This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2020.2963978, IEEE Transactions on Dependable and Secure Computing

Secure Keyword Search and Data Sharing Mechanism for Cloud Computing

Chunpeng Ge[1], *Member, IEEE,* Willy Susilo[2,*], *Senior Member, IEEE,* Zhe Liu[1], *Senior Member, IEEE,* Jinyue Xia[3], Pawel Szalachowski[4], and Liming Fang[1]

***Abstract*—The emergence of cloud infrastructure has significantly reduced the costs of hardware and software resources in computing infrastructure. To ensure security, the data is usually encrypted before it's outsourced to the cloud. Unlike searching and sharing the plain data, it is challenging to search and share the data after encryption. Nevertheless, it is a critical task for the cloud service provider as the users expect the cloud to conduct a quick search and return the result without losing data confidentiality. To overcome these problems, we propose a ciphertext-policy attribute-based mechanism with keyword search and data sharing (CPAB-KSDS) for encrypted cloud data. The proposed solution not only supports attribute-based keyword search but also enables attribute-based data sharing at the same time, which is in contrast to the existing solutions that only support either one of two features. Additionally, the keyword in our scheme can be updated during the sharing phase without interacting with the PKG. In this paper, we describe the notion of CPAB-KSDS as well as its security model. Besides, we propose a concrete scheme and prove that it is against chosen ciphertext attack and chosen keyword attack secure in the random oracle model. Finally, the proposed construction is demonstrated practical and efficient in the performance and property comparison.**

***Index Terms*—Cloud Data Sharing, Searchable Attribute-based Encryption, Attribute-based Proxy Re-encryption, Keyword Update.**

I. Introduction

CLOUD computing has been the remedy to the problem of personal data management and maintenance due to the growth of personal electronic devices. It is because users can outsource their data to the cloud with ease and low cost. The emergence of cloud computing has also influenced and dominated Information Technology industries. It is unavoidable that cloud computing also suffers from security and privacy challenges.

Encryption is the basic method for enabling data confidentiality and attribute-based encryption is a prominent representative due to its expressiveness in user's identity and data [1]–[4]. After the attribute-based encrypted data is uploaded in the cloud, authorized users face two basic operations: data searching and data sharing. Unfortunately, traditional attribute-based encryption just ensures the confidentiality of data. Hence, it does not support searching and sharing.

Suppose in a Person Health Record (PHR) system [5]–[7], a group of patients store their encrypted personal health reports $Enc(D_1, P_1, KW_1), \cdots, Enc(D_n, P_n, KW_n)$ in the cloud, where $Enc(D_i, P_i, KW_i)$ is an attribute-based encryption of the health report D_i under an access policy P_i and a keyword KW_i. Doctors satisfying the policy P_i can recover the record D_i. However, they could not retrieve the specific record by simply typing the keyword. Instead, a doctor Alice needs to first download and decrypt the encrypted records. After decryption, she can use the keyword to search the specific one from a bunch of the decrypted health records. Another inconvenient scenario is that Alice attempts to share a record with her colleague, in the case like she needs to consult the report with a specialist. In this situation, she must download the encrypted files, then decrypt them. Then, after she has acquired the underlying record, she encrypts the record using the policy of the specialist. As a result, this system is very inefficient in terms of searching and sharing.

Additionally, the traditional attribute-based encryption (ABE) technology used in the current PHR systems might cause another issue for keyword maintenance because the ABE algorithm could not scale well for keyword updates once the number of the records significantly increases. For example, after reviewing a health report with the patient self marked "contagious" tag, Alice from hospital A confirmed it is not the contagious condition and corrected the tag to "non-contagious". In order for Alice to share a health report that is encrypted with a tag "contagious" with another doctor from hospital B, she needs to change the tag as "non-contagious" without decrypting the report. As the traditional attribute-based encryption with keyword search can not support keyword updating, Alice has to generate a new tag for all shared ciphertexts so as to keep the privacy of the keyword.

From above scenarios, the traditional attribute-based encryption is not flexible for data searching and sharing. Additionally, attribute-based encryption is not well scaled when there is an update request to the keyword. In order to search and share a specific record, Alice downloads and decrypts the ciphertexts. However, this process is impractical to Alice especially when there is a tremendous number of ciphertexts. The worse situation is the data owner Alice should stay online all the time because Alice needs to provide her private key for the data decryption. Thus, ABE solution does not take the

* Corresponding author.

1. College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics. E-mail: {gecp, zheliu, fangliming}@nuaa.edu.cn.

2. School of Computing and Information Technology, University of Wollongong. E-mail: wsusilo@uow.edu.au.

3. IBM. E-mail: jinyue.xia@ibm.com.

4. Singapore University of Technology and Design, Singapore. E-mail: pawel@sutd.edu.sg.

advantages of cloud computing.

An alternative method is to delegate a third party to do the search, re-encrypt and keyword update work instead of Alice. Alice can store her private key in the third party's storage, and thus the third party can do the heavy job on behalf of Alice. In such an approach, however, we need to fully trust the third party since it can access to Alice's private key. If the third party is compromised, all the user data including sensitive privacy will be leaked as well. It would be a severe disaster to the users.

A. Related Work

In an ABE, the users' identities are described by a list of attributes [1]. After ABE's pioneering work [1], several scholars extended the notion of ABE. For example, key-policy attribute-based encryption (KP-ABE) [2], where the private key of a user is related to an access policy and the ciphertext corresponds to an attribute set. In contrast, there is another example called ciphertext-policy attribute-based encryption (CP-ABE) [3], where the private key is generated with an attribute set and the ciphertext is related to an access policy. In both KP-ABE and CP-ABE, the ciphertext length is linear with the size of the access policy. To reduce the ciphertext length, Emura et al. [8] proposed a ciphertext-policy attribute-based encryption scheme with constant ciphertext length. Although it supports the AND-gates on multi attributes, it doesn't support the monotonic express on attributes. After that, a number of constructions have come out to enhance the efficiency, security and expressiveness [4], [9], [10]. To illustrate the ABE's application, Li et al. [11] adopted the notion of attribute-based encryption in the PHR system to achieve fine-grained access control on personal health records. A ciphertext policy attribute-based encryption with hidden policy [12] was proposed to hide the access policy which may leak the user's privacy in the PHR system. The concept of outsourcing decryption attribute-based encryption was introduced to enable a computation-constrained mobile device to outsource most of the decryption work to a service provider [13]. However, there is no guarantee that the service provider could return the correct partial decryption ciphertext. To overcome this issue, Lai [14] and Li [15] proposed attribute-based encryption with verifiable outsourced decryption schemes respectively.

Proxy re-encryption was designed to delegate the decryption [16]. Prior work has focused on the scheme's functionality, efficiency, and security model [17] [18] [19], [20]. Later, Liang et al. [21] presented an attribute-based proxy re-encryption (AB-PRE) scheme by using proxy re-encryption to a attribute-based setting. Meanwhile, another AB-PRE scheme was proposed to support "AND" gates on positive and negative attributes [22]. Following their work, Liang et al. [23] proposed a ciphertext-policy attribute-based proxy re-encryption (CPAB-PRE) scheme supporting a monotonic access formula in the selective model. Later, the security has been improved in an adaptive model [24]. Ge et al. [25], [26] presented two KP-ABE schemes that are secure in the selective and adaptive model respectively. Liang et al. [27] proposed a deterministic finite automata (DFA) based PRE scheme, where the access policy is viewed as a DFA. Unfortunately, the privacy could not be preserved in keyword search in all of these schemes.

Allowing the search ability in public key encryption is another research direction that has gained popularity. The primitive of searchable encryption in a symmetric key setting was first introduced by Song et al. [28]. Following their work, many searchable encryption schemes with different functionalities were proposed such as the ranking search on keyword [29] and fuzzy keyword searching [30]. To extend the searchable encryption to the public key setting, Boneh et al. [31] proposed the notion of public key encryption with keyword search (PEKS). A PEKS scheme supporting range, subset and conjunctive queries on keywords was presented by Boneh and Waters [32] in TCC 2007. Later, attribute-based keyword search was proposed via the combination of a PEKS and ABE [33]. A more efficient attribute-based searchable encryption scheme was achieved by involving the data owner to issue keys for a data user [34]. A ciphertext policy attribute-based keyword search scheme was introduced in the shared multi-owner setting [35]. However, none of the above schemes could support the data sharing function.

A KP-ABPRE with keyword search scheme was designed to allow a server not only can search for a certain ciphertext but also re-encrypt it [36]. The PKG in this scheme controls the access policy in a traditional key policy ABE scheme, and the data owner loses the ability to assign access policy on his encrypted data. It is, however, worth noting here that in a PHR system [11], [12], the data owner should have full control on the data to be shared. Thus, a ciphertext policy attribute-based encryption with keyword search and data sharing scheme is desired. One additional issue with the work [36] is that the data owner must interact with the PKG and request the PKG to generate a search token which will greatly increase the burden of PKG. Moreover, it is the delegator that needs to share the data with the delegatee, which is unrelated with the PKG. Therefore, they left it as an open problem to construct an attribute-based encryption scheme supporting data searching and data sharing without the help of PKG during the searching and sharing phase.

B. Motivation

Prior work did not demonstrate that the existing attribute-based mechanisms could both support keyword search and data sharing in one scheme without resorting to PKG. Therefore, a new attribute-based mechanism is needed to achieve the goal for the above PHR scenario. One may argue that the problem can be trivially solved by combining an AB-PRE scheme and attribute-based keyword search scheme (AB-KS). However, the combination could result in two major issues: 1) the combined scheme is not CCA secure, 2) it is vulnerable to collusion attack. The detailed explanation will be given later in subsection IV-A.

Therefore, a secure scheme is desired to fully support keyword searching, data sharing as well as the protection of

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2020.2963978, IEEE Transactions on Dependable and Secure Computing

the privacy of keyword. All of these concerns motivate us to design a mechanism that:

1) allows the data owner to search and share the encrypted health report without the unnecessary decryption process.
2) supports keyword updating during the data sharing phase.
3) more importantly, does not need the exist of the PKG, either in the phase of data sharing or keyword updating.
4) the data owner can fully decide who could access the data he encrypted.

In this paper we first point out a notion of ciphertext-policy attribute-based mechanism with keyword search and data sharing (CPAB-KSDS), which also supports keyword updating.

C. Our Contribution

We first introduce a ciphertext-policy attribute-based mechanism with keyword search and data sharing (CPAB-KSDS) for encrypted cloud data. The searching and sharing functionality are enabled in the ciphertext-policy setting. Furthermore, our scheme supports the keyword to be updated during the sharing phase. After presenting the construction of our mechanism, we proof its chosen ciphertext attack (CCA) and chosen keyword attack (CKA) security in the random oracle model. The proposed construction is demonstrated practical and efficient in the performance and property comparison.

II. System Architecture and Definitions

In this section, we first present the architecture of our CPAB-KSDS scheme. Following that, we will describe the definition of the proposed scheme and its security model.

A. System Architecture

The CPAB-KSDS system, shown in Fig 1, consists of five entities: the PKG, the cloud server (act as the proxy), the health record owner, the delegator (recipient of the original ciphertext) and the delegatee (recipient of the re-encrypted ciphertext). The workflow for the system is described as follows.

System Initialization: This phase is executed by the PKG. The PKG generates the system public parameters that are publicly available for all the participants of the system and the master secret key which is kept private by the PKG.

Registration: The registration phase is executed by the PKG. When each user issues a registration request to the PKG, the PKG generates a private corresponds to his attribute set.

Ciphertext Upload: The personal health record owner encrypts his record with the original recipient's policy and the keyword, and then upload the encrypted record to the cloud server.

Ciphertext Search: The recipient generates a search token and issues a search request contains the search token to the cloud server. The cloud server searches the ciphertext via the $Test$ algorithm and returns the search result to the recipient.

Re-encryption: The delegator generates a re-encryption key and issues a re-encryption request contains the re-encryption key to the cloud server. The cloud server converts the original encrypted record to a re-encrypted ciphertext under a new access policy.

