Published in ACM Transaction on Storage (2023) 19 (4). DOI: 10.1145/3607255

Owner-free Distributed Symmetric Searchable Encryption Supporting Conjunctive Queries

QIUYUN TONG, School of Cyber Engineering, Xidian University, China
XINGHUA LI, School of Cyber Engineering, Xidian University, China and Engineering Research Center of Big data Security, Ministry of Education, China
YINBIN MIAO and YUNWEI WANG, School of Cyber Engineering, Xidian University, China
XIMENG LIU, College of Computer and Data Science, Fuzhou University, China
ROBERT H. DENG, School of Information Systems, Singapore Management University, Singapore

Symmetric Searchable Encryption (SSE), as an ideal primitive, can ensure data privacy while supporting retrieval over encrypted data. However, existing multi-user SSE schemes require the data owner to share the secret key with all query users or always be online to generate search tokens. While there are some solutions to this problem, they have at least one weakness, such as non-supporting conjunctive query, result decryption assistance of the data owner, and unauthorized access. To solve the above issues, we propose an Owner-free Distributed Symmetric searchable encryption supporting Conjunctive query (ODiSC). Specifically, we first evaluate the Learning-Parity-with-Noise weak Pseudorandom Function (LPN-wPRF) in dual-cloud architecture to generate search tokens with the data owner free from sharing key and being online. Then, we provide fine-grained conjunctive query in the distributed architecture using additive secret sharing and symmetric-key hidden vector encryption. Finally, formal security analysis and empirical performance evaluation demonstrate that ODiSC is adaptively simulation-secure and efficient.

CCS Concepts: • **Security and privacy → Management and querying of encrypted data;**

Additional Key Words and Phrases: Symmetric searchable encryption, multi-user, conjunctive query, dual-cloud architecture

ACM Reference format:
Qiuyun Tong, Xinghua Li, Yinbin Miao, Yunwei Wang, Ximeng Liu, and Robert H. Deng. 2023. Owner-free Distributed Symmetric Searchable Encryption Supporting Conjunctive Queries. *ACM Trans. Storage* 19, 4, Article 38 (September 2023), 25 pages.
https://doi.org/10.1145/3607255

This work was supported by the National Natural Science Foundation of China (No. 62125205, No. 62072361), the Key Research and Development Program of Shaanxi (No. 2023KXJ-190), and the Fundamental Research Funds for the Central Universities (No. XJSJ23188, No. YJSJ23007).

Authors' addresses: Q. Tong, Y. Miao (corresponding author), and Y. Wang, School of Cyber Engineering, Xidian University, Xi'an, China, 710071; emails: qytong0820@163.com, ybmiao@xidian.edu.cn, wywxidian@foxmail.com; X. Li (corresponding author), School of Cyber Engineering, Xidian University, Xi'an, China, 710071, and Engineering Research Center of Big Data Security, Ministry of Education, Xi'an, China, 710071; email: xhli1@mail.xidian.edu.cn; X. Liu, College of Computer and Data Science, Fuzhou University, Fuzhou, China, 350108; email: snbnix@gmail.com; R. H. Deng, School of Information Systems, Singapore Management University, Singapore, Singapore, 188065; email: robertdeng@smu.edu.sg.

1553-3077/2023/09-ART38 $15.00
https://doi.org/10.1145/3607255

1 INTRODUCTION

Cloud computing, as a promising computing paradigm, motivates resource-constrained clients to outsource their data for cost saving and flexibility. However, **Cloud Service Providers (CSPs**, e.g., Amazon AWS, Microsoft Azure, etc.) are often treated as untrusted entities and may leak the private data [29]. **Symmetric Searchable Encryption (SSE)** [27] is an ideal primitive to ensure data privacy while supporting ciphertext retrieval. A natural extension of SSE is the multi-user setting, where an arbitrary group of users can issue queries [10]. Existing **Multi-user SSE (MSSE)** schemes have realized more expressive keyword search, among which conjunctive query is an important function and has been widely used in reality. For example, a doctor wants to find the medical records containing two query keywords *fever and hyposmia* from an electronic medical record database.

But existing MSSE schemes [6, 14, 17, 20, 31, 32, 35, 39, 42] cannot achieve "owner-free," as shown in Figure 1. Specifically, in the previous MSSE schemes [17, 20, 32, 35, 39], the data owner shares the secret key with query users to generate search tokens seen in Figure 1(a). Since each query user has access to the secret key, the adversary (usually CSP) will obtain the key by corrupting any one query user, which seriously increases the risk of key leakage and thus exposes the whole outsourced database. In Figure 1(b), some other MSSE schemes [6, 14, 31, 42] perform a per-query interaction between the data owner and each query user, such that the query user obtains a search token without learning the secret key, while the data owner learns nothing about the search query. Nevertheless, the data owner must be online at all times, and considerable computation and communication burdens are incurred on it, which defeats the initial purpose of data outsourcing.

Recently, the combination of secret sharing and dual-cloud architecture in the previous schemes [8, 19, 26, 37] has been used to achieve privacy-preserving owner-free retrieval. In the architecture, both outsourced data and search queries are additively secret-shared among two non-colluding CSPs, who then execute a series of secure **Multi-parity Computation (MPC)** protocols to achieve secure top-k queries. However, there are several weaknesses if such implementation is directly applied to MSSE, where the secret key instead of outsourced data is secretly shared among two non-colluding CSPs. First, such implementation is only used to support similar sequence query and k-nearest neighbor classification. If it is used for index-based retrieval, it only supports top-k queries, not conjunctive queries. Additionally, decryption of desired results still requires the assistance of the data owner.[1] Second, without constraint of key-based authorization, any query user is able to retrieve the entire database, which poses a significant risk of unauthorized access.[2]

To address the above issues, we propose an owner-free distributed[3] SSE scheme supporting conjunctive queries (called ODiSC). We evaluate the **Learning-Parity-with-Noise weak Pseudorandom Function (LPN-wPRF)** [13] under the dual-cloud architecture to generate search tokens, enabling MSSE to be jointly implemented by two clouds when the data owner becomes free from sharing key and being online. Moreover, we combine **Additive Secret Sharing (ASS)** [12] and **Symmetric-key Hidden Vector Encryption (SHVE)** [35] to support fine-grained conjunctive queries in the dual-cloud/distributed architecture, such that unauthorized access is prevented

[1]The combination of secret sharing and dual-cloud architecture only frees the data owner from encrypting search queries. The secret keys used to encrypt the outsourced database remain exclusively possessed by the data owner. Consequently, decryption of the ciphertexts retrieved by the query user requires the assistance of the data owner.

[2]Unauthorized access means that the search result set $\mathcal{R}$ contains the ciphertext c_D of such document, which matches the query Q regardless of whether its access policy Γ_D matches the attribute set S. For example, given that the documents D_1, D_2, D_3, D_4, D_5 match the query Q, and that the access policies of the documents D_4, D_5 match the attribute set S, we have $\mathcal{R} = \{c_{D_1}, c_{D_2}, c_{D_3}, c_{D_4}, c_{D_5}\}$, and the documents D_1, D_2, D_3 are beyond the query user's authorization.

[3]To distinguish from traditional SSE in the single-cloud architecture, we use distributed SSE to represent SSE in the dual-cloud architecture.

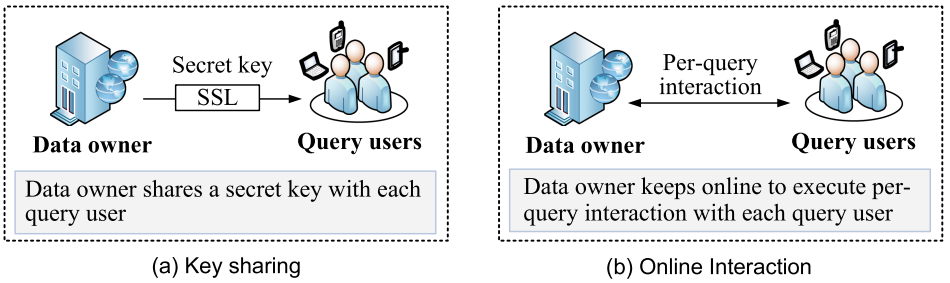


Fig. 1. Limitations in existing MSSE schemes.

and decryption of desired results is realized without the assistance of the data owner. The main contributions of our work are summarized as follows:

- We propose an owner-free MSSE scheme in the dual-cloud architecture. Specifically, we build an MPC protocol for LPN-wPRF, in which two CSPs take the secret-shared secret key and a secret-shared search query as input to output a secret-shared search token while keeping the secret key and search query secret.
- We design a privacy-preserving fine-grained conjunctive query. Specifically, a fine-grained conjunctive query is converted into vector matching using a Bloom filter and hidden vector-based access structure, and then secure vector matching is performed using ASS and SHVE. It makes up for the weaknesses brought by the dual-cloud architecture.
- We give a formal security analysis to show that ODiSC is adaptively simulation-secure. We also conduct extensive experiments using the real-world dataset to demonstrate that ODiSC is efficient and feasible for practical applications, whose search time is less than 1 second for 10,000 data.

The remaining article is organized as follows: In Sections 2 and 3, we review the existing literature and relevant background knowledge, respectively. Section 4 describes the system model, threat model, problem definition, security definition, and design goals. In Section 5, we present the technical overview and concrete construction of ODiSC and discuss the identity impersonation in it. Sections 6 and 7 present the security and performance evaluations of our proposed scheme, respectively. Section 8 concludes this article.

2 RELATED WORK

MSSE is first formally defined and constructed by Curtmola et al. [10]. The scheme first builds an inverted index to support efficient single keyword search, then is extended to MSSE by sharing the secret key with a group of authorized users based on broadcast encryption. After that, there are many follow-up works proposed.

Similar to [10], some MSSE works [17, 20, 32, 35, 39] use the idea of key sharing. The schemes [20, 32, 35] support multi-keyword ranked search, Boolean range query, and fuzzy multi-keyword search in the multi-user setting also via broadcast encryption. In these schemes, authorized users can access the entire database. Zhang et al. [39] achieved fine-grained Boolean query by combining SSE primitive with the attribute-based authorization policy, where partial secret keys are shared with all query users. If the adversary corrupts a query user, he or she can use the partial secret key to decrypt the partial index, which results in data privacy leakage. Li et al. [17] provided efficient search privilege refinement by utilizing the polynomial-based access strategy and achieved highly accurate ciphertext retrieval by combining **Term Frequency-Inverse Document Frequency (TF-IDF)** with the secure **k-Nearest Neighbor (kNN)** technique. In

these schemes, a query user may decrypt others' search tokens, resulting in query privacy leakage. To solve it, Tong et al. [30] distributed different secret keys to query users and used the dual secure kNN technique to convert the multi-key trapdoors into the same-key trapdoors.

Other works [6, 14, 31, 42] achieve MSSE by performing an interaction between the data owner and the query user in each query. Specifically, Jarecki et al. [14] executed **Oblivious PRF (OPRF)** between the data owner and each query user to generate a search token without disclosing the secret key and plaintext query to each other. In addition, it also allowed CSP to verify that the search tokens are authorized by the data owner using homomorphic signature. Cash et al. [6] used the same method to support dynamic multi-user ciphertext retrieval. The schemes [31, 42] designed a trapdoor generation protocol to encrypt a query point with a round of communication between the data owner and the query user, where noise addition is introduced to mask plaintext query and secret key. However, they cannot resist the known plaintext attack. In these interaction-based schemes, the data owner needs to stay online all the time to receive and encrypt each query request, which defeats the initial purpose of data outsourcing. If a large number of queries are issued in a short time, the data owner will become the bottleneck of the system.

The schemes [5, 8, 15, 19, 26, 28, 37, 41] solved the contradiction between key sharing and online interaction. In the scheme [41], query users locally encrypted search queries using the limited information about the secret key, which claims to be secure against known-plaintext attacks. Sun et al. [28] proposed a non-interactive fine-grained Boolean query based on RSA function and **Attribute-based Encryption (ABE)**, where the access control works on an encrypted database. Query users would infer some sensitive information (e.g., theme) from the documents that match query keywords but are inaccessible, thereby damaging inaccessibility. Kermanshahi et al. [15] secretly shared the secret key to authorized query users using the Shamir secret sharing and allowed query users to compute search tokens by using **Randomizable Distributed key-homomorphic PRF (RDPRF)**. However, the decryption of search results still needs the help of the data owner. Cheng et al. [8] used ASS to split each genomic sequence and search query into two shares, then executed a series of secure MPC protocols to support secure **Similar Sequence Query (SSQ)** in the model involving two non-colluding and honest-and-curious clouds. It allows any query user to retrieve the entire database, which leads to unauthorized access. In addition, the number of communication rounds in the search process increases linearly with the size of the outsourced database, which brings huge communication overhead. The schemes [19, 26, 37] used the same system model to support faster secure SSQ, privacy-preserving kNN classification, and encrypted image retrieval, respectively. Boneh et al. [5] implemented an MPC-friendly PRF protocol among two or more servers to obtain the decryption key for the database entries matching the selected query keyword. In this scheme, unauthorized access will occur since anyone is allowed to retrieve the entire database, whether they have access or not. Note that asymmetric SE schemes [9, 23, 34] can easily enable owner-free retrieval, but they rely on a fully trusted authority and incur significant computation overhead. In this article, we focus on how to implement owner-free MSSE. Compared with existing schemes, our proposed scheme has versatile features demonstrated in Table 1.

3 PRELIMINARIES

In this section, we introduce some related knowledge used in our work, which includes additive secret sharing [12], LPN-wPRF [13], and SHVE construction [16]. Table 2 gives a summary of notations used in the article.

3.1 Additive Secret Sharing

Two-out-of-two ASS is defined over a field $\mathbb{Z}_p$. ASS splits a secret $x \in \mathbb{Z}_p$ into a pair of random shares $\langle x \rangle = \{\langle x \rangle_0, \langle x \rangle_1\} \subseteq \mathbb{Z}_p$ such that $\langle x \rangle_0 + \langle x \rangle_1 \equiv x \mod p$ and only the party P_ℓ knows $\langle x \rangle_\ell$.

Table 1. Comparative Summary between ODiSC and Existing Schemes

Schemes	Search Type	Avoiding Key Sharing	Offline Data Owner	Fine-grained Access Control
[14]	Boolean	OPRF	✗	Signature
[17]	kNN (TF-IDF)	✗	Key sharing	Polynomial
[31]	kNN (Euclidean)	Mask	✗	✗
[39]	Boolean	✗	Key sharing	ABE
[5]	Single	MPC-friendly PRF†		✗
[15]	Boolean	Shamir + RDPRF†		✗
[26]	kNN (edit)	ASS + Dual-cloud†		✗
[28]	Boolean	RSA function†		ABE
ODiSC	Conjunctive	Distributed LPN-wPRF†		SHVE

Notes. "†": The technique can be used to achieve privacy-preserving retrieval, where key sharing and online data owner are both avoided.

Table 2. Notation Descriptions

Notations	Descriptions
$[i]$	$1, 2, \ldots, i$
$\mathbf{0}$	All-zeros vector
$\mathbf{1}$	All-ones vector
d	Number of attributes
N_{B}	Length of Bloom filter
N	Size of outsourced database DB
M	Size of keyword dictionary W
Q	Search query $Q = \{w_{j_1}, w_{j_2}, \ldots, w_{j_q}\} \subseteq W$
K	Secret key for index and token generation
sk	Secret key used to encrypt the outsourced data
$\circledast$	Hadamard (component-wise) product
$\langle x \rangle_\ell$	Additive secret sharing of x held by party P_ℓ
$\mathsf{DB}(w)$	Document set containing the keyword w
$\mathsf{Add}_p(u, v)$	$u + v \mod p$
$\mathsf{Lin}_p^B(x)$	$Bx \mod p$ where B is public
$\mathsf{BL}_p(B, x)$	$Bx \mod p$ where B is secret
$\mathsf{Convert}_{(p,q)}(x)$	Convert $x \in \mathbb{Z}_p$ into $x \in \mathbb{Z}_q$
$\mathsf{TopkSim}(D, Q) = 1$	Similarity of document D and search query Q is top-k
$\mathsf{AcMatch}(\Gamma_D, S) = 0$	Access policy Γ_D does not match attribute set S

To reconstruct the secret x, the parity P_ℓ collects all shares and computes $x = \langle x \rangle_0 + \langle x \rangle_1$ over $\mathbb{Z}_p$. ASS is additive homomorphic, i.e., $\langle x + y \rangle = \langle x \rangle + \langle y \rangle$. For multiplication, given $\langle x \rangle$ and $\langle y \rangle$, the parties jointly compute the secret shares of $z = xy$ based on Beaver's technique [2].

