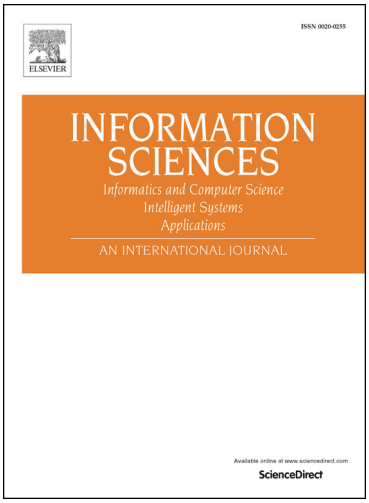
Journal Pre-proof



Key-Aggregate Searchable Encryption Supporting Conjunctive Queries for Flexible Data Sharing in the Cloud

Jinlu Liu, Bo Zhao, Jing Qin, Xinyi Hou and Jixin Ma

PII: S0020-0255(23)00921-0
DOI: https://doi.org/10.1016/j.ins.2023.119336
Reference: INS 119336

To appear in: *Information Sciences*

Received date: 7 November 2021
Revised date: 14 June 2023
Accepted date: 16 June 2023

Please cite this article as: J. Liu, B. Zhao, J. Qin et al., Key-Aggregate Searchable Encryption Supporting Conjunctive Queries for Flexible Data Sharing in the Cloud, *Information Sciences*, 119336, doi: https://doi.org/10.1016/j.ins.2023.119336.

Key-Aggregate Searchable Encryption Supporting Conjunctive Queries for Flexible Data Sharing in the Cloud

Jinlu Liu[a], Bo Zhao[a], Jing Qin[a,*], Xinyi Hou[a], Jixin Ma[b]

[a]*School of Mathematics, Shandong University, Jinan, 250100, Shandong, China*
[b]*School of Computing and Mathematical Sciences University of Greenwich, London, UK*

Abstract

Searchable encryption (SE) meets users' demand for the keyword search on encrypted data. Key-aggregate searchable encryption (KASE) improves data owners' ability to selectively share encrypted data with users. In KASE, the data owner encrypts different documents/document classes with distinct keys and can share any selected subset of documents by simply transmitting an aggregate key to the user. The user only uploads an aggregate trapdoor to the server for querying these shared documents. However, the existing KASE schemes have some limitations: the security definition is incomplete, only single-keyword search is supported, and the provable security scheme relies on the random oracle model. For these reasons, in this paper, we propose the Key-Aggregate Searchable Encryption supporting Conjunctive Queries (KASE-CQ) framework and its two security models: indistinguishability against selective-document chosen keyword attack and existential unforgeability against selective-document chosen keyword attack. These models reflect the indistinguishability of ciphertext and the unforgeability of the aggregate key, respectively. Our system supports flexible data sharing and the conjunctive keyword search on encrypted data. Furthermore, we design a concrete KASE-CQ construction, which can be proven secure in the

*Corresponding author
Email addresses: jinlulmath@163.com (Jinlu Liu), zhaobomath@163.com (Bo Zhao), qinjing@sdu.edu.cn (Jing Qin), hxy2353@163.com (Xinyi Hou), j.ma@greenwich.ac.uk (Jixin Ma)

standard model. We also demonstrate that our construction is secure against the insider trapdoor attack presented by Zhou et al. [40]. Finally, performance analysis and comparisons with Cui et al.'s scheme [10] illustrate the superior efficiency of our scheme.

Keywords: Data sharing, Key-aggregate, Conjunctive keyword query, Standard model.

1. Introduction

In today's era of data explosion, data storage and transmission problems have been solved with the advent of cloud technology [32, 18, 23, 28, 17]. Cloud storage is an innovative information storage technology. With the Internet, especially 5G communication technology, users can quickly transfer their files, photos, and videos to cloud terminals anytime and anywhere [6]. When data is outsourced, users cannot control data directly, making data privacy a significant concern for cloud storage [31, 35, 39, 29]. A basic strategy is "encryption-before-outsourcing" [37, 28, 41]. However, when data is encrypted using traditional encryption schemes with semantic security (such as AES and ElGamal), the encrypted data is indistinguishable from a ciphertext randomly selected from the ciphertext space. This leads to the loss of semantic information [27], posing challenges for users to efficiently search for the data of interest.

To support data retrieval without compromising confidentiality, searchable encryption (SE) [20, 12, 11, 24, 36, 38] technology has been proposed. In an SE scheme, a data owner encrypts potential keywords and uploads them to the cloud server along with the encrypted data. When searching for data matching a specific keyword, the user submits the corresponding keyword trapdoor (i.e. encrypted keyword) to the server. This allows the server to test whether any ciphertext matches the trapdoor, namely ciphertexts embedded with the same keyword as the trapdoor. The SE is mainly classified into searchable symmetric encryption (SSE) and public key encryption with keyword search (PEKS). SSE permits only the secret key holder to produce ciphertexts and generate trapdoors

for searching. On the other hand, PEKS allows certain users who know the public key to generate ciphertexts, but only the secret key holder can create trapdoors.

1.1. The Need for Key-Aggregate Searchable Encryption

Although SE technology has attracted broad attention from both industry and academia, it still has limitations. One particular limitation is the ability to selectively share encrypted data, which is a crucial function of cloud systems. Data owners outsource much of their data to the cloud server but want to authorize various users to access different parts of them. In large-scale applications with numerous users and files, implementing systems that support both keyword search and selective sharing of encrypted data using traditional searchable encryption techniques is hindered by practical problems. These problems include the complex and expensive key management in SSE and the need for multiple copies of ciphertexts in PEKS. To further illustrate this point, we consider the following scenario.

Assume that Alice uses a public storage server (e.g., Dropbox) to store her private photos. Before uploading them to the cloud storage, Alice encrypts her photos to prevent potential data leakage and generates keyword ciphertexts based on time, place, and person, enabling her to search for photos of interest. Alice then selectively shares these photos with her friends. For instance, she shares photos 1, 2, and 3 with Bob, photos 1 and 2 with Carol, and photos 2 and 4 with Dan. For her friends to view the shared photos, Alice must delegate permission for keyword search and decryption of these photos to them. There are two extreme methods for Alice using the traditional SE technology [12, 8, 11, 7]:

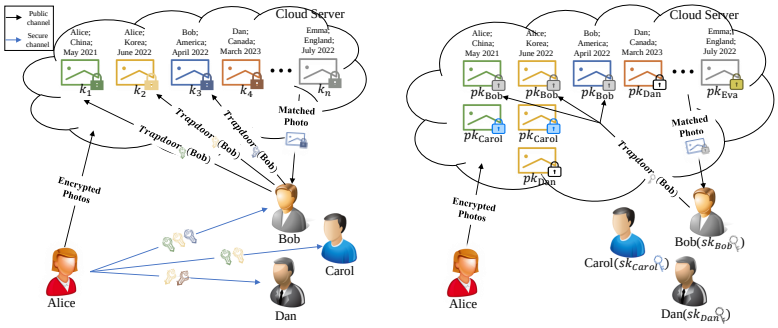
1) Using SSE, Alice encrypts her photos (and their keywords) with different secret keys. She then sends the secret keys for photos 1, 2, and 3 to Bob, the secret keys for photos 1 and 2 to Carol, and the secret keys for photos 2 and 4 to Dan. Bob securely stores the received secret keys. When searching for photos matching a keyword, he utilizes these keys to generate the trapdoors for photos 1, 2, and 3, respectively. Then, Bob

submits these trapdoors to the cloud server. Carol and Dan can also perform similar operations.

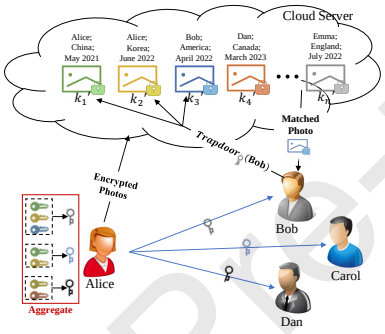
2) Using PEKS, Alice encrypts photos 1, 2, and 3 with Bob's public key, photos 1 and 2 with Carol's public key, and photos 2 and 4 with Dan's public key. Then, Bob can utilize his secret key to generate a trapdoor and submit it to the cloud server to conduct keyword searches on photos 1, 2, and 3. Carol and Dan can also perform similar operations.

Both of these methods are very heavy and expensive to implement. For the first method, first, the number of secret keys that need to be distributed to a user is proportional to the number of shared photos. These keys not only need to be sent via secure channels but also to be securely stored and managed by users within their devices. Second, to perform the keyword search on shared photos, a large number of trapdoors must be generated and submitted to the server. The communication, storage, and computational complexity involved generally increase with the number of secret keys to be shared. For the second method, a photo is encrypted multiple times using the public keys of all authorized users. The resulting multiple ciphertexts for each photo need to be uploaded and stored in the cloud server. The communication, computational, and (cloud) storage complexity involved generally increase with the number of users authorized to access a photo. Therefore, both methods render the system inefficient and impractical.

The Key-Aggregate Searchable Encryption (KASE) proposed by Cui et al. [10] is an excellent technology to address the abovementioned problems. In KASE, the data owner divides his data into different classes and encrypts each piece of data not only under his public key but also under an identifier that indicates the data's class. The data owner can extract an aggregate key of constant size from any number of different classes by embedding the secret key into the product of public keys associated with these data classes. The aggregate key is a single key but aggregates the ability to search ciphertexts of these different classes. Therefore, with KASE, the data owner can share any subset



(a) SSE (b) PEKS



(c) KASE

Figure 1: Keyword search in data sharing

of data classes by solely sending the corresponding aggregate key to the user via secure channels. The user merely generates a single aggregate trapdoor using the obtained aggregate key and submits it to the cloud for searching all shared documents. The server can adjust this aggregate trapdoor to trapdoors for each document using some public information [1]. Compared with the methods using SSE and PEKS, the most prominent property of the KASE system is that the ciphertext, aggregate key, and trapdoor are all of constant size. This property significantly reduces the computational, communication, and storage overhead.

[1] We notice that although the server needs to adjust the aggregate trapdoor to trapdoors for each document, it can be seen from our concrete KASE-CQ construction in Section 4 that the process only requires the server to perform multiplication operations, which is acceptable for the computation-rich cloud server.

