FABESA: Fast (and Anonymous) Attribute-Based Encryption under Standard Assumptions

Long Meng
University of Surrey
Guildford, United Kingdom
long.meng@surrey.ac.uk

Liqun Chen
University of Surrey
Guildford, United Kingdom
liqun.chen@surrey.ac.uk

Yangguang Tian
University of Surrey
Guildford, United Kingdom
yangguang.tian@surrey.ac.uk

Mark Manulis
Universität der Bundeswehr München
München, Germany
mark.manulis@unibw.de

ABSTRACT

Attribute-Based Encryption (ABE) provides fine-grained access control to encrypted data and finds applications in various domains. The practicality of ABE schemes hinges on the balance between security and efficiency. The state-of-the-art adaptive secure ABE scheme, proven to be adaptively secure under standard assumptions (FAME, CCS'17), is less efficient compared to the fastest one (FABEO, CCS'22) which is only proven secure under the Generic Group Model (GGM). These traditional ABE schemes focus solely on message privacy. To address scenarios where attribute value information is also sensitive, Anonymous ABE (A^2BE) ensures the privacy of both the message and attributes. However, most A^2BE schemes suffer from intricate designs with low efficiency, and the security of the fastest key-policy A^2BE (proposed in FEASE, USENIX'24) relies on the GGM.

In this paper, we propose novel fast key-policy and ciphertext-policy ABE schemes that (1) support both AND and OR gates for access policies, (2) have no restriction on the size and type of policies or attributes, (3) achieve adaptive security under the standard DLIN assumption, and (4) only need 4 pairings for decryption. As our ABE constructions automatically provide ciphertext anonymity, we easily transform our ABE schemes to A^2BE schemes while maintaining the same features and high-level efficiency.

The implementation results show that all our schemes achieve the best efficiency comparing to other schemes with adaptive security proven under standard assumptions. Specifically, our ABE schemes perform better than FAME and are close to FABEO. Our key-policy A^2BE scheme performs close to the one in FEASE and our ciphertext-policy A^2BE outperforms the state-of-the-art (Cui et al., ProvSec'16).

KEYWORDS

Attribute-Based Encryption, DLIN assumption, Anonymity

1 INTRODUCTION

Attribute-Based Encryption (ABE) [25, 51] elaborates upon the foundations of classical public-key encryption that introduces the novel feature of fine-grained access control over encrypted data. ABE has applications in a number of scenarios including electronic medical records [4], cloud security [52], verifiable computation [45], and online social networks [9]. Companies like CipherCloud [1] already provide cloud security solutions, including data encryption and tokenization through ABE to control data access.

ABE can be categorized into Key-Policy ABE (KP-ABE) [25] and Ciphertext-Policy ABE (CP-ABE) [11]. In KP-ABE, a ciphertext is associated with a set of attributes, and a secret key is associated with an access policy. The encrypted message can only be decrypted if the attribute set in the ciphertext satisfies the access policy in the secret key. A CP-ABE is the dual of KP-ABE with access policies associated with the ciphertext and attribute sets attached to the secret key.

Generally, *expressiveness*, *security*, and *efficiency* are three primary factors that decide the practicality of an ABE scheme. First, an ABE scheme should support *expressive* access policies described as Boolean formulas (AND, OR gates), allowing for high-level granularity in access control. For example, a commonly-used policy in healthcare systems could be like "(Title: Professor **OR** Years: 10) **AND** (Subject: Surgery)", which allows access for surgery professors, or surgery doctors with over 10 years of experience. Second, ensuring the *adaptive security* for an ABE scheme provides confidentiality for data encrypted under attributes or policies chosen anytime during a system's life cycle, is natural and stronger than schemes with selective security that only allow attributes or policies declared before the system is deployed. Besides, it is preferred to reduce the security of an ABE scheme to *standard assumptions* that are well-studied and time-tested (e.g., Bilinear Diffie-Hellman, Decisional Linear, etc.), because it instills a higher level of confidence in security than the schemes proven under the Generic Group Model (GGM) which lacks concrete security assurances and real-world instantiations. Third, efficiency is pivotal with the need to minimize both *communication* and *computational* overhead to avoid delays in systems such as healthcare data-sharing or cloud computing, where inefficiencies can compromise patient care and user experience, respectively.

The current landscape of practical ABE schemes excelling in the dimensions of expressiveness, security, and efficiency include [3, 6, 11, 16, 20, 25, 49, 50, 55]. Among them, the state-of-the-art schemes are (1) FAME KP-ABE and CP-ABE schemes by Agrawal and Chase [3], satisfying adaptive security under standard assumptions, and (2) FABEO KP-ABE and CP-ABE schemes by Riepel and Wee [49], emphasizing efficiency while satisfying adaptive security under the GGM. Although FAME achieves stronger security arguments, its use of the dual system framework [33, 54] results in complex constructions with a considerable efficiency trade-off. Compared to FABEO with 100 attributes in both sets and policies, FAME is 8.9 and 8.5 times slower for key generation in KP-ABE and CP-ABE respectively, and

5.7 and 3.4 times slower for encryption in KP-ABE and CP-ABE respectively. Despite FAME achieving fast decryption with a constant 6 pairings, FABEO surpasses it with needing only constant 2 (for KP-ABE) or 3 (for CP-ABE) pairings. In essence, the cost of achieving strong security arguments remains substantial. Thus, our first question arises: *Can we construct faster KP-ABE and CP-ABE schemes that satisfy adaptive security under standard assumptions?*

Continuing on our research, traditional ABE schemes only protect message privacy rather than attribute privacy due to the following two reasons: (1) The ciphertext is sent with an explicit attribute set or access policy, which we call a "payload"[1] that is directly accessible to attackers. (2) Even if the payload is not sent with the ciphertext, the ciphertext itself does not provide anonymity. For example, the fastest KP-ABE scheme FABEO [49] constructs the ciphertext with $ct_{1,u} = \mathsf{H}(u)^s$, $ct_2 = g_2^s$, in which H is a hash function that maps an attribute u to a group element in $\mathbb{G}_1$, s is a randomness on $\mathbb{Z}_p$, and g_2 is a public generator in $\mathbb{G}_2$. In this case, an attacker can distinguish $b \in \{0, 1\}$ when provided with two attributes u_0 and u_1 and a ciphertext ct_{u_b} by $e(\mathrm{ct}_{1,u_b}, g_2) = e(\mathsf{H}(u_b), \mathrm{ct}_2)$. However, this is not appropriate for applications where the attributes contain sensitive information such as healthcare [18, 62], edge computing [56], blockchain-based access control [21], etc.

To address this problem, Anonymous ABE (A^2BE) schemes [28, 42] were proposed to protect both the message privacy and attribute privacy with the following two conditions: (1) *Payload privacy*: The payload does not reveal information about attribute values. (2) *Ciphertext anonymity*: the ciphertext does not reveal information about the message and attribute values [28]. Similarly, A^2BE has two variants: Key-Policy Anonymous ABE (KP-A^2BE) and Ciphertext-Policy Anonymous ABE (CP-A^2BE, or policy-hiding ABE) dependent on if the access policy is associated with the key or the ciphertext.

In the literature, there are two categories of A^2BE: (1) *Fully A^2BE* achieved by the Inner Product Encryption (IPE) [17, 28, 43, 44]. Specifically, IPE encodes the policy and attribute set as separate vectors, $\mathbf{x}$ and $\mathbf{y}$, into the secret key and ciphertext, with decryption succeeding if the inner product $\mathbf{x} \cdot \mathbf{y} = 0$. While these schemes fully hide the payload and achieve strong security arguments for ciphertext anonymity, they are deemed impractical due to restrictions on the size of policy and attribute sets during setup and a superpolynomial degradation in efficiency resulting from the encoding of policies or attributes into vectors (detailed in Sec. 6). (2) *Partially A^2BE* mainly focuses on enabling ciphertext anonymity in ABE constructions and adopts the partially hidden structure [29] for payload privacy. This structure protects the privacy of attribute values and exposes attribute names in the payload, in exchange for efficient matching between policies and attribute sets. The efficiency of such schemes depends on the ABE efficiency, making them more practical for real-world applications.

In this paper, our primary focus is on the study of partially A^2BE. Numerous works have been developed in the literature for both KP-A^2BE [19, 31, 38, 41] and CP-A^2BE [18, 29, 42, 60–62] schemes. The state-of-the-art A^2BE schemes that support expressive policies include (1) the KP-A^2BE scheme in FEASE [40] [2] from Long et al. that achieves the best efficiency but its adaptive security is proven under the GGM, and (2) the CP-A^2BE scheme CDW$^+$ [18] that achieves only selective security and their intricate constructions lead to low efficiency. In summary, the state-of-the-art A^2BE schemes suffer from deficiencies in security and efficiency, which unavoidably limit their practicality in real-world applications. Given these challenges, our second question arises: *Can we develop fast KP-A^2BE and CP-A^2BE schemes that achieve adaptive security under standard assumptions?*

Contributions. To answer both of our questions, we first propose a novel fast KP-ABE and a CP-ABE scheme. These two schemes share the following features:

(1) *Expressiveness*: Support expressive access policies that can be expressed as any monotonic Boolean (AND, OR) formulas.
(2) *Group setting*: Constructed in prime-order groups with the efficient Type-III pairing.
(3) *Attribute usage*: No restrictions on the size of the attributes and allow any arbitrary string to be used as an attribute.
(4) *Security*: Satisfy adaptive security under the Decional Linear (DLIN) assumption. The security is proved by using the random oracle model [3].
(5) *Efficiency*: The complexity of the encryption and key generation algorithm is linear to the number of attributes and the decryption only requires constant 4 pairings.
(6) *Ciphertext anonymity*: The ciphertext protects the privacy of both the message and attribute values.

Then, we easily bridge our ABE schemes to A^2BE schemes by adopting the partially hidden structure [29] to protect payload privacy. The resulting A^2BE schemes inherit the features 1-4 from our ABE schemes and maintain close efficiency as our ABE schemes.

Our implementation results show that all our schemes reach the best efficiency among the schemes that are adaptively secure under standard assumptions. Our ABE schemes perform better than FAME [3] and close to FABEO [49]. Our KP-A^2BE scheme performs close to FEASE [40] and our CP-A^2BE outperforms CDW$^+$ [18].

The following results are obtained when the size of the attribute set and policy is set to 100: Our KP-ABE scheme runs 0.19s and 0.14s for key generation and encryption, which are 6.1 and 2.9 times faster than FAME, 1.4 and 1.9 times slower than FABEO, respectively. Our CP-ABE scheme runs 0.12s and 0.21s for key generation and encryption, which are 4.6 and 2.3 times faster than FAME and 1.8 and 1.5 times slower than FABEO, respectively. For decryption, both our schemes need 4 pairings that are less than FAME (6 pairings) and closely match FABEO (2-3 pairings). Our KP-A^2BE is 1.2 times faster and 1.8 times slower than FEASE for key generation and encryption respectively. Our CP-A^2BE is 35 times and 2.3 times faster than CDW$^+$ for key generation and encryption respectively. For decrypting a conjunctive set of 100 attributes, our KP-A^2BE is 1.2 times slower than FEASE and our CP-A^2BE is 168 times faster than CDW$^+$.

2 PRELIMINARIES

We first define the notation that will be used throughout the paper. We denote the set $1, ..., n$ as $[n]$. For a prime p, let $\mathbb{Z}_p$ denote the set $[0, ..., p-1]$ where addition and multiplication are computed modulo p. $\mathbb{Z}_p^*$ is same as $\mathbb{Z}_p$ but with 0 removed. Let λ denote the security

[1]The definition of "payload" in this paper does not include the message to be encrypted.
[2]In [40], a KP-A^2BE scheme is proposed and transformed into the searchable encryption scheme called FEASE. In this paper, we use FEASE as a name for the KP-A^2BE scheme from [40].

[3]Random oracle is fairly common in cryptographic protocols. The state-of-the-art ABE schemes FAME [3] and FABEO [49] also use random oracle for security proofs.

parameter. For a set S, $s \xleftarrow{\$} S$ denotes that s is sampled uniformly at random from S. A probabilistic algorithm is called probabilistic polynomial time (PPT) if its running time is bounded by some polynomial in the length of its input. A vector $\mathbf{v}$ is treated as a column vector. $\mathbf{v}_k$ denotes the k-th element of $\mathbf{v}$ and $\|$ denotes concatenation of vectors. M_i and $\mathrm{M}_{i,j}$ denote the i-th row and the (i, j)-th element of a matrix M, respectively. We use M^{T} for the transpose of M.

2.1 Access structure

Monotone means that an authorized user who acquires more attributes will not lose any privileges. A (monotone) Boolean formula consists of **AND** and **OR** gates, where each input is associated with an attribute in an attribute universe $\mathcal{U}$. We say a set of attributes $S \subseteq \mathcal{U}$ satisfies a Boolean formula if we set all inputs of the formula that map to an attribute in S to true and the others to false.

Monotone span programs (MSP) [10] are a more general class of functions and include Boolean formulas. We encode an access structure by a policy (M, π) where M of size $\ell \times n$ over $\mathbb{Z}_p$ and a general mapping function $\pi : \{1, ..., \ell\} \rightarrow \mathcal{U}$. In [32], Lewko and Waters describe a simple and efficient method to convert any (monotone) Boolean formula F into an MSP(M, π) such that every row of M corresponds to input in F and the number of columns is same as the number of **AND** gates in F. Furthermore, each entry in M is either a 0, 1, or -1. Let $\mathbb{S} = \{u_i\}_{i\in[m]} \subseteq \mathcal{U}$ be a set of m attributes and $I = \{i \,|\, i \in \{1, ..., \ell\}, \pi(i) \in \mathbb{S}\}$ be the set of rows in M that belong to $\mathbb{S}$. We say that (M, π) accepts $\mathbb{S}$ if there exists a linear combination of rows in I that gives $(1, 0,..., 0)$. This means, there exist constants $\gamma_i \in \mathbb{Z}_p$ for $i \in I$ such that $\sum_{i\in I} \gamma_i \mathbf{M}_i = (1, 0, ..., 0)$. These constants can be computed in time polynomial in the size of M. It is worth noting that if Lewko and Water's method is applied to Boolean formulas, then it is always possible to pick coefficients that are either 0 or 1 for the resulting MSPs, irrespective of the set $\mathbb{S}$. Note that the above notation will be used in Sec. 3.1 and Sec. 3.2.

2.2 Partially hidden structure

One naive solution to protect payload privacy of expressive A^{2}BE schemes is simply to remove the payload, but then it is not aware of which set of attributes satisfies the policy. Consequently, decryption involves attempting every key and ciphertext component in all possible combinations, leading to extremely low efficiency. To address this problem, Lai et al. proposed the "partially hidden structure" [29, 31] that works as follows. First, we define an attribute set $\mathbb{S} = \{u_i\}_{i\in[m]}$ that has m attributes with each attribute belonging to a different category (attribute name). Let n_i and v_i denote the attribute name and attribute value of an attribute u_i respectively, i.e., $u_i = \{n_i, v_i\}$. Second, we express an access policy as $\mathbb{A} = (\mathrm{M}, \pi, \{\pi(i)\}_{i\in[\ell]})$, where M is a $\ell \times n$ share-generating matrix, M_i denotes the i^{th} row of M, π is a mapping function from M_i to an attribute $\pi(i)$. Let $n_{\pi(i)}$ and $v_{\pi(i)}$ denote the attribute name and attribute value of attribute $\pi(i)$ respectively, i.e., $\pi(i) = \{n_{\pi(i)}, v_{\pi(i)}\}$. By applying this structure, the attribute values $v_{\pi(i)}$ of an access policy $\mathbb{A}$ and the attribute values v_i of an attribute set $\mathbb{S}$ are not exposed in the ciphertext or secret key, while the access policy information $(\mathrm{M}, \pi, \{n_{\pi(i)}\}_{i\in[\ell]})$ and attribute names $\{n_i\}_{i\in[m]}$ are disclosed. A user's attribute set $\mathbb{S} = \{u_i\} = \{n_i, v_i\}_{i\in[m]}$ satisfies an access policy $(\mathrm{M}, \pi, \pi(i) = \{n_{\pi(i)}, v_{\pi(i)}\}_{i\in[\ell]})$ if and only if there exists $\mathcal{I} \subseteq \{1, ..., \ell\}$ and constants $\{\gamma_i\}_{i\in\mathcal{I}}$ such that

$$\sum_{i\in\mathcal{I}} \gamma_i \mathrm{M}_i = (1, 0, ..., 0) \text{ and } \pi(i) = u_i \text{ for } \forall i \in \mathcal{I}.$$

Note that the above notation will be used in Sec. 3.3. Taking the policy "(Title: Professor **OR** Years: 10) **AND** (Subject: Surgery)" and attribute set "[Title: Doctor, Years: 5, Subject: Surgery]" as an example, the partially hidden policy is "(Title **OR** Years) **AND** Subject", and the partially hidden attribute set is "[Title, Years, Subject]" separately. This technique, although leaking a certain level of information (i.e., attribute names), provides high efficiency. Attribute names, being less sensitive than attribute values, enable efficient policy matching and fast location of the attribute values under corresponding names without involving pairing or exponentiation operations, thereby significantly enhancing decryption efficiency.

2.3 Bilinear maps and assumption

Bilinear maps. Let GroupGen be a PPT algorithm that takes as input a security parameter 1^λ and outputs a set of group parameters $\mathsf{par} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$, where p is the prime order of $\Theta(\lambda)$ bits, $\mathbb{G}_1$,$\mathbb{G}_2$ and $\mathbb{G}_T$ are cyclic groups of order p, g_1 and g_2 are the generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively. $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is an asymmetric Type-III pairing group where there exists no efficiently computable homomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$.

Decisional Linear (DLIN) assumption. We refer to the asymmetric version of the DLIN problem introduced in [3]. We define $\mathsf{par} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2) \leftarrow \mathsf{GroupGen}(1^\lambda)$, $x_1, x_2, y_1, y_2, R \xleftarrow{\$} \mathbb{Z}_p$, $D = (g_1^{x_1}, g_1^{x_2}, g_2^{x_1}, g_2^{x_2}, g_1^{x_1 y_1}, g_1^{x_2 y_2}, g_2^{x_1 y_1}, g_2^{x_2 y_2})$. $T_0 = (g_1^{y_1+y_2}, g_2^{y_1+y_2})$, $T_1 = (g_1^R, g_2^R)$. Then we define the advantage of an algorithm $\mathcal{A}$ in deciding the DLIN problem

$$\mathsf{Adv}_{\mathcal{A}}^{\mathrm{DLIN}}(\lambda) := \Big|\Pr[\mathcal{A}(\mathsf{par}, D, T_0) = 1] - \Pr[\mathcal{A}(\mathsf{par}, D, T_1) = 1]\Big|$$

is negligible in λ. The probability is over the uniform random choice of the parameters and over the coin tosses of $\mathcal{A}$. We say that an algorithm $\mathcal{A}(t, \epsilon)$ decides DLIN problem in $\mathbb{G}_1$ and $\mathbb{G}_2$ if $\mathcal{A}$ runs in time at most t, $\mathsf{Adv}_{\mathcal{A}}^{\mathrm{DLIN}}$ is at least ϵ.

Definition 1. *(DLIN assumption.) The (t, ϵ) DLIN assumption holds in both $\mathbb{G}_1$ and $\mathbb{G}_2$ if no t-time algorithm has advantage at least ϵ in solving the DLIN problem.*

2.4 Security definitions of ABE and A^{2}BE

We provide a general syntax and security model for ABE and A^{2}BE schemes. In the following, we define the description x, y as an access policy $\mathbb{A}$ and an attribute set $\mathbb{S}$. For KP-ABE (or KP-A^{2}BE), $x = \mathbb{A}$, $y = \mathbb{S}$. For CP-ABE (or CP-A^{2}BE), $x = \mathbb{S}$, $y = \mathbb{A}$. We define $\mathsf{P}(x, y) = 1$ as $\mathbb{S}$ satisfies $\mathbb{A}$ and define $\mathsf{P}(x, y) = 0$ as $\mathbb{S}$ does not satisfy $\mathbb{A}$, no matter which one is associated with x or y. For an ABE scheme, we define that the payload $\mathsf{pl} = x$ is associated with the secret key and $\mathsf{pl} = y$ is associated with the ciphertext. For a partially A^{2}BE scheme, we define $\mathsf{pl} = \{n_i\}_{i\in[m]}$ for $\mathbb{S}$, and $\mathsf{pl} = (\mathrm{M}, \pi, \{n_{\pi(i)}\}_{i\in[\ell]})$ for $\mathbb{A}$.

Syntax. An ABE (or A^{2}BE) scheme over a message space $\mathsf{msg} \in \mathcal{M}$ consists of the following algorithms:

- $\mathsf{Setup}(1^\lambda)$. The setup algorithm takes input a security parameter 1^λ, outputs a public key pk and a master secret key msk.

- $\mathsf{KeyGen}(\mathsf{pk}, \mathsf{msk}, x)$. The key generation algorithm takes input pk, msk, and x, outputs a secret key sk_x and a payload pl_x.
- $\mathsf{Enc}(\mathsf{pk}, y, \mathsf{msg})$. The encryption algorithm takes input pk, y, and $\mathsf{msg} \in \mathcal{M}$, outputs a ciphertext ct_y and a payload pl_y.
- $\mathsf{Dec}(\mathsf{pl}_x, \mathsf{pl}_y, \mathsf{ct}_y, \mathsf{sk}_x)$. The decryption algorithm takes input pl_x, pl_y, sk_x and ct_y, outputs a message msg if $\mathsf{P}(x, y) = 1$, or a special symbol $\perp$.

Correctness. For any input x and y with $\mathsf{P}(x, y) = 1$ and $\mathsf{msg} \in \mathcal{M}$, we require $\Pr[\mathsf{Dec}(\mathsf{pl}_x, \mathsf{pl}_y, \mathsf{ct}_y, \mathsf{sk}_x) = \mathsf{msg} : (\mathsf{pk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)),$ $(\mathsf{sk}_x, \mathsf{pl}_x) \leftarrow \mathsf{KeyGen}(\mathsf{pk}, \mathsf{msk}, x)$, $(\mathsf{ct}_y, \mathsf{pl}_y) \leftarrow \mathsf{Enc}(\mathsf{pk}, y, \mathsf{msg})] = 1$.

An ABE scheme addresses the property that a ciphertext does not reveal any information about the encrypted message, which is called "Indistinguishability against Chosen Plaintext Attack (IND-CPA)" security. A partially A^{2}BE scheme addresses the property that a ciphertext does not reveal the encrypted message and attribute values, which is called "Anonymity (Anon)".

