Security and Safety, Vol. 4, 2024022 (2025)
https://doi.org/10.1051/sands/2024022
sands.edpsciences.org

Security and Safety of Data in Cloud Computing

Research Article

Open Access

Other Fields

Efficient verifiable searchable encryption with search and access pattern privacy

Axin Wu[1,*], Dengguo Feng[1], Min Zhang[2], Jialin Chi[2], and Yinghui Zhang[3]

[1] *State Key Laboratory of Cryptology, Beijing 100878, China*
[2] *TCA Laboratory, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China*
[3] *National Engineering Research Center for Secured Wireless (NERCSW), School of Cyberspace Security, Xi'an University of Posts & Telecommunications, Xi'an 710121, China*

Received: 16 October 2024 / Revised: 8 November 2024 / Accepted: 17 December 2024 / Published online: 30 January 2025

Abstract Searchable encryption (SE) enables data users to securely search encrypted data stored in untrusted cloud servers. However, most SE schemes allow for leakages of access and search patterns to maximize efficiency and functionality. Recent attacks have shown that adversaries can recover query keywords with prior knowledge of the database by exploiting these leakages. Unfortunately, the existing schemes that protect access and search patterns result in frequent communications and high computational costs. Furthermore, complex calculation processes also raise challenges for verifying search results. To address these concerns, we first design an efficient conjunctive SE scheme with search and access pattern privacy using private set intersection. In the proposed scheme, we utilize random numbers to obfuscate the values of polynomials and randomly divide the results into two parts, which simplifies the search process, improves search efficiency, and eliminates the need for time-consuming ciphertext multiplication operations. We also extend this scheme to support search result verifiability. Specifically, by embedding a random number as the root of the return polynomial, we achieve verifiability of search results. Furthermore, we prove the security of both schemes employing the simulation-based method. Finally, we implement the schemes in a real database and thorough performance analyses demonstrate their efficiency.

Keywords Searchable encryption, Private set intersection, Search and access pattern, Verifiability, Conjunctive search

Citation Wu A, Feng D and Zhang M et al. Efficient verifiable searchable encryption with search and access pattern privacy. Security and Safety 2025; **4**: 2024022. https://doi.org/10.1051/sands/2024022

1 Introduction

Searchable encryption [1] (SE) ensures the confidentiality of files while enabling search functionality over encrypted data. Its abilities to perform searches over encrypted data have made SE widely used in cloud storage and encrypted databases. In this context, SE is imperative in various applications, including medical cloud systems and industrial internet of things. To cater the various application scenarios, a wide range of SE schemes [2–5] have been proposed. These schemes make different trade-offs among efficiency, functionality, and security depending on specific requirements. For example, most SE schemes [3, 6] may result in access pattern leakage, where the document identifiers in a query are revealed, and search pattern leakage, which exposes whether different queries are related to the same keyword, to achieve higher efficiency and enhanced functionality. However, existing attacks [7–11] have demonstrated

* Corresponding author (email: waxinsec@163.com)

Table 1. Functionality comparison of pattern hidden searchable encryption

Schemes	Verification	Access pattern	Search pattern	Conjunctive query	No-rebuild	Techniques
INFSCI'14 [7]	×	×	√	×	√	GBC
CRYPTO'16 [12]	×	√	√	×	×	ORAM
CCS'18 [13]	×	√	×	√	√	SHVE
INFOCOM'18 [14]	×	√	×	×	√	DF
TSC'20 [15]	×	×	√	√	√	k-anonymity
TSC'22 [16]	×	√	√	√	√	PSI
TIFS'20 [17]	×	√	√	×	×	RE
NDSS'21 [18]	×	√	√	×	√	DP
IoT-J'24 [19]	√	√	√	×	√	OPE
Our scheme	√	√	√	√	√	PSI

×: Not support the functionality. √: Support the functionality. GBC: Grouping-based construction. SHVE: Symmetric-key hidden vector encryption. DP: Differential privacy. PSI: Private set intersection. RE: Re-Encryption cryptosystems. OPE: Oblivious Polynomial Evaluation.

that adversaries can recover the underlying keywords in the keyword trapdoor with a high probability and accuracy, given some prior knowledge of the outsourced database or query subset. This violates the privacy requirements of SE, which is a significant concern. Recent studies [9–11] have also highlighted the potential security threats posed by the disclosure of search and access patterns.

Ensuring privacy for only one of the access pattern or search pattern is insufficient because they are interdependent. Access patterns can be derived directly from search patterns, and deterministic search tokens, as well as the access pattern and response equality pattern, can also generate search patterns. Achieving access and search pattern privacy in a single-round interaction with cloud servers is challenging because the user's access needs to be oblivious during each query. Existing schemes that aim to protect search and access patterns come with additional computation and communication costs. Ideally, the schemes based on Oblivious RAM (ORAM) [12] and Private Information Retrieval (PIR) [20] offer high security guarantees, but they require extensive computation and communication resources, making them impractical for real-world applications. Fortunately, some novel schemes have been proposed aiming to protect the pattern privacy. Originally, researchers [7, 12, 15, 17–19] primarily concentrated on ensuring pattern privacy in single keyword scenarios. Nevertheless, the demands for multi-keyword search requests [21, 22] are frequent. Computing the intersection outcomes of each individual keyword search to obtain multi-keyword search results not only incurs higher computation and communication expenses but can also lead to additional leakage beyond the results. Recently, researchers [13, 16] have proposed some multi-keyword SE schemes with search and access pattern privacy. However, to our knowledge, none of these schemes are capable of verifying the search results in the context of search and access pattern privacy with conjunctive queries. To provide clarity, we summarize these schemes and related solutions in Table 1.

Achieving verifiability of search results in the context of maintaining search and access pattern privacy is both challenging and necessary. Firstly, currently available schemes [16–19] for protecting search and access patterns require more computational and communication burdens. For example, to achieve the above functionalities, Wang *et al.* [16] used a special homomorphic encryption (*i.e.*, additive homomorphic encryption with a double trapdoor decryption mechanism) to simulate the product of two ciphertexts on two non-collusive cloud servers. For one multiplication operation of two ciphertexts, this approach necessitates multiple encryption, homomorphic addition, scalar multiplication, and decryption operations. These processes make it difficult to verify the correctness of the returned results due to complex processes such as the multiple rounds of interactions [17] between entities and the introduction of redundant information [18]. Additionally, implementing pattern privacy requires more resources. In an attempt to conserve computational power and network bandwidths, cloud servers might provide inaccurate or incomplete results. By ensuring the verifiability of results, integrity and correctness can be guaranteed, which consequently increases the trust of data users and reduces the risk of disputes between cloud service providers and data users.

Motivated by the aforementioned considerations, we enhance the efficiency of Wang *et al.*'s scheme [16] and incorporate the feature of search result verifiability. To optimize efficiency, we employ random numbers to blind the polynomial values during the generation of the keyword index and randomly divide

the results into two parts, which are stored on two non-collusive cloud servers, respectively. This procedure can be regarded as a form of secret sharing for the blinded polynomial values. Significantly, our approach avoids the need for homomorphic encrypting algorithms in the keyword index generation, thus eliminating the need for the product of two ciphertexts and the reliance on specialized homomorphic encryption techniques, such as additive homomorphic encryption with a double trapdoor decryption mechanism. As a result, the proposed method also reduces communication costs arising from ciphertext expansion. Through these enhancements, our schemes effectively improve the efficiency of both the keyword index generation and search processes. Furthermore, with respect to additional functionality, we introduce a random number into the keyword trapdoor during its generation process. Subsequently, we verify whether this random number corresponds to the root of the returned polynomial in the received results, and whether the non-colluding cloud servers consistently produce valid results. If the verification is successful, the search process is correct. In summary, our contributions can be summarized as follows:

- Firstly, we present SAP-ECKS, an efficient conjunctive searchable encryption with the search and access pattern privacy, which reduces the computation and communication costs of Wang *et al.*'s scheme [16]. Moreover, in SAP-ECKS, the file identifiers can be easily and rapidly identified.
- Secondly, we extend SAP-ECKS to guarantee the verifiability of search results, resulting in a novel scheme referred to as SAP-VECKS. To our knowledge, SAP-VECKS is the first scheme to consider result verifiability in the context of access and search pattern privacy and conjunctive queries.
- Finally, we demonstrate the security of the proposed schemes using simulation-based methods. Additionally, we have analyzed and evaluated the performance of the schemes from both theoretical and experimental viewpoints. This result shows that our schemes outperform existing alternatives in terms of efficiency and provide enhanced functionalities.

The rest of the paper consists of the following sections. In Section 2, we review the related works. Then, we present the basic knowledge and cryptographic primitives in Section 3. Next, the system model and definition of the proposed schemes are introduced in Section 4. In Section 5, we propose the concrete construction of SAP-ECKS and extend it to SAP-VECKS. Later, the proof and performance analysis are presented in Section 6 and 7 respectively. Finally, we summarize the paper in Section 8.

2 Related works

The concept of SE was originally proposed by Song *et al.* [1]. Owing to its favorable characteristics, it has attracted widespread attention from both academia and industry. Numerous schemes [23–27] have been proposed since then. In practice, an effective SE scheme should meet at least the requirement of sub-linear complexity in relation to the database size. Among the various implementation methods available for SE, the inverted index stands out due to its efficient performance. Curtmola *et al.* [6] first introduced this method, while also providing the formal security definition. To support the updating of outsourced data, Kamara *et al.* [28] presented the first dynamic SE scheme with sub-linear complexity. However, these methods were only focused on single keyword searches. To enable rich search functionalities, Golle *et al.* [29] introduced conjunctive keyword searches, followed by Boolean SE schemes [3, 30] that aimed to further enrich search expressiveness. Nonetheless, one major drawback of these schemes reveals the access or search pattern of keyword searches.

With some prior knowledge of the query subset and outsourced database, scholars have presented a series of attacks by leveraging the access pattern or the search pattern to recover the underlying keywords of search trapdoors. Islam *et al.* [31] leveraged the prior knowledge of the entire database to launch the access-pattern attack for the first time. Later, Cash *et al.* [8, 32] recovered some query keywords by leveraging the prior knowledge about the keyword co-occurrence probability matrix and access patterns of keywords, which employed less prior knowledge based on Islam *et al.*'s method. Zhang *et al.* [9] exploited the injected files that could be associated with previous keyword trapdoors to attack non-forward-secure SE schemes, which could recover underlying keywords in the keyword trapdoor with higher accuracy. Additionally, Liu *et al.* [7] emphasized that the adversary could learn some query keywords by exploiting information about the data user's search habits and search pattern leakages. Recently, some attack schemes against access and search patterns [10, 11, 33, 34] with higher threat have been proposed.

