Privacy-Preserving and Trusted Keyword Search for Multi-Tenancy Cloud

Xiaojie Zhu, *Member, IEEE*, Peisong Shen, Yueyue Dai, *Member, IEEE*, Lei Xu, *Member, IEEE*, and Jiankun Hu, *Senior Member, IEEE*

***Abstract*— Cloud service models intrinsically cater to multiple tenants. In current multi-tenancy model, cloud service providers isolate data within a single tenant boundary with no or minimum cross-tenant interaction. With the booming of cloud applications, allowing a user to search across tenants is crucial to utilize stored data more effectively. However, conducting such a search operation is inherently risky, primarily due to privacy concerns. Moreover, existing schemes typically focus on a single tenant and are not well suited to extend support to a multi-tenancy cloud, where each tenant operates independently. In this article, to address the above issue, we provide a privacy-preserving, verifiable, accountable, and parallelizable solution for "privacy-preserving keyword search problem" among multiple independent data owners. We consider a scenario in which each tenant is a data owner and a user's goal is to efficiently search for granted documents that contain the target keyword among all the data owners. We first propose a verifiable yet accountable keyword searchable encryption (VAKSE) scheme through symmetric bilinear mapping. For verifiability, a message authentication code (MAC) is computed for each associated piece of data. To maintain a consistent size of MAC, the computed MACs undergo an exclusive OR operation. For accountability, we propose a keyword-based accountable token mechanism where the client's identity is seamlessly embedded without compromising privacy. Furthermore, we introduce the parallel VAKSE scheme, in which the inverted index is partitioned into small segments and all of them can be processed synchronously. We also conduct formal security analysis and comprehensive experiments to demonstrate the data privacy preservation and efficiency of the proposed schemes, respectively.**

***Index Terms*— Symmetric searchable encryption, verification, accountability, fine-grained access control, parallel search, multi-tenancy.**

Manuscript received 11 July 2023; revised 11 December 2023 and 22 February 2024; accepted 8 March 2024. Date of publication 13 March 2024; date of current version 6 May 2024. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Alptekin Küpçü. *(Corresponding author: Xiaojie Zhu.)*

Xiaojie Zhu is with the Computer, Electrical and Mathematical Sciences and Engineering Division, King Abdullah University of Science and Technology (KAUST), Thuwal 23955-6900, Saudi Arabia (e-mail: xiaojie.zhu@kaust.edu.sa).

Peisong Shen is with the Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China (e-mail: shenpeisong@iie.ac.cn).

Yueyue Dai is with the School of Cyber Science and Engineering, Huazhong University of Science and Technology, Wuhan 430074, China (e-mail: yueyuedai@ieee.org).

Lei Xu is with the School of Mathematics and Statistics, Nanjing University of Science and Technology, Nanjing 210094, China (e-mail: xuleicrypto@gmail.com).

Jiankun Hu is with the School of Engineering and Information Technology, The University of New South Wales, Canberra, NSW 2600, Australia (e-mail: j.hu@adfa.edu.au).

Digital Object Identifier 10.1109/TIFS.2024.3377549

I. Introduction

CLOUD computing has had a profound impact on data management. It offers massive storage and computing resources, payment-on-demand, and flexible scalability. Motivated by these advantages, thousands of clients are opting for cloud services. One typical application area is healthcare, and some applications are Healthvana [1] and CDPHP [2]; both the platforms are the tenants of Amazon [3]. Healthvana stores patient reports and CDPHP stores doctor information. It is desirable for a patient to search both the datasets to find the most suitable doctor by matching the patient data with the doctor information. For example, HIV patients store their reports in Healthvana and seek for suitable doctors from CDPHP. However, such a search across tenancies is challenging. Each tenant is an independent data owner and must abide the privacy laws, such as HIPAA [4], which are enforced to protect individuals' medical data privacy. In addition, for their own interests, companies treat patient data as an asset and tend to maintain complete control over it.

Data encryption is the best practice for maintaining data privacy. Each data owner encrypts their data before outsourcing it to the cloud. This guarantees the confidentiality of the data but greatly reduces their utility. A user must download an entire dataset in order to retrieve one piece of data. Considering data utility and privacy, Song et al. [5] introduced the primitives of symmetric searchable encryption (SSE). SSE is a keyword search technique that allows search over the ciphertext without decryption. Goh et al. [6] proposed a secure index to improve search efficiency. Subsequently, Curtmola et al. [7] formalized the security definition of SSE and proposed two constructions that corresponded to nonadaptive semantic security and adaptive semantic security.

In early research, most works on SSE focused on the honest-but-curious cloud service provider (CSP). In such a model, the search result is fully trusted and the CSP is assumed to honestly follow the protocol specification. Search results in practice may contain corrupted data due to underlying hardware/software failures. In addition, for self-interest, the CSP may deviate from the protocol specification. For example, to reduce computational costs, CSP may randomly choose data as a search result. To mitigate this problem, Chai and Gong [8] proposed verifiable SSE, where the search result includes not only retrieved documents but also proof of the correctness and completeness of the search. The correctness of the search means that the returned search result matches the query. The

completeness of the search means that the retrieved data has not been tampered with. In addition, Chen et al. [9] proposed an authenticated Merkle hash tree to verify the search result. Although significant progress has been made by the existing constructions [8], [9], the verifiable property comes at the high cost of extra storage and computation. There is still room for solutions that are more practical.

Recently, with increasing demand for users (e.g., physicians), previous SSE constructions, providing a client either full access to the data or no access, expose their short term. It is desirable to design a fine-grained access control mechanism to enable data owners to selectively grant grant clients access to their data. To achieve this goal, Han et al. [10] proposed to apply attribute-based encryption [11] to solve this problem and provided a general solution in the context of public-key keyword search scenarios. With this design, only a keyword search request that matches the predefined access structure can retrieve the target document. The above searchable encryption schemes rely on public key encryption, which is inefficient compared with symmetric encryption. Moreover, none of them are suitable for use with dynamic dynamic access structures since the access structure is associated with either a key or ciphertext. Any change in the access structure may result in all of the ciphertext or keys being renewed.

Furthermore, all the mentioned works failed to allow a client to search the data from multiple data owners, where each data owner encrypts their data with a unique key. The existing SSE schemes only support a client to search over a single data owner [5], [12], [13]. However, in our scenario, a client needs to search for data outsourced by multiple independent data owners. For example, to identify a medical treatment for a cancer patient, a physician may need to analyze medical data from thousands of contributors (e.g., patients). An intuitive solution for this scenario is to deploy existing schemes [5], [12], [13] for each data owner, where each data owner manages their outsourced data independently. For each service request, the physician generates a specialized (authorized) request for each owner's data and sends it to the CSP. This results in a high volume of requests for a single query. Another approach is to adopt the recent *Multi-Writer Encrypted Database* [14], which allows multiple data owners to store data and allows clients to search across data owners. However, this scheme presumes a fixed and predefined number of data owners who share a master secret key and public key. This assumption does not hold in our scenario, where data owners operate independently and have the freedom to join or leave the system at any time. This is a critical feature that allows a flexible number of data owners in the system, who have complete control of their own data, and allows a legitimate client to search across parties.

Moreover, in the above works, no accountability and verifiability mechanisms are available. If a client abuses his/her right (e.g., by sharing search tokens with unauthorized clients), there is no way to identify the client. In addition, the search result returned by the CSP may be incorrect due to deliberate modifications or unpredicted data loss. For example, instead of returning target documents, the CSP may send the client an advertisement. Moreover, the hardware and software are not stable, which may result in data loss due to hardware or software failure. Additionally, packet loss during communication is common.

To address the problem of user accountability, Li et al. [15] proposed privacy-aware attribute-based encryption with user accountability and applied it to the file storage system [16]. However, their scheme relied on attribute-based encryption, which is different from SSE because SSE uses symmetric encryption. To address the problem of the integrity of the search result, Chai and Gong [8] constructed a verifiable data structure based on the hash function and Soleimanian et al. [17] proposed a scheme by delegating a third party to conduct the verification. However, these approaches require either a third party or a large amount of extra storage and computation to ensure verifiability.

Finally, with the availability of GPUs and TPUs, the requirement of parallelism is essential. Although efficient SSE constructions are available [6], [9], existing solutions are still highly sequential.

Herein, we propose a framework that is, to the best of our knowledge, the first to to tackle all the above challenges. In the proposed scheme, each data owner encrypts its own dataset (with its unique key) and outsources the storage and processing tasks for search operations to the CSP. For privacy, all the data are encrypted using a standard symmetric encryption algorithm and an index is established for privacy-preserving search. The proposed indexing mechanism provides not only privacy but also the ability to search over the datasets of several data owners. Each data owner encrypts its own dataset independently, and the searchability of ciphertext is enabled across all the data owners through this indexing mechanism.

To support the search result verification, we propose a novel construction of message authentication code (MAC), in which all the associated data is required to compute MAC, and then exclusive or (XOR) operation is conducted between all the computed MACs. Finally, the result is encrypted and embedded into the index. In addition, to enable fine-grained access control and user accountability, we propose a keyword-based accountable token mechanism. Each user needs to obtain a token for a specific keyword before they are able to launch a query. The token is utilized to transform a user's query to support ciphertext search. Moreover, with the construction of a token, the identity of the client is properly embedded into the token without privacy violation. Because of the way the token is constructed, it can be used to trace back to the client. Finally, to enhance the search efficiency, we propose a parallelism mechanism, where the inverted index is partitioned into segments and all of them are executed in parallel. In summary, our paper makes the following contributions:

1) To the best of our knowledge, we are the first to tackle the privacy-preserving keyword search problem in a multi-tenancy cloud where each tenant operates independently. We propose a privacy-preserving data outsourcing framework that enables efficient secure retrieval of the outsourced data from the CSP with dynamic updates, fine-grained access control, verifiable search results, user accountability, and parallelism. For

our proposed scheme, we design a privacy-preserving framework that supports backward indexing with verifiable yet accountable keyword searchable encryption (VAKSE). Moreover, due to the design of the index and token generation mechanism, we ensure fine-grained access control, the ability to search over several parties, and user accountability. Furthermore, we propose a novel approach to constructing MAC to realize search result verification. Finally, to enhance the search efficiency, we propose PVAKSE to parallelize VAKSE.

2) We formally analyze the security of the proposed schemes and demonstrate that our proposed scheme can attain our design goals. In addition, we conduct comprehensive experiments to evaluate the proposed scheme, and our results show the effectiveness of VAKSE and PVAKSE.

The remainder of this paper is organized as follows. In Section II, we introduce related work. We then review the preliminaries in Section III. After that, we present the system model, threat model, and design goals in Section IV, which is followed by the proposed scheme in Section V. Security and performance analyses are presented in Section VI and Section VII, respectively. In Section VIII, we discuss the approach to mitigating the storage requirement of data owners, detection of denial of query service, and real-world use cases and deployment scenarios. Finally, we draw conclusions in Section IX.