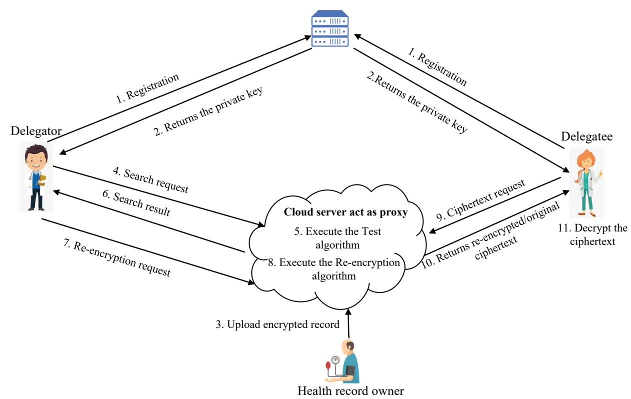


Fig. 1. System architecture.

Decryption: The recipient (a delegatee or a delegator) requests a re-encrypted (or an original) ciphertext from the cloud server and then decrypts the ciphertext with his own private key to get the underlying record. Note that, a delegatee may act as a delegator for other participants.

B. CPAB-KSDS

Definition 1 (CPAB-KSDS). A CPAB-KSDS scheme is described as follows:

- $Setup(\lambda, U) \rightarrow (PK, MK)$: The $Setup$ algorithm is executed by the PKG. Input a security parameter λ and the description of attribute universe U. Output public parameters PK and a master secret key MK.
- $KeyGen(MK, S) \rightarrow sk_S$: The $KeyGen$ algorithm is executed by the PKG. Input MK and an attribute set S. Output a private key sk_S.
- $Enc(m, (M, \rho), KW) \rightarrow CT$: The Enc algorithm is executed by the health record owner. Input a message m, an access policy (M, ρ)[1] and a keyword KW. Output an original ciphertext CT.
- $TokenGen(sk_S, KW') \rightarrow \tau_{KW'}$: The $TokenGen$ algorithm is executed by the delegator. Input the private key sk_S and a keyword KW'. Output a search token $\tau_{KW'}$ for the keyword KW'.
- $Test(CT, \tau_{KW'}) \rightarrow 1/0$: The $Test$ algorithm is executed by the cloud server. Input a ciphertext CT under KW and a search token $\tau_{KW'}$. Output returns 1 if $KW = KW'$, otherwise, simply returns 0.
- $RKeyGen(sk_S, (M', \rho'), KW') \rightarrow rk$: The $RKeyGen$ algorithm is executed by the delegator. Input a private key sk_S, an access structure (M', ρ') and a keyword KW'. Output the re-encryption key rk. Here, S satisfies (M, ρ) but not satisfies (M', ρ'). Note that, the keyword input KW' may not equal to the keyword KW in the $RKeyGen$ algorithm. If $KW' \neq KW$, it means that the delegator wants to update the keyword in the ciphertext

[1]We adopt the definition of an access policy as [37].

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2020.2963978, IEEE Transactions on Dependable and Secure Computing

and the keyword in the ciphertext will be updated in the re-encryption phase.

- $ReEnc(CT, rk) \rightarrow CT$: The $ReEnc$ algorithm is executed by the cloud server. Input an original ciphertext CT and rk computed from $RKeyGen$. Output the re-encrypted ciphertext CT under a new access policy and keyword.
- $Dec(sk_S, CT) \rightarrow m/\perp$: The Dec algorithm is executed by the delegator/delegatee to decrypt the original/re-encrypted ciphertext. Input a ciphertext CT under access policy (M, ρ) and a private key sk_S. Output the plaintext m, if $S \models (M, \rho)$, and $\perp$ otherwise.

In the above algorithms, for simplicity, we omit PK as input.

Consistency: Generally, a CPAB-KSDS scheme is consistent if using a corresponding search token can search the correctly generated ciphertext and a legal secret key can decrypt the correct ciphertext. Formally, for a message $m \in G_T$, $KW \in \{0,1\}^*$, $Setup(\lambda, U) \rightarrow (PK, MK)$, $KeyGen(MK, S) \rightarrow sk_S$, $TokenGen(sk_S, KW) \rightarrow \tau_{KW}$, $TokenGen(sk_S, KW) \rightarrow \tau_{KW'}$, $RKeyGen(sk_S, (M', \rho'), KW') \rightarrow rk$:

$$Dec(sk_S, Enc(m, (M, \rho), KW)) = m;$$
$$Test(\tau_{KW}, Enc(m, (M, \rho), KW)) = 1;$$
$$Dec(sk_{S'}, ReEnc(Enc(m, (M, \rho), KW), rk)) = m;$$
$$Test(\tau_{KW'}, ReEnc(Enc(m, (M, \rho), KW), rk)) = 1;$$

if $S \models (M, \rho)$ and $S' \models (M', \rho')^2$.

C. Threat Model for CPAB-KSDS

Our threat model considers the confidentiality for the plaintext and the keyword. We use three security games that consider the security of the original ciphertext, re-encrypted ciphertext, and keyword individually.

Definition 2 (IND-CCA-Or). If there does not exist an PPT (probability polynomial time) adversary can win the game described below with a non-negligible advantage, then the CPAB-KSDS scheme is indistinguishable chosen ciphertext secure at original ciphertext (IND-CCA-Or).

1) **Init.** $\mathcal{A}$ chooses the challenge policy (M^*, ρ^*) that is a $l^* \times n^*$ matrix.
2) **Setup.** Challenger $\mathcal{C}$ executes $Setup(\lambda, U)$ to retrieve PK and MK then forwards PK to the $\mathcal{A}$.
3) **Phase I.** $\mathcal{A}$ queries:
   a) $\mathcal{O}_{sk}(S)$: $\mathcal{A}$ queries on S, the challenger $\mathcal{C}$ executes $KeyGen(mk, S)$ to obtain sk_S, and forwards it to the $\mathcal{A}$.
   b) $\mathcal{O}_{token}(S, KW)$: $\mathcal{A}$ queries on S and a keyword KW, $\mathcal{C}$ runs $KeyGen(msk, S)$ and $\tau_{KW} \leftarrow TokenGen(sk_S, KW)$, returns τ_{KW} to the adversary $\mathcal{A}$.
   c) $\mathcal{O}_{test}(CT, KW)$: $\mathcal{A}$ queries on a ciphertext CT and a keyword KW, the challenger $\mathcal{C}$ runs algorithms $sk_S \leftarrow KeyGen(msk, S)$ and $\tau_{KW} \leftarrow TokenGen(sk_S, KW)$. Returns the test result $1/0 \leftarrow Test(CT, \tau_{KW})$ to the adversary $\mathcal{A}$.
   d) $\mathcal{O}_{rk}(S, (M', \rho'), KW')$: $\mathcal{A}$ queries on S, (M', ρ') and KW', where S does not satisfy (M', ρ'), the challenger $\mathcal{C}$ executes $sk_S \leftarrow KeyGen(MK, S)$ and $rk \leftarrow RKeyGen(sk_S, (M', \rho'), KW')$. Returns rk to $\mathcal{A}$.
   e) $\mathcal{O}_{re}(CT, S, (M', \rho'), KW')$: $\mathcal{A}$ queries on an original ciphertext CT under an access policy (M, ρ) and keyword KW, attribute set S, access policy $(M'\rho')$ and keyword KW', the challenger $\mathcal{C}$ executes $CT/\perp \leftarrow ReEnc(rk, CT)$, where $rk = RKeyGen(sk_S, (M', \rho'), KW')$, $sk_S = KeyGen(msk, S)$ and S satisfies (M, ρ). Returns the result to adversary $\mathcal{A}$.
   f) $\mathcal{O}_{dec}(S, CT)$: $\mathcal{A}$ queries on an attribute set S and ciphertext CT, the challenger $\mathcal{C}$ runs $sk_S = KeyGen(msk, S)$, $m/\perp \leftarrow Dec(sk_S, CT)$. Return the decryption result to the adversary $\mathcal{A}$.

   During Phase I, $\mathcal{A}$ is restrict not to make queries as:
   - $\mathcal{O}_{sk}(S)$ if $S \models (M^*, \rho^*)$;
   - $\mathcal{O}_{rk}(S, (M', \rho'), KW')$, if $S \models (M^*, \rho^*)$ and $\mathcal{A}$ has queried $\mathcal{O}_{sk}(S')$, where $S' \models (M', \rho')$;
4) **Challenge.** $\mathcal{A}$ sends messages (m_0, m_1) with equal length and a challenge keyword KW^* to the challenger $\mathcal{C}$. $\mathcal{C}$ randomly choose a bit $b \in \{0,1\}$, then computes challenge ciphertext $CT^* = Enc(m_b, (M^*, \rho^*), KW^*)$, and sends CT^* to $\mathcal{A}$.
5) **Phase II.** $\mathcal{A}$ queries as in the phase I except:
   - $\mathcal{O}_{sk}(S)$, if S satisfies (M^*, ρ^*);
   - $\mathcal{O}_{rk}(S, (M', \rho'), KW')$ and $\mathcal{O}_{sk}(S')$, if S, S' satisfy (M^*, ρ^*), (M', ρ') respectively;
   - $\mathcal{O}_{re}(CT^*, S, (M', \rho'), KW')$ and $\mathcal{O}_{sk}(S')$, if S, S' satisfy (M^*, ρ^*), (M', ρ') respectively;
   - $\mathcal{O}_{dec}(S, CT)$, if S satisfies (M^*, ρ^*) and CT is a derivative[3] of CT^*.
6) **Guess.** $\mathcal{A}$ makes a guess b' and wins if $b' = b$.

The adversary's advantage is defined as

$$Adv_{\mathcal{A}}^{IND-CCA-Or}(\lambda) = |Pr[b' = b] - \frac{1}{2}|.$$

Definition 3 (IND-CCA-Re). If there does not exist an PPT adversary can win the game described below with a non-negligible advantage, we say a CPAB-KSDS scheme is indistinguishable chosen ciphertext secure at re-encrypted ciphertext (IND-CCA-Re).

1) **Init.** $\mathcal{A}$ chooses the challenge policy (M^*, ρ^*) that is a $l^* \times n^*$ matrix.
2) **Setup.** Challenger $\mathcal{C}$ executes $Setup(\lambda, U)$ to retrieve PK and SK, then forwards PK to the adversary $\mathcal{A}$.

[2] Here, $S \models (M, \rho)$ indicates S satisfies (M, ρ).

[3] The definition of derivative defined in [17].

3) **Phase I.** $\mathcal{A}$ queries as below:
   a) $\mathcal{O}_{sk}(S)$: Given an attribute set S, $\mathcal{C}$ executes the $KeyGen(SK, S)$ to get the private key sk_S, and forwards sk_S to $\mathcal{A}$.
   b) $\mathcal{O}_{token}(S, KW)$: On input an attribute set S and a keyword KW, challenger $\mathcal{C}$ runs algorithms $KeyGen(SK, S)$ and $TokenGen(sk_S, KW)$. Returns τ_{KW} to the adversary $\mathcal{A}$.
   c) $\mathcal{O}_{test}(CT, KW)$: On input a ciphertext CT and a keyword KW, the challenger $\mathcal{C}$ runs algorithms $sk_S \leftarrow KeyGen(SK, S)$ and $\tau_{KW} \leftarrow TokenGen(sk_S, KW)$. Returns to $\mathcal{A}$ the test result of $1/0 \leftarrow Test(CT, \tau_{KW})$.
   d) $\mathcal{O}_{rk}(S, (M', \rho'), KW')$: On input an attribute set S, access policy (M', ρ') and keyword KW', where S does not satisfy (M', ρ'), the challenger runs $\mathcal{C}$ runs $sk_S \leftarrow KeyGen(SK, S)$ and $rk \leftarrow RKeyGen(sk_S, (M', \rho'), KW')$. Returns rk to the adversary $\mathcal{A}$.
   e) $\mathcal{O}_{dec}(S, CT)$: On input an attribute set S and ciphertext CT, the challenger $\mathcal{C}$ runs the result of $sk_S = KeyGen(SK, S)$, $m/\perp \leftarrow Dec(sk_S, CT)$ to the adversary $\mathcal{A}$.