Table 2. Table of notations

Notations	Descriptions
λ	The system security parameter
id	The document identifier
W_i	The keyword set of id_i
$W = \cup_{i=1}^{d} W_i$	The keyword dictionary in the system
w_i	The i-th keyword in a keyword set W
$DB(w_i)$	The document identifier set containing w_i
$L = \max_{w_i \in W} \{\lvert DB(w_i)\rvert\}$	The upper bound of cardinalities of the document identifier set containing w_i
$m = \lvert W\rvert$	The total number of keywords in W
$+_h$	The addition of two ciphertexts
$*_h$	The multiplication of ciphertext and random number
$P_{w_i}(x)$	The L-degree polynomial corresponding to $DB(w_i)$
X_i	The coefficients of polynomial $P_{w_i}(x)$
I	The index matrix
Tr	The keyword trapdoor
φ_i, ψ_i	The random L-degree polynomial
$P(x), \mathcal{P}(x)$	The encrypted search result

To prevent leakage of access patterns and search patterns, various pattern protection schemes have been proposed by scholars. Theoretically, the schemes based on ORAM [12] and PIR [20] can achieve a higher-level security by fully hiding access patterns. However, due to frequent communications and high computational costs, these schemes are impractical for widespread deployment. To alleviate these issues, certain lightweight cryptographic techniques have been applied in other schemes. For example, Liu *et al.* [7] utilized the obfuscation and chameleon hash functions to hide search patterns. Bosch *et al.* [35] protected search patterns across multiple clouds by obfuscating the entire search index before each query through a proxy server. Lai *et al.* [13] employed symmetric-key hidden vector encryption to hide result patterns through multiple-round interactions between data owners and cloud servers. Zheng *et al.* [15], based on k-anonymity, also necessitated redundancy and two rounds of interaction to obscure search patterns. However, these schemes [7, 13, 15, 35] resulted in increased search complexity and decreased search throughput. Some schemes based on differential privacy [14] and padding approaches [36] protect access patterns and strike a balance between security and efficiency. Nevertheless, the aforementioned schemes only protect either search patterns or access patterns, not both simultaneously. Recently, there have been schemes that protect both search pattern and search pattern. Specifically, Song *et al.* [17] employed re-encryption cryptosystems to shuffle and update index entries across multiple cloud servers to fulfill search requests. Shang *et al.* [18] based on differential privacy introduced redundant information to hide access and search patterns, resulting in increased computational and communication costs for users (their experiment took between 27.5 and 1099.1 minutes to respond to a search request). Yang *et al.* [19] further proposed a verifiable solution, however, its implementation relies on the interaction between cloud servers and users, and its functionality is limited to only supporting single keyword retrieval. These schemes introduced redundant information or required scrambling and re-encrypting the keyword index after each search, as well as the interaction between multiple cloud servers, thereby increasing computational costs, communication costs, storage costs, and search delays.

In summary, existing SE schemes that ensure both access and search pattern privacy do not provide functionalities for conjunction keyword search and the verifiability of search results simultaneously.

3 Preliminaries

The notations commonly used in this paper are summarized in Table 2. For simplicity of description, a document is treated as a keyword set. A database is a set of a series of documents, ordered alphabetically by their identifiers.

3.1 Additive homomorphic encryption

The addition property of homomorphic encryption allows calculations on the ciphertexts of messages m_1 and m_2 to obtain the ciphertext of the message $m_1 + m_2$. The additive homomorphic encryption we use here is Paillier encryption scheme [37], of which steps are as follows.

- **KeyGen**: According to the security parameter 1^λ, two distinct random large prime numbers p and q are selected. Then the algorithm sets $N = p \cdot q$. Let u be the least common multiple of $p-1$ and $q-1$, denoted by $u = lcm(p-1, q-1)$. Let $L(x) = \frac{(x-1)}{N}$. $g \in \mathbb{Z}^*_{N^2}$ is randomly selected and ensure that
$$s = (L(g^u \ mod \ N^2))^{-1} \bmod N$$
exists. Finally, the public and private keys of the algorithm are (N, g) and (u, s), respectively.
- **Enc**: When encrypting a message $m \in \mathbb{Z}_N$, the algorithm randomly selects a random number $r \in \mathbb{Z}^*_{N^2}$ and generates the ciphertext by
$$C = E_{pk}(m) = g^m \cdot r^N \bmod N^2.$$
- **Dec**: When decrypting a ciphertext C, the algorithm recovers the plaintext by
$$m = D_{sk}(C) = L(C^u \bmod N^2) \cdot s \bmod N.$$

Additive homomorphic property: Given two ciphertexts $E_{pk}(m_1)$ and $E_{pk}(m_2)$ of plaintexts m_1 and m_2, then the ciphertext of $(m_1 + m_2)$ can be obtained through $E_{pk}(m_1)$ and $E_{pk}(m_2)$. For simplicity, the addition of homomorphic encryption $E_{pk}(m_1 + m_2) = E_{pk}(m_1) \cdot E_{pk}(m_2)$ is defined as

$$C = C_1 +_h C_2.$$

In addition, the ciphertext of $l \cdot m_1$ can be easily obtained by the ciphertext $E_{pk}(m_1)$ and numerical value l. For simplicity, the ciphertext generation of $E_{pk}(l \cdot m_1) = E_{pk}(m_1)^l$ is denoted as

$$C' = l *_h C_1.$$

3.2 Representing sets by polynomials

An element set can be represented by a polynomial [38], which is widely used [39, 40]. Specifically, the elements in a set come from a ring $\mathbb{R}$, and polynomials are represented over the ring $\mathbb{R}$. Besides, the ring $\mathbb{R}$ should be large enough to encode the universe of set elements $\mathbb{U}$. For an element $u_i \in \mathbb{U}$, it is coded as $e_i = u_i || H(u_i)$, where H expresses the cryptographic hash function. Additionally, there is such a relationship $L_\mathbb{R} = L_\mathbb{U} + L_H$, where the bit length of elements in $\mathbb{R}$ is $L_\mathbb{R}$, the bit length of elements in $\mathbb{U}$ is $L_\mathbb{U}$, and the output bit length of hash function H is L_H. Given $s_i \in \mathbb{R}$, if s_i can be parsed into a and $b = H(a)$, and then $s_i = a||b$ holds, we say that s_i is valid. Otherwise, it is invalid. A valid element can be easily distinguished since the probability of randomly selecting an element from $\mathbb{R}$ with the correct structure can be ignored due to the randomness of hash function output. The coefficients of a polynomial are derived from the ring $\mathbb{R}$. That is, the element set S over $\mathbb{R}$ can be expressed as a polynomial

$$\rho(x) = \Pi_{i=1}^{|S|}(x - s_i),$$

where $s_i \in S$.

For two polynomials $\rho^{(A)}$ and $\rho^{(B)}$, representing sets $S^{(A)}$ and $S^{(B)}$ respectively, the union of sets $S^{(A)}$ and $S^{(B)}$ is represented by the polynomial $\rho^{(A)} \cdot \rho^{(B)}$. The greatest common divisor $gcd(\rho^{(A)}, \rho^{(B)})$ represents the intersection of sets $S^{(A)}$ and $S^{(B)}$. For two L-degree polynomials $\rho^{(A)}$ and $\rho^{(B)}$ over $R^L[x]$, and two L-degree random polynomials $\gamma^{(A)}$ and $\gamma^{(B)}$ over $R^L[x]$, Kissner *et al.* [39] proved that

$$\rho^{(A)} \cdot \gamma^{(A)} + \rho^{(B)} \cdot \gamma^{(B)} = \eta \cdot gcd(\rho^{(A)}, \rho^{(B)}),$$

where η expresses a uniformly random $(2L - |S_1 \cap S_2|)$-degree polynomial. Therefore, we can obtain the intersection of sets $S^{(A)}$ and $S^{(B)}$ from $\eta \cdot gcd(\rho^{(A)}, \rho^{(B)})$, and the other elements outside the intersection in $S^{(A)} \cap S^{(B)}$ are not disclosed.

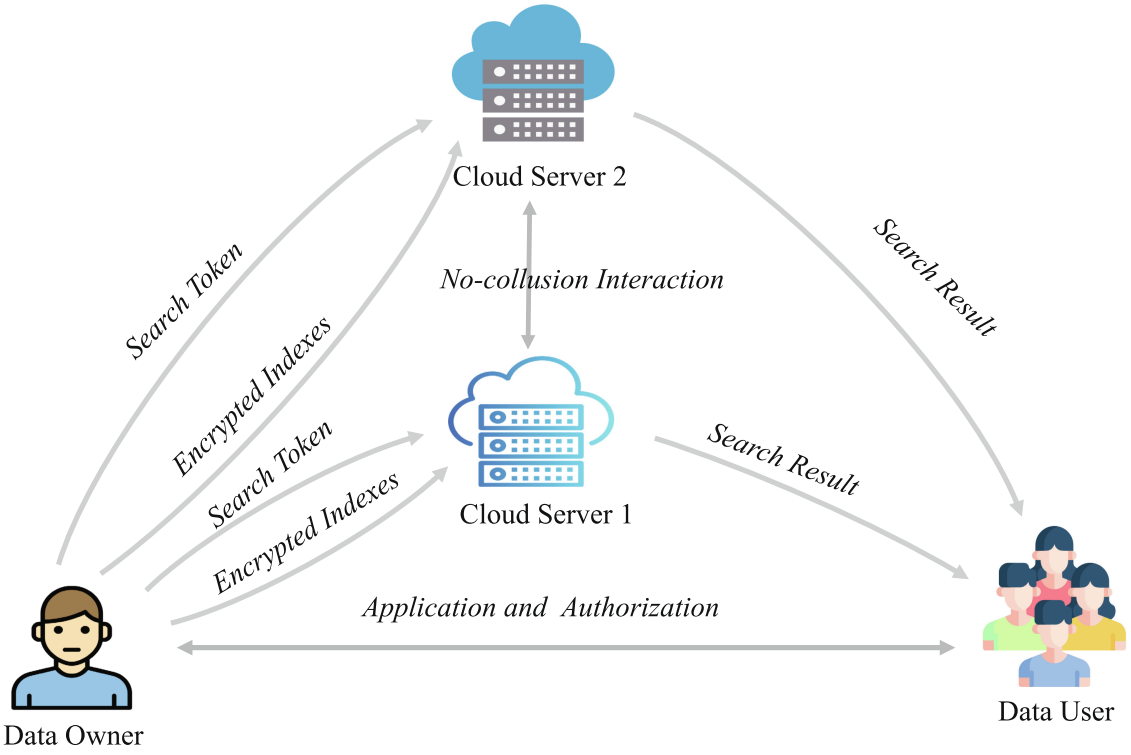

Figure 1. System model

Through the polynomial

$$\beta = \rho^{(A)} \cdot \gamma^{(A)} + \rho^{(B)} \cdot \gamma^{(B)} = \eta \cdot gcd(\rho^{(A)}, \rho^{(B)}),$$

one can calculate roots of the polynomial β and further find the elements in the intersection set $|S^{(A)} \cap S^{(B)}|$ by filtering the invalid elements. When computing the intersection of $S^{(A)}$ and $S^{(B)}$, roots of the polynomial ρ come from either η or $gcd(\rho^{(A)}, \rho^{(B)})$. Since η is a random polynomial, its root will also come from the random number in $\mathbb{R}$. By encoding elements $u_i \in \mathbb{U}$, we can effectively exclude invalid roots.

