Privacy-Preserving Multi-Keyword Searchable Encryption for Distributed Systems

Xueqiao Liu, Guomin Yang*, *Senior Member, IEEE,* Willy Susilo, *Senior Member, IEEE,* Joseph Tonien, Ximeng Liu, *Member, IEEE,* and Jian Shen

Abstract—As cloud storage has been widely adopted in various applications, how to protect data privacy while allowing efficient data search and retrieval in a distributed environment remains a challenging research problem. Existing searchable encryption schemes are still inadequate on desired functionality and security/privacy perspectives. Specifically, supporting multi-keyword search under the multi-user setting, hiding search pattern and access pattern, and resisting keyword guessing attacks (KGA) are the most challenging tasks. In this paper, we present a new searchable encryption scheme that addresses the above problems simultaneously, which makes it practical to be adopted in distributed systems. It not only enables multi-keyword search over encrypted data under a *multi-writer/multi-reader* setting but also guarantees the data and search pattern privacy. To prevent KGA, our scheme adopts a multi-server architecture, which accelerates search response, shares the workload, and lowers the key leakage risk by allowing only authorized servers to jointly test whether a search token matches a stored ciphertext. A novel subset decision mechanism is also designed as the core technique underlying our scheme and can be further used in applications other than keyword search. Finally, we prove the security and evaluate the computational and communication efficiency of our scheme to demonstrate its practicality.

Index Terms—Searchable Encryption, Multi-Keyword Search, Multi-User Access, Search Pattern, Access Pattern.

1 Introduction

Since the emergence of cloud computing, cloud storage has become one of the most popular and essential cloud services for both industrial and personal users due to its appealing advantages in comparison to the traditional data storage. According to the forecast from the statistics portal website *statista*, the data center storage capacity worldwide will stand at 2,300 exabytes by 2021 [1]. With such a rapid growth in cloud storage, data security and privacy are indispensable considerations that must be well-addressed to avoid monetary loss or damage of reputation due to cloud data leaks. Hence, it is natural to apply cryptographic approaches such as data encryption mechanisms to ensure the privacy of sensitive information stored in the cloud. Nevertheless, such a straightforward privacy protection mechanism does not work for cloud storage facilities with considerable capacity since it disallows the cloud server to perform a quick search over the stored data based on the user request. To resolve this problem, searchable encryption schemes have been introduced in the literature.

In the seminal work by Boneh et al. [2], the notion of Public-key Encryption with Keyword Search (PEKS) was introduced. In a PEKS scheme, it is assumed that there are three entities: a data owner (or writer), a data user (or reader) and a storage server. To share data to the user via the storage server, the owner first extracts a keyword from the data and then generates the encryption of the keyword (called a searchable ciphertext) with the intended user's public key. The actual data, which can be encrypted separately, is submitted together with its searchable ciphertext to the server. Then only the intended user can generate a search token (a.k.a. trapdoor) based on his/her private key and a keyword of interest and then passes the token to the server, who will test whether the trapdoor matches a searchable ciphertext and inform the search result to the user. In this model, if the owner wants to share the same data to different users, it needs to repeat the above operation and generate multiple searchable ciphertexts, which is not practical or scalable for distributed environments. For instance, if an administrative staff of a large corporation with ten thousand employees uploads a regulation document to their subscribed cloud server, it would require ten thousand searchable ciphertexts to be generated, which results in huge computation and storage overhead. Thus, an efficient searchable encryption scheme supporting multi-user access is more desirable for multi-user environments, where a searchable ciphertext can match trapdoors from different authorized users.

Similar to the demand for multi-user search, multi-keyword search is another desirable feature of searchable encryption. For a data document with multiple keywords, the plain PEKS scheme demands the same number of searchable ciphertexts to be generated. Moreover, given a set of trapdoors for multiple searching keywords, each trapdoor needs to be repeatedly tested against all the searchable ciphertexts associated with a document. Hence, a more efficient searchable encryption supporting multi-keyword search is also desirable.

In a secure cloud storage supporting keyword search,

- *Xueqiao Liu, Guomin Yang, Willy Susilo and Joseph Tonien are with the Institute of Cybersecurity and Cryptology, University of Wollongong, Australia, 2522. Email: {xl691,gyang,wsusilo,dong}@uow.edu.au.*
- *Ximeng Liu is with the College of Mathematics and Computer Science, Fuzhou University, Fuzhou, China, 350002. Email: snbnix@gmail.com.*
- *Jian Shen is with the School of Computer and Software Nanjing University of Information Science and Technology Nanjing, China, 210044. Email: s_shenjian@126.com.*
- ** Corresponding author.*

Manuscript received ××, × × ××; revised ××, × × ××.

security concerns involve not only data privacy, but search query privacy as well. Unfortunately, the PEKS scheme by Boneh et al. [2] cannot guarantee the search query privacy against offline keyword guessing attacks (KGAs) [3] since whether a trapdoor matches a searchable ciphertext can be checked by anyone (including the cloud server). Thus, making searchable encryption schemes immune to KGA is essential to protect user privacy.

Different solutions have been proposed to resist KGA attacks. There are generally two means to resist KGA: to make the server unable to generate the searchable ciphertext by itself and then launch KGA; and to disable public testing. The first method has led to new cryptographic primitives such as public-key authenticated encryption with keyword search (PAEKS) [4] where private key of the data owner is used to generate and authenticate the searchable ciphertext. However, PAEKS also takes the public key of the data user as input to generate the searchable ciphertext, thereby failing to support multi-user search.

The second approach, namely disabling public testing, inevitably requires a secret to be used in the testing algorithm. Since the testing is performed by the storage server, if the secret is known to the server, then it can still perform KGA without being detected. Therefore, a *distributed* testing mechanism becomes necessary to reduce the trust on a single server. Specifically, the secret used for testing can be split into two (or multiple) shares, one kept by the public cloud storage server and the other kept by an internal server of an organisation. The job of the internal server is to cooperate with the cloud storage server in performing the searching operation while preventing the latter from performing KGA. Such an approach can also be extended to a multi-internal-server setting. For example, two internal servers residing in two departments/branches/clusters can be deployed so that each internal server can handle search queries from one department. In such a distributed environment, one important requirement is to allow multi-reader and multi-writer in data uploading and retrieval. It means we should allow the encrypted document uploaded by a data owner (or writer) to be searched by multiple data users (or readers), and vise versa.

Lastly, it is also important to achieve search pattern privacy, which means the cloud storage server can't identify the matching documents corresponding to a search query (i.e., the cloud storage can't tell whether two search queries produce the same or different results, even if they are made by the same user).

1.1 Our Contributions

In this work, we present a new public-key searchable encryption scheme that can address the above security, privacy and functionality issues. Our scheme is suitable for a distributed environment which comprises multiple data writers and readers and can deploy multiple designated servers to assist the public cloud storage server to perform privacy-preserving keyword search over encrypted data. Our solution is called **S**earchable **E**ncryption based on **E**fficient **P**rivacy-preserving **O**utsourced calculation framework with **M**ultiple keys (SE-EPOM). The contributions of our work are three-fold:

- We design a new subset decision mechanism to determine whether one input set is the subset of the other input set. The proposed mechanism provides the basis for enabling multi-keyword search in a two-sever architecture. It may also be applied to other applications that require private subset testing.
- We present an SE-EPOM scheme based on the above subset decision mechanism. The proposed scheme has the merits of supporting multi-user access, supporting multi-keyword search and achieving data and search query privacy. Specifically, different from existing works, our multi-user access refers to accommodating both multiple writers (or data owners) and multiple readers (or data users) simultaneously, which is important for adoption in a distributed system. Also, by applying the multi-server architecture in the searching/testing operation, search queries are handled with the assistance from multiple parallel servers to accelerate the response and balance the workload, at the same time Keyword Guessing Attack from the cloud storage server is effectively resisted. Moreover, the trapdoor and the searchable ciphertext are delicately designed to achieve constant size. A comparison between our scheme and the existing ones is presented in Table 3.
- We evaluate the computational and communication overhead of our scheme and two other keyword search schemes. The experimental results demonstrate that our scheme is practical and more advantageous than the compared ones.

1.2 Related Work

After the concept of Searchable Encryption (SE) was put forth in [5], it was divided into two categories, Searchable Symmetric Encryption (SSE) and Public key Encryption with Keyword Search (PEKS) [2]. SSE evolves from the prototype of sequential scanning the ciphertext stream without any index aside [5] to various sophisticated constructions [6], [7], [8] with delicate encrypted indexes for significantly accelerating the search operation.

As PEKS attracts more attention from researchers in past two decades, PEKS works with distinct features and functionalities are designed. A number of schemes supporting multi-keyword search were proposed in the literature [9], [10], [11]. However, these proposed schemes can't support multiple readers and writers simultaneously and do not achieve satisfactory performance. Specifically, the trapdoor and ciphertext size of [9], [11] is linear to either the number of keywords contained in the processed document or the number of keywords represented in the query. In addition, the public key size of [10] is also in proportion to the set size. Thus, it is a challenge to design public key, trapdoor, and ciphertext with short or constant size, reduce computational cost on trapdoor and ciphertext, and make the universal keyword set easy to expand.

According to the syntax of PEKS, to enable multi-reader access to the same message, multiple searchable ciphertexts of the same keyword should be generated for different readers, thereby multiplying the computation and storage overhead. Existing works supporting multi-user access [7],

[12], [13] more or less base their implementations on SSE or broadcast encryption [14]. Their multi-user access refers to one writer and multiple readers. Sun et al. utilized Ciphertext-Policy Attribute-Based Encryption (CP-ABE) in combination with the cross-tag proposed in [15] to support the multi-reader access in addition to the multi-keyword functionality [16]. In their scheme, a writer grants readers secret keys, which means to access data outsourced by distinct writers, each reader should maintain a set of secret keys. Moreover, the number of stored searchable ciphertexts is $\sum_{w\in\mathcal{W}}|DB[w]|$, where $DB[w]$ represents all documents that include the keyword w and $\mathcal{W}$ represents the universal keyword set. It means each document may accompany with multiple searchable ciphertexts, which could result in large storage overhead in a large scale system. In 2019, Xu et al. proposed a lattice-based PEKS scheme transferred from an anonymous identity-based (ID-based) scheme by replacing identities with keywords [17]. Their construction is actually an ID-based PEKS which maps the reader's identity to a matrix so that the writer can use the reader's identity to do encryption.

Keyword guessing attack (KGA) is a typical attack against PEKS [3], [18]. Since readers' public keys are known, anyone can generate searchable ciphertexts for desired keywords and perform the testing against a searching trapdoor. To resist KGA, cryptographic primitives such as public-key authenticated encryption with keyword search (PAEKS) [4] and public-key encryption with fuzzy keyword search (PEFKS) [19] were proposed. KGA undermines the search pattern privacy [7]. Hiding the search pattern and the access pattern should also be taken into consideration when building searchable encryption schemes. However, search pattern privacy is not preserved in many existing schemes [2], [20], where the adversary can tell whether the underlying keywords of two queries are identical or not. The access pattern [7] refers to the identifiers of matching documents, which is revealed in most searchable encryption schemes [5], [6], [7], [21], [22]. Although Oblivious RAM [23] is a potential solution to solve the problem, current ORAM constructions are still too expensive to be practical.