   During Phase I, adversary $\mathcal{A}$ is restrict not to make the $\mathcal{O}_{sk}(S)$ query, where $S \models (M^*, \rho^*)$.
4) **Challenge.** $\mathcal{A}$ sends two messages (m_0, m_1) with equal length and a challenge keyword KW^* to $\mathcal{C}$. $\mathcal{C}$ chooses a random bit $b \in \{0, 1\}$ and returns the challenge ciphertext $CT^* = ReEnc(Enc(m_b, (M, \rho), KW), rk)$, where $rk \leftarrow RKeyGen(sk_S, (M^*, \rho^*), KW^*)$, $S \models (M, \rho)$ to $\mathcal{A}$.
5) **Phase II.** $\mathcal{A}$ makes queries same as phase I except:
   - $\mathcal{O}_{sk}(S)$, if $S \models (M^*, \rho^*)$;
   - $\mathcal{O}_{dec}(S, CT^*)$, $S \models (M^*, \rho^*)$.
6) **Guess.** $\mathcal{A}$ makes the guess b' and wins if $b' = b$.

The adversary's advantage is defined as

$$Adv_{\mathcal{A}}^{IND-CCA-Re}(\lambda) = |Pr[b' = b] - 1/2|.$$

In this game, since the adversary can make any re-encryption key query without restrictions, he can execute the re-encryption himself. Thus, the re-encryption query is useless.

Definition 4 (IND-CKA). A CPAB-KSDS scheme is indistinguishable chosen keyword secure (IND-CKA) if there doesn't exist a PPT adversary $\mathcal{A}$ who can win the following game with a non-negligible advantage. Let oracle $\mathcal{O}_1 = \{\mathcal{O}_{sk}, \mathcal{O}_{token}, \mathcal{O}_{test}, \mathcal{O}_{rk}, \mathcal{O}_{dec}\}$, where $\mathcal{O}_{sk}$, $\mathcal{O}_{token}$, $\mathcal{O}_{test}$, $\mathcal{O}_{rk}$, $\mathcal{O}_{dec}$ are the same as in IND-CCA-Or game.

1) **Setup.** The challenger $\mathcal{C}$ runs $Setup(\lambda, U)$ to get PK and MK. And then forwards PK to the adversary $\mathcal{A}$.
2) **Phase I.** $\mathcal{A}$ queries in $\mathcal{O}_1$.
3) **Challenge.** $\mathcal{A}$ sends two keywords (KW_0, KW_1) with equal length, a challenge message m^* and access policy (M^*, ρ^*) to $\mathcal{C}$. The restriction is that $\mathcal{A}$ cannot has made any $\mathcal{O}_{token}(S, KW)$ queries, where $S \models (M^*, \rho^*)$. Challenger $\mathcal{C}$ randomly choose a bit $b \in \{0, 1\}$ and then computes $CT^* = Enc(m^*, (M^*, \rho^*), KW_b)$. Returns CT^* to $\mathcal{A}$.
   Note that, CT^* can also be $CT^* = ReEnc(Enc(m^*, (M, \rho), KW'), rk)$, where $rk \leftarrow RKeyGen(sk_S, (M^*, \rho^*), KW_b)$, $S \models (M, \rho)$.
4) **Phase II.** Like in the query phase I $\mathcal{A}$ continues querying except:
   - $\mathcal{O}_{test}(CT^*, KW)$;
   - $\mathcal{O}_{token}(S, KW)$, where $S \models (M^*, \rho^*)$.
5) **Guess.** $\mathcal{A}$ makes the guess b' and wins if $b' = b$.

$\mathcal{A}$'s advantage is defined as

$$Adv_{\mathcal{A}}^{IND-CKA}(\lambda) = |Pr[b' = b] - 1/2|.$$

Remarks: As illustrated in [38], in the public key searchable encryption setting, an adversary can conduct the statistical attack. Detailly, an adversary can issue token queries to get the search tokens and generate a keyword ciphertext for any keywords he wants. Then the adversary can execute the $Test$ algorithm to test whether the keyword in the token equal to the keyword in the ciphertext. To capture the statistical attack, Zheng et al. [33] defined two types of keyword security: the chosen keyword attack security and the keyword secrecy. The chosen keyword attack security indicates that the adversary cannot deduce any information about the keyword from the keyword ciphertext. While the keyword secrecy means that the probability of an adversary knowing the keyword from the ciphertext and the search token is no more than the probability of guessing a random element from the possible keyword space. The key secrecy captures the fact that the keyword embedded in the token cannot be protected since an adversary can choose a keyword and generate a corresponding keyword ciphertext. Then the adversary executes the $Test$ algorithm to check whether the keyword embedded in the token equals to the keyword in the keyword ciphertext. In our scheme, we adopt the chosen keyword attack security definition of [33]. In our IND-CKA definition, though the adversary can choose a keyword KW as he likes and gets the corresponding token τ_{KW} via the $\mathcal{O}_{token}(S, KW)$ query. However, the restriction is that S does not satisfy (M^*, ρ^*). Whenever the adversary executes the $Test(\tau_{KW}, CT^*)$ algorithm, the algorithm will return 0 since S does not satisfy (M^*, ρ^*). Thus, the adversary cannot gain any extra information about the keyword in the keyword ciphertext through the $Test$ algorithm that will lead to the failure of the statistical attack.

A CPAB-KSDS scheme is said to be chosen ciphertext and chosen keyword secure if $Adv_{\mathcal{A}}^{IND-CCA-Or}(\lambda)$, $Adv_{\mathcal{A}}^{IND-CCA-Re}(\lambda)$ and $Adv_{\mathcal{A}}^{IND-CKA}(\lambda)$ are negligible.

III. PRELIMINARIES

A. Bilinear Map

G and G_T are two multiplicative cyclic groups of prime order p, $e : G \times G \to G_T$, A tuple (G, G_T, p, e) is a bilinear map tuple, if for $\forall \mu, \nu \in G, r, s \in Z_p^*$

1) $e(\mu^r, \nu^s) = e(\mu, \nu)^{rs}$;

2) $e(\mu, \nu) \neq 1$.
3) $e(\mu, \nu)$ can be computed efficiently.

B. q-BDHE Assumption

G is a group of prime order p. Randomly choose $g, \nu, s \in Z_p$. Denote g^{ν^i} as g_i. Given a vector $\vec{v} = (g, g_s, g_1, \cdots, g_q, g_{q+2}, \cdots, g_{2q}) \in G^{2q+1}$, the adversary cannot distinguish $e(g,g)^{\nu^{q+1}s} \in G_T$ from a random element in G_T.

Formally, the probability :

$$| \, Pr[\mathcal{A}(\vec{v}, T = e(g,g)^{\nu^{q+1}s})] - Pr[\mathcal{A}(\vec{v}, T = R)] \, |,$$

where $R \xleftarrow{r} G_T$, is negligible for all PPT adversary $\mathcal{A}$, then the decisional $q-$Bilinear Diffie-Hellman Exponent assumption (q-BDHE) [4] holds.

C. DL Assumption

G is a group of prime order p. Randomly choose $g, z, h \in G$, $r_1, r_2 \in Z_p$. Given a vector $\vec{v} = (g, z, h, z^{r_1}, g^{r_2}) \in G^5$, the adversary is hard to distinguish $h^{r_1+r_2} \in G$ from a random element in G.

Formally, the probability:

$$| \, Pr[\mathcal{A}(\vec{v}, T = h^{r_1+r_2})] - Pr[\mathcal{A}(\vec{v}, T = R)] \, |,$$

where $R \xleftarrow{r} G$, is negligible for all PPT adversaries $\mathcal{A}$, the following then the decisional linear assumption (DL) [33] holds.

IV. CPAB-KSDS System

A. Challenges and Our Techniques

Here we demonstrate why a simple combination of an AB-PRE scheme and attribute-based keyword search scheme (AB-KS) does not solve our design challenge. Assume the combined CPAB-KSDS ciphertext is $C_{CPAB-KSDS} = (C_{AB-PRE}, C_{AB-KS})$, where C_{AB-PRE} is an AB-PRE ciphertext and C_{AB-KS} is an AB-KS ciphertext, an adversary may issue decryption oracle of a manipulated ciphertext (C_{AB-PRE}, C'_{AB-KS}) to get the underlying plaintext. Another problem is that it is vulnerable to the collusion attack [19]. The proxy and the delegatee can collude to reveal the delegator's private key. Suppose the first part delegator's private is $K = g^\alpha f^t$. If we set the re-encryption key as $rk = K^{H(\delta)}$, where δ is an randomly chosen element and encrypted with the delegatee's attribute set S, the delegatee can first recover δ with his own private and further get the delegator's private key part K.

In our construction, we utilize the ciphertext-policy attribute-based encryption scheme [4] as the basic component since it supports any monotonic access policy and achieves the CCA security. To overcome the first issue, we bind the AB-PRE ciphertext and the AB-KS ciphertext tightly via a same random element. In such a manner, if one part of the CPAB-KSDS ciphertext is changed, the another part will update accordingly. Furthermore, in the decryption algorithm, the decryptor first checks the validity of the ciphertext and then conducts the decryption. Regarding the collusion attack issue, we introduce a random value to randomize the delegator's private key. In the detailed construction, which will be shown in the following subsection, the re-encryption is set to be $rk = K^{H(\delta)} \cdot Q^\theta$, where Q and θ are randomly chosen. Thus only with the value of δ and rk, the delegatee colludes with the proxy cannot reveal the private key part K. When it is needed to remove the random value Q^θ in the decryption algorithm, we leverage the bilinear property of the bilinear pairing to get rid of it.

B. Proposed Construction

In our scheme, ciphertexts are encrypted with an access policy and a keyword, and the private key is connected with an attribute set S. U is the attribute universe whose size is polynomial of λ. $KW \in \{0,1\}^*$ denotes a keyword. The following describes our proposed CPAB-KSDS scheme.

1) $Setup(\lambda, U)$: Chooses a bilinear map tuple (p, g, G, G_T, e), and randomly select $\alpha, \beta, a, b, c \in Z_p^*$, $f, \tilde{g} \in G$, compute $f_1 = g^c, f_2 = g^b, Q = g^\beta$. For $\forall i, 1 \leqslant i \leqslant |U|$, choose $h_1, \cdots, h_{|U|} \in G$. Choose collision-resistant hash functions: $H_1 : \{0,1\}^* \rightarrow G$, $H_2 : G_T \rightarrow \{0,1\}^*$, $H_3 : \{0,1\}^* \rightarrow Z_p^*$, $H_4 : \{0,1\}^* \times G_T \rightarrow Z_p^*$. Choose a CCA-secure symmetric key encryption $SY = (S.Enc, S.Dec)$. Output $msk = (g^\alpha, a, b)$ and $mpk = (e(g,g)^\alpha, g^a, \tilde{g}, f, f_1, f_2, Q, H_1, H_2, H_3, H_4, h_1, \cdots, h_{|U|}, SY)$.
2) $KeyGen(msk, S)$: Randomly choose $t, r \in Z_p^*$ and compute the secret key sk_S as
$$K = g^\alpha f^t, \quad L = g^t,$$
$$V = g^{(ac-r)/b}, \quad Y = g^r, \quad Z = \tilde{g}^r,$$
$$\forall x \in S, \{K_x = h_x^t, \quad Y_x = H_1(x)^r\}.$$
Note that, V can be computed as $V = f_1^{a/b}/g^{r/b}$. The secret key sk_S implicitly contains S.
3) $Enc(m, (M, \rho), KW)$: Choose a random element $R \in G_T$, then compute $s = H_4(m, R)$. Choose two random vectors $\vec{v} = (s, k_2, \cdots, k_n) \in Z_p^{*n}$, $\vec{\eta} = (s_2, k_{n+1}, \cdots, k_{2n-1}) \in Z_p^{*n}$, where $s_2, k_2 \cdots, k_{2n-1}$ are randomly chosen from Z_p^*. For $i = 1$ to l, compute $\lambda_i = \vec{v} \cdot M_i$ and $\varphi_i = \vec{\eta} \cdot M_i$, where M_i is the vector related to the i-th row of M. Randomly choose $s_1 \in Z_p^*$ and compute
$$C_0 = m \oplus H_2(R), \quad C = R \cdot e(g,g)^{\alpha s}, \quad C' = g^s,$$
$$C'' = Q^s, \quad \forall 1 \leqslant i \leqslant l, C_i = f^{\lambda_i} h_{\rho(i)}^{-s},$$
$$W = f_1^{s_1}, \quad W_0 = g^{a(s_1+s_2)} f_2^{s_1 H_2(KW)},$$
$$W_1 = f_2^{s_2}, \quad D = g^{s_2},$$
$$\forall 1 \leqslant i \leqslant l, E_i = \tilde{g}^{\varphi_i} H_1(\rho(i))^{-s_2},$$
$$E = H_1(C_0, C, C', C'', D, \{C_i, E_i\}_{i \in [1,l]}, W, W_0, W_1)^s.$$

Output the ciphertext $CT = (C_0, C, C', C'', D, \{C_i, E_i\}_{i\in[1,l]}, W, W_0, W_1, E)$. Note that, CT implicitly includes (M, ρ).

