Dual-Server Boolean Data Retrieval for Highly-Scalable Secure File Sharing Services

Kai Zhang, Xiwen Wang, Jianting Ning, *Member, IEEE*, and Xinyi Huang

***Abstract*— Searchable encryption (SE) is a promising strategy for cloud-based file retrieval services, via structuring correspondences between files and keywords. Public key encryption with keyword search (PEKS) has been generally employed in file-sharing services, as compared to searchable symmetric encryption (SSE). However, PEKS is inherently vulnerable to keyword guessing attacks (KGA) launched by a malicious server. To resist such attacks, classic solutions are dual-server PEKS (DS-PEKS) [TIFS'2015] and server-aided PEKS (SA-PEKS) [TIFS'2016]. However, the query model in these two solutions only support single keyword search pattern, which inevitably limits their wide deployments in practice due to efficiency concern. In this work, we present DSB-SE, a new cloud-based file sharing & retrieval system that supports boolean queries while retaining KGA-resistance. Compared to DS-PEKS and SA-PEKS, the cost of documents searching in DSB-SE is $25,000$ times (resp. $6,600$ times) faster when #keyword $= 10$ and s-term $= 1$, where s-term is the least frequent keyword in the query pattern. Technically, the performance gain derives from revisiting traditional boolean SSE by: (i) introducing a pairing-free DDH-based transformation key modular that allows a data reader's query pattern to be treated as a data writer's; (ii) employing the dual-server methodology to support boolean query with efficient validity checks. In particular, the client-to-cloud communication cost for retrieving index of a single document is bounded to $10^{-2}s$, and the cost of sending a token ranges from $8 \times 10^{-2}s \sim 13 \times 10^{-2}s$. Nevertheless, DSB-SE is $1.5 \times 10^{-2}s$ slightly slower than DS-PEKS (but $1.35 \times 10^{-2}s$ faster than SA-PEKS) for key generation cost. Overall, the experiments show that the DSB-SE is practical and sufficient for real cloud applications, which is conducted over Enron dataset under a real-world cloud platform.**

***Index Terms*— Cloud security, searchable encryption, data sharing, keyword search, keyword guessing attacks.**

Manuscript received 13 September 2021; revised 14 April 2022 and 5 October 2022; accepted 12 November 2022. Date of publication 24 November 2022; date of current version 7 December 2022. This work was supported in part by the Shanghai Rising-Star Program under Grant 22QA1403800, in part by the Program of Shanghai Academic Research Leader under Grant 21XD1421500, and in part by the National Natural Science Foundation of China under Grant 61972094 and Grant 62032005. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Frederik Armknecht. *(Corresponding author: Jianting Ning.)*

Kai Zhang and Xiwen Wang are with the College of Computer Science and Technology, Shanghai University of Electric Power, Shanghai 350117, China (e-mail: kzhang@shiep.edu.cn; xwWang486@gmail.com).

Jianting Ning is with the Fujian Provincial Key Laboratory of Network Security and Cryptology, College of Computer and Cyber Security, Fujian Normal University, Fuzhou 350117, China, and also with the State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China (e-mail: jtning88@gmail.com).

Xinyi Huang is with the Thrust of Artificial Intelligence, Information Hub, Hong Kong University of Science and Technology (Guangzhou), Guangzhou 511458, China (e-mail: xinyi@ust.hk).

Digital Object Identifier 10.1109/TIFS.2022.3224669

I. Introduction

CLOUD-BASED data outsourcing services provide limited or unlimited resource pool for users, allowing data to be migrated from the user side to the cloud server. Although the cost of local data managements is greatly reduced, the security concerns are introduced in remote data storage, retrieval and sharing for users. As surveyed in a report about "cloud security solutions" of 300 CISOs by International Data Corporation (IDC) [1], the top priorities for cloud access are maintaining sensitive data confidential, compliance and the right level of access. Therefore, enabling highly-scalable cloud storage with secure data access and retrieval is captured as a CLOUDSEC industry research hotspot.

Generally, remotely outsourcing data via *encryption-then-outsourcing* with support of efficient retrieval have become a primary approach to secure cloud data access [2], [3]. Towards achieving effective data retrieval, the data structure of keyword index is usually employed for users to filter documents via the correspondences between files and keywords. Hence, the encrypted keyword index is certainly required for encrypted cloud storage retrieval and access services. In this way, a data owner encrypts its documents along with associated keywords before uploading them to the cloud, where the deployed data encryption methodologies are asked to support efficient keyword search over encrypted data.

Searchable Encryption: The primitive of *searchable encryption* (SE) [4], [5] has been a widely employed strategy in practical data retrieval services [6], [7], [8], [9], which not only prevents data breach but also realizes efficient encrypted data searching. Generally, there are two kinds of SE: *searchable symmetric encryption* (SSE) and *public key encryption with keyword search* (PEKS). In an SSE system, the data can be only encrypted and accessed by a data owner, while the data encrypted by a data owner in a PEKS system can be accessed by a data reader.

Researchers have reached a consensus in the following [4], [10], [11]: (i) SSE enjoys more expressive query models and higher searching efficiency but only achieves some basic forms of security and applications; (ii) PEKS has more application deployments and considers relatively strong security models, but its searching efficiency and query expressiveness are usually limited.

PEKS Against Inside Keyword Guessing Attacks: The traditional PEKS [5] encrypts a set of keywords $W = (w_1, \cdots, w_n)$ and generates a search token token_Q for a query formula Q, and sends token_Q to a server. However, it rarely considers an inherent threat that an adversarial server may launch *keyword guessing attacks* (KGA) [12], [13] to recover keywords from received tokens. To address the KGA security problem, there are two classic solutions proposed in the literature: dual-server PEKS (DS-PEKS) [14] and server-aided PEKS (SA-PEKS) [15]. In [14], Chen et al. separated server into two independent parts: front server (FS) and back server (BS), where they cannot collude together to launch KGA attacks. However, [14] suffers from high communication cost between FS and BS, this is because FS needs to generate an internal testing-state for each document. In SA-PEKS [15], two independent servers, called keyword server (KS) and storage server (SS), still need to be introduced to resist KGA attacks. For a query, the client needs to first interact with KS to obtain KS-derived keyword. Similar to [14], SS cannot recover keyword from received tokens without the secret key of KS.

To sum up, the PEKS schemes [13], [14], [15] against KGA attacks only focus on single or fuzzy keyword search. Thus, the searching efficiency may be dramatically decreased in practical applications, when to deal with expressive query model of conjunctive and boolean formula.

SSE With Support of Boolean Queries: Song et al. [4] proposed the notion of SSE that enables clients to search over encrypted data, but the scheme only realizes single keyword search. To rich query models, Cash et al. [16] presented a highly-scalable SSE system supporting sub-linear boolean queries. They introduced s-term that used to reduce the search range from all encrypted data to the least set of files. Nevertheless, [16] only considers the situation that a data writer's documents can be only searched by itself. To enable file sharing, several multi-user SSE [17], [18], [19] were recently presented. Based on identity-based encryption (IBE) or attribute-based encryption (ABE), encrypted data in these systems is produced by involving a reader's secret key.

Although the systems [17], [18], [19] already consider data sharing scenario, the inherent inside KGA attack has not yet been formally considered. In addition, trivially combining these schemes with existing KGA security approaches [14], [15] seems straightforward, since the utilized IBE or ABE modular may lead to expensive cost for data sharing.

A. Motivation and Candidate Solution

▷ *Motivation:* To be summarized, state-of-the-art solutions have the following main limitations: (i) most of them against KGA attacks only support single keyword, and rarely consider conjunctive even boolean query models; (ii) previous boolean SE systems for data sharing have never considered KGA security concerns. Hence, an efficient SE system against inside KGA supporting boolean queries for practical scalable cloud-based file sharing services is motivated.

▷ *A Candidate Solution:* Trivially combining DS-PEKS [14] with SSE [16] sounds a naive way to obtain a boolean SE system against KGA attacks, however, it is not a solid solution due to the following reasons:

1) Numerous EDB and $\mathsf{XSet} = \{\mathsf{xtag}\}$ are produced for a data writer: To support data sharing and data search with KGA-resistance, trivially employing DS-PEKS [14], into SSE [16] may make the encryption tuples (i.e., EDB and $\mathsf{XSet} = \{\mathsf{xtag}\}$) in SSE [16] be encrypted repeatedly for different data readers. Moreover, the dual-server framework only works for single keyword search [14] while not easy to be directly employed in [16] that supports conjunctive/boolean queries.
2) Additional computation overhead for data sharing: To extend SSE for data sharing, current solution boolean SE systems are usually designed via a combination of identity-based encryption (IBE) [18] or attribute-based encryption (ABE) [17], [19], which inevitably brings about additional computation cost for data sharing.
3) High communication cost against KGA attacks: Directly applying [14]'s dual-server framework may lead to high communication cost (i.e., $\mathcal{O}(|\mathsf{DB}|)$) between front server and back server for Cash et al.'s SSE [16]. In particular, by receiving a trapdoor from the client, the front server needs to generate internal testing-state for *every encrypted tuple* in the database.

B. Our Results

This work proposes a Dual-Server Boolean Searchable Encryption scheme (DSB-SE) for cloud-based data sharing services. The scheme enables a data writer's cloud-based documents to be accessed by another data reader, and effectively prevents the cloud from exhaustively guessing the underlying keywords from search tokens. To this end, we first revisit boolean searchable symmetric encryption, and introduce a new algorithm that enables a data reader's search token to act as a data writer's token. Moreover, we employ the dual-server KGA-resilient approach to separate the authority of cloud into two independent servers, in order to complete validity checks over query patterns. Furthermore, extensive experiments on Enron dataset under a real-world cloud platform is conducted. In particular, our contributions are described as follows.

- **Sub-linear Boolean Queries for Scalable Cloud-based Documents Sharing.** To enable a data reader to issue boolean queries over a data writer's documents with sub-linear complexity, we revisit [16] and introduce a transformation key tk. The tk is used to efficiently convert the search token of a data reader to a valid search token, which is similar to the role of a data writer's token in traditional boolean SE schemes. Note that the tk is a simple, efficient DDH-based instance as $\mathsf{tk} = (g^r, X \cdot g^{\beta \cdot r}, g^{-\alpha} \cdot H_1(X), g^{\beta/\alpha})$, where r, X are randomnesses, H_1 is a hash function and $\mathsf{pk}_{\mathsf{rea}} = g^{\beta}$ is the shared data reader's public key. Note that in DSB-SE, a data writer only needs to encrypt the documents once for different data readers, while the candidate solution needs a set of encryptions for each data reader.
- **Security Against Inside Keyword Guessing Attacks.** To achieve the desirable security against inside KGA attacks launched by the cloud, we employ an efficient dual-server methodology to separate the cloud server into front server

TABLE I
FEATURE COMPARISON WITH RELATED WORK

Work	Boolean Query	KGA Security	Data Sharing	Hardness Assumption	Search Cost	Token Size
[13], [14], [15]	✗	✓	✓	Specific number-theoretic	$\mathcal{O}(\#\mathsf{Doc})$	$q \cdot \mathcal{O}(1)$
[16], [20]	✓	✗	✗	PRF, DL	$\mathcal{O}(c_{w_1})$	$\mathcal{O}(q \cdot c_{w_1})$
[17], [18], [19]	✓	✗	✓	PRF, DL, IBE/ABE	$\mathcal{O}(n \cdot c_{w_1})$	$\mathcal{O}(q \cdot c_{w_1})$
Our work	✓	✓	✓	PRF, DL, DDH	$\mathcal{O}(c_{w_1})$	$\mathcal{O}(q \cdot c_{w_1})$

"IBE" denotes Identity-Based Encryption and "ABE" denotes Attribute-Based Encryption. "#Doc" denotes the number of documents in a database; "c_{w_1}" denotes the number of the least frequent keyword in a query pattern; "n" denotes the number of users in an IBE system or the size of attribute universe in an ABE system; "q" denotes the number of keywords in a query pattern.

and back server. Each of them only holds partial knowledge of each $\mathsf{Trap} = (g^{r_2}, H(w)^{1/(\beta \cdot z_i)} \cdot (g^{\gamma} \cdot g^{\eta})^{r_2})$ of a reader's search token. In particular, the front server treats the received search token $\mathsf{token}_{\mathsf{rea}}$ as a $\mathsf{token}_{\mathsf{FS}}$, and utilize it to generate another $\mathsf{token}_{\mathsf{BS}}$ for the back server. Then, the back server transforms the token $\mathsf{token}_{\mathsf{BS}}$ with a corresponding transformation key tk, and checks whether the underlying query of $\mathsf{token}_{\mathsf{BS}}$ exists in XSet, and finally returns target files. In addition, we reduce communication cost between front server and back server from $\mathcal{O}(|\mathsf{DB}|)$ in a candidate solution to $\mathcal{O}(c_{w_1})$.

