PPDF: A Privacy-Preserving Cloud-Based Data Distribution System With Filtering

Yudi Zhang, Willy Susilo, *Fellow, IEEE*, Fuchun Guo, and Guomin Yang, *Senior Member, IEEE*

***Abstract*—Cloud computing has emerged as a popular choice for distributing data among both individuals and companies. Ciphertext-policy attribute-based encryption (CP-ABE) has been extensively used to provide data security and enable fine-grained access control. With this encryption technique, only users whose attributes satisfy the access policy can access the plaintext. In order to mitigate the computational overhead on users, particularly on lightweight devices, partial decryption has been introduced, where the cloud assists in performing the decryption computations without revealing sensitive information. However, in this process, the cloud obtains the user's attributes, thus infringing on the user's privacy. To address this issue, this article proposes a privacy-preserving cloud-based data distribution system with filtering (PPDF) to enable partial decryption without revealing the user's attributes. The proposed system also employs an edge server to assist the user in filtering out invalid ciphertexts, i.e., ciphertexts where the user's attributes do not satisfy the access policy, and transmit only the valid partially decrypted ciphertexts to the data receiver. Consequently, the proposed PPDF scheme achieves constant decryption cost for the data receiver. We provide a security proof and a performance evaluation of the proposed scheme, which confirms its effectiveness and practicality in various real-world applications.**

***Index Terms*—Privacy-preserving, date sharing, date filtering, outsource decryption, attributes test.**

I. Introduction

THE exponential growth of data has prompted a surge in the adoption of cloud storage in recent years. Cloud storage offers the benefits of cost efficiency, scalability with multiple users, and disaster recovery, which has been leveraged by IT giant companies like Xerox, Netflix, and Best Buy to accelerate their business operations. However, the security of cloud-stored data also needs to be carefully considered to ensure data confidentiality, integrity, and availability. According to the Verizon Data Breach Investigations Report (DBIR), the number of cloud security breaches in 2021 surpassed on-premises breaches for the first time, highlighting the need for robust security measures [1]. To securely share data with the intended recipient, data owners can encrypt the plaintext before uploading it to the cloud server, ensuring that the cloud server cannot access the data without the secret key. However, traditional public key encryption approaches do not provide flexible data sharing, as data owners would need to encrypt the plaintext under different public keys, resulting in redundant ciphertexts. Attribute-based encryption (ABE) [2], [3], [4], [5], [6] has been proposed to ensure end-to-end data security in cloud storage, enabling data owners to specify access policies during encryption and realize flexible access control.

Despite the potential benefits of ABE in cloud storage, many secure data sharing systems using ABE suffer from inefficiency, particularly due to the high computational cost of the decryption algorithm. For some schemes, the computational cost grows linearly with the complexity of the access policy, which poses significant challenges for practical implementation. To address this issue, several cloud computing-based schemes have been proposed in recent years [7], [8], [9], [10]. Fig. 1(a) depicts a typical cloud computing-based data subscription architecture, where the data owner encrypts the data on different access policies and uploads the ciphertexts to the cloud server. The authority issues the secret key to the data receiver based on his/her attributes, while also generating a transformation key and sending it to the cloud. Prior to pushing the subscribed data to the data user, the cloud server verifies whether the attributes in the transformation key satisfy the policy in the ciphertext. If they do not satisfy, the cloud server outputs nothing; otherwise, the cloud server executes the partial decryption algorithm and outputs the partially decrypted ciphertext to the data receiver. In addition, some cloud computing-based schemes provide mechanisms for the data receiver to verify whether the cloud server runs the partial decryption algorithm honestly or outputs a junk value.

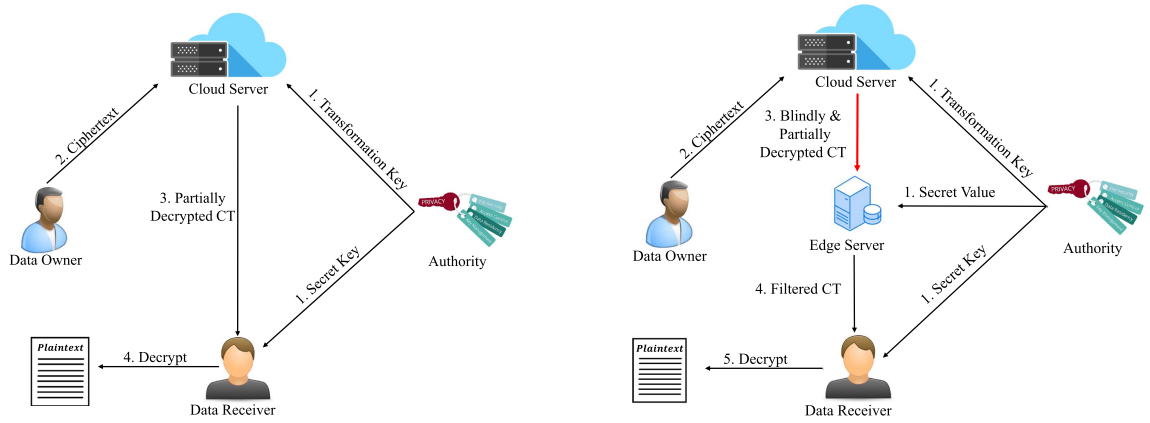
While cloud computing has made data sharing more convenient for users, protecting their privacy poses ongoing challenges. Specifically, when the cloud is utilized for partial decryption of ciphertext, the user's attributes are directly exposed to the cloud server. These attributes contain sensitive information and can potentially reveal the user's true identity. One approach to preserve user privacy is to use a transformation key with hidden attributes. The cloud server performs partial decryption without attribute matching and sends the partially decrypted ciphertext to the user. However, this method incurs significant computing and storage overhead for the user to determine whether the received

Manuscript received 27 February 2023; revised 8 June 2023; accepted 15 July 2023. Date of publication 20 July 2023; date of current version 13 December 2023. This work was supported in part by Australian Research Council Discovery Project under Grant DP200100144, in part by ARC Future Fellowship under Grant FT220100046, and in part by Australian Laureate Fellowship under grant FL230100033. Recommended for acceptance by Y. Zhang. *(Corresponding author: Yudi Zhang.)*

Yudi Zhang, Willy Susilo, and Fuchun Guo are with the Institute of Cybersecurity and Cryptology, School of Computing and Information Technology, University of Wollongong, Wollongong, NSW 2522, Australia (e-mail: zhangyudi007@gmail.com; wsusilo@uow.edu.au; fuchun@uow.edu.au).

Guomin Yang is with the School of Computing and Information Systems, Singapore Management University, Singapore 188065 (e-mail: gyang@uow.edu.au).

Digital Object Identifier 10.1109/TSC.2023.3297175



(a) The architecture of computing based data subscribing

(b) The architecture of our proposed system

Fig. 1. The different architectures between general system and our proposed system.

data belongs to them, which may not be feasible for lightweight devices with limited resources. Alternatively, certain attribute-based encryption schemes with hidden policy properties have been proposed, such as those discussed in [11], [12], [13], which offer privacy protection. Nevertheless, when delegating partial decryption power to the cloud, the cloud server still needs to transmit all partially decrypted ciphertexts to the user, who then needs to identify which one belongs to them.

In summary, the computational complexity of performing several bilinear operations on a device with limited resources is inefficient. Therefore, our objective is to leverage cloud servers to reduce the computational burden on users, while simultaneously utilizing edge computing [14], [15], [16] to protect user privacy. As such, the objective of this paper is to not only safeguard user privacy, but also optimize user computational efficiency.

A. Application Case

Consider the scenario of firmware distribution within an Internet of Things (IoT) network, comprising various device types, each with distinct attributes. The encrypted firmware is uploaded to a cloud server. It's presupposed that both the cloud server and the edge server are semi-honest, meaning they adhere to the protocol but may attempt to derive as much information as possible from received messages. Moreover, it is assured that no collusion will occur between them. Upon the IoT device requesting firmware, it first transmits this request to the edge server. The edge server then communicates this request to the cloud server to retrieve the ciphertext. The cloud server uses the transformation key to execute the partial decryption algorithm, then, returns the results to the edge server. The edge server tests whether the end device's attributes satisfy the access policy in the ciphertext. If the test passes (i.e., the content belongs to the specified device), the edge server pushes the ciphertext to the end device. By employing the proposed PPDF scheme, it dramatically reduces the computation overhead of the end device while hiding the content and recipients of the firmware.

B. Our Contributions

To achieve our goal, we propose a Privacy-Preserving Cloud-Based Data Distribution System with Filtering (PPDF) that help users partially decrypt ciphertexts and accurately push valid ciphertexts to the users. Our proposed scheme, illustrated in Fig. 1(b), utilizes the cloud server to perform partial decryption, while concealing the user's attributes (i.e., blindly and partially decryption). Subsequently, the edge server filters the ciphertexts for the data receiver, returning only the partially decrypted ciphertext that satisfies the access policy of the user's attributes. Invalid ciphertexts are filtered out. We implemented our proposed scheme on a personal computer and a Raspberry Pi 3b, demonstrating its security, flexibility, and practicality in real-world applications. The main contributions of our work are as follows:

1) We designed a privacy-preserving cloud-based data distribution system with filtering, which allows the data owner to upload encrypted messages without revealing the access policy to the cloud server. The cloud server is only able to obtain the index of the attributes, which are blinded. Finally, the edge server pushes the filtered ciphertexts to the data user.
2) Our scheme supports the AND gates access structures. We prove that our scheme is selective-CPA (chosen-plaintext attacks) secure under the n-BDHEasym assumption (n denotes the total number of attributes in the universe).
3) We implemented our scheme using the RELIC cryptographic meta-toolkit [17] on a PC and a Raspberry Pi 3b. Compared with previous work, our scheme improves the computational efficiency on the user side. For the data receiver, the computational cost does not increase with the growth of the number of attributes.