3.2 LPN-wPRF

LPN-wPRF is constructed by mixing linear functions over different moduli. The motivation is to maximize simplicity and minimize complexity in MPC applications. LPN-wPRF is a function $F_\lambda : \mathbb{Z}_2^{m \times n} \times \mathbb{Z}_2^n \to \mathbb{Z}_2^t$ with key-space, input space, and output space in $\mathbb{Z}_2^{m \times n}$, $\mathbb{Z}_2^n$, and $\mathbb{Z}_2^t$, respectively, where m, n, t are the functions of security parameter λ. The concrete construction is shown in Figure 2, which is succinctly represented using five basic gates: mod-2 linear gate Lin_2^B, mod-2

LPN-wPRF

For a secret key $K \in \mathbb{Z}_2^{m\times n}$ and a public matrix $B \in \mathbb{Z}_2^{t\times m}$, we define $F_K^B(x) = F_\lambda(K, x)$ as follows:

(1) On input $x \in \mathbb{Z}_2^n$, compute $u = \mathsf{BL}_2(K, x)$;
(2) Compute $K^* = \mathsf{Convert}_{(2,3)}(K), x^* = \mathsf{Convert}_{(2,3)}(x)$;
(3) Compute $v = \mathsf{Convert}_{(3,2)}(\mathsf{BL}_3(K^*, x^*))$;
(4) Compute and output $y = \mathsf{Lin}_2^B(\mathsf{Add}_2(u, v))$.

Fig. 2. Concrete construction of LPN-wPRF.

SHVE construction

- SHVE.Setup(1^λ): Given the security parameter λ, the algorithm returns the master secret key $msk \xleftarrow{\$} \{0,1\}^\lambda$ and the message space $\mathcal{M} = \{\text{“}True''\}$.
- SHVE.KeyGen$(msk, \boldsymbol{v})$: Given a predicate vector $\boldsymbol{v} = (v_1, v_2, \ldots, v_s) \in \Sigma_*^s$, the algorithm generates

$$d_0 = \oplus_{i\in Y}(F_0(msk, v_i||i)) \oplus sk,\ d_1 = \mathsf{Sym.Enc}(sk, 0^{\lambda+\log\lambda}), \tag{1}$$

  where $Y = \{i \in [s] | v_i \neq *\}, F_0 : \{0,1\}^\lambda \times \{0,1\}^{\lambda+\log\lambda} \to \{0,1\}^{\lambda+\log\lambda}$ is a PRF function, and $sk \xleftarrow{\$} \{0,1\}^{\lambda+\log\lambda}$ is the secret key for a symmetric encryption scheme (Sym.Enc, Sym.Dec). Then, the algorithm outputs the decryption key $\mathbf{s} = (d_0, d_1, Y)$.
- SHVE.Enc$(msk, \mu, \boldsymbol{x})$: Given an index vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_s) \in \Sigma^s$ and a message μ, the algorithm outputs the ciphertext $\boldsymbol{y} = (y_1, y_2, \ldots, y_s)$, where

$$y_i = F_0(msk, x_i||i), i \in [s]. \tag{2}$$

- SHVE.Dec$(\mathbf{s}, \boldsymbol{y})$: The algorithm computes

$$sk' = (\oplus_{i\in Y} y_i) \oplus d_0,\ \mu' = \mathsf{Sym.Dec}(sk', d_1). \tag{3}$$

  If $\mu' = 0^{\lambda+\log\lambda}$, it outputs “$True''$” and $\perp$ otherwise.

Fig. 3. Detailed procedure of SHVE construction.

addition gate Add_2, mod-p bilinear gate BL_p, $\mathbb{Z}_2 \to \mathbb{Z}_3$ conversion gate $\mathsf{Convert}_{(2,3)}$, and $\mathbb{Z}_3 \to \mathbb{Z}_2$ conversion gate $\mathsf{Convert}_{(3,2)}$. To improve efficiency and enable short key, the secret key K can be taken as a structured matrix (e.g., uniformly random Toeplitz matrix, generator matrix of quasi-cyclic codes) rather than a uniformly random matrix, which reduces the size of the secret key from mn to $m + n$ or n. For λ-bit security, it is suggested to set $m = n = 2\lambda, t = \lambda$.

3.3 SHVE Construction

SHVE construction is a lightweight predicate encryption scheme used to solve the resulting pattern leakage problem in the **Oblivious Cross-tags (OXT)** [7]. It is defined over $\Sigma_* = \Sigma \cup \{*\}$, where Σ denotes a finite set of attributes and “$*$” denotes a wildcard symbol not in Σ. The detailed procedure of SHVE construction is shown in Figure 3.

4 PROBLEM FORMULATION

In this section, we introduce the system model, threat model, problem definition, security definition, and design goals of ODiSC.

4.1 System Model and Threat Model

In this article, we consider a document database outsourcing scenario. As shown in Figure 4, the system model of ODiSC mainly consists of three entities: **Data Owner (DO)**, **Query Users (QUs)**, and two **Cloud Service Providers (CSPs)**. The role of each entity is described as follows:

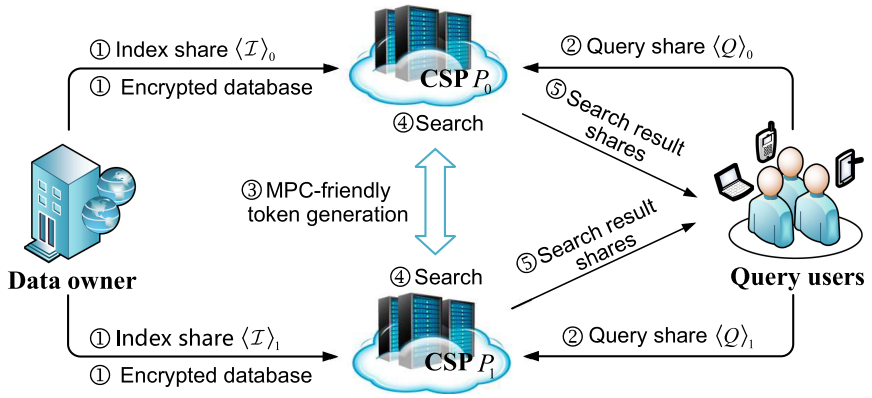


Fig. 4. System model of ODiSC.

- *Data owner.* The DO constructs a secure index for the document database, then secretly shares it among CSPs P_0, P_1.
- *Query users.* The QU wants to perform conjunctive query over the encrypted document database. He or she secretly shares his or her search query among CSPs P_0, P_1.
- *Cloud service providers.* In addition to generating search tokens for QUs, reliable[4] CSPs P_0, P_1 also provide storage and search services for DO and QUs, respectively.

In a smart medical setting, the hospital can act as a DO, who plans to outsource medical data to two CSPs. Before encrypting the database, it first builds a secure index and splits it into two shares $\langle \mathcal{I} \rangle_0, \langle \mathcal{I} \rangle_1$, then uploads an index share and the encrypted database to CSP P_ℓ for $\ell \in \{0, 1\}$ (Step ①). When the QU (e.g., patient, research institution) attempts to issue a search query Q, he or she splits it into two shares $\langle Q \rangle_0, \langle Q \rangle_1$ and then submits them to CSPs P_0, P_1, respectively (Step ②). Upon receiving the search request, CSPs P_0, P_1 first interactively execute the MPC-friendly token generation protocol to generate a secret-shared search token (Step ③). Then, each CSP locally evaluates the stored index share using a respective search token share to obtain the search result shares (Step ④) and returns them to the QU (Step ⑤). Finally, the QU reconstructs search results to obtain desired documents. The storage system with dual-cloud architecture not only reduces the risk of document data loss through cross-cloud backup but also enables privacy-preserving retrieval by distributing index and query data across two CSPs. Moreover, it eliminates the need for the DO to perform key management and remain online during retrieval.

Threat Model. Similar to [8, 19, 31], the entities in the system model are assumed to be honest-but-curious. They honestly follow the established protocols, but the DO is curious about QUs' query contents, QUs are curious about unauthorized data, and CSPs are curious to infer valuable information from the outsourced data. Moreover, we assume that two CSPs cannot collude with each other or be compromised simultaneously, which means that they should not reveal any extra information beyond what is specified by the protocols.

Remark. According to the 2023 State of the Cloud Report[5] published by Flexera, a well-known software asset management company, few enterprises are now locked in a single cloud provider, and a vast majority of enterprises have embraced multi-cloud strategies, with 87% of them

[4]The dual-cloud architecture, utilizing replication [36] and erasure code [18], can provide fault tolerance for stored data. Thus, we assume that CSPs are reliable and can guarantee data persistence and availability without loss. Our focus remains on owner-free MSSE supporting conjunctive query.

[5]https://info.flexera.com/CM-REPORT-State-of-the-Cloud?lead_source=Website%20Visitor&id=Flexera.com-PR

implementing a multi-cloud approach and 72% using a hybrid cloud approach. This indicates that the dual-cloud architecture is practical. Furthermore, the assumption of two non-colluding CSPs is common in the secret-sharing-based data processing schemes such as [8, 19, 21, 38, 40]. In practice, the architecture of two non-colluding CSPs is realistic due to business reputation and competitive relationship. Once caught colluding with each other, they will lose massive market shares as customers no longer trust them. For outsider adversaries, the probability of compromising two CSPs at the same time is significantly low as CSPs adopt different defense strategies to resist network attacks. Thus, we just consider insider adversaries, i.e., CSPs.

4.2 Problem Definition

In MSSE, a certain DO uses the secret key K to construct a secure index for a document database $\mathsf{DB} = \{D_1, D_2, \ldots, D_N\}$. Before sending a search request to the CSP, the search query $Q = \{w_{j_1}, w_{j_2}, \ldots, w_{j_q}\}(q \geq 2)$ of each QU needs to be encrypted using the secret key K. One way is that the DO shares K with each QU, denoted as $f_{\mathsf{KeyShare}} = 1$, which increases the risk of key leakage and thus exposes the entire outsourced database. Another way is that the DO remains online to execute a per-query interaction with each QU, denoted as $f_{\mathsf{Online}} = 1$, which defeats the initial purpose of data outsourcing. Although the combination of secret sharing and dual-cloud architecture in the previous schemes can be used to achieve privacy-preserving owner-free retrieval (i.e., $f_{\mathsf{KeyShare}} = 0, f_{\mathsf{Online}} = 0$), it has the following weaknesses if directly applied to MSSE:

- A conjunctive query $w_{j_1} \wedge w_{j_2} \wedge \cdots \wedge w_{j_q}$ is not supported.
- Decryption of each desired result c_D requires the assistance of the DO, where $c_D = \mathsf{Sym.Enc}(sk_D, D)$ and the secret key sk_D is only owned by the DO.
- The desired result set $\mathcal{R}$ may contain the documents beyond the QU's authorization, i.e., $\mathcal{R} = \{c_D | \mathsf{TopkSim}(D, Q) = 1, \mathsf{AcMatch}(\Gamma_D, S) = 1\} \cup \{c_D | \mathsf{TopkSim}(D, Q) = 1, \mathsf{AcMatch}(\Gamma_D, S) = 0\}$, where Γ_D is the access policy of the document D and S is the QU's attribute set.

Therefore, in this article, we aim to propose a MSSE scheme satisfying

$$\mathcal{R} \leftarrow \mathsf{MSSE}(\mathsf{DB}, Q, K)$$
$$s.t. \begin{cases} f_{\mathsf{KeyShare}} = 0, f_{\mathsf{Online}} = 0, \\ \mathcal{R} = \{c_D | Q \subseteq D, \mathsf{AcMatch}(\Gamma_D, S) = 1\}, \\ sk_D \text{ can be obtained by QU if } c_D \in \mathcal{R}. \end{cases} \tag{4}$$

4.3 Security Definition

We consider the simulation-based definition introduced by [11, 25], where the adversary $\mathcal{A}$ is a certain CSP P_ℓ in the honest-but-curious model. Thus, for the security of ODiSC, we need to ensure that the view of the adversary $\mathcal{A}$ is simulatable, such that the adversary $\mathcal{A}$ cannot computationally distinguish between the simulated view $\mathsf{View}_{\mathcal{A},\mathsf{SIM}}$ and the real view $\mathsf{View}_{\mathcal{A},\mathsf{REAL}}$ with a non-negligible probability. We now define the experiments **SIM** and **REAL** in Figure 5. Here, $\mathcal{L}_1(\mathsf{DB})$ denotes the database leakage function, where $\mathcal{A}$ knows the size of database N, the size of keyword dictionary M, the size of each index list N_w, the number of attributes d, and the number of 0-ciphertext and 1-ciphertext in each column of each index list $\{n_1^{\varrho,j}, n_2^{\varrho,j}\}_{\varrho \in [M], j \in [N_\mathsf{B}]}$. $\mathcal{L}_2(\mathsf{DB}, Q, S)$ denotes the search leakage function, where $\mathcal{A}$ learns which entry of the inverted index is retrieved for the specific query keyword and whether the same user attribute was uploaded in the past or not. Then, the security of ODiSC is defined as:

Definition 1. Our ODiSC scheme $\Pi = \{\mathsf{Setup}, \mathsf{IndexBulid}, \mathsf{QueryIssue}, \mathsf{Search}\}$ is said to be adaptively simulation-secure if for any PPT adversary $\mathcal{A}$ making polynomial $q(\lambda)$-times queries and a

$\textsc{Real}^{\Pi}_{\mathcal{A}}$: The real experiment between a challenger C and a Probabilistic Polynomial Time (PPT) adversary $\mathcal{A}$ is shown as follows:

(1) C runs $\mathsf{Setup}(1^{\lambda})$ to generate the parameter $\mathcal{P}$ and secret key K, then sends $\langle \mathcal{P} \rangle_{\ell}$ to $\mathcal{A}$.
(2) $\mathcal{A}$ chooses a database $\mathsf{DB} = \{D_1, D_2, \ldots, D_N\}$ and their policy vector set $A = \{\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_N\}$, then sends them to C.
(3) C runs $\mathsf{IndexBulid}(K, \mathsf{DB}, A)$ to output the index shares $\langle \mathcal{I} \rangle$ and encrypted database EDB, then sends $\langle \mathcal{I} \rangle_{\ell}, \mathsf{EDB}$ to $\mathcal{A}$.
(4) $\mathcal{A}$ chooses a search query Q_1 and an attribute set S_1, then runs $\mathsf{QueryIssue}(Q_1)$ to generate the query shares $\langle Q_1 \rangle$, finally sends $S_1, \langle Q_1 \rangle_{1-\ell}$ to C. There is a restriction that S_1 cannot satisfy A.
(5) C runs SHVE.Enc to encrypt S_1 as Eu_1 and sends it to $\mathcal{A}$. In addition, C and $\mathcal{A}$ cooperate to execute *token generation* to obtain the search token shares $\langle \mathsf{tk}_1 \rangle$.
(6) Based on $\langle \mathcal{P} \rangle_{\ell}, \langle \mathcal{I} \rangle_{\ell}, \mathsf{EDB}, \{\langle \mathsf{tk}_{\iota} \rangle_{\ell}, \mathsf{Eu}_{\iota}\}_{\iota \in [i-1]}$ for $i \geq 2$, $\mathcal{A}$ chooses a search query Q_i and along with an attribute set S_i, then sends $S_i, \langle Q_i \rangle_{1-\ell}$ to C. There is a restriction that S_i cannot satisfy A.
(7) C executes the step (5) with $S_i, \langle Q_i \rangle_{1-\ell}$ as input and returns Eu_i to $\mathcal{A}$.
(8) Until the query reaches the polynomial number of times $q(\lambda)$, $\mathcal{A}$ outputs the view $\mathsf{View}_{\mathcal{A},\textsc{Real}} = \{\langle \mathcal{P} \rangle_{\ell}, \langle \mathcal{I} \rangle_{\ell}, \mathsf{EDB}, \{\langle \mathsf{tk}_{\iota} \rangle_{\ell}, \mathsf{Eu}_{\iota}\}_{\iota \in [q(\lambda)]}\}$.