To clarify practical performance, we employ HUAWEI Cloud platform to simulate an experiment environment from the sides of client, cloud and client-to-cloud communication. In particular, we implement our DSB-SE and related work over a real-world dataset Enron. By achieving small 4KB∼27KB token size, our DSB-SE runs 6,600× even 25,000× faster for documents searching compared with related work. In addition, the client-to-cloud communication cost for retrieving index of a single document is always bounded with 0.01 seconds.

1) Comparison: Table I shows that DSB-SE supports more functionality-features and achieves high document search efficiency with comparable token size, whose security can be reduced to simple assumptions. Current PEKS with KGA-resistance schemes [13], [14], [15] employed the traditional index model that leads to searching complexity as $\mathcal{O}(\#\mathsf{Doc})$. That is, the server needs to perform a match between every document and a search trapdoor. The inverted index model and the least frequency keyword are introduced to efficiently locate the least documents as in [16], which enables our DSB-SE to achieve sub-linear searching complexity of $\mathcal{O}(c_{w_1})$. For a conjunctive query that includes q keywords, the searching complexity of [13], [14], and [15] suffers from $q \cdot \mathcal{O}(\#\mathsf{Doc})$, while DSB-SE always achieves $\mathcal{O}(c_{w_1})$ searching overhead. The token size in our DSB-SE is c_{w_1} times bigger than [13], [14], and [15], but is highly acceptable due to support of highly-efficient expressive query models. Compared to [16] and [20], our DSB-SE supports data sharing property and KGA-resistance security property while still maintains sub-linear searching complexity. Moreover, the security of our DSB-SE can be reduced to rather simple assumptions (i.e., DL or DDH) instead of complicated primitives, such as IBE and ABE [17], [18], [19]. In addition, we implement related work and our DSB-SE, and give a performance comparison about the experiment results in Section VI.

TABLE II
NOTATIONS

Notation	Meaning
$[n]$	$\{1, 2, \cdots, n\}$.
$\mathbf{A}$	An array.
$\mathbf{A}[i]$	The i-th element of $\mathbf{A}$.
$\vert\mathbf{A}\vert$	The length of $\mathbf{A}$.
ind	The index of a document Doc.
w	A keyword.
W_{ind}	A set of keywords $W = (w_1, w_2, \cdots, w_q)$.
Doc	A document Doc is labeled with (ind, W_{ind}).
s-term	The least frequent keyword in a query pattern.
xterm	Any queried keyword in a query.
EDB	An encrypted index of a document.
XSet	An encrypted keyword set of a document.
tk	A transformation key for converting a search token.
$\mathsf{token}_{\mathsf{FS}}$	A token sent to the front server.
$\mathsf{token}_{\mathsf{BS}}$	A token sent to the back server.

2) Organization: Section II reviews some background knowledge and Section III formalizes the system model and security model. In Section IV, we present a SE system, and analyze its security in Section V. Section VI gives a performance analysis on DSB-SE and existing work. Finally, we survey related work in Section VII, and conclude this work in Section VIII.

II. BACKGROUND KNOWLEDGE

In this section, we review background knowledge, where the notations that used in this paper are shown in Table II

Definition 1 (Inverted Index): An inverted index data structure builds a set of keyword pointer linked to documents, which enables search engines to proceed less searching time-consuming cost than forward index. Specifically, the inverted index is set up by a dictionary δ to record the state about the size of $\mathsf{DB}[w]$ for each keyword, which includes two functions:

- $c \leftarrow \mathsf{Get}(w, \delta)$: Inputs a keyword w and the dictionary δ, it outputs c as a counter that records the current size of $\mathsf{DB}[w]$. It returns 0 if w does not exist in δ.
- $\mathsf{Update}(\delta, w, c)$: Inputs the dictionary δ, a keyword w and a number c, updates the recorded size of $\mathsf{DB}[w]$ in δ to c. It inserts (w, c) into δ if w does not exist in δ.

Definition 2 (PRF): 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 $\mathcal{A}$. That is, for any PPT adversary $\mathcal{A}$, the advantage is defined as

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

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

Definition 3 (DL Assumption): Let $\mathbb{G}$ be a cyclic group with a prime order p, g be a generator of $\mathbb{G}$. Let h also be an element of $\mathbb{G}$. Discrete Logarithm problem is to distinguish g^a from h for any positive integer a. For any PPT distinguisher $\mathcal{D}$, the advantage is defined as

$$\mathsf{Adv}^{\mathsf{DL}}_{\mathcal{D},\mathbb{G}}(\kappa) = |\Pr[\mathcal{D}(g, g^a)] - \Pr[\mathcal{D}(g, h)]|.$$

The DL assumption says $\mathsf{Adv}^{\mathsf{DL}}_{\mathcal{D},\mathbb{G}}(\kappa)$ is negligible in κ for any PPT distinguisher $\mathcal{D}$.

Definition 4 (DDH Assumption): Let $\mathbb{G}$ 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 $\mathbb{G}$ and a, b, r are randomly selected from $\mathbb{Z}_p$. For any PPT distinguisher $\mathcal{D}$, the advantage is defined as

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

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

III. Problem Formulation

A. System Model

There are three different entities in the DSB-SE system: *Key Generation Center (KGC), Cloud Server (Cloud)* and *Clients*.

- KGC: It is a trusted entity that initializes a system and generates public parameter, and produces public key and private key pair for cloud and clients.
- Cloud: The cloud is a *semi-honest* server that provides encrypted documents storage and file-sharing services for clients, but it may launch inside keywords guessing attacks, which consists of a *front server* and a *back server*.
- Clients: The clients include multiple data writers and multiple data readers. In particular, a data writer uploads encrypted documents to cloud and shares its documents with a data reader, in which the data reader can issue any boolean query over them.

B. System Running Flow

The DSB-SE system for secure and highly-scalable cloud-based files sharing services include the following running flows (as depicted in Fig. 1), where the formal function definitions are described in Section IV.

1) **System Initialization:** The KGC runs global setup ($\mathsf{GlobalSetup}$) algorithm inputs a security parameter and generates a system public parameter. Moreover, the KGC runs key generation algorithms ($\mathsf{KeyGen}_{\mathsf{Ser}}$ and $\mathsf{KeyGen}_{\mathsf{Clnt}}$) and distributes public key and secret key pair to front server, back server and clients.
2) **Encrypted Documents Generation:** A data owner runs a document encryption algorithm ($\mathsf{EDBSetup}$) to encrypt documents under its secret key $\mathsf{sk}_{\mathsf{wri}}$, and outputs encrypted documents tuples.
3) **Query Issuing of A data Reader:** A data reader runs a trapdoor generation algorithm ($\mathsf{TrapGen}$) to issue a boolean query and generates a search token. Later, the data writer calls the transformation key generation algorithm (TKGen) to generate data-sharing transformation key to the cloud.
4) **Document Retrieval in Cloud:** The front server runs front_server test algorithm ($\mathsf{FrontTest}$) to produce a new search token under the received data reader's search token. Then, the back server first checks the validity of the front_server's token and runs the documents searching algorithm (Search) to locate the corresponding files and return them to the data reader.

$\triangleright$ **Design Challenges.** The design challenges of our system are assumed as follows: (i) The case of supporting boolean queries; (ii) The case of insider keyword guessing attacks; (iii) The case of supporting cloud-based file sharing; (iv) The case of practical utility based on provably security.

C. Design Goals

To address the design challenges, we formalize the design goals of the system as follows.

- **Documents retrieval with boolean query models.** A client can proceed expressive boolean queries over encrypted documents in the cloud.
- **Resilient to inside keyword guessing attacks.** A malicious server learns nothing about sensitive information/keywords from client's search tokens.
- **Documents sharing over scalable cloud storage.** A data reader is able to search the data writer's documents in scalable cloud-based data access services.
- **Practical efficiency.** The time-consuming cost for realizing query models and data sharing can be practically deployed in real-world environment.
- **Running securely.** Only the allowed information is leaked to public/cloud, and the security is based on simple hardness assumptions.

D. Security Guarantee Model

The cloud and clients are assumed to be semi-honest and not collude with each other, as well as no collusion attack between clients. Although they perform protocols honestly, (i) the cloud tries to get private information of clients; (ii) the client tries to gain private information beyond its authorization.

1) Security Against Adversary Server: This security model captures the case of cloud server trying to get sensitive information of encrypted data and clients' query pattern, which is parameterized by a leakage function $\mathcal{L}$ (that describes the information of outsourced database and queries) that allowed to be known by cloud server.

Definition 5: Let Π be a scheme that described in Section IV-A. For two efficient algorithms $\mathcal{A}$ and $\mathcal{S}$, we define the security via the following two experiments:

Real$^{\Pi}_{\mathcal{A}}(\lambda)$: $\mathcal{A}(\lambda)$ repeatedly chooses an encryption tuple $(\mathbf{D}, \mathsf{pk}_{\mathsf{wri}})$ or a query tuple $(Q, \mathsf{sk}_{\mathsf{rea}})$, where $\mathsf{pk}_{\mathsf{wri}}$ is the public key of the writer of the document and $\mathsf{sk}_{\mathsf{rea}}$ is the secret key of a reader who issues a query. If an encryption tuple is chosen, the experiment runs $\mathsf{EDBSetup}(\mathsf{pp}, \mathbf{D}, \mathsf{sk}_{\mathsf{wri}})$ and returns $(\mathsf{EDB}, \mathsf{XSet})$ to $\mathcal{A}$; otherwise it runs $\mathsf{TrapGen}(\mathsf{pp}, \mathsf{sk}_{\mathsf{rea}}, Q)$ and $\mathsf{Search}(\mathsf{token}_{\mathsf{BS}}, \mathsf{tk})$, and sends $(\mathsf{token}_{\mathsf{FS}}, \mathsf{Res})$ to $\mathcal{A}$. Finally, $\mathcal{A}$ returns a bit as an output of the experiment.

Ideal$^{\Pi}_{\mathcal{A},\mathcal{S}}(\lambda)$: The experiment initializes two empty lists $\mathbf{d}$ and $\mathbf{q}$, sets two counters $i = 1$ and $j = 1$. $\mathcal{A}(1^{\lambda})$ repeatedly chooses an encryption tuple $(\mathbf{D}, \mathsf{pk}_{\mathsf{wri}})$ or a query tuple $(Q, \mathsf{sk}_{\mathsf{rea}})$. if an encryption tuple is chosen, $\mathcal{A}$ records this tuple as $\mathbf{d}[i]$ and increments i, the experiment runs $\mathbf{S}(\mathcal{L}(\mathbf{d}, \mathbf{q}))$ and returns $(\mathsf{EDB}, \mathsf{XSet})$ to $\mathcal{A}$; otherwise the experiment records the search tuple as $\mathbf{q}[j]$, increases j and runs $\mathcal{S}(\mathcal{L}(\mathbf{d}, \mathbf{q}))$ to output a transcript to $\mathcal{A}$. In the end, $\mathcal{A}$ returns a bit as the output of the experiment.

We say that Π is $\mathcal{L}$-semantically secure against an adaptive adversary if there exists an algorithm $\mathcal{S}$ s.t.

$$\Pr[\mathbf{Real}^{\Pi}_{\mathcal{A}}(\lambda) = 1] - \Pr[\mathbf{Ideal}^{\Pi}_{\mathcal{A},\mathcal{S}}(\lambda) = 1] \le \mathsf{negl}(\lambda).$$

2) Security Against Adversary Client: This security model considers the case of a client trying to get information beyond its authorization via forging valid search tokens of others. We define the security via the following game $\mathsf{Game}^{\mathsf{UF}}_{\mathcal{A},\mathsf{token}}$ between a challenger and an adversary $\mathcal{A}$.