C. Organization of This Paper

The rest of this paper is organized as follows. In Section II, we show the related work about attribute-based encryption and

outsourcing decryption schemes briefly. In Section III, we describe the notations and access policy. In Section IV, we show the PPDF architecture, the threat model and the protocol workflow. In Section V, we give the algorithm definitions and the system design in detail. In Section VI, we prove the proposed system can preserve privacy and is CPA (chosen-plaintext attacks) secure. We demonstrate our evaluation results in Section VII, before concluding this paper in the last section.

II. Related Work

A. Attribute-Based Encryption

After Sahai and Waters [2] introduced the fuzzy identity-based encryption (IBE) as the first ABE scheme, many ABE schemes have been proposed in the past decade. There are two categories of ABE: KP-ABE (key-policy ABE) [18], [19] and CP-ABE (ciphertext-policy ABE) [5], [6], [20], [21]. In CP-ABE, user's secret key is associated with an attribute list, and the ciphertext specifies an access policy. On the contrary, in KP-ABE, the access policy is encoded into a user's secret key, and the ciphertext is generated with a related attribute list. Due to the data owner can control the access policy determination in CP-ABE, therefore, CP-ABE has received wider attention, especially in cloud-based data sharing.

Bethencourt et al. [3] introduced a first CP-ABE scheme which supports tree-based access policy, they use a two-level random masking methodology to construct a private key randomization technique. Cheung and Newport [4] proposed CP-ABE scheme in which AND gate policies which access structures are AND gates on positive and negative attributes. They proved their scheme is CPA (chosen-plaintext attacks) secure under DBDH assumption. However, these schemes either support a limited access structure or the security proof should in the generic group model.

To solve this problem, Goyal et al. [22] introduced the first construction of CP-ABE scheme, the security proof is based on a number theoretic assumption, and it supports advanced access structures. Following this work, Liang et al. [23] designed a new bounded CP-ABE scheme which can reduce the computation cost during the encryption and decryption phase. Lewko et al. [24] proposed a fully secure ABE scheme, which is proved secure from three static assumptions.

Nevertheless, the ciphertext in most previous schemes are linear increasing of the numbers of access policy. Herranz et al. [25] proposed an ABE scheme with constant size ciphertexts which supports threshold policies. Susilo et al. [26] introduced a more efficient ABE scheme with constant size ciphertexts. In addition, Waters [5] introduced a new CP-ABE scheme which can support LSSS access policies. Ding et al. [27] presented an extended framework of privacy-preserving computation with flexible access control which can adapt to different application scenarios with sound scalability. While, the computational cost of decryption algorithm is still heavy to the data receiver.

B. ABE With Outsourcing

To reduce the computational cost for data receiver when executing the decryption algorithm, Green et al. [7] considered outsourcing the billiard computation to a proxy, which can greatly reduce the computational overhead of the data receiver. In their scheme, a transformation key is generated for the proxy, and it will not reveal any information about the secret key. Also, the proxy can learn nothing about the plaintext. However, it cannot guarantee whether the proxy runs the algorithm correctly.

Lai et al. [8] designed a ABE scheme with verifiable outsourced decryption, the main technique is to add an extra encryption instance in the ciphertext. Lin et al. [28] proposed a more efficient ABE scheme with verifiable outsourced decryption, they used symmetric-key encryption and commitment to construct their scheme. Mao et al. [29] proposed generic constructions of CPA-secure and RCCA-secure ABE systems with verifiable outsourced decryption. Xu et al. [30] introduced a CP-ABE outsourced decryption scheme with public verification. However, Li et al. [31] showed that the efficient of [30] is low, and it is not feasible.

Besides, the above schemes will reveal data receiver's attributes to the proxy, due to the transformation key contains user's attributes, the policy in the ciphertext can also expose some information about user's attributes. In addition, there are also some schemes [10], [32] that outsource the user's computation to edge servers.

However, the existing literature lacks a feasible solution that simultaneously minimizes users' computation cost while preserving their privacy. To address this gap, our paper introduces a novel scheme called PPDF. In PPDF, we leverage the cloud for storing and performing partial decryption of ciphertext, thereby minimizing the computation cost for users. Additionally, we employ an edge server to safeguard users' privacy and filter out invalid ciphertext. By combining these elements, PPDF offers an effective solution that addresses both the computational and privacy concerns in attribute-based encryption systems.

III. Preliminaries

Let S denote a random distribution or set, and $a \xleftarrow{r} S$ denote that a is selected from S randomly. If $\forall c > 0$, $\exists \lambda'$, $\forall \lambda \geq \lambda'$, $\mu(\lambda) \leq \frac{1}{\lambda^c}$, the function $\mu(\lambda)$ is called negligible. P.P.T denotes a probabilistic-polynomial time algorithm, and PKG (Private Key Generator) denotes a trusted private key generator. H is a secure hash function, such that $H : \{0,1\}^* \rightarrow \mathbb{Z}_p$.

A. Access Policy

An access policy W, namely a ciphertext policy in CP-ABE, is a rule that returns either 0 or 1 given an attributes set L. We say that L satisfies W if and only if W answers 1 on L. Notation $L \models W$ denotes that L satisfies W, and the case of L does not satisfy W is denoted by $L \not\models W$. In this paper, an attributes set $L = \{v_{1,l_1}, v_{2,l_2}, \ldots, v_{i,l_i}, \ldots, v_{n,l_n}\}$ and an access policy $W = \{W_1, W_2, \ldots, W_i, \ldots, W_n\} = \wedge_{i \in I_W} W_i$, where I_W is a subscript index set and $I_W = \{1 \leq i \leq n, W_i \neq *\}$ (wildcard $*$ in W plays the role of "don't care" value), we say $L \models W$ if $L_i = W_i$ or $W_i = *$ for all $1 \leq i \leq n$, otherwise, $L \not\models W$.

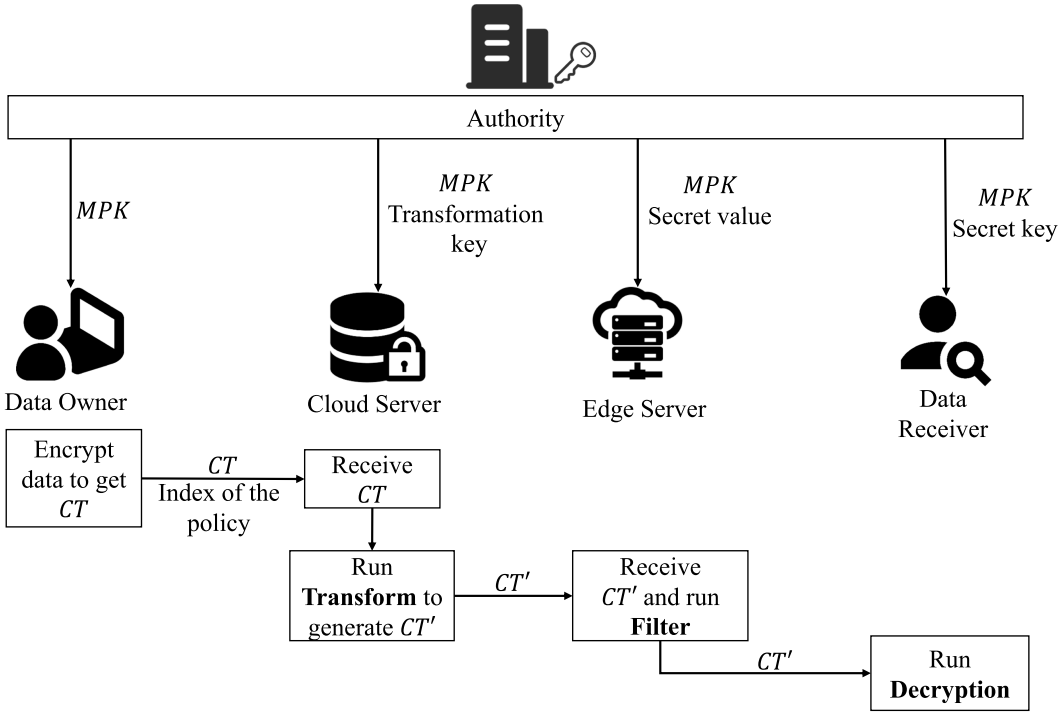


Fig. 2. PPDF architecture.

IV. PPDF in a Nutshell

The proposed PPDF system aims to minimize users' computation overhead without leaking any user's privacy, including the attributes. It allows a cloud to partially decrypt the ciphertext, but learns nothing about the plaintext, user's secret key, or the attributes. Then, the cloud server sends all the partially decrypted ciphertexts to the edge server. When the user's attributes satisfy the ciphertext policy, the edge server will send the ciphertext to the user. Finally, the user does an ElGamal-like decryption to obtain the plaintext.