$\textsc{Sim}^{\Pi}_{\mathcal{A},\mathcal{S}}$: The simulation experiment between a simulator $\mathcal{S}$ and a PPT adversary $\mathcal{A}$ is shown as follows:

(1) $\mathcal{S}$ simulates and sends the paramater share $\langle \mathcal{P} \rangle_{\ell}$ to $\mathcal{A}$.
(2) $\mathcal{A}$ outputs a database DB and their policy vectors A.
(3) Given the leakage $\mathcal{L}_1(\mathsf{DB})$, $\mathcal{S}$ generates $\langle \mathcal{I} \rangle_{\ell}$ and EDB, and sends them to $\mathcal{A}$.
(4) $\mathcal{A}$ outputs Q_1 along with S_1. There is a restriction that S_1 cannot satisfy A.
(5) Given the leakage $\mathcal{L}_2(\mathsf{DB}, Q_1, S_1)$, $\mathcal{S}$ simulates $\mathsf{Eu}_1, \langle \mathsf{tk}_1 \rangle$, and sends them to $\mathcal{A}$.
(6) Based on $\langle \mathcal{P} \rangle_{\ell}, \langle \mathcal{I} \rangle_{\ell}, \mathsf{EDB}, \{\langle \mathsf{tk}_{\iota} \rangle_{\ell}, \mathsf{Eu}_{\iota}\}_{\iota \in [i-1]}$ for $i \geq 2$, $\mathcal{A}$ outputs Q_i along with S_i. There is a restriction that S_i cannot satisfy A.
(7) Given $\mathcal{L}_2(\mathsf{DB}, Q_1, \ldots, Q_i, S_1, \ldots, S_i)$, $\mathcal{S}$ simulates Eu_i and $\langle \mathsf{tk}_i \rangle_{\ell}$, then sends them to $\mathcal{A}$.
(8) Until the query reaches $q(\lambda)$, $\mathcal{A}$ outputs $\mathsf{View}_{\mathcal{A},\textsc{Sim}} = \{\langle \mathcal{P} \rangle_{\ell}, \langle \mathcal{I} \rangle_{\ell}, \mathsf{EDB}, \{\langle \mathsf{tk}_{\iota} \rangle_{\ell}, \mathsf{Eu}_{\iota}\}_{\iota \in [q(\lambda)]}\}$.

Fig. 5. The real and simulation experiments.

PPT distinguisher $\mathcal{D}$, there exists an efficient simulator $\mathcal{S}$ such that

$$|\Pr[\mathcal{D}(\mathsf{View}_{\mathcal{A},\textsc{Real}}) = 1] - \Pr[\mathcal{D}(\mathsf{View}_{\mathcal{A},\textsc{Sim}}) = 1]| \leq \mathsf{neg}(\lambda). \tag{5}$$

In layman's terms, adaptive simulation security means that the view of adversary $\mathcal{A}$ (CSP P_{ℓ}) can be simulated in a way that is indistinguishable from the real view when the adversary $\mathcal{A}$ is given the ability to adaptively choose each future query based on previously chosen queries as well as their search tokens and search results.

4.4 Design Goals

ODiSC should achieve the following goals in terms of functionality, security, efficiency, and accuracy:

- *Owner-free fine-grained conjunctive query.* Conjunctive query can be done when the DO is free from sharing the secret key and being online. Moreover, the search results can be decrypted if and only if the QU's attributes satisfy their access policies.
- *Adaptive simulation security.* Adaptive CSPs should not obtain the shares owned by each other or infer underlying plaintext information from the knowledge they have.
- *Efficient retrieval.* The search time complexity of our scheme should be insensitive to the size of the outsourced database, making it suitable for large-scale data retrieval scenarios.

Table 3. Distributed Computation of Five Circuit Gates in LPN-wPRF

Protocols	Public Inputs	Shared Inputs	Correlated Randomness	Output Shares
$\pi_{\text{Lin}}^{B,2}$	B	$\langle w\rangle_\ell$	—	$\langle y\rangle_\ell = B \cdot \langle w\rangle_\ell \mod 2$
π_{Add}^{2}	—	$\langle u\rangle_\ell, \langle v\rangle_\ell$	—	$\langle w\rangle_\ell = \langle u\rangle_\ell + \langle v\rangle_\ell \mod 2$
π_{BL}^{p}	$\hat{K}, \hat{x}$	—	$\langle\tilde{K}\rangle_\ell, \langle\tilde{x}\rangle_\ell, \left\langle\tilde{K}\tilde{x}\right\rangle_\ell$	$\langle u\rangle_\ell = \ell \cdot \hat{K}\hat{x} - \hat{K}\langle\tilde{x}\rangle_\ell - \hat{x}\left\langle\tilde{K}\right\rangle_\ell + \left\langle\tilde{K}\tilde{x}\right\rangle_\ell$
$\pi_{\text{Convert}}^{(2,3)}$	$\hat{x}$ over $\mathbb{Z}_2$	—	$\tilde{r}_1 = \tilde{x} \mod 3$ $\tilde{r}_2 = \tilde{x} + \mathbf{1} \mod 2 \mod 3$	$\langle x^*\rangle_\ell = (\hat{x} \oplus \mathbf{1}) \circledast \langle\tilde{r}_1\rangle_\ell + \hat{x} \circledast \langle\tilde{r}_2\rangle_\ell \mod 3$
$\pi_{\text{Convert}}^{(3,2)}$	$\hat{v}^*$ over $\mathbb{Z}_3$	—	$\bar{r}_1 = \bar{v} \mod 2$ $\bar{r}_2 = \bar{v} + \mathbf{1} \mod 3 \mod 2$	$\langle v\rangle_\ell[i] = \ell \oplus (\oplus_{j=1}^{2}\langle\bar{r}_j\rangle_\ell[i])$ if $\hat{v}^*[i] = 0$ $\langle v\rangle_\ell[i] = \langle\bar{r}_2\rangle_\ell[i]$ if $\hat{v}^*[i] = 1$ $\langle v\rangle_\ell[i] = \langle\bar{r}_1\rangle_\ell[i]$ if $\hat{v}^*[i] = 2$

- *Accurate search result.* Our scheme should not compromise accuracy while preserving privacy, and its retrieval accuracy should be comparable to plaintext retrieval.

5 OUR PROPOSED SCHEME

In this section, we first introduce the technical overview of ODiSC, then specify corresponding concrete construction, and finally extend ODiSC to consider CSPs impersonating QUs' identities to break data privacy.

5.1 Technical Overview

As described in Section 4.2, traditional MSSE schemes generate search tokens via key sharing or online interaction, which incurs an increased key leakage risk or defeats the initial purpose of data outsourcing. Although the combination of ASS and dual-cloud architecture, Shamir secret sharing and RDPRF, and RSA function and MPC-friendly PRF can be used to realize privacy-preserving owner-free retrieval, they have one of the following weaknesses: non-supporting conjunctive query, result decryption assistance of the DO, and unauthorized access. Therefore, in this article, we aim to propose an MSSE scheme satisfying Equation (4). To do this, we first introduce a distributed LPN-wPRF protocol $\pi_{\text{DLPN-wPRF}}$, which is a building block of search token generation with $f_{\text{KeyShare}} = 0, f_{\text{Online}} = 0$.

The distributed LPN-wPRF protocol $\pi_{\text{DLPN-wPRF}}$ is to evaluate LPN-wPRF in dual-cloud architecture using a secret-shared key and a secret-shared element as input to obtain a secret-shared output. We first need to evaluate the five gates of LPN-wPRF in the distributed architecture, denoted as linear gate protocol $\pi_{\text{Lin}}^{B,2}$, addition gate protocol π_{Add}^{2}, bilinear gate protocol π_{BL}^{p}, $\mathbb{Z}_2 \to \mathbb{Z}_3$ conversion protocol $\pi_{\text{Convert}}^{(2,3)}$, and $\mathbb{Z}_3 \to \mathbb{Z}_2$ conversion protocol $\pi_{\text{Convert}}^{(3,2)}$. Their concrete constructions are shown in Table 3, which are derived from [13]. Then, we compose them to construct $\pi_{\text{DLPN-wPRF}}$ with a preprocessing phase. The detailed procedure is shown in Figure 6.

After that, we realize the privacy-preserving fine-grained conjunctive query without decryption assistance or unauthorized access, which can be divided into the following two steps:

(1) Converting conjunctive query into secure vector matching. Each document D with access policy Γ_D is represented as an N_{B}-bit Bloom filter $\boldsymbol{b}$, obtained by mapping each keyword in D to $\boldsymbol{b}$. Similarly, each search query Q from QU with the attribute set S is represented as an N_{B}-bit Bloom filter $\boldsymbol{q}$. If $\boldsymbol{q}[i] = \boldsymbol{b}[i]$ or $\boldsymbol{q}[i] = 0$ for $\forall i \in [N_{\text{B}}]$ (i.e., $Q \subseteq D$), the document D is a conjunctive query result of the query Q. For secure vector matching, we first generate the secret-shared search token $\langle\text{tk}\rangle$ using the protocol $\pi_{\text{DLPN-wPRF}}$. To prevent privacy leakage during the retrieval process, we take the shares $\langle H(\boldsymbol{q}[i]||i)\rangle$ as input when $\boldsymbol{q}[i] = 1$. Otherwise, we select a random element $r_i \in \mathbb{Z}_2^n$ and take $\langle r_i\rangle$ as input. Then, we encrypt the data vector $\boldsymbol{b}$ to obtain the index ciphertext $\text{Eb} = (y_1, y_2, \ldots, y_{N_{\text{B}}})$ via SHVE.Enc, where the PRF function F_0 is replaced with LPN-wPRF F_K^B. To eliminate the noises $\{r_i | \boldsymbol{q}[i] = 0\}$ introduced in the search token generation, we split Eb into two

Distributed LPN-wPRF protocol $\pi_{\text{DLPN-wPRF}}$

procedure: DLPN-wPRF$^{\text{off}}(1^\lambda)$

DO generates secret key shares and correlated randomness shares as follows:

(1) Generate a secret key $K \xleftarrow{\$} \mathbb{Z}_2^{m\times n}$.
(2) Randomly sample masks $\tilde{K} \in \mathbb{Z}_2^{m\times n}, \tilde{x} \in \mathbb{Z}_2^n$ for K, x over $\mathbb{Z}_2$ and $\bar{K} \in \mathbb{Z}_3^{m\times n}, \bar{x} \in \mathbb{Z}_3^n$ for K, x over $\mathbb{Z}_3$.
(3) Compute $\tilde{r}_1 = \tilde{x} \mod 3, \tilde{r}_2 = (\tilde{x}+\mathbf{1}) \mod 2 \mod 3$, and $\tilde{R}_1 = \tilde{K} \mod 3, \tilde{R}_2 = (\tilde{K}+\mathbf{1}) \mod 2 \mod 3$.
(4) Randomly sample mask $\bar{v} \in \mathbb{Z}_3^m$.
(5) Compute $\bar{r}_1 = \bar{v} \mod 2, \bar{r}_2 = (\bar{v}+\mathbf{1}) \mod 3 \mod 2$.
(6) Create additive shares for $K, \tilde{K}, \tilde{x}, \tilde{K}\tilde{x}, \bar{K}, \bar{x}, \bar{K}\bar{x}+\bar{v}, \tilde{r}_1, \tilde{r}_2, \tilde{R}_1, \tilde{R}_2, \bar{r}_1, \bar{r}_2$ to obtain $\langle\mathcal{P}\rangle$.
(7) Output $\langle\mathcal{P}\rangle_\ell$ to party P_ℓ for $\ell \in \{0,1\}$.

procedure: DLPN-wPRF$^{\text{on}}(\langle\mathcal{P}\rangle, \langle x\rangle)$

(1) Parties P_0, P_1 locally mask their key shares $\langle K\rangle$ and input shares $\langle x\rangle$ in Eq. 6 over $\mathbb{Z}_2$.

$$\left\langle\hat{K}\right\rangle = \langle K\rangle + \left\langle\tilde{K}\right\rangle, \langle\hat{x}\rangle = \langle x\rangle + \langle\tilde{x}\rangle. \tag{6}$$

(2) Reconstruct $\hat{K}, \hat{x}$ by exchanging their shares.
(3) P_ℓ locally runs $\pi^2_{\text{BL}}(\hat{K}, \hat{x}, \left\langle\tilde{K}\right\rangle_\ell, \langle\tilde{x}\rangle_\ell, \left\langle\tilde{K}\tilde{x}\right\rangle_\ell)$ to obtain $\langle u\rangle_\ell$ for $\ell \in \{0,1\}$.
(4) P_ℓ locally runs $\pi^{(2,3)}_{\text{Convert}}(\hat{x}, \langle\tilde{r}_1\rangle_\ell, \langle\tilde{r}_2\rangle_\ell), \pi^{(2,3)}_{\text{Convert}}(\hat{K}, \left\langle\tilde{R}_1\right\rangle_\ell, \left\langle\tilde{R}_2\right\rangle_\ell)$ to obtain $\langle x^*\rangle, \langle K^*\rangle$ over $\mathbb{Z}_3$.
(5) Parties P_0, P_1 compute $\langle\hat{v}^*\rangle = \langle K^*x^* + \bar{v}\rangle$ using the triple $(\left\langle\bar{K}\right\rangle, \langle\bar{x}\rangle, \left\langle\bar{K}\bar{x}+\bar{v}\right\rangle)$ via π^3_{BL} protocol and open it to obtain $\hat{v}^*$.
(6) P_ℓ locally runs $\pi^{(3,2)}_{\text{Convert}}(\hat{v}^*, \langle\bar{r}_1\rangle_\ell, \langle\bar{r}_2\rangle_\ell)$ to obtain $\langle v\rangle_\ell$.
(7) P_ℓ locally runs $\pi^2_{\text{Add}}(\langle u\rangle_\ell, \langle v\rangle_\ell)$ to obtain $\langle w\rangle_\ell$.
(8) P_ℓ locally runs $\pi^{B,2}_{\text{Lin}}(\langle w\rangle_\ell)$ and outputs the result $\langle y\rangle_\ell$.

Fig. 6. Detailed procedure of $\pi_{\text{DLPN-wPRF}}$.

