Multi-client Sub-Linear Boolean Keyword Searching for Encrypted Cloud Storage with Owner-enforced Authorization

Kai Zhang, Mi Wen, Member, IEEE, Rongxing Lu Senior Member, IEEE, and Kefei Chen

Abstract—To date, cloud computing has emerged as a primary utility for providing remote data storage services for users, since users can thus be relieved from cumbersome document maintenance. Despite of the benefits brought by data outsourcing, the unexpected data breaches raise concerns about data confidentiality and privacy. To deal with this, a straightforward and convincing strategy is to encrypt data before outsourcing them to the cloud. However, securely sharing and searching over outsourced encrypted data has turned into a challenge due to the hindrance led by data encryption. To address the challenge, this paper proposes a new highly-scalable searchable encryption scheme for encrypted cloud storage. The scheme achieves sub-linear Boolean keyword searching, and moreover allows the data owner to authorize which clients could search or access the documents in cloud. Technically, we revisit searchable symmetric encryption primitive by non-trivially combining it with a novel access control technique, and build inverted index data structure for both attribute-based access control and sub-linear search process. Furthermore, we introduce a formalized security definition for the system, and prove its security in the simulation-based security model. Finally, we conduct a couple of experiments over a representative real-world dataset to show practicality.

Index Terms—Cloud Security, Searchable Encryption, Access Control, Keyword Search, Boolean Query.

1 INTRODUCTION

CLOUD services have taken a promising trend in providing ubiquitous and on-demand access to a shared pool of configurable storage/computing resources [1] for users in recent years, thereby users are increasingly outsource personal documents to cloud server. The personal documents usually involve some sensitive information (such as passwords, identifiable information and health information), there have been many incidents of outsourced data leaking to adversarial attackers or to public [2]. As discovered by academic [3] and insustry [4], data breaches are considered to be primary security risks in cloud computing (untrusted outsiders) over the past decade. Hence, data confidentiality and privacy have been a primary area of focus.

To deal with the security challenge, it has been strongly suggested to employ an “encryption-before-outsourcing” mechanism: a data owner encrypts his/her documents before outsources them to cloud. Nevertheless, simple encryption of outsourced data will certainly hinder the efficiency of data processing compared to plaintext data domain. Hence, searching documents archive and sharing documents in cloud have turned into an arduous challenge. Therefore, the focus of this work, efficient document retrieval/sharing and the privacy preservation in data usage should not be conflicting goals.

A Motivation Case. Considering an example of documents retrieval and sharing in encrypted cloud storage. Assume the data warehouse of a company is hosted in cloud, there are three staffs Bob, Charles and David from different departments collaboratively processing routine work based on cloud., i.e., searching documents archive and sharing documents. To preserve data confidential and privacy, the outsourced data are encrypted before outsourcing to the cloud. For instance, a financial office staff Bob encrypts a set of financial reports where these reports origins from North America Division is accepted by the authorization policy. Here, the owner-enforced authorization policy for which staffs could access these files is formulated as “Financial Office” AND “North America Division”. To efficiently retrieve target documents from massive documents stored in cloud, a staff carries out documents searching with submitting a search query expression based on documents’ keywords.

Hence, a manager Charles of the financial office in Asia-Pacific division is certainly unable to access these reports, since his description attributes set (“Financial Office”, “Asia-Pacific Division”) fail to satisfy the authorization policy. Nevertheless, a VP David of the finical office in North America Division is accepted by the authorization policy. Moreover, when to check the 2017-year or 2018-year annual financial reports except for advertising expenses, David submits an searching expression: “North America Division” AND (“2017-Year” OR “2018-Year”) NOT “Advertising”, and
generates a corresponding search token to cloud. After a check on whether David satisfies the authorization policy, the cloud uses the search token to search and return financial reports whose keywords set matches David’s search query expression.

We can observe that the data owner enforces an attribute-based authorization policy for determining which staffs have documents access/searching rights. Therefore, only staffs whose description attributes satisfy authorization policy could retrieve target documents, where the associated keywords match a queried boolean keyword search formula. Hence, a problem from this example arises naturally: is there a secure and efficient multi-user documents retrieval and sharing system that supports expressive search patterns and access control sharing policy?

Single Keyword Search over Encrypted Data. Song et al. [15] introduced the searchable encryption (SE) primitive that allowed a remote server to search over encrypted data with an authorized search token from clients. This enables efficient searching over encrypted documents and makes the server learn nothing about plaintext data. Nevertheless, [15] with the following work [16], [17], [18] only considered symmetric case (single data owner and single client), which cannot be deployed in real-world. This is because documents are usually shared across a couple of clients rather simple “one-to-one” data sharing scene in public cloud. To accommodate multiple clients supporting fine-grained authorization, the attribute-based searchable encryption (ABSE) primitive was introduced in [5], which combined attribute-based encryption with other cryptographic primitives (e.g., proxy based re-encryption). Recently, the ABSE works have been intensively researched [6], [7], [8].

Generally, a data owner in ABSE systems is able to specify an authorization policy and thus determine which clients could search over encrypted data in cloud. Nevertheless, the query clients can only insert just one keyword (e.g., “w1”) into a query pattern, and unfortunately retrieve all documents associated with the keyword “w1”. As can be seen, single keyword searching SE systems bring about forced search expression for query clients, which still suffer from expensive linear searching costs $O(\#doc)$ (where “$\#doc$” is the total number of all outsourced documents). When extended to process a conjunctive keyword search (e.g. “$w_1 \land w_2 \land \cdots \land w_q$”) across different keywords, the searching costs in ABSE systems [5], [6], [7], [8], [9] are still in relation with the total number of all outsourced documents. Particularly for highly-scalable documents outsourcing, such ABSE systems may not be directly deployed due to unacceptable search costs and/or limited search expressions. Hence, more efficient and expressive searchable encryption systems are highly appreciated.

Boolean Keyword Search over Encrypted Data. Quite recently, Cash et al. [11] designed a searchable symmetric encryption (SSE) system supporting boolean queries, that is, $w_1 \land \psi(w_2, \cdots, w_q)$ where $\psi$ is a boolean formula over keywords ($w_2, \cdots, w_q$). The searching costs are only associated with the least frequent term in the conjunction and thus reduced to a sub-linear complexity (independent of the total number of all stored documents). And the work [11] used inverted-index data structure to organize documents. Given a set of documents invert-indexed with a keywords set ($w_1, w_2, w_3, w_4$), the boolean keyword search pattern “$w_1 \land (w_2 \lor w_3 \land w_4)$” supports negations, disjunctions, threshold etc. formulas for keywords searching, which certainly involves conjunctive and single keyword search.

However [11] only considered a symmetric situation, that is, a same secret key was used in both documents encryption and documents searching process. This greatly limits its wide deployments in multi-client collaborative cloud services. Following it, the works [12], [13] considered richer outsourcing scenarios for SSE schemes, supporting single data owner and multiple query clients.

Motivation and Utility. So far, existing SE-based solutions either only considered owner-enforced authorization but suffer from strong search expression and/or expensive search costs; or only achieve sub-linear Boolean keyword search but not support fine-grained authorization towards multiple clients. Hence, a searchable encryption system that supports both sub-linear boolean query and fine-grained authorization across multiple clients simultaneously is still an unaddressed problem.

1.1 Our Contributions

This work proposes a novel multi-client searchable encryption scheme, which achieves sub-linear boolean query and supports owner-enforced attribute-based authorization across multiple clients. Formally, the key features of the SE system are summarized as follows, where Table 1 also shows a functionality comparison with related works.

