Flexibly and Securely Shape Your Data Disclosed to Others

Qingqing Xie*
Jiangsu University
Zhenjiang, Jiangsu

Yantian Hou†
Boise State University
Boise, Idaho
yantianhou@boisestate.edu

Ke Cheng*
Xidian University
Xi'an, China
kechengstu@gmail.com

Gaby G. Dagher
Boise State University
Boise, Idaho, USA
gabydagher@boisestate.edu

Liangmin Wang
Jiangsu University
Zhenjiang, Jiangsu, China
wanglm@ujs.edu.cn

Shucheng Yu
Stevens Institute of Technology
Hoboken, New Jersey, USA
syu19@stevens.edu

ABSTRACT

This work is to enhance existing fine-grained access control to support a more expressive access policy over arithmetic operation results. We aim to enable data owners to flexibly bind a user's identity with his/her authorized access target according to a given access control policy, which indicates how a piece of data obfuscated by different noises. To this end, we design a cryptographic primitive that decouples the noisy data to two components, one associated with user identity, and the other one shared and dynamically changes, with the composite of these two components evaluated and revealed at user sides. The security of our scheme is formally proven using game based approach. We implement our system on a commercial cloud platform and use extensive experiments to validate its functionality and performance.

CCS CONCEPTS

• **Security and privacy → Access control**; *Privacy protections*; • **Theory of computation → Cryptographic primitives**.

KEYWORDS

fine-grained access control; homomorphic encryption; privacy-preserving cloud computing

ACM Reference Format:
Qingqing Xie, Yantian Hou, Ke Cheng, Gaby G. Dagher, Liangmin Wang, and Shucheng Yu. 2019. Flexibly and Securely Shape Your Data Disclosed to Others. In *ACM Asia Conference on Computer and Communications Security (AsiaCCS '19), July 9–12, 2019, Auckland, New Zealand.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3321705.3329851

*Part of this work is done by Qingqing Xie and Ke Cheng as visiting students under Dr. Yantian Hou's advisory in Boise State University.
†Yantian Hou is the corresponding author.

1 INTRODUCTION

Cryptography-based fine-grained access control allows a data owner to flexibly bind each of his/her data with an arbitrary user identity for using the data. With this approach, data owners do not need to trust any centralized gateway as used in the traditional access control [10], which has been reported for abusing our data without permissions [3]. However, the usages of fine-grained access control are restrained in practice because they lack support for an expressive function-level control, i.e., the access control targets are limited to only the data itself, instead of any of its functional operations.

Adding functional features to fine-grained access control has been an open problem, which indicates not only which user could access what data, but also how to use that data. Enhancing the access control from data level to function level is critical for realizing a more expressive access control mechanism enforced by data owner. Recently several researchers have started combining attribute based encryption [2, 14] with homomorphic encryption [13], which allows arithmetic operations over ciphertexts that are associated with different attributes. Though facing several challenges such as the exponentially increased ciphertext size, these studies could advance the applications that demand both complicated on-line operations and flexible access control over the data hosted by untrusted clouds/servers.

In this work, we aim to empower data owners to control the access to their data in a more expressive and efficient manner, by combining fine-grained access control with homomorphism. Different from previous works, our goal is to perform additive operation across an identity-associated (the identity is denoted by user's attributes) component, and a dynamically shared component, as shown in Fig. 1. Our realization is driven by the need for fine-grained access control on the same data, but with different precisions for different data users. For example, a data owner deploys the ciphertexts of zero noise, light noise, and heavy noise to his family, common friends, and the "finding-nearest-restaurant" application respectively. Meanwhile, the data owner publishes the ciphertexts of the geo-location coordinates (x, y) in runtime. Each of the three types of users could only access its authorized noisy version through online homomorphic operations. Our solution should meet the following requirements: 1) Flexibility. Each user could be arbitrarily associated with a precision version according to a fine-grained access policy enacted by the data owner. 2) Security. Each user could only access its associated version according to its identity defined by his/her attributes. 3) Efficiency. The data owner should be able to efficiently enact and enforce the fine-grained access policy, regardless of the policy's size.

To meet these requirements simultaneously, we propose a cryptographic primitive with a novel decoupling technique. In our approach, the shared and identity-associated components are encrypted separately while holding additive homomorphism. When decrypting, each user first "plugs" his/her ciphertext of identity-associated component into the ciphertext of the shared-component, and then unlocks the composite with only his/her secret key to derive the noisy value.

The decoupling of the two components is a desirable feature for data owner in practice. Enabled by it, the data owner only needs to focus on the encryption of the single data component in runtime by treating it as the shared component, which yields constant overhead regardless of the granularity (the number of noises). Meanwhile the enactment of fine-grained access policy and the noise-component (which is treated as the identity-associated component) encryptions could be entirely offloaded (e.g., to a separate policy enactment module, a professional agent whom the data owner trusts, or a parent of the child). This advantage could help the non-professional data owners, who are deficient in access control knowledge or resources, to better protect their data privacy.

None of previous efforts could satisfy our requirements. Recently, several cryptography works have made efforts to establish fine-grained access control over general arithmetic operation results by combining homomorphic encryptions with attribute-based encryptions (ABE) [2, 9, 14]. Compared to our work, they consider a different homomorphism across the ciphertexts associated with the same/different attributes, by enabling authorized user/users to access not only the computational result, but also the operands. In addition, several of their schemes suffer from the exploded ciphertext sizes, yielding exponentially large overhead for both the data owners and the data storage in practice.

Our cryptographic scheme is formally proven secure through the proof-by-reduction technique and the game-based approach. We implement our system and deploy it on a commercial cloud platform, through which the data owner performs the 1-to-n data sharing towards multiple users with different precision levels. We use extensive experiments to validate its complexity.

Summary of Contributions: 1) We propose a cryptographic primitive, which is named as fine-grained access precision control (FAPC). Our design is the first generic cryptographic primitive that enables fine-grained access control over the additive computational results of different identity-associated components and a shared component, with only compact ciphertext size. 2) The security of our scheme is formally presented through the game-based approach. 3) We establish a prototype system on a commercial cloud platform, where the performance of our design is validated through extensive experiments.

$$\boxed{key_i} + \boxed{[x]_i} \xrightarrow{\text{Decrypt}} \boxed{x}$$

$$\boxed{key_i} + \boxed{[x_i]_i} + \boxed{[y]} \xrightarrow{\text{Decrypt}} \boxed{x_i + y}$$

Figure 1: The traditional fine-grained access control (top), and our fine-grained control combined with homomorphic operation of a shared component and an identity-associated component (bottom). $[x]_i$ **denotes the encryption of** x **under identity i's key.**

The rest of the paper is organized as follows:

In Sec. 2 we introduce our system model and formally define the problem. The FAPC primitive design is presented in Sec. 3. The security and complexity analysis of our design, as well as its experimental evaluations are presented in Sec. 4. Sec. 5 reviews the related work. Sec. 6 concludes this paper.

