IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 5 June 2024, accepted 21 June 2024, date of publication 24 June 2024, date of current version 9 July 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3418886

RESEARCH ARTICLE

Attribute-Based Searchable Encryption With Forward Security for Cloud-Assisted IoT

DILXAT GHOPUR
School of Cyber Engineering, Xidian University, Xi'an 710071, China
e-mail: bayawan2012@163.com

ABSTRACT Ciphertext-Policy Attribute-Based Searchable Encryption (CP-ABSE) is one of the most suitable encryption mechanisms in cloud environments for its fine-grained access structure and keyword retrieval capability over the ciphertext. However, in the CP-ABSE schemes, guaranteeing the forward security of the outsourced cloud data and securely deleting those no longer needed data without relying on the cloud are challenging problems. To handle such challenges, we propose a Puncturable CP-ABSE (Pun-CP-ABSE) scheme that achieves self-controlled data deletion with a fine-grained access structure under the searchable mechanism. The data owner punctures the trapdoor to accomplish the data deletion. Then, the deletion process does not need to communicate with a trusted third party and can guarantee forward security. After the puncturation, the cloud server can no longer search for the corresponding ciphertext. Furthermore, we prove the Pun-CP-ABSE scheme is secure against the Chosen-Plaintext Attack (CPA) and Chosen-Keyword Attack (CKA). We have also implemented the Pun-CP-ABSE scheme to show its efficiency and feasibility.

INDEX TERMS Attribute-based encryption, puncturable encryption, searchable encryption, self-controlled data deletion, forward security.

I. INTRODUCTION

The pervasive adoption of Internet of Things (IoT) technology has propelled the widespread integration of IoT devices into various facets of individuals' daily lives, such as smart homes, intelligent healthcare, smart industries, intelligent transportation, and other fields [1], [2], [3], [4]. The extensive employment of IoT devices brings a tremendous volume of data. The analysis and prediction of massive IoT data can contribute to developing high-quality health treatment, disaster prediction, service improvement, and efficient management. However, IoT devices' storage and processing capabilities are very limited. Therefore, the cloud-assisted IoT technology [5], [6] has emerged, which takes advantage of cloud service's powerful computing and storage resources at a lower cost. Furthermore, the encryption-before-outsourcing method has been used in the semi-honest cloud environment to guarantee the security of the outsourced data generated by IoT devices. However, this method hinders the searchability of the encrypted data. The Searchable Encryption (SE) [7], [8], [9] ingeniously achieves the retrieve function over ciphertext. Furthermore, efficient access control over tremendous amounts of encrypted data in the cloud is another tricky problem. Fortunately, Ciphertext-Policy Attribute-Based Searchable Encryption (CP-ABSE) [10], [11], [12], [13] effectively solves this issue with three features, ie., the one-to many encryption method, fine-grained access policy and searchability, which allows the data owner to manage, search, and decrypt the ciphertext efficiently. As shown in Fig. 1, the data collected by IoT devices is encypted as ciphertexts with corresponding indexes before uploading the to the cloud. Whenever the data owner accesses the data, he sends the trapdoor to the cloud server. Then, the cloud server returns the search result if the attributes in the trapdoor satisfy the access structure and the keyword matches the indexes.

However, CP-ABSE implemented in cloud-based IoT environments still suffers from security problems. For example, some sensitive data after access or when no longer needed should be securely deleted from the cloud reliably and

The associate editor coordinating the review of this manuscript and approving it for publication was Martin Reisslein.

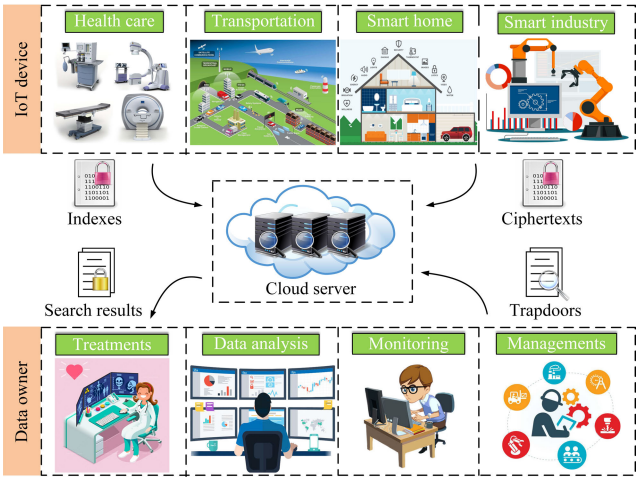


FIGURE 1. CP-ABSE in the cloud-based IoT scenario.

flexibly. If the data is deleted by the cloud, the cloud may back up the sensitive data for profit. Therefore, implementing secure data deletion without relying on the cloud and ensuring that deleted data no longer appears is a challenging problem. The best solution for this problem is regularly updating the trapdoor to withdraw the search ability for ciphertexts. This way, data deletion was successfully achieved without depending on the cloud.

To the best of our knowledge, the data deletion was widely studied previously, and several corresponding data deletion schemes were proposed. For example, Green and Miers [14] first proposed Puncturable Encryption (PE) to achieve data deletion in public key encryption. In their work, the ciphertext contains a set of labels, and data deletion is performed by puncturing the secret key with a specific negative label. If the negative label could match labels in the ciphertext, then the secret key would lose the decryption capability. Phuong et al. [15] first proposed a puncturable ABE (Pun-ABE) to implement data deletion in the ABE system. Sun et al. [16] first combined Searchable Symmetric Encryption (SSE) with PE to propose a Pun-SSE scheme. In their scheme, puncture method was applied to achieve data deletion in the SSE system. After the puncturation, the cloud could not retrieve the deleted data.

However, the problem of data deletion has yet to be solved in the CP-ABSE scheme. To handle this problem, we combine PE [14] with CP-ABE [17] and add a search function to propose a new searchable encryption mechanism, Pun-CP-ABSE, which achieves forward secure data deletion in CP-ABSE. In our scheme, the data owner generates two kinds of trapdoors named general trapdoor and puncturable trapdoor, respectively, for searching and securely deleting the outsourced data. The general trapdoor possesses the attributes to achieve access control. The puncturable trapdoor is generated by the original secret key of PE that can achieve fine-grained data deletion executed by the puncture algorithm. After the puncturable trapdoor is punctured with specific tags, the cloud server will lose searchability toward the ciphertext, which contains the same tags as in the puncturable trapdoor. The main contributions of the Pun-CP-ABSE are presented as follows:

1) We combine PE with CP-ABE to propose a Pun-CP-ABSE scheme that achieves precisely, permanently, fine-grandly, and self-controlled data deletion in a searchable CP-ABE mechanism. In our scheme, the data deletion process does not relay to trusted third parties such that the communication overhead could be omitted. And the forward security is guaranteed at the same time.
2) The Pun-CP-ABSE scheme achieves fine-grained access control over the ciphertext with a keyword search, which enables the data owner to retrieve the ciphertext corresponding to the keywords if his attributes match the access structure.
3) We prove that the Pun-CP-ABSE scheme is secure under the Chosen-Plaintext Attack (CPA) and Chosen Keyword Attack (CKA). Furthermore, we present the efficiency and practicability of the Pun-CP-ABSE scheme by implementing simulations.

II. RELATED WORK

A. ATTRIBUTE-BASED ENCRYPTION

Sahai et al. [18] proposed an Attribute-Based Encryption (ABE) scheme. According to the position of the access structure, the ABE scheme is separated into two types: Key-Policy ABE (KP-ABE) [19] and Ciphertext-Policy ABE (CP-ABE) [20]. The fine-grained access control ability of ABE makes it suitable for widely use in cloud environment. Consequently, the ABE scheme draws significant attention from researchers, thus developing several new research fields. Notable examples include Online/Offline ABE (OO-ABE) [21], Outsourced Decryption ABE (OD-ABE) [3], Revocable ABE (RABE) [22], Multi-Authority ABE (MA-ABE) [23], Policy Hidden ABE (PH-ABE) [24], and various others. In the OO-ABE scheme, the data owner pre-computed some parts of the encryption using the proxy server before knowing the access structure. Then, the encryption is completed with less computation with the smart device. In the OD-ABE scheme, the ciphertext was partially decrypted by the proxy server using the transformation key. In this way, it reduced the user's decryption burden. The RABE scheme mainly discusses withdrawing user access rights according to permission changes. The MA-ABE scheme was proposed to enhance the security and efficiency of the original ABE. The PH-ABE scheme mainly studied preventing information leakage from the access policy. However, these schemes did not consider the searchable function over the encrypted data in ABE scheme.

B. SEARCHABLE ENCRYPTION

Song et al. [7] first proposed the SSE to achieve retrieval function over the ciphertext. Although their scheme possesses highly efficient searchability, it suffers from poor key management. Therefore, Boneh et al. [25] proposed Public-key

Encryption with Keyword Search (PEKS) scheme. Subsequently, various PEKS schemes have been proposed, for example, ranked-keyword search [26], fuzzy-keyword search [27], multi-keyword search [28], and verifiable-keyword search [10]. However, these searchable schemes lack fine-grained access control. For this, Zheng et al. [29] first described the CP-ABSE scheme's construction. This scheme achieves keyword searchability over the ciphertext and is proved secure against CKA. Unfortunately, this scheme only achieves a single keyword search. Yu et al. [30] introduced a multi-authority CP-ABSE to achieve user revocation and effectively decrease the calculation amount of the decryption phase. Furthermore, this scheme can resist CPA and CKA. Miao et al. [31] proposed a multi-authority CP-ABSE. In their work, they trace the malicious attribute authority that incorrectly produces the user's secret keys, and revokes the malicious attribute authority. Wang et al. [9] proposed a multi-owner and multi-user single-key searchable ABSE scheme. In their scheme, they efficiently achieve user-level revocation and give proof of resisting CKA and Keyword Guessing Attack (KGA). Li et al. [32] proposed a Key-Policy Attribute-Based Searchable Encryption (KP-ABSE) scheme. In their scheme, they use an outsourced decryption mechanism to reduce the computation burden of the users in the decryption and security proof of CPA. Miao et al. [33] introduced a multi-owner setting CP-ABSE. In their work, they hide the access structure to prevent leaking sensitive information and trace the malicious user. Chen et al. [34] introduced a flexible retrieval CP-ABSE. In their work, they achieve a multi-keyword ranked retrieve to enhance the search efficiency of the CP-ABSE. Furthermore, their scheme can resist CKA and KGA. Although these schemes mentioned above have studied the CP-ABSE in many aspects, they did not consider the function of secure data deletion in the CP-ABSE scheme.

C. DATA DELETION

Geambasu et al. [35] first proposed a data self-destructing symmetric encryption. In their work, the data is encrypted by symmetric encryption. Then, the secret sharing protocol is used to decompose the key into n key elements and deliver them to a large-scale Distributed Hash Table (DHT) network. DHT network nodes would periodically and automatically delete key components, resulting in achieving self-destructing data deletion. Xiong et al. [36] introduced a data self-destructing KP-ABE. In their work, data is encrypted with a time interval, and the secret key is produced with a time instant. When the time instantaneously exceeds the interval, the secret key cannot decrypt the ciphertext. In this way, they achieved self-destructing data deletion. Yu et al. [37] achieved a data deletion in CP-ABE for a fog-based cloud environment. In their scheme, all users' secret keys commonly possess one particular "dummy" attribute. When data needs to be deleted, the fog device changes the ciphertext's access structure (i.e., updates the "dummy" attribute). In this way, the system user cannot decrypt the data anymore. After the data deletion, the correctness of the deletion process could be verified by the signature method. Xue et al. [38] achieved a data deletion in KP-ABE. They updated the ciphertext and secret key to accomplish the data deletion and use Merkle Hash Tree (MHT) to verify the data deletion process.