II. Related Work

Due to the boom in cloud computing and increasing privacy concerns, keyword search over encrypted data has been a hot topic over the past twenty years. In this section, we investigate SSE and ABE to present relevant work and position our study.

A. Symmetric Searchable Encryption

Song et al. [5] came up with the problem of keyword search for encrypted data and proposed the first symmetric searchable encryption (SSE) scheme. In their scheme, each keyword is encrypted in two layers. The first layer is implemented via pseudo-random permutation and used to hide the keyword. The second layer is realized by the pseudo-random function and used to support encrypted keyword searches. To improve search efficiency, Goh et al. [6] proposed a secure index against adaptive chosen keyword attack (IND-CKA) based on pseudo-random functions and bloom filters. Thereafter, Curtmola et al. [7] formalized the security definition of SSE and proposed two constructions that corresponded to nonadaptive semantic security and adaptive semantic security by assuming the the existence of a pseudo-random permutation and an encryption algorithm that provides security against chosen plaintext attacks. Following these definitions, various SSE schemes have been proposed to enrich queries and enhance search efficiency, such as ranked keyword search [9], [18], fuzzy keyword search [19], similarity search [20], semantic search [21], and parallel search [22].

All the above works assume that the CSP is honest but curious. This means that the CSP honestly executes the intended protocol. However, this assumption is not always valid in real-world scenarios since cloud services are susceptible to external attacks, internal misconfigurations, software glitches, and even insider threats [23], [24]. All these elements can lead a CSP to diverge from the intended protocol and to operate outside the boundaries of the honest-but-curious model. In our work, we address the potential risk that a compromised CSP is not detected by users. Specifically, we consider the scenario where the server continues to provide its service but deviates from the intended protocol by operating beyond the boundaries of the honest-but-curious model.

B. Verifiable Symmetric Searchable Encryption

To enhance security, stronger models against threats are considered. To prevent the honest-but-curious cloud server from returning partial search results, Chai and Gong [8] proposed the first verifiable SSE scheme based on hash functions and block ciphers, where the search result included not only retrieved documents but also proof of the search process. To formalize a stronger threat model for a malicious cloud server, Kurosawa and Ohtaki [25] introduced the definition of universally composable (UC) secure SSE against non-adaptive adversaries and constructed a verifiable SSE based on pseudo-random permutation and unforgeable Message Authentication Code (MAC) to achieve UC security. Although UC security offers higher security, its construction requires a powerful client to conduct verification. To remove the heavy verification load of clients, Soleimanian and Khazaei [17] presented a public VSSE scheme based on pseudo-random functions, one-way functions, digital signatures and the DDH assumption, which delegated a third party to accomplish the verification and achieved $\mathcal{L}$-adaptive security under the DDH assumption. To support multi-user settings, Zhu et al. [26] proposed a generic VSSE scheme based on Merkle Patricia Trie (MPT) and pseudo-random functions, which decoupled the proof from SSE and achieved $\mathcal{L}$-adaptively-secure.

However, all the above approaches require either a third party or a large amount of extra storage and computation resources to ensure verifiability.

C. Accountable Attribute-Based Encryption

In the above schemes, the client is trusted. In reality, dishonest users may attempt to access data without authorization. Even worse, some users may give away some of their original or transformed keys such that nobody can tell who has distributed these keys. The first problem is called unauthorized access. The second problem is called key abuse. The first problem can be prevented by fine-grained access control, and the second problem can be discouraged by user accountability.

The issue of fine-grained access control and user accountability has been widely discussed in the area of ABE [15], [16], [27], [28]. Yu et al. [28] considered how to defend against the key abuse problem in KP-ABE schemes. Hinek et al. [27] introduced a trusted party to support decryption operations to prevent the key abuse problem. Li et al. [15] embedded user-specific information into the attribute private key issued to that user to realize user accountability and applied the construction into the file storage system [16].

Although fine-grained access control and user accountability can be achieved using the above ABE schemes, they are inefficient compared with symmetric encryption. Moreover, these ABE schemes can not support ciphertext searchability, which is provided by SSE.

D. Attribute-Based Keyword Search

To enable fine-grained access control and ciphertext search, Zheng et al. [29] proposed the first attribute-based keyword search (ABKS) scheme that combined public key encryption with keyword search (PEKS) [30] with CPABE. Instead of CPABE, Liang and Susilo [31] constructed an ABKS scheme using KPABE. To enrich the query, Huang et al. [32] proposed an ABKS scheme with ranked keyword search. Considering the consistency of the secret key of ABKS, Ge et al. [33] designed a flexible and secure keyword search scheme for cloud-based data sharing.

The above ABKS schemes have the same shortcomings as existing ABE-based schemes: they are less efficient than SSE. Moreover, none of them are suitable for dynamic access structures since the access structure is associated with either a key or ciphertext. Any change in the access structure may result in all the ciphertext or keys being renewed.

E. Technical Challenges of the Proposal

The proposed scheme initially necessitates the acceptance of multiple data owners, each of whom possesses no prior shared knowledge. The first technical challenge lies in querying across data owners, each of whom independently establishes their access policies. Verifying the retrieved data, regardless of its source among the data owners, is the second technical challenge. The third technical challenge is identifying users who misuse or abuse their rights, regardless of the data owner by whom those rights were granted. The fourth technical challenge involves designing a privacy-preserving keyword search mechanism that incorporates parallelism while satisfying the aforementioned requirements.

III. Preliminaries

In this section, we briefly introduce the concepts of the discrete logarithm (DL) problem and the symmetric bilinear group. Both are a foundation for the proposed schemes.

A. Discrete Logarithm (DL) Problem

The discrete logarithm problem is defined as follows: Given a group G, a generator g of the group, and an element $h = g^x$ of G, the problem is to find the DL x to the base g of h in the group G.

The problem is hard if the order of G is a prime number p that equals $2q + 1$, where q is a large prime number. This guarantees that $p - 1 = 2q$ has a large prime factor and the Pohlig-Hellman algorithm [34] cannot solve the DL problem easily.

B. Symmetric Bilinear Group

Let G be a multiplicative cyclic group of prime order p, and let g be the generator of the group G. Let e: $G \times G \leftarrow G_T$ be a function that maps two elements from G to a target group G_T of prime order p. The tuple (G, G_T, p, e) is a symmetric bilinear group if the following properties hold:

1) group operations in G and G_T are efficiently computable.
2) mapping from G to G_T is efficiently computable.
3) mapping e is nondegenerate: $e(g, g) \neq 1$.
4) mapping e is bilinear: $e(g^a, g^b) = e(g, g)^{ab}$.

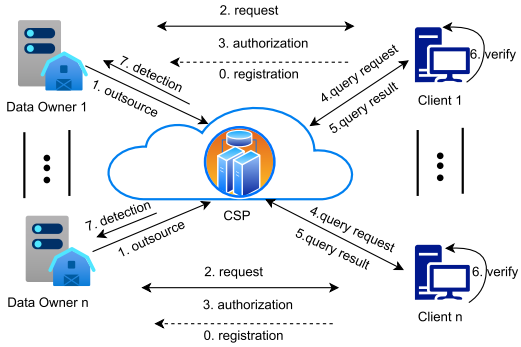


Fig. 1. System model. In the model, each data owner encrypts his/her data independently and outsources it to the CSP. To access the data, a client must register with the system and request permission from the data owner. After the authorization is obtained, the client can raise a query to retrieve outsourced data. Upon receiving the retrieved data, the client can verify the search result. If the CSP detects abnormal queries, it can raise a user accountability request to identify the client.

IV. Models and Design Goals

In this section, we first formalize our system model and threat model and then identify our design goal.

A. System Model

In our system model, which is shown in Fig. 1, we consider a typical outsourcing model, that consists of three entities: data owner, Cloud Service Provider (CSP), and client. The data owner has a document dataset D. Each document consists of document identity id and words. Since the size of the dataset may be large, a data owner may not be able to store it or compute with it. The data owner tends to outsource the dataset to the CSP. Meanwhile, to ensure data privacy, the data owner may encrypt the data before outsourcing it to the CSP. The CSP supplies resources for storage and powerful computing. It receives the outsourced dataset from the data owner and processes queries from the client. Before a client is allowed to use the system, the client needs to register with the data owner and obtain permission to be enrolled in the system. After the client obtains approval, the client is able to raise queries and receive the result returned from the CSP.

We emphasize that our system supports multiple data owners, unlike an existing work that focuses on only one data owner [9]. Thus, our system supports various clients. For example, client 1 may be granted access to the data from data owners 1 and 2, while client 2 may be granted access to the data from data owners 1 and 3. Moreover, our system supports search result verification. Our system is capable of detecting the return of an incorrect search result. Furthermore, it enables user accountability and allows the identification of any legitimate client that abuses his/her rights.

B. Threat Model

In our threat model, the data owner is considered to be fully trusted because the data owner owns the dataset. However, the CSP is considered malicious, which indicates that the CSP may not follow the proposed scheme honestly, and may behave arbitrarily.

Herein, we mainly consider the following three threats from a malicious CSP:

1) Since the CSP stores the encrypted data, the CSP has the opportunity to extract information and infer the content of the encrypted data.
2) Due to self-interest, the CSP may select random documents as the search result or may return documents that do not match the query as the search result.
3) Due to unexpected errors, the CSP may execute the search operation incorrectly.

We assume that all legitimate clients strictly follow the protocol, although they may share the assigned token with illegitimate clients. Herein, we only cover user accountability. For the user misbehavior detection, readers can refer to [35] and [36].

C. Design Goal

In this work, our goal is to present a privacy-preserving keyword search scheme that has multiple independent data owners. In particular, we aim to fulfill the following objectives.

Privacy Preservation: Our proposed scheme aims to protect the privacy of outsourced data and achieve adaptive semantic security (which is formally defined in Section VI), in which an adversary cannot learn any additional information about the plaintext other than what is already revealed by the ciphertext.

Search Over Several Data Owners: The proposed scheme is supposed to support multiple data owners that independently outsource data to the CSP. Meanwhile, a client can perform data search over several data owners.

Fine-Grained Access Control: In the proposed scheme, the granularity of the access control should be the keyword. For each keyword, a client is required to obtain permission before they are able to generate a query to retrieve documents that contain the target keyword.

Verifiable Search Result: The proposed scheme needs to offer a mechanism to verify the correctness and completeness of the search result. Correctness guarantees that the retrieved result corresponds to the target keyword, and completeness ensures that no adversary has tampered with the retrieved data.