2 PROBLEM STATEMENT

The problem we are studying is fine-grained access precision control (FAPC). Specifically, our goal is to enable the data owner (DO) to associate each shared data with various noises, and disclose the noisy data only to authorized user identity/identities. Different from state-of-the-art works, our solution requires this task of fine-grained access control to be lightweight for the data owner, thus scalable to support a large number of users simultaneously. Meanwhile, it should be privacy-preserving, thus no unauthorized entity (e.g., unauthorized users, untrusted cloud platform for data sharing) could access the noisy data. Before formulating our problem, we first define the noise function and the access precision control policy as follows:

Definition 2.1 (Noisy Data). *Use $\mathbb{DS}$ to denote the data space. Given a data $d \in \mathbb{DS}$, a noise $\epsilon \in \mathbb{DS}$, and any noise function f, a noisy data is denoted as $d_\epsilon = f(d, \epsilon)$.*

Definition 2.2 (Access Precision Control Policy). *Use $\mathbb{US}$ to denote the user space. The access precision control policy set Γ is a set of identity-precision pairs (i, ϵ_j), each associating a user's identity $i \in \mathbb{US}$ with a noise version ϵ_j.*

The noise function f could be in any form. In this work, we consider the basic additive function. Given the definition of noisy data and the access precision control policy, we could then formulate the FAPC problem as follows:

Definition 2.3 (FAPC problem). *Given any data $d \in \mathbb{DS}$ and a policy set Γ, find a scheme Π, such that the noisy data $d_{\epsilon_j} = f(d, \epsilon_j)$ could be revealed to user i, iff $(i, \epsilon_j) \in \Gamma$.*

Briefly, the FAPC problem is to find a scheme Π such that a user i could only access its permitted noisy data $f(d, \epsilon_j)$ according to a pre-defined policy Γ. In practice, the policy Γ is normally defined by the owner of data d. In the cases where the data owner demands the finest policy granularity, each noisy version is associated with a unique user identity.

3 OUR SOLUTION

To solve our problem, we design a cryptographic primitive, that is partially based on the CP-ABE scheme [1]. Our scheme decouples the ciphertext into a dynamically shared component and identity-associated components, and then reuses the shared component to pair with each identity-associated component in running time when decrypted.

3.1 Framework

The FAPC scheme consists of five algorithms:

• $Setup(1^{\lambda}) \rightarrow mk, pk$: is a probabilistic algorithm. It takes as input a security parameter 1^{λ}, then outputs a master key mk and a public key pk.

• $Encrypt'(pk, m, k, T_{\forall}) \rightarrow CT_m$: is a probabilistic algorithm. It takes the public key pk, a plaintext m (the shared component), a session key $k \in \mathbb{G}_T$ and a corresponding access tree $T_{\forall}$, then outputs the ciphertext CT_m of data m, where $T_{\forall}$ is constructed in such a way that any authorized user, whose attribute set satisfying any tree $T_i, i \in [1, N]$, should satisfy $T_{\forall}$ as well.

• $Encrypt''(pk, \mathcal{N}, \mathcal{T}, k) \rightarrow \mathcal{E}_{T_{\forall}}(k), CT_{\mathcal{N}}$: is a probabilistic algorithm. It takes as input a public key pk, a plaintext set $\mathcal{N} = \{\tau_i\}_{i \in [1,N]}$ (the identity-associated components), a set of access trees (as described in Appendix. $\mathcal{T} = \{T_i\}_{i \in [1,N]}$ [1], a session key k, then outputs the ciphertext $\mathcal{E}_{T_{\forall}}(k)$ of session key k and the ciphertext set $CT_{\mathcal{N}} = \{CT_{\tau_i}\}_{i \in [1,N]}$ of plaintext set $\mathcal{N}$.

• $Update(pk, m, CT_m, k, m') \rightarrow CT_{m'}$: is a probabilistic algorithm. It takes as input the data m and its corresponding ciphertext CT_m, the session key k, and the new data m', then outputs the ciphertext $CT_{m'}$ of m'.

• $KeyGenerate(pk, mk, \mathbb{A}^u) \rightarrow \mathbb{SK}_u$: is a probabilistic algorithm. It takes as input a public key pk, a master key mk, and an attribute set $\mathbb{A}^u$ corresponding to a user u's identity, then outputs a secret attribute key $\mathbb{SK}_u$ associated with $\mathbb{A}^u$.

• $Decrypt(pk, CT_m^{\tau_i}, \mathcal{E}_{T_{\forall}}(k), \mathbb{SK}_u) \rightarrow f'(m, \tau_i)$ or *null*: is a deterministic algorithm. It takes as input the public key pk, a ciphertext pair $CT_m^{\tau_i} = \{CT_m, CT_{\tau_i}\}$ of a certain data-noise pair (m, τ_i), the ciphertext $\mathcal{E}_{T_{\forall}}(k)$ of session key k, and an attribute key $\mathbb{SK}_u$ for an attribute set $\mathbb{A}^u$. If $\mathbb{A}^u$ satisfies the corresponding tree T_i, this algorithm will decrypt the ciphertexts and return the corresponding composite $f'(m, \tau_i)$, otherwise, it will return *null*.

Table 1: Major notations

$m = e(g,g)^d \in \mathbb{G}_T$	shared (data) component in $\mathbb{G}_T$ transformed from original data $d \in \mathbb{Z}_q$
$\tau_i = e(g,g)^{\epsilon_i} \in \mathbb{G}_T$	i-th piece of identity-associated (noise) component transformed from original noise $\epsilon_i \in \mathbb{Z}_q$
$\mathcal{N} = \{\tau_i\}_{i \in [1,N]}$	set of noise
$f : \mathbb{Z}_q \times \mathbb{Z}_q \rightarrow \mathbb{Z}_q$	noise function before space transformation
$f' : \mathbb{G}_T \times \mathbb{G}_T \rightarrow \mathbb{G}_T$	noise function after space transformation
$d_{\epsilon_i} = f(d, \epsilon_i)$	i-th noisy data before transformation
$m_{\tau_i} = f'(m, \tau_i)$	i-th noisy data after transformation
$\mathcal{T} = \{T_i\}_{i \in [1,N]}$	set of access trees, where T_i determines who can access the noised data $f'(m, \tau_i)$
$\mathbb{A}^u$	user's attribute set
$\mathbb{SK}_u$	user's attribute key

The major notations used in our scheme are shown in Table 1. The correctness and security are defined as follows.

1. Correctness Definition

The correctness of the FAPC scheme is that the user is able to access the noisy data if and only if the user's attribute set satisfies the corresponding access tree. Formally:

[1]We refer the reader to [1] for details.