However, previous CP-ABSE schemes have not considered the data deletion function yet. To compensate for this space, we combine PE with CP-ABE to achieve a fine-grained self-controlled Pun-CP-ABSE. The functions comparison is presented in Table 1.

III. PRELIMINARIES

A. LAGRANGE POLYNOMIAL INTERPOLATION

The polynomial of degree d can be reconstructed by the different $d+1$ coordinates $(a_0, b_0), (a_1, b_1), \ldots, (a_d, b_d)$ as follows:

$$q(a) = L_d(a) = \sum_{j=0}^{d} b_j \Delta_j(x),$$

where

$$\Delta_k(a) = \prod_{0 \le k, k \ne i \le d} \frac{a - a_i}{a_k - a_i}.$$

B. ACCESS STRUCTURE

Let $P = \{\psi_1, \psi_2, \ldots, \psi_n\}$ denotes a participants set. The collection $\mathbb{W} \subseteq 2^{\{\psi_1, \psi_2, \ldots, \psi_n\}}$ is monotone if for all random sets F, R: if $F \in \mathbb{W}$ and $F \subseteq R$, then $R \in \mathbb{W}$. An access structure is a set $\mathbb{W}$ of non-empty subsets of $\{\psi_1, \psi_2, \ldots, \psi_n\}$. The sets in $\mathbb{W}$ are considered authorised. Otherwise, the sets are considered not authorised.

C. LINEAR SECRET-SHARING SCHEMES (LSSS)

Let $\mathbb{W}$ be an access strategy on P, then there is a LSSS access structure $(\mathbb{A}, \rho)$ where $\mathbb{A}$ is a $\ell \times n$ matrix, and ρ is a function that associates each row of $\mathbb{A}$ to party. When the secret value s is shared, the vector $\vec{u}$ can be chosen as $\vec{u} = (s, y_2, \ldots, y_n) \in \mathbb{Z}_p^n$ and the value $\pi = \mathbb{A} \cdot \vec{u}$ is calculated. Assuming that S is an authorized group described by $(\mathbb{A}, \rho)$, and I is the set of rows in matrix $\mathbb{A}$ such that $I \subset \{1, 2, \ldots, \ell\}$, and defined as $I = \{i : \rho(i) \in S\}$. Then, there is a constant number set $\{\omega_i \in \mathbb{Z}_p\}$ that satisfies the $\sum_{i \in I} \omega_i \pi_i = s$, where $\pi_i = (\mathbb{A} \cdot \vec{u})_i$.

D. DECISIONAL BILINEAR DIFFIE-HELLMAN (DBDH) ASSUMPTION

Assuming that x_1, x_2, x_3, x_4 are random values picked from $\mathbb{Z}_p$ and g is a generater of G_0. Let $X_1 = g^{x_1}$, $X_2 = g^{x_2}$ and $X_3 = g^{x_3}$. Then there is no Probabilistic Polynomial Time (PPT) algorithm $\mathcal{A}$ can distinguish between $e(g, g)^{x_1 x_2 x_3}$ and $e(g, g)^{x_4}$ more than a negligible advantage ϵ as:

$$\left| \begin{array}{l} Pr[\mathcal{A}(X_1, X_2, X_3, e(g, g)^{x_1 x_2 x_3}) = 0] \\ -Pr[\mathcal{A}(X_1, X_2, X_3, e(g, g)^{x_4}) = 0] \end{array} \right| \le \epsilon.$$

TABLE 1. Function comparison.

Schemes	Scheme type	Fine-grained access control	Keyword search	Data deletion	Security model
Green [14]	PE	✗	✗	✓	CPA
Sun [16]	Pun-SSE	✗	✓	✓	-
Boneh [25]	PEKE	✗	✓	✗	CPA
Bethencourt [20]	CP-ABE	✓	✗	✗	CPA
Phuong [15]	Pun-ABE	✓	✗	✓	CPA
Miao [31]	CP-ABSE	✓	✓	✗	CPA, CKA
Our	Pun-CP-ABSE	✓	✓	✓	CPA, CKA

IV. SYSTEM FORMULATION

A. SYSTEM AND THREAT MODELS

As shown in Fig.2, the system model of the Pun-CP-ABSE consists of the following four entities: Cloud Service (**CS**), Key Generation Center (**KGC**), IoT devices (**Dev**), and the Data Owner (**DO**).

1) **Key Generation Center**. **KGC** generates public and private keys in the system.
2) **Cloud Service**. **CS** provides data storage and retrieval services. When **DO** issues a search query using trapdoor, the **CS** will return the result if **DO**'s attributes satisfy the access strategy, and the keywords in trapdoor can match the index in ciphertext, while the punctured tags in trapdoor doesn't match the labels in ciphertext.
3) **IoT devices**. **Dev** is a series of IoT devices (e.g., smart home device, smart clinic device, or smart industry device) that are managed by the **DO**. They are responsible for generating or collecting the data. **Dev** encrypts the data using some tags and keywords with access structure and sends the ciphertext to the **CS**.
4) **Data Owner**. **DO** is the owner of the data stored in the cloud collected by IoT devices, such as a smart grid manager, a smart homeowner, or a patient who uses health care devices. **DO** is responsible for managing, searching, accessing, and deleting the data in the **CS**. The **DO** generates the trapdoor to search for corresponding data. If the attributes in the trapdoor can match the access structure, and the keyword can match the index, the **DO** can search and decrypt the corresponding ciphertext. Furthermore, **DO** executes the puncture algorithm using some tags to permanently and precisely delete the specific data.

Considering the threat model above the Pun-CP-ABSE scheme, **CS** is a semi-honest entity that honestly executes the assignment tasks (i.e., storage and search corresponding ciphertext) and tries to gain information about storage data or the keyword from the trapdoors generated by the **DO**. We assume **KGC**, **Dev**, and **DO** are the fully trusted entity that honestly performs their tasks.

B. PUN-CP-ABSE SYSTEM DEFINITION

The Pun-CP-ABSE scheme consists of seven algorithms, namely: Setup, Encrypt, KeyGen, TrapdoorGen, Puncture, Search, Decrypt. The interactions of each process are presented in Fig.3.

1) $\mathsf{Setup}(1^\kappa, U, d) \rightarrow (\mathrm{PK_{KGC}}, \mathrm{MK_{KGC}}), (\mathrm{PK_{CS}}, \mathrm{MK_{CS}})$: Given the security parameter κ, number of tags d and system attributes U as input. **KGC**, **CS** generate corresponding public and private key pairs $(\mathrm{PK_{KGC}}, \mathrm{MK_{KGC}})$ and $(\mathrm{PK_{CS}}, \mathrm{MK_{CS}})$, respectively.
2) $\mathsf{Encrypt}(\mathcal{M}, W', \mathrm{PK_{KGC}}, \mathrm{T}, (\mathbb{A}, \rho)) \rightarrow \mathrm{CT}$: Given the public key $\mathrm{PK_{KGC}}$, labels $\mathrm{T} = (t_1, t_2, \ldots, t_d)$, LSSS access structure $(\mathbb{A}, \rho)$, where $\mathbb{A}$ is a $\ell * n$ matrix, and ρ is a function that associates each row of $\mathbb{A}$ to attributes, and message $\mathcal{M}$. **Dev** generates a ciphertext CT associated with keyword W′.
3) $\mathsf{KeyGen}(\mathrm{MK_{KGC}}, \mathrm{PK_{KGC}}, \mathrm{S}) \rightarrow (\mathrm{SK}, \mathrm{PSK})$: Given the public and private key $\mathrm{MK_{KGC}}$,$\mathrm{PK_{KGC}}$, and an attribute set S. **KGC** generates the secret key SK and original puncturable key PSK.
4) $\mathsf{TrapdoorGen}(\mathrm{PK_{CS}}, \mathrm{SK}, \mathrm{PSK}, W) \rightarrow (\widehat{\mathrm{T}}_W, \widetilde{\mathrm{T}}_W^{(0)})$: Given the public key $\mathrm{PK_{CS}}$, secret key SK, original puncturable key PSK, and keyword W. **DO** generates the general trapdoor $\widehat{\mathrm{T}}_W$ and puncturable trapdoor $\widetilde{\mathrm{T}}_W^{(0)}$.
5) $\mathsf{Puncture}(\mathrm{PK_{KGC}}, \widetilde{\mathrm{T}}_W^{(k-1)}, t) \rightarrow \widetilde{\mathrm{T}}_W^{(k)}$: Given the public key $\mathrm{PK_{KGC}}$, puncturable trapdoor $\widetilde{\mathrm{T}}_W^{(k-1)}$, and label t. **DO** generate a new puncturable trapdoor $\widetilde{\mathrm{T}}_W^{(k)}$.
6) $\mathsf{Search}(\mathrm{MK_{CS}}, \widehat{\mathrm{T}}_W, \widetilde{\mathrm{T}}_W^{(k)}, \mathrm{CT}) \rightarrow (\widetilde{C}, C', A)/\perp$: Given the private key $\mathrm{MK_C}$, general trapdoor $\widehat{\mathrm{T}}_W$, puncturable trapdoor $\widetilde{\mathrm{T}}_W^{(k)}$, and ciphertext CT associated with the keyword W′. If the attributes in the $\widehat{\mathrm{T}}_W$ can satisfy the access strategy, and the tags in the $\widetilde{\mathrm{T}}_W^{(k)}$ do not match the labels in the ciphertext CT, while the W of $(\widehat{\mathrm{T}}_W, \widetilde{\mathrm{T}}_W^{(k)})$ matches the W' of CT, then **CS** returns the search result and ciphertext components $(\widetilde{C}, C', A)$ to **DO**. Otherwise, **CS** returns $\perp$.
7) $\mathsf{Decrypt}(\mathrm{SK}, \mathrm{W}, A, C', \widetilde{C}) \rightarrow \mathcal{M}$: After receiving the search result and ciphertext components (A,C',$\widetilde{C}$), **DO** uses the secret key SK and corresponding keyword W to obtain the plaintext $\mathcal{M}$.

C. SECURITY MODELS OF PUN-CP-ABSE

For the security of Pun-CP-ABSE, we mainly consider two kinds of attacks: CPA and CKA, which are proceeded by the adversary $\mathcal{A}$ and the challenger $\mathcal{C}$. The formal definitions

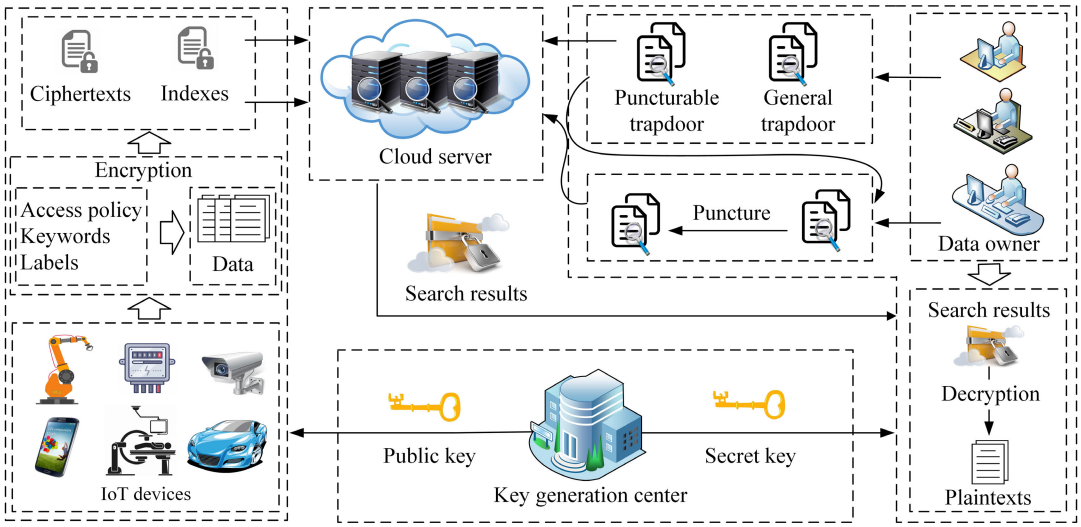


FIGURE 2. System model of the Pun-CP-ABSE.