A. System Architecture

As shown in Fig. 2, the proposed system contains five different entities: an authority, a cloud server, an edge server, a data owner, and data receivers.

- *The authority* is a trusted entity in our system, it holds the master secret key and conducts the entire system. The authority issues the secret key to the data receiver according to the user's attributes. It also generates the corresponding transformation key to the edge server. Note that, we assume that the authority neither colludes with any other entities nor is compromised.
- *The cloud server* is a semi-honest entity in our system, the encryption of the data is stored on the cloud server, and it helps the data receiver partially decrypt the ciphertext.
- *The edge server* is a semi-honest entity in our system, it receives all the partial decrypted ciphertext, and helps the data receiver to filter the message (i.e., only the user's attributes satisfy the policy, it will send the partial decrypted ciphertext to the data receiver).
- *The data owner* encrypts the message M under the policy W, and generates a corresponding ciphertext CT. Then he/she uploads the ciphertext with the index of the policy I_W. Therefore, the cloud server cannot obtain the exact policy in the ciphertext.
- *The data receiver* receives the subscribed data from the edge server. It first obtains a partially decrypted ciphertext from the edge server. Then, the data receiver decrypts the ciphertext and outputs the plaintext M.

B. Design Goals

The PPDF system is aimed to achieve the following goals.

- *Minimize Computation:* PPDF allows the data receiver delegate the heavy computation to the cloud server. It is impossible for the cloud server to obtain any sensitive information such as the plaintext, user's secret key.
- *Privacy-Preserving:* PPDF ensures the cloud server cannot access the data receiver's attributes, which improved the privacy of the data receiver.
- *File Filtering:* PPDF allows an edge server do the file filtering, while the edge sever also cannot obtain any information about the plaintext.

C. Threat Model

Suppose there is a malicious external adversary, who wants to obtain the plaintext. The adversary can access the public parameters, all the data which is transferred in the public channel, and the ciphertext stored on the cloud server. The adversary can also join the system as a receiver but will not be issued a valid secret key corresponding to a specified access policy.

The cloud server is a semi-honest entity, it will execute all the algorithm correctly, but will attempt to learn all possible information from legitimately received messages (such as user's attributes).

In addition, the cloud server will not collude with the edge server, which means that, the edge server will only receive the partial decrypted ciphertext from the cloud server, they will not reveal any other information to each other.

D. Protocol Workflow

A PPDF protocol consists of the following phases:

- *System Initialization:* This phase is run by the authority. In this phase, the authority should initialize the system, and generate the public parameters and the master secret key. Every entity in the system can access the public parameters, while the authority keeps the master secret key secret.
- *User Registration:* In this phase, the user sends registration request to the authority, if the user is valid, the authority will generate a secret key for the user according to his/her attributes. Meanwhile, the authority should generate a transformation key for the cloud server and a secret value for the edge.
- *Data Sharing:* This phase is run by the data owner. The data owner encrypts the plaintext under a specific access policy, then uploads the ciphertext to the cloud server. Only a registered user whose attributes match the policy can decrypt the ciphertext.
- *Partial Decryption:* This phase is mainly run by the cloud server, upon receiving the request from the data receiver, the cloud server partially decrypts the ciphertext to generate a partial decrypted ciphertext, then sends it to the edge server.
- *Data Filtering:* This phase is mainly run by the edge server. After receiving the partial decrypted ciphertext from the cloud server, the edge server can filter the ciphertext, if the ciphertext cannot be decrypted by the data receiver (i.e., the data receiver's attributes do not satisfy the policy), it will be filtered out. Otherwise, the edge server returns the partial decrypted ciphertext to the data receiver.
- *Data Access:* This phase is mainly run by the data receiver. After receive the partial decrypted ciphertext from the edge server, the data receiver uses the secret key to decrypt it and obtains the plaintext.

The progress of the above phases will be introduced in detailed in next section.

V. System Design

In this section, we first present the formal algorithm definition, and describe the detailed system design. This system is built on top of outsourced decryption ABE with filtering. The data owner encrypts the data using the ABE **Encrypt** algorithm, then uploads the ciphertext to the cloud. The authority acts as a PKG that delivers the secret keys to different entities. The cloud not only stores the data, but also helps the data receiver to do partial decryption. Finally, the edge server filters the ciphertext and returns the valid ciphertext to data receiver.

A. Algorithm Definitions

An outsourced decryption ABE with filtering contains the following algorithms.

- **Setup**: Taking as input a security parameter λ, and n attributes in universe $U = \{W_1, W_2, \ldots, W_i, \ldots, W_n\}$, this algorithm outputs the public parameters MPK and the master secret key MSK.
- **Encrypt**: Taking as input an access structure W, the public parameters MPK, and a message M, this algorithm outputs the ciphertext CT such that only the user possesses a set of attributes satisfies the policy can decrypt the ciphertext.
- $\mathbf{KeyGen}_{out}$: Taking as input the public parameters MPK, the master secret key MSK, an attribute set L, this algorithm outputs a transformation key TK, a secret value t, and a partial secret key SK.
- **Transform**: Taking as input the public parameters MPK, a transformation key TK, and a ciphertext CT, this algorithm outputs the partially decrypted ciphertext CT'.
- **Filter**: Taking as input the public parameters MPK, a secret value t, a partially decrypted ciphertext CT', this algorithm output the CT' or $\perp$.
- **Decrypt**: Taking as input the public parameters MPK, a partially decrypted ciphertext CT', and a partial secret key SK, this algorithm outputs the message M.

Correctness: The correctness of PPDF scheme is as follows. For all MPK, MSK, W, L such that the attribute set L satisfies the access policy W, if $(SK, TK) \leftarrow \mathbf{KeyGen}_{out}(L, MSK, MPK)$, $CT \leftarrow \mathbf{Encrypt}(W, M, MPK)$ and $CT' \leftarrow \mathbf{Transform}(TK, Params, CT)$, we have $M = \mathbf{Decrypt}(CT', SK, MPK)$.

B. Designed System

1) System Initialization: First, the authority initializes the system to generate the private and public parameters, where the input security parameter is λ. It runs the **Setup** algorithm to generate MSK and MPK. Assume there are n attributes in universe and the attribute set is $U = \{W_1, W_2, \ldots, W_i, \ldots, W_n\}$. Each attribute has **Positive** (1) and **Negative** (0) options, i.e., $W_i = (w_{i,0}, w_{i,1})$. $\mathbb{G}_1, \mathbb{G}_2$ are two cyclic groups of prime order p, g_1 and g_2 are the generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively, there exists a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. The authority chooses $x, y \xleftarrow{r} \mathbb{Z}_p^*$, computes $X_{i,k_i} = g_1^{-H(x||i||k_i)}, Y_{i,k_i} = e(g_1, g_2)^{H(y||i||k_i)}$ where k_i is 1 or 0. The authority sets MPK as $(g_1, g_2, \{X_{i,0}, X_{i,1}, Y_{i,0}, Y_{i,1}\}_{1 \le i \le n})$ and MSK as (x, y). All the entities in the system can access MPK, the authority keeps MSK secret.

2) User Registration: If a data receiver wants to join the system, he/she should register to the system, and gets a partial decryption secret key from the authority. The user's attributes $L = (v_{1,l_1}, v_{2,l_2}, \ldots, v_{i,l_i}, \ldots, v_{n,l_n})$ should satisfy that $L \in U$. After receiving the registration request from the data receiver, the authority checks whether the data receiver is a valid one. If the data receiver is valid, then the authority uses his/her attributes L, public parameters MPK, and master secret key

MSK to generate the secret key SK_L. The authority runs $\mathbf{KeyGen}_{out}$ algorithm on input (MPK, MKS, L), it selects random numbers $r_1, r_2, t, \xleftarrow{r} \mathbb{Z}_p^*$, computes

$$\sigma_{i,l_i} = g_2^{H(y||i||l_i)} g_2^{r_1 H(x||i||l_i)}, R_1 = g_2^{r_1}$$
$$\gamma_{i,l_i} = g_2^{tH(y||i||l_i)} g_2^{r_2 H(x||i||l_i)}, R_2 = g_2^{r_2},$$

sets the secret key as $SK_L = (t, \{\sigma_{i,l_i}\}_{1\le i\le n}, \{\gamma_{i,l_i}\}_{1\le i\le n}, R_1, R_2)$. Then, it selects $z \xleftarrow{r} \mathbb{Z}_p^*$ randomly, computes $TK_1 = \sigma_{i,l_i}^{1/z}$, $TK_2 = \gamma_{i,l_i}^{1/z}$, $TK_3 = R_1^{1/z}$, $TK_4 = R_2^{1/z}$. Then, the authority sends the transformation key $TK = (TK_1, TK_2, TK_3, TK_4)$ to the cloud, sends the secret value t to the edge server, and returns the partial decryption secret key $SK = (TK, z)$ to the data receiver.