4) $TokenGen(sk_S, KW')$: Choose a random element $\gamma \in Z_p^*$ and compute

$$\tau_1 = \left(g^a f_2^{H_1(KW')}\right)^{\gamma}, \quad \tau_2 = f_1^{\gamma},$$

$$\tau_3 = V^{\gamma}, \quad Y' = Y^{\gamma} \quad Z' = Z^{\gamma},$$

Then, for each $x \in S$, compute ${Y_x}' = {Y_x}^{\gamma}$. Set the trapdoor as $\tau = (\tau_1, \tau_2, \tau_3, Y', Z', \{{Y_x}'\}_{\forall x\in S})$.

5) $Test(CT, \tau)$: Input a ciphertext $CT = (C, C', C'', D, \{C_i, E_i\}_{i\in[1,l]}, W, W_0, W_1, E)$ and a search token $\tau = (\tau_1, \tau_2, \tau_3, Y', Z', \{{Y_x}'\}_{\forall x\in S})$. If S associated with the search token τ does not satisfy (M, ρ) in CT, the algorithm returns $\perp$. Otherwise, let $I \subseteq \{1, \cdots, l\}$ be a set of indices, such that for all $i \in I$, $\rho(i) \in S$ and $\Sigma_{i\in I}\omega_i M_i = (1, 0, \cdots, 0)$. Denote $\Delta = \{x : \exists i \in I, \rho(i) = x\}$, compute

$$F = e(Y'Z', D) / \left(\prod_{i\in I} (e(Y', E_i) \cdot e(D, {Y_x}'))^{\omega_i}\right).$$

The algorithm returns 1, means $KW = KW'$, if $e(W, \tau_1)e(W_1, \tau_3)F = e(W_0, \tau_2)$. Otherwise returns 0, means $KW \neq KW'$.

Note that, if CT is a re-encrypted ciphertext, the algorithm first computes $F' = e(Y'Z', D') / \left(\prod_{i\in I} (e(Y', {E_i}') \cdot e(D', {Y_x}'))^{\omega_i}\right) = e(g, g)^{r{s_2}'\gamma}$. And then verifies whether $e(W', \tau_1)e({W_1}', \tau_3)F' \stackrel{?}{=} e({W_0}', \tau_2)$. If the equation holds, outputs 1, means $KW = KW'$, otherwise outputs 0.

6) $RKeyGen(sk_S, (M', \rho'), KW')$: Choose random elements $\delta \in \{0, 1\}^*$ and $\theta \in Z_p^*$. Compute

$$rk_1 = K^{H_3(\delta)} Q^{\theta}, \quad rk_2 = g^{\theta},$$

$$rk_3 = L^{H_3(\delta)}, \quad \forall x \in S, rk_{4,x} = K_x^{H_3(\delta)}.$$

Randomly choose $R' \in G_T$, compute $s' = H_4(\delta, R')$. Choose two random vectors $\vec{v}' = (s', {k_2}', \cdots, {k_n}') \in {Z_p^*}^n$, $\vec{\eta}' = ({s_2}', {k_{n+1}}', \cdots, {k_{2n-1}}') \in {Z_p^*}^n$, where ${s_2}', {k_2}', \cdots, {k_{2n-1}}'$ are randomly chosen from Z_p^*. For $i = 1$ to l, compute ${\lambda_i}' = \vec{v}' \cdot {M_i}'$ and and ${\varphi_i}' = \vec{\eta}' \cdot {M_i}'$, where ${M_i}'$ is the vector related to the i-th row of M'. Randomly choose ${s_1}' \in Z_p^*$ and compute

$$\widetilde{rk_5} = \delta \oplus H_2(R'), \quad rk_5 = R' \cdot e(g, g)^{\alpha s'},$$

$$rk_6 = g^{s'}, \quad \forall 1 \leqslant i \leqslant l, rk_{7,i} = f^{{\lambda_i}'} h_{\rho(i)}^{-s'},$$

$$W' = f_1^{{s_1}'}, \quad {W_0}' = g^{a({s_1}' + {s_2}')} f_2^{{s_1}' H_1(KW')},$$

$${W_1}' = f_2^{{s_2}'}, \quad D' = g^{{s_2}'},$$

$$\forall 1 \leqslant i \leqslant l, {E_i}' = \tilde{g}^{{\varphi_i}'} H_1(\rho(i))^{-{s_2}'},$$

$$E' = H_1(\widetilde{rk_5}, rk_5, rk_6, D', \{rk_{7,i}, {E_i}'\}_{i\in[1,l]}, W', {W_0}', {W_1}')^{s'}.$$

Set the re-encryption key as $rk = (rk_1, rk_2, rk_3, \{rk_{4,x}\}_{x\in S}, \widetilde{rk_5}, rk_5, rk_6, D', \{rk_{7,i}, {E_i}'\}_{i\in[1,l]}, W', {W_0}', {W_1}', E')$.

7) $ReEnc(CT, rk)$: On input an original ciphertext CT and a re-encryption key rk, compute $\bar{t} = H_1(C_0, C, C', C'', D, \{C_i, E_i\}_{i\in[1,l]}, W, W_0, W_1)$, and check whether the following equalities hold:

$$e(g, E) \stackrel{?}{=} e(C', \bar{t}), \qquad (1)$$

$$e(C', Q) \stackrel{?}{=} e(g, C''), \qquad (2)$$

$$\forall 1 \leqslant i \leqslant l, e(g, C_i) \stackrel{?}{=} e(g, f^{\lambda_i}) e(C', h_{\rho(i)})^{-1}. \qquad (3)$$

If one of them fails, the algorithm outputs $\perp$. Otherwise, it continues.

If S does not satisfy (M, ρ) in CT, it output $\perp$. Else let $I \subseteq \{1, \cdots, l\}$ be a set of indices, such that for all $i \in I$, $\rho(i) \in S$ and $\Sigma_{i\in I}\omega_i M_i = (1, 0, \cdots, 0)$. Denote $\Delta = \{x : \exists i \in I, \rho(i) = x\}$. Compute

$$\Gamma = \frac{e(rk_1, C')}{e(rk_2, C'') \cdot \prod_{i\in I} e(C_i, rk_3)^{\omega_i} \cdot e(C', \prod_{x\in\Delta} rk_{4,x})^{\omega_i}}.$$

Compute $CT_1 = S.Enc(CT||\Gamma, \delta)$, $CT_2 = (\widetilde{rk_5}, rk_5, rk_6, D', \{rk_{7,i}, {E_i}'\}_{i\in[1,l]}, W', {W_0}', {W_1}', E')$. Output the re-encrypted ciphertext $CT = (CT_1, CT_2)$.

Note that, via the $ReEnc$ algorithm, a new keyword KW' is embedded in the re-encrypted ciphertext part of W_0'. In such a manner, the keyword in the re-encrypted ciphertext was updated. For example, the original ciphertext CT is encrypted with the keyword KW. If the delegator wants to update the keyword KW to KW' in the re-encryption phase, he can issue a re-encryption key rk with the keyword KW' in the $RKeyGen$ algorithm. When the cloud server re-encrypts the original ciphertext via the $ReEnc(CT, rk)$ algorithm, the new keyword is embedded in W_0' part of the re-encrypted ciphertext.

8) $Dec(sk_S, CT)$:

(1) CT is an original ciphertext.

a) If one of them $(1)-(3)$ fails, the algorithm outputs $\perp$. Otherwise, it continues.

b) If S does not satisfy (M, ρ) in CT, it output $\perp$. Else let $I \subseteq \{1, \cdots, l\}$ be an index set, such that for all $i \in I$, $\rho(i) \in S$ and $\Sigma_{i\in I}\omega_i M_i = (1, 0, \cdots, 0)$. Define $\Delta = \{x : \exists i \in I, \rho(i) = x\}$. Compute

$$\frac{e(K, C')}{\prod_{i\in I} e(C_i, L)^{\omega_i} \cdot e(C', \prod_{x\in\Delta} K_x)^{\omega_i}} = e(g, g)^{\alpha s}.$$

Compute $R = C/e(g, g)^{\alpha s}$, $m = C_0 \oplus H_2(R)$ and $s = H_4(m, R)$. Output m if $C' = g^s$, $C'' = Q^s$ and $E = H_1(C_0, C, C', C'', D, \{C_i, E_i\}_{i\in[1,l]}, W, W_0, W_1)^s$. Otherwise output $\perp$.

(2) CT is a re-encrypted ciphertext.

a) Phase $CT_2 = (\widetilde{rk_5}, rk_5, rk_6, D', \{rk_{7,i}, {E_i}'\}_{i\in[1,l]}, W', {W_0}', {W_1}', E')$, compute $\tilde{t} = H_1(\widetilde{rk_5}, rk_5,$

$rk_6, D', \{rk_{7,i}, E_i'\}_{i\in[1,l]}, W', W_0', W_1')$. For $\forall 1 \leqslant i \leqslant l$, verify

$$e(g, E') \stackrel{?}{=} e(rk_6, \tilde{t}), \qquad (4)$$

$$e(g, rk_{7,i}) \stackrel{?}{=} e(g, f^{\lambda_i'})e(rk_6, h_{\rho(i)})^{-1}. \qquad (5)$$

Check whether equations $(4) - (5)$ hold. If not, output $\perp$. Otherwise proceed.

b) If S associated with sk does not satisfy (M, ρ) in CT, it output $\perp$. Else let $I \subseteq \{1, \cdots, l\}$ be a set of indices, such that for all $i \in I$, $\rho(i) \in S$ and $\Sigma_{i\in I}\omega_i M_i = (1, 0, \cdots, 0)$. Define $\Delta = \{x : \exists i \in I, \rho(i) = x\}$. Compute

$$\frac{e(K, rk_6)}{\prod\limits_{i\in I} e(rk_{7,i}, L)^{\omega_i} \cdot e(rk_6, \prod\limits_{x\in\Delta} K_x)^{\omega_i}} = e(g,g)^{\alpha s'}.$$

Next, compute $R' = rk_5/e(g,g)^{\alpha s'}$, $\delta = \widetilde{rk_5} \oplus H_2(R')$ and $s' = H_4(\delta, R')$. Output δ if $rk_6 = g^{s'}$ and $E' = H_1(\widetilde{rk_5}, rk_5, rk_6, D', \{rk_{7,i}, E_i'\}_{i\in[1,l]}, W', W_0', W_1')^{s'}$. Otherwise output $\perp$.

c) Compute $CT||\Gamma = S.Dec(CT_1, \delta)$, and $m = C/\Gamma^{H_3(\delta)^{-1}}$.