1.3 Organization

In Section 2, we define the notations which will be mentioned throughout this work and retrospect techniques which will be building blocks of our scheme. Problem formulation is presented in Section 3. Then syntax, correctness and security definitions of SE-EPOM are shown in Section 4. The proposed scheme is described in details in Section 5. Security analysis is given in Section 6 followed by the comparison of performance evaluation in Section 7. Finally we conclude this work in Section 8.

2 PRELIMINARIES

2.1 Notation

Notations and terminologies used throughout this paper are given in Table 1.

Suppose the number of keywords is μ and the universal keyword set is $\mathcal{W} = \{w_{\mu-1}, \cdots, w_0\}$. We use a decimal integer $T \in \{0, \cdots, 2^\mu - 1\}$ whose binary representation is $(T_{\mu-1}, \cdots, T_0)$ to represent the inclusion relationship between each keyword of $\mathcal{W}$ and a document D where

$$T_i = \begin{cases} 1 \text{ if } w_i \text{ is contained in } D, \\ 0 \text{ otherwise.} \end{cases}$$

$\mathcal{W}_T = \{w_i | w_i \in \mathcal{W}, T_i = 1\}$ denotes the keyword set corresponding to T. In a search query launched by a request user (RU), we use a decimal integer $t \in \{0, \cdots, 2^\mu - 1\}$ whose binary representation is $(t_{\mu-1}, \cdots, t_0)$ to represent the inclusion relationship between each keyword of $\mathcal{W}$ and RU's interest where

$$t_i = \begin{cases} 1 \text{ if RU is interested in } w_i, \\ 0 \text{ otherwise.} \end{cases}$$

$\mathcal{W}_t = \{w_i | w_i \in \mathcal{W}, t_i = 1\}$ denotes the keyword set corresponding to t. Obviously, if t matches T, $\mathcal{W}_t \subseteq \mathcal{W}_T$ holds.

TABLE 1: Notations and terminologies

Notation	Meaning
$\mathcal{W}$	The universal keyword set
$\mathcal{W}_x$	The keyword set represented by x
D	A document
$\mathcal{W}_{id}$	All keywords contained in document with identifier id
w	A keyword
DB	All documents in storage
$DB[w]$	All documents which include w
Q	The keyword set represented in a search query
$(x_{\mu-1}, \cdots, x_0)$	The unsigned binary representation of a positive decimal integer x
μ	The bit length of x, the number of keywords in $\mathcal{W}$
$x_{\mu-1}$	The most significant bit
x_0	The least bit
$\neg x$	The complement of x
T	Plaintext of searchable ciphertext, a decimal integer whose binary representation indicates each keyword inclusion relationship between $\mathcal{W}$ and $\mathcal{W}_T$
t	Plaintext of trapdoor, a decimal integer binary whose representation indicates each keyword inclusion relationship between $\mathcal{W}$ and $\mathcal{W}_t$
SUM	The decimal integer $2^\mu - 1$
td	Trapdoor
SC	Searchable ciphertext
$PK_i,\ pk_i$	Public key of party i
$SK_i,\ sk_i$	Secret key of party i
CT	Ciphertext
$[\![\cdot]\!]_{pk}$	The encryption of $\cdot$ under the public key pk
$\mathcal{L}(\cdot)$	The bit-length of $\cdot$

2.2 Secure Bit-Decomposition Protocol (SBD)

SBD [24] can convert the encryption of x into the encryption of the individual bits of x, leaking no information about x to both parties. SBD protocol will be one of the building blocks of our scheme and SBD could be defined as follows:

$$\text{SBD}([\![x]\!]_{pk}) \rightarrow ([\![x_{\mu-1}]\!]_{pk}, \cdots, [\![x_0]\!]_{pk}).$$

2.3 DT-PKC

Distributed Two-Trapdoor Public-Key Cryptosystem (DT-PKC) is an useful tool for dealing with integer operations across different encrypted domains by splitting a strong key into shares [25], which is based on partial homomorphic encryption (PHE) [26] and threshold cryptosystems [27], attaining more competitive computation performance than solutions using fully homomorphic encryption (FHE) [28].

2.3.1 Basic Algorithms

The DT-PKC algorithms in [25] are as follows:

`KeyGen` : Given a security parameter k, it outputs the strong private key SK, the public key pk_i and the weak private key sk_i of party i.

`Encryption` (`Enc`) : Given a message m and the public key pk_i of party i, it outputs the ciphertext $[\![m]\!]_{pk_i}$.

`Decryption With Weak Private Key` (`WDec`) : Given the ciphertext $[\![m]\!]_{pk_i}$ and the weak private key sk_i, it outputs the original message m.

`Strong Private Key Splitting` (`SKeyS`) : Given the strong private key SK, it outputs two partial strong private keys $SK^{(1)}$ and $SK^{(2)}$.

`Partial Decryption With Partial Strong Private Key Step One` (`PSDec1`) : Given the ciphertext $[\![m]\!]_{pk_i}$ and the partial strong private key $SK^{(1)}$, it runs the partial decryption algorithm $PDO_{SK^{(1)}}(\cdot)$ and outputs the step one partial ciphertext $CT_i^{(1)}$.

`Partial Decryption With Partial Strong Private Key Step Two` (`PSDec2`) : Given the step one partial ciphertext $CT_i^{(1)}$, the ciphertext $[\![m]\!]_{pk_i}$ and the partial strong private key $SK^{(2)}$, it runs the partial decryption algorithm $PDT_{SK^{(2)}}(\cdot,\cdot)$ and outputs the original message m.

`Ciphertext Refresh` (`CR`) : Given the ciphertext $[\![m]\!]_{pk_i}$, it outputs another ciphertext $[\![m]\!]'_{pk_i}$ of the same message.

2.3.2 Derived Protocols

The following DT-PKC derived protocols all take the same inputs, which are two ciphertexts $[\![x]\!]_{pk_a}$, $[\![y]\!]_{pk_b}$, the partial strong private keys $SK^{(1)}$, $SK^{(2)}$, and the public keys pk_a, pk_b, pk_c. The syntax is as follows:

`Secure Addition Protocol Across Domains` (`SAD`): Given the input, it outputs the ciphertext of the addition $[\![x+y]\!]_{pk_c}$.

`Secure Multiplication Protocol Across Domains` (`SMD`): Given the input, it outputs the ciphertext of the multiplication $[\![x \cdot y]\!]_{pk_c}$.

`Secure Less Than Protocol` (`STL`): Given the input, it outputs the ciphertext $[\![u^*]\!]_{pk_c}$ where $u^* = 0$ means $x \geq y$ and $u^* = 1$ means $x < y$.

`Secure Equivalent Testing Protocol` (`SEQ`): Given the input, it outputs the ciphertext $[\![f]\!]_{pk_c}$ where $f = 0$ means $x = y$ and otherwise means $x \neq y$.

3 Problem Formulation

3.1 System Model

In our design, the system consists of the following parties: a Key Generation Center (KGC), a Cloud Platform (CP), multiple Internal Servers (IS's), Data Providers (DPs) and Request Users (RUs). Figure 1 outlines the sketch of our system model.

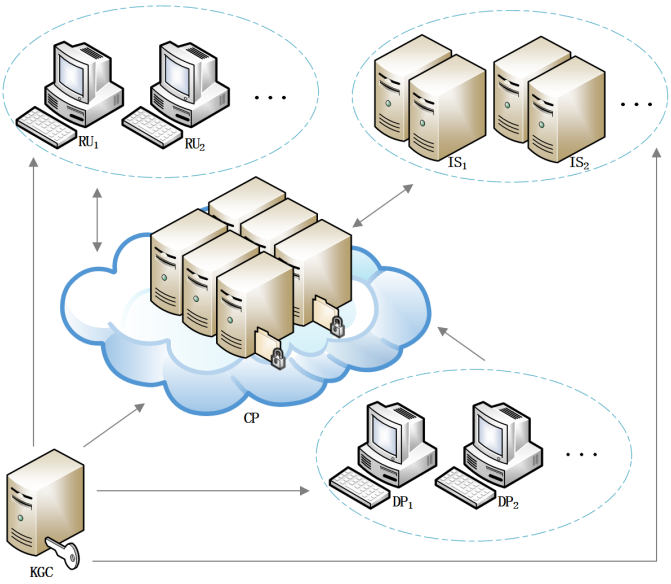


Fig. 1: System model.

- KGC: The Key Generation Center is in charge of generating public parameters, system keys, and keys of CP and IS's, as well as distributing corresponding keys to CP and IS's. For instance, the administrator of an organization can play the role of KGC.
- CP: The Cloud Platform is responsible for storing documents and corresponding searchable ciphertexts uploaded by DPs. It handles search queries with the assistance from IS's, and generates the (encrypted) searching result for RUs.
- IS: Internal Servers of an organization undertake partial computation to assist CP in handling a search query. In practice, IS's can be designated servers in an organization.
- DP: Each Data Provider generates its own public and secret key pair based on the public parameters, computes the searchable ciphertexts according to the keywords associated with a document, and stores the document with the searchable ciphertext on CP.
- RU: Each Request User generates its own public and secret key pair based as DP, and computes the trapdoor for specific keywords of interest. It decrypts the search results sent back from CP and obtains the indexes of the documents satisfying the search query.

3.2 Threat Model

In this work, we assume that KGC is honest and follows the scheme to generate and distribute corresponding key pairs to CP and IS's.

CP and any IS are assumed to be a pair of non-colluding semi-honest adversaries[1]. That is, even though they follow the protocol honestly, they are still interested in finding

1. Under the definition of non-colluding semi-honest adversaries in [29], the threat model is subject to the restriction that CP and any IS cannot be compromised concurrently.

information about other parties from either intermediate values or computation results. For example, CP may try to tell the underlying keywords from a trapdoor or learn which documents match a trapdoor.

DPs are also assumed to be semi-honest adversaries who follow the protocol but may try to extract privacy about other parties. For instance, a DP may eavesdrop the communications among other parties and attempt to learn sensitive information about a query, e.g., which documents are searched by an RU.

We assume RUs would honestly follow the protocol when performing a search query by themselves but may seek to learn sensitive information related to the searchable ciphertexts and searching queries generated by other DPs and RUs.

3.3 Design Goals

We aim to design an SE-EPOM satisfying the following functions and security requirements.

1) *Data Privacy:* CP is unable to learn information about the data or keyword encrypted and uploaded by DP.
2) *Query Privacy:* CP is prohibited from learning which keyword(s) an RU is searching.
3) *Multi-Keyword Search:* Every authorized RU is permitted to perform both single-keyword and multi-keyword search queries. In addition, the trapdoor of a multi-keyword query should be indistinguishable from that of a single-keyword query.
4) *Short Trapdoor and Ciphertext:* The size of trapdoor and ciphertext is constant and independent on the number of keywords.
5) *Multi-User Access:* The system allows multiple DPs to outsource their data resource to the cloud and multiple RUs to search the same set of data, i.e. multi-writer/multi-reader setting.
6) *Search Pattern Hiding:* Given two search queries possibly performed by the same RU, CP cannot link their underlying keywords.
7) *Access Pattern Hiding:* Search result of a query such as identifiers of documents satisfying the query is hidden from CP.

4 SE-EPOM

4.1 Syntax

A Searchable Encryption based on Efficient Privacy-preserving Outsourced calculation framework with Multiple keys is a protocol among a KGC, a CP, multiple IS's, multiple DPs, and multiple RUs as follows [2].