$N_{\text{B}} \times 2$-dimensional matrices $\text{Eb}^{(0)}, \text{Eb}^{(1)}$ as shown in Equation (7). Note that the noise r_i in Equation (7) is generated independently of QU and is equal to $H(w||i)$, where w is a keyword extracted from the document D and stored in the address field. More details are provided in Section 5.2.

$$\text{Eb}^{(\ell)}[i][0] = \left\langle F^B_K(r_i)\right\rangle_\ell, \text{Eb}^{(\ell)}[i][1] = y_i \oplus \left\langle F^B_K(r_i)\right\rangle_{1-\ell}, i \in [N_{\text{B}}]. \tag{7}$$

It is worth noting that all locations containing 1 in the query vector $\boldsymbol{q}$ are leaked if we directly use SHVE construction [16] for secure vector matching, thereby leaking query privacy and data privacy. To solve it, we use ASS to split $\boldsymbol{q}$ into two random vectors $\langle\boldsymbol{q}\rangle$, such that the locations containing 1 are hidden. In this way, privacy-preserving conjunctive query is achieved by computing $\langle\text{tk}\rangle_\ell \oplus \text{Ind}^{(\ell)}$, where

$$\text{Ind}^{(\ell)} = \text{Eb}^{(\ell)} \circ \langle\boldsymbol{q}\rangle_\ell = \oplus_{i\in[N_{\text{B}}]}\text{Eb}^{(\ell)}\left[i\right]\left[\langle\boldsymbol{q}\rangle_\ell[i]\right]. \tag{8}$$

(2) Converting access control into secure vector matching. Let $\mathcal{X} = \{X_1, \ldots, X_d\}$ be a set of attributes and $V_i = \{v_1, v_2, \ldots\}$ be the value set for the attribute X_i (e.g., role, department, location, etc.). For example, for the attribute *role*, the attribute values are a set of unique numbers corresponding to *nurse, pharmacist, physician, and so forth*. We define $\Sigma = V_1 \cup \cdots \cup V_d$ and $\Sigma_* = \Sigma \cup \{*\}$. The access policy Γ_D is encoded as a policy vector $\boldsymbol{p} \in \Sigma_*^d$. The user attribute set S is encoded as an attribute vector $\boldsymbol{a} \in \Sigma^d$. If $\boldsymbol{a}[j] = \boldsymbol{p}[j]$ for all $\boldsymbol{p}[j] \neq *$, the attribute set S matches the access policy Γ_D, i.e., $\text{AcMatch}(\Gamma_D, S) = 1$. For fine-grained access control and result decryption without the assistance of the DO, we conceal the secret key sk_D under the policy vector $\boldsymbol{p}$ via SHVE.KeyGen to

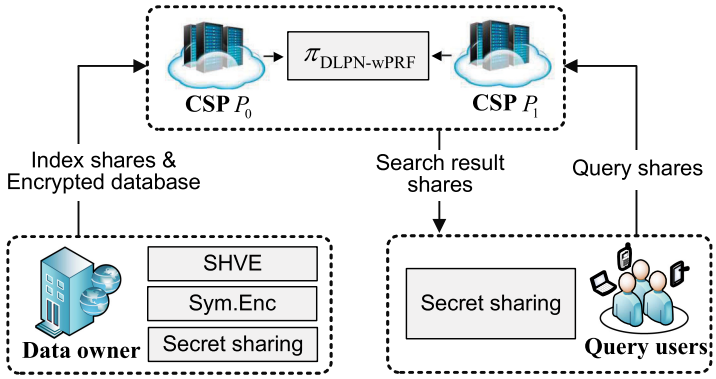


Fig. 7. Implementation workflow of ODiSC.

obtain a key ciphertext Ek. Moreover, we encrypt the attribute vector $\boldsymbol{a}$ as an attribute ciphertext Eu via SHVE.Enc. To hide the attributes that the document D does not care about, the policy vector $\boldsymbol{p}$ is transformed into a binary vector bv, which is then split into two random binary vectors $\langle \mathsf{bv} \rangle$. To protect sk_D from CSPs, the key ciphertext Ek is also additively split. In this way, fine-grained access control is achieved by computing $\langle \mathsf{Ek} \rangle_\ell \oplus \mathsf{Att}^{(\ell)}$, where

$$\mathsf{Att}^{(\ell)} = \mathsf{Eu} \odot \langle \mathsf{bv} \rangle_\ell = \oplus_{i \in [d]} \mathsf{Eu}[i] \cdot \langle \mathsf{bv} \rangle_\ell [i]. \tag{9}$$

Finally, each CSP P_ℓ computes the search result share $\langle sk' \rangle_\ell$ in Equation (10) and returns it to the QU, who then reconstructs sk'. If the search query Q matches the document D and the attribute set S satisfies the access policy Γ_D, then sk' is equal to sk_D and the QU can use it to decrypt the ciphertext c_D without the help of the DO. Since the computation of $\langle sk' \rangle_\ell$ is performed locally, the number of communication rounds among CSPs is not affected by the size of the outsourced database. CSPs just require three rounds of communication to generate the search token.

$$\langle sk' \rangle_\ell = (\langle \mathsf{tk} \rangle_\ell \oplus \mathsf{Ind}^{(\ell)}) \oplus (\langle \mathsf{Ek} \rangle_\ell \oplus \mathsf{Att}^{(\ell)}). \tag{10}$$

5.2 Concrete Construction

ODiSC is an MSSE scheme that supports fine-grained conjunctive query with DOs free from sharing key and being online. Its implementation workflow is shown in Figure 7. DO uses SHVE, Sym.Enc, and the secret sharing technique to build a secure index and encrypt the document database. DU employs the secret sharing technique to issue search request in a privacy-preserving manner. CSPs P_0, P_1 perform the MPC protocol $\pi_{\text{DLPN-wPRF}}$ to retrieve accessible documents including all query keywords. The concrete construction can be divided into four algorithms: Setup, IndexBuild, QueryIssue, and Search. The specific process is shown as follows.

Setup(1^λ): Given the security parameter λ, the DO runs DLPN-wPRF$^{\text{off}}(1^\lambda)$ to generate the parameter shares $\langle \mathcal{P} \rangle$ and the secret key K, then sends $\langle \mathcal{P} \rangle_\ell$ to CSP P_ℓ for $\ell \in \{0, 1\}$. In addition, when a QU with identity id wants to join, the DO first transforms his or her attribute set S into an attribute vector $\boldsymbol{a}$, then encrypts it as the attribute ciphertext Eu in Equation (11), and finally sends the tuple $(H_3(id), \mathsf{Eu})$ to each CSP, where $H : \{0,1\}^* \to \{0,1\}^n, H_3 : \{0,1\}^* \to \{0,1\}^\lambda$ are two hash functions.

$$\mathsf{Eu}[i] = F_K^B(H(\boldsymbol{a}[i] || (N_\mathsf{B} + i))), i \in [d]. \tag{11}$$

IndexBuild(K, DB, A): Given an outsourced database $\mathsf{DB} = \{D_1, D_2, \ldots, D_N\}$ and their policy vector set A, the DO extracts keywords from DB to obtain the keyword dictionary $W = \{w_1, w_2, \ldots, w_M\}$. After that, the DO builds and secretly shares an inverted index for efficient

ALGORITHM 1: Index Building Process

Input: Secret key K, database DB, policy vector set A
Output: Index shares $\langle \mathcal{I} \rangle$, encrypted database EDB

1 **for** $j = 1; j \leq N; j++$ **do**
2 **for** $i = 1; i \leq N_B; i++$ **do**
3 $\boldsymbol{b} \xleftarrow{encode} D_j$;
4 Compute $y_i = F_K^B(H(\boldsymbol{b}[i]||i))$;
5 Set $\mathsf{Eb} = (y_1, y_2, \ldots, y_{N_\mathsf{B}})$;
6 Select $\mathsf{str} \in \{0,1\}^\lambda$ satisfying $\mathsf{str} = \mathbf{0}^l||\mathsf{str}_1||\mathsf{str}_2$;
7 Compute $sk_{D_j} = H_1(\mathsf{str}_1), p_{D_j} = H_2(\mathsf{str}_2)$;
8 Compute $c_j = \mathsf{Sym.Enc}(sk_{D_j}, D_j)$ and link it to p_{D_j};
9 Set $Y_{D_j} = \{i \in [d] | \boldsymbol{p}_j[i] \neq *\}$;
10 Compute $\mathsf{Ek} = \oplus_{i \in Y_{D_j}} F_K^B(H(\boldsymbol{p}_j[i]||(N_\mathsf{B} + i))) \oplus \mathsf{str}$;
11 Generate bv satisfying $\mathsf{bv}[i] = 1$ if $i \in Y_{D_j}$ and 0 otherwise;
12 Set $C_{D_j} = (\mathsf{Eb}, \mathsf{Ek}, \mathsf{bv})$, $\mathsf{EDB} = \{c_1, c_2, \ldots, c_N\}$;
13 Construct an inverted index $\mathcal{I}$ containing M address-value pairs $\{\langle \mathsf{Ew}_i, L_i \rangle\}_{i \in [M]}$, where $\mathsf{Ew}_i = F_K^B(H(w_i))$ and $L_i = \{C_{D_j}\}_{D_j \in \mathsf{DB}(w_i)}$;
14 **for** $\varrho = 1; \varrho \leq M; \varrho++$ **do**
15 **for** $C_{D_j} \in L_\varrho$ **do**
16 $\langle \mathsf{Ek} \rangle \xleftarrow{ASS} \mathsf{Ek}$, $\langle \mathsf{bv} \rangle \xleftarrow{ASS} \mathsf{bv}$;
17 **for** $i = 1; i \leq N_B; i++$ **do**
18 Compute $r_i = H(w_\varrho||i)$;
19 Compute $\mathsf{Eb}^{(\ell)}[i][0], \mathsf{Eb}^{(\ell)}[i][1]$ via Equation (7);
20 Set $\langle C_{D_j} \rangle_\ell = (\mathsf{Eb}^{(\ell)}, \langle \mathsf{Ek} \rangle_\ell, \langle \mathsf{bv} \rangle_\ell)$;
21 Set $\langle L_\varrho \rangle_\ell = \{\langle C_{D_j} \rangle_\ell\}_{D_j \in \mathsf{DB}(w_\varrho)}$;
22 Set $\langle \mathcal{I} \rangle_\ell = \{\langle \mathsf{Ew}_i, \langle L_i \rangle_\ell \rangle\}_{i \in [M]}$;
23 **return** $\langle \mathcal{I} \rangle$, EDB.

ciphertext retrieval, which can be divided into the following three steps. The specific process is shown in Algorithm 1.

- *Encryption.* For each document $D_j \in \mathsf{DB}$ with the policy vector $\boldsymbol{p} \in A$, the DO generates the content $C_{D_j} = (\mathsf{Eb}, \mathsf{Ek}, \mathsf{bv})$ below:
  - The DO represents D_j as a data vector $\boldsymbol{b}$ of size N_B as described in Section 5.1 and then runs LPN-wPRF F_K^B to encrypt $\boldsymbol{b}$ as an index ciphertext Eb (Lines 2–5).
  - The DO selects a string $\mathsf{str} \in \{0,1\}^\lambda$ such that its first l-bit is $\mathbf{0}$ and then parses its last $\lambda - l$ bits into two strings $\mathsf{str}_1, \mathsf{str}_2$ (Line 6). Based on the string str_1, the DO computes the secret key sk_{D_j} that will be used to encrypt D_j. Based on the string str_2, the DO computes the pointer p_{D_j} and links it to the ciphertext c_j (Lines 7–8). The pointer p_{D_j} serves as an indirect address to locate the entry storing c_j. Here, $H_1 : \{0,1\}^* \rightarrow \{0,1\}^t$, $H_2 : \{0,1\}^* \rightarrow \{0,1\}^{\lceil \log N \rceil}$ are two hash functions.
  - For the policy vector $\boldsymbol{p}$, the DO obtains the set Y_{D_j} and encrypts the string str as a key ciphertext Ek (Lines 9–10). In addition, the DO transforms Y_{D_j} as a d-dimensional binary vector bv (Line 11). Note that the matrix in $\mathbb{Z}_2^t$ is converted to a t-bit string by default in the XOR operation.

ALGORITHM 2: Search Process

Input: Parameter shares $\langle \mathcal{P} \rangle$, index shares $\langle \mathcal{I} \rangle$, query shares $\langle Q \rangle$, identity $H_3(id)$
Output: Desired result set $\mathcal{R}$

1 CSPs P_0, P_1:
2 Find Eu via $H_3(id)$;
3 Compute $\langle \mathsf{tk}_{\mathsf{add}} \rangle, \langle \mathsf{tk}_{\mathsf{val}} \rangle$ via Equation (12);
4 Reconstruct $\mathsf{tk}_{\mathsf{add}} = F_K^B(H(w))$ and locate the index list $\langle L_w \rangle_\ell$;
5 **for** $\langle C_{D_j} \rangle_\ell \in \langle L_w \rangle_\ell$ **do**
6 Compute $\mathsf{Ind}^{(\ell)}, \mathsf{Att}^{(\ell)}$ via Equations (8) and (9), respectively;
7 Compute $\langle \mathsf{str}' \rangle_\ell = (\langle \mathsf{tk}_{\mathsf{val}} \rangle_\ell \oplus \mathsf{Ind}^{(\ell)}) \oplus (\langle \mathsf{Ek} \rangle_\ell \oplus \mathsf{Att}^{(\ell)})$;
8 $\langle R \rangle_\ell = \langle R \rangle_\ell \cup \{\langle \mathsf{str}' \rangle_\ell\}$; // $\langle R \rangle_\ell$ is initialized as $\varnothing$
9 QU:
10 **for** $\langle str' \rangle_\ell \in \langle R \rangle_\ell$ **do**
11 Reconstruct $\mathsf{str}' = \langle \mathsf{str}' \rangle_0 \oplus \langle \mathsf{str}' \rangle_1$;
12 **if** *the first l-bit of str′ is* **0** **then**
13 Extract the last $\lambda - l$ bits of the string str′ and derive sk_{D_j} and p_{D_j};
14 Retrieve c_j using p_{D_j} and obtain $D_j = \mathsf{Sym.Dec}(sk_{D_j}, c_j)$;
15 $\mathcal{R} = \mathcal{R} \cup \{D_j\}$; // $\mathcal{R}$ is initialized as $\varnothing$
16 **return** $\mathcal{R}$.

- *Index construction.* The DO builds an inverted index $\mathcal{I}$ containing M pairs $\langle$address, value$\rangle$. The address field stores the ciphertext Ew_i of each keyword $w_i \in W$, and the value field stores an index list L_i of the documents $\{D_j | D_j \in \mathsf{DB}(w_i)\}$ (Line 13).
- *Splitting and uploading.* For each content C_{D_j}, the DO first splits Eb, Ek, bv into $\mathsf{Eb}^{(\ell)}, \langle \mathsf{Ek} \rangle_\ell$, $\langle \mathsf{bv} \rangle_\ell$, respectively, to obtain $\langle C_{D_j} \rangle_\ell$ (Lines 15–20). Then, the DO replaces C_{D_j} with $\langle C_{D_j} \rangle_\ell$ to obtain the index share $\langle \mathcal{I} \rangle_\ell$ (Lines 21–22). Finally, the DO uploads the index share $\langle \mathcal{I} \rangle_\ell$ and encrypted database EDB to CSP P_ℓ for $\ell \in \{0, 1\}$.

QueryIssue(Q): When the QU with identity id attempts to issue a search query Q, he or she generates the query shares $\langle Q \rangle$ as follows. First, the QU randomly selects a keyword w from the search query Q and additively splits its hashed value $H(w)$ into $\langle H(w) \rangle$ (Lines 1–2). Next, the QU represents Q as a query vector $\boldsymbol{q}$ as Section 5.1 does, which is then additively split into $\langle \boldsymbol{q} \rangle$ (Line 3). In addition, the QU generates an N_{B}-dimensional vector $\boldsymbol{q}^*$ satisfying $\boldsymbol{q}^*[i] = H(\boldsymbol{q}[i] || i)$ if $\boldsymbol{q}[i] = 1$ and $\boldsymbol{q}^*[i] = H(w || i)$ otherwise, and then splits it into $\langle \boldsymbol{q}^* \rangle$ (Lines 4–7). Finally, the QU encodes its identity id as $H_3(id)$ and sends it along with the query share $\langle Q \rangle_\ell = (\langle H(w) \rangle_\ell, \langle \boldsymbol{q}^* \rangle_\ell, \langle \boldsymbol{q} \rangle_\ell)$ to CSP P_ℓ for $\ell \in \{0, 1\}$.