3) Data Sharing: Each data owner can run this phase to sharing their data on the cloud server. Taking as input the public parameters MPK, the message M, and the policy W, the data owner executes the **Encrypt** algorithm to encrypt the message M under the access policy W. Suppose policy $W = \wedge_{i\in I_W} W_i$, where $W_i = v_{i,k_i}$. The data owner chooses $s \xleftarrow{r} \mathbb{Z}_p^*$ randomly, and computes $(X_W, Y_W) = (\prod_{i\in I_W} \bar{X}_i, \prod_{i\in I_W} \bar{Y}_i)$ with $(\bar{X}_i, \bar{Y}_i) = (X_{i,k_i}, Y_{i,k_i})$, and sets

$$\begin{cases} C_0 = M \cdot Y_W^s, \\ C_1 = g_1^s, \\ C_2 = X_W^s, \end{cases}$$

Then, the data owner sets the ciphertext as $CT = (I_W, C_0, C_1, C_2)$, where I_W is the index of the policy W, and uploads this ciphertext to the cloud server. Note that, the cloud can not obtain the exact policy from the ciphertext, the only thing it can learn is the index of the policy.

4) Partial Decryption: Upon receiving the access request from the edge server, the cloud server first locates the corresponding transformation key TK from the request. It then takes TK, along with MPK and the data CT as input and runs **Transform** algorithm to get the partially decrypted ciphertext:

$$\bar{C}_1 = e\left(C_1, \prod_{i\in I_W} TK_1\right) e\left(C_2, TK_3\right),$$
$$\bar{C}_2 = e\left(C_1, \prod_{i\in I_W} TK_2\right) e\left(C_2, TK_4\right),$$

sets the partially decrypted ciphertext $CT' = (\bar{C}_1, \bar{C}_2)$, and returns CT' to the edge server.

5) Data Filtering: Upon receiving CT' from the cloud server, the edge server uses the public parameters MPK, the secret value t together with CT' as the input, and runs **Filter** algorithm to test whether the equation $\bar{C}_1^{\ t} = \bar{C}_2$ holds. If the equation holds, it pushes CT' to the data receiver. Otherwise, it means the attributes do not satisfy the policy, the edge server filters out it, and outputs $\perp$.

6) Data Access: Upon receiving the partial decrypted request data CT' from the edge server, the data receiver can output the plaintext such that:

$$M = \frac{C_0}{\bar{C}_1^{\ z}}.$$

C. Correctness

The edge server obtains the ciphertext $(\bar{C}_1, \bar{C}_2)$, where

$$\begin{aligned}\bar{C}_1 &= e\left(C_1, \prod_{i\in I_W} TK_1\right) e\left(C_2, TK_3\right) \\ &= e\left(g_1, g_2\right)^{sH(y||i||l_i)/z} e(g_1, g_2)^{sr_1(H(x||i||l_i) - H(x||i||l_i)')/z},\end{aligned}$$

$$\begin{aligned}\bar{C}_2 &= e\left(C_1, \prod_{i\in I_W} TK_2\right) e(C_2, TK_4) \\ &= e(g_1, g_2)^{stH(y||i||l_i)/z} e(g_1, g_2)^{sr_2(H(x||i||l_i) - H(x||i||l_i)')/z}.\end{aligned}$$

Only when $H(x||i||l_i)$ equals to $H(x||i||l_i)'$ (i.e., the attributes satisfy the policy in the ciphertext), the equation $\bar{C}_1^{\ t} = \bar{C}_2$ holds. If the equation holds, the edge server returns $(\bar{C}_1, \bar{C}_2)$ to the data receiver. Finally, data receiver computes

$$M = \frac{C_0}{\bar{C}_1^{\ z}}.$$

Therefore, our proposed encryption algorithm and decryption algorithm are correct.

VI. Security Analysis

In this section, we first prove the property of privacy-preserving. Then we prove that our proposed scheme is selective CPA secure.

A. Mathematical Assumptions

Definition 1: The Twin Decisional Diffie-Hellman (Twin-DDH) assumption [33]: Let $\mathbb{G}_2$ be a cyclic group of prime order p with generator g_2, given $g_2^{\alpha}, g_2^{\beta}, g_2^{\gamma}, Z$ where α, β, γ are chosen randomly from $\mathbb{Z}_p^*$, Z is a random element in $\mathbb{G}_2$. An algorithm $\mathcal{B}$ that outputs $b \in \{0, 1\}$ has advantage ϵ in solving twin DDH if

$$|\Pr[\mathcal{B}(g_2, g_2^{\alpha}, g_2^{\beta}, g_2^{\gamma}, g_2^{\alpha\gamma}) = 0]$$
$$- \Pr[\mathcal{B}(g_2, g_2^{\alpha}, g_2^{\beta}, g_2^{\gamma}, Z) = 0]| \ge \epsilon,$$

We say that the Twin-DDH assumption holds in $\mathbb{G}_2$ if no t-time algorithm has advantage at least ϵ in solving the Twin-DDH problem.

Definition 2: Decisional (t, ϵ, l)-BDHEasym: $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are three cyclic groups of prime order together with an asymmetric pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. Let $\mathbf{y}_{g,h,\alpha,l} = (g_1, g_2, \ldots, g_l, h_1, h_2, \ldots, h_l, h_{l+2}, \ldots, h_{2l})$, where $g_i = g^{\alpha^i}$, $h_i = h^{\alpha^i}$. An algorithm $\mathcal{B}$ that outputs $b \in \{0, 1\}$ has advantage ϵ in solving the decisional l-BDHEasym in $\mathbb{G}_T$ if

$$|\Pr[\mathcal{B}(g, h, g', h', \mathbf{y}_{g,h,\alpha,l}, e(g', h_{l+1})) = 0]$$
$$- \Pr[\mathcal{B}(g, h, g', h', \mathbf{y}_{g,h,\alpha,l}, Z) = 0]| \ge \epsilon,$$

where the probability is over the random choice of generators $g \xleftarrow{r} \mathbb{G}_1, h \xleftarrow{r} \mathbb{G}_2$, the random choice $\alpha \xleftarrow{r} \mathbb{Z}_p^*$, $Z \xleftarrow{r} \mathbb{G}_T$, and the random bits consumed by $\mathcal{B}$. We say that the decisional (t, ϵ, l)-BDHEasym assumption holds in $\mathbb{G}_T$ if no t-time algorithm has advantage at least ϵ in solving the decisional l-BDHE problem in $\mathbb{G}_T$.

B. Security Model of Privacy-Preserving

Definition 3: If $\mathcal{A}$ is a P.P.T algorithm, $\mathcal{C}$ be a challenger. The definition of privacy-preserving of user attributes is as follows:

- *Setup*: The challenger $\mathcal{C}$ generates the master secret key MSK and the mast public key MPK by using a security parameter λ. Then, it sends MPK to the adversary $\mathcal{A}$.
- *Phase 1*: The adversary can issue a polynomial number of queries to the challenger.
  Secret Key Query: The adversary queries an attribute list L to the challenger $\mathcal{C}$. Then $\mathcal{C}$ returns the corresponding secret key SK_L.
- *Challenge*: When $\mathcal{A}$ decides to finish **Phase 1**, it outputs an index value i to challenger. Assume that $TK^{(0)}$ denotes the attribute value of i in one of the keys is 0, $TK^{(1)}$ denotes the attribute value of i in one of the keys is 1. The challenger chooses $b \xleftarrow{r} \{0, 1\}$, and produces a transformation key $TK^{(b)}$. Then it returns $TK^{(b)}$ to the adversary.
- *Phase 2*: The same as Phase 1.
- *Guess*: The adversary $\mathcal{A}$ outputs a guess $b' \in \{0, 1\}$, and wins the game if $b' = b$.

The advantage of an adversary $\mathcal{A}$ in the above game is defined as $\epsilon = \Pr[b' = b] - \frac{1}{2}$.

The scheme can achieve the privacy-preserving property if for all P.P.T algorithms, we have ϵ is a negligible function of λ.

C. Security Proof of Privacy-Preserving

Theorem 1: If the decisional (τ, ϵ, n)-BDHEasym assumption holds in $\mathbb{G}_T$, then the cloud server in our system cannot obtain any information of user's attribute from the transformation keys.

Proof: Suppose that there exists an adversary $\mathcal{A}$, which breaks the privacy-preserving with $\mathbf{Adv}^{\mathbf{pp}}_{\mathbf{ABE}}(\mathcal{A}) \geq \epsilon$. We build a simulator $\mathcal{S}$ that has advantage ϵ in solving the DDH problem in $\mathbb{G}_2$. $\mathcal{S}$ takes as input a random DDH challenge $(g_2^\alpha, g_2^\beta, g_2^\gamma, Z)$, where Z is either $g_2^{\alpha\gamma}$ or a random element in $\mathbb{G}_2$. Without of loss of generality, we assume that the transformation key only contains the target attribute i. The simulator $\mathcal{S}$ interacts with the adversary $\mathcal{A}$ as follows.

Setup: The simulator selects $x, y \xleftarrow{r} \mathbb{Z}_p^*$ randomly, and generates system public parameters MPK as $(g_1, g_2, \{X_{i,0}, X_{i,1}, Y_{i,0}, Y_{i,1}\}_{1 \leq i \leq n})$. Then $\mathcal{S}$ returns MPK to $\mathcal{A}$.

Phase 1: The adversary makes the following queries:

Hash Oracle $\mathcal{O}_H(\cdot)$: When there is a query on H for input '$\cdot$', $\mathcal{S}$ first looks if there is an item containing '$\cdot$' in the list $\mathcal{L}$. If it is, the previous defined value is returned. Otherwise, it chooses $t \in \mathbb{Z}_p^*$, adds the entry $(\cdot, t)$ to $\mathcal{L}$ and returns t.