Fig. 1 depicts the comparison of using SSE, PEKS, and KASE to realize the above photo sharing scenario.

1.2. Current Research States

Since the introduction of the KASE system [10], various KASE schemes [19, 21, 22] have been proposed. However, none of these schemes provide concrete security model definitions for KASE. Zhou et al. [40] showed an attack on [10] (none of [19, 21, 22] can resist such an attack), which we call the inside trapdoor attack (ITA). In ITA, the inside attacker of the system can guess the aggregate keys of users who share the document access with him. [40] also designed a file-centric multi-key aggregate keyword searchable encryption (Fc-MKA-KSE) scheme that can resist ITA and formally defined two security models: indistinguishability against selective-file chosen keyword attack (IND-sF-CKA) and indistinguishability against selective-file keyword guessing attack (IND-sF-KGA). The IND-sF-CKA ensures that an adversary who can perform aggregate key queries and trapdoor queries cannot obtain the relationship between the challenge ciphertext and the corresponding keyword. Hence, it captures the privacy of the keyword ciphertext, that is, the keyword ciphertext will not reveal any information about the corresponding keyword to unauthorized users. The IND-sF-KGA ensures that an adversary who can perform aggregate key queries and keyword ciphertext queries cannot obtain the relationship between the challenge trapdoor and the corresponding keyword. So it captures the privacy of the keyword trapdoor, that is, the keyword trapdoor will not reveal any information about the corresponding keyword to unauthorized users.

Although KASE is an invaluable cryptographic tool for achieving both keyword search and selective sharing of encrypted data, existing schemes still face the following problems. These problems limit their practical utilization and motivate our work.

- **Security Definition.** In most previous studies on key aggregate searchable encryption, the security definition was not formally discussed. Zhou

et al. [40] defined the IND-sF-CKA and IND-sF-KGA security models. However, the IND-sF-CKA does not capture that the adversary can obtain the ciphertext stored on the server, because the adversary is not allowed to perform ciphertext queries (Note that the generation of the keyword ciphertext requires a secret key). Moreover, in the trapdoor query of IND-sF-CKA, only the non-challenged file set is allowed to be queried, which can be calculated by itself after the adversary conducts the aggregate key query. Thus, the trapdoor query is unnecessary. Additionally, although Zhou et al. defined the IND-sF-KGA security model, they did not provide proof under this model. [40] also did not prove that their scheme could resist ITA. At last, search controllability is necessary in the KASE system [10], which means the attacker cannot generate a new aggregate key from the known aggregate keys, to search for files that are not authorized to access. However, no related security model is given in [40].

- **Query Type.** Existing key-aggregate searchable encryption systems solely allow the single keyword search. However, the conjunctive query [13, 1, 25] is essential for the efficient utilization of the data repository, because the single keyword search always produces extremely coarse results. For example, Bob may not want to search all the photos in which "Alice" appeared, but only the photos taken in "China" in "May 2021" in which "Alice" appeared. In this case, the capacity to search "$Alice \wedge China \wedge May$ 2021" is needed. To achieve this goal with existing schemes, the user is required to submit trapdoors for each keyword. Subsequently, the server utilizes each trapdoor to locate the documents matching each keyword and returns the intersection of those documents to the user. This approach is insecure and inefficient because the server can know which documents contain each individual keyword in addition to the result of the conjunctive query, and the computational and communication overhead of the trapdoor is linear in the number of conjuncts being searched.

- **Security Model.** Zhou et al. [40] proved that their Fc-MKA-KSE

scheme meets the IND-sF-CKA security model. Regrettably, this can only be proved in the random oracle model (ROM). In the security proof, ROM is an ideal substitute for the hash function in reality. The scheme proved secure in the ROM is not sure secure in the actual implementation. Hence, it is preferable to design a scheme that can prove security in the standard model (SM).

1.3. Our Contributions

To fill the gap in the literature, we propose a novel KASE system called Key-Aggregate Searchable Encryption supporting Conjunctive Queries (KASE-CQ). We also present two security models to capture the keyword ciphertext security and controlled searching respectively, as well as a concrete KASE-CQ construction. Compared with previously proposed KASE schemes, we make the following primary contributions:

- **Enhanced Security Definition.** We formalize the indistinguishability against selective-document chosen keyword attack (IND-sDOC-CKA) and existential unforgeability against selective-document chosen keyword attack (EU-sDOC-CKA) security models on the KASE-CQ system. The definition of IND-sDOC-CKA is the same as that of IND-sF-CKA [40], except that our model also captures the ability to obtain the ciphertext stored on the server by allowing the adversary to perform ciphertext queries. The EU-sDOC-CKA model captures the search controllability. Specifically, an adversary owing aggregate keys $K_{agg}^{S_1}, K_{agg}^{S_2}, ..., K_{agg}^{S_q}$ corresponding to document class sets S_1, S_2,...,S_q, respectively, cannot generate a new aggregate key corresponding to set S^*, such that the challenge document i^* satisfies $i^* \in S^*$ and $i^* \notin (S_1 \cup S_2 \cup ... \cup S_q)$. In other words, the attacker cannot generate an aggregate key for a document set containing an unauthorized document from known aggregate keys.

- **Conjunctive Keyword Query.** Our KASE-CQ construction can support conjunctive queries, i.e. given a keyword set $\{w_1, w_2, ..., w_l\}$, the user

can generate the trapdoor for "$w_1 \wedge w_2 \wedge ... \wedge w_l$". The server can then use this trapdoor to directly locate and return the set of documents containing all of these keywords. This eliminates the need for generating separate trapdoors for each keyword and returning the intersection of documents matching each keyword. Therefore, in our KASE-CQ scheme, the computational and communication overhead of trapdoors is independent of the number of conjuncts being searched. The server does not know the documents that match each individual keyword in the conjunctive query.

- **Standard Model.** We formally prove that KASE-CQ construction meets the ciphertext indistinguishability under the decisional Bilinear Diffie-Hellman Exponent (BDHE) assumption in the IND-sDOC-CKA security model and the aggregate key unforgeability under the Diffie-Hellman Exponent (DHE) assumption in the EU-sDOC-CKA security model without random oracle. We also prove the KASE-CQ is secure against ITA.

1.4. Related Work

Chu et al. [9] defined the key-aggregate cryptosystem (KAC). Thereafter, many KAC schemes have been proposed [14, 26, 34, 30, 15]. The KAC system allows users to decrypt multiple documents encrypted with different keys using a single aggregate key, realizing the effective authorization of decryption permission on any ciphertext set. However, KAC does not provide functionality for searching encrypted data. Cui et al.[10] applied the aggregate key technology to propose a KASE system. The data owner uses different keys to encrypt different documents. When sharing multiple documents, the data owner only transmits an aggregate key to the user. Users only generate an aggregate trapdoor to search for documents encrypted with different keys. However, [10] did not give the formal security model definitions for its scheme. Considering that the server may return only a portion of the search results to save its computing resources, Li et al. [19] introduced a verifiable searchable encryption with aggregate keys (VSEAK) scheme using bloom filter [3] based on [10]. But the

verification mechanism of this scheme causes a significant computational burden to the user. When multiple data owners share documents with the same user, [10] and [19] require users to securely store aggregate keys from multiple owners and generate trapdoors corresponding to each aggregate key for search. Users have a great burden of computation and storage. Accordingly, Liu et al. [21] designed a VSEAK scheme under the multi-owner setting. Building upon the scheme proposed in [19], they use a key uk to encrypt each aggregate key and convert them into auxiliary values that can be publicly stored on the server. The user just stores uk locally and uses it to generate a single trapdoor to query multiple owners' data.

In 2018, Zhou et al. [40] proposed a concrete attack on the scheme proposed in [10], which we call the inside trapdoor attack (ITA). In addition, Zhou et al. [40] presented an Fc-MKA-KSE system and its two security models: IND-sF-CKA and IND-sF-KGA, which capture the keyword ciphertext security and trapdoor privacy, respectively. However, in the concrete Fc-MKA-KSE construction proposed, the public/secret keys of the data owner are proportional to the documents' quantity, and the security is proved in ROM. Since the cloud server probably tampers with data stored by data owners and unauthorized users may send search queries to obtain information about the data owner, Liu et al. [22] introduced a verifiable and authenticated KASE scheme. In 2019, Wang et al. [33] presented an efficient key-aggregate keyword searchable encryption scheme, which is secure against ITA, but the elements contained in each document index are linearly related to the data owner's documents.

In Table 1, we compare our work with KASE [10] and Fc-MKA-KSE [40] in terms of functionality and security. Note we use — to denote "not applicable". For all we know, our scheme is the first KASE system that simultaneously formalizes the security model, supports conjunctive keyword search, is provably secure in the SM, and is secure against ITA.

Table 1: Functionality and Security Comparise with [10], [40]

Scheme	Query Type	Against ITA	Security Definition	ROM or SM
KASE [10]	Single	✗	✗	—
Fc-MKA-KSE [40]	Single	✓(no proof provided)	IND-sF-CKA IND-sF-KGA	ROM
Ours	Conjunctive	✓(proof is provided)	IND-sDOC-CKA EU-sDOC-CKA	SM

1.5. Organization

In the following Section, we present the relevant notations, abbreviations, and primitives. In Section 3, we propose the system model, system framework, and requirements of KASE-CQ. The KASE-CQ scheme and related security analysis are provided in Section 4, and in Section 5 we show performance analysis. The conclusion is in Section 6.

2. PRELIMINARIES

In this section, we introduce some cryptography backgrounds relevant to our study, including the description of main notations and abbreviations, bilinear pairing, and the complexity assumptions on which the security of our scheme is based.

2.1. Notations and Abbreviations

The main notations and abbreviations used in this paper are illustrated in Table 2 and Table 3, respectively.

2.2. Bilinear Pairing

$\mathcal{G}$ and $\mathcal{G}_1$ are two cyclic multiplicative groups of prime order p, and g is a generator of $\mathcal{G}$. The map $e : \mathcal{G} \times \mathcal{G} \to \mathcal{G}_1$ is a bilinear pairing if the following properties are satisfied [4].

- Bilinearity: For $\forall\ g_1, g_2 \in \mathcal{G}$, $a, b \in \mathcal{Z}_p$, $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$.
- Non-degeneracy: $e(g, g) \neq 1$.