1) Supporting owner-enforced attribute-based authorization for multiple clients data sharing. The data owner encrypts the outsourced documents under a specified attribute-based authorization policy, in such a way that, it can non-interactively determine which clients could search or access the stored documents in cloud. This authorization in our SE system is similar to the attribute-based control paradigm in ABE primitive, where all clients are depicted by a description attributes set and the fine-grained authorization policy is an “AND” formula over attributes. We note that the attribute-based authorization paradigm is not employed in a trivial way, since a subtle combination between a boolean keyword search and an “AND”-gate access policy has to be built in indispensable (c.f. technical details in Section 4.1).

2) Supporting Boolean keyword search with sub-linear complexity. For documents searching, a query client submits a boolean expression of keywords and thus generates a search token for cloud. With the search token, the cloud returns target documents under a condition: not only the documents’ associated keywords match the submitted boolean expression but also the attributes of a query client satisfy owner-enforced authorization policy. To process a boolean query with sub-linear search costs (independent of the total number of stored documents) for cloud, we introduce a new inverted index method for attribute-based authorization paradigm, to deal with both documents encryption and attribute-based authorization tuples in a two-fold way (c.f. technical details in Section 4.1).
Moreover, we introduce a formal security definition for the SE system and moreover give a rigorous simulation-based security analysis. To further illustrate practical usability, we conduct a couple of experiments over a real-world dataset [19] (Enron: http://www.cs.cmu.edu/~./enron/) for the proposed system and related work.

**Organization.** Notations and background knowledge are described in Section 2. Section 3 introduces formal system model, security guarantee model and design goals of this work. We provide a main construction supporting conjunctive keyword search and its security analysis in Section 4 and Section 5, then extend it to process boolean query in Section 6. A comparison between related work and a simulated experiment are given in Section 7. Section 8 concludes this work.

## 2 Background Knowledge

The notations that used in this paper are shown in Table 2.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\kappa$</td>
<td>A security parameter.</td>
</tr>
<tr>
<td>$[n]$</td>
<td>{1, 2, \ldots, n}.</td>
</tr>
<tr>
<td>$A$</td>
<td>An array.</td>
</tr>
<tr>
<td>$</td>
<td>A</td>
</tr>
<tr>
<td>att</td>
<td>An attribute.</td>
</tr>
<tr>
<td>$\Gamma$</td>
<td>An attribute set $\Gamma := (\text{att}_1, \text{att}_2, \ldots, \text{att}_n)$.</td>
</tr>
<tr>
<td>$N'$</td>
<td>An attribute universe.</td>
</tr>
<tr>
<td>$\text{ind}$</td>
<td>The indice of a document.</td>
</tr>
<tr>
<td>$\omega$</td>
<td>A keyword.</td>
</tr>
<tr>
<td>$W$</td>
<td>A set of keywords $W := (\omega_1, \omega_2, \ldots, \omega_q)$.</td>
</tr>
<tr>
<td>Doc</td>
<td>A document Doc is labeled with $(\text{ind}, W_{\text{ind}})$.</td>
</tr>
<tr>
<td>ACC</td>
<td>An access policy.</td>
</tr>
<tr>
<td>id</td>
<td>A client’s identity.</td>
</tr>
<tr>
<td>DB</td>
<td>An outsourced database.</td>
</tr>
<tr>
<td>DB[$w$]</td>
<td>The indices of documents that associated with keyword $w$ in DB.</td>
</tr>
<tr>
<td>DB[$w$, id]</td>
<td>The indices of documents that the access policy can be satisfied by client id in DB[$w$].</td>
</tr>
<tr>
<td>$id \in$ ACC</td>
<td>The client id can satisfy the access policy ACC.</td>
</tr>
<tr>
<td>s-term</td>
<td>The least frequent keyword among the keywords in a query.</td>
</tr>
<tr>
<td>xterm</td>
<td>Any queried keyword in a query.</td>
</tr>
</tbody>
</table>

Fig. 1 lists two “keyword search index” data structures in search engines for finding target documents where keyword occurs: forward index and inverted index, in which each document is labeled with a set of keywords and indice pair. The traditional forward index needs to list all documents with all related keywords based on existing keywords list for each document, which brings about expensive time-consuming costs and memory storage for documents searching. Instead, as the most popular data structure in search engines, the inverted index builds a set of pointers to documents for each same keyword. As a result, search engines could search documents with a search token to reduce keywords into core meaning, which efficiently reduce time-consuming costs and memory storage.

To deal with encrypted domain for search engines, the inverted index are usually used in symmetric key cryptography systems and cannot be directly deployed in our multiple clients scenario. Aiming to achieve owner-enforced attribute-based authorization towards multiple clients with boolean keyword search, we introduce a new encrypted inverted index utilizing attribute-based access control technique as [20]. Hence, both fine-grained and sub-linear boolean keyword search with inverted index are achieved simultaneously.

### 2.1 Inverted Keyword Search Index

<table>
<thead>
<tr>
<th>Forward Index</th>
<th>Inverted Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>Document id</td>
<td>Keyword</td>
</tr>
<tr>
<td>Doc 1</td>
<td>$w_1$, $w_2$, $w_3$</td>
</tr>
<tr>
<td>Doc 2</td>
<td>$w_2$, $w_3$</td>
</tr>
<tr>
<td>Doc 3</td>
<td>$w_4$</td>
</tr>
<tr>
<td>Doc 4</td>
<td>$w_2$, $w_4$</td>
</tr>
</tbody>
</table>

Fig. 1. Keyword Search Index Data Structure

A keyword dictionary $\delta$ maintains a couple of tuples $(w, c)$, where $w$ is a keyword and $c$ is a counter. This is used to extend a static SE to a dynamic SE with supporting database changes (i.e., documents adding, documents deleting and documents modifying). Generally, the $\delta$ has the following two functions:

- $c \leftarrow \text{Get}(\delta, w)$ : Outputs the counter of a keyword $w$. If $w$ does not exist in $\delta$, outputs 0 as the answer.
- $\text{Update}(\delta, w, c)$ : Updates the counter of a keyword $w$ to $c$. If $w$ does not exist in the dictionary, inserts the tuple $(w, c)$ into it.

### 2.2 Keyword Dictionary

A keyword dictionary $\delta$ maintains a couple of tuples $(w, c)$, where $w$ is a keyword and $c$ is a counter. This is used to extend a static SE to a dynamic SE with supporting database changes (i.e., documents adding, documents deleting and documents modifying). Generally, the $\delta$ has the following two functions:

- $c \leftarrow \text{Get}(\delta, w)$ : Outputs the counter of a keyword $w$. If $w$ does not exist in $\delta$, outputs 0 as the answer.
- $\text{Update}(\delta, w, c)$ : Updates the counter of a keyword $w$ to $c$. If $w$ does not exist in the dictionary, inserts the tuple $(w, c)$ into it.
### 2.3 Cryptographic Assumptions and Primitives

**Definition 1 (PRF [21]).** A pseudo-random function (PRF) $F$ is a polynomial time computable function that cannot be distinguished from random functions $F'$ by any probabilistic polynomial time (PPT) adversary $A$. That is, for any PPT adversary $A$, the advantage is defined as

$$
\text{Adv}_{A,F}(\kappa) = |\Pr[A(F(K,\cdot)(1^\kappa)) - \Pr[A(F'(\cdot)(1^\kappa))],
$$

where $K \sim \{0,1\}^\kappa$. The $F$ is a PRF if $\text{Adv}_{A,F}(\kappa)$ is negligible for any PPT adversary $A$.