Initialization. The challenger runs $\mathsf{GlobalSetup}(\lambda)$ and returns the public parameter pp to the $\mathcal{A}$.

Key extraction. Receiving a key query request, the challenger runs $\mathsf{KeyGen}_{\mathsf{Clnt}}(\mathsf{pp})$ and returns a key pair $(\mathsf{pk}_{\mathsf{rea}}, \mathsf{sk}_{\mathsf{rea}})$ to $\mathcal{A}$.

Challenge. $\mathcal{A}$ chooses a challenge data reader rea^*, and the challenger runs $\mathsf{KeyGen}_{\mathsf{Clnt}}(\mathsf{pp})$ and returns $\mathsf{pk}_{\mathsf{rea}^*}$ to $\mathcal{A}$.

Output. $\mathcal{A}$ outputs a search token for the challenge client, and the challenger outputs 1 if the search token is valid.

Definition 6: We say that a search token is said to be unforgeable in Π if the advantage $\Pr[\mathsf{Game}^{\mathsf{UF}}_{\mathcal{A},\mathsf{token}}(\lambda) = 1]$ is negligible for all PPT adversary $\mathcal{A}$.

3) Security Against Keyword Guessing Attacks: This security model considers the case of front server and back server trying to get query information by launching KGA attacks. Let $\mathcal{A}$ be an attacker whose running time is bounded by a polynomial in a security parameter λ. We consider the following two games:

$\triangleright$ **Game 1:** The adversary $\mathcal{A}$ is assumed to be a front server.

Initialization. Runs $\mathsf{GlobalSetup}(\lambda)$, $\mathsf{KeyGen}_{\mathsf{Clnt}}(\lambda)$ and $\mathsf{KeyGen}_{\mathsf{Ser}}(\lambda)$, and generates public parameter pp, key pairs of the server $(\mathsf{sk}_{\mathsf{FS}}, \mathsf{pk}_{\mathsf{FS}}, \mathsf{sk}_{\mathsf{BS}}, \mathsf{pk}_{\mathsf{BS}})$ and a data reader $(\mathsf{sk}_{\mathsf{rea}}, \mathsf{pk}_{\mathsf{rea}})$. Sends $(\mathsf{pp}, \mathsf{pk}_{\mathsf{rea}}, \mathsf{pk}_{\mathsf{BS}}, \mathsf{sk}_{\mathsf{FS}}, \mathsf{pk}_{\mathsf{FS}})$ to $\mathcal{A}$, while keeps $\mathsf{sk}_{\mathsf{rea}}$ and $\mathsf{sk}_{\mathsf{BS}}$ secret from $\mathcal{A}$.

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

- Token query: $\mathcal{A}$ adaptively asks the challenger for the query token token towards a query Q. The challenger responds a $\mathsf{token}_{\mathsf{FS}} \leftarrow \mathsf{TrapGen}(\mathsf{pp}, \mathsf{sk}_{\mathsf{rea}}, Q)$ to $\mathcal{A}$.
- Test query: $\mathcal{A}$ adaptively asks the challenger for a token query $\mathsf{token}_{\mathsf{FS}}$ towards a query Q. The challenger returns a bit $b \leftarrow \mathsf{BackTest}(\mathsf{pp}, \mathsf{token}_{\mathsf{BS}}, \mathsf{sk}_{\mathsf{BS}}, \mathsf{tk})$ to $\mathcal{A}$.

Challenge. $\mathcal{A}$ chooses a target keyword set pair (W_0^*, W_1^*). Receiving this, $\mathcal{C}$ picks $\beta \in \{0, 1\}$ uniformly at random and generates a challenge token $\mathsf{token}_{\mathsf{FS}}{}^* \leftarrow \mathsf{TrapGen}(\mathsf{pp}, \mathsf{sk}_{\mathsf{rea}}, Q)$ and returns it to $\mathcal{A}$.

Phase 2. $\mathcal{A}$ issues a number of trapdoor extraction queries as before, with the restriction that W_0 and W_1 are not allowed to be queried as trapdoor extraction queries.

Guess. $\mathcal{A}$ outputs its guess $\beta' \in 0, 1$ of β and wins the game if $\beta' = \beta$.

Note that the trapdoor reveals no information about the underlying keyword to the adversarial front server. We refer to such an adversarial front server in **Game 1** as an IND-KGA adversary and define its winning advantage as

$$\mathsf{Adv}^{\mathsf{IND-KGA}}_{\mathsf{FS},\mathcal{A}}(\lambda) = \Pr[\beta' = \beta] - 1/2.$$

$\triangleright$ **Game 2:** The adversary $\mathcal{A}$ is assumed to be a back server.

This security model is the same as the **Game 1**, except that the adversary owns the secret key $\mathsf{sk}_{\mathsf{BS}}$ of the back server instead of $\mathsf{sk}_{\mathsf{FS}}$ of the front server. Similarly, We refer to such an adversarial back server in **Game 2** as an IND-KGA adversary and define its winning advantage as

$$\mathsf{Adv}^{\mathsf{IND-KGA}}_{\mathsf{BS},\mathcal{A}}(\lambda) = \Pr[\beta' = \beta] - 1/2.$$

IV. DSB-SE: The Proposed Scheme

In this section, we first present a dual-server conjunctive SE system (DSC-SE) for highly-scalable cloud-based data sharing, and then enable it with boolean queries (DSB-SE) as an enhanced extension.

A. DSC-SE: Formal Construction

Let $\mathbb{G}, \mathbb{G}_T$ be groups with prime order p and a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, where g is a generator of $\mathbb{G}$. Let F be a pseudo-random function (PRF) with range in $\{0, 1\}^*$, F_p be a PRF with range in $\mathbb{Z}_p$, $H_1 : \{0, 1\}^n \to \mathbb{G}$, and $H_2 : \{0, 1\}^* \to \{0, 1\}^n$ are two collision-resistant hash functions, where the formal system routine is shown in Fig. 1.

- $\mathsf{GlobalSetup}(\lambda) \to \mathsf{pp}$: Takes as input a secure parameter λ, KGC selects two keys $K_x, K_z \leftarrow_R \{0, 1\}^n$ for F_p and a key $K_l \leftarrow_R \{0, 1\}^n$ for F. The algorithm returns the public parameter $\mathsf{pp} = (\mathbb{G}, \mathbb{G}_T, p, e, g, F, F_p, H_1)$.
- $\mathsf{KeyGen}_{\mathsf{Ser}}(\mathsf{pp}) \to (\mathsf{pk}_{\mathsf{FS}}, \mathsf{sk}_{\mathsf{FS}}, \mathsf{pk}_{\mathsf{BS}}, \mathsf{sk}_{\mathsf{BS}})$: Inputs pp, the KGC selects $\gamma, \eta \leftarrow_R \mathbb{Z}_p$ and computes g^{γ}, g^{η}.

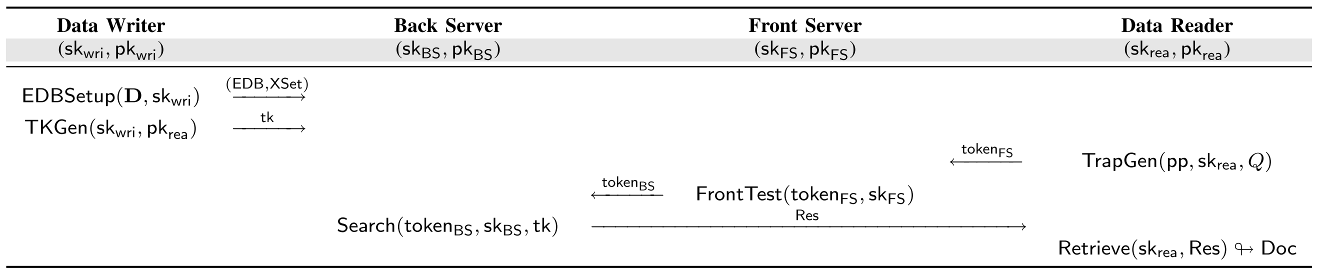

Data Writer
$(\mathsf{sk_{wri}}, \mathsf{pk_{wri}})$
Back Server
$(\mathsf{sk_{BS}}, \mathsf{pk_{BS}})$
Front Server
$(\mathsf{sk_{FS}}, \mathsf{pk_{FS}})$
Data Reader
$(\mathsf{sk_{rea}}, \mathsf{pk_{rea}})$
$\mathsf{EDBSetup}(\mathbf{D}, \mathsf{sk_{wri}}) \xrightarrow{(\mathsf{EDB},\mathsf{XSet})}$
$\mathsf{TKGen}(\mathsf{sk_{wri}}, \mathsf{pk_{rea}}) \xrightarrow{\mathsf{tk}}$
$\xleftarrow{\mathsf{token_{FS}}}$
$\mathsf{TrapGen}(\mathsf{pp}, \mathsf{sk_{rea}}, Q)$
$\xleftarrow{\mathsf{token_{BS}}}$
$\mathsf{FrontTest}(\mathsf{token_{FS}}, \mathsf{sk_{FS}})$
$\mathsf{Search}(\mathsf{token_{BS}}, \mathsf{sk_{BS}}, \mathsf{tk}) \xrightarrow{\mathsf{Res}}$
$\mathsf{Retrieve}(\mathsf{sk_{rea}}, \mathsf{Res}) \nrightarrow \mathsf{Doc}$

Fig. 1. System model of dual-sever searchable encryption supporting boolean queries for highly-scalable cloud-based data sharing services.

The algorithm outputs the public key and secret key pair for the front server as $\mathsf{pk_{FS}} = g^{\gamma}$, $\mathsf{sk_{FS}} = (\gamma, K_x, K_z, K_l)$ and for the back server $\mathsf{pk_{BS}} = g^{\eta}$, $\mathsf{sk_{BS}} = (\eta, K_x, K_z, K_l)$.

- $\mathsf{KeyGen_{Clnt}}(\mathsf{pp}) \rightarrow (\mathsf{pk_u}, \mathsf{sk_u})$: Inputs pp, KGC selects $\beta \leftarrow_R \mathbb{Z}_p$ and computes g^{β}. The algorithm outputs the public key and secret key pair for the client as $\mathsf{pk_u} = g^{\beta}$ and $\mathsf{sk_u} = (\beta, K_x, K_z, K_l)$.
- $\mathsf{EDBSetup}(\mathsf{pp}, \mathbf{D}, \mathsf{sk_{wri}}) \rightarrow (\mathsf{EDB}, \mathsf{XSet})$: Inputs a document $\mathbf{D} = (ind, W_{ind})$ and a data writer's secret key $\mathsf{sk_{wri}} = \alpha$, it encrypts the document by performing the Algorithm 1. Assume each document has a unique indice ind, and the original document of ind is encrypted by a symmetric key encryption algorithm (e.g., AES) with a key K_{ind}. We state that δ cannot be retrieved by more than one client for encrypting documents.
- $\mathsf{TrapGen}(\mathsf{pp}, \mathsf{sk_{rea}}, Q) \rightarrow \mathsf{token_{FS}}$: To proceed a conjunctive query $Q = (w_1 \wedge \cdots \wedge w_n)$, a data reader uses its secret key $\mathsf{sk_R}$ to generate a search token towards Q by performing the Algorithm 1 where w_1 is assumed as the s-term of Q.
- $\mathsf{TKGen}(\mathsf{pp}, \mathsf{sk_{wri}}, \mathsf{pk_{rea}}) \rightarrow \mathsf{tk}$: Takes as input pp, $\mathsf{sk_{wri}} = (\alpha, K_x, K_z, K_l)$ and $\mathsf{pk_{rea}} = g^{\beta}$. The writer randomly selects $r, X \leftarrow_R \mathbb{G}$ and outputs a transformation key $\mathsf{tk} = (g^r, X \cdot g^{\beta \cdot r}, g^{-\alpha} \cdot H_1(X), g^{\beta/\alpha})$ for the back server.
- $\mathsf{FrontTest}(\mathsf{pp}, \mathsf{token_{FS}}, \mathsf{sk_{FS}}, \mathsf{tk}) \rightarrow \mathsf{token_{BS}}$: Takes as input $\mathsf{pp}, \mathsf{token_{FS}}$ and $\mathsf{sk_{FS}}$, the front server outputs a token $\mathsf{token_{BS}}$ for back server by performing Algorithm 1.
- $\mathsf{BackTest}(\mathsf{pp}, \mathsf{token_{BS}}, \mathsf{sk_{BS}}, \mathsf{tk}) \rightarrow b$: Takes as input $\mathsf{pp}, \mathsf{token}$ and $\mathsf{sk_{BS}}$, the back server checks whether the underlying query of $\mathsf{token_{BS}}$ exists in XSet and outputs a bit by performing Algorithm 1.
- $\mathsf{Search}(\mathsf{token_{BS}}, \mathsf{sk_{BS}}, \mathsf{tk}) \rightarrow \mathsf{Res}$: Given a $\mathsf{token_{BS}}$ from a front server and a transformation key tk from writer, the back server returns results as in Algorithm 1.
- $\mathsf{Doc} \leftarrow \mathsf{Retrieve}(\mathsf{Res}, \mathsf{sk_{rea}})$: Finally, the data reader recovers $(ind||K_{ind}) \leftarrow \mathsf{Dec}(\mathsf{sk_{rea}}, e_0)$ for e_0 in Res and sends inds to the cloud server. After fetches corresponding encrypted documents, it descrpts them with the corresponding key K_{ind}.