Table 2: Notations

Notation	Description
$a \in G$	a is an element of set G
$\|A\|$	the number of elements in set A
n	the maximum number of documents owned by a data owner
m	the number of keyword fields of each document
W_i	the keyword set of ith document
C_i	the keyword ciphertext of ith document
Q	the queried keyword set
S_A	the set of document identifiers that authorized user A can access
k_{agg}^S	the aggregate key of document set S

- Computability: For $\forall\ u, v \in \mathcal{G}$, there is an efficient algorithm to calculate $e(u, v)$.

2.3. Complexity Assumptions

The security of our concrete KASE-CQ construction is based on complexity assumptions called the Divisible Computational Diffie-Hellman (DCDH) assumption, Diffie-Hellman Exponent (DHE) assumption, and decisional Bilinear Diffie-Hellman Exponent (BDHE) assumption. We recall the formal definitions of them as follows.

2.3.1. Divisible Computational Diffie-Hellman (DCDH) Assumption

DCDH Problem [2]: Given a 3-tuple $(g, g^a, g^b) \in \mathcal{G}^3$, where a, b are randomly chosen from $\mathcal{Z}_p$, compute $g^{\frac{a}{b}}$.

The advantage of algorithm $\mathcal{A}$ to solve the DCDH problem in $\mathcal{G}$ is ε if

$$Pr\left[\mathcal{A}(g, g^a, g^b) = g^{\frac{a}{b}}\right] \geq \varepsilon.$$

Definition 1. *The (t, ε)-DCDH assumption holds in $\mathcal{G}$ if there exists no t-time algorithm that can solve the DCDH problem with advantage greater than ε.*

Table 3: Abbreviations

Abbr.	Full Name	Abbr.	Full Name
SE	Searchable Encryption	IND-sDOC-CKA	Indistinguishability against Selective-Document Chosen Keyword Attack
SSE	Symmetric Searchable Encrytpion	EU-sDOC-CKA	Existential Unforgeability against Selective-Document Chosen Keyword Attack
PEKS	Public Key Encryption with Keyword Search	Fc-MKA-KSE	File-Centric Multi-Key Aggregate Keyword Searchable Encryption
KAC	Key-Aggregate Cryptosystem	IND-sF-CKA	Indistinguishability against Selective-File Chosen Keyword Attack
KASE	Key-Aggregate Searchabel Encryption	IND-sF-KGA	Indistinguishability against Selective-File Keyword Guessing Attack
KASE-CQ	Key-Aggregate Searchable Encryption Supporting Conjunctive Queries	ITA	Inside Trapdoor Attack
VSEKA	Verifiable Searchable Encryption with Aggregate Keys	DCDH	Divisible Computational Diffie-Hellman
ROM	Random Oracle Model	DHE	Diffie-Hellman Exponent
SM	Standard Model	BDHE	Bilinear Diffie-Hellman Exponent

2.3.2. Diffie-Hellman Exponent (DHE) Assumption

n-DHE Problem [16]: Given a tuple of $2n$ elements $(g, g^{\alpha}, g^{(\alpha^2)}, ..., g^{(\alpha^n)}, g^{(\alpha^{n+2})}, ..., g^{(\alpha^{2n})}) \in \mathcal{G}^{2n}$, compute $g^{(\alpha^{n+1})}$. For shorthand, when g and α are specified, we use g_i to denote $g^{(\alpha^i)}$.

The advantage of algorithm $\mathcal{A}$ to solve the n-DHE problem in $\mathcal{G}$ is ε if

$$Pr[\mathcal{A}(g, g_1, g_2, ..., g_n, g_{n+2}, .., g_{2n}) = g_{n+1}] \geq \varepsilon.$$

Definition 2. *The (t, ε, n)-DHE assumption holds in $\mathcal{G}$ if there exists no t-time algorithm that can solve the n-DHE problem with advantage greater than ε.*

2.3.3. Bilinear Diffie-Hellman Exponent (BDHE) Assumption

n-BDHE Problem [5]: Given a vector of $2n+1$ elements

$$(h, g, g^{\alpha}, g^{(\alpha^2)}, ..., g^{(\alpha^n)}, g^{(\alpha^{n+2})}, ..., g^{(\alpha^{2n})}) \in \mathcal{G}^{2n+1}$$

as input, output $e(g,h)^{(\alpha^{n+1})} \in \mathcal{G}_1$. As n-DHE Problem, we use g_i to represent $g^{(\alpha^i)}$.

The advantage of algorithm $\mathcal{A}$ to solve the n-BDHE problem in $\mathcal{G}$ is ε if

$$Pr[\mathcal{A}(h, g, g_1, g_2, ..., g_n, g_{n+2}, .., g_{2n}) = e(g_{n+1}, h)] \geq \varepsilon.$$

Decisional n-BDHE Problem: Given a vector of $2n+1$ elements

$$(h, g, g_1, g_2, ..., g_n, g_{n+2}, ..., g_{2n}) \in \mathcal{G}^{2n+1}$$

and $Z \in \mathcal{G}_1$, the decisional version of n-BDHE problem is to decide $Z \stackrel{?}{=} e(g_{n+1}, h)$. For shorthand, let $y_{g,\alpha,n} = (g_1, g_2, ..., g_n, g_{n+2}, ..., g_{2n})$.

The advantage of algorithm $\mathcal{A}$ with output $b \in \{0, 1\}$ to solve decisional n-BDHE problem is ε if

$$|Pr[\mathcal{A}(g, h, y_{g,\alpha,n}, e(g_{n+1}, h)) = 1] - Pr[\mathcal{A}(g, h, y_{g,\alpha,n}, Z) = 1]| \geq \varepsilon.$$

We denote the distribution on the left as T_{BDHE} and the distribution on the right as F_{BDHE}.

Definition 3. *The (t, ε, n)-(decisional) BDHE assumption holds in $\mathcal{G}$ if there exists no t-time algorithm that can solve the (decisional) n-DBHE problem with advantage greater than ε.*

3. Problem Formulation of KASE-CQ

In this section, we introduce a new cryptography primitive called *Key-Aggregate Searchable Encryption supporting Conjunctive Queries* (KASE-CQ).

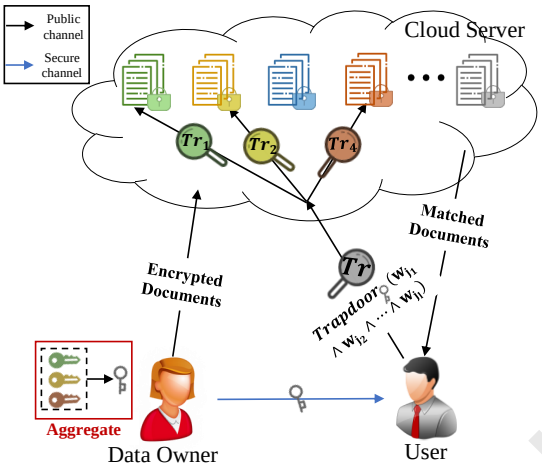


Figure 2: The System Model of KASE-CQ

We also propose the system model, system framework, and function and security requirements of the KASE-CQ.

3.1. System Model

The KASE-CQ system model is shown in Fig. 2. A KASE-CQ system consists of three types of entities: a data owner, users, and a cloud server. We introduce the detailed description of each entity as follows.

- Data Owner. The entity is fully credible. He is responsible for setting up the system to generate system parameters. He divides his documents into different classes and encrypts each document under his public key, the cloud server's public key and an identifier that indicates the document's class, then outsources the generated ciphertexts to the cloud server. He also uses his secret key to compute the aggregate key for the set of shared document classes and transmits it to the user through the secure channel.

- Users. After obtaining the aggregate key, this entity can generate the aggregate trapdoor based on his interested keywords and submit the trapdoor to the cloud server to issue the conjunctive search query. The entities

may collude to obtain as much secret information as possible by combining the aggregate keys they possess.

- Cloud Server. The entity is honest-but-curious, i.e. it honestly performs the protocols but is curious to spy out the valuable information. It stores the ciphertexts from the data owner. After getting the aggregate trapdoor from the user, the server adjusts the aggregate trapdoor to trapdoors for each different document class. Subsequently, it performs the ciphertext search operation and transmits the matched ciphertexts to the user.

3.2. System Framework

We consider the data owner storing encrypted documents on the untrusted cloud server. The maximum number of documents or document classes owned by a data owner is n. Suppose each document has m keyword fields, and different keyword fields have different keywords. For the ith document D_i, we use $W_i = \{w_{i,1}, w_{i,2}, ..., w_{i,m}\}$ to represent its keyword set, where $w_{i,j}$ is the jth keyword of D_i. $Q = \{w'_{j_1}, w'_{j_2}, ..., w'_{j_l}\}$ denotes the queried keyword set, where $j_i, 1 \leq i \leq l$ is the keyword field and $|Q| \leq m$. About document encryption, we can use the key-aggregate cryprosystem [9] to get the ciphertext c_i of each document D_i. This is not the focus of our research. We mainly focus on how to encrypt keywords so that the keyword ciphertexts can be searched. Therefore, in our algorithm, we will not dwell on the encryption of the document itself.

