IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 27, 2019, accepted January 10, 2020, date of publication January 20, 2020, date of current version February 7, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2967793

Key-Aggregate Searchable Encryption, Revisited: Formal Foundations for Cloud Applications, and Their Implementation

MASAHIRO KAMIMURA[1], NAOTO YANAI[1], (Member, IEEE), SHINGO OKAMURA[2], (Member, IEEE), AND JASON PAUL CRUZ[1], (Member, IEEE)

[1]Graduate School of Information Science and Technology, Osaka University, Suita 565-0871, Japan
[2]National Institute of Technology, Nara College, Nara 639-1080, Japan

Corresponding author: Naoto Yanai (yanai@ist.osaka-u.ac.jp)

This work was supported in part by the Japan Society for the Promotion of Science KAKENHI Number 18K18049, in part by the Innovation Platform for Society 5.0 at MEXT, and Secom Science and Technology Foundation.

ABSTRACT In the use of a cloud storage, sharing of data with efficient access control is an important requirement in addition to data security and privacy. Cui et al. (IEEE Trans. on Comp. 2016) proposed *key-aggregate searchable encryption (KASE)*, which allows a data owner to issue an *aggregate key* that enables a user to search in an authorized subset of encrypted files by generating an encrypted keyword called *trapdoor*. While the idea of KASE is elegant, to the best of our knowledge, its security has never been discussed formally. In this paper, we discuss the security of KASE formally and propose provably secure schemes. We first introduce our provably secure scheme, named *first construction*, with respect to encrypted files and aggregate keys in a single-server setting. In comparison with the scheme of Cui et al., the first construction is secure without increased computational costs. Then, we introduce another provably secure scheme, named *main construction*, with respect to trapdoors in a two-server setting. The main construction guarantees the privacy of a search, encrypted files, and aggregate keys. Considering 5,000 encrypted keywords, the first construction can finish search within three seconds and the main construction can finish search within six seconds.

INDEX TERMS Key-aggregate searchable encryption, searchable encryption, data sharing, provable security.

I. INTRODUCTION

A. BACKGROUND

A cloud storage service provides a solution for storing, accessing, and sharing files over the Internet. However, such a service may be vulnerable and stored data may be leaked without the permission or knowledge of the data owners. To prevent data leakage, data owners would want to encrypt their files before uploading them to a cloud storage. However, simply encrypting files makes searching for specific files inefficient because a user would need to download and decrypt all files and then check which files are needed. Moreover, to maximize the capabilities of the cloud and features of a cloud storage, data owners should also be able to share their files to intended users. Searchable encryption [1] allows users to search over encrypted data with a chosen keyword without decryption of the encrypted data. Searchable encryption is suitable for storage of data even in a vulnerable cloud storage service. In particular, even if an adversary exploits control of a cloud storage and leaks stored data, the stored data remain safe and protected by encryption. In recent years, several researchers [2]–[4] created schemes that authorize other users to search over encrypted data, therefore allowing data owners to share their encrypted data in the multi-user setting.

On this background, this paper aims to introduce a scheme that authorizes search over encrypted data, i.e., the searchability, *efficiently*. Suppose that a data owner who owns a set of original data can issue a key that enables other users to search in a subset of its encrypted data without decryption. In such setting, the following features are desirable: (1) the data owner issues only a *single short key* that is independent of

The associate editor coordinating the review of this manuscript and approving it for publication was Fan Zhang.

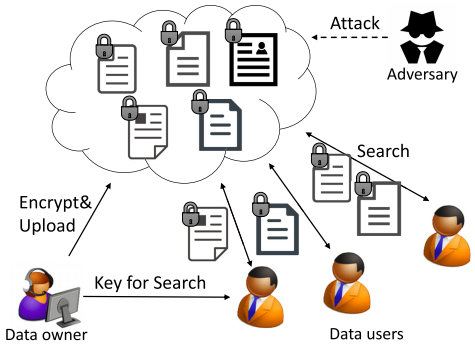

FIGURE 1. Overview of key-aggregate searchable encryption (KASE).

both the number of encrypted data and the number of users the encrypted data will be shared to, (2) and the encrypted data can be shared with users *without changing or reproducing the encrypted data*. These features can make the operations of a cloud storage service efficient because the keys and encrypted data become easier to manage for both a data owner and users. We note that these *features are not implied in previous systems,* namely, multi-user searchable encryption [5]–[8] and multi-key searchable encryption [2], [4], which do not include efficient management as described above. Therefore, achieving the features described above is a non-trivial problem.

Cui et al. [9] proposed *key-aggregate searchable encryption (KASE)* for the underlying purpose. An overview of KASE is shown in Figure 1. In KASE, a data owner issues aggregate keys that allow data users to search over authorized data only, i.e., data users generate trapdoors to search over encrypted data. The data sizes of ciphertexts and aggregate keys are independent of the number of data users the ciphertexts will be shared to, and the data size of ciphertexts is independent of the number of users. Therefore, KASE can improve the efficiency of the operations of a cloud storage service under the problem setting described above.

However, Cui et al. did not provide formal definitions of the security of KASE and its security proofs. Moreover, Kiayias et al. [3] introduced an attack against the scheme of Cui et al. in which encrypted keywords in ciphertexts are distinguishable for an adversary. To the best of our knowledge, no KASE scheme with formal security definitions and proofs has been introduced, even in subsequent works [10], [11], making it an open problem.

B. OUR CONTRIBUTIONS

In this paper, we propose KASE schemes with provable security.[1] To the best of our knowledge, our proposed schemes are the first provably secure constructions. We also define a syntax and its security formally. We note that constructing a provably secure KASE scheme is *non-trivial*. As will be described in Section III-D in detail, KASE requires a data owner to control the searchability via only a single short key and without creating different ciphertexts of the same keyword for different users. The algebraic structures that can be used to construct a KASE scheme are limited by the use of the single short key and keeping encrypted keywords.

In this paper, we introduce two provably secure KASE schemes named *first construction* and *main construction*. To create the schemes, we focus on the mathematical features of the cryptographic primitives used in constructing KASE and combine the existing instantiations of these cryptographic primitives. As will be described in Section IV-A in detail, some instantiations of broadcast encryption [12], [13] and aggregate signatures [14] are generally combined as foundations.

The first construction achieves keyword privacy, which guarantees the confidentiality of encrypted keywords, and aggregate key unforgeability, which authorizes the searchability of keywords from a data owner to users. In addition to keyword privacy and aggregate key unforgeability, the main construction also achieves trapdoor privacy, which guarantees the privacy of search for the users, by using secret sharing [15].

The computational cost and the size of ciphertexts in the first construction are identical to those of the scheme by Cui et al. [9], but the first construction is provably secure. We performed experiments by considering 5,000 keywords, and the results show that our two constructions can encrypt all keywords within one second, the first construction can perform search within three seconds, and the main construction can perform search within six seconds. We leave the construction of a generic scheme based on any instantiation of the primitives we used as an open problem. We have also published our source codes (https://github.com/naotoyanai/kase).

C. POTENTIAL APPLICATIONS

KASE has many potential applications and we consider that it can generally be applied in systems that require the storage of sensitive data that will be shared to multiple users. As specific examples, we describe cloud-assisted content-sharing networks with cryptography [16] and privacy-preserving authenticated communication in smart home environment [17] below.

Content sharing networks are networks that are scalable in terms of the number of users, storage size, and network bandwidth. Cloud-assisted content sharing networks with cryptography mainly aim to enable both a service provider and users to control the privacy of data flexibly and scalably. According to Wu et al. [16], cloud-assisted content sharing networks with cryptography allows: (1) an individual to freely produce any number and any kind of online media, such as texts, images, and videos; (2) an individual to grant any access to his/her media to anyone at any time; (3) an individual to reveal a large number of attributes (e.g., age, address, and gender), some of which can be dynamic; and (4) an individual to share contents using various devices and bandwidths, and hence demand different access privileges for the same media. Wu et al. proposed an instanton of cloud-assisted content sharing networks with

cryptography by utilizing attribute-based encryption [18]. However, attribute-based encryption does not provide the searchability for ciphertexts, and hence the content of ciphertexts cannot be searched without decryption. This limitation is inefficient and may not be appealing to users because users take a long time to decrypt all of the content of ciphertexts. Moreover, attribute-based encryption requires a trusted third-party to generate secret keys, creating a potential single point of failure in the entire system. Furthermore, the number of secret keys increases with the number of attributes, and thus the size of a storage and management costs of the keys also increase with the number of users and the management of authorization for content sharing. In contrast, KASE provides searchability of contents of ciphertexts and a fixed size of keys for both a data owner and users, solving the problems of cloud-assisted content-sharing networks described above. Therefore, KASE is desirable for use in cloud-assisted content-sharing networks.

The example of service utilizing content sharing networks is the movie contents sharing service. Consider for example a scenario where customers (data users) want to watch movies (contents) that are managed by a company (data owner). The company gives each user a single aggregate key depending on the customers' interests. If a customer is interested in ''Documentary'' and ''Sci-Fi'' genres, then the company generates a single aggregate key with the indexes of Documentary and Sci-Fi and gives this aggregate key to the customer. Using this aggregate key, the customer can search for Documentary and Sci-Fi movies on the company's encrypted contents and watch them. As a concrete example, the service of Amazon Prime Video in Japan offers about 1,000 video contents for each genre, e.g., 955 documentary movies and 303 Sci-Fi movies. Our experimental results show that our KASE schemes can search thousands of data within a few seconds, and therefore we imagine that our KASE schemes can be used effectively in searching for movies in the content sharing network example above. In general, the number of keys increases as the number of genres the customers are interested in increases. Nevertheless, in our KASE schemes, the company only needs to give a single aggregate key regardless of the number of genres, and therefore the management of customers' keys is easy and efficient.

The privacy-preserving authenticated communication in smart home environment by Poh et al. [17] is an application where a user can securely utilize smart devices that accumulate private information, such as sleeping patterns and medical information. Poh et al. realized such an application in the single-user setting by integrating searchable encryption and authenticated key-establishment protocol. However, smart devices are continuously becoming more popular, and thus the use of smart devices in the multi-user setting should be considered. For example, in a case with a large number of users, e.g., employees in a workplace, each employee has a separated data access per device. KASE can be used to control access of each employee to devices and their data by the use of a single key. Thus, a more efficient and attractive usage of device management can be expected with the use of KASE.

D. RELATED WORKS

Following the original KASE proposed by Cui et al. [9], many KASE schemes have been proposed in [10], [11], [19], [20], [21]. We note that the constructions of the schemes proposed in [10], [11] are essentially the same as the construction by Cui et al. and their security have never been proven. Liu et al. [11] introduced the verifiability of search results, and Li et al. [10] discussed situations with multiple data owners. Yao et al. [19] proposed a lattice-based KASE scheme, but they did not discuss its security proofs. Wang et al. [20] proposed a scheme wherein an adversary cannot generate a new aggregate key by combining some aggregate keys, but they also did not prove the security of their scheme. Meanwhile, Mukti et al. [21] proposed KASE with multiple data owners and discussed its security formally. Unfortunately, their definitions are incomplete and their proofs are wrong. In particular, in their syntax, any keyword can be searched for any documents, i.e., even in unauthorized documents, as long as the keyword is included as a part of ciphertexts. Moreover, in their proposed construction, the bilinear maps are nested for the test algorithm, which is an unworkable setting for bilinear maps. In addition, the distribution of the simulated secret keys and public keys in their proofs is different from that of their proposed algorithms.

As a special research alleviating the conditions of KASE, Zhou et al. [22] proposed a searchable and secure scheme in the situation where remote sensor devices encrypt data. This scheme individually changes the key for aggregate key issuance and the key for data encryption. However, this feature raises an issue of increased key management for a data owner. Patranabis et al. [23] also considered variants of KASE, but their scheme has no search delegation to other users. Therefore, the situation that KASE normally handles is different, so it can be said that it is different from the problem dealt with in this paper.

In a framework of conventional searchable encryption, the multi-user setting [5]–[8], [24]–[27] and the multi-key setting [2], [4], [28] have been known to control the searchability for each document. However, an efficient control of the searchability, such as issuance of aggregate keys which is one of the main problems in KASE, is out of the scope of such settings.

Searchable attribute-based encryption (SABE) [29]–[33] is an encryption scheme similar to KASE. SABE is a searchable encryption scheme in which documents corresponding to attributes are searchable for users who own secret keys of the attributes. While SABE provides searchability along with attributes for each document, the size of secret keys depends on the number of attributes, i.e., the key size is linear. This problem in SABE is a different problem from KASE.

As additional related works, key-aggregate cryptosystems [34]–[36] outsource the decryption of data.

Among them, Patranabis et al. [36] showed a provably secure scheme. However, the searchability for ciphertexts is out of the scope of these works. Since the searchability is not implied generally, constructing a provably secure KASE remains an open problem.

