Puncturable Search: Enabling Authorized Search in Cross-data Federation

Lin Mei, Chungen Xu(✉), and Qianmu Li

Nanjing University of Science and Technology, Nanjing 210094, JS, China
{meilin,xuchung,qianmu}@njust.edu.cn

Abstract. Recently, most popular cloud storage solutions offer data syncing and federation, but it also brings security risks. Generally, users can deploy encryption technology to dispel data privacy concerns, but the usability of data will be hindered, e.g., searchability. Multi-user searchable encryption (MUSE) scheme is a paradigm that enables search over encrypted data federated from different users. However, to the best of our knowledge, most of the existing MUSEs are vulnerable to the insider keyword guessing attack (IKGA). Besides, therein real-time team collaboration incurs significant communication and computation costs, leading to high latency. To this end, in this paper, we focus on developing a secure and efficient encrypted search scheme with little cost. Specifically, we first propose a dual-server Public-key Puncturable Encryption with Keyword Search (dPPEKS) scheme in a cross data federation scenario by extending the Puncturable Encryption (PE) and searchable encryption (SE) technologies. Our scheme realizes the efficient and dynamic update of teammates, including user joining and exiting. Through analysis, we prove that our scheme can achieve IND-CKA2 security and resist IKGA. In addition, the performance analysis demonstrates that our scheme gains a better balance in security and efficiency.

Keywords: Cloud storage · Data federation · Searchable encryption · Puncturable encryption · Keyword guessing attack

1 Introduction

Working remotely against the COVID-19 pandemic has become the norm since 2020. Many enterprises are requiring their employees to use applications like Dropbox, Google Drive to facilitate office collaboration on the cross-data from various team members [1]. Take Dropbox as an example, a manager creates the file on the Dropbox server, together with the detailed access policy to regulate other teammates' authority on this file, namely which user can read or write these files. However, these files generally involve some sensitive information of the company like financial budget, federating and storing them on the thirty-party may bring potential privacy challenges. Untrusted server and unauthorized users can be free to fetch and abuse these files, which may bring enterprises significant economic damages.

X. Yuan et al. (Eds.): QShine 2021, LNICST 402, pp. 303–322, 2021.
https://doi.org/10.1007/978-3-030-91424-0_18

Preventing unauthorized access with encryption mechanism is the key to mitigating above privacy issues, however, directly encrypting them will incur many practical issues. One of the most fundamental problem is how to search on the encrypted data. It is clear that traditional plaintext search methods on plaintext data will fail naturally unless the user downloads and decrypts all outsourced ciphertexts. Moreover, the encrypted files should also be amenable to fine-grained access control. That is, the data owner can set access barriers on some teammates transferred out of the current project team to prevent them from retrieving files.

To achieve the goal for secure multi-client search, multi-user searchable encryption (MUSE) [2,9] is proposed, which enables data users to securely search and selectively share files according to user's privilege friendly. In early stages, a straightforward but expensive approach is to leverage the broadcast encryption [9] to revoke the key of all users and assign the new key to authorized ones. Some other works propose to introduce the attribute-based encryption (ABE) [13,14,16,20,23], but doing so will suffer from a high overhead for data authorization. In addition, there are some works propose to integrate a proxy server to authorize legal teammates to access the outsourced files, which burdens the communication costs of data owner [12,21].

In addition to above performance issues, another problem that hinders the development of MUSEs is its security issue. Specifically, most of MUSE schemes [13,14,16,20] are designed under the public-key framework, in this case, the cloud server (i.e., the insider attacker) can use the public key to generate the ciphertexts of all keywords freely. For a token space with low entropy, the keyword corresponding to each token can be gradually identified by performing test with the generated ciphertexts. The above attack is called the insider keyword guessing attack (IKGA), which has been demonstrated can destroy the security of most MUSE schemes.

To tackle with the above challenges, in this paper, our first goal is to develop an efficient authorized encrypted keyword search scheme, with the hope of realizing fine-grained authorization in team collaboration. We achieve this goal by adopting the philosophy of puncturable encryption (PE) [10]. Specifically, we use the tags of unauthorized users to update the access policy of ciphertexts. Compared to directly applying PE to SE schemes for updating secret keys repeatedly, doing so can reduce the communication cost between data owners and users. And when the total number of users remains unchanged and the number of authorized users increases, the communication cost between data owners and cloud server and the computation cost of the encryption will be further reduced.

Based on the above construction, we further study how to harden it to against IKGA. In prior arts, researchers try to address this problem by using authentication encryption [11,17,18]. Specifically, the data owner and the user use their own secret key and the other party's public key to encrypt keyword and generate token, respectively. As a result, given the token, the cloud server can only search over the ciphertexts generated by the specific owner whose public key is used in the token. However, such a solution is only feasible to the single-user scenario

because the authentication process is performed one-to-one between the data owner and the user. In this work, we propose to leverage the dual-server [6,7] technology, which generally requires two servers to cooperate to complete the test algorithm.

In the following, we summarize our contributions:

- We motivate the need of MUSE in achieving authorized search in cloud-based team cooperation. Specifically, we devise a multi-user public-key puncturable encryption with keyword search scheme and optimize it to support IKGA-resistance.
- Compared with existing MUSE schemes, our scheme has the following two merits: 1) Efficient update. Any teammate can efficiently update the access policy when some users exit or join without affecting the unchanged user. 2) Non-interactive. There is no need for any third-party agencies to intervene in the authorization or revocation process.
- Our scheme is proved to be secure to resist IKGA, which is missing in most existing MUSE schemes. In addition, it also has better performance than existing works. More specifically, the size of ciphertexts in our scheme is about 6 times smaller than that of [8,15], and the computation time of ciphertexts is about 6 and 12 times less than that in [8,15], respectively.

Organization. The rest of this paper is organized as follows. Section 2 introduces related works. In Sect. 3, we give the system overview and the threat model of our scheme. Then the notations used throughout this paper, the bilinear pairing used in the scheme construction, and the hardness assumptions applied in security proof are all presented in Sect. 4. In Sect. 5, we define the dual-server Public-key Puncturable Encryption with Keyword Search (dPPEKS) scheme and its security model. Section 6 describes the dPPEKS scheme in detail, and Sect. 7 gives strict security proof. We analyze the performance of our scheme through experiments in Sect. 8. Finally, we present a brief conclusion in Sect. 9.

2 Related Work

The first PEKS scheme was proposed to enable one to search over encrypted data generated by public key system from different writers [3]. Following the work of Boneh et al. [3], researchers paid a number of efforts on this research direction and developed versatile PEKS schemes. MUSE is one of them which focuses on meeting the requirement of data sharing among multiple users. Zheng et al. [27] first introduced attribute-based keyword search (ABKS) to enable the data owner to encrypt the keyword with access control policy so that multiple users with proper cryptographic credentials can launch valid queries. Later, in order to delegate heavy system update workload to the cloud server, Sun et al. [20] incorporated proxy re-encryption and lazy re-encryption techniques to their ABKS scheme. Moreover, considering the requirement of multi-keyword search, Yang et al. [24] developed a multi-keyword rank search system which

supports flexible search authorization and time-controlled revocation for multiple users. Closely-related to the above mentioned schemes, some MUSE works based on symmetric SE are proposed as well. Curtmola et al. [9] used broadcast encryption to naturally extend the symmetric SE to support multiple users in keyword queries as long as they are authorized. In [2,25], a trusted third-party is introduced to manage the key for user authorization and revocation.

In addition to the achievement in enriching search functionality, there are some works focused on the investigation of PEKS security, particularly the potential risk from keyword guessing attack (KGA). The main reason KGA can work is aforementioned limited size of keyword space in real life. And with the follow-up to the investigation, the KGA has led to the launch of three research directions: 1) Off-line KGA from the external adversary, which means anyone other than the server can continuously generate the ciphertext and test its correspondence with eavesdropping token [5]; 2) On-line KGA from the external adversary, a unique attack method, which requires the server to be on-line during the attack. In another word, the adversary computes the ciphertexts for his choice of candidate keywords and sends the crafted ciphertexts to the server [26]. Finally, upon observing the returning ciphertexts, including one of the crafted ciphertext, the adversary can determine which keyword is relate to the token [26]. 3) IKGA, namely the KGA from the internal adversary, where the cloud server generates the keyword ciphertext of its choice and then performs the test operation to find out the keyword of the search token [5].