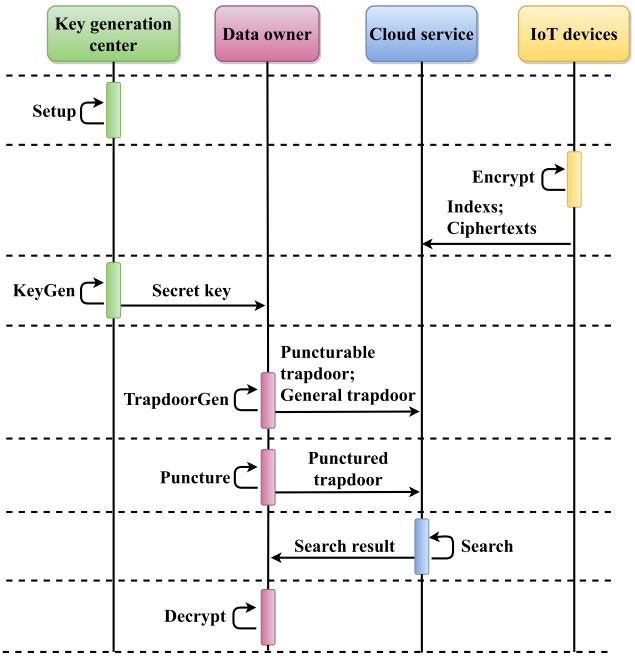


FIGURE 3. System flow of the Pun-CP-ABSE.

of the CPA and CKA are given as **Game 1** and **Game 2**, respectively, as follows:

■ **Game 1**

1) **Init**. The adversary $\mathcal{A}$ sends the access policy $(\mathbb{A}^*, \rho)$ with the label set $\{t_1^*, \ldots, t_d^*\}$ to the C.
2) **Setup**. C calls the $\mathsf{Setup}(1^\kappa, U, d)$ to produce the public keys, then gives them to the $\mathcal{A}$.
3) **Query Phase 1**. $\mathcal{A}$ issues for attribute set S, where S does not match the access policy $(\mathbb{A}^*, \rho)$. The challenger C calls $\mathsf{KeyGen}(\mathrm{PK_{KGC}}, \mathrm{MK_{KGC}}, \mathrm{S}) \rightarrow (\mathrm{SK}, \mathrm{PSK})$.
4) **Challenge**. $\mathcal{A}$ delivers two same-length messages $\mathcal{M}_0$ and $\mathcal{M}_1$ to the C. Then C picks random bit $\vartheta \in \{0, 1\}$ to generate the ciphertext CT_ϑ, and sends it to $\mathcal{A}$.
5) **Query Phase 2**. This phase is to repeat the Phase 1.
6) **Guess**. $\mathcal{A}$ outputs a guess ϑ' of ϑ. If $\vartheta' = \vartheta$, $\mathcal{A}$ wins the game.

The advantage of $\mathcal{A}$ is defined as $Adv_{\mathcal{A}}^{\mathrm{CPA}} = |Pr[\vartheta' = \vartheta] - \frac{1}{2}|$.

Definition 1: The Pun-CP-ABSE scheme achieves security against CPA as long as no PPT adversary possesses a significant advantage in the aforementioned security game.

■ **Game 2**

1) **Init**. $\mathcal{A}$ declares the challenging access policy and the set of tags as $((\mathbb{A}^*, \rho), \{t_1^*, \ldots, t_d^*\})$.
2) **Setup**. The C calls the $\mathsf{Setup}(1^\kappa, U, d)$ to generate the public keys and delivers them to the $\mathcal{A}$. Then C selects an empty set R and a counter $k = 0$.
3) **Query Phase 1**. $\mathcal{A}$ repeatedly issues the followed four kinds of queries and receives responses from challenger:
   - **Secret key query**: $\mathcal{A}$ queries for the attribute set S, where S does not match the $(\mathbb{A}^*, \rho)$. C executes $\mathsf{KeyGen}(\mathrm{PK_{KGC}}, \mathrm{MK_{KGC}}, \mathrm{S}) \rightarrow (\mathrm{SK}, \mathrm{PSK})$ to generates the secret key SK and puncturable key PSK.
   - **Trapdoor-query**: $\mathcal{A}$ submits attribute set S and keyword W to C. C sends general trapdoor $\widehat{\mathrm{T}}_W$ and puncturable trapdoor $\widetilde{\mathrm{T}}_W^{(0)}$ to the $\mathcal{A}$.
   - **Puncture**: $\mathcal{A}$ increases the k, and calls $\mathsf{Puncture}$ algorithm to generate new puncturable trapdoor $\widetilde{\mathrm{T}}_W^{(k)}$, then puts the label t to the set R.
   - **Corrupt**: C returns the most recent puncturable trapdoor $\widetilde{\mathrm{T}}_W^{(k)}$ to the $\mathcal{A}$ when this query is issued for the first time. For the subsequent queries, C returns $\perp$.

TABLE 2. Notations used in Pun-CP-ABSE algorithm.

Notations	Descriptions.
$(\mathrm{PK_{KGC}}, \mathrm{MK_{KGC}})$	public and private key pairs of KGC.
$(\mathrm{PK_{CS}}, \mathrm{MK_{CS}})$	public and private key pairs of CS.
U	number of attributes in the system.
d	number of tags
$\mathrm{T} = (t_1, t_2, ..., t_d)$	tag set
$(\mathbb{A}, \rho)$	access policy
W'	keyword in the ciphertext
W	keyword in the trapdoor
$\mathcal{M}$	plaintext
CT	ciphertext
SK	secret key
PSK	puncturable secret key
S	attribute set in the secret key
$\widehat{\mathrm{T}}_W = (\hat{T}_1, \hat{T}_2, \hat{T}_{3,x}, \hat{T}_4)$	general trapdoor
$\widetilde{\mathrm{T}}_0^{(0)} = (T_{0,1}^{(0)}, T_{0,2}^{(0)}, T_{0,3}^{(0)}, T_{0,4}^{(0)})$	initial puncturable trapdoor
$\widetilde{\mathrm{T}}_W^{(k)} = [\widetilde{\mathrm{T}}_k^{(0)}, \widetilde{\mathrm{T}}^{(1)}, ..., \widetilde{\mathrm{T}}^{(k)}]$	puncturable trapdoor after the k-th punctuation
$\widetilde{\mathrm{T}}_k^{(0)} = (T_{k,1}^{(0)}, T_{k,2}^{(0)}, T_{k,3}^{(0)}, T_{k,4}^{(0)})$	the first component of the $\widetilde{\mathrm{T}}_W^{(k)}$
$\widetilde{\mathrm{T}}^{(k)} = (T_{k,1}^{(k)}, T_{k,2}^{(k)}, T_{k,3}^{(k)}, T_{k,4}^{(k)})$	the k-th component of the $\widetilde{\mathrm{T}}_W^{(k)}$

4) **Challenge**. $\mathcal{A}$ submits two exact size keywords, W_0 and W_1, which were not issued previously to C. Then, C randomly picks bit $\vartheta \in \{0, 1\}$ for generating the index C_{W_ϑ} and return it to $\mathcal{A}$.
5) **Query Phase 2**. Phase 2 entails the same requirements as Phase 1, but requires for an additional condition that the two keywords must not have been issued beforehand.
6) **Guess**. $\mathcal{A}$ outputs a guess $\vartheta' \in \{0, 1\}$. If $\vartheta' = \vartheta$, $\mathcal{A}$ wins the game. The advantage of $\mathcal{A}$ is denoted as $Adv_{\mathcal{A}}^{\mathrm{CKA}} = |Pr[\vartheta' = \vartheta] - \frac{1}{2}|$.

Definition 2: The security of the Pun-CP-ABSE scheme against CKA is ensured if an adversary with PPT can only achieve a negligible advantage when attempting to break the security game mentioned above.

V. PUN-CP-ABSE SCHEME

A. CONCRETE CONSTRUCTION OF PUN-CP-ABSE

In this section, we describe the detailed processes of the Setup, Encrypt, KeyGen, Puncture, TrapdoorGen, Search and Decrypt algorithms. The notations used in this scheme are shown in TABLE 2.

1) $\mathrm{Setup}(1^\kappa, U, d) \to (\mathrm{PK_{KGC}}, \mathrm{MK_{KGC}}), (\mathrm{PK_{CS}}, \mathrm{MK_{CS}})$: First, **KGC** selects two bilinear groups G_0 and G_1 that the generator is g with prime order p. Then, it defines the bilinear map $e : G_0 \times G_0 \to G_1$, and two hash functions $H_1 : \{0,1\}^* \to \mathbb{Z}_p$, $H_2 : G_1 \to \{0,1\}^{logp}$, respectively. Next, **KGC** randomly selects group elements $\{h_i\}_{1 \le i \le |U|} \in G_0$ that corresponding to the system attributes U, and randomly chooses exponents $\alpha, a, \beta \in \mathbb{Z}_p$. Then **KGC** sets a polynomial $q(x)$ with the condition of $q(0) = \beta$, and defines $Q(x) = g^{q(x)}$. Finally, **KGC** computes the public values $\{g^{q(j)}\}_{1 \le j \le d}$. Note that t_0 is a spatial label not used in normal operations. Output:

$$\mathrm{PK_{KGC}} = \Big(g, g_1 = g^a, g_2 = g^\beta, e(g,g)^\alpha, \{h_i\}_{1 \le i \le |U|}, \{g^{q(j)}\}_{1 \le j \le d}, t_0\Big),$$
$$\mathrm{MK_{KGC}} = \alpha. \tag{1}$$

The **CS** randomly selects a private key $\mathrm{MK_{CS}} = x_c \in \mathbb{Z}_p$. Then, its public key is $\mathrm{PK_{CS}} = g^{x_c}$.