**Definition 2 (DDH Assumption).** Let $G$ be a cyclic group with a prime order $p$, the Decisional Diffie-Hellman problem is to distinguish $(g,g^a,g^b,g^{ab})$ from $(g,g^a,g^b,g^r)$, where $g$ is an element randomly selected from $G$ and $a,b,r$ are randomly selected from $Z_p$. For any PPT distinguisher $D$, the advantage is defined as

$$
\text{Adv}_{D,G}(\kappa) = |\Pr[D(g,g^a,g^b,g^{ab}) - \Pr[D(g,g^a,g^b,g^r)]].
$$

The DDH assumption says $\text{Adv}_{D,G}(\kappa)$ is negligible in $\kappa$ for any PPT distinguisher $D$.

**Definition 3 (SXDH Assumption).** Let $G_1$, $G_2$ and $G_T$ be three cyclic groups with a same prime order $p$, and an efficient asymmetric bilinear pairing $e: G_1 \times G_2 \rightarrow G_T$ satisfying: (1) Non-degenerate: if $g_1$ is a generator of $G_1$ and $g_2$ is a generator of $G_2$, then $e(g_1, g_2)$ is a generator of $G_T$. (2) Bilinear: $\forall a, b \in Z_p$, we have $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$. The Symmetric eXternal Diffie-Hellman [22] assumption says that the three groups $G_1, G_2, G_T$ are DDH groups.

**Definition 4 (CP-ABE [23]).** In general, a ciphertext-policy attribute-based encryption consists of four algorithms:

- **ABE.Setup**: The setup algorithm inputs a secure parameter $p$ and an attribute universe, and outputs a public parameter $\text{mpk}_{\text{ABE}}$ and a master key $\text{msk}_{\text{ABE}}$.

- **ABE.Enc**: The encryption algorithm takes as input $\text{mpk}_{\text{ABE}}$, a message $M$ and an access structure $\text{ACC}$, it generates a ciphertext $CT$.

- **ABE.Extract**: The key generation algorithm takes as input $\text{msk}_{\text{ABE}}$ and a set of attributes set $\Gamma$ of a client, outputs a private key $\text{pkv}_{\text{ABE}}$ for the client.

- **ABE.Dec**: The decryption algorithm inputs a $CT$ and a $\text{pkv}_{\text{ABE}}$ associated with an attribute set $\Gamma$, it decrypts the ciphertext if and only if $\Gamma$ satisfies the access policy $\text{ACC}$ that involved in $CT$.

### 3 Problem Formulation

#### 3.1 System Model

There are three different entities in our SE system as shown in Fig. 2: Authority, Cloud Server and Clients. In the system, the authority is a trusted party that maintains system and deals with registration for different parties; the semi-honest cloud server honestly runs algorithms and provides data outsourcing storage/searching services; the (multiple) clients consist of multiple data owners and multiple query clients, where each client can outsource personal documents to cloud and search documents that are contributed from other clients. Once system initialized by the authority, any legal client receives a secret key by registering in the system with submitting his/her attributes set to the authority. Then, a data owner enforces an authorization policy to encrypt his/her documents and outsources the encrypted documents to cloud; to process documents searching, a query client generates a search token for search query and sends it to cloud. Finally, the cloud uses the search token to search over the encrypted database and returns corresponding documents. Note that there is one condition should be satisfied, i.e., not only the document’s keywords set matches search query expression but also the owner-enforced authorization policy accepts the attributes set of a query client. Especially, the system running routine can be seen in Fig. 2.

#### 3.2 Function Definition

- **Setup**: $(1^\kappa, \mathcal{N}) \rightarrow (\text{PP}, \text{MK})$: With a security parameter $\kappa$ and an attribute universe description $\mathcal{N}$, this algorithm outputs a public parameter $\text{PP}$ and a master secret key $\text{MK}$ for the system.

- **KeyGen**(Γ, MK) → sk : The key generation algorithm inputs an attributes set $\Gamma$ of a client and $\text{MK}$, and outputs a secret key $\text{sk}$ for the client.

- **Encrypt**(PP, Doc, sk, ACC) → (EDB, XSet) : The encryption algorithm inputs PP, a document Doc, a sk of a client who encrypts a document and an access policy ACC, and outputs searchable ciphertext.

- **TrapGen**(Q, sk) → Token : The search token generation algorithm accepts a client’s secret key sk along with a search query $Q$ as input, and outputs a search token $\text{Token}$ for the query.

- **Search**(Token) → $R$ : Input a search token $\text{Token}$, the search algorithm outputs encrypted search results $R$.

- **Retrieve**(R, sk) → Documents : This retrieval algorithm inputs an encrypted search results $R$ and a client’s secret key $sk$, and returns target original documents.

#### 3.3 Security Guarantee Model

Different from previous sub-linear boolean keyword search SE schemes [11], [12], [13], security guarantees against two types of honest-but-curious adversaries should be seriously considered: one is an adversary server and one is the colluders. Although the cloud honestly runs the designed protocols, it may snoop privacy information involved in stored documents during searching; and a client may collude with other clients to access/search documents beyond permission. Same as in [11], [12], [13], we assume that there is no collusion between cloud and clients to get privacy information of documents and/or during searching.

#### 3.3.1 Security against adversary server

Recall the security definition in [11] under the simulation-based security [24], where the view of adversary can be simulated by given only permitted leakage during an adaptive attack, in the sense that, the adversary cannot learn anything beyond the permitted leakage. The security is parameterized by a leakage function $\mathcal{L}$ that describes some information being known to the adversary during each operation. Hence, it is possible that the outputs of a simulator have the same distribution with $\mathcal{L}$ as in real-world.
remark that original documents retrieval is not necessarily modeled as that in [11].

**Definition 5.** Let $\Pi$ be a scheme described in Section 3.2, $\mathcal{L}$ be a leakage function, $\mathcal{A}$ be a stateful semi-honest adversary and $\mathcal{S}$ be a simulator, we define the security via the following two experiments $\text{Real}^\Pi_{\mathcal{A}}(\kappa)$ and $\text{Ideal}^\Pi_{\mathcal{A},\mathcal{S}}(\kappa)$.

$\text{Real}^\Pi_{\mathcal{A}}(\kappa)$: $\mathcal{A}(1^\kappa)$ repeatedly chooses an encryption tuple $(\text{Doc}, \mathcal{ACC}, id)^1$ or a searching query tuple $(Q, id)^2$. For an encryption tuple $(\text{Doc}, \mathcal{ACC}, id)$ chosen by the adversary, this game runs $\text{Encrypt}(\text{PP}, \text{Doc}, sk_{id}, \mathcal{ACC})$ to generate $(\text{EDB}, XSet)$ and gives it to $\mathcal{A}$; otherwise, this game runs $\text{Token} \leftarrow \text{TrapGen}(Q, sk_{id})$ and $R \leftarrow \text{Search}(\text{Token})$, then gives the transcript to $\mathcal{A}$. In the end $\mathcal{A}$ returns a bit as an output of this game.

$\text{Ideal}^\Pi_{\mathcal{A},\mathcal{S}}(\kappa)$: This game initializes two empty lists $d$ and $q$, and initializes two counters $i = 1$ and $j = 1$. $\mathcal{A}(1^\kappa)$ repeatedly chooses an encryption tuple $(\text{Doc}, \mathcal{ACC}, id)$ or a searching query tuple $(Q, id)$. If it chooses an encryption tuple, $\mathcal{A}$ records $(\text{Doc}, \mathcal{ACC}, id)$ as $d[i]$ and increases $i$, this game runs $S(\mathcal{L}(d, q))$ to get $(\text{EDB}, XSet)$, and gives it to $\mathcal{A}$; Otherwise, this game records $(Q, id)$ as $q[i]$, increases $i$ and runs $S(\mathcal{L}(d, q))$ to output a transcript to $\mathcal{A}$. In the end $\mathcal{A}$ returns a bit as an output of this game.