To mitigate the potential risk of these three IKGAs, many efforts have been made, yet, most solutions are concentrated on PEKS with single user [6,7,11, 17,18] and few on MUSE schemes. Li et al. [15] discussed how to defeat off-line KGA for MUSE from the external adversary, while the influence of the internal adversary is not mentioned. Regarding IKGA, two typical solutions are: 1) authentication encryption [11,17,18], which enables only legitimate senders can generate the keyword ciphertext; and 2) dual-server technology [6,7], which makes sure that only two servers cooperate to complete the test operation. Most recently, Chen et al. [8] proposed an IKGA secure MUSE scheme. They first introduced one server to generate the temporary search results, which later be transformed into the final results by another server. In this way, neither server can independently perform the test algorithm, and hence resisting the IKGA. However, the empirical result shows that their scheme comes with expensive computation and communication cost. To this end, in this paper, we concentrate on addressing how to realize IKGA secure in an efficient manner.

3 Problem Statement

In this section, we first describe different parties involved in the proposed system and provide a high-level description of our system. Then, to capture the capability of each participant, we define the threat model and make the information each one holds and the behaviors they act explicit.

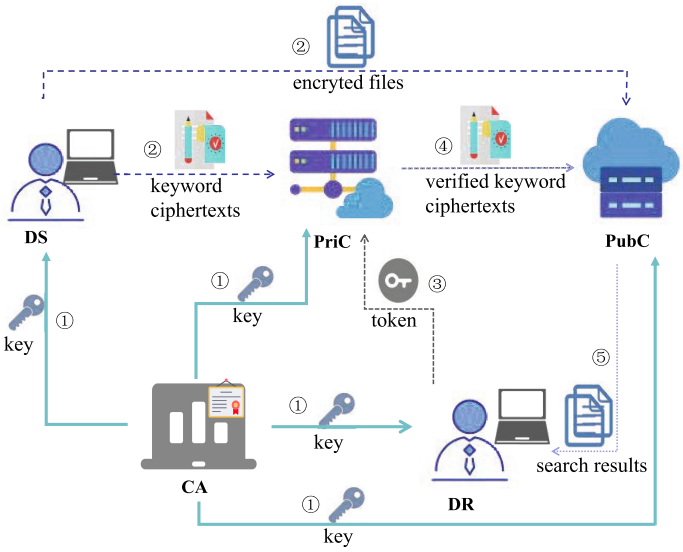


Fig. 1. The system model of the dPPEKS scheme

3.1 System Model

Figure 1 illustrates the architecture of the encrypted file search in cloud storage system. The data hosting service we consider to design involves four entities: central authority, users, private cloud, and public cloud.

1. Central authority (CA). CA plays the role of a key generator, who is in charge of generating system parameters and master secret key for the whole system. Generating secret keys for the server and team members is the responsibility of CA as well.
2. Users. Each user in our model has dual roles: the data sender (DS) and the data receiver (DR). When a user acts as the DS, he generates and sends the keyword ciphertext to the private cloud. Meanwhile, the file is encrypted and sent to the public cloud. In another case, when a user plays the role of DR, he will generate the token of the wanted keyword and send it to the private cloud.
3. Private cloud (PriC). PriC is responsible for storing the encrypted files with the keyword ciphertext and performing the access policy verification with the token submitted by users.
4. Public cloud (PubC). PubC conducts the keyword test operation on the authenticated ciphertext sent from the PriC.

3.2 Threat Model

Being consistent with most previous MUSE schemes, CA is supposed to be wholly trusted in our model, who knows the secret key of the whole system

but never launches an attack on the system. Then, we primarily stress that both servers in our model behave entirely in an "honest-but-curious" fashion. Namely, they will honestly follow the algorithm we set up and return the correct results. However, both servers will be curious about the files the user searches for and try to extract additional information from what they own, e.g., conducting the IKGA. Furthermore, we assume that the two servers will never collude with each other, in other words, they will not share their secret key. Lastly, the team members in our model, who have the search capability, are considered reliable. Yet, once they leave the current team, they will be regarded as the external adversary.

4 Preliminaries

In this section, we introduce the bilinear pairing in Sect. 4.1. In Sect. 4.2, we present the hardness assumptions that the security of our scheme relies on.

4.1 Bilinear Pairing

Let $\mathbb{G}$ and $\mathbb{G}_T$ be two cyclic groups with the same prime order p. A map e: $\mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ is bilinear if it satisfies the following three conditions [4]:

1. Bilinear: $\forall\ u, v \in \mathbb{G}$ and $x, y \in Z_p^*$, we have $e(u^x, v^y) = e(u, v)^{xy}$;
2. Non-degeneracy: If g is a generator of $\mathbb{G}$, then we have $e(g, g) \neq 1$;
3. Computability: For any group elements $x, y \in Z_p^*$, there is a polynomial time algorithm to compute the value of $e(g^x, g^y)$.

4.2 Hardness Assumption

Given a tuple $(g,\ g^a, g^b)$, where $g, g^a, g^b \in \mathbb{G}_1$. The Computational Diffie-Hellman (CDH) problem in $\mathbb{G}_1$ is to compute g^{ab}. The advantage for a probabilistic polynomial-time (PPT) adversary $\mathcal{A}$ to solve the above problem is: $Adv_{CDH}(\mathcal{A}) = |\Pr[\mathcal{A}(g, g^a, g^b) = g^{ab}]|$.

Definition 1. *(CDH assumption) We say that the CDH assumption holds if $Adv_{CDH}(\mathcal{A})$ is negligible for $\mathcal{A}$.*

Given a tuple $(g,\ g^a, g^b, g^c, g')$, where $g, g^a, g^b, g^c \in \mathbb{G}_1$ and $g' \in \mathbb{G}_2$. The Decisional Bilinear Diffie-Hellman (DBDH) problem [22] in $(\mathbb{G}_1, \mathbb{G}_2)$ is to decide if $g' = e(g, g)^{abc}$. The advantage for a PPT adversary $\mathcal{A}$ to solve the problem is: $Adv_{DBDH}(\mathcal{A}) = |\Pr[\mathcal{A}(g, g^a, g^b, g^c, e(g, g)^{abc}) = 1] - \Pr[\mathcal{A}(g, g^a, g^b, g^c, g') = 1]|$.

Definition 2. *(DBDH assumption) We say that the DBDH assumption [22] holds if $Adv_{DBDH}(\mathcal{A})$ is negligible for $\mathcal{A}$.*

Given a tuple (g, g^a), where $g, g^a \in \mathbb{G}_1$. The Discrete Logarithms (DL) problem in $\mathbb{G}_1$ is to compute the value of a. The advantage for a PPT adversary $\mathcal{A}$ to solve the above problem is: $Adv_{DL}(\mathcal{A}) = |\Pr[\mathcal{A}(g, g^a) = a]|$.

Definition 3. *(DL assumption) We say that the DL assumption holds if $Adv_{DL}(\mathcal{A})$ is negligible for $\mathcal{A}$.*

5 Definition and Security

Definition 4. *A dPPEKS scheme can be constructed from the following six algorithms:*

- **Setup**$(1^k, d)$. This probabilistic algorithm is run by the CA to setup the system. It inputs a security parameter k, the threshold d and returns the system parameter SP, the master secret key MSK, PubC's secret key SK_{PubC}, PriC's secret key SK_{PriC} and servers' joint public key PK_{js}.
- **Derive**(SP, MSK, τ_i). This probabilistic algorithm is run by the CA to derive keys for users. It inputs the system parameter SP, the master secret key MSK and the user's tag τ_i, and returns the authentication secret key $SK_{au}^{\tau_i}$, public key pk_{τ_i} and identity secret key $SK_{id}^{\tau_i}$ to the user.
- **PunEnc**(SP, $SK_{au}^{\tau_i}$, PK_{js}, w, $T^{\dagger}$). This probabilistic algorithm is run by DS to generate the keyword ciphertext. It inputs the system parameter SP, authentication secret key $SK_{au}^{\tau_i}$, servers' joint public key PK_{js}, keyword w and access policy $T^{\dagger}$, and returns the keyword ciphertext CT_w.
- **TokGen**(SP, w^*, $SK_{id}^{\tau_j}$). This probabilistic algorithm is run by DR whenever he wants to search for some files containing the specific keyword w^*. It inputs the system parameter SP, the keyword w^* and DR's identity secret key $SK_{id}^{\tau_j}$, and finally returns the search token ST_{w^*}.
- **PolicyTest**$(SK_{PriC}, ST_{w^*}, CT_w)$. This algorithm is run by the PriC to verify if the ST_{w^*} satisfies the access policy contained in CT_w. It inputs the PriC's secret key SK_{PriC}, the search token ST_{w^*} and the keyword ciphertext CT_w, and returns authenticated ciphertexts C, whose tag satisfies $\tau \notin T^{\dagger}$.
- **KeywordTest**(SK_{PubC}, C). This algorithm is run by PubC to test if the authenticated ciphertexts C contains the same keyword in ST_{w^*}. It inputs the PubC's secret key SK_{PubC} and the authenticated keyword ciphertext C, and returns 1 if $w = w^*$, otherwise returns 0.