Definition 3.1 (FAPC Scheme Correctness). *For algorithms (Enc', Enc'', Dec), we have $Dec_{sk_i}(Enc'(m), Enc''(\tau_i)) = f'(m, \tau_i)$, iff $\mathbb{A}^u$ satisfies the corresponding tree T_i. Otherwise, the Dec algorithm outputs null.*

2. Security Definition

Our scheme should be secure in the following security model. Briefly, the adversary in the following game tries to distinguish an arbitrary but equal-length $\{m, \{\tau\}\}$ pair, with arbitrary size of the set $\{\tau\}$. Using Π to denote our FAPC scheme, the security definition is stated via the following experiment $\boldsymbol{Exp}_{\mathcal{A},\Pi}^{\mathcal{T}-cpa}$, which is a variation of indistinguishability under chosen plaintext attack (CPA):

Setup. The challenger runs the *Setup* algorithm and gives the public key pk to the adversary $\mathcal{A}$.

Phase 1. The adversary $\mathcal{A}$ repeatedly makes attribute key queries corresponding to attributes $\mathbb{A}^1, \mathbb{A}^2, \ldots, \mathbb{A}^{n_1}$. The challenger responds by running the attribute key generation algorithm to generate the corresponding attribute key $\mathbb{SK}_j$, for each $j \in [1, n_1]$.

Challenge. The adversary $\mathcal{A}$ submits two sets of plaintexts $\{m_0, \mathcal{N}_0 = \{\tau_i^0\}_{i \in [1,N]}\}$ and $\{m_1, \mathcal{N}_1 = \{\tau_i^1\}_{i \in [1,N]}\}$, where all counterparts are of equal length. In addition, the adversary $\mathcal{A}$ gives a challenge set of access trees $\mathcal{T}^* = \{T_i\}_{i \in [1,N]}$ and $T_{\forall}^*$ such that none of $\mathbb{A}^u$ from Phase 1 satisfies any tree. Here $T_{\forall}^*$ is established such that any authorized user's attribute set could satisfy $T_{\forall}^*$, if and only if it satisfies any tree T_i, $i \in [1, N]$. The challenger first flips a random coin $\mu \in \{0, 1\}$, and chooses a random session key k, then computes the corresponding ciphertexts $\mathcal{E}_{T_{\forall}^*}(k)$, $CT_{m_\mu}^*$ and $CT_{\mathcal{N}_\mu}^*$. The ciphertexts are given to the adversary $\mathcal{A}$.

Phase 2. Phase 1 is repeated under the condition that none of the sets of attributes $\mathbb{A}^{n_1+1}, \mathbb{A}^{n_1+2}, \ldots, \mathbb{A}^{n_2}$ satisfies any access tree in Challenge.

Guess. The adversary $\mathcal{A}$ outputs a guess μ' of μ.

We define the advantage of the adversary $\mathcal{A}$ in this game as $Adv_{\mathcal{A}} = Pr[\mu' = \mu] - 1/2$.

Definition 3.2 (Scheme Security). *We say that our FAPC scheme is secure if all probabilistic polynomial-time (PPT) $\mathcal{A}$ have at most a negligible advantage $Adv_{\mathcal{A}}$ in the above experiment.*

3.2 FAPC Scheme Details

The details of each algorithm are as follows:

3.2.1 Setup. This algorithm calls the setup algorithm of CP-ABE [1], to compute the master key mk and the public key pk as $(mk, pk) = CP\text{-}ABE.Setup(1^{\lambda})$.

3.2.2 Encrypt'. Given the public key pk, a plaintext m (the data component), a session key $k \in \mathbb{G}_T$ and a corresponding access tree $T_{\forall}$, this encryption algorithm computes the shared component's ciphertext by encrypting data m as CT_m under the session key k.

• The Shared Component (Data Component)

We choose two random numbers $\tilde{s}, \hat{s} \in \mathbb{Z}_q$, and compute :

$$C_m = m \cdot e(g,g)^{\alpha \cdot \tilde{s}}, \quad (1)$$

$$P_{up} = e(g,g)^{\alpha \cdot \hat{s}}. \quad (2)$$

Where $e(g,g)^{\alpha}$ is a part of the pk. The shared component is denoted as:

$$CT_m = \{C_m, E_k(P_{up})\}, \quad (3)$$

where $E_k(\cdot)$ is an efficient symmetric encryption algorithm using the session key k. P_{up} is an auxiliary parameter for later data update, which can be accessed by all the authorized users, but must be protected against the cloud. A straight-forward approach for securely sharing the P_{up} with authorized users is using the traditional CP-ABE scheme to encrypt it under the tree $T_\forall$, which however, introduces large overhead at the DO. To reduce the computation overhead, we adopt the hybrid encryption scheme by encrypting P_{up} with a symmetric-session key k. Then k is encrypted under tree $T_\forall$ by CP-ABE (Equ. (4)) . Since the session key k could be reused continuously for multi-round data updating, the computation overhead is remarkably reduced.

3.2.3 Encrypt". Given the public key pk, a set of plaintexts $\mathcal{N} = \{\tau_i\}_{i\in[1,N]}$ (the identity-associated components), a set of access trees $\mathcal{T} = \{T_i\}_{i\in[1,N]}$, this encryption algorithm computes the ciphertexts by: 1) encrypting the session key k as $\mathcal{E}_{T_\forall}(k)$ under the corresponding access tree $T_\forall$, where $T_\forall$ is constructed in such a way that any user, whose attribute set satisfies any tree $T_i \in \mathcal{T}$, also satisfies $T_\forall$; 2) encrypting the identity-associated components $\mathcal{N} = \{\tau_i\}_{i\in[1,N]}$ as $CT_{\mathcal{N}}$ under a set of access trees $\mathcal{T} = \{T_i\}_{i\in[1,N]}$, where each access tree T_i denotes which identity could access the i-th noisy version.

• Session Key Component

Encrypt the session key k as ciphertext $\mathcal{E}_{T_\forall}(k)$ under the tree $T_\forall$ by the traditional CP-ABE scheme, as shown in Equ. (4). We direct readers to the related work [1] for details.

$$\mathcal{E}_{T_\forall}(k) = \textit{CP-ABE.Encrypt}(T_\forall, k, pk) \tag{4}$$

• The Identity-Associated Component (Noise Component)

First each $\tau_i \in \mathcal{N}$ is randomized as A_{τ_i} by Equ. (5). Then A_{τ_i} is encrypted as $\mathcal{E}_{T_i}(A_{\tau_i})$ under the corresponding access tree T_i by the traditional CP-ABE scheme, as shown in Equ. (6).

$$A_{\tau_i} = (\tau_i)^{-1} \cdot e(g,g)^{-\alpha\cdot(\tilde{s}-\hat{s})} \tag{5}$$

$$\mathcal{E}_{T_i}(A_{\tau_i}) = \textit{CP-ABE.Encrypt}(T_i, A_{\tau_i}, pk) \tag{6}$$

The identity-associated component's ciphertext is denoted as:

$$CT_{\mathcal{N}} = \left\{\mathcal{E}_{T_i}(A_{\tau_i})\right\}_{i\in[1,N]} \tag{7}$$

For expression simplicity, we use $CT_m^{\tau_i}$ to denote each pair of shared component and identity-associated component as:

$$CT_m^{\tau_i} = \left\{CT_m, \mathcal{E}_{T_i}(A_{\tau_i})\right\}. \tag{8}$$

3.2.4 Update. This algorithm is to replace the shared component m with a newer version m', by updating the ciphertext $CT_m = \left\{C_m, E_k\left(P_{up}\right)\right\}$. It includes two steps:

Step 1. Update generation.