4 System model and definition

4.1 System model

The system model of the schemes includes the following entities and their respective functionalities (see Figure 1):

- **Data owner**: The data owner possesses a large number of documents that have been reorganized in reverse order. Afterwards, the owner encrypts the corresponding indexes to generate a keyword index matrix before outsourcing it to the cloud servers. Furthermore, the data owner can authorize data users to access the data.
- **Data user**: To access specific documents, a data user must obtain permission from the data owner. The data owner initiates an encryption algorithm, generates a private key and random number, and provides these to the data users for secure access. Afterward, the data owner generates a keyword trapdoor based on the searched keyword set and another random number and uploads it to the cloud servers. The data user receives the search result from the cloud servers and can then verify it.
- **Cloud servers 1 and 2**: The cloud servers 1 and 2 are responsible for storing encrypted documents and the keyword index matrix. When responding to search requests, one server needs to interact with one another. To improve search efficiency, the cloud servers can execute the search process in parallel.

In addition, the cloud servers are not collusive. Two non-collusive cloud servers are usually operated by two competing service providers. To effectively respond to unexpected situations such as server crashes, cloud service providers may implement a series of preventive measures, such as data backup, to enhance the robustness and reliability of the entire system.

The flowchart process of the schemes operates as follows: after generating the encrypted index matrix, the data owner randomly divide it into two parts and upload them to cloud server 1 and cloud server 2 respectively. When a data user needs to search for some files containing a specific set of keywords, he must submit the keyword set to the data owner. Subsequently, the data owner creates a keyword trapdoor on behalf of the data user and uploads it to cloud server 1 and cloud server 2 respectively. Meanwhile, the data owner securely returns the private key to the data user. After receiving the keyword trapdoor, cloud server 1 and cloud server 2 interactively execute the search process, and provide the encrypted search results to the data user. The data user then uses the private key to decrypt the search results, thereby restoring valid information. Besides, the data user has the capability to verify the correctness of the search results. It is worth noting that in the above process, the data owner learns the keywords searched by the data user. If the keyword trapdoor is generated by the data user himself, the data user needs to interact with the data owner and confirm the position of the queried keywords in the dictionary. Although this may optimize the model, it would also increase the complexity of the interaction between both parties. To streamline the interaction process, we have currently adopted the model described above.

4.2 Design goals

The goals include the following:

- **Confidentiality**: The proposed schemes should be capable of protecting keyword privacy in the keyword index matrix and keyword trapdoor. Additionally, the proposed schemes can also hide the search and access pattern during the search process.
- **Verifiability**: The proposed verifiable scheme should have the ability to allow data users to verify search results, enabling them to detect incomplete results from cloud servers, and errors in calculation or transmission processes.
- **Efficiency**: The proposed schemes should allow data users to obtain search results as quickly as possible and support parallel operation during the search phase.

4.3 Formal definition

The proposed schemes consist of the following probabilistic polynomial time (PPT) algorithms:

- **Setup**$(1^\lambda, DB) \rightarrow (pp, I', I'', K)$: After inputting the security parameter 1^λ and database DB, the data owner outputs the public parameter pp, the keyword index matrixes I' and I'' and the secret information K, where I' and I'' are uploaded to cloud servers 1 and 2, respectively.
- **Trapdoor**$(pp, Q) \rightarrow (Tr, pk_u, sk_u, \beta)$: The data owner initializes the Paillier encryption algorithm according to the security parameter 1^λ, and obtains the public and private keys (pk_u, sk_u). The data owner generates the keyword trapdoor based on the queried keyword set Q and a random number β, and then uploads the keyword trapdoor Tr to the cloud servers. Besides, the data owner issues the private key sk_u and the random number β to the data user.
- **Search**$(Tr, I', I'') \rightarrow P$ and $\mathcal{P}$: After receiving the search request, the cloud servers compute the polynomial implying document identifier intersection set according to Tr, I' and I'' and returns the encrypted polynomials P and $\mathcal{P}$ to the data user.
- **Decrypt**$(P, \mathcal{P}, sk_u) \rightarrow F$ or $\perp$: The data user decrypts the returned results P and $\mathcal{P}$ by sk_u, obtains the result polynomials, and then verifies them. If the verification is passed, the data user obtains the document identifier set F. Otherwise, the result is rejected.

Remark: Our schemes are similar to most SE schemes [3, 16, 41] in that it does not prioritize the retrieval of encrypted files. Instead, these schemes focus on the storage and search of metadatas. SE can be categorized into two types based on search results: response-hiding [13] and response-revealing [5]. The former hides the matched document identifiers while the latter reveals them in plaintext, where the

response-hiding type hides the access pattern without considering encrypted payloads [42]. Additionally, we assume that there are independent authentication methods [43] and access control mechanisms [44] on the cloud server-side, which falls beyond the scope of our discussion.

If the data user wants to obtain encrypted documents, an additional round is required after obtaining the document metadata. This can be applicable in scenarios where matching document previews are displayed before they are downloaded. Before obtaining the encrypted documents in the second round, the data user has already acquired the required document identifiers. In this system, there are two non collusive cloud servers. Following a similar approach as described in [17], the data user randomly divides the document identifier set into two parts to retrieve the encrypted files. To further enhance privacy, the user can incorporate random false positives and negatives [18] during the aforementioned operation to obfuscate the search process and results. To support second-round verifiability [24], we can store the hash values of the documents locally and compare the hash values of returned outcomes with local hash values. Here, we mainly focus on the first round.

4.4 Security definition

Similar to SE schemes [16, 41], we assume that the data owner is honest. In the first scheme, the cloud servers are assumed to be semi-honest, which may be curious about the underlying keywords in the keyword index matrix and keyword trapdoors. In the second scheme, the cloud servers are assumed to be malicious, where the cloud servers may return the error or incomplete results. The data user may want to learn information other than search results. Here, we only consider the storage and processing of the metadata without paying attention to the retrieval of encrypted documents. Then, we define the security of the schemes from the cloud server-side and the data user side. Similar to the schemes [3, 41], the security against the cloud servers is formalized by the simulation-based approach between a PPT adversary $\mathcal{A}$ and a simulator $\mathcal{S}$, where the real game $\mathbf{Real}_{\mathcal{A}}^{\Pi}$ and the ideal game $\mathbf{Ideal}_{\mathcal{A},\mathcal{S}}^{\Pi}$ are computationally indistinguishable. The essence of this proof approach is that the adversary cannot learn more leakage than the definition of the leakage function $\mathcal{L}$. Assuming that there are t conjunctive queries $\mathbf{Q} = (Q_1, Q_2, \ldots, Q_t)$, the leakage function with the input DB and $\mathbf{Q}$ is defined as follows:

- $m = |W|$ is the number of keywords in the keyword dictionary.
- L is the upper bound of cardinalities of the document identifier set containing w_i.

In our schemes, the leakage function $\mathcal{L}$ reveals the number of rows and columns of the keyword index matrix I' or I''. $\mathbf{Real}_{\mathcal{A}}^{\Pi}(1^{\lambda})$ and $\mathbf{Ideal}_{\mathcal{A},\mathcal{S}}^{\Pi}(1^{\lambda})$ are defined as follows:

- $\mathbf{Real}_{\mathcal{A}}^{\Pi}(1^{\lambda})$: After receiving the database DB submitted by the adversary $\mathcal{A}$, the experiment returns the public parameter pp and encrypted index matrixes I' and I'' to $\mathcal{A}$ by running the **Setup** algorithm. Then, $\mathcal{A}$ adaptively selects a series of search queries with polynomial size interactions. For each query Q_i, the experiment gives transcripts of the **Trapdoor** algorithm and the **Search** algorithm to $\mathcal{A}$. Finally, the game outputs the result $b \in \{0, 1\}$ that is output by $\mathcal{A}$.
- $\mathbf{Ideal}_{\mathcal{A},\mathcal{S}}^{\Pi}(1^{\lambda})$: After receiving the database DB submitted by the adversary $\mathcal{A}$, the experiment returns the public parameter pp and encrypted index matrixes I' and I'' to $\mathcal{A}$ by running the $\mathcal{S}(\mathcal{L}(1^{\lambda}, DB))$ algorithm. Then, $\mathcal{A}$ adaptively selects a series of search queries with polynomial size interactions. For each query Q_i, the experiment gives transcripts of the $\mathcal{S}(\mathcal{L}(DB, Q))$ algorithm to $\mathcal{A}$. Finally, the game outputs the result $b \in \{0, 1\}$ that is output by $\mathcal{A}$.

A SE scheme is $\mathcal{L}$-*adaptively-secure* if for any PPT $\mathcal{A}$, there is a simulator $\mathcal{S}$ so that $\mathcal{A}$ cannot distinguish whether it is interacting with $\mathbf{Real}_{\mathcal{A}}^{\Pi}(1^{\lambda})$ or $\mathbf{Ideal}_{\mathcal{A},\mathcal{S}}^{\Pi}(1^{\lambda})$. The formal definition is as follows:

Definition 1. The proposed scheme is $\mathcal{L}$-*adaptively-secure* if for any PPT $\mathcal{A}$ there is a simulator $\mathcal{S}$ such that

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

As for the user side, the data user does not learn any other information beyond the search result, whose argument will be postponed to a later section.

5 Concrete constructions

5.1 Overview

The overview is presented as follows: Firstly, the documents are reorganized in a reverse-index manner, where each keyword corresponds to a polynomial generated by the document identifiers that contain it. The coefficients of the polynomials are blinded and randomly divided into two parts, I' and I'', which are then uploaded to cloud servers 1 and 2 respectively by the data owner. Secondly, the keyword trapdoor can be generated based on the searched keyword set Q_i and the homomorphic encryption algorithm. Specifically, for each $w_i \in W$ in the keyword dictionary, a random number r_i is chosen only if $w_i \in W \cap Q_i$, otherwise, $r_i = 0$ is set. The trapdoor keyword is generated by encrypting r_i and uploaded to the cloud servers. Thirdly, the cloud servers utilize the keyword trapdoor and the index matrices to calculate the polynomials. During this process, the polynomial corresponding to the keyword in the searched keyword set Q is not calculated to 0. Consequently, the polynomial corresponding to the file identifiers that contain the conjunctive keywords Q is returned. Fourthly, after decrypting the returned result, the data user identifies the valid root of the polynomial and recovers the file identifier set.