User Accountability: If a legitimate client abuses the assigned token, the proposed scheme is supposed to identify that client. Thus, user accountability acts as a targeted deterrent against token abuse.

V. Proposed Scheme

In this section, we first construct a VAKSE scheme. Then, we present the PVAKSE scheme, which is based on VAKSE.

Before we present the details of different constructions, we present frequently used notation in Table I.

TABLE I
NOTATIONS

λ	security parameter
H	hash function
k_i	secret key of i-th data owner for symmetric encryption/decryption and binding keywords
pk_i	public key of i-th data owner
pk_c	public key of the client
w	a keyword
c_w	ciphertext of the keyword w
α_w	hash result of a keyword w
$r_{c,w}$	a random number selected by a client to obtain authorization
d	content of a document
id	identifier of a document
L	list of document identifiers
D_i	document sets of i-th data owner
I_i	inverted index of i-th data owner
n	total number of keywords
$\mathcal{EI}_i$	encrypted inverted index of i-th data owner
$\mathcal{ED}_i$	encrypted dataset of i-th data owner
$tk_{i,w}$	token generated by i-th data owner for keyword w
D_c	a dictionary that stores used numbers for keyword authorization
Φ	set of encrypted document identifiers
Ψ	set of encrypted documents
S_{ids}	set of decrypted document identifiers
S_{ds}	set of decrypted documents

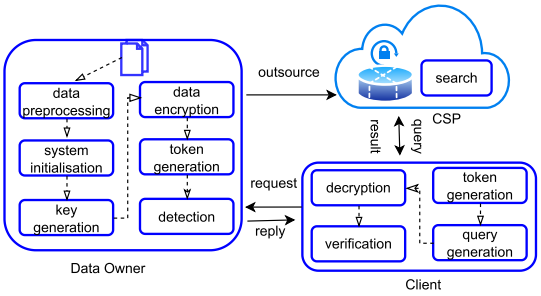


Fig. 2. Overview of the VAKSE scheme.

A. Construction of the VAKSE Scheme

In this section, we first present the overview of each component of the VAKSE scheme and then explain each component in detail.

1) Overview: As shown in Fig. 2, the proposed VAKSE scheme consists of ten modules: data preprocessing, system initialization, key generation, data encryption, token generation, query generation, search, decryption, verification, and detection. The function of each module is detailed below.

Data Preprocessing: In this module, as shown in Fig. 3, all the keywords are first extracted from documents and then form an inverted index, where each entry of the inverted index consists of a keyword and a list of document identifiers.

System Initialization: This is the initial procedure of the system. This module accepts the input of system parameters and produces the required parameters for the following procedures. For brief and clear presentation, the **key generation** module is incorporated into this component, which outputs required keys.

Data Encryption: To ensure data privacy, the data owner must encrypt the data before outsourcing the data to the CSP. This module implements all the encryption algorithms that are required to perform data encryption.

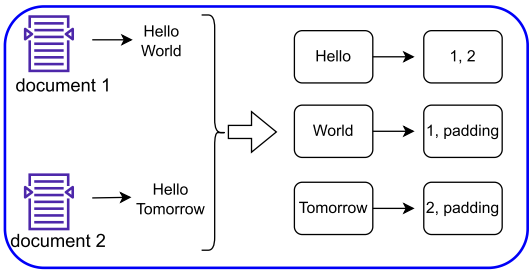


Fig. 3. Data preprocessing.

Token Generation: Before a client launches a query to search target documents, the client needs to get a token first from the data owner. This module implements the algorithms to deal with the token request from the client and generate tokens for the client.

Query Generation: Before a client raises a query to search target documents, the client must obtain a token from the data owner. This module implements the algorithms that execute the token request from the client and generate tokens for the client.

Search: When provided with the encrypted data, token, and query, the CSP conducts a search operation to find the documents that match the requirement and return them to the client. The search module is implemented to conduct a privacy-preserving search.

Decryption: Upon receiving the retrieved ciphertext, the client uses the decryption algorithm to decrypt the ciphertext. This module is designed to decrypt either the identities of the encrypted document or the documents themselves.

Verification: When provided with a search result, the client is capable of raising a verification operation. The objective is to check the correctness and completeness of the returned result. The verification module implements the mechanism to verify the search result.

Detection: When provided with the abused token and assistant information, the data owner owner can learn the identity of the client who misuses the allocated token. The detection module prevents token abuse and realizes user accountability.

2) Components of VAKSE: In this section, we present the details of each component.

Data Preprocessing: The i-th data owner collects all the documents D_i, extracts all the keywords from those documents, and uses them to create a dictionary. Thereafter, as shown in Fig. 3, the data owner creates an inverted index I_i to store the (key, value) pairs. The key is used to store the keyword and the value is utilized to store the identities of documents that contain the keyword. To prevent the adversary from learning the size of the document identities, we pad zeros to the shortlist so that all the keywords correspond to an equal number of document identities. As shown in Fig. 3, there are three key-value pairs: {Hello, (1,2)}, {World, (1, padding)}, {Tomorrow, (2, padding)}.

System Initialization: Table II shows all the parameters generated during the initialization phase. For initialization, the system first selects a symmetric encryption mechanism (e.g., AES), which consists of encryption algorithm $\mathcal{SE}nc$ and decryption algorithm $\mathcal{SD}ec$. The application of a symmetric encryption algorithm results in greater efficiency than the public-key encryption algorithms that were deployed in [10] and [29]. Moreover, it outputs a bilinear mapping $e: G \times G \rightarrow G_T$, where the order of the groups is p and the group generator of G is g, along with the hash function $H: \{0,1\}^* \rightarrow \mathbb{Z}_p$. Next, the system generates a pair with the private key and public key for each data owner and client. For the i-th data owner, the system first selects the private key $sk_i \leftarrow \mathbb{Z}_p^*$ and then computes public key $pk_i \leftarrow g^{sk_i}$. For each client, the system similarly generates sk_c and pk_c. Additionally, it generates a secret key for each data owner, which is used to encrypt documents. For example, $k_i \leftarrow \{0,1\}^\lambda$ is generated for i-th data owner.

TABLE II
PARAMETERS USED IN INITIALIZATION

Public info	$\lambda, G, G_T, g, p, e, H$
Info of data owner	sk_i, pk_i, k_i
Info of client	sk_c, pk_c
Info that data owner gives to legitimate client	k_i

Algorithm 1 Data Encryption ($\mathcal{E}nc$)

Input: (sk_i, k_i, I_i, D_i)
Output: $\mathcal{EI}_i, \mathcal{ED}_i$

for all $(w, L) \in I_i$ **do**
  $r \leftarrow \mathbb{Z}_p^*$
  $c_0 \leftarrow g^r$
  $\alpha_w \leftarrow H(w)$
  $c_1 \leftarrow g^{-r/(sk_i - \alpha_w)}$
  $c_w \leftarrow (c_0, c_1)$
  $k_{i,w} \leftarrow F(k_i, w)$
  $c_{ids} \leftarrow \mathcal{SE}nc(k_{i,w}, L)$
  $\mathcal{EI}_i[c_w] \leftarrow c_{ids}$
for all $(id, d) \leftarrow D_i$ **do**
  $\mathcal{ED}_i[id] \leftarrow \mathcal{SE}nc(k_i, d)$
return $\mathcal{EI}_i, \mathcal{ED}_i$

Data Encryption: The data encryption step can be divided into two parts. The first step is the inverted index encryption and the second step is the document encryption. As shown in Algorithm 1, index encryption encrypts all the entries of I_i. Each entry is a pair (w, L), where w is the keyword and L is a list of document identities, e.g., 1 and 2 in Fig 3. For each keyword w, the data owner computes $c_0 \leftarrow g^r$ as the first component of the encrypted keyword c_w, where r is randomly selected from $\mathbb{Z}_p^*$. The data owner then computes $c_1 \leftarrow g^{-r/(sk_i - \alpha_w)}$ as the second component of the encrypted keyword c_w, where α_w is the hash result of keyword w. For the encryption of document identities, the data owner executes the symmetric encryption algorithm $\mathcal{SE}nc$ to encrypt the list L of document identities, where the secret key $k_{i,w}$ is generated by calling a pseudorandom function F with inputs w and k_i, where k_i is the secret key of the data owner that is generated during the initialization. Here, the goal of using the pseudorandom function F is to bind the keyword w and k_i. After the index encryption, all the documents D_i are encrypted using the symmetric encryption algorithm

TABLE III
SYSTEM CORE APIS

API Call	Input	Purpose of the call	Output
		Data Owner	
$\mathcal{E}nc$	sk_i, k_i, I_i, D_i	Encrypt inverted index and dataset	$\mathcal{E}I_i, \mathcal{E}D_i$
$\mathcal{T}okenGen$	$sk_i, pk_i, pk_c, \alpha_w, k_i$	1.Receive parameters from a client 2.Generate and send a token to the client	D_c
		Client	
$\mathcal{T}okenGen$	w, pk_c	1. Generate and send required parameters to a data owner for token generation 2. Receive a token from the data owner	$r_{c,w}, tk_{i,w}$
$\mathcal{Q}ueryGen$	$w, pk_c, r_{c,w}$	Generate a query with target keyword	$q_{c,w}$
$\mathcal{D}ecrypt\mathcal{DI}$	Φ, k_i, w	Decrypt the encrypted document identities	S_{ids}
$\mathcal{D}ecrypt$	Ψ, k_i	Decrypt the encrypted documents	S_{ds}
$\mathcal{V}erify$	Φ, w, k_i, Ψ	Verify the search result	True/False
$\mathcal{D}etection$	$D_c, tk_{i,w}, w$	Identify the client from its abused token	pk_c
		CSP	
$\mathcal{S}earch$	$\mathcal{E}I_i, tk_{i,w}, q_{c,w}$	Search the target encrypted document identities	Φ

Algorithm 2 Token Generation ($\mathcal{T}okenGen$)

Input: $(pk_i,\ pk_c,\ sk_i,\ w,\ k_i)$
Output: $r_{c,w},\ tk_{i,w},\ D_c$

Client

$\alpha_w \leftarrow H(w)$
$r_{c,w} \leftarrow \mathbb{Z}_p$
send $r_{c,w}$, α_w and pk_c to the data owner
return $r_{c,w}$

Data Owner

$tk_{i,w} \leftarrow \frac{pk_i}{pk_c}^{\frac{-r_{c,w}}{sk_i-\alpha_w}}$
send $tk_{i,w}$ and k_i to the client
$D_c[\alpha_w] \leftarrow r_{c,w}$
return D_c

Algorithm 3 Query Generation ($\mathcal{Q}ueryGen$)

Input: $(\ w,\ pk_c,\ r_{c,w})$
Output: $q_{c,w}$

$\alpha_w \leftarrow H(w)$
compute $g^{r_{c,w}}$ and $g^{-r_{c,w}\alpha_w} pk_c^{r_{c,w}}$
$q_{c,w} \leftarrow (g^{r_{c,w}}, g^{-r_{c,w}\alpha_w} pk_c^{r_{c,w}})$
return $q_{c,w}$

Algorithm 4 Keyword Search ($\mathcal{S}earch$)

Input: $(\mathcal{EI}_i,\ tk_{i,w},\ q_{c,w})$
Output: Φ

$\Phi \leftarrow \phi$
for all $(c_w, c_{ids}) \in \mathcal{EI}_i$ **do**
  for all $(c_0, c_1) \in c_w$ **do**
    $(g^{r_{c,w}}, g^{-r_{c,w}\alpha_w} pk_c^{r_{c,w}}) \leftarrow q_{c,w}$
    $(q_0, q_1) \leftarrow (g^{r_{c,w}}, g^{-r_{c,w}\alpha_w} pk_c^{r_{c,w}})$
    if $e(c_0, q_0)e(c_0, tk_{i,w})e(c_1, q_1) = 1$ **then**
      $\Phi \leftarrow \Phi \cup c_{ids}$
return Φ

$\mathcal{SE}nc$ with the secret key k_i. Finally, the module outputs the encrypted inverted index $\mathcal{EI}_i$ and dataset $\mathcal{ED}_i$.