Search($\langle \mathcal{P} \rangle, \langle \mathcal{I} \rangle, \langle Q \rangle, H_3(id)$): Upon receiving the search request, CSPs P_0, P_1 first find the attribute ciphertext Eu via $H_3(id)$. Then, they generate a secret-shared search token and use it to perform privacy-preserving retrieval to find the document that is accessible to the QU and contains all query keywords. The specific search process is shown in Algorithm 2, which can be divided into the following four steps in Figure 8:

- *Token generation.* CSPs P_0, P_1 cooperatively run DLPN-wPRF$^{\mathsf{on}}$ to generate the search token shares $\langle \mathsf{tk} \rangle = (\langle \mathsf{tk}_{\mathsf{add}} \rangle, \langle \mathsf{tk}_{\mathsf{val}} \rangle)$ as shown in Equation (12) (Line 3).

$$\begin{aligned} \langle \mathsf{tk}_{\mathsf{add}} \rangle &\leftarrow \mathsf{DLPN\text{-}wPRF}^{\mathsf{on}}(\langle \mathcal{P} \rangle, \langle H(w) \rangle), \\ \langle \mathsf{tk}_{\mathsf{val}} \rangle &\leftarrow \oplus_{i \in [N_{\mathsf{B}}]} \mathsf{DLPN\text{-}wPRF}^{\mathsf{on}}(\langle \mathcal{P} \rangle, \langle \boldsymbol{q}^* \rangle). \end{aligned} \tag{12}$$

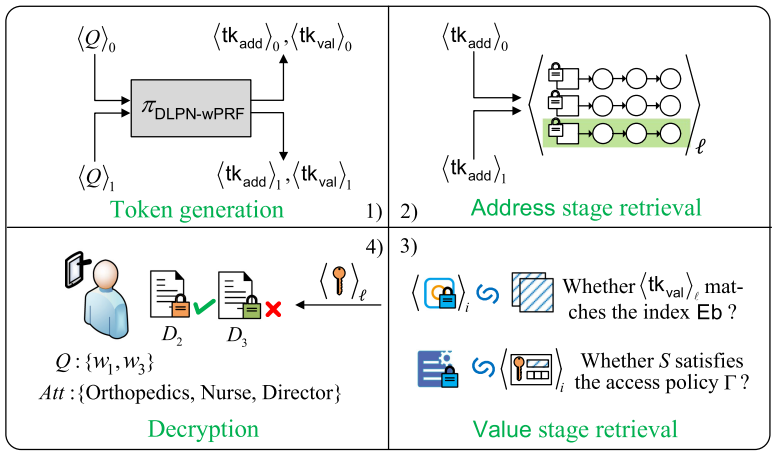


Fig. 8. Example of search process.

- address *field retrieval*. CSP P_ℓ reconstructs $\mathsf{tk}_{\mathsf{add}}$ and uses it to locate the index list $\langle L_w \rangle_\ell$ for $\ell \in \{0, 1\}$ (Line 4).
- value *field retrieval*. For each content share $\langle C_{D_j} \rangle_\ell$ in the index list $\langle L_w \rangle_\ell$, CSP P_ℓ first locally computes $\mathsf{Ind}^{(\ell)}$ and $\mathsf{Att}^{(\ell)}$, then uses them to compute the share $\langle \mathsf{str}' \rangle_\ell$ according to Equation (10) and returns it to the QU (Lines 5–8). One potential drawback of the retrieval process is that CSP P_ℓ reveals the number of documents containing the specific query keyword w, known as volume pattern leakage. Fortunately, existing schemes such as [10, 24, 33] offer solutions for implementing volume-hiding retrieval, which can be used in our scheme to hide the leakage.
- *Decryption*. The QU reconstructs $\mathsf{str}' = \langle \mathsf{str}' \rangle_0 \oplus \langle \mathsf{str}' \rangle_1$ (Line 11). If the first l-bit of str' is $\mathbf{0}$, the QU first extracts the strings $\mathsf{str}_1, \mathsf{str}_2$ to derive the secret key sk_{D_j} and pointer p_{D_j}, respectively, then uses sk_{D_j} to decrypt the ciphertext c_j to obtain the desired document D_j (Lines 12–15). Note that the ciphertext c_j can be retrieved using the pointer p_{D_j} via keyword-based **private information retrieval (PIR)**, which has been extensively studied in recent literature such as [1, 22]. By leveraging keyword-based PIR, our scheme ensures that CSPs are unable to determine which specific document satisfies the search request by monitoring memory access. It guarantees adaptive simulation security.

Correctness. We prove that $\mathsf{str}' = \mathsf{str}$ if the document D_j matches the QU's search query Q (i.e., $\boldsymbol{b}[i] = \boldsymbol{q}[i]$ for $\forall i \in Y_{\boldsymbol{q}} = \{i | \boldsymbol{q}[i] = 1\}$) and the QU's attribute set S satisfies the access policy Γ_{D_j} (i.e., $\boldsymbol{a}[i] = \boldsymbol{p}[i]$ for $\forall i \in Y_{D_j}$), simultaneously:

$$
\begin{aligned}
\mathsf{str}' =& \langle \mathsf{str}' \rangle_0 \oplus \langle \mathsf{str}' \rangle_1 \\
=& \mathsf{tk}_{\mathsf{val}} \oplus (\mathsf{Ind}^{(0)} \oplus \mathsf{Ind}^{(1)}) \oplus (\mathsf{Att}^{(0)} \oplus \mathsf{Att}^{(1)}) \oplus \mathsf{Ek} \\
=& (\oplus_{i \in Y_{\boldsymbol{q}}} \mathsf{tk}_{\mathsf{val}}[i]) \oplus (\oplus_{i \in Y_{\boldsymbol{q}}} \mathsf{Eb}[i]) \oplus (\oplus_{i \in Y_{D_j}} \mathsf{Eu}[i]) \oplus \mathsf{Ek} \\
=& \oplus_{i \in Y_{\boldsymbol{q}}} \left(F_K^B\Big(H(\boldsymbol{b}[i] || i)\Big) \oplus F_K^B\Big(H(\boldsymbol{q}[i] || i)\Big) \right) \\
& \oplus \left(\oplus_{i \in Y_{D_j}} F_K^B\Big(H(\boldsymbol{a}[i] || (N_{\mathsf{B}} + i))\Big) \oplus F_K^B\Big(H(\boldsymbol{p}[i] || (N_{\mathsf{B}} + i))\Big) \oplus \mathsf{str} \right).
\end{aligned}
$$

We present a toy example of single document retrieval. The document $D = \{w_1, w_2, w_3\}$ with the access policy $\Gamma_D = \{Orthopedics, Nurse, *\}$ is encoded as $\boldsymbol{b} = (1, 1, 1)$ and Γ_D is encoded as $\boldsymbol{p} = (6, 1, *)$. The search query $Q = \{w_1, w_3\}$ for the QU with attribute set $S = \{Orthopedics, Nurse,$

Director} is encoded as $\boldsymbol{q} = (1, 0, 1)$ and S is encoded as $\boldsymbol{a} = (6, 1, 5)$. For privacy-preserving owner-free retrieval, the DO, QU, and CSPs do as follows.

DO: The DO first encrypts $\boldsymbol{b} = (1, 1, 1)$ as the index ciphertext $\mathsf{Eb} = (y_1, y_2, y_3)$, where $y_i = F_K^B(H(\boldsymbol{b}[i]||i)) = F_K^B(H(1||i))$. Then, the DO computes $k_i = F_K^B(H(w_1||i))$ for $i \in [3]$ and splits Eb as two matrices $\mathsf{Eb}^{(0)}, \mathsf{Eb}^{(1)}$, as shown in Equation (13). Next, the DO encrypts the secret key sk_D under the policy vector $\boldsymbol{p} = (6, 1, *)$ to obtain the key ciphertext $\mathsf{Ek} = y_4 \oplus y_5 \oplus sk_D$, where $y_4 = F_K^B(H(6||4))$, $y_5 = F_K^B(H(1||5))$. In addition, the DO transforms $\boldsymbol{p} = (6, 1, *)$ into the binary vector $\mathsf{bv} = (1, 1, 0)$ and splits it into two vectors $\langle \mathsf{bv} \rangle_0 = (1, 0, 0), \langle \mathsf{bv} \rangle_1 = (0, 1, 0)$. Finally, the content share $\langle C_D \rangle_\ell = (\mathsf{Eb}^{(\ell)}, \langle \mathsf{Ek} \rangle_\ell, \langle \mathsf{bv} \rangle_\ell)$ is sent to CSP P_ℓ.

$$\mathsf{Eb}^{(0)} = \begin{bmatrix} \langle k_1 \rangle_0 & y_1 \oplus \langle k_1 \rangle_1 \\ \langle k_2 \rangle_0 & y_2 \oplus \langle k_2 \rangle_1 \\ \langle k_3 \rangle_0 & y_3 \oplus \langle k_3 \rangle_1 \end{bmatrix}, \mathsf{Eb}^{(1)} = \begin{bmatrix} \langle k_1 \rangle_1 & y_1 \oplus \langle k_1 \rangle_0 \\ \langle k_2 \rangle_1 & y_2 \oplus \langle k_2 \rangle_0 \\ \langle k_3 \rangle_1 & y_3 \oplus \langle k_3 \rangle_0 \end{bmatrix}. \tag{13}$$

QU: For the query vector $\boldsymbol{q} = (1, 0, 1)$, the QU first splits it into $\langle \boldsymbol{q} \rangle_0 = (0, 1, 1), \langle \boldsymbol{q} \rangle_1 = (1, 1, 0)$. Then, the QU generates $\boldsymbol{q}^* = (H(1||1), F_K^B(w_1||2), H(1||3))$ and additively splits it into $\langle \boldsymbol{q}^* \rangle_0, \langle \boldsymbol{q}^* \rangle_1$. After that, the QU sends the query share $\langle Q \rangle_\ell = \{\langle H(w_1) \rangle_\ell, \langle \boldsymbol{q}^* \rangle_\ell, \langle \boldsymbol{q} \rangle_\ell\}$ to CSP P_ℓ. In addition, the QU's attribute vector $\boldsymbol{a} = (6, 1, 5)$ is encrypted as the attribute ciphertext $\mathsf{Eu} = (y_4, y_5, y_6)$, where $y_6 = F_K^B(H(5||6))$.

CSPs: After receiving the search request, two CSPs cooperatively generate a secret-shared search token $\langle \mathsf{tk} \rangle$ satisfying $\mathsf{tk} = (y_1, k_2, y_3)$ via $\pi_{\mathsf{DLPN\text{-}wPRF}}$, and CSP P_ℓ obtains the search token share $\langle \mathsf{tk} \rangle_\ell = \langle y_1 \rangle_\ell \oplus \langle k_2 \rangle_\ell \oplus \langle y_3 \rangle_\ell$. Then, P_ℓ locally computes $\langle sk' \rangle_\ell$ and sends it to the QU, who then reconstructs sk' in Equation (14), which is equal to sk_D.

$$\begin{aligned} sk' =& \Big(\mathsf{tk} \oplus (\mathsf{Eb} \circ \boldsymbol{q})\Big) \oplus \Big(\mathsf{Ek} \oplus (\mathsf{Eu} \odot \mathsf{bv})\Big) \\ =& \Big((y_1 \oplus y_3 \oplus k_2) \oplus (y_1 \oplus y_3 \oplus k_2)\Big) \oplus \Big((y_4 \oplus y_5 \oplus sk_D) \oplus (y_4 \oplus y_5)\Big) \\ =& sk_D. \end{aligned} \tag{14}$$

Remark. ODiSC can also be deployed in multi-cloud architecture ($s > 2$ is the number of CSPs), which enhances security at the expense of performance. To achieve this, it is necessary to generate a secret-shared search token $\langle \mathsf{tk} \rangle = \{\langle \mathsf{tk} \rangle_1, \ldots, \langle \mathsf{tk} \rangle_s\}$ and to split the index ciphertext Eb into s shares. The scheme [13] has demonstrated that the distributed LPN-wPRF protocol $\pi_{\mathsf{DLPN\text{-}wPRF}}$ can work securely for any number of parties, indicating that the search token shares $\langle \mathsf{tk} \rangle$ can be generated when the query vector $\boldsymbol{q}$ and secret key K are secretly shared among s clouds. Moreover, the index ciphertext Eb of each document D_j can be split into s $N_\mathsf{B} \times 2$-dimensional matrices $\mathsf{Eb}^{(\ell)} (\ell \in [s])$ such that

$$\mathsf{Eb}^{(\ell)}[i][0] = k_{i,\ell}, \mathsf{Eb}^{(\ell)}[i][1] = y_i \oplus (\oplus_{j \in [s] \setminus \{\ell\}} k_{i,j}), i \in [N_\mathsf{B}], \tag{15}$$

where $k_{i,1}, k_{i,2}, \ldots, k_{i,s}$ are s random numbers whose sum is equal to $F_K^B(H(w||i))$, and w is a keyword extracted from the document D and stored in the address field. Then, privacy-preserving conjunctive query among s CSPs can be achieved by computing $\oplus_{\ell \in [s]}(\langle \mathsf{tk} \rangle_\ell \oplus \mathsf{Ind}^{(\ell)})$. In s cloud setting, the security of ODiSC can be ensured as long as at least one CSP does not conspire. However, the communication overhead for the DO and QUs, as well as the computation overhead for s clouds, increases by a factor of $s/2$ compared to the dual-cloud architecture.

5.3 Discussion

In the algorithm Search, CSP $P_\ell (\forall \ell \in \{0, 1\})$ retrieves the attribute ciphertext Eu using the identity $H_3(id)$ sent by the QU and then utilizes Eu to locate accessible search results. The operation may result in data privacy leakage. That is, CSP P_ℓ may impersonate the QU's identity id and send a

search request to the other CSP $P_{1-\ell}$. Then, $P_{1-\ell}$ performs the algorithm Search to return the search result share set $R_{1-\ell}$ to CSP P_ℓ. With the complete result shares, CSP P_ℓ can reconstruct the secret keys of the documents accessible to QU id. To prevent this, we perform identity authentication using the RSA algorithm and ASS before retrieval.

We secretly share the private key K_{pri} of the RSA algorithm among θ users/agents. In each query, the QU with identity id generates a new identity signature $S_{id,\delta}$ by interacting with $\theta-1$ users/agents. Upon receiving the search request, each CSP $P_\ell(\ell \in \{0,1\})$ verifies that the identity signature is newly generated (i.e., not being used before). Under the assumption that the number of corrupted users/agents is less than θ, CSPs cannot reconstruct the private key K_{pri}, and thus cannot generate a new signature. Therefore, if the identity signature is newly generated, then a corresponding search request is made by the QU. Otherwise, the search request is made by a certain CSP P_ℓ with previous identity signatures, and the other CSP $P_{1-\ell}$ terminates the retrieval. Without the search result share set $\langle R\rangle_{1-\ell}$ returned by CSP $P_{1-\ell}$, CSP P_ℓ is unable to learn any useful information from his or her set $\langle R\rangle_\ell$. The detailed process of identity authentication is shown as follows:

- In the registration phase, the DO generates a pair of RSA public and private keys (K_{pub}, K_{pri}) for the registered QU and secretly shares K_{pri} into θ shares $\langle K_{pri}\rangle$ satisfying $\sum_{i=1}^{\theta}\langle K_{pri}\rangle_\ell = K_{pri}$. Then, the DO sends K_{pub} to each CSP, one key share $\langle K_{pri}\rangle_0$ to the registered QU, and the last shares to $\theta-1$ users/agents, respectively.
- When the QU with identity id attempts to issue a search request, he or she first sends $(\delta \cdot H_3(id))^r$ to $\theta-1$ users/agents, where δ, r are two random elements. Then, $\theta-1$ users/agents locally compute the signature shares $\langle S_{id,\delta}\rangle_{l_j}$ $(j \in [\theta-1])$ in Equation (16) and send them to the QU. It is worth noting that δ needs to be different for each query to prevent CSP from using previous identity signatures to pass identity authentication.

$$\langle S_{id,\delta}\rangle_{l_j} = \mathsf{Sign}(\langle K_{pri}\rangle_{l_j}, (\delta H_3(id))^r). \tag{16}$$

- After receiving $\theta-1$ signature shares, the QU reconstructs the identity signature $S_{id,\delta}$ in Equation (17) and sends the tuple $(H_3(id), S_{id,\delta})$ along with the query share $\langle Q\rangle_\ell$ to CSP P_ℓ for $\ell \in \{0,1\}$.

$$S_{id,\delta} = \mathsf{Sign}(\langle K_{pri}\rangle_{l_0}, \delta H_3(id)) \cdot \left(\prod_{j=1}^{\theta-1}\langle S_{id,\delta}\rangle_{l_j}\right)^{1/r} = \mathsf{Sign}(K_{pri}, \delta H_3(id)). \tag{17}$$

- When receiving the search request, each CSP $P_\ell(\ell \in \{0,1\})$ verifies the user identity by extracting the element δ in Equation (18).

$$\delta = \mathsf{Ver}(K_{pub}, S_{id,\delta})/H_3(id). \tag{18}$$

If δ is an unused element, each CSP P_ℓ performs subsequent retrieval as the algorithm Search does. Otherwise, the signature $S_{id,\delta}$ has been used before, which indicates that one CSP P_ℓ impersonates the QU's identity id. In this case, the other CSP $P_{1-\ell}$ terminates the search, and thus CSP P_ℓ cannot reconstruct the desired results.