KeyGen$(1^k) \rightarrow (PK_{DP}, SK_{DP}, PK_{RU}, SK_{RU}, SK_{CP}, SK_{IS})$: Given the security parameter k, KGC generates the public parameter PP[3], the system secret key SK, the secret key SK_{CP} of CP, and the secret key SK_{IS} of IS. Each DP generates its secret key SK_{DP} and its public key PK_{DP}. Each RU generates its secret key SK_{RU} and its public key PK_{RU}.

Store$(PK_{DP}, \mathcal{W}_T) \rightarrow SC$: Given DP's public key PK_{DP} and a document keyword set $\mathcal{W}_T$ to be processed, DP computes the searchable ciphertext SC.

Trapdoor$(PK_{RU}, \mathcal{W}_t) \rightarrow td$: Given RU's public key PK_{RU} and the keyword set $\mathcal{W}_t$ of interest, RU computes the trapdoor td.

Test$(PK_{DP}, SK_{CP}, SK_{IS}, PK_{RU}, SK_{RU}, td, SC) \rightarrow 0/1$: Given DP's public key PK_{DP}, CP's secret key SK_{CP}, IS's secret key SK_{IS}, RU's public key PK_{RU} and secret key SK_{RU}, RU's trapdoor td and a searchable ciphertext SC, CP and IS computes an intermediate value which implies whether SC matches td. RU computes the test result from the intermediate value and outputs 1 if $\mathcal{W}_t \subseteq \mathcal{W}_T$ or 0 otherwise.

4.2 Correctness

For a searchable encryption scheme, the most important requirement is that the returning result from the server must be what the client wants to obtain. To be detailed, we formulate the correctness as follows:

- **Test** $(PK_{DP}, SK_{CP}, SK_{IS}, PK_{RU}, SK_{RU}, td, SC) \rightarrow 1$ if and only if $\mathcal{W}_t \subseteq \mathcal{W}_T$.
- **Test** $(PK_{DP}, SK_{CP}, SK_{IS}, PK_{RU}, SK_{RU}, td, SC) \rightarrow 0$ if and only if $\mathcal{W}_t \nsubseteq \mathcal{W}_T$.

4.3 Security Definitions

4.3.1 Ciphertext Indistinguishability

When an encrypted document is uploaded to CP, DP should also attach the corresponding encrypted searchable ciphertext. It is required that the encrypted searchable ciphertext should not leak any information about its underlying keyword. It is worth noting that, even though CP can interact with IS to run **Test**, the final test result can only be accessed by RU. Another consideration is that since the trapdoor is the encryption under RU's public key and the searchable ciphertext is the encryption under DP's public key, the adversary can generate trapdoors and searchable ciphertexts for any keyword set of its choice by itself. Thus, in the game, there is no need for the challenger to respond to trapdoor or ciphertext queries which usually simulate the chosen keyword attacks (CKA). Due to the multi-keyword search feature of our system model, the distinguishing game can be described as follows: the adversary chooses two distinct keyword sets of interest and sends them to the challenger, the challenger randomly chooses one of them to derive the corresponding encrypted searchable ciphertext and returns it; then the adversary tries to guess which one is the underlying keyword set.

The indistinguishability game $IND-SC$ depicting the security requirement is defined as follows:

Setup. The challenger $\mathcal{C}$ runs **KeyGen**$(1^k) \rightarrow (PK_{DP}, SK_{DP}, PK_{RU}, SK_{RU}, SK_{CP}, SK_{IS})$, sends $(PK_{DP}, PK_{RU}, SK_{CP})$ to the adversary $\mathcal{A}$ and keeps $(SK_{DP}, SK_{RU}, SK_{IS})$ secret.

Challenge. The adversary $\mathcal{A}$ picks two different keyword sets $\mathcal{W}_0, \mathcal{W}_1 \subseteq \mathcal{W}$. $\mathcal{A}$ sends $\mathcal{W}_0, \mathcal{W}_1$ to the challenger

2. Unless otherwise specified, our scheme refers to the architecture with only one IS, one DP, and one RU for ease of description.

3. PP is the input of each algorithm and will not be included explicitly.

$\mathcal{C}$. $\mathcal{C}$ picks $b \in_R \{0,1\}$, runs **Store**$(PK_{DP}, \mathcal{W}_b) \rightarrow SC_b$ and sends SC to $\mathcal{A}$.

Output. The adversary $\mathcal{A}$ gives its guess b' and wins the game if $b' = b$.

Definition 1. We say that a SE-EPOM satisfies indistinguishability in the above game, if for any probabilistic polynomial-time (PPT) adversary $\mathcal{A}$, the advantage

$$\mathsf{Adv}^{IND-SC}_{SE-EPOM,\mathcal{A}}(\lambda) = |\Pr[\mathcal{A}^{b=0} \text{ wins}] - \Pr[\mathcal{A}^{b=1} \text{ wins}]| \tag{1}$$

is negligible.

4.3.2 Trapdoor Privacy

As one of the security concerns in PEKS, trapdoor privacy refers to hiding the information about the underlying keyword of a trapdoor. For our system model which considers a multi-keyword search, it means CP cannot learn anything about the underlying keywords from the trapdoor it receives from an RU. Similar to the previous security requirement, trapdoor queries and ciphertext queries are not considered. However, we allow CP to communicate with IS to perform the test operation and obtain the intermediate result. The indistinguishability game can be described as follows: the adversary chooses two distinct keyword sets of interest and sends them to the challenger, the challenger randomly chooses one of them to derive the corresponding trapdoor and returns it; then the adversary tries to guess which one is the underlying keyword set of the challenge trapdoor.

The indistinguishability game $IND-TD$ depicting the security requirement is defined as follows:

Setup. The challenger $\mathcal{C}$ runs **KeyGen**$(1^k) \rightarrow (PK_{DP}, SK_{DP}, PK_{RU}, SK_{RU}, SK_{CP}, SK_{IS})$, sends $(PK_{DP}, PK_{RU}, SK_{CP})$ to the adversary $\mathcal{A}$ and keeps $(SK_{DP}, SK_{RU}, SK_{IS})$ secret.

Challenge. The adversary $\mathcal{A}$ picks two different keyword sets $\mathcal{W}_0, \mathcal{W}_1 \subseteq \mathcal{W}$. $\mathcal{A}$ sends $\mathcal{W}_0, \mathcal{W}_1$ to the challenger $\mathcal{C}$. $\mathcal{C}$ picks $b \in_R \{0,1\}$, runs **Trapdoor**$(PK_{RU}, \mathcal{W}_b) \rightarrow td_b$, sends td_b to $\mathcal{A}$.

Query. The adversary $\mathcal{A}$ is allowed to interact with the challenger $\mathcal{C}$ who acts as IS according to the **Test** protocol.

Output. The adversary $\mathcal{A}$ gives its guess b' and wins the game if $b' = b$.

Definition 2. We say that a SE-EPOM satisfies indistinguishability in the above game, if for any polynomial probabilistic-time (PPT) adversary $\mathcal{A}$, the advantage

$$\mathsf{Adv}^{IND-TD}_{SE-EPOM,\mathcal{A}}(\lambda) = |\Pr[\mathcal{A}^{b=0} \text{ wins}] - \Pr[\mathcal{A}^{b=1} \text{ wins}]| \tag{2}$$

is negligible.

4.3.3 Search Pattern Privacy

From intuition, when two search queries are launched by RU, CP should not be able to tell whether these two queries have the same underlying keyword set. This kind of security requirement is called search pattern privacy [30]. Due to the syntax of SE-EPOM, it means given two trapdoors, the adversary should not tell whether their underlying keyword sets are the same. Similar to the previous security requirement, trapdoor queries and ciphertext queries are not considered. Inspired by the security definitions for PEKS in [30], we formulate the search pattern privacy (SPP) in the following game:

Setup. The challenger $\mathcal{C}$ runs **KeyGen**$(1^k) \rightarrow (PK_{DP}, SK_{DP}, PK_{RU}, SK_{RU}, SK_{CP}, SK_{IS})$, sends $(PK_{DP}, PK_{RU}, SK_{CP})$ to the adversary $\mathcal{A}$ and keeps $(SK_{DP}, SK_{RU}, SK_{IS})$ secret.

Challenge. The adversary $\mathcal{A}$ picks two different keyword sets $\mathcal{W}_0, \mathcal{W}_1 \subseteq \mathcal{W}$. $\mathcal{A}$ sends $\mathcal{W}_0, \mathcal{W}_1$ to the challenger $\mathcal{C}$. $\mathcal{C}$ picks $b \in_R \{0,1\}$, runs **Trapdoor**$(PK_{RU}, \mathcal{W}_0) \rightarrow td_0$ and **Trapdoor**$(PK_{RU}, \mathcal{W}_b) \rightarrow td_b$, sends td_0, td_b to $\mathcal{A}$.

Query. The adversary $\mathcal{A}$ is allowed to interact with the challenger $\mathcal{C}$ who acts as IS according to the **Test** protocol.

Output. The adversary $\mathcal{A}$ gives its guess b' and wins the game if $b' = b$.

Definition 3. We say that a SE-EPOM satisfies search pattern privacy in the above SPP game, if for any PPT adversary $\mathcal{A}$, the advantage

$$\mathsf{Adv}^{SPP}_{SE-EPOM,\mathcal{A}}(\lambda) = |\Pr[\mathcal{A}^{b=0} \text{ wins}] - \Pr[\mathcal{A}^{b=1} \text{ wins}]| \tag{3}$$

is negligible.

4.3.4 Relation between Trapdoor Privacy and Search Pattern Privacy

From our security definitions, the adversaries of the $IND-TD$ game and the SPP game are both aiming to collect information about the underlying keywords of a trapdoor. Therefore, it is natural to explore the relationship between the two security requirements.

Theorem 1. *A SE-EPOM scheme satisfies search pattern privacy if it satisfies trapdoor privacy.*

Proof. Assume that SE-EPOM does not satisfy search pattern privacy. We show that it does not satisfy trapdoor privacy, either.

When SE-EPOM does not satisfy search pattern privacy, it means there exists an adversary $\mathcal{B}$ which can win the SPP game with non-negligible probability. Suppose there is an adversary $\mathcal{A}$ who runs $\mathcal{B}$ as a subroutine in the $IND-TD$ game as follows:

Setup. The challenger $\mathcal{C}$ runs **KeyGen**$(1^k) \rightarrow (PK_{DP}, SK_{DP}, PK_{RU}, SK_{RU}, SK_{CP}, SK_{IS})$, sends $(PK_{DP}, PK_{RU}, SK_{CP})$ to the adversary $\mathcal{A}$ and keeps $(SK_{DP}, SK_{RU}, SK_{IS})$ secret.

$\mathcal{A}$ sends $(PK_{DP}, PK_{RU}, SK_{CP})$ to the adversary $\mathcal{B}$. Then the adversary $\mathcal{A}$ runs the adversary $\mathcal{B}$ as a subroutine, $\mathcal{B}$ picks two different keyword sets $\mathcal{W}_0, \mathcal{W}_1 \subseteq \mathcal{W}$ and sends them to $\mathcal{A}$.

Challenge. $\mathcal{A}$ sends $\mathcal{W}_0, \mathcal{W}_1$ to the challenger $\mathcal{C}$. $\mathcal{C}$ picks $b \in_R \{0,1\}$, runs **Trapdoor**$(PK_{RU}, \mathcal{W}_b) \rightarrow td_b$ and sends td_b to $\mathcal{A}$.