Token Generation: To search a document with specific keywords, a client must obtain tokens for these keywords from the data owner. First, the client selects the target keyword w, and computes the hash result α_w of the keyword. Next, a random number $r_{c,w} \leftarrow \mathbb{Z}_p^*$ is selected. Finally, the client sends $r_{c,w}$, α_w, and its public key pk_c to the data owner to obtain the authorization of keyword search. Upon receiving the request, the data owner decides whether to approve the request. If the data owner approves the request, the data owner computes the value of $tk_{i,w} \leftarrow \frac{pk_i}{pk_c}^{-r_{c,w}/(sk_i-\alpha_w)}$. After the computation, the data owner sends $tk_{i,w}$ and k_i to the client. In addition, $r_{c,w}$ is stored into the local dictionary $D_c[\alpha_w]$. In Section VIII, we discuss the requirement of the dictionary D_c and propose an approach for mitigating the storage requirements.

Query Generation: To generate the query, as shown in Algorithm 3, the client first invokes the hash function with the selected keyword w as input and outputs α_w. Next, the client computes $g^{r_{c,w}}$ as the first component of the query $q_{c,w}$ and $g^{-r_{c,w}\alpha_w} pk_c^{r_{c,w}}$ as the second component of the query $q_{c,w}$. Note that both the keyword w and random number $r_{c,w}$ should be consistent with token generation for documents containing the target keyword to be retrieved.

Search: The search operation is conducted by the CSP. Given the encrypted index $\mathcal{EI}_i$, token $tk_{i,w}$, and query $q_{c,w}$, as shown in Algorithm 4, the CSP needs to check whether the query matches the entry of the encrypted inverted index. Specifically, for each pair $(c_w, c_{ids}) \in \mathcal{EI}_i$, the CSP computes $e(c_0, q_0)e(c_0, tk_{i,w})e(c_1, q_1)$ to check whether the result is equal to 1, where c_0 and c_1 are the two components of c_w and q_0 and q_1 are the two components of query $q_{c,w}$. If the computed result is 1, it means that the query matches the index. The ciphertext c_{ids} of the corresponding document identities is thene stored in the set Φ. Finally, the set Φ is sent to the client.

Decrypt: After the retrieved result Φ is received, as shown in Algorithm 5, the client invokes the symmetric decryption algorithm $\mathcal{SD}ec$ to decrypt the ciphertext and obtain the plaintext S_{ids} of the matching document identities. The client sends the document identities to the CSP to retrieve all corresponding encrypted documents Ψ. The client uses Algorithm 6 to decrypt the encrypted documents when they are received and to obtain the plaintext S_{ds} of the documents.

Detection: To support *user accountability*, as shown in Algorithm 7, the data owner maintains a dictionary D_c, in which the hash result of keyword w is associated with the

Algorithm 5 Decrypt Document Identities ($\mathcal{D}ecrypt\mathcal{DI}$)

Input: $(\Phi,\ k_i,\ w)$
Output: S_{ids}
$k_{i,w} \leftarrow F(k_i, w)$
$S_{ids} \leftarrow \phi$
for all $c_{ids} \in \Phi$ **do**
  $L \leftarrow \mathcal{SD}ec(k_{i,w}, c_{ids})$
  $S_{ids} \leftarrow S_{ids} \cup L$
return S_{ids}

Algorithm 6 Decrypt Documents ($\mathcal{D}ecrypt$)

Input: $(\Psi,\ k_i)$
Output: S_{ds}
$S_{ds} \leftarrow \phi$
for all $(id, ed) \in \Psi$ **do**
  $d \leftarrow \mathcal{SD}ec(k_i, ed)$
  $S_{ds} \leftarrow S_{ds} \cup (id, d)$
return S_{ds}

Algorithm 7 User Accountability ($\mathcal{D}etection$)

Input: $(D_c,\ tk_{i,w},\ w)$
Output: pk_c
$\alpha_w \leftarrow H(w)$
$r_{c,w} \leftarrow D_c[\alpha_w]$
$pk_c \leftarrow pk_i \cdot tk_{i,w}^{\frac{sk_i - \alpha_w}{r_{c,w}}}$
return pk_c

value $r_{c,w}$ sent from the client. If token abuse occurs, *i.e.*, if tokens are illegally shared between clients, the data owner computes $pk_i \cdot tk_{i,w}^{\frac{sk_i-\alpha_w}{r_{c,w}}}$ to identify client pk_c.

B. Search Result Verification

The verification of the search result includes the correctness and completeness of the search result, as shown in Algorithm 8.

The correctness of the search result requires the correctness of the keyword search process and document retrieval process. The correctness of the keyword search process is verified through the decryption process because generating the secret key for decrypting the retrieved document identities requires the target keyword. If an incorrect keyword is used, the generated secret key cannot be used to decrypt the encrypted document identities. The correctness of the retrieved documents is evaluated by comparing the retrieved document identities with the document identities held by the client.

The completeness of the search result also includes two parts. The first part is the completeness of the retrieved document identities and the second part is the completeness of the retrieved documents. The data encryption algorithm is modified to guarantee the completeness of both parts, as shown in Algorithm 9. The algorithm first generates the $\mathcal{MAC}$ of the document identity list. Then, for each document that belongs to the document list, it also generates the $\mathcal{MAC}_d$ and conducts an XOR operation, which is denoted by $\oplus$, with the $\mathcal{MAC}$ of the

Algorithm 8 Search Result Verification ($\mathcal{V}erify$)

Input: $(\Phi,\ w,\ k_i,\ \Psi)$
Output: True/False
Correctness
$S_{ids} \leftarrow \mathcal{D}ecrypt\mathcal{DI}(\Phi, k_i, w)$
$(L, \mathcal{MAC}) \leftarrow S_{ids}$
$S_{ds} \leftarrow \mathcal{D}ecrypt(\Psi, k_i)$
if the identities of S_{ds} are same with L **then**
  return True
return False
Completeness
$k_{i,w} \leftarrow F(k_i, w)$
$\mathcal{MAC}^* \leftarrow \mathcal{HMAC}(k_{i,w}, L)$
for all $(id, d) \in S_{ds}$ **do**
  $\mathcal{MAC}_d^* \leftarrow \mathcal{HMAC}(k_{i,w}, d)$
  $\mathcal{MAC}^* \leftarrow \mathcal{MAC}^* \oplus \mathcal{MAC}_d^*$
if $\mathcal{MAC}^* = \mathcal{MAC}$ **then**
  return True
return False

Algorithm 9 Data Encryption With MAC ($\mathcal{E}nc$)

Input: $(sk_i,\ k_i,\ I_i,\ D_i)$
Output: $\mathcal{EI}_i,\ \mathcal{ED}_i$
for all $(w, L) \in I_i$ **do**
  $c_w \leftarrow \mathcal{ASMT}.\mathcal{E}nc(sk_i, w)$
  $k_{i,w} \leftarrow F(k_i, w)$
  $\mathcal{MAC} \leftarrow \mathcal{HMAC}(k_{i,w}, L)$
  for all $id \in L$ **do**
    $\mathcal{MAC}_d \leftarrow \mathcal{HMAC}(k_{i,w}, D_i(id))$
    $\mathcal{MAC} \leftarrow \mathcal{MAC} \oplus \mathcal{MAC}_d$
  $c_{ids} \leftarrow \mathcal{SE}(k_{i,w}, L||\mathcal{MAC})$
  $\mathcal{EI}_i[c_w] \leftarrow c_{ids}$
for all $d \leftarrow D_i$ **do**
  $\mathcal{ED}_i \leftarrow \mathcal{SE}nc(k_i, d)$
return $\mathcal{EI}_i,\ \mathcal{ED}_i$

document identity list. The XOR result is stored in the $\mathcal{MAC}$. Finally, $\mathcal{MAC}$ is concatenated with the document identity list and encrypted by the symmetric encryption algorithm. During the verification phase, the client verifies the completeness of the document identity list and all the retrieved documents based on the retrieved $\mathcal{MAC}$ value. In particular, the client first calls the pseudo-random function F with the secret key k_i and keyword w as input and generates the key $k_{i,w}$. Then, the secret key $k_{i,w}$ and list L of the document identities are used to generate $\mathcal{MAC}^*$. Next, for each retrieved document d, the client computes $\mathcal{MAC}_d^*$ and runs the XOR operation using $\mathcal{MAC}^*$. If the $\mathcal{MAC}^*$ is equal to the retrieved $\mathcal{MAC}$ value, the retrieved documents and document identity list are complete.

C. PVAKSE: VAKSE Based on Inverted Index Partition

We observe that the search efficiency can be enhanced in two ways based on the construction of VAKSE. The first way

is to parallelize the index processing of each data owner. Since each data owner outsources its data independently, the query can be processed in parallel. The another way is to reduce the time consumption of searching over the inverted index of each data owner. Inside the inverted index, the entries are independent and can also be dealt with in parallel. Considering both the ways to enhance efficiency, the goal is to parallelize the inverted index.

One way to fulfill the above goal is to divide the inverted index into entries. Parallelizing the search of the entries requires the same number of threads as the total number of entries in the inverted index. For example, the Enron dataset [37] has more than half a million entries. However, Amazon AWS, which is the most prevalent cloud service, provides an instance with a maximum of 64 CPU cores [38], which means that it needs approximately 10,000 most advanced instances. This example shows that it is impractical to divide the inverted index into entries. To reduce the demand for instances from the CSP and satisfy the constraint on the maximum number of concurrent requests from a client, we divide the inverted index into the same number of partitions with the maximum number of concurrently supported requests. This results in a balance between resource requirements and efficiency.