B. Correctness Guarantee

$\triangleright$ **(i) Correctness for recovering xtag with a trapdoor.** Given a trapdoor (T_1, T_2) and key pair tuples of $(\mathsf{pk_{FS}}, \mathsf{sk_{FS}}), (\mathsf{pk_{BS}}, \mathsf{sk_{BS}}), (\mathsf{pk_{wri}}, \mathsf{sk_{wri}})$ and $(\mathsf{pk_{rea}}, \mathsf{sk_{rea}})$ the front server and back server compute (T_γ, U_γ) and (T_η, U_η) with their secret key as:

$$\begin{aligned}
T_\gamma &= T_2^{\gamma}/T_1^{\gamma^2} = H_1(w)^{\gamma/(\beta \cdot z_i)} \cdot g^{\gamma \cdot \eta \cdot r_2},\\
U_\gamma &= e(g^{\beta/\alpha \cdot e_1}, T_\gamma) = e(g^{\beta \cdot \mathsf{xind} \cdot z}, H_1(w)^{\gamma/(\beta \cdot z_i)} \cdot g^{\gamma \eta r_2}),\\
T_\eta &= T_2^{\eta}/T_1^{\eta^2} = H_1(w)^{\eta/(\beta \cdot z_i)} \cdot g^{\gamma \cdot \eta \cdot r_2},\\
U_\eta &= e(g^{\beta/\alpha \cdot e_1}, T_\eta^{-1}) = e(g^{\beta \cdot \mathsf{xind} \cdot z}, H_1(w)^{-\eta/(\beta \cdot z_i)} \cdot g^{-\gamma \eta r_2}).
\end{aligned}$$

With the knowledge of $g^{\beta/\alpha}$ in the transformation key tk, xtag can be recovered from the following:

$$\begin{aligned}
T_\gamma T_\eta^{-1} &= H_1(w)^{(\gamma - \eta) \cdot /(\beta \cdot z_i)},\\
U_\gamma U_\eta &= e(g^{\beta/\alpha \cdot e_1}, T_\gamma) e(g^{\beta/\alpha \cdot e_1}, T_\eta^{-1})\\
&= e(g^{\beta/\alpha \cdot e_1}, T_\gamma T_\eta^{-1}) = e(g^{\gamma - \eta}, H_1(w)^{\mathsf{xind}}) = \mathsf{xtag}.
\end{aligned}$$

$\triangleright$ **(ii) Correctness for documents decryption with a transformation key.** Given transformed encrypted tuples $e = (c_1, c_2, c_3) = (g^{r_1}, m \cdot g^{\beta \cdot r_1} \cdot H_1(X), (g^{r_3}, X \cdot g^{\beta \cdot r_3}))$, a data reader is able to decrypt c_3 with its secret key β to obtain X. Hence, the message m can be certainly recovered by decrypting c_2 with the knowledge of g^{r_1}, β and X,

C. DSB-SE: Extension Supporting Boolean Queries

We continue to extend DSC-SE to support boolean queries, i.e. $Q = (w_1 \wedge \psi(w_2, \cdots, w_n)$, where ψ is an arbitrary boolean formula over $(w_2, \cdots, w_n)$. In particular, a data reader computes $\mathsf{token_{FS}}$ as in DSC-SE, where the front server sends a computed $\mathsf{token_{BS}}$ under a boolean formula $\bar{\psi}$ to the back server. Note that $\bar{\psi}$ is the same as ψ, except that the keywords are replaced by $(w'_2, \cdots, w'_n)$. Later, the back server uses l_i to retrieve the tuples (e_0, e_1) that is associated with w_1, with the difference for determining which tuples (e_0, e_1) match $\bar{\psi}$. For each tuple $(e_0, e_1) \leftarrow \mathsf{EDB}[l_i]$, the back server computes $(w'_2, \cdots, w'_n)$ as

$$w_j = \begin{cases} 1, & \text{if } U_\gamma U_\beta \in \mathsf{XSet};\\ 0, & \text{otherwise.} \end{cases}$$

where $j = 2, \cdots, n$. If the value of $\bar{\psi}$ is true and $U_\gamma U_\beta \in \mathsf{XSet}$, then e_0 can be added to Res since the tuple matches this query indicated. In DSB-SE, the time complexity of processing a boolean query is $\mathcal{O}(|\mathsf{DB}[w_1]|)$ (where w_1 is assumed as the s-term) is the same as a conjunctive query in DSC-SE. Moreover, the leakage information is also the same as processing a conjunctive search, except that $\bar{\psi}$ is leaked.

Algorithm 1 DSC-SE: Dual-Server Conjunctive Searchable Encryption for Secure Cloud-Based Data Sharing

```
function EDBSetup(pp, $\mathbf{D}$, sk$_{\mathsf{wri}}$))
    EDB $\leftarrow \{\}$, XSet $\leftarrow \{\}$
    sk$_{\mathsf{wri}} = \alpha$ pk$_{\mathsf{FS}} = g^{\gamma}$, pk$_{\mathsf{BS}} = g^{\eta}$
    for $w \in W_{ind}$ do
        $c_w \leftarrow$ Get$(H_2(w), \delta)$, $c_w = c_w + 1$
        Update$(\delta, H_2(w), c_w)$
    end for
    $xind \leftarrow F_p(K_x, ind)$
    for $w \in W_{ind}$ do
        $t \leftarrow_R \mathbb{Z}_p$, $l \leftarrow F(K_l, w||c_w)$, $z \leftarrow F_p(K_z, w||c_w)$
        $e_0 \leftarrow (g^{r_1}, (ind||K_{ind}) \cdot e(g^{r_1}, H_1(g^{\alpha}))^{\alpha}$
        $e_1 \leftarrow$ xind $\cdot z \cdot \alpha$
        xtag $\leftarrow e(g^{\gamma-\eta}, H_1(w)^{\mathsf{xind}})$
        EDB$[l] = (e_0, e_1)$, XSet $\leftarrow$ XSet $\cup$ xtag
    end for
    return EDB, XSet
end function

function TrapGen($Q$, pk$_{\mathsf{FS}}$, pk$_{\mathsf{BS}}$, sk$_{\mathsf{rea}}$)
    $Q = (w_1 \wedge \cdots \wedge w_n)$, $r_2 \leftarrow_R \mathbb{Z}_p$
    pk$_{\mathsf{FS}} = g^{\gamma}$, pk$_{\mathsf{BS}} = g^{\eta}$, sk$_{\mathsf{rea}} = (\beta, K_x, K_z, K_l)$
    for $i = 1, 2, \cdots$ until cloud server sends stop do
        $l_i \leftarrow F(K_l, w_1||i)$, $z_i \leftarrow F_p(K_z, w_1||i)$
        for $j \in [n]$ do
            $(T_1, T_2) = (g^{r_2}, H(w)^{1/(\beta \cdot z_i)} \cdot (g^{\gamma} \cdot g^{\eta})^{r_2})$
            Trap$[i][j] = (T_1, T_2)$
        end for
        Send $(l_i$, Trap$[i])$ to the cloud server.
    end for
    return token$_{\mathsf{FS}} = ((l_i)_{i=1}^{c}$, Trap)
end function

function FrontTest(pp, token$_{\mathsf{FS}}$, sk$_{\mathsf{FS}}$)
    token$_{\mathsf{FS}} = ((l_1$, Trap$[1]), (l_2$, Trap$[2]), \cdots)$
    sk$_{\mathsf{FS}} = (\gamma, K_x, K_z, K_l)$
    $i = 1$, STag $\leftarrow \{ \}$
    while $l_i \in$ EDB do
        $(e_0, e_1) \leftarrow$ EDB$[l_i]$, Trap$[i][j] = (T_{1,j}, T_{2,j})$
        $T_{\gamma,j} = T_{2,j}^{\gamma} / T_{1,j}^{\gamma^2}$
        STag$[i][j] = T_{\gamma,j}$
    end while
    return token$_{\mathsf{BS}} = ((l_i)_{i=1}^{c}$, Trap, STag)
end function

function BackTest(pp, token$_{\mathsf{BS}}$, sk$_{\mathsf{BS}}$, tk)
    token$_{\mathsf{BS}}$ = $((l_1$, Trap$[1]$, STag$[1])$, $(l_2$, Trap$[2]$, STag$[2]), \cdots)$
    $i = 1$, $b = 1$, sk$_{\mathsf{BS}} = (\eta, K_x, K_z, K_l)$
    tk $= (g^{r_3}, X \cdot g^{\beta \cdot r_3}, H_1(g^{\alpha})^{-\alpha} \cdot H_1(X), g^{\beta/\alpha})$
    while $l_i \in$ EDB do
        $(e_0, e_1) \leftarrow$ EDB$[l_i]$, Trap$[i][j] = (T_{1,j}, T_{2,j})$
        STag$[i][j] = T_{\gamma,j}$, $U_{\gamma,j} = e(g^{\beta/\alpha \cdot e_1}, T_{\gamma,j})$
        $T_{\eta,j} = T_{2,j}^{\eta} / T_{1,j}^{\eta^2}$, $U_{\eta,j} = e(g^{\beta/\alpha \cdot e_1}, T_{\eta,j}^{-1})$
        if $U_{\gamma} U_{\eta} \in$ XSet not for all $j$ then
            $b = 0$
        end if
        $i = i + 1$
    end while
    return $b$
end function

function TKGen(pp, sk$_{\mathsf{wri}}$, pk$_{\mathsf{rea}}$))
    $r_3 \leftarrow_R \mathbb{Z}_p$,$X \leftarrow_R \mathbb{G}$
    sk$_{\mathsf{wri}} = (\alpha, K_x, K_z, K_l)$, pk$_{\mathsf{rea}} = g^{\beta}$
    tk $= (g^{r_3}, X \cdot g^{\beta \cdot r_3}, g^{-\alpha} \cdot H_1(X), g^{\beta/\alpha})$
    return tk
end function

function Search(token$_{\mathsf{BS}}$, sk$_{\mathsf{BS}}$, tk)
    token$_{\mathsf{BS}}$ = $((l_1$, Trap$[1]$, STag$[1])$, $(l_2$, Trap$[2]$, STag$[2]), \cdots)$
    $i = 1$, $R \leftarrow \{\}$, pk$_{\mathsf{rea}} = g^{\beta}$,sk$_{\mathsf{BS}} = (\eta, K_x, K_z, K_l)$
    tk $= (g^{r_3}, X \cdot g^{\beta \cdot r_3}, g^{-\alpha} \cdot H_1(X), g^{\beta/\alpha})$
    while $l_i \in$ EDB do
        $(e_0, e_1) \leftarrow$ EDB$[l_i]$, Trap$[i][j] = (T_{1,j}, T_{2,j})$
        $T_{\eta,j} = T_{2,j}^{\eta} / T_{1,j}^{\eta^2}$, $U_{\eta,j} = e(g^{\beta/\alpha \cdot e_1}, T_{\eta,j}^{-1})$
        STag$[i][j] = T_{\gamma,j}$, $U_{\gamma,j} = e(g^{\beta/\alpha \cdot e_1}, T_{\gamma,j})$
        if $U_{\gamma} U_{\eta} \in$ XSet for all $j$ then
            $e_0 \leftarrow (g^{r_1}, m \cdot e(g^{r_1}, H_1(g^{\alpha}))^{\alpha}$
            $c_1 = g^{r_1}$, $c_2 = m \cdot e(g^{r_1}, H_1(X))$
            $c_3 = (g^{r_3}, X \cdot g^{\beta \cdot r_3})$
            $e = (c_1, c_2, c_3)$
            Res $\leftarrow$ Res $\cup\, e$
        end if
        $i = i + 1$
    end while
    Send stop
    return Res
end function
```

V. Security Analysis

We first introduce leakage functions and then present three theorems to give security analysis of the DSB-SE scheme.