$\mathcal{A}$ runs **Trapdoor**$(PK_{RU}, \mathcal{W}_0) \rightarrow td_0$ and sends td_0, td_b to $\mathcal{B}$. In $\mathcal{B}$'s view, td_0, td_b correspond to the challenge td_0, td_b in the SPP game. As the above assumption, $\mathcal{B}$ outputs b' such that $b' = b$ with non-negligible probability.

Query. $\mathcal{B}$'s queries can be answered by $\mathcal{A}$ by directly forwarding the same queries to $\mathcal{C}$.

Output. The adversary $\mathcal{A}$ gives its guess b' if and only if the adversary $\mathcal{B}$ outputs b'. $\mathcal{A}$ wins the game if $b' = b$.

Therefore, $\mathcal{A}$ definitely wins the game with non-negligible probability. That is, SE-EPOM does not satisfy trapdoor privacy.

4.4 Simulation-based Security Definition

We also consider a simulation-based security model for non-colluding semi-honest adversaries presented in [25], [29]. As we will show later, this more abstract model can simultaneously capture the security requirements defined above. Due to the scenario of SE-EPOM, there is a DP (D_a), a CP (S_1) and an IS (S_2). We refer readers to [29] for details.

Let $\mathcal{P} = (D_a, S_1, S_2)$ be the set of all protocol parties and there are three kinds of adversaries $\mathcal{A}_{D_a}, \mathcal{A}_{S_1}, \mathcal{A}_{S_2}$ which corrupt D_a, S_1, S_2 respectively.

In the real world, D_a run with inputs x, y (with additional auxiliary inputs z_x, z_y), while S_1, S_2 receive auxiliary inputs z_1, z_2 respectively. Let $\mathcal{H} \subset \mathcal{P}$ be the set of honest parties. If P is honest, i.e., $P \in \mathcal{H}$, out_P is the output of party P. If P is corrupted, i.e., $P \in \mathcal{P}\backslash H$, out_P is the view of P during the protocol.

For each $P^* \in \mathcal{P}$, the partial view of P^* in a real world execution of protocol Π between parties $\mathcal{P} = (D_a, S_1, S_2)$ with adversaries $\mathcal{A} = (\mathcal{A}_{D_a}, \mathcal{A}_{S_1}, \mathcal{A}_{S_2})$ present is defined as follows:

$$REAL^{P^*}_{\Pi,\mathcal{A},\mathcal{P},\mathbf{z}}(k, x, y) = \{output_P : P \in H\} \cup out_{P^*}.$$

In the ideal world, all the parties interact with a trusted party that evaluates f. The challenge $DP\ a$ sends x, y to f. If either x or y is $\perp$, then f returns $\perp$. Finally f returns $f(x, y)$ to the challenge $DP\ a$. If P is honest, i.e., $P \in \mathcal{H}$, let out_P be the output returned by f to party P. If P is corrupted, i.e., $P \in \mathcal{P}\backslash H$, let out_P be some value output by P. For each $P^* \in \mathcal{P}$, the partial view of P^* in an ideal world execution of protocol Π between parties $\mathcal{P} = (D_a, S_1, S_2)$ with independent simulators $\mathcal{S} = (\mathcal{S}_{D_a}, \mathcal{S}_{S_1}, \mathcal{S}_{S_2})$ present is defined as follows:

$$IDEAL^{P^*}_{f,\mathcal{S},\mathcal{P},\mathbf{z}}(k, x, y) = \{output_P : P \in H\} \cup out_{P^*}.$$

Informally, a protocol Π is considered secure against non-colluding semi-honest adversaries if it partially emulates, in the real world, an evaluation of f in the ideal world.

Definition 4. Let f be a deterministic functionality among parties $\mathcal{P} = (D_a, S_1, S_2)$ and Π be a protocol among parties $\mathcal{P} = (D_a, S_1, S_2)$. Furthermore, let $\mathcal{H} = \emptyset$, i.e., each party $P \in \mathcal{P}$ is semi-honest non-colluding parties. We say that $\Pi(\mathcal{P})$-securely computes f if there exists a set $\mathbf{Sim} = (\mathbf{Sim}_{D_a}, \mathbf{Sim}_{S_1}, \mathbf{Sim}_{S_2})$ of PPT transformations such that for all semi-honest non-colluding adversaries $\mathcal{A} = (\mathcal{A}_{D_a}, \mathcal{A}_{S_1}, \mathcal{A}_{S_2})$, for all $x, y \in \{0, 2^\mu - 1\}$, $z \in \{0, 2^\mu - 1\}$ and for all parties $P \in \mathcal{P}$,

$$\{REAL^{P^*}_{\Pi,\mathcal{A},\mathcal{P},\mathbf{z}}(k, x, y)\}_{k\in\mathbb{N}} \overset{c}{\approx} \{IDEAL^{P^*}_{f,\mathcal{S},\mathcal{P},\mathbf{z}}(k, x, y)\}_{k\in\mathbb{N}} \tag{3}$$

where $\mathcal{S} = (\mathcal{S}_{D_a}, \mathcal{S}_{S_1}, \mathcal{S}_{S_2})$ and $\mathcal{S}_{D_a} = \mathbf{Sim}_{D_a}(\mathcal{A}_{D_a}), \mathcal{S}_{S_1} = \mathbf{Sim}_{S_1}(\mathcal{A}_{S_1}), \mathcal{S}_{S_2} = \mathbf{Sim}_{S_2}(\mathcal{A}_{S_2})$.

4.5 Relation between Security Definitions

In this section, we prove that if an SE-EPOM is securely realized according to Definition 4, then it satisfies the security requirements of ciphertext indistinguishability (Definition 1) and trapdoor privacy (Definition 2).

Theorem 2. *An SE-EPOM satisfies ciphertext indistinguishability and trapdoor privacy if it is securely realized according to Definition 4 with adversaries $\mathcal{A} = (\mathcal{A}_{D_a}, \mathcal{A}_{S_1}, \mathcal{A}_{S_2})$ present.*

Proof. Assume that SE-EPOM does not satisfy (one of) ciphertext indistinguishability or trapdoor privacy. We show that SE-EPOM cannot be securely realized according to Definition 4.

There is a distinguisher $\mathcal{Z}$ trying to distinguish the real world from the ideal world.

1) Assume that SE-EPOM does not satisfy the searchable ciphertext indistinguishability in Definition 1. That is, there exists an adversary $\mathcal{B}$ such that Equation (1) is non-negligible.
The distinguisher $\mathcal{Z}$ asks $\mathcal{A}$ or $\mathcal{S}$ to corrupt S_1 (CP) so that S_1 relays each message which it received from D_a (DP) to $\mathcal{Z}$ (in the real world). S_1 behaves honestly. $\mathcal{Z}$ internally runs the adversary $\mathcal{B}$:
If $\mathcal{B}$ sends $\mathcal{W}_{T,0}, \mathcal{W}_{T,1} \subseteq \mathcal{W}$ to the challenger, then
   a) $\mathcal{Z}$ activates D_a with input (**Store**, sid, $\mathcal{W}_{T,b}$) where $b \in \{0, 1\}$ is a random bit.
   b) In the real world,
      D_a sends SC to $\mathcal{S}_1(\mathcal{A})$, then $\mathcal{S}_1(\mathcal{A})$ replays it to $\mathcal{Z}$.
      In the ideal world,
      D_a sends (**Store**, sid, $\mathcal{W}_{T,b}$) to f.
      f sends $|\mathcal{W}_{T,b}|$ to $\mathcal{S}$.
      $\mathcal{S}$ computes SC' and sends it to $\mathcal{Z}$.

   Finally $\mathcal{Z}$ outputs 1 if and only if $\mathcal{B}$ outputs 1.
   If $\mathcal{Z}$ interacts with the protocol Π, SC is simulated for $\mathcal{B}$ since $\mathcal{A}$ plays the role of $\mathcal{A}_{S_1}$. While, if $\mathcal{Z}$ interacts with $\mathcal{S}_{S_1}$, SC' is simulated for $\mathcal{B}$ since the ideal world adversary $\mathcal{S}$ plays the role of $\mathcal{S}_{S_1}$.
   From our assumption, there exists an adversary $\mathcal{B}$ which distinguishes searchable ciphertexts in the real world, outputting 1 with non-negligible advantage over 0, while in the ideal world outputs 1 with probability $\frac{1}{2}$. Obliviously, the distinguisher $\mathcal{Z}$ which runs $\mathcal{B}$ as a subroutine can distinguish the partial view of the party S_1 in the real world execution from that of the ideal world execution. That is, the protocol cannot securely realize SE-EPOM.
2) Assume that SE-EPOM does not satisfy the trapdoor privacy in Definition 3. That is, there exists an adversary $\mathcal{B}'$ such that Equation (2) is non-negligible.
The distinguisher $\mathcal{Z}$ asks $\mathcal{A}$ or $\mathcal{S}$ to corrupt S_1 (CP) so that S_1 relays each message which it received from RU to $\mathcal{Z}$ (in the real world). S_1 behaves honestly. $\mathcal{Z}$ internally runs the adversary $\mathcal{B}'$:
If $\mathcal{B}'$ sends $\mathcal{W}_{t,0}, \mathcal{W}_{t,1} \subseteq \mathcal{W}$ to $\mathcal{Z}$, then
   a) $\mathcal{Z}$ activates RU with input (**Trapdoor**, sid, $\mathcal{W}_{t,b}$) where $b \in \{0, 1\}$ is a random bit.
   b) In the real world,
      RU sends td to $\mathcal{S}_1(\mathcal{A})$, then $\mathcal{S}_1(\mathcal{A})$ replays it to $\mathcal{Z}$.
      In the ideal world,

RU sends (**Trapdoor**, sid, $\mathcal{W}_{t,b}$) to f.
f sends $|\mathcal{W}_{t,b}|$ to $\mathcal{S}$.
$\mathcal{S}$ computes td' and sends it to $\mathcal{Z}$.

Finally $\mathcal{Z}$ outputs 1 if and only if $\mathcal{B}'$ outputs 1.
If $\mathcal{Z}$ interacts with the protocol Π, td is simulated for $\mathcal{B}'$ since $\mathcal{A}$ plays the role of $\mathcal{A}_{S_1}$. While, if $\mathcal{Z}$ interacts with $\mathcal{S}_{S_1}$, td' is simulated for $\mathcal{B}'$ since the ideal world adversary $\mathcal{S}$ plays the role of $\mathcal{S}_{S_1}$.
From our assumption, there exists an adversary $\mathcal{B}'$ which distinguishes trapdoors in the real world, outputting 1 with non-negligible advantage over 0, while in the ideal world outputs 1 with probability $\frac{1}{2}$. Obviously, the distinguisher $\mathcal{Z}$ which runs $\mathcal{B}'$ as a subroutine can distinguish the partial view of the party S_1 in the the real world execution from that of the ideal world execution. That is, the protocol cannot securely realize SE-EPOM.

5 Our SE-EPOM Construction

5.1 An Overview of Our Construction

Given trapdoor t which represents a set $\mathcal{W}_t$ of keywords of interest and a searchable ciphertext T which represents a set $\mathcal{W}_T$ of containing keywords, the match of t and T means all keywords of $\mathcal{W}_t$ are included in $\mathcal{W}_T$, i.e., $\mathcal{W}_t \subseteq \mathcal{W}_T$. Then deciding whether t matches T turns to checking whether $\mathcal{W}_t \subseteq \mathcal{W}_T$.