IND-CPA Security. We model the adaptive IND-CPA security [4] in a game $\prod_1$ between an adversary $\mathcal{A}$ and a challenger C as follows:

- **Setup.** C runs $\mathsf{Setup}(1^\lambda)$ to obtain a public key pk and a master secret key msk. It sends pk to $\mathcal{A}$ and keeps msk secret.
- **Phase 1.** $\mathcal{A}$ issues queries to a key generation oracle for polynomial many times:
  - Key generation oracle: Given a description x, the oracle generates $(\mathsf{sk}_x, \mathsf{pl}_x) \leftarrow \mathsf{KeyGen}(\mathsf{pk}, \mathsf{msk}, x)$ for $\mathcal{A}$.
- **Challenge.** $\mathcal{A}$ outputs a challenge description y^* and two equal-length messages msg_0^*, msg_1^* with the restriction that $\mathsf{P}(x, y^*) = 0$ for any x that has been queried in Phase 1. Then C selects a random bit $b \in \{0, 1\}$, runs the algorithm $(\mathsf{ct}_b^*, \mathsf{pl}_y^*) \leftarrow \mathsf{Enc}(\mathsf{pk}, y^*, \mathsf{msg}_b^*)$ and returns the challenge $(\mathsf{ct}_b^*, \mathsf{pl}_y^*)$ to $\mathcal{A}$.
- **Phase 2.** Same as Phase 1 except $\mathsf{P}(x, y^*) = 0$ for any input x.
- **Guess.** $\mathcal{A}$ outputs $b' \in \{0, 1\}$ and wins the game if $b' = b$.

An ABE scheme is adaptively IND-CPA secure if the advantage function refers to the security game $\prod_1$

$$\mathsf{Adv}^{\mathsf{CPA}}_{\prod_1, \mathcal{A}}(\lambda) = \left| \Pr[b' = b] - \frac{1}{2} \right|$$

is negligible in the parameter λ for any PPT adversary $\mathcal{A}$.

Anonymity. For a partially A^{2}BE scheme to achieve anonymity, it needs to satisfy two properties defined as follows:

(1) Payload privacy: The payload sent along with the ciphertext does not reveal the attribute values.
(2) Ciphertext anonymity: The ciphertext construction does not reveal the message and attribute values.

Then we model the adaptive anonymity of a partially A^{2}BE scheme in a game $\prod_2$ that addresses both the above properties, which is running between an adversary $\mathcal{A}$ and a challenger C as follows:

- **Setup.** Same as defined in IND-CPA.
- **Phase 1.** Same as defined in IND-CPA.
- **Challenge.** $\mathcal{A}$ outputs two equal-size messages msg_0^*, msg_1^* and descriptions y_0^*, y_1^* with the restriction that y_0^*, y_1^* have the same pl^* and $\mathsf{P}(x, y_0^*) = \mathsf{P}(x, y_1^*) = 0$ for any x that has been queried in Phase 1. C selects $b \in \{0, 1\}$, runs $(\mathsf{ct}_b^*, \mathsf{pl}_y^*) \leftarrow \mathsf{Enc}(\mathsf{pk}, y_b^*, \mathsf{msg}_b^*)$ and returns the $(\mathsf{ct}_b^*, \mathsf{pl}_y^*)$ to $\mathcal{A}$.
- **Phase 2.** Same as Phase 1 with the restriction that $\mathsf{P}(x, y_0^*) = \mathsf{P}(x, y_1^*) = 0$ for any input x.
- **Guess.** Same as defined in IND-CPA.

A partially A^{2}BE scheme is adaptively anonymous [5] if the advantage function refers to the security game $\prod_2$

$$\mathsf{Adv}^{\mathsf{Anon}}_{\prod_2, \mathcal{A}}(\lambda) = \left| \Pr[b' = b] - \frac{1}{2} \right|$$

is negligible in the parameter λ for any PPT adversary $\mathcal{A}$.

3 OUR SCHEMES

As shown in Fig. 1, our technical roadmap unfolds in stages. We start from the fastest KP-ABE scheme FABEO [49] because it (1) supports expressive access policies, (2) has no restrictions on the attribute size and type, (3) is constructed on prime order group with Type-III pairing, and (4) only requires a constant 2 pairings for decryption.

Based on the facts (1) FABEO KP-ABE does not satisfy anonymity, (2) the adaptive IND-CPA security of FABEO KP-ABE is proven in the GGM, and (3) our target is to design fast ABE and A^{2}BE schemes in which the IND-CPA security and anonymity can be reduced to standard assumptions, our first step is to transform FABEO KP-ABE into a KP-ABE scheme with ciphertext anonymity (which also implies IND-CPA security). This step has been already achieved by FEASE KP-A^{2}BE [40] by using a "randomness splitting" technique, so we briefly review this step and explain the construction of the "FEASE KP-ABE" scheme in Sec. 3.1. However, the ciphertext anonymity of FEASE KP-ABE is still proven in the GGM. As the main technical contribution in this work, we propose novel techniques in Sec. 3.2, allowing modifications in FEASE KP-ABE to achieve ciphertext anonymity under the DLIN assumption, leading to our proposed KP-ABE scheme. After that, we transform our KP-ABE into our CP-ABE scheme in terms of their difference in syntax. Finally, we apply the partially hidden structure (as defined in Sec. 2.2) in our ABE schemes for reaching payload privacy, resulting in our KP-A^{2}BE and CP-A^{2}BE schemes in Sec. 3.3. In this section, we provide a step-by-step guidance explaining our designs.

3.1 From FABEO KP-ABE to FEASE KP-ABE

The construction of FABEO KP-ABE scheme [49] is presented in Fig. 2. The notation is defined in Sec. 2.1. Besides, the r value in sk_1 would have been a vector $\mathbf{r}'$, and the original version should be $\mathsf{sk}_{1,j} = g_2^{\mathbf{r}'[j]}$ where $j \in [\tau]$ indicates the number of attribute re-use. We simplify it and let $j = 1$ since it is easier for further illustrations.

In terms of the definition in Sec. 2.4, the FABEO construction does not satisfy anonymity because of the following two reasons:

(1) No payload privacy: The payload in FABEO includes the exposed attribute set $\mathbb{S}$ as an element, making it directly accessible to attackers.
(2) No ciphertext anonymity: The ciphertext construction does not provide anonymity. Specifically, when provided with two attributes, u_0 and u_1, and a ciphertext $(\mathsf{ct}_{1,u_b}, \mathsf{ct}_2)$ where $b \in \{0, 1\}$, attackers can determine b from the equation $e(\mathsf{ct}_{1,u_b}, g_2) = e(\mathsf{H}(u_b), \mathsf{ct}_2)$.

[4]The IND-CPA security of our schemes can be extended to IND-CCA security by using generic transformations such as [12, 22]. The details are out the scope of this paper.

[5]The anonymity is defined similar to the "weakly attribute-hidng" in [43]. The only difference is that our model allows the exposure of the payload pl^* while [43] does not.

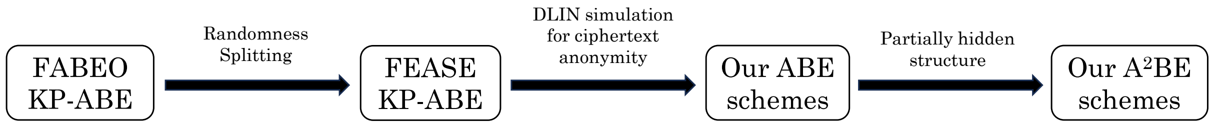


Figure 1: The technical roadmap for the design of our schemes. The text above the arrows indicates the techniques used for the transformations.

$(\mathsf{pk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$.

Run $\mathsf{GroupGen}(1^\lambda)$ to obtain the group parameters $\mathsf{par} := (p, e, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2)$. Pick $\alpha \xleftarrow{\$} \mathbb{Z}_p$ and a hash function $\mathsf{H} : \{0,1\}^* \to \mathbb{G}_1$. Compute the public key pk and master secret key msk as

$$\mathsf{pk} = (\mathsf{par}, \mathsf{H}, e(g_1, g_2)^\alpha),\ \mathsf{msk} = \alpha.$$

$(\mathsf{sk}, \mathsf{pl}_{\mathbb{A}}) \leftarrow \mathsf{KeyGen}(\mathsf{pk}, \mathsf{msk}, \mathbb{A} = (\mathrm{M}, \pi, \{\pi(i)\}_{i\in[\ell]}))$.

Pick $r \xleftarrow{\$} \mathbb{Z}_p^\tau, \mathbf{v} \xleftarrow{\$} \mathbb{Z}_p^{n-1}$. Compute

$$\mathsf{sk}_1 = g_2^r,\ \mathsf{sk}_{2,i} = g_1^{\mathrm{M}_i(\alpha\|\mathbf{v})^\top} \cdot \mathsf{H}(\pi(i))^r.$$

Output $\mathsf{sk} = (\mathsf{sk}_1, \{\mathsf{sk}_{2,i}\}_{i\in[\ell]})$, $\mathsf{pl}_{\mathbb{A}} = \mathbb{A}$.

$(\mathsf{ct}, \mathsf{pl}_{\mathbb{S}}) \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathbb{S} = \{u_i\}_{i\in[m]}, \mathsf{msg})$. Pick $s \xleftarrow{\$} \mathbb{Z}_p$. Compute

$$\mathsf{ct}_{1,i} = \mathsf{H}(u_i)^s,\ \mathsf{ct}_2 = g_2^s,\ \mathsf{ct}_3 = e(g_1, g_2)^{\alpha s} \cdot \mathsf{msg}.$$

Output $\mathsf{ct} = (\{\mathsf{ct}_{1,i}\}_{i\in[m]}, \mathsf{ct}_2, \mathsf{ct}_3)$, $\mathsf{pl}_{\mathbb{S}} = \mathbb{S}$.

$\mathsf{msg}/\bot \leftarrow \mathsf{Dec}(\mathsf{ct}, \mathsf{sk})$.

Tests if there is any subset $\mathcal{I}$ that matches $\mathbb{S}$ in ct with $\mathbb{A}$ in sk. If not, return $\bot$. Otherwise, it finds constants $\{\gamma_i\}_{i\in\mathcal{I}}$ s.t. $\sum_{i\in\mathcal{I}} \gamma_i \mathrm{M}_i = (1, 0, \ldots, 0)$ and reconstructs the message msg by computing:

$$\frac{e\Big(\prod_{i\in\mathcal{I}} (\mathsf{ct}_{1,\pi(i)})^{\gamma_i}, \mathsf{sk}_1\Big) \cdot \mathsf{ct}_3}{e\Big(\prod_{i\in\mathcal{I}} (\mathsf{sk}_{2,i})^{\gamma_i}, \mathsf{ct}_2\Big)}.$$

Figure 2: The FABEO KP-ABE scheme [49].

$(\mathsf{pk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$.

Run $\mathsf{GroupGen}(1^\lambda)$ to obtain the group parameters $\mathsf{par} := (p, e, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2)$. Pick $\alpha, b_1, b_2 \xleftarrow{\$} \mathbb{Z}_p$ and a hash function $\mathsf{H} : \{0,1\}^* \to \mathbb{G}_1$. Compute

$$\mathsf{pk} = (\mathsf{par}, \mathsf{H}, g_2^{b_1}, g_2^{b_2}, e(g_1, g_2)^\alpha),\ \mathsf{msk} = (\alpha, b_1, b_2).$$

$(\mathsf{sk}, \mathsf{pl}_{\mathbb{A}}) \leftarrow \mathsf{KeyGen}(\mathsf{pk}, \mathsf{msk}, \mathbb{A} = (\mathrm{M}, \pi, \{\pi(i)\}_{i\in[\ell]}))$.

Pick $r \xleftarrow{\$} \mathbb{Z}_p, \mathbf{v} \xleftarrow{\$} \mathbb{Z}_p^{n-1}$. Compute $\mathsf{sk}_1 = g_2^r$,

$$\mathsf{sk}_{2,i} = (g_1^{\mathrm{M}_i(\alpha\|\mathbf{v})^\top} \mathsf{H}(\pi(i))^r)^{\frac{1}{b_1}},\ \mathsf{sk}_{3,i} = (g_1^{\mathrm{M}_i(\alpha\|\mathbf{v})^\top} \mathsf{H}(\pi(i))^r)^{\frac{1}{b_2}}.$$

Output $\mathsf{sk} = (\mathsf{sk}_1, \{\mathsf{sk}_{2,i}, \mathsf{sk}_{3,i}\}_{i\in[\ell]})$, $\mathsf{pl}_{\mathbb{A}} = \mathbb{A}$.

$(\mathsf{ct}, \mathsf{pl}_{\mathbb{S}}) \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathbb{S} = \{u_i\}_{i\in[m]}, \mathsf{msg})$.

Pick $s_1, s_2 \xleftarrow{\$} \mathbb{Z}_p$, let $s = s_1 + s_2$. Compute

$$\mathsf{ct}_{1,i} = \mathsf{H}(u_i)^s,\ \mathsf{ct}_2 = g_2^{b_1 s_1},\ \mathsf{ct}_3 = g_2^{b_2 s_2},\ \mathsf{ct}_4 = e(g_1, g_2)^{\alpha s} \cdot \mathsf{msg}.$$

Output $\mathsf{ct} = (\{\mathsf{ct}_{1,i}\}_{i\in[m]}, \mathsf{ct}_2, \mathsf{ct}_3, \mathsf{ct}_4)$, $\mathsf{pl}_{\mathbb{S}} = \mathbb{S}$.

$\mathsf{msg}/\bot \leftarrow \mathsf{Dec}(\mathsf{ct}, \mathsf{pl}_{\mathbb{S}}, \mathsf{sk}, \mathsf{pl}_{\mathbb{A}})$.

Tests if there is any subset $\mathcal{I}$ that matches $\mathbb{S}$ in ct with $\mathbb{A}$ in sk. If not, return $\bot$. Otherwise, it finds constants $\{\gamma_i\}_{i\in\mathcal{I}}$ s.t. $\sum_{i\in\mathcal{I}} \gamma_i \mathrm{M}_i = (1, 0, \ldots, 0)$ and reconstructs the message msg by computing:

$$\frac{e\Big(\prod_{i\in\mathcal{I}} (\mathsf{ct}_{1,\pi(i)})^{\gamma_i}, \mathsf{sk}_1\Big) \cdot \mathsf{ct}_4}{e\Big(\prod_{i\in\mathcal{I}} (\mathsf{sk}_{2,i})^{\gamma_i}, \mathsf{ct}_2\Big) \cdot e\Big(\prod_{i\in\mathcal{I}} (\mathsf{sk}_{3,i})^{\gamma_i}, \mathsf{ct}_3\Big)}.$$

Figure 3: The FEASE KP-ABE scheme from FEASE KP-A^{2}BE [40] by removing the partially hidden structure.

To enable FABEO KP-ABE into a KP-A^{2}BE scheme, FEASE [40] protects payload privacy by using the partially hidden structure. As we first focus on designing non-anonymous ABE schemes, we remove the partially hidden structure and form a “FEASE KP-ABE scheme” as shown in Fig. 3. FEASE KP-ABE uses a “randomness splitting technique” to realize ciphertext anonymity on FABEO KP-ABE. This technique divides the randomness s into two distinct components $s_1, s_2 \in \mathbb{Z}_p$ in which $s = s_1 + s_2$, and forms a DLIN type of construction $\mathsf{ct}_{1,i} = \mathsf{H}(u_i)^s$, $\mathsf{ct}_2 = g_2^{b_1 s_1}$, $\mathsf{ct}_3 = g_2^{b_2 s_2}$, where $g_2^{b_1}$ and $g_2^{b_2}$ are parts of the public key. Then the secret key $\mathsf{sk}_{2,i} = g_1^{\mathrm{M}_i(\alpha\|\mathbf{v})^\top} \cdot \mathsf{H}(\pi(i))^r$ is doubled and exponentiated by $\frac{1}{b_1}$ and $\frac{1}{b_2}$ correspondingly. Now given two attributes, u_0 and u_1, and a ciphertext $(\mathsf{ct}_{1,u_b}, \mathsf{ct}_2, \mathsf{ct}_3, \mathsf{ct}_4)$ where $b \in \{0, 1\}$, an attacker who owns $g_2^{b_1}$ and $g_2^{b_2}$ can no longer discern the attribute u_b. However, the adaptive ciphertext anonymity (including IND-CPA security) of FEASE KP-ABE can be only proven in the GGM instead of relying on the DLIN assumption [40]. This is due to the following technical problem: FEASE KP-ABE uses DLIN-format construction to prevent the attack for distinguishing attributes as above. However, when reducing the ciphertext anonymity of FEASE KP-ABE to the DLIN assumption, only the ciphertext can be simulated by a DLIN tuple, the secret key $\mathsf{sk}_{2,i}$ and $\mathsf{sk}_{3,i}$ involving terms $g_1^{\frac{1}{b_j}}$, $\mathsf{H}(\pi(i))^{\frac{1}{b_j}}$ for $j = \{1, 2\}$ cannot be simulated by a DLIN tuple. Therefore, our technical challenge is:

How to enable the simulation of both the ciphertext and (especially) the secret key of FEASE KP-ABE by using a DLIN tuple without trading-off too much efficiency?

To overcome this challenge, we do the following modifications.

3.2 From FEASE KP-ABE to our ABE schemes

DLIN simulation for ciphertext anonymity. The ciphertext anonymity addresses the indistinguishability for both the attribute terms $\mathsf{ct}_{1,i}$, and the message term ct_4. We first aim to reduce the attribute term $\mathsf{ct}_{1,i}$ into the DLIN hard problem. In the secret key, we observe that

$\underline{(\mathsf{pk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda).}$

Run $\mathsf{GroupGen}(1^\lambda)$ to obtain the group parameters $\mathsf{par} = (p, e, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2)$. Pick $\alpha, b_1, b_2 \xleftarrow{\$} \mathbb{Z}_p$ and a hash function $\mathsf{H}: \{0,1\}^* \rightarrow \mathbb{G}_1$. Compute

$$\mathsf{pk} = (\mathsf{H}, \mathsf{par}, g_2^{b_1}, g_2^{b_2}, e(g_1, g_2)^\alpha), \mathsf{msk} = (\alpha, b_1, b_2).$$

$\underline{(\mathsf{sk}, \mathsf{pl}_\mathbb{A}) \leftarrow \mathsf{KeyGen}(\mathsf{pk}, \mathsf{msk}, \mathbb{A} = (\mathrm{M}, \pi, \{\pi(i)\}_{i \in [\ell]})).}$

Pick $r \xleftarrow{\$} \mathbb{Z}_p$, $\mathbf{v} \xleftarrow{\$} \mathbb{Z}_p^{n-1}$. Compute

$$\mathsf{sk}_1 = g_2^r, \mathsf{sk}_{2,i} = g_1^{\mathrm{M}_i(\alpha \| \mathbf{v})^\top} \cdot \mathsf{H}(\pi(i))^{-r},$$

$$\mathsf{sk}_{3,i} = \mathsf{H}(0 \,\|\, \pi(i))^{\frac{r}{b_1}}, \mathsf{sk}_{4,i} = \mathsf{H}(1 \,\|\, \pi(i))^{\frac{r}{b_2}}.$$

Output $\mathsf{sk} = (\mathsf{sk}_1, \{\mathsf{sk}_{2,i}, \mathsf{sk}_{3,i}, \mathsf{sk}_{4,i}\}_{i \in [\ell]})$, $\mathsf{pl}_\mathbb{A} = \mathbb{A}$.

$\underline{(\mathsf{ct}, \mathsf{pl}_\mathbb{S}) \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathbb{S} = \{u_i\}_{i \in [m]}, \mathsf{msg}).}$

Pick $s_1, s_2 \xleftarrow{\$} \mathbb{Z}_p$, let $s = s_1 + s_2$. Compute

$$\mathsf{ct}_{1,i} = \mathsf{H}(u_i)^s \cdot \mathsf{H}(0 \,\|\, u_i)^{s_1} \cdot \mathsf{H}(1 \,\|\, u_i)^{s_2}, \mathsf{ct}_2 = g_2^s,$$

$$\mathsf{ct}_3 = g_2^{b_1 s_1}, \mathsf{ct}_4 = g_2^{b_2 s_2}, \mathsf{ct}_5 = e(g_1, g_2)^{\alpha s} \cdot \mathsf{msg}.$$

Output $\mathsf{ct} = (\{\mathsf{ct}_{1,i}\}_{i \in [m]}, \mathsf{ct}_2, \mathsf{ct}_3, \mathsf{ct}_4, \mathsf{ct}_5)$, $\mathsf{pl}_\mathbb{S} = \mathbb{S}$.

$\underline{\mathsf{msg}/\bot \leftarrow \mathsf{Dec}(\mathsf{ct}, \mathsf{pl}_\mathbb{S}, \mathsf{sk}, \mathsf{pl}_\mathbb{A}).}$

Tests if there is any subset $\mathcal{I}$ that matches $\mathbb{S}$ in ct with $\mathbb{A}$ in sk. If not, return $\bot$. Otherwise, it finds constants $\{\gamma_i\}_{i \in \mathcal{I}}$ s.t. $\sum_{i \in \mathcal{I}} \gamma_i \mathrm{M}_i = (1, 0, ..., 0)$ and reconstructs the message msg by computing:

$$\frac{\mathsf{ct}_5 \cdot e\Big(\prod_{i \in \mathcal{I}} (\mathsf{sk}_{3,i})^{\gamma_i}, \mathsf{ct}_3\Big) \cdot e\Big(\prod_{i \in \mathcal{I}} (\mathsf{sk}_{4,i})^{\gamma_i}, \mathsf{ct}_4\Big)}{e\Big(\prod_{i \in \mathcal{I}} (\mathsf{ct}_{1,\pi(i)})^{\gamma_i}, \mathsf{sk}_1\Big) \cdot e\Big(\prod_{i \in \mathcal{I}} (\mathsf{sk}_{2,i})^{\gamma_i}, \mathsf{ct}_2\Big)}.$$

Figure 4: The construction of our KP-ABE scheme.

$\underline{(\mathsf{pk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda).}$

Run $\mathsf{GroupGen}(1^\lambda)$ to obtain the group parameters $\mathsf{par} = (p, e, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2)$. Pick $\alpha, b_1, b_2 \xleftarrow{\$} \mathbb{Z}_p$, $g_3 \in \mathbb{G}_1$, and a hash function $\mathsf{H}: \{0,1\}^* \rightarrow \mathbb{G}_1$. Compute

$$\mathsf{pk} = (\mathsf{H}, \mathsf{par}, g_3, g_2^{b_1}, g_2^{b_2}, e(g_1, g_2)^\alpha), \mathsf{msk} = (\alpha, b_1, b_2).$$

$\underline{(\mathsf{sk}, \mathsf{pl}_\mathbb{S}) \leftarrow \mathsf{KeyGen}(\mathsf{pk}, \mathsf{msk}, \mathbb{S} = \{u_i\}_{i \in [m]}).}$

Pick $r \xleftarrow{\$} \mathbb{Z}_p$. Compute

$$\mathsf{sk}_1 = g_2^r, \mathsf{sk}_2 = g_1^\alpha g_3^{-r},$$

$$\mathsf{sk}_{3,i} = \mathsf{H}(0 \,\|\, u_i)^{\frac{r}{b_1}}, \mathsf{sk}_{4,i} = \mathsf{H}(1 \,\|\, u_i)^{\frac{r}{b_2}}.$$

Output $\mathsf{sk} = (\mathsf{sk}_1, \mathsf{sk}_2, \{\mathsf{sk}_{3,i}, \mathsf{sk}_{4,i}\}_{i \in [m]})$, $\mathsf{pl}_\mathbb{S} = \mathbb{S}$.