A. Leakage Analysis

We describe the leakage function in DSC-SE and use it to analyze security, where the security analysis of DSB-SE with same security guarantee can be easily obtained. With leakage profile $\mathbf{d}, \mathbf{q}$, the outputs of leakage $\mathcal{L}$ are:

- op is an array with the length is $|\mathsf{op}| = |\mathbf{d}| + |\mathbf{q}|$ that records each operation type, i.e., "encrypt" or "search".
- N is an array that records the total number of keywords in each document, i.e., the size of each EDB and XSet.
- $\bar{\mathbf{s}}$ is the *equality pattern* of the s-terms, indicating which queries have the same s-term. Roughly speaking, if $\mathbf{s} = (a, b, a, c, b)$, then we have $\bar{\mathbf{s}} = (1, 2, 1, 3, 2)$.
- $\mathsf{RP}[i, \alpha] = \mathsf{DB}[\mathbf{s}[i]] \cap \mathsf{DB}[\mathbf{x}[i, \alpha]]$. It reveals the indices in the intersection of the s-term and any xterm in the

same query. Furthermore, we use $\mathsf{RP}[i,\alpha,d]$ to denote the ind in $\mathsf{RP}[i,\alpha]$ that contained in $\mathbf{d}[d]$.

- $\mathsf{SRP}[i] = \mathsf{DB}[\mathbf{s}[i]]$ is the matching results of s-term of the i-th query. For supporting newly documents added to the outsourced database, the matching results of same s-terms in different queries may be different.
- $\mathsf{IP}[i_1,i_2,\alpha,\beta] = \begin{cases} \mathsf{DB}[\mathbf{s}[i_1]] \cap \mathsf{DB}[\mathbf{s}[i_2]] & (1) \\ \emptyset & (2) \end{cases}$. Here, $\mathsf{IP}[i_1,i_2,\alpha,\beta]$ holds the case (1) when $\mathbf{s}[i_1] \neq \mathbf{s}[i_2]$, $\mathbf{x}[i_1,\alpha] = \mathbf{x}[i_2,\beta]$, otherwise it is an empty set.
- $\mathsf{dRP}[i][j] == 1$ implies a $\mathsf{EDB}[l]$ that generated by $\mathbf{d}[i]$ is retrieved by the s-term $\mathbf{s}[j]$, otherwise it equals to 0.
- $\mathsf{xt[i]} = |\mathbf{x}[i,\cdot]|$ records the number of xterms in i-th query.

$\triangleright$ **Understanding leakage.** That is, op is directly to be leaked, since the storage server is unavoidable to know the type of each operation. N shows the size of each EDB and XSet, a simple way to prevent it is to add some random entries. The equality pattern indicates which queries have the same s-term, this is a consequence of taking inverted index to guarantee optimal search. RP captures the intersection of s-term and any xterm in a same query, SRP represents the results corresponding to any s-term and IP reveals the part results of intersection of each two s-term under a specified condition, we note that the leakage in RP,SRP and SRP is overstated as in [16] and [17] for designing security proof. The components dRP and xt are straightforward.

B. Security Analysis

We sketch a security analysis for the proposed scheme against non-adaptive attacks by designing a simulator, and then discuss the security against keyword guessing attacks.

Theorem 1: Our scheme is $\mathcal{L}$-semantically secure against non-adaptive attacks where $\mathcal{L}$ is the leakage function defined in Definition 5, assuming that the F and F_p are secure PRFs.

Proof: The non-adaptive means the adversary submits the completed encryption tuple list $\mathbf{d}$ and search tuple list $\mathbf{q}$ at the same time. Given the leakage function $\mathcal{L}(\mathbf{d},\mathbf{q}) = \{\mathsf{op}, N, \bar{\mathbf{s}}, \mathsf{RP}, \mathsf{SRP}, \mathsf{dRP}, \mathsf{IP}, \mathsf{xt}\}$, we construct a simulator as in the Algorithm 2.

$\triangleright$ **Understanding simulation in Algorithm 2.** Generally, we randomly select parameters from $\mathbb{G}$ and $\mathbb{Z}_p$ to replace the parameters in the real game $\mathbf{Real}^{\Pi}_{\mathcal{A}}(\lambda)$. If an adversary $\mathcal{A}$ can distinguish the random parameters in $\mathbf{Ideal}^{\Pi}_{\mathcal{A},\mathcal{S}}(\lambda)$ and used parameters in $\mathbf{Real}^{\Pi}_{\mathcal{A}}(\lambda)$, it can break DL and DDH assumptions. In particular, we use α' and β' to simulate a secret key of data writer and data reader. In the simulator $\mathcal{S}$, we introduce an array H_2 and an array H_3 to record trapdoor and xtag respectively, and use H_5 to simulate the hash function $H(w)$. In addition, the simulator simulates $\mathsf{EDBSetup}$ and $\mathsf{XsetSetup}$ functions to generate EDB and XSet. Note that the entries of EDB are filled with random tuples (e_0, e_1) and XSet is also filled with random elements.

With the knowledge of α' and β' generated in the $\mathsf{initialize}$ function, the simulator simulates the TKGen function. Roughly speaking, RP reveals indices of documents and SRP reveals the results associated with any s-term. Moreover, the simulator computes $\mathsf{ResInds}$ by using RP instead of decrypting the invalid ciphertexts in Res.

Hence, we have $\Pr[\mathbf{Real}^{\Pi}_{\mathcal{A}}(\lambda) = 1] - \Pr[\mathbf{Ideal}^{\Pi}_{\mathcal{A},\mathcal{S}}(\lambda) = 1] \leq \mathsf{Adv}^{\mathrm{DDH}}_{\mathbb{G},\mathcal{B}_1}(\lambda) + \mathsf{Adv}^{\mathrm{DDH}}_{\mathbb{G}_T,\mathcal{B}_2}(\lambda) + 2\cdot\mathsf{Adv}^{\mathrm{PRF}}_{F_p,\mathcal{B}_3}(\lambda)$ where the advantage of adversary $\mathcal{B}_1, \mathcal{B}_2$ breaking the DDH assumption in $\mathbb{G}$ and $\mathbb{G}_T$ is $\mathsf{Adv}^{\mathrm{DDH}}_{\mathbb{G},\mathcal{B}_1}(\lambda)$ and $\mathsf{Adv}^{\mathrm{DDH}}_{\mathbb{G}_T,\mathcal{B}_2}(\lambda)$ respectively, and the advantage of adversary $\mathcal{B}_3$ breaking the PRF F_p is $\mathsf{Adv}^{\mathrm{PRF}}_{F_p,\mathcal{B}_3}(\lambda)$. ■

Theorem 2: Our scheme is $\mathsf{IND\text{-}KGA}$ secure against keyword guessing attacks, if the DL assumption holds in $\mathbb{G}$.

Proof: To prove the theorem, we have to show that for any PPT adversary $\mathcal{A}$, the advantage functions $\mathsf{Adv}^{\mathsf{IND-KGA}}_{\mathsf{FS},\mathcal{A}}(\lambda)$ and $\mathsf{Adv}^{\mathsf{IND-KGA}}_{\mathsf{BS},\mathcal{A}}(\lambda)$ are negligible. Suppose given a trapdoor $(T_1, T_2) = (g^{r_2}, H(w)^{1/(\beta\cdot z_i)} \cdot (g^{\gamma} \cdot g^{\eta})^{r_2})$, what $\mathcal{A}$ knows are $(g, \gamma, g^{r_2}, g^{\beta}, g^{\eta})$. $\mathcal{A}$ guesses a keyword w' and gets $(H(w'), z_i')$, where $z_i = F_p(K_z, w||i)$ is related to w and we can only get $(H(w), z_i)$ under a keyword w. For $H(w)^{1/(\beta\cdot z_i)}\cdot(g^{\gamma}\cdot g^{\eta})^{r_2}$, it is hard to guess $H(w')^{1/(\beta\cdot z_i')}\cdot g^{\eta})^{r_2}$ due to DL assumption. Thus, we have $\mathsf{Adv}^{\mathsf{IND-KGA}}_{\mathsf{FS},\mathcal{A}}(\lambda) \leq \mathsf{negl}(\lambda)$. Similarly, we can easily aslo have $\mathsf{Adv}^{\mathsf{IND-KGA}}_{\mathsf{BS},\mathcal{A}}(\lambda) \leq \mathsf{negl}(\lambda)$. ■

Theorem 3: The search token is unforgeable in our scheme, if the DL assumption holds in $\mathbb{G}$.

Proof: As the β in a client's secret key is randomly selected, the adversary $\mathcal{A}$ in the game can win with a negligible advantage since it can not resolve β from g^{β} as long as DL assumption holds in $\mathbb{G}$. We can observe that no one can generate a valid search token beyond its identity, e.g., Bob generates a search token of Alice's, except that Bob can succeed to guess the β of Alice's. ■

VI. Performance Analysis

This section gives a general theoretical analysis and a detailed simulated experiment performance analysis between the DSB-SE and existing PEKS against KGA attacks.

A. Theoretical Analysis

The work [13], [14], and [15] encrypt the keywords and upload them to cloud, while the DSB-SE encrypts real documents. In front test phase, [14] has to generate $\mathsf{token}_{\mathsf{BS}}$ for all documents in database while ours only generates $\mathsf{token}_{\mathsf{BS}}$ according to $\mathsf{token}_{\mathsf{FS}}$ received from client, which reduces much time-consuming cost. In search phase, our system only searches documents associated with s-term while others have to search the whole database to check documents whether they conform to conditions. Moreover, the underlying hardness assumptions of our work are simple assumptions, such as DDH and DL.