For clarity, hereafter we consider the two algorithms **PolicyTest**$(SK_{PriC}, ST_{w^*}, CT_w)$ and **KeywordTest**(SK_{PubC}, C) as one algorithm **Test**$(SK_{PriC}, SK_{PubC}, ST_{w^*}, CT_w)$ in all security game and proof, which will not influence the security proof. It is reasonable as we assume that the challenger holds the ability of both policy test and keyword with the secret key of PriC and PubC.

Definition 5. *(CT-IND-CKA game) Given the security parameter k and the threshold parameter d, let $\mathcal{A}$ be the adversary, $\mathcal{B}$ be the challenger, and W be the keyword space.*

- **Setup**. *$\mathcal{B}$ runs the setup algorithm* **Setup**$(1^k, d)$*, gives the system parameters* SP *to the $\mathcal{A}$, and keeps the master secret key* MSK *and the secret key of both two servers SK_{PriC}, SK_{PubC}.*
- **Query phase 1**. *$\mathcal{A}$ is allowed to adaptively makes the following four queries:*
  - *Key query $\langle \tau_i \rangle$: $\mathcal{A}$ can adaptively ask $\mathcal{B}$ the secret key for any tag $\tau_i \in \{0,1\}^*$ of his choice. $\mathcal{B}$ generates the secret key following the algorithm* **Derive**(SP, MSK, τ_i) *and returns it to $\mathcal{A}$.*

⋆ *Ciphertext query* $\langle w \rangle$*:* $\mathcal{A}$ *can adaptively choose a random keyword* w *from* W*, and ask* $\mathcal{B}$ *the ciphertext of* w*. After receiving the request,* $\mathcal{B}$ *calls the algorithm* **PunEnc** *to generate the keyword ciphertext* $CT_w = \mathbf{PunEnc}(\mathrm{SP}, SK_{au}^{\tau_i}, PK_{js}, w, T^{\dagger})$ *and finally returns it to* $\mathcal{A}$*.*
⋆ *Token query* $\langle w \rangle$ *:* $\mathcal{A}$ *can adaptively choose a random keyword* w *from* W*, and ask* $\mathcal{B}$ *the search token of keyword* w*. After receiving the request,* $\mathcal{B}$ *calls the algorithm* **TokGen** *to generate the search token of keyword* $ST_w = \mathbf{TokGen}(\mathrm{SP}, w^*, SK_{id}^{\tau_j})$ *and finally returns it to* $\mathcal{A}$*.*
⋆ *Test query* $\langle CT_w, ST_{w'} \rangle$ *:* $\mathcal{A}$ *can adaptively test if the* CT_w *and* $ST_{w'}$ *containing the same keyword by asking* $\mathcal{B}$*. After receiving the test request from* $\mathcal{A}$*,* $\mathcal{B}$ *runs the* $\mathbf{Test}(SK_{PriC}, SK_{PubC}, ST_{w^*}, CT_w)$*.* $\mathcal{B}$ *returns 1 to* $\mathcal{A}$ *if* ST_w *and* CT_{w_i} *containing the same keyword and the tag in* ST_w *does not exist in the access policy of* CT_{w_i} *or 0 otherwise.*

- **Challenge**. *Once* $\mathcal{A}$ *decides to finish the phase 1, he will choose two different keywords* w_0 *and* w_1 *from* W *and a tag* τ^**, which are then sent to* $\mathcal{B}$ *as* $\mathcal{A}$*'s challenge. The only restriction is that* $\mathcal{A}$ *cannot query the ciphertext of* w_0 *and* w_1 *in phase 1. After receiving the two challenge keywords,* $\mathcal{B}$ *selects a random bit* $b \in \{0, 1\}$ *and sends a challenge ciphertext* CT_{w_b} *to* $\mathcal{A}$*.*
- **Query phase 2**. $\mathcal{A}$ *continues to adaptively query to oracles as in phase 1. The only restriction is that* $\mathcal{A}$ *cannot query the keyword ciphertexts of* w_0 *and* w_1 *in query phase 1.*
- **Guess**. $\mathcal{A}$ *outputs his guess* $b' \in \{0, 1\}$*. If* $b' = b$*, we say* $\mathcal{A}$ *wins the game.*

We define the advantage of $\mathcal{A}$ *in CT-IND-CKA game is* $Adv_{\mathcal{A}}^{CT}(k) = |Pr[b' = b] - 1/2|$.

Definition 6. *A dPPEKS scheme is assumed to be CT-IND-CKA secure if for all sufficiently large* k *and PPT adversary* $\mathcal{A}$*, there exists a negligible function* $negl$ *such that* $Adv_{\mathcal{A}}^{CT}(k) \leq negl(k)$.

Definition 7. *(ST-IND-CKA game) Given the security parameter* k *and the threshold parameter* d*, and let* $\mathcal{A}$ *be the adversary,* $\mathcal{B}$ *be the challenger, and* W *be the keyword space.*

- **Setup**. $\mathcal{B}$ *runs the setup algorithm* $\mathbf{Setup}(1^k, d)$*, gives the system parameters* SP *to* $\mathcal{A}$*, and keeps the master secret key* MSK *and the secret key of both two servers* SK_{PriC}*,* SK_{PubC}*.*
- **Query phase 1**. $\mathcal{A}$ *is allowed to adaptively makes four queries as in Definition 5.*
- **Challenge**. *Once* $\mathcal{A}$ *decides to finish the phase 1, he will choose two different keywords* w_0 *and* w_1 *from* W *and a tag* τ^* *as his challenge. The only restriction is that* $\mathcal{A}$ *cannot query the secret key of* τ^* *and the search token of* w_0 *and* w_1 *in query phase 1. After receiving the two challenge keywords,* $\mathcal{B}$ *selects a random bit* $b \in \{0, 1\}$ *and sends a challenge search token* $ST_{w_b} = \mathbf{TokGen}(w_b, SK_{\tau^*}, \mathrm{SP})$ *to* $\mathcal{A}$*.*
- **Query phase 2**. $\mathcal{A}$ *continues to adaptively query to oracles as in phase 1. There are two restrictions here. The first is that no secret key query is*

allowed on tag τ^*, and the second is that both keywords w_0 and w_1 should not be queried to the **TokGen** *oracle.*

– **Guess**. *$\mathcal{A}$ outputs the guess $b' \in \{0,1\}$. If $b' = b$, we say $\mathcal{A}$ wins the game.*

We define the advantage of $\mathcal{A}$ in ST-IND-CKA game is $Adv_{\mathcal{A}}^{ST}(k) = |Pr[b' = b] - 1/2|$.

Definition 8. *A dPPEKS scheme is assumed to be ST-IND-CKA secure if for all sufficiently large k and PPT adversary $\mathcal{A}$, there exists a negligible function $negl$ such that $Adv_{\mathcal{A}}^{ST}(k) \leq negl(k)$.*

Definition 9. *A dPPEKS scheme is assumed to be semantically secure against insider keyword guessing attack if for any PPT adversary $\mathcal{A}$, both $Adv_{\mathcal{A}}^{ST}(k)$ and $Adv_{\mathcal{A}}^{CT}(k)$ are negligible in the security parameter k.*

6 Our dPPEKS Scheme

As introduced in Sect. 4, our scheme is composed of six algorithms. In this section, we will describe the dPPEKS scheme algorithm by algorithm.

6.1 Construction

First of all, the tag of each user in our paper is randomly chosen from $\{0,1\}^*$, and each user is required to submit his tag before the whole process. The tag set of all team members is denoted by T. The threshold d represents the number of unauthorized users (i.e., leaving users). The access policy $T^\dagger$ is consisted in the tags of all leaving members and the random strings if the number of leaving members is less than the threshold d. Then our design details are given as follows.