$\underline{(\mathsf{ct}, \mathsf{pl}_\mathbb{A}) \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathbb{A} = (\mathrm{M}, \pi, \{\pi(i)\}_{i \in [\ell]}, \mathsf{msg}).}$

Pick $s_1, s_2 \xleftarrow{\$} \mathbb{Z}_p$, $\mathbf{v} \xleftarrow{\$} \mathbb{Z}_p^{n-1}$, let $s = s_1 + s_2$. Compute

$$\mathsf{ct}_{1,i} = g_3^{\mathrm{M}_i(s \| \mathbf{v})^\top} \cdot \mathsf{H}(0 \,\|\, \pi(i))^{s_1} \cdot \mathsf{H}(1 \,\|\, \pi(i))^{s_2}, \mathsf{ct}_2 = g_2^s,$$

$$\mathsf{ct}_3 = g_2^{b_1 s_1}, \mathsf{ct}_4 = g_2^{b_2 s_2}, \mathsf{ct}_5 = e(g_1, g_2)^{\alpha s} \cdot \mathsf{msg}.$$

Output $\mathsf{ct} = (\{\mathsf{ct}_{1,i}\}_{i \in [\ell]}, \mathsf{ct}_2, \mathsf{ct}_3, \mathsf{ct}_4, \mathsf{ct}_5)$, $\mathsf{pl}_\mathbb{A} = \mathbb{A}$.

$\underline{\mathsf{msg}/\bot \leftarrow \mathsf{Dec}(\mathsf{ct}, \mathsf{pl}_\mathbb{A}, \mathsf{sk}, \mathsf{pl}_\mathbb{S}).}$

Tests if there is any subset $\mathcal{I}$ that matches $\mathbb{S}$ in sk with $\mathbb{A}$ in ct. If not, return $\bot$. Otherwise, it finds constants $\{\gamma_i\}_{i \in \mathcal{I}}$ s.t. $\sum_{i \in \mathcal{I}} \gamma_i \mathrm{M}_i = (1, 0, ..., 0)$ and reconstructs the message msg by computing:

$$\frac{\mathsf{ct}_5 \cdot e\Big(\prod_{i \in \mathcal{I}} (\mathsf{sk}_{3,i})^{\gamma_i}, \mathsf{ct}_3\Big) \cdot e\Big(\prod_{i \in \mathcal{I}} (\mathsf{sk}_{4,i})^{\gamma_i}, \mathsf{ct}_4\Big)}{e\Big(\prod_{i \in \mathcal{I}} (\mathsf{ct}_{1,\pi(i)})^{\gamma_i}, \mathsf{sk}_1\Big) \cdot e\Big(\mathsf{sk}_2, \mathsf{ct}_2\Big)}.$$

Figure 5: The construction of our CP-ABE scheme.

exponents $\frac{1}{b_1}$ and $\frac{1}{b_2}$ are on both $g_1^{\mathrm{M}_i(\alpha \| \mathbf{v})^\top}$ and $\mathsf{H}(\pi(i))^r$ in $\mathsf{sk}_{2,i}$ and $\mathsf{sk}_{3,i}$. In the security reduction, the hash function H can be modeled as a random oracle that simulates the attribute values by using the DLIN tuple listed in Sec. 2.3, but the term $g_1^{\frac{1}{b_j} \cdot \mathrm{M}_i(\alpha \| \mathbf{v})^\top}$ for $j = 1, 2$ cannot be simulated. Thus, we change $\mathsf{sk}_{2,i}$ and $\mathsf{sk}_{3,i}$ respectively into $g_1^{\mathrm{M}_i(\alpha \| \mathbf{v})^\top} \mathsf{H}(\pi(i))^{\frac{r}{b_1}}$ and $g_1^{\mathrm{M}_i(\alpha \| \mathbf{v})^\top} \mathsf{H}(\pi(i))^{\frac{r}{b_2}}$, and change ct_4 into $e(g_1, g_2)^{\alpha b_1 s_1 + \alpha b_2 s_2}$ such that the exponents $\frac{1}{b_1}$ and $\frac{1}{b_2}$ only occur on random oracle H while the decryption is correct. Then we apply the design technique used in FAME [3] that separates random oracles to simulate different results. In specific, if we simulate the random oracle $\mathsf{H}(0 \,\|\, \pi(i)) = g_1^{b_1 \cdot t_i}$ and $\mathsf{H}(1 \,\|\, \pi(i)) = g_1^{b_2 \cdot \delta_i}$ where $g_1^{b_1}$ and $g_1^{b_2}$ are from the DLIN tuple, t_i and δ_i are randomly chosen for each $0 \,\|\, \pi(i)$ and $1 \,\|\, \pi(i)$ respectively, then the $\frac{1}{b_1}$ and $\frac{1}{b_2}$ on exponents could be eliminated by the $g_1^{b_1}, g_1^{b_2}$ simulated from the random oracle. After that, we found that the term $\mathsf{H}(u_i)^s$ in $\mathsf{ct}_{1,i}$ no longer matched with the ones in $\mathsf{sk}_{2,i}$ and $\mathsf{sk}_{3,i}$ for decryption as their hash inputs are different. Thus, we change $\mathsf{ct}_{1,i}$ into $\mathsf{H}(0 \,\|\, u_i)^{s_1} \mathsf{H}(1 \,\|\, u_i)^{s_2}$ so that the ciphertext can be decrypted successfully, while $\mathsf{ct}_{1,i}$ can be simulated as $g_1^{b_1 s_1 t_i} \cdot g_1^{b_2 s_2 \delta_i}$ in the security reduction where $g_1^{b_1 s_1}$ and $g_1^{b_2 s_2}$ are from the DLIN tuple. Nevertheless, $\mathsf{ct}_{1,i}$ must be indistinguishable from the DLIN hard problem terms g_1^s or g_2^s ($s = s_1 + s_2$), which do not exist on $\mathsf{ct}_{1,i}$. Therefore, we multiply $\mathsf{H}(u_i)^s$ to $\mathsf{ct}_{1,i}$ to make it complete, i.e., $\mathsf{ct}_{1,i} = \mathsf{H}(u_i)^s \cdot \mathsf{H}(0 \,\|\, u_i)^{s_1} \mathsf{H}(1 \,\|\, u_i)^{s_2}$. In this case, if $\mathsf{H}(u_i)$ is simulated at a random point on $\mathbb{G}_1$, $\mathsf{H}(u_i)^s$ can be associated with g_1^s. After making the scheme correct, the secret key becomes $\mathsf{sk}_1 = g_2^r$, $\mathsf{sk}_2 = \mathsf{H}(\pi(i))^{-r}$, $\mathsf{sk}_{3,i} = g_1^{\mathrm{M}_i(\alpha \| \mathbf{v})^\top} \mathsf{H}(0 \,\|\, \pi(i))^{\frac{r}{b_1}}$, $\mathsf{sk}_{4,i} = g_1^{\mathrm{M}_i(\alpha \| \mathbf{v})^\top} \mathsf{H}(1 \,\|\, \pi(i))^{\frac{r}{b_2}}$. The construction of the ciphertext becomes $\mathsf{ct}_{1,i} = \mathsf{H}(u_i)^s \cdot \mathsf{H}(0 \,\|\, u_i)^{s_1} \mathsf{H}(1 \,\|\, u_i)^{s_2}$, $\mathsf{ct}_2 = g_2^s$, $\mathsf{ct}_3 = g_2^{b_1 s_1}$, $\mathsf{ct}_4 = g_2^{b_2 s_2}$, $\mathsf{ct}_5 = e(g_1, g_2)^{\alpha b_1 s_1 + \alpha b_2 s_2} \cdot \mathsf{msg}$.

Now the attribute term $\mathsf{ct}_{1,i}$ can be reduced to the DLIN assumption, the next step is to reduce the message term ct_5 into the DLIN hard problem. It is clear that we can simulate ct_5 by using the DLIN tuples $g_2^{b_1 s_1}$ and $g_2^{b_2 s_2}$, but the DLIN hard problem terms g_1^s or g_2^s ($s = s_1 + s_2$) do not exist on ct_5. Therefore, our target is to change ct_5 as the message multiplied with $e(g_1, g_2)^{\alpha s}$. Through the above observation, we believe that it is difficult to achieve this form if α is on the exponents of $\mathsf{sk}_{3,i}$ and $\mathsf{sk}_{4,i}$. Thus, our technique is to move the secret sharing part $g_1^{\mathrm{M}_i(\alpha \| \mathbf{v})^\top}$ from $\mathsf{sk}_{3,i}, \mathsf{sk}_{4,i}$ to $\mathsf{sk}_{2,i}$. i.e., $\mathsf{sk}_{2,i} = g_1^{\mathrm{M}_i(\alpha \| \mathbf{v})^\top} \cdot \mathsf{H}(\pi(i))^{-r}$. In this case, we simultaneously achieve that (1) the pairing between the new $\mathsf{sk}_{2,i}$ and ct_2 can recover the $e(g_1, g_2)^{\alpha s}$, and (2) The connection between the secret sharing term $g_1^{\mathrm{M}_i(\alpha \| \mathbf{v})^\top}$ and the attribute terms $\mathsf{H}(0 \| \pi(i))^{\frac{r}{b_1}}$, $\mathsf{H}(1 \| \pi(i))^{\frac{r}{b_2}}$ is bridged through the randomness r. The final version of our KP-ABE scheme is presented in Fig. 4.

Natural transformation to CP-ABE. Finally, it is easy to obtain our CP-ABE scheme from our KP-ABE scheme in terms of the natural difference in their syntax: the access policy should be associated with the ciphertext rather than the secret key. Accordingly, the only change is to set the $\mathsf{H}(u_i)^s$ in $\mathsf{ct}_{1,i}$ with the access policy by making it into the secret share values. i.e., $\mathsf{ct}_{1,i} = g_3^{\mathsf{M}_i(s\|\mathsf{v})^\top}\mathsf{H}(0 \| u_i)^{s_1}\mathsf{H}(1 \| u_i)^{s_2}$ where $g_3 \in \mathbb{G}_1$, and correspondingly remove the secret sharing and hashes in $\mathsf{sk}_{2,i}$. Meanwhile, we maintain the security arguments on both ciphertext anonymity and IND-CPA for our CP-A^{2}BE scheme. We present our CP-ABE scheme in Fig 5.

3.3 From our ABE to our A^{2}BE schemes

As discussed in Sec. 3.2, our ABE schemes have already achieved ciphertext anonymity, so they can be transformed into the corresponding A^{2}BE schemes by simply adopting the partially hidden structure in Sec. 2.2 for payload privacy. The key changes made to our ABE schemes are listed as follows:

(1) Each attribute $\pi(i)$ in access policy $\mathbb{A}$ is separated into a name $n_{\pi(i)}$ and a value $v_{\pi(i)}$, i.e., $\mathbb{A} = (\mathsf{M}, \pi, \{\pi(i)\}_{i\in[\ell]} = \{n_{\pi(i)}, v_{\pi(i)}\}_{i\in[\ell]})$. The payload $\mathsf{pl}_{\mathbb{A}} = (\mathsf{M}, \pi, \{n_{\pi(i)}\}_{i\in[\ell]})$.
(2) Each attribute u_i in an attribute set $\mathbb{S}$ is separated into a name n_i and a value v_i, i.e., $\mathbb{S} = \{u_i\}_{i\in[m]} = \{n_i, v_i\}_{i\in[m]}$. Then the payload $\mathsf{pl}_{\mathbb{S}} = \{n_i\}_{i\in[m]}$.
(3) In decryption algorithm, first to test if there is any subset $\mathcal{I}$ that matches $\{n_i\}_{i\in[m]}$ with $(\mathsf{M}, \pi, \{n_{\pi(i)}\}_{i\in[\ell]})$. If not, return $\perp$. Otherwise, it finds constants $\{\gamma_i\}_{i\in\mathcal{I}}$ s.t. $\sum_{i\in\mathcal{I}}\gamma_i\mathsf{M}_i = (1, 0, ..., 0)$ and reconstructs the message msg by using the same equation as shown in Fig. 4 and Fig. 5. If the message is not correct, find another subset of $\mathcal{I}$ and repeat the checking. If the message cannot be recovered for all subsets, return $\perp$.

3.4 Further remarks

Technical novelty. When compared to FAME [3] that achieves fast ABE schemes under the DLIN assumption, and FEASE [40] that achieves ciphertext anonymity under GGM, the proposed techniques used in our schemes have the following differences and advantages:

- Compared to both: In secret key, our techniques avoid to set $\frac{1}{b_j}$ (for $j = \{1, 2\}$) as the exponents on generators g_1, g_2, which has not been used in FAME and FEASE.
- Compared to FAME: In ciphertext, our techniques directly associate the DLIN hard problem value s on both the exponents of the message term and attribute term, while FAME only associates s_1 and s_2 on these terms. Thus, our schemes enable simulation-based security reduction to DLIN rather than using the dual system framework, which leads to more efficient construction for all algorithms (Setup, KeyGen, Enc and Dec).
- Compared to FEASE: Our techniques use two different random oracles to simulate the $\frac{1}{b_j}$ (for $j = \{1, 2\}$) terms and reduce the security to DLIN assumption, while FEASE only uses one random oracle and reduce the security to GGM.

Practical features. Our schemes inherit the following features from FABEO [49] and FEASE [40]:

- Large universe of attributes: by using a hash function H that can map any attribute string to a group element in $\mathbb{G}_1$, which eliminates the need to specify attributes during setup.
- Expressiveness: by representing policies as monotone span programs (MSPs), which can encode any monotone Boolean formula without size restrictions.
- Attribute multi-use: by doing minor changes in our constructions in a similar way as in FABEO [49] (Page 4, Fig. 1). In short, the secret key (in KP-ABE) or ciphertext (in CP-ABE) should contain vectors whose lengths depend on the maximum number of times and an attribute is repeated in the policy (denoted by τ). When an attribute is reused, it will be associated with different randomness in the vector that is indexed by its occurrence number (denoted by $\rho(i)$). This results in a small overhead in the decryption process corresponding to the multi-use parameter τ. We show the details in Appendix A for readers' interest.

Design rationale. Through our techniques, we naturally obtain ABE schemes with ciphertext anonymity, which includes indistinguishability for both the message and attribute terms. Thus, we simultaneously obtain fast ABE and A^{2}BE schemes under the DLIN assumption. Since ciphertext anonymity comes as a by-product of our techniques, an interesting open question is whether our ABE scheme can be simplified further to achieve solely IND-CPA security for ABE while still relying on a standard assumption.

4 SECURITY ANALYSIS OF OUR SCHEMES

In this section, we prove the correctness, IND-CPA security and anonymity of our ABE and A^{2}BE schemes.

4.1 Correctness

For our ABE schemes in Fig. 4 and Fig. 5, we show that when $\mathbb{S}$ satisfies $\mathbb{A}$, decryption recovers the correct message with probability 1. The correctness of our A^{2}BE schemes can be proved in the same way.

Our KP-ABE scheme: Let $\mathsf{D}_1 = e(\prod_{i\in\mathcal{I}}(\mathsf{ct}_{1,\pi(i)})^{\gamma_i}, \mathsf{sk}_1)$, we have

$$\mathsf{D}_1 = e(\prod_{i\in\mathcal{I}}(\mathsf{H}(u_i)^s \cdot \mathsf{H}(0 \| u_i)^{s_1} \cdot \mathsf{H}(1 \| u_i)^{s_2})^{\gamma_i}, g_2^r) = e(\mathsf{H}(u_i)^{\sum_{i\in\mathcal{I}} sr\gamma_i}, g_2) \cdot e(\mathsf{H}(0 \| u_i)^{\sum_{i\in\mathcal{I}} s_1 r\gamma_i}, g_2) \cdot e(\mathsf{H}(1 \| u_i)^{\sum_{i\in\mathcal{I}} s_2 r\gamma_i}, g_2).$$

Let $\mathsf{D}_2 = e(\prod_{i\in\mathcal{I}}(\mathsf{sk}_{2,i})^{\gamma_i}, \mathsf{ct}_2)$, we have

$$\begin{aligned}\mathsf{D}_2 &= e(\prod_{i\in\mathcal{I}}(g_1^{\mathsf{M}_i(\alpha\|\mathsf{v})^\top} \cdot \mathsf{H}(\pi(i))^{-r})^{\gamma_i}, g_2^s) \\ &= e(g_1, g_2)^{\alpha s} \cdot e(\mathsf{H}(\pi(i))^{-\sum_{i\in\mathcal{I}} sr\gamma_i}, g_2).\end{aligned}$$

Let $\mathsf{D}_3 = e(\prod_{i\in\mathcal{I}}(\mathsf{sk}_{3,i})^{\gamma_i}, \mathsf{ct}_3)$, we have

$$\mathsf{D}_3 = e(\prod_{i\in\mathcal{I}}\mathsf{H}(0 \| \pi(i))^{\frac{r}{b_1}\cdot\gamma_i}, g_2^{b_1 s_1}) = e(\mathsf{H}(0 \| \pi(i))^{\sum_{i\in\mathcal{I}} s_1 r\gamma_i}, g_2).$$

Let $\mathsf{D}_4 = e(\prod_{i\in\mathcal{I}}(\mathsf{sk}_{4,i})^{\gamma_i}, \mathsf{ct}_4)$, we have

$$\mathsf{D}_4 = e(\prod_{i\in\mathcal{I}}\mathsf{H}(1 \| \pi(i))^{\frac{r}{b_2}\cdot\gamma_i}, g_2^{b_2 s_2}) = e(\mathsf{H}(1 \| \pi(i))^{\sum_{i\in\mathcal{I}} s_2 r\gamma_i}, g_2).$$

Finally, the decryption works as $\frac{\mathsf{ct}_5\cdot\mathsf{D}_3\cdot\mathsf{D}_4}{\mathsf{D}_1\cdot\mathsf{D}_2} = \mathsf{msg}$.

Our CP-ABE scheme: Let $\mathsf{D}_1 = e(\prod_{i\in\mathcal{I}}(\mathsf{ct}_{1,\pi(i)})^{\gamma_i}, \mathsf{sk}_1)$, we have

$$\mathsf{D}_1 = e(\prod_{i\in\mathcal{I}}(g_3^{\mathsf{M}_i(s\|\mathbf{v})^\top}\cdot\mathsf{H}(0\|\pi(i))^{s_1}\cdot\mathsf{H}(1\|\pi(i))^{s_2})^{\gamma_i}, g_2^r)$$
$$= e(g_3^{sr}\cdot\mathsf{H}(0\|\pi(i))^{\sum_{i\in\mathcal{I}}s_1 r\gamma_i}\cdot\mathsf{H}(1\|\pi(i))^{\sum_{i\in\mathcal{I}}s_2 r\gamma_i}, g_2).$$

Let $\mathsf{D}_2 = e(\mathsf{sk}_2, \mathsf{ct}_2) = e(g_1^\alpha g_3^{-r}, g_2^s) = e(g_1, g_2)^{\alpha s}\cdot e(g_3, g_2)^{-rs}$.
Let $\mathsf{D}_3 = e(\prod_{i\in\mathcal{I}}(\mathsf{sk}_{3,i})^{\gamma_i}, \mathsf{ct}_3)$, we have

$$\mathsf{D}_3 = e(\prod_{i\in\mathcal{I}}\mathsf{H}(0\|u_i)^{\frac{r}{b_1}\cdot\gamma_i}, g_2^{b_1 s_1}) = e(\mathsf{H}(0\|u_i)^{\sum_{i\in\mathcal{I}}s_1 r\gamma_i}, g_2).$$

Let $\mathsf{D}_4 = e(\prod_{i\in\mathcal{I}}(\mathsf{sk}_{4,i})^{\gamma_i}, \mathsf{ct}_4)$, we have

$$\mathsf{D}_4 = e(\prod_{i\in\mathcal{I}}\mathsf{H}(1\|u_i)^{\frac{r}{b_2}\cdot\gamma_i}, g_2^{b_2 s_2}) = e(\mathsf{H}(1\|u_i)^{\sum_{i\in\mathcal{I}}s_2 r\gamma_i}, g_2).$$

Finally, the decryption works as $\frac{\mathsf{ct}_5\cdot\mathsf{D}_3\cdot\mathsf{D}_4}{\mathsf{D}_1\cdot\mathsf{D}_2} = \mathsf{msg}$.

4.2 IND-CPA security and anonymity

Then we prove the IND-CPA security of our ABE schemes and the anonymity of our A^{2}BE schemes by following the security model defined in Sec. 2.4.

THEOREM 1. *Our KP-ABE scheme is adaptively IND-CPA secure in the random oracle model under the DLIN assumption.*

PROOF. Suppose there exists a PPT adversary $\mathcal{A}$ that can break our KP-ABE scheme in the IND-CPA game with advantage ϵ. We build an algorithm $\mathcal{B}$ that solves the DLIN problem by advantage $\epsilon/2$. Algorithm $\mathcal{B}$ is given a DLIN tuple, where we replace the variable names in the DLIN tuple defined in Sec. 2.3 into the same notation as our construction as follows:

$$x_1 = b_1, x_2 = b_2, y_1 = s_1, y_2 = s_2.$$

Then the tuple $(A, B, C, D, E, F, G, H, I, J, Y, Z)$ given to $\mathcal{B}$ is defined as:

$$A = g_1, B = g_2, C = g_1^{b_1}, D = g_1^{b_2}, E = g_2^{b_1}, F = g_2^{b_2},$$
$$G = g_1^{b_1 s_1}, H = g_1^{b_2 s_2}, I = g_2^{b_1 s_1}, J = g_2^{b_2 s_2}, Y, Z.$$

Let $s = s_1 + s_2$, $R \xleftarrow{\$} \mathbb{Z}_p$. The challenger flips a random binary coin μ outside of $\mathcal{B}$'s view. If $\mu = 0$, the challenger sets $Y = g_1^s, Z = g_2^s$. If $\mu = 1$, the challenger sets $Y = g_1^R, Z = g_2^R$. $\mathcal{B}$'s goal is to output 0 if $Y = g_1^s$ and $Z = g_2^s$, or output 1 otherwise. $\mathcal{B}$ works by interacting with $\mathcal{A}$ in a game as follows:

Setup. To generate the system parameters, $\mathcal{B}$ picks $\alpha \xleftarrow{\$} \mathbb{Z}_p$. Then $\mathcal{B}$ gives $\mathcal{A}$ the public key

$$\mathsf{pk} = (\mathsf{H}, p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, A, B, E, F, e(A, B)^\alpha).$$

and keeps the master secret key $\mathsf{msk} = \alpha$ unknown to $\mathcal{A}$.

Phase 1. $\mathcal{A}$ can query the random oracle $\mathsf{H}(\cdot)$ and the key generation oracle $\mathcal{O}_K(\cdot)$. They are defined as follows:

Random oracle $\mathsf{H}(\cdot)$: $\mathcal{B}$ maintains a list P with entries of the form $< x_i, h_i, k_i >$, a list L with entries $< 0\|x_i, h_{0i}, t_i >$ and a list Q with entries $< 1\|x_i, h_{1i}, \delta_i >$. The list P, L and Q are initially empty. The adversary $\mathcal{A}$ can make one of the three types of oracle queries as follows:

(1) x_i: $\mathcal{B}$ checks if x_i already appears on the list P in a tuple $< x_i, h_i, k_i >$. If yes, then $\mathcal{B}$ responds with $\mathsf{H}(x_i) = h_i \in \mathbb{G}_1$. Otherwise, $\mathcal{B}$ picks $k_i \xleftarrow{\$} \mathbb{Z}_p$ and computes $h_i \leftarrow A^{k_i} \in \mathbb{G}_1$. Then $\mathcal{B}$ adds the tuple $< x_i, h_i, k_i >$ to list L and responds to $\mathcal{A}$ by setting $\mathsf{H}(x_i) = h_i$. h_i is uniform in $\mathbb{G}_1$ and is independent of $\mathcal{A}$'s current view as required.