Definition 4. *(KASE-CQ) A KASE-CQ system contains the following eight algorithms:*

- $\boldsymbol{Setup(1^k, n)}$**.** *On input n and security parameter k, the data owner runs this algorithm to generate the public system parameter params. We implicitly assume that all other algorithms take params as input.*
- $\boldsymbol{KeyGen}$**.** *The data owner outputs his public-secret key pair (pk, sk).*
- $\boldsymbol{KeyGen_s}$ *The cloud server outputs its public-secret key pair (pk_s, sk_s).*

- ***Encrypt*(pk, sk, pk_s, i, W_i).** *On input W_i, (pk, sk), and pk_s, the data owner runs this algorithm to generate the keyword ciphertext $C_i = \{CW_i, \Delta_i\}$, where $CW_i = \{c_{i,1}, c_{i,2}, ..., c_{i,m}\}$ is obtained by encrypting each keyword in W_i and Δ_i is the auxiliary value. Finally, C_i is uploaded to the cloud.*

- ***Extract*(sk, S).** *The data owner runs this algorithm to generate the aggregate searchable encryption key for the user who has access to the selected document set. He inputs sk and a document identifier set S, then outputs aggregate key k_{agg}^S and transmits it to the user over the secure channel. Users who get the key have the keyword search right for documents of the ith class, where $i \in S$.*

- ***Trapdoor*(pk, k_{agg}^S, Q).** *The user runs this algorithm to generate an aggregate trapdoor Tr using aggregate key k_{agg}^S and queried keyword set Q, and then submits T_r and S to the cloud.*

- ***Adjust*(Tr, i, S).** *The cloud server runs this algorithm to adjust Tr to the trapdoor Tr_i of the document with identifier i belonging to S.*

- ***Test*(Tr_i, S, C_i, sk_s).** *The cloud server runs this algorithm. It takes the C_i and trapdoor Tr_i as input, and output 1 if the expression*

$$(w_{i,j_1} = w'_{j_1}) \wedge (w_{i,j_2} = w'_{j_2}) \wedge ... \wedge (w_{i,j_l} = w'_{j_l}) \tag{1}$$

holds; otherwise, output 0.

Example. For better understanding, we give an example to illustrate how our KASE-CQ system works. The security parameter is denoted as k, and its value can be set according to specific security requirements. We assume that a data owner has four documents (for simplicity, assume that a document class contains one document), and each document has three keyword fields, i.e., $n = 4$ and $m = 3$. Therefore, we have $W_1 = \{w_{1,1}, w_{1,2}, w_{1,3}\}$, $W_2 = \{w_{2,1}, w_{2,2}, w_{2,3}\}$, $W_3 = \{w_{3,1}, w_{3,2}, w_{3,3}\}$ and $W_4 = \{w_{4,1}, w_{4,2}, w_{4,3}\}$. The data owner generates the public system parameter *params* by running the

$Setup(1^k, 4)$ algorithm, and generates his public-secret key pair (pk, sk) by running the $KeyGen$ algorithm. The cloud server runs the $KeyGen_s$ algorithm to generate its public-secret key pair (pk_s, sk_s). Then, the data owner runs algorithms $Encrypt(pk, sk, pk_s, 1, W_1)$,..., $Encrypt(pk, sk, pk_s, 4, W_4)$ to generate keyword ciphertexts C_1,..., C_4 and uploads them to the cloud server. When the data owner wants to share the 1st, 3rd, and 4th documents to a user, that is $S = \{1, 3, 4\}$, he runs the $Extract(sk, \{1, 3, 4\})$ algorithm to generate an aggregate key $k_{agg}^{\{1,3,4\}}$ and transmits it to the user over the secure channel. With $k_{agg}^{\{1,3,4\}}$, the user can perform conjunctive keyword queries on the 1st, 3rd, and 4th documents. For example, when he wants to search for documents whose first keyword field is the keyword w_1' and the fourth keyword field is the keyword w_4', that is $Q = \{w_1', w_4'\}$, the user runs $Trapdoor(pk, k_{agg}^{\{1,3,4\}}, \{w_1', w_4'\})$ to generate the trapdoor Tr. Then, he submits Tr and $S = \{1, 3, 4\}$ to the cloud server. The server runs algorithms $Adjust(Tr, 1, \{1, 3, 4\})$, $Adjust(Tr, 3, \{1, 3, 4\})$, and $Adjust(Tr, 4, \{1, 3, 4\})$ to adjust Tr to the trapdoors Tr_1, Tr_3, and Tr_4 of the 1st document, 3rd document, and 4th document respectively. Finally, for $i = 1, 3, 4$, the server runs $Test(Tr_i, S, C_i, sk_s)$ to test whether the expression $(w_{i,1} = w_1') \wedge (w_{i,4} = w_4')$ is held, and if it is held, the corresponding encrypted document c_i is returned to the user.

3.3. Requirements of KASE-CQ

3.3.1. Function Requirements

Considering the motivation, a valid KASE-CQ scheme must meet the following two function requirements.

Definition 5 (Correctness)**.** *For any sets $S \subseteq \{1, 2, ..., n\}$ and $Q = \{w_{j_1}', w_{j_2}', ..., w_{j_l}'\}$ ($|Q| \leq m$), any identifier $i \in S$, $(pk, sk) \leftarrow KeyGen$, $(pk_s, sk_s) \leftarrow KeyGen_s$, $C_i \leftarrow Encrypt(pk, sk, pk_s, i, W_i)$, $k_{agg}^S \leftarrow Extract(sk, S)$, $Tr \leftarrow Trapdoor(pk, k_{agg}^S, Q)$, and $Tr_i \leftarrow Adjust(Tr, i, S)$, if the ith document contains the same keywords as the queried keyword set in corresponding keyword fields, then the algorithm $Test(Tr_i, S, C_i, sk_s)$ outputs 1.*

Definition 6 (Compactness)**.** *For any document identifier set* $S \subseteq \{1, 2, ..., n\}$*, the algorithms* $Extract(sk, S)$ *and* $Trapdoor(pk, k_{agg}^{S}, Q)$ *output a fixed-length aggregate key and a fixed-length aggregate trapdoor, respectively.*

3.3.2. Security Requirements

The cloud server is "honest-but-curious", meaning it runs algorithms correctly but obtains secret information from encrypted data whenever possible. Unauthorized users may collude to expose keywords in keyword ciphertexts and generate an aggregate key for a new document set utilizing known aggregate keys. The user may also carry out the inside trapdoor attack. Hence, a KASE-CQ scheme must meet the following security requirements, where $\mathcal{A}$ is the polynomial-time adversary and $\mathcal{C}$ is the challenger. Both $\mathcal{A}$ and $\mathcal{C}$ are given n and m.

1) **Ciphertext Privacy** : ensure that *Encrypt* does not disclose any information of relative keyword set to unauthorized attackers. We define the indistinguishability against selective-document chosen keyword attack (IND-sDOC-CKA) security model for ciphertext privacy. Security is defined by the game between $\mathcal{A}$ and $\mathcal{C}$ as below.

Init. $\mathcal{A}$ declares the document identifier $i^*(i^* \in [1, n])$ that he wants to be challenged.

Setup. The system parameter generation algorithm $Setup(1^k, n)$ is executed to generate system parameter $params$. The two key generation algorithms $KeyGen$ and $KeyGen_s$ are executed to generate public-secret key pairs (pk, sk) and (pk_s, sk_s) for data owner and server, respectively. Then, $\mathcal{C}$ sends $params$, pk, and pk_s to $\mathcal{A}$.

Phase1. $\mathcal{A}$ makes the following queries adaptively.

Aggregate key query (S): Once the document identifier set S is received, $\mathcal{C}$ runs $Extract(sk, S)$ to generate the aggregate key k_{agg}^{S} and sends it to $\mathcal{A}$. We constrain $i^* \notin S$.

Ciphertext query (W, i): Once the keyword set W and related document identifier i are received, $\mathcal{C}$ runs $Encrypt(pk, sk, pk_s, i, W)$ to generate the keyword

ciphertext C_i and sends it to $\mathcal{A}$.

Challenge. $\mathcal{A}$ outputs two distinct challenge keyword sets W_0 and W_1 ($|W_0| = |W_1| = m$). $\mathcal{C}$ randomly selects a bit c and computes challenge ciphertext $CT^* = Encrypt(pk, sk, pk_s, i^*, W_c)$, which is sent to $\mathcal{A}$.

We restrict $W_0 \neq W$, $W_1 \neq W$ when $i^* = i$, where (W, i) is for the ciphertext query.

Phase2. Same as in Phase 1, $\mathcal{C}$ responds to aggregate key queries and ciphertext queries. The restriction is that $(W, i) \neq (W_0, i^*)$ and $(W, i) \neq (W_1, i^*)$ for ciphertext query (W, i).

Guess. Finally, $\mathcal{A}$ outputs a bit c'. If $c' = c$, $\mathcal{A}$ wins.

We define the advantage of $\mathcal{A}$'s winning game as

$$Adv_{\mathcal{A}}^{IND-sDOC-CKA}(k) = |Pr[c' = c] - \frac{1}{2}|.$$

Definition 7 (IND-sDOC-CKA-Secure)**.** *A KASE-CQ scheme is $(t, q_k, q_c, \varepsilon)$-secure in IND-sDOC-CKA security model if $Adv_{\mathcal{A}}^{IND-sDOC-CKA}(k)$ is less than or equal to ε for any $\mathcal{A}$ after making q_k aggregate key queries and q_c ciphertext queries in time t.*

2) **Inside Trapdoor Attack Secure** : ensure that an inside attacker can't infer the aggregate key of the user who shares document access with him.

In 2018, Zhou et al. [40] showed an attack on the scheme in [10], which we call it the inside trapdoor attack (ITA). We briefly review the attack.

The trapdoor of a keyword w in [10] is $k_{agg}^S \cdot H'(w)$, where $H' : \{0,1\}^* \to \mathcal{G}$ is a hash function. $\mathcal{A}$ is an attacked user who has the right to access document set S_A, so the aggregate key is $k_{agg}^{S_A}$. $\mathcal{B}$ is an inside attacker who can access document set S_B. S_A and S_B satisfy $S_A \neq S_B$, $S_A \cap S_B \neq \emptyset$, and $S_A \nsubseteq S_B$. We assume $D \in (S_A \cap S_B)$ and D contains the keywords w_1, w_2, and w_3.

$\mathcal{A}$ submits the trapdoor $Tr = k_{agg}^{S_A} \cdot H'(w_*)$ of keyword w_* for querying, and the returned results contain document D. If $\mathcal{B}$ eavesdrops on the trapdoor Tr, he knows the key $k_{agg}^{S_A}$ must belong to set $K = \{k_1, k_2, k_3\} = \left\{\frac{Tr}{H'(w_1)}, \frac{Tr}{H'(w_2)}, \frac{Tr}{H'(w_3)}\right\}$. Therefore, $\mathcal{B}$ can use each k_i, in turn, to generate the

trapdoor of w_j ($j \neq i$) for keyword search, and then check whether k_i is $\mathcal{A}$'s aggregate key based on the search results. For example, $\mathcal{B}$ guesses $k_{agg}^{S_A} = k_1$, namely $w_* = w_1$, and computes $Tr_i = k_1 \cdot H'(w_i)$, i=2 or 3. After the cloud server receives Tr_i for search, if D is in search results, Tr_i is valid, which means $k_1 = k_{agg}^{S_A}$. Otherwise, Tr_i is invalid, and $\mathcal{B}$ continues to perform a similar operation on k_2.