VI. VAKSE Security Analysis

According to our design goal for privacy preservation, we demonstrate that the proposed schemes protect the privacy of outsourced data and queries. Because PVAKSE uses the same security mechanism as VAKSE, we conduct a security analysis of only VAKSE. Before we present the detailed security analysis, we first introduce four auxiliary notations following [39].

Definition 1 (History): Let Δ be a dictionary and D be a document collection containing terms from Δ. A q-query history over D is the tuple $H = (D, \Delta)$ that includes the collection of documents D, and a vector of q-set of keywords $\Delta = (w_1, \cdots, w_q)$, where w_j $(1 \leq j \leq q)$ is the keyword used in j-th query.

Definition 2 (Search Path Pattern): Let Δ be a dictionary; let D be a document collection set over Δ; and let I be the inverted index of D. The search path pattern induced by the q-query history $H = (D, \Delta)$ is the tuple $\sigma(H) = (I(w_1), \cdots, I(w_q))$.

Definition 3 (Access Pattern): Let Δ be a dictionary; let D be a document collection set over Δ. The access pattern induced by the q-query history $H = (D, \Delta)$ is the tuple $\beta(H) = (D(w_1), \cdots, D(w_q))$.

Definition 4 (Trace): Let Δ be a dictionary and D be a document collection set over Δ. The trace induced by the q-query history H is the sequence $\gamma(H) = (|D_1|, \cdots, |D_n|, |I_1|, \cdots, |I_l|, \sigma(H), \beta(H))$, where $|D_j|$ $(1 \leq j \leq n)$ represents the length of the j-th document in D, and $|I_j|$ $(1 \leq j \leq n)$ represents the length of the j-th entry in I.

We now present our simulation-based definition that the view of an adversary generated from an adaptively chosen history can be simulated given only the trace, which is explained in Definition 5. Based on the definition, Theorem 1 is proposed and followed by its proof.

Definition 5 (Adaptive Semantic Security): Let $\Pi = (Setup, Enc, TokenGen, QueryGen, Search, Decrypt, Verify, Detection)$ be the proposed scheme; let $\lambda \in \mathbb{N}$ be the security parameter; let $\mathcal{A} = (\mathcal{A}_0, \cdots, \mathcal{A}_q)$ be a PPT adversary such that $q \in \mathbb{N}$; and let $\mathcal{S} = (\mathcal{S}_0, \cdots, \mathcal{S}_q)$ be a PPT simulator. Consider the following probabilistic experiments $\boldsymbol{Real}_{\Pi,\mathcal{A}}(\lambda)$, and $\boldsymbol{Sim}_{\Pi,\mathcal{A},\mathcal{S}}(\lambda)$.

$\boldsymbol{Real}_{\Pi,\mathcal{A}}(\lambda)$
$(sk_i, pk_i, sk_c, pk_c, e, p, k_i) \leftarrow Setup(\lambda)$
$(I_i, D_i, st_{\mathcal{A}}) \leftarrow \mathcal{A}_0(1^\lambda)$
$(\mathcal{EI}_i, \mathcal{ED}_i) \leftarrow \mathcal{E}nc(sk_i, k_i, I_i, D_i)$
$(w_1, st_{\mathcal{A}}) \leftarrow \mathcal{A}_1(st_{\mathcal{A}}, \mathcal{EI}_i, \mathcal{ED}_i)$
$tk_{i,w_1}, r_{c,w_1} \leftarrow TokenGen(pk_i, pk_c, sk_i, sk_c, w_1)$
$q_{c,w_1} \leftarrow QueryGen(pk_c, w_1, r_{c,w_1})$
$for\ 2 \leq j \leq q$
  $(w_j, st_{\mathcal{A}}) \leftarrow \mathcal{A}_j(st_{\mathcal{A}}, \mathcal{EI}_i, \mathcal{ED}_i, (tk_{i,w_1}, r_{c,w_1}, q_{c,w_1}),$
  $\cdots, (tk_{i,w_{j-1}}, r_{c,w_{j-1}}, q_{c,w_{j-1}}))$
  $(tk_{i,w_j}, r_{c,w_j}) \leftarrow TokenGen(pk_i, pk_c, sk_i, sk_c, w_j)$
  $q_{c,w_j} \leftarrow QueryGen(pk_c, w_j, r_{c,w_j})$
$let\ \mathcal{T}rp_i = ((tk_{i,w_1}, q_{c,w_1}), \cdots, (tk_{i,w_q}, q_{c,w_q}))$
$output\ V_i = (\mathcal{EI}_i, \mathcal{ED}_i, \mathcal{T}rp_i)\ and\ st_{\mathcal{A}}$
$\boldsymbol{Sim}_{\Pi,\mathcal{A},\mathcal{S}}(\lambda)$
$(I_i, D_i, st_{\mathcal{A}}) \leftarrow \mathcal{A}_0(1^\lambda)$
$(\mathcal{EI}_i, \mathcal{ED}_i, st_{\mathcal{S}}) \leftarrow \mathcal{S}_0(\gamma(D_i))$
$(w_1, st_{\mathcal{A}}) \leftarrow \mathcal{A}_1(st_{\mathcal{A}}, \mathcal{EI}_i, \mathcal{ED}_i)$
$(tk_{i,w_1}, r_{c,w_1}, st_{\mathcal{S}}) \leftarrow \mathcal{S}_1(st_{\mathcal{S}}, \gamma(D_i, w_1))$
$(q_{c,w_1}, st_{\mathcal{S}}) \leftarrow \mathcal{S}_1(st_{\mathcal{S}}, \gamma(D_i, w_1))$
$for\ 2 \leq j \leq q$
  $(w_j, st_{\mathcal{A}}) \leftarrow \mathcal{A}_j(st_{\mathcal{A}}, \mathcal{EI}_i, \mathcal{ED}_i, (tk_{i,w_1}, r_{c,w_1}, q_{c,w_1}), \cdots,$
  $(tk_{i,w_{j-1}}, r_{c,w_{j-1}}, q_{c,w_{j-1}}))$
  $(tk_{i,w_j}, r_{c,w_j}, st_{\mathcal{S}}) \leftarrow \mathcal{S}_j(st_{\mathcal{S}}, \gamma(D, w_1, \cdots, w_j))$
  $(q_{c,w_j}, st_{\mathcal{S}}) \leftarrow \mathcal{S}_j(st_{\mathcal{S}}, \gamma(D, w_1, \cdots, w_j))$
$let\ \mathcal{T}rp_i = ((tk_{i,w_1}, q_{c,w_1}), \cdots, (tk_{i,w_q}, q_{c,w_q}))$
$output\ V_i = (\mathcal{EI}_i, \mathcal{ED}_i, \mathcal{T}rp_i)\ and\ st_{\mathcal{S}}$

We say that VAKSE is adaptively and semantically secure if, for all polynomial-size adversaries $\mathcal{A} = (\mathcal{A}_0, \cdots, \mathcal{A}_q)$ such that $q = poly(\lambda)$, there exists a nonuniform polynomial-size simulator $\mathcal{S} = (\mathcal{S}_0, \cdots, \mathcal{S}_q)$ such that for all polynomial-size distinguishers $\mathcal{D}$, it has a negligible advantage in distinguishing the output of $\boldsymbol{Real}_{\Pi,\mathcal{A}}$ from $\boldsymbol{Sim}_{\Pi,\mathcal{A},\mathcal{S}}$.

Theorem 1: If the DL problem is hard and the adopted symmetric encryption is semantically secure, the proposed VAKSE scheme is adaptively secure.

Proof: We attempt to construct a polynomial-sized simulator $\mathcal{S} = (\mathcal{S}_0, \cdots, \mathcal{S}_q)$ such that for all polynomial-sized adversaries $\mathcal{A} = (\mathcal{A}_0, \cdots, \mathcal{A}_q)$, the outputs of $\mathbf{Real}_{\Pi,\mathcal{A}}(\lambda)$ and $\mathbf{Sim}_{\Pi,\mathcal{A},\mathcal{S}}(\lambda)$ are computationally indistinguishable. The simulator $\mathcal{S} = (\mathcal{S}_0, \cdots, \mathcal{S}_q)$

adaptively generates a string $V_i^* = (\mathcal{EI}_i^*, \mathcal{ED}_i^*, \mathcal{T}rp_i^*) = (\mathcal{EI}_i^*, \mathcal{ED}_i^*, (tk_{i,w_1}^*, q_{c,w_1}^*), \cdots, (tk_{i,w_q}^*, q_{c,w_q}^*))$ as follows.

$\mathcal{S}_0(\gamma(D_i))$: Using the trace of $\gamma(D_i)$, the simulator first calculates the total number n_d of documents. It randomly selects the same number of bytes as the size of each encrypted document to simulate the encrypted document. It thereby obtains an encrypted document set $\mathcal{ED}_i^*$. To simulate the encrypted inverted index $\mathcal{EI}_i$, it simulates each entry one by one. The encrypted keyword c_w is simulated by randomly selecting two numbers from the group G. The corresponding encrypted document identities c_{ids} are then simulated by randomly selecting bytes with the same size as the original ciphertext. Then it forms an encrypted index $\mathcal{EI}_i^*$ with the simulated entries.

Since, with all but negligible probability, $st_{\mathcal{A}}$ does not include sk_i and k_i, $\mathcal{EI}_i^*$ is indistinguishable from a real inverted index. Otherwise, one is able to distinguish either $\mathcal{EI}_i^*$ from $\mathcal{EI}_i$ or $\mathcal{ED}_i^*$ from $\mathcal{ED}_i$. With the assumption that the adopted symmetric encryption is semantically secure, it has a negligible advantage in distinguishing $\mathcal{ED}_i^*$ from $\mathcal{ED}_i$. Similarly, the advantage in distinguishing the encrypted document identities c_{ids}^* from c_{ids} is negligible. Distinguishing the encrypted keyword c_w^* from c_w requires analyzing the components of the ciphertext. Since the first component of both c_w^* from c_w is randomly selected, it is indistinguishable from each other. For the second component, with the assumption that the DL problem is hard, the probability to solve the problem is negligible.