First, we choose a random number $\Delta s \in \mathbb{Z}_q$, and compute

$$\Delta P_{up} = e(g,g)^{\alpha\cdot\Delta s}, \Delta C_{m'} = \frac{m'}{m}\cdot e(g,g)^{\alpha\cdot\Delta s}. \tag{9}$$

Next, we compute P'_{up} as:

$$P'_{up} = P_{up}\cdot\Delta P_{up} = e(g,g)^{\alpha\cdot(\hat{s}+\Delta s)}. \tag{10}$$

Then we encrypt P'_{up} as $E_k(P'_{up})$ with the session key k. The update ciphertext is denoted as: $CT_{m\to m'} = \left\{\Delta C_{m'}, E_k(P'_{up})\right\}$.

Step 2. Update execution.

Given $CT_{m\to m'} = \left\{\Delta C_{m'}, E_k(P'_{up})\right\}$, we replace the original $E_k(P_{up})$ with new $E_k(P'_{up})$, and compute the new $C_{m'}$ as:

$$C_{m'} = C_m\cdot\Delta C_{m'} = m'\cdot e(g,g)^{\alpha\cdot(\tilde{s}+\Delta s)} \tag{11}$$

Finally, this update algorithm outputs the ciphertext $CT_{m'}$ for the newer shared component's plaintext m' as:

$$CT_{m'} = \left\{C_{m'}, E_k\left(P'_{up}\right)\right\}. \tag{12}$$

Note that the update algorithm only modifies the *shared component* without changing any identity-associated component or session key component, yielding constant data-update complexity regardless of the size of set $\mathcal{N}$.

3.2.5 KeyGenerate. This step of attribute key generation is the same as the key generation algorithm in CP-ABE [1]. In this algorithm, we compute the secret attribute key $\mathbb{SK}_u$ for the user with attribute set $\mathbb{A}^u$, as:

$$\mathbb{SK}_u = \textit{CP-ABE.KeyGenerate}(pk, mk, \mathbb{A}^u) \tag{13}$$

3.2.6 Decrypt. Given the public key pk, the ciphertext $\left\{\mathcal{E}_{T_\forall}(k), \mathcal{E}_{T_i}(A_{\tau_i})\right\}$, a secret attribute key $\mathbb{SK}_u$ associated with an attribute set $\mathbb{A}^u$, this decrypt algorithm will output the composite $f'(m,\tau_i)$ if $\mathbb{A}^u$ satisfies T_i. Otherwise it outputs *null*. This decrypt algorithm includes three steps, where the first and second steps are offline, and the third step is repeated with each update in running time:

Step 1. Decrypt the ciphertext $\mathcal{E}_{T_\forall}(k)$

We call *CP-ABE.Decrypt*$(pk, \mathcal{E}_{T_\forall}(k), \mathbb{SK}_u)$ algorithm to decrypt the session key k [1]. If the user's attribute set $\mathbb{A}^u$ doesn't satisfy the tree $T_\forall$, *CP-ABE.Decrypt* will return *null*. Otherwise, it returns the session key k, as:

$$\textit{CP-ABE.Decrypt}\left(pk, \mathcal{E}_{T_\forall}(k), \mathbb{SK}_u\right) = k. \tag{14}$$

Step 2. Decrypt the ciphertext $\mathcal{E}_{T_i}(A_{\tau_i})$

In this step, we call *CP-ABE.Decrypt*$(pk, \mathcal{E}_{T_i}(A_{\tau_i}), \mathbb{SK}_u)$ algorithm. If the user's attribute set $\mathbb{A}^u$ doesn't satisfy the tree T_i, the *CP-ABE.Decrypt* algorithm outputs *null*; otherwise, we have:

$$\textit{CP-ABE.Decrypt}\left(pk, \mathcal{E}_{T_i}(A_{\tau_i}), \mathbb{SK}_u\right) = A_{\tau_i}. \tag{15}$$

Note that the A_{τ_i} does not leak any information of the identity-associated component τ_i.

Step 3. Compute $f'(m,\tau_i)$

First, we obtain P_{up} by decrypting $E_k(P_{up})$ with the session key k obtained in **Step 1**, denoted as:

$$D_k(E_k(P_{up})) = P_{up}. \tag{16}$$

Then we obtain the composite by computing:

$$\begin{aligned}\frac{C_m\cdot A_{\tau_i}}{P_{up}} &= \frac{m\cdot e(g,g)^{\alpha\cdot\tilde{s}}\cdot(\tau_i)^{-1}\cdot e(g,g)^{-\alpha\cdot(\tilde{s}-\hat{s})}}{e(g,g)^{\alpha\cdot\hat{s}}}\\ &= \frac{m}{\tau_i} = f'(m,\tau_i).\end{aligned} \tag{17}$$

4 EVALUATION

The complexity and security analysis of our scheme are shown in Appendix. We also implement a system on cloud and use a user-cloud interactive scenario to experimentally evaluate our scheme. Our system includes an Amazon EC2 t2.large machine with 8GB of RAM, and multiple desktops with the Intel(R) CORE(TM) 2 Duo

CPU E8400 @ 3.00GHz and 8.00G RAM. We deploy Setup, Key Generate, Encrypt, Decrypt, and Update generation algorithms on the desktops. The Update execution runs on the cloud. All algorithms in our FAPC scheme are implemented using the Java Pairing-based Cryptography (JPBC) library [8].

4.1 Performance Analysis

We use several experiments to test the three offline algorithms, including *Setup*, *KeyGenerate*, *Encrypt*$''$, and the online *Encrypt*$'$, *Update*, *Decrypt* algorithms.

Setup. In this step, our scheme has constant time overhead of computing the master key *mk* and the public key *pk*, which is the same as the counterpart in the traditional CP-ABE scheme [1]. The average running time of 1000 independent tests is 0.212s.

Key Generate. This step's computational complexity is $O(W)$, where W is the number of user's attributes. To validate this, we select different numbers of user attributes, ranging from 10 to 100. According to our experiment results, the key generation time increases linearly from 0.75s to 6.4s.

Encrypt. In the *Encrypt*$''$ algorithm, for each piece of noise, the DO needs to compute the ciphertext. Theoretically, the overall computational complexity for encryption is $O(N \cdot X)$, where N is the number of noise and X is the average number of leaves in an access tree. To validate it, we choose different number of noise, ranging from 1 to 100, and test the encryption time under two different access trees with different leaf number: 16-leaf and 64-leaf. According to our result, the encryption time latency increases linearly from 24s to 210s in the 16-leaf case, and increases linearly from 90s to 840s in the 64-leaf case. As the comparison, the CP-ABE scheme outperforms with only a negligible advantage. We should note that most of the encryption overhead is one-time cost. Only the encryption of the data component in *Encrypt*$'$ is online, which is irrelevant to the number of leaf nodes as well as the number of noises, and costs 0.04s.