We say the II is $\mathcal{L}$-semantically secure against an adaptive adversary if there exists an algorithm/simulator $\mathcal{S}$ such that

$$\Pr[\text{Real}^\Pi_{\mathcal{A}}(\kappa) = 1] - \Pr[\text{Ideal}^\Pi_{\mathcal{A},\mathcal{S}}(\kappa) = 1] \leq \text{negl}(\kappa)$$

where $\text{negl}(\cdot)$ is a negligible function under a security parameter $\kappa$. This security guarantee lets the adversary be unable to learn any knowledge beyond the permitted leakage information, otherwise, it successfully distinguishes two games with a non-negligible advantage.

### 3.3.2 Security against colluded clients

In the system, the clients may collude each other to search/access the documents beyond permission, thus a valid search token is successfully computed for making the merging attribute set satisfies the aiming challenged authorization policy. Therefore, the security against collusion attacks should be considered, where the game is sketched by the following.

- **Init.** The challenger runs Setup to initialize the game and returns public parameter to an adversary $\mathcal{A}$.
- **Key extraction.** Receiving a secret key query for an attribute set $\Gamma$ from $\mathcal{A}$, the challenger runs KeyGen to return a secret key associated with $\Gamma$ to $\mathcal{A}$.
- **Output.** $\mathcal{A}$ outputs a search token where the involved attribute set has not been queried before. If the search token is valid, the challenger outputs 1.

We say a scheme II is secure against colluded clients if the challenger outputs 1 in the above game with a negligible probability. This security guarantee lets colluded clients be unable to search/access documents beyond permission according to the owner-enforced authorization policy.

### 3.4 Design Goals

The design goals of our system are formalized as follows:

1. **Multiple clients (including data owners and query clients).** To accommodate multiple clients, each client could search outsourced documents that contributed from multiple data owners.
2. **Fine-grained authorization towards document searching/accessing.** The data owner can non-interactively determine which clients search over his/her documents based on owner-enforced attribute-based authorization policy across multiple clients who are described by a set of attributes.
3. **Boolean keyword search expression.** In the system, a query client is able to issue practical boolean queries that have been intensively deployed in searching engines.
4. **Sub-linear Search Costs.** The search costs are independent of the total number of all stored documents, and only related to the least frequent term in the conjunction.
5. **Running securely.** During the process of outsourced encrypted documents sharing and searching query, no other knowledge are leaked to the cloud/outsiders.
attribute and assume the employed two functions $W$.

### 4.2 Formal Construction

To encrypt a document choose to use a symmetric key encryption algorithm (e.g., $\delta, w, c$).

$$\text{Update}(\delta, w, c) = (\prod_{\text{att} \in I} x_{\text{att}}^{x_{\text{ind}}}) \cdot e((\prod_{\text{att} \in I} t_{\text{att}}^{r_{\text{att}}}) H(w_{\text{att}}), g_{2}^{x_{\text{ind}}})$$

$$= e(u_{\text{att}}^{H(w_{\text{att}})}, z_{e}) \cdot e((\prod_{\text{att} \in I} \sigma_{\text{att}})^{H(w_{\text{att}})}, g_{2}^{z_{e} \cdot x_{\text{ind}}})$$

$$= e(u_{\text{att}}^{H(w_{\text{att}})}, g_{2}^{z_{e} \cdot x_{\text{ind}}}) \cdot e((\prod_{\text{att} \in I} t_{\text{att}}^{r_{\text{att}}}) H(w_{\text{att}}), g_{2}^{x_{\text{ind}}})$$

Correctness Guarantee. Provided the owner-enforced authorization policy $\text{ACC}$ can be satisfied, the correctness guarantee for document searching $\text{Search}$ holds as follows:

$$e(u_{\text{att}}^{H(w_{\text{att}})}, z_{e}) \cdot e((\prod_{\text{att} \in I} \sigma_{\text{att}})^{H(w_{\text{att}})}, g_{2}^{z_{e} \cdot x_{\text{ind}}})$$

The security analysis of our main construction for conjunctive keyword search in Section 4 is studied.

### 5 Security Analysis

#### 5.1 Leakage Analysis

We begin with analyzing the leakage information considering a trade-off between security and efficiency. As illustrated in Section 3.3, two lists $d$ and $q$ are employed to store respective encryption and searching tuples. Here, let a $f$-th query be $q[f] = (s[f], x[f], \cdot, \cdot)$, where $s[f]$ is the $s$-term, $x[f], \cdot$ is the terms and $id[f]$ is the identity of the query client. The concrete descriptions on leakage function $L$ and simulation processes are given in Section 9: Appendix with Algorithm 1. Additionally, we emphasize that the leakage formation in $id, RP, SRP$ and IP are overstated for designing security proof, and hence not be revealed in actual scheme.

#### 5.2 Security Analysis

Similar to [11, 13], we first prove the security under a non-adaptive attack, which means the adversary will submit the completed lists $d$ and $q$ together. Then, we discuss the security against an adaptive attack.

**Theorem 1.** Our scheme is $L$-semantically secure against non-adaptive attacks, if the employed PRFs are secure, the underlying CP-ABE is IND-sCP-CCA secure and the SXDH assumption holds in $G_{1}$, $G_{2}$ and $G_{\mathcal{T}}$.

**Proof.** By using the outputs of the leakage function $L$, we construct a simulator who has a same distribution as the real game, which is shown in Section 9: Appendix.

From the construction of simulator, we have

$$\text{Pr}[\text{Real}_{A}^{\mathcal{L}}(\kappa)] = 1 - \text{Pr}[\text{Ideal}_{A,S}^{\mathcal{L}}(\kappa)]$$

$$\leq \text{Adv}_{\mathcal{A},G_{1}}^{\mathcal{L}}(\kappa) + \text{Adv}_{\mathcal{A},G_{2}}^{\mathcal{L}}(\kappa) + \text{Adv}_{\mathcal{A},G_{3}}^{\mathcal{L}}(\kappa) + \text{Adv}_{\mathcal{A},F_{p}}^{\mathcal{L}}(\kappa)$$

$$+ \text{Adv}_{\mathcal{A},\text{Prp}}^{\mathcal{L},\text{CP-CPA}}(\kappa).$$

**Theorem 2.** Our scheme is $L$-semantically secure against adaptive attacks, if the employed PRFs are secure, the underlying CP-ABE is IND-sCP-CCA secure and the SXDH assumption holds in $G_{1}$, $G_{2}$ and $G_{\mathcal{T}}$.

**Proof.** The adaptive attack indicates that the adversary performs attacks as the security definition described in
Setup($\kappa, \mathcal{N}$) : Inputs a secure parameter $\kappa$ and an attribute universe $\mathcal{N} = \{\text{att}_1, \ldots, \text{att}_n\}$ whose size is $n$, the authority randomly selects $r_1, r_2, \ldots, r_{2n} \in \mathbb{Z}_p$ and $t_1, t_2, \ldots, t_{2n} \in \mathbb{G}_1$. For $k = [2n]$ sets $x_k = g_2^{r_k}$ and $y_k = e(t_k, g_2)$. Let $H : \{0, 1\}^* \rightarrow \mathbb{Z}_p$ be a collision-resistant hash function, $F$ be a PRF with range in $\{0, 1\}^*$ and $F_p$ be a PRF with range in $\mathbb{Z}_p$, and ABE be a non-monotonic ABE scheme. The authority runs setup algorithm of ABE to generate key pair

$$(\text{mpk}_{\text{ABE}}, \text{msk}_{\text{ABE}}) \leftarrow \text{ABE.Setup}(\kappa, \mathcal{N})$$

and randomly selects two keys $K_x$ and $K_z$ for a PRF $F_p$, and selects a key $K_i$ for a PRF $F$. The public parameter is

$$\text{PP} = \{g_1, g_2, e, H, F, F_p, \text{mpk}_{\text{ABE}}, \{(x_k, y_k)\}_{k=[2n]}\},$$

and the master secret key is

$$\text{MK} = \{\text{msk}_{\text{ABE}}, K_x, K_z, K_i, \{(r_k, t_k)\}_{k=[2n]}\}.$$ Note that, for $k \in \{1, \ldots, n\}$, the public parameter information $(x_k, y_k)$ in PP corresponds to the positive type of attribute $\text{att}_k$ and $(x_{k+n}, y_{k+n})$ corresponds to the negative type of it; the master secret key information $(r_k, t_k)$ in MK corresponds to the positive type of attribute $\text{att}_k$ and $(x_{k+n}, y_{k+n})$ corresponds to the negative type of it.