$\mathcal{S}_1(st_{\mathcal{S}}, \gamma(D, w_1))$: Since we know that $D(w_1)$ corresponds to a set of document identities. The simulator needs to guarantee that, given the token and query, the corresponding identities should be retrieved. In order to achieve that, it first select two random numbers $r_{i,3}$ and $r_{i,4}$ from $\mathbb{Z}_p$ for the w_1. Then, it randomly chooses one encrypted keyword $c_w^* = (g^{r_{i,1}}, g^{r_{i,2}})$ from $\mathcal{EI}_i^*$. The corresponding components of the token and query are then constructed as $g^{(1+\frac{r_{i,4}r_{i,2}}{r_{i,1}r_{i,3}})*r_{i,3}}$ and $(g^{r_{i,4}}, g^{-r_{i,4}+\frac{r_{i,1}(r_{i,3}+r_{i,4})}{r_{i,2}}})$ respectively. Based on the token and query construction, the search result matches the requirement. Consequently, $\mathcal{T}rp_{i,1}^* = (tk_{i,w_1}^*, q_{c,w_1}^*)$ is the pair $(g^{(1+\frac{r_{i,4}r_{i,2}}{r_{i,1}r_{i,3}})*r_{i,3}}, (g^{r_{i,4}}, g^{-r_{i,4}+\frac{r_{i,1}(r_{i,3}+r_{i,4})}{r_{i,2}}}))$.

Since, with all but negligible probability, $st_{\mathcal{A}}$ does not include sk_i, tk_{i,w_1}^* is indistinguishable from a real tk_{i,w_1}. Otherwise, the DL problem can be solved in polynomial time. Similarly, with the assumption of the DL problem, the query q_{c,w_1}^* is indistinguishable from a real query.

$\mathcal{S}_j(st_{\mathcal{S}}, \gamma(D, w_1, \cdots, w_j))$ $(2 \le j \le q)$: for each w_j, $\mathcal{S}_j$ generates a token and query in the same way as $\mathcal{S}_1$ does. Similarly, since $st_{\mathcal{A}}$ includes the sk_i with negligible probability, the generated token and query are indistinguishable from real ones. □

VII. Performance

In this section, we first evaluate the performance of the VAKSE scheme and then that of the PVAKSE scheme, focusing on the data encryption, token generation, query generation, search, decryption, verification, and detection phases.

Experimental Setting: All the experiments are conducted using the Java programming language. The JPBC library [40] is used to implement the pairing operation. The symmetric pairing is constructed on the curve $y^2 = x^3 + x$ and the order of the group G is 160 bits. SHA-256 [41] is invoked to implement the hash function. The symmetric encryption algorithm is instantiated by using AES and implemented by invoking Java Crypto library [42] with 128 bits of the secret key. The experiments are executed on a laptop running on the Windows 10 Enterprise operating system, Intel(R) Core(TM) i7-7600U CPU, and 16GB RAM. To reduce the risk of system error, all the reported results are the averaged runtimes for the same operation over ten iterations.

Data Model: We use the publicly available and well-known Enron Email Dataset [37] as the experimental dataset. In total, we extract 517,400 valid emails and 133,321 keywords from the dataset.

A. Performance of VAKSE Scheme

In this section, we present the results of performing VAKSE for a case with one data owner and then demonstrate the results of performing VAKSE for a case with multiple independent data owners to vividly simulate the multitenant scenario.

1) Experiments With Single Data Owner: We evaluate the performance of the VAKSE scheme with a single data owner, especially considering data encryption, token generation, query generation, search, decryption, and verification processes.

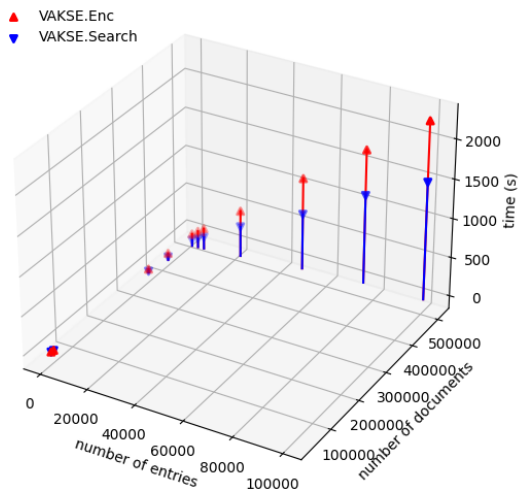
Evaluation of Data Encryption: In Section V-A, Algorithm 1 shows the details of the data encryption process. From this algorithm, we learn that the performance of the encryption algorithm is affected by the size of the index I and the total number of documents in D. As shown in Fig. 4a, the encryption time increases linearly and significantly with an increase in the number of documents and entries.

Evaluation of Token Generation: As shown in Algorithm 2, for each keyword w, the token generation algorithm is invoked. As shown in Fig. 4b, the time cost of token generation is almost constant (0.01 seconds), regardless of the increase in the number of entries and documents.

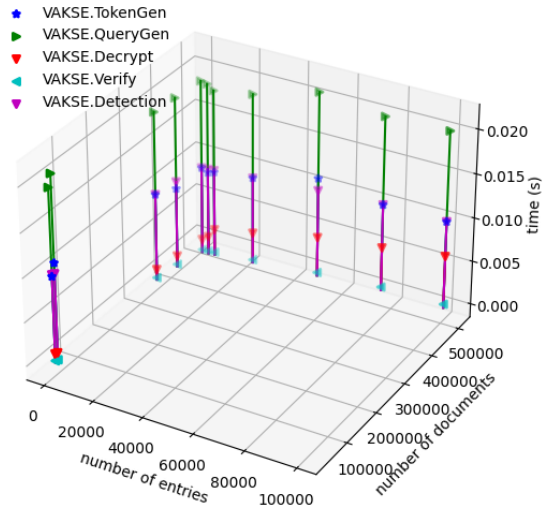
Evaluation of Query Generation: As shown in Algorithm 3, the query generation algorithm is invoked by the client. Fig. 4b shows that for each keyword w, the time cost of the query generation algorithm is almost constant at 0.02 seconds.

Evaluation of Search Phase: As shown in Algorithm 4, the search algorithm checks all the entries of the encrypted inverted index using the query and token. As shown in Fig. 4a, the time cost of the search algorithm increases linearly with the increase in the entries of the inverted index.

Evaluation of Decryption: The decryption phase includes two steps. The first step decrypts the encrypted document identities and the second step decrypts the encrypted documents. As shown in Algorithm 5 and 6, the retrieved encrypted document identities and documents are decrypted using the symmetric decryption algorithm. From Fig. 4b, we can see that with an increase in the number of documents and entries, the decryption time slowly increases. After analyzing the data,



(a) Performance of the VAKSE.$\mathcal{E}nc$ and VAKSE.$\mathcal{S}earch$ algorithms.



(b) Performance of the VAKSE.$\mathcal{T}okenGen$, VAKSE.$\mathcal{D}ecrypt$, VAKSE.$\mathcal{V}erify$, and VAKSE.$\mathcal{D}etection$ algorithms.

Fig. 4. Performance of VAKSE scheme. The graph shows the time cost of running various algorithms of VAKSE scheme over different datasets.

we learn that the size of the retrieved result tends to be larger for a larger dataset, which results in a higher time cost of decryption.

Evaluation of Verification: The verification algorithm is used to verify if the retrieved documents are correct and complete. From Algorithm 8, we observe that the verification performance is slightly affected by the size of the retrieved result. Due to the efficiency of SHA-256 [41], the time cost of the verification algorithm is approximately 0.0001 seconds.

Evaluation of Detection Algorithm: The detection algorithm is built to identify any client who misuses the allocated token. The detection algorithm (Algorithm 7) reveals the identity of such a client and has a fixed number of operations. As presented in Fig. 4b, the time cost is 0.01 seconds.

2) Experiments With Multiple Data Owners: From Section V-A, we learn that VAKSE supports multiple data owners. In the above experiment, we evaluate the performance of only the VAKSE scheme with one data owner. In a scenario with multiple data owners, there is a major difference in token generation and search processes. Note that the time cost of query generation is the same for either one or multiple data owners. To search across data owners, the client must obtain permission from the data owners. This results in multiple tokens. With the tokens, a search operation can be executed over access-granted data, instead of over all the data. To evaluate the token generation and search algorithm, we execute them over 2, 4, 8, 16, and 32 data owners who allows a client to access their data. Each data owner holds a dataset containing 3125 keywords and 412,477 documents. The results of sequentially running VAKSE.$\mathcal{T}okenGen$ and VAKSE.$\mathcal{S}earch$ algorithms are shown in Fig. 5. From the figure, we can observe that the time cost of both token generation and search increases approximately linearly with the increase in the number of data owners. However, the time consumption of token generation is minor compared with the time cost of search.

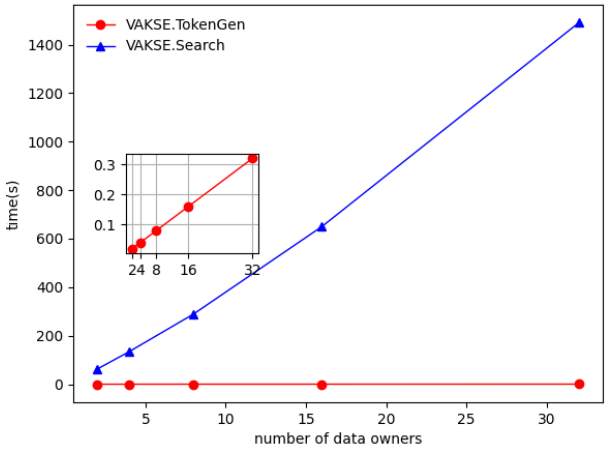


Fig. 5. Performance of the VAKSE.$\mathcal{T}okenGen$ and VAKSE.$\mathcal{S}earch$ algorithms. The graph shows the time cost of sequentially running the VAKSE.$\mathcal{T}okenGen$ and VAKSE.$\mathcal{S}earch$ algorithms over different number of data owners.