6 SECURITY ANALYSIS

Before analyzing the security of the protocol $\pi_{\mathsf{DLPN\text{-}wPRF}}$, we introduce the following lemmas.

Lemma 1 ([3]). *A protocol is perfectly simulatable if all its sub-protocols are perfectly simulatable.*

Lemma 2 ([4]). *If a random element r is uniformly distributed on $\mathbb{Z}_p$ and independent from any variable $x \in \mathbb{Z}_p$, $r \pm x$ is also uniformly random and independent from x.*

According to Lemma 1, if the sub-protocols $\pi^2_{\mathsf{Add}}, \pi^{B,2}_{\mathsf{Lin}}, \pi^{p}_{\mathsf{BL}}, \pi^{(2,3)}_{\mathsf{Convert}}, \pi^{(3,2)}_{\mathsf{Convert}}$ are proved secure or simulatable, then $\pi_{\mathsf{DLPN\text{-}wPRF}}$ is secure. Thus, we prove the security of $\pi_{\mathsf{DLPN\text{-}wPRF}}$ in Theorem 1.

Theorem 1. *$\pi_{DLPN\text{-}wPRF}$ is secure in the honest-but-curious model if Lemmas 1 and 2 hold.*

Proof. The sub-protocols π^2_{Add} and $\pi^{B,2}_{\mathsf{Lin}}$ are performed without interaction, and thereby their inputs and parameters can be replaced by randomness. Thus, $\pi^2_{\mathsf{Add}}, \pi^{B,2}_{\mathsf{Lin}}$ are secure in the honest-but-curious model and we prove the security of $\pi^{p}_{\mathsf{BL}}, \pi^{(2,3)}_{\mathsf{Convert}}$, and $\pi^{(3,2)}_{\mathsf{Convert}}$ below.

For the sub-protocol π^{p}_{BL}, $P_\ell(\ell \in \{0,1\})$ holds the view $view_{\mathsf{BL},i} = \{\langle K\rangle_\ell, \langle x\rangle_\ell, \langle \hat{K}\rangle_\ell, \langle \hat{x}\rangle_\ell, \langle \tilde{K}\rangle_\ell, \langle \tilde{x}\rangle_\ell, \langle \tilde{K}\tilde{x}\rangle_\ell, \hat{K}, \hat{x}\}$, where $\langle \hat{K}\rangle_\ell = \langle K\rangle_\ell + \langle \tilde{K}\rangle_\ell, \langle \hat{x}\rangle_\ell = \langle x\rangle_\ell + \langle \tilde{x}\rangle_\ell, \hat{K} = \langle \hat{K}\rangle_\ell + \langle \hat{K}\rangle_{1-\ell}, \hat{x} = \langle \hat{x}\rangle_\ell + \langle \hat{x}\rangle_{1-\ell}$. The parameters $\langle \tilde{K}\rangle_\ell, \langle \tilde{x}\rangle_\ell, \langle \tilde{K}\tilde{x}\rangle_\ell$ are randomly generated by the DO. The inputs $\langle K\rangle_\ell, \langle x\rangle_\ell$ are uniformly random on $\mathbb{Z}_p^{m\times n}$ and $\mathbb{Z}_p^n$, respectively. $\langle \hat{K}\rangle_\ell, \langle \hat{x}\rangle_\ell, \hat{K}, \hat{x}$ can be derived by the linear computations of $\langle \tilde{K}\rangle_\ell, \langle \tilde{x}\rangle_\ell, \langle K\rangle_\ell, \langle x\rangle_\ell$ such that they are also uniformly random according to Lemma 2. The output $out_{\mathsf{BL},i} = \{\langle u\rangle_\ell\}$ is derived by the linear computations of $\langle \tilde{K}\rangle_\ell$, $\langle \tilde{x}\rangle_\ell, \langle \tilde{K}\tilde{x}\rangle_\ell \langle K\rangle_\ell, \langle x\rangle_\ell$, which is also uniformly random according to Lemma 2. Thus, both view and output of $P_\ell(\ell \in \{0,1\})$ are simulatable by the simulator $\mathcal{S}$, which proves that π^{p}_{BL} is secure in the honest-but-curious model.

For the sub-protocol $\pi^{(2,3)}_{\mathsf{Convert}}$, $P_\ell(\ell \in \{0,1\})$ holds the view $view_{\mathsf{Conv}(2,3),i} = \{\langle x\rangle_\ell, \langle \tilde{x}\rangle_\ell, \hat{x}, \langle \tilde{r}_1\rangle_\ell, \langle \tilde{r}_2\rangle_\ell\}$, where $\hat{x} = x + \tilde{x}$. The parameters $\langle \tilde{x}\rangle_\ell, \langle r_1\rangle_\ell, \langle r_2\rangle_\ell$ are randomly generated by the DO. The input $\langle x\rangle_\ell$ is uniformly random on $\mathbb{Z}_2^n$ (or $\mathbb{Z}_2^{m\times n}$), and $\hat{x}$ can be derived by the linear computations of $\langle x\rangle_\ell, \langle \tilde{x}\rangle_\ell$ such that it is also uniformly random according to *Lemma 2*. The output $out_{\mathsf{Conv}(2,3),i} = \{\langle x^*\rangle_\ell\}$ is derived by the linear computations of $\hat{x}, \langle \tilde{r}_1\rangle_\ell, \langle \tilde{r}_2\rangle_\ell$. According to Lemma 2, the output $out_{\mathsf{Conv}(2,3),i}$ is also uniformly random. Thus, the simulator $\mathcal{S}$ can simulate the view and output of $P_\ell(\ell \in \{0,1\})$, which proves that $\pi^{(2,3)}_{\mathsf{Convert}}$ is secure in the honest-but-curious model.

For the sub-protocol $\pi^{(3,2)}_{\mathsf{Convert}}$, $P_\ell(\ell \in \{0,1\})$ holds the view $view_{\mathsf{Conv}(3,2),i} = \{\langle \bar{v}\rangle_\ell, \langle v^*\rangle_\ell, \hat{v}^*, \langle \bar{r}_1\rangle_\ell, \langle \bar{r}_2\rangle_\ell\}$, where $\hat{v}^* = v^* + \bar{v}$. The parameters $\langle \bar{v}\rangle_\ell, \langle \bar{r}_1\rangle_\ell, \langle \bar{r}_2\rangle_\ell$ are randomly generated by the DO. The input $\langle v^*\rangle_\ell$ is uniformly random on $\mathbb{Z}_3^m$. $\hat{v}^*$ can be derived by the linear computations of $\langle \bar{v}\rangle_\ell, \langle v^*\rangle_\ell$ such that it is also uniformly random according to Lemma 2. The output $out_{\mathsf{Conv}(3,2),i} = \{\langle v\rangle_\ell\}$ is derived by the linear computations of $\hat{v}^*, \langle \bar{r}_1\rangle_\ell, \langle \bar{r}_2\rangle_\ell$. According to Lemma 2, the output $out_{\mathsf{Conv}(3,2),i}$ is also uniformly random. Thus, the simulator $\mathcal{S}$ can simulate the view and output of $P_\ell(\ell \in \{0,1\})$, which proves that $\pi^{(3,2)}_{\mathsf{Convert}}$ is secure in the honest-but-curious model.

Therefore, the protocol $\pi_{\mathsf{DLPN\text{-}wPRF}}$ is secure in the honest-but-curious model. □

Theorem 2. *ODiSC Π is adaptively simulation-secure if LPN-wPRF F^B_K is indistinguishable from random function and $\pi_{DLPN\text{-}wPRF}$ is secure in the honest-but-curious model.*

Proof. Given the leakage functions $\mathcal{L}_1, \mathcal{L}_2$ defined in Section 4.3, the simulator $\mathcal{S} = (\mathcal{S}_0, \ldots, \mathcal{S}_{q(\lambda)})$ generates $\mathsf{View}_{\mathcal{A},\mathsf{SIM}} = \{\langle \mathcal{P}^*\rangle_\ell, \langle \mathcal{I}^*\rangle_\ell, \mathsf{EDB}^*, \{\mathsf{Eu}^*_\iota, \langle \mathsf{tk}^*_\iota\rangle_i\}_{\iota\in[q(\lambda)]}\}$ as follows:

$\mathcal{S}_0(1^\lambda, \mathcal{L}_1(\mathsf{DB}))$: $\mathcal{S}_0$ first randomly samples

$$\langle K\rangle_\ell, \langle \tilde{K}\rangle_\ell \xleftarrow{\$} \mathbb{Z}_2^{m\times n}, \langle \tilde{x}\rangle_\ell \xleftarrow{\$} \mathbb{Z}_2^n,$$
$$\langle \bar{K}\rangle_\ell, \langle \tilde{R}_1\rangle_\ell, \langle \tilde{R}_2\rangle_\ell \xleftarrow{\$} \mathbb{Z}_3^{m\times n},$$
$$\langle \bar{x}\rangle_\ell, \langle \tilde{r}_1\rangle_\ell, \langle \tilde{r}_2\rangle_\ell, \langle \bar{K}\bar{x} + \bar{v}\rangle_\ell \xleftarrow{\$} \mathbb{Z}_3^m,$$
$$\langle \tilde{K}\tilde{x}\rangle_\ell, \langle \tilde{r}_1\rangle_\ell, \langle \tilde{r}_2\rangle_\ell \xleftarrow{\$} \mathbb{Z}_2^m,$$

then sets $\langle \mathcal{P}^* \rangle_\ell = \{\langle K \rangle_\ell, \langle \tilde{K} \rangle_\ell, \langle \tilde{x} \rangle_\ell, \langle \bar{K} \rangle_\ell, \langle \tilde{R}_1 \rangle_\ell, \langle \tilde{R}_2 \rangle_\ell, \langle \bar{x} \rangle_\ell, \langle \tilde{r}_1 \rangle_\ell, \langle \tilde{r}_2 \rangle_\ell, \langle \bar{K}\bar{x} + \bar{v} \rangle_\ell, \langle \tilde{K}\tilde{x} \rangle_\ell, \langle \bar{r}_1 \rangle_\ell, \langle \bar{r}_2 \rangle_\ell\}$.

To simulate the index share $\langle \mathcal{I} \rangle_\ell$, $\mathcal{S}_0$ randomly samples $y_{i,0}, y_{i,1}$ from $\{0,1\}^\lambda$ for $i \in [N_\mathrm{B}]$ and constructs an inverted index $\langle \mathcal{I}^* \rangle_\ell$ containing M address-value pairs. For the ϱth address-value pair, $\mathcal{S}_0$ randomly samples e_ϱ from $\{0,1\}^\lambda$ and stores it in the address field. In addition, $\mathcal{S}_0$ simulates the list $L^*_\varrho = \{(\mathsf{Eb}_i^{(\ell)}, \langle \mathsf{Ek}_i \rangle_\ell, \langle \mathsf{bv}_i \rangle_\ell)\}_{i \in [N_{w_\varrho}]}$ and stores it in the value field. Specifically, $\mathcal{S}_0$ first generates N_{w_ϱ} null matrices of the size $N_\mathrm{B} \times 2$. Then, for each $i \in [N_\mathrm{B}]$, $\mathcal{S}_0$ randomly samples k_i from $\{0,1\}^\lambda$; additively splits it into two random shares $\langle k_i \rangle_\ell, \langle k_i \rangle_{1-\ell}$; arranges $(\langle k_i \rangle_\ell, y_{i,0} \oplus \langle k_i \rangle_{1-\ell})$ to the ith row of $n_0^{\varrho,i}$ matrices randomly selected from the null matrix set; and arranges $(\langle k_i \rangle_\ell, y_{i,1} \oplus \langle k_i \rangle_{1-\ell})$ to the ith row of the rest of the $n_1^{\varrho,i}$ matrices. In this way, $\mathcal{S}_0$ obtains N_{w_ϱ} index ciphertext shares $\{\mathsf{Eb}_i^{(\ell)}\}_{i \in [N_{w_\varrho}]}$. In addition, $\mathcal{S}_0$ randomly generates a d-dimensional binary vector $\langle \mathsf{bv}_i \rangle_\ell$ and randomly samples $\langle \mathsf{Ek}_i \rangle_\ell$ from $\{0,1\}^t$ for $i \in [N_{w_\varrho}]$.

To simulate the encrypted database EDB, $\mathcal{S}_0$ randomly samples a string c_i^* from $\{0,1\}^{|c_i|}$ for $i \in [N]$ and outputs the simulated database cipherext $\mathsf{EDB}^* = \{c_1^*, c_2^*, \ldots, c_N^*\}$.

$\mathcal{S}_1(\mathcal{L}_2(\mathsf{DB}, Q_1, S_1))$: For the selected key w_ϱ in the search query Q_1, $\mathcal{S}_1$ sets e_ϱ as its ciphertext according to the leakage function $\mathcal{L}_2(\mathsf{DB}, Q_1, S_1)$, then randomly samples $\langle \mathsf{tk}^*_{\mathsf{add}} \rangle_\ell$ from $\{0,1\}^t$ and sets $\langle \mathsf{tk}^*_{\mathsf{add}} \rangle_{1-\ell} = e_\varrho \oplus \langle \mathsf{tk}^*_{\mathsf{add}} \rangle_\ell$. To simulate the sub-token $\langle \mathsf{tk}_{\mathsf{val}} \rangle_\ell$, $\mathcal{S}_1$ randomly samples ξ_i from $\{0,1\}^t$ for $i \in [N_\mathrm{B}]$, then sets $\langle \mathsf{tk}^*_{\mathsf{val}} \rangle_\ell = \oplus_{i \in [N_\mathrm{B}]} \xi_i$. To simulate the attribute ciphertext Eu_1, $\mathcal{S}_1$ randomly samples a d-dimensional vector from $\mathbb{Z}_{2^t}^d$ as Eu_1^*. In addition, $\mathcal{S}_1$ remembers the association between $S_1[\iota]$ and $\mathsf{Eu}_1^*[\iota]$. Finally, $\mathcal{S}_1$ sends Eu_1^* and $\langle \mathsf{tk}_1^* \rangle_\ell$ to $\mathcal{A}$, where $\langle \mathsf{tk}_1^* \rangle_\ell = (\langle \mathsf{tk}^*_{\mathsf{add}} \rangle_\ell, \langle \mathsf{tk}^*_{\mathsf{val}} \rangle_\ell)$.

$\mathcal{S}_i(\mathcal{L}_2(\mathsf{DB}, Q_1, \ldots, Q_i, S_1, \ldots, S_i))$: For the user attribute set $S_i (i \geq 2)$, $\mathcal{S}_i$ checks whether the attribute $S_i[j] (j \in [d])$ has appeared before based on the leakage function $\mathcal{L}_2(\mathsf{DB}, Q_1, \ldots, Q_i, S_1, \ldots, S_i)$. Let U be the attribute set not appearing before. $\mathcal{S}_i$ randomly samples g_ι from $\{0,1\}^t$ for $\iota \in [|U|]$ and combines them with the ciphertexts of appeared attributes to obtain the attribute ciphertext vector Eu_i^*. For the search query $Q_i (i \geq 2)$, $\mathcal{S}_i$ generates $\langle \mathsf{tk}_i^* \rangle_\ell$ the same way $\mathcal{S}_1$ does.