We first use the binary representations and decimal integers to denote each set in accordance with Section 2.1, then perform some calculations on these binary representations and decimal integers, and finally tell whether $\mathcal{W}_t \subseteq \mathcal{W}_T$ according to these calculation results. A novel *Subset Decision Mechanism* (shown in the next subsection) is proposed for such a purpose.

Note that for ease of understanding, calculations in *Subset Decision Mechanism* are all performed on plaintexts. However, to enforce data and query privacy, calculations should be all performed on ciphertexts instead. Moreover, the involved ciphertexts sometimes may come from encryptions under different public keys of different parties, e.g. a trapdoor from RU and a searchable ciphertext from DP. Thus, the remaining issue is to enable computation on ciphertexts under different public keys. Here we utilize the secure computation protocols across domains in [25] where a pair of servers are deployed to apply our *Subset Decision Mechanism* on ciphertexts, which are built on the base of homomorphic encryption.

As depicted in Figure 1, our scheme consists of a CP and multiple IS's. KGC generates multiple independent shares of the strong private key and distributes each key pair between the CP and an IS. When a new IS is introduced into the system, the KGC can just repeat the process. Since these key pairs are all independent, even the IS's collude, they are not able to derive the strong private key. Then CP can ask one or multiple IS's to assist in handling the same search query, where each IS handles a part of keywords. For instance, 10 keywords could be split into 2 5-keyword parts and handled by 2 IS's using the *Subset Decision Mechanism* introduced later. Under such a distributed system where multiple parallel IS's are in service for the same query, the more IS's are used, the faster a query will be responded. Also, it can effectively support load balancing: if an IS is offline or overloaded, then other IS's could share the workload. In addition, replication, an important feature of distributed systems, can also be obtained by letting different IS's test on the same data. The usability and reliability are enhanced in this way. To simplify description, below we will present our scheme with only one IS deployed and it is straightforward to add new IS's by distributing an independent key pair between the CP and each new IS.

5.2 Subset Decision Mechanism

Assume there is a universal set $\mathcal{W} = \{w_{\mu-1}, \cdots, w_0\}$ whose binary representation is $(b_{\mu-1}, \cdots, b_0) = (1, \cdots, 1)$, and its corresponding decimal integer is SUM. There are two subsets $\mathcal{W}_T$ and $\mathcal{W}_t$ of $\mathcal{W}$, whose binary representation are $(T_{\mu-1}, \cdots, T_0)$ and $(t_{\mu-1}, \cdots, t_0)$, respectively. Their corresponding decimal integer are T and t, respectively.

Our method of deciding $\mathcal{W}_t \subseteq \mathcal{W}_T$ is to make sure that there is not any i s.t. $t_i = 1$ and $T_i = 0$. The deciding procedure is elaborated as follows. Given the μ bit binary representations of T, its complement is $(\neg T_{\mu-1}, \cdots, \neg T_0)$, whose corresponding integer is $\neg T$. Then we compute the bitwise addition c_i of t_i and $\neg T_i$. Finally, if none of $c_i = 2$, $\mathcal{W}_t \subseteq \mathcal{W}_T$; otherwise, $\mathcal{W}_t \nsubseteq \mathcal{W}_T$. The formal mechanism is outlined in Algorithm 1.

Algorithm 1 Subset Decision

Input: A universal set $\mathcal{W} = \{w_{\mu-1}, \cdots, w_0\}$, two subsets $\mathcal{W}_T, \mathcal{W}_t \subseteq \mathcal{W}$.
Output: Whether $\mathcal{W}_t \subseteq \mathcal{W}_T$.

1: Compute the binary representations $(T_{\mu-1}, \cdots, T_0)$, $(t_{\mu-1}, \cdots, t_0)$ of $\mathcal{W}_T, \mathcal{W}_t$.
2: Compute the complement $(\neg T_{\mu-1}, \cdots, \neg T_0)$ of $(T_{\mu-1}, \cdots, T_0)$.
3: Set $i = 0, R = 1$.
4: **while** $i < \mu$ **do**
5: $\quad c_i = \neg T_i + t_i$,
6: $\quad d_i = 2 - c_i$,
7: $\quad R = R \cdot d_i$,
8: **end while**
9: **if** $R = 0$ **then**
10: $\quad$ **return** $\mathcal{W}_t \nsubseteq \mathcal{W}_T$.
11: **else**
12: $\quad$ **return** $\mathcal{W}_t \subseteq \mathcal{W}_T$.
13: **end if**

Besides, our another observation can somehow accelerate the algorithm when $\mathcal{W}_t \nsubseteq \mathcal{W}_T$. That is, when $\mathcal{W}_t \subseteq \mathcal{W}_T$, $sum = t + \neg T \leq SUM = 2^\mu - 1$. Accordingly, if $sum > SUM$, we must have $\mathcal{W}_t \nsubseteq \mathcal{W}_T$. The mechanism with this modification is outlined in Algorithm 2. For ease of understanding, we take Table 2 as a toy example to illustrate the subset decision mechanism in Algorithm 2.

Correctness Analysis: Here we will demonstrate the correctness of Algorithm 2.

- For the output that $\mathcal{W}_t \nsubseteq \mathcal{W}_T$, it splits into two cases.

`Case-1` When $sum \leq SUM$ but there exist carries in the addition bitwise of μ bits, suppose the least

Algorithm 2 Subset Decision with Modification

Input: A universal set $\mathcal{W} = \{w_{\mu-1}, \cdots, w_0\}$, two subsets $\mathcal{W}_T, \mathcal{W}_t \subseteq \mathcal{W}$.
Output: Whether $\mathcal{W}_t \subseteq \mathcal{W}_T$.

1: Besides computations in Step 1, 2 of Algorithm 1, compute the decimal integers T, t of $\mathcal{W}_T, \mathcal{W}_t$, $SUM = 2^\mu - 1$ of $\mathcal{W}$, $\neg T = SUM - T$, $sum = \neg T + t$.
2: **if** $sum > SUM$ **then**
3: **return** $\mathcal{W}_t \not\subseteq \mathcal{W}_T$.
4: **else**
5: Go to Step 3 of Algorithm 1.
6: **end if**

TABLE 2: A Toy Example

i	μ	$\mu-1$	$\cdots$				0
	6	5	4	3	2	1	0
SUM	0	1	1	1	1	1	1
T		0	0	0	1	1	0
$\neg T$		1	1	1	0	0	1
t^1		0	0	0	1	0	1
sum	0	1	1	1	1	1	0
c		1	1	1	1	0	2
d		1	1	1	1	2	0
t^2		0	0	1	1	1	0
sum	1	0	0	0	1	1	1
t^3		0	0	0	1	0	0
sum	0	1	1	1	1	0	1
c		1	1	1	1	0	1
d		1	1	1	1	2	1

[0] sum is the addition of $\neg T$ and t. c and d are intermediate values in Algorithm 1. Here $\mu = 6, SUM = 63, \mathcal{W} = \{w_5, \cdots, w_0\}, T = 6, \mathcal{W}_T = \{w_2, w_1\}, \neg T = 57$. Each bit of the binary representation is listed on their right side.

[1] In this case, $t = 5, \mathcal{W}_t = \{w_2, w_0\}$. Though $sum = 62 \leq SUM$, there exists a carry in the addition bitwise of 6 bits and the least bit where there is a carry generated from is the 0th bit. Then we have $sum_0 = 0, c_0 = 2, d_0 = 0$ caused by $\neg T_0 = 1, T_0 = 0, t_0 = 1$. Thus, $w_0 \in \mathcal{W}_t, w_0 \notin \mathcal{W}_T, \mathcal{W}_t \not\subseteq \mathcal{W}_T$.

[2] In this case, $t = 14, \mathcal{W}_t = \{w_3, w_2, w_1\}$. $sum = 71 > SUM$, without additional checking, we have that there must exist at least one carry in the addition bitwise of the 6 bits so that the carry is delivered forward to the 6th bit, i.e., $sum_6 = 1$ s.t. $sum > SUM$ since SUM only has 6 bits, i.e., $SUM_6 = 0$. Then the reasoning is similar to the above case and the least bit where there is a carry generated from is the 3th bit caused by $\neg T_3 = 1, T_3 = 0, t_3 = 1$. Thus, $w_3 \in \mathcal{W}_t, w_3 \notin \mathcal{W}_T, \mathcal{W}_t \not\subseteq \mathcal{W}_T$.

[3] In this case, $t = 4, \mathcal{W}_t = \{w_2\}$. $sum = 61 \leq SUM$ and there is not any carry in the addition bitwise of 6 bits, then we have $c_i = \neg T_i + t_i = 0/1$ s.t. $d_i = 2 - c_i \neq 0$ for each $i \in \{0, \cdots, 5\}$ caused by $\neg T_i, t_i$ are not 1 simultaneously, i.e., they satisfy $\neg T_i = 0, T_i = 1, t_i = 0$ or $\neg T_i = 0, T_i = 1, t_i = 1$ or $\neg T_i = 1, T_i = 0, t_i = 0$. That is, for each $i \in \{0, \cdots, 5\}$, if $w_i \in \mathcal{W}_t$, we must have $w_i \in \mathcal{W}_T$. Thus, $\mathcal{W}_t \subseteq \mathcal{W}_T$.

bit where there is a carry generated from is the ith bit for $i \in \{0, \cdots, \mu - 1\}$, i.e., $d_i = 2 - c_i = 0, c_i = 2$. Then we must have $\neg T_i = 1, T_i = 0, t_i = 1$ which means the keyword $w_i \in \mathcal{W}_t, w_i \notin \mathcal{W}_T$. Thus, $\mathcal{W}_t \not\subseteq \mathcal{W}_T$.

`Case-2` When $sum > SUM$, there must exist at least one carry in the addition bitwise of the μ bits so that the carry is delivered forward to the μth bit, i.e., $sum_\mu = 1 > SUM_\mu = 0$ since $SUM = 2^\mu - 1$ only has μ bits. Then the reasoning is similar to `Case-1` and we come to the conclusion that $\mathcal{W}_t \not\subseteq \mathcal{W}_T$.

- For the output that $\mathcal{W}_t \subseteq \mathcal{W}_T$, there is only one case:

`Case-3` When $sum \leq SUM$ and there is not any carry in the addition bitwise of μ bits, we must have $c_i = \neg T_i + t_i = 0/1$ s.t. $d_i = 2 - c_i \neq 0$ for each $i \in \{0, \cdots, \mu - 1\}$. It means $\neg T_i, t_i$ should not be both 1, then we have

$$\begin{cases} \neg T_i = 0, T_i = 1, t_i = 0 & \text{or} \\ \neg T_i = 0, T_i = 1, t_i = 1 & \text{or} \\ \neg T_i = 1, T_i = 0, t_i = 0. \end{cases}$$

In the above three items, we have if $w_i \in \mathcal{W}_t$, then $w_i \in \mathcal{W}_T$. Thus, we come to the conclusion that $\mathcal{W}_t \subseteq \mathcal{W}_T$.