To realize the verifiability of search results, a random number β is embedded in the keyword trapdoor. The polynomial returned by the cloud servers implies β as the root. After decrypting, the data user verifies whether β is the root of the polynomial. If so, the calculation process is correct, and then the algorithm continues. Otherwise, the error result is rejected. Since the verifiable scheme supports parallel operations, it may also pass the verification when the server returns incomplete results. Specifically, the cloud server misses some results during the processing. It only calculates a subset of keywords $Q' \subseteq Q$ for one query set Q. To avoid this situation, both the cloud servers 1 and 2 return the search results. After the search results are verified, the data user extracts the valid file identifiers. If they are the same, the results are received; otherwise, the results are rejected.

5.2 SAP-ECKS construction

The proposed SE scheme is divided into the following concrete phases.

- **Setup**: After inputting the security parameter 1^λ and the database DB, the data owner outputs the system parameter pp and the keyword index matrix I.
  - The data owner chooses a pseudo-random function $f : (k, x) \to y$ with the key according to the security parameter λ. Besides, the data owner also selects a hash function $H : R \to \{0,1\}^{L_H}$, where the elements in ring R are mapped to L-size bit strings. (λ, f, H) are shared by participants as the system parameter pp.
  - For each document identifier set $DB(w_i)$ containing the keyword w_i, the data user employs the hash function H to encode the elements in $DB(w_i)$ so that they have the form $id' = id||H(id)$. The coded $DB(w_i)$ is represented a polynomial of degree L by

$$P_{w_i}(x) = x^{L-|DB(w_i)|} \prod_{id \in DB(w_i)} (x - id'),$$

    where id belongs to a domain $\mathbb{U}$ ($id \in \mathbb{U} \subseteq \mathbb{R}$). After that, the polynomial $P_{w_i}(x)$ can be represented by a coefficient vector $X_i = (\tau_{i0}, \tau_{i1}, \ldots, \tau_{iL})$ such that $P_{w_i}(x) = \Sigma_{j=0}^{j=L} \tau_{ij} \cdot x^j$.
  - The data owner randomly generates the private key $k_z \in \mathbb{R}$ of a pseudo-random function f to blind $X = (X_1, X_2, \ldots, X_m)$. Specifically, he computes

$$I_i = z_i \cdot X_i = (z_i \cdot \tau_{i0}, z_i \cdot \tau_{i1}, \ldots, z_i \cdot \tau_{iL}),$$

    where $z_i = f(k_z, i)$, $1 \le i \le m$. The secret information K held by data owner includes k_z and the order of keywords in the keyword dictionary. Finally, the data owner randomly divides the index matrix $I = (I_1, I_2, \ldots, I_m)$ into two parts I' and I'' such that $I = I' + I''$, and uploads I' and I'' to the cloud server 1 and cloud server 2, respectively.
- **Trapdoor**: When the data user wants to access documents from the data owner, he first submits a request for access permission to the data owner.

  - The data owner runs the *KeyGen* algorithm of Paillier to obtain the public and private keys (pk_u, sk_u), where sk_u is returned to the data user.
  - The data owner finds the position $L_i = (l'_1, l'_2, \ldots, l'_q)$ of all searched keyword set $Q_i = (w'_1, w'_2, \ldots, w'_q)$ in the keyword dictionary W. For $1 \leq i \leq |W|$, if $i \in L_i$, he randomly selects r_i and calculates $Tr_i = E_{pk_u}(r_i)$; otherwise, he sets $r_i = 0$ and calculates $Tr_i = E_{pk_u}(r_i)$. Finally, the data owner uploads the keyword trapdoor $Tr = (Tr_1, Tr_2, \ldots, Tr_m)$ to the cloud server 1 and cloud server 2.
- **Search**: When the cloud servers receive the search request from the data owner, the cloud server 1 returns the search result to the data user as follows:
  - It first generates m L-degree polynomials $(\varphi_1, \varphi_2, \ldots, \varphi_m)$. Then it transforms the blinding factor of I_i by φ_i to obtain $T'_i(x) = (I'_i(x) \cdot \varphi_i(x)) *_h Tr_i = E_{pk_u}(r_i \cdot I'_i(x) \cdot \varphi_i(x))$.
  - The cloud server 1 sends $(\varphi_1, \varphi_2, \ldots, \varphi_m)$ to the cloud server 2. Then, the cloud server 2 computes

$$T''_i(x) = (I''_i(x) \cdot \varphi_i(x)) *_h Tr_i = E_{pk_u}(r_i I''_i(x)\varphi_i(x))$$

    and returns the result $T''_i(x)$ to the cloud server 1.
  - After that, the cloud server 1 computes

$$P'_{w_i}(x) = T'_i(x) +_h T''_i(x) = E_{pk_u}(r_i I_i(x)\varphi_i(x)).$$

    Finally, it returns the search result

$$P(x) = P'_{w_1}(x) +_h P'_{w_2}(x) +_h \ldots +_h P'_{w_m}(x)$$

    to the data user.
- **Decrypt**: After receiving the search result, the data user performs the operations as follows:
  - The data user fist decrypts the search result $P(x)$ by the *Decryption* algorithm of Paillier and gets a polynomial

$$P'(x) = r_1 z_1 \varphi_1(x) P_{w_1}(x) + r_2 z_2 \varphi_2(x) P_{w_2}(x) + \ldots + r_m z_m \varphi_m(x) P_{w_m}(x).$$

  - The data user finds valid roots of the polynomial $P'(x)$ and obtains the file identifier set F.

Correctness analysis: The correctness of the scheme requires that all the files (*i.e.*, file identifiers) containing queried keyword set Q are returned and the data user can correctly recover the valid roots of the polynomial (*i.e.*, file identifiers) from the returned result by filtering out random roots. In *Setup* phase, the file identifiers $DB(w_i)$ containing a keyword forms a polynomial $P_{w_i}(x)$, and then the coefficient $X_i = (\tau_{i1}, \tau_{i2}, \ldots, \tau_{iL})$ of $P_{w_i}(x)$ is blinded by a random number z_i. Next, $I_i = z_i X_i$ is randomly divided into two parts I' and I'' such that $I = I' + I''$. During the process of generating keyword trapdoors, for $1 \leq i \leq m$, if $w'_i \in W \cap Q$, $Tr_i = E_{pk_u}(r_i)$, where r_i is a random number. Otherwise $w'_i \notin Q$, $Tr_i = E_{pk_u}(r_i)$, where $r_i = 0$. During the search phase, the cloud servers 1 and 2 employ random polynomials $(\varphi_1, \varphi_2, \ldots, \varphi_m)$ to compute

$$T'_i = E_{pk_u}(r_i \cdot I'_i(x) \cdot \varphi_i(x)), T''_i = E_{pk_u}(r_i \cdot I''_i(x) \cdot \varphi_i(x)).$$

Then the result returned is

$$P(x) = Enc_{pk_u}(P'(x)) = Enc_{pk_u}(\Sigma_{i=1}^{m}(r_i z_i \varphi_i(x) P_{w_i}(x))).$$

After decryption, the data user can obtain

$$P'(x) = \Sigma_{i=1}^{m}(r_i z_i \varphi_i(x) P_{w_i}(x)).$$

On the one hand, if $P_{w_i}(a) = 0$ for all $w_i \in Q$, then $P'(a) = 0$, which means a is the common root of the polynomials. $F = \cap_{w_i \in Q} DB(w_i)$ is the result of the conjunction query. On the other hand, it can filter out the wrong root in the form $a||H(a)$ of valid roots. As a result, the data user is able to obtain the results they want to search with an overwhelming probability.

5.3 SAP-VECKS construction

The proposed SE scheme is divided into the following concrete phases.

- **Setup**: This process is the same as the above scheme.
- **Trapdoor**: When the data user wants to access documents from the data owner, he first applies for the access permission to the data owner,
  - The data owner runs the *KeyGen* algorithm of Paillier to obtain the public and private keys (pk_u, sk_u).
  - The data owner finds the position $L_i = (l'_1, l'_2, \ldots, l'_q)$ of all searched keyword set $Q_i = (w'_1, w'_2, \ldots, w'_q)$ in the keyword dictionary W. Then, the data owner randomly selects a random number β. Next, for $1 \leq i \leq m$, if $i \in L_i$, he randomly selects r_i and calculates $Tr_{i1} = E_{pk_u}(r_i)$ and $Tr_{i2} = E_{pk_u}(r_i \cdot \beta)$. Otherwise, he sets $r_i = 0$ and calculates $Tr_{i1} = E_{pk_u}(r_i)$ and $Tr_{i2} = E_{pk_u}(r_i \cdot \beta)$.
  - The data owner uploads the search trapdoor $Tr = (Tr_1, Tr_2, \ldots, Tr_m)$ to the cloud server 1 and cloud server 2, where $Tr_i = (Tr_{i1}, Tr_{i2})$. Besides, the data owner returns the private key sk_u and the random β to the data user.
- **Search**: When the cloud server receives the search request from the data owner, it can return the search result to the data user.
  - The cloud servers generates m L-degree polynomials $(\varphi_1, \varphi_2, \ldots, \varphi_m)$ and $(\psi_1, \psi_2, \ldots, \psi_m)$, respectively. Then the cloud server 1 transforms the blinding factor of I_i by φ_i to obtain

$$T'_{i1}(x) = x \cdot (I'_i(x) \cdot \varphi_i(x)) *_h Tr_{i1} = E_{pk_u}(x \cdot r_i \cdot I'_i(x) \cdot \varphi_i(x))$$

    and

$$T'_{i2}(x) = (-I'_i(x) \cdot \varphi_i(x)) *_h Tr_{i2} = E_{pk_u}(-r_i \cdot \beta \cdot I'_i(x) \cdot \varphi_i(x)).$$

    The cloud server 2 computes

$$\mathcal{T}'_{i1}(x) = x \cdot (I''_i(x) \cdot \psi_i(x)) *_h Tr_{i1} = E_{pk_u}(x \cdot r_i \cdot I''_i(x) \cdot \psi_i(x))$$

    and

$$\mathcal{T}'_{i2}(x) = (-I''_i(x) \cdot \psi_i(x)) *_h Tr_{i2} = E_{pk_u}(-r_i \cdot \beta \cdot I''_i(x) \cdot \psi_i(x)).$$

  - The cloud server 1 also sends m polynomials $(\varphi_1, \varphi_2, \ldots, \varphi_m)$ to the cloud server 2. Then, the cloud server 2 computes

$$T''_{i1}(x) = (x \cdot I''_i(x) \cdot \varphi_i(x)) *_h Tr_{i1} = E_{pk_u}(x \cdot r_i I''_i(x) \varphi_i(x))$$

    and

$$T''_{i2}(x) = (-I''_i(x) \cdot \varphi_i(x)) *_h Tr_{i2} = E_{pk_u}(-r_i \cdot \beta I''_i(x) \varphi_i(x)).$$