Notes: Given the public key $\mathrm{PK_{KGC}}$ parameters of $\Big(g_2 = Q(0), Q(1), \ldots, Q(d)\Big)$, then $Q(x)$ can be reconstructed by the Lagrange Polynomial Interpolation as follows.

$$Q(x) = g^{\sum_{j=0}^{d} q(j)\Delta_j(x)} = \prod_{j=0}^{d} g^{q(j)^{\Delta_j(x)}} = \prod_{j=0}^{d} Q(j)^{\Delta_j(x)}. \tag{2}$$

2) $\mathrm{Encrypt}(\mathcal{M}, W', \mathrm{PK_{KGC}}, \mathrm{T}, (\mathbb{A}, \rho)) \to \mathrm{CT}$: **Dev** encrypts the data with keyword W' under the set of labels $\mathrm{T} = (t_1, t_2, \ldots, t_d)$ and the LSSS access structure $(\mathbb{A}, \rho)$, where $\mathbb{A}$ is a $\ell \times n$ matrix, and ρ is a function that associates each row of $\mathbb{A}$ to attributes. The **Dev** randomly selects vector $\vec{u} = (s, y_2, \ldots, y_n) \in \mathbb{Z}_p^n$. The secret value s is calculated as $\pi_i = \vec{u} \cdot \mathbb{A}_i$, where $i = 1$ to ℓ. Furthermore, **Dev** randomly selects exponents $\tilde{r}_1, \tilde{r}_2, \ldots, \tilde{r}_\ell \in \mathbb{Z}_p$, then outputs ciphertext as $\mathrm{CT} = \{\widetilde{C}, C_{W'}, C', \{C_{1,i}, C_{2,i}\}_{1 \le i \le \ell}, \{C_{3,j}\}_{1 \le j \le d}\}$, where

$$\widetilde{C} = \mathcal{M} \cdot e(g,g)^{\alpha s}, C' = g^s,$$
$$C_{W'} = H_2\Big(e(g,g)^{\alpha s H_1(W')}\Big),$$
$$C_{1,i} = g^{a\pi_i} h_{\rho(i)}^{-\tilde{r}_i}, C_{2,i} = g^{\tilde{r}_i}, C_{3,j} = Q(H_1(t_j))^s. \tag{3}$$

3) $\mathrm{KeyGen}(\mathrm{MK_{KGC}}, \mathrm{PK_{KGC}}, \mathrm{S}) \to (\mathrm{SK}, \mathrm{PSK})$: The **KGC** randomly selects exponents $\tau, r_0 \in \mathbb{Z}_p$, then uses public and private key pairs to generate secret key SK and puncturable secret key PSK as:

$$\mathrm{SK} = \Big\{\widetilde{D} = g^\alpha \cdot g^{a\tau}, D_1 = g^\alpha \cdot g^{\tau(a+\beta)}, D_2 = g^\tau, \forall x \in S : D_x = h_x^\tau\Big\},$$
$$\mathrm{PSK} = \Big\{psk_1 = g_2^{(\tau + r_0)}, psk_2 = Q(H_1(t_0))^{r_0}, psk_3 = g^{r_0}, psk_4 = t_0\Big\}. \tag{4}$$

4) $\mathrm{TrapdoorGen}(\mathrm{PK_{CS}}, \mathrm{SK}, \mathrm{PSK}, W) \to (\widehat{\mathrm{T}}_W, \widetilde{\mathrm{T}}_W^{(0)})$: The **DO** randomly selects exponents $\sigma \in \mathbb{Z}_p$ and outputs general trapdoor $\widehat{\mathrm{T}}_W = (\hat{T}_1, \hat{T}_2, \hat{T}_{3,x}, \hat{T}_4)$ and initial puncturable trapdoor $\widetilde{\mathrm{T}}_W^{(0)} = \Big[\widetilde{\mathrm{T}}_0^{(0)}\Big] = \Big[(T_{0,1}^{(0)}, T_{0,2}^{(0)}, T_{0,3}^{(0)}, T_{0,4}^{(0)})\Big]$, where

$$T_{0,1}^{(0)} = \{psk_1\}^{H_1(W)}, \quad \hat{T}_1 = D_1^{H_1(W)} \cdot \mathrm{PK_{CS}}^\sigma,$$
$$T_{0,2}^{(0)} = \{psk_2\}^{H_1(W)}, \quad \hat{T}_2 = D_2^{H_1(W)},$$
$$T_{0,3}^{(0)} = \{psk_3\}^{H_1(W)}, \quad \hat{T}_{3,x} = \{D_x^{H_1(W)}\}_{x \in S},$$
$$T_{0,4}^{(0)} = t_0, \quad \hat{T}_4 = g^\sigma. \tag{5}$$

5) $\mathrm{Puncture}(\mathrm{PK_{KGC}}, \widetilde{\mathrm{T}}_W^{(k-1)}, t) \to \widetilde{\mathrm{T}}_W^{(k)}$: The **DO** uses label t to puncture the existing puncturable trapdoor $\widetilde{\mathrm{T}}_W^{(k-1)}$ which denotes as $\Big[\widetilde{\mathrm{T}}_{k-1}^{(0)}, \widetilde{\mathrm{T}}^{(1)}, \ldots, \widetilde{\mathrm{T}}^{(k-1)}\Big]$. Further $\widetilde{\mathrm{T}}_{k-1}^{(0)}$ is expressed by $\Big(T_{k-1,1}^{(0)}, T_{k-1,2}^{(0)}, T_{k-1,3}^{(0)},$

$T^{(0)}_{k-1,4}\Big)$. The **DO** randomly selects exponents $\lambda_k, r_k, r'_k \in \mathbb{Z}_p$ and computes:

$$\widetilde{\mathrm{T}}^{(0)}_k = \Big(T^{(0)}_{k-1,1} \cdot g_2^{r_k-\lambda_k}, T^{(0)}_{k-1,2} \cdot Q(H_1(t_0))^{r_k}, T^{(0)}_{k-1,3} \cdot g^{r_k}, t_0\Big),$$
$$\widetilde{\mathrm{T}}^{(k)} = \Big(g_2^{\lambda_k+r'_k}, Q(H_1(t))^{r'_k}, g^{r'_k}, t\Big). \quad (6)$$

Then **DO** outputs: $\widetilde{\mathrm{T}}^{(k)}_W = \Big[\widetilde{\mathrm{T}}^{(0)}_k, \ldots, \widetilde{\mathrm{T}}^{(k-1)}, \widetilde{\mathrm{T}}^{(k)}\Big]$.

6) Search$(\mathrm{MK_{CS}}, \widehat{\mathrm{T}}_W, \widetilde{\mathrm{T}}^{(k)}_W, \mathrm{CT}) \to (E, \widetilde{C})/\perp$: First, **CS** computes $\hat{T}' = \hat{T}_1/\hat{T}_4^{\mathrm{MK_{CS}}} = D_1^{H_1(W)}$. If the attribute set S meets the access strategy $(\mathbb{A}, \rho)$, there is a constant number set $\{\omega_i \in \mathbb{Z}_p\}$ that satisfies the $\sum_{i\in I} \omega_i \pi_i = s$, where $I = \{i : \rho(i) \in \mathrm{S}\}$. Then **CS** computes:

$$\begin{aligned} A &= \prod_{i\in I} (e(C_{1,i}, \hat{T}_2) e(C'_{2,i}, \hat{T}_{3,i}))^{\omega_i} \\ &= \prod_{i\in I} (e(g^{a\pi_i} h_{\rho(i)}^{-\tilde{r}_i}, g^{\tau}) e(g^{\tilde{r}_i}, h_x^{\tau}))^{\omega_i} \\ &= \prod_{i\in I} e(g,g)^{a\tau\pi_i\omega_1 H_1(W)} \\ &= e(g,g)^{as\tau H_1(W)}. \end{aligned} \quad (7)$$

Next, **CS** computes:

$$\begin{aligned} B &= \frac{e(C', T')}{A} \\ &= \frac{e(g^s, g^{\alpha H_1(W)} \cdot g^{\tau(a+\beta)H_1(W)})}{e(g,g)^{as\tau H_1(W)}} \\ &= \frac{e(g,g)^{\alpha s H_1(W)} \cdot e(g,g)^{as\tau H_1(W)} \cdot e(g,g)^{\tau\beta s H_1(W)}}{e(g,g)^{as\tau H_1(W)}} \\ &= e(g,g)^{\alpha s H_1(W)} \cdot e(g,g)^{\tau\beta s H_1(W)}. \end{aligned} \quad (8)$$

For $j = 0, 1, \ldots, i$, puncturable trapdoor $\widetilde{\mathrm{T}}^{(j)}$, **CS** computes a set of coefficients $\{w_1, w_2, \ldots, w_d, w^*\}$ such that $\sum_{m=1}^{d}(w_m \cdot q(H_1(t_m))) + w^* \cdot q(H_1(T_4^{(i)})) = q(0) = \beta$. Then **CS** computes:

$$Z_i = \frac{e(T^{(i)}_{i,1}, C')}{e(T^{(i)}_{i,3}, \prod_{m=1}^{d}(C_{3,m})^{w_k}) \cdot e(T^{(i)}_{i,2}, C')^{w^*}}. \quad (9)$$

Furthermore, **CS** computes components Z as:

$$Z = \prod_{i=0}^{d} Z_i = e(g,g)^{\tau\beta s H_1(W)}. \quad (10)$$

Then, **CS** computes E as:

$$\begin{aligned} E = \frac{B}{Z} &= \frac{e(g,g)^{\alpha s z H_1(W)} \cdot e(g,g)^{\tau\beta s H_1(W)}}{e(g,g)^{\tau\beta s H_1(W)}} \\ &= e(g,g)^{\alpha s z H_1(W)}. \end{aligned} \quad (11)$$

Finally, **CS** checks the equation $H_2(E) \stackrel{?}{=} C_{W'}$. If true, then **CS** returns the $(\widetilde{C}, C', A)$ to **DO**. Otherwise, **CS** returns $\perp$.

7) Decrypt$(\mathrm{SK}, \mathrm{W}, A, C', \widetilde{C}) \to \mathcal{M}$: After receiving the search result and ciphertext components $(\widetilde{C}, C', A)$ from the **CS**, **DO** can decrypt the ciphertex as follows:

$$\mathcal{M} = \widetilde{C} / \Big(e(C', \widetilde{D})/A^{1/H_1(W)}\Big). \quad (12)$$

B. CORRECTNESS OF SEARCH

1) NON-PUNCTURED CASE

Here, we'd like to present the correctness of the non-punctured case of the trapdoor. If the trapdoor is not punctured with specific tags, the cloud computes the component B as follows:

$$\begin{aligned} Z &= Z_0 \\ &= \frac{e\Big(T^{(0)}_{0,1}, C'\Big)}{e\Big(T^{(0)}_{0,3}, \prod_{k=1}^{d}(C_{3,m})^{w_k}\Big) \cdot e\Big(T^{(0)}_{0,2}, C'\Big)^{w^*}} \\ &= \frac{e\Big(g_2^{(\tau+r_0)H_1(W)}, g^s\Big)}{e\Big(g^{r_0 H_1(W)}, \prod_{k=1}^{d}(Q(H_1(t_k))^s)^{w_k}\Big)} \\ &\quad \cdot \frac{1}{e\Big(Q(H_1(t_0))^{r_0 H_1(W)}, g^s\Big)^{w^*}} \\ &= \frac{e(g,g)^{s\beta(\tau+r_0)H_1(W)}}{e\Big(g^{r_0 H_1(W)}, g^{s\sum_{k=1}^{d} q(H_1(t_k))w_k}\Big) \cdot e\Big(g^{r_0 q(H_1(t_0))H_1(W)}, g^s\Big)^{w^*}} \\ &= \frac{e(g,g)^{s\beta\tau H_1(W) + s\beta r_0 H_1(W)}}{e(g,g)^{r_0 s H_1(W)(\sum_{k=1}^{d} q(H_1(t_k))w_k + q(H_1(t_0))w^*)}} \\ &= \frac{e(g,g)^{s\beta\tau H_1(W)} \cdot e(g,g)^{s\beta r_0 H_1(W)}}{e(g,g)^{r_0 s\beta H_1(W)}} \\ &= e(g,g)^{s\beta\tau H_1(W)} \end{aligned}$$

2) PUNCTURED CASE

Here, we'd like to present the correctness of the situation after the j-th puncturation of the trapdoor. If the trapdoor is punctured with tags that are not included in the ciphertext, the cloud calculates the component B, as shown in the equation at the bottom of the next page.