For further security analysis, recent works [37], [38] have proposed several attacks against searchable encryption. Zhang et al. [37] proposed the file injection attack against the query privacy of single-keyword and conjunctive searchable encryption schemes. In their attack, a server (adversary) can extract information from the user's queries with very few injected files. Ning et al. [38] proposed the passive attacks against symmetric searchable encryption schemes. Although these attacks are out of the scope of provable security framework, protecting KASE from these attacks is desirable in a practical scenario. We plan to study these attacks as future work.

1) PAPER ORGANIZATION

The rest of this paper is organized as follows. The mathematical background to understanding this work is described in Section II, and then a definition of KASE and the technical difficulty in constructing a KASE scheme is presented in Section III. The main idea to overcome the difficulty and the proposed KASE schemes are discussed in Section IV, and then their security proofs are presented in Section V. Implementation, evaluation, and analysis are discussed in Section VI. Finally, the conclusion and future direction are presented in Section VII.

II. PRELIMINARIES

In this section, we present the background on groups with bilinear maps and their security assumptions.

A. BILINEAR MAPS

The proposed schemes are based on bilinear maps and bilinear groups. In this paper, we recall the standard description defined in [39]. Here, let $\mathbb{G}$, $\mathbb{H}$, and $\mathbb{G}_T$ be groups with the same prime order p. We then define bilinear groups $(\mathbb{G}, \mathbb{H}, \mathbb{G}_T)$ and bilinear maps $e : \mathbb{G} \times \mathbb{H} \to \mathbb{G}_T$ as follows:

1) Bilinear groups $(\mathbb{G}, \mathbb{H}, \mathbb{G}_T)$ are two cyclic groups with a prime order p.
2) $g \in \mathbb{G}$ and $h \in \mathbb{H}$ are generators of $\mathbb{G}$ and $\mathbb{H}$, respectively.
3) A bilinear map $e : \mathbb{G} \times \mathbb{H} \to \mathbb{G}_T$ is a map with the following properties:
   a) Bilinearity. For any value $u \in \mathbb{G}, v \in \mathbb{H}$ and $a, b \in \mathbb{Z}_p^*$, $e(u^a, v^b) = e(u, v)^{ab}$ holds.
   b) Non-degeneracy. $e(g, h) \neq 1$ holds, where 1 means an identity element over $\mathbb{G}_T$.
   c) Computability. For any value $u \in \mathbb{G}$ and $h \in \mathbb{H}$, $e(u, v)$ can be calculated efficiently.

When $\mathbb{G} = \mathbb{H}$, bilinear groups are said to be *symmetric* and denoted by $(\mathbb{G}, \mathbb{G}_T)$ for the sake of convenience. Likewise, $\mathbb{G} \neq \mathbb{H}$, bilinear groups are said to be *asymmetric* and denoted by $(\mathbb{G}, \mathbb{H}, \mathbb{G}_T)$.

B. COMPLEXITY ASSUMPTIONS

In this section, we define security assumptions utilized in the proposed schemes.

1) BDHE ASSUMPTION IN $(\mathbb{G}, \mathbb{G}_T)$

The bilinear Diffie-Hellman exponentiation (BDHE) assumption in $(\mathbb{G}, \mathbb{G}_T)$ is an assumption introduced by Boneh et al. [12].

Definition 1 ((ϵ, l)-BDHE Assumption in $(\mathbb{G}, \mathbb{G}_T)$): We say the l-BDHE problem in $(\mathbb{G}, \mathbb{G}_T)$ with a security parameter 1^k as, for a given $(g, h, g^{\alpha}, g^{\alpha^2}, \ldots, g^{\alpha^l}, g^{\alpha^{l+2}}, \ldots, g^{\alpha^{2l}}, Z)$ with uniformly random $(g, h) \in \mathbb{G}$, $\alpha \in \mathbb{Z}_p^*$ and $(\mathbb{G}, \mathbb{G}_T)$ as input, determining whether $Z \in \mathbb{G}_T$ is $e(g^{\alpha^{l+1}}, h)$ or a random value R. We say that a polynomial time algorithm $\mathcal{A}$ can solve the l-BDHE problem in $(\mathbb{G}, \mathbb{G}_T)$ with an advantage ϵ if the following relation holds:

$$|Pr[\mathcal{A}(g, h, \boldsymbol{y}_{g,\alpha,l}, e(g^{\alpha^{l+1}}, h), \mathbb{G}, \mathbb{G}_T) = 0] - Pr[\mathcal{A}(g, h, \boldsymbol{y}_{g,\alpha,l}, R, \mathbb{G}, \mathbb{G}_T) = 0]| \geq \epsilon,$$

where $\boldsymbol{y}_{g,\alpha,l} = (g^{\alpha}, g^{\alpha^2}, \ldots, g^{\alpha^l}, g^{\alpha^{l+2}}, \ldots, g^{\alpha^{2l}})$. We say the (ϵ, l)-BDHE assumption holds in $(\mathbb{G}, \mathbb{G}_T)$ if there is no polynomial-time algorithm that can solve the l-BDHE problem in $(\mathbb{G}, \mathbb{G}_T)$ with ϵ.

2) DHE ASSUMPTION IN $\mathbb{G}$

The Diffie-Hellman exponentiation (DHE) assumption in $\mathbb{G}$ is an assumption introduced by Herranz et al. [40].

Definition 2 ((ϵ, l)-DHE Assumption in $\mathbb{G}$): We say the l-DHE problem with a security parameter 1^k as, for a given $(g, g^{\alpha}, g^{\alpha^2}, \ldots, g^{\alpha^l}, g^{\alpha^{l+2}}, \ldots, g^{\alpha^{2l}})$ with uniformly random $g \in \mathbb{G}$, $\alpha \in \mathbb{Z}_p^*$ and $(\mathbb{G}, \mathbb{G}_T)$ as input, computing $g^{\alpha^{l+1}}$. We say that a polynomial time algorithm $\mathcal{A}$ can solve the l-DHE problem in $\mathbb{G}$ with an advantage ϵ if the following relation holds:

$$Pr[\mathcal{A}(g, \boldsymbol{y}_{g,\alpha,l}, g^{\alpha^{l+1}}, \mathbb{G}, \mathbb{G}_T)] \geq \epsilon,$$

where $\boldsymbol{y}_{g,\alpha,l} = (g^{\alpha}, g^{\alpha^2}, \ldots, g^{\alpha^l}, g^{\alpha^{l+2}}, \ldots, g^{\alpha^{2l}})$. We say the (ϵ, l)-DHE assumption holds in $\mathbb{G}$ if there is no polynomial-time algorithm that can solve the l-DHE problem in $\mathbb{G}$ with ϵ.

3) XDH ASSUMPTION IN $(\mathbb{G}, \mathbb{H})$

The external Diffie-Hellman (XDH) assumption in $(\mathbb{G}, \mathbb{H})$ is an assumption introduced in [41], [42]. Note that, unlike the BDHE assumption and the DHE assumption described above, the XDH assumption holds on only *asymmetric* bilinear groups $(\mathbb{G}, \mathbb{H}, \mathbb{G}_T)$.

Definition 3 (ϵ-XDH Assumption in $(\mathbb{G}, \mathbb{H})$): We say the XDH problem in $(\mathbb{G}, \mathbb{H})$ as, for a given (g, h, g^a, g^b, Z) with uniformly random $g \in \mathbb{G}$, $h \in \mathbb{H}$, $(a, b) \in \mathbb{Z}_p^*$ and $(\mathbb{G}, \mathbb{H}, \mathbb{G}_T)$ as input, determining whether $Z \in \mathbb{G}$ is g^{ab} or is a random

value R. We say that a polynomial time algorithm $\mathcal{A}$ can solve XDH problem in $(\mathbb{G}, \mathbb{H})$ with advantage ϵ if the following relation holds:

$$|Pr[\mathcal{A}(g, h, g^a, g^b, g^{ab}, \mathbb{G}, \mathbb{H}, \mathbb{G}_T) = 0] - Pr[\mathcal{A}(g, h, g^a, g^b, R, \mathbb{G}, \mathbb{H}, \mathbb{G}_T) = 0]| \geq \epsilon.$$

We say ϵ-XDH assumption holds in $(\mathbb{G}, \mathbb{H})$ if there is no polynomial-time algorithm that can solve the XDH problem in $(\mathbb{G}, \mathbb{H})$ with ϵ.

III. KEY-AGGREGATE SEARCHABLE ENCRYPTION(KASE)

In this section, we describe the main problem statement of key-aggregate searchable encryption (KASE) [9]. Then, we newly define a syntax of algorithms and the security for KASE. These definitions are our contributions.

A. PROBLEM STATEMENT

In key-aggregate searchable encryption (KASE) [9], a data owner provides a ''single-and-short'' aggregate key that enables a user to access documents for authorization of search. Each user, which we call data user for the sake of convenience, is given an aggregate key as secret information and then generates a ''single-and-short'' trapdoor to search for a keyword from the documents. In doing so, the following requirements should be satisfied by KASE:

- Searchability: A user can generate trapdoors for any keyword to search in encrypted documents.
- Compactness: The size of both aggregate keys and trapdoors should be independent of the number of documents and number of users. In addition, the size of encrypted keywords should be independent of the number of users.
- Keyword Privacy: An adversary cannot extract information about the original keywords from encrypted keywords. That is, a person who does not have an aggregate key corresponding to indexes of the documents cannot get any information from the encrypted keyword.
- Aggregate Key Unforgeability: An adversary cannot search for any keyword without authorization from a data owner. That is, an adversary cannot perform keyword search over the documents that are not related to the known aggregate key and it cannot generate new aggregate keys for other sets of documents from the known keys.

These requirements are also shown in the original work of KASE [9]. The compactness and the searchability are functionality for KASE while the keyword privacy and the aggregate key unforgeability are security for KASE.

As described above, the compactness must be satisfied because the main motivation of KASE is to keep aggregate keys and trapdoors short while providing the searchability. We note that *the size of encrypted keywords may depend on the number of documents* because the data size of the encrypted keywords with respect to the number of users is the out of the scope of the compactness. The keyword privacy is a requirement that prevents an adversary from getting information contained in encrypted keywords. Meanwhile, aggregate key is a new notion required in KASE and has not been discussed in general searchable encryption. However, because documents outside the scope of authorization should be unsearchable, the aggregate key unforgeability should be discussed. Even if the keyword privacy is satisfied, there is a possibility that an adversary can search over documents that are outside the scope of authorization. Thus, both the keyword privacy and the aggregate key unforgeability should be discussed.

As another security requirement, the following should be considered:

- Trapdoor Privacy: An adversary cannot determine a keyword embedded in a given trapdoor. That is, even when a user asks an untrusted cloud server to search, the server cannot obtain the keyword except for the search results.

We note that the trapdoor privacy is an additional security requirement, i.e., only a few schemes [3], [43], [44] satisfy the trapdoor privacy even in the conventional searchable encryption. However, satisfying the trapdoor privacy is an important feature. When keywords embedded in trapdoors are revealed, the original keywords can be analyzed from encrypted keywords by looking up the search results. In other words, if the trapdoor privacy is unsatisfied, then the keyword privacy may be threatened. Thus, we consider that a KASE scheme should satisfy all requirements described above.

To the best of our knowledge, no provably secure KASE scheme or formal security definitions have been proposed. Thus, in this paper, we define the requirements above formally.

B. ALGORITHMS

The algorithms of KASE are defined as follows:

- $params \leftarrow Setup(1^\lambda, n)$: This algorithm is run by a cloud service provider to set up the scheme. On input of a security parameter 1^λ and the maximum number n of possible documents owned by a data owner, the algorithm outputs a public parameter $params$.
- $sk \leftarrow KeyGen(params)$: This algorithm is run by a data owner to generate a secret key sk.
- $c_{i,l} \leftarrow Encrypt(params, sk, i, w_l)$: This algorithm is run by a data owner to encrypt a keyword which belongs to the ith document and generate an encrypted keyword. On input of the data owner's secret key sk, a document index i, and a keyword $w_l \in \mathcal{KS}$ where $\mathcal{KS}$ is a keyword space, the algorithm outputs an encrypted keyword $c_{i,l}$.
- $k_{agg} \leftarrow Extract(params, sk, S)$: This algorithm is run by a data owner to generate an aggregate key for delegating the keyword search capability for a certain set of documents to other data users. On input of the data owner's secret key sk and a set S of indexes of documents, the algorithm outputs an aggregate key k_{agg}.
- $Tr \leftarrow Trapdoor(params, k_{agg}, S, w_l)$: This algorithm is run by a data user who performs the keyword search.

On input of an aggregate key k_{agg} and a keyword w_l, the algorithm outputs a single trapdoor Tr.