Update. Theoretically, the computational complexity of update generation is constant and irrelevant to the number of noises, because of the decoupled encryptions of the shared and identity-associated components. In practice, the update is generated by the data owner, and then sent to a cloud for publishing. To validate the complexity, we conduct a set of experiments using different number of noise. From Fig. 2 and 3, we can see that in our scheme, both the communication overhead and the computation time overhead for update generation are constant, thus much less than those by using the CP-ABE approach, which are linear with the number of noise. Specifically for 100 pieces of noise, the communication overhead is around 900KB when using CP-ABE. Meanwhile, the computational time overhead is around 210s when dealing with 100 noises, meaning that the maximum supported update frequency is one update per three-and-a-half minutes. As the comparison, FAPC has only 512Byte overhead per update. The computational time overhead is only 0.04s, yielding an update frequency of 25 per second. Meanwhile, our computational time overhead for update execution is also constant at about 0.1ms, but slightly heavier than that of the CP-ABE approach (which is zero since no computational operation is performed). Though a small amount of computation overhead is introduced at the cloud side, it is negligible because only one

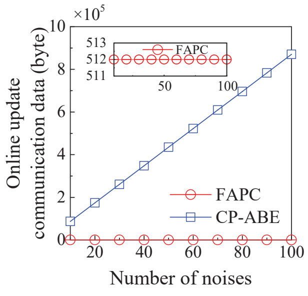

Figure 2: Update communication overhead, 16 leaves per access tree.

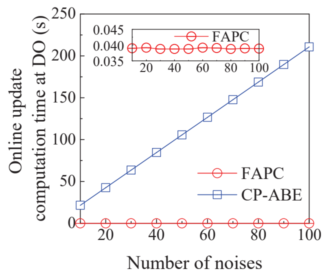

Figure 3: Update computation overhead at DO, 16 leaves per access tree.

multiplication operation in the group $\mathbb{G}_T$ is performed, which is computationally efficient. Note that the CP-ABE approach's overhead will increase to linear if considering the storage operations on cloud. Due to the constant communication and computation overhead, our scheme is scalable to support an even larger number of users simultaneously.

Decrypt. In this algorithm, three components are decrypted: 1) the session key component; 2) the noise components after randomized A_{τ_i}; 3) the composite of the shared and identity-associate components. For session key and noise components, we set different leaf number in an access tree ranging from 10 to 100. According to our results, the time latency increases linearly from 0.25s to 2.25s for the decryption of session key, and increases linearly from 0.5s to 4.5s for the decryption of noise components A_{τ_i}. For the decryption of the composite (Equ. (16) and Equ. (17)), we conduct 1000 independent tests and find the decryption of the composite costs 1.06s in average.

5 RELATED WORK

Fine-grained Access Control. Fine-grained data access control [10] is to flexibly associate each user's identity with his/her permitted access targets. As an advancement, attribute based encryption (ABE) defines a user's identity by his/her attribute set. Sahai and Waters first proposed this method [24] to exert access control over encrypted data. Goyal et al. later extended it to key-based attribute based encryption (KP-ABE) [15]. Bethencourt et al. extended it to the ciphertext-based attribute based encryption (CP-ABE) [1]. In KP-ABE, the user's secret key is associated with an access policy over attributes. The user can decrypt the ciphertext if and only if the attribute set of the ciphertext satisfies the access policy specified in his/her secret key. In CP-ABE, the ciphertext is associated with an access policy over attributes and the attribute set generates the user's secret/attribute key. The user can decrypt the ciphertext if and only if his/her attributes set satisfies the access policy specified in the ciphertext. The data owner is able to decide who have access to the encrypted data. Multiple ABE approaches have been proposed to implement secure data outsourcing [12], sharing [21], and keyword searching [16, 20]. However, these works only study how to control the access to different data. Directly applying them into our access-precision problems could generate large overhead for data owners. *Recently, researchers have been making efforts to combine ABE with homomorphic encryptions, which are most similar* *to ours [2, 6, 9, 14, 22, 23]*. However, none of their works consider the homomorphic operation between a shared component and identity-associated components, where only the composite of the two types of components is accessible by an authorized user. The enlarged ciphertext-size is also a problem in most of these works.

Homomorphic Encryptions. In our scheme the process of combining the shared and identity-associated components relies on the support of homomorphism. Homomorphic encryption has been extensively studied in privacy-preserving computing domain. Partially homomorphic encryption or fully homomorphic encryption allows arithmetic operations on the cloud with the operation results disclosed [7, 13]. Leveraging these cryptographic primitives, we could evaluate various utility functions on untrusted nodes. Topics include range search [27], keyword/string search [4, 11], genomic data search [5], image search [29], clustering [25], and image feature extraction [17], etc. Researchers even proposed schemes that support multiple types of functions simultaneously with inter-operable operators [28], with the assist from the data owner staying on-line. However, none of these approaches support fine-grained access control, i.e., how to differentiate data or its usage for different user identities is not addressed.

6 CONCLUSION

In this work we first present a cryptographic design that combines fine-grained access control with homomorphism. We design a decoupling technique that split the ciphertext into a dynamically shared component, and multiple identity-associated components. The user could combine the two components and reveal the corresponding composite in plaintext on the fly, iff his/her identity denoted by the attribute set satisfies the fine-grained access policy enacted by the data owner. Our algorithms enable data owners to precisely and efficiently control the form of their data revealed to the public in a fine-grained manner. We formally prove the security of our design with game-based approach. We test our algorithms on a commercial cloud platform and use extensive experiments to validate its performance in practice.

REFERENCES

[1] John Bethencourt, Amit Sahai, and Brent Waters. 2007. Ciphertext-policy attribute-based encryption. In *IEEE symposium on security and privacy (SP)*. IEEE, 321–334.

[2] Zvika Brakerski, David Cash, Rotem Tsabary, and Hoeteck Wee. 2016. Targeted homomorphic attribute-based encryption. In *Theory of Cryptography Conference*. Springer, 330–360.

[3] Carole Cadwalladr and Emma Graham-Harrison. 2018. Revealed: 50 million Facebook profiles harvested for Cambridge Analytica in major data breach. *The Guardian* 17 (2018).

[4] Ning Cao, Cong Wang, Ming Li, Kui Ren, and Wenjing Lou. 2014. Privacy-preserving multi-keyword ranked search over encrypted cloud data. *IEEE Transactions on parallel and distributed systems* 25, 1 (2014), 222–233.

[5] Ke Cheng, Yantian Hou, and Liangmin Wang. 2018. Secure Similar Sequence Query on Outsourced Genomic Data. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security (ASIACCS '18)*. ACM, New York, NY, USA, 237–251. https://doi.org/10.1145/3196494.3196535

[6] Michael Clear and Ciarán McGoldrick. 2015. Multi-identity and Multi-key Leveled FHE from Learning with Errors. In *Advances in Cryptology – CRYPTO 2015*, Rosario Gennaro and Matthew Robshaw (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 630–656.