C. SECURITY PROOF OF PUN-CP-ABSE

Theorem 1: If an adversary $\mathcal{A}$ capable of compromising the Pun-CP-ABSE scheme, it implies that a simulator $\mathcal{B}$ possesses a significant advantage in the DBDH game.

Proof: Assuming the existence of an adversary $\mathcal{A}$ capable of attacking the Pun-CP-ABSE with an advantage of ϵ, we can create a simulator $\mathcal{B}$ that achieves an advantage of $\epsilon/2$ in the DBDH game. First, the challenger C selects the groups G_0 and G_1 with the bilinear map e. Then C randomly chooses the number $\eta \in \{0, 1\}$. If $\eta = 0$, then C sets $(g, X_1, X_2, X_3, Y) = (g, g^{x_1}, g^{x_2}, g^{x_3}, e(g,g)^{x_1 x_2 x_3})$. Otherwise, C sets $(g, X_1, X_2, X_3, Y) = (g, g^{x_1}, g^{x_2}, g^{x_3}, e(g,g)^{x_4})$ for random numbers $x_1, x_2, x_3, x_4 \in \mathbb{Z}_p$.

Init: $\mathcal{A}$ chooses a access strategy $(\mathbb{A}^*, \rho^*)$, where $\mathbb{A}^*$ is a $\ell \times n$ matrix. $\mathcal{A}$ also selects a label set $T = \{t_1^*, \ldots, t_d^*\}$ as the challenge set, and sends them to $\mathcal{B}$.

Setup: $\mathcal{B}$ chooses the vector $y^* = (y_1^*, y_2^* \ldots, y_n^*)$, where $y_1^* = -1$, it holds that $\mathbb{A}_i^* \cdot y^* = 0$ for all $\rho^*(i) \in S$. Then $\mathcal{B}$ randomly selects α' and implicitly sets $\alpha = x_1 x_2 + \alpha'$ by letting $e(g,g)^{\alpha} = e(g^{x_1}, g^{x_2}) \cdot e(g,g)^{\alpha'}$. It also randomly picks $t \in \mathbb{Z}_p$, and sets as $t = x_1$. Further $\mathcal{B}$ selects empty set R, and a counter $k = 0$. Then it computes $h_x = g^{z_x} g^{\mathbb{A}_{i,x}^*}$ for all attributes $1 \le x \le U$, where z_x is a random value and $\rho^*(i) = x$. Next, $\mathcal{B}$ selects $d+1$ points $\{\phi_{t_0}, \phi_{t_1}, \ldots, \phi_{t_d}\} \in \mathbb{Z}_p$. Note that ϕ_{t_0} is a unique value not utilized in the simulation. Then it defines $q(0) = \beta = x_2$, and $q(t_i) = \phi_{t_i}$, then $Q(H(t_i)) = g^{q(t_i)} = g^{\phi_{t_i}}$. Finally, $\mathcal{B}$ sets public key as:

$$
\begin{aligned}
Z_0 &= \frac{e\left(T_{j,1}^{(0)}, C'\right)}{e\left(T_{j,3}^{(0)}, \prod_{k=1}^{d}(C_{3,m})^{w_k}\right) \cdot e\left(T_{j,2}^{(0)}, C'\right)^{w^*}} \\
&= \frac{e\left(g_2^{(\tau+r_0)H_1(W)} \cdot g_2^{r_1+\ldots+r_j-\lambda_1-\ldots-\lambda_j}, g^s\right)}{e\left(g^{r_0 H_1(W)} \cdot g^{r_1+\ldots+r_j}, \prod_{k=1}^{d}(Q(H_1(t_k))^s)^{w_k}\right)} \\
&\quad \cdot \frac{1}{e\left(Q(H_1(t_0))^{r_0 H_1(W)} \cdot g^{q(H_1(t_0))(r_1+\ldots+r_j)}, g^s\right)^{w^*}} \\
&= \frac{e(g,g)^{s\beta(\tau+r_0)H_1(W)}}{e\left(g^{r_0 H_1(W)}, g^{s\sum_{k=1}^{d} q(H_1(t_k))w_k}\right) \cdot e\left(g^{r_1+\ldots+r_j}, g^{s\sum_{k=1}^{d} q(H_1(t_k))w_k}\right)} \\
&\quad \cdot \frac{e(g,g)^{s\beta(r_1+\ldots+r_j-\lambda_1-\ldots-\lambda_j)}}{e\left(g^{r_0 q(H_1(t_0))H_1(W)}, g^s\right)^{w^*} \cdot e\left(g^{q(H_1(t_0))(r_1+\ldots+r_j)}, g^s\right)^{w^*}} \\
&= \frac{e(g,g)^{s\beta\tau H_1(W)+s\beta r_0 H_1(W)}}{e(g,g)^{r_0 s H_1(W)(\sum_{k=1}^{d} q(H_1(t_k))w_k + q(H_1(t_0))w^*)}} \\
&\quad \cdot \frac{e(g,g)^{s\beta(r_1+\ldots+r_j-\lambda_1-\ldots-\lambda_j)}}{e(g,g)^{(r_1+\ldots+r_j)s(\sum_{k=1}^{d} q(H_1(t_k))w_k + q(H_1(t_0))w^*)}} \\
&= \frac{e(g,g)^{s\beta\tau H_1(W)+s\beta r_0 H_1(W)}}{e(g,g)^{r_0 s\beta H_1(W)}} \cdot \frac{e(g,g)^{s\beta(r_1+\ldots+r_j-\lambda_1-\ldots-\lambda_j)}}{e(g,g)^{(r_1+\ldots+r_j)s\beta}} \\
&= e(g,g)^{s\beta\tau H_1(W)} \cdot e(g,g)^{s\beta(-\lambda_1\ldots-\lambda_j)} \\
Z_j &= \frac{e\left(T_{j,1}^{(j)}, C'\right)}{e\left(T_{j,3}^{(j)}, \prod_{k=1}^{d}(C_{3,m})^{w_k}\right) \cdot e\left(T_{j,2}^{(j)}, C'\right)^{w^*}} \\
&= \frac{e\left(g_2^{(\lambda_j+r_j')}, g^s\right)}{e\left(g^{r_j'}, \prod_{k=1}^{d}(Q(H_1(t_k))^s)^{w_k}\right) \cdot e\left(Q(H_1(t))^{r_j'}, g^s\right)^{w^*}} \\
&= \frac{e(g,g)^{s\beta(\lambda_j+r_j')}}{e\left(g^{r_j'}, g^{s\sum_{k=1}^{d} q(H_1(t_k))w_k}\right) \cdot e\left(g^{r_j' q(H_1(t))}, g^s\right)^{w^*}} \\
&= \frac{e(g,g)^{s\beta(\lambda_j+r_j')}}{e(g,g)^{r_j' s(\sum_{k=1}^{d} q(H_1(t_k))w_k + q(H_1(t))w^*)}} \\
&= \frac{e(g,g)^{s\beta(\lambda_j+r_j')}}{e(g,g)^{r_j' s\beta}} \\
&= e(g,g)^{s\beta\lambda_j} \\
Z &= \prod_{j=0}^{i} Z_j = e(g,g)^{s\beta\tau H_1(W)}
\end{aligned}
$$

$\mathrm{PK}_{\mathrm{KGC}} = (g, g_1 = g^a, g_2 = g^\beta, e(g,g)^\alpha, \{h_i\}_{1\le i\le|U|}, \{g^{q(j)}\}_{1\le j\le d}, t_0)$.

Phase 1: $\mathcal{A}$ adaptively submits an attribute set S to $\mathcal{B}$. $\mathcal{B}$ generates the corresponding secret key SK and puncturable key PSK. First, $\mathcal{B}$ randomly picks two value $\tau, r_0 \in Z_p$, and implicitly let $\tau = -x_1$, $r_0 = r_0' + x_1$. Then $\mathcal{B}$ simulates the secret key as $\widetilde{D} = g^{\alpha+a\tau} = g^{x_1x_2+\alpha'+x_1x_1} = g^{\alpha'}X_1^{x_1+x_2}$, $D_1 = g^\alpha \cdot g^{\tau(a+\beta)} = g^{x_1x_2}g^{\alpha'}g^{-x_1(x_1+x_2)} = g^{\alpha'}/X_1^2$, $D_2 = g^\tau = 1/X_1$, $D_x = \left(g^{z_x}\cdot g^{\mathbb{A}^*_{i,x}}\right)^{-x_1} = g^{-x_1z_x - x_1\mathbb{A}^*_{i,x}} = 1/X_1^{z_x+\mathbb{A}^*_{i,x}}$. Moreover, $\mathcal{B}$ simulates the puncturable key as $psk_1 = g_2^{(\tau+r_0)} = g^{\beta(\tau+r_0)} = X_2^{r_0'}$, $psk_2 = Q(H_1(t_0))^{r_0} = g^{\phi_{t_0}(r_0'+x_1)} = g^{\phi_{t_0}r_0'}\cdot X_1^{\phi_{t_0}}$, $psk_3 = g^{r_0} = g^{r_0'+x_1} = g^{r_0'}\cdot X_1$. Finally, $\mathcal{B}$ gives SK and PSK to $\mathcal{A}$.

Challenge: $\mathcal{A}$ delivers two messages of equal size, $\mathcal{M}_0$ and $\mathcal{M}_1$, to the simulator $\mathcal{B}$. Subsequently, $\mathcal{B}$ picks random bit $\vartheta \in \{0,1\}$ to generate the ciphertext as follows. First, $\mathcal{B}$ computes $\widetilde{C} = \mathcal{M}\cdot e(g,g)^{s\alpha} = \mathcal{M}\cdot e(g,g)^{x_3(x_1x_2+\alpha')} = \mathcal{M}\cdot Y\cdot e(g,g)^{x_3\alpha'}$. Then $\mathcal{B}$ computes $C' = g^s = g^{x_3} = X_3$, and for $i = 1,\ldots,\ell$ randomly selects exponent $r_i \in \mathbb{Z}_p$ and implicitly lets $r_i = -x_1$, then computes $C_{1,i} = g^{a\pi_i}h_{\rho(i)}^{-r_i} = g^{x_1\pi_i}g^{x_1z_x}g^{x_1\mathbb{A}^*_{i,x}} = X_1^{\pi_i+z_x+\mathbb{A}^*_{i,x}}$, $C_{2,i} = g^{r_i} = g^{-x_1} = 1/X_1$. For labels t_j, where $j = 1,\ldots,d$. $\mathcal{B}$ computes $C_{2,j} = Q(H_1(t_j))^s = g^{s\phi_{t_j}} = X_3^{\phi_{t_j}}$. $\mathcal{B}$ delivers $CT^*_{\mathcal{M}} = \{\widetilde{C}, C', \{C_{1,i}, C_{2,i}\}_{1\le i\le\ell}, \{C_{3,j}\}_{1\le j\le d}\}$ to $\mathcal{A}$.

Query Phase 2. This phase is to repeat the Phase 1.

Guess: $\mathcal{A}$ gives a guess ϑ' about ϑ.