System Setup. The **Setup** algorithm, performed by CA, with inputs security parameter k and the threshold d, first generates the system parameters for the whole system, and then outputs the master secret key for key derivation. When generating the master secret key, the most important point is that choosing a tag t_0, which should be different with all tags in T. As for server's key, it chooses two random elements y, z from Z_p as the secret key of the PriC and PubC, respectively. Then, computing g^{yz} as the joint public key of these two servers, where g is an element of the system parameters. In this paper, we assume that system parameters and the joint public key are shared for all participants within the system, while the master secret key is only known to the CA. Given part of the system parameters $(g_2, V(1), \ldots, V(d))$, it's easy for any participants to compute $V(\cdot)$ by interpolating in the exponent. The **Setup** algorithm is formulated in Algorithm 1.

Key Derivation. The **Derive** algorithm, performed by CA, is responsible for computing the secret and public key pairs for each member in the team. Without loss of generality, here we take the ith user as an example. First, a random number chosen from the group Z_p is set to be the authentication secret key

Algorithm 1. Setup

Require: Security parameter k; the threshold value d.

Ensure: System parameters SP; master secret key MSK; PubC's secret key SK_{PubC}; PriC's secret key SK_{PriC}; the server's joint public key PK_{js};.

1: Choose a group $\mathbb{G}$ of prime order p, a generator g and a hash function H:$\{0,1\}^* \rightarrow Z_p$

2: $(a_d, a_{d-1}, \ldots, a_1, a_0) \xleftarrow{R} Z_p^{d+1}, q(x) \leftarrow a_d x^d + a_{d-1} x^{d-1} + \ldots + a_1 x + a_0$

3: $r, \alpha \xleftarrow{R} Z_p, t_0 \xleftarrow{R} \{0,1\}^*, g_1 \leftarrow g^{\alpha}, g_2 \leftarrow g^{a_0}, V(x) \leftarrow g^{q(x)}$

4: SP$\leftarrow (g, g_1, g_2, V(1), \ldots, V(d)), sk_0^{(1)} \leftarrow g_2^{\alpha+r}, sk_0^{(2)} \leftarrow V(H(t_0))^r$

5: $sk_0^{(3)} \leftarrow g^r, sk_0^{(4)} \leftarrow V(H(t_0)), sk_0^{(5)} \leftarrow t_0$

6: MSK$\leftarrow (sk_0^{(1)}, sk_0^{(2)}, sk_0^{(3)}, sk_0^{(4)}, sk_0^{(5)})$

7: $y, z \xleftarrow{R} Z_p, SK_{PubC} \leftarrow y, SK_{PriC} \leftarrow z, PK_{js} \leftarrow g^{yz}$

Algorithm 2. Derive

Require: System parameters SP; master secret key MSK; the ith user's tag τ_i.

Ensure: The ith user's authentication secret key $SK_{au}^{\tau_i}$; public key $PK_{au}^{\tau_i}$; identity secret key $SK_{id}^{\tau_i}$.

1: $x \xleftarrow{R} Z_p, SK_{au}^{\tau_i} \leftarrow x$, $PK_{au}^{\tau_i} \leftarrow g^x, r_0, r_1, \lambda' \xleftarrow{R} Z_p$

2: $sk_{i0} \leftarrow (sk_0^{(1)} \cdot g_2^{r_0 - \lambda'}, sk_0^{(2)} \cdot (sk_0^{(5)})^{r_0}, sk_0^{(3)} \cdot g^{r_0}, sk_0^{(4)}, sk_0^{(5)})$

3: $sk_{i1} \leftarrow (g_2^{\lambda'+r_1}, V(H(\tau_i))^{r_1}, g^{r_1}, \tau_i, V(H(\tau_i))), SK_{id}^{\tau_i} \leftarrow [sk_{i0}, sk_{i1}]$

of the ith user. Then, the public key of the ith user can be derived from the authentication secret key (line 3 in Algorithm 2). Taking the system parameters, master secret key and the tag of the ith user as inputs, this algorithm computes the identity secret key and sends it with the authentication secret key to the ith user through a secure channel. The public key of the ith user will be sent to the user publicly. The detailed generation process is described in Algorithm 2.

Keyword Encryption. Given a keyword w, the **PunEnc** algorithm will first choose a bilinear paring map and a hash function (line 1 in Algorithm 3), with which it can compute the keyword ciphertext. Only the tags of unauthorized users will be added into the access policy $T^{\dagger}$ (line 3 in Algorithm 3). If the number of leaving users is smaller than the threshold, some random bit string will be chosen to fill the gap. More succinctly, once someone's tag, usually the tag of leaving members, belongs to the access policy, he cannot pass the Algorithm 5 and get the DS's ciphertext. Except for the access policy, system parameters and authentication secret key are used to generate the keyword ciphertext as well. Algorithm 3 displays the details of **PunEnc**.

Token Generation. For the purpose of searching the encrypted files containing specific keyword w^*, DR will run the **TokGen** algorithm to obtain the token with his identity secret key. Without loss of generality, we take the jth user as an example in this algorithm. Input the identity secret key of the jth user, this algorithm outputs the search token, a tuple (st_{j0}, st_{j1}), where st_{j0} and st_{j1} are both consisted of four parts. Note that, the random number r' (line 1 in

Algorithm 3. PunEnc

Require: System parameters SP; the server's joint public key PK_{js}; the ith user's authentication secret key $SK_{au}^{\tau_i}$; keyword w; access policy $T^{\dagger}$.

Ensure: Ciphertext CT_w.

1: Choose a bilinear pairing map e: $\mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ and a hash function H_1: $G \rightarrow G_T$

2: $s \xleftarrow{R} Z_p, ct^{(1)} \leftarrow e(g_1, g_2)^{sH(w)} \cdot H_1(g^{xyz}), ct^{(2)} \leftarrow g^s, ct^{(3)} \leftarrow g^x$

3: $ct^{(3,1)} \leftarrow V(H(t_1))^s, \ldots, ct^{(3,d)} \leftarrow V(H(t_d))^s$

4: $CT_w \leftarrow (ct^{(1)}, ct^{(2)}, ct^{(3)}, ct^{(3,1)}, \ldots, ct^{(3,d)}, T^{\dagger})$

Algorithm 4. TokGen

Require: System parameters SP; user's identity secret key $SK_{id}^{\tau_j}$; keyword w^*

Ensure: Token ST_{w^*}.

1: $r' \xleftarrow{R} Z_p, st_{j0}^{(1)} \leftarrow (sk_{j0}^{(1)})^{H(w^*)} \cdot g_2^{r'}, st_{j0}^{(2)} \leftarrow (sk_{j0}^{(2)})^{H(w^*)} \cdot (sk_{j0}^{(5)})^{r'}$

2: $st_{j0}^{(3)} \leftarrow (sk_{j0}^{(3)})^{H(w^*)} \cdot g^{r'}, st_{j0}^{(4)} \leftarrow sk_{j0}^{(4)}, st_{j0} \leftarrow (st_{j0}^{(1)}, st_{j0}^{(2)}, st_{j0}^{(3)}, st_{j0}^{(4)})$

3: $st_{j1}^{(1)} \leftarrow (sk_{j1}^{(1)})^{H(w^*)} \cdot g_2^{r'}, st_{j1}^{(2)} \leftarrow (sk_{j1}^{(2)})^{H(w^*)} \cdot (sk_{j1}^{(5)})^{r'}$

4: $st_{j1}^{(3)} \leftarrow (sk_{j1}^{(3)})^{H(w^*)} \cdot g^{r'}, st_{j1}^{(4)} \leftarrow sk_{j1}^{(4)}, st_{j1} \leftarrow (st_{j1}^{(1)}, st_{j1}^{(2)}, st_{j1}^{(3)}, st_{j1}^{(4)})$

5: $ST_{w^*} \leftarrow (st_{j0}, st_{j1})$

Algorithm 4) ensures the indistinguishability of the token. Algorithm 4 displays the details of **TokGen**.

Policy Test. Upon receiving the search token (st_{j0}, st_{j1}) from the jth user, PriC searches the keyword ciphertexts sent from different DS. Here we take the keyword ciphertext generated by the ith user as an example. Note that, the pseudo-code showed below ignores a step that the PriC will first verify the validity of ciphertext by comparing the $ct^{(3)}$ with the public key set of the whole team. If the $ct^{(3)}$ is not contained in the public key set of the team, the subsequent test process of this ciphertext will be terminated. This step is added to prevent malicious external users from uploading illegal ciphertexts. Then, based on the Lagrange interpolation polynomial, it calculates the coefficients by exploiting the access policy, a part of the search token (lines 7 to 11 in Algorithm 5). The results make an explanation to the fact that once the DR's tag is contained in the access policy, then he cannot get corresponding ciphertexts. If the jth user is authorized to the keyword, this algorithm will continue with PriC's secret key. Finally, PriC sends the verified keyword ciphertext to PubC. The detail of **PolicyTest** is described by the Algorithm 5.