5.3 Detailed Construction

For clarity, we denote the Distributed Two Trapdoors Public-Key Cryptosystem as DT-PKC. Our protocol is outlined based on the workflow of Algorithm 1 as follows:

KeyGen. Given the security parameter k, KGC, DPs and RUs perform the following operations:

1) KGC finds two large primes p, q s.t. $\mathcal{L}(p) = \mathcal{L}(q) = k$. Then KGC first runs `KeyGen` to get the strong private key $SK = \lambda$ and runs `SKeyS` to generate two partial strong private key $SK^{(1)} = \lambda_1, SK^{(2)} = \lambda_2$. KGC also initializes a keyword set $\mathcal{W}$ which contains μ keywords as the universal keyword set. Then KGC publishes the public parameter $PP = (\mathcal{W}, \mu, N, g)$, sends $SK_{CP} = SK^{(1)} = \lambda_1$ to CP, the secret key $SK_{IS} = SK^{(2)} = \lambda_2$ to IS, and keeps $SK = \lambda$ secret.
2) Each DP runs `KeyGen` to generate its own key pair of the public key $pk_{DP} = (N, g, h_{DP})$ and the corresponding weak private key $sk_{DP} = \theta_{DP}$, then publishes pk_{DP} and keeps sk_{DP} secret. Each RU runs `KeyGen` to generate its own key pair of the public key $pk_{RU} = (N, g, h_{RU})$ and the corresponding weak private key $sk_{RU} = \theta_{RU}$, then publishes pk_{RU} and keeps sk_{RU} secret.

Store. Given DP's public key $PK_{DP} = pk_{DP}$ and a document keyword set $\mathcal{W}_T \subseteq \mathcal{W}$ to be processed, due to $\mathcal{W}$, DP computes the corresponding searchable ciphertext $T \in \{0, \cdots, 2^\mu - 1\}$ whose binary representation is $(T_{\mu-1}, \cdots, T_0)$ in the same way that we mentioned in Section 2.1, runs `Enc` to get the encrypted searchable ciphertext $[\![T]\!]_{pk_{DP}}$ and sends it to CP.

Trapdoor. Given RU's public key $PK_{RU} = pk_{RU}$ and the keyword set $\mathcal{W}_t$ of interest, due to $\mathcal{W}$, RU extracts the keyword set $\mathcal{W}_t \subseteq \mathcal{W}$ of interest and computes the corresponding trapdoor $t \in \{0, \cdots, 2^\mu - 1\}$ whose binary representation is $(t_{\mu-1}, \cdots, t_0)$ in the same way that we mentioned in Section 2.1, runs `Enc` to get the encrypted trapdoor $[\![t]\!]_{pk_{RU}}$ and sends it to CP.

Test. Given DP's public key $PK_{DP} = pk_{DP}$, RU's public key $PK_{RU} = pk_{RU}$, CP's secret key $SK_{CP} = SK^{(1)}$, IS's secret key $SK_{IS} = SK^{(2)}$, RU's encrypted trapdoor $[\![t]\!]_{pk_{RU}}$

and an encrypted searchable ciphertext $[\![T]\!]_{pk_{DP}}$, CP and IS perform the following 4 steps which will be later elaborated on in Algorithm 3, 4, 5 and 6:

`Step-1`: Given $[\![t]\!]_{pk_{RU}}$ and $[\![T]\!]_{pk_{DP}}$, CP jointly computes the ciphertext of each bit of $\neg T$ and t, i.e., $[\![\neg T_i]\!]_{pk_{DP}}$ and $[\![t_i]\!]_{pk_{RU}}$ for $i \in \{0, \cdots, \mu-1\}$ with IS.

`Step-2`: Given $[\![\neg T_i]\!]_{pk_{DP}}$ and $[\![t_i]\!]_{pk_{RU}}$ for $i \in \{0, \cdots, \mu-1\}$, CP jointly computes the ciphertext of each c_i and d_i mentioned in our subset deciding mechanism, i.e., $[\![c_i]\!]_{pk_{RU}}$ and $[\![d_i]\!]_{pk_{RU}}$ with IS.

`Step-3`: Given $[\![d_i]\!]_{pk_{RU}}$, CP jointly computes the multiplication of each d_i for $i \in \{0, \cdots, \mu-1\}$ and the value after randomization, i.e., $[\![R]\!]_{pk_{RU}}$ and $[\![f]\!]_{pk_{RU}}$ with IS.

`Step-4`: Given $[\![f]\!]_{pk_{RU}}$, RU outputs 0 which means T does not match t, RU does not ask CP for the current document; outputs 1 which means T matches t, RU asks CP for the current document.

Algorithm 3 `Step-1`

Input: $[\![t]\!]_{pk_{RU}}$, $[\![T]\!]_{pk_{DP}}$, pk_{DP}, pk_{RU}, sk_{CP}, sk_{IS}.
Output: $[\![\neg T_i]\!]_{pk_{DP}}$ and $[\![t_i]\!]_{pk_{RU}}$ for $i \in \{0, \cdots, \mu-1\}$.

1: CP computes:

$$[\![SUM]\!]_{pk_{DP}} = [\![2^{\mu} - 1]\!]_{pk_{DP}},$$
$$[\![\neg T]\!]_{pk_{DP}} = [\![SUM]\!]_{pk_{DP}} \cdot ([\![T]\!]_{pk_{DP}})^{N-1},$$

2: CP runs `SBD` protocol with IS:

$$\texttt{SBD}([\![\neg T]\!]_{pk_{DP}}) \rightarrow ([\![\neg T_{\mu-1}]\!]_{pk_{DP}}, \cdots, [\![\neg T_0]\!]_{pk_{DP}}),$$
$$\texttt{SBD}([\![t]\!]_{pk_{RU}}) \rightarrow ([\![t_{\mu-1}]\!]_{pk_{RU}}, \cdots, [\![t_0]\!]_{pk_{RU}}).$$

Algorithm 4 `Step-2`

Input: $[\![\neg T_i]\!]_{pk_{DP}}$ and $[\![t_i]\!]_{pk_{RU}}$ for $i \in \{0, \cdots, \mu-1\}$, pk_{DP}, pk_{RU}, sk_{CP}, sk_{IS}.
Output: $[\![d_i]\!]_{pk_{RU}}$.

1: CP runs `SAD` protocol with IS for $i \in \{0, \cdots, \mu-1\}$:

$$\texttt{SAD}([\![\neg T_i]\!]_{pk_{DP}}, [\![t_i]\!]_{pk_{RU}}) \rightarrow [\![c_i]\!]_{pk_{RU}},$$

2: CP computes for $i \in \{0, \cdots, \mu-1\}$:

$$[\![d_i]\!]_{pk_{RU}} = [\![2]\!]_{pk_{RU}} \cdot ([\![c_i]\!]_{pk_{RU}})^{N-1}.$$

Algorithm 5 `Step-3`

Input: $[\![d_i]\!]_{pk_{RU}}$ for $i \in \{0, \cdots, \mu-1\}$, pk_{RU}, sk_{CP}, sk_{IS}.
Output: $[\![f]\!]_{pk_{RU}}$.

1: CP sets $[\![R]\!]_{pk_{RU}} = [\![1]\!]_{pk_{RU}}$ and $i = 0$,
2: CP runs `SMD` protocol with IS:
3: **while** $i < \mu$ **do**
4: $\quad \texttt{SMD}([\![R]\!]_{pk_{RU}}, [\![d_i]\!]_{pk_{RU}}) \rightarrow [\![R]\!]_{pk_{RU}}$,
5: **end while**
6: CP chooses $r \xleftarrow{R} \mathbb{Z}_N^*$ s.t. $gcd(r, N) = 1$ and computes:

$$[\![f]\!]_{pk_{RU}} = [\![r \cdot R]\!]_{pk_{RU}} = [\![R]\!]^r_{pk_{RU}}.$$

Algorithm 6 `Step-4`

Input: $[\![f]\!]_{pk_{RU}}$, sk_{RU}.
Output: 0 or 1.

1: RU decrypts $[\![f]\!]_{pk_{RU}}$ with its weak private key sk_{RU}:

$$D_{sk_{RU}}([\![f]\!]_{pk_{RU}}) \rightarrow f.$$

2: **if** $f = 0$ **then**
3: $\quad$ **return** 0.
4: **else**
5: $\quad$ **return** 1.
6: **end if**

6 Security

Since we have proved that the secure realization of SE-EPOM according to Definition 4 implies the security requirements of searchable ciphertext indistinguishability and search pattern privacy, in this section we prove the security of our scheme based on the security model defined in Definition 4 so that we can directly obtain searchable ciphertext indistinguishability and search pattern privacy of our scheme.

6.1 Security of Protocol

Theorem 3. *The protocol described in Section 5.3 securely realize SE-EPOM according to Definition 4 with adversaries* $\mathcal{A} = (\mathcal{A}_{D_a}, \mathcal{A}_{S_1}, \mathcal{A}_{S_2})$ *present.*

Proof. $\mathbf{Sim}_{D_a}$ receives x as input and simulates $\mathcal{A}_{D_a}$ as follows: it computes $[\![T]\!]_{pk_{DP}} \leftarrow Enc_{pk_a}(T)$, returns $[\![T]\!]_{pk_{DP}}$ to $\mathcal{A}_{D_a}$ and outputs $\mathcal{A}_{D_a}$'s entire view. The view of $\mathcal{A}_{D_a}$ consists of $[\![T]\!]_{pk_{DP}}$. The view of $\mathcal{A}_{D_a}$ in both the real world and the ideal world executions are indistinguishable due to the semantic security of DT-PKC (See Section VI-A Theorem 1 in [25] for details).

$\mathbf{Sim}_{S_1}$ simulates $\mathcal{A}_{S_1}$ as follows: it generates (fictious) encryptions of inputs $[\![\hat{T}]\!]_{pk_{DP}}, [\![\hat{t}]\!]_{pk_{RU}}$ by running $Enc_{pk_{DP}}(\hat{T}), Enc_{pk_{RU}}(\hat{t})$ on randomly chosen $\hat{T}, \hat{t} \in \{0, \cdots 2^{\mu} - 1\}$, computes $[\![SUM]\!]_{pk_{DP}}, [\![\neg\hat{T}]\!]_{pk_{DP}}$. It then generates (fictious) intermediate values and encryptions of $[\![\neg\hat{T}_i]\!]_{PK_{DP}}, [\![\hat{t}_i]\!]_{PK_{RU}}$ for $i \in \{0, \cdots, \mu-1\}$ of $\texttt{SBD}(\cdot)$ according to $\hat{T}, \hat{t}$, then computes intermediate values in the same way of Proof of Theorem 3 Section VI-C in [25], $[\![\hat{c}_i]\!]_{PK_{RU}}$ and $[\![\hat{d}_i]\!]_{PK_{RU}}$ based on according to $\hat{T}, \hat{t}$ for $i \in \{0, \cdots, \mu-1\}$. It computes intermediate values of $\texttt{SMD}(\cdot,\cdot)$ and (fictious) encryption of the product $[\![\hat{R}]\!]_{PK_{RU}}$ according to all $\hat{d}_i$. It then computes (fictious) $[\![\hat{f}]\!]_{pk_{RU}}$ based on the randomly chosen $\hat{r}$ and $[\![\hat{R}]\!]_{PK_{RU}}$. $\mathbf{Sim}_{S_1}$ sends encryptions of all the intermediate values in execution to $\mathcal{A}_{S_1}$. If $\mathcal{A}_{S_1}$ returns $\perp$, $\mathbf{Sim}_{S_1}$ returns $\perp$. The view of $\mathcal{A}_{S_1}$ consists of the encrypted values it creates. In both real and the ideal world, it receives the encryptions of $\hat{R}, \hat{f}$. The views of $\mathcal{A}_{S_1}$ in both the real and the ideal world executions are indistinguishable, guaranteed by the fact that DP is honest and the semantic security of DT-PKC.