If $\vartheta \neq \vartheta'$, $\mathcal{B}$ returns 1 to imply that $Y = R$ is a random tuple of group G_1. The advantage of $\mathcal{B}$ is denotes as $Pr[\mathcal{B}(g^{x_1}, g^{x_2}, g^{x_3}, Y = R) = 1] = 1/2$. If $\vartheta = \vartheta'$, $\mathcal{B}$ returns 0 to imply $Y = e(g,g)^{x_1x_2x_3}$. Then $\mathcal{B}$ has an advantage $Pr[\mathcal{B}(g^{x_1}, g^{x_2}, g^{x_3}, Y = e(g,g)^{x_1x_2x_3}) = 0] = 1/2 + \epsilon$ to break the DBDH assumption. Therefore, the superiority of $\mathcal{B}$ in achieving success in the CPA game is determined as:

$$\begin{aligned} Adv_{\mathcal{B}}^{CPA} &= |\frac{1}{2}Pr[\mathcal{B}(g^{x_1}, g^{x_2}, g^{x_3}, Y = e(g,g)^{x_1x_2x_3}) = 0] \\ &\quad + \frac{1}{2}Pr[\mathcal{B}(g^{x_1}, g^{x_2}, g^{x_3}, Y = e(g,g)^{x_4}) = 1] - \frac{1}{2}| \\ &= \frac{1}{2}(\frac{1}{2}+\epsilon) + \frac{1}{2}\cdot\frac{1}{2} - \frac{1}{2} = \frac{\epsilon}{2}. \end{aligned}$$

Therefore, the Theorem 1 is proved. ■

Theorem 2: If an adversary $\mathcal{A}$ possesses the ability to successfully undermine the CKA security of the Pun-CP-ABSE with an advantage of ϵ, then $\mathcal{B}$ will possess a non-negligible advantage of $\epsilon/2$ in the DBDH game.

Proof: C selects public parameters g, G_0, G_1 and e, then randomly chooses the value $\vartheta \in \{0,1\}$. If $\vartheta = 0$, then C sets $(g, X_1, X_2, X_3, Y) = (g, g^{x_1}, g^{x_2}, g^{x_3}, e(g,g)^{x_1,x_2,x_3})$. Otherwise, C sets $(g, X_1, X_2, X_3, Y) = (g, g^{x_1}, g^{x_2}, g^{x_3}, e(g,g)^{x_4})$ for random numbers $x_1, x_2, x_3, x_4 \in \mathbb{Z}_p$.

Init: $\mathcal{A}$ picks an access strategy and a label set as $\left\{(\mathbb{A}^*, \rho^*), T = \{t_1^*, \ldots, t_d^*\}\right\}$, where $\mathbb{A}^*$ is a $\ell \times n$ matrix. Then $\mathcal{A}$ sends them to $\mathcal{B}$.

Setup: $\mathcal{B}$ randomly picks α' and implicitly sets $\alpha = x_1x_2$ by letting $e(g,g)^\alpha = e(g^{x_1}, g^{x_2})$. The rest of the processes are consistent with the proofs in Theorem 1.

Query Phase 1. $\mathcal{A}$ repeatedly issues the following four kinds of queries and receives responses from challenger:

- **Private-key-query**: $\mathcal{B}$ randomly selects value $\tau \in Z_p$, and implicitly let $\tau = -x_1$. Then $\mathcal{B}$ simulates the secret key as $\widetilde{D} = g^{\alpha+a\tau} = g^{x_1x_2+x_1x_1} = X_1^{x_1+x_2}$, $D_1 = g^\alpha \cdot g^{\tau(a+\beta)} = g^{x_1x_2}g^{-x_1(x_1+x_2)} = 1/X_1^2$. The rest of the settings are the same as the proofs in Theorem 1.
- **Trapdoor-query**: First, $\mathcal{B}$ computes $W_h = H_1(W)$ for generating the puncturable trapdoor $\widetilde{\mathrm{T}}_W^{(0)} = \left[(T_{0,1}^{(0)}, T_{0,2}^{(0)}, T_{0,3}^{(0)}, T_{0,4}^{(0)})\right]$, where $T_{0,1}^{(0)} = \{psk_1\}^{H_1(W)} = X_2^{r_0'W_h}$, $T_{0,2}^{(0)} = \{psk_2\}^{H_1(W)} = g^{\phi_{t_0}r_0'W_h}\cdot X_1^{\phi_{t_0}W_h}$, $T_{0,3}^{(0)} = \{psk_3\}^{H_1(W)} = g^{r_0'W_h}\cdot X_1^{W_h}$, $T_{0,4}^{(0)} = t_0$. Then $\mathcal{B}$ randomly selects exponents $\sigma \in \mathbb{Z}_p$ to generate general trapdoor $\widehat{\mathrm{T}}_W = (\hat{T}_1, \hat{T}_2, \hat{T}_{3,x}, \hat{T}_4)$, where $\hat{T}_1 = D_1^{H_1(W)}\cdot \mathrm{PK}_{\mathrm{CS}}{}^\sigma = g^{x_c\sigma}/X_1^{2W_h}$, $\hat{T}_2 = D_2^{H_1(W)} = 1/X_1^{W_h}$, $\hat{T}_{3,x} = \{D_x^{H_1(W)}\}_{x\in S} = 1/X_1^{(z_x+2\mathbb{A}^*_{i,x})W_h}$, $\hat{T}_4 = g^\sigma$.
- **Puncture**: $\mathcal{B}$ randomly selects exponents $\lambda_k, r_k, r_k' \in \mathbb{Z}_p$ and implicitly set $\lambda_k = x_1 - \hat{r}_k$, $r_k = x_1$, $r_k' = x_2 + \hat{r}_k$. Then $\mathcal{B}$ uses the label t to puncture $\widetilde{\mathrm{T}}_W^{(k-1)}$ and puts the label t to the set R. For generating the $\widetilde{\mathrm{T}}_W^{(k)}$, $\mathcal{B}$ computes $\widetilde{\mathrm{T}}_k^{(0)} = \left(T_{k,1}^{(0)}, T_{k,2}^{(0)}, T_{k,3}^{(0)}, T_{k,4}^{(0)}\right)$, and $\widetilde{\mathrm{T}}^{(k)} = \left(T_1^{(k)}, T_2^{(k)}, T_3^{(k)}, T_4^{(k)}\right)$, where $T_{k,1}^{(0)} = T_{k-1,1}^{(0)}\cdot g_2^{r_k-\lambda_k} = X_2^{r_0'W_h+\hat{r}_1+\ldots+\hat{r}_k}$, $T_{k,2}^{(0)} = T_{k-1,2}^{(0)}\cdot Q(H_1(t_0))^{r_k} = g^{\phi_{t_0}r_0'W_h}\cdot X_1^{\phi_{t_0}(W_h+k)}$, $T_{k,3}^{(0)} = T_{k-1,3}^{(0)}\cdot g^{r_k} = g^{r_0'W_h}\cdot X_1^{W_h+k}$, $T_{k,4}^{(0)} = t_0$, $T_1^{(k)} = g_2^{\lambda_k+r_k'} = X_2^{x_1+x_2}$, $T_2^{(k)} = Q(H_1(t))^{r_k'} = g^{\phi_{t_0}(x_2+\hat{r}_k)} = X_2^{\phi_{t_0}}\cdot g^{\phi_{t_0}\hat{r}_k}$, $T_3^{(k)} = g^{r_k'} = X_2\cdot g^{\hat{r}_k}$, $\widetilde{T}_4^{(k)} = t$. Then $\mathcal{B}$ returns the new punctured trapdoor $\widetilde{\mathrm{T}}_W^{(k)}$ to the $\mathcal{A}$.
- **Corrupt**: $\mathcal{B}$ returns the most latest puncturable trapdoor $\widetilde{\mathrm{T}}_W^{(k)} = \left[\widetilde{\mathrm{T}}_k^{(\tilde{0})}, \ldots, \widetilde{\mathrm{T}}^{(k-1)}, \widetilde{\mathrm{T}}^{(k)}\right]$ to the $\mathcal{A}$ when this query is first issued. For the subsequent queries, $\mathcal{B}$ returns $\perp$.

Challenge: $\mathcal{A}$ submits two exact size keywords, W_0 and W_1, to $\mathcal{B}$. Then $\mathcal{B}$ randomly chooses bit $\vartheta \in \{0,1\}$ to calculate $W_\vartheta = H_1(W)$, then generates the index $C_W = H_2(e(g,g)^{\alpha s W_\nu})$. The remaining process is identical to the proofs in Theorem 1. Finally, $\mathcal{B}$ delivers $CT^*_W = \{C_W, C', \{C_{1,i}, C_{2,i}\}_{1\le i\le\ell}, \{C_{3,j}\}_{1\le j\le d}\}$ to $\mathcal{A}$.

Query Phase 2. Phase 2 entails the same requirements as Phase 1, but requires for an additional condition that the two keywords must not have been issued beforehand.

Guess: $\mathcal{A}$ returns a guess ϑ' about ν. If $\vartheta = \vartheta'$, $\mathcal{B}$ returns 0 to express that $Y = e(g,g)^{x_1x_2x_3}$. Otherwise, Y is a random tuple of group G_1. The definition of the advantage of $\mathcal{B}$ in winning the security game can be stated as follows:

$$\begin{aligned} Adv_{\mathcal{B}}^{CKA} &= |\frac{1}{2}Pr[\vartheta = \vartheta' \mid Y = e(g,g)^{x_1x_2x_3}] \\ &\quad + \frac{1}{2}Pr[\vartheta = \vartheta' \mid Y = e(g,g)^{x_4}] - \frac{1}{2}| \end{aligned}$$

TABLE 3. Notations.

Notation	Definition
M_{G_0}, M_{G_1}	the multiplication in group G_0, G_1, respectively.
E_{G_0}, E_{G_1}	the exponentiation in group G_0, G_1, respectively.
Z, C_0, C_1	bit-length of element in $\mathbb{Z}_p$, G_0, G_1, respectively.
P	the pairing computation.
ℓ	the total count of rows of the LSSS matrix.
n_{max}	the maximum columns of the LSSS matrix
d	the total count of tags in the ciphertext.
t	the total count of tags punctured in the trapdoor.
s	the total count of attributes in the secret key.
u	the total count of attributes in the universe.
w	the total count of keywords.

$$= \frac{1}{2}(\frac{1}{2} + \epsilon) + \frac{1}{2} \cdot \frac{1}{2} - \frac{1}{2} = \frac{\epsilon}{2}.$$

Therefore, the Theorem 2 is proved. ■

VI. PERFORMANCE EVALUATION

In this part, we give the numerical analysis of the Pun-CP-ABSE and implement extensive simulations to evaluate our proposed scheme's performance in practical applications.

A. NUMERICAL ANALYSIS

Here we mainly analyze the algorithm's computation and communication costs generated by **DO**, **Dev**, **KGC**, and **CS**.In real applications, **Dev** encrypts the data and sends it to the **CS**. Moreover, **KGC** produces the public and secret keys, then distributes them to the **Dev** and **DO**, respectively. **DO** uses their trapdoor to access the data. Whenever **DO** wants to delete the data, **DO** punctures the trapdoor to achieve data deletion. When considering the computation costs, the computation cost of hash functions and computations associated with the Lagrange Interpolation are disregarded due to their notable efficiency compared to other operations. The notations are given in the TABLE 3 for ease.