So $\mathcal{B}$ can get $\mathcal{A}$'s aggregate key by guessing three times at most. Furthermore, after obtaining the aggregate key of $\mathcal{A}$, $\mathcal{B}$ can utilize the same strategy to obtain the aggregate keys of other users who shares document access with $\mathcal{A}$.

Definition 8 (ITA Secure)**.** *A KASE scheme is ITA secure if it can resist the above ITA attack, namely no inside attacker $\mathcal{A}$ can obtain other users' aggregate keys by performing an inside trapdoor attack.*

3) **Aggregate Key Unforgeability** : ensure that attackers can't generate an aggregate key for a document set containing an unauthorized document from the known aggregate keys. We define the existential unforgeability against selective-document chosen keyword attack (EU-sDOC-CKA) security model for aggregate key unforgeability. Security is defined by the game between $\mathcal{A}$ and $\mathcal{C}$ as below.

Init. $\mathcal{A}$ proclaims the document identifier $i^*(i^* \in [1, n])$ that he wants to be challenged.

Setup. The system parameter generation algorithm $Setup(1^k, n)$ is executed to generate system parameter $params$. The two key generation algorithms $KeyGen$ and $KeyGen_s$ are executed to generate public-secret key pairs (pk, sk) and (pk_s, sk_s) for data owner and server, respectively. Then, $\mathcal{C}$ sends $params$, pk, and pk_s to $\mathcal{A}$.

Query. $\mathcal{A}$ makes the aggregate key queries and ciphertext queries adaptively.

Aggregate key query (S): Once the document identifier set S is received, $\mathcal{C}$ runs $Extract(sk, S)$ to generate aggregate key k_{agg}^S and sends it to $\mathcal{A}$. We constrain $i^* \notin S$.

Ciphertext query (W, i): Once the keyword set W and related document identifier i are received, $\mathcal{C}$ runs $Encrypt(pk, sk, pk_s, i, W)$ to generate the keyword

ciphertext C_i and sends it to $\mathcal{A}$.

Forgery. $\mathcal{A}$ outputs an aggregate key $k_{agg}^{S^*}$ and wins if

- $k_{agg}^{S^*}$ is a valid aggregate key of document set S^*.
- $i^* \in S^*$.

The probability of $\mathcal{A}$ returning forged aggregate key is the advantage $Adv_{\mathcal{A}}^{EU-sDOC-CKA}(k)$ of winning the game.

Definition 9 (EU-sDOC-CKA-Secure)**.** *A KASE-CQ scheme is* $(t, q_k, q_c, \varepsilon)$*-secure in EU-sDOC-CKA security model if* $Adv_{\mathcal{A}}^{EU-sDOC-CKA}(k)$ *is less than or equal to* ε *for any* $\mathcal{A}$ *after making* q_k *aggregate key queries and* q_c *ciphertext queries in time* t*.*

4. THE PROPOSED KASE-CQ SCHEME

In this section, we first instantiate the proposed KASE-CQ primitive by providing a concrete construction for KASE-CQ. We then show that this construction satisfies the proposed function and security requirements.

4.1. Construction

1) $Setup(1^k, n) \rightarrow params$. On input n and security parameter k , the data owner initializes the system parameters:

- Generates a symmetric bilinear map group $\mathbb{PG} = (\mathcal{G}, \mathcal{G}_1, p, e(\cdot, \cdot))$, where p is the order of $\mathcal{G}$ and $\mathcal{G}_1$, and $2^k \leq p \leq 2^{k+1}$;
- Randomly chooses a generator g of $\mathcal{G}$ and $\alpha \in Z_p^*$, and calculates $g_i = g^{(\alpha^i)}$ for $i = \{1, 2, ..., n, n+2, ..., 2n\}$;
- Chooses a cryptographic hash function $H : \{0,1\}^* \rightarrow Z_p^*$.

Output the public system parameters

$$params = \{\mathbb{PG}, (g_1, g_2, ..., g_n, g_{n+2}, ..., g_{2n}), H\}.$$

2) $KeyGen \rightarrow (pk, sk)$. Randomly pick $\beta, \gamma \in Z_p^*$ and compute $v_1 = g^\beta$, $v_2 = g^\gamma$. Output the data owner's public-secret key pair (pk, sk):

$$pk = (v_1, v_2), sk = (\beta, \gamma).$$

3) $KeyGen_s \rightarrow (pk_s, sk_s)$. The cloud server randomly picks $\lambda \in Z_p^*$, computes $v = g^\lambda$, and outputs the public-secret key pair (pk_s, sk_s) as:

$$(pk_s, sk_s) = (v, \lambda).$$

4) $Encrypt(pk, sk, pk_s, i, W_i) \rightarrow C_i$. Input document identifier i and keyword set W_i, randomly choose $t, r \in Z_p^*$, and then generate the keyword ciphertext as follows:

- Generate the auxiliary value $\Delta_i = \{c_1, c_2, c_3, c_4, c_5\}$ by computing:

$$c_1 = g^t, c_2 = (v_2 \cdot g_i)^t, c_3 = e(g_1, g_n)^t, c_4 = g^{\frac{t}{\beta}}, c_5 = e(g, g)^{rt}.$$

- Encrypt each keyword in W_i to generate ciphertext set CW_i as follows:

$$CW_i = \{c_{i,j}\}_{j=1}^m = \left\{pk_s^{H(w_{i,j})+\beta r}\right\}_{j=1}^m.$$

Therefore, the output of this algorithm is $C_i = (\Delta_i, CW_i)$.

5) $Extract(sk, S) \rightarrow k_{agg}^S$. Given the document identifier subset $S \subseteq \{1, ..., n\}$, the aggregate key is calculated as

$$k_{agg}^S = \prod_{j \in S} g_{n+1-j}^\gamma.$$

Then, the data owner sends k_{agg}^S and S to the user securely, thus delegating the user the right to access documents in the set S.

6) $Trapdoor(pk, k_{agg}^S, Q) \rightarrow Tr$. Given the queried keyword set $Q = \left\{w'_{j_1}, w'_{j_2}, ..., w'_{j_l}\right\}$, generate an aggerate trapdoor $Tr = \{T_1, T_2, T_3\}$ for all documents related

to the key k_{agg}^S. Specifically, the user first randomly picks $s \in Z_p^*$. Then, the user calculates $Tr = \{T_1, T_2, T_3\}$ as follows:

$$T_1 = k_{agg}^S \cdot g^s,\ T_2 = \frac{v_1^s}{g^{\sum_{k=1}^{l} H(w'_{j_k})}},\ T_3 = \{j_k\}_{k=1}^{l}.$$

We can see that T_3 is the keyword field set of the queried keyword set. Finally, the Tr and S are submitted to the server.

7) $Adjust(Tr, i, S) \rightarrow Tr_i$. Upon receiving the trapdoor Tr and document identifier set S, the server can adjust Tr to the actual trapdoor $Tr_i = \{T_{i,1}, T_{i,2}, T_{i,3}\}$ for the document class with identifier $i \in S$ as:

$$T_{i,1} = T_1 \cdot \prod_{j \in S, j \neq i} g_{n+1-j+i},\ T_{i,2} = T_2,\ T_{i,3} = T_3.$$

That is, the server only needs to adjust T_1, while T_2 and T_3 are independent of the document being searched, so no adjustment is required.

8) $Test(Tr_i, S, C_i, sk_s) \rightarrow 1/0$. Once the trapdoor Tr_i is obtained, the server can locate the corresponding keyword ciphertext set $\{c_{i,j_k}\}_{k=1}^{l}$ based on the queried keyword field set $T_{i,3}$. Then, the server verifies the following equation:

$$\frac{e(T_{i,1}, c_1)}{e(\prod_{j \in S} g_{n+1-j}, c_2)} \stackrel{?}{=} \frac{e((\prod_{k=1}^{l} c_{i,j_k})^{\frac{1}{sk_s}} \cdot T_2, c_4)}{c_3 c_5^l}. \tag{2}$$

If the equation 2 holds, it indicates that expression 1 holds and therefore outputs 1, otherwise output 0.

Correctness: First, we have

$$\begin{aligned} &\frac{e(T_{i,1}, c_1)}{e(\prod_{j \in S} g_{n+1-j}, c_2)} \\ &= \frac{e(k_{agg}^S \cdot g^s \cdot \prod_{j \in S, j \neq i} g_{n+1-j+i}, g^t)}{e(\prod_{j \in S} g_{n+1-j}, (v_2 \cdot g_i)^t)} \\ &= \frac{e(k_{agg}^S, g^t) e(g^s, g^t) e(\prod_{j \in S, j \neq i} g_{n+1-j+i}, g^t)}{e(\prod_{j \in S} g_{n+1-j}, g^{\gamma t}) e(\prod_{j \in S} g_{n+1-j}, g_i^t)} \\ &= \frac{e(g, g)^{st}}{e(g_1, g_n)^t}. \end{aligned} \tag{3}$$

In addition, if Equation 1 holds, then

$$\begin{aligned}
&\frac{e((\prod_{k=1}^{l} c_{i,j_k})^{\frac{1}{sk_s}} \cdot T_2, c_4)}{c_3 c_5^l} \\
&= \frac{e(g^{\sum_{k=1}^{l} H(w'_{j_k}) + l\beta r} \cdot g^{\beta s - \sum_{k=1}^{l} H(w'_{j_k})}, g^{\frac{t}{\beta}})}{e(g_1, g_n)^t e(g,g)^{rtl}} \\
&= \frac{e(g^{l\beta r + \beta s}, g^{\frac{t}{\beta}})}{e(g_1, g_n)^t e(g,g)^{rtl}} \\
&= \frac{e(g,g)^{st}}{e(g_1, g_n)^t}
\end{aligned} \tag{4}$$

Therefore, combining Equations 3 and 4, we can see that if the aggregate key is valid, and the keywords of the document and the queried keywords are the same in the corresponding fields, then $\frac{e(T_{i,1}, c_1)}{e(\prod_{j \in S} g_{n+1-j}, c_2)} = \frac{e((\prod_{k=1}^{l} c_{i,j_k})^{\frac{1}{sk_s}} \cdot T_2, c_4)}{c_3 c_5^l}$.