KeyGen Oracle $\mathcal{O}_{KeyGen}(L)$: The simulator $\mathcal{S}$ chooses $i^* \xleftarrow{r} [1, n]$. On input the query attribute $L = (v_{1,l_1}, v_{2,l_2}, \ldots, v_{i,l_i}, \ldots, v_{n,l_n})$, $\mathcal{C}$ selects randomly $r \xleftarrow{r} \mathbb{Z}_p^*$, computes $R = g_2^r$, if $i = i^*$, sets

$$\sigma_{i^*,l_{i^*}} = \begin{cases} g_2^\alpha g_2^{rH(x||i^*||l_{i^*})} & \text{if } l_{i^*} = 0, \\ g_2^\beta g_2^{rH(x||i^*||l_{i^*})} & \text{if } l_{i^*} = 1. \end{cases}$$

Otherwise,

$$\sigma_{i,l_i} = g_2^{H_0(y||i||l_i)} g_2^{rH(x||i||l_i)}$$

Returns the secret key $SK_L = (\{\sigma_{i,l_i}\}_{1 \leq i \leq n}, R)$.

Challenge: The adversary outputs an index value i^* to challenger. The challenger chooses $b \xleftarrow{r} \{0, 1\}$ and $r_1, r_2 \xleftarrow{r} \mathbb{Z}_p^*$, and produces a transformation key $TK^{(b)}$ as follows:

$$T_1 = Z \cdot g_2^{\gamma H(x||i^*||b) r_1}, T_2 = (g_2^\gamma)^{r_1},$$

$$T_3 = Z^t \cdot g_2^{\gamma H(x||i^*||b) r_2}, T_4 = (g_2^\gamma)^{r_2}.$$

Phase 2: The same as Phase 1.

Guess: The adversary $\mathcal{A}$ outputs a guess bit b' of b. If $b' = b$, the simulator $\mathcal{S}$ outputs 1 in the twin DDH game to guess that $Z = g_2^{\alpha\gamma}$. Otherwise, it outputs 0 to indicate that Z is a random element in $\mathbb{G}_2$.

This completes the description of the simulation and the solution of twin DDH problem. It is not hard to verify that the simulation is indistinguishable from the real scheme. If $Z = g_2^{\alpha\gamma}$, then $TK^{(b)}$ is a valid transformation key, according to the assumption, we have

$$\Pr[\mathcal{S}(g_2, g_2^\alpha, g_2^\beta, g_2^\gamma, g_2^{\alpha\gamma}) = 1] = \frac{1}{2} + \mathbf{Adv}^{\mathbf{pp}}_{\mathbf{ABE}}(\mathcal{A}) \geq \frac{1}{2} + \epsilon.$$

If Z is another element in $\mathbb{G}_2$, as r_1, r_2, t are random chosen, we have T_1, T_2, T_3, T_4 are random and independent. Therefore, the transformation key TK is completely hidden from the challenger transformation key $TK^{(b)}$, and we have

$$\Pr[\mathcal{S}(g_2, g_2^\alpha, g_2^\beta, g_2^\gamma, Z) = 1] = \frac{1}{2}.$$

Therefore, the advantage of the simulator $\mathcal{S}$ in solving the twin DDH problem is

$$\begin{aligned} \mathsf{Adv} &= |\Pr[\mathcal{S}(g_2, g_2^\alpha, g_2^\beta, g_2^\gamma, g_2^{\alpha\gamma}) = 1] \\ &\quad - \Pr[\mathcal{S}(g_2, g_2^\alpha, g_2^\beta, g_2^\gamma, Z) = 1]| \\ &= \frac{1}{2} + \epsilon - \frac{1}{2} \\ &= \epsilon. \end{aligned}$$

□

D. Security Model of Outsourced Decryption ABE With Filtering

Definition 4: If $\mathcal{A}$ is a P.P.T algorithm, $\mathcal{C}$ be a challenger. The selective ciphertext-policy and chosen-plaintext attacks (IND-sCP-CPA) security game for outsourced decryption ABE with filtering is defined as follows:

- *Initiation*: The adversary $\mathcal{A}$ outputs a challenge ciphertext policy W^*.

- *Setup*: The challenger $\mathcal{C}$ generates the master secret key MSK and the mast public key MPK by using a security parameter λ. Then, it gives MPK to the adversary $\mathcal{A}$.
- *Phase 1*: The adversary can issue a polynomial number of queries to the challenger.
  Secret key query:The adversary queries an attribute list L to the challenger $\mathcal{C}$, where $L \not\models W^*$. C returns the corresponding secret key SK_L.
  Transformation key query: On a query attribute list L, the challenger $\mathcal{C}$ returns the corresponding transformation key TK.
- *Challenge*: When $\mathcal{A}$ decides to finish **Phase 1**, it outputs two messages (M_0, M_1) to challenge. The challenger randomly chooses a bit $b \xleftarrow{r} \{0,1\}$, and computes $CT^* = \mathbf{Encrypt}(M_b, W^*)$. Then, the challenger $\mathcal{C}$ returns CT^* to the adversary.
- *Phase 2*: The same as Phase 1.
- *Guess*: The adversary $\mathcal{A}$ outputs a guess $b' \in \{0,1\}$, and wins the game if $b' = b$.

The advantage of an adversary $\mathcal{A}$ in the above game is defined as $\epsilon = \Pr[b' = b] - \frac{1}{2}$.

The scheme is secure against selective ciphertext-policy and chosen-plaintext attacks if for all P.P.T algorithms, we have ϵ is a negligible function of λ.

E. Security Proof of CPA Secure

Theorem 2: If the decisional (τ, ϵ, n)-BDHEasym assumption holds in $\mathbb{G}_T$, then our the privacy-preserving cloud-based data distribution scheme with filtering above is IND-sCP-CPA secure.

Proof: Suppose that there exists a τ-time adversary $\mathcal{A}$, which breaks the proposed scheme with $\mathbf{Adv}^{\mathbf{IND-sCP-CPA}}_{\mathbf{CP-ABE}}(\mathcal{A}) \geq \epsilon$. We build a simulator $\mathcal{S}$ that has advantage ϵ in solving the decisional n-BDHEasym problem in $\mathbb{G}_T$. $\mathcal{S}$ takes as input a random decisional n-BDHEasym challenge $(g, h, \mathbf{y}_{g,h,\alpha,n}, Z)$, where $\mathbf{y}_{g,h,\alpha,n} = (g_1, g_2, \ldots, g_n, h_1, h_2, \ldots, h_n, h_{n+2}, \ldots, h_{2n})$ and Z is either $e(g, h_{n+1})$ or a random element in $\mathbb{G}_T$. The simulator $\mathcal{S}$ plays the role of the challenger in the IND-sCP-CPA game, and interacts with the adversary $\mathcal{A}$ as follows.

Initial: The simulator $\mathcal{S}$ receives a challenge access structure $W^* = \wedge_{i \in I_{W^*}} W_i$ specified by the adversary $\mathcal{A}$, where $I_{W^*} = \{1, 2, \ldots, w\}$ with $w \leq n$ represents the attribute index set specified in W^*.

Setup: The simulator $\mathcal{S}$ needs to generate a system public key MPK. $\mathcal{S}$ chooses $j^* \in \{1, 2 \ldots, w\}$ and $x, x', y, y' \xleftarrow{r} \mathbb{Z}_p^*$. Then, it does the following:

1) If $j \in I_{W^*} - \{j^*\}$, suppose $W_j = v_{j,k_j}$, then $\mathcal{S}$ computes:
   $(X_{j,k_j}, Y_{j,k_j}) = (g^{-H(x||j||k_j)} g_{n+1-j}^{-1}, e(g,h)^{H(y||j||k_j)})$.
   For $k \neq k_j$, $\mathcal{S}$ computes:
   $(X_{j,k_j}, Y_{j,k_j}) = (g^{-H(x'||j||k_j)}, e(g,h)^{H(y'||j||k_j)})$.
2) For j^*, suppose $W_{j^*} = v_{j^*,k_{j^*}}$, $\mathcal{S}$ computes:

$$(X_{j^*}, Y_{j^*}) = \left(g^{-H(x||j^*||k_{j^*})} \prod_{t \in I_{W^*} - \{j^*\}} g_{n+1-t}, \right.$$
$$\left. e(g,h)^{H(y||j^*||k_{j^*})} e(g,h)^{\alpha^{n+1}} \right).$$

   For $k \neq k_{j^*}$, $\mathcal{S}$ computes:

$$(X_{j^*,k}, Y_{j^*,k}) = (g^{-H(x'||j^*||k)}, e(g,h)^{H(y'||j^*||k)}).$$

3) If $j \notin I_{W^*}$, for $k_j \in \{0,1\}$, $\mathcal{S}$ computes:

$$(X_{j,k_j}, Y_{j,k_j}) = (g^{-H(x||j||k_j)}, e(g,h)^{H(y||j||k_j)}).$$

Then $MPK = (g, h, \{X_{i,0}, X_{i,1}, Y_{i,0}, Y_{i,1}\}_{1 \leq i \leq n})$, and $\mathcal{S}$ sends MPK to $\mathcal{A}$.