1) COMPUTATION COST

In TABLE 4, we compare the algorithm computation costs of previous CP-ABSE [34],CP-ABSE [11], Pun-CP-ABE [15] with the proposed Pun-CP-ABSE. From the TABLE 4, we can observe that the computation cost of Setup in our proposed Pun-CP-ABSE is the same as that in the Pun-CP-ABE. The computation cost of Setup in our scheme is lower than CP-ABSE [11] and higher than CP-ABSE [35]. The computation cost of our proposed Pun-CP-ABSE in the KeyGen, Encrypt algorithms are lower than others. The computation cost of Trapdoor in our proposed Pun-CP-ABSE is related to the attributes, while in the CP-ABSE [34] and CP-ABSE [11], they are extra related to the keywords. The Pun-CP-ABE [15] did not consider the search function. Thus, the Trapdoor is ignored. The computation cost of Puncture in our proposed Pun-CP-ABSE are the same as that in Pun-CP-ABE [15], while CP-ABSE [34] and CP-ABSE [11] do not consider the deletion function. The computation cost of Search in our scheme is higher than that in CP-ABSE [34] and lower than CP-ABSE [11]. The Pun-CP-ABE [15] does not consider the search function. The CP-ABSE [34] and CP-ABSE [11] do not consider the Decrypt, and the computation cost of the Pun-CP-ABE [15] is higher than our proposed scheme's.

2) STORAGE COST

In TABLE 5, we compare the storage costs of previously proposed CP-ABSE [34], CP-ABSE [11], Pun-CP-ABE [15] with the proposed Pun-CP-ABSE. There, we discuss storage costs of the Public key, Secret key, Encrypt (i.e., ciphertext), Trapdoor, and Search (i.e., search result). We can see from the TABLE 5 that the storage cost of the Public Key in our proposed Pun-CP-ABSE is higher than that in scheme CP-ABSE [34], and less than that in Pun-CP-ABE [15] and Pun-CP-ABE [11]. Because in our proposed Pun-CP-ABSE, the Public Key is related to the attributes and tags. In the Pun-CP-ABE [15], it's related to the tags and the maximum number of columns of a matrix. In the Pun-CP-ABE [11], it's related to the universe attribute, a maximum number of columns' matrix, and some keywords. And other schemes are only related to the attributes or other public parameters. The storage cost of the Secret Key in our scheme is lower than those in CP-ABSE [34], CP-ABSE [11], and Pun-CP-ABE [15]. Towards the storage cost of the Encrypt, the Pun-CP-ABE [15] and Pun-CP-ABE [11] schemes are higher than others, and our scheme's storage costs are similar to the CP-ABSE [34] scheme. The storage cost of the Trapdoor in our scheme is lower than that in the CP-ABSE [34] and CP-ABSE [11]. The storage cost of the search result in our scheme is acceptable.

B. IMPLEMENTATION

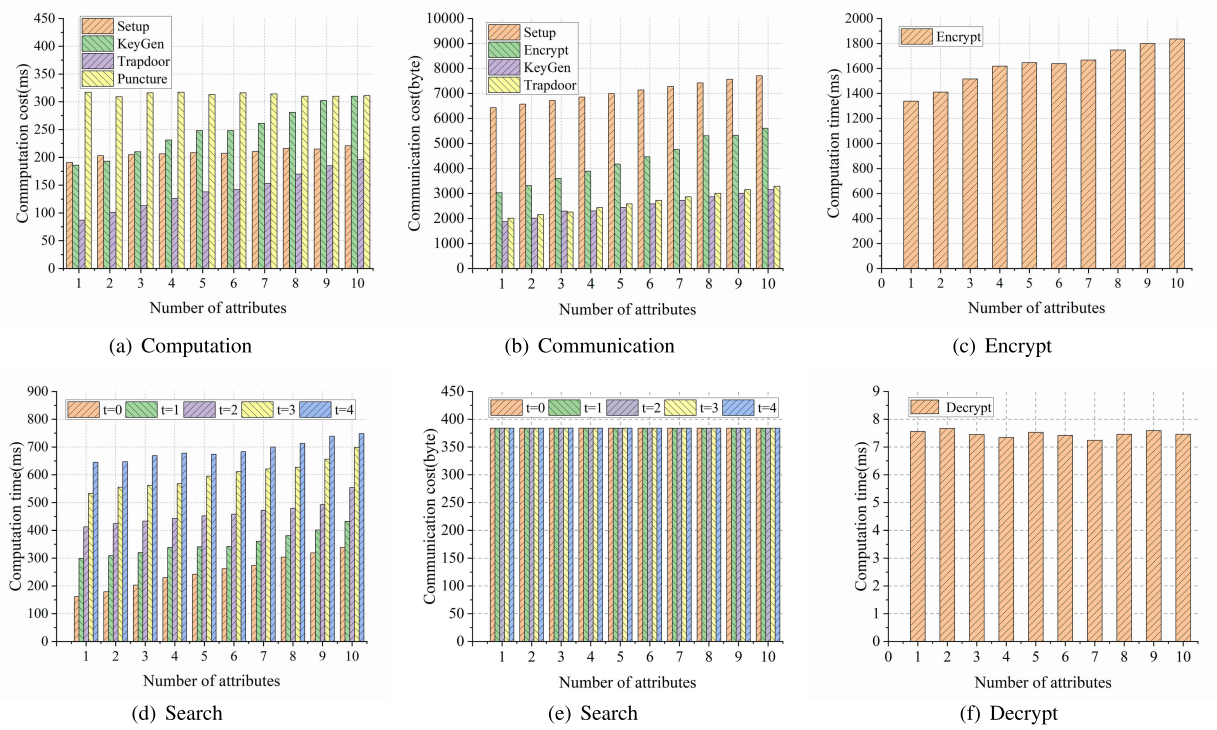
Our Pun-CP-ABSE scheme was executed on an Intel(R) Core(TM) i7-8565U CPU running at 1.80GHz and equipped with 8 GB RAM. The operating system used was ubuntu-22.04.1, and the implementation was based on JPBC (Java Pairing-Based Cryptography) in Java 1.8.0. When we simulate our scheme, the attributes of the access policy and tags of the ciphertext change from 1 to 10, respectively. The simulation result represents the average duration of 100 iterations. Fig. 4 shows the effect of attributes on the algorithm. We define the attributes of the access policy changing from 1 to 10, and tags related to the ciphertext are defined as 10 constantly. Fig. 4 (a) presents the operation time of the algorithms of Setup, KeyGen, Trapdoor, and Puncture. The operation time of those algorithms rises with the number of attributes except the Puncture algorithm. The computation cost of the Puncture algorithm is constant when the attributes grow. In Fig. 4 (b), we evaluate the communication costs of the Setup, Encrypt, KeyGen, and Trapdoor.

TABLE 4. Comparisons of computational cost.

scheme	CP-ABSE [34]	CP-ABSE [11]	Pun-CP-ABE [15]	Pun-CP-ABSE(Ours)
Sutup	$4E_{G_0} + E_{G_1}$	$(3w+3)E_{G_0} + E_{G_1} + P$	$(3+d)\mathrm{E}_{G_0} + \mathrm{E}_{G_1} + P$	$(3+d)\mathrm{E}_{G_0} + \mathrm{E}_{G_1} + P$
KeyGen	$(3w+s)E_{G_0} + E_{G_1}$	$(s \times n_{max} + n_{max} + 1)E_{G_0} + (n_{max} + s + 1)M_{G_0}$	$(4+\ell+n_{max})\mathrm{E}_{G_0} + (1+n_{max})\mathrm{M}_{G_0}$	$(8+s)\mathrm{E}_{G_0} + 3\mathrm{M}_{G_0}$
Encrypt	$(3w+2s)E_{G_0} + 2E_{G_1} + (w+s)M_{G_0}$	$(2\ell \times n_{max} + 3w + 1)E_{G_0} + E_{G_1} + \ell \times n_{max}\mathrm{M}_{G_0}$	$(1+d+\ell+\ell \times n_{max})\mathrm{E}_{G_0} + \mathrm{M}_{G_1}$	$(1+d+3\ell)\mathrm{E}_{G_0} + \mathrm{M}_{G_0} + 2\mathrm{E}_{G_1} + \mathrm{M}_{G_1}$
Trapdoor	$(s+3w)E_{G_0} + 2E_{G_1} + wM_{G_0}$	$(4w+2)\mathrm{E}_{G_0} + (2w+1)\mathrm{M}_{G_0}$	-	$(7+s)\mathrm{E}_{G_0} + \mathrm{M}_{G_0}$
Puncture	-	–	$6\mathrm{E}_{G_0} + 3\mathrm{M}_{G_0}$	$6\mathrm{E}_{G_0} + 3\mathrm{M}_{G_0}$
Search	$2E_{G_0} + 3E_{G_1} + E_{G_T} + 2sM_{G_T} + (2s+3w)P$	$(s \times n_{max} + s)\mathrm{E}_{G_0} + w\mathrm{E}_{G_1} + (s \times n_{max} + w + 2)\mathrm{M}_{G_1} + (4w + n_{max} + s + 1)P$	-	$d \times (t+1)\mathrm{E}_{G_0} + (1+\ell+t)\mathrm{E}_{G_1} + (1+t+\ell)\mathrm{M}_{G_1} + (4+\ell+3t)P$
Decrypt	–	–	$(n_{max} \times \ell + (t+1) \times d)\mathrm{E}_{G_0} + (t+1)\mathrm{E}_{G_1} + (3+\ell+t+n_{max})\mathrm{M}_{G_1} + (4 + n_{max} \times \ell + 3t)P$	$\mathrm{E}_{G_1} + P$

TABLE 5. Comparisons of storage cost.

Scheme	Public key	Secret key	Encrypt	Trapdoor	Search
CP-ABSE [34]	$4\mathrm{C}_0 + \mathrm{C}_1$	$(2w+s)\mathrm{C}_0 + \mathrm{C}_1$	$(2w+s)\mathrm{C}_0 + 2\mathrm{C}_1$	$(2w+s)\mathrm{C}_0 + 2\mathrm{C}_1$	-
CP-ABSE [11]	$(u \times n_{max} + 3w + 4)\mathrm{C}_0 + \mathrm{C}_1$	$(n_{max} + s + 1)\mathrm{C}_0$	$(\ell \times n_{max} + 3w + 1)\mathrm{C}_0 + \mathrm{C}_1$	$(n_{max} + 3w + s + 2)\mathrm{C}_0$	-
Pun-CP-ABE [15]	$(n_{max} \times d + d + 2)C_0 + C_1 + Z$	$(4+\ell+d)\mathrm{C}_0 + \mathrm{Z}$	$(\ell \times n_{max} + d)\mathrm{C}_0 + C_1$	-	-
Pun-CP-ABSE(Ours)	$(3+u+d)\mathrm{C}_0 + \mathrm{C}_1 + \mathrm{Z}$	$(6+s)\mathrm{C}_0 + \mathrm{Z}$	$(1+d+2\ell)\mathrm{C}_0 + 2\mathrm{C}_1$	$(6+s)\mathrm{C}_0 + \mathrm{Z}$	$\mathrm{C}_0 + 2\mathrm{C}_1$



(a) Computation (b) Communication (c) Encrypt

(d) Search (e) Search (f) Decrypt

FIGURE 4. The effect of attributes on the algorithms.

The costs of communication for those schemes also increase slightly, with the number of attributes increasing. The time required to perform the Encrypt operation is illustrated in Figure 4 (c). The operation time of the Encrypt algorithm increases linearly with the number of attributes increasing. Fig. 4 (d), (e) show the execution time and communication

(a) Computation (b) Communication

FIGURE 5. The effect of tags on the algorithms.

cost of the Search algorithm, in which the effect of the Puncture operation is considered. The t is the number of tags punctured to the trapdoor. The $t = 0$ is the case of a non-punctured state. Based on the information presented in Fig. 4 (d), it is evident that the duration of the Search algorithm increases when more tags are punctured to the trapdoor and the number of attributes grows in the access policy. The reason is that in the Search algorithm, three pairing operations are calculated for each tag, and two pairing operations and one multiplication operation are calculated for each attribute. From Fig. 4 (e), we can observe that the communication cost of the Search algorithm is constant and unrelated to the attributes and punctured tags. Fig. 4 (f) presents the operation time of the Decrypt algorithm. The execution time of the Decrypt is constant and unrelated to the attribute in the system. It's evident in Fig. 4 (f) that the computation time of our scheme's Decrypt operation is significantly lower than others. Because most of the computation operations related to the attributes and tags are completed in the Search algorithm by powerful cloud servers. Therefore, **DO** can quickly obtain the data with minimal computation in the Decrypt phase.