The indistinguishability of the simulated view $\mathsf{View}_{\mathcal{A},\mathrm{SIM}}$ from the real view $\mathsf{View}_{\mathcal{A},\mathrm{REAL}}$ follows directly from the following facts:

- Since the PPT distinguisher $\mathcal{D}$ does not know the secret key of AES, the CPA-security of AES will guarantee that the simulated ciphertext c_i^* and real ciphertext c_i are indistinguishable for $\mathcal{D}$, which has been proved in [11]. Formally, $\mathsf{Adv}_{\mathrm{AES}}(\mathcal{D}(c_i, c_i^*)) \leq \mathsf{neg}_1(1^\lambda)$. Thus, we have
$$\begin{aligned}\mathsf{Adv}(\mathcal{D}(\mathsf{EDB}, \mathsf{EDB}^*)) &= 1 - (1 - \mathsf{Adv}_{\mathrm{AES}}(\mathcal{D}(c_i, c_i^*)))^N \\ &\leq 1 - (1 - \mathsf{neg}_1(\lambda))^N \leq \mathsf{neg}_1(\lambda).\end{aligned}$$
- Since the PPT distinguisher $\mathcal{D}$ does not know the secret key K and the inputs of LPN-wPRF are independently uniformly random strings ensured by the collision-free hash function H, the output of LPN-wPRF is indistinguishable from the output of a truly random function, i.e., $\mathsf{Adv}_{\text{LPN-wPRF}}(\mathcal{D}(y_i, y_i^*)) \leq \mathsf{neg}_2(1^\lambda)$, which has been proven in [13]. The difference between the real index share $\langle \mathcal{I} \rangle_\ell$ and the simulated index share $\langle \mathcal{I}^* \rangle_\ell$ is to replace all outputs of LPN-wPRF with randomness. Thus, we have Equation (19). For the same reason, we also have $\mathsf{Adv}(\mathcal{D}(\langle \mathsf{Eu}_i \rangle_\ell, \langle \mathsf{Eu}_i^* \rangle_\ell)) \leq \mathsf{neg}_2(\lambda)$.
$$\begin{aligned}\mathsf{Adv}(\mathcal{D}(\langle \mathcal{I} \rangle_\ell, \langle \mathcal{I}^* \rangle_\ell)) &= 1 - (1 - \mathsf{Adv}_{\text{LPN-wPRF}}(\mathcal{D}(y_i, y_i^*)))^{2N_\mathrm{B}} \\ &\leq 1 - (1 - \mathsf{neg}_2(\lambda))^{2N_\mathrm{B}} \leq \mathsf{neg}_2(\lambda).\end{aligned} \tag{19}$$
- Theorem 1 proves that the output of the protocol $\pi_{\text{DLPN-wPRF}}$ is uniformly random. Thus, we have $\mathsf{Adv}(\mathcal{D}(\langle \mathsf{tk}_i \rangle_\ell, \langle \mathsf{tk}_i^* \rangle_\ell)) \leq \mathsf{neg}_3(\lambda)$.

Table 4. Theoretical Computation and Communication Costs in Different Schemes

Types	Schemes	IndexBuild	QueryIssue	Search
Comp. costs	MRSF	$2MN \cdot \mathsf{T}_{\mathrm{DOT}}$	$2M \cdot \mathsf{T}_{\mathrm{DOT}}$	$2N \cdot \mathsf{T}_{\mathrm{DOT}}$
	NIMC-SSE-Π_1	$N_W(3\mathsf{T}_{\mathrm{PRF}} + \mathsf{T}_{\mathrm{EXP}}) + N\mathsf{T}_{\mathrm{ABE.Enc}}$ $+M(\mathsf{T}_{\mathrm{PRF}} + \mathsf{T}_{\mathrm{Inv}})$	$N_w N_Q \cdot (2\mathsf{T}_{\mathrm{PRF}}$ $+3\mathsf{T}_{\mathrm{EXP}})$	$N_w N_Q \mathsf{T}_{\mathrm{EXP}}$
	ODiSC	$(2N_{\mathrm{B}} + Nd + M + MN_{\mathrm{B}})$ $\cdot (\mathsf{T}_{\text{LPN-wPRF}} + \mathsf{T}_{\mathrm{SHA}})$	$N_{\mathrm{B}} \cdot \mathsf{T}_{\mathrm{SHA}}$	$(N_{\mathrm{B}} + 1)\mathsf{T}_{\text{DLPN-wPRF}}$ $+N_w(2N_{\mathrm{B}} + d)\mathsf{T}_{\mathrm{XOR}}$
Comm. costs	MRSF	$2MN \cdot b_{\mathrm{Float}}$	$2M \cdot b_{\mathrm{Float}}$	–
	NIMC-SSE-Π_1	$N_W(\lvert Z_p\rvert + 2d\lvert G\rvert + \lvert G_T\rvert)$	$N_w N_Q \lvert G\rvert + t$	$N_{\mathrm{CQ}}(2d\lvert G\rvert + \lvert G_T\rvert)$
	ODiSC	$2(N_W(2N_{\mathrm{B}} \cdot t + d + t) + Mt)$	$2N_{\mathrm{B}}(n+1) + 2n$	$N_{\mathrm{B}}(8n + 2m) + 2N_w \cdot t$

Notes. $\mathsf{T}_{\mathrm{DOT}}$: Time complexity of inner product; $\mathsf{T}_{\mathrm{PRF}}$: Time complexity of PRF; $\mathsf{T}_{\mathrm{EXP}}$: Time complexity of modular exponentiation; $\mathsf{T}_{\mathrm{Inv}}$: Time complexity of inverse operation; b_{Float}: Size of a floating number; $\lvert Z_p\rvert$, $\lvert G\rvert$, $\lvert G_T\rvert$: Element length in the groups Z_p, G, G_T; N_Q: Number of query keywords; N_{CQ}: Number of search results.

Therefore, we have Equation (20), which means that ODiSC is adaptively simulation-secure.

$$\begin{aligned}
&|\Pr[\mathcal{D}(\mathrm{View}_{\mathcal{A},\mathrm{REAL}}) = 1] - \Pr[\mathcal{D}(\mathrm{View}_{\mathcal{A},\mathrm{SIM}}) = 1]| \\
&= \mathsf{Adv}(\mathcal{D}(\mathsf{EDB}, \mathsf{EDB}^*)) + \mathsf{Adv}\Big(\mathcal{D}(\langle \mathcal{I} \rangle_\ell, \langle \mathcal{I}^* \rangle_\ell)\Big) \\
&+ \mathsf{Adv}\Big(\mathcal{D}(\langle \mathsf{Eu}_i \rangle_\ell, \langle \mathsf{Eu}_i^* \rangle_\ell)\Big) + \mathsf{Adv}\Big(\mathcal{D}(\langle \mathsf{tk}_i \rangle_\ell, \langle \mathsf{tk}_i^* \rangle_\ell)\Big) \\
&\leq \mathsf{neg}_1(\lambda) + 2\mathsf{neg}_2(\lambda) + \mathsf{neg}_3(\lambda) \stackrel{\mathrm{def}}{=} \mathsf{neg}(\lambda).
\end{aligned} \tag{20}$$

□

7 PERFORMANCE ANALYSIS

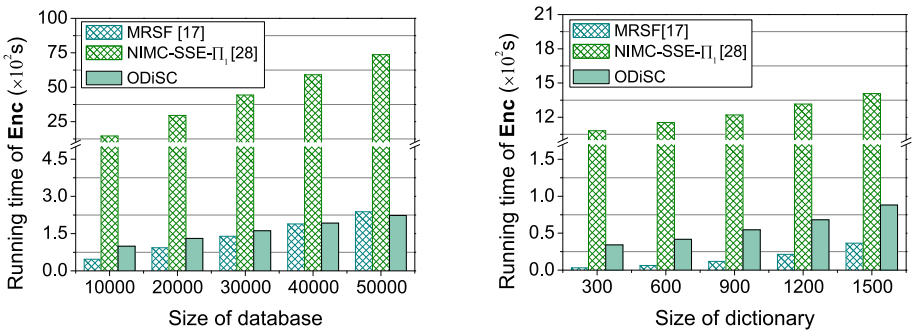
We analyze the performance of ODiSC theoretically and experimentally by comparing it with MRSF [17] and NIMC-SSE-Π_1 [28].

7.1 Theoretical Analysis

We present the computation and communication costs of ODiSC and comparison schemes in Table 4.

As for the computation cost in ODiSC, we mainly consider several time-consuming operations, i.e., LPN-wPRF, DLPN-wPRF$^{\mathsf{on}}$, hash function. Let $\mathsf{T}_{\text{LPN-wPRF}}, \mathsf{T}_{\text{DLPN-wPRF}}, \mathsf{T}_{\mathrm{SHA}}$ be corresponding time complexities. In IndexBulid, the DO first encrypts N_{B}-dimensional data vector and d-dimensional policy vector for each outsourced data, which costs $(2N_{\mathrm{B}} + Nd)(\mathsf{T}_{\text{LPN-wPRF}} + \mathsf{T}_{\mathrm{SHA}})$. Then, the DO encrypts M keywords and generates $M \cdot N_{\mathrm{B}}$ noises $\{k_{\varrho,l}\}$ to split the lists $\{L_{w_\varrho}\}_{\varrho \in [M]}$, which costs $M(\mathsf{T}_{\text{LPN-wPRF}} + \mathsf{T}_{\mathrm{SHA}})$ and $MN_{\mathrm{B}}(\mathsf{T}_{\text{LPN-wPRF}} + \mathsf{T}_{\mathrm{SHA}})$, respectively. In QueryIssue, the QU computes n-bit digest for each dimension of a query vector and then converts the digest to an n-dimensional vector, which costs $N_{\mathrm{B}}\mathsf{T}_{\mathrm{SHA}}$. In Search, CSPs first cooperate to execute $(N_{\mathrm{B}} + 1)$ DLPN-wPRF$^{\mathsf{on}}$, which costs $(N_{\mathrm{B}} + 1) \cdot \mathsf{T}_{\text{DLPN-wPRF}}$. Then, each CSP executes $N_w(2N_{\mathrm{B}} + d)$ XOR operations to compute N_w key shares, where N_w denotes the size of the selected list in address *field retrieval*.

As for the communication cost of ODiSC, we represent the number of (keyword, *id*) pairs extracted from the outsourced database as N_W. The protocol DLPN-wPRF$^{\mathsf{on}}$ needs three rounds of communication to open the masked intermediate values. When the secret key K is a generator matrix of quasi-cyclic codes, the cost per party is $2n$ bits in the first and second rounds, and m bits in the third round. In IndexBulid, the communication cost is derived from uploading two index shares. The size of each index share is $N_W(2N_{\mathrm{B}} \cdot t + d + t) + M \cdot t$ bits. In QueryIssue, the communication cost originates from sending two query shares. The size of each query share is $n + N_{\mathrm{B}} \cdot n + N_{\mathrm{B}}$ bits. In



(a) $M = 1761$ (b) $N = 10^4$

Fig. 9. Computation cost of IndexBulid.

Table 5. Communication Cost of IndexBulid When $N = 10^4$

M	300	600	900	1,200	1,500
N_B	205	322	380	410	497
N_W	7,756	14,112	19,549	27,195	34,998
MRSF (MB)	25.18	48.07	70.95	93.84	116.73
NIMC-SSE-Π_1 (MB)	86.30	157.03	217.53	302.61	389.44
ODiSC (MB)	97.37	277.94	454.24	681.69	1,063.13

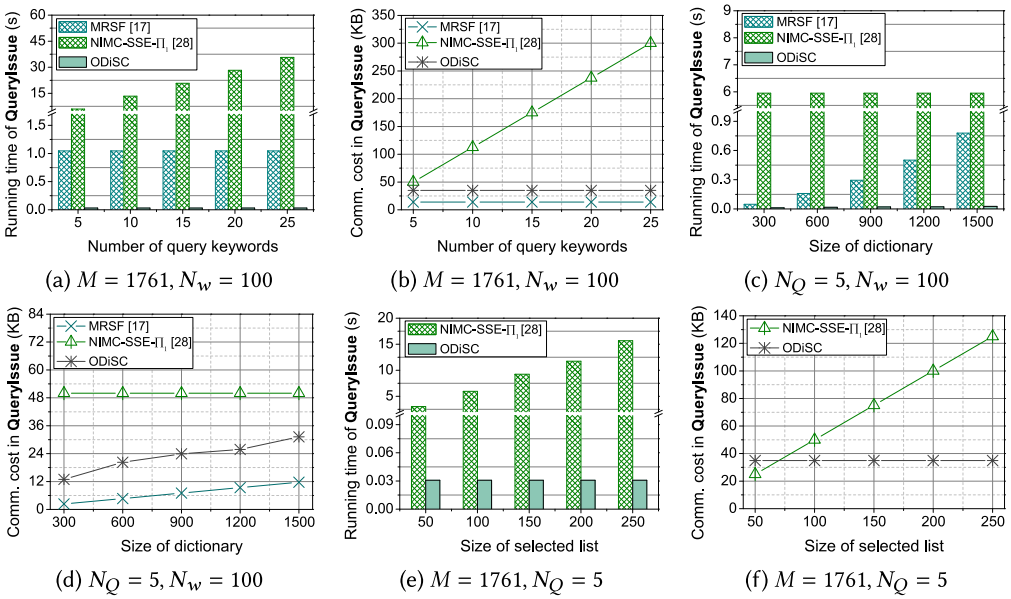
Search, the communication cost stems from executing $(N_B + 1)$ DLPN-wPRF^{on} in *token generation* and returning search results in value *field retrieval*, which costs $(N_B + 1)\cdot 2(4n + m)$ and $N_w \cdot 2t$, respectively, when the secret key K is a generator matrix of quasi-cyclic codes. Note that in the theoretical analysis and the following tests, we do not consider the encryption time and storage cost of the outsourced database because ODiSC and the comparison schemes have the same.

7.2 Experimental Tests

We conduct experiments on a server with 3.20 GHz 3.19 GHz Intel(R) Core(TM) i7-8700K CPU using Python and the **Paring-based Cryptography (PBC)** library. We randomly choose 10,000 business data from the Yelp dataset[6] as the test dataset. We extract 1,761 unique keywords from their category descriptions. The maximum number of extracted keywords in each data is $\omega = 19$, and the number of (keyword, *id*) pairs is $N_W = 42{,}156$. Each data also contains 44 attributes. We set the size of the **Bloom filter (BF)** in ODiSC as $N_B = -16\omega/\ln(1 - (10^{-6})^{1/16})$, such that the false positive of the BF constructed from 16 independent hash functions is about 10^{-6}. We set the number of access roles and random numbers in MRSF as 30. We select Type A elliptic curve for NIMC-SSE-Π_1, where the parameters p and q are set to 160 bits and 512 bits, respectively. Then, we have $|Z_p| = 160$ bits and $|G| = |G_T| = 1{,}024$ bits. In addition, we set the security parameter as $\lambda = 128$ bits, then $m = 256, n = 256, t = 128$. Note that the index structure of ODiSC is maintained via key-value storage.

Figures 9(a) and 9(b) show that the index construction time of ODiSC increases with the size of database N and the size of dictionary M, but is at least 10× faster than that of NIMC-SSE-Π_1. The index construction time of MRSE is about the same as ODiSC, and it grows faster than ODiSC as the size of the database increases. When $N = 10{,}000, M = 1{,}761$, ODiSC takes a few minutes to build the secure index. Table 5 shows that the communication cost of ODiSC in the index construction

[6] https://www.yelp.com/dataset



(a) $M = 1761, N_w = 100$ (b) $M = 1761, N_w = 100$ (c) $N_Q = 5, N_w = 100$

(d) $N_Q = 5, N_w = 100$ (e) $M = 1761, N_Q = 5$ (f) $M = 1761, N_Q = 5$

Fig. 10. Performance of QueryIssue.