Keyword Test. In this algorithm, the relationship between the ciphertexts and the search token will be test. PubC will conduct the keyword test with its secret key to obtain all ciphertext containing the queried keyword. The detailed process of **KeywordTest** is shown in Algorithm 6.

Algorithm 5. PolicyTest

Require: Token ST_{w^*}; ciphertext CT_w; PriC's secret key SK_{PriC}.
Ensure: Output: Authenticated ciphertext.

1: **for** i=0 to 1 **do**
2: $T_i^\dagger \leftarrow (st_{ji}^{(4)}, t_1, t_2, \ldots, t_d), u_i^* \leftarrow \prod_{t_m \in T_i^\dagger \setminus \{st_{ji}^{(4)}\}} \frac{-H(t_m)}{H(st_{ji}^{(4)}) - H(t_m)}$
3: **for** k=1 to d **do**
4: $u_{ki} \leftarrow \prod_{t_m \in T_i^\dagger \setminus \{t_k\}} \frac{-H(t_m)}{H(t_k) - H(t_m)}$
5: **end for**
6: $z_{i1} \leftarrow e(st_{ji}^{(1)}, ct^{(2)}), z_{i2} \leftarrow e(st_{ji}^{(2)}, ct^{(2)})^{u_i^*}$
7: $z_{i3} \leftarrow e(st_{ji}^{(3)}, \prod_{k=1}^{d} (ct^{(3,k)})^{u_{ki}}), Z_i \leftarrow \frac{z_{i1}}{z_{i2} \cdot z_{i3}}$
8: **end for**
9: $C \leftarrow (ct^{(1)}, {ct^{(3)}}^z, \prod_{i=0}^{1} Z_i)$

Algorithm 6. KeywordTest

Require: Authenticated ciphertext C; PubC's secret key SK_{PubC}.
Ensure: Authenticated ciphertext.

1: $c \leftarrow \frac{C(1)}{H_1(C(2)^y) \cdot C(3)}$

Correctness. Let $SK_{au}^{\tau_i}$, pk_{τ_i} be the authentication secret key and public key of the ith user, $SK_{id}^{\tau_j}$ be the identity secret key of the jth user, whose tag is τ_j. Let CT_w be the ciphertext that contains the keyword w and access policy $T^\dagger$ sent from the ith user, ST_{w^*} be the search token of keyword w^* that generated by the jth user. Then we can verify the correctness of our scheme as follow:

- First of all, find a tuple $(u_0^*, u_{10}, \ldots, u_{d0})$ that meets $q(0) = u_0^* \cdot q(H(st_{j0}^{(4)})) + \sum_{k=1}^{d} (u_{k0} \cdot q(H(t_k)))$. In this step, since $st_{j0}^{(4)} = t_0$ and t_0 is a tag that does not belong to T, we can work out the correct coefficient value $u_0^*, u_{10}, \ldots, u_{d0}$ and compute as following:

$$u_{k0} = \prod_{t_i \in T_0^\dagger \setminus \{t_k\}} \frac{-H(t_i)}{H(t_k) - H(t_i)}, u_0^* = \prod_{t_i \in T_0^\dagger \setminus \{st_{j0}^{(4)}\}} \frac{-H(t_i)}{H(st_{j0}^{(4)}) - H(t_i)},$$

$$z_{01} = e(st_{j0}^{(1)}, ct^{(2)}), z_{02} = e(st_{j0}^{(2)}, ct^{(2)})^{u_0^*}, z_{03} = e(st_{j0}^{(3)} \prod_{k=1}^{d} (ct^{(3,k)})^{u_{k0}}),$$

$$z_{02} \cdot z_{03} = e(g^{(r+r_0)H(w^*)+r'}, g^{sa_0}), Z_0 = \frac{z_{01}}{z_{02} \cdot z_{03}} = e(g^{(\alpha-\lambda')H(w^*)}, g^{sa_0}).$$

- Second, find a tuple $(u_1^*, u_{11}, \ldots, u_{d1})$ that meets $q(0) = u_1^* \cdot q(H(st_{j1}^{(4)})) + \sum_{k=1}^{d} (u_{k1} \cdot q(H(t_k)))$. In this step, the value of $st_{j1}^{(4)}$ is the tag of the receiver τ_j. Once the receiver's tag does not belong to the access policy $T^\dagger$, the coefficient can be properly computed. This characteristic is determined by the properties of the Lagrange interpolation polynomial. Now we suppose that

the receiver is authorized by sender for the keyword w, then we can work out the correct coefficient value $u_1^*, u_{11}, \ldots, u_{d1}$ and compute as following:

$$u_{k1} = \prod_{t_i \in T_1^\dagger \setminus \{t_k\}} \frac{-H(t_i)}{H(t_k) - H(t_i)}, u_1^* = \prod_{t_i \in T_1^\dagger \setminus \{st_{j1}^{(4)}\}} \frac{-H(t_i)}{H(st_{j1}^{(4)}) - H(t_i)}$$

$$z_{11} = e(st_{j1}^{(1)}, ct^{(2)}), z_{12} = e(st_{j1}^{(2)}, ct^{(2)})^{u_1^*}, z_{13} = e(st_{j1}^{(3)}, \prod_{k=1}^{d} (ct^{(3,k)})^{u_{k1}}),$$

$$z_{12} \cdot z_{13} = e(g^{r_1 \cdot H(w^*) + r'}, g^{sa_0}), Z_1 = \frac{z_{11}}{z_{12} \cdot z_{13}} = e(g^{\lambda' H(w^*)}, g^{sa_0}).$$

- Finally, set $C = (ct^{(1)}, ct^{(3)^z}, \prod_{i=0}^{1} Z_i)$, and test the relationship between the ciphertext CT_w and the search token ST_{w^*} as follows:

$$c = \frac{C(1)}{H_1(C(2)^y) \cdot C(3)} = \frac{e(g_1, g_2)^{sH(w)} \cdot H_1(g^{xyz})}{H_1(g^{xyz}) \cdot e\left(g^{(\alpha - \lambda')H(w^*)}, g^{sa_0}\right) \cdot e(g^{\lambda' H(w^*)}, g^{sa_0})}$$

$$= \frac{e(g_1, g_2)^{sH(w)}}{e(g^{\alpha H(w^*)}, g^{sa_0})} = \frac{e(g_1, g_2)^{sH(w)}}{e(g_1, g_2)^{sH(w^*)}}$$

Then it can be discussed from the following two cases: if $w = w^*$ holds, then set the value of c to 1; otherwise, set the value of c to 0. In summary, the correctness of our scheme has been proved.

7 Security Analysis

We evaluate the security of our dPPEKS scheme by analyzing its achievement of the definitions in Sect. 5. The dPPEKS scheme that can resist KGA defined in this paper needs to simultaneously satisfy the ciphertext indistinguishability and the search token indistinguishability under adaptively chosen keyword attack [19]. In the game described in this section, the adversary can access the key oracle, the ciphertext oracle, the token oracle and the test oracle arbitrarily. In the following, we give the security proof of our proposed scheme.

Theorem 1. *Under the DBDH and DL assumptions in the standard model, our dPPEKS scheme is secure against the off-line keyword guessing attack from internal attackers.*

This theorem will be proved through Lemmas 1 and 2 stated as follows.

Lemma 1. *Our dPPEKS scheme satisfies the CT-IND-CKA security under the DBDH assumption in the standard model.*

Proof. Here we will show that if there is an adversary $\mathcal{A}_{CT}$ that can break the CT-IND-CKA security of our dPPEKS scheme with a non-negligible advantage ε, it is equivalent to the existence of an algorithm $\mathcal{A}_{DBDH}$ that can solve the DBDH problem with the same advantage.