- $Tr_i \leftarrow Adjust(params, i, S, Tr, \{f_{1,i}\}_{i\in[1,m_1]})$: This algorithm is run by a cloud server to adjust the given aggregate trapdoor for each document. On input of a set S of indexes of documents, the index i of the target document, an aggregate trapdoor Tr, and auxiliary functions $\{f_i\}_{i\in[1,m_1]}(m_1 \in \mathbb{N})$ possibly, the algorithm outputs each trapdoor Tr_i for the ith target document in S.
- $b \leftarrow Test(params, Tr_i, S, c_{i,l}, \{f_{2,i}\}_{i\in[1,m_2]})$: This algorithm is run by a cloud server to perform keyword search over an encrypted keyword. On input of a trapdoor Tr_i, the document index i and auxiliary functions $\{f_i\}_{i\in[1,m_2]}(m_2 \in \mathbb{N})$ possibly, the algorithm outputs *true* or *false* to denote whether the ith document contains the keyword w_l.

We note that the syntax above represents a multi-server setting that includes multiple cloud servers. The syntax can contain multiple servers by setting auxiliary functions separately for each cloud server in the Adjust and Test algorithms.

We define the correctness of the syntax of KASE as follows:

Definition 4 (Correctness): For any document containing the keyword w_l with index $i \in S$, We say that a KASE scheme satisfies the correctness if the following statement holds: for all $1^\lambda, n \in \mathbb{N}, i \in [1,n], w_l \in \mathcal{KS}$, when a public parameter $params \leftarrow Setup(1^\lambda, n)$ and a secret key $sk \leftarrow KeyGen(params)$, $c_{i,l} \leftarrow Encrypt(params, sk, i, w_l)$ are used, $Test(params, Tr_i, S, c_{i,l}, \{f_{2,i}\}_{i\in[1,m_2]}) = true$ if $Tr \leftarrow Trapdoor(params, k_{agg}, S, w_l)$ and $Tr_i \leftarrow Adjust(params, i, S, Tr, \{f_{1,i}\}_{i\in[1,m_1]})$.

The correctness defined above imposes the searchability of KASE because the correctness guarantees that a data user can search for any keyword without decryption. Moreover, the syntax described above is identical to the abstraction of the algorithms proposed in the original work of KASE [9] except that the symmetric key setting instead of the public key setting in [9].

We define the compactness of KASE as follows:

Definition 5 (Compactness): We say that KASE satisfies the compactness if the sizes of both aggregate keys and trapdoors are independent of the number n of encrypted documents and the number m of data users, i.e., $\mathcal{O}(1)$ with respect to n and m, and the size of encrypted keywords is independent of m, i.e., $\mathcal{O}(n)$ with respect to n and m.

C. SECURITY DEFINITIONS

In this section, we define three security requirements for KASE, namely, the keyword privacy, the aggregate key unforgeability, and the trapdoor privacy. The security requirements are defined by the following game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. For each game, both $\mathcal{C}$ and $\mathcal{A}$ are given $(1^\lambda, n)$ as input, $\mathcal{A}$ is allowed to get aggregate keys, encrypted keywords, and trapdoors in the query phase by accessing the key extraction oracle $\mathcal{O}_{Extract}$, the encryption oracle $\mathcal{O}_{Encrypt}$, and the trapdoor oracle $\mathcal{O}_{Trapdoor}$, respectively. In particular, $\mathcal{A}$ accesses each oracle as follows:

- $\mathcal{O}_{Extract}$: by taking $S \subseteq [1,n]$ as input, return $k_{agg} \leftarrow Extract(params, sk, S)$.
- $\mathcal{O}_{Encrypt}$: by taking $i \in [1,n], w_l \in \mathcal{KS}$ as input, return $c_{i,l} \leftarrow Encrypt(params, sk, i, w_l)$.
- $\mathcal{O}_{Trapdoor}$: by taking $S \subseteq [1,n], w_l \in \mathcal{KS}$ as input, return $Tr \leftarrow Trapdoor(params, Extract(params, sk, S), S, w_l)$.

Definition 6 ((ϵ, n)-Keyword Privacy): In this game, an adversary $\mathcal{A}$ tries to distinguish a challenge keyword or a random keyword from a challenge encrypted keyword.

- *Init: $\mathcal{A}$ declares the index $i^* \in [1,n]$ of a challenge document used in the guess phase and sends it to $\mathcal{C}$.*
- *Setup: $\mathcal{C}$ generates $params \leftarrow Setup(1^\lambda, n)$ and $sk \leftarrow KeyGen(params)$, and sends params to $\mathcal{A}$.*
- *Query: $\mathcal{A}$ can query to $\mathcal{O}_{Extract}$ at most $n-1$ times[2] and can query to $\mathcal{O}_{Encrypt}$ at arbitrary times. Here, when $\mathcal{A}$ queries to $\mathcal{O}_{Extract}$, it imposes the constraint $S \subseteq [1,n] \setminus \{i^*\}$.*
- *Guess: $\mathcal{A}$ declares a challenge keyword w_{l^*} and sends it to $\mathcal{C}$. $\mathcal{C}$ randomly chooses $\theta \in \{0,1\}$. If $\theta = 0$, then $\mathcal{C}$ sets $w_\theta = w_{l^*}$. Otherwise, i.e., $\theta = 1$, $\mathcal{C}$ sets a random keyword as w_θ, where $|w_0| = |w_1|$. $\mathcal{C}$ sends $c_{i^*,\theta} \leftarrow Encrypt(params, sk, i^*, w_\theta)$ to $\mathcal{A}$. $\mathcal{A}$ then selects $\theta' \in \{0,1\}$.*

We say KASE satisfies the (ϵ, n)-keyword privacy if the following relation holds for $\mathcal{A}$'s advantage with any probabilistic polynomial time algorithm and 1^λ with any sufficiently large size:

$$Adv := |Pr[\theta = \theta'] - 1/2| < \epsilon$$

Definition 7 ((ϵ, n)-Aggregate Key Unforgeability): In this game, an adversary $\mathcal{A}$ tries to forge a valid aggregate key where $\mathcal{A}$ can search encrypted keywords with the aggregate key.

- *Setup: $\mathcal{C}$ randomly chooses $i^* \in [1,n]$. $\mathcal{C}$ generates $params \leftarrow Setup(1^\lambda, n)$ and $sk \leftarrow KeyGen(params)$, and then sends $params, i^*$ to $\mathcal{A}$.*
- *Query: $\mathcal{A}$ can query to $\mathcal{O}_{Extract}$ at most $n-1$ times and can query to $\mathcal{O}_{Encrypt}$ at arbitrary times. Here, when $\mathcal{A}$ queries to $\mathcal{O}_{Extract}$, it imposes the constraint $i^* \notin S$.*
- *Forge: $\mathcal{A}$ outputs $S^* \subseteq [1,n]$ and k^*_{agg}, where S^* includes i^*, i.e., $i^* \in S^*$.*

We say KASE satisfies the (ϵ, n)-aggregate key unforgeability if the following relation holds for $\mathcal{A}$'s advantage with any probabilistic polynomial time algorithm, keyword w_l and 1^λ with any sufficiently large size:

$$Adv := Pr[Test(params, Adjust(params, i^*, S^*, Trapdoor(params, k^*_{agg}, S^*, w_l))) = Test(params,$$

[2] If $\mathcal{A}$ can access to $\mathcal{O}_{Extract}$ more than n times, $\mathcal{A}$ can generate trapdoors for every index. $\mathcal{A}$ can trivially break any scheme without loss of generality. The restriction is also necessary for the remaining definitions.

$$Adjust(params, i^*, S^*, Trapdoor(params, Extract(params, sk, S^*), S^*, w_l)))] < \epsilon$$

Definition 8 ((ϵ, n)-Trapdoor Privacy): In this game, an adversary $\mathcal{A}$ tries to distinguish a challenge keyword or a random keyword from the given challenge trapdoor.

- *Init: $\mathcal{A}$ declares a set $S^* \subseteq [1, n]$ of indexes and a challenge keyword w_{l^*} used in the guess phase, and sends it to $\mathcal{C}$.*
- *Setup: $\mathcal{C}$ generates $params \leftarrow Setup(1^\lambda, n)$ and $sk \leftarrow KeyGen(params)$, and then sends params to $\mathcal{A}$.*
- *Query: $\mathcal{A}$ can query to $\mathcal{O}_{Trapdoor}$ at most $n - |S^*|$ times and can query to $\mathcal{O}_{Encrypt}$ at arbitrary times. Here, when $\mathcal{A}$ queries to $\mathcal{O}_{Encrypt}$, it imposes the constraint $w_l \neq w_{l^*} \wedge i \notin S^*$.*
- *Guess: $\mathcal{C}$ randomly chooses $\theta \in \{0, 1\}$. If $\theta = 0$ then $\mathcal{C}$ sets $w_\theta = w_{l^*}$. Otherwise, i.e., $\theta = 1$, $\mathcal{C}$ sets a random keyword as w_θ, where $|w_0| = |w_1|$ holds. $\mathcal{C}$ sends $Tr^* \leftarrow Trapdoor(params, k^*_{agg}, S^*, w_\theta)$ to $\mathcal{A}$. $\mathcal{A}$ then selects $\theta' \in \{0, 1\}$.*

We say KASE satisfies the (ϵ, n)-trapdoor privacy if the following relation holds for $\mathcal{A}$'s advantage with any probabilistic polynomial time algorithm and 1^λ with any sufficiently large size:

$$Adv := |Pr[\theta = \theta'] - 1/2| < \epsilon$$

D. TECHNICAL DIFFICULTY

Conventional searchable encryption [1], [45], [46] can be thought of performing the same search functionality as KASE, but the number of secret keys possessed to a data user and the number of trapdoors are proportional to the number of documents stored in cloud. Thus, the compactness cannot be satisfied.

The intuition of KASE's difficulty lies in the trade-off between security and features. That is, there is a possibility that security cannot be satisfied if the focus is only satisfying the compactness. In the case of KASE, the algebraic structure is limited because the sizes of aggregate key and trapdoor need to be $\mathcal{O}(1)$ regardless of the number of documents and the number of users. Therefore, potential configurations that satisfy searchability are limited. Furthermore, for the trapdoor privacy, the aggregate key must already have a concrete algebraic structure, and thus a trapdoor that uses only the aggregate key necessarily has more restrictive algebraic structures. This makes the trapdoor privacy even more difficult to satisfy. In fact, Kiayias et al. [3] showed the attack against encrypted keywords and trapdoors in Cui et al.'s scheme. This implies that it is difficult to construct a KASE scheme that satisfies security and features at the same time. In addition, note that Kiayias et al.'s attack scenario against Cui et al.'s scheme can be included in this oracle definition, considering the difference between the syntax of the secret key and the public key.

IV. CONSTRUCTIONS

In this section, we propose two constructions, namely, the first construction and the main construction, following the security requirements described in Section III-C. The first construction satisfies the keyword privacy and the aggregate key unforgeability. The main construction satisfies the trapdoor privacy in addition to the keyword privacy and the aggregate key unforgeability.

A. IDEA

Our main idea of KASE is to combine broadcast encryption (BE) [12], [13] and aggregate signatures (AS) [14]. BE is an encryption scheme that allows a specified set of users to decrypt ciphertext whereby a set of indexes corresponding to the user index is embedded in ciphertexts. Intuitively, the searchability can be realized by treating a decryption algorithm of BE as a test algorithm of KASE. The keyword privacy can then be satisfied by utilizing the ciphertext security of BE. Furthermore, we construct an aggregate key in the form of AS [14]. The signature size can be aggregated to a fixed length regardless of the number of users, and hence the compactness can be satisfied by keeping the construction of AS in aggregate keys. This also implies that the aggregate key unforgeability can be satisfied via the unforgeability of AS.

When we considered the idea of combining BE and AS, we found that the BE proposed by Boneh et al. [12] and the AS proposed by Boneh et al. [14] could be combined. Our first construction is close to a simple combination and can be constructed by a single server. However, trapdoor is out of the scope of this construction, i.e., the trapdoor privacy is unsatisfied.

For this reason, we also propose the main construction that satisfies the trapdoor privacy. In our main construction, we embed random values in trapdoors to make the trapdoors probabilistic. In doing so, to satisfy the searchability with the trapdoors, it is necessary to embed the same random values in encrypted keywords on a cloud. However, if the random values are sent to the cloud, the cloud can also extract the original keywords from the given trapdoor. Thus, we further utilize the idea of secret sharing for these random values. We also prepare for two servers that do not collude with each other. A data user distributes the random values embedded in the trapdoors into two shares, and then sends the shares to each server individually. By constructing the test algorithm in a way such as *n-out-of-n* threshold decryption [47], [48], the random values can be embedded in encrypted keywords and public parameters without knowing the original random values themselves. The approach described above allows search over ciphertexts and satisfies the trapdoor privacy.