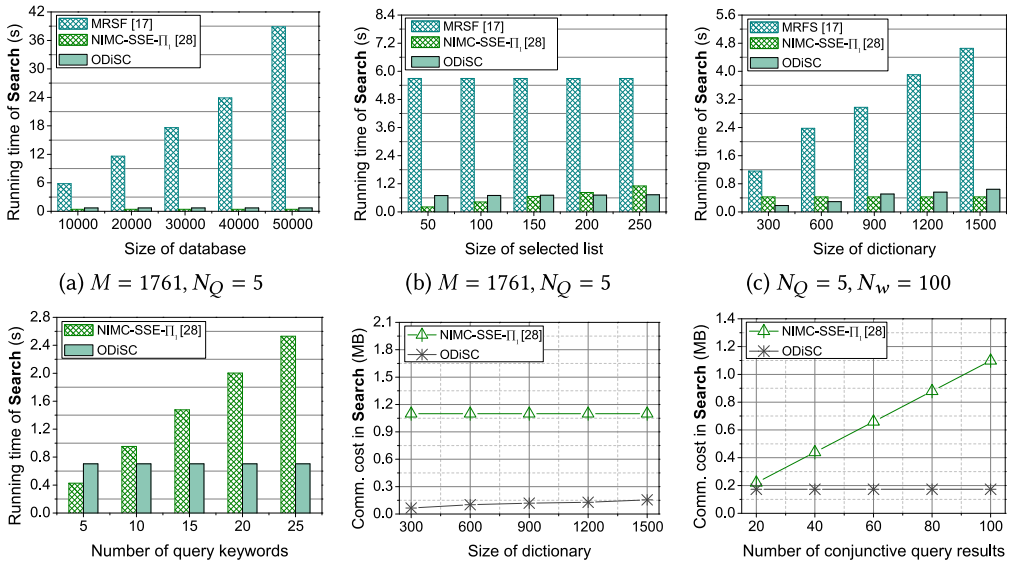
phase super-linearly increases with the number of (keyword, *id*) pairs N_W. This is because the length of the Bloom filter increases with N_W to maintain the false positive at 10^{-6}. Besides, we find the communication cost of ODiSC is higher than that of MRSF and NIMC-SSE-Π_1. Fortunately, index outsourcing is performed once only.

Figure 10 shows that the query generation time and outsourcing cost in ODiSC both increase with the size of dictionary M but are not affected by the number of query keywords N_Q and the size of selected list N_w, which are opposite to NIMC-SSE-Π_1. Compared with MRSF, ODiSC is efficient in query generation but consumes more communication cost. When M = 1,761, ODiSC takes about 30 ms to generate a secret-shared query and 35 KB to sends it to CSPs, which is at least 180× faster and 1× more communication cost saving than NIMC-SSE-Π_1. The microsecond-level query generation time and KB-level query size are friendly to resource-limited devices.

Figures 11(a) and 11(b) show that the search time of ODiSC is almost not affected by the size of database N and the size of selected list N_w, which indicates that the search time of ODiSC is mainly derived from the execution of $N_B + 1$ DLPN-wPRF^{on} and the effect of XOR operation is negligible. When N = 10,000, M = 1,761, ODiSC takes about 0.7 s for retrieval, which is about 7× faster than MRSF. Figures 11(c) and 11(d) show that the search time of ODiSC increases with the size of dictionary M but is not affected by the number of query keywords N_Q. When M = 1,761, N_Q = 5, N_w = 100, the search time of ODiSC is higher than that of NIMC-SSE-Π_1, but it is less than 1 s. Figures 11(e) and 11(f) show that the communication cost of ODiSC in Search increases with M but is not affected by the number of search results N_{CQ}. Table 6 shows that the communication cost of returning search results is negligible compared with the communication cost in search token generation. When M = 1,761, N_{CQ} = 100, N_w = 100, the communication cost of ODiSC is about 0.2 MB, which is at least 4× more cost saving than NIMC-SSE-Π_1.

8 CONCLUSION

In this work, we proposed an owner-free distributed SSE scheme supporting conjunctive queries, namely ODiSC. In ODiSC, search tokens were generated with the data owner free from sharing



(a) $M = 1761, N_Q = 5$ (b) $M = 1761, N_Q = 5$ (c) $N_Q = 5, N_w = 100$

(d) $M = 1761, N_w = 100$ (e) $N_w = 100, N_{CQ} = 100$ (f) $M = 1761, N_w = 100$

Fig. 11. Performance of Search.

Table 6. Communication Cost of Search When $M = 1,761$

Size of Selected List N_w	50	100	150	200	250
NIMC-SSE-Π_1 (MB)	1.0986	1.0986	1.0986	1.0986	1.0986
ODiSC (MB)	0.1712	0.1727	0.1743	0.1758	0.1773

key or staying online via the distributed LPN-wPRF protocol. Moreover, it realized fine-grained conjunctive query in the distributed architecture by combining additive secret sharing with symmetric-key hidden vector encryption, such that ciphertext retrieval is achieved without unauthorized access and result decryption is completed without assistance from the data owner. Both security and performance analysis demonstrated that ODiSC could guarantee data confidentiality and achieve efficient retrieval.

As part of our future work, we will focus on achieving volume-hiding retrieval and providing fault tolerance for stored data.

REFERENCES

[1] Asra Ali, Tancrede Lepoint, Sarvar Patel, Mariana Raykova, Phillipp Schoppmann, Karn Seth, and Kevin Yeo. 2021. Communication–computation trade-offs in PIR. In *Proc. USENIX Security Symposium (USENIX'21)*. 1811–1828.

[2] Donald Beaver. 1991. Efficient multiparty protocols using circuit randomization. In *Proc. Annual International Cryptology Conference (CRYPTO'91)*. Springer, 420–432.

[3] Dan Bogdanov, Sven Laur, and Jan Willemson. 2008. Sharemind: A framework for fast privacy-preserving computations. In *Proc. European Symposium on Research in Computer Security (ESORICS'08)*. Springer, 192–206.

[4] Dan Bogdanov, Margus Niitsoo, Tomas Toft, and Jan Willemson. 2012. High-performance secure multi-party computation for data mining applications. *International Journal of Information Security* 11, 6 (2012), 403–418.

[5] Dan Boneh, Yuval Ishai, Alain Passelègue, Amit Sahai, and David Wu. 2018. Exploring crypto dark matter: New simple PRF candidates and their applications. In *Proc. Theory of Cryptography Conference (TCC'18)*, Vol. 11240. Springer, 699–729.

[6] David Cash, Joseph Jaeger, Stanislaw Jarecki, Charanjit Jutla, Hugo Krawczyk, Marcel-Cătălin Roşu, and Michael Steiner. 2014. Dynamic searchable encryption in very-large databases: Data structures and implementation. *Proc. Annual Network and Distributed System Security Symposium (NDSS'14)* (2014), 1–16.

[7] David Cash, Stanislaw Jarecki, Charanjit Jutla, Hugo Krawczyk, Marcel-Cătălin Roşu, and Michael Steiner. 2013. Highly-scalable searchable symmetric encryption with support for boolean queries. In *Proc. Annual Cryptology Conference (CRYPTO'13)*. Springer, 353–373.

[8] Ke Cheng, Yantian Hou, and Liangmin Wang. 2018. Secure similar sequence query on outsourced genomic data. In *Proc. Asia Conference on Computer and Communications Security (AsiaCCS'18)*. ACM, 237–251.

[9] Jie Cui, Han Zhou, Yan Xu, and Hong Zhong. 2019. OOABKS: Online/offline attribute-based encryption for keyword search in mobile cloud. *Information Sciences* 489 (2019), 63–77.

[10] Reza Curtmola, Juan Garay, Seny Kamara, and Rafail Ostrovsky. 2006. Searchable symmetric encryption: Improved definitions and efficient constructions. In *Proc. ACM Conference on Computer and Communications Security (CCS'06)*. ACM, 79–88.

[11] Reza Curtmola, Juan Garay, Seny Kamara, and Rafail Ostrovsky. 2011. Searchable symmetric encryption: Improved definitions and efficient constructions. *Journal of Computer Security* 19, 5 (2011), 895–934.

[12] Daniel Demmler, Thomas Schneider, and Michael Zohner. 2015. ABY-A framework for efficient mixed-protocol secure two-party computation. In *Proc. Annual Network and Distributed System Security Symposium (NDSS'15)*. The Internet Society, 1–15.

[13] Itai Dinur, Steven Goldfeder, Tzipora Halevi, Yuval Ishai, Mahimna Kelkar, Vivek Sharma, and Greg Zaverucha. 2021. MPC-friendly symmetric cryptography from alternating moduli: Candidates, protocols, and applications. In *Proc. Annual International Cryptology Conference (CRYPTO'21)*. Springer, 517–547.

[14] Stanislaw Jarecki, Charanjit Jutla, Hugo Krawczyk, Marcel Rosu, and Michael Steiner. 2013. Outsourced symmetric private information retrieval. In *Proc. ACM SIGSAC Conference on Computer & Communications Security (CCS'13)*. ACM, 875–888.

[15] Shabnam Kasra Kermanshahi, Joseph K. Liu, Ron Steinfeld, Surya Nepal, Shangqi Lai, Randolph Loh, and Cong Zuo. 2021. Multi-client cloud-based symmetric searchable encryption. *IEEE Transactions on Dependable and Secure Computing* 18, 5 (2021), 2419–2437.

[16] Shangqi Lai, Sikhar Patranabis, Amin Sakzad, Joseph K. Liu, Debdeep Mukhopadhyay, Ron Steinfeld, Shi-Feng Sun, Dongxi Liu, and Cong Zuo. 2018. Result pattern hiding searchable encryption for conjunctive queries. In *Proc. ACM SIGSAC Conference on Computer and Communications Security (CCS'18)*. ACM, 745–762.

[17] Jiayi Li, Jianfeng Ma, Yinbin Miao, Ruikang Yang, Ximeng Liu, and Kim-Kwang Raymond Choo. 2020. Practical multi-keyword ranked search with access control over encrypted cloud data. *IEEE Transactions on Cloud Computing* 10, 3 (2020), 2005–2019.

[18] Hsiao-Ying Lin and Wen-Guey Tzeng. 2011. A secure erasure code-based cloud storage system with secure data forwarding. *IEEE Transactions on Parallel and Distributed Systems* 23, 6 (2011), 995–1003.

[19] Lin Liu, Jinshu Su, Ximeng Liu, Rongmao Chen, Kai Huang, Robert H. Deng, and Xiaofeng Wang. 2019. Toward highly secure yet efficient KNN classification scheme on outsourced cloud data. *IEEE Internet of Things Journal* 6, 6 (2019), 9841–9852.

[20] Xueqiao Liu, Guomin Yang, Yi Mu, and Robert H. Deng. 2018. Multi-user verifiable searchable symmetric encryption for cloud storage. *IEEE Transactions on Dependable and Secure Computing* 17, 6 (2018), 1322–1332.

[21] Yang Liu, Zhuo Ma, Ximeng Liu, Siqi Ma, and Kui Ren. 2022. Privacy-preserving object detection for medical images with faster R-CNN. *IEEE Transactions on Information Forensics and Security* 17 (2022), 69–84.

[22] Rasoul Akhavan Mahdavi and Florian Kerschbaum. 2022. Constant-weight PIR: Single-round keyword PIR via constant-weight equality operators. In *Proc. 31st USENIX Security Symposium (USENIX'22)*. 1723–1740.

[23] Yinbin Miao, Robert H. Deng, Kim-Kwang Raymond Choo, Ximeng Liu, Jianting Ning, and Hongwei Li. 2019. Optimized verifiable fine-grained keyword search in dynamic multi-owner settings. *IEEE Transactions on Dependable and Secure Computing* 18, 4 (2019), 1804–1820.

[24] Sarvar Patel, Giuseppe Persiano, Kevin Yeo, and Moti Yung. 2019. Mitigating leakage in secure cloud-hosted data structures: Volume-hiding for multi-maps via hashing. In *Proc. ACM SIGSAC Conference on Computer and Communications Security (CCS'19)*. 79–93.

[25] Sikhar Patranabis and Debdeep Mukhopadhyay. 2017. Lightweight symmetric-key hidden vector encryption without pairings. *Cryptology ePrint Archive* (2017).

[26] Thomas Schneider and Oleksandr Tkachenko. 2019. EPISODE: Efficient privacy-preserving similar sequence queries on outsourced genomic databases. In *Proc. ACM Asia Conference on Computer and Communications Security (AsiaCCS'19)*. ACM, 315–327.

[27] Dawn Xiaoding Song, David Wagner, and Adrian Perrig. 2000. Practical techniques for searches on encrypted data. In *Proc. IEEE Symposium on Security and Privacy (S&P'00)*. IEEE, 44–55.

[28] Shi-Feng Sun, Cong Zuo, Joseph K. Liu, Amin Sakzad, Ron Steinfeld, Tsz Hon Yuen, Xingliang Yuan, and Dawu Gu. 2022. Non-interactive multi-client searchable encryption: Realization and implementation. *IEEE Transactions on Dependable and Secure Computing* 19, 1 (2022), 452–467.

[29] Qiuyun Tong, Xinghua Li, Yinbin Miao, Ximeng Liu, Jian Weng, and Robert Deng. 2023. Privacy-preserving boolean range query with temporal access control in mobile computing. *IEEE Transactions on Knowledge and Data Engineering* 35, 5 (2023), 5159–5172.

[30] Qiuyun Tong, Yinbin Miao, Lei Chen, Jian Weng, Ximeng Liu, Kim-Kwang Raymond Choo, and Robert H. Deng. 2021. Vfirm: Verifiable fine-grained encrypted image retrieval in multi-owner multi-user settings. *IEEE Transactions on Services Computing* 15, 6 (2021), 3606–3619.

[31] Qiuyun Tong, Yinbin Miao, Ximeng Liu, Kim-Kwang Raymond Choo, Robert H. Deng, and Hongwei Li. 2020. VPSL: Verifiable privacy-preserving data search for cloud-assisted Internet of Things. *IEEE Transactions on Cloud Computing* 10, 4 (2020), 2964–2976.

[32] Qiuyun Tong, Yinbin Miao, Jian Weng, Ximeng Liu, Kim-Kwang Raymond Choo, and Robert Deng. 2023. Verifiable fuzzy multi-keyword search over encrypted data with adaptive security. *IEEE Transactions on Knowledge and Data Engineering* 35, 5 (2023), 5386–5399.

[33] Jianfeng Wang, Shi-Feng Sun, Tianci Li, Saiyu Qi, and Xiaofeng Chen. 2022. Practical volume-hiding encrypted multi-maps with optimal overhead and beyond. In *Proc. ACM SIGSAC Conference on Computer and Communications Security (CCS'22)*. 2825–2839.

[34] Mingyue Wang, Yinbin Miao, Yu Guo, Cong Wang, Hejiao Huang, and Xiaohua Jia. 2021. Attribute-based encrypted search for multi-owner and multi-user model. In *Proc. IEEE International Conference on Communications (ICC'21)*. IEEE, 1–7.

[35] Xiangyu Wang, Jianfeng Ma, Ximeng Liu, Robert H. Deng, Yinbin Miao, Dan Zhu, and Zhuoran Ma. 2020. Search me in the dark: Privacy-preserving boolean range query over encrypted spatial data. In *Proc. IEEE Conference on Computer Communications (INFOCOM'20)*. IEEE, 2253–2262.

[36] Yijie Wang and Sijun Li. 2006. Research and performance evaluation of data replication technology in distributed storage systems. *Computers & Mathematics with Applications* 51, 11 (2006), 1625–1632.

[37] Zhihua Xia, Qi Gu, Lizhi Xiong, Wenhao Zhou, and Jian Weng. 2020. Privacy-preserving image retrieval based on additive secret sharing. *arXiv preprint arXiv:2009.06893* (2020).

[38] Yang Yang, Ke Mu, and Robert H. Deng. 2022. Lightweight privacy-preserving GAN framework for model training and image synthesis. *IEEE Transactions on Information Forensics and Security* 17 (2022), 1083–1098.

[39] Kai Zhang, Mi Wen, Rongxing Lu, and Kefei Chen. 2020. Multi-client sub-linear boolean keyword searching for encrypted cloud storage with owner-enforced authorization. *IEEE Transactions on Dependable and Secure Computing* 18, 6 (2020), 2875–2887.

[40] Yandong Zheng, Rongxing Lu, Yunguo Guan, Songnian Zhang, Jun Shao, and Hui Zhu. 2022. Efficient and privacy-preserving similarity query with access control in eHealthcare. *IEEE Transactions on Information Forensics and Security* 17 (2022), 880–893.

[41] Lu Zhou, Youwen Zhu, and Aniello Castiglione. 2017. Efficient k-NN query over encrypted data in cloud with limited key-disclosure and offline data owner. *Computers & Security* 69 (2017), 84–96.

[42] Youwen Zhu, Rui Xu, and Tsuyoshi Takagi. 2013. Secure k-NN computation on encrypted cloud data without sharing key with query users. In *Proc. International workshop on Security in Cloud Computing (AsiaCCS Workshop'13)*. ACM, 55–60.