Phase 1: The adversary $\mathcal{A}$ makes the following queries.

- *Hash Oracle* $\mathcal{O}_H(\cdot)$: When there is a query on H for input '$\cdot$', $\mathcal{S}$ first looks if there is an item containing '$\cdot$' in the list $\mathcal{L}$. If it is, the previous defined value is returned. Otherwise, it chooses $t \in \mathbb{Z}_p^*$, adds the entry $(\cdot, t)$ to $\mathcal{L}$ and returns t.
- *KeyGen Oracle* $\mathcal{O}_{KeyGen}(L)$: Suppose $\mathcal{A}$ submits an attribute list L in a secret key query, where $L \not\models W$. There must exist $j \in I_{W^*}$ such that $L_j \notin W_j$, $\mathcal{S}$ chooses such j. Without loss of generality, assume $L_j = v_{j,\hat{k}_j}$, and $W_j = v_{j,k_j}$. $\mathcal{S}$ computes $\sigma_{j,\hat{k}_j} = h^{H(y'||j||\hat{k}_j)} (h_j^r)^{H(x'||j||\hat{k}_j)}$. For $t \neq j$, $\mathcal{S}$ computes σ_{t,k_t} as follows:
  Case 1 If $t \in I_{W^*} - \{j^*\}$, suppose $L_t = v_{t,k_t}$, $\mathcal{S}$ computes

$$\sigma_{t,k_t} = h^{H(y||t||k_t)} (h_j^r)^{H(x||t||k_t)} h_{n+1-t+j}, h_j^r.$$

  Case 2 If $t = j^*$, suppose $L_{j^*} = v_{j^*,k_{j^*}}$, $\mathcal{S}$ computes

$$\sigma_{j^*,k_{j^*}} = h^{H(y||j^*||k_{j^*})} (h_j^r)^{H(x||j^*||k_{j^*})} \left(\prod_{k \in I_{W^*} - \{ij^*,j\}} h_{n+1-k+j}^{-1} \right), h_j^r.$$

  Case 3 If $t \notin I_{W^*}$, suppose $L_t = v_{t,k_t}$, $\mathcal{S}$ computes

$$\sigma_{t,k_t} = h^{H(y||t||k_t)} (h_j^r)^{H(x||t||k_t)}, h_j^r.$$

  Finally, $\mathcal{S}$ returns $SK_L = (\sigma_i, h_j^r)$.
- *Transformation Oracle* Suppose $\mathcal{A}$ submits an attribute list L in a secret key query.
  If $L \models W$, then it chooses a "fake" transformation key as follows: randomly selects $d \xleftarrow{r} \mathbb{Z}_p^*$, and runs **KeyGen** to obtain SK'_L. The simulator $\mathcal{S}$ sets $TK = SK'_L$ and $SK_L = (d, TK)$. If d was replaced by $z = \alpha/d$, then the transformation key is properly distributed.
  Otherwise, it runs the **KeyGen** oracle to obtain the secret key SK_L. Note that, the simulator $\mathcal{S}$ should select another random value $d', r_2 \xleftarrow{r} \mathbb{Z}_p^*$ to generate $(\gamma_i, h_j^{r_2})$, then it sets $SK_L = (d', \sigma_i, h_j^r, \gamma_i, h_j^{r_2})$. $\mathcal{S}$ selects $z \xleftarrow{r} \mathbb{Z}_p^*$ randomly and sets the transformation key $TK = (\sigma_{i,l_i}^{1/z}, \gamma_{i,l_i}^{1/z}, (h_j^r)^{1/z}, (h_j^{r_2})^{1/z})$.

The simulator $\mathcal{S}$ returns TK to the adversary.

Challenge: The simulator $\mathcal{S}$ sets

$$x_{W^*} = \sum_{t \in I_{W^*}} H(x||t||k_t) = \sum_{j=1}^{w} H(x||j||k_j)$$

$$y_{W^*} = \sum_{j=1}^{w} H(y||j||k_j)$$

and defines (X_{W^*}, Y_{W^*}) as follows:

$$\begin{aligned} X_{W^*} &= \bar{X_{j^*}} \prod_{t \in I_{W^*}-\{j^*\}} \bar{X_t} \\ &= (g^{-H(x||j^*||k_{j^*})} \prod_{t \in I_{W^*}-\{j^*\}} g_{n+1-t}) \cdot \prod_{t \in I_{W^*}-\{j^*\}} g^{-H(x||t||k_t)} g_{n+1-t}^{-1} \\ &= g^{-x_{W^*}}, \\ Y_{W^*} &= \bar{Y_{j^*}} \prod_{t \in I_{W^*}-\{j^*\}} \bar{Y_t} \\ &= e(g,h)^{H(x||j^*||k_{j^*})} e(g,h)^{\alpha^{n+1}} \cdot \prod_{t \in I_{W^*}-\{j^*\}} e(g,h)^{H(y||t||k_t)} \\ &= e(g,h)^{\sum_{j=1}^{w} H(y||j||k_j)+\alpha^{n+1}}. \end{aligned}$$

The adversary $\mathcal{A}$ submits two messages M_0 and M_1 of equal length. The simulator $\mathcal{S}$ can challenge $\mathcal{A}$ as follows. $\mathcal{S}$ chooses $b \in \{0,1\}$, and computes

$$\begin{cases} C_0^* = M_b \cdot Y_W^s = M_b \cdot Ze(g',h)^{y_{W^*}}, \\ C_1^* = g^s = g', \\ C_2^* = (g')^{-x_{W^*}}, \end{cases}$$

The challenge ciphertext $CT_{W^*} = (W^*, C_0^*, C_1^*, C_2^*)$ is a valid encryption of M_b if $Z = e(g, h_{n+1})$. On the other hand, when Z is a random element in $\mathbb{G}_T$, CT_{W^*} is independent of b in the adversary's view.

Phase 2: The same as **Phase 1**.

Guess: The adversary $\mathcal{A}$ outputs a guess bit b' of b. If $b' = b$, the simulator $\mathcal{S}$ outputs 1 in the decisional n-BDHEasym game to guess that $Z = e(g, h_{n+1})$. Otherwise, it outputs 0 to indicate that Z is a random element in $\mathbb{G}_T$.

This completes the description of the simulation and the solution of n-BDHEasym problem. It is not hard to verify that the simulation is indistinguishable from the real scheme. If $Z = e(g, h_{n+1})$, then CT_{W^*} is a valid ciphertext, according to the assumption, we have

$$\begin{aligned} &\Pr[\mathcal{S}(g, h, \mathbf{y}_{g,h,\alpha,n}, e(g, h_{n+1})) = 1] \\ &= \frac{1}{2} + \mathbf{Adv}_{\mathbf{CP-ABE}}^{\mathbf{IND-sCP-CPA}}(\mathcal{A}) \geq \frac{1}{2} + \epsilon. \end{aligned}$$

If Z is a random element in $\mathbb{G}_T$, as g', x, y, g_i, h_i are random chosen, we have $Ze(g', h)^{y_{w^*}}, g', (g')^{-x_{w^*}}$ are random and independent. Therefore, the message M_b is completely hidden from $\mathcal{A}$, and we have

$$\Pr[\mathcal{S}(g, h, \mathbf{y}_{g,h,\alpha,n}, Z) = 1] = \frac{1}{2}.$$

Therefore, the advantage of the simulator $\mathcal{S}$ in solving the decisional n-BDHEasym problem is

$$\begin{aligned} \mathsf{Adv}_{\mathcal{S}} &= |\Pr[\mathcal{S}(g, h, \mathbf{y}_{g,h,\alpha,n}, e(g, h_{n+1})) = 1] \\ &- \Pr[\mathcal{S}(g, h, \mathbf{y}_{g,h,\alpha,n}, Z) = 1]| \\ &= \frac{1}{2} + \epsilon - \frac{1}{2} \\ &= \epsilon. \end{aligned}$$

□

TABLE I
RUNNING TIME ON PC

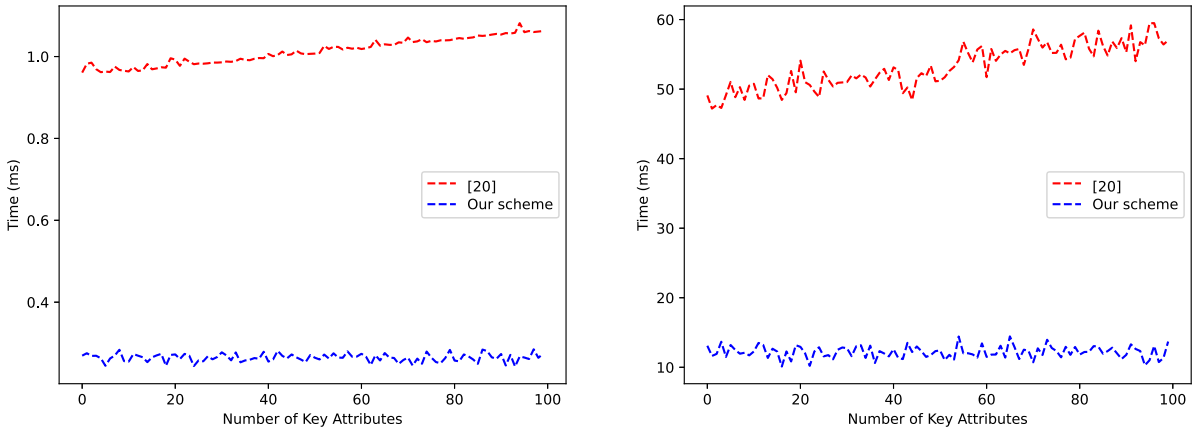
n \ Algorithm	**Setup**	**KeyGen**$_{out}$	**Encrypt**	**Transform**
10	1.28 ms	9.78 ms	0.47 ms	1.92 ms
20	2.08 ms	18.87 ms	0.49 ms	1.95 ms
30	2.89 ms	27.96 ms	0.52 ms	1.97 ms
40	3.68 ms	37.01 ms	0.54 ms	1.99 ms
50	4.47 ms	46.12 ms	0.57 ms	2.02 ms
60	5.27 ms	55.15 ms	0.60 ms	2.04 ms
70	6.07 ms	64.24 ms	0.62 ms	2.06 ms
80	6.86 ms	73.33 ms	0.65 ms	2.08 ms
90	7.66 ms	82.30 ms	0.67 ms	2.11 ms
100	8.45 ms	91.41 ms	0.70 ms	2.13 ms