B. FIRST CONSTRUCTION

The algorithms for the first construction are as follows:

- $params \leftarrow Setup(1^\lambda, n)$: Generate $\mathbb{B} = (p, \mathbb{G}, \mathbb{G}_T, e(\cdot, \cdot))$ as a bilinear map and bilinear groups, where p is an order

such that $\mathbb{G}$ and $2^{\lambda} < p < 2^{\lambda+1}$. Set n as the maximum number of documents. For $i \in \{1, 2, \ldots, n, n+2, \ldots, 2n\}$, pick a random generator $g \in \mathbb{G}$ and a random $\alpha \in \mathbb{Z}_p$, and then compute $g_i = g^{(\alpha^i)} \in \mathbb{G}$. Select a one-way hash function $H : \{0,1\}^* \to \mathbb{G}$. Finally, output a public parameter $params = (\mathbb{B}, PubK, H)$, where $PubK = (g, g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n}) \in \mathbb{G}^{2n}$.

- $sk \leftarrow KeyGen(params)$: Pick a random $\beta \in \mathbb{Z}_p$ and output a secret key $sk = \beta$.
- $c_{i,l} \leftarrow Encrypt(params, sk, i, w_l)$: Pick a random $t_{i,l} \in \mathbb{Z}_p$ and output an encrypted keyword $c_{i,l} = (c_{1,i,l}, c_{2,i,l}, c_{3,i,l})$ by computing the following:

$$c_{1,i,l} = g^{t_{i,l}}, c_{2,i,l} = (g^{\beta} \cdot g_i)^{t_{i,l}}, c_{3,i,l} = \frac{e(H(w_l), g)^{t_{i,l}}}{e(g_1, g_n)^{t_{i,l}}}.$$

- $k_{agg} \leftarrow Extract(params, sk, S)$: For the given subset $S \subseteq [1, n]$ which contains the indexes of documents, output an aggregate key k_{agg} by computing the following:

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

- $Tr \leftarrow Trapdoor(params, k_{agg}, S, w_l)$: For all documents relevant to the given aggregate key k_{agg}, generate a single trapdoor Tr for the keyword w_l by computing the following:

$$Tr = k_{agg} \cdot H(w_l).$$

- $Tr_i \leftarrow Adjust(params, i, S, Tr)$: For each document in the given set S, output trapdoor Tr_i by computing the following:

$$Tr_i = Tr \cdot \Pi_{j \in S, j \neq i} g_{n+1-j+i}.$$

- $b \leftarrow Test(params, Tr_i, S, c_{i,l})$: For the ith document and the keyword embedded in Tr_i, output *true* or *false* by judging whether the following equation holds or not:

$$\frac{e(Tr_i, c_{1,i,l})}{e(c_{2,i,l}, pub)} \stackrel{?}{=} c_{3,i,l},$$

where $pub = \Pi_{j \in S} g_{n+1-j}$.

Note: The first construction may seem to be the same as the construction by the Cui et al. [9], but these constructions are rigorously different. In the encrypt algorithm of Cui et al.'s scheme, each random number t embedded in encrypted keywords is different for each document. However, the same random number is embedded in encrypted keywords with the same document index. For any document with index i, all the encrypted keywords contain a common random number t_i, i.e., for any indexes i, j such that $i \neq j, t_i \neq t_j$ holds. By utilizing this feature, an adversary can extract the original keyword w_l to be encrypted from multiple encrypted keywords. In contrast, the first construction embeds an individual random number $t_{i,l}$ for not only each document but also for the same document index. Thus, an adversary cannot extract the original keyword w_l even when the adversary obtains multiple encrypted keywords.

The first construction described above satisfies the correctness, i.e., the searchability, because equation (1), as shown at the bottom of this page holds.

Furthermore, the sizes of an aggregate key and a trapdoor are $|\mathbb{G}|$ independent of the number of indexes in S because $k_{agg} = \Pi_{j \in S} g_{n+1-j}^{\beta} \in \mathbb{G}$ and $Tr = k_{agg} \cdot H(w_l) \in \mathbb{G}$. A search over encrypted keywords can be executed without changing the encrypted keywords themselves for any keyword. Hence, the first construction satisfies the compactness.

We will show that the first construction satisfies the keyword privacy and the aggregate key unforgeability in Section V-A. The first construction does not satisfy the trapdoor privacy because trapdoors are deterministic with respect to keywords. In other words, an adversary can extract keywords from trapdoors when the keywords used-so-far are sent again. In the next subsection, we will introduce the main construction, which satisfies the trapdoor privacy, under a two-server setting.

C. MAIN CONSTRUCTION

To construct a scheme that satisfies the trapdoor privacy, random values should be embedded in trapdoors. Likewise, the same random values should be embedded in the encrypted keyword and public parameters to satisfy the correctness.

Intuitively, the approach to use random values in trapdoors seems to require a data user to send the same random values to a cloud. However, if the data user sends the same random value as those in trapdoors, the cloud can extract $k_{agg} \cdot H(w_l)$ from the given trapdoors and the random values. Consequently, the trapdoor privacy is unsatisfied.

To overcome this limitation, instead of sending random values to a cloud, we aim to send the random values in a

$$\begin{aligned}
&\frac{e(Tr_i, c_{1,i,l})}{e(c_{2,i,l}, pub)} \\
&= \frac{e(\Pi_{j \in S} g_{n+1-j}^{\beta} \cdot H(w_l) \cdot \Pi_{j \in S, j \neq i} g_{n+1-j+i}, g^{t_{i,l}})}{e((g^{\beta} \cdot g_i)^{t_{i,l}}, \Pi_{j \in S} g_{n+1-j})} = \frac{e(\Pi_{j \in S} g_{n+1-j}^{\beta}, g^{t_{i,l}}) \cdot e(H(w_l), g^{t_{i,l}}) \cdot e(\Pi_{j \in S, j \neq i} g_{n+1-j+i}, g^{t_{i,l}})}{e(g^{\beta t_{i,l}}, \Pi_{j \in S} g_{n+1-j}) \cdot e(g_i^{t_{i,l}}, \Pi_{j \in S} g_{n+1-j})} \\
&= \frac{e(\Pi_{j \in S} g_{n+1-j}, g^{\beta t_{i,l}}) \cdot e(H(w_l), g^{t_{i,l}}) \cdot e(\Pi_{j \in S, j \neq i} g_{n+1-j+i}, g^{t_{i,l}})}{e(\Pi_{j \in S} g_{n+1-j}, g^{\beta t_{i,l}}) \cdot e(\Pi_{j \in S} g_{n+1-j+i} g^{t_{i,l}})} = \frac{e(H(w_l), g^{t_{i,l}}) \cdot e(\Pi_{j \in S, j \neq i} g_{n+1-j+i}, g^{t_{i,l}})}{e(\Pi_{j \in S} g_{n+1-j+i}, g^{t_{i,l}})} \\
&= \frac{e(H(w_l), g^{t_{i,l}}) \cdot e(\Pi_{j \in S} g_{n+1-j+i}, g^{t_{i,l}})}{e(\Pi_{j \in S} g_{n+1-j+i}, g^{t_{i,l}}) \cdot e(g_{n+1} \cdot g^{t_{i,l}})} = \frac{e(H(w_l), g^{t_{i,l}})}{e(g_{n+1} \cdot g^{t_{i,l}})} = \frac{e(H(w_l), g)^{t_{i,l}}}{e(g_1 \cdot g_n)^{t_{i,l}}}.
\end{aligned} \tag{1}$$

manner that nobody except for a data user itself can extract. Simultaneously, embedding the random values in encrypted keywords and a public parameter on the cloud without revealing the random values themselves. To do this, we construct trapdoors by using secret sharing. Consider a data user that distributes random values into two shares and then sends the shares to two servers. By utilizing these shares, a cloud can embed the random values in encrypted keywords and a public parameter without knowing the random values themselves.

In the main construction that will be described below, two servers $\mathcal{C}_{main}$ and $\mathcal{C}_{aid}$ are assumed to be semi-honest and to not collude with each other without loss of generality. Both servers store the same encrypted keywords. When a data user generates a trapdoor, he/she also generates a random value r and embeds r in the resulting trapdoor. The data user then distributes r into two shares and sends either of the shares to $\mathcal{C}_{main}$ and $\mathcal{C}_{aid}$, respectively. After receiving the trapdoor and the share, $\mathcal{C}_{main}$ and $\mathcal{C}_{aid}$ embed the received share in the stored encrypted keyword and public parameter provisionally. Then, these values are gathered on the $\mathcal{C}_{main}$ side. $\mathcal{C}_{main}$ then combines the encrypted keywords and the public parameter with the shares to recover r. That is, $\mathcal{C}_{main}$ can obtain the encrypted keywords and the public parameter where the random r is embedded without knowing r itself. Consequently, $\mathcal{C}_{main}$ is able to search over encrypted keywords and return the search results to the data user.

The main construction is described as follows. The asymmetric bilinear map is used in the main construction to satisfy the trapdoor privacy. Furthermore, in the Adjust and Test algorithms, we instantiate a function $f : \mathbb{Z}_p \times \mathbb{G} \to \mathbb{G}$ and $f_T : \mathbb{Z}_p \times \mathbb{G}_T \to \mathbb{G}_T$ as auxiliary functions of the input defined in Section III-B. f is a function that takes two arbitrary inputs $x \in \mathbb{Z}_p$, $g \in \mathbb{G}$ and outputs $g^x \in \mathbb{G}$. f_T is a function that takes two arbitrary inputs $x \in \mathbb{Z}_p$, $g_T \in \mathbb{G}_T$ and outputs $g_T^x \in \mathbb{G}_T$.

- $params \leftarrow Setup(1^\lambda, n)$: Generate a bilinear map group system $\mathbb{B} = (p, \mathbb{G}, \mathbb{H}, \mathbb{G}_T, e(\cdot,\cdot))$, where p is the order of $\mathbb{G}, \mathbb{H}$ and $2^\lambda < p < 2^{\lambda+1}$. Set n as the maximum possible number of documents that belong to a data owner. Pick a generator $g \in \mathbb{G}, h \in \mathbb{H}$ and a random $\alpha \in \mathbb{Z}_p$, and then compute $g_i = g^{(\alpha^i)} \in \mathbb{G}$ for $i \in \{1, 2, \ldots, n, n+2, \ldots, 2n\}$, $h_i = h^{(\alpha^i)} \in \mathbb{H}$ for $i \in [1, n]$. Select a one-way hash function $H : \{0,1\}^* \to \mathbb{G}$. Finally, output a public parameter $params = (\mathbb{B}, PubK, H)$, where $PubK = (g, g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n}, h, h_1, \ldots, h_n) \in (\mathbb{G}^{2n} \times \mathbb{H}^{n+1})$.
- $sk \leftarrow KeyGen(params)$: Pick a random $\beta \in \mathbb{Z}_p$ and output a secret key $sk = \beta$.
- $c_{i,l} \leftarrow Encrypt(params, sk, i, w_l)$: Pick a random $t_{i,l} \in \mathbb{Z}_p$ and output an encrypted keyword $c_{i,l} = (c_{1,i,l}, c_{2,i,l}, c_{3,i,l})$ by computing the following:

$$c_{1,i,l} = h^{t_{i,l}} \in \mathbb{H},$$
$$c_{2,i,l} = (g^\beta \cdot g_i)^{t_{i,l}} \in \mathbb{G},$$
$$c_{3,i,l} = \frac{e(H(w_l), h)^{t_{i,l}}}{e(g_1, h_n)^{t_{i,l}}} \in \mathbb{G}_T.$$