[7] Ronald Cramer, Ivan Damgård, and Jesper Nielsen. 2001. Multiparty computation from threshold homomorphic encryption. *EUROCRYPT 2001* (2001), 280–300.

[8] Angelo De Caro and Vincenzo Iovino. 2011. JPBC: Java pairing based cryptography. In *IEEE Symposium on Computers and Communications (ISCC)*. IEEE, 850–855.

[9] W. Ding, Z. Yan, and R. Deng. 2017. Privacy-Preserving Data Processing with Flexible Access Control. *IEEE Transactions on Dependable and Secure Computing* (2017), 1–1. https://doi.org/10.1109/TDSC.2017.2786247

[10] David F. Ferraiolo, Ravi Sandhu, Serban Gavrila, D. Richard Kuhn, and Ramaswamy Chandramouli. 2001. Proposed NIST Standard for Role-based Access Control. *ACM Trans. Inf. Syst. Secur.* 4, 3 (Aug. 2001), 224–274.

[11] Benjamin Fuller, Mayank Varia, Arkady Yerukhimovich, Emily Shen, Ariel Hamlin, Vijay Gadepally, Richard Shay, John Darby Mitchell, and Robert K Cunningham. 2017. Sok: Cryptographically protected database search. In *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 172–191.

[12] William C Garrison, Adam Shull, Steven Myers, and Adam J Lee. 2016. On the practicality of cryptographically enforcing dynamic access control policies in the cloud. In *2016 IEEE Symposium on Security and Privacy (SP)*. IEEE, 819–838.

[13] Craig Gentry. 2009. *A fully homomorphic encryption scheme*. Ph.D. Dissertation. Stanford University.

[14] Craig Gentry, Amit Sahai, and Brent Waters. 2013. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *Advances in Cryptology–CRYPTO 2013*. Springer, 75–92.

[15] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. 2006. Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of the 13th ACM conference on Computer and communications security*. ACM, 89–98.

[16] Kai He, Jun Guo, Jian Weng, Jiasi Weng, Joseph K Liu, and Xun Yi. 2018. Attribute-Based Hybrid Boolean Keyword Search over Outsourced Encrypted Data. *IEEE Transactions on Dependable and Secure Computing* (2018).

[17] Linzhi Jiang, Chunxiang Xu, Xiaofang Wang, Bo Luo, and Huaqun Wang. 2017. Secure outsourcing SIFT: Efficient and Privacy-preserving Image Feature Extraction in the Encrypted Domain. *IEEE Transactions on Dependable and Secure Computing* (2017).

[18] Taeho Jung, Xiang-Yang Li, Zhiguo Wan, and Meng Wan. 2015. Control cloud data access privilege and anonymity with fully anonymous attribute-based encryption. *IEEE Transactions on Information Forensics and Security* 10, 1 (2015), 190–199.

[19] Jonathan Katz and Yehuda Lindell. 2014. *Introduction to modern cryptography, second edition*. CRC press.

[20] H. Li, Y. Yang, T. H. Luan, X. Liang, L. Zhou, and X. S. Shen. 2016. Enabling Fine-Grained Multi-Keyword Search Supporting Classified Sub-Dictionaries over Encrypted Cloud Data. *IEEE Transactions on Dependable and Secure Computing* 13, 3 (May 2016), 312–325. https://doi.org/10.1109/TDSC.2015.2406704

[21] Ming Li, Shucheng Yu, Yao Zheng, Kui Ren, and Wenjing Lou. 2013. Scalable and secure sharing of personal health records in cloud computing using attribute-based encryption. *IEEE Transactions on Parallel and Distributed Systems* 24, 1 (2013), 131–143.

[22] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. 2012. On-the-fly Multiparty Computation on the Cloud via Multikey Fully Homomorphic Encryption. In *Proceedings of the Forty-fourth Annual ACM Symposium on Theory of Computing (STOC '12)*. ACM, New York, NY, USA, 1219–1234.

[23] Pratyay Mukherjee and Daniel Wichs. 2016. Two Round Multiparty Computation via Multi-key FHE. In *Proceedings of the 35th Annual International Conference on Advances in Cryptology — EUROCRYPT 2016 - Volume 9666*. Springer-Verlag New York, Inc., New York, NY, USA, 735–763.

[24] Amit Sahai and Brent Waters. 2005. Fuzzy identity-based encryption. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 457–473.

[25] Jaideep Vaidya and Chris Clifton. 2003. Privacy-preserving k-means clustering over vertically partitioned data. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 206–215.

[26] Zhiguo Wan, Jun'e Liu, and Robert H Deng. 2012. HASBE: A Hierarchical Attribute-Based Solution for Flexible and Scalable Access Control in Cloud Computing. *IEEE Transactions on Information Forensics and Security* 7, 2 (April 2012), 743–754. https://doi.org/10.1109/TIFS.2011.2172209

[27] B. Wang, M. Li, and H. Wang. 2016. Geometric Range Search on Encrypted Spatial Data. *IEEE Transactions on Information Forensics and Security* 11, 4 (2016), 704–719. https://doi.org/10.1109/TIFS.2015.2506145

[28] Wai Kit Wong, Ben Kao, David Wai Lok Cheung, Rongbin Li, and Siuming Yiu. 2014. Secure query processing with data interoperability in a cloud database environment. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*. ACM, 1395–1406.

[29] Z. Xia, X. Wang, L. Zhang, Z. Qin, X. Sun, and K. Ren. 2016. A Privacy-Preserving and Copy-Deterrence Content-Based Image Retrieval Scheme in Cloud Computing. *IEEE Transactions on Information Forensics and Security* 11, 11 (Nov 2016), 2594–2608. https://doi.org/10.1109/TIFS.2016.2590944

[30] Shucheng Yu, Cong Wang, Kui Ren, and Wenjing Lou. 2010. Attribute based data sharing with attribute revocation. In *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security*. ACM, 261–270.

A COMPLEXITY ANALYSIS

We use N to denote the number of noise, W as the average number of one user's attributes, X as the average number of leaves in an access tree.

Setup algorithm computes a master key mk and a system public key pk, with computational complexity $O(1)$. The *Encrypt′* algorithm yields constant overhead by encrypting the C_m and P_{up}. In *Encrypt′′* algorithm, the DO chooses a polynomial p_x at each node x in each tree $T_{i, i\in[1,N]}$ and $T_\forall$, which is the dominating overhead with complexity $O(N \cdot X)$. Nevertheless, this overhead is one-time for each session. Note that the tree $T_\forall$ is not necessarily more complicated in structure. Optimizing the tree structure could further reduce the complexity, but is not the focus of this paper. In *Update* algorithm, the DO just needs to compute and send two ciphertexts $\{\Delta C_{m'}, E_k(P'_{up})\}$ to the cloud. The cloud needs one computation to generate $C_{m'}$. *Therefore the update complexity at both the DO and cloud sides, as well as the communication overhead towards the cloud are* $O(1)$. *KeyGenerate* algorithm computes attribute key from each user attribute in $\mathbb{A}^u$. Therefore the computational complexity is $O(W)$. In *Decrypt* algorithm, the computational overhead is linear with the number of leaf nodes in each tree. Therefore, the complexity is $O(X)$. The complexity analysis is summarized in Table 2.