B. Performance of PVAKSE

To evaluate the performance of PVAKSE, we execute PVAKSE over a dataset containing 509,537 emails and 100,000 keywords. The inverted index built from the dataset is partitioned into 2, 4, 8, 16, and 32 parts on which to run different experiments. To simulate the parallel process in the experiments, each partition is independently processed, and the average time cost is reported. Since none of the modules, except encryption and search, are affected by parallelism, we focus on the time cost of data encryption and search. We name the parallel encryption algorithm as PVAKSE.$\mathcal{E}nc$ and parallel search algorithm as PVAKSE.$\mathcal{S}earch$. The performance of invoking PVAKSE.$\mathcal{E}nc$ and PVAKSE.$\mathcal{S}earch$ is presented in Fig. 6. From this figure, we learn that with the increasing number of partitions, the degree of parallel is also improved. With a linear increase in the number of partitions, the total time needed for encryption and search decreases

TABLE IV

CHARACTERISTIC COMPARISON WITH THE RELATED WORK. √ DENOTES THAT THE CORRESPONDING FUNCTION IS SUPPORTED AND × INDICATES THAT THE CORRESPONDING FUNCTION IS NOT CONSIDERED

Schemes	ciphertext search	search across data owners	verifiability	user accountability	fine-grained access control	parallelism
PDSSE [22]	√	×	√	×	×	√
GSSE [26]	√	×	√	×	√	×
CPAB-KSDS [33]	√	×	×	×	√	×
MACPABE [16]	×	×	×	√	√	×
VAKSE	√	√	√	√	√	×
PVAKSE	√	√	√	√	√	√

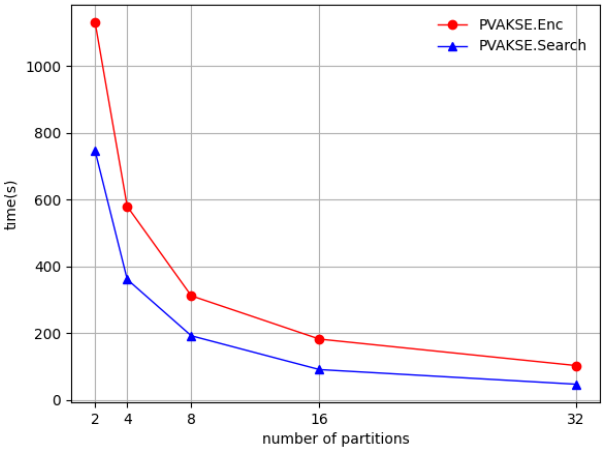


Fig. 6. Performance of the PVAKSE.$\mathcal{E}nc$ and PVAKSE.$\mathcal{S}earch$ algorithms. The graph shows the time cost of executing PVAKSE.$\mathcal{E}nc$ and PVAKSE.$\mathcal{S}earch$ algorithms on different number of partitions.

TABLE V

EFFICIENCY COMPARISON WITH RELATED WORK. n IS THE TOTAL NUMBER OF KEYWORDS, r IS THE NUMBER OF DOCUMENTS CONTAINING A KEYWORD, ρ IS THE NUMBER OF USER ATTRIBUTES, AND $|S|$ REPRESENTS THE SIZE OF THE SEARCH RESULT

Schemes	Ciphertext Search	Search Result Verification	Detection
PDSSE [22]	$O(r \log n/p)$	×	×
GSSE [26]	$O(n)$	$O(\log n)$	×
CPAB-KSDS [33]	$O(n)$	×	×
MACPABE [16]	×	×	$O(2^{\rho})$
VAKSE	$O(n)$	$O(\|S\|)$	$O(1)$
PVAKSE	$O(n/p)$	$O(\|S\|)$	$O(1)$

significantly. In particular, when the number of partitions increases from 1 to 8, the reduction in time cost is the most significant.

C. Experimental Comparison With Related Work

In this section, we conduct an experimental comparison of our scheme with those presented in Table IV. The focus is on encryption, search, token generation, and decryption. It is noteworthy that we exclude MACPABE [16] and GSSE [26] from consideration. MACPABE [16] lacks support for ciphertext search, and GSSE [26] is designed to provide a proof for validating search results independently from any SSE construction.

Experimental Setting: All the experiments are conducted on a cluster named Ibex, which contains more than 400 nodes. Each node has more than 40 cores and 350GB usable memory. The operating system on the nodes is CentOS 7.9 and the job scheduler is SLURM 20.11.6. All the remaining settings are the same as the previously mentioned experimental setup.

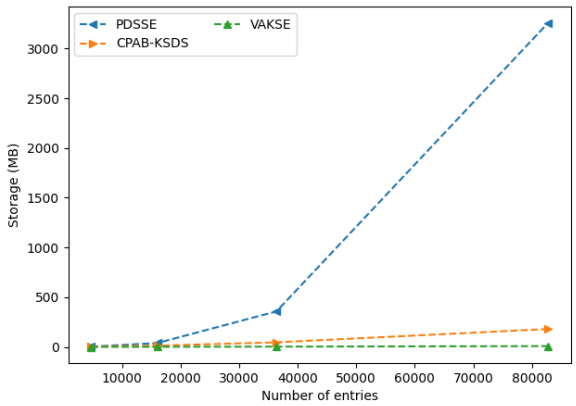


Fig. 7. Ciphertext storage cost. We compare the cipertext storage cost of PDSSE, CPAB-KSDS, and VAKSE with various numbers of entries.

Data Model: We use the publicly available Wikipedia dataset [43] as the experimental dataset. In particular, we choose the content written in English, namely *20220301.en* dataset. The total number of files is 6,458,670 and the total number of extracted keywords is 9,198,084.

1) Performance Comparison: In the experiment, to maintain consistency, the number of data owners is uniformly set to one across all schemes, and the number of threads is configured to 10 for parallel search in both PDSSE and PVAKSE. In addition, encryption operations are performed on the extracted keywords. From the experimental result, the public parameter size and private parameter size are 18 bytes and 48 bytes for PDSSE, respectively; 6290 bytes and 406 bytes for CPAB-KSDS, respectively; and 1066 bytes and 96 bytes for our schemes, respectively. For communication costs, given that the retrieved results are the same, we focus on the size of the query request. In PDSSE, the query request contains only one token, with a total size of 30 bytes. In CPAB-KSDS, the query size has two components: the first part includes the fixed five group members, and the second part is determined by the attribute size of the client. In our experiment, assuming three client attributes, the query size comprises eight group members, resulting in 2480 bytes. In VAKSE, the query request includes three group members and totals 930 bytes.

The storage cost for the ciphertext is illustrated in Fig. 7. PDSSE exhibits the highest storage cost, whereas our scheme demonstrates the lowest cost. The substantial storage cost associated with PDSSE is attributed to the maintenance of the tree structure, which nonetheless offers significant advantages in search operations. The encryption time cost is depicted in Fig. 8. The encryption operation of CPAB-KSDS incurs

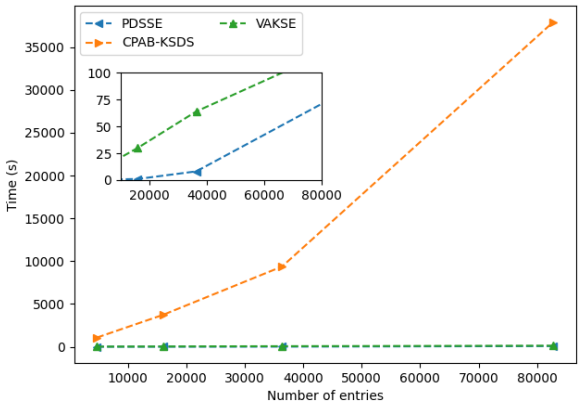


Fig. 8. Time cost of encryption operation. We compare the encryption time cost of PDSSE, CPAB-KSDS, and VAKSE with various numbers of entries.

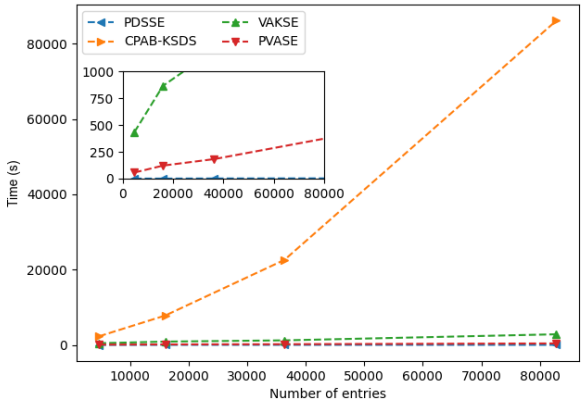


Fig. 9. Time cost of search operation. We compare the search time cost of PDSSE, CPAB-KSDS, VAKSE, and PVAKSE with various numbers of entries.

the heaviest time cost, and PDSSE incurs the lowest time cost. The significant encryption time cost of CPAB-KSDS is attributed to its large number of exponentiation and pairing operations, which are not required by PDSSE. The search time cost is illustrated in Fig. 9. In this figure, we compare the search time costs of PDSSE, CPAB-KSDS, VAKSE, and PVAKSE. PDSSE exhibits the best search performance, PVAKSE takes the second position, VAKSE ranks third, and CPAB-KSDS is the least efficient. Additionally, the time cost of token generation for PDSSE, CPAB-KSDS, and VAKSE is approximately 34ms, 900ms, and 169ms, respectively. For decryption, CPAB-KSDS does not perform this operation. The decryption time costs for PDSSE and our scheme are similar, and both require approximately 0.02ms for decrypting one ciphertext.

VIII. Discussion

A. Mitigating the Storage Requirement of Data Owners

In the above schemes, the data owner needs to maintain dictionary D_c to support detection. Each entry in D_c stores a number that is selected by a client for keyword authorization, which may result in a large size. To mitigate the demand for storage, the $r_{c,w}$ could be generated by the data owner instead of random selection by the client. However, the chosen number should match the requirement for distinguishing between different clients and keywords. To meet the requirements, the data owner conducts $r_{c,w} \leftarrow H(pk_c, \alpha_w)$ to generate the number, where pk_c is the public key of a client, α_w is the hash result of the submitted keyword, and H is a hash function. Using this design, the data owner does not need to store D_c. As the side effect, the data owner is required to check all the public keys of clients before able to detect the malicious user.

B. Detection of Denial of Query Service

A malicious CSP may reject a query or directly return an empty search result. To detect such misbehavior, the token generation process needs to be modified. Before a data owner decides to approve the authorization request, the data owner needs to verify whether the requested keyword is contained in the dataset. If it is not present in the dataset, the data owner rejects the authorization request. Otherwise, the data owner processes the authorization request as before. Based on the existing configuration, the situation in which an empty search result is considered to be valid should not occur. As a result, a client has the capability to identify instances where the CSP refuses to process queries or return empty search results.

C. Real-World Use Cases and Deployment Scenarios

The proposed scheme is versatile and well-suited for deployment in environments that require selective data sharing or utilization across multiple tenants. One specific application is for managing patient data. With the increasing prevalence of cloud services, hospitals are opting to outsource their data to a CSP and to become tenants of its cloud service. To ensure data protection, the outsourced data requires safeguarded in accordance with self-interest and regulations. In addition, patients may visit different hospitals for various medical conditions. Ultimately, when patients seek access to their historical records for specific diseases, it becomes essential to search across multiple tenants. Another example application is in academic certificates. Currently, universities and schools are utilizing the services of a CSP and enjoying the benefits as tenants. An individual may often receive requests to prove their educational background. In such a scenario, individuals need access to access data from multiple tenants to gather all their educational information.