Consistency. The consistency is verified as:

1) For the search token, in the $Test$ algorithm we have

$$\begin{aligned} F &= e(Y'Z', D)/\left(\prod_{i\in I}(e(Y', E_i)\cdot e(D, Y_x'))^{\omega_i}\right) \\ &= \frac{e(g^{r\gamma}\cdot \tilde{g}^{r\gamma}, g^{s_2})}{\prod\limits_{i\in I}(e(g^{r\gamma}, \tilde{g}^{\varphi_i}H_1(\rho(i))^{-s_2})\cdot e(g^{s_2}, H_1(x)^{r\gamma}))^{\omega_i}} \\ &= \frac{e(g^{r\gamma}\tilde{g}^{r\gamma}, g^{s_2})}{e(g^{r\gamma}, \tilde{g})^{\sum_{i\in I}\varphi_i\omega_i}} \\ &= \frac{e(g^{r\gamma}\tilde{g}^{r\gamma}, g^{s_2})}{e(g^{r\gamma}, \tilde{g})^{s_2}} \\ &= e(g,g)^{rs_2\gamma}. \end{aligned}$$

Further, if $KW = KW'$, it can be verified that

$$\begin{aligned} & e(W, \tau_1)e(W_1, \tau_3)F \\ =& e(f_1^{s_1}, (g^a f_2^{H_1(KW')})^{\gamma})e(f_2^{s_2}, g^{\gamma(ac-r)/b})e(g,g)^{rs_2\gamma} \\ =& e(g^{cs_1}, (g^a g^{bH_1(KW')})^{\gamma})e(g^{s_2}, g^{\gamma ac}) \\ =& e(g^{c\gamma}, g^{a(s_1+s_2)} f_2^{s_1 H_1(KW')}) \\ =& e(W_0, \tau_2) \end{aligned}$$

Thus, the consistency of keyword can be verified. Note that, if CT is a re-encrypted ciphertext, it can be verified in the same manner.

2) For an original ciphertext, we have

$$\begin{aligned} & \frac{e(K, C')}{\prod\limits_{i\in I} e(C_i, L)^{\omega_i}\cdot e(C', \prod\limits_{x\in\Delta} K_x)^{\omega_i}} \\ =& \frac{e(g^{\alpha} f^t, g^s)}{\prod\limits_{i\in I} e(f^{\lambda_i} h_{\rho(i)}^{-s}, g^t)^{\omega_i}\cdot e(g^s, \prod\limits_{x\in\Delta} h_x^t)^{\omega_i}} \\ =& \frac{e(g^{\alpha} f^t, g^s)}{e(f, g^t)^{\sum_{i\in I}\lambda_i\omega_i}} \\ =& e(g,g)^{\alpha s} \end{aligned}$$

3) For a re-encrypted ciphertext, we have

$$\begin{aligned} \Gamma &= \frac{e(rk_1, C')}{e(rk_2, C'')\cdot\prod\limits_{i\in I} e(C_i, rk_3)^{\omega_i}\cdot e(C', \prod\limits_{x\in\Delta} rk_{4,x})^{\omega_i}} \\ &= \frac{e(K^{H_3(\delta)}Q^{\theta}, g^s)}{e(g^{\theta}, Q^s)\prod\limits_{i\in I} e(f^{\lambda_i} h_{\rho(i)}^{-s}, L^{H_3(\delta)})^{\omega_i} e(g^s, \prod\limits_{x\in\Delta} K_x^{H_3(\delta)})^{\omega_i}} \\ &= e(g,g)^{\alpha s H_3(\delta)} \end{aligned}$$

Later, In the Dec algorithm for a re-encrypted ciphertext, δ can be computed in the same way as above. Then, it can compute $m = C/\Gamma^{H_3(\delta)^{-1}}$.

C. Security Proof

Now we demonstrate the proof of chosen ciphertext and chosen keyword security for our CPAB-KSDS scheme. For simplicity, we assume H_1, H_2, H_3 are TCR hash functions, $SY = (S.Enc, S.Dec)$ is a symmetric encryption.

Theorem 1. CPAB-KSDS scheme is IND-CCA-Or secure if the decisional $|U|$-BDHE assumption holds.

Proof. Suppose a PPT adversary $\mathcal{A}$ can attack the IND-CCA-Or security, we could build a simulator $\mathcal{B}$ to break the $|U|$-BDHE assumption. Given a $|U|$-BDHE sample $(\vec{y} = (g, g_s, g_1, \cdots, g_{|U|}, g_{|U|+2}, \cdots, g_{2|U|}), T) \in G^{2q+1} \times G_T$, the task for $\mathcal{B}$ is to determine if $T \stackrel{?}{=} e(g,g)^{\nu^{|U|+1}s}$.

Initially, $\mathcal{B}$ maintains the following empty values.

- sk^{list}: stores tuples of (S, sk_s).
- rk^{list}: stores tuples of $(S, (M', \rho'), KW', rk, flag)$, where $flag \in \{true, false\}$, where $flag = ture$ indicates rk is a valid re-encryption key, and $flag = false$ indicates rk is random.

$\mathcal{B}$ controls random oracles H_1, H_2, H_4 as follows. $\mathcal{B}$ maintains hash lists H_1^{list}, H_2^{list}, H_4^{list} which are initially empty.

- H_1^{list}: $\mathcal{A}$ queries to H_1, if $(C_0, C, C', C'', D, \{C_i, E_i\}_{i\in[1,l]}, W, W_0, W_1, \sigma, g^{\sigma})$ exists in H_1^{list}, returns g^{σ}. Otherwise, choose a random $\sigma \in Z_p^*$ and returns g^{σ} as the answer. Adds $(C_0, C, C', C'', D, \{C_i, E_i\}_{i\in[1,l]}, W, W_0, W_1, \sigma, g^{\sigma})$ to H_1^{list}.
- H_2^{list}: $\mathcal{A}$ queries to H_2, if (R, ϕ) exists in H_2^{list}, returns ϕ. Otherwise, choose a random $\phi \in \{0,1\}^*$ as the answer. Adds (R, ϕ) to H_2^{list}.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2020.2963978, IEEE Transactions on Dependable and Secure Computing

- H_4^{list}: $\mathcal{A}$ queries to H_4, if (m, R, s) exists in H_4^{list}, returns s. Otherwise, choose a random $s \in Z_p^*$ as the answer. Adds (m, R, s) to H_4^{list}.

1) Init. The challenge $\mathcal{A}$ outputs an access policy (M^*, ρ^*) he wants to challenge. M^* is an $l^* \times n^*$ matrix, where $n^* \leq |U|$.
2) Setup. Simulator $\mathcal{B}$ chooses a random $\alpha' \in Z_p$ and sets $f = g^\nu$, $e(g,g)^\alpha = e(g,g)^{\alpha'} \cdot e(g_1, g_{|U|})$. This implicitly sets $\alpha = \alpha' + \nu^{|U|+1}$. For $\forall x, 1 \leq x \leq |U|$. Choose a random value $z_x \in Z_p$. If there exists an $i \in [1, l]$ such that $\rho^*(i) = x$, then sets $h_x = g^{z_x} g_1^{M^*_{i,1}} \cdot g_2^{M^*_{i,2}} \cdots g_{n^*}^{M^*_{i,n^*}}$. Otherwise sets $h_x = g^{z_x}$. Next, $\mathcal{B}$ randomly choose $\beta, a, b, c \in Z_p^*$, $\tilde{g} \in G$ and a symmetric encryption $SY = (S.Enc, S.Dec)$. Computes $f_1 = g^c$, $f_2 = g^b$, $Q = g^\beta$. The master secret key is (g^α, a, b), whereby g^α is unknown to $\mathcal{B}$.
3) Phase I.
   a) $\mathcal{O}_{sk}(S)$: $\mathcal{B}$ first searches sk^{list}, if (S, sk_S) exists, returns sk_S. Otherwise,
      - if $S \models (M^*, \rho^*)$, $\mathcal{B}$ aborts and outputs $\perp$.
      - Otherwise, $\mathcal{B}$ randomly choose $\mu, r \in Z_p^*$. Finds a vector $\vec{\omega} = (\omega_1, \cdots, \omega_{n^*}) \in Z_p^*$ such that $\omega_1 = -1$ and for all i where $\rho^*(i) \in S$, $\vec{\omega} \cdot M_i^* = 0$. By the definition of LSSS [37], such $\vec{\omega}$ must exists if S does not satisfy (M^*, ρ^*). Computes $L = g^\mu \prod_{i=1}^{n^*} (g_{|U|+1-i})^{\omega_i} \triangleq g^t$. This implicitly sets $t = \mu + \omega_1 \nu^{|U|} + \omega_2 \nu^{|U|-1} + \cdots + \omega_{n^*} \nu^{|U|+1-n^*}$. By this setting, $K = g^\alpha f^t = g^{\alpha' + \nu^{|U|+1}} \cdot g^{\nu(\mu + \omega_1 \nu^{|U|} + \omega_2 \nu^{|U|-1} + \cdots + \omega_{n^*} \nu^{|U|+1-n^*})} = g^{\alpha'} g^{\mu\nu} \prod_{i=2}^{n^*} (g_{|U|+2-i})^{\omega_i}$.
        For each $x \in S$, if there doesn't exist i so that $\rho^*(i) = x$, $\mathcal{B}$ computes $K_x = L^{z_x}$. Otherwise, suppose $\rho^*(i) = x$, $\mathcal{B}$ calculates K_x as
        $$K_x = L^{z_x} \prod_{j=1}^{n^*} \left(g^\mu \prod_{\substack{k=1 \\ k \neq j}}^{n^*} (g_{|U|+1+j-k})^{\omega_k} \right)^{M_{i,j}}.$$
        Next, $\mathcal{B}$ can compute V, Y, Z and Y_x as he knows a, b, c, r. Finally, $\mathcal{B}$ adds (S, sk_S) to sk^{list}.
   b) $\mathcal{O}_{token}(S, KW)$: $\mathcal{B}$ first searches sk^{list}, if (S, sk_S) exists, using sk_S to generate τ_{KW} via the $TokenGen$ algorithm. If such an entry doesn't exist, $\mathcal{B}$ queries $\mathcal{O}_{sk}(S)$ to get sk_S and then generates τ_{KW}. Adds (S, sk_S) to sk^{list}.
   c) $\mathcal{O}_{test}(CT, KW)$: $\mathcal{B}$ first queries $\mathcal{O}_{token}$ to get a search token τ_{KW}. Then runs $Test(CT, \tau)$ and returns the result to $\mathcal{A}$.
   d) $\mathcal{O}_{rk}(S, (M', \rho'), KW')$: $\mathcal{B}$ first searches rk^{list}, if $(S, (M', \rho'), KW', rk, *)$ exists, where $*$ denotes the wildcard, outputs rk. Otherwise proceeds,
      - If $S \models (M^*, \rho^*)$ and $(S', sk_{S'})$ in sk^{list}, where $S' \models (M', \rho')$, $\mathcal{B}$ aborts and outputs $\perp$. Otherwise,
      - If $S \models (M^*, \rho^*)$ but there is no tuple $(S', sk_{S'})$ in sk^{list}, where $S' \models (M', \rho')$, $\mathcal{B}$ randomly selects values for each element of rk. Adds $(S, (M', \rho'), KW', rk, false)$ to rk^{list} list. Otherwise,
      - $\mathcal{B}$ first queries $\mathcal{O}_{sk}(S)$ to get sk_S and then generates rk using sk_S via $RKeyGen$ algorithm. Adds (S, sk_S) and $(S, (M', \rho'), KW', rk, true)$ to sk^{list} and rk^{list} respectively.
   e) $\mathcal{O}_{re}(CT, S, (M', \rho'), KW')$: If $S \models (M^*, \rho^*)$ and there is a tuple $(S', sk_{S'})$ in sk^{list}, where $S' \models (M', \rho')$, $\mathcal{B}$ aborts and outputs $\perp$. Else if the equations $(1) - (3)$ do not hold, outputs $\perp$. Otherwise if there is a tuple $(S, (M', \rho'), KW', rk, *)$ in rk^{list}, re-encrypts CT with rk. Otherwise, $\mathcal{B}$ first issues $\mathcal{O}_{rk}(S, (M', \rho'), KW')$ to get rk. Next, $\mathcal{B}$ re-encrypts CT with rk, then adds $(S, (M', \rho'), KW', rk, 1)$ to rk^{list}.
   f) $\mathcal{O}_{dec}(S, CT)$: $\mathcal{B}$ proceeds,
      - If CT is a original ciphertext, $\mathcal{B}$ first verifies whether $(1) - (3)$ hold, if not, outputs $\perp$. Otherwise, $\mathcal{B}$ checks whether there exists tuples (R, ϕ) in H_2^{list} and (m, R, s) in H_4^{list}, such that $C_0 = m \oplus \phi$, $C' = g^s$. If yes, returns m to $\mathcal{A}$. Otherwise outputs $\perp$.
      - If CT is a re-encrypted ciphertext, $\mathcal{B}$ first verifies equations $(4) - (5)$, if these verification fail, outputs $\perp$. Otherwise, $\mathcal{B}$ checks whether there exists tuples (R', ϕ') in H_2^{list} and (δ, R', s') in H_4^{list}, such that $rk_5 = \delta \oplus \phi'$, $rk_6 = g^{s'}$. If yes, returns δ to $\mathcal{A}$. Otherwise outputs $\perp$. Finally $\mathcal{B}$ computes $CT || \Gamma = S.Dec(CT_1, \delta)$, and $m = C / \Gamma^{H_3(\delta)^{-1}}$. Returns m to $\mathcal{A}$.
4) Challenge. $\mathcal{A}$ selects two equal length message (m_0, m_1) and a challenge keyword KW^*. Challenger $\mathcal{C}$ randomly choose a bit $b \in \{0, 1\}$ and constructs $C_0^* = m_b \oplus H_2(R^*)$, $C^* = R^* \cdot T \cdot e(g^s, g^{\alpha'})$, $C'^* = g^s$ and $C''^* = (g^s)^\beta$.
   Then, $\mathcal{B}$ chooses random values $y'_2, \cdots, y'_{n^*} \in Z_p$. For $i = 1, \cdots, l^*$, computes
   $$C_i^* = \left(\prod_{j=1,\cdots,n^*} (g^\nu)^{y'_j M^*_{i,j}} \right) (g^s)^{-z_{\rho^*(i)}}.$$
   Randomly choose $s_1, s_2, k_2, \cdots, k_{2n-1}$ and computes $W^* = f_1^{s_1}$, $W_0^* = g^{a(s_1+s_2)} f_2^{s_1 H_2(KW^*)}$, $W_1^* = f_2^{s_2}$, $D^* = g^{s_2}$ and $\forall 1 \leqslant i \leqslant l^*$, $E_i^* = \tilde{g}^{\varphi_i} H_1(\rho^*(i))^{-s_2}$.
   Next,$\mathcal{B}$ computes $g^{\sigma^*} = H_1(C_0^*, C^*, C'^*, C''^*, D^*, \{C_i^*, E_i^*\}_{i \in [1, l^*]}, W^*, W_0^*, W_1^*)$, $E^* = (g^s)^{\sigma^*}$.
   Note that, by this setting, there exists an tuple $(C_0^*, C^*, C'^*, C''^*, D^*, \{C_i^*, E_i^*\}_{i \in [1, l^*]}, W^*, W_0^*, W_1^*, \sigma^*, g^{\sigma^*})$ in H_1^{list}. If there no such tuple, adds it to H_1^{list}.
   If $T = e(g,g)^{\nu^{|U|+1} s}$, we have $CT^* = R^* \cdot T \cdot e(g^s, g^{\alpha'}) = R^* \cdot e(g,g)^{\nu^{|U|+1} s} \cdot e(g^s, g^{\alpha'}) = R^* \cdot e(g,g)^{s\alpha}$ that is simulated perfectly.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2020.2963978, IEEE Transactions on Dependable and Secure Computing