VII. PERFORMANCE EVALUATION

We use RELIC to evaluate the performance of our proposed scheme on a PC with VMware ESXi (an Intel Xeon E5-2678 v3 @ 12x 2.494 GHz CPU, 32 GB RAM, 64-bit Ubuntu 20.04 operating system) and a Raspberry Pi 3b (an ARM Cortex-A53 @ 4x 1.2 GHz CPU, 1 GB RAM, 32-bit Raspbian operating system). The curve we used in our experiment is BN-254 curve [34] (*x64-pbc-bn254.sh* on PC, *arm-pbc-bn254.sh* on Raspberry Pi 3b) with embedding degree $k = 12$, the bit lengths of elements in $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ are 64 bytes, 128 bytes and 384 bytes, respectively, which can achieve the 128-bit security level.

First, we analyze the communication costs of each phase. Assume that there are n attributes in universe, the data user has l attributes, the policy in the ciphertext is exactly same as the data user's attributes. $|\mathbb{Z}_p|$, $|\mathbb{G}_1|$, $|\mathbb{G}_2|$ and $|\mathbb{G}_T|$ denote the size of the element in the groups $\mathbb{Z}_p$, $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ respectively. In *User Registration* phase, PKG sends transformation key to the cloud server, the communication cost is $(2l+2)|\mathbb{G}_2|$, sends secret value to the edge server, the communication cost is $|\mathbb{Z}_p|$, and the communication cost of sending partial decryption secret key to the data receiver is $(2l+2)|\mathbb{G}_2| + |\mathbb{Z}_p|$. In *Data Sharing* phase, the communication cost is a constant size $2|\mathbb{G}_1| + |\mathbb{G}_T|$ plus the size of the index. In *Partial Decryption* phase, the cloud server sends partially decrypted ciphertext to the edge server, the communication cost is $2|\mathbb{G}_T|$. In *Data Filtering* phase, if the data receiver's attributes satisfy the policy, the edge server will push the partially decrypted ciphertext to the data receiver, it costs $2|\mathbb{G}_T|$.

Then, we implement our scheme on the PC, and we vary the number of the attributes from 1 to 100. Suppose every attribute in the **Setup** phase will be used in the **Encrypt** and **Decrypt** (i.e., the policy does not contain "DON't CARE" attribute, and all the attributes initialed by **Setup** algorithm should be contained). All the experiments are running 5,000 times, the averaged results are shown in Table I (n denotes the number of attributes). The computational cost of **Setup**, $\mathbf{KeyGen}_{out}$, **Encrypt** and **Transform** are almost linearly dependent on the number of the attribute. As shown from Table I, when the number of the attribute achieves 100, the data owner should take only 0.70 ms to encrypt the data, $\mathbf{KeyGen}_{out}$ takes longer than other algorithms, it costs 91.41 ms to generate the corresponding keys. However, in the real applications, this algorithm is run by a trusted third party with strong computing power. Therefore, the whole scheme is efficient and acceptable. The time cost of



(a) Decyption Time Cost on PC

(b) Decyption Time Cost on Raspberry Pi 3b

Fig. 3. Benchmark results between our scheme and [20]. Timing results are provided for both PC and Raspberry Pi 3b based on the number of key attributes.

the **Filter** algorithm and **Decrypt** algorithm are independent of the number of attributes, they are about 0.26 ms and 0.27 ms respectively. Besides, the size of the ciphertext is constant, it does not influence by the number of attributes.

Furthermore, we compared the user side (the data receiver) computational cost between our scheme and [20]. As shown in Fig. 3, the computational cost of the user side in [20] is associated with the number of attributes, it increases linearly with the number of attributes. On the contrary, in our scheme, the computational cost of the user side is constant, on PC the decryption costs about 0.26 ms, on Raspberry Pi 3b, it costs about 12.18 ms. Obviously, our scheme is more efficient and parasitical especially on some lightweight devices.

VIII. Conclusion

In this article, we proposed a privacy-preserving cloud-based data distribution system with filtering. Specifically, it allows a semi-trusted cloud to do partial decryption for the user without knowing user's attributes, while the user can test whether his/her attributes satisfy the policy in the ciphertext. Besides, the user can also delegate the test phase to an edge server. According to the performance evaluation, our proposed scheme is potentially useful in the real world applications.

In the future, we intend to extend it to a more complex environment, such as when the partial decryption is run by a malicious edge server.

References

[1] Verizon, 2021 data breach investigations report. Accessed: 2021. [Online]. Available: https://www.verizon.com/business/resources/reports/dbir/2021/masters-guide/

[2] A. Sahai and B. R. Waters, "Fuzzy identity-based encryption," in *Proc. Annu. Int. Conf. Theory Appl. Cryptographic Techn.*, ser. Lecture Notes in Computer Science, R. Cramer, Ed., Aarhus, Denmark: Springer, May 22–26, 2005, pp. 457–473.

[3] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proc. IEEE Symp. Secur. Privacy*, Oakland, CA, USA, May 20–23, 2007, pp. 321–334.

[4] L. Cheung and C. C. Newport, "Provably secure ciphertext policy ABE," in *Proc. 14th Conf. Comput. Commun. Secur.*, P. Ning, S. De Capitani di Vimercati, and P. F. Syverson, Eds., Alexandria, VA, USA: ACM Press, Oct. 28–31, 2007, pp. 456–465.

[5] B. Waters, "Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization," in *Proc. 14th Int. Conf. Theory Pract. Public Key Cryptogr.*, ser. Lecture Notes in Computer Science, D. Catalano, N. Fazio, R. Gennaro, and A. Nicolosi, Eds., Taormina, Italy: Springer, Mar. 6–9, 2011, pp. 53–70.

[6] J. Han, W. Susilo, Y. Mu, J. Zhou, and M. H. Au, "PPDCP-ABE: Privacy-preserving decentralized ciphertext-policy attribute-based encryption," in *Proc. 19th Eur. Symp. Res. Comput. Secur.*, ser. Lecture Notes in Computer Science, M. Kutylowski and J. Vaidya, Eds., Wroclaw, Poland: Springer, Sep. 7–11, 2014, pp. 73–90.

[7] M. Green, S. Hohenberger, and B. Waters, "Outsourcing the decryption of ABE ciphertexts," in *Proc. 20th USENIX Secur. Symp.*, San Francisco, CA, USA, Aug. 8–12, 2011, pp. 34–34.

[8] J. Lai, R. H. Deng, C. Guan, and J. Weng, "Attribute-based encryption with verifiable outsourced decryption," *IEEE Trans. Inf. Forensics Secur.*, vol. 8, no. 8, pp. 1343–1354, Aug. 2013.

[9] B. Qin, R. H. Deng, S. Liu, and S. Ma, "Attribute-based encryption with efficient verifiable outsourced decryption," *IEEE Trans. Inf. Forensics Secur.*, vol. 10, no. 7, pp. 1384–1393, Jul. 2015.

[10] C. Feng, K. Yu, M. Aloqaily, M. Alazab, Z. Lv, and S. Mumtaz, "Attribute-based encryption with parallel outsourced decryption for edge intelligent IoV," *IEEE Trans. Veh. Technol*, vol. 69, no. 11, pp. 13 784–13 795, Nov. 2020.

[11] T. Nishide, K. Yoneyama, and K. Ohta, "Attribute-based encryption with partially hidden encryptor-specified access structures," in *Proc. 6th Int. Conf. Appl. Cryptogr. Netw. Secur.*, ser. Lecture Notes in Computer Science, S. M. Bellovin, R. Gennaro, A. D. Keromytis, and M. Yung, Eds., New York, NY, USA: Springer, Jun. 3–6, 2008, pp. 111–129.

[12] T. V. X. Phuong, G. Yang, and W. Susilo, "Hidden ciphertext policy attribute-based encryption under standard assumptions," *IEEE Trans. Inf. Forensics Secur.*, vol. 11, no. 1, pp. 35–45, Jan. 2016.