- $k_{agg} \leftarrow Extract(params, sk, S)$: For the given subset $S \subseteq [1, n]$ which contains the indexes of documents, output an aggregate key k_{agg} by computing the following:

$$k_{agg} = \Pi_{j \in S} g_{n+1-j}^\beta \in \mathbb{G}.$$

- $Tr \leftarrow Trapdoor(params, k_{agg}, S, w_l)$: Randomly generate $r \in \mathbb{Z}_p$ and calculate $Tr = (k_{agg} \cdot H(w_l))^r \in \mathbb{G}$. Then, r is broken into $r = r_{main} + r_{aid}$, and $Tr_{main} = (Tr, r_{main})$, $Tr_{aid} = r_{aid}$.
- $Tr_i \leftarrow Adjust(params, i, S, Tr, f(r_{main}, pub_i), f(r_{aid}, pub_i))$: Calculate $pub_i = \Pi_{j \in S, j \neq i} g_{n+1-j+i} \in \mathbb{G}$ on the two servers. $\mathcal{C}_{aid}$ sends $f(r_{aid}, pub_i) = pub_i^{r_{aid}}$ to $\mathcal{C}_{main}$. Next, $\mathcal{C}_{main}$ calculates $(f(r_{main}, pub_i)) \cdot (f(r_{aid}, pub_i)) = pub_i^{r_{main}} \cdot pub_i^{r_{aid}} = pub_i^r$ and calculates $Tr_i = Tr \cdot pub_i^r \in \mathbb{G}$.
- $b \leftarrow Test(params, Tr_i, S, c_{i,l}, f_T(r_{main}, c_{2,i,l}^\#), f_T(r_{aid}, c_{2,i,l}^\#), f_T(r_{main}, c_{3,i,l}), f_T(r_{aid}, c_{3,i,l}))$: Calculate $pub = \Pi_{j \in S} h_{n+1-j} \in \mathbb{H}$ and $c_{2,i,l}^\# = e(c_{2,i,l}, pub)$ on the two servers. $\mathcal{C}_{aid}$ sends $f_T(r_{aid}, c_{2,i,l}^\#) = e(c_{2,i,l}, pub)^{r_{aid}}$ to $\mathcal{C}_{main}$. Next, $\mathcal{C}_{main}$ calculates $(f_T(r_{main}, c_{2,i,l}^\#)) \cdot (f_T(r_{aid}, c_{2,i,l}^\#)) = e(c_{2,i,l}, pub)^{r_{main}} \cdot e(c_{2,i,l}, pub)^{r_{aid}} = e(c_{2,i,l}, pub)^r$. In addition, $\mathcal{C}_{aid}$ sends $f_T(r_{aid}, c_{3,i,l}) = c_{3,i,l}^{r_{aid}}$ to $\mathcal{C}_{main}$. Then, $\mathcal{C}_{main}$ calculates $(f_T(r_{main}, c_{3,i,l})) \cdot (f_T(r_{aid}, c_{3,i,l})) = c_{3,i,l}^{r_{main}} \cdot c_{3,i,l}^{r_{aid}} = c_{3,i,l}^r$ and outputs *true* or *false* by judging whether the following equation holds

$$\begin{aligned}
&\frac{e(Tr_i, c_{1,i,l})}{e(c_{2,i,l}, pub)^r} \\
&= \frac{e(\Pi_{j \in S} g_{n+1-j}^{\beta r} \cdot H(w_l)^r \cdot \Pi_{j \in S, j \neq i} g_{n+1-j+i}^r, h^{t_{i,l}})}{e((g^\beta \cdot g_i)^{t_{i,l}}, \Pi_{j \in S} h_{n+1-j}^r)} = \frac{e(\Pi_{j \in S} g_{n+1-j}^{\beta r}, h^{t_{i,l}}) \cdot e(H(w_l)^r, h^{t_{i,l}}) \cdot e(\Pi_{j \in S, j \neq i} g_{n+1-j+i}^r, h^{t_{i,l}})}{e(g^{\beta t_{i,l}}, \Pi_{j \in S} h_{n+1-j}^r) \cdot e(g_i^{t_{i,l}}, \Pi_{j \in S} h_{n+1-j}^r)} \\
&= \frac{e(\Pi_{j \in S} g_{n+1-j}^r, h^{\beta t_{i,l}}) \cdot e(H(w_l)^r, h^{t_{i,l}}) \cdot e(\Pi_{j \in S, j \neq i} g_{n+1-j+i}^r, h^{t_{i,l}})}{e(\Pi_{j \in S} g_{n+1-j}^r, h^{\beta t_{i,l}}) \cdot e(\Pi_{j \in S} g_{n+1-j+i}^r, h^{t_{i,l}})} = \frac{e(H(w_l)^r, h^{t_{i,l}}) \cdot e(\Pi_{j \in S, j \neq i} g_{n+1-j+i}^r, h^{t_{i,l}})}{e(\Pi_{j \in S} g_{n+1-j+i}^r, h^{t_{i,l}})} \\
&= \frac{e(H(w_l)^r, h^{t_{i,l}}) \cdot e(\Pi_{j \in S} g_{n+1-j+i}^r, h^{t_{i,l}})}{e(\Pi_{j \in S} g_{n+1-j+i}^r, h^{t_{i,l}}) \cdot e(g_{n+1}^r . h^{t_{i,l}})} = \frac{e(H(w_l)^r, h^{t_{i,l}})}{e(g_{n+1}^r, h^{t_{i,l}})} = \frac{e(H(w_l), h)^{r t_{i,l}}}{e(g_1 . h_n)^{r t_{i,l}}} = c_{3,i,l}^r
\end{aligned} \tag{2}$$

or not:

$$\frac{e(Tr_i, c_{1,i,l})}{e(c_{2,i,l}, pub)^r} =^? c_{3,i,l}^r.$$

The main construction satisfies the correctness, i.e., the searchability, as shown in equation (2), as shown at the bottom of the previous page.

Furthermore, similar to the first construction, the sizes of an aggregate key and a trapdoor are $|\mathbb{G}|$ independent of the number n of indexes in S. In addition, the data size of encrypted keywords is independent of the number of users because the algorithms do not change the encrypted keywords themselves. Thus, the main construction also satisfies the compactness.

We note that r_{aid} is known by only $\mathcal{C}_{aid}$ and r_{main} is known by only $\mathcal{C}_{main}$. In the trapdoor algorithm, a data user sends Tr_{main} to $\mathcal{C}_{main}$ and Tr_{aid} to $\mathcal{C}_{aid}$. Then, $\mathcal{C}_{aid}$ sends $pub_i^{r_{aid}}$ in the adjust algorithm and $c_{3,i,l}^{r_{aid}}, e(c_{2,i,l}, pub)^{r_{aid}}$ in the test algorithm to $\mathcal{C}_{main}$. $\mathcal{C}_{aid}$ does not send r_{aid} itself, and thus $\mathcal{C}_{main}$ does not know r_{aid}.

V. SECURITY PROOFS

In this section, we will show the security proofs of the first construction and main construction. The security of the main construction is proved in the two-server setting because it uses two servers. The proof statement is consistent with the security definitions because our definitions have captured the multi-server setting by applying an auxiliary function individually for each server.

A. PROOFS OF THE FIRST CONSTRUCTION

The first construction satisfies the (ϵ', n)-keyword privacy and the (ϵ', n)-aggregate key unforgeability. In this section, we prove these two securities.

Theorem 1 ((ϵ', n)-Keyword Privacy): The first construction satisfies the (ϵ', n)-keyword privacy under the (ϵ, n)-BDHE Assumption, where $\epsilon \geq \epsilon'$.

Proof: Suppose there exists an adversary $\mathcal{A}$, whose advantage is ϵ', against the first construction. We then build an algorithm $\mathcal{B}$ that solves the BDHE problem. Let $\mathcal{C}$ be a challenger for the BDHE problem. Algorithm $\mathcal{B}$ proceeds as follows.

- Init: $\mathcal{A}$ declares challenge document index $i^* \in [1, n]$ and sends it to $\mathcal{B}$.
- Setup: $\mathcal{C}$ sends $(g, h, g_1, g_2, \ldots, g_n, g_{n+2}, \ldots, g_{2n}, Z)$ to $\mathcal{B}$. $\mathcal{B}$ randomly generates $sk = \beta$ and calculates $v' = g^\beta g_{i^*}^{-1}$. $\mathcal{B}$ sends $params = (g, g_1, g_2, \ldots, g_n, g_{n+2}, \ldots, g_{2n})$ to $\mathcal{A}$.
- Query: When $\mathcal{A}$ queries for $\mathcal{O}_{Extract}$, $\mathcal{B}$ responds as follows:
  - If an aggregate key for $i^* \in S$ is queried, return $\perp$.
  - If an aggregate key for $i^* \notin S$ is queried, return $k_{agg} = (\Pi_{j\in S} g_{n+1-j}^\beta) \cdot (\Pi_{j\in S} g_{n+1-j+i^*})^{-1} = \Pi_{j\in S} g_{n+1-j}^{\beta-\alpha^{i^*}}$. If $j = i^*$, $(\Pi_{j\in S} g_{n+1-j+i^*})^{-1}$ cannot be calculated, but it can be calculated because of $i^* \notin S$.

  When $\mathcal{A}$ queries for $\mathcal{O}_{Encrypt}$, $\mathcal{B}$ randomly generates $t_{i,l} \in \mathbb{Z}_p^*$, calculates the following $c_{i,l} = (c_{1,i,l}, c_{2,i,l}, c_{3,i,l})$ and responds to $\mathcal{A}$ ($c_{1,i,l} = g^{t_{i,l}}, c_{2,i,l} = (v' \cdot g_i)^{t_{i,l}}, c_{3,i,l} = \frac{e(H(w_l),g)^{t_{i,l}}}{e(g_1,g_n)^{t_{i,l}}}$).
- Guess: $\mathcal{A}$ declares the challenge keyword w_{l^*} and sends it to $\mathcal{B}$. $\mathcal{B}$ calculates the challenge encrypted keyword $c_{1,i^*,\theta} = h, c_{2,i^*,\theta} = h^\beta, c_{3,i^*,\theta} = \frac{e(H(w_{l^*}),h)}{Z}$.
  Here, we define $h = g^t$ (t is a random value). Then, when $Z = e(g_{n+1}, h)$, $c_{1,i^*,\theta} = g^t = h, c_{2,i^*,\theta} = ((g^\beta g_{i^*}^{-1}) \cdot g_{i^*})^t = g^{\beta t} = h^\beta, c_{3,i^*,\theta} = \frac{e(H(w_{l^*}),g)^t}{e(g_1,g_n)^t} = \frac{e(H(w_{l^*}),h)}{e(g_{n+1},h)} = \frac{e(H(w_{l^*}),h)}{Z}$. Therefore, the calculation results are identical to the results of the Encrypt algorithm of the first construction. $\mathcal{B}$ sends $c_{i^*,\theta} = (c_{1,i^*,\theta}, c_{2,i^*,\theta}, c_{3,i^*,\theta})$ to $\mathcal{A}$. $\mathcal{A}$ chooses $\theta' \in \{0, 1\}$ and sends it to $\mathcal{B}$. Then, $\mathcal{B}$ sends θ' to $\mathcal{C}$ as a guess of θ.

In the guess phase, if Z is a random value, then $\Pr[\theta = \theta'] = 1/2$. On the other hand, if $Z = e(g_{n+1}, h)$, then $|\Pr[\theta = \theta'] - 1/2| > \epsilon'$. This indicates that $\mathcal{B}$ has an advantage over ϵ' for solving the (ϵ, n)-BDHE problem. Thus, if the (ϵ, n)-BDHE assumption holds, the first construction satisfies the (ϵ', n)-keyword privacy.

Theorem 2 ((ϵ', n)-Aggregate key Unforgeability): The first construction satisfies the (ϵ', n)-aggregate key unforgeability under the (ϵ, n)-DHE Assumption, where $\epsilon = \epsilon'$.

Proof: Suppose there exists as adversary $\mathcal{A}$, whose advantage is ϵ', against the first construction. We then build an algorithm $\mathcal{B}$ that solves the DHE problem. Let $\mathcal{C}$ be a challenger for the DHE problem. Algorithm $\mathcal{B}$ proceeds as follows.

- Setup: $\mathcal{C}$ sends $(g, g^\alpha, g^{\alpha^2}, \ldots, g^{\alpha^n}, g^{\alpha^{n+2}}, \ldots, g^{\alpha^{2n}})$ to $\mathcal{B}$. $\mathcal{B}$ randomly generates $sk = \beta$ and calculates $v' = g^\beta g_{i^*}^{-1}$. $\mathcal{B}$ sends $params = (g, g_1, g_2, \ldots, g_n, g_{n+2}, \ldots, g_{2n})$ to $\mathcal{A}$.
- Query: When $\mathcal{A}$ queries for $\mathcal{O}_{Extract}$, $\mathcal{B}$ responds as follows:
  - If an aggregate key for $i^* \in S$ is queried, return $\perp$.
  - If an aggregate key for $i^* \notin S$ is queried, return $k_{agg} = (\Pi_{j\in S} g_{n+1-j}^\beta) \cdot (\Pi_{j\in S} g_{n+1-j+i^*})^{-1} = \Pi_{j\in S} g_{n+1-j}^{\beta-\alpha^{i^*}}$. Here, if $j = i^*$, then $(\Pi_{j\in S} g_{n+1-j+i^*})^{-1}$ cannot be calculated, but it can be calculated because of $i^* \notin S$.

  When $\mathcal{A}$ queries for $\mathcal{O}_{Encrypt}$, $\mathcal{B}$ randomly generates $t_{i,l} \in \mathbb{Z}_p^*$, calculates $c_{i,l} = (c_{1,i,l}, c_{2,i,l}, c_{3,i,l})$ and responds to $\mathcal{A}$ ($c_{1,i,l} = g^{t_{i,l}}, c_{2,i,l} = (v' \cdot g_i)^{t_{i,l}}$, $c_{3,i,l} = \frac{e(H(w_l),g)^{t_{i,l}}}{e(g_1,g_n)^{t_{i,l}}}$).
- Forge: $\mathcal{A}$ outputs S^*, k_{agg}^* and sends them to $\mathcal{B}$.
  - If $i^* \notin S^*$, abort
  - If $i^* \in S^*$, $k_{agg}^* = (\Pi_{j\in S, j\neq i^*} g_{n+1-j})^{\beta-\alpha^{i^*}} \cdot (g_{n+1-i^*})^{\beta-\alpha^{i^*}}$. By using this k_{agg}^*, $\mathcal{B}$ calculates (3), as shown at the bottom of the next page, and outputs results.