As the comparison, we compute the complexity by using the traditional CP-ABE scheme [1]. In this case, the DO encrypts each piece of noisy data under the corresponding access tree. When updating the data, the DO needs to encrypt all the updated noisy data before uploading to the cloud for storage. The cloud directly replaces the obsolete data with the new one. Therefore the update complexity by using the traditional CP-ABE approach is $O(N \cdot X)$ at both the DO and the cloud sides.

Table 2: The computational complexity of the FAPC algorithms

Algorithms	Complexity
FAPC.*Setup*	$O(1)$
FAPC.*Encrypt′*	$O(1)$
FAPC.*Encrypt′′*	$O(N \cdot X)$
FAPC.*Update*	$O(1)$
FAPC.*KeyGenerate*	$O(W)$
FAPC.*Decrypt*	$O(X)$

B SECURITY ANALYSIS

In this section we first formally define the security requirements. Then we will prove that our FAPC scheme is secure through a reduction chain from a publicly recognized secure scheme.

B.1 Security Proof

1. Correctness

If the user's attribute set $\mathbb{A}^u$ satisfies certain access tree $T_i, i \in [1, N]$, he/she will first successfully decrypt the session key k encrypted with the traditional CP-ABE scheme, since the access tree $T_\forall$ is satisfied by all authorized users. Then the user is able to obtain P_{up} and A_{τ_i}. Finally by Equ. (17), the user will get the corresponding noisy data $f'(m, \tau_i), i \in [1, N]$.

Otherwise, the *CP-ABE.Decrypt* $\left(pk, \mathcal{E}_{T_i}(A_{\tau_i}), \mathbb{SK}_u\right)$ algorithm will output *null*, even though the user may decrypt k (other authorized users that satisfy other trees $T_j, j \neq i$). Therefore our scheme is correct since the user will obtain the authorized noisy data $f'(m, \tau_i), i \in [1, N]$ if and only if his/her attribute set $\mathbb{A}^u$ satisfies the corresponding access tree T_i.

2. Scheme Security

Briefly, the proof logic is as follows: we denote P_Π as the problem to break the security of our scheme (defined in Sec. 3.1), P_Ω and $P_{\Omega'}$ as the problem to break the security of the traditional CP-ABE scheme and the problem to break the security of the traditional CP-ABE scheme with multiple encryptions, respectively (in our paper, multiple encryptions are performed by using the same public key to encrypt multiple messages under different access trees). We could prove the security of our FAPC scheme by first reducing $P_{\Omega'}$ to P_Π, then prove the hardness of $P_{\Omega'}$ based on the hardness of P_Ω. Given that P_Ω has been proven to be computationally hard [1, 26], we can prove the hardness of our problem P_Π through the reduction chain.

Reduction from $P_{\Omega'}$ to P_Π: Here we construct the reduction from the $P_{\Omega'}$ problem of the traditional CP-ABE scheme to the P_Π problem of our FAPC scheme. The reduction process is in the proof of Theorem 1.

THEOREM 1. *If a PPT adversary has a non-negligible advantage in our security model (Sec. 3.1), then there exists another PPT adversary that can break the indistinguishable multiple encryptions of the traditional CP-ABE scheme [1] with a non-negligible advantage.*

The proof is in Appendix C.1.

Hardness of $P_{\Omega'}$: The remaining work is to prove the hardness of $P_{\Omega'}$ based on P_Ω. Based on the security of the traditional CP-ABE scheme, we can prove that no PPT adversary could break the security of the traditional CP-ABE scheme with multiple encryptions, i.e., $P_{\Omega'}$ is hard if P_Ω is hard. The corresponding theorem is as follows.

THEOREM 2. *If the traditional CP-ABE scheme [1] is CPA-secure, then it is also CPA-secure for CP-ABE scheme with multiple encryptions.*

We refer readers to article [19], Theorem 11.6 for the general form. Given Theorems 1 and 2, we can conclude that our FAPC scheme is secure under the security model defined in Sec. 3.1.

3. Collusion Resistance

For the collusion attack launched by multiple unauthorized users, the traditional CP-ABE scheme has been proven to be collusion-resistant [18, 30] by introducing random numbers to tie together different components in an attribute key, making the users not able to generate a new attribute key through collusion. Thus they fail in decrypting the corresponding ciphertext. Since our key generation phase is the same as the counterpart used in CP-ABE, our scheme is collusion-resistant.

C SECURITY PROOF

C.1 Proof of Theorem 1

Proof. Suppose a PPT adversary $\mathcal{A}$'s advantage in our security model defined in Sec. 3.1 is $Adv_{\mathcal{A}}$. Then we will show that a PPT adversary $\mathcal{A}'$ can be constructed based on $\mathcal{A}$, such that $\mathcal{A}'$ is able to break the indistinguishable multiple encryptions of the traditional CP-ABE scheme with the same advantage $Adv_{\mathcal{A}}$ in the following security game.

Setup. $\mathcal{A}'$ obtains the public key $pk = \left\langle \mathbb{G}_0, g, h = g^{\beta}, e(g,g)^{\alpha} \right\rangle$ of the traditional CP-ABE scheme, and sends pk to the adversary $\mathcal{A}$. Note that the corresponding master key $mk = \langle \beta, g^{\alpha} \rangle$ is only known to the traditional CP-ABE challenger C.

Phase 1. $\mathcal{A}$ submits multiple attribute key queries by sending sets of attributes $\mathbb{A}^1, \mathbb{A}^2, \ldots, \mathbb{A}^{n_1}$ to $\mathcal{A}'$. To generate the corresponding attribute keys, $\mathcal{A}'$ makes an attribute key query to the challenger C for each attribute set $\mathbb{A}^j$, $j \in [1, n_1]$. Then $\mathcal{A}'$ is given the corresponding keys $\mathbb{SK}_j^* = \left\{ D = g^{(r+\alpha)/\beta}, \forall l \in \mathbb{A}^j : D_l = g^r \cdot H(l)^{r_l}, D_l' = g^{r_l} \right\}$, $j \in [1, n_1]$. All these attribute keys are sent to the adversary $\mathcal{A}$.