As shown in Table I, the search complexity of DSB-SE achieves $\mathcal{O}(c_{w_1})$ with the least frequent keyword s-term. For a conjunctive query composed of 10 keywords, the systems [13], [14], [15] need to search over 2,025 documents (including 10∼15 keywords) with 10 times; while our DSB-SE only needs to search over c_{w_1} documents for 1 time. As depicted in Section VI-B.1, c_{w_1} (i.e. the number of s-term) is relatively

Algorithm 2 *Simulator*:

```
function initialize($\mathcal{L}(\mathbf{d}, \mathbf{q})$)
    for $w \in \bar{\mathbf{x}}$ and $ind \in \mathsf{RP} \cup \mathsf{IP}$ do
        $r \leftarrow_R \mathbb{Z}_p$ , $H_5[w] \leftarrow g^r$
    end for
    for $i \in \bar{\mathbf{s}}$ do
        $c_i$=0
    end for
    $d = q = 1$
    for  $i = 1$ to $|\mathsf{op}|$ do
        if $\mathsf{op}[i]$==encrypt then
            $\mathbf{t}[i] = \mathsf{EDBSetup}(\mathcal{L}(\mathbf{d}, \mathbf{q}))$, $d++$
        end if
        if $\mathsf{op}[i]$==search then
            $\mathbf{t}[i] = \mathsf{TrapGen}(\mathcal{L}(\mathbf{d}, \mathbf{q}))$, $q++$
        end if
    end for
    return t
end function

function EDBSetup($\mathcal{L}(\mathbf{d}, \mathbf{q})$)
    $j = 0$, Dup $\leftarrow \{\}$
    for  $\bar{\mathbf{s}}[i] \in \{\bar{\mathbf{s}}[q] \cdots \bar{\mathbf{s}}[|\bar{\mathbf{s}}|]\}$ and $\mathsf{dRP}[d][i] == 1$ and $\bar{\mathbf{s}}[i] \notin$
Dup do
        $c_{\bar{\mathbf{s}}[i]}++$, $r_1 \leftarrow \mathbb{Z}_p$, $z \leftarrow_R \mathbb{Z}_p$, $\mathsf{xind} \leftarrow_R \mathbb{Z}_p$
        $t \leftarrow_R \mathbb{Z}_p$, $t' \leftarrow \mathbb{Z}_p$
        $e_0 \leftarrow (g^{r_1}, 0^{\kappa} \cdot g^{\alpha' r_1})$, $e_1 = \mathsf{xind} \cdot z\alpha'$
        $H_1[\bar{\mathbf{s}}[i], c_{\bar{\mathbf{s}}[i]}] \leftarrow e_1$, $H_6[\bar{\mathbf{s}}[i], c_{\bar{\mathbf{s}}[i]}] \leftarrow (z, t)$
        $H_3[\bar{\mathbf{s}}[i], c_{\bar{\mathbf{s}}[i]}] \leftarrow e(g^{(\gamma - \eta)}, H_5[\bar{\mathbf{s}}[i]]^{\mathsf{xind}})$
        $l \leftarrow_R \{0, 1\}^*$, $\mathsf{EDB}[l] = (e_0, e_1)$ $l[\bar{\mathbf{s}}[i], c_{\bar{\mathbf{s}}[i]}] = l$
        Dup $\leftarrow \bar{\mathbf{s}}[i] \cup$ Dup, $j++$
    end for
    for $i = j + 1, \cdots, N[d]$  do
        $l \leftarrow_R \{0, 1\}^k$, $e_0 \leftarrow (r_1, r_2)$, $e_1 \leftarrow_R \mathbb{Z}_p$
        $\mathsf{EDB}[l] = (e_0, e_1)$
    end for
    XSet $\leftarrow$ XsetSetup($\mathcal{L}(\mathbf{d}, \mathbf{q})$)
    return (EDB, XSet)
end function
function XsetSetup$\mathcal{L}(\mathbf{d}, \mathbf{q})$
$j = 0$, XSet $\leftarrow \{\}$
for $\mathsf{RP}[t \geq q, \alpha, d] \neq \emptyset$
$ind \leftarrow \mathsf{RP}[t, \alpha, d]$, xtag $\leftarrow H_3[\bar{\mathbf{x}}[t, \alpha], ind]$
XSet $\leftarrow$ XSet $\cup$ xtag, $j++$
end for
for $i = j + 1, \cdots, N[d]$  do
    xtag $\leftarrow_R \mathbb{G}_T$, XSet $\leftarrow$ XSet $\cup$ xtag
end for
return XSet
end function

function TKGen($\mathcal{L}(\mathbf{d}, \mathbf{q})$)
    $r \leftarrow_R \mathbb{G}$,$X \leftarrow_R \mathbb{G}$
    $\mathsf{tk} = (r, X \cdot r^{\beta'}, g^{-\alpha'} \cdot H_1(X), g^{\beta'/\alpha'})$
    return tk
end function

function TrapGen($\mathcal{L}(\mathbf{d}, \mathbf{q})$)
    for $i = 1, \ldots, c_{\bar{\mathbf{s}}[q]}$ do
        $l_i = l[\bar{\mathbf{s}}[q], i]$
    end for
    $\mathbf{l} = \{l_i\}_{i=1}^{c_{\bar{\mathbf{s}}[q]}}$, $(ind_1, \ldots, ind_{c_{\bar{\mathbf{s}}[q]}}) \leftarrow \mathsf{SRP}[q]$
    for $\alpha \in [\mathsf{xt[q]}]$ do
        $R \leftarrow \mathsf{RP}[q, \alpha] \cup_{q' \in [|\bar{\mathbf{s}}|], \beta \in [\mathsf{xt[q']}]} \mathsf{IP}[q, q', \alpha, \beta]$
        for $c \in [c_{\bar{\mathbf{s}}[q]}]$ do
            if $ind_c \in R$ then
                $(z, t) \leftarrow H_6[\bar{\mathbf{x}}[q, \alpha], ind_c]$,$t' \leftarrow_R \mathbb{Z}_p$
                $H_2[\bar{\mathbf{x}}[q, \alpha], ind_c] \leftarrow (g^{t'}, (H_5[\bar{\mathbf{x}}[q, \alpha]])^{1/(\beta' \cdot z)} \cdot$
$g^{t'(\gamma+\eta)})$
                $\mathsf{Trap}[c, \alpha] \leftarrow H_2[\bar{\mathbf{x}}[q, \alpha], ind_c]$
            else
                if $\exists H_4[\bar{\mathbf{s}}[q], \bar{\mathbf{x}}[q, \alpha], c]$ then
                    $\mathsf{Trap}[c, \alpha] = H_4[\bar{\mathbf{s}}[q], \bar{\mathbf{x}}[q, \alpha], c]$
                else
                    $r_1, r_2 \leftarrow_R \mathbb{G}$, $\mathsf{Trap}[c, \alpha] \leftarrow (r_1, r_2)$
                    $H_4[\bar{\mathbf{s}}[q], \bar{\mathbf{x}}[q, \alpha], c] \leftarrow \mathsf{Trap}[c, \alpha]$
                end if
            end if
        end for
    end for
    $\mathsf{token_{FS}} \leftarrow (\mathbf{l}, \mathsf{Trap})$, $\mathsf{token_{BS}} \leftarrow \mathsf{FrontTest}(\mathsf{token_{FS}})$
    Res $\leftarrow$ Search($\mathsf{token_{BS}}$, tk)
    ResInds $\leftarrow \cap\ \mathsf{RP}[q, \alpha]$ for all $\alpha \in [\mathsf{xt[q]}]$
    return (token, Res, ResInds)
end function
```

small (roughly 1∼10). For c_{w_1} documents, each document is performed a match with a search token for 10 times (i.e. the number of keywords in a search query Q). In summary, the systems [13], [14], [15] need to search 202,500∼303,750 times, but our DSB-SE roughly need 10∼100 times. Nevertheless, the token size of DSB-SE is roughly c_{w_1} times bigger than PEKS solutions. We can say that the searching cost of our DSB-SE is more efficient than PEKS schemes with relatively comparable token size, while realizing boolean query.

B. Experiment Results Analysis

We conduct a number of experiments on state-of-art public-key SE schemes against KGA attacks for realizing practical conjunctive searching, which includes [13], [14], and [15] and our DSB-SE. Based on HUAWEI Cloud, a real client-to-server outsourcing cloud storage & searching environment is simulated for illustrating the practical utility. Moreover, a couple of careful and detailed observations on experiment results under a real-world dataset are formally concluded.

1) Experimental Bed-up and Data Preprocessing: The HUAWEI Cloud platform is conducted on an Ubuntu 18.04 system with an Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40GHz and 8.00GB RAM, while the client side is deployed on an Ubuntu 18.04 system with an Intel(R) Core(TM) i5-6200U CPU @ 2.30GHz and 4.00GB RAM. The implementation codes of Fuzzy-PEKS [13], DS-PEKS [14],

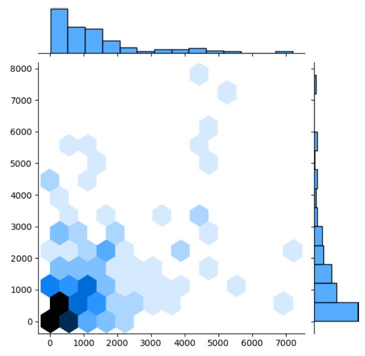
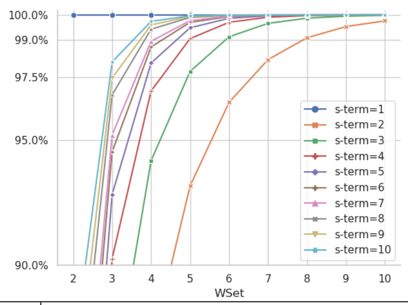


Fig. 2. The file entries distribution in the enron dataset.



s-term	10	20	30	40	49
|WSet| = 1	13.7%	5.25%	2.56%	1.60%	1.09%
|WSet| = 2	1.88%	0.28%	0.66%	0.26%	0.12%
|WSet| = 3	0.25%	0.15%	0.15%	0.04‰	0.01‰

Fig. 3. Documents Coverage with the value of WSet and s-term.

SA-PEKS [15] and our DSB-SE are implemented with Python 3 language by using Pypbc 0.2 library. We employ SHA-256 as the underlying hash function and curve $y^2 = x^3 + x$ for Type-A pairings, while parameters are q-bits=512 and r-bits=160. In addition, an AES-CBC modular is employed to encrypt files whose key is 256 bits and Initialization Vector is 128 bits.

A well-known representative real-world Enron Email Dataset is taken into consideration. Through a careful observation on the distribution of Enron dataset in Fig. 2, the majority file numbers in MAILDIR are around 2,000, and hence select "MAILDIR/CORMAN-S" including 2,025 documents from Enron dataset to encrypt with AES.

At the same time, the s-term serves as a main factor that influences system performance and a special keyword that is associated with the smallest set of documents. Actually, the range of s-term in practical applications is relatively small (roughly 1~10) under a conjunctive/boolean query formula, since a large s-term may lead to less than 1% and even no target files of documents searching. To obtain a practical and efficient range for s-term, we show a searching document cover rate with the changes of WSet and s-term in Fig 3. We find that when WSet=8, the rate exceeds 99% despite s-term value. In addition, for s-term=10, the rate exceeds 97.5%. As a result, we could pay less attention to the condition that s-term>10 and WSet>3. Moreover, considering people are incline to search target emails with "email address" and "subject" as keywords (instead of email body) in practice, we choose to use email address as s-term and subject as other keywords.

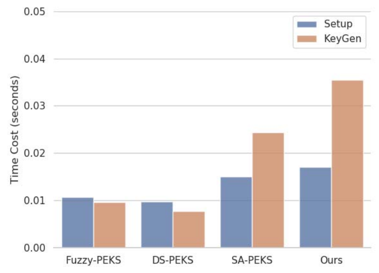
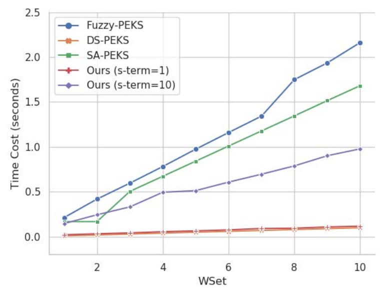
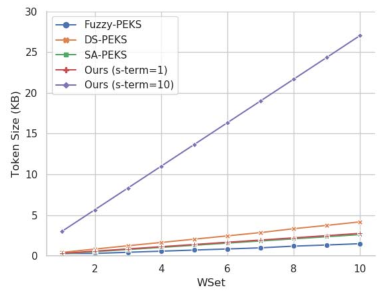


Fig. 4. Time cost of system setup phase.



(a) time cost



(b) storage cost

Fig. 5. Time and storage cost of trapdoor generation phase.

To well study and understand the experiments about this work, we illustrate the results in the following perspectives: dataset statistic, document encryption phase, setup phase, key generation phase, trapdoor generation and search efficiency.

2) System Initialization: The system initialization includes the Setup and Key Generation algorithm. The setup phase initializes the system by generating public parameters, while the key generation phase produces public key and secret key for servers and clients. Concretely, from the time-cost distribution in Fig. 4, the time-cost performance is almost same (relatively small) in different systems, whose differences of order of magnitude are 10^{-2} seconds.

In trapdoor generation phase, Fig. 5(a) and Fig. 5(b) show the time and storage cost distribution, in which we find that our system yields high time efficiency than others for s-term=1. Even though s-term=10, our system is still more efficient than the work [13] and [15] as shown in Fig. 5(a). As can be seen from Fig. 5(b), our system yields high storage efficiency whose size is around 2KB for s-term=1. For s-term=10, the token takes up storage a lot with comparison to other systems, nearly about 4KB~27KB while others are no more than 5KB.