(2) $0 \| x_i$: $\mathcal{B}$ checks if $0 \| x_i$ already appears on the list L in a tuple $< 0 \| x_i, h_{0i}, t_i >$. If yes, then $\mathcal{B}$ responds with $\mathsf{H}(0\|x_i) = h_{0i} \in \mathbb{G}_1$. Otherwise, $\mathcal{B}$ picks $t_i \xleftarrow{\$} \mathbb{Z}_p$ and computes $h_{0i} \leftarrow C^{t_i} \in \mathbb{G}_1$. Then $\mathcal{B}$ adds the tuple $< 0\|x_i, h_{0i}, t_i >$ to list L and responds to $\mathcal{A}$ by setting $\mathsf{H}(0\|x_i) = h_{0i}$. h_{0i} is uniform in $\mathbb{G}_1$ and is independent of $\mathcal{A}$'s current view as required.

(3) $1 \| x_i$: $\mathcal{B}$ checks if $1 \| x_i$ already appears on the list Q in a tuple $< 1 \| x_i, h_{1i}, \delta_i >$. If yes, then $\mathcal{B}$ responds with $\mathsf{H}(1 \| x_i) = h_{1i} \in \mathbb{G}_1$. Otherwise, $\mathcal{B}$ picks $\delta_i \xleftarrow{\$} \mathbb{Z}_p$ and computes $h_{1i} \leftarrow D^{\delta_i} \in \mathbb{G}_1$. Then $\mathcal{B}$ adds the tuple $< 1\|x_i, h_{1i}, \delta_i >$ to list Q and responds to $\mathcal{A}$ by setting $\mathsf{H}(1\|x_i) = h_{1i}$. h_{1i} is uniform in $\mathbb{G}_1$ and is independent of $\mathcal{A}$'s current view as required.

Key generation oracle $\mathcal{O}_K(\cdot)$: When $\mathcal{A}$ issues a query for the secret key of an access policy $\mathbb{A} = (\mathsf{M}, \pi, \{\pi(i)\}_{i\in[\ell]})$. $\mathcal{B}$ runs the random oracle $\mathsf{H}(\cdot)$ to obtain $h_i, h_{0i}, h_{1i} \in \mathbb{G}_1$ for each attribute $\pi(i)$ such that $\mathsf{H}(\pi(i)) = h_i$, $\mathsf{H}(0 \| \pi(i)) = h_{0i}$ and $\mathsf{H}(1 \| \pi(i)) = h_{1i}$. Let $< \pi(i), h_i, k_i >$, $< 0\|\pi(i), h_{0i}, t_i >$ and $< 1\|\pi(i), h_{1i}, \delta_i >$ be the corresponding tuple on list P, L and Q respectively. Then $\mathcal{B}$ chooses $r \xleftarrow{\$} \mathbb{Z}_p$ and a vector $\mathbf{v} \xleftarrow{\$} \mathbb{Z}_p^{n-1}$, and simulates the secret key as follows:

$$\mathsf{sk}_1 = B^r, \mathsf{sk}_{2,i} = A^{\mathsf{M}_i(\alpha\|\mathbf{v})^\top - k_i r}, \mathsf{sk}_{3,i} = A^{t_i r}, \mathsf{sk}_{4,i} = A^{\delta_i r}.$$

We can see that the exponents $\frac{1}{b_1}$, $\frac{1}{b_2}$ in $\mathsf{sk}_{3,i}$ and $\mathsf{sk}_{4,i}$ are canceled by the random oracle outputs C^{t_i} and D^{δ_i} respectively, in which $\mathsf{H}(0 \| \pi(i))^{\frac{r}{b_1}} = C^{\frac{t_i r}{b_1}} = A^{t_i r}$, $\mathsf{H}(1 \| \pi(i))^{\frac{r}{b_2}} = D^{\frac{\delta_i r}{b_2}} = A^{\delta_i r}$, and all other elements can be directly simulated by $\mathcal{B}$ by using the known values, so the secret key $\mathsf{sk} = (\mathsf{sk}_1, \{\mathsf{sk}_{2,i}, \mathsf{sk}_{3,i}, \mathsf{sk}_{4,i}\}_{i\in[\ell]})$ is valid for the access policy $\mathbb{A}$. Then $\mathcal{B}$ gives $(\mathsf{sk}, \mathbb{A})$ to $\mathcal{A}$.

Challenge. $\mathcal{A}$ outputs two equal-size messages msg_0^*, msg_1^* and an attribute set $\mathbb{S}^* = \{x_i^*\}_{i\in[m]}$ that it intends to attack. $\mathcal{B}$ checks if $\mathbb{S}^*$ satisfies any access policy $\mathbb{A}$ queried in Phase 1. If yes, then $\mathcal{B}$ rejects $\mathbb{S}^*$. Otherwise, $\mathcal{B}$ randomly picks $\beta \in \{0, 1\}$ and chooses to encrypt a message msg_β^* with attribute set $\mathbb{S}^*$. In specific, $\mathcal{B}$ first runs the random oracle $\mathsf{H}(\cdot)$ to obtain $h_i^*, h_{0i}^*, h_{1i}^* \in \mathbb{G}_1$ for each attribute value $x_i^* \in \mathbb{S}^*$ such that $\mathsf{H}(x_i^*) = h_i^*$, $\mathsf{H}(0\|x_i^*) = h_{0i}^*$ and $\mathsf{H}(1\|x_i^*) = h_{1i}^*$. Let $< x_i^*, h_i^*, k_i^* >$, $< 0 \| x_i^*, h_{0i}^*, t_i^* >$ and $< 1 \| x_i^*, h_{1i}^*, \delta_i^* >$ be the corresponding tuple on list P, L and Q respectively, $\mathcal{B}$ simulates the challenge ciphertext $\mathsf{ct}_\beta^* = (\{\mathsf{ct}_{1,i}\}_{i\in[m]}, \mathsf{ct}_2, \mathsf{ct}_3, \mathsf{ct}_4, \mathsf{ct}_5)$ as follows:

$$\mathsf{ct}_{1,i} = Y^{k_i^*}\cdot G^{t_i^*}\cdot H^{\delta_i^*}, \mathsf{ct}_2 = Z, \mathsf{ct}_3 = I, \mathsf{ct}_4 = J, \mathsf{ct}_5 = e(Y, B)^\alpha\cdot\mathsf{msg}_\beta^*.$$

We can see that $\mathsf{H}(0\|x_i^*)^{s_1} = C^{t_i^* s_1} = G^{t_i^*}$, $\mathsf{H}(1\|x_i^*)^{s_2} = D^{\delta_i^* s_2} = H^{\delta_i^*}$. If $\mu = 0$, then $Y = g_1^s, Z = g_2^s$, $Y^{k_i^*} = (g_1^s)^{k_i^*}$, $e(Y, B)^\alpha = e(g_1, g_2)^{\alpha s}$. Thus, the ciphertext ct_β^* is a valid random encryption for attribute set $\mathbb{S}^*$ and message msg_β^*. If $\mu = 1$, then $Y = g_1^R, Z = g_2^R$, $\mathsf{ct}_5 = e(g_1, g_2)^{\alpha R}\cdot\mathsf{msg}_\beta^*$. As R is random, ct_5 is a random element on $\mathbb{G}_T$ from $\mathbb{A}$'s view and ct_β^* contains no information about msg_β^*. Then $\mathcal{B}$ gives $(\mathsf{ct}_\beta^*, \mathbb{S}^*)$ to $\mathcal{A}$.

Phase 2. Same as Phase 1 with the restriction that any input access policy $\mathbb{A}$ can not be satisfied by $\mathbb{S}^*$.

Guess. $\mathcal{A}$ outputs a bit $\beta' \in \{0, 1\}$. If $\beta' = \beta$, $\mathcal{B}$ outputs $\mu' = 0$ indicating (Y, Z) is a valid DLIN tuple. Otherwise it outputs $\mu' = 1$ indicating (Y, Z) is given as a random tuple. In the case of $\mu = 1$,

$\mathcal{A}$ gains no information about β, so we have $\Pr[\beta \neq \beta' | \mu = 1] = \frac{1}{2}$. As $\mathcal{B}$ guesses $\mu = 1$ when $\beta \neq \beta'$, we have $\Pr[\mu' = \mu | \mu = 1] = \frac{1}{2}$. If $\mu = 0$, then $\mathcal{A}$ sees an encryption of msg^*_β, with an advantage ϵ of breaking the scheme. Thus, we have $\Pr[\beta = \beta' | \mu = 0] = \frac{1}{2} + \epsilon$. As $\mathcal{B}$ guesses $\mu = 0$ when $\beta = \beta'$, we have $\Pr[\mu' = \mu | \mu = 0] = \frac{1}{2} + \epsilon$. Therefore, the overall advantage of $\mathcal{B}$ solves the DLIN problem is $\frac{1}{2}\Pr[\mu' = \mu | \mu = 0] + \frac{1}{2}\Pr[\mu' = \mu | \mu = 1] - \frac{1}{2} = \frac{1}{2} \cdot (\frac{1}{2} + \epsilon) + \frac{1}{2} \cdot \frac{1}{2} - \frac{1}{2} = \frac{\epsilon}{2}$. Then we complete the proof of Theorem 1. □

THEOREM 2. *Our CP-ABE scheme is adaptively IND-CPA secure in the random oracle model under the DLIN assumption.*

PROOF. Suppose there exists a PPT adversary $\mathcal{A}$ that can break our CP-ABE scheme in the IND-CPA game with advantage ϵ. We build an algorithm $\mathcal{B}$ that solves the DLIN problem by advantage $\epsilon/2$. Same as before, $\mathcal{B}$ is given a DLIN tuple defined as the following:

$$A = g_1,\, B = g_2,\, C = g_1^{b_1},\, D = g_1^{b_2},\, E = g_2^{b_1}, F = g_2^{b_2},$$
$$G = g_1^{b_1 s_1},\, H = g_1^{b_2 s_2},\, I = g_2^{b_1 s_1},\, J = g_2^{b_2 s_2},\, Y,\, Z.$$

Let $s = s_1 + s_2$, $R \xleftarrow{\$} \mathbb{Z}_p$. The challenger flips a random binary coin μ outside of $\mathcal{B}$'s view. If $\mu = 0$, the challenger sets $Y = g_1^s, Z = g_2^s$. If $\mu = 1$, the challenger sets $Y = g_1^R, Z = g_2^R$. $\mathcal{B}$'s goal is to output 0 if $Y = g_1^s$ and $Z = g_2^s$, or output 1 otherwise. $\mathcal{B}$ works by interacting with $\mathcal{A}$ in a game as follows:

Setup. To generate the system parameters, $\mathcal{B}$ picks $\alpha,\, c \xleftarrow{\$} \mathbb{Z}_p$. Then $\mathcal{B}$ gives $\mathcal{A}$ the public key

$$\mathsf{pk} = (\mathsf{H}, p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, A, B, A^c, E, F, e(A, B)^\alpha).$$

and keeps the master secret key $\mathsf{msk} = \alpha$ unknown to $\mathcal{A}$.

Phase 1. $\mathcal{A}$ can query the random oracle $\mathsf{H}(\cdot)$ and the key generation oracle $\mathcal{O}_K(\cdot)$. They are defined as follows:

Random oracle $\mathsf{H}(\cdot)$: The adversary $\mathcal{A}$ can make queries for $0 \| x_i$ or $1 \| x_i$ that have been defined in proof of theorem 1.

Key generation oracle $\mathcal{O}_K(\cdot)$: When $\mathcal{A}$ issues a query for the secret key of an attribute set $\mathbb{S} = \{x_i\}_{i \in [m]}$. $\mathcal{B}$ runs the random oracle $\mathsf{H}(\cdot)$ to obtain $h_{0i}, h_{1i} \in \mathbb{G}_1$ for each attribute x_i such that $\mathsf{H}(0 \| x_i) = h_{0i}$ and $\mathsf{H}(1 \| x_i) = h_{1i}$. Let $< 0 \| x_i, h_{0i}, t_i >$ and $< 1 \| x_i, h_{1i}, \delta_i >$ be the corresponding tuple on list L and Q respectively. Then $\mathcal{B}$ chooses $r \xleftarrow{\$} \mathbb{Z}_p$, and simulates the secret key as follows:

$$\mathsf{sk}_1 = B^r,\, \mathsf{sk}_2 = A^{\alpha - cr},\, \mathsf{sk}_{3,i} = A^{t_i r},\, \mathsf{sk}_{4,i} = A^{\delta_i r}.$$

We can see that the exponents $\frac{1}{b_1}, \frac{1}{b_2}$ in $\mathsf{sk}_{3,i}$ and $\mathsf{sk}_{4,i}$ are canceled by the random oracle outputs C^{t_i} and D^{δ_i} respectively, in which $\mathsf{H}(0 \| x_i)^{\frac{r}{b_1}} = C^{\frac{t_i r}{b_1}} = A^{t_i r}$, $\mathsf{H}(1 \| x_i)^{\frac{r}{b_2}} = D^{\frac{\delta_i r}{b_2}} = A^{\delta_i r}$, and all other elements can be directly simulated by $\mathcal{B}$ by using the known values, so the secret key $\mathsf{sk} = (\mathsf{sk}_1, \mathsf{sk}_2, \{\mathsf{sk}_{3,i}, \mathsf{sk}_{4,i}\}_{i \in [m]})$ is valid for the attribute set $\mathbb{S}$. Then $\mathcal{B}$ gives $(\mathsf{sk}, \mathbb{S})$ to $\mathcal{A}$.

Challenge. $\mathcal{A}$ outputs two equal-size messages msg^*_0, msg^*_1 and an access policies $\mathbb{A}^* = (\mathrm{M}^*, \pi^*, \{\pi(i)^*\}_{i \in [\ell]})$ that it intends to attack. $\mathcal{B}$ checks if $\mathbb{A}^*$ can be satisfied by any attribute set $\mathbb{S}$ queried from the $\mathcal{O}_K$ in Phase 1. If yes, $\mathcal{B}$ rejects $\mathbb{A}^*$. Otherwise, $\mathcal{B}$ randomly picks $\beta \in \{0, 1\}$ and chooses to encrypt a message msg^*_β with access policy $\mathbb{A}^*$. In specific, $\mathcal{B}$ first runs the random oracle $\mathsf{H}(\cdot)$ to obtain $h^*_{0i}, h^*_{1i} \in \mathbb{G}_1$ for each attribute value $\pi(i)^* \in \mathbb{A}^*$ such that $\mathsf{H}(0 \| \pi(i)^*) = h^*_{0i}$ and $\mathsf{H}(1 \| \pi(i)^*) = h^*_{1i}$. Then $\mathcal{B}$ picks a vector $\mathbf{v} \xleftarrow{\$} \mathbb{Z}_p^{n-1}$. Let $< 0 \| \pi(i)^*, h^*_{0i}, t^*_i >$ and $< 1 \| \pi(i)^*, h^*_{1i}, \delta^*_i >$ be the corresponding tuple on list L and Q respectively, $\mathcal{B}$ simulates the challenge ciphertext $\mathsf{ct}^*_\beta = (\{\mathsf{ct}_{1,i}\}_{i \in [m]}, \mathsf{ct}_2, \mathsf{ct}_3, \mathsf{ct}_4, \mathsf{ct}_5)$ as follows:

$$\mathsf{ct}_{1,i} = Y^{c \cdot \mathrm{M}^*_{i,1}} \cdot A^{\sum_{j=2}^{n} \mathrm{M}^*_{i,j} \mathrm{v}_j} \cdot G^{t^*_i} \cdot H^{\delta^*_i}, \mathsf{ct}_2 = Z,\, \mathsf{ct}_3 = I,$$
$$\mathsf{ct}_4 = J,\, \mathsf{ct}_5 = e(Y, B)^\alpha \cdot \mathsf{msg}^*_\beta.$$

Similarly, $\mathsf{H}(0 \| \pi(i)^*)^{s_1} = C^{t_i s_1} = G^{t^*_i}$, $\mathsf{H}(1 \| \pi(i)^*)^{s_2} = D^{\delta^*_i s_2} = H^{\delta^*_i}$. If $\mu = 0$, then $Y = g_1^s, Z = g_2^s$, $Y^{c \cdot \mathrm{M}^*_{i,1}} = (g_1^s)^{c \cdot \mathrm{M}^*_{i,1}}$, $e(Y, B)^\alpha = e(g_1, g_2)^{\alpha s}$. Thus, the ciphertext ct^*_β is a valid random encryption for policy $\mathbb{A}^*$ and message msg^*_β. If $\mu = 1$, then $Y = g_1^R$, $Z = g_2^R$, $\mathsf{ct}_5 = e(g_1, g_2)^{\alpha R} \cdot \mathsf{msg}^*_\beta$. As R is random, ct_5 is a random element on $\mathbb{G}_T$ and ct^*_β contains no information about msg^*_β. Then $\mathcal{B}$ gives $(\mathsf{ct}^*_\beta, \mathbb{A}^*)$ to $\mathcal{A}$.

Phase 2. Same as Phase 1 with the restriction that any input attribute set $\mathbb{S}$ can not satisfy $\mathbb{A}^*$.

Guess. $\mathcal{A}$ outputs a bit $\beta' \in \{0, 1\}$. If $\beta' = \beta$, $\mathcal{B}$ outputs $\mu' = 0$ indicating (Y, Z) is a valid DLIN tuple. Otherwise it outputs $\mu' = 1$ indicating (Y, Z) is given as a random tuple. In the case of $\mu = 1$, $\mathcal{A}$ gains no information about β, so we have $\Pr[\beta \neq \beta' | \mu = 1] = \frac{1}{2}$. As $\mathcal{B}$ guesses $\mu = 1$ when $\beta \neq \beta'$, we have $\Pr[\mu' = \mu | \mu = 1] = \frac{1}{2}$. If $\mu = 0$, then $\mathcal{A}$ sees an encryption of msg^*_β, with an advantage ϵ of breaking the scheme. Thus, we have $\Pr[\beta = \beta' | \mu = 0] = \frac{1}{2} + \epsilon$. As $\mathcal{B}$ guesses $\mu = 0$ when $\beta = \beta'$, we have $\Pr[\mu' = \mu | \mu = 0] = \frac{1}{2} + \epsilon$. Therefore, the overall advantage of $\mathcal{B}$ solves the DLIN problem is $\frac{1}{2}\Pr[\mu' = \mu | \mu = 0] + \frac{1}{2}\Pr[\mu' = \mu | \mu = 1] - \frac{1}{2} = \frac{1}{2} \cdot (\frac{1}{2} + \epsilon) + \frac{1}{2} \cdot \frac{1}{2} - \frac{1}{2} = \frac{\epsilon}{2}$. Then we complete the proof of Theorem 2. □

THEOREM 3. *Our KP-A^{2}BE scheme is adaptively anonymous in the random oracle model under the DLIN assumption.*

PROOF. Suppose there exists a PPT adversary $\mathcal{A}$ that can break our KP-A^{2}BE scheme in the anonymity game with advantage ϵ'. We build an algorithm $\mathcal{B}$ that solves the DLIN problem by advantage $\epsilon'/2$. Same as before, $\mathcal{B}$ is given a DLIN tuple defined as the following:

$$A = g_1,\, B = g_2,\, C = g_1^{b_1},\, D = g_1^{b_2},\, E = g_2^{b_1}, F = g_2^{b_2},$$
$$G = g_1^{b_1 s_1},\, H = g_1^{b_2 s_2},\, I = g_2^{b_1 s_1},\, J = g_2^{b_2 s_2},\, Y,\, Z.$$

Let $s = s_1 + s_2$, $R \xleftarrow{\$} \mathbb{Z}_p$. The challenger flips a random binary coin μ outside of $\mathcal{B}$'s view. If $\mu = 0$, the challenger sets $Y = g_1^s, Z = g_2^s$. If $\mu = 1$, the challenger sets $Y = g_1^R, Z = g_2^R$. $\mathcal{B}$'s goal is to output 0 if $Y = g_1^s$ and $Z = g_2^s$, or output 1 otherwise. $\mathcal{B}$ works by interacting with $\mathcal{A}$ in a game as follows:

Setup. To generate the system parameters, $\mathcal{B}$ picks $\alpha \xleftarrow{\$} \mathbb{Z}_p$. Then $\mathcal{B}$ gives $\mathcal{A}$ the public key

$$\mathsf{pk} = (\mathsf{H}, p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, A, B, E, F, e(A, B)^\alpha).$$

and keeps the master secret key $\mathsf{msk} = \alpha$ unknown to $\mathcal{A}$.

Phase 1. $\mathcal{A}$ can query the random oracle $\mathsf{H}(\cdot)$ and the key generation oracle $\mathcal{O}_K(\cdot)$. They are defined as follows:

Random oracle $\mathsf{H}(\cdot)$: Same as in proof of theorem 1.

Key generation oracle $\mathcal{O}_K(\cdot)$: When $\mathcal{A}$ issues a query for the secret key of an access policy $\mathbb{A} = (\mathrm{M}, \pi, \{\pi(i)\} = \{n_{\pi(i)}, v_{\pi(i)_1}\}_{i \in [\ell]})$. $\mathcal{B}$ runs the random oracle $\mathsf{H}(\cdot)$ to obtain $h_i,\; h_{0i},\; h_{1i} \in \mathbb{G}_1$ for each attribute $\pi(i)$ such that $\mathsf{H}(\pi(i)) = h_i$, $\mathsf{H}(0 \parallel \pi(i)) = h_{0i}$ and $\mathsf{H}(1 \parallel \pi(i)) = h_{1i}$. Let $< \pi(i),\ h_i,\ k_i >$, $< 0 \parallel \pi(i),\ h_{0i},\ t_i >$ and $< 1 \parallel \pi(i),\ h_{1i},\ \delta_i >$ be the corresponding tuple on list P, L and Q respectively. Then $\mathcal{B}$ chooses $r \xleftarrow{\$} \mathbb{Z}_p$ and a vector $\mathbf{v} \xleftarrow{\$} \mathbb{Z}_p^{n-1}$, and simulates the secret key as follows:

$$\mathsf{sk}_1 = B^r,\ \mathsf{sk}_{2,i} = A^{\mathrm{M}_i(\alpha\|\mathrm{v})^\top - k_i r},\ \mathsf{sk}_{3,i} = A^{t_i r},\ \mathsf{sk}_{4,i} = A^{\delta_i r}.$$

We can see that the exponents $\frac{1}{b_1}$, $\frac{1}{b_2}$ in $\mathsf{sk}_{3,i}$ and $\mathsf{sk}_{4,i}$ are canceled by the random oracle outputs C^{t_i} and D^{δ_i} respectively, in which $\mathsf{H}(0 \parallel \pi(i))^{\frac{r}{b_1}} = C^{\frac{t_i r}{b_1}} = A^{t_i r}$, $\mathsf{H}(1 \parallel \pi(i))^{\frac{r}{b_2}} = D^{\frac{\delta_i r}{b_2}} = A^{\delta_i r}$, and all other elements can be directly simulated by $\mathcal{B}$ by using the known values, so the secret key $\mathsf{sk} = (\mathsf{sk}_1, \{\mathsf{sk}_{2,i}, \mathsf{sk}_{3,i}, \mathsf{sk}_{4,i}\}_{i\in[\ell]})$ is valid for the access policy $\mathbb{A}$. Then $\mathcal{B}$ gives $(\mathsf{sk}, (\mathrm{M}, \pi, \{n_{\pi(i)}\}_{i\in[\ell]}))$ to $\mathcal{A}$.