[13] H. Cui, R. H. Deng, G. Wu, and J. Lai, "An efficient and expressive ciphertext-policy attribute-based encryption scheme with partially hidden access structures," in *Proc. Int. Conf. Provable Secur.*, Springer, 2016, pp. 19–38.

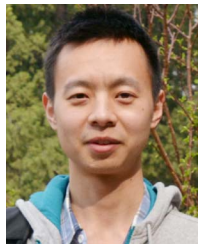
[14] H. Xiong, Y. Zhao, L. Peng, H. Zhang, and K.-H. Yeh, "Partially policy-hidden attribute-based broadcast encryption with secure delegation in edge computing," *Future Gener. Comput. Syst.*, vol. 97, pp. 453–461, 2019.

[15] X. Xia, F. Chen, Q. He, J. Grundy, M. Abdelrazek, and H. Jin, "Online collaborative data caching in edge computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 2, pp. 281–294, Feb. 2021.

[16] Q. He, Z. Dong, F. Chen, S. Deng, W. Liang, and Y. Yang, "Pyramid: Enabling hierarchical neural networks with edge computing," in *Proc. ACM Web Conf.*, 2022, pp. 1860–1870.

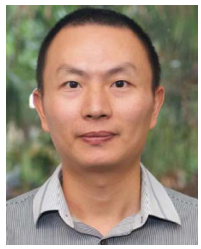
[17] D. F. Aranha, C. P. L. Gouvêa, T. Markmann, R. S. Wahby, and K. Liao, RELIC is an efficient library for cryptography. [Online]. Available: https://github.com/relic-toolkit/relic

[18] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proc. 13th Conf. Comput. Commun. Secur.*, A. Juels, R. N. Wright, and S. De Capitani di Vimercati, Eds., Alexandria, VA, USA: ACM Press, Oct. 30–Nov. 3, 2006, pp. 89–98.
[19] R. Ostrovsky, A. Sahai, and B. Waters, "Attribute-based encryption with non-monotonic access structures," in *Proc. 14th Conf. Comput. Commun. Secur.*, P. Ning, S. De Capitani di Vimercati, and P. F. Syverson, Eds., Alexandria, VA, USA: ACM Press, Oct. 28–31, 2007, pp. 195–203.
[20] Y. Zhang, D. Zheng, X. Chen, J. Li, and H. Li, "Computationally efficient ciphertext-policy attribute-based encryption with constant-size ciphertexts," in *Proc. 8th Int. Conf. Provable Secur.*, ser. Lecture Notes in Computer Science, S. S. M. Chow, J. K. Liu, L. C. K. Hui, and S.-M. Yiu, Eds., Hong Kong, China: Springer, Oct. 9–10, 2014, pp. 259–273.
[21] Q. M. Malluhi, A. Shikfa, and V. C. Trinh, "A ciphertext-policy attribute-based encryption scheme with optimized ciphertext size and fast decryption," in *Proc. 12th ACM Symp. Inf. Comput. Commun. Secur.*, R. Karri, O. Sinanoglu, A.-R. Sadeghi, and X. Yi, Eds., Abu Dhabi, UAE: ACM Press, Apr. 2–6, 2017, pp. 230–240.
[22] V. Goyal, A. Jain, O. Pandey, and A. Sahai, "Bounded ciphertext policy attribute based encryption," in *Proc. 35th Int. Colloq. Automata Lang. Program. Part II*, ser. Lecture Notes in Computer Science, L. Aceto, I. Damgård, L. A. Goldberg, M. M. Halldórsson, A. Ingólfsdóttir, and I. Walukiewicz, Eds., Reykjavik, Iceland: Springer, Jul. 7–11, 2008, pp. 579–591.
[23] X. Liang, Z. Cao, H. Lin, and D. Xing, "Provably secure and efficient bounded ciphertext policy attribute based encryption," in *Proc. 4th ACM Symp. Inf., Comput. Commun. Secur.*, W. Li, W. Susilo, U. K. Tupakula, R. Safavi-Naini, and V. Varadharajan, Eds., Sydney, Australia: ACM Press, Mar. 10–12, 2009, pp. 343–352.
[24] A. B. Lewko, T. Okamoto, A. Sahai, K. Takashima, and B. Waters, "Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption," in *Proc. Annu. Int. Conf. Theory Appl. Cryptographic Techn.*, ser. Lecture Notes in Computer Science, H. Gilbert, Ed., French Riviera: Springer, May 30–Jun. 3, 2010, pp. 62–91.
[25] J. Herranz, F. Laguillaumie, and C. Ràfols, "Constant size ciphertexts in threshold attribute-based encryption," in *Proc. 13th Int. Conf. Theory Pract. Public Key Cryptogr.*, ser. Lecture Notes in Computer Science, P. Q. Nguyen and D. Pointcheval, Eds., Paris, France: Springer, May 26–28, 2010, pp. 19–34.
[26] W. Susilo, G. Yang, F. Guo, and Q. Huang, "Constant-size ciphertexts in threshold attribute-based encryption without dummy attributes," *Inf. Sci.*, vol. 429, pp. 349–360, 2018.
[27] W. Ding et al., "An extended framework of privacy-preserving computation with flexible access control," *IEEE Trans. Netw. Service Manag.*, vol. 17, no. 2, pp. 918–930, Jun. 2020.
[28] S. Lin, R. Zhang, H. Ma, and M. Wang, "Revisiting attribute-based encryption with verifiable outsourced decryption," *IEEE Trans. Inf. Forensics Secur.*, vol. 10, no. 10, pp. 2119–2130, Oct. 2015.
[29] X. Mao, J. Lai, Q. Mei, K. Chen, and J. Weng, "Generic and efficient constructions of attribute-based encryption with verifiable outsourced decryption," *IEEE Trans. Dependable Secure Comput.*, vol. 13, no. 5, pp. 533–546, Sep./Oct. 2016.
[30] J. Xu, Q. Wen, W. Li, and Z. Jin, "Circuit ciphertext-policy attribute-based hybrid encryption with verifiable delegation in cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 1, pp. 119–129, Jan. 2016.
[31] J. Li, Y. Wang, Y. Zhang, and J. Han, "Full verifiability for outsourced decryption in attribute based encryption," *IEEE Trans. Serv. Comput.*, vol. 13, no. 3, pp. 478–487, May/Jun. 2020.
[32] L. Yuan et al., "CSEdge: Enabling collaborative edge storage for multi-access edge computing based on blockchain," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 8, pp. 1873–1887, Aug. 2022.
[33] D. Cash, E. Kiltz, and V. Shoup, "The twin Diffie-Hellman problem and applications," in *Proc. Annu. Int. Conf. Theory Appl. Cryptographic Techn.*, ser. Lecture Notes in Computer Science, N. P. Smart, Ed., Istanbul, Turkey: Springer, Apr. 13–17, 2008, pp. 127–145.
[34] P. S. L. M. Barreto and M. Naehrig, "Pairing-friendly elliptic curves of prime order," in *Proc. 12th Annu. Int. Workshop Sel. Areas Cryptogr.*, ser. Lecture Notes in Computer Science, B. Preneel and S. Tavares, Eds., Kingston, ON, Canada: Springer, Aug. 11–12, 2006, pp. 319–331.



Yudi Zhang received the master's degree from Hubei University of Technology, China, in 2017 and the PhD degree from Wuhan University, China, in 2020. He is currently a post-doctor with the School of Computing and Information Technology, University of Wollongong, Australia. His main research interests include cryptography and information security, in particular, cryptographic protocols.

Willy Susilo (Fellow, IEEE) received the PhD degree in computer science from the University of Wollongong, Australia. He is currently a distinguished professor and the head of the School of Computing and Information Technology and the director of the Institute of Cybersecurity and Cryptology, University of Wollongong. Recently, he has been awarded a prestigious Australian Laureate Fellowship. He has published more than 400 research papers in the area of cybersecurity and cryptology. His main research interests include cybersecurity, cryptography, and information security. He was a recipient of the prestigious Australian Research Council (ARC) Future fellow by the ARC and the researcher of the Year Award by the University of Wollongong in 2016. He is the editor-in-chief of the *Elsevier's Computer Standards and Interfaces* and *MDPI's Information Journal*. He has served as a program committee member in dozens of international conferences. He is currently serving as an associate editor in several international journals, including the *IEEE Transactions on Dependable and Secure Computing* and the *International Journal of Information Security* (Springer). His work has been cited more than 19,000 times in Google Scholar. He is also a fellow of the Australian Computer Society (ACS).



Fuchun Guo received the PhD degree from University of Wollongong, Australia, in 2013. He is currently a senior lecturer of the Institute of Cybersecurity and Cryptology, University of Wollongong. Fuchun received the prestigious Australian Research Council DECRA Fellowship award in 2017. In 2023, he has been awarded a prestigious ARC Future Fellowship. His primary research interest is the public-key cryptography; in particular, protocols, encryption and signature schemes, and security proof.

Guomin Yang (Senior Member, IEEE) received the PhD degree from the Computer Science Department, City University of Hong Kong in 2009. Previously, he worked as a research scientist with the Temasek Laboratories at National University of Singapore. He is currently an associate professor with the School of Computing and Information Systems, Singapore Management University, Singapore. Prior to this appointment, he was an Associate Professor with the University of Wollongong, Australia. He was awarded a prestigious Australian Research Council DECRA fellowship award. His research interests are cryptography and network security.