TABLE III
THE TIME COST OF DOCUMENTS ENCRYPTION

Scheme	Time cost	Description
Fuzzy-PEKS [13]	7522.351s	Encrypt keywords only†
DS-PEKS [14]	350.719s	
SA-PEKS [15]	5225.454s	
Ours	553.864s	Encrypt real documents‡

† : encryption only includes (ind, w).
‡ : encryption includes both (ind, w) and every real document Doc.

TABLE IV
THE SIZE OF GENERATED FILES AT DOCUMENTS ENCRYPTION PHASE

File Name	File Size	Description
EDB.dat	16MB	For all documents†
Xset.dat	9.2MB	
Res.dat	0.658KB	For one target document‡
Files_Enc.dat	3.468KB	

† : for 2,025 documents in the Enron dataset.
‡ : for one target document that satisfies an issued query pattern.

In conclusion, the performance of setup and key generation is efficient. The performance of trapdoor generation is efficient for s-term=1, it still remains high efficiency for s-term=10; while the performance of token storage is efficient for s-term=1, but little expensive than other systems for s-term=10.

3) Documents Tuples Encryption: To evaluate the document encryption performance, we choose mode from Enron Email dataset and encrypt these files with AES. Each document is indexed with 10∼15 keywords that involve email address and subject. Concretely, Table III shows that the cost of our system remains high efficient compared with Fuzzy-PEKS [13] and SA-PEKS [15] though we encrypt actual documents and keywords while others only encrypt keywords.

In the experiment, the data writer encrypts all 2,025 documents and accordingly generates EDB.dat and Xset.dat. The EDB.dat stores s-term, document index ind, corresponding key K_{ind} and others (e.g., file link info) for all documents. The Xset.dat stores encrypted keywords xterm for all documents. From Table IV, we can observe the size of encrypted files. The search result file Res.dat is a set of $\mathsf{Enc}(ind||K_{ind})$, where a data reader can recover an encrypted document with K_{ind}. If only one document satisfies the issued query pattern, the returned result file Res.dat and the associated encrypted document Files_Enc.dat are 0.658 KB and 3.468 KB respectively. In addition, the cloud maintains s-term set in EDB.dat and keywords set in Xset.dat with 16 MB and 9.2 MB sizes respectively for all documents.

We may conclude that although actual documents are encrypted in our system, it still enjoys high efficiency compared to those work under encrypting only keywords. In addition, the sizes of encrypted documents seems no additional cost for the next-step encrypted documents retrieval.

4) Documents Retrieval: During searching, the cloud uses the received trapdoor from a data reader to search indices of documents in cloud, as depicted in Table V, our documents searching time cost is much more lightweight than others. For s-term=1 and #KWD=10, the time cost of DSB-SE is about 6,600 times faster than [13], [15] and even 25,000 times faster than [14]. For s-term=10 and #KWD=10, DSB-SE runs 1200 times faster than [13] and [15].

The KGA-resistance secure PEKS schemes [13], [14], [15] only support single keyword search, but they can be extended to deal with conjunctive queries with repeatedly running the underlying search algorithm multiple times. Here, we take the extensions of [13], [14], and [15] that support conjunctive queries as a comparison with our boolean DSB-SE. Although the time complexity of processing boolean queries is the same as that of conjunctive queries, we have still implemented our scheme that supports boolean queries in the experiment. We note that the shown efficiency of their schemes may decrease for further extending to process boolean queries. In particular, Table V shows a document searching efficiency comparison between our boolean DSB-SE and conjunctive PEKS schemes.

To observe the influence of s-term, we illustrate it by changing different s-term in the experiment and show it in Fig 6. For s-term=1, we can see the searching time is no more than 0.2s. Even under the worst case (i.e. s-term=10 and #KWD=10), the consumed time is only 1.2s. The searching time increases along with the increment of s-term and #KWD. In addition, there is a growing time trend with mounting s-term and WSet, but we do not consider it in further due to the practical documents coverage in Fig. 3.

Hence, s-term plays an important role in improving searching efficiency, which locates the least set of document that conform to search keyword #KWD. Moreover, the inverted index data structure is employed in the DSB-SE system.

5) Client-to-Cloud Communication: To evaluate the performance of the simulated Client-to-Cloud environment, we record the time of client sending token, retrieving inds of document, receiving document and decrypting document. Figure 7(a) shows the retrieve time, along with the increasing of querying WSet, we can see that the retrieve time decrease and converge to $0.015s$. Figure 7(b) shows the time difference between client sending request and server returning result. Most of receiving time are around 1s. Figure 7(c) shows client decryption time, and decryption is efficient which takes only 0.01s. As depicted in Figure 8, the time of sending token are about 0.1s which can be neglected in communication.

Hence, we conclude that along with the increasing of querying WSet, the time on retrieving encrypted indices, receiving encrypted documents and documents decryption converge to a fixed value respectively. This observation conforms to reality that along with search conditions increasing, the number of corresponding results are decreasing.

C. Summary

Previous solutions [13], [14], [15] need to perform a match between keywords and every document, which leads to high

TABLE V
THE DOCUMENTS SEARCHING TIME COST COMPARISON

Time(s) / #KWD / Work		1	2	3	4	5	6	7	8	9	10
Fuzzy-PEKS [13]		117.33	235.06	352.23	469.78	588.18	703.92	824.61	940.32	1058.26	1174.15
DS-PEKS [14]		456.83	913.78	1374.25	1834.26	2292.64	2756.77	3216.51	3674.73	4126.27	4587.59
SA-PEKS [15]		119.86	239.72	360.04	479.85	600.20	719.36	840.29	958.90	1080.26	1200.41
Ours	s-term=1	0.0207	0.0309	0.0573	0.0700	0.0940	0.0997	0.1249	0.1301	0.1389	0.1776
	s-term=10	0.1479	0.1555	0.2751	0.3679	0.4966	0.6004	0.7006	0.8244	0.9336	1.0576

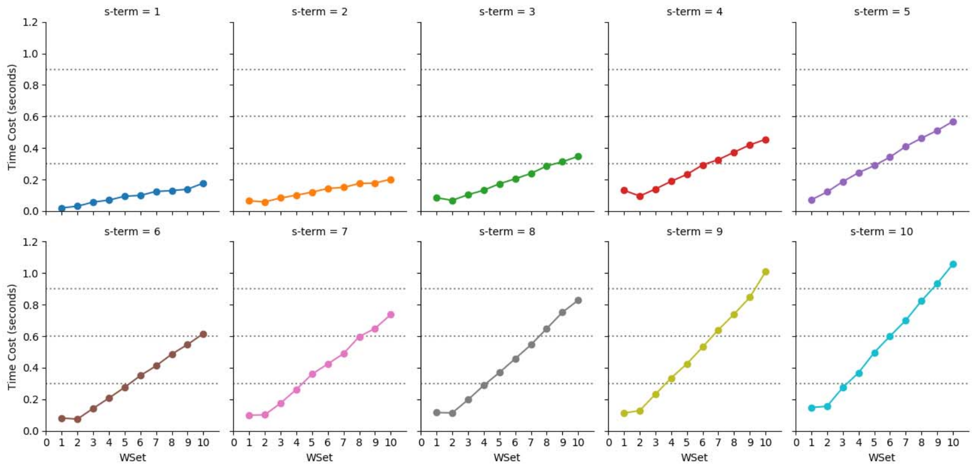
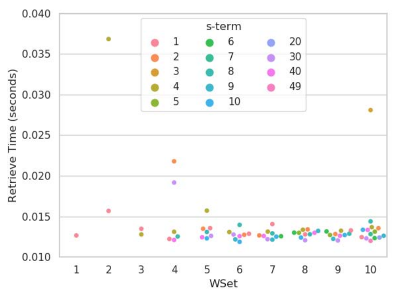
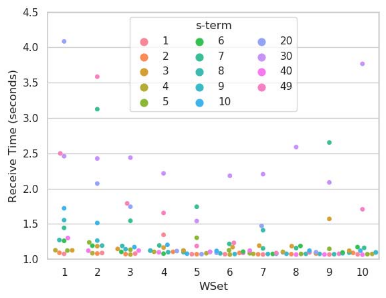


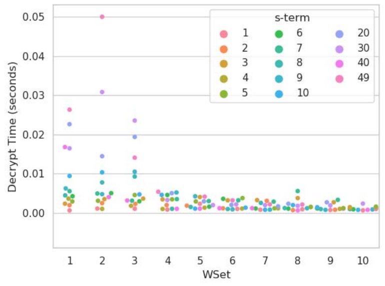
Fig. 6. Time cost of different s-term ranging from 1 to 10 towards different WSet=1∼10.



(a) Encrypted Indexes Retrival



(b) Encrypted Dcouments Receiving



(c) Encrypted Documents Decryption

Fig. 7. Time cost for retrieving encrypted file index, receiving/decryption encrypted files *where only one document meets the query expressions*.

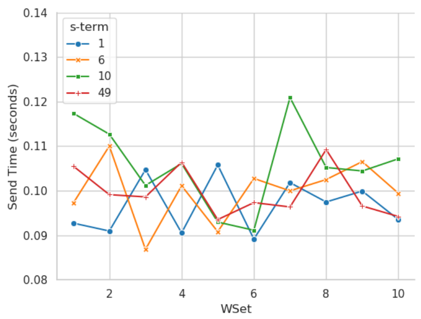


Fig. 8. Time cost for sending a token from a data reader to cloud server.

searching time cost. Although they resist KGA attacks, only single keyword search is considered. When to deal with a conjunctive query, the search algorithm should be repeatedly performed multiple times. However, our DSB-SE enjoys better time efficiency in both document encryption and search. The main factors that influence the performance are the s-term (the least frequent keyword in a query) and the number of keywords in a query Q. Hence, DSB-SE only needs to perform a match between keywords and every document in a small amount of documents. Furthermore, DSB-SE supports conjunctive and boolean queries, which avoids the search algorithm being run repeatedly.

VII. RELATED WORK AND DISCUSSION

1) PEKS-KGA: Byun et al. [21] firstly considered and proposed offline KGA attacks for PEKS [5]. To resist KGA attacks, Xu et al. [13] proposed a Fuzzy-PEKS scheme. In [13], the malicious server only obtains the fuzzy search trapdoor instead of the exact search trapdoor. However, this leads to additional communication overhead since there are some unmatched files returned. Moreover, Huang and Li [22] proposed a novel notion of public-key authenticated encryption to capture offline KGA attacks. Subsequently, Li et al. [23] proposed a scheme that resisted offline KGA attacks by authenticating cipher-keyword, thus an adversary cannot forge a valid encrypted keyword. To address inside KGA attacks, Chen et al. [14], [15] gave a DS-PEKS and SA-PEKS framework against inside KGA attacks by malicious servers. In DS-PEKS [14], the server is separated into two independent parts: a front server (FS) and a back server (BS). A ciphertext CT_{w_1} and a trapdoor T_{w_2} are generated under both public key of FS and BS, where FS generates an internal testing-state C_{ITS} with its secret key and sends C_{ITS} to BS. Later, BS uses C_{ITS} and its secret key to perform an equality test on w_1 and w_2 that involved in CT_{w_1} and T_{w_2}. In SA-PEKS [15], a semi-honest keyword server (KS) is introduced, where a client first needs to interact with the KS to obtain KS-derived keyword ksd_w. Roughly speaking, the ksd_w simulates keyword w in the search phase, thus a storage server (SS) cannot launch KGA attacks on ksd_w without the secret key of KS. In overall, both DS-PEKS and SA-PEKS separate server rights into two independent parts to resist inside KGA attacks, where the difference is that DS-PEKS focuses on keeping privacy of search trapdoor while SA-PEKS focuses on keeping privacy of original keyword. Besides, a framework of KGA-secure verifiable SE [24] was presented for supporting a variety of dynamic data updates including data modification, insertion, and deletion. Very recently, Noroozi et al. [25] employed a ciphertext re-randomization technique to propose a generic construction that resists online and offline KGA attacks. Li et al. [26] also considered the property of KGA-resistance to further support fine-grained access control with the employment of attribute-based encryption.

We may conclude that these works mainly focus on basic single keyword search against offline or online KGA attacks, but fail to consider more expressive query models. Hence, this may bring about large search cost and communication cost.