Challenge. $\mathcal{A}$ outputs two equal-size messages msg_0^*, msg_1^* and two attribute sets $\mathbb{S}_0^* = \{x_{i0}^*\}_{i\in[m]} = \{n_i^*, v_{i_0}^*\}_{i\in[m]}$, $\mathbb{S}_1^* = \{x_{i1}^*\}_{i\in[m]} = \{n_i^*, v_{i_1}^*\}_{i\in[m]}$ that it intends to attack. Note that $\mathbb{S}_0^*$, $\mathbb{S}_1^*$ must have the same attribute names $\{n_i^*\}_{i\in[m]}$. $\mathcal{B}$ checks if $\mathbb{S}_0^*$ or $\mathbb{S}_1^*$ satisfies any access policy $\mathbb{A}$ queried in Phase 1. If yes, then $\mathcal{B}$ rejects $\mathbb{S}_0^*$ and $\mathbb{S}_1^*$. Otherwise, $\mathcal{B}$ randomly picks $\beta \in \{0, 1\}$ and chooses to encrypt a message msg_β^* with attribute set $\mathbb{S}_\beta^*$. In specific, $\mathcal{B}$ first runs the random oracle $\mathsf{H}(\cdot)$ to obtain $h_i^*, h_{0i}^*, h_{1i}^* \in \mathbb{G}_1$ for each attribute value $x_i^* \in \mathbb{S}_\beta^*$ such that $\mathsf{H}(x_i^*) = h_i^*$, $\mathsf{H}(0 \parallel x_i^*) = h_{0i}^*$ and $\mathsf{H}(1 \parallel x_i^*) = h_{1i}^*$. Let $< x_i^*,\ h_i^*,\ k_i^* >$, $< 0 \parallel x_i^*,\ h_{0i}^*,\ t_i^* >$ and $< 1 \parallel x_i^*,\ h_{1i}^*,\ \delta_i^* >$ be the corresponding tuple on list P, L and Q respectively, $\mathcal{B}$ simulates the challenge ciphertext $\mathsf{ct}_\beta^* = (\{\mathsf{ct}_{1,i}\}_{i\in[m]}, \mathsf{ct}_2, \mathsf{ct}_3, \mathsf{ct}_4, \mathsf{ct}_5)$ as follows:

$$\mathsf{ct}_{1,i} = Y^{k_i^*} \cdot G^{t_i^*} \cdot H^{\delta_i^*},\ \mathsf{ct}_2 = Z,\ \mathsf{ct}_3 = I,\ \mathsf{ct}_4 = J,\ \mathsf{ct}_5 = e(Y, B)^\alpha \cdot \mathsf{msg}_\beta^*.$$

We can see that $\mathsf{H}(0 \parallel x_i^*)^{s_1} = C^{t_i^* s_1} = G^{t_i^*}$, $\mathsf{H}(1 \parallel x_i^*)^{s_2} = D^{\delta_i^* s_2} = H^{\delta_i^*}$. If $\mu = 0$, then $Y = g_1^s$, $Z = g_2^s$, $Y^{k_i^*} = (g_1^s)^{k_i^*}$, $e(Y, B)^\alpha = e(g_1, g_2)^{\alpha s}$. Thus, the ciphertext ct_β^* is a valid random encryption for attribute set $\mathbb{S}_\beta^*$ and message msg_β^*. If $\mu = 1$, then $Y = g_1^R$, $Z = g_2^R$, $\mathsf{ct}_{1,i} = g_1^{R\cdot k_i^*} \cdot g_1^{b_1 s_1 t_i^*} \cdot g_1^{b_2 s_2 \delta_i^*}$, $\mathsf{ct}_5 = e(g_1, g_2)^{\alpha R} \cdot \mathsf{msg}_\beta^*$. As R is random, $\mathsf{ct}_{1,i}$ are random elements on $\mathbb{G}_1$, and ct_5 is a random element on $\mathbb{G}_T$ from $\mathbb{A}$'s view hence ct_β^* contains no information about $\mathbb{S}_\beta^*$ and msg_β^* respectively. Then $\mathcal{B}$ gives $(\mathsf{ct}_\beta^*, \{n_i^*\}_{i\in[m]})$ to $\mathcal{A}$.

Phase 2. Same as Phase 1 with the restriction that any input access policy $\mathbb{A}$ can not be satisfied by $\mathbb{S}_0^*$ or $\mathbb{S}_1^*$.

Guess. $\mathcal{A}$ outputs a bit $\beta' \in \{0, 1\}$. If $\beta' = \beta$, $\mathcal{B}$ outputs $\mu' = 0$ indicating (Y, Z) is a valid DLIN tuple. Otherwise it outputs $\mu' = 1$ indicating (Y, Z) is given as a random tuple. In the case of $\mu = 1$, $\mathcal{A}$ gains no information about β, so we have $\Pr[\beta \neq \beta' | \mu = 1] = \frac{1}{2}$. As $\mathcal{B}$ guesses $\mu = 1$ when $\beta \neq \beta'$, we have $\Pr[\mu' = \mu | \mu = 1] = \frac{1}{2}$. If $\mu = 0$, then $\mathcal{A}$ sees an encryption of $\mathbb{S}_\beta^*$ and msg_β^*, with an advantage ϵ' of breaking the scheme. Thus, we have $\Pr[\beta = \beta' | \mu = 0] = \frac{1}{2} + \epsilon'$. As $\mathcal{B}$ guesses $\mu = 0$ when $\beta = \beta'$, we have $\Pr[\mu' = \mu | \mu = 0] = \frac{1}{2} + \epsilon'$. Therefore, the overall advantage of $\mathcal{B}$ solves the DLIN problem is $\frac{1}{2}\Pr[\mu' = \mu | \mu = 0] + \frac{1}{2}\Pr[\mu' = \mu | \mu = 1] - \frac{1}{2} = \frac{1}{2} \cdot (\frac{1}{2} + \epsilon') + \frac{1}{2} \cdot \frac{1}{2} - \frac{1}{2} = \frac{\epsilon'}{2}$. Then we complete the proof of Theorem 3. □

Theorem 4. *Our CP-A^{2}BE scheme is adaptively anonymous in the random oracle model under the DLIN assumption.*

Proof. Suppose there exists a PPT adversary $\mathcal{A}$ that can break our CP-A^{2}BE scheme in the anonymity game with advantage ϵ'. We build an algorithm $\mathcal{B}$ that solves the DLIN problem by advantage $\epsilon'/2$. Same as before, $\mathcal{B}$ is given a DLIN tuple defined as the following:

$$A = g_1,\ B = g_2,\ C = g_1^{b_1},\ D = g_1^{b_2},\ E = g_2^{b_1}, F = g_2^{b_2},$$
$$G = g_1^{b_1 s_1},\ H = g_1^{b_2 s_2},\ I = g_2^{b_1 s_1},\ J = g_2^{b_2 s_2},\ Y,\ Z.$$

Let $s = s_1 + s_2$, $R \xleftarrow{\$} \mathbb{Z}_p$. The challenger flips a random binary coin μ outside of $\mathcal{B}$'s view. If $\mu = 0$, the challenger sets $Y = g_1^s$, $Z = g_2^s$. If $\mu = 1$, the challenger sets $Y = g_1^R$, $Z = g_2^R$. $\mathcal{B}$'s goal is to output 0 if $Y = g_1^s$ and $Z = g_2^s$, or output 1 otherwise. $\mathcal{B}$ works by interacting with $\mathcal{A}$ in a game as follows:

Setup. To generate the system parameters, $\mathcal{B}$ picks α, $c \xleftarrow{\$} \mathbb{Z}_p$. Then $\mathcal{B}$ gives $\mathcal{A}$ the public key

$$\mathsf{pk} = (\mathsf{H}, p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, A, B, A^c, E, F, e(A, B)^\alpha).$$

and keeps the master secret key $\mathsf{msk} = \alpha$ unknown to $\mathcal{A}$.

Phase 1. $\mathcal{A}$ can query the random oracle $\mathsf{H}(\cdot)$ and the key generation oracle $\mathcal{O}_K(\cdot)$. They are defined as follows:

Random oracle $\mathsf{H}(\cdot)$: Same as in proof of theorem 2.

Key generation oracle $\mathcal{O}_K(\cdot)$: When $\mathcal{A}$ issues a query for the secret key of an attribute set $\mathbb{S} = \{x_i\}_{i\in[m]} = \{n_i, v_i\}_{i\in[m]}$. $\mathcal{B}$ runs the random oracle $\mathsf{H}(\cdot)$ to obtain $h_{0i}, h_{1i} \in \mathbb{G}_1$ for each attribute x_i such that $\mathsf{H}(0 \parallel x_i) = h_{0i}$ and $\mathsf{H}(1 \parallel x_i) = h_{1i}$. Let $< 0 \parallel x_i, h_{0i}, t_i >$ and $< 1 \parallel x_i, h_{1i}, \delta_i >$ be the corresponding tuple on list L and Q respectively. Then $\mathcal{B}$ chooses $r \xleftarrow{\$} \mathbb{Z}_p$, and simulates the secret key as follows:

$$\mathsf{sk}_1 = B^r,\ \mathsf{sk}_2 = A^{\alpha - cr},\ \mathsf{sk}_{3,i} = A^{t_i r},\ \mathsf{sk}_{4,i} = A^{\delta_i r}.$$

We can see that the exponents $\frac{1}{b_1}$, $\frac{1}{b_2}$ in $\mathsf{sk}_{3,i}$ and $\mathsf{sk}_{4,i}$ are canceled by the random oracle outputs C^{t_i} and D^{δ_i} respectively, in which $\mathsf{H}(0 \parallel x_i)^{\frac{r}{b_1}} = C^{\frac{t_i r}{b_1}} = A^{t_i r}$, $\mathsf{H}(1 \parallel x_i)^{\frac{r}{b_2}} = D^{\frac{\delta_i r}{b_2}} = A^{\delta_i r}$, and all other elements can be directly simulated by $\mathcal{B}$ by using the known values, so the secret key $\mathsf{sk} = (\mathsf{sk}_1, \mathsf{sk}_2, \{\mathsf{sk}_{3,i}, \mathsf{sk}_{4,i}\}_{i\in[m]})$ is valid for the attribute set $\mathbb{S}$. Then $\mathcal{B}$ gives $(\mathsf{sk}, \{n_i\}_{i\in[m]})$ to $\mathcal{A}$.

Challenge. $\mathcal{A}$ outputs two equal-size messages msg_0^*, msg_1^* and two access policies $\mathbb{A}_0^* = (\mathrm{M}^*, \pi^*, \{\pi^*(i)_0\} = \{n_{\pi^*(i)}, v_{\pi^*(i)_0}\}_{i\in[\ell]})$, $\mathbb{A}_1^* = (\mathrm{M}^*, \pi^*, \{\pi^*(i)_1\} = \{n_{\pi^*(i)}, v_{\pi^*(i)_1}\}_{i\in[\ell]})$ that it intends to attack. Note that $\mathbb{A}_0^*$, $\mathbb{A}_1^*$ must have the same $(\mathrm{M}^*, \pi^*, \{n_{\pi^*(i)}\}_{i\in[\ell]})$. $\mathcal{B}$ checks if $\mathbb{A}_0^*$ or $\mathbb{A}_1^*$ can be satisfied by any attribute set $\mathbb{S}$ queried from the $\mathcal{O}_K$ in Phase 1. If yes, $\mathcal{B}$ rejects $\mathbb{A}_0^*$ and $\mathbb{A}_1^*$. Otherwise, $\mathcal{B}$ randomly picks $\beta \in \{0, 1\}$ and chooses to encrypt a message msg_β^* with access policy $\mathbb{A}_\beta^*$. In specific, $\mathcal{B}$ first runs the random oracle $\mathsf{H}(\cdot)$ to obtain $h_{0i}^*, h_{1i}^* \in \mathbb{G}_1$ for each attribute value $\pi(i)^* \in \mathbb{A}_\beta^*$ such that $\mathsf{H}(0 \parallel \pi(i)^*) = h_{0i}^*$ and $\mathsf{H}(1 \parallel \pi(i)^*) = h_{1i}^*$. Then $\mathcal{B}$ picks a vector $\mathbf{v} \xleftarrow{\$} \mathbb{Z}_p^{n-1}$. Let $< 0 \parallel \pi(i)^*,\ h_{0i}^*,\ t_i^* >$ and $< 1 \parallel \pi(i)^*,\ h_{1i}^*,\ \delta_i^* >$ be the corresponding tuple on list L and Q respectively, $\mathcal{B}$ simulates the challenge ciphertext $\mathsf{ct}_\beta^* = (\{\mathsf{ct}_{1,i}\}_{i\in[m]}, \mathsf{ct}_2, \mathsf{ct}_3, \mathsf{ct}_4, \mathsf{ct}_5)$ as follows:

$$\mathsf{ct}_{1,i} = Y^{c\cdot \mathrm{M}_{i,1}^*} \cdot A^{\sum_{j=2}^{n} \mathrm{M}_{i,j}^* \mathrm{v}_j} \cdot G^{t_i^*} \cdot H^{\delta_i^*}, \mathsf{ct}_2 = Z,\ \mathsf{ct}_3 = I,$$
$$\mathsf{ct}_4 = J,\ \mathsf{ct}_5 = e(Y, B)^\alpha \cdot \mathsf{msg}_\beta^*.$$

Similarly, $\mathsf{H}(0 \parallel \pi(i)^*)^{s_1} = C^{t_i^* s_1} = G^{t_i^*}$, $\mathsf{H}(1 \parallel \pi(i)^*)^{s_2} = D^{\delta_i^* s_2} = H^{\delta_i^*}$. If $\mu = 0$, then $Y = g_1^s$, $Z = g_2^s$, $Y^{c\cdot \mathrm{M}_{i,1}^*} = (g_1^s)^{c\cdot \mathrm{M}_{i,1}^*}$, $e(Y,\ B)^\alpha =$

$e(g_1, g_2)^{\alpha s}$. Thus, the ciphertext ct^*_β is a valid random encryption for policy $\mathbb{A}^*_\beta$ and message msg^*_β. If $\mu = 1$, then $Y = g_1^R$, $Z = g_2^R$, $\mathrm{ct}_{1,i} = g_1^{R\cdot c\cdot \mathrm{M}^*_{i,1}} \cdot g_1^{\sum_{j=2}^n \mathrm{M}^*_{i,j}\mathrm{v}_j} \cdot g_1^{b_1 s_1 t^*_i} \cdot g_1^{b_2 s_2 \delta^*_i}$, $\mathrm{ct}_5 = e(g_1, g_2)^{\alpha R} \cdot \mathrm{msg}^*_\beta$. As R is random, $\mathrm{ct}_{1,i}$ are random elements on $\mathbb{G}_1$, ct_5 is a random element on $\mathbb{G}_T$ hence ct^*_β contains no information about $\mathbb{A}^*_\beta$ and msg^*_β respectively. Then $\mathcal{B}$ gives $(\mathrm{ct}^*_\beta, (\mathrm{M}^*, \pi^*, \{n_{\pi^*(i)}\}_{i\in[\ell]}))$ to $\mathcal{A}$.

Phase 2. Same as Phase 1 with the restriction that any input attribute set $\mathbb{S}$ can not satisfy $\mathbb{A}^*_0$ or $\mathbb{A}^*_1$.

Guess. $\mathcal{A}$ outputs a bit $\beta' \in \{0, 1\}$. If $\beta' = \beta$, $\mathcal{B}$ outputs $\mu' = 0$ indicating (Y, Z) is a valid DLIN tuple. Otherwise it outputs $\mu' = 1$ indicating (Y, Z) is given as a random tuple. In the case of $\mu = 1$, $\mathcal{A}$ gains no information about β, so we have $\Pr[\beta \neq \beta' | \mu = 1] = \frac{1}{2}$. As $\mathcal{B}$ guesses $\mu = 1$ when $\beta \neq \beta'$, we have $\Pr[\mu' = \mu | \mu = 1] = \frac{1}{2}$. If $\mu = 0$, then $\mathcal{A}$ sees an encryption of $\mathbb{A}^*_\beta$ and msg^*_β, with an advantage ϵ' of breaking the scheme. Thus, we have $\Pr[\beta = \beta' | \mu = 0] = \frac{1}{2} + \epsilon'$. As $\mathcal{B}$ guesses $\mu = 0$ when $\beta = \beta'$, we have $\Pr[\mu' = \mu | \mu = 0] = \frac{1}{2} + \epsilon'$. Therefore, the overall advantage of $\mathcal{B}$ solves the DLIN problem is $\frac{1}{2}\Pr[\mu' = \mu | \mu = 0] + \frac{1}{2}\Pr[\mu' = \mu | \mu = 1] - \frac{1}{2} = \frac{1}{2} \cdot (\frac{1}{2} + \epsilon') + \frac{1}{2} \cdot \frac{1}{2} - \frac{1}{2} = \frac{\epsilon'}{2}$. Then we complete the proof of Theorem 4. □

5 IMPLEMENTATIONS

We implement our ABE and A^{2}BE schemes and compare them with several state-of-the-art schemes in their corresponding fields. Our experiments are run in Python 3.9.16 using the Charm 0.5 framework [5] and the BN254 curve for pairings because (1) it is considered secure based on current cryptographic knowledge, (2) it is used in industrial applications, such as SNARKs [7]. All running times below were measured on a PC with a 3.59 GHz AMD Ryzen 5 3600 6-Core Processor and 16GB RAM. The code is available on Github [2].

In the ABE field, we compare our ABE schemes with the state-of-the-art ABE schemes with adaptive security, which include: (1) BSW CP-ABE [11], (2) CGW KP-ABE and CP-ABE [16] with the DLIN instantiations, (3) ABGW KP-ABE and CP-ABE [6], (4) FAME KP-ABE and CP-ABE [3], (5) FABEO KP-ABE and CP-ABE [49], and (6) FEASE KP-ABE [40]. FAME and FABEO has transferred all the above ABE schemes into Type-III setting. In the A^{2}BE field, we compare our KP-A^{2}BE scheme with FEASE [40] and CWD$^+$ [19] and compare our CP-A^{2}BE scheme with CDW$^+$ [18] [6]. We transfer CWD$^+$ and CDW$^+$ from Type-I to the Type-III setting as shown in Appendix B.

For ABE schemes, we form AND gates between attributes in the access policies to simulate the case that attributes are all required for decryption. We test these schemes against policies and attribute sets of size 10, 20, ..., 100 since large policy sizes are quite likely in typical use cases [26]. Besides, we convert an access policy into a Boolean formula and then to an MSP using Lewko-Waters' method [32] (see Sec. 2.1 for a detailed discussion) so that the matrix M has only entries in $\{0, 1, -1\}$ and the reconstruction coefficients $\{\gamma_i\}$ are always 0 or 1, which reduces the number of exponentiations.

For A^{2}BE schemes, we first choose random words from the English vocabulary to form attribute names and randomly assign a positive integer between 1 - 100 as an attribute value to each attribute name, such as "Name: 2", "Gender: 6", "Department: 7". The attribute values are the input of the key generation and encryption algorithms and the attribute names are exposed. We ensure that the attribute names can always match regardless of the policy, but the attribute values are chosen randomly. i.e., the attribute values only have little probability to match. In this way, we can simulate the worst case that the decryption has to traverse every subset of the matched attribute names to maximize the decryption time. We test both the secret key and encryption for 10, 20, ..., 100 attributes, and assign "AND" or "OR" gates between the attributes to form various policies as we need.

[6]The background of these works are illustrated in Sec. 6.

Groups	Choose	Multiply	Exp.	Hash	Pairing
$\mathbb{G}_1$	1.01	0.0017	0.74	0.03	
$\mathbb{G}_2$	1.46	0.0038	1.32	0.07	20.7
$\mathbb{G}_T$	-	0.0309	5.44	-	

Table 1: Average time (in ms) for operations on BN254 curve.

Schemes	Time (s)
CGW	0.485
ABGW	0.028
FAME	0.035
FABEO	0.026
FEASE	0.028
Ours	0.029

Schemes	Time (s)
BSW	0.026
CGW	0.487
ABGW	0.03
FAME	0.037
FABEO	0.027
Ours	0.031

Table 2: Setup time (in sec) for KP-ABE schemes (left) and CP-ABE schemes (right). The setup time of CGW is measured when the attribute universe is set to 100. Other schemes support large universe and have constant setup time.

Schemes	Time (s)
CWD$^+$	0.036
FEASE	0.028
Ours	0.029

Schemes	Time (s)
CDW$^+$	0.04
Ours	0.031

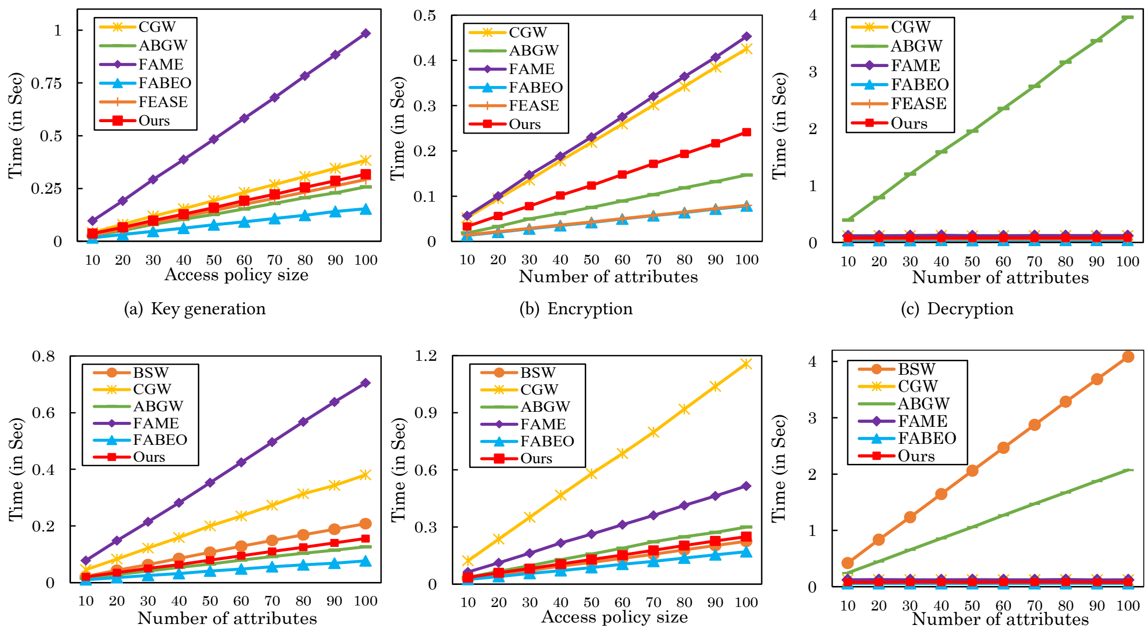
Table 3: Setup time (in ms) for KP-A^{2}BE schemes (left) and CP-A^{2}BE schemes (right).