Compactness: We show that the size of the aggregate key and aggregate trapdoor has nothing to do with $|S|$. For any S, the *Extract* algorithm always outputs an element of group $\mathcal{G}$. The output of the $Trapdoor$ algorithm consists of three parts. Both T_1 and T_2 are elements of group $\mathcal{G}$. Although the size of T_3 is related to the quantity of queried keywords, it has nothing to do with $|S|$. Therefore, given the queried keywords, $Trapdoor$'s output is also of fixed length for any S.

4.2. Security analysis

Theorem 1. *The KASE-CQ is $(t, q_k, q_c, \varepsilon)$-IND-sDOC-CKA secure if decisional (t', ε, n)-BDHE assumption holds, where $t' = t + \mathcal{O}(q_k + q_c)$.*

Proof. Assuming there is an adversary $\mathcal{A}$ that can break the IND-sDOC-CKA security model with advantage ε after making q_k aggregate key queries and q_c ciphertext queries, we can construct a simulator $\mathcal{B}$ to solve the decisional n-BDHE problem.

Let $g_i = g^{(\alpha^i)}$ for a generator $g \in \mathcal{G}$ and $\alpha \in Z_p$. Given as input a problem instance $(h, g, g_1, g_2, ..., g_n, g_{n+2}, ..., g_{2n}, Z)$ over the symmetric bilinear group $\mathbb{PG} = (\mathcal{G}, \mathcal{G}_1, p, e)$, $\mathcal{B}$ works as follows by running $\mathcal{A}$.

Init. Adversary $\mathcal{A}$ outputs a challenge document identifier $i^*(i^* \in [1, n])$.

Setup. $H : \{0,1\}^* \to Z_p^*$ is a hash function. $\mathcal{B}$ randomly chooses $\beta, \gamma', \lambda \in Z_p^*$ and computes public key $pk = (v_1, v_2') = (g^\beta, g^{\gamma'} g_{i^*}^{-1} = g^{\gamma' - \alpha^{i^*}})$, $pk_s = v = g^\lambda$. Therefore, $params = (\mathbb{PG}, (g_1, g_2, ..., g_n, g_{n+2}, .., g_{2n}), H)$, pk, and pk_s are available from the problem instance and chosen parameters. $\mathcal{B}$ sends $params$, pk, and pk_s to adversary $\mathcal{A}$.

Phase1.

- For an aggregate key query (S), since $i^* \notin S$, $\mathcal{B}$ can calculate

$$k_{agg}^S = (\prod_{j \in S} g_{n+1-j}^{\gamma'})(\prod_{j \in S} g_{n+1-j+i^*})^{-1} = \prod_{j \in S} g_{n+1-j}^{\gamma' - \alpha^{i^*}}.$$

  Therefore, k_{agg}^S is a valid aggregate key for S.

- For a ciphertext query (W, i), since $\mathcal{B}$ knows the params, public key v_2', v, and secret key β used to generate ciphertext, it always can respond to the valid ciphertext to $\mathcal{A}$.

Challenge. $\mathcal{A}$ outputs two distinct keyword sets W_0 and W_1 to be challenged. The restriction is that W_0 and W_1 over document i^* have not been queried. $\mathcal{B}$ randomly selects $r \in Z_p^*$ and a bit c and responds the challenge ciphertext

$$\begin{aligned} CT^* &= (c_1^*, c_2^*, c_3^*, c_4^*, c_5^*, CW^*) \\ &= (h, h^{\gamma'}, Z, h^{1/\beta}, e(g^r, h), \left\{ v^{H(w_i)+\beta r} \right\}_{w_i \in W_c}). \end{aligned}$$

Let $t = log_g h$. When $Z = e(g_{n+1}, h)$, we have

$$\begin{aligned} CT^* &= (g^t, g^{t\gamma'}, e(g_{n+1}, g)^t, g^{\frac{t}{\beta}}, e(g^r, g^t), \left\{ v^{H(w_i)+\beta r} \right\}_{w_i \in W_c}) \\ &= (g^t, (v_2' \cdot g_{i^*})^t, e(g_1, g_n)^t, g^{\frac{t}{\beta}}, e(g, g)^{rt}, \left\{ v^{H(w_i)+\beta r} \right\}_{w_i \in W_c}). \end{aligned}$$

Accordingly, CT^* is a valid challenge keyword set ciphertext for W_c.

Phase2. Same as in Phase 1. The restriction is that $(W, i) \neq (W_0, i^*)$ and $(W, i) \neq (W_1, i^*)$ for ciphertext query (W, i).

Guess. $\mathcal{A}$ outputs a bit c'. If $c' = c$, $\mathcal{B}$ outputs *true* meaning $Z = e(g_{n+1}, h)$.

Otherwise, it outputs $false$ meaning Z is random in $\mathcal{G}_1$.

This completes the simulation of interaction with $\mathcal{A}$. The advantage and time to solve the problem instance are calculated as follows.

If the problem instance is sampled from T_{BDHE}, simulation and real attack game are indistinguishable, and therefore $\mathcal{A}$ has probability $\frac{1}{2}+\varepsilon$ of guessing $c'=c$. If the problem instance is sampled from F_{BDHE}, $Pr[c'=c]=\frac{1}{2}$. Therefore, we have

$$\begin{aligned}&|Pr[\mathcal{A}(g,h,y_{g,\alpha,n},e(g_{n+1},h))=1]-Pr[\mathcal{A}(g,h,y_{g,\alpha,n},Z)=1]|\\=&(\frac{1}{2}+\varepsilon)-\frac{1}{2}\\=&\varepsilon.\end{aligned}$$

T_s is used to represent simulation time, then $T_s=\mathcal{O}(q_k+q_c)$, which depends on the aggregate key and ciphertext generation. So $\mathcal{B}$ will solve the n-BDHE problem in $t+\mathcal{O}(q_k+q_c)$-time with advantage ε.

The proof is completed. □

Theorem 2. *The KASE-CQ is ITA secure if DCDH assumption holds.*

Proof. Using the same notations as in the description of ITA, we use a concrete example to show that ITA is invalid for the proposed KASE-CQ scheme. $\mathcal{A}$ is an attacked user who has the right to access document set S_A and the aggregate key is $k_{agg}^{S_A}$. $\mathcal{B}$ is an inside attacker who can access document set S_B. S_A and S_B satisfy $S_A\neq S_B$, $S_A\cap S_B\neq\emptyset$, and $S_A\nsubseteq S_B$. We assume $D\in(S_A\cap S_B)$ and the keyword set of D is $W_D=\{w_1,w_2,w_3\}$.

Suppose $\mathcal{A}$ submits the trapdoor $Tr=\{T_1,T_2,T_3\}=\{k_{agg}^{S_A}\cdot g^s,\frac{v_1^s}{g^{\sum_{k=1}^{2}H(w_{j_k})}},\{j_k\}_{k=1}^{2}\}$ to the server and document D is included in search results. Therefore, $\mathcal{B}$ can know the queried keyword set corresponding to Tr. However, from T_1, we can see that $\mathcal{B}$ needs to get g^s to obtain the aggregate key $k_{agg}^{S_A}$ and even if $\mathcal{B}$ guesses the keyword set, he can only get $g^{\beta s}$ by computing $\frac{v_1^s}{g^{\sum_{k=1}^{2}H(w_{j_k})}}\cdot$

$g^{\sum_{k=1}^{2} H(w_{j_k})} = v_1^s = g^{\beta s}$. Therefore, if the DCDH assumption holds, it is intractable for $\mathcal{B}$ to compute the g^s from $g^{\beta s}$ and public key g^{β}.

The proof is completed. □

Theorem 3. *The KASE-CQ is $(t, q_k, q_c, \varepsilon)$-EU-sDOC-CKA secure if (t', ε, n)-DHE assumption holds, where $t' = t + \mathcal{O}(q_k + q_c)$.*

Proof. In the same way as Theorem 1, we prove by contradiction. Assuming there is an adversary $\mathcal{A}$ that can break the EU-sDOC-CKA security model with advantage ε after making q_k aggregate key queries and q_c ciphertext queries, then we can use $\mathcal{A}$ to construct a simulator $\mathcal{B}$ to break the DHE assumption.

Let $g_i = g^{(\alpha^i)}$ for a generator $g \in \mathcal{G}$ and $\alpha \in Z_p$. Given as input a problem instance $(g, g_1, g_2, ..., g_n, g_{n+2}, ..., g_{2n})$ over the group $\mathcal{G}$, $\mathcal{B}$ works as follows by running $\mathcal{A}$.

The first three algorithms **"Init"**, **"Setup"**, and **"Query"** are the same as **"Init"**, **"Setup"**, and **"Phase1"** in theorem 1 respectively, so we will not repeat them.

Forgery. $\mathcal{A}$ returns a forged aggregate key $k_{agg}^{S^*}$ of the document set S^*.

According to the definition of aggregate key and the above simulation process, we have

$$k_{agg}^{S^*} = \prod_{j \in S^*} g_{n+1-j}^{\gamma' - \alpha^{i^*}} = g_{n+1-i^*}^{\gamma' - \alpha^{i^*}} \cdot \prod_{j \in S^*, j \neq i^*} g_{n+1-j}^{\gamma' - \alpha^{i^*}}.$$

$\mathcal{B}$ computes

$$\begin{aligned}
&\frac{g_{n+1-i^*}^{\gamma'} \cdot \prod_{j \in S^*, j \neq i^*} g_{n+1-j}^{\gamma'} \cdot \prod_{j \in S^*, j \neq i^*} g_{n+1-j+i^*}^{-1}}{k_{agg}^{S^*}} \\
&= \frac{g_{n+1-i^*}^{\gamma'} \cdot \prod_{j \in S^*, j \neq i^*} g_{n+1-j}^{\gamma'} \cdot \prod_{j \in S^*, j \neq i^*} g_{n+1-j+i^*}^{-1}}{g_{n+1-i^*}^{\gamma'} \cdot (g_{n+1-i^*}^{\alpha^{i^*}})^{-1} \cdot \prod_{j \in S^*, j \neq i^*} g_{n+1-j}^{\gamma'} \cdot \prod_{j \in S^*, j \neq i^*} (g_{n+1-j}^{\alpha^{i^*}})^{-1}} \\
&= \frac{\prod_{j \in S, j \neq i^*} g_{n+1-j+i^*}^{-1}}{(g_{n+1-i^*}^{\alpha^{i^*}})^{-1} \cdot \prod_{j \in S^*, j \neq i^*} (g_{n+1-j}^{\alpha^{i^*}})^{-1}} \\
&= g^{(\alpha^{n+1})}
\end{aligned}$$

to get the solution of the DHE problem.