Let $(p, \mathbb{G}_1, \mathbb{G}_2, e, g, A = g^a, B = g^b, C = g^c, X)$ be an instance of the DBDH problem. $\mathcal{A}_{DBDH}$ tries to distinguish X and $e(g,g)^{abc}$, then it interacts with the adversary $\mathcal{A}_{CT}$ as follows:

Setup. The algorithm $\mathcal{A}_{DBDH}$ randomly chooses a d-degree polynomial $q(\cdot)$ whose constant term is equal to a_0, sets $V(x) = g^{q(x)}$ and $g_2 = g^{a_0}$. Then it selects one hash function $H : \{0,1\}^* \rightarrow Z_p$, two random numbers $\alpha, r \in Z_p$, $t_0 \in \{0,1\}^*$, and sets $g_1 = g^{\alpha}$. Set the master key MSK, a five-tuple, which consists of the following five parts: $sk_0^{(1)} = {g_2}^{\alpha+r}, sk_0^{(2)} = V(H(t_0))^r, sk_0^{(3)} = g^r, sk_0^{(4)} = t_0, sk_0^{(5)} = V(H(t_0))$. Set the secret key and joint public key of two servers: $SK_{PubC} = y, SK_{PriC} = z, PK_{js} = g^{yz}$, where y, z is randomly chosen from Z_p. Finally, it sends $PK_{js} = g^{yz}$ and $\mathrm{SP} = (g, A^{\alpha}, B^{a_0}, V(1), \ldots, V(d))$ to the adversary $\mathcal{A}_{CT}$.

Query Phase 1. The adversary $\mathcal{A}_{CT}$ adaptively makes queries to oracles. The algorithm $\mathcal{A}_{DBDH}$ responds in the following form:

Key query $\langle \tau_i \rangle$. The algorithm $\mathcal{A}_{DBDH}$ randomly samples $x, r_0, r_1, \lambda' \in Z_p$ and a tag $\tau_i \in \{0,1\}^*$ and sets authentication key pair $(SK_{au}^{\tau_i}, PK_{au}^{\tau_i}) = (x, g^x)$, identity secret key $SK_{id}^{\tau_i} = (sk_{n0}, sk_{n1})$, where

$$sk_{n0} = ({g_2}^{\alpha+r+r_0-\lambda'}, V(H(t_0)^{r+r_0}, g^{r+r_0}, t_0, V(H(t_0)))$$
$$sk_{n1} = ({g_2}^{\lambda'+r_1}, V(H(\tau_i))^{r_1}, g^{r_1}, \tau_i, V(H(\tau_i)))$$

Finally, $\mathcal{A}_{DBDH}$ returns $(SK_{au}^{\tau_i}, PK_{au}^{\tau_i})$ and $SK_{id}^{\tau_i}$ to $\mathcal{A}_{CT}$.

Ciphertext query $\langle w \rangle$. The algorithm $\mathcal{A}_{DBDH}$ samples a random number $s \in Z_p$ to generate the keyword ciphertext $CT_w = (ct^{(1)}, ct^{(2)}, ct^{(3,1)}, ct^{(3,2)}, \ldots, ct^{(3,d)}, \mathrm{T}^{\dagger})$, where $ct^{(1)} = e(A^{\alpha}, B^{a_0})^{sH(w)} \cdot H_1(g^{xyz})$, $ct^{(2)} = g^s$, $ct^{(3)} = g^x, ct^{(3,1)} = V(H(t_1^{'}))^s, \ldots, ct^{(3,d)} = V(H(t_d^{'}))^s$. Finally, $\mathcal{A}_{DBDH}$ returns CT_w to $\mathcal{A}_{CT}$.

Token query $\langle w \rangle$. The algorithm $\mathcal{A}_{DBDH}$ randomly selects a number $r' \in Z_p$, generates the search token of keyword $ST_w = (st_{n0}, st_{n1})$, where

$$st_{n0} = (st_{n0}^{(1)}, st_{n0}^{(2)}, st_{n0}^{(3)}, st_{n0}^{(4)}), st_{n0}^{(1)} = {g_2}^{(\alpha+r+r_0-\lambda')H(w)+r'},$$
$$st_{n0}^{(2)} = V(H(t_0))^{(r+r_0)H(w)+r'}, st_{n0}^{(3)} = g^{(r+r_0)H(w)+r'}, st_{n0}^{(4)} = t_0,$$
$$st_{n1} = (st_{n1}^{(1)}, st_{n1}^{(2)}, st_{n1}^{(3)}, st_{n1}^{(4)}), st_{n1}^{(1)} = {g_2}^{(\lambda'+r_1)H(w)+r'},$$
$$st_{n1}^{(2)} = V(H(t_\tau))^{r_1 H(w)+r'}, st_{n1}^{(3)} = g^{r_1 H(w)+r'}, st_{n1}^{(4)} = \tau.$$

and returns it to $\mathcal{A}_{CT}$.

Test query $\langle CT_w, ST_{w'} \rangle$. The algorithm $\mathcal{A}_{DBDH}$ runs the **Test**$(SK_{PriC}, SK_{PubC}, ST_{w^*}, CT_w)$ and returns 1 to $\mathcal{A}_{CT}$ if the keyword corresponding to ST_w and CT_{w_i} is the same and the tag in ST_w does not exist in the access policy of CT_{w_i} or 0 otherwise.

Challenge. In this step, the adversary $\mathcal{A}_{CT}$ will submit his challenge: two different keywords w_0, w_1 and a tag τ^* with the restriction that both two keywords and tag τ^* never be queried in phase 1. The algorithm $\mathcal{A}_{DBDH}$ randomly selects

a bit $\beta \in \{0,1\}$. Then it chooses a random number $s^* \in Z_p$ and computes $CT_{w_\beta} = (X^{\alpha a_0 s^* H(w_\beta)} \cdot H_1(g^{xyz}), g^{\alpha a_0 c s^*}, g^x, V(H(t_1))^{\alpha a_0 c s^*}, \ldots, V(H(t_d))^{\alpha a_0 c s^*})$. It should be noted that there should be one tag t_i equals to τ. Finally, it returns the challenge ciphertext CT_{w_β} to the adversary $\mathcal{A}_{CT}$.

Query Phase 2. The adversary $\mathcal{A}_{CT}$ adaptively makes the queries to oracles as in phase 1. Notice that both keywords w_0 and w_1 should not be queried to get their ciphertexts and the secret key should not be queried on the tag τ^*.

Guess. The adversary $\mathcal{A}_{CT}$ outputs the guess $\beta' \in \{0,1\}$. If $\beta = \beta'$ which means that $X = e(g,g)^{abc}$, the algorithm $\mathcal{A}_{DBDH}$ outputs 1 or 0 otherwise.

Next, we analyze the advantage of the algorithm $\mathcal{A}_{DBDH}$ in solving the DBDH problem. In the challenge phase, if $X = e(g,g)^{abc}$, then let $s' = s^* \cdot \alpha a_0 c$. We will get $CT_{w_\beta} = (ct^{(1)}, ct^{(2)}, ct^{(3)}, ct^{(3,1)}, \ldots, ct^{(3,d)})$, where

$$
\begin{aligned}
ct^{(1)} &= e(g,g)^{ab\alpha a_0 c s^* H(w_\beta)} \cdot H_1(g^{xyz}) = e(A,B)^{s' H(w_\beta)} \cdot H_1(g^{xyz}) \\
ct^{(2)} &= g^{\alpha a_0 c s^*} = g^{s'}, ct^{(3)} = g^x \\
ct^{(3,1)} &= g^{\alpha a_0 c s^* q(H(t_1))} = V(H(t_1))^{s'}, \ldots, ct^{(3,d)} = g^{\alpha a_0 c s^* q(H(t_d))} = V(H(t_d))^{s'}
\end{aligned}
$$

It is clear that when $X = e(g,g)^{abc}$, CT_{w_β} is a valid ciphertext of the keyword w_β. And the adversary $\mathcal{A}_{CT}$ wins the game with the probability: $|Pr[\beta' = \beta] - 1/2| = \varepsilon$. In another case, when X is a random element in the group $\mathbb{G}_2$, the adversary $\mathcal{A}_{CT}$ can obtain no additional information from the ciphertext of the keyword w_β. Therefore, the guess β' satisfies $Pr[\beta' = \beta] = 1/2$. In summary, the advantage of the algorithm $\mathcal{A}_{DBDH}$ to solve the DBDH problem satisfies the following equation: $|Pr[\mathcal{A}_{DBDH}(p, \mathbb{G}_1, \mathbb{G}_2, e, g, g^a, g^b, g^c, e(g,g)^{abc}) = 1 | a,b,c \in Z_p] - Pr[\mathcal{A}_{DBDH}(p, \mathbb{G}_1, \mathbb{G}_2, e, g, g^a, g^b, g^c, X) = 1 | a,b,c \in Z_p \wedge X \in \mathbb{G}_2]| = |(1/2 \pm \varepsilon) - 1/2| = \varepsilon$. This completes the proof of Lemma 1.