The result in the above (ϵ', n)-aggregate key unforgeability game is identical to the answer of the (ϵ, n)-DHE problem.

That is, the advantage of the (ϵ', n)-aggregate key unforgeability game is equal to the advantage of the (ϵ, n)-DHE problem. Thus, if the (ϵ, n)-DHE assumption holds, the first construction satisfies the (ϵ', n)-aggregate key unforgeability.

B. PROOFS OF THE MAIN CONSTRUCTION

In this section, we prove that the main construction satisfies the (ϵ', n)-trapdoor privacy. Note that the (ϵ', n)-keyword privacy and the (ϵ', n)-aggregate key unforgeability can be proved similarly to the proofs for the first construction except for the use security assumptions in asymmetric bilinear groups (see the Appendix for details). As described in the previous section, the main construction is based on two servers. In our security proof, a challenge ciphertext and a challenge trapdoor for both $\mathcal{C}_{main}$ and $\mathcal{C}_{aid}$ are simulated by a reduction algorithm.

Theorem 3 ((ϵ', n)-Trapdoor Privacy): Let a hash function H be modeled as a random oracle. The main construction satisfies the (ϵ', n)-trapdoor privacy under the ϵ-XDH assumption, where $\epsilon \geq \epsilon'$.

Proof: Suppose there exists as adversary $\mathcal{A}$, whose advantage is ϵ', against the main construction. We then build an algorithm $\mathcal{B}$ that solves the XDH problem. Let $\mathcal{C}$ be a challenger for the XDH problem. Algorithm $\mathcal{B}$ proceeds as follows.

- Init: $\mathcal{A}$ declares challenge document index set $S^* \subseteq U$ and challenge keyword w_{l^*} and sends them to $\mathcal{B}$.
- Setup: $\mathcal{C}$ sends (g, h, g^a, g^b, Z) to $\mathcal{B}$. $\mathcal{B}$ randomly generates $\alpha, \omega, r'_{main} \in \mathbb{Z}_p^*$ and calculates $g_i = g^{\alpha^i} (i \in \{1, 2, \ldots, n, n+2, \ldots, 2n\})$, $h_i = h^{\alpha^i} (i \in [1, n+1])$. ω is used to generate both a response from the random oracle H and the challenge trapdoor. Simultaneously, r'_{main} corresponds to r_{main} of the challenge trapdoor. In this proof, the random value r of challenge trapdoor is mapped to the challenge a of g^a. In doing so, calculating $r'_{aid} = a - r'_{main}$ is necessary in accordance with the Trapdoor algorithm. However, since $\mathcal{B}$ does not know a, $\mathcal{B}$ cannot calculate r'_{aid} itself. Then, instead of calculating r'_{aid} as behavior for $\mathcal{C}_{aid}$ in the challenge phase, $\mathcal{B}$ calculates the value including r'_{aid} as $\mathcal{C}_{aid}$. Specifically, $\mathcal{B}$ calculates $g^a \cdot g^{-r'_{main}} = g^{a-r'_{main}} = g^{r'_{aid}}$ and generates the value including $g^{r'_{aid}}$ as the behavior of $\mathcal{C}_{aid}$. This implicitly means that, in the main construction, $\mathcal{C}_{aid}$, who receives r_{aid}, embeds r_{aid} in $e(c_{2,i,l}, pub)$, pub_i and $c_{3,i,l}$ in the Adjust and Test algorithms. To execute the Adjust and Test algorithms correctly for the challenge trapdoor, $\mathcal{C}_{aid}$ is also considered to use the following values:

$$pub^* = \Pi_{j \in S^*} h_{n+1-j},$$
$$pub_i^* = \Pi_{j \in S^*, j \neq i} (g^{r'_{aid}})^{\alpha^{n+1-j+i}} (i \in S^*).$$

Next, $\mathcal{B}$ calculates the following values:

$$c'^{\#}_{2,i,l} = (Z \cdot g^{br'_{main}}) \cdot (g^{r'_{aid}})^{\alpha^i} (i \in S^*),$$
$$A_i = e(c'^{\#}_{2,i,l}, pub^*),$$
$$B = \frac{e((g^{r'_{aid}})^{\omega}, h)}{e(g^{r'_{aid}}, h_{n+1})} = \left(\frac{e(g^{\omega}, h)}{e(g_1, h_n)}\right)^{r'_{aid}}.$$

These values are used to calculate encrypted keywords, which satisfy the correctness for the challenge trapdoor. $\mathcal{B}$ sends $params = (g, g_1, g_2, \ldots, g_n, g_{n+2}, \ldots, g_{2n}, h, h_1, h_2, \ldots, h_n)$ to $\mathcal{A}$. Here, $\mathcal{B}$ does not know $sk = b$ because it is a part of the secret of the challenge.

- Hash: When $\mathcal{A}$ queries for random oracle model, $\mathcal{B}$ responds as follows. Note that $\mathcal{B}$ has a hash list $L(w_l, y_l)$. In the initial state, L is an empty set.
  - If w_l is in L, return the corresponding y_l stored in L.
  - If w_l is not in L, setting y_l as follows, add (w_l, y_l) to L and return y_l.
    - If $w_l = w_{l^*}$, let $y_l = g^{\omega}$.
    - If $w_l \neq w_{l^*}$, choose $x \in \mathbb{Z}_p^*$ uniformly randomly and let $y_l = g^x$.
- Query: If $\mathcal{A}$ queries for $\mathcal{O}_{Encrypt}$, $\mathcal{B}$ returns as follows:
  - If $\mathcal{A}$ queries for a keyword which satisfies $w_l = w_{l^*} \vee i \in S^*$, return $\perp$.
  - If $\mathcal{A}$ queries for a keyword which satisfies $w_l \neq w_{l^*} \wedge i \notin S^*$, randomly generate $t_{i,l} \in \mathbb{Z}_p^*$ and return $c_{i,l} = (c_{1,i,l}, c_{2,i,l}, c_{3,i,l})$ $(c_{1,i,l} = h^{t_{i,l}}, c_{2,i,l} = (g^b \cdot g_i)^{t_{i,l}}, c_{3,i,l} = \frac{e(H(w_l), h)^{t_{i,l}}}{e(g_1, h_n)^{t_{i,l}}})$

  If $\mathcal{A}$ queries for $\mathcal{O}_{Trapdoor}$, $\mathcal{B}$ randomly generates $r \in \mathbb{Z}_p^*$ and calculates $Tr = (\Pi_{j \in S} (g^b)^{\alpha^{n+1-j}} \cdot H(w_l))^r = (\Pi_{j \in S} g^b_{n+1-j} \cdot H(w_l))^r$. Then, $\mathcal{B}$ randomly generates $r_{main} \in \mathbb{Z}_p^*$ and calculates $r_{aid} = r - r_{main}$. $\mathcal{B}$ returns (Tr, r_{main}) to $\mathcal{A}$.
- Guess: $\mathcal{B}$ calculates the challenge trapdoor as $Tr^* = \Pi_{j \in S^*} Z^{\alpha^{n+1-j}} \cdot (g^a)^{\omega}$. At this time, if $Z = g^{ab}$, $Tr = \Pi_{j \in S^*} (g^{ab})^{\alpha^{n+1-j}} \cdot (g^a)^{\omega} = (\Pi_{j \in S^*} g^b_{n+1-j} \cdot g^{\omega})^a$ holds. From the simulation of the hash phase, $H(w_{l^*}) = g^{\omega}$ holds, and therefore $Tr = (\Pi_{j \in S^*} g^b_{n+1-j} \cdot H(w_{l^*}))^a$ holds. Therefore, Tr has the same distribution as that calculated by the Trapdoor algorithm in the main construction. $\mathcal{B}$ sends Tr^* and r'_{main} to $\mathcal{A}$. $\mathcal{A}$ then chooses $\theta' \in \{0, 1\}$

$$\frac{(\Pi_{j \in S^*, j \neq i^*} g_{n+1-j})^{\beta} \cdot (\Pi_{j \in S^*, j \neq i^*} g_{n+1-j+i^*})^{-1} \cdot (g_{n+1-i^*})^{\beta}}{k^*_{agg}}$$
$$= \frac{(\Pi_{j \in S^*, j \neq i^*} g_{n+1-j})^{\beta} \cdot (\Pi_{j \in S^*, j \neq i^*} g_{n+1-j+i^*})^{-1} \cdot (g_{n+1-i^*})^{\beta}}{(\Pi_{j \in S^*, j \neq i^*} g_{n+1-j})^{\beta} \cdot (\Pi_{j \in S^*, j \neq i^*} g_{n+1-j})^{-\alpha^{i^*}} \cdot (g_{n+1-i^*})^{\beta} \cdot (g_{n+1-i^*})^{-\alpha^{i^*}}} = (g_{n+1-i^*})^{\alpha^{i^*}} = g^{\alpha^{n+1-i^*+i^*}} = g^{\alpha^{n+1}} \quad (3)$$

and returns it to $\mathcal{B}$. Finally, $\mathcal{B}$ returns the received θ' to $\mathcal{C}$ as a guess of θ.

Note that if $Z = g^{ab}$, not only the trapdoor generated in the query phase but also the challenge trapdoor satisfies the correctness.

In the guess phase, if Z is a random value, then $Pr[\theta = \theta'] = 1/2$. On the other hand, if $Z = g^{ab}$, then $|Pr[\theta = \theta'] - 1/2| = Adv > \epsilon'$. This indicates that $\mathcal{B}$ has an advantage over ϵ' for solving the ϵ-XDH problem. Thus, if the ϵ-XDH assumption holds, the main construction satisfies the (ϵ', n)-trapdoor privacy.

VI. DISCUSSION

In this section, we discuss the performance of our proposed schemes. We first implement the proposed schemes to measure their actual performance. Next, we theoretically compare the computational cost and the storage cost of the proposed schemes with those of related works. We also compare the security features of our schemes with those of related works.

A. IMPLEMENTATION AND PERFORMANCE

We implement the first construction and the main construction to evaluate the performance of each algorithm. Our reference implementations are available on GitHub (`https://github.com/naotoyanai/kase`). The implementation environment and performance evaluation are as follows:

1) IMPLEMENTATION ENVIRONMENT

In our implementation, we use the mcl library version 0.94,[3] which is a C++ library for pairing computation. We also use the BLS12-381 curves. The curves are asymmetric bilinear maps, and hence the parameters in the first construction are dually generated for each input group of the bilinear maps. We also evaluate each cryptographic operation in the C++ platform with Mac OS named Mojave. In our environment, CPU is 1.4 GHz Intel Core i5-4260U and memory is 4 GB 1600 MHz DDR3. We note that the communication latency to interact between $\mathcal{C}_{main}$ and $\mathcal{C}_{aid}$ in the main construction is not measured because the communication process between $\mathcal{C}_{main}$ and $\mathcal{C}_{aid}$ is not implemented due to the lack of communication function in the mcl library.

2) PERFORMANCE EVALUATION

The results of the performance evaluation of our schemes are shown in Figures 2–7. Among the KASE algorithms, the Setup, Encrypt, and Extract algorithms are common to the first construction and the main construction because the mcl library supports asymmetric bilinear maps. Thus, the evaluation of these algorithms, i.e., Figures 2–4, contains only the first construction. The results shown in Figures 2–4 are common also for those in the main construction. On the other hand, the results of the Adjust and Test algorithms listed in Figures 6 and 7 do not include the communication latency between the two servers in the main construction.

[3]mcl library: `https://github.com/herumi/mcl`

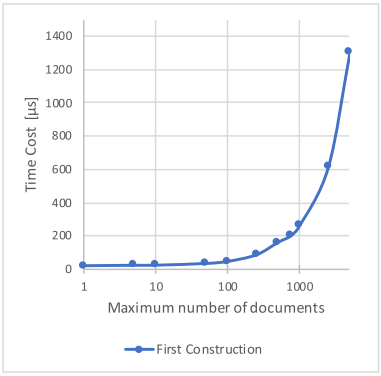

FIGURE 2. Time cost of setup.

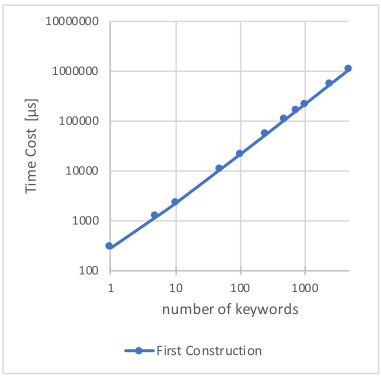

FIGURE 3. Time cost of encrypt.