Challenge. The adversary $\mathcal{A}$ gives $\mathcal{A}'$ a challenge access tree set $\left\langle \mathcal{T}^* = \{T_i\}_{i \in [1,N]}, T_{\forall}^* \right\rangle$, and two pieces of plaintext $\{m_0, \mathcal{N}_0 = \{\tau_i^0\}_{i \in [1,N]}\}$ and $\left\{ m_1, \mathcal{N}_1 = \{\tau_i^1\}_{i \in [1,N]} \right\}$, where all counterparts are of equal length, i.e., $|m_0| = |m_1|$, and $|\tau_i^0| = |\tau_i^1|, \forall i \in [1, N]$. Note that it is required that none of the sets $\mathbb{A}^1, \mathbb{A}^2, \ldots, \mathbb{A}^{n_1}$ from Phase 1 satisfies any tree in $\left\langle \mathcal{T}^* = \{T_i\}_{i \in [1,N]}, T_{\forall}^* \right\rangle$. $\mathcal{A}'$ is expected to return the challenge ciphertext back to the adversary $\mathcal{A}$ after running the following two steps:

Step 1. $\mathcal{A}'$ randomly chooses a secret session key k, and encrypts k under the corresponding access tree $T_{\forall}^*$. The corresponding ciphertext is shown in Equ. (18).

$$\mathcal{E}_{T_{\forall}^*}(k) = CP - ABE.encrypt(T_{\forall}^*, k, pk) \tag{18}$$

Step 2. $\mathcal{A}'$ randomly chooses $\hat{s}^* \in \mathbb{Z}_q$, then transforms the two lists of plaintexts as $\left\{ m_0, \frac{m_0}{\tau_1^0} \cdot e(g,g)^{\alpha \cdot \hat{s}^*}, \ldots, \frac{m_0}{\tau_N^0} \cdot e(g,g)^{\alpha \cdot \hat{s}^*} \right\}$ and $\left\{ m_1, \frac{m_1}{\tau_1^1} \cdot e(g,g)^{\alpha \cdot \hat{s}^*}, \ldots, \frac{m_1}{\tau_N^1} \cdot e(g,g)^{\alpha \cdot \hat{s}^*} \right\}$. Then $\mathcal{A}'$ sends these two lists of messages and the corresponding tree set $\{T_x, T_1, \ldots, T_N\}$ to the challenger C, where $T_x \in \mathcal{T}^*$ is an arbitrarily selected access tree from $\mathcal{T}^*$. The challenger C first randomly flips a binary coin μ, and uses CP-ABE to encrypt m_μ under T_x as $\mathcal{E}_{T_x}(m_\mu)$:

$$\begin{aligned} &\mathcal{E}_{T_x}(m_\mu) \\ =&CP - ABE.encrypt(T_x, m_\mu, pk) \\ =&\left\{ \begin{array}{l} T_x, C_{m_\mu} = m_\mu \cdot e(g,g)^{\alpha \cdot \tilde{s}^*}, \hat{C} = h^{\tilde{s}^*}, \\ \left\{ \left\langle C_x^y = g^{p_y(0)}, C_x'^y = H(att(y))^{p_y(0)} \right\rangle \right\}_{y \in Y_x} \end{array} \right\}. \end{aligned} \tag{19}$$

Then similarly for each $i \in [1, N]$, C encrypts $\frac{m_\mu}{\tau_i^\mu} \cdot e(g,g)^{\alpha \cdot \hat{s}^*}$ under the tree T_i as $\mathcal{E}_{T_i}\left(\frac{m_\mu}{\tau_i^\mu} \cdot e(g,g)^{\alpha \cdot \hat{s}^*} \right)$:

$$\begin{aligned} &\mathcal{E}_{T_i}\left(\frac{m_\mu}{\tau_i^\mu} \cdot e(g,g)^{\alpha \cdot \hat{s}^*} \right) \\ =&CP - ABE.encrypt(T_i, \frac{m_\mu}{\tau_i^\mu} \cdot e(g,g)^{\alpha \cdot \hat{s}^*}, pk) \\ =&\left\{ \begin{array}{l} T_i, C_i' = \frac{m_\mu}{\tau_i^\mu} \cdot e(g,g)^{\alpha \cdot (\hat{s}^* + \dot{s}_i^*)}, \hat{C}_i = h^{\dot{s}_i^*}, \\ \left\{ \left\langle C_i^y = g^{p_y(0)}, C_i'^y = H(att(y))^{p_y(0)} \right\rangle \right\}_{y \in Y_i} \end{array} \right\}. \end{aligned} \tag{20}$$

The ciphertexts are given to the adversary $\mathcal{A}'$. Then for all $y \in Y_i$, $i \in [1, N]$, $\mathcal{A}'$ computes $\tilde{C}_i^*$ as

$$\tilde{C}_i^* = \frac{C_i'}{C_{m_\mu}} = (\tau_i^\mu)^{-1} \cdot e(g,g)^{-\alpha(\tilde{s}^* - \hat{s}^* - \dot{s}_i^*)}, \tag{21}$$

and sets $C_{m_\mu}^* = C_{m_\mu}$, $\hat{C}_i^* = \hat{C}_i$, $C_i^{y*} = C_i^y$, $C_i'^{y*} = C_i'^y$, $\mathcal{E}_{T_i}(A_{\tau_i^*}) = \left\langle T_i, \tilde{C}_i^*, \hat{C}_i^*, \left\{ \left\langle C_i^{y*}, C_i'^{y*} \right\rangle \right\}_{y \in Y_i} \right\rangle$.

In addition, $\mathcal{A}'$ sets $P_{up}^* = e(g,g)^{\alpha \cdot \hat{s}^*}$, and encrypts P_{up}^* as $E_k^*(P_{up}^*)$ using a secure symmetric encryption algorithm, such as AES, where k is the secret session key from **Step 1**. Till now $\mathcal{A}'$ obtains the ciphertexts as shown in Equ. (22):

$$CT_{m_\mu} = \left\{ C_{m_\mu}^*, E_k^*\left(P_{up}^*\right) \right\}, CT_{\mathcal{N}_\mu *} = \left\{ \mathcal{E}_{T_i}(A_{\tau_i^*}) \right\}_{i \in [1,N]} \tag{22}$$

Finally, the challenge ciphertext $\mathcal{E}_{T_{\forall}^*}(k)$, CT_{m_μ} and $CT_{\mathcal{N}_\mu *}$ are returned to $\mathcal{A}$.

Phase 2. Phase 1 is repeated under the condition that no set of attributes can satisfy any access tree in Challenge.

Guess. First, the adversary $\mathcal{A}$ outputs a guess $\mu' \in \{0, 1\}$ of μ to the adversary $\mathcal{A}'$. Then, the adversary $\mathcal{A}'$ outputs μ' to conclude its own game.

According to the security model defined in Sec. 3.1, the advantage of the adversary $\mathcal{A}'$ breaking the security of the traditional CP-ABE scheme with multiple encryptions, is:

$$Adv_{\mathcal{A}'} = Pr[\mu' = \mu] - 1/2 = Adv_{\mathcal{A}}. \tag{23}$$

Equ. (23) shows that if $Adv_{\mathcal{A}}$ is non-negligible in our security model, then the adversary $\mathcal{A}'$ also has non-negligible advantage $Adv_{\mathcal{A}'} = Adv_{\mathcal{A}}$ to break the security of the traditional CP-ABE scheme with multiple encryptions. Theorem 1 is proven. □