Table 1 lists the average time taken by various operations on BN254. The setup times for the ABE and A^{2}BE schemes are listed in Table 2 and Table 3 respectively. Then we show the running times for the ABE schemes in Fig. 6 and the running times for A^{2}BE schemes in Fig. 7. These results are supported by the theoretical comparison in Table 4 and Table 6 which lists the number of multiplications, exponentiations, hashings and pairings. Besides, we provide the number of group elements of key and ciphertext in Table 5 and 7.

5.1 Basic operations

According to Table 1, we can see that operations on group $\mathbb{G}_2$ are more expensive than on $\mathbb{G}_1$, in which it has nearly 1.5 times slower for choosing an element, 2 times slower for exponentiation, 15 times slower for multiplication, and 2.3 times slower for hashing. Pairing is the most expensive operation that is nearly 28 and 16 times slower than exponentiation on $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively. In addition, note that the size of an element in $\mathbb{G}_2$ is 1.8 times that of $\mathbb{G}_1$.

In Table 2 and Table 3, we show the setup time of the schemes listed in our evaluation. Except for CGW schemes, all the other schemes support large universes of attributes, and hence have a constant setup time and are almost equally fast. Specifically, the setup time of our schemes is a bit faster than FAME, CWD$^+$, and CDW$^+$.



(a) Key generation
(b) Encryption
(c) Decryption
(d) Key generation
(e) Encryption
(f) Decryption

Figure 6: Running times for KP-ABE (top) and CP-ABE (bottom) schemes.

Schemes	Key generation					Encryption					Decryption			
	$\mathbb{G}_1$			$\mathbb{G}_2$		$\mathbb{G}_1$			$\mathbb{G}_2$		$\mathbb{G}_1$	$\mathbb{G}_2$	$\mathbb{G}_T$	Pairing
	Mul	Exp	Hash	Mul	Exp	Mul	Exp	Hash	Mul	Exp	Mul	Mul	Mul	
CGW	-	-	-	$\sim 6\ell n$	$6\ell+9n$	-	$3m+3$	-	-	-	$3x_2$	$3x_2$	6	6
ABGW	-	-	-	-	2ℓ	$2m$	$3m+1$	-	-	-	-	-	$2x_2$	$2x_2$
FAME	$9\ell n+3n$	$9\ell+3n$	$6\ell+6n$	-	3	$3m$	$6m$	$6m$	-	3	$6x_2$	-	6	6
FABEO	ℓ	2ℓ	ℓ	-	1	-	m	m	-	1	$2x_2$	-	2	2
FEASE	2ℓ	4ℓ	ℓ	-	1	-	m	m	-	2	$3x_2$	-	2	3
Ours	ℓ	4ℓ	3ℓ	-	1	$2m$	$3m$	$3m$	-	3	$4x_2$	-	4	4
BSW	$m+1$	$m+2$	m	-	m	-	ℓ	ℓ	-	$\ell+1$	-	-	$2x_2+1$	$2x_2+1$
CGW	-	-	-	-	$3m+6$	$\sim 6\ell n$	$6\ell+9n$	-	-	-	$3x_2$	$3x_2$	6	6
ABGW	-	-	-	-	$2m+1$	2ℓ	5ℓ	-	-	-	$2x_2$	-	x_2+2	x_2+2
FAME	$6m+9$	$9m+9$	$6m+6$	-	3	$6\ell n+3\ell$	6ℓ	$6\ell+6n$	-	3	$6x_2+3$	-	6	6
FABEO	1	$m+2$	$m+1$	-	1	ℓ	2ℓ	$\ell+1$	-	2	$2x_2$	-	3	3
Ours	1	$2m+2$	$2m$	-	1	2ℓ	3ℓ	2ℓ	-	3	$3x_2$	-	4	4

Table 4: Computational Overhead for key generation, encryption and decryption between KP-ABE (top) and CP-ABE schemes (bottom). m denotes the number of attributes in the attribute set, ℓ and n are the number of rows and columns of the MSP matrix. x_2 denotes the total number of attributes used for decryption. The number for CGW multiplications in $\mathbb{G}_1$ and $\mathbb{G}_2$ are $3(2\ell+3n+2\ell n-1)$.

5.2 ABE schemes

KP-ABE schemes. The running times and computational overhead for KP-ABE schemes are shown in the upper part of Fig. 6 and Table 4 respectively. For key generation, when the policy contains 100 attributes, our scheme (0.32s) outperforms CGW (0.38s) and FAME (0.98s), and is close to ABGW (0.26s), FEASE (0.3s) and FABEO (0.15s). This is because our scheme has 4ℓ exponentiations and 3ℓ hashes on $\mathbb{G}_1$, and only 1 exponentiation on $\mathbb{G}_2$. CGW and ABGW have more exponentiations on $\mathbb{G}_2$ (CGW has $6\ell+9n$ and ABGW has 2ℓ), FAME has more than 2 times exponentiations in $\mathbb{G}_1$ and more than 3 times multiplication and hash calculations in $\mathbb{G}_1$ than our scheme. But ABGW has no calculations on $\mathbb{G}_1$, FEASE and FABEO has less exponentiations and hashes on $\mathbb{G}_1$ than ours. For encryption, when the attribute set size is 100, our scheme runs 0.24s, which is faster than CGW (0.43s) and FAME (0.45s), and a bit slower than ABGW (0.15s), FEASE (0.09s) and FABEO (0.08s). The reason is that our scheme has $3m$ exponentiations on $\mathbb{G}_1$, while FAME has nearly three times more ($6m$) and CGW has nearly 1.5 times more ($3m+3$). In contrast, our scheme has more exponentiations on $\mathbb{G}_2$ than ABGW, and has more $\mathbb{G}_1$ exponentiations and hashes than FABEO and FEASE.

For decryption, except ABGW has a linear increase with the number of attributes, all other schemes have nearly constant decryption time. Among them, our scheme (0.08s) is faster than CGW (0.12s) and FAME (0.12s) and is very close to FABEO (0.04s) and FEASE (0.06s). The reason is that our scheme has a constant 4 pairings and $4x_2$ multiplications on $\mathbb{G}_1$, which is less than CGW (6 pairings and $3x_2$

Schemes	Key size		Ciphertext size	
	$\mathbb{G}_1$	$\mathbb{G}_2$	$\mathbb{G}_1$	$\mathbb{G}_2$
CGW	-	$3\ell+3$	$3m+3$	-
ABGW	-	2ℓ	$2m$	-
FAME	3ℓ	3	$3m$	3
FABEO	ℓ	1	m	1
FEASE	2ℓ	1	m	2
Ours	3ℓ	1	m	3
BSW	$m+1$	m	ℓ	$\ell+1$
CGW	-	$3m+6$	$3\ell+3$	-
ABGW	-	$m+2$	3ℓ	-
FAME	$3m+3$	3	3ℓ	3
FABEO	$m+1$	1	ℓ	2
Ours	$2m+1$	1	ℓ	3

Table 5: Comparison of Communication Overhead between KP-ABE schemes (top) and CP-ABE schemes (bottom). m **denotes the number of attributes in the attribute set,** ℓ **and** n **are the number of rows and columns of the MSP matrix.**

multiplications on $\mathbb{G}_2$), and FAME (6 pairings and $6x_2$ multiplications on $\mathbb{G}_1$), and is more than FABEO (2 pairings and $2x_2$ multiplications on $\mathbb{G}_1$) and FEASE (3 pairings and $3x_2$ multiplications on $\mathbb{G}_1$).

The communication overhead of KP-ABE schemes is shown in the upper part of Table 5. For key size, our scheme has group elements 3ℓ on $\mathbb{G}_1$ and 1 on $\mathbb{G}_2$, which is less than CGW ($3\ell+3$ on $\mathbb{G}_2$), ABGW (2ℓ on $\mathbb{G}_2$), and FAME (3ℓ on on $\mathbb{G}_1$ and 3 on $\mathbb{G}_2$) as the group elements on $\mathbb{G}_2$ is 1.8 times longer than the ones on $\mathbb{G}_1$. FABEO and FEASE only has ℓ and 2ℓ elements on $\mathbb{G}_1$ respectively so they have shorter sizes than our scheme. For ciphertext size, our scheme has m elements on $\mathbb{G}_1$ and 3 on $\mathbb{G}_2$, which is less than CGW ($3m+3$ on $\mathbb{G}_1$), ABGW ($2m$ on $\mathbb{G}_1$), and FAME ($3m$ on $\mathbb{G}_1$ and 3 on $\mathbb{G}_2$). FABEO and FEASE has m elements on $\mathbb{G}_1$ that is the same as ours but less elements on $\mathbb{G}_2$ so they are shorter than our scheme.

CP-ABE schemes. The running times and computational overhead for CP-ABE schemes are presented in the lower part of Fig. 6 and Table 4 respectively. For key generation, when the attribute set size is 100, our scheme (0.15s) is close to ABGW (0.13s) and FABEO (0.08s), and is faster than BSW (0.2s), CGW (0.38s), and FAME (0.7s). The reason is, our scheme has exponentiations $2m+2$ on $\mathbb{G}_1$ and 1 on $\mathbb{G}_2$, which is less than FAME ($9m+9$ on $\mathbb{G}_1$ and 3 on $\mathbb{G}_2$), CGW ($3m+6$ on $\mathbb{G}_2$), and BSW ($m+1$ on $\mathbb{G}_1$ and m on $\mathbb{G}_2$). ABGW is a bit faster than our scheme since it has $2m+1$ exponentiations on $\mathbb{G}_2$ and no calculations on $\mathbb{G}_1$. FABEO is faster than our scheme since it has nearly half of the exponentiations ($m+2$) and hashes ($m+1$) on $\mathbb{G}_1$ than our scheme. For encryption, when the policy size is 100, our scheme (0.24s) runs faster than CGW (1.15s), ABGW (0.3s) and FAME (0.52s), and is close to BSW (0.22s) and FABEO (0.17s). The reason is that our scheme has fewer exponentiations on $\mathbb{G}_1$ (3ℓ) than CGW ($6\ell+9n$), ABGW (5ℓ), and FAME (6ℓ). Despite BSW has a linear increase ($\ell+1$) for exponentiations on $\mathbb{G}_2$ while our scheme has a constant of 3, it has fewer $\mathbb{G}_1$ exponentiations (ℓ) than us. Our scheme is slower than FABEO since it has 1.5 times exponentiations more than FABEO on both groups $\mathbb{G}_1$ and $\mathbb{G}_2$.

For decryption, except BSW and ABGW have a linear increase with the number of attributes, the other schemes have nearly constant decryption time. Specifically, our scheme has constant 4 pairings and $3x_2$ multiplications on $\mathbb{G}_1$, which is less than CGW (6 pairings and $3x_2$ multiplications on both $\mathbb{G}_1$ and $\mathbb{G}_2$), FAME (6 pairings and $6x_2+3$ multiplications on $\mathbb{G}_1$), and is very close to FABEO (3 pairings and $2x_2$ multiplications on $\mathbb{G}_1$).

The communication overhead of CP-ABE schemes is in the lower part of Table 5. For key size, our scheme has $2m+1$ elements on $\mathbb{G}_1$ and 1 on $\mathbb{G}_2$, which is less than BSW ($m+1$ on $\mathbb{G}_1$ and m on $\mathbb{G}_2$), CGW ($3m+6$ on $\mathbb{G}_2$), and FAME ($3m+3$ on on $\mathbb{G}_1$ and 3 on $\mathbb{G}_2$) as the group elements on $\mathbb{G}_2$ is 1.8 times bigger than the ones on $\mathbb{G}_1$. ABGW only has $m+2$ on $\mathbb{G}_2$ and no elements on $\mathbb{G}_1$. FABEO only has $m+1$ elements on $\mathbb{G}_1$ and 1 on $\mathbb{G}_2$ so these two schemes has shorter sizes than our scheme. For ciphertext size, our scheme has $\ell+3$ elements on $\mathbb{G}_1$ and 3 on $\mathbb{G}_2$, which is less than BSW (ℓ on $\mathbb{G}_1$ and $\ell+1$ on $\mathbb{G}_2$), CGW ($3\ell+3$ on $\mathbb{G}_1$), ABGW (3ℓ on $\mathbb{G}_1$), and FAME (3ℓ on $\mathbb{G}_1$ and 3 on $\mathbb{G}_2$). FABEO has ℓ elements on $\mathbb{G}_1$ that is the same as ours but only 1 element on $\mathbb{G}_2$ so it is shorter than our scheme.

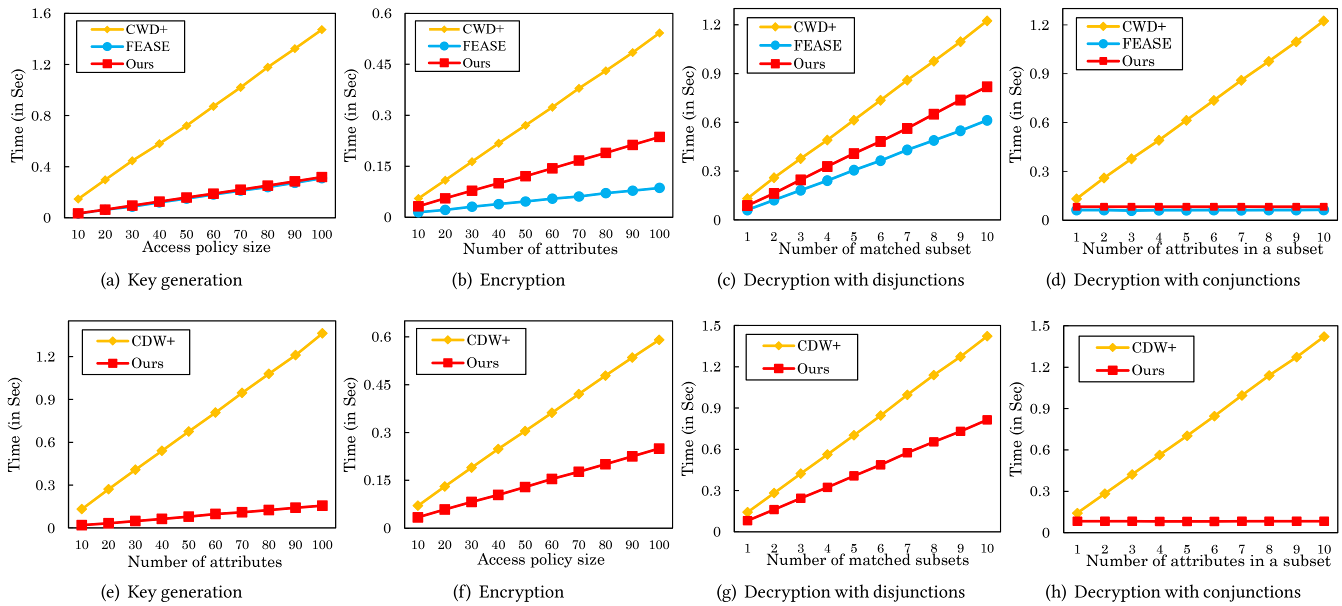
5.3 Anonymous ABE schemes

KP-A^{2}BE schemes. The running times and computational overhead for KP-A^{2}BE schemes are presented in the upper part of Fig. 7 and Table 6 respectively. For key generation, our scheme runs 0.32s for generating a policy that contains 100 attributes, which is basically the same as FEASE (0.31s) and is 6.2 times faster than CWD$^+$ (1.47s). The reason is that our scheme has 4ℓ exponentiations and 3ℓ hashes on $\mathbb{G}_1$, but all the multiplications and exponentiations of CWD$^+$ are calculated on $\mathbb{G}_2$ (2ℓ multiplications and 8ℓ exponentiations). FEASE is a bit faster since it has less hashes (ℓ) than ours. For encryption, our scheme runs 0.24s for encrypting 100 attributes, which is faster than CWD$^+$ (0.54s) and slower than FEASE (0.09s). The reason is that our scheme has $3m$ exponentiations on $\mathbb{G}_1$ while CWD$^+$ has $6m+2$. Our scheme is slower than FEASE as it triples the exponentiations and hashes on $\mathbb{G}_1$ and has one more exponentiation on $\mathbb{G}_2$.

For decryption, Fig. 7 (c) shows that our scheme runs 0.82s for 10 disjunctions in a matched subset, which is faster than CWD$^+$ (1.22s) and a bit slower than FEASE (0.61s). Fig. 7 (d) shows that for decrypting a conjunctive set of 10 attributes, our scheme runs for 0.08s, which is 15.2 times faster than CWD$^+$ (1.22s) and close to FEASE (0.06s). The reason is that the pairing number of FEASE and our scheme only relates to x_1 - the number of disjunctions (OR gates) in the matched subset, while CWD$^+$ is related to x_2 - the total number of attributes needed for decryption. The reason that FEASE is a bit faster than our scheme is that it has less pairing ($3x_1$) than ours ($4x_1$).

The communication overhead of KP-A^{2}BE schemes is shown in the upper part of Table 7. For key size, our scheme has 3ℓ elements on $\mathbb{G}_1$ and 1 element on $\mathbb{G}_2$, which is less than CWD$^+$ (6ℓ elements on $\mathbb{G}_2$). For ciphertext size, our scheme has m elements on $\mathbb{G}_1$, which is much less than CWD$^+$ ($5m+1$ on $\mathbb{G}_1$).

CP-A^{2}BE schemes. The running times and computational overhead for CP-A^{2}BE schemes are presented in the lower part of Fig. 7 and Table 6 respectively. For key generation, for encrypting 100 attributes, our scheme (0.16s) outperforms CDW$^+$ (1.36s) 8.5 times faster. The reason is that our scheme needs $2m+2$ exponentiations on $\mathbb{G}_1$ while CDW$^+$ has $3m+1$ exponentiations and $8m+5$ multiplications on the expensive group $\mathbb{G}_2$. For encryption, our scheme runs 0.25s which is 2.4 times faster than CDW$^+$ (0.59s) for generating a policy with 100 attributes because our scheme has fewer exponentiations (3ℓ) than CDW$^+$ ($8\ell+1$) on $\mathbb{G}_1$. For decryption, Fig. 7 (g) shows that our scheme runs 0.81s for 10 disjunctions in a matched subset, which is faster



(a) Key generation (b) Encryption (c) Decryption with disjunctions (d) Decryption with conjunctions

(e) Key generation (f) Encryption (g) Decryption with disjunctions (h) Decryption with conjunctions

Figure 7: Running times for KP-A^{2}BE schemes (top) and CP-A^{2}BE schemes (bottom). Figures (c) and (g) measure the running time of the decryption algorithm regarding the number of "OR" gates in a matched attribute name subset (assume no "AND" gates). Figures (d) and (h) measure the running time of the decryption algorithm regarding the number of "AND" gates in a matched attribute name subset (assume no "OR" gates).

Schemes	Key generation					Encryption					Decryption		
	$\mathbb{G}_1$			$\mathbb{G}_2$		$\mathbb{G}_1$			$\mathbb{G}_2$		$\mathbb{G}_1$	$\mathbb{G}_T$	Pairing
	Mul	Exp	Hash	Mul	Exp	Mul	Exp	Hash	Mul	Exp	Mul	Mul	
CWD$^+$	-	-	-	2ℓ	8ℓ	$2m$	$6m+2$	-	-	-	-	$5x_2+1$	$6x_2$
FEASE	2ℓ	4ℓ	ℓ	-	1	-	m	m	-	2	$3x_2$	2	$3x_1$
Ours	ℓ	4ℓ	3ℓ	-	1	$2m$	$3m$	$3m$	-	3	$4x_2$	4	$4x_1$
CDW$^+$	-	-	-	$3m+1$	$8m+5$	2ℓ	$8\ell+1$	-	-	-	-	$5x_2+2$	$6x_2+1$
Ours	1	$2m+2$	$2m$	-	1	2ℓ	3ℓ	2ℓ	-	3	$3x_2$	4	$4x_1$

Table 6: Comparison of Computational Overhead for key generation, encryption and decryption between KP-A^{2}BE schemes (top) and CP-A^{2}BE schemes (bottom). m denotes the number of attributes in the attribute set, ℓ and n are the number of rows and columns of the MSP matrix. x_1 denotes the total number of matched attribute names subset. x_2 denotes the total number of attributes in all matched names subset.

Schemes	Key size		Ciphertext size	
	$\mathbb{G}_1$	$\mathbb{G}_2$	$\mathbb{G}_1$	$\mathbb{G}_2$
CWD$^+$	-	6ℓ	$5m+1$	-
FEASE	2ℓ	1	m	2
Ours	3ℓ	1	m	3
CDW$^+$	-	$5m+2$	$6\ell+1$	-
Ours	$2m+1$	1	ℓ	3

Table 7: Comparison of Communication Overhead between KP-A^{2}BE schemes (top) and CP-A^{2}BE schemes (bottom). m denotes the number of attributes in the attribute set, ℓ and n are the number of rows and columns of the MSP matrix.

than CDW$^+$ (1.42s). Fig. 7 (h) shows that for decrypting a conjunctive set of 100 attributes, our scheme (0.08s) runs 17.7 times faster than CDW$^+$ (1.42s). Similar to the KP-A^{2}BE, the reason is that the pairing number of our scheme only relates to x_1 while CDW$^+$ is related to x_2.

The communication overhead of CP-A^{2}BE schemes is shown in the lower part of Table 7. For key size, our scheme has $2m+1$ elements on $\mathbb{G}_1$ and 1 element on $\mathbb{G}_2$, which is less than CDW$^+$ ($5m+2$ elements on $\mathbb{G}_2$). For ciphertext size, our scheme has only elements ℓ on $\mathbb{G}_1$ and 3 on $\mathbb{G}_2$, which is less than CDW$^+$ ($6\ell+1$ on $\mathbb{G}_1$).

Discussions. It is not hard to see that the performance of both our ABE and A^{2}BE schemes are comparable to the fastest ones: FABEO [49] and FEASE [40]. Considering that our schemes realize the adaptive security under the DLIN assumption while FABEO and FEASE are proven under the GGM, the degradation of efficiency is a trade-off for stronger security arguments. Besides, normally A^{2}BE schemes have lower efficiency than traditional ABE schemes. When transforming an ABE scheme to an A^{2}BE scheme, the degradation of efficiency comes from (1) protecting the payload privacy, and (2) realizing the ciphertext anonymity. Since our ABE schemes naturally reaches ciphertext anonymity, our transformation from ABE to

A^{2}BE schemes only incurs an efficiency degradation in the decryption algorithm brought by the partially hidden structure for payload privacy. Since the match of attribute names does not guarantee the match of attribute values, the testing time of attribute values still requires a linear increase in the number of OR gates in the matched subset. Nevertheless, the partially hidden structure has been the most efficient technique to protect payload privacy.

6 RELATED WORKS

6.1 State-of-the-art ABE schemes

In the ABE field, a diverse range of trade-offs exists between expressiveness, efficiency, and security. The state-of-the-art expressive ABE schemes include the following pairing-based schemes: 1) BSW CP-ABE [11], 2) GPSW KP-ABE [25], 3) Waters CP-ABE [55], 4) CGW KP-ABE and CP-ABE [16], 5) ABGW CP-ABE and KP-ABE [6], 6) FAME CP-ABE and KP-ABE [3], and 7) FABEO CP-ABE and KP-ABE [49]. Table 8 provides a property-wise comparison between the state-of-the-art ABE schemes with various features. Among these schemes, only BSW and FABEO simultaneously support the large universe, arbitrary attribute strings, and attribute multi-use. As illustrated in Sec. 3, our ABE schemes build on FABEO and thus inherit all these features.