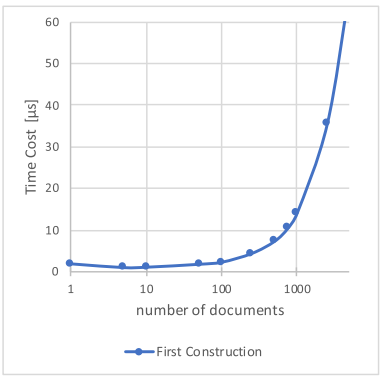

FIGURE 4. Time cost of extract.

The execution time of the Setup algorithm is linear with respect to the maximum number of documents belonging to a single owner. When the number of documents increases to 5,000, the Setup algorithm only needs one millisecond of execution time and therefore remains reasonable.

The execution time of the Encrypt algorithm is linear with respect to the number of keywords. The execution time of the Encrypt algorithm is larger than those of other algorithms because of the use of bilinear maps with heavy calculations. Nevertheless, when the number of keywords increases to 5,000, the execution time of the Encrypt algorithm finishes within one second and is therefore still practical.

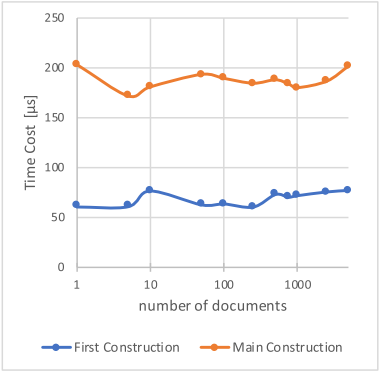

FIGURE 5. Time cost of trapdoor.

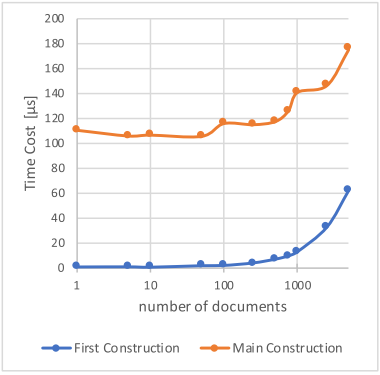

FIGURE 6. Time cost of adjust.

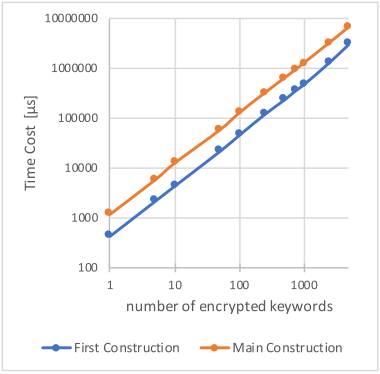

FIGURE 7. Time cost of test.

The execution time of the Extract algorithm is linear with respect to the number of shared documents. When the number of documents increases to 5,000, the Extract algorithm needs only 0.07 milliseconds of execution time. The Extract algorithm can be performed faster than the other algorithms.

The execution time of the Trapdoor algorithm is constant with respect to the number of documents, i.e., 0.07 milliseconds in the first construction and 0.2 milliseconds in the main construction. The difference in the execution time depends on the number of scalar multiplications. In particular, the first construction does not utilize scalar multiplication, whereas the main construction utilizes a single scalar multiplication. Since the data sizes of r_{main} and r_{aid} are small because of integers, the entire execution time can be minimized even when the communication latency is included.

The execution time of the Adjust algorithm is linear with respect to the number of documents. When the number of documents increases to 5,000, the Adjust algorithm takes only 0.06 milliseconds in the first construction and only 0.18 milliseconds in the main construction. Similar to the Extract algorithm, the Adjust algorithm in the first construction can be performed faster than the other algorithms. In the main construction, the Adjust algorithm includes scalar multiplications and hence the computation is slightly heavy. However, we note that the computations for $\mathcal{C}_{main}$ and $\mathcal{C}_{aid}$ can be done in parallel to improve the performance. Although we did not implement the parallelization, this could improve the performance twice faster.

The execution time of the Test algorithm is linear with respect to the number of encrypted keywords. When the number of keywords increases to 5,000, the algorithm takes three seconds in the first construction and six seconds in the main construction. Similar to the Encrypt algorithm, the high cost is caused by the use of bilinear maps, which is a bottleneck in all algorithms that use it. However, the search process can be fully parallelized because it is individual for each encrypted keyword. Thus, the performance can be improved by parallelization, e.g., by the use of the OpenMP library.[4]

Finally, search for any keyword is a summation of the execution times of the Trapdoor, Adjust, and Test algorithms. For instance, a search in 5,000 encrypted keywords is executed within about three seconds in the first construction and about six seconds in the main construction.

B. COMPUTATIONAL AND STORAGE COST ANALYSIS

In this section, we compare the computational cost and the storage cost of our schemes with other schemes [9], [11], [22] of KASE. The results are shown in Tables 1 and 2. Li et al. [11] proposed two constructions, i.e., the single-owner setting and the multi-owner setting. Therefore, we only compare our schemes in the single-owner setting because our schemes have a single-owner setting.

The computational cost and the storage size for the first construction are less-than-or-equal to those of the schemes by Cui et al. [9] and by Li et al. [11] in spite of achieving the provable security, which is an open problem in these works.

Even in the main construction, the computational cost for the Encrypt algorithm is identical to that of the scheme by Cui et al. [9], and the computational cost for the Trapdoor algorithm is smaller than that of the scheme by Zhou et al. [22]. Although the computational costs for the Adjust and Test algorithms are greater than those of other schemes, the number of scalar multiplications in $\mathbb{G}$ and exponentiations in $\mathbb{G}_T$, whose computations are heavy, are constant with respect to the number of documents.

[4]OpenMP library: https://www.openmp.org/wp-content/uploads/cspec20.pdf

TABLE 1. The computational cost of KASE: The operation time of hash operations, scalar multiplication, point addition, exclusive or in $\mathbb{G}_T$, exponentiation in $\mathbb{G}_T$, multiplication in $\mathbb{G}_T$, and pairing operation are identified as T_h, T_{sm}, T_a, T_x, T_{exp}, T_{mul}, and T_p, respectively. Li et al. [11] uses a Bloom filter to verify whether the keyword really exists in the document set. The time taken for the operation is represented as T_{bf}. The random generation and integer addition are ignored. The time of *Adjust* + *Test* refers to the computational cost that takes file per one index.

	Cui et al. [9]	Li et al. [11]	Zhou et al. [22]	First Construction	Main Construction
$Encrypt$	$T_h + 2T_{sm} + T_a +$ $2T_p + T_{mul} + T_{exp}$	$T_h + 2T_{sm} + 2T_a + 2T_p +$ $T_{mul} + 2T_{exp} + T_x + T_{bf}$	$T_h + 4T_{sm} + T_a$	$T_h + 2T_{sm} + T_a +$ $2T_p + T_{mul} + T_{exp}$	$T_h + 2T_{sm} + T_a +$ $2T_p + T_{mul} + T_{exp}$
$Trapdoor$	$T_h + T_a$	$T_h + T_a$	$T_h + 3T_{sm} + T_a$	$T_h + T_a$	$T_h + T_a + T_{sm}$
$Adjust + Test$	$2\|S\| \cdot T_a +$ $T_{mul} + 2T_p$	$2\|S\| \cdot T_a +$ $T_{mul} + 2T_p$	$2\|S\| \cdot T_a +$ $2T_{mul} + 4T_p$	$2\|S\| \cdot T_a +$ $T_{mul} + 2T_p$	$6\|S\| \cdot T_a + 2\|S\| \cdot T_{mul} +$ $4T_{sm} + 2T_{exp} + 2T_p$

TABLE 2. The storage cost of KASE: $|\mathbb{G}|$, $|\mathbb{G}_T|$ refers to the size of $\mathbb{G}$, $\mathbb{G}_T$.

	Cui et al. [9]	Li et al. [11]	Zhou et al. [22]	First Construction	Main Construction
encrypted keyword	$2\|\mathbb{G}\| + \|\mathbb{G}_T\|$	$2\|\mathbb{G}\| + 3\|\mathbb{G}_T\|$	$3\|\mathbb{G}\|$	$2\|\mathbb{G}\| + \|\mathbb{G}_T\|$	$3\|\mathbb{G}\| + 2\|\mathbb{G}_T\|$
trapdoor	$\|\mathbb{G}\|$	$\|\mathbb{G}\|$	$2\|\mathbb{G}\|$	$\|\mathbb{G}\|$	$\|\mathbb{G}\| + 2\|\mathbb{Z}_p^*\|$
aggregate key	$\|\mathbb{G}\|$	$\|\mathbb{G}\|$	$\|\mathbb{G}\|$	$\|\mathbb{G}\|$	$\|\mathbb{G}\|$

TABLE 3. The Security of KASE: Checkmark means achievement of the provable security.

	Cui et al. [9]	Li et al. [11]	Zhou et al. [22]	First Construction	Main Construction
Compactness	✓	✓		✓	✓
Keyword Privacy			✓	✓	✓
Aggregate key Unforgeability				✓	✓
Trapdoor Privacy			✓		✓

Thus, the computational cost of the main construction can be considered to be practical. Moreover, for the storage cost, in spite of two additional components in $\mathbb{Z}_p$, the storage size is fairly identical to that of the scheme by Zhou et al. In particular, an element in $\mathbb{G}$ is constructed by two integers on an elliptic curve, i.e., x-coordinate and y-coordinate, whose bit lengths are the same as the bit length of an element in $\mathbb{Z}_p$. Thus, the entire bit length of $2\mathbb{Z}_p$ is equal to that of $\mathbb{G}$. The main construction can achieve a similar storage size as other schemes.

As will be discussed in detail in the next subsection, the performance of the proposed schemes described above has been achieved as well as the provable security, which is the open problem in existing works.

C. SECURITY

In this section, we discuss the security required in KASE. The results are shown in Table 3.

In Table 3, the schemes by Cui et al. [9] and by Li et al. [11] do not satisfy the keyword privacy, the aggregate key unforgeability, and the trapdoor privacy. On the other hand, the scheme by Zhou et al. [22], the first construction, and the main construction satisfy the keyword privacy. Although Cui et al. and Li et al. have discussed the keyword privacy informally, their discussions do not include the provable security with reduction algorithms.

Although the scheme by Zhou et al. satisfies the keyword privacy and the trapdoor privacy, it does not satisfy the compactness. Zhou et al.'s scheme assumes a special situation where a remote sensor device encrypts its sensing data. This requires each sensor device to have an extra key for encryption, and the number of keys increases linearly with respect to the number of sensors when viewed across the system. Thus, the compactness cannot be satisfied. We also note that the scheme by Zhou et al. deals with a problem different from ours. Moreover, the aggregate key unforgeability has not been discussed explicitly in other works.

VII. CONCLUSION

In this paper, we proposed provably secure KASE scheme and defined the security of KASE formally. To the best of our knowledge, this is the first paper to provide a formal security discussion of KASE. Our main idea was to combine broadcast encryption and aggregate signatures, and we proposed the scheme called first construction by combining the broadcast encryption scheme by Boneh et al. and the aggregate signature scheme by Boneh et al. The security is provably secure with respect to the keyword privacy under the BDHE-assumption and the aggregate key unforgeability under the DHE assumption. Furthermore, by constructing trapdoors that utilize random numbers distributed via secret sharing, we proposed another scheme called main construction that satisfies the trapdoor privacy. We then implemented the proposed schemes and showed that both schemes could encrypt 5,000 keywords within one second. Moreover, a search in the 5,000 keywords can be executed within about three seconds in the first construction and about six seconds in the main construction. These results show that the proposed schemes are practical while achieving provably security. As future work, we plan to propose a generic construction through any broadcast encryption and any aggregate signatures because the proposed schemes in this paper are based on the specific constructions described above. We also plan to optimize implementation, e.g., parallelization of processes, to improve the performance of the proposed schemes. Furthermore, to guarantee stronger security of KASE, we plan to study file injection attacks [37] and passive attacks [38] against our schemes as further security analysis, which is not contained in the provable security.

ACKNOWLEDGMENT

The authors would like to thank members of the study group "Shin-Akarui-Angou-Benkyou-Kai" for the valuable discussions and helpful comments.

APPENDIX. PROOFS OF MAIN CONSTRUCTION

A. COMPLEXITY ASSUMPTIONS

First, we define the l-(D-)BDHE assumption [49]. This is an assumption in asymmetric bilinear groups.