Lemma 2. *Our dPPEKS scheme satisfies the ST-IND-CKA security under the DL assumption in the standard model.*

Proof. Here we will show a series of games based on the Definition 8.

Game 0. This is the original game as described in Definition 7, and we can redefine the advantage that the adversary wins in **Game 0** as $Adv^{\mathbf{Game0}}(\mathcal{A}) = Adv^{ST}(\mathcal{A})$.

Game 1. Let **Game** 1 be the same as **Game** 0, except for the choice of challenge token. The token of w_β, referred as $ST^*_{w_\beta} = (st^*_{n0}, st^*_{n1})$, is computed in the following way:

$$
\begin{aligned}
st^*_{n0} &= (st^{(1)*}_{n0}, st^{(2)*}_{n0}, st^{(3)*}_{n0}, st^{(4)*}_{n0}), st^{(1)*}_{n0} = g_2^{c_1 \cdot H(w_\beta) + r'}, \\
st^{(2)*}_{n0} &= (V(H(t_0)))^{c_2 \cdot H(w_\beta) + r'}, st^{(3)*}_{n0} = g^{c_3 \cdot H(w_\beta) + r'}, st^{(4)*}_{n0} = t_0, \\
st^*_{n1} &= (st^{(1)*}_{n1}, st^{(2)*}_{n1}, st^{(3)*}_{n1}, st^{(4)*}_{n1}), st^{(1)*}_{n1} = g_2^{c_4 \cdot H(w_\beta) + r'}, \\
st^{(2)*}_{n1} &= (V(H(t_\tau)))^{c_5 \cdot H(w_\beta) + r'}, st^{(3)*}_{n1} = g^{c_6 \cdot H(w_\beta) + r'}, st^{(4)*}_{n1} = \tau
\end{aligned}
$$

where $c_1, c_2, c_3, c_4, c_5, c_6$ are chosen randomly from Z_p. We can find that it is impossible for any PPT adversary to compute the value of r' based on the DL assumption, even if SP, pk and pk_s are known to the adversary. Furthermore, due to the randomness of r', the distribution of $ST^*_{w_\beta}$ and ST_{w_β} cannot be distinguished from the perspective of adversary. Thus, we have $\mid Adv^{\mathbf{Game1}}(\mathcal{A}) - Adv^{\mathbf{Game0}}(\mathcal{A}) \mid \leq Adv_{DL}(\mathcal{A})$, and the $Adv_{DL}(\mathcal{A})$ is negligible if the DL assumption holds (Definition 3).

Game 2. Let **Game** 2 be the same as **Game** 1, except for the choice of challenge token. Randomly choose st^{**}_{n0} and st^{**}_{n1} from $\mathbb{G}_1$ as the component of $ST^{**}_{w_\beta}$. Due to the randomness of r' and the chosen approach of $ST^{**}_{w_\beta}$, it is impossible for adversary to distinguish $ST^*_{w_\beta}$ and $ST^{**}_{w_\beta}$. Thus, we have $Adv^{\mathbf{Game2}}(\mathcal{A}) = Adv^{\mathbf{Game1}}(\mathcal{A})$. Moreover, we know that the adversary wins in **Game** 2 with probability $\frac{1}{2}$ because $ST^{**}_{w_\beta}$ is independent of β. Therefore, the advantage that the adversary wins in the **Game** 2 can be presented as $Adv^{\mathbf{Game2}}(\mathcal{A}) = \mid \frac{1}{2} - \frac{1}{2} \mid = 0$.

Finally, based on the game defined above, we have $\mid Adv^{\mathbf{Game2}}(\mathcal{A}) - Adv^{ST}(\mathcal{A}) \mid \leq Adv_{DL}(\mathcal{A})$, where $Adv^{\mathbf{Game2}}(\mathcal{A}) = 0$ and $Adv_{DL}(\mathcal{A})$ is negligible. In all, the advantage $Adv^{ST}(\mathcal{A})$ is negligible.

This completes the proof of Lemma 2.

8 Performance Analysis

Comparing with the prior dual-server scheme [6] and MUSE schemes [8,15], we briefly conduct theoretical analysis and actual experiments of our dPPEKS scheme in this section. All the experiments in this section are conducted on a laptop with Windows 10 Intel (R) Core (TM) i5-5200U CPU @ 2.20 GHz and 4 GB RAM with Java language. In our implementation, the cryptographic operations are implemented with the help of source library jpbc. As for the experimental configuration, the Type A which can be constructed on the curve $y^2 = x^3 + x$ over the finite field F_p for the prime number $p = 3 \mod 4$, is chosen to do the pairing operation.

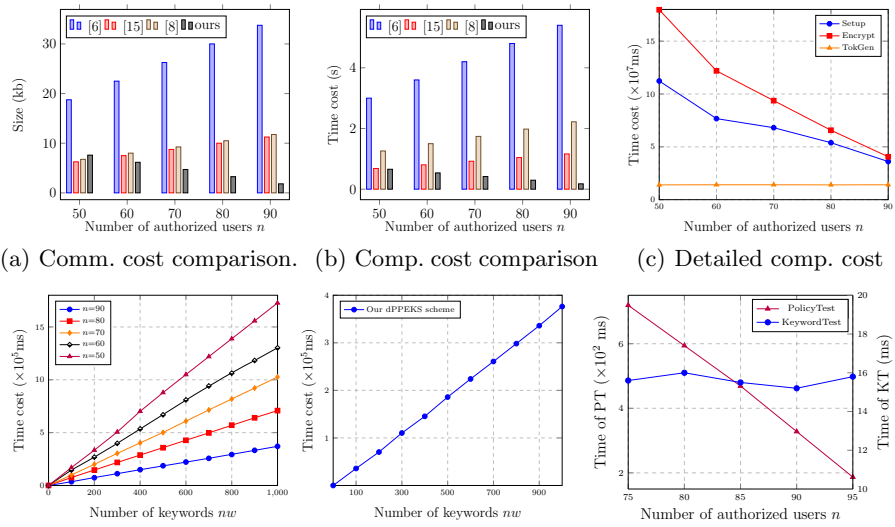
8.1 Theoretical Analysis

When conducting theoretical analysis on [6,8,15] and the proposed dPPEKS scheme, for the sake of fairness, we exclude the communication and computation cost of encrypting and decrypting steps existing in [15]. Furthermore, for attribute-based schemes [8,15], the attribute contained in their access policy is equivalent to the authorized tags in dPPEKS scheme. The number of keywords is fixed to one in conjunctive keyword search scheme [8]. Note that, only time-consuming operations like the exponential operation (E), bilinear pairing operation (P) and three kinds of hash operation ($H : \{0, 1\}^* \rightarrow \mathbb{G}_1$, H_1: $\mathbb{G}_T \rightarrow \{0, 1\}^k$, H_T: $\mathbb{G}_1 \rightarrow \mathbb{G}_T$ respectively) are considered in our theoretical computation cost analysis.

To demonstrate the efficiency of our scheme in keyword encryption intuitively, we compare the communication (comm.) and computation (comp.) costs

by histogram in Fig. 2(a) and (b). In terms of communication cost, Fig. 2(a) clearly shows that with the increase of authorized users, the size of ciphertexts in [6,8,15] increases linearly. While the proposed dPPEKS scheme is superior with the aforementioned three schemes. The difference of [8,15] and our scheme in ciphertext size is slight when n equals to 50, but when n increases to 90, the communication cost in our scheme is only 1.8 Kb, which is about 6 times smaller than that of [8,15].

The proposed dPPEKS scheme has much less computational burden than those in [6,8,15]. As shown in Fig. 2(b), the average encryption time of our scheme is far less than the other three schemes. For example, when setting $n = 90$, it only costs 0.177 s to generate the ciphertext containing 90 users in our scheme, which is about 6 and 12 times less than that in [8,15].



(a) Comm. cost comparison. (b) Comp. cost comparison (c) Detailed comp. cost

(d) Comp. cost of **PunEnc** (e) Time cost of test (f) The time cost for **PolicyTest** and **KeywordTest**.

Fig. 2. Demonstration of effectiveness for our dPPEKS scheme.

8.2 Experiment Analysis

As for the performance analysis, we conduct experiment simulation of our dPPEKS scheme. In our experiments, the tag and keyword space are consists of 100 and 1000 strings randomly selected in $\{0,1\}^*$, respectively. The security parameter k is set to 128 bits. For comparison, we set $N = 100$ in the whole paper. Note that, the reason why we omit the experimental comparison between our scheme and other schemes are listed as follows: First, the scheme in [6] is only suitable for single user scenario. Second, the authorization mechanism of

our scheme is completely different from other MUSE schemes. Through theoretical analysis, we prove that the size of the team does not affect the time cost in our proposed scheme. Other schemes sacrifice communication costs to achieve efficient privilege matching, i.e., the authorization matrix [15] and the access tree [8].