2) SSE With Dynamic Updates: SSE enables a client to outsource an encrypted database to a remote server and later search over the encrypted database with a search token. In a seminal work [16], Cash et al. proposed the technique of Oblivious Cross Tags (OXT) that supports sub-linear boolean queries for highly-scalable SSE, where a tradeoff between information leakage and search efficiency is well conducted. Since then, numerous SSE schemes [17], [18], [19] have been proposed for concerning different practical scenarios. To achieve dynamic updates for SSE, Karama et al. [6] proposed a dynamic SSE (DSSE) in which the outsourced database can be updated (i.e., data addition and deletion). Moreover, the works [27], [28] additionally achieve boolean queries and access control over documents without per-query interaction between the data owner and each client. Nevertheless, the update operation leads to the leakage of query or data privacy, which motivates the security property of forward privacy and backward privacy. Zeng et al. [29] proposed a forward secure SE system that binds a search token and its generation time together and checks whether the encrypted data is generated before the search token. In addition, some forward and backward secure SE schemes were studied in a line of works [30], [31], [32]. Nevertheless, previous schemes mainly focused on single-writer/multi-reader setting, which limits its employments in some practical applications, i.e., a reader let gateway search over encrypted emails from writers. To address the problem, Sharma et al. [33] presented a multi-writer/multi-reader SE scheme and Xu et al. [35] presented a multi-writer SE with aggregated keywords search that supports subset encrypted data validation. Very recently, Wang et al. [34] proposed a new paradigm of searchable encryption (i.e., hybrid SE), which not only supports sub-linear boolean queries but also guarantees forward privacy for multi-writer SE with dynamic updates.

In summary, there are numerous advanced features are studied in existing works, such as boolean queries, data sharing, forward and backward security or multi-writer setting. However, resisting KGA attacks has not been formally considered in the context. This work targets enabling boolean SE schemes for data sharing to resist KGA attacks.

VIII. Conclusion and Future Research

In this work, we have presented a dual-server searchable encryption scheme supporting boolean queries for secure cloud storage and file-sharing services, and prove its security under the simulation-based security model. Nevertheless, this work mainly focuses on the data sharing situation of multiple-writer/multiple-reader (single-writer/single-reader) in an interactive way. It seems an interesting work to enhance this work to support multiple-writer/multiple-reader in a non-interactive way by using identity-based encryption or attribute-based encryption. In addition, we may also borrow the technique presented in hybrid SE to enhance our scheme to a forward and backward secure SE scheme for multi-writer SSE with dynamic updates.

References

[1] *State of Cloud Security 2021: More Aware Yet Very Exposed*. [Online]. Available: https://bit.ly/2MmZkDt

[2] S. Kamara and K. Lauter, "Cryptographic cloud storage," in *Proc. Int. Conf. Financial Cryptography Data Secur.* Berlin, Germany: Springer, 2010, pp. 136–149.

[3] C. Ge, W. Susilo, Z. Liu, J. Xia, P. Szalachowski, and L. Fang, "Secure keyword search and data sharing mechanism for cloud computing," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 6, pp. 2787–2800, Dec. 2020.

[4] D. Xiaoding Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proc. IEEE Symp. Secur. Privacy. (S&P)*, May 2000, pp. 44–55.

[5] D. Boneh, G. D. Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *Advances in Cryptology*. Berlin, Germany: Springer, 2004, pp. 506–522.

[6] S. Kamara, C. Papamanthou, and T. Roeder, "Dynamic searchable symmetric encryption," in *Proc. ACM Conf. Comput. Commun. Secur. (CCS)*, 2012, pp. 965–976.

[7] K. Xue, W. Chen, W. Li, J. Hong, and P. Hong, "Combining data owner-side and cloud-side access control for encrypted cloud storage," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 8, pp. 2062–2074, Aug. 2018.

[8] J. Zhu, Q. Li, C. Wang, X. Yuan, Q. Wang, and K. Ren, "Enabling generic, verifiable, and secure data search in cloud services," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 8, pp. 1721–1735, Aug. 2018.

[9] K. Liang, X. Huang, F. Guo, and J. K. Liu, "Privacy-preserving and regular language search over encrypted cloud data," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 10, pp. 2365–2376, Oct. 2016.

[10] D. Boneh and B. Waters, "Conjunctive, subset, and range queries on encrypted data," in *Proc. Theory cryptography Conf.* Berlin, Germany: Springer, 2007, pp. 535–554.

[11] C. Bösch, P. Hartel, W. Jonker, and A. Peter, "A survey of provably secure searchable encryption," *ACM Comput. Surveys*, vol. 47, no. 2, pp. 1–51, Jan. 2014.

[12] I. R. Jeong, J. O. Kwon, D. Hong, and D. H. Lee, "Constructing PEKS schemes secure against keyword guessing attacks is possible?" *Comput. Commun.*, vol. 32, no. 2, pp. 394–396, Feb. 2009.

[13] P. Xu, H. Jin, Q. Wu, and W. Wang, "Public-key encryption with fuzzy keyword search: A provably secure scheme under keyword guessing attack," *IEEE Trans. Comput.*, vol. 62, no. 11, pp. 2266–2277, Nov. 2012.

[14] R. Chen, Y. Mu, G. Yang, F. Guo, and X. Wang, "Dual-server public-key encryption with keyword search for secure cloud storage," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 4, pp. 789–798, Apr. 2015.

[15] R. Chen et al., "Server-aided public key encryption with keyword search," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 12, pp. 2833–2842, Dec. 2016.

[16] D. Cash, S. Jarecki, C. S. Jutla, H. Krawczyk, M. Rosu, and M. Steiner, "Highly-scalable searchable symmetric encryption with support for Boolean queries," in *Advances in Cryptology*. Berlin, Germany: Springer, 2013, pp. 353–373.

[17] S. Sun, J. K. Liu, A. Sakzad, R. Steinfeld, and T. H. Yuen, "An efficient non-interactive multi-client searchable encryption with support for Boolean queries," in *Computer Security*. Cham, Switzerland: Springer, 2016, pp. 154–172.

[18] M. Zeng, K. Zhang, H. Qian, X. Chen, and J. Chen, "A searchable asymmetric encryption scheme with support for Boolean queries for cloud applications," *Comput. J.*, vol. 62, no. 4, pp. 563–578, Apr. 2019.

[19] K. Zhang, M. Wen, R. Lu, and K. Chen, "Multi-client sub-linear Boolean keyword searching for encrypted cloud storage with owner-enforced authorization," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 6, pp. 2875–2887, Nov. 2021.

[20] S. Jarecki, C. Jutla, H. Krawczyk, M. Rosu, and M. Steiner, "Outsourced symmetric private information retrieval," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, 2013, pp. 875–888.

[21] J. W. Byun, H. S. Rhee, H.-A. Park, and D. H. Lee, "Off-line keyword guessing attacks on recent keyword search schemes over encrypted data," in *Proc. Workshop Secure Data Manage.* Berlin, Germany: Springer, 2006, pp. 75–83.

[22] Q. Huang and H. Li, "An efficient public-key searchable encryption scheme secure against inside keyword guessing attacks," *Inf. Sci.*, vol. 403, pp. 1–14, Sep. 2017.

[23] H. Li, Q. Huang, J. Shen, G. Yang, and W. Susilo, "Designated-server identity-based authenticated encryption with keyword search for encrypted emails," *Inf. Sci.*, vol. 481, pp. 330–343, May 2019.

[24] Y. Miao, Q. Tong, R. H. Deng, K.-K.-R. Choo, X. Liu, and H. Li, "Verifiable searchable encryption framework against insider keyword-guessing attack in cloud storage," *IEEE Trans. Cloud Comput.*, vol. 10, no. 2, pp. 835–848, Apr. 2022.

[25] M. Noroozi and Z. Eslami, "Public-key encryption with keyword search: A generic construction secure against online and offline keyword guessing attacks," *J. Ambient Intell. Humanized Comput.*, vol. 11, no. 2, pp. 879–890, Feb. 2020.

[26] J. Li, M. Wang, Y. Lu, Y. Zhang, and H. Wang, "ABKS-SKGA: Attribute-based keyword search secure against keyword guessing attack," *Comput. Standards Interfaces*, vol. 74, Feb. 2021, Art. no. 103471.

[27] L. Du, K. Li, Q. Liu, Z. Wu, and S. Zhang, "Dynamic multi-client searchable symmetric encryption with support for Boolean queries," *Inf. Sci.*, vol. 506, pp. 234–257, Jan. 2020.

[28] L. Sun, C. Xu, and Y. Zhang, "A dynamic and non-interactive Boolean searchable symmetric encryption in multi-client setting," *J. Inf. Secur. Appl.*, vol. 40, pp. 145–155, Jun. 2018.

[29] M. Zeng, H. Qian, J. Chen, and K. Zhang, "Forward secure public key encryption with keyword search for outsourced cloud storage," *IEEE Trans. Cloud Comput.*, vol. 10, no. 1, pp. 426–438, Jan. 2019.

[30] C. Zuo, S.-F. Sun, J. K. Liu, J. Shao, J. Pieprzyk, and L. Xu, "Forward and backward private DSSE for range queries," *IEEE Trans. Dependable Secure Comput.*, vol. 19, no. 1, pp. 328–338, Jan. 2022.

[31] S. Sun et al., "Practical non-interactive searchable encryption with forward and backward privacy," in *Proc. 28th Annu. Netw. Distrib. Syst. Secur. Symp. (NDSS)*, 2021, pp. 1–18, doi: 10.14722/ndss.2021.24162.

[32] J. Wang and S. S. M. Chow. (2019). *Forward and Backward-Secure Range-Searchable Symmetric Encryption*. Cryptology ePrint Archive. [Online]. Available: https://eprint.iacr.org/2019/497

[33] D. Sharma and D. C. Jinwala, "Multi-writer multi-reader conjunctive keyword searchable encryption," *Int. J. Inf. Comput. Secur.*, vol. 15, nos. 2–3, pp. 141–162, 2021.

[34] J. Wang and S. S. M. Chow, "Omnes pro uno: Practical multi-writer encrypted database," in *Proc. 31st USENIX Secur. Symp. (USENIX Security)*, Aug. 2022, pp. 2371–2388.

[35] L. Xu, C. Xu, J. Liu, B. Dou, and X. Jin, "Enabling privacy-preserving data validation from multi-writer encryption with aggregated keywords search," *Wireless Netw.*, 2022, doi: 10.1007/s11276-022-03117-3.

Kai Zhang received the bachelor's degree in computer science and technology from Shandong Normal University, China, in 2012, and the Ph.D. degree in computer science and technology from East China Normal University, China, in 2017. He visited Nanyang Technological University in 2017. He is currently an Associate Professor with the Shanghai University of Electric Power, China. His research interests include applied cryptography and information security.

Xiwen Wang received the bachelor's degree from the Shanghai University of Electric Power, China, in 2021, where he is currently pursuing the master's degree with the College of Computer Science and Technology. His research interests include cloud security and applied cryptography.

Jianting Ning (Member, IEEE) received the Ph.D. degree from the Department of Computer Science and Engineering, Shanghai Jiao Tong University, in 2016. He is currently a Professor with the Fujian Provincial Key Laboratory of Network Security and Cryptology, College of Computer and Cyber Security, Fujian Normal University, China. Previously, he was a Research Scientist at the School of Computing and Information Systems, Singapore Management University, and a Research Fellow at the Department of Computer Science, National University of Singapore. He has published papers in major conferences/journals, such as ACM CCS, NDSS, ASIACRYPT, ESORICS, ACSAC, IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, and IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING. His research interests include applied cryptography and information security.

Xinyi Huang received the Ph.D. degree from the School of Computer Science and Software Engineering, University of Wollongong, Australia, in 2009. He is currently an Associate Professor at the Thrust of Artificial Intelligence, Information Hub, Hong Kong University of Science and Technology (Guangzhou), China. His research interests include cryptography and information security. He has published over 160 research papers in refereed international conferences and journals, such as ACM CCS, Crypto, Asiacrypt, IEEE TRANSACTIONS ON COMPUTERS, IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, and IEEE TRANSACTIONS ON INFORMATION SECURITY AND FORENSICS. His work has been cited more than 10000 times at Google Scholar. He is in the Editorial Board of *International Journal of Information Security* and *Science China Information Sciences*. He has served as the program chair/the general chair or a program committee member in over 120 international conferences.