KeyGen($\Gamma$, MK) : Inputs a client’s attributes set $\Gamma \subseteq \mathcal{N}$ (a non-empty subset of $\mathcal{N}$) and the master secret key MK. The authority randomly selects an element $v$ from $\mathbb{G}_1$, for each $i \in [n]$: computes $\sigma_i = t_i v^{r_i}$ if $a \in \Gamma$ (i.e., one attribute $\text{att}_i$ in $\mathcal{N}$ exists in a client’s attribute set $\Gamma$); else sets $\sigma_i = t_i v^{r_i+n}$ if $a \notin \Gamma$. Outputs the client’s secret key sk as

$$\text{sk} = \{K_x, K_z, K_i, \text{pvk}_{\text{ABE}}, v, \{\sigma_i\}_{i\in[n]}\}$$

where $\text{pvk}_{\text{ABE}} \leftarrow \text{ABE.Extract}(\text{msk}_{\text{ABE}}, \Gamma)$.

Encrypt($\text{PP}, \text{Doc}, \text{sk}, \text{ACC}$) : Inputs public parameter PP, a client’s secret key $\text{sk} = \{K_x, K_z, K_i, \text{pvk}_{\text{ABE}}, v, \{\sigma_i\}_{i\in[n]}\}$, a document $\text{Doc} = (\text{ind}, \text{W}_{\text{ind}})$ and an access policy $\text{ACC} = \bigwedge_{\text{att} \in \Gamma} \{\text{att} \in \{\text{att}_1, \ldots, \text{att}_n\} \text{ (an “AND” operation over some attributes att}_i \text{ in a non-empty subset I of } \mathcal{N}\}$. Computes $\text{Xind} \leftarrow F_p(K_x, \text{ind})$ and $(x_i, y_i)$ as

$$(x_i, y_i) = \begin{cases} (x_i, y_i) = (g_2^{r_i}, e(t_i, g_2)) & \text{if } \text{att}_i = \text{att}_i^+ \\ (x_{i+n}, y_{i+n}) = (g_2^{r_{i+n}}, e(t_i, g_2)) & \text{if } \text{att}_i = \text{att}_i^- \end{cases}$$

And sets $(x_I, y_I) = (\prod_{\text{att}_i \in \Gamma} x_i, \prod_{\text{att}_i \in \Gamma} y_i)$. For each keyword $w \in \text{W}_{\text{ind}}$, it does the following:

1) Computes an internal counter $c \leftarrow \text{Get}(\delta, w)$, $c \leftarrow c+1, l \leftarrow F(K_i, c||w), z \leftarrow F_p(K_z, c||w)$ and runs Update($\delta, w, c$) to updates the counter of each keyword to $c$.

2) Computes encrypted tuples $e_0 \leftarrow \text{ABE.Enc}(\text{mpk}_{\text{ABE}}, \text{ind}||\text{ACC}, \text{sk}), e_1 \leftarrow g_2^{x_i \cdot \text{ind}}, e_2 \leftarrow x_i^{y_i \cdot \text{ind}}$ then sets the item in $\text{EDB}[l] = (\text{ACC}, e_0, e_1, e_2)$ and appends a value $\text{stag} = y_I^{f(H(w)) \cdot \text{ind}}$ to s set data structure $\text{XSet}$. The cloud outsources (EDB, XSet) and the encrypted original document by using a symmetric key algorithm (e.g., AES) with a secret key is $\text{K}_{\text{ind}}$ to the cloud.

TrapGen($Q, \text{sk}$) : Given a conjunctive keyword searching query $Q = (w_1 \land w_2 \land \cdots \land w_q)$ and $w_1$ is assumed to be s-term (the least frequent term among the queried terms/keywords in Q), and a client’s secret key $\text{sk} = \{K_x, K_z, K_i, \text{pvk}_{\text{ABE}}, v, \{\sigma_i\}_{i\in[n]}\}$. The clients does the following steps until the cloud sends stop symbol Failure, for an internal counter $c = 1, 2, \ldots$ in keyword dictionary,

1) Computes $l_c \leftarrow F(K_i, c||w_1), \text{z_c} \leftarrow F(K_z, c||w_1), \text{Trap}[c][j] = (v^{H(w_j)||\text{z_c}}, \{\sigma_i^{H(w_j)||\text{z_c}}\}_{i\in[n]})$, for $j = 1, \ldots, q$.

2) Sends generated tokens $\text{Token}[c] = (\{l_c, \text{Trap}[c])$ where $\text{Trap}[c] = \{\text{Trap}[c][j]\}_{j\in[q]}$ to the cloud for $c = 1, 2, \ldots$

Search($\text{Token}$) : Receiving a search token $\text{Token}$ from a client, the cloud proceeds as follows:

1) Initializes an empty set $R$ denoted as searching result, for an internal counter $c = 1, 2, \ldots$ do
   a) Retrieves tuples $(\text{ACC} = \bigwedge_{\text{att}_i \in \Gamma} \text{att}_i, e_0, e_1, e_2) \leftarrow \text{EDB}[l_c]$, if this action fails, jump to “Step 2”.
   b) Checks if $e(v^{H(w_j)||\text{z_c}}) = e(\sigma_j, e_1) \in \text{XSet}$ for all $j \in [q]$, where $v^{H(w_j)||\text{z_c}}$ and $\sigma_j = (\prod_{\text{att}_i \in \Gamma} \sigma_i)^{H(w_j)||\text{z_c}}$ comes from $\text{Trap}[c][j] = (v^{H(w_j)||\text{z_c}}, \{\sigma_i^{H(w_j)||\text{z_c}}\}_{i\in[n]})$. If it exists in XSet for all $j \in [q]$ then add $e_0$ to the set $R$.
2) Sends the stop symbol Failure and returns $R$ to clients as the searching result, end off this search process.

Retrieve($\text{R}$) : To retrieve the original documents, the client proceeds as follows:

1) Decrypts each $e_0$ in the returned searching result $R$ as $(\text{ind}||\text{K}_{\text{ind}}) \leftarrow \text{ABE.Dec}(\text{pvk}_{\text{ABE}}, e_0)$.
2) Sends $\text{inds}$ in “Step 1)” to the cloud to fetching the encrypted original documents.
3) Decrypts the encrypted original documents with a corresponding secret key $\text{K}_{\text{ind}}$, respectively.
Section 3.3, instead of submitting two completed lists \(d\) and \(q\) in together. Generally, the security follows the techniques of [11], where the simulator pads EDB and XSet with random chosen elements to deal with adaptive attacks. To simulate search tokens, it adaptively assigns the random chosen elements by hashing into tables. The main idea is to write EDB and XSet in a random way, and then adaptively initializes the hash tables.