The simulation methods (or techniques) used to prove the security of ABE schemes include: (1) **Random Oracle Model (ROM)**: This model assumes that some hash functions used in the schemes behave like random oracles. (2) **Standard Model (STM)**: This model does not assume the idealized random oracle, making it preferable to ROM. In related works, the security of BSW, Waters, FAME, FABEO are proved by using the ROM, while the security of GPSW, Waters [7], CGW, and ABGW are proved by using the STM.

The security of an ABE scheme can be reduced to (1) **Generic Group Model (GGM)**: Generic properties of mathematical groups rather than specific hardness assumptions, or (2) **Standard Assumption**: Well-studied mathematical hard problem, which are preferred because they offer greater assurance of security than GGM. In related works, BSW, ABGW, and FABEO prove their security under the GGM. In contrast, CGW and FAME base their security on the k-linear (k-LIN) assumptions [8] by adopting the dual system encryption framework [33, 54]. However, this framework results in more complex constructions and security proofs [49]. Instead, several ABE schemes, e.g., GPSW [25], and Waters [55], achieve selective security under standard assumptions by using simulation-based proofs.

In this work, our schemes achieve adaptive security while relying on a standard assumption: DLIN assumption, and we use the ROM as our technique. The form of our proof relies on straightforward simulations rather than the dual system framework. Although the use of ROM is not ideal, it enables us to meet both of our goals: relying on a *standard assumption* and achieving *high-level efficiency*. Finding a way to replace ROM with the STM while still meeting these two objectives remains an open problem.

In terms of efficiency, only CGM, FAME and FABEO achieve fast decryption that only needs a constant number of pairings regardless of the size of the attributes. Specifically, CGW with DLIN instantiation (when $k=2$) and FAME need 6 pairings for both KP-ABE and CP-ABE schemes. FABEO needs 2 pairings for KP-ABE and 3 for the CP-ABE scheme. Our ABE schemes, designed based on FABEO KP-ABE, inherit the feature of fast decryption, necessitating a constant 4 pairings while providing security under the DLIN assumption.

6.2 Anonymous ABE schemes

Fully A^{2}BE schemes. Katz, Sahai, and Waters (KSW) [28] proposed Inner Product Encryption (IPE), capable of realizing expressive KP-A^{2}BE and CP-A^{2}BE schemes. The IPE encodes policy and attribute set into equal-size vectors $\mathbf{x}$ and $\mathbf{y}$ and embeds them into the key and ciphertext separately. The decryption successes if $\mathbf{x}\cdot\mathbf{y}=0$. The KSW scheme is based on composite order groups and is selectively secure. Subsequently, Okamoto and Takashima [43, 44] constructed IPE schemes based on prime-order groups and achieved adaptive security based on the DLIN assumption. Chen et al. [17] further reduced the public key size of [43] while maintaining the security arguments. Xiong et al. [57] proposed an efficient IPE scheme that achieves constant size of public key and secret key.

IPE restrictions. Although IPE schemes offer stronger security arguments, ensuring full payload privacy while satisfying adaptive security under standard assumptions, they remain impractical due to the following limitations. Firstly, the system requires fixed parameters at the setup phase, including the maximum number of attributes, maximum degree d, and the vector size N. This inflexibility necessitates parameter updates every time the system changes. Secondly, the computational overhead of IPE schemes is directly related to N, but the encoding process from policy and attributes into vectors significantly penalizes efficiency. The existing encoding technique [28] involves representing a policy as a multivariable polynomial [9], resulting in a superpolynomial increase in the size of the attribute set vector (assume it is $\mathbf{y}$) for applications with a large number of attributes. For instance, with 100 attributes and a maximum degree of 3, the length of $\mathbf{y}$ is approximately 10^6 [44]. As the length of $\mathbf{x}$ and $\mathbf{y}$ must be equal, even $\mathbf{x}$ only contains only a few attributes in the policy, it needs to be encoded to the same length of $\mathbf{y}$ around 10^6. Despite efforts to mitigate this limitation by Okamoto and Takashima [44], which reduced the length of $\mathbf{x}$ proportionally to the effective dimension, $\mathbf{y}$ still imposes significant computational and communication overhead.

Partially A^{2}BE schemes. The concept of partially A^{2}BE schemes are firstly proposed by Nishide et al. [42], but their scheme and a number of other works [13, 15, 30, 35, 36, 46–48, 60, 61] [8, 21, 24, 39, 59] only support AND gate policies. Besides, KP-A^{2}BE has been paid much less attention than CP-A^{2}BE schemes. It has been studied that expressive KP-A^{2}BE shares similar syntax and security properties from the expressive Asymmetric Searchable Encryption (ASE) schemes, one can generically transform an expressive ASE scheme to a KP-A^{2}BE by treating the keywords as the attributes [40]. Therefore, we look into the expressive CP-A^{2}BE and KP-A^{2}BE (ASE) schemes.

Lai et al. introduced the first LSSS-based expressive CP-A^{2}BE [29] and KP-A^{2}BE [31]. They introduced the "partially hidden structure"

[7] The Waters CP-ABE scheme [55] have different versions with or without using the ROM.
[8] The dimension of the CGW construction depends on k thus can be proven under the general k-LIN assumptions. FAME is proven under the specific DLIN assumption (instantiated with $k=2$).

[9] E.g., a policy $\mathbb{A}$ = (a_1 **OR** b_1) **AND** c_1 is set to $P(x_1,\ x_2,\ x_3) = r_1(x_1-a_1)(x_2-b_1)+r_2(x_3-c_1)$ where $r_1, r_2 \in \mathbb{Z}_p$. Assume d is set to 2, then both vectors are set as the coefficients of $(1, x_1, x_2, x_3, x_1^2, x_2^2, x_3^2, x_1x_2, x_1x_3, x_2x_3)$. The policy vector $\mathbf{x} = (a_1b_1r_1 - c_1r_2, -b_1r_1, -a_1r_1, r_2, 0, 0, 0, r_1, 0, 0)$. An attribute set $\mathbb{S} = [a_1, b_4, c_1]$ is encoded as $\mathbf{y} = (1, a_1, b_4, c_1, 0, 0, 0, a_1b_4, 0, 0)$. We can see that $\mathbf{x}\cdot\mathbf{y}=0$.

Scheme	Large Universe	Arbitrary attributes	Attribute multi-use	Adaptive security	Assumption	Simulation method	Fast decryption
GPSW [25]	✓	×	✓	×	DBDH	STM	×
BSW [11]	✓	✓	✓	✓	GGM	ROM	×
Waters [55]	✓	×	✓	×	q-BDHE	STM / ROM	×
CGW [16]	×	×	×	✓	k-LIN	STM	✓
ABGW [6]	✓	✓	✓	✓	GGM	STM	×
FAME [3]	✓	✓	×	✓	DLIN	ROM	✓
FABEO [49]	✓	✓	✓	✓	GGM	ROM	✓
Ours (Fig. 4, 5)	✓	✓	✓	✓	DLIN	ROM	✓

Table 8: A property-wise comparison between the state-of-the-art ABE schemes. Here, we refer BSW, Waters and GPSW to their Type-III versions specified in [3]. CGW, ABGW, FAME, FABEO, and our schemes include both their KP-ABE and CP-ABE schemes which share the same features. Besides, "DBDH" stands for "Decisional Bilinear Diffie Hellman", "q-BDHE" stands for "Decisional q-parallel Bilinear Diffie Hellman Exponents", "ROM" stands for "Random Oracle Model", "STM" stands for "Standard Model".

Scheme	Prime order	Large universe	Arbitrary attributes	Payload privacy	Security	Assumption	Simulation method	KeyGen	Enc
KSW [28]	×	×	×	Full	Selective	COA	STM	$O(N)$	$O(N)$
OT^1 [43]	✓	×	×	Full	Adaptive	DLIN	STM	$O(N^2)$	$O(N^2)$
OT^2 [44]	✓	×	×	Full	Adaptive	DLIN	STM	$O(N)$	$O(N)$
CGW [17]	✓	×	×	Full	Adaptive	k-LIN	STM	$O(N \cdot k)$	$O(N \cdot k)$
XYY$^+$ [57]	✓	×	✓	Full	Adaptive	DLIN	ROM	$O(N)$	$O(N)$
LDL [29]	×	×	×	Partial	Adaptive	COA	STM	$O(n)$	$O(\ell)$
LZD$^+$ [31]	×	×	×	Partial	Adaptive	COA	STM	$O(\ell)$	$O(n)$
CWD$^+$ [19]	✓	✓	×	Partial	Selective	$q-1$, DLIN	STM	$O(\ell)$	$O(m)$
CDW$^+$ [18]	✓	✓	×	Partial	Selective	$q-2$, DLIN	STM	$O(m)$	$O(\ell)$
ZZD [62]	×	✓	×	Partial	Adaptive	COA	STM	$O(m)$	$O(\ell)$
FEASE [40]	✓	✓	✓	Partial	Adaptive	GGM	ROM	$O(\ell)$	$O(m)$
Our KP-A^{2}BE (Sec. 3.3)	✓	✓	✓	Partial	Adaptive	DLIN	ROM	$O(\ell)$	$O(m)$
Our CP-A^{2}BE (Sec. 3.3)	✓	✓	✓	Partial	Adaptive	DLIN	ROM	$O(m)$	$O(\ell)$

Table 9: A property-wise comparison of state-of-the-art expressive A^{2}BE schemes in the field, including IPE schemes (top) and partially A^{2}BE schemes (bottom). "COA" stands for "composite order group assumptions". "ROM" stands for "Random Oracle Model", "STM" stands for "Standard Model". "KeyGen" and "Enc" separately represent the computational complexity of the key generation and encryption algorithms. N is the size of IPE vectors, n is the maximum number of attributes in the system, k is the parameter used in the k-LIN family, m and ℓ are the number of attributes in the attribute set and policy respectively.

to enhance efficiency by sacrificing the privacy of attribute names. Their scheme is based on Lewko's ABE scheme [34], which is limited by its small universe, composite-order group construction, and the restriction that each attribute can only be used once in an access policy. Lv et al. [38] extended the construction of [31] to support more expressive "negation (NOT gate)" keyword policies. Liu et al. [37] and Zhang et al. [62] improved [29] by removing the restriction of one-use and enabling large universe construction respectively. Hu et al. [27] identified vulnerabilities in existing CP-A^{2}BE schemes and proposed a resilient scheme similar to [62]. Subsequently, the works [23, 53, 58] separately applied the scheme in [62] into the scenarios of vehicular fog computing, Internet-of-Medial Things, and smart health.

Cui et al. proposed the first expressive CP-A^{2}BE [18] and KP-A^{2}BE [19] on prime-order groups, incorporating Rouselakis and Waters's ABE scheme [50] with partially hidden structure for payload privacy and the linear splitting technique [14] for ciphertext anonymity. This scheme exhibits improved efficiency compared to those based on composite-order groups, achieving a large universe and selective security in the standard model. Meng et al. [41] developed [19] to reach constant size ciphertext while trading off the trapdoor efficiency into superpolynomial complexity.

Recently, Meng et al. proposed a fast and expressive KP-A^{2}BE in FEASE [40] that adjusts the construction of FABEO for achieving ciphertext anonymity and adopting the partially hidden structure for payload privacy. Their scheme achieved the best efficiency but their security can be only proven under the GGM. Table 9 compare various features between the state-of-the-art A^{2}BE schemes.

7 CONCLUSION

This paper proposed fast ABE schemes with adaptive security under the DLIN assumption. The security proof of our schemes uses simple simulation-based reduction instead of the complex dual system framework. Besides, the construction satisfies ciphertext anonymity and hence they are easily transferred to anonymous ABE schemes by adding a partially hidden structure. The efficiency of both our ABE and anonymous ABE schemes reach the top level in their corresponding fields and is comparable to the fastest ones. Our future works aims to design a fast ABE scheme under a standard assumption without using the ROM.

ACKNOWLEDGEMENT

This work is supported by the EU's research and innovation program: 952697 (ASSURED), 101019645 (SECANT), 101069688 (CONNECT), 101070627 (REWIRE), and 101095634 (ENTRUST). These projects are funded by the UK government Horizon Europe guarantee and administered by UKRI. We would like to thank Junichi Tomida for identifying a security flaw in the initial design of our KP-ABE scheme, which we have rectified in this publication.

REFERENCES

[1] 2010. CipherCloud. https://cpl.thalesgroup.com/partners/ciphercloud.
[2] 2024. FABESA: Fast Attribute-Based Encryption with Adaptive Security under Standard Assumption. https://github.com/ACMCCS2024/FABESA.git.
[3] Shashank Agrawal and Melissa Chase. 2017. FAME: Fast Aattribute-Based Message Encryption. In CCS.
[4] Joseph A Akinyele et al. 2011. Securing Electronic Medical Records Using Attribute-Based Encryption on Mobile Devices. In SPSM.
[5] Joseph A Akinyele et al. 2013. Charm: a Framework for Rapidly Prototyping Cryptosystems. Journal of Cryptographic Engineering (2013).
[6] Miguel Ambrona et al. 2017. Attribute-Based Encryption in the Generic Group Model: Automated Proofs and New Constructions. In CCS.
[7] Diego F Aranha, Youssef El Housni, and Aurore Guillevic. 2023. A Survey of Elliptic Curves for Proof Systems. Designs, Codes and Cryptography (2023).
[8] Yang Ba et al. 2021. A Blockchain-Based CP-ABE Scheme with Partially Hidden Access Structures. Security Comm. Networks (2021).
[9] Randy Baden, Adam Bender, Neil Spring, Bobby Bhattacharjee, and Daniel Starin. 2009. Persona: an Online Social Network with User-Defined Privacy. In SIGCOMM.
[10] Amos Beimel. 1996. Secure Schemes for Secret Sharing and Key Distribution. In PhD thesis.
[11] John Bethencourt, Amit Sahai, and Brent Waters. 2007. Ciphertext-Policy Attribute-Based Encryption. In IEEE S&P.
[12] Dan Boneh and Jonathan Katz. 2005. Improved Efficiency for CCA-Secure Cryptosystems Built using Identity-Based Encryption. In CT-RSA.
[13] Dan Boneh and Brent Waters. 2007. Conjunctive, Subset, and Range Queries on Encrypted Data. In TCC.
[14] Xavier Boyen and Brent Waters. 2006. Anonymous Hierarchical Identity-Based Encryption (Without Random Oracles). In CRYPTO.
[15] Payal Chaudhari, Manik Lal Das, and Anish Mathuria. 2015. On Anonymous Attribute Based Encryption. In ICISS.
[16] Jie Chen, Romain Gay, and Hoeteck Wee. 2015. Improved Dual System ABE in Prime-Order Groups via Predicate Encodings. In EUROCRYPT.
[17] Jie Chen, Junqing Gong, and Hoeteck Wee. 2018. Improved Inner-Product Encryption with Adaptive Security and Full Attribute-Hiding. In ASIACRYPT.
[18] Hui Cui, Robert H Deng, Guowei Wu, and Junzuo Lai. 2016. An Efficient and Expressive Ciphertext-Policy Attribute-Based Encryption Scheme with Partially Hidden Access Structures. In ProvSec.
[19] Hui Cui, Zhiguo Wan, Robert H Deng, Guilin Wang, and Yingjiu Li. 2016. Efficient and Expressive Keyword Search over Encrypted Data in Cloud. IEEE TDSC (2016).
[20] Antonio de la Piedra et al. 2023. ACABELLA: Automated (Crypt) Analysis of Attribute-Based Encryption Leveraging Linear Algebra. In CCS.
[21] Ruizhong Du and Tianhe Zhang. 2022. Blockchain-Based Ciphertext Policy-Hiding Access Control Scheme. In SecureComm.
[22] Eiichiro Fujisaki and Tatsuaki Okamoto. 1999. Secure Iintegration of Asymmetric and Symmetric Encryption Schemes. In CRYPTO.
[23] Tingyun Gan et al. 2021. Partial Policy Hiding Attribute-Based Encryption in Vehicular Fog Computing. Soft Computing (2021).
[24] Sheng Gao, Guirong Piao, Jianming Zhu, Xindi Ma, and Jianfeng Ma. 2020. Trustaccess: A Trustworthy Secure Ciphertext-Policy and Attribute Hiding Access Control Scheme Based on Blockchain. IEEE TVT (2020).
[25] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. 2006. Attribute-Based Encryption for Fine-Grained Access Control of Encrypted Data. In CCS.
[26] Matthew Green, Susan Hohenberger, Brent Waters, et al. 2011. Outsourcing the Decryption of ABE Ciphertexts.. In USENIX.
[27] Gongcheng Hu et al. 2019. Analysis of Ciphertext Policy Hidden Attribute-Based Encryption and Its Improved Method. In Frontiers in Cyber Security.
[28] Jonathan Katz, Amit Sahai, and Brent Waters. 2008. Predicate Encryption Supporting Disjunctions, Polynomial Equations, and Inner Products. In EUROCRYPT.
[29] Junzuo Lai, Robert H Deng, and Yingjiu Li. 2012. Expressive CP-ABE with Partially Hidden Access Structures. In AsiaCCS.
[30] Junzuo Lai et al. 2011. Fully Secure Cipertext-Policy Hiding CP-ABE. In ISPEC.
[31] Junzuo Lai et al. 2013. Expressive Search on Encrypted Data. In AsiaCCS.
[32] Allison Lewko and Brent Waters. 2011. Unbounded HIBE and Attribute-Based Encryption. In EUROCRYPT.
[33] Allison Lewko and Brent Waters. 2012. New Proof Methods for Attribute-Based Encryption: Achieving Full Security Through Selective Techniques. In CRYPTO.
[34] Allison Lewko et al. 2010. Fully Secure Functional Encryption: Attribute-Based Encryption and (Hierarchical) Inner Product Encryption. In EUROCRYPT.
[35] Jin Li, Kui Ren, Bo Zhu, and Zhiguo Wan. 2009. Privacy-Aware Attribute-Based Encryption with User Accountability. In ISC.
[36] Xiaohui Li, Dawu Gu, Yanli Ren, Ning Ding, and Kan Yuan. 2012. Efficient Ciphertext-Policy Attribute Based Encryption with Hidden Policy. In IDCS.
[37] Lixian Liu, Junzuo Lai, Robert H Deng, and Yingjiu Li. 2016. Ciphertext-Policy Attribute-Based Encryption with Partially Hidden Access Structure and its Application to Privacy-Preserving Electronic Medical Record System in Cloud Environment. Security Comm. Networks (2016).
[38] Zhiquan Lv, Cheng Hong, Min Zhang, and Dengguo Feng. 2014. Expressive and Secure Searchable Encryption in the Public Key Setting. In ISC.
[39] Mriganka Mandal. 2020. Privacy-Preserving Fully Anonymous Ciphertext Policy Attribute-Based Broadcast Encryption with Constant-Size Secret Keys and Fast Decryption. JISA (2020).
[40] Long Meng, Liqun Chen, Yangguang Tian, Mark Manulis, and Suhui Liu. 2024. FEASE: Fast and Expressive Asymmetric Searchable Encryption. Accepted in USENIX Security Symposium. https://eprint.iacr.org/2024/054 https://eprint.iacr.org/2024/054.
[41] Ru Meng, Yanwei Zhou, Jianting Ning, Kaitai Liang, Jinguang Han, and Willy Susilo. 2017. An Efficient Key-Policy Attribute-Based Searchable Encryption in Prime-Order Groups. In ProvSec.
[42] Takashi Nishide, Kazuki Yoneyama, and Kazuo Ohta. 2008. Attribute-Based Encryption with Partially Hidden Encryptor-Specified Access Structures. In ACNS.
[43] Tatsuaki Okamoto and Katsuyuki Takashima. 2012. Adaptively Attribute-Hiding (Hierarchical) Inner Product Encryption. In EUROCRYPT.
[44] Tatsuaki Okamoto and Katsuyuki Takashima. 2012. Fully Secure Unbounded Inner-Product and Attribute-Based Encryption. In ASIACRYPT.
[45] Bryan Parno et al. 2012. How to delegate and verify in public: Verifiable computation from attribute-based encryption. In TCC.
[46] Tran Viet Xuan Phuong et al. 2015. Hidden Ciphertext Policy Attribute-Based Encryption Under Standard Assumptions. IEEE TIFS (2015).
[47] Y Sreenivasa Rao and Ratna Dutta. 2013. Recipient Anonymous Ciphertext-Policy Attribute Based Encryption. In ICISS.
[48] Y Rao Sreenivasa et al. 2015. Fully Secure Bandwidth-Efficient Anonymous Ciphertext-Policy Attribute-Based Encryption. Security Comm. Networks (2015).
[49] Doreen Riepel and Hoeteck Wee. 2022. FABEO: Fast Attribute-Based Encryption with Optimal Security. In CCS.
[50] Yannis Rouselakis and Brent Waters. 2013. Practical Constructions and New Proof Methods for Large Universe Attribute-Based Encryption. In CCS.
[51] Amit Sahai and Brent Waters. 2005. Fuzzy Identity-Based Encryption. In EUROCRYPT.
[52] Nuno Santos, Rodrigo Rodrigues, K. Gummadi, and Stefan Saroiu. 2012. Policy-Sealed Data: A New Abstraction for Building Trusted Cloud Services. In USENIX.
[53] Huiyong Wang, Jialing Liang, Yong Ding, Shijie Tang, and Yujue Wang. 2023. Ciphertext-Policy Attribute-Based Encryption Supporting Policy-Hiding and Cloud Auditing in Smart Health. Computer Standards & Interfaces (2023).
[54] Brent Waters. 2009. Dual System Encryption: Realizing Fully Secure IBE and HIBE under Simple Assumptions. In CRYPTO.
[55] Brent Waters. 2011. Ciphertext-Policy Attribute-Based Encryption: An Expressive, Efficient, and Provably Secure Realization. In PKC.
[56] Hu Xiong et al. 2019. Partially Policy-Hidden Attribute-Based Broadcast Encryption with Secure Delegation in Edge Computing. FGCS (2019).
[57] Hu Xiong et al. 2021. Efficient Unbounded Fully Attribute Hiding Inner Product Encryption in Cloud-Aided WBANs. IEEE Systems Journal (2021).
[58] Peng Zeng, Zhiting Zhang, Rongxing Lu, and Kim-Kwang Raymond Choo. 2021. Efficient Policy-Hiding and Large Universe Attribute-Based Encryption with Public Traceability for Internet of Medical Tthings. IEEE IoT Journal (2021).
[59] Leyou Zhang, Yilei Cui, and Yi Mu. 2019. Improving Security and Privacy Attribute Based Data Sharing in Cloud Computing. IEEE Systems Journal (2019).
[60] Yinghui Zhang, X. Chen, Jin Li, Duncan S Wong, and Hui Li. 2013. Anonymous attribute-based encryption supporting efficient decryption test. In AsiaCCS.
[61] Yinghui Zhang, Xiaofeng Chen, Jin Li, Duncan S Wong, Hui Li, and Ilsun You. 2017. Ensuring Attribute Privacy Protection and Fast Decryption for Outsourced Data Security in Mobile Cloud Computing. Information Sciences (2017).
[62] Yinghui Zhang et al. 2018. Security and Privacy in Smart Health: Efficient Policy-Hiding Attribute-Based Access Control. IEEE IoT Journal (2018).