5) Phase II. Other than the restrictions in the IND-CCA-Or game, $\mathcal{A}$ queries as it does phase I
6) **Guess.** $\mathcal{A}$ makes the guess b' and wins if $b' = b$.

When $T = e(g,g)^{\nu^{|U|+1}s}$, $\mathcal{B}$ simulators perfectly if the simulation does not abort. If T is a random element in G_T, Then CT^* is a random ciphertext, and the value b reveals nothing about CT^*. The probability of $Pr[b' = b] = \frac{1}{2}$. Thus, $\mathcal{B}$ can solve the decisional $|U|$-BDHE assumption with non-negligible advantage.

Theorem 2. Our proposed CPAB-KSDS scheme is IND-CCA-Re secure if the decisional $|U|$-BDHE assumption holds.

Proof. The Init, Setup and query Phase I is similar to these steps in the proof of Theorem 1.

1) Challenge. $\mathcal{A}$ selects two message (m_0, m_1) with equal length and a challenge keyword KW^*. Challenger $\mathcal{C}$ chooses a random bit $b \in \{0,1\}$ and constructs as follows.
   a) Generate a secret key sk_S and a re-encryption key rk, where $rk \leftarrow RKeyGen(sk_S, (M^*, \rho^*), KW^*)$.
   b) $\mathcal{B}$ generates an original ciphertext $CT \leftarrow Enc(m_b, (M, \rho), KW)$ using the same way as in Challenge phase in the proof of Theorem 1.
   c) Re-encrypts CT with re-encryption key rk to get challenge ciphertext CT^* via $CT^* \leftarrow ReEnc(CT, rk)$.
   d) Outputs the challenge ciphertext CT^* to $\mathcal{A}$.

   If $T = e(g,g)^{\nu^{|U|+1}s}$, CT^* is a valid challenge ciphertext. If T is a random value in G_T, the challenge ciphertext CT^* is independent of b from the adversary's perspective.
2) Phase II. Other than the restrictions in the IND-CCA-Re game, $\mathcal{A}$ queries as it does phase I
3) **Guess.** $\mathcal{A}$ makes the guess b' and wins if $b' = b$.

When T is randomly chosen in G_T, Then CT^* is a random ciphertext, and the value b reveals nothing about CT^*. The probability of $Pr[b' = b] = \frac{1}{2}$. Therefore, $\mathcal{B}$ can solve the decisional $|U|$-BDHE assumption with non-negligible advantage.

Theorem 3. Our proposed CPAB-KSDS scheme is IND-CKA secure if the DL assumption holds.

Proof. Suppose there exists a PPT adversary $\mathcal{A}$ can break the IND-CKA security, we built a simulator $\mathcal{B}$ to break the DL assumption. Given a DL sample $(\vec{y} = (g, z, h, z^{r_1}, g^{r_2}, T) \in G^6$, the task for $\mathcal{B}$'s is to determine if $T \stackrel{?}{=} h^{r_1+r_2}$.

$\mathcal{B}$ controls random oracle H_1 as follows. $\mathcal{B}$ maintains hash lists H_1^{list} which is initially empty.

- H_1^{list}: $\mathcal{A}$ queries to H_1, if $(x, *, \sigma_x, g^{\sigma_x})$ exists in H_1^{list}, returns g^{σ_x}. Otherwise, choose a random $\sigma_x \in Z_p^*$ and returns g^{σ_x} as the answer. Adds $(x, *, \sigma_x, g^{\sigma_x})$ to H_1^{list}.

1) Setup. $\mathcal{B}$ randomly choose $\alpha, \beta, d, \upsilon \in Z_p^*$, $f, h_1, \cdots, h_{|U|} \in G$. Sets $f_1 = z = g^c$, $h = g^a$, $g^b = z^d$, $\tilde{g} = g^\upsilon$ and $Q = g^\beta$ for some unknown a, b, c. This implicitly sets $b = cd$. Chooses a symmetric encryption $SY = (S.Enc, S.Dec)$. The master secret key is $msk = (g^\alpha, a, b)$, where a, b are unknown to $\mathcal{B}$.
2) Phase I.
   a) $\mathcal{O}_{sk}(S)$: $\mathcal{B}$ chooses random values $t, r' \in Z_p^*$ and computes the secret key as $K = g^\alpha f^t$, $= g^t$, $V = h^{1/d}/g^{r'}$, $Y = (z^d)^{r'}$, $Z = (z^d)^{\upsilon r'}$. For each $x \in S$, $\mathcal{B}$ first queries (x) to H_1 and gets σ_x and g^{σ_x}. Then $\mathcal{B}$ computes $\forall x \in S, \{K_x = h_x^t, Y_x = (z^d)^{\sigma_x r'}\}$. Note that, K, L, K_x are generated the same as the real algorithm. Denote $r \triangleq br'$, we have $V = h^{1/d}/g^{r'} = g^{a/d}/g^{r/b} = g^{ac/b}/g^{r/b} = g^{(ac-r)/b}$, $Y = (z^d)^{r'} = (g^b)^{r'} = g^r$, $Z = (z^d)^{\upsilon r'} = (g^b)^{\upsilon r'} = \tilde{g}^r$ and $Y_x = (z^d)^{\sigma_x r'} = (g^b)^{\sigma_x r'} = H_1(x)^r$. Thus,$sk_S$ is a valid secret key for S.
   b) $\mathcal{O}_{token}(S, KW)$: $\mathcal{B}$ first queries $\mathcal{O}_{sk}(S)$ to get sk_S and then generates τ_{KW}.
   c) $\mathcal{O}_{test}(CT, KW)$: $\mathcal{B}$ first queries $\mathcal{O}_{token}$ to get a search token τ_{KW}. Then runs $Test(CT, \tau)$ and returns the result to $\mathcal{A}$.
   d) $\mathcal{O}_{rk}(S, (M', \rho'), KW')$: $\mathcal{B}$ first queries $\mathcal{O}_{sk}(S)$ to get a private key sk_S. Then runs $RKeyGen(sk_S, (M', \rho'), KW')$ and returns the result to $\mathcal{A}$.
   e) $\mathcal{O}_{dec}(S, CT)$: $\mathcal{B}$ uses α to generate a corresponding sk_S and returns the decryption $Dec(sk_S, CT)$ result to $\mathcal{A}$.
3) Challenge. $\mathcal{A}$ chooses two keywords (KW_0, KW_1) with equal length, a challenge message m^* and access policy (M^*, ρ^*), where M^* is a $l^* \times n^*$ matrix. If $\mathcal{A}$ has made a query $\mathcal{O}_{token}(S, KW)$, $S \models (M^*, \rho^*)$, $\mathcal{B}$ aborts and outputs $\perp$. Otherwise, $\mathcal{B}$ chooses a random bit $b \in \{0,1\}$, $s \in Z_p^*$. Constructs $C_0^* = m^* \oplus H_2(R^*)$, $C^* = R^* \cdot e(g,g)^{\alpha s}$, $C'^* = g^s$ and $C''^* = Q^s$. For $i = 1, \cdots, l^*$, computes $C_i^* = f^{\lambda_i} h_{\rho^*(i)}^{-s}$. Computes $W^* = z^{r_1}$, $W_0^* = T \cdot z^{r_1 d H_2(KW_b)}$, $W_1^* = z^{r_2 d}$, $D^* = g^{r_2}$ and $\forall 1 \leqslant i \leqslant l^*, E_i^* = \tilde{g}^{\varphi_i} g^{r_2 \sigma_{\rho^*(i)}}$. Next,$\mathcal{B}$ computes $g^{\sigma^*} = H_1(C_0^*, C^*, C'^*, C''^*, D^*, \{C_i^*, E_i^*\}_{i \in [1, l^*]}, W^*, W_0^*, W_1^*)$, $E^* = g^{s\sigma^*}$.

   If $T = h^{r_1+r_2}$, we have $W_0^* = T \cdot z^{r_1 d H_2(KW_b)} = h^{r_1+r_2} \cdot z^{r_1 d H_2(KW_b)} = g^{a(r_1+r_2)} f^{s_1 H_2(KW_b)}$. Thus, CT^* is a correctly generated challenge ciphertext.

   Note that, CT^* can also be $CT^* = ReEnc(Enc(m^*, (M, \rho), KW'), rk)$, where $rk \leftarrow RKeyGen(sk_S, (M^*, \rho^*), KW_b)$, $S \models (M, \rho)$.
4) Phase II. $\mathcal{A}$ makes queries as in phase I other than the restrictions in the IND-CKA game.
5) **Guess.** $\mathcal{A}$ makes the guess b' and wins if $b' = b$.