**Theorem 3.** Our scheme is secure to resist the attacks carried out by the colluded clients, if the underlying CP-ABE is IND-scp-CPA secure and the SXDH assumption holds in \(G_1, G_2\) and \(G_T\).

**Proof 3.** The proof of this theorem is trivial, we can see that no colluded clients can generate a valid token beyond permission with the guarantee provided by the SXDH assumption. For the colluded clients, successfully generating a valid token implies that they can generate a new secret key beyond permission. In such a sense, they have to combine secret keys in together to solve for \(t_1\) and \(v_1\), but this is impossible since each secret key of a client is related to a random value \(\tilde{v}\) and the DDH assumption holds in \(G_1\) as well. Therefore, the colluded clients cannot generate a new secret key and a valid search token beyond their permissions.

### 6 Enhanced Construction: Supporting Boolean Keyword Search

This section extends the main construction supporting conjunctive queries in Section 4 to support boolean queries.

**Boolean queries.** The boolean query is more expressive for documents searching since it includes negations, disjunctions, threshold queries and more. Concretely, a boolean query formulated as \(w_1 \land \psi(w_2, \ldots, w_q)\) enables us to search a document that contains a keyword \(w_1\) and additionally satisfies an arbitrary boolean expression \(\psi\) on the remaining keywords \((w_2, \ldots, w_q)\). With employing boolean queries, the search complexity is just proportional to the number of documents that contain the keyword \(w_1\) rather than the total number of all stored documents in remote server. Without loss of generality, the keyword \(w_1\) is assumed to be the estimated least frequent keyword.

**Technical implementations.** The used techniques is similar as [11] for extending conjunctive queries \(w_1 \land w_2 \land \cdots \land w_q\) to boolean queries \(w_1 \land \psi(w_2, \ldots, w_q)\), by overkeywords \((w_1, w_2, \ldots, w_q)\). A client computes \(l_1\) and \(\text{Trap}[\epsilon]\) as same as in processing a conjunctive search but sends them with a boolean expression \(\psi\) to the cloud, where \(\psi\) is a copy of \(\psi\) except that the keywords are replaced by \((v_2, \ldots, v_q)\). The cloud uses \(l_1\) to retrieve tuples \((\text{ACC}, c_0, e_1, e_2)\) that associated with an estimated least frequent keyword \(w_1\), where the difference with conjunctive queries is the way to determine which tuples match \(\psi\). For each record \((\text{ACC}, c_0, e_1, e_2) \leftarrow \text{EDB}[l_1]\), the cloud computes \((v_2, \ldots, v_q)\) as

\[
v_j = \begin{cases} 
1 & \text{if } e(H(w_j)/\epsilon, v_j \cdot e(\sigma_j, e_1)) \in \text{XSet} \\
0 & \text{otherwise}
\end{cases}
\]

where \(j = 2, \ldots, q\). If \(e(H(w_j)/\epsilon, v_j \cdot e(\sigma_1, e_1)) \in \text{XSet}\) and \(\psi\) holds, this implies that the tuple matches the query, then the cloud appends \(e_0\) to the result set \(R\).

Note that the search complexity for such boolean query is \(O(c_{w_1})\) where \(w_1\) is the s-term of the query. And the leakage profiling process is consistent with processing the leakage incurred in a conjunctive keyword searching over \((w_1, w_2, \ldots, w_q)\). Hence, the leakage information is similar as conjunctive queries except for the cloud server gets \(\psi\).

### 7 Performance Evaluation

This section gives a theoretical analysis comparison with boolean query SE work (not support owner-enforced authorization) and an implementation analysis comparison with representative ABSE work (only support conjunctive query).

#### 7.1 Theoretical Analysis

**Table 3** Numerical Evaluation. The \(\text{Mul}_1, \text{Mul}_2\) and \(\text{Mul}_\tau\) respectively denotes the multiplication operation in group \(G_1, G_2\) and \(G_T\), \(E_1, E_2\) denotes the exponentiation operation in group \(G_1, G_2\) respectively and \(BP\) is the bilinear pairing operation; \(\text{Time}_{\text{ABE Setup}}\) and \(\text{Time}_{\text{ABE Extract}}\) represent the running time of the Setup and Extract algorithm in ABE.

<table>
<thead>
<tr>
<th>Running Process</th>
<th>Computation complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Setup</td>
<td>(n(E_2 + BP) + \text{Time}_{\text{ABE Setup}})</td>
</tr>
<tr>
<td>Key Generation</td>
<td>(n(E_1 + \text{Mul}<em>1) + \text{Time}</em>{\text{ABE Extract}})</td>
</tr>
<tr>
<td>Encryption</td>
<td>(</td>
</tr>
<tr>
<td>Token Generation</td>
<td>(q\text{Mul}_1 + q</td>
</tr>
</tbody>
</table>

Table 3 shows a theoretical computation analysis on group operations of the proposed boolean query SE. We assume the size of attribute universe in the system is \(n\) and a document \((\text{ind}, W_{ind})\) being in encryption under an access policy \(\text{ACC}\), and consider the size of \(W_{ind}\) as \(|W_{ind}|\) and the number of attributes in an “AND” gate access policy \(\text{ACC}\) as \(|\text{ACC}|\). For processing a boolean query \(w_1 \land \psi(w_2, \ldots, w_q)\), we assume \(w_j\) as its s-term and \(c_{w_1}\) as a counter across \(q\) keywords, and \(|\text{ACC}|\) denotes the sum of the number of attributes in all access control policies that related to theedocuments in DB[\(w_1\)].

Moreover, we give a feature and efficiency comparison with boolean SE work [11, 12, 13] in Table 4. The efficiency side mainly focuses on generating a search token between a data writer/owner and a data reader/client, where [11], [12, 13] cannot support owner-enforced authorization and only support conjunctive queries \((w_1 \land w_2 \land \cdots \land w_q)\). As the expected attribute-based search control is achieved across multiple clients for data sharing, we need to introduce public key encryption techniques. This may bring about additional costs in reader’s computation side than those depend on simple symmetric-key operations.

#### 7.2 Conducted Experiment Analysis

We implement both the enhanced SE system supporting boolean query \(w_1 \land \psi(w_2, \ldots, w_q)\) where \(\psi\) is an arbitrary boolean expression and the “attribute-based keyword search with fine-grained owner-enforced search authorization” work [7] that supports conjunctive keyword search \(w_1 \land w_2 \land \cdots \land w_q\), then give an all-around implementation analysis for
comparison. Here, we take the work [7] as a representative ABSE since it inspires brand works in the area of ABSE and achieves better search efficiency among related work. It additionally mainly focuses on fundamental multi-client keyword search with owner-enforced authorization in cloud rather than concerning more security concerns. Moreover only [7] employs the same “AND”-gate search authorization “att1 ∧ att2 ∧ ···” towards multiple clients as this paper.

7.2.1 Experimental bed-up

The work [7] and our SE system are conducted on an Ubuntu 16.04 system with an Intel(R) Core(TM) i3-4130 CPU of 3.40GHz and 4.00GB RAM. The codes are implemented with python 3 language under Charm 0.43 library [25] (a widely deployed open-source framework for rapidly prototyping advanced cryptosystems). To optimize the running efficiency of conducted schemes in implementation side, we choose to use an asymmetric elliptic curve “MNT159” to implement them and employ the scheme [26] as the underlying ABE scheme in our SE system.