    Next, it sends the result $T''_i = T''_{i1} +_h T''_{i2}$ to the cloud server 1.
  - The cloud server 2 also sends m polynomials $(\psi_1, \psi_2, \ldots, \psi_m)$ to the cloud server 1. Then, the cloud server 1 computes

$$\mathcal{T}''_{i1}(x) = (x \cdot I'_i(x) \cdot \psi_i(x)) *_h Tr_{i1} = E_{pk_u}(x \cdot r_i I'_i(x) \psi_i(x))$$

    and

$$\mathcal{T}''_{i2}(x) = (-I'_i(x) \cdot \psi_i(x)) *_h Tr_{i2} = E_{pk_u}(-r_i \cdot \beta I'_i(x) \psi_i(x)).$$

    Next, it sends the results $\mathcal{T}''_i = \mathcal{T}''_{i1} +_h \mathcal{T}''_{i2}$ to the cloud server 2.
  - The cloud server 1 computes

$$\begin{aligned} P'_{w_i}(x) &= T'_{i1}(x) +_h T'_{i2}(x) +_h T''_i(x) \\ &= E_{pk_u}(x \cdot r_i \cdot I'_i(x) \cdot \varphi_i(x) - r_i \cdot \beta \cdot I'_i(x) \cdot \varphi_i(x) + x \cdot r_i \cdot I''_i(x) \varphi_i(x) - r_i \cdot \beta I''_i(x) \varphi_i(x)) \\ &= E_{pk_u}(x \cdot r_i \cdot z_i \cdot P_{w_i}(x) \cdot \varphi_i(x) - \beta \cdot r_i \cdot z_i \cdot P_{w_i}(x) \cdot \varphi_i(x)). \end{aligned}$$

    Next, it returns the search result

$$P(x) = P'_{w_1}(x) +_h P'_{w_2}(x) +_h \ldots +_h P'_{w_m}(x)$$

    to the data user.

– The cloud server 2 computes

$$\begin{aligned}\mathcal{P}'_{w_i}(x) &= \mathcal{T}'_{i1}(x) +_h \mathcal{T}'_{i2}(x) +_h \mathcal{T}''_i(x) \\ &= E_{pk_u}(x \cdot r_i \cdot I''_i(x) \cdot \psi_i(x) - r_i \cdot \beta \cdot I''_i(x) \cdot \psi_i(x) + x \cdot r_i \cdot I'_i(x)\psi_i(x) - r_i \cdot \beta I'_i(x)\psi_i(x)) \\ &= E_{pk_u}(x \cdot r_i \cdot z_i \cdot P_{w_i}(x) \cdot \psi_i(x) - \beta \cdot r_i \cdot z_i \cdot P_{w_i}(x) \cdot \psi_i(x)).\end{aligned}$$

Next, it returns the search result

$$\mathcal{P}(x) = \mathcal{P}'_{w_1}(x) +_h \mathcal{P}'_{w_2}(x) +_h \ldots +_h \mathcal{P}'_{w_m}(x)$$

to the data user.

– **Decrypt**: After receiving the search result, the data user performs the operations as follows:
  – The data user fist decrypts the search results $P(x)$ and $\mathcal{P}(x)$ and gets polynomials

$$P'(x) = r_1 z_1 \cdot (x - \beta) \cdot \varphi_1(x) P_{w_1}(x) + \ldots + r_m z_m \cdot (x - \beta) \cdot \varphi_m(x) P_{w_m}(x)$$

and

$$\mathcal{P}'(x) = r_1 z_1 \cdot (x - \beta) \cdot \psi_1(x) P_{w_1}(x) + \ldots + r_m z_m \cdot (x - \beta) \cdot \psi_m(x) P_{w_m}(x).$$

  – Then, the data user verifies whether β is the root of $P'(x)$ and $\mathcal{P}'(x)$. If not, the result is rejected; Otherwise, the data user finds valid roots of the polynomial $P'(x)$ and $\mathcal{P}'(x)$. If the valid roots are the same, the file identifier set F is output. Otherwise, the result is rejected.

Correctness analysis: In addition to correctness, the scheme also provides the verifiability of search results, which requires that the data user can verify the correctness of the returned results. During the process of generating keyword trapdoors, if $w'_i \in W \cap Q$, $Tr_{i1} = E_{pk_u}(r_i)$ and $Tr_{i2} = E_{pk_u}(r_i\beta)$, where $r_i, \beta \in \mathbb{R}$. If $w'_i \notin Q$, $Tr_{i1} = E_{pk_u}(r_i)$ and $Tr_{i2} = E_{pk_u}(r_i\beta)$, where $r_i = 0$. During the search phase, the cloud servers employ random polynomials $(\varphi_1, \varphi_2, \ldots, \varphi_m)$ and $(\psi_1, \psi_2, \ldots, \psi_m)$ to compute

$$P'_{w_i}(x) = E_{pk_u}((x - \beta) \cdot r_i \cdot I_i(x) \cdot \varphi_i(x)), \mathcal{P}'_{w_i}(x) = E_{pk_u}((x - \beta) \cdot r_i \cdot I_i(x) \cdot \psi_i(x)).$$

Then the returned results are

$$P(x) = Enc_{pk_u}(P'(x)) = Enc_{pk_u}(\Sigma_{i=1}^m (r_i z_i \sigma(x) \varphi_i(x) P_{w_i}(x)))$$

and

$$\mathcal{P}(x) = Enc_{pk_u}(\mathcal{P}'(x)) = Enc_{pk_u}(\Sigma_{i=1}^m (r_i z_i \sigma(x) \psi_i(x) P_{w_i}(x))),$$

where $\sigma(x) = x - \beta$. After decryption, the data user can obtain

$$P'(x) = \Sigma_{i=1}^m (r_i z_i \sigma(x) \varphi_i(x) P_{w_i}(x)), \mathcal{P}'(x) = \Sigma_{i=1}^m (r_i z_i \sigma(x) \psi_i(x) P_{w_i}(x))$$

The data user first verifies whether β is the root of the polynomials $P'(x)$ and $\mathcal{P}'(x)$. If not, the results are rejected. Otherwise, the data user finds valid roots of the polynomials $P'(x)$ and $\mathcal{P}'(x)$. If the valid roots are the same, the file identifier set F is output. Otherwise, the result is rejected.

6 Security proof

Before we prove our schemes, let's first review leakage patterns. For t conjunctive queries $\mathbf{Q} = (Q_1, Q_2, \ldots, Q_t)$, *Search Pattern*, *Access Pattern*, *Response Equality Pattern* and *Response Length Pattern* are defined as follows:

– *Search Pattern* reveals whether two searches can be associated with the same keyword. It can be represented by a binary $t \times t$ matrix SP, where $SP[i, j] = 1$ if $Q[i] = Q[j]$ and $SP[i, j] = 0$ if $Q[i] \neq Q[j]$.
– *Access Pattern* reveals which document identifiers are included in a query. It is expressed as $\{DB(Q_1), DB(Q_2), \ldots, DB(Q_t)\}$
– *Response Equality Pattern* indicates whether the two responses are the same. It can be represented by a binary $t \times t$ matrix REP, where $REP[i, j] = 1$ if $R[i] = R[j]$ and $REP[i, j] = 0$ if $R[i] \neq R[j]$.

– *Response Length Pattern* exposes the cardinality of the document identifier set returned in each query. It is expressed as $\{|DB(Q_1)|, |DB(Q_2)|, \ldots, |DB(Q_t)|\}$. It is worth noting that response equality pattern and response length pattern can be obtained through access patterns.

The high-level intuition regarding the security is as follows: For two queries with the same keyword set, the keyword trapdoor is different since the algorithm *Trapdoor* is random for each search. The returned search result always has the same degree for the encryption polynomial $Enc_{pk_u}(P(x))$ (or $Enc_{pk_u}(\mathcal{P}(x))$), enabling us to hide the response length pattern. For the same retrieval request, the encryption polynomial returned each time is different, effectively hiding the response equality pattern. Additionally, the keyword trapdoor generation algorithm is random and both cloud server 1 and cloud server 2 utilize random polynomials to obscure the result. The respond-hiding results are encrypted, and the response equality pattern and response length pattern are hidden. Therefore, without considering the encrypted documents, the access pattern can also be kept hidden.

6.1 The proof of SAP-ECKS

Theorem 1. Assuming that the Paillier encryption scheme is IND-CPA secure, the proposed scheme SAP-ECKS denoted as Π is $\mathcal{L}$-*semantically-secure* against adaptive attacks, where $\mathcal{L}$ represents the leakage function.

Proof: The above theorem is proved by a series of indistinguishable games. The first game G_0 is exactly the same as $\text{Real}_{\mathcal{A}}^{\Pi}(1^\lambda)$. In the second game G_1, the keyword index matrixes are replaced by random numbers. In the third game G_2, we choose m random numbers and encrypt them by the Paillier encryption scheme in the keyword trapdoor generation. The last game $\text{Ideal}_{\mathcal{A},\mathcal{S}}^{\Pi}(1^\lambda)$ is simulated by $\mathcal{S}_1$ and $\mathcal{S}_2$ by leveraging the leakage function $\mathcal{L}$, where $\mathcal{S}_1$ and $\mathcal{S}_2$ simulate the cloud server 1 and cloud server 2, respectively. Through the indistinguishability of the above-mentioned games, we finally come to the conclusion that $\text{Real}_{\mathcal{A}}^{\Pi}(1^\lambda)$ and $\text{Ideal}_{\mathcal{A},\mathcal{S}}^{\Pi}(1^\lambda)$ are indistinguishable.

Game G_0: G_0 and the real game $\text{Real}_{\mathcal{A}}^{\Pi}(1^\lambda)$ are exactly the same. Therefore,

$$Pr[\text{Real}_{\mathcal{A}}^{\Pi}(1^\lambda) = 1] = Pr[G_0 = 1].$$

Game G_1: G_0 and G_1 are identical except for replacing the coefficients of polynomials with random numbers from the ring $\mathbb{R}$. Specifically, we select $m \cdot L$ random numbers to form $I' = \{x_{ij}\}_{1\le i\le L, 1\le j\le m}$. Then, we also choose $m \cdot L$ random numbers $\{r_{ij}\}_{1\le i\le L, 1\le j\le m}$ and calculate $x'_{ij} = r_{ij} - x_{ij}$ to form $I'' = \{x'_{ij}\}_{1\le i\le L, 1\le j\le m}$. The simulated I' and I'' are identical to the real game G_0. Therefore, we have

$$Pr[G_0 = 1] = Pr[G_1 = 1].$$

Game G_2: In the game, instead of leveraging the Paillier encryption algorithm and the query keyword set Q to generate the keyword trapdoor, we select m random numbers from the ring and encrypt them under the same key. Since the Paillier encryption scheme is semantically secure, the adversary $\mathcal{A}$ is not able to distinguish between G_2 and G_1. If $\mathcal{A}$ can distinguish G_2 from G_1, we can build a reduction against the IND-CPA Paillier encryption scheme. Therefore,

$$|Pr[G_1 = 1] - Pr[G_2 = 1]| \le \mathbf{Adv}_{\Pi,\mathcal{A}}^{\text{IND-CPA}}(1^\lambda).$$

Simulator: Until now, the keyword index matrixes and keyword trapdoor do not contain any information of keywords and file identifiers. Then, the details of constructing the simulator $\mathcal{S}_1$ and $\mathcal{S}_2$ are given. Specifically, $\mathcal{S}_1$ (or $\mathcal{S}_2$) takes the leakage function $\mathcal{L}(DB, Q) = (m, L)$ as input, and the simulated keyword index matrix I' or I'' and transcripts $Res = (Tr, F)$ as the output.