Fig. 5 shows the effect of tags on the Setup, Encrypt and Search algorithm. We define the tags changing from 1 to 10, and attributes are defined as 10 constantly. Fig. 5 (a), (b) show the effect of the tags on the execution time and communication cost, respectively. From Fig. 5 (a), we can see that the execution time of the Encrypt algorithm is significantly influenced by tags, while Setup and Search algorithms are slightly effected by tags. From Fig. 5 (b), we can see that the tags affect both the communication costs of the Setup and Encrypt, which linearly rise with the number of attributes. Meanwhile, tags do not influence the Search algorithm, as the communication cost is constant.

VII. CONCLUSION

In this paper, we have introduced an advanced self-controlled data deletion scheme within a searchable mechanism. This scheme empowers **DO**s to effectively and permanently delete the ciphertext by puncturing their trapdoors with specific labels corresponding to the ciphertext. The data deletion process does not relay to trusted third parties, which will not generate additional communication overhead, and the data deletion process ensures forward security. Therefore, we give the algorithm constructions of the Pun-CP-ABSE scheme. The Pun-CP-ABSE scheme achieves fine-grained access control over encrypted data with a keyword search that enables the **DO** to retrieve the data corresponding to the keywords if the **DO** attribute can satisfy the access strategy. Moreover, we prove our scheme is secure against the CPA and CKA. Last, We implemented simulations of the Pun-CP-ABSE scheme to show its efficiency and practicability.

REFERENCES

[1] S. Das and S. Namasudra, "Multiauthority CP-ABE-based access control model for IoT-enabled healthcare infrastructure," *IEEE Trans. Ind. Informat.*, vol. 19, no. 1, pp. 821–829, Jan. 2023.

[2] Y. Wan, X. Lin, K. Xu, F. Wang, and G. Xue, "Extracting spatial information of IoT device events for smart home safety monitoring," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, May 2023, pp. 1–10.

[3] Y. Miao, F. Li, X. Li, J. Ning, H. Li, K. R. Choo, and R. H. Deng, "Verifiable outsourced attribute-based encryption scheme for cloud-assisted mobile e-health system," *IEEE Trans. Dependable Secure Comput.*, early access, Jul. 4, 2023, doi: 10.1109/TDSC.2023.3292129.

[4] D. Ghopur, J. Ma, X. Ma, Y. Miao, J. Hao, and T. Jiang, "Puncturable ciphertext-policy attribute-based encryption scheme for efficient and flexible user revocation," *Sci. China Inf. Sci.*, vol. 66, no. 7, Jul. 2023, Art. no. 172104.

[5] X. Feng, J. Ma, S. Liu, Y. Miao, X. Liu, and K. R. Choo, "Transparent ciphertext retrieval system supporting integration of encrypted heterogeneous database in cloud-assisted IoT," *IEEE Internet Things J.*, vol. 9, no. 5, pp. 3784–3798, Mar. 2022.

[6] T. Liu, Y. Miao, K. R. Choo, H. Li, X. Liu, X. Meng, and R. H. Deng, "Time-controlled hierarchical multikeyword search over encrypted data in cloud-assisted IoT," *IEEE Internet Things J.*, vol. 9, no. 13, pp. 11017–11029, Jul. 2022.

[7] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proc. IEEE Symp. Secur. Privacy.*, May 2000, pp. 44–55.

[8] F. Li, J. Ma, Y. Miao, X. Liu, J. Ning, and R. H. Deng, "A survey on searchable symmetric encryption," *ACM Comput. Surv.*, vol. 56, no. 5, pp. 1–42, May 2024.

[9] Z. Li, J. Ma, Y. Miao, X. Liu, and K.-K.-R. Choo, "Forward and backward secure keyword search with flexible keyword shielding," *Inf. Sci.*, vol. 576, pp. 507–521, Oct. 2021.

[10] Y. Miao, R. H. Deng, K. R. Choo, X. Liu, J. Ning, and H. Li, "Optimized verifiable fine-grained keyword search in dynamic multi-owner settings," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 4, pp. 1804–1820, Jul. 2021.

[11] Q. Huang, G. Yan, and Q. Wei, "Attribute-based expressive and ranked keyword search over encrypted documents in cloud computing," *IEEE Trans. Services Comput.*, vol. 16, no. 2, pp. 957–968, Mar. 2023.
[12] Y. Miao, F. Li, X. Li, Z. Liu, J. Ning, H. Li, K. R. Choo, and R. H. Deng, "Time-controllable keyword search scheme with efficient revocation in mobile e-health cloud," *IEEE Trans. Mobile Comput*, vol. 23, no. 1, pp. 1–15, May 2023.
[13] Y. Miao, Q. Tong, K. R. Choo, X. Liu, R. H. Deng, and H. Li, "Secure online/offline data sharing framework for cloud-assisted industrial Internet of Things," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 8681–8691, Oct. 2019.
[14] M. D. Green and I. Miers, "Forward secure asynchronous messaging from puncturable encryption," in *Proc. IEEE Symp. Secur. Privacy*, May 2015, pp. 305–320.
[15] T. V. Xuan Phuong, R. Ning, C. Xin, and H. Wu, "Puncturable attribute-based encryption for secure data delivery in Internet of Things," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Apr. 2018, pp. 1511–1519.
[16] S.-F. Sun, X. Yuan, J. K. Liu, R. Steinfeld, A. Sakzad, V. Vo, and S. Nepal, "Practical backward-secure searchable encryption from symmetric puncturable encryption," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2018, pp. 763–780.
[17] B. Waters, "Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization," in *Proc. Int. Workshop Public Key Cryptogr.* Taormina, Italy: Springer, 2011, pp. 53–70.
[18] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Proc. Adv. Cryptol. (EUROCRYPT)*, Aarhus, Denmark. Alexandria, VA, USA: Springer, 2005, pp. 457–473.
[19] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proc. 13th ACM Conf. Comput. Commun. Secur.*, Alexandria, VA, USA, Oct. 2006, pp. 89–98.
[20] U. C. Yadav and S. T. Ali, "Ciphertext policy-hiding attribute-based encryption," in *Proc. Int. Conf. Adv. Comput., Commun. Informat. (ICACCI)*, Kochi, India, Aug. 2015, pp. 2067–2071.
[21] S. Hohenberger and B. Waters, "Online/offline attribute-based encryption," in *Public-Key Cryptography—PKC*, Buenos Aires, Argentina. Springer, 2014, pp. 293–310.
[22] S. Chen, J. Li, Y. Zhang, and J. Han, "Efficient revocable attribute-based encryption with verifiable data integrity," *IEEE Internet Things J.*, vol. 11, no. 6, pp. 10441–10451, Mar. 2024.
[23] C. K. Chaudhary, R. Sarma, and F. A. Barbhuiya, "RMA-CPABE : A multi-authority CPABE scheme with reduced ciphertext size for IoT devices," *Future Gener. Comput. Syst.*, vol. 138, pp. 226–242, Jan. 2023.
[24] J. Li, Y. Zhang, J. Ning, X. Huang, G. S. Poh, and D. Wang, "Attribute based encryption with privacy protection and accountability for CloudIoT," *IEEE Trans. Cloud Comput.*, vol. 10, no. 2, pp. 762–773, Apr. 2022.
[25] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *Proc. Adv. Cryptol. (EUROCRYPT)*, Interlaken, Switzerland. Springer, 2004, pp. 506–522.
[26] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 1, pp. 222–233, Jan. 2014.
[27] Z. Fu, X. Wu, C. Guan, X. Sun, and K. Ren, "Toward efficient multi-keyword fuzzy search over encrypted outsourced data with accuracy improvement," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 12, pp. 2706–2716, Dec. 2016.
[28] Y. Miao, J. Ma, X. Liu, X. Li, Z. Liu, and H. Li, "Practical attribute-based multi-keyword search scheme in mobile crowdsourcing," *IEEE Internet Things J.*, vol. 5, no. 4, pp. 3008–3018, Aug. 2018.
[29] Q. Zheng, S. Xu, and G. Ateniese, "VABKS: Verifiable attribute-based keyword search over outsourced encrypted data," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Toronto, ON, Canada, Apr. 2014, pp. 522–530.
[30] J. Yu, S. Liu, M. Xu, H. Guo, F. Zhong, and W. Cheng, "An efficient revocable and searchable MA-ABE scheme with blockchain assistance for C-IoT," *IEEE Internet Things J.*, vol. 10, no. 3, pp. 2754–2766, Feb. 2023.
[31] Y. Miao, R. H. Deng, X. Liu, K. R. Choo, H. Wu, and H. Li, "Multi-authority attribute-based keyword search over encrypted cloud data," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 4, pp. 1667–1680, Jul. 2021.
[32] J. Li, X. Lin, Y. Zhang, and J. Han, "KSF-OABE: Outsourced attribute-based encryption with keyword search function for cloud storage," *IEEE Trans. Services Comput.*, vol. 10, no. 5, pp. 715–725, Sep. 2017.
[33] Y. Miao, X. Liu, K. R. Choo, R. H. Deng, J. Li, H. Li, and J. Ma, "Privacy-preserving attribute-based keyword search in shared multi-owner setting," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 3, pp. 1080–1094, May 2021.
[34] Y. Chen, W. Li, F. Gao, Q. Wen, H. Zhang, and H. Wang, "Practical attribute-based multi-keyword ranked search scheme in cloud computing," *IEEE Trans. Services Comput.*, vol. 15, no. 2, pp. 724–735, Mar. 2022.
[35] R. Geambasu, T. Kohno, A. A. Levy, and H. M. Levy, "Vanish: Increasing data privacy with self-destructing data," in *Proc. USENIX Secur. Symp.*, vol. 316, Montreal, QC, Canada, 2009, p. 5555.
[36] J. Xiong, X. Liu, Z. Yao, J. Ma, Q. Li, K. Geng, and P. S. Chen, "A secure data self-destructing scheme in cloud computing," *IEEE Trans. Cloud Comput.*, vol. 2, no. 4, pp. 448–458, Oct. 2014.
[37] Y. Yu, L. Xue, Y. Li, X. Du, M. Guizani, and B. Yang, "Assured data deletion with fine-grained access control for fog-based industrial applications," *IEEE Trans. Ind. Informat.*, vol. 14, no. 10, pp. 4538–4547, Oct. 2018.
[38] L. Xue, Y. Yu, Y. Li, M. H. Au, X. Du, and B. Yang, "Efficient attribute-based encryption with attribute revocation for assured data deletion," *Inf. Sci.*, vol. 479, pp. 640–650, Apr. 2019.

DILXAT GHOPUR received the B.S. and M.S. degrees from the School of Mathematics and System Science, Xinjiang University, Ürümqi, China, in 2015 and 2019, respectively. He is currently pursuing the Ph.D. degree with the School of Cyber Engineering, Xidian University. His current research interests include cloud security and applied cryptography.