First, we explore how n affects the time cost of the three algorithms containing in our scheme: **Setup**, **PunEnc** and **TokGen**. As shown in Fig. 2(c), the algorithm **TokGen** costs an average of 141 ms, which is independent with the value of n. Besides, Fig. 2(c) also lists the **Setup** and **PunEnc** time cost against the value of n increases, which indicates that the more users are authorized, the less is the computational cost derived by **Setup** and **PunEnc**.

To have a better understanding of the effect of n and nw on algorithm **PunEnc**, our experiment shows the total per keyword encrypting time for different values of n. Figure 2(d) shows the encryption time it takes for each keyword is 1.7 s when $n = 50$, while it only consumes 0.37 s on average when $n = 90$.

Finally, given a search token, we evaluate the performance of ciphertext retrieval in test operation including **PolicyTest** (PT) and **KeywordTest** (KT) when the number of keywords increases. By setting $n = 95$, Fig. 2(e) shows that the number of keywords is positively correlated with the time cost of test operation. Furthermore, considering the detailed time cost of **PolicyTest** and **KeywordTest**, we analyze the two algorithms separately in Fig. 2(f). By varying the number of authorized users from 75 to 95, the computational cost of **PolicyTest** is almost negatively related to the variable n. This can be explained based on the fact that the size of keyword ciphertexts is linear with the number of unauthorized users, which will decrease when the number of users n increases. At the same time, the time cost of **KeywordTest** approaches to constant 16ms.

9 Conclusion

In this paper, we propose a new scheme named dPPEKS that can be appropriately applied in the cloud storage system. This scheme deals with the problem of data sharing within a team implementing encryption strategies. The heart of this scheme is the exploration of a new perspective to realize efficient data sharing by changing the object contained in the access policy from authorized users to unauthorized users. It enables data senders to set the access policy of their encrypted files and allows other teammates to search and read the ciphertexts without the help of data owners. Besides, when teammates change, the data owner can efficiently update the user's search rights without affecting other users. The scheme has been proved to be KGA-secure against any internal adversary through thorough security proof under the standard model. Furthermore, we also analyze the communication and computational cost of our scheme in detail. Extensive results prove that our scheme is superior as compared with other schemes in efficiency and security. Last but not least, we hope our research can take a new perspective for designing the MUSE schemes and promote the research on access strategies for cloud-based data deploying encryption.

Acknowledgment. This work was supported by the National Natural Science Foundation of China (No: 62072240) and the National Key Research and Development Program of China (No. 2020YFB1804604).

References

1. Coronavirus: Six tech challenges small businesses face when working from home. https://www.stuff.co.nz/business/prosper/advice/121410081/coronavirus-six-tech-challenges-small-businesses-face-when-working-from-home. Accessed 10 June 2021
2. Bao, F., Deng, R.H., Ding, X., Yang, Y.: Private query on encrypted data in multi-user settings. In: Chen, L., Mu, Y., Susilo, W. (eds.) ISPEC 2008. LNCS, vol. 4991, pp. 71–85. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-79104-1_6
3. Boneh, D., Di Crescenzo, G., Ostrovsky, R., Persiano, G.: Public key encryption with keyword search. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 506–522. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24676-3_30
4. Boneh, D., Franklin, M.: Identity-based encryption from the Weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44647-8_13
5. Byun, J.W., Rhee, H.S., Park, H.-A., Lee, D.H.: Off-line keyword guessing attacks on recent keyword search schemes over encrypted data. In: Jonker, W., Petković, M. (eds.) SDM 2006. LNCS, vol. 4165, pp. 75–83. Springer, Heidelberg (2006). https://doi.org/10.1007/11844662_6
6. Chen, B., Wu, L., Zeadally, S., He, D.: Dual-server public-key authenticated encryption with keyword search. IEEE Trans. Cloud Comput. (2019). https://doi.org/10.1109/TCC.2019.2945714
7. Chen, R., Mu, Y., Yang, G., Guo, F., Wang, X.: Dual-server public-key encryption with keyword search for secure cloud storage. IEEE Trans. Inf. Forensics Secur. **11**(4), 789–798 (2016)
8. Chen, Y., Li, W., Gao, F., Wen, Q., Zhang, H., Wang, H.: Practical attribute-based multi-keyword ranked search scheme in cloud computing. IEEE Trans. Serv. Comput. (2019). https://doi.org/10.1109/TSC.2019.2959306
9. Curtmola, R., Garay, J.A., Kamara, S., Ostrovsky, R.: Searchable symmetric encryption: improved definitions and efficient constructions. In: CCS, pp. 79–88. ACM, New York (2006)
10. Green, M.D., Miers, I.: Forward secure asynchronous messaging from puncturable encryption. In: S&P, pp. 305–320. IEEE Computer Society, New York (2015)
11. Huang, Q., Li, H.: An efficient public-key searchable encryption scheme secure against inside keyword guessing attacks. Inf. Sci. **403**, 1–14 (2017)
12. Jie, C., Han, Z., Hong, Z., Yan, X.: AKSER: attribute-based keyword search with efficient revocation in cloud computing. Inf. Sci. **423**, 343–352 (2017)
13. Kaushik, K., Varadharajan, V., Nallusamy, R.: Multi-user attribute based searchable encryption. In: MDM, pp. 200–205. IEEE Computer Society, New York (2013)
14. Li, J., Shi, Y., Zhang, Y.: Searchable ciphertext-policy attribute-based encryption with revocation in cloud storage. Int. J. Commun. Syst. **30**(1), 1–13 (2017)
15. Li, Z., Zhao, M., Jiang, H., Xu, Q.: Multi-user searchable encryption with a designated server. Ann. des Télécommunications **72**, 617–629 (2017)

16. Liang, K., Susilo, W.: Searchable attribute-based mechanism with efficient data sharing for secure cloud storage. IEEE Trans. Inf. Forensics Secur. **10**(9), 1981–1992 (2015)
17. Lu, Y., Wang, G., Li, J.: Keyword guessing attacks on a public key encryption with keyword search scheme without random oracle and its improvement. Inf. Sci. **479**, 270–276 (2019)
18. Miao, Y., Tong, Q., Deng, R., Choo, K.R., Liu, X., Li, H.: Verifiable searchable encryption framework against insider keyword-guessing attack in cloud storage. IEEE Trans. Cloud Comput. (2020). https://doi.org/10.1109/TCC.2020.2989296
19. Rhee, H.S., Park, J.H., Susilo, W., Lee, D.H.: Trapdoor security in a searchable public-key encryption scheme with a designated tester. J. Syst. Softw. **83**(5), 763–771 (2010)
20. Sun, W., Yu, S., Lou, W., Hou, Y.T., Li, H.: Protecting your right: verifiable attribute-based keyword search with fine-grained owner-enforced search authorization in the cloud. IEEE Trans. Parallel Distrib. Syst. **27**(4), 1187–1198 (2016)
21. Wang, S., Zhang, X., Zhang, Y.: Efficiently multi-user searchable encryption scheme with attribute revocation and grant for cloud storage. Plos One **11**(11), 1–23 (2016)
22. Waters, B.: Efficient identity-based encryption without random oracles. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005). https://doi.org/10.1007/11426639_7
23. Xu, L., Xu, C., Liu, J.K., Zuo, C., Zhang, P.: Building a dynamic searchable encrypted medical database for multi-client. Inf. Sci. **527**, 394–405 (2020)
24. Yang, Y., Liu, X., Deng, R.H.: Multi-user multi-keyword rank search over encrypted data in arbitrary language. IEEE Trans. Dependable Secur. Comput. **17**(2), 320–334 (2020)
25. Yang, Y.: Towards multi-user private keyword search for cloud computing. In: IEEE Cloud, pp. 758–759. IEEE Computer Society, New York (2011)
26. Yau, W., Phan, R.C., Heng, S., Goi, B.: Keyword guessing attacks on secure searchable public key encryption schemes with a designated tester. Int. J. Comput. Math. **90**(12), 2581–2587 (2013)
27. Zheng, Q., Xu, S., Ateniese, G.: VABKS: verifiable attribute-based keyword search over outsourced encrypted data. In: INFOCOM, pp. 522–530. IEEE, New York (2014)