$\mathbf{Sim}_{S_2}$ simulates $\mathcal{A}_{S_2}$ as follows: it generates the (fictious) encryption of intermediate values of protocols $\texttt{SBD}(\cdot)$, $\texttt{SAD}(\cdot,\cdot)$ and $\texttt{SMD}(\cdot,\cdot)$ by computing on and encrypting randomly chosen numbers in the same way of Proof of Theorem

3 Section VI-C in [25]. $\mathbf{Sim}_{S_2}$ sends these encryptions of intermediate values to $\mathcal{A}_{S_2}$. If $\mathcal{A}_{S_2}$ returns $\perp$, $\mathbf{Sim}_{S_2}$ returns $\perp$. The view of $\mathcal{A}_{S_2}$ consists of the encrypted values it creates. In both the real and the ideal world, it receives the intermediate values in execution. The views of $\mathcal{A}_{S_2}$ in both the real world and the ideal world executions are indistinguishable, guaranteed by the fact that DP is honest and the semantic security of DT-PKC.

6.2 Ciphertext Indistinguishability, Trapdoor Privacy and Search Pattern Privacy

Obviously, due to Theorem 1, Theorem 2 and Theorem 3, we can come to the following conclusion:

Corollary 1. *Our SE-EPOM scheme satisfies searchable ciphertext indistinguishability, trapdoor privacy, and search pattern privacy.*

6.3 Access Pattern Privacy

Access pattern indicates the matching document identifiers revealed in a searching operation. Access pattern leakage generally comes from the following two aspects: the output of the test algorithm returned by the CP (within the search procedure), and documents downloaded by the user after obtaining the searching result (outside the searching procedure). Many searchable encryption schemes reveal access pattern in both aspects. In particular, in most of the existing searchable encryption schemes, the result of the test algorithm obtained by the CP directly indicates whether a document contains the keyword(s) or not. In contrast, our scheme guards the testing result via a semantically secure encryption so that the CP cannot learn the matching document identifiers corresponding to a search query. Our Trapdoor Privacy model (Definition 2) has also captured this feature: the adversary can generate a searchable ciphertext using one of the challenging keywords, if the adversary can tell the matching result between the searchable ciphertext and the challenging trapdoor, then the adversary can win the game. Therefore, our scheme achieves access pattern privacy within the search procedure.

On the other hand, access pattern can also be leaked to the CP when the user later retrieves the documents corresponding to the searching result. This is outside the scope of the searching procedure and should be handled using other countermeasures. For example, the user may batch the searching results when accessing the documents or swap the access orders based on the results of multiple searching requests. The user may also access some irrelevant documents on top of the target ones to confuse the storage server. We should note that these measures can be applied to any searchable encryption scheme that can protect the test result within the searching procedure.

7 Performance Evaluation

7.1 Experimental Analysis

Our SE-EPOM is simulated with Java. The communication is implemented in a distributed system architecture as depicted in Figure 1. KGC, CP and DP are deployed on PCs with 3.4GHz eight-core processors and 8GB RAM, and IS's and RU are deployed on PCs with 2.4GHz four-core processors and 4G RAM. For the compared schemes [10], [11] which support multi-keyword search, the server side is deployed on a PC with 3.4GHz eight-core processors and 8GB RAM, and the client side is deployed on a PC with 2.4GHz four-core processors and 4G RAM. All experiments are conducted under 80-bit security where our parameter N is a 1024 bit-length positive integer. The experimental results are rendered when the maximal number of keywords in the system is 5, 10, 15 or 20. We evaluate three schemes on both computational cost and communication cost (number of bits) in Figure 2.

Figure 2a illustrates the comparison on the time cost of store (build index) algorithm. [10] uses polynomial interpolation to derive ciphertext based on keywords, therefore the time consumption is quadratic with the maximal keywords in the system and is around 1886.75 ms when there are 20 keywords allowed in the system. The time cost of [11] is linear to the number of the maximal keywords and is about 36.6 ms for 20 keywords. In comparison, our cost is only 15.4 ms to generate a ciphertext and is independent to the number of keywords.

Figure 2b demonstrates the performance of trapdoor computation. The time cost of [10] is linear to the maximal number of keywords in the system and is around 163.4 ms when there are 20 keywords. [11] uses polynomial interpolation to calculate trapdoor which results in a quadratic curve with the maximal number of keywords and needs roughly 2913 ms to generate a trapdoor for 20 keywords. In contrast, ours is constant and only about 23 ms.

Figure 2c shows the contrast of testing process. The time cost of [10], [11] seems to be better than ours since our testing requires rounds of interaction and network latency heavily contributes to the time consumption. However, the performance of [10], [11] degrades with the increase of the number of keywords, while ours is nearly constant due to the distributed architecture where testing is undertaken by multiple parallel IS's. As the number of keywords increases, the difference in response time will become smaller and less noticeable.

Figure 2d shows the transcript (public parameters and keys) size in setup and key generation processes. Ours is not as small as that of the compared schemes [10], [11] but this will not disadvantage our scheme too much due to the fact that setup and key generation are both one-time execution. Moreover, their transcript size keeps increasing with the maximal number of keywords. [10]'s public parameter size and [11]'s user key size are both linear to the number of the maximal number of keywords in the system, thus their size will surpass our constant transcript size when there are more keywords.

Figure 2e and 2f demonstrate the comparisons of the trapdoor size and ciphertext size, respectively. Both [10], [11] derive multiple transcripts in trapdoor generation and store, and their sizes heavily depend on the maximal number of keywords in the system. Our trapdoor and ciphertext are both a homomorphic ciphertext, which is constant and independent of the number of keywords. To be noted, Figure 2f only corresponds to the scenario of one reader in the system. To enable multiple readers, the size of ciphertext of [10], [11] will accordingly multiply as the number of readers,

resulting in far worse performance. While in our scheme, one ciphertext supports all users' access.

Though our testing is not as efficient as the compared ones when the number of keywords is small, our computational cost of store and trapdoor generation, and communication cost of trapdoor and ciphertext are much better than those of [10], [11]. In addition, our scheme achieves better functionality and security as shown in Table 3.

7.2 Functionality Comparison with Existing Schemes

A functionality comparison between our scheme and the existing ones is presented in Table 3.

Researches have already reached a consensus on the classification of searchable encryption schemes, which can be divided into the following four types. Single-reader/single-writer setting [9] only allows the writer (owner) itself to launch queries and refers to symmetric searchable schemes where the writer and the reader are the same party. Single-writer/multi-reader setting [16] refers to schemes that enable multiple users to search on encrypted data produced by a particular writer while multi-writer/single-reader setting [2], [17] means the opposite. Multi-writer/multi-reader setting supports each user to encrypt and upload data and search on stored encrypted data from all the users as well.

For single-keyword schemes [2], [17], to query multiple keywords, the user needs to prepare as many trapdoors as the number of keywords in the query. Similarly, the number of searchable ciphertexts stored with the document is also proportional to the number of keywords included in the current (target) document $|\mathcal{W}_{id}|$. For multi-keyword searchable encryption schemes, the size of the trapdoor and the ciphertext is an important indicator of efficiency. The size of both trapdoor and ciphertext of [9] is also linear to $|\mathcal{W}_{id}|$, resulting in that the user needs to keep in mind the number of keywords of the document that it intends to search. The size of both trapdoor and ciphertext of [10], [11] increases with the maximal number of keywords $|\mathcal{W}|$ in the system. [16] builds a connection between the document and its each contained keyword, hence the ciphertext size is linear to $|\mathcal{W}_{id}|$. Their search trapdoor is to generate the pointers of possible connections between each keyword in query and each identifier of all stored documents. Hence, the number of such connections is linear to both the number of keywords in the query $|Q|$ and the number of documents in storage $|DB|$. Both our trapdoor and ciphertext are the homomorphic encryption of a decimal integer whose length is irrelevant to the number of keywords. Each document only needs one searchable ciphertext to represent all its underlying keywords, hence the number of searchable ciphertexts stored in the cloud is $|DB|$. In addition, our scheme essentially supports multi-reader access, while [10], [11] need to prepare one separate ciphertext for each reader to enable multi-reader access, i.e., $l \cdot |DB|$ ciphertexts where l is the number of readers.

Since the intended reader's public key is known in traditional PEKS schemes, those schemes inevitably suffer from KGA. [10] utilizes Type-3 bilinear map to lower KGA success probability. [9] is actually a symmetric scheme where both ciphertext and trapdoor generation need the user's secret key, therefore attackers other than the user are unable to generate ciphertexts for the test. [11] delegates the testing ability to the server by taking the server's secret key as input to prevent outside attackers from freely testing but it is still vulnerable to internal KGA from the curious server. [16] is a single-writer/multi-reader scheme which means only the writer can generate ciphertexts.

Our scheme takes secret keys of CP and IS's as input for testing so that attackers including the CP or a set of IS's are unable to freely test and learn anything about the underlying keywords, thereby resisting KGA. In summary, the performance of our solution is better than related work.

8 Conclusion and Open Problems

In this paper, we introduced the notion of SE-EPOM and formalized its security definitions. Subsequently, we designed a concrete SE-EPOM scheme in a distributed architecture with our novel subset decision mechanism and proved it satisfies our proposed security requirements. Besides, the scheme possesses attractive features including supporting multi-keyword search, constant size trapdoor and ciphertext, hiding search pattern and access pattern during searching, and enabling the multi-writer/multi-reader setting. Finally, the evaluation on compared schemes and our scheme shows that the overall performance of our distributed SE-EPOM scheme outperforms other solutions. We leave designing a secure SE-EPOM with fewer rounds of communication between CP and IS's as our future work.

Acknowledgement

We thank the Associate Editor and anonymous Reviewers for their insightful and constructive comments on this work.

References

[1] "Forecast number of personal cloud storage consumers/users worldwide from 2014 to 2020 (in millions)," https://www.statista.com/statistics/638593/worldwide-data-center-storage-capacity-cloud-vs-traditional/, accessed August 30, 2018.

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

[3] 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 *Workshop on Secure Data Management*, 2006, pp. 75–83.

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

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

[6] E.-J. Goh *et al.*, "Secure indexes." *IACR Cryptology ePrint Archive*, vol. 2003, p. 216, 2003.

[7] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: improved definitions and efficient constructions," *Journal of Computer Security*, vol. 19, no. 5, pp. 895–934, 2011.

[8] K. Kurosawa and Y. Ohtaki, "Uc-secure searchable symmetric encryption," in *International Conference on Financial Cryptography and Data Security*, 2012, pp. 285–298.

[9] P. Wang, H. Wang, and J. Pieprzyk, "Keyword field-free conjunctive keyword searches on encrypted data and extension for dynamic groups," in *International conference on cryptology and network security*, 2008, pp. 178–195.