The simulation of interaction with $\mathcal{A}$ and the solution of the DHE problem are completed. The advantage and time to solve the problem instance are analyzed as follows.

From the above simulation and problem-solving process, it can be seen that as long as adversary $\mathcal{A}$ returns a valid forged aggregate key, $\mathcal{B}$ can get the solution of the DHE problem using the forged key. According to the assumption, $\mathcal{A}$ has the advantage ε to return the forged aggregate key, thus $\mathcal{B}$ can solve the DHE problem with advantage ε. Let T_s denote the cost of simulation, then $T_s = \mathcal{O}(q_k + q_c)$, which mainly depends on the aggregate key and ciphertext responses. Therefore, $\mathcal{B}$ will solve the DHE problem in $t + \mathcal{O}(q_k + q_c)$-time with advantage ε, that is, the (t', ε, n)-DHE assumption in $\mathcal{G}$ will be broken.

The proof is completed. □

5. PERFORMANCE

5.1. Theoretical Analysis

We compare the computational and communication cost of our scheme with that of KASE [10].

Table 4: Computation Comparison with [10]

	KASE [10]	**Ours**
Setup	$(2n-1)E_{\mathcal{G}} + (2n-2)M_{Z_p}$	$(2n-1)E_{\mathcal{G}} + (2n-2)M_{Z_p}$
KeyGen	$E_{\mathcal{G}}$	$2E_{\mathcal{G}}$
KeyGen$_s$	—	$E_{\mathcal{G}}$
Encrypt	$2mP + 2E_{\mathcal{G}} + 2m_{\mathcal{G}_1} + M_{\mathcal{G}} + mM_{\mathcal{G}_1}$	$2P + (m+3)E_{\mathcal{G}} + 2E_{\mathcal{G}_1} + M_{\mathcal{G}} + (m+1)M_{Z_p}$
Extract	$(\lvert S\rvert - 1)M_{\mathcal{G}} + E_{\mathcal{G}}$	$(\lvert S\rvert - 1)M_{\mathcal{G}} + E_{\mathcal{G}}$
Trapdoor	$lM_{\mathcal{G}}$	$2M_{\mathcal{G}} + 3E_{\mathcal{G}}$
Adjust	$l(\lvert S\rvert - 1)M_{\mathcal{G}}$	$(\lvert S\rvert - 1)M_{\mathcal{G}}$
Test	$2lP + (\lvert S\rvert - 1)M_{\mathcal{G}} + lM_{\mathcal{G}_1}$	$3P + (\lvert S\rvert + l - 1)M_{\mathcal{G}} + 3M_{\mathcal{G}_1} + E_{\mathcal{G}} + E_{\mathcal{G}_1}$

We compare the computational cost in Table 4. For the sake of comparison, we take into account only a few time-consuming operations, including bilinear pairing operation P, exponentiation operation $E_{\mathcal{Z}_p}$ in $\mathcal{Z}_p$, exponentiation operation $E_{\mathcal{G}}$ in $\mathcal{G}$, exponentiation operation $E_{\mathcal{G}_1}$ in $\mathcal{G}_1$, multiplication operation $M_{\mathcal{Z}_p}$ in $\mathcal{Z}_p$, multiplication operation $M_{\mathcal{G}}$ in $\mathcal{G}$, and multiplication operation $M_{\mathcal{G}_1}$ in $\mathcal{G}_1$. As in Table 1, in Table 4, we also use — to denote "not applicable".

In **Setup**, the two schemes perform the same operation except for the hash function selection; In **KeyGen**, KASE generates one public-secret key pair and our scheme generates two public-secret key pairs to meet the proposed security model definitions; In $\mathbf{KeyGen_s}$, our scheme generates the server's public-secret pair; In **Encrypt**, KASE needs two bilinear pairing operations for every keyword contained in the document, so its bilinear pairing operations are proportional to the number of keywords fields, while our scheme merely has two bilinear pairing operations in total; In **Extract**, the two schemes need to take the same actions to generate the aggregate key; In **Trapdoor**, for conjunctive query, KASE needs to generate separate trapdoor components for each keyword, and our scheme merely calculates a whole trapdoor for all queried keywords; In **Adjust**, KASE adjusts the trapdoor of each keyword, while our scheme only needs to adjust T_1; In **Test**, our scheme is more efficient, because KASE searches for each keyword in queried keyword set and find the intersection of the search results to obtain the target documents.

Table 5: Communication Comparison with [10]

	KASE [10]	**Ours**
Ciphertext (Data Owner→Cloud Server)	$2\|\mathcal{G}\| + m\|\mathcal{G}_1\|$	$(m+3)\|\mathcal{G}\| + 2\|\mathcal{G}_1\|$
Aggregate Key (Data Owner→User)	$\|\mathcal{G}\|$	$\|\mathcal{G}\|$
Trapdoor (User→Cloud Server)	$l\|\mathcal{G}\| + l\|k\|$	$2\|\mathcal{G}\| + l\|k\|$

"$|\mathcal{G}|$": the bit-length of an element in group $\mathcal{G}$;
"$|\mathcal{G}_1|$": the bit-length of an element in group $\mathcal{G}_1$;
"$|k|$": the length of security parameter k.

In Table 5, we show the communication comparison. The **Ciphertext** size and **Aggregate Key** size of our scheme are almost the same as KASE's. But for **Trapdoor**, because our scheme supports conjunctive keyword search and KASE needs to search each queried keyword individually, the size of our scheme is much smaller than KASE.

5.2. Practical Analysis

We use C programming language and Paring Based Cryptography (PBC) Library to calculate the system running time and select Type-A pairing to perform the specific algorithm. We set the security parameter to 512 bits. The whole experiment is run on Ubuntu 20.04.1 LTS operating system and is implemented by GCC 9.3.0 compile on a Linux Server with AMD Ryzen 5 4600H with Radeon Graphics CPU@ 3.00 GHz and 2GB memory. The documents and source codes of KASE-CQ are on GitHub https://github.com/jinlu06/KASE-CQ.

System running time comparison is shown in Fig. 3. Fig. 3 (a) and Fig. 3 (c) suggest that the running time in KASE **Setup** and **Extract** are nearly the same as in our scheme, respectively. And the time overhead in **Setup** and **Extract** has a linear relation with the maximum number of documents and the number of shared documents, respectively. These are all consistent with theoretical analysis. Fig. 3 (b) gives the time spent in **Encrypt** when $n = 1000$, indicating that our scheme needs much less computational cost. From Fig. 3 (d), the **Trapdoor** running time of our scheme is constant, but the **Trapdoor** time of KASE grows linearly with the amount of queried keywords. In Fig.3 (e) and Fig. 3 (f), we demonstrate the sum of the computational cost in **Adjust** and **Test**. Since both **Adjust** and **Test** are algorithms that need to run for the server to **Search**, we test the total running time of these two algorithms. In Fig. 3 (e), we fix $|S| = 100$, we notice that the **Search** time of KASE is affected by the number of queried keywords, and increases linearly with l increases, while the time cost of our scheme remains almost unchanged. In Fig.3 (f), we fix $l = 2$ to test the effect of $|S|$ on **Search** efficiency. The performance of our scheme is superior to KASE.

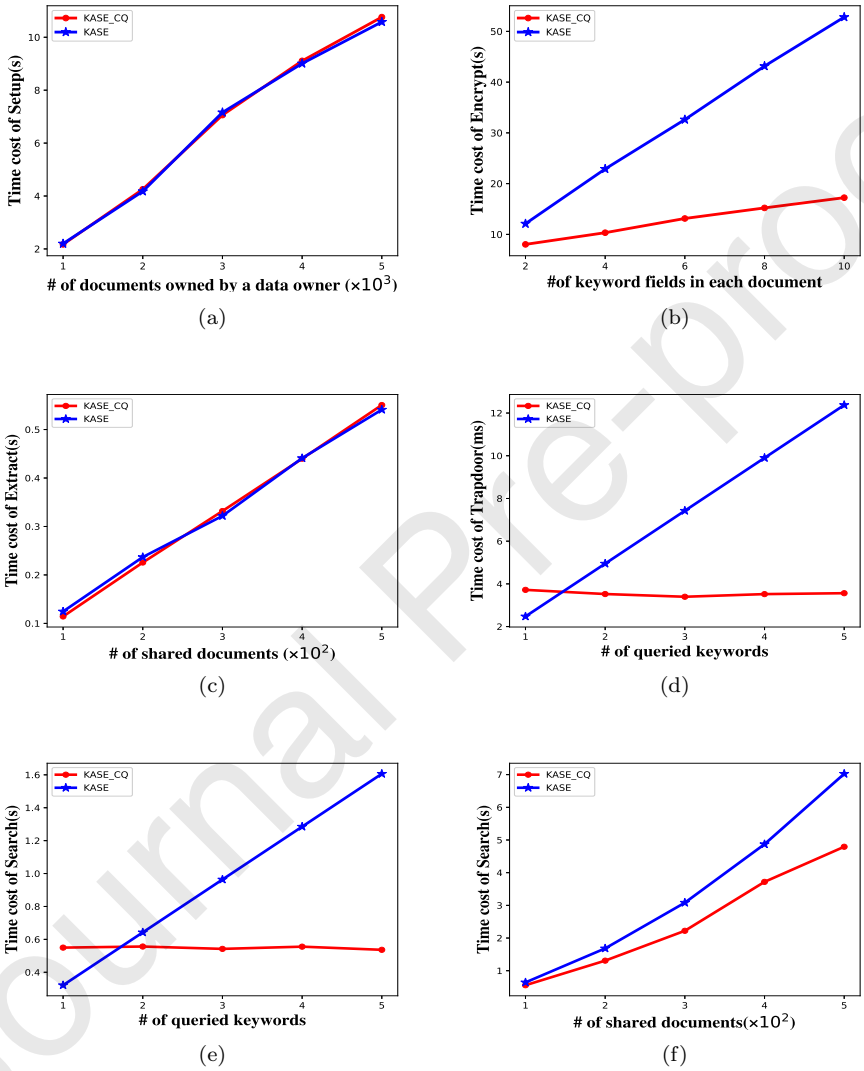


Figure 3: Time cost Comparison

In short, our scheme achieves stronger security guarantees (IND-sDOC-CKA, EU-sDOC-CKA, and ITA secure) in SM, a more practical function (conjunctive query), and the performance is more efficient.