A well-known representative real-world dataset Enron [19] is taken as a testing dataset for further illustrating convincing practicality, where the MySQL database is used to store encrypted data. We randomly select 1000 different documents from Enron dataset to encrypt under randomly generated access policies ACC. And assume the scale of attribute universe ranges from 10 to 100 (n : 1 ~ 100), the number of the associated keywords |W_{ind}| with each document ranges from 1 to 50 (#keywords: 1 ~ 50), and the assumed least frequent term in the conjunction ranges 20 to 60 (∼term: 20 ~ 60). As other works, the experiment does not consider the original documents encryption process.

To well study and understand the experimental results about [7] and this work, we mainly focus the searching efficiency that includes trapdoor generation phase along with searching phase and documents encryption phase, as well as the system setup phase and key generation phase.

7.2.2 Results and Comparison

7.2.2.1 System Initialization: The system initialization of both [7]’s conjunctive query SE system and our boolean query SE system respectively includes the Setup and KeyGen algorithm. The setup phase initializes the system by generating public parameters and master secret key, while the key generation phase grants corresponding secret keys for clients. Overall, the system initialization performance comparison are drawn in Fig. 4, whose running time costs are both related to the scale of attribute universe.

We may conclude that, though the performance of this work is efficient but still a little expensive than [7] due to invoking the setup and key extraction algorithm of an ABE modular. Nevertheless, this is highly acceptable since the expressive boolean query pattern (i.e., “w1 ∧ ψ(w2, ··· , wq)”) are achieved, while previous multi-client SEs with owner-enforced authorization only support conjunctive keyword query. Moreover, once an one-time investment for initializing the system completed, the provided highly-scalable documents searching/sharing service are more efficient than conjunctive query SEs, which is studied in the following.

7.2.2.2 Documents Tuples Encryption: To evaluate the document encryption performance towards randomly chosen 1000 different documents, which are indexed with 1 ~ 50 keywords and encrypted under randomly specified authorization policies across 10 ~ 100 attributes. Concretely, from the time-costs distribution in Fig. 5(a) and the storage-costs distribution in Fig. 5(b) that related to the number of attributes, the general efficiency performance in this work is almost same with different #attributes. Nevertheless, the time costs for generating encrypted documents in [7] increases with #attributes and is expensive than ours. Fig. 6(a) and Fig. 6(b) show the time-costs and storage-costs distribution related to the number of keywords involved in documents, in which the consumed costs in [7] and ours are both linear to #keywords. As can be seen, our system yields high time and storage efficiency than [7]. Specifically, the time costs in this work are about 2.5 ~ 3.5 seconds even the #keywords reaches 50 across different #attributes, while that in [7] are about 3.5 ~ 6 seconds. A similar result can be observed for the storage costs side, only 23KB ~ 26KB memory space is needed to maintain (EDB, Xset) tuple for a document who contains 50 keywords across different #attributes, while that in [7] are nearly about 100KB ~ 550KB.

Hence, we can conclude from Fig. 5 and Fig. 6, the primary factor that influences the Encrypt algorithm in this work is the number of associated keywords (#keywords), while that in [7] is determined by both the number of keywords and the number of attributes.

7.2.2.3 Documents Searching: During documents searching, a client runs TrapGen algorithm to wake up
the cloud to process Search algorithm, hence the documents searching overhead involve both time-costs and communication-costs of the TrapGen and Search algorithm. Generally, the documents searching performance in our boolean query SE and [7]'s conjunctive SE are both mainly related to #keywords and #attributes, while the least frequent term in the conjunction (i.e., s-term) is still an important factor for this work.

Concretely, by setting #keywords=5 and increasing the value of #attribute from 1 to 100, we can observe from Fig. 7 that: (1) the counter value of s-term (#s-term) included in a query in our scheme is still a very important factor that influence the searching performance in Fig. 7(a); (2) our work enjoys better time efficiency than [7] for a same #keywords (resp. runs 4500 times faster with s-term=20 and 1500 times faster with s-term=60 when #attributes=100) in Fig. 7(b); (3) [7] enjoys better communication efficiency than this work due to no consideration of expected expressive boolean queries in Fig. 7(c). [7] achieves roughly respective 10 times and 125 time savings than ours s-term=20 and s-term=60 realizing expressive boolean query patterns, but it is considered to be acceptable in real-world.

To measure the influence of the #attribute and #keywords for documents searching performance, we assume the s-term as 40 in our system and provide a detailed result comparison on time costs between [7] and ours in Table 5. As can be seen, our boolean query SE certainly achieve high time efficiency and slightly increase with #attributes and #keywords, and moreover achieve roughly 300~2000 times savings than [7]. We remark that if the s-term is set as 20, more time-costs savings are certainly obtained according to the conclusion from Fig. 7. This is because, processing a single/conjunctive-keyword search [5], [6], [7], [8], [9], [10], a client needs to send a keyword to cloud for fetching the documents where this keyword occurs, then finds and downloads the final results by itself. However, this work employs inverted index as the underlying data structure to manage encrypted documents for speeding up performance.

Fig. 5. The time costs and storage costs of documents encryption phase related to the number of Attributes.

Fig. 6. The time costs and storage costs of documents encryption of related to the number of Keywords.

Fig. 7. The time and storage costs of documents searching with setting #keywords as 5.

Fig. 8. The time and storage costs of document searching with setting s-term as 40 in our boolean query SE system.
To observe how the #keywords influences the documents searching efficiency for our system, whose searching costs slightly increase with the growth of the amount of attribute in Fig. 8, where the s-term is set 40. Hence, we can conclude that the documents searching costs are related to the least frequent term (s-term) in the conjunction and independent of the total number of stored documents in cloud. This greatly brings about high efficiency for documents searching particularly for highly-scalable documents.

### 7.3 Related Work

**Encrypted Boolean Query.** Boolean query is an indispensable kind of keyword search and has been widely deployed in applications. With a boolean keyword search expression (i.e. $w_1 \land \lnot w_2 \lor w_3$), document searching is to retrieve documents whose keywords set matches the boolean expression. Considering keywords as a vector, Moataz et al. [27] designed a SE scheme that supported generic boolean queries over encrypted data, but consumed-time costs were linear to the total number of documents in database. Later on, the milestone work by Cash et al. [11] introduced the first SE scheme with the support of sub-linear boolean keyword search, and then widely researched in [12], [13], [14].

**Attribute-Based Searchable Encryption.** With employing proxy re-encryption technique, Liang et al. [6] designed an ABSE scheme while still preserved encrypted data functionality of a proxy. A number of ABSE works focused on preserving the privacy of the access policy [10], [28], outsourced ABSE works with keyword search and enhanced access control [8], [29], and as well as a trade-off between appreciated functionalities [9], [30], [31] and adversarial attack models [17], [18], [32], [33]. Nevertheless, all existing ABSE schemes can only work with single keyword search or conjunctive-keyword search, but lose highly-efficient search costs (i.e., $O(\#doc)$, where $\#doc$ is the total number of the stored documents in cloud).

### 8 Conclusion

This work presents a privacy-preserving documents searching/sharing system for encrypted cloud storage. In the system, a data owner specifies an authorization on which clients could search/access outsourced data, and the cloud can process a client’s boolean keyword search with only sub-linear costs. Note that the system does not consider attribute dynamic operations (e.g., attribute adding) at present. Since the size of the attribute universe is required to be fixed as $n$ in the setup algorithm, where the associated random variables should be also assigned to corresponding attributes. As a result, we cannot let the attribute universe vary with the attribute dynamic changes. It seems not a easy job to realize dynamic attribute operations for the system, since once the attribute universe is not fixed when initializes the system, the CNF expression over both keywords and attributes could not be achieved. We leave this as an interesting future work to study.

### 9 Appendix

Remarks on leakage function and formal simulation process on non-adaptive attacks are described by Algorithm 1.