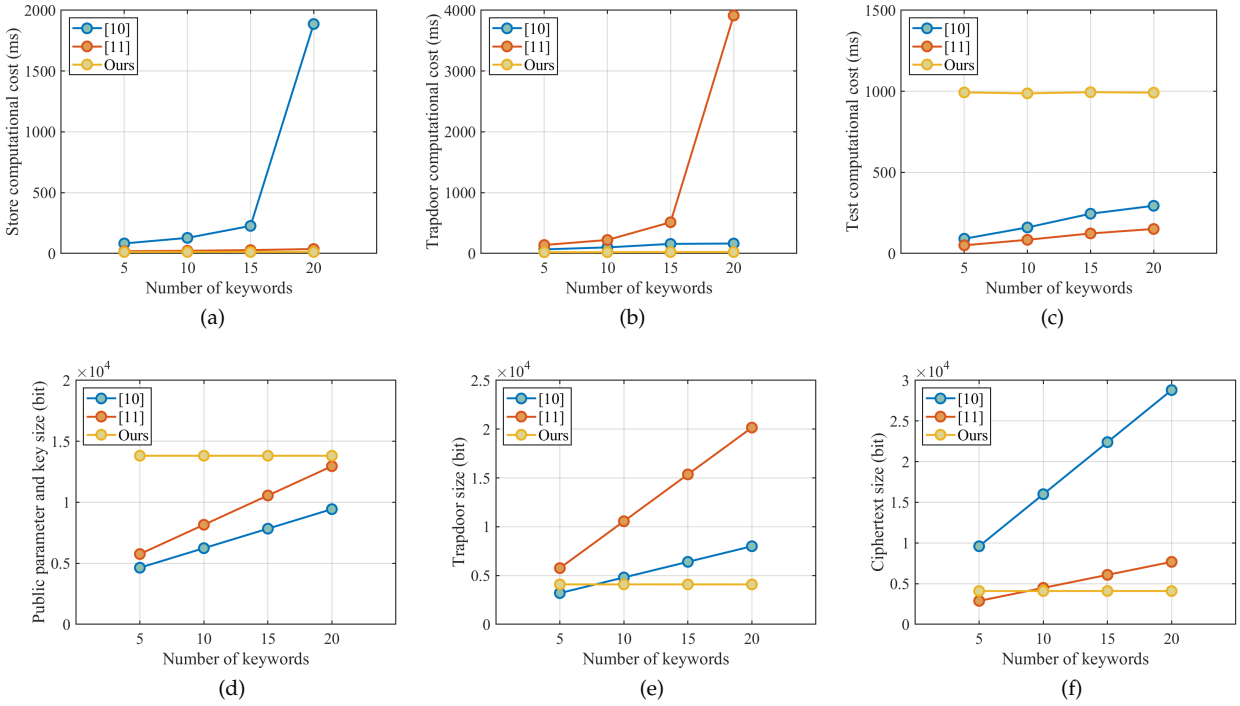


Fig. 2: Experimental Results

TABLE 3: Functionality Comparison

	Multi-reader	Multi-writer	Multi-keyword	Server KGA resistance	# of ciphertexts	Trapdoor size	Ciphertext size
[2]	✗	✓	✗	✗	$\sum_{w\in\mathcal{W}} \|DB[w]\|$	$\mathcal{O}(\|Q\|)$	$\mathcal{O}(\|\mathcal{W}_{id}\|)$
[17]	✗	✓	✗	✗	$\sum_{w\in\mathcal{W}} \|DB[w]\|$	$\mathcal{O}(\|Q\|)$	$\mathcal{O}(\|\mathcal{W}_{id}\|)$
[9]	✗	✗	✓	✓	$\|DB\|$	$\mathcal{O}(\|\mathcal{W}_{id}\|)$	$\mathcal{O}(\|\mathcal{W}_{id}\|)$
[10]	✗	✓	✓	✗	$\|DB\|$	$\mathcal{O}(\|\mathcal{W}\|)$	$\mathcal{O}(\|\mathcal{W}\|)$
[11]	✗	✓	✓	✗	$\|DB\|$	$\mathcal{O}(\|\mathcal{W}\|)$	$\mathcal{O}(\|\mathcal{W}\|)$
[16]	✓	✗	✓	✓	$\sum_{w\in\mathcal{W}} \|DB[w]\|$	$\mathcal{O}(\|DB\| * \|Q\|)$	$\mathcal{O}(\|\mathcal{W}_{id}\|)$
Ours	✓	✓	✓	✓	$\|DB\|$	$\mathcal{O}(1)$	$\mathcal{O}(1)$

# of ciphertexts: the number of searchable ciphertexts in one reader setting. Trapdoor size: the size of trapdoor when launching a query of multiple keywords. Ciphertext size: the size of the searchable ciphertext when the document includes multiple keywords.

[10] B. Zhang and F. Zhang, "An efficient public key encryption with conjunctive-subset keywords search," *Journal of Network and Computer Applications*, vol. 34, no. 1, pp. 262–267, 2011.

[11] K. Huang, R. Tso, and Y.-C. Chen, "Somewhat semantic secure public key encryption with filtered-equality-test in the standard model and its extension to searchable encryption," *Journal of Computer and System Sciences*, vol. 89, pp. 400–409, 2017.

[12] G. Wang, C. Liu, Y. Dong, P. Han, H. Pan, and B. Fang, "Idcrypt: A multi-user searchable symmetric encryption scheme for cloud applications," *IEEE Access*, vol. 6, pp. 2908–2921, 2018.

[13] X. Liu, G. Yang, Y. Mu, and R. Deng, "Multi-user verifiable searchable symmetric encryption for cloud storage," *IEEE Transactions on Dependable and Secure Computing*, 2018.

[14] A. Fiat and M. Naor, "Broadcast encryption," in *Annual International Cryptology Conference*, 1993, pp. 480–491.

[15] D. Cash, S. Jarecki, C. Jutla, H. Krawczyk, M.-C. Roşu, and M. Steiner, "Highly-scalable searchable symmetric encryption with support for boolean queries," in *Annual Cryptology Conference*, 2013, pp. 353–373.

[16] S.-F. 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 *European symposium on research in computer security*, 2016, pp. 154–172.

[17] L. Xu, X. Yuan, R. Steinfeld, C. Wang, and C. Xu, "Multi-writer searchable encryption: an lwe-based realization and implementation," in *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security*, 2019, pp. 122–133.

[18] W.-C. Yau, S.-H. Heng, and B.-M. Goi, "Off-line keyword guessing attacks on recent public key encryption with keyword search schemes," in *International Conference on Autonomic and Trusted Computing*, 2008, pp. 100–105.

[19] 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 Transactions on computers*, vol. 62, no. 11, pp. 2266–2277, 2012.

[20] S. Kamara, C. Papamanthou, and T. Roeder, "Dynamic searchable symmetric encryption," in *Proceedings of the 2012 ACM conference on Computer and communications security*, 2012, pp. 965–976.

[21] R. Chen, Y. Mu, G. Yang, F. Guo, X. Huang, X. Wang, and Y. Wang, "Server-aided public key encryption with keyword search," *IEEE Trans. Information Forensics and Security*, vol. 11, no. 12, pp. 2833–2842, 2016.

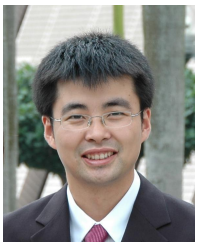
[22] R. Chen, Y. Mu, G. Yang, F. Guo, and X. Wang, "Dual-server public-key encryption with keyword search for secure cloud storage," *IEEE Trans. Information Forensics and Security*, vol. 11, no. 4, pp. 789–798, 2016.

[23] O. Goldreich and R. Ostrovsky, "Software protection and simulation on oblivious rams," *Journal of the ACM (JACM)*, vol. 43, no. 3, pp. 431–473, 1996.

[24] B. K. Samanthula, H. Chun, and W. Jiang, "An efficient and proba-

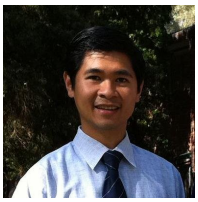
bilistic secure bit-decomposition," in *Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security*, 2013, pp. 541–546.
[25] X. Liu, R. H. Deng, K.-K. R. Choo, and J. Weng, "An efficient privacy-preserving outsourced calculation toolkit with multiple keys," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 11, p. 2401, 2016.
[26] E. Bresson, D. Catalano, and D. Pointcheval, "A simple public-key cryptosystem with a double trapdoor decryption mechanism and its applications," in *Advances in Cryptology - ASIACRYPT 2003, 9th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, November 30 - December 4, 2003, Proceedings*, 2003, pp. 37–54.
[27] P.-A. Fouque and D. Pointcheval, "Threshold cryptosystems secure against chosen-ciphertext attacks," in *International Conference on the Theory and Application of Cryptology and Information Security*, 2001, pp. 351–368.
[28] C. Gentry, *A fully homomorphic encryption scheme*, 2009, vol. 20, no. 09.
[29] S. Kamara, P. Mohassel, and M. Raykova, "Outsourcing multi-party computation." *IACR Cryptology ePrint Archive*, vol. 2011, p. 272, 2011.
[30] M. Nishioka, "Perfect keyword privacy in peks systems," in *International Conference on Provable Security*, 2012, pp. 175–192.

Xueqiao Liu received the B.S. degree from Hefei University of Technology, China, in 2011 and the M.S. degree from Jinan University, China, in 2014. She is currently pursuing the Ph.D. degree in the School of Computing and Information Technology, University of Wollongong, Australia. Her major research interests include cryptography, data security and privacy in cloud computing and network security.



Guomin Yang received his Ph.D. degree in computer science from the City University of Hong Kong in 2009. He was a Research Scientist with the Temasek Laboratories, National University of Singapore from 2009 to 2012. He is currently an Associate Professor at the School of Computing and Information Technology, University of Wollongong. His research mainly focuses on applied cryptography and network security. He received a prestigious ARC DECRA Fellowship in 2015.

Willy Susilo received the PhD degree in computer science from the University of Wollongong, Australia. He is currently a professor and the head of the School of Computing and Information Technology, University of Wollongong, Australia. He is also the director of the Centre for Computer and Information Security Research, University of Wollongong. His main research interests include cloud security, cryptography, and information security. He has received the prestigious ARC Future Fellowship from the Australian Research Council. He has served as a program committee member in major international conferences.



Joseph Tonien obtained his PhD in Information Security at the University of Wollongong in 2005 and has nearly a decade of experience in enterprise software development. He was a recipient of a research grant and a postdoctoral fellowship from Australian Research Council. His main research interest include cryptography and number theory. He is a member of the Institute of Cybersecurity and Cryptology at the University of Wollongong.

Ximeng Liu (S'13-M'16) received the B.Sc. degree in electronic engineering from Xidian University, Xi'an, China, in 2010 and the Ph.D. degree in Cryptography from Xidian University, China, in 2015. Now he is the full professor in the College of Mathematics and Computer Science, Fuzhou University. Also, he is a research fellow at the School of Information System, Singapore Management University, Singapore. He has published more than 100 papers on the topics of cloud security and big data security including papers in IEEE TOC, IEEE TII, IEEE TDSC, IEEE TSC, IEEE IoT Journal, and so on. He awards "Minjiang Scholars" Distinguished Professor, "Qishan Scholars" in Fuzhou University, and ACM SIGSAC China Rising Star Award (2018). His research interests include cloud security, applied cryptography and big data security. He is a member of the IEEE, ACM, CCF.

Jian Shen received the M.E. and Ph.D. degrees in computer science from Chosun University, South Korea, in 2009 and 2012, respectively. Since 2012, he has been a Professor with the Nanjing University of Information Science and Technology, Nanjing, China. His research interests include public cryptography, cloud computing and security, data auditing and sharing, and information security systems.