To simulate the keyword index matrix, $\mathcal{S}_1$ (or $\mathcal{S}_2$) randomly selects $m \times L$ random elements from the ring to form the keyword index matrix I' (or I''). The distribution of I' (or I'') in both Game G_1 and simulator $\mathcal{S}_1$ (or $\mathcal{S}_2$) is indistinguishable because the elements in I' and I'' are random.

To simulate the keyword trapdoor, $\mathcal{S}_1$ (or $\mathcal{S}_2$) selects m random numbers from the ring and then utilizes the Paillier encryption algorithm to encrypt these random numbers. The distribution of keyword trapdoors in both Game G_2 and simulator $\mathcal{S}_1$ (or $\mathcal{S}_2$) is indistinguishable because the search pattern is hidden and the Paillier encryption scheme is IND-CPA. The details of simulating the keyword index matrix and keyword trapdoor are shown in Algorithm 1.

Algorithm 1 Simulator $\mathcal{S}$

Input: $\mathcal{L}$
Output: $\mathcal{I}', \mathcal{I}'', Tr, Res$
Generating $\mathcal{I}'$
for $1 \leq i \leq m$ **do**
  for $1 \leq j \leq L$ **do**
    $z_{ij} \xleftarrow{a} \mathcal{R}$; $E_i \leftarrow z_{ij}$
  end for
  Parse $E_i = \{z_{i1}, z_{i2}, \ldots, z_{in}\}$; Store E_i to the $\mathcal{I}'$
end for
Generating $\mathcal{I}''$
for $1 \leq i \leq m$ **do**
  for $1 \leq j \leq L$ **do**
    $z_{ij} \xleftarrow{a} \mathcal{R}$; $E_i \leftarrow z_{ij}$
  end for
  Parse $E_i = \{z_{i1}, z_{i2}, \ldots, z_{in}\}$; Store E_i to the $\mathcal{I}''$
end for
Generating Tr, Res
for $1 \leq i \leq m$ **do**
  $e_i \xleftarrow{a} \mathcal{R}$; $Tr_i = Enc_{pk_u}(e_i)$
end for
$Res \leftarrow Search(I', I'', Tr, pp)$

Algorithm 2 Simulator *Search*

Input: I', I'', Tr, pp
Output: Res
$\underline{\mathcal{S}_1}$:
Generate m L-degree polynomials
for $1 \leq i \leq m$ **do**
  for $1 \leq j \leq L$ **do**
    $a_{ij} \xleftarrow{a} \mathcal{R}$; $E_i \leftarrow a_{ij}$
  end for
  Parse $E_i = \{\varphi_{i1}, \varphi_{i2}, \ldots, \varphi_{in}\}$; $\varphi_i \leftarrow E_i$
end for
Parse $Tr = \{Tr_1, Tr_2, \ldots, Tr_m\}$
Compute T_i' as the real game does
Send $(\varphi_1, \varphi_2, \ldots, \varphi_m)$ to $\mathcal{S}_2$
Store $(\varphi_1, \varphi_2, \ldots, \varphi_m)$ to Res
$\underline{\mathcal{S}_2}$:
Compute T_i'' as the real game does
Send T_i'' to $\mathcal{S}_1$
Store T_i'' to Res
$\underline{\mathcal{S}_1}$:
Compute $P(x)$ as the real game does
Store $P(x)$ to Res

Under the interaction of cloud server 1 and cloud server 2, the search result can be obtained. $\mathcal{S}_1$ randomly chooses m L-degree polynomials and send them to $\mathcal{S}_1$. Then, $\mathcal{S}_1$ and $\mathcal{S}_2$ perform calculations as the real game dose. The details of simulating the search phase are shown in Algorithm 2.

Theorem 2. The data user does not learn any additional information except the search result $F = \cap_{w_i \in Q} DB(w_i)$.

Proof: It can be seen from the correctness analysis that after the data owner sends the query request $Tr = (Tr_1, Tr_2, , \ldots, Tr_m)$, the cloud servers return an encrypted polynomial

$$P(x) = Enc_{pk_u}(P'(x)) = Enc_{pk_u}(\Sigma_{i=1}^{m}(r_i z_i \varphi_i(x) P_{w_i}(x)))$$

to the data user. After decryption, the data user obtains the polynomial $P'(x) = \Sigma_{i=1}^{m}(r_i z_i \varphi_i(x) P_{w_i}(x))$. Next, the data user recovers valid roots, from which document identifiers are obtained. From other random roots, the data user does not learn any information about keyword index matrix and documents. As a

result, from the returned polynomial, the data user does not learn any additional information except the the desired search result $F = \cap_{w_i \in Q} DB(w_i)$.

6.2 The proof of SAP-VECKS

Theorem 3. Assuming that the Paillier encryption scheme is IND-CPA secure, the proposed scheme SAP-VECKS denoted as Π_2 is $\mathcal{L}$-*semantically-secure* against adaptive attacks, where $\mathcal{L}$ represents the leakage function.

Proof: The idea of the proof is the same as ***Theorem 1***. The difference is the trapdoor generation and search processes. In trapdoor generation, we choose $2m$ random numbers instead of m random numbers. Then, the $2m$ random numbers are encrypted as Algorithm 1 does. Since the Paillier encryption scheme is IND-CPA secure, the adversary cannot distinguish the keyword trapdoor between in $\text{Real}_{\mathcal{A}}^{\Pi_2}(1^\lambda)$ and in $\text{Ideal}_{\mathcal{A},\mathcal{S}}^{\Pi_2}(1^\lambda)$. $\mathcal{S}_1$ and $\mathcal{S}_2$ first simulate the search process as Algorithm 2 does. Then $\mathcal{S}_1$ and $\mathcal{S}_2$ exchange identities and simulate again. As analyzed in ***Theorem 1***, the adversary cannot distinguish the search process in $\text{Real}_{\mathcal{A}}^{\Pi_2}(1^\lambda)$ and in $\text{Ideal}_{\mathcal{A},\mathcal{S}}^{\Pi_2}(1^\lambda)$. Due to space limitation, we omit the detailed process of simulation.

Theorem 4. The data user does not learn any additional information except the search result $F = \cap_{w_i \in Q} DB(w_i)$.

Proof: The proof process is the same as **Theorem 3**, except that the keyword trapdoor generation and search process. During the process of generating a keyword trapdoor, a random number β is embedded in the trapdoor $Tr = ((Tr_{11}, Tr_{12}), \ldots, (Tr_{m1}, Tr_{m2}))$, where $Tr_{i1} = E_{pk_u}(r_i)$, $Tr_{i2} = E_{pk_u}(r_i \cdot \beta)$. The cloud servers return the encrypted polynomials

$$P(x) = Enc_{pk_u}(P'(x)) = Enc_{pk_u}(\Sigma_{i=1}^{m}(r_i z_i \varphi_i(x)\sigma(x)P_{w_i}(x)))$$

and

$$\mathcal{P}(x) = Enc_{pk_u}(\mathcal{P}'(x)) = Enc_{pk_u}(\Sigma_{i=1}^{m}(r_i z_i \psi_i(x)\sigma(x)P_{w_i}(x))).$$

After decryption, the data user obtains the polynomials

$$P'(x) = \Sigma_{i=1}^{m}(r_i z_i \varphi_i(x)\sigma(x)P_{w_i}(x))$$

and

$$\mathcal{P}'(x) = \Sigma_{i=1}^{m}(r_i z_i \psi_i(x)\sigma(x)P_{w_i}(x)).$$

Then the data user verifies whether β is the root of $P'(x)$ and $\mathcal{P}'(x)$. After verification, the data user recovers valid roots, from which document identifiers are obtained. From other random roots, the data user does not learn any information about keyword index matrix and documents. As a result, from the returned polynomials, the data user does not learn any additional information except the the desired search result $F = \cap_{w_i \in Q} DB(w_i)$.

7 Performance analysis

In this section, we evaluate the performance of our schemes by comparing them with schemes [16, 18] in terms of computation costs, communication costs, and storage costs. Additionally, since the scheme [16] and our schemes are implemented by the same technique (*i.e.*, PSI), we further make a thorough comparison among our schemes and the scheme [16]. In our experimental evaluation, we implement our schemes and the schemes [16, 18] on the same platform and compare them from various perspectives.

7.1 Theoretical analysis

General analysis: The performance comparison among the schemes [16, 18] and our schemes is presented in Table 3. From Table 3, it can be observed that the computation costs, communication costs, and storage costs in [18] increase in proportion to the database size and the maximum number of distinct keywords in a document. Similarly, the computation costs, communication costs, and storage costs associated with for

Table 3. Efficiency comparison among the schemes and ours

Schemes	Computation costs			Communication costs		Storage costs
	Setup	Trapdoor	Search	Setup	Search	Cloud server
Shang *et al.*'s scheme [18]	$O(\|DB\|S)$	$O((\|DB\|+F+F\cdot L)S)$	$O(\|DB\|L)$	$O(\|DB\|S)$	$O((\|DB\|+F+F\cdot L)S)$	$O(\|DB\|S)$
Wang *et al.*'s scheme [16]	$O(mL)$	$O(m)$	$O(mL)$	$O(mL)$	$O(mL)$	$O(mL)$
SAP-ECKS	$O(mL)$	$O(m)$	$O(mL)$	$O(mL)$	$O(mL)$	$O(mL)$
SAP-VECKS	$O(mL)$	$O(m)$	$O(mL)$	$O(mL)$	$O(mL)$	$O(mL)$

$|BD|$, m, S, F and L represents the number of documents in the database, the number of distinct keywords in the database, the maximum number of distinct keywords in a document, the number of distinct document labels in the database and the maximum cardinality of a document set of files containing a keyword, respectively.