IX. Conclusion

Herein, we propose a privacy-preserving, efficient, verifiable, accountable, and parallel solution for the keyword search problem in a multitenant cloud environment. To achieve this, we devised a privacy-preserving inverted index to enable a verifiable ciphertext search. Each entry contains encrypted keyword and document identity pairs and the compressed MAC for all corresponding documents. Then, we designed a fine-grained access control mechanism through keyword-based token generation. Moreover, we embedded the user identity into the token to achieve user accountability. All those components were built into the VAKSE scheme. To further improve search efficiency, we introduced the PVAKSE, in which the inverted index was partitioned into small segments that

could be searched synchronously. Finally, we formally analyzed the security of our proposed schemes and conducted extensive experiments to show their effectiveness. For future work, we intend to enhance the security and performance of PVAKSE further.

REFERENCES

[1] (2022). *Healthvana*. [Online]. Available: https://healthvana.com/

[2] (2022). *CDPHP*. [Online]. Available: https://www.cdphp.com/

[3] (2022). *Customer Success Stories*. [Online]. Available: https://aws.amazon.com/solutions/case-studies/

[4] (2022). *HiPAA*. [Online]. Available: http://www.cms.hhs.gov/HIPAAGenInfo/

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

[6] E.-J. Goh, "Secure indexes," *Cryptol. ePrint Arch.*, Oct. 2003.

[7] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: Improved definitions and efficient constructions," *J. Comput. Secur.*, vol. 19, no. 5, pp. 895–934, 2011.

[8] Q. Chai and G. Gong, "Verifiable symmetric searchable encryption for semi-honest-but-curious cloud servers," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2012, pp. 917–922.

[9] C. Chen et al., "An efficient privacy-preserving ranked keyword search method," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 4, pp. 951–963, Apr. 2016.

[10] F. Han, J. Qin, H. Zhao, and J. Hu, "A general transformation from KP-ABE to searchable encryption," *Future Gener. Comput. Syst.*, vol. 30, pp. 107–115, Jan. 2014.

[11] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proc. 13th ACM Conf. Comput. Commun. Secur.*, Oct. 2006, pp. 89–98.

[12] R. Brinkman, L. Feng, J. Doumen, P. H. Hartel, and W. Jonker, "Efficient tree search in encrypted data," *Inf. Syst. Secur.*, vol. 13, no. 3, pp. 14–21, May 2004.

[13] R. A. Popa, C. M. S. Redfield, N. Zeldovich, and H. Balakrishnan, "CryptDB: Protecting confidentiality with encrypted query processing," in *Proc. 23rd ACM Symp. Oper. Syst. Principles (SOSP)*, 2011, pp. 85–100.

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

[15] J. Li, K. Ren, B. Zhu, and Z. Wan, "Privacy-aware attribute-based encryption with user accountability," in *Proc. Int. Conf. Inf. Secur.* Berlin, Germany: Springer, 2009, pp. 347–362.

[16] J. Li, X. Chen, S. S. M. Chow, Q. Huang, D. S. Wong, and Z. Liu, "Multi-authority fine-grained access control with accountability and its application in cloud," *J. Netw. Comput. Appl.*, vol. 112, pp. 89–96, Jun. 2018.

[17] A. Soleimanian and S. Khazaei, "Publicly verifiable searchable symmetric encryption based on efficient cryptographic components," *Des., Codes Cryptogr.*, vol. 87, no. 1, pp. 123–147, 2019.

[18] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 1, pp. 222–233, Jan. 2013.

[19] J. Wang et al., "Efficient verifiable fuzzy keyword search over encrypted data in cloud computing," *Comput. Sci. Inf. Syst.*, vol. 10, no. 2, pp. 667–684, 2013.

[20] W. Sun et al., "Privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking," in *Proc. 8th ACM SIGSAC Symp. Inf., Comput. Commun. Secur.*, 2013, pp. 71–82.

[21] Z. Fu, L. Xia, X. Sun, A. X. Liu, and G. Xie, "Semantic-aware searching over encrypted data for cloud computing," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 9, pp. 2359–2371, Sep. 2018.

[22] S. Kamara and C. Papamanthou, "Parallel and dynamic searchable symmetric encryption," in *Proc. Int. Conf. Financial Cryptography Data Secur.* Berlin, Germany: Springer, 2013, pp. 258–274.

[23] W. Sun, X. Liu, W. Lou, Y. T. Hou, and H. Li, "Catch you if you lie to me: Efficient verifiable conjunctive keyword search over large dynamic encrypted cloud data," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2015, pp. 2110–2118.

[24] R. Bost, P.-A. Fouque, and D. Pointcheval, "Verifiable dynamic symmetric searchable encryption: Optimality and forward security," *Cryptol. ePrint Arch.*, Jan. 2016.

[25] K. Kurosawa and Y. Ohtaki, "UC-secure searchable symmetric encryption," in *Proc. Int. Conf. Financial Cryptogr. Data Secur.* Berlin, Germany: Springer, 2012, pp. 285–298.

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

[27] M. Hinek, S. Jiang, R. Safavi-Naini, and S. Shahandashti, "Attribute based encryption with key cloning protection," Tech. Rep. 2008/478, 2008.

[28] S. Yu, K. Ren, W. Lou, and J. Li, "Defending against key abuse attacks in KP-ABE enabled broadcast systems," in *Proc. Int. Conf. Secur. Privacy Commun. Syst.* Berlin, Germany: Springer, 2009, pp. 311–329.

[29] Q. Zheng, S. Xu, and G. Ateniese, "VABKS: Verifiable attribute-based keyword search over outsourced encrypted data," in *Proc. IEEE Conf. Comput. Commun.*, Apr. 2014, pp. 522–530.

[30] D. Boneh, G. D. Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *Proc. Int. Conf. Theory Appl. Cryptograph. Techn.* Berlin, Germany: Springer, 2004, pp. 506–522.

[31] K. Liang and W. Susilo, "Searchable attribute-based mechanism with efficient data sharing for secure cloud storage," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 9, pp. 1981–1992, Sep. 2015.

[32] Q. Huang, G. Yan, and Q. Wei, "Attribute-based expressive and ranked keyword search over encrypted documents in cloud computing," *IEEE Trans. Services Comput.*, vol. 16, no. 2, pp. 957–968, Mar. 2023.

[33] C. Ge, W. Susilo, Z. Liu, J. Xia, P. Szalachowski, and L. Fang, "Secure keyword search and data sharing mechanism for cloud computing," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 6, pp. 2787–2800, Nov. 2021.

[34] R. A. Mollin, *An Introduction to Cryptography*. Boca Raton, FL, USA: CRC Press, 2006.

[35] R. W. van der Heijden, S. Dietzel, T. Leinmüller, and F. Kargl, "Survey on misbehavior detection in cooperative intelligent transportation systems," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 1, pp. 779–811, 4th Quart., 2018.

[36] S. Ruj, M. A. Cavenaghi, Z. Huang, A. Nayak, and I. Stojmenovic, "On data-centric misbehavior detection in VANETs," in *Proc. IEEE Veh. Technol. Conf. (VTC Fall)*, Sep. 2011, pp. 1–5.

[37] (2022). *Enron Email Dataset*. [Online]. Available: https://www.cs.cmu.edu/

[38] (2022). *CPU Cores Threads per CPU Core per Instance Type*. [Online]. Available: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/cpu-options-supported-instances-values.html

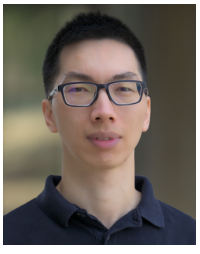
[39] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: Improved definitions and efficient constructions," in *Proc. 13th ACM Conf. Comput. Commun. Secur.*, 2006, pp. 79–88.

[40] A. De Caro and V. Iovino, "JPBC: Java pairing based cryptography," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, Jun. 2011, pp. 850–855.

[41] (2022). *Java Security Message Digest*. [Online]. Available: https://docs.oracle.com/javase/7/docs/api/java/security/MessageDigest.html

[42] Oracle. *Java™ Platform, Standard Edition 8 API Specification*. Accessed: Feb. 1, 2023. [Online]. Available: https://docs.oracle.com/javase/8/docs/api/javax/crypto/Cipher.html

[43] Wikimedia Foundation. *Wikimedia Downloads*. Accessed: Feb. 1, 2023. [Online]. Available: https://dumps.wikimedia.org



Xiaojie Zhu (Member, IEEE) received the M.S. degree from the University of Chinese Academy of Sciences in 2015 and the Ph.D. degree from the University of Oslo in 2021. He is currently a Staff Research Scientist with KAUST. His research interests include cloud security, applied cryptography, and distributed systems.

Peisong Shen received the bachelor's degree from the University of Science and Technology of China in 2012 and the Graduate degree from the University of Chinese Academy of Sciences in 2018. Since then, he has been with the Institute of Information Engineering as a Research Assistant. He is currently an Assistant Researcher with the Institute of Information Engineering, Chinese Academy of Sciences. His research interests include applied cryptography and data privacy protection.

Yueyue Dai (Member, IEEE) was a Researcher with Nanyang Technological University, Singapore, in 2020. She is currently an Associate Professor with Huazhong University of Science and Technology. Her current research interests include edge intelligence, the Internet of Vehicles, and blockchain. She serves/has served as a Guest Editor for many leading journals *IEEE Network*, *Future Generation Computer Systems*, and *Digital Communications and Networks*; a PC Member for IEEE Symposium on Blockchain; and a TPC Member for IEEE ICC 2022, IEEE GLOBECOM 2021, IEEE ICC 2021, and VTC2020-Spring.

Lei Xu (Member, IEEE) received the Ph.D. degree from Nanjing University of Science and Technology in 2019. He is currently an Associate Professor with the School of Mathematics and Statistics, Nanjing University of Science and Technology. Before that, he was a Post-Doctoral Researcher with the Department of Computer Science, City University of Hong Kong, Hong Kong. He was a Visiting Ph.D. Student with the Faculty of Information Technology, Monash University, from April 2017 to April 2018. His main research interests focus on applied cryptography and information security.

Jiankun Hu (Senior Member, IEEE) is currently a Full Professor in cyber security with the School of Engineering and Information Technology, The University of New South Wales, Defence Force Academy (UNSW@ADFA), Australia. His major research interests include computer networking and computer security, especially biometric security. He has been awarded ten Australia Research Council Grants. He served as the Security Symposium Co-Chair for IEEE GLOBECOM '08 and IEEE ICC '09. He was the Program Co-Chair of the 2008 International Symposium on Computer Science and its Applications. He is serving as an Associate Editor for the following journals: *Journal of Security and Communication Networks* (Wiley), a Senior Area Editor for IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, and an Area Editor for *KSII TIIS*, *IET CPS*, IEEE OPEN JOURNAL OF THE COMMUNICATIONS SOCIETY, and S&P (Wiley).