### References


Algorithm 1 Simulator

Remarks on Leakage Function \( \mathcal{L} \):

With taking as input \( d \) and \( q \), the leakage function \( \mathcal{L} \) outputs the following items:

- \( op \) is an array that records the type of each operation type. We let \( op[f] \) denote the type of \( f \)-th operation and it is either ”encrypt” or ”search”.
- \( ACC \) is an array that records the access policy in each encryption operation. Let \( ACC_{ind} \) be an access policy of the document labeled with \( ind \).
- \( id \) is an array records the identity of client in each search operation. Moreover, we assume the attributes set of a client \( id \) is also leaked.
- \( N \) is an array that records the size of each EDB and XSet, namely, the number of keywords in each document, and \( |N| = \) equals with \( |d| \).
- \( t[a,b] \) is the equality pattern of \( a \), it reveals whether the queries have the same \( t \)-term. If \( s = \{a, b, a, b\} \), then we have \( t[a,b] = \{1, 2, 1, 3, 2\} \).
- \( EDB \) is the equality pattern of \( x \), it reveals whether the queries have the same \( EDB \).
- \( SP \) is the size of each query pattern, it reveals the numbers of indices that matches the \( s \)-term in each query, i.e., \( SP[f] = |DB[s[f]]| \).
- \( RP \) reveals the indices that the access policy can be satisfied by a client \( id[\] in the intersection of \( DB[s[f]] \) and \( DB[x_[f]], \) i.e., \( RP[f, \alpha, d] = DB[s[f], id[\_\_f]] \) \& \( DB[x_[f]], \) Denote \( RP[f, \alpha, d] \) as the element in \( RP[f, \alpha] \) that produced by \( d[d] \).
- \( SRP \) reveals the matching results of the \( s \)-term of \( f \)-th query, i.e, \( SRP[f] = DB[s[f]] \).
- \( dRP[f][d'] \) \( = 1 \) implies the indice of \( d'[f] \) is contained in \( DB[h][d] \), otherwise, the value is 0.
- \( IP[f, j, \alpha, \beta] = \begin{cases} \db, & \text{if id}_d[f] = id[j] \text{ and } \exists x[f, \alpha] = x[j, \beta] \\ DB[s[f], id[\_\_f]], & \text{if id}_d[f] \neq id[j] \text{ and } \exists x[f, \alpha] = x[j, \beta] \\ \emptyset, & \text{Otherwise.} \end{cases} \)
- \( -term \) records the number of terms \( x \) in \( i-th \) query.

Simulation Process:

function Initialize(\( \mathcal{L}(d, q) \))

for each \( h \in s[q] \) do
    \( c_{op}=0 \)
end for

for each \( w \in x', \) \( ind \in RP \uplus IP \) do
    \( H_2[w, \_\_ind] = e(g_1, g_2) H_2[w, \_\_ind] \)
end for

for each \( w \in x', \) \( ind \in RP \uplus IP, id \in id \) do
    if \( id \in (ACC_{ind} = \bigwedge_{i \in \text{att.}} f[t, \alpha]) \) then
        \( \{y_0, \ldots, y_n\} \leftarrow Z_{p+1} ^{\oplus t} \)
        \( H_2[w, \_\_ind] = y_0 + \sum_{i \in \text{att.}} f[t, \alpha], f[t, \alpha] \)
    else
        \( \{y_0, \ldots, y_n\} \leftarrow Z_{p+1} ^{\oplus t} \)
        \( H_2[w, \_\_ind] = y_0 + \sum_{i \in \text{att.}} f[t, \alpha], f[t, \alpha] \)
    end if
end for

for \( d = q = 1 \) do
    for \( h = 1 \) to \( |op| \) do
        if \( op[h] == \text{encrypt} \) then
            \( t[h, \_] = \text{Encrypt} (\mathcal{L}(d, q)) \)
        else
            \( t[h, \_] = \text{Encrypt} (\mathcal{L}(d, q)) \)
        end if
    end for
end for

end function

function Encrypt(\( \mathcal{L}(d, q) \))

h = 0, \( \text{Dupp} \leftarrow \{ \}

for \( s[q'] \in (s[q] \cdots s[q')) \notin \text{Dupp} \) and \( dRP[d][q'] == 1 \) do
    \( c_{\text{w}[q']} \leftarrow c_{\text{w}[q']} + 1 \)
    \( l \leftarrow 1, 0 \)
    \( l[w[q']], l[w[q']] \leftarrow l \)
    \( e_0 \leftarrow ABE, \text{Enc}(mpk_{\text{ABE}}, 0^k, \text{ACC}[d]) \)
    \( y \leftarrow Z_{p}, H_1[s[q'], c_{\text{w}[q']} = y \)
    \( y \leftarrow Z_{p}, H_1[s[q'], c_{\text{w}[q']} = y \)
    \( e_1 \leftarrow g^s \cdot H_1[s[q'], c_{\text{w}[q']} \times e_2 \leftarrow g_2 \cdot H[s[q'], c_{\text{w}[q']} \times \text{E} \leftarrow \text{ACC}[d], e_0, e_1, e_2 \)
    \( \text{Dupp} \leftarrow \text{Dupp} \cup s[q'], h += 1 \)
end for

end function

function Search(\( \mathcal{L}(d, q) \))

l \leftarrow \{0, 1\}
\( e_0 \leftarrow ABE, \text{Enc}(mpk_{\text{ABE}}, 0^k, \text{ACC}[d]) \)
\( e_1 \leftarrow g_2, e_2 \leftarrow g_2 \)
\( \text{E} \leftarrow \text{ACC}[d], e_0, e_1, e_2 \)

end function

function XSetSetup(\( \mathcal{L}(d, q) \))

XSet \leftarrow \{ \}
\( \text{return} \text{E} \text{\&} \text{X} \)

end function

function TranGen(\( \mathcal{L}(d, q) \))

end for

function XSet(\( \mathcal{L}(d, q) \))

XSet \leftarrow \{ \}
\( \text{xtag} = ++ \)
\( \text{XSet} \leftarrow \text{XSet} \cup \text{xtag}, h += 1 \)
end for

end function

end for

for \( h = 1 \) to \( N[d] \) do
    \( \text{xtag} = ++ \)
    \( \text{XSet} \leftarrow \text{XSet} \cup \text{xtag} \)
    \( \text{end for} \)

end function

for \( c \in [e_{\text{w}[q]}] \) do
    if \( \text{ind}_c \in K \) then
        \( \{y_0, \ldots, y_n\} \leftarrow H_2[x[q], \alpha, \text{ind}_c, e_{\text{w}[q]}] \)
        \( \text{Tran}[c, \alpha] = \{g_1^{y_0 \times g_2}, \ldots, g_1^{y_n \times g_2} | k \in [n]\} \)
        \( \text{else} \)
        \( \exists H_{0}[s[q]], x[q], c, e_{\text{w}[q]} \) \text{ then}
        \( \text{Tran}[c, \alpha] = H_0(s[q], x[q], c, e_{\text{w}[q]}] \)
        \( \text{else} \)
        \( \text{Tran}[c, \alpha] = H_0(s[q], x[q], c, e_{\text{w}[q]}] \)
        \( \text{end if} \)
    end if
end for

end function

for \( \text{Token} \leftarrow \{1, \text{Tran}\} \)

end for

\( \text{Res} \leftarrow \text{Search}(\text{Token}) \)
\( \text{Resl} \leftarrow \cap \text{RP}[q, \alpha] \) \text{ for } \alpha \in [\text{xt}[q]]
\( \text{return} \text{(Token, Res, Resl)} \)

end function