Table 4. Efficiency comparison between [16] and our schemes

Schemes	Computation costs			Communation costs		Storage costs
	Setup	Trapdoor	Search	Setup	Search	Cloud server 1
Wang *et al.*'s scheme [16]	$m(L+1)T_E$	mT_E	$(L^2+2L+3)mT_M+(L^2+4L)$ $mT_A+4LmT_E+2LmT_D$	$2(L+1)m\|\mathbb{Z}_{N^2}\|$	$(4mL+8m+4L+2)\|\mathbb{Z}_{N^2}\|$	$2(L+1)m\|\mathbb{Z}_{N^2}\|$
SAP-ECKS	$m(L+1)T_R$	mT'_E	$(4L+2)mT_M+(4mL+$ $2m-2L-1)T_A$	$2(L+1)m\|\mathbb{R}\|$	$(L+1)m\|\mathbb{R}\|+(2mL+2m+$ $2L+1)\|\mathbb{Z}_{N^2}\|$	$2(L+1)m\|\mathbb{R}\|$
SAP-VECKS	$m(L+1)T_R$	$2mT'_E$	$(8L+6)mT_M+(4mL+$ $2m+8L+2)T_A$	$2(L+1)m\|\mathbb{R}\|$	$(2L+2)m\|\mathbb{R}\|+(4mL+4m+$ $4L+2)\|\mathbb{Z}_{N^2}\|$	$2(L+1)m\|\mathbb{R}\|$

T_E, T'_E, T_D, T_A, T_M, and T_R represent the time cost of a BCP encryption operation, a Paillier encryption operation, a decryption operation, a homomorphic addition operation, a multiplication operation between a ciphertext and a random number, and a multiplication operatation, respectively. m is the number of keywords in the keyword dictionary. $|\mathbb{R}|$ and $|\mathbb{Z}_{N^2}|$ are the bit lengths of the elements in $\mathbb{R}$ and $\mathbb{Z}_{N^2}$.

the scheme [18] and our schemes are proportional to the number of keywords and the maximum number of keywords contained in a document. In a database system, the number of documents $|DB|$ is often several orders of magnitude larger than the number m of distinct keywords. Additionally, the maximum number of distinct keywords S in a document and the maximum cardinality L of a document set containing a keyword are not significantly different in terms of orders of magnitude. Therefore, Wang *et al.*'s and our schemes result in better efficiency because they employ the reverse-order index structure compared to the scheme [18]. Moreover, as shown in Table 3, Wang's scheme [16] is of the same order of magnitude as ours. Further comparison analysis are provided below.

Thorough analysis: To further compare the efficiency of the scheme [16] and our schemes, we make a thorough comparison shown in Table 4, focusing on computation costs, communication costs, and storage costs. From the table, our schemes demonstrate superior performance in terms of computational costs across all phases. Specifically, when generating the keyword index matrix, our schemes employ random numbers to obfuscate the values of polynomials instead of homomorphic encryption. Moreover, during the keyword trapdoor generation phase, a BCP encryption operation is more time-consuming compared to a Paillier encryption operation. In the search phase, our schemes eliminate the need for computing the product between a polynomial and an encrypted polynomial. Regarding communication costs, our schemes surpass Wang *et al.*'s scheme [16] in both the keyword index generation and search phases, mainly due to message expansion that occurs during encryption operations. Similarly, our schemes are superior to Wang *et al.*'s scheme [16] concerning storage costs. Additionally, it is worth noting that the scheme proposed by Wang *et al.* [16] relies on a special additive homomorphic encryption scheme with a double trapdoor decryption mechanism, while our schemes only require a classical additive homomorphic encryption, making our approach more versatile and applicable.

Conclusion of theoretical analysis: Based on the analysis above, it is evident that our schemes outperform those proposed by Wang *et al.* [16] and Shang *et al.* [18] overall. Additionally, our schemes are not dependent on a specific additive homomorphic encryption scheme, making them more versatile.

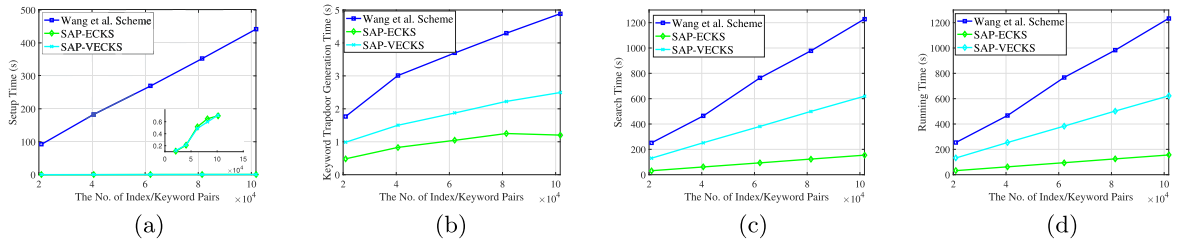

Figure 2. Computation costs. (a) Setup phase. (b) Trapdoor generation phase. (c) Search phase. (d) The whole process of obtaining file indexes

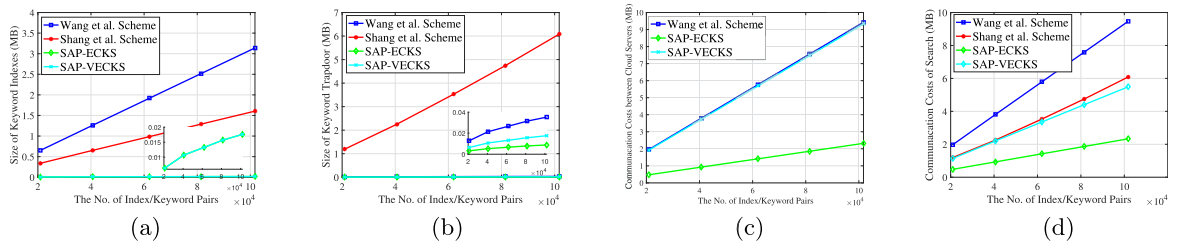

Figure 3. Communication costs. (a) Setup phase. (b) Trapdoor generation phase. (c) Search phase. (d) The whole process of obtaining file indexes

7.2 Experimental analysis

To compare our schemes with Wang *et al.*'s scheme [16] and Shang *et al.*'s scheme [18] in a more intuitive manner, we implement them using GMP[1], NTL library[2] and PBC[3] on the real-world Wikimedia dataset[4] (100000 keyword/index pairs are extracted), then test and compare them on the same platform from different aspects. The experiment runs on Ubuntu 18.04 LTS with 4 cores, 8 threads (Intel(R) Core(TM) i5- 8265U CPU, 1.60 GHz), 8 GB RAM, and 512GB disk. The security parameter of Wang *et al.*'s scheme [16] and our schemes is set to 1024 bit while Shang *et al.*'s scheme [18] is implemented on the default 160-bit Type A elliptic curve.

Computation costs: The comparison of computation costs among our schemes and the scheme proposed in [16] is presented in Figure 2. Here, we do not compare our schemes with the scheme introduced in [18]. After we implement their scheme [18] on the same platform, we learn that their scheme is very time-consuming. Specifically, after converting the keyword/document pairs (e.g. 20000 keyword/index pairs) to the document/keyword-set pairs, the *Setup*, *Trapdoor* and *Search* phases take 550.078, 171.716 and 6241.359 seconds, respectively, which is far greater than the cost of our schemes and the one in [16]. Figure 2a illustrates that our schemes are highly efficient in computing the keyword index matrix. Unlike other schemes, our approaches do not require encryption operations during this process. Instead, we simply randomize the coefficients of polynomials, which involves randomizing polynomial coefficients and dividing them randomly into two parts. In the keyword trapdoor generation process, as shown in Figure 2b, our schemes are more efficient. This can be attributed to the fact that one BCP encryption operation is more time-consuming than one Paillier encryption operation. In terms of search efficiency, as depicted in Figure 2c, our schemes outperform others. This is because our schemes do not involve multiplying a random polynomial by an encrypted polynomial, which is a very time-consuming operation. Figure 2d shows the comparison of the computation costs from generating keyword trapdoor to obtain file identifiers. Compared with [16], our schemes obtain document identifiers faster.Overall, our schemes show superiority over [16, 18] in terms of computation costs.

Communication costs: Figure 3 displays a comparison of communication costs among our schemes and the schemes [16, 18]. As shown in Figure 3a, our schemes have a better communication cost in the

[1] https://gmplib.org/
[2] https://libntl.org/
[3] https://crypto.stanford.edu/pbc/
[4] https://dumps.wikimedia.org/

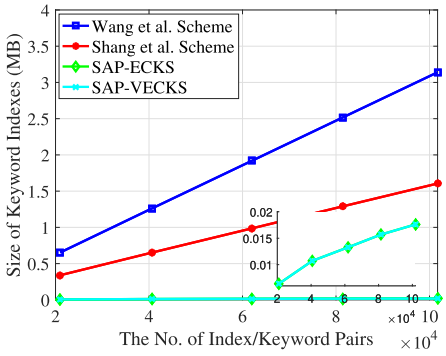

Figure 4. Storage costs at server side

keyword index generation phase. This is mainly because our schemes do not involve encryption operations in this phase. Figure 3b indicates that our schemes and [16] have superior communication costs compared to [18]. The main reason is that their scheme introduces redundancy to hide access and search patterns. From Figure 3c, our schemes require fewer communication resources than the scheme [16] in the search process, which results from the observation that our schemes avoid homomorphic multiplication between two ciphertexts. Figure 3d presents a comparison of the communication costs from generating keyword trapdoor to obtaining file identifiers, indicating that our schemes provide a better network bandwidth. In short, our schemes have smaller communication costs.

Storage costs: Figure 4 displays the storage costs of the server-side keyword index. As illustrated in Figure 4, our schemes save more server-side storage costs. This is primarily attributable to the fact that our schemes do not employ homomorphic encryption in the keyword index generation phase. Instead, we utilize randomization techniques such as blinding polynomial coefficients and dividing them randomly into two parts to obfuscate the values of polynomials.

Conclusion of performance analysis: After conducting thorough theoretical and experimental analysis, it is evident that our schemes require less computation and network bandwidths. Although the scheme [18] realizes the search and access pattern privacy on a single server in a single-round interaction, its computation costs and communication costs in each phase are very expensive, which negates this advantage of the single server and single-round interaction. While our schemes and Wang *et al.*'s scheme [16] share the same system model, our schemes offer superior efficiency and the rich functionality to verify search results. In summary, our schemes outperform existing alternatives in terms of efficiency and offer richer functionalities.

8 Conclusion

In this study, we propose SAP-ECKS, an efficient conjunctive searchable encryption with the privacy of search and access patterns. We also extend the initial scheme to support the verifiability of search results. Our proposed schemes primarily utilizes PSI to maintain search and access pattern privacy and improve efficiency by randomly dividing the index matrix into two parts. During the search process, the cloud servers compute the intersection of file identifier sets associated with each keyword and provide the result to the data user. After decryption, the data user can extract the file identifiers containing the search keywords. In the verifiable scheme, data users can verifies the correctness of the search process. Additionally, we prove the security of our proposed schemes through simulation-based methods. Furthermore, we implement our schemes and conduct a comparison with several related schemes using a real database. The experimental results indicate that our schemes exhibit better efficiency in terms of computation costs, communication costs, and storage costs. In the following work, we will achieve the same goals on the dynamic database.