When $T = h^{r_1+r_2}$, $\mathcal{B}$ simulators perfectly if the simulation does not abort. If T is randomly chosen in G, KW_b is hidden from the adversary and b reveal nothing about CT^*. The probability of $Pr[b' = b] = \frac{1}{2}$. Therefore, $\mathcal{B}$ can solve the DL assumption with non-negligible advantage.

V. PERFORMANCE

To evaluate the performance, our scheme is compared with the recently proposed search encryption scheme [30], attribute based keyword search schemes [34], [35] and KPAB-PRE-KS

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2020.2963978, IEEE Transactions on Dependable and Secure Computing

TABLE I
FUNCTIONALITY COMPARISON WITH [30], [34], [35], [36].

Schemes	Keyword Search?	Data Sharing?	Access Policy	Without interactive with PKG?	private key or public key setting?
[30]	✓	×	×	✓	private key
[34]	✓	×	Ciphertext policy	✓	public key
[35]	✓	×	Ciphertext policy	✓	public key
[36]	✓	✓	Key policy	×	public key
Ours	✓	✓	Ciphertext policy	✓	public key

TABLE II
COMPUTATION COMPARISON WITH [30], [34], [35], [36].

Schemes	Enc	TokenGen	Test	ReEnc	Dec(Or)	Dec(Re)
[30]	$\mathcal{O}(\lambda^2) \cdot m$	$\mathcal{O}(\lambda^2) \cdot m$	$\mathcal{O}(\lambda) \cdot m$	$\perp$	$\perp$	$\perp$
[34]	$\mathcal{O}(l) \cdot e + \mathcal{O}(1) \cdot p$	$\mathcal{O}(\|S\|) \cdot e$	$\mathcal{O}(\|S\|) \cdot (e+p)$	$\perp$	$\perp$	$\perp$
[35]	$\mathcal{O}(l) \cdot e$	$\mathcal{O}(\|S\|) \cdot e$	$\mathcal{O}(\|S\|) \cdot p + \mathcal{O}(1) \cdot e$	$\perp$	$\perp$	$\perp$
[36]	$\mathcal{O}(\|S\|) \cdot e + \mathcal{O}(1) \cdot p$	$\mathcal{O}(l) \cdot e$	$\mathcal{O}(\|S\|) \cdot e + \mathcal{O}(1) \cdot p$	$\mathcal{O}(\|S\|) \cdot e + \mathcal{O}(1) \cdot p$	$\mathcal{O}(\|S\|) \cdot e + \mathcal{O}(1) \cdot p$	$\mathcal{O}(\|S\|) \cdot e + \mathcal{O}(1) \cdot p$
Ours	$\mathcal{O}(l) \cdot e + \mathcal{O}(1) \cdot p$	$\mathcal{O}(\|S\|) \cdot e$	$\mathcal{O}(\|S\|) \cdot (e+p)$	$\mathcal{O}(\|M\|) \cdot (e+p)$	$\mathcal{O}(\|M\|) \cdot (e+p)$	$\mathcal{O}(\|M\|) \cdot (e+p)$

TABLE III
IMPLEMENTATION TIME.

Algorithms	KeyGen (ms)	Enc (ms)	TokenGen (ms)	Test (ms)	RKenGen (ms)	ReEnc (ms)	Dec(Or) (ms)	Dec(Re) (ms)
$\|S\| = 5$	12.954	40.003	7.232	16.171	51.667	33.914	9.463	31.640
$\|S\| = 10$	19.934	67.811	10.515	24.083	86.230	61.216	17.682	58.685
$\|S\| = 15$	26.146	98.433	13.810	32.006	120.610	88.598	25.720	87.315
$\|S\| = 20$	33.624	125.362	17.106	39.826	157.500	117.622	34.438	116.067
$\|S\| = 25$	40.479	152.616	20.392	47.753	191.435	142.800	42.745	141.702
$\|S\| = 30$	46.616	181.117	23.673	55.647	226.063	171.027	50.708	169.145

scheme [36]. We have made a thorough comparison based on the following aspects: functionality, theoretical analysis efficiency and implementation time.

A. Functionality Comparison

Table I summarizes that our scheme supports the data sharing and keyword search functionality whereas schemes [30], [34], [35] cannot provide the data sharing property. Moreover, the scheme [30] works in the private key setting while [34], [35], [36] and our scheme work in the public key setting. When compared with the KPAB-PRE-KS scheme [36], it requires the delegator to interactive with the PKG to generate the re-encryption key every time. Our proposed scheme, instead, works in a ciphertext-policy model without involving the PKG to generate the re-encryption key which reduces the burden for PKG.

B. Efficiency Theoretical Analysis

Table II illustrates the difference of our scheme, searchable encryption scheme [30], CPAB-KS scheme [34], [35] and KPAB-PRE-KS scheme [36], regarding the computation cost. In Table II, λ denotes the security parameter in scheme [30], $|S|$ is the size of the attributes in an attribute set S. l is the total row numbers in an access policy (M, ρ), p is the cost of a bilinear pairing computation, e is the computation of an exponentiation operation in an group G or G_T and m is the computation cost of the multiplication of two real numbers. $Dec(Or)$ is the decryption of an original ciphertext while $Dec(Re)$ is the decryption computation of a re-encrypted ciphertext. Let $|M| = max\{|S|, l\}$ denote the larger one between $|S|$ and l. Compared to the complexity of computing an exponentiation, the cost of the hash operation in our scheme is neglected here as it has minimal impact on the efficiency.

As shown in Table II, in the private key searchable encryption scheme [30], the computation costs of Enc and $TokenGen$ algorithms are linear with the square of the security parameter and the $Test$ algorithm cost is linear as well. Considering the public key searchable encryption schemes, the efficiency of our scheme is almost identical to

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2020.2963978, IEEE Transactions on Dependable and Secure Computing

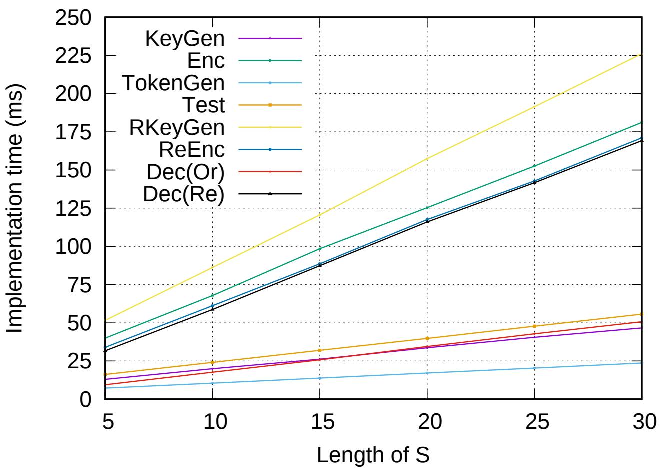


Fig. 2. Implementation Time.

the CPAB-KS scheme [34] while our scheme does cost more in the $Test$ phase compared to [35]. It is because our scheme supports the data sharing functionality, which requires extra operations in the computation. When compared to KPAB-PRE-KS scheme [36], the $KeyGen$, Enc and $TokenGen$ computation cost of our scheme are almost the same with [36]. Regarding the computation cost of $Test$, $ReEnc$, $Dec(Or)$ and $Dec(Rec)$, our scheme cost a little more than KPAB-PRE-KS scheme since our scheme needs more bilinear pairing computation. The main reason is that interaction with a PKG is not required and we need separate each attribute as the input to a bilinear map while the KPAB-PRE-KS scheme uses the continuously multiply of attributes as one input to the bilinear map. However, the one input in the KPAB-PRE-KS scheme requires the participation of the PKG. So we believe our scheme is still better since no PKG involving is beneficial to reduce the computational cost. In our scheme, no more interaction with the PKG at the stage when the delegator computes the re-encryption key. The elimination of PKG can significantly decrease the overall burden of the PKG.

C. Implementation

We use Go language to take the advantage of open source Golang PBC package [39] which supports a wrapper to a Pairing-Based Cryptography library (PBC) [40] written in C. The CPU used in the implementation is Intel i5-8250U @1.60GHZ with a 8GB RAM. The chosen elliptic curve is $Y^2 = X^3 + X$ and the order of the group is 160 bit. In order to get a more accurate average execution time, the experiment was done 20 different times.

The universal attribute is set to $|U| = 1000$. Let $|S| = 5$ in the $KenGen$ algorithm. Let the row $l = 5$ for an access policy (M, ρ) and for each row $1 \leq i \leq l$, $\rho(i)$ corresponds to a distinct attribute is S. Table III summarizes the running time. Further, $|S|$ and l have been varied from 5 to 30 with step 5.

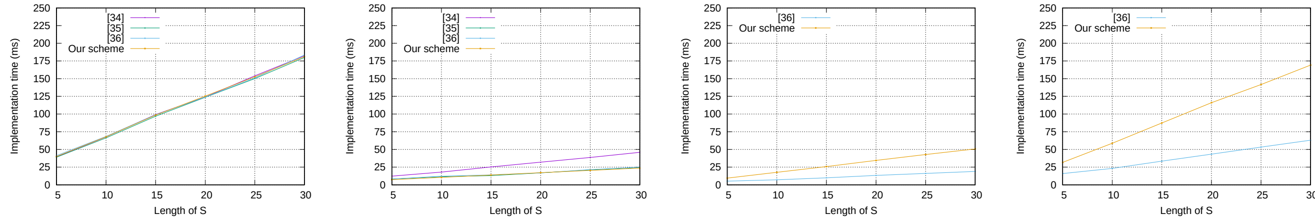
We compare the execution time of the algorithms in Table III and Figure 2. It is clear that the execution time of $KeyGen$, Enc, $TokenGen$, $Test$, $RKeyGen$, $ReEnc$, $Dec(Or)$ and $Dec(Re)$ algorithms are nearly linear to the size of S, which matches our theoretical analysis. From Table III, one may think that the re-encryption functionality is useless since the Enc algorithm only takes about 80% of the running time of the $RKeyGen$ algorithm. The delegator can re-execute the Enc algorithm to generate a ciphertext with the new policy and keyword. However, applying the proposed proxy re-encryption manner offers two benefits over re-running Enc. First, once the re-encryption key is generated, it can be used to re-encrypt the delegator's ciphertext multiple times and reduces the delegator's computation cost in total. Second, if the delegator chooses to re-execute the Enc algorithm, he should first download the ciphertext from the cloud server, decrypt the ciphertext to retrieve the underling plaintext and then encrypt the plaintext with the new policy and keyword. Moreover, downloading data from the cloud brings a new problem for data maintenance.

We also compare the implementation time of our scheme with the previous schemes [34], [35], [36] as they all work in the public key setting and support the access policy on the user's identity. Note that, we did not compare the implementation time with scheme [30] as scheme [30] works in the private key setting and does not support the access policy on the user's identity. Here, we make a comparison of the Enc, $TokenGen$, $Dec(Or)$ and $Dec(Re)$ algorithms as these algorithms are executed on the user's side. Fig 3(a) shows that the Enc algorithm computation cost of our scheme is almost identical to the schemes [34], [35] and [36]. From Fig 3(b), we can see that the $TokenGen$ algorithm of our scheme is almost as efficient as [35] and [36], and more efficient than scheme [34]. As shown in Fig 3(c) and 3(d), the $Dec(Or)$ and $Dec(Re)$ algorithms computation costs of ours scheme are higher than that of scheme [36]. However, as we analyzed in subsection V-B, our scheme does not need to interact with the PKG and thus reduces the burden of the PKG.