6. Conclusion

Key-aggregate searchable encryption is a useful tool for searching and sharing encrypted data. Considering the limitations of existing KASE systems, in this paper, we proposed the key-aggregate searchable encryption supporting conjunctive queries (KASE-CQ) framework and its security notions IND-sDOC-CKA and EU-sDOC-CKA. We also designed a concrete KASE-CQ construction. This construction supports conjunctive keyword search, satisfies the proposed security notions in SM, and is secure against ITA. Additionally, the performance analysis shows the scheme's efficiency.

In the future, we will try to investigate other expressive searches, such as disjunctive search, fuzzy keyword search, wildcard search, etc. In addition, revocation functionality is vital in any data sharing system because some users may exit. Therefore, we will also work on the revocation of the delegated search for KASE.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (No.62072276, No.61772311).

References

[1] Lucas Ballard, Seny Kamara, and Fabian Monrose. Achieving efficient conjunctive keyword searches over encrypted data. In *International conference on information and communications security*, pages 414–426. Springer, 2005.

[2] Feng Bao, Robert H Deng, and Huafei Zhu. Variations of diffie-hellman problem. In *Information and Communications Security: 5th International Conference, ICICS 2003, Huhehaote, China, October 10-13, 2003. Proceedings 5*, pages 301–312. Springer, 2003.

[3] Burton H Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970.

[4] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *International conference on the theory and applications of cryptographic techniques*, pages 416–432. Springer, 2003.

[5] Dan Boneh, Craig Gentry, and Brent Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In *Annual international cryptology conference*, pages 258–275. Springer, 2005.

[6] Ning Cao, Shucheng Yu, Zhenyu Yang, Wenjing Lou, and Y Thomas Hou. Lt codes-based secure and reliable cloud storage service. In *2012 Proceedings IEEE INFOCOM*, pages 693–701. IEEE, 2012.

[7] David Cash, Stanislaw Jarecki, Charanjit Jutla, Hugo Krawczyk, Marcel-Cătălin Roşu, and Michael Steiner. Highly-scalable searchable symmetric encryption with support for boolean queries. In *Annual cryptology conference*, pages 353–373. Springer, 2013.

[8] Yan-Cheng Chang and Michael Mitzenmacher. Privacy preserving keyword searches on remote encrypted data. In *International conference on applied cryptography and network security*, pages 442–455. Springer, 2005.

[9] Cheng-Kang Chu, Sherman SM Chow, Wen-Guey Tzeng, Jianying Zhou, and Robert H Deng. Key-aggregate cryptosystem for scalable data sharing in cloud storage. *IEEE transactions on parallel and distributed systems*, 25(2):468–477, 2013.

[10] Baojiang Cui, Zheli Liu, and Lingyu Wang. Key-aggregate searchable encryption (kase) for group data sharing via cloud storage. *IEEE Transactions on computers*, 65(8):2374–2385, 2015.

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

[12] Eu-Jin Goh et al. Secure indexes. *IACR Cryptol. ePrint Arch.*, 2003:216, 2003.

[13] Philippe Golle, Jessica Staddon, and Brent Waters. Secure conjunctive keyword search over encrypted data. In *International conference on applied cryptography and network security*, pages 31–45. Springer, 2004.

[14] Cheng Guo, Ningqi Luo, Md Zakirul Alam Bhuiyan, Yingmo Jie, Yuanfang Chen, Bin Feng, and Muhammad Alam. Key-aggregate authentication cryptosystem for data sharing in dynamic cloud storage. *Future Generation Computer Systems*, 84:190–199, 2018.

[15] Cheng Guo, Ruhan Zhuang, Yingmo Jie, Yizhi Ren, Ting Wu, and Kim-Kwang Raymond Choo. Fine-grained database field search using attribute-based encryption for e-healthcare clouds. *Journal of medical systems*, 40(11):1–8, 2016.

[16] Javier Herranz, Fabien Laguillaumie, Benoît Libert, and Carla Rafols. Short attribute-based signatures for threshold predicates. In *Cryptographers' Track at the RSA Conference*, pages 51–67. Springer, 2012.

[17] Yinghui Huang, Wenting Shen, Jing Qin, and Huiying Hou. Privacy-preserving certificateless public auditing supporting different auditing frequencies. *Computers & Security*, 128:103181, 2023.

[18] Seny Kamara and Kristin Lauter. Cryptographic cloud storage. In *International Conference on Financial Cryptography and Data Security*, pages 136–149. Springer, 2010.

[19] Tong Li, Zheli Liu, Ping Li, Chunfu Jia, Zoe L Jiang, and Jin Li. Verifiable searchable encryption with aggregate keys for data sharing in outsourcing storage. In *Australasian Conference on Information Security and Privacy*, pages 153–169. Springer, 2016.

[20] Jinlu Liu, Bo Zhao, Jing Qin, Xi Zhang, and Jixin Ma. Multi-keyword ranked searchable encryption with the wildcard keyword for data sharing in cloud computing. *The Computer Journal*, 2021.

[21] Zheli Liu, Tong Li, Ping Li, Chunfu Jia, and Jin Li. Verifiable searchable encryption with aggregate keys for data sharing system. *Future Generation Computer Systems*, 78:778–788, 2018.

[22] Zhenhua Liu and Yaohui Liu. Verifiable and authenticated searchable encryption scheme with aggregate key in cloud storage. In *2018 14th International Conference on Computational Intelligence and Security (CIS)*, pages 421–425. IEEE, 2018.

[23] Peter Mell, Tim Grance, et al. The nist definition of cloud computing. 2011.

[24] Brice Minaud and Michael Reichle. Dynamic local searchable symmetric encryption. In *Advances in Cryptology–CRYPTO 2022: 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15–18, 2022, Proceedings, Part IV*, pages 91–120. Springer, 2022.

[25] Dong Jin Park, Kihyun Kim, and Pil Joong Lee. Public key encryption with conjunctive field keyword search. In *International Workshop on Information Security Applications*, pages 73–86. Springer, 2004.

[26] Sikhar Patranabis, Yash Shrivastava, and Debdeep Mukhopadhyay. Provably secure key-aggregate cryptosystems with broadcast aggregate keys for online data sharing on the cloud. *IEEE Transactions on Computers*, 66(5):891–904, 2016.

[27] Darren Quick and Kim-Kwang Raymond Choo. Dropbox analysis: Data remnants on user machines. *Digital Investigation*, 10(1):3–18, 2013.

[28] Kui Ren, Cong Wang, and Qian Wang. Security challenges for the public cloud. *IEEE Internet computing*, 16(1):69–73, 2012.

[29] Wenting Shen, Jia Yu, Ming Yang, and Jiankun Hu. Efficient identity-based data integrity auditing with key-exposure resistance for cloud storage. *IEEE Transactions on Dependable and Secure Computing*, 2022.

[30] Jiann-Cherng Shieh, Cheng-Chi Lee, and Shu-Yu Lee. A dynamic key aggregate cryptosystem in cloud environment. In *2015 IIAI 4th International Congress on Advanced Applied Informatics*, pages 73–78. IEEE, 2015.

[31] Ashish Singh and Kakali Chatterjee. Cloud security issues and challenges: A survey. *Journal of Network and Computer Applications*, 79:88–115, 2017.

[32] Luis M Vaquero, Luis Rodero-Merino, Juan Caceres, and Maik Lindner. A break in the clouds: towards a cloud definition, 2008.

[33] Xuqi Wang, Yu Xie, Xiangguo Cheng, and Zhengtao Jiang. An efficient key-aggregate keyword searchable encryption for data sharing in cloud storage. In *2019 IEEE Globecom Workshops (GC Wkshps)*, pages 1–6. IEEE, 2019.

[34] Z. Wang. Provably secure key-aggregate cryptosystems with auxiliary inputs for data sharing on the cloud. *Future generation computer systems*, 93(APR.):770–776, 2019.

[35] Haining Yang, Ye Su, Jing Qin, and Huaxiong Wang. Privacy-preserving outsourced inner product computation on encrypted database. *IEEE Transactions on Dependable and Secure Computing*, 2020.

[36] Lisha Yao, Jian Weng, Anjia Yang, Xiaojian Liang, Zhenghao Wu, Zike Jiang, and Lin Hou. Scalable cca-secure public-key authenticated encryption with keyword search from ideal lattices in cloud computing. *Information Sciences*, 624:777–795, 2023.

[37] PENG Yong, ZHAO Wei, XIE Feng, Zhong-hua DAI, Gao Yang, and Dongqing CHEN. Secure cloud storage based on cryptographic techniques. *The Journal of China Universities of Posts and Telecommunications*, 19:182–189, 2012.

[38] Hongjie Zhang, Shengke Zeng, and Jiali Yang. Backward private dynamic searchable encryption with update pattern. *Information Sciences*, 624:1–19, 2023.

[39] Xi Zhang, Bo Zhao, Jing Qin, Wei Hou, Ye Su, and Haining Yang. Practical wildcard searchable encryption with tree-based index. *International Journal of Intelligent Systems*, 2021.

[40] Rang Zhou, Xiaosong Zhang, Xiaojiang Du, Xiaofen Wang, Guowu Yang, and Mohsen Guizani. File-centric multi-key aggregate keyword searchable encryption for industrial internet of things. *IEEE Transactions on Industrial Informatics*, 14(8):3648–3658, 2018.

[41] Binrui Zhu, Jiameng Sun, Jing Qin, and Jixin Ma. A secure data sharing scheme with designated server. *Secur. Commun. Networks*, 2019:4268731:1–4268731:16, 2019.