A SUPPORTING ATTRIBUTE MULTI-USE

We define $\rho(i) = |\{z \mid \pi(z) = \pi(i),\ z \le i\}|$ to denote the $\rho(i)$-th occurence of the attribute $\pi(i)$, and define $\tau = \max_{i\in[\ell]} \rho(i)$ to represent the maximum number of times an attribute is used in M. Other notation is the same as in Fig. 4 and Fig. 5.

$\underline{(\mathsf{pk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)}$.

Run $\mathsf{GroupGen}(1^\lambda)$ to obtain the group parameters $\mathsf{par} = (p,\ e,\ \mathbb{G}_1,\ \mathbb{G}_2,\ \mathbb{G}_T,\ g_1,\ g_2)$. Pick $\alpha,\ b_1,\ b_2 \xleftarrow{\$} \mathbb{Z}_p$ and a hash function $\mathsf{H}: \{0,1\}^* \to \mathbb{G}_1$. Compute

$$\mathsf{pk} = (\mathsf{H}, \mathsf{par}, g_2^{b_1}, g_2^{b_2}, e(g_1, g_2)^\alpha),\ \mathsf{msk} = (\alpha, b_1, b_2).$$

$\underline{(\mathsf{sk}, \mathsf{pl}_{\mathbb{A}}) \leftarrow \mathsf{KeyGen}(\mathsf{pk}, \mathsf{msk}, \mathbb{A} = (\mathrm{M}, \pi, \{\pi(i)\}_{i\in[\ell]}))}$.

Pick $\mathbf{r} \xleftarrow{\$} \mathbb{Z}_{\mathbf{p}}^\tau$, $\mathbf{v} \xleftarrow{\$} \mathbb{Z}_p^{n-1}$. Compute

$$\mathsf{sk}_{1,j} = g_2^{\mathbf{r}[j]},\ \mathsf{sk}_{2,i} = g_1^{\mathrm{M}_i(\alpha\|\mathbf{v})^\top} \cdot \mathsf{H}(\pi(i))^{-\mathbf{r}[\rho(i)]},$$

$$\mathsf{sk}_{3,i} = \mathsf{H}(0\,\|\,\pi(i))^{\frac{\mathbf{r}[\rho(i)]}{b_1}},\ \mathsf{sk}_{4,i} = \mathsf{H}(1\,\|\,\pi(i))^{\frac{\mathbf{r}[\rho(i)]}{b_2}}.$$

Output $\mathsf{sk} = (\{\mathsf{sk}_{1,j}\}_{j\in[\tau]}, \{\mathsf{sk}_{2,i}, \mathsf{sk}_{3,i}, \mathsf{sk}_{4,i}\}_{i\in[\ell]})$, $\mathsf{pl}_{\mathbb{A}} = \mathbb{A}$.

$\underline{(\mathsf{ct}, \mathsf{pl}_{\mathbb{S}}) \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathbb{S} = \{u_i\}_{i\in[m]}, \mathsf{msg})}$.

Pick $s_1, s_2 \xleftarrow{\$} \mathbb{Z}_p$, let $s = s_1 + s_2$. Compute

$$\mathsf{ct}_{1,i} = \mathsf{H}(u_i)^s \cdot \mathsf{H}(0\,\|\,u_i)^{s_1} \cdot \mathsf{H}(1\,\|\,u_i)^{s_2},\ \mathsf{ct}_2 = g_2^s,$$

$$\mathsf{ct}_3 = g_2^{b_1 s_1},\ \mathsf{ct}_4 = g_2^{b_2 s_2},\ \mathsf{ct}_5 = e(g_1, g_2)^{\alpha s} \cdot \mathsf{msg}.$$

Output $\mathsf{ct} = (\{\mathsf{ct}_{1,i}\}_{i\in[m]}, \mathsf{ct}_2, \mathsf{ct}_3, \mathsf{ct}_4, \mathsf{ct}_5)$, $\mathsf{pl}_{\mathbb{S}} = \mathbb{S}$.

$\underline{\mathsf{msg}/\bot \leftarrow \mathsf{Dec}(\mathsf{ct}, \mathsf{pl}_{\mathbb{S}}, \mathsf{sk}, \mathsf{pl}_{\mathbb{A}})}$.

Tests if there is any subset $\mathcal{I}$ that matches $\mathbb{S}$ in ct with $\mathbb{A}$ in sk. If not, return $\bot$. Otherwise, it finds constants $\{\gamma_i\}_{i\in\mathcal{I}}$ s.t. $\sum_{i\in\mathcal{I}} \gamma_i \mathrm{M}_i = (1, 0, \ldots, 0)$ and reconstructs the message msg by computing:

$$\frac{\mathsf{ct}_5 \cdot e\Big(\prod_{i\in\mathcal{I}} (\mathsf{sk}_{3,i})^{\gamma_i}, \mathsf{ct}_3\Big) \cdot e\Big(\prod_{i\in\mathcal{I}} (\mathsf{sk}_{4,i})^{\gamma_i}, \mathsf{ct}_4\Big)}{\prod_{j\in[\tau]} e\Big(\prod_{i\in\mathcal{I},\rho(i)=j} (\mathsf{ct}_{1,\pi(i)})^{\gamma_i}, \mathsf{sk}_{1,j}\Big) \cdot e\Big(\prod_{i\in\mathcal{I}} (\mathsf{sk}_{2,i})^{\gamma_i}, \mathsf{ct}_2\Big)}.$$

Figure 8: Our KP-ABE scheme with attribute multi-use.

$\underline{(\mathsf{pk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)}$.

Run $\mathsf{GroupGen}(1^\lambda)$ to obtain the group parameters $\mathsf{par} = (p,\ e,\ \mathbb{G}_1,\ \mathbb{G}_2,\ \mathbb{G}_T,\ g_1,\ g_2)$. Pick $\alpha,\ b_1,\ b_2 \xleftarrow{\$} \mathbb{Z}_p$, $g_3 \in \mathbb{G}_1$, and a hash function $\mathsf{H}: \{0,1\}^* \to \mathbb{G}_1$. Compute

$$\mathsf{pk} = (\mathsf{H}, \mathsf{par}, g_3, g_2^{b_1}, g_2^{b_2}, e(g_1, g_2)^\alpha),\ \mathsf{msk} = (\alpha, b_1, b_2).$$

$\underline{(\mathsf{sk}, \mathsf{pl}_{\mathbb{S}}) \leftarrow \mathsf{KeyGen}(\mathsf{pk}, \mathsf{msk}, \mathbb{S} = \{u_i\}_{i\in[m]})}$.

Pick $r \xleftarrow{\$} \mathbb{Z}_p$. Compute

$$\mathsf{sk}_1 = g_2^r,\ \mathsf{sk}_2 = g_1^\alpha g_3^{-r},$$

$$\mathsf{sk}_{3,i} = \mathsf{H}(0\,\|\,u_i)^{\frac{r}{b_1}},\ \mathsf{sk}_{4,i} = \mathsf{H}(1\,\|\,u_i)^{\frac{r}{b_2}}.$$

Output $\mathsf{sk} = (\mathsf{sk}_1, \mathsf{sk}_2, \{\mathsf{sk}_{3,i}, \mathsf{sk}_{4,i}\}_{i\in[m]})$, $\mathsf{pl}_{\mathbb{S}} = \mathbb{S}$.

$\underline{(\mathsf{ct}, \mathsf{pl}_{\mathbb{A}}) \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathbb{A} = (\mathrm{M}, \pi, \{\pi(i)\}_{i\in[\ell]}, \mathsf{msg})}$.

Pick $\mathbf{s_1}, \mathbf{s_2} \xleftarrow{\$} \mathbb{Z}_p^\tau$, $\mathbf{v} \xleftarrow{\$} \mathbb{Z}_p^{n-1}$, let $s = \mathbf{s_1}[1] + \mathbf{s_2}[1]$. Compute

$$\mathsf{ct}_{1,i} = g_3^{\mathrm{M}_i(s\|\mathbf{v})^\top} \cdot \mathsf{H}(0\,\|\,\pi(i))^{\mathbf{s_1}[\rho(i)]} \cdot \mathsf{H}(1\,\|\,\pi(i))^{\mathbf{s_2}[\rho(i)]},\ \mathsf{ct}_2 = g_2^s,$$

$$\mathsf{ct}_{3,j} = g_2^{b_1\cdot \mathbf{s_1}[j]},\ \mathsf{ct}_{4,j} = g_2^{b_2\cdot \mathbf{s_2}[j]},\ \mathsf{ct}_5 = e(g_1, g_2)^{\alpha s} \cdot \mathsf{msg}.$$

Output $\mathsf{ct} = (\{\mathsf{ct}_{1,i}\}_{i\in[\ell]}, \mathsf{ct}_2, \{\mathsf{ct}_{3,j}, \mathsf{ct}_{4,j}\}_{j\in[\tau]}, \mathsf{ct}_5)$, $\mathsf{pl}_{\mathbb{A}} = \mathbb{A}$.

$\underline{\mathsf{msg}/\bot \leftarrow \mathsf{Dec}(\mathsf{ct}, \mathsf{pl}_{\mathbb{A}}, \mathsf{sk}, \mathsf{pl}_{\mathbb{S}})}$.

Tests if there is any subset $\mathcal{I}$ that matches $\mathbb{S}$ in sk with $\mathbb{A}$ in ct. If not, return $\bot$. Otherwise, it finds constants $\{\gamma_i\}_{i\in\mathcal{I}}$ s.t. $\sum_{i\in\mathcal{I}} \gamma_i \mathrm{M}_i = (1,\ 0,\ \ldots,\ 0)$. Let $D_3 = \prod_{j\in[\tau]} e\Big(\prod_{i\in\mathcal{I},\rho(i)=j} (\mathsf{sk}_{3,i})^{\gamma_i},\ \mathsf{ct}_{3,j}\Big)$, $D_4 = \prod_{j\in[\tau]} e\Big(\prod_{i\in\mathcal{I},\rho(i)=j} (\mathsf{sk}_{4,i})^{\gamma_i},\ \mathsf{ct}_{4,j}\Big)$. Reconstructs the message msg by computing:

$$\frac{\mathsf{ct}_5 \cdot D_3 \cdot D_4}{e\Big(\prod_{i\in\mathcal{I}} (\mathsf{ct}_{1,\pi(i)})^{\gamma_i}, \mathsf{sk}_1\Big) \cdot e\Big(\mathsf{sk}_2, \mathsf{ct}_2\Big)}.$$

Figure 9: Our CP-ABE scheme with attribute multi-use.

B CUI ET AL.'S SCHEMES IN TYPE-III SETTING

Fig. 10 and Fig. 11 display the KP-A^{2}BE and CP-A^{2}BE schemes [19] and [18] that we implemented in Sec. 5. We transformed both schemes into the asymmetric Type-III setting, and we omitted the message authentication as it is out of the scope of this paper.

$\underline{(\mathsf{pk}, \mathsf{msk}) \leftarrow \mathsf{KeyGen}(1^\lambda)}$. Run $\mathsf{GroupGen}(1^\lambda)$ to obtain group parameters $\mathsf{par} = (p, e, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2)$. Pick $\delta_1, h_1, w_1 \xleftarrow{\$} \mathbb{G}_1, \delta_2, h_2, w_2 \xleftarrow{\$} \mathbb{G}_2, \alpha, d_1, d_2, d_3, d_4 \xleftarrow{\$} \mathbb{Z}_p$. Compute the public key and master secret key as

$$\mathsf{pk} = (\mathsf{par}, g_1, \delta_1, h_1, w_1, g_1^{d_1}, g_1^{d_2}, g_1^{d_3}, g_1^{d_4}, e(g_1, g_2)^\alpha).$$
$$\mathsf{msk} = (\alpha, g_2, \delta_2, h_2, w_2, d_1, d_2, d_3, d_4).$$

$\underline{(\mathsf{sk}, \mathsf{pl}_{\mathbb{A}}) \leftarrow \mathsf{KeyGen}(\mathsf{pk}, \mathsf{msk}, \mathbb{A} = (\mathrm{M}, \pi, \{\pi(i)\}_{i\in[\ell]}))}$. Let $\{\pi(i)\}_{i\in[\ell]} = \{n_{\pi(i)}, v_{\pi(i)}\}_{i\in[\ell]}$. Pick $r, r', t_{1,1}, t_{1,2}, \ldots, t_{\ell,1}, t_{\ell,2} \xleftarrow{\$} \mathbb{Z}_p, \mathbf{v} \xleftarrow{\$} \mathbb{Z}_p^{n-1}$. Compute

$$\mathsf{sk}_{1,i} = g_2^{\mathrm{M}_i(\alpha\|\mathbf{v})^\top} \cdot w_2^{d_1 d_2 t_{i,1} + d_3 d_4 t_{i,2}}, \mathsf{sk}_{2,i} = g_2^{d_1 d_2 t_{i,1} + d_3 d_4 t_{i,2}},$$
$$\mathsf{sk}_{3,i} = (\delta_2^{\pi(i)} h_2)^{-d_2 t_{i,1}}, \mathsf{sk}_{4,i} = (\delta_2^{\pi(i)} h_2)^{-d_1 t_{i,1}},$$
$$\mathsf{sk}_{5,i} = (\delta_2^{\pi(i)} h_2)^{-d_4 t_{i,2}}, \mathsf{sk}_{6,i} = (\delta_2^{\pi(i)} h_2)^{-d_3 t_{i,2}}$$

$\mathsf{sk} = (\{\mathsf{sk}_{1,i}, \mathsf{sk}_{2,i}, \mathsf{sk}_{3,i}, \mathsf{sk}_{4,i}, \mathsf{sk}_{5,i}, \mathsf{sk}_{6,i}\}_{i\in[\ell]})$, $\mathsf{pl}_{\mathbb{A}} = (\mathrm{M}, \pi, \{n_{\pi(i)}\}_{i\in[\ell]})$.

$\underline{(\mathsf{ct}, \mathsf{pl}_{\mathbb{S}}) \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathbb{S} = \{u_i\}_{i\in[m]} = \{n_i, v_i\}_{i\in[m]}, \mathsf{msg})}$.

Pick $\mu, s_{1,1}, s_{1,2}, \ldots, s_{m,1}, s_{m,2}, z_1, \ldots, z_m \xleftarrow{\$} \mathbb{Z}_p$. Compute $\mathsf{ct}_1 = g_1^\mu, \mathsf{ct}_{2,i} = w_1^{-\mu} (\delta_1^{u_i} h_1)^{z_i}$,

$$\mathsf{ct}_{3,i} = g_1^{d_1(z_i - s_{i,1})}, \mathsf{ct}_{4,i} = g_1^{d_2 s_{i,1}}, \mathsf{ct}_{5,i} = g_1^{d_3(z_i - s_{i,2})},$$
$$\mathsf{ct}_{6,i} = g_1^{d_4 s_{i,2}}, \mathsf{ct}_7 = e(g_1, g_2)^{\alpha\mu} \cdot \mathsf{msg}$$

$\mathsf{ct} = (\mathsf{ct}_1, \{\mathsf{ct}_{2,i}, \mathsf{ct}_{3,i}, \mathsf{ct}_{4,i}, \mathsf{ct}_{5,i}, \mathsf{ct}_{6,i}\}_{i\in[m]}, \mathsf{ct}_7)$, $\mathsf{pl}_{\mathbb{S}} = \{n_i\}_{i\in[m]}$.

$\underline{\mathsf{msg}/\bot \leftarrow \mathsf{Dec}(\mathsf{ct}, \mathsf{sk})}$. Tests if there is any subset $\mathcal{I}$ that matches $\{n_i\}_{i\in[m]}$ in ct with $(\mathrm{M}, \pi, \{n_{\pi(i)}\})$ in sk. If not, return $\bot$. Otherwise, it finds constants $\{\gamma_i\}_{i\in\mathcal{I}}$ s.t. $\sum_{i\in\mathcal{I}} \gamma_i \mathrm{M}_i = (1, 0, \ldots, 0)$ and computes

$$\mathsf{T} = \prod_{i\in\mathcal{I}} (e(\mathsf{ct}_1, \mathsf{sk}_{1,i}) e(\mathsf{ct}_{2,i}, \mathsf{sk}_{2,i}) e(\mathsf{ct}_{3,i}, \mathsf{sk}_{3,i})$$
$$e(\mathsf{ct}_{4,i}, \mathsf{sk}_{4,i}) e(\mathsf{ct}_{5,i}, \mathsf{sk}_{5,i}) e(\mathsf{ct}_{6,i}, \mathsf{sk}_{6,i}))^{\gamma_i},$$

and then calculate $\mathsf{msg} = \mathsf{ct}_7 \cdot \mathsf{T}^{-1}$. If the message is not correct, find another subset of $\mathcal{I}$ and repeat the checking. If the message cannot be recovered for all subsets, return $\bot$.

Figure 10: The construction of CWD+ [19] transformed into a KP-A^{2}BE scheme in Type-III setting

$\underline{(\mathsf{pk}, \mathsf{msk}) \leftarrow \mathsf{KeyGen}(1^\lambda)}$. Run $\mathsf{GroupGen}(1^\lambda)$ to obtain group parameters $\mathsf{par} = (p, e, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2)$. Pick $g_1, \delta_1, h_1, v_1, w_1 \in \mathbb{G}_1, g_2, \delta_2, h_2, v_2, w_2 \in \mathbb{G}_2, \alpha, d_1, d_2, d_3, d_4 \in \mathbb{Z}_p$. Compute the public key and master secret key as

$$\mathsf{pk} = (\mathsf{par}, g_1, \delta_1, h_1, v_1, w_1, g_1^{d_1}, g_1^{d_2}, g_1^{d_3}, g_1^{d_4}, e(g_1, g_2)^\alpha).$$
$$\mathsf{msk} = (g_2^\alpha, g_2, \delta_2, h_2, v_2, w_2, d_1, d_2, d_3, d_4).$$

$\underline{(\mathsf{sk}, \mathsf{pl}_{\mathbb{S}}) \leftarrow \mathsf{KeyGen}(\mathsf{pk}, \mathsf{msk}, \mathbb{S} = \{u_i\}_{i\in[m]} = \{n_i, v_i\}_{i\in[m]})}$. Pick $r, r', r_1, \ldots, r_m, r'_1, \ldots, r'_m \leftarrow \mathbb{Z}_p$. Compute

$$\mathsf{sk}_1 = g_2^\alpha w_2^{d_1 d_2 r + d_3 d_4 r'},$$
$$\mathsf{sk}_2 = g_2^{d_1 d_2 r + d_3 d_4 r'}, \mathsf{sk}_{3,i} = g_2^{d_1 d_2 r_i + d_3 d_4 r'_i},$$
$$\mathsf{sk}_{4,i} = ((\delta_2^{u_i} h_2)^{r_i} v_2^{-r})^{d_2}, \mathsf{sk}_{5,i} = ((\delta_2^{u_i} h_2)^{r_i} v_2^{-r})^{d_1},$$
$$\mathsf{sk}_{6,i} = ((\delta_2^{u_i} h_2)^{r'_i} v_2^{-r'})^{d_4}, \mathsf{sk}_{7,i} = ((\delta_2^{u_i} h_2)^{r'_i} v_2^{-r'})^{d_3}.$$

$\mathsf{sk} = (\mathsf{sk}_1, \mathsf{sk}_2, \{\mathsf{sk}_{3,i}, \mathsf{sk}_{4,i}, \mathsf{sk}_{5,i}, \mathsf{sk}_{6,i}, \mathsf{sk}_{7,i}\}_{i\in m})$, $\mathsf{pl}_{\mathbb{S}} = \{n_i\}_{i\in[m]}$.

$\underline{(\mathsf{ct}, \mathsf{pl}_{\mathbb{A}}) \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathbb{A} = (\mathrm{M}, \pi, \{\pi(i)\}_{i\in[\ell]}, \mathsf{msg})}$. Let $\{\pi(i)\}_{i\in[\ell]} = \{n_{\pi(i)}, v_{\pi(i)}\}_{i\in[\ell]}$. Pick $\mu, s_{i,1}, \ldots, s_{i,\ell}, z_1, \ldots, z_\ell \leftarrow \mathbb{Z}_p, \mathbf{v} \leftarrow \mathbb{Z}_p^{n-1}$. Compute

$$\mathsf{ct}_1 = g_1^\mu, \mathsf{ct}_{2,i} = w_1^{\mathrm{M}_i(\mu\|\mathbf{v})^\top} v_1^{z_i},$$
$$\mathsf{ct}_{3,i} = (\delta_1^{\pi(i)} h_1)^{-z_i}, \mathsf{ct}_{4,i} = g_1^{d_1 s_{i,1}},$$
$$\mathsf{ct}_{5,i} = g_1^{d_2(z_i - s_{i,1})}, \mathsf{ct}_{6,i} = g_1^{d_3 s_{i,2}},$$
$$\mathsf{ct}_{7,i} = g_1^{d_4(z_i - s_{i,2})}, \mathsf{ct}_8 = e(g_1, g_2)^{\alpha\mu} \cdot \mathsf{msg}.$$

$\mathsf{ct} = (\mathsf{ct}_1, \{\mathsf{ct}_{2,i}, \mathsf{ct}_{3,i}, \mathsf{ct}_{4,i}, \mathsf{ct}_{5,i}, \mathsf{ct}_{6,i}, \mathsf{ct}_{7,i}\}_{i\in[\ell]}, \mathsf{ct}_8)$, $\mathsf{pl}_{\mathbb{A}} = (\mathrm{M}, \pi, \{n_{\pi(i)}\}_{i\in[\ell]})$.

$\underline{\mathsf{msg}/\bot \leftarrow \mathsf{Dec}(\mathsf{ct}, \mathsf{sk})}$. Tests if there is any subset $\mathcal{I}$ that matches $\{n_i\}_{i\in[m]}$ in sk with $(\mathrm{M}, \pi, \{n_{\pi(i)}\})$ in ct. If not, return $\bot$. Otherwise, it finds constants $\{\gamma_i\}_{i\in\mathcal{I}}$ s.t. $\sum_{i\in\mathcal{I}} \gamma_i \mathrm{M}_i = (1, 0, \ldots, 0)$ and computes:

$$\mathsf{T} = e(\mathsf{ct}_1, \mathsf{sk}_1)^{-1} \prod_{i\in\mathcal{I}} (e(\mathsf{ct}_{2,i}, \mathsf{sk}_2) e(\mathsf{ct}_{3,i}, \mathsf{sk}_{3,i})$$
$$e(\mathsf{ct}_{4,i}, \mathsf{sk}_{4,i}) e(\mathsf{ct}_{5,i}, \mathsf{sk}_{5,i}) e(\mathsf{ct}_{6,i}, \mathsf{sk}_{6,i}) e(\mathsf{ct}_{7,i}, \mathsf{sk}_{7,i}))^{\gamma_i},$$

and then calculate $\mathsf{msg} = \mathsf{ct}_8 \cdot \mathsf{T}$. If the message is not correct, find another subset of $\mathcal{I}$ and repeat the checking. If the message cannot be recovered for all subsets, return $\bot$.

Figure 11: The construction of CDW+ [18] CP-A^{2}BE in Type-III setting