VI. CONCLUSION

In this work, a new notion of ciphertext-policy attribute-based mechanism (CPAB-KSDS) is introduced to support keyword searching and data sharing. A concrete CPAB-KSDS scheme has been constructed in this paper and we prove its CCA security in the random oracle model. The proposed scheme is demonstrated efficient and practical in the performance and property comparison. This paper provides an affirmative answer to the open challenging problem pointed out in the prior work [36], which is to design an attribute-based encryption with keyword searching and data sharing without the PKG during the sharing phase. Furthermore, our work motivates interesting open problems as well including designing CPAB-KSDS scheme without random oracles or proposing a new scheme to support more expressive keyword search.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2020.2963978, IEEE Transactions on Dependable and Secure Computing



(a) Enc algorithm (b) $TokenGen$ algorithm (c) $Dec(Or)$ algorithm (d) $Dec(Re)$ algorithm

Fig. 3. Implementation Time Comparison.

Acknowledgments

The work has been supported by NSF of China (61702236, 61672270, 61872181), etc.

We would like to thank the anonymous reviewers for very useful comments to an earlier version of this paper.

References

[1] A. Sahai and B. Waters, “Fuzzy identity-based encryption,” in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 457–473, Springer, 2005.

[2] V. Goyal, O. Pandey, A. Sahai, and B. Waters, “Attribute-based encryption for fine-grained access control of encrypted data,” in *Proceedings of the 13th ACM conference on Computer and communications security*, pp. 89–98, Acm, 2006.

[3] J. Bethencourt, A. Sahai, and B. Waters, “Ciphertext-policy attribute-based encryption,” in *Security and Privacy, 2007. SP’07. IEEE Symposium on*, pp. 321–334, IEEE, 2007.

[4] B. Waters, “Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization,” in *International Workshop on Public Key Cryptography*, pp. 53–70, Springer, 2011.

[5] H. Qian, J. Li, Y. Zhang, and J. Han, “Privacy-preserving personal health record using multi-authority attribute-based encryption with revocation,” *International Journal of Information Security*, vol. 14, no. 6, pp. 487–497, 2015.

[6] J. Liu, X. Huang, and J. K. Liu, “Secure sharing of personal health records in cloud computing: Ciphertext-policy attribute-based signcryption,” *Future Generation Computer Systems*, vol. 52, pp. 67–76, 2015.

[7] L. Fang, W. Susilo, C. Ge, and J. Wang, “Interactive conditional proxy re-encryption with fine grain policy,” *Journal of Systems and Software*, vol. 84, no. 12, pp. 2293–2302, 2011.

[8] K. Emura, A. Miyaji, A. Nomura, K. Omote, and M. Soshi, “A ciphertext-policy attribute-based encryption scheme with constant ciphertext length,” in *International Conference on Information Security Practice and Experience*, pp. 13–23, Springer, 2009.

[9] S. Hohenberger and B. Waters, “Attribute-based encryption with fast decryption,” in *Public-Key Cryptography–PKC 2013*, pp. 162–179, Springer, 2013.

[10] A. Lewko and B. Waters, “New proof methods for attribute-based encryption: Achieving full security through selective techniques,” in *Advances in Cryptology–CRYPTO 2012*, pp. 180–198, Springer, 2012.

[11] M. Li, S. Yu, Y. Zheng, K. Ren, and W. Lou, “Scalable and secure sharing of personal health records in cloud computing using attribute-based encryption,” *IEEE transactions on parallel and distributed systems*, vol. 24, no. 1, pp. 131–143, 2012.

[12] L. Zhang, G. Hu, Y. Mu, and F. Rezaeibagha, “Hidden ciphertext policy attribute-based encryption with fast decryption for personal health record system,” *IEEE Access*, vol. 7, pp. 33202–33213, 2019.

[13] M. Green, S. Hohenberger, B. Waters, *et al.*, “Outsourcing the decryption of abe ciphertexts.,” in *USENIX Security Symposium*, vol. 2011, 2011.

[14] J. Lai, R. H. Deng, C. Guan, and J. Weng, “Attribute-based encryption with verifiable outsourced decryption,” *IEEE Transactions on information forensics and security*, vol. 8, no. 8, pp. 1343–1354, 2013.

[15] J. Li, X. Huang, J. Li, X. Chen, and Y. Xiang, “Securely outsourcing attribute-based encryption with checkability,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 8, pp. 2201–2210, 2013.

[16] M. Blaze, G. Bleumer, and M. Strauss, “Divertible protocols and atomic proxy cryptography,” in *International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 127–144, Springer, 1998.

[17] G. Ateniese, K. Fu, M. Green, and S. Hohenberger, “Improved proxy re-encryption schemes with applications to secure distributed storage,” *ACM Transactions on Information and System Security (TISSEC)*, vol. 9, no. 1, pp. 1–30, 2006.

[18] B. Libert and D. Vergnaud, “Unidirectional chosen-ciphertext secure proxy re-encryption,” *IEEE Transactions on Information Theory*, vol. 57, no. 3, pp. 1786–1802, 2011.

[19] M. Green and G. Ateniese, “Identity-based proxy re-encryption,” in *Applied Cryptography and Network Security*, pp. 288–306, Springer, 2007.

[20] C. Ge, W. Susilo, J. Wang, and L. Fang, “Identity-based conditional proxy re-encryption with fine grain policy,” *Computer Standards & Interfaces*, vol. 52, pp. 1–9, 2017.

[21] X. Liang, Z. Cao, H. Lin, and J. Shao, “Attribute based proxy re-encryption with delegating capabilities,” in *Proceedings of the 4th International Symposium on Information, Computer, and Communications Security*, pp. 276–286, ACM, 2009.

[22] S. Luo, J. Hu, and Z. Chen, “Ciphertext policy attribute-based proxy re-encryption,” in *International Conference on Information and Communications Security*, pp. 401–415, Springer, 2010.

[23] K. Liang, L. Fang, W. Susilo, and D. S. Wong, “A ciphertext-policy attribute-based proxy re-encryption with chosen-ciphertext security,” in *Intelligent Networking and Collaborative Systems (INCoS), 2013 5th International Conference on*, pp. 552–559, IEEE, 2013.

[24] K. Liang, M. H. Au, W. Susilo, D. S. Wong, G. Yang, and Y. Yu, “An adaptively cca-secure ciphertext-policy attribute-based proxy re-encryption for cloud data sharing,” in *International Conference on Information Security Practice and Experience*, pp. 448–461, Springer, 2014.

[25] C. Ge, W. Susilo, J. Wang, Z. Huang, L. Fang, and Y. Ren, “A key-policy attribute-based proxy re-encryption without random oracles,” *The Computer Journal*, vol. 59, no. 7, pp. 970–982, 2016.

[26] C. Ge, W. Susilo, L. Fang, J. Wang, and Y. Shi, “A cca-secure key-policy attribute-based proxy re-encryption in the adaptive corruption model for dropbox data sharing system,” *Designs, Codes and Cryptography*, pp. 1–17, 2018.

[27] K. Liang, M. H. Au, J. K. Liu, W. Susilo, D. S. Wong, G. Yang, T. V. X. Phuong, and Q. Xie, “A dfa-based functional proxy re-encryption scheme for secure public cloud data sharing,” *IEEE Transactions on Information Forensics and Security*, vol. 9, no. 10, pp. 1667–1680, 2014.

[28] D. X. Song, D. Wagner, and A. Perrig, “Practical techniques for searches on encrypted data,” in *Proceeding 2000 IEEE Symposium on Security and Privacy. S&P 2000*, pp. 44–55, IEEE, 2000.

[29] C. Wang, N. Cao, J. Li, K. Ren, and W. Lou, “Secure ranked keyword search over encrypted cloud data,” in *2010 IEEE 30th international conference on distributed computing systems*, pp. 253–262, IEEE, 2010.

[30] B. Wang, S. Yu, W. Lou, and Y. T. Hou, “Privacy-preserving multi-keyword fuzzy search over encrypted data in the cloud,” in *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*, pp. 2112–2120, IEEE, 2014.

[31] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, “Public key encryption with keyword search,” in *International conference on the theory and applications of cryptographic techniques*, pp. 506–522, Springer, 2004.

[32] D. Boneh and B. Waters, “Conjunctive, subset, and range queries on encrypted data,” in *Theory of Cryptography Conference*, pp. 535–554, Springer, 2007.
[33] Q. Zheng, S. Xu, and G. Ateniese, “Vabks: verifiable attribute-based keyword search over outsourced encrypted data,” in *Infocom, 2014 proceedings IEEE*, pp. 522–530, IEEE, 2014.
[34] H. Yin, J. Zhang, Y. Xiong, L. Ou, F. Li, S. Liao, and K. Li, “Cp-abse: A ciphertext-policy attribute-based searchable encryption scheme,” *IEEE Access*, vol. 7, pp. 5682–5694, 2019.
[35] Y. Miao, X. Liu, K.-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 Transactions on Dependable and Secure Computing*, 2019.
[36] K. Liang and W. Susilo, “Searchable attribute-based mechanism with efficient data sharing for secure cloud storage,” *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 9, pp. 1981–1992, 2015.
[37] Beimel and Amos, *Secure schemes for secret sharing and key distribution*. Technion-Israel Institute of technology, Faculty of computer science, 1996.
[38] L. Fang, W. Susilo, C. Ge, and J. Wang, “Public key encryption with keyword search secure against keyword guessing attacks without random oracle,” *Information Sciences*, vol. 238, pp. 221–241, 2013.
[39] Nik-U, “Pbc package,” *Online: https://github.com/Nik-U/pbc*, 2015.
[40] B. Lynn *et al.*, “Pbc library,” *Online: http://crypto.stanford.edu/pbc*, 2006.

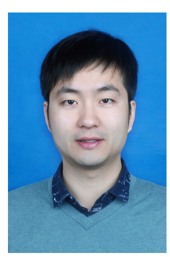
Chunpeng Ge received the Ph.D degree in Computer Science from Nanjing University of Aeronautics and Astronautics in 2016. He is now a research fellow of Singapore University of Technology and Design. His current research interests include cryptography, information security and privacy preserving for blockchain. His recent work has focused on the topics of public key encryption with keyword search, proxy re-encryption, identity-based encryption, and techniques for resistance to CCA attacks.

Willy Susilo is a Professor in the School of Computing and Information Technology, Faculty of Informatics in University of Wollongong, Australia. He is the director of Institute of Cybersecurity and Cryptology, School of Computing and Information Technology, University of Wollongong. He obtained his PhD from the University of Wollongong in 2001. He has published more than 400 papers in journals and conference proceedings in cryptography and network security. He has served as the program committee member of several international conferences. He is a senior member of IEEE.

Zhe Liu received the BS and MS degrees in Shandong University in 2008 and 2011, respectively. He is a professor in College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics(NUAA), China. Before joining NUAA, he was a researcher in SnT, University of Luxembourg, Luxembourg. He received his Ph.D degree Laboratory of Algorithmics, Cryptology and Security (LACS), University of Luxembourg, Luxembourg in 2015. His Ph.D thesis has received the prestigious FNR Awards 2016 – Outstanding PhD Thesis Award for his contributions in cryptographic engineering on IoT devices. His research interests include computer arithmetic and information security. He has co-authored more than 70 research peer-reviewed journal and conference papers.

Jinyue Xia received the Ph.D degree in Computer Science from University of North Carolina at Charlotte, USA in 2017. His current research interests include data security, cryptography and information security. His recent work has focused on the topics of public key encryption with proxy re-encryption and identity-based encryption.

Pawel Szalachowski received his PhD degree in Computer Science from Warsaw University of Technology in 2012. He is currently an Assistant Professor at the Information Systems Technology and Design Pillar, SUTD. Prior to joining SUTD, he was a senior researcher at ETH Zurich, where he led the design and implementation of the SCION architecture. His current research interests include blockchain and system security.



Liming Fang received the Ph.D degree in Computer Science from Nanjing University of Aeronautics and Astronautics in 2012, and has been a postdoctor in the information security from City University of Hong Kong. He is the associate professor at the School of Computer Science, Nanjing University of Aeronautics and Astronautics. Now, he is a visiting scholar of the Department of Electrical and Computer Engineering New Jersey Institute of Technology. His current research interests include cryptography and information security. His recent work has focused on the topics of public key encryption with keyword search, proxy re-encryption, identity-based encryption, and techniques for resistance to CCA attacks.