Acknowledgments

No acknowledgments.

Funding

This work was supported by the National Key Research and Development Program of China under Grant No. 2022YFB4501500 and No. 2022YFB4501503, the National Natural Science Foundation of China (62072369), The Youth Innovation Team of Shaanxi Universities (23JP160), the Shaanxi Special Support Program Youth Top-notch Talent Program, the Technology Innovation Leading Program of Shaanxi (2023-YD-CGZH-31), and the China Postdoctoral Science Foundation under Grant Number 2024T170080.

Conflicts of interest

The authors declare that they have no conflicts of interest.

Data availability statement

No data are associated with this article.

Author contribution statement

Axin Wu conceived of the presented idea, developed the theory and composed the main of the paper. Dengguo Feng helped to develop the theory, and supervised the paper. Min Zhang and Jialin Chi assisted in writing the paper. Yinghui Zhang verified the proposed schemes, analyzed the evaluation results, and reviewed the paper.

References

[1] Song DX, Wagner D and Perrig A. Practical techniques for searches on encrypted data. In: Proceeding 2000 IEEE Symposium on Security and Privacy, IEEE, 2000, 44–55.
[2] Mondal P, Chamani JG, Demertzis I, et al. I/O-efficient dynamic searchable encryption meets forward & backward privacy. In: 33rd USENIX Security Symposium, 2024, 2527–2544.
[3] Cash D, Jarecki S, Jutla C, et al. Highly-scalable searchable symmetric encryption with support for boolean queries. In: Annual Cryptology Conference, Springer, 2013, 353–373.
[4] Luo F, Wang H and Yan X. Re-paeks: Public-key authenticated re-encryption with keyword search. IEEE Trans Mobile Comput 2024; 1–14.
[5] Bost R. $\sum o\varphi o\varsigma$: Forward secure searchable encryption. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, 2016, 1143–1154.
[6] Curtmola R, Garay J, Kamara S, et al. Searchable symmetric encryption: improved definitions and efficient constructions. J Comput Secur 2011; **19**: 895–934.
[7] Liu C, Zhu L, Wang M, et al. Search pattern leakage in searchable encryption: Attacks and new construction. Inf. Sci. 2014; **265**: 176–188.
[8] Cash D, Grubbs P, Perry J, et al. Leakage-abuse attacks against searchable encryption. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, 2015, 668–679.
[9] Zhang Y, Katz J and Papamanthou C. All your queries are belong to us: The power of file-injection attacks on searchable encryption. In: 25th USENIX Security Symposium, 2016, 707–720.
[10] Blackstone L, Kamara S and Moataz T. Revisiting leakage abuse attacks. In: Network and Distributed System Security Symposium, 2020, 1–18.
[11] Oya S and Kerschbaum F. Hiding the access pattern is not enough: Exploiting search pattern leakage in searchable encryption. In: 30th USENIX Security Symposium, 2021, 127–142.
[12] Garg S, Mohassel P and Papamanthou C. TWORAM: Efficient oblivious ram in two rounds with applications to searchable encryption. In: Annual International Cryptology Conference, Springer, 2016, 563–592.
[13] Lai S, Patranabis S, Sakzad A, et al. Result pattern hiding searchable encryption for conjunctive queries. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, 2018, 745–762.
[14] Chen G, Lai TH, Reiter MK, et al. Differentially private access patterns for searchable symmetric encryption. In: IEEE INFOCOM 2018 IEEE Conference on Computer Communications, IEEE, 2018, 810–818.
[15] Zheng Y, Lu R, Shao J, et al. Achieving practical symmetric searchable encryption with search pattern privacy over cloud. IEEE Trans Serv Comput 2020; **15**: 1358–1370.
[16] Wang Y, Sun SF, Wang J, et al. Achieving searchable encryption scheme with search pattern hidden. IEEE Trans Serv Comput 2022; **15**: 1012–1025.
[17] Song Q, Liu Z, Cao J, et al. SAP-SSE: Protecting search patterns and access patterns in searchable symmetric encryption. IEEE Trans Inf Forens Secur 2020; **16**: 1795–1809.
[18] Shang Z, Oya S, Peter A, et al. Obfuscated access and search patterns in searchable encryption. In: Network and Distributed System Security Symposium, 2021, 1–18.
[19] Yang Y, Hu Y, Dong X, et al. Opense: Efficient verifiable searchable encryption with access and search pattern hidden for cloud-iot. IEEE Internet Things J 2024; **11**: 13793–13809.

[20] Corrigan-Gibbs H and Kogan D. Private information retrieval with sublinear online time. In: 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Springer, 2020, 44–75.
[21] Zhang Y, Zhu T, Guo R, et al. Multi-keyword searchable and verifiable attribute-based encryption over cloud data. IEEE Trans Cloud Comput 2023; **11**: 971–983.
[22] Hong C, Li Y, Zhang M, et al. Fast multi-keywords search over encrypted cloud data. In: 17th International Conference: Web Information Systems Engineering, Springer, 2016, 433–446.
[23] Wang Q, Lai C, Lu R, et al. Searchable encryption with autonomous path delegation function and its application in healthcare cloud. IEEE Trans Cloud Comput 2023; **11**: 879–896.
[24] Wu A, Yang A, Luo W, et al. Enabling traceable and verifiable multi-user forward secure searchable encryption in hybrid cloud. IEEE Trans Cloud Comput 2023; **11**: 1886–1898.
[25] Lv Z, Zhang M and Feng D. Multi-user searchable encryption with efficient access control for cloud storage. In: 2014 IEEE 6th International Conference on Cloud Computing Technology and Science, IEEE, 2014, 366–373.
[26] Ming Y, Liu H, Wang C, et al. Generic construction: Cryptographic reverse firewalls for public key encryption with keyword search in cloud storage. IEEE Trans Cloud Comput 2024; **12**: 405–418.
[27] Wu A, Li F, Xin X, et al. Efficient public-key searchable encryption against inside keyword guessing attacks for cloud storage. J Syst Archit 2024; **149**: 103104.
[28] Kamara S, Papamanthou C and Roeder T. Dynamic searchable symmetric encryption. In: Proceedings of the 2012 ACM Conference on Computer and Communications Security, 2012, 965–976.
[29] Golle P, Staddon J and Waters B. Secure conjunctive keyword search over encrypted data. In: International Conference on Applied Cryptography and Network Security, Springer, 2004, 31–45.
[30] Kamara S and Moataz T. Boolean searchable symmetric encryption with worst-case sub-linear complexity. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques, Springer, 2017, 94–124.
[31] Islam MS, Kuzu M and Kantarcioglu M. Access pattern disclosure on searchable encryption: Ramification, attack and mitigation. In: Network and Distributed System Security Symposium, Citeseer, 2012, 12.
[32] Cash D, Jaeger J, Jarecki S, et al. Dynamic searchable encryption in very-large databases: Data structures and implementation. In: Network and Distributed System Security Symposium, Citeseer, 2014, 23–26.
[33] Lambregts S, Chen H, Ning J, et al. VAL: Volume and access pattern leakage-abuse attack with leaked documents. In: 27th European Symposium on Research in Computer Security, Springer, 2022, 653–676.
[34] Markatou EA and Tamassia R. Full database reconstruction with access and search pattern leakage. In: Information Security: 22nd International Conference, Springer, 2019, 25–43.
[35] Bösch C, Peter A, Leenders B, et al. Distributed searchable symmetric encryption. In: 2014 Twelfth Annual International Conference on Privacy, Security and Trust, IEEE, 2014, 330–337.
[36] Xu L, Yuan X, Wang C, et al. Hardening database padding for searchable encryption. In: IEEE INFOCOM 2019-IEEE Conference on Computer Communications, IEEE, 2019, 2503–2511.
[37] Paillier P. Public-key cryptosystems based on composite degree residuosity classes. In: International Conference on the Theory and Applications of Cryptographic Techniques, Springer, 1999, 223–238.
[38] Freedman MJ, Nissim K and Pinkas B. Efficient private matching and set intersection. In: International Conference on the Theory and Applications of Cryptographic Techniques, Springer, 2004, 1–19.
[39] Kissner L and Song D. Privacy-preserving set operations. In: Annual International Cryptology Conference, Springer, 2005, 241–257.
[40] Abadi A, Terzis S, Metere R, et al. Efficient delegated private set intersection on outsourced private datasets. IEEE Trans Depend Secur Comput 2017; **16**: 608–624.
[41] Sun SF, Liu JK, Sakzad A, et al. An efficient non-interactive multi-client searchable encryption with support for boolean queries. In: European Symposium on Research in Computer Security, Springer, 2016, 154–172.
[42] Kamara S, Moataz T and Ohrimenko O. Structured encryption and leakage suppression. In: Annual International Cryptology Conference, Springer, 2018, 339–370.
[43] Yang A, Weng J, Yang K, et al. Delegating authentication to edge: A decentralized authentication architecture for vehicular networks. IEEE Trans Intell Transp Syst 2022; **23**: 1284–1298.
[44] Zhang Y, Deng RH, Xu S, et al. Attribute-based encryption for cloud computing access control: A survey. ACM Comput Surv 2020; **53**: 1–41.

Axin Wu received his Ph.D. degree in the College of Cyber security from the Jinan University, Guangzhou, China, in 2023. He is currently a postdoctoral fellow in the State Key Laboratory of Cryptology, Beijing, China. His research interests include cloud security and wireless network security.

Dengguo Feng is currently a professor with the Institute of Software, Chinese Academy of Sciences, Beijing, China. His research interests include trusted computing, confidential computing, data security and privacy protection. He has authored or coauthored more than 200 publications in international journals and conferences, and presided over the research work and project formulation of more than 20 international and national standards in the field of information security technology.

Min Zhang received her Ph.D. degree from University of Chinese Academy of Sciences, China, in 2007. She is currently an professor of the TCA Lab., Institute of Software, Chinese Academy of Sciences, China. Her current research interests include data Security and privacy protection in Cloud Computing, Differential Privacy and confidential computing.

Jialin Chi received her B.S. degree from Shandong University, China, in 2012, and the Ph.D. degree from Institute of Software, Chinese Academy of Sciences, China, in 2018. She is currently a research assistant with Institute of Software, Chinese Academy of Sciences. Her research interests include data security and privacy computing.

Yinghui Zhang received his Ph.D. degree in Cryptography from Xidian University, China, in 2013. He is a professor at School of Cyberspace Security, National Engineering Research Center for Secured Wireless (NERCSW), Xi'an University of Posts & Telecommunications. He served for the program committee of several conferences and the editorial members of several international journals in information security. His research interests include public key cryptography, cloud security and wireless network security.