Definition 9 ((ϵ, l)-(D-)BDHE Assumption in $(\mathbb{G}, \mathbb{H}, \mathbb{G}_T)$): We say the l-(D-)BDHE problem in $(\mathbb{G}, \mathbb{H}, \mathbb{G}_T)$ with a security parameter 1^k as, for a given $(g, g^{\alpha}, g^{\alpha^2}, \ldots, g^{\alpha^l}, g^{\alpha^{l+2}}, \ldots, g^{\alpha^{2l}}, h, h^s, \alpha, s \in \mathbb{Z}_p^*$ and $(\mathbb{G}, \mathbb{H}, \mathbb{G}_T)$ as input, determining whether $Z \in \mathbb{G}_T$ is $e(g^{\alpha^{l+1}}, h^s)$ or a random value R. We say that a polynomial time algorithm $\mathcal{A}$ can solve the l-(D-)BDHE problem in $(\mathbb{G}, \mathbb{H}, \mathbb{G}_T)$ with an advantage ϵ if the following relation holds:

$$|Pr[\mathcal{A}(g, h, h^s, \boldsymbol{y}_{g,h,\alpha,l}, e(g_{l+1}, h^s)) = 0] - Pr[\mathcal{A}(g, h, h^s, \boldsymbol{y}_{g,h,\alpha,l}, R) = 0]| \geq \epsilon,$$

where $\boldsymbol{y}_{g,h,\alpha,l} = (g_1, \ldots, g_l, g_{l+2}, \ldots, g_{2l})$. We say the l-(D-)BDHE assumption holds in $(\mathbb{G}, \mathbb{H}, \mathbb{G}_T)$ if there is no polynomial-time algorithm that can solve the l-(D-)BDHE problem in $(\mathbb{G}, \mathbb{H}, \mathbb{G}_T)$ with ϵ.

The difference of the assumption described above from the assumption is only the input, i.e., $h \in \mathbb{G}$ for the l-BDHE assumption and g^s for the l-(D-)BDHE assumption. Namely, the notation of the input is different.

Next, we define l-BDHE assumption and l-DHE assumption in asymmetric bilinear groups. We use these assumptions for the security proofs of the main construction. We note that the l-BDHE assumption in asymmetric bilinear groups is naturally extended from the l-(D-)BDHE assumption. In particular, in the following assumptions, when $\mathbb{G} = \mathbb{H}$ and $g = h$, the l-BDHE assumption in $(\mathbb{G}, \mathbb{H}, \mathbb{G}_T)$ is identical to the l-(D-)BDHE assumption in this section and the l-DHE assumption in $(\mathbb{G}, \mathbb{H}, \mathbb{G}_T)$ is identical to the l-DHE assumption in Section II-B.

Definition 10 ((ϵ, l)-BDHE Assumption in $(\mathbb{G}, \mathbb{H}, \mathbb{G}_T)$): We say the l-BDHE problem in $(\mathbb{G}, \mathbb{H}, \mathbb{G}_T)$ with a security parameter 1^k as, for a given $(g, g^s, g^{\alpha}, g^{\alpha^2}, \ldots, g^{\alpha^l}, g^{\alpha^{l+2}}, \ldots, g^{\alpha^{2l}}, h, h^s, h^{\alpha}, h^{\alpha^2}, \ldots, h^{\alpha^l}, Z)$ with uniformly random $g \in \mathbb{G}, h \in \mathbb{H}, \alpha, s \in \mathbb{Z}_p^*$ and $(\mathbb{G}, \mathbb{H}, \mathbb{G}_T)$ as input, determining whether $Z \in \mathbb{G}_T$ is $e(g^{\alpha^{l+1}}, h^s)$ or a random value R. We say that a polynomial time algorithm $\mathcal{A}$ can solve the l-BDHE problem in $(\mathbb{G}, \mathbb{H}, \mathbb{G}_T)$ with an advantage ϵ if the following relation holds:

$$|Pr[\mathcal{A}(g, g^s, h, h^s, \boldsymbol{y}_{g,h,\alpha,l}, e(g_{l+1}, h^s)) = 0] - Pr[\mathcal{A}(g, g^s, h, h^s, \boldsymbol{y}_{g,h,\alpha,l}, R) = 0]| \geq \epsilon,$$

where $\boldsymbol{y}_{g,h,\alpha,l} = (g_1, \ldots, g_l, g_{l+2}, \ldots, g_{2l}, h_1, \ldots, h_l)$. We say the l-BDHE assumption holds in $(\mathbb{G}, \mathbb{H}, \mathbb{G}_T)$ if there is no polynomial-time algorithm that can solve the l-BDHE problem in $(\mathbb{G}, \mathbb{H}, \mathbb{G}_T)$ with ϵ.

Definition 11 ((ϵ, l)-DHE Assumption in $(\mathbb{G}, \mathbb{H})$): We say the l-DHE problem in $(\mathbb{G}, \mathbb{H})$ with a security parameter 1^k as, for a given $(g, g^{\alpha}, g^{\alpha^2}, \ldots, g^{\alpha^l}, g^{\alpha^{l+2}}, \ldots, g^{\alpha^{2l}}, h, h^{\alpha}, h^{\alpha^2}, \ldots, h^{\alpha^l})$ with uniformly random $g \in \mathbb{G}, h \in \mathbb{H}, \alpha \in \mathbb{Z}_p^*$ and $(\mathbb{G}, \mathbb{H}, \mathbb{G}_T)$ as input, computing $g^{\alpha^{l+1}}$. We say that a polynomial time algorithm $\mathcal{A}$ can solve the l-DHE problem in $(\mathbb{G}, \mathbb{H})$ with an advantage ϵ if the following relation holds:

$$Pr[\mathcal{A}(g, \boldsymbol{y}_{g,h,\alpha,l}, g^{\alpha^{l+1}})] \geq \epsilon,$$

where $\boldsymbol{y}_{g,h,\alpha,l} = (g^{\alpha}, g^{\alpha^2}, \ldots, g^{\alpha^l}, g^{\alpha^{l+2}}, \ldots, g^{\alpha^{2l}}, h^{\alpha}, h^{\alpha^2}, \ldots, h^{\alpha^l})$. We say the l-DHE assumption holds in $(\mathbb{G}, \mathbb{H})$ if there is no polynomial-time algorithm that can solve the l-DHE problem in $(\mathbb{G}, \mathbb{H})$ with ϵ.

B. PROOF FOR KEYWORD PRIVACY

In this section, we show that the main construction satisfies the keyword privacy.

Theorem 4 ((ϵ', n)-Keyword Privacy): The main construction satisfies the (ϵ', n)-keyword privacy under the (ϵ, n)-BDHE assumption in $(\mathbb{G}, \mathbb{H}, \mathbb{G}_T)$, where $\epsilon \geq \epsilon'$.

Proof. Suppose there exists as adversary $\mathcal{A}$, whose advantage is ϵ', against the main construction. We then build an algorithm $\mathcal{B}$ that solves the BDHE problem in $(\mathbb{G}, \mathbb{H}, \mathbb{G}_T)$. Let $\mathcal{C}$ be a challenger for the BDHE problem in $(\mathbb{G}, \mathbb{H}, \mathbb{G}_T)$. Algorithm $\mathcal{B}$ proceeds as follows.

- Init: $\mathcal{A}$ declares a challenge document index $i^* \in [1, n]$ and sends it to $\mathcal{B}$.
- Setup: $\mathcal{C}$ sends $(g, g^s, g_1, g_2, \ldots, g_n, g_{n+2}, \ldots, g_{2n}, h, h^s, h_1, h_2, \ldots, h_n, h_{n+2}, \ldots, h_{2n}, Z)$ to $\mathcal{B}$. $\mathcal{B}$ randomly generates $sk = \beta$ and calculates $v' = g^{\beta} g_{i^*}^{-1}$. $\mathcal{B}$ sends $params = (g, g_1, g_2, \ldots, g_n, g_{n+2}, \ldots, g_{2n}, h, h_1, h_2, \ldots, h_n)$ to $\mathcal{A}$.
- Query: When $\mathcal{A}$ queries for $\mathcal{O}_{Extract}$, $\mathcal{B}$ responds as follows:
  - If an aggregate key for $i^* \in S$ is queried, return $\perp$.
  - If an aggregate key for $i^* \notin S$ is queried, return $k_{agg} = (\Pi_{j \in S} g_{n+1-j}^{\beta}) \cdot (\Pi_{j \in S} g_{n+1-j+i^*})^{-1} = \Pi_{j \in S} g_{n+1-j}^{\beta - \alpha^{i^*}}$. Note that, for $i^* \notin S$, $(\Pi_{j \in S} g_{n+1-j+i^*})^{-1}$ can be calculated.

  When $\mathcal{A}$ queries for $\mathcal{O}_{Encrypt}$, $\mathcal{B}$ randomly generates $t_{i,l} \in \mathbb{Z}_p^*$, calculates the following $c_{i,l} = (c_{1,i,l}, c_{2,i,l}, c_{3,i,l})$, and responds to $\mathcal{A}$ ($c_{1,i,l} = h^{t_{i,l}}, c_{2,i,l} = (v' \cdot g_i)^{t_{i,l}}, c_{3,i,l} = \frac{e(H(w_l), h)^{t_{i,l}}}{e(g_1, h_n)^{t_{i,l}}}$).
- Guess: $\mathcal{A}$ declares the challenge keyword w_{l^*} and sends it to $\mathcal{B}$. $\mathcal{B}$ calculates the challenge encrypted keyword $c_{1,i^*,\theta} = h^s, c_{2,i^*,\theta} = (g^s)^{\beta}, c_{3,i^*,\theta} = \frac{e(H(w_{l^*}), h^s)}{Z}$. Then, when $Z = e(g_{n+1}, h^s)$, $c_{1,i^*,\theta} = h^s, c_{2,i^*,\theta} = ((g^{\beta} g_{i^*}^{-1}) \cdot g_{i^*})^s = g^{\beta s} = (g^s)^{\beta}, c_{3,i^*,\theta} = \frac{e(H(w_{l^*}), h)^s}{e(g_1, h_n)^s} = \frac{e(H(w_{l^*}), h^s)}{e(g_{n+1}, h^s)} = \frac{e(H(w_{l^*}), h^s)}{Z}$. Therefore, the calculation results are identical to the Encrypt algorithm of the main construction. $\mathcal{B}$ sends $c_{i^*,\theta} = (c_{1,i^*,\theta}, c_{2,i^*,\theta}, c_{3,i^*,\theta})$ to $\mathcal{A}$. $\mathcal{A}$ chooses $\theta' \in \{0, 1\}$ and sends it to $\mathcal{B}$. Then, $\mathcal{B}$ sends θ' to $\mathcal{C}$ as a guess of θ.

In the guess phase, if Z is a random value, then $\Pr[\theta = \theta'] = 1/2$. On the other hand, if $Z = e(g_{n+1}, h^s)$, $|\Pr[\theta = \theta'] - 1/2| > \epsilon'$. This indicates that $\mathcal{B}$ has an advantage over ϵ' for solving the (ϵ, n)-BDHE problem in $(\mathbb{G}, \mathbb{H}, \mathbb{G}_T)$. Thus, if the (ϵ, n)-BDHE assumption holds in $(\mathbb{G}, \mathbb{H}, \mathbb{G}_T)$, the main construction satisfies the (ϵ', n)-keyword privacy.

C. PROOF FOR AGGREGATE KEY UNFORGEABILITY

In this section, we show that the main construction satisfies the aggregate key unforgeability.

Theorem 5 ((ϵ', n)-Aggregate key Unforgeability): The main construction satisfies the (ϵ', n)-aggregate key unforgeability under the (ϵ, n)-DHE Assumption in $(\mathbb{G}, \mathbb{H})$, where $\epsilon = \epsilon'$.

Proof: Suppose there exists as adversary $\mathcal{A}$, whose advantage is ϵ', against the main construction. We then build an algorithm $\mathcal{B}$ that solves the DHE problem. Let $\mathcal{C}$ be a challenger for the DHE problem. Algorithm $\mathcal{B}$ proceeds as follows.

- Setup: $\mathcal{C}$ sends $(g, g_1, g_2, \ldots, g_n, g_{n+2}, \ldots, g_{2n}, h, h_1, h_2, \ldots, h_n, h_{n+2}, \ldots, h_{2n})$ to $\mathcal{B}$. $\mathcal{B}$ randomly generates $sk = \beta$ and calculates $v' = g^{\beta} g_{i^*}^{-1}$. $\mathcal{B}$ sends $params = (g, g_1, g_2, \ldots, g_n, g_{n+2}, \ldots, g_{2n}, h, h_1, h_2, \ldots, h_n)$ to $\mathcal{A}$.
- Query: When $\mathcal{A}$ queries for $\mathcal{O}_{Extract}$, $\mathcal{B}$ responds as follows:
  - If an aggregate key for $i^* \in S$ is queried, return $\perp$.
  - If an aggregate key for $i^* \notin S$ is queried, return $k_{agg} = (\Pi_{j \in S} g_{n+1-j}^{\beta}) \cdot (\Pi_{j \in S} g_{n+1-j+i^*})^{-1} = \Pi_{j \in S} g_{n+1-j}^{\beta - \alpha^{i^*}}$. Here, for $i^* \notin S$, $(\Pi_{j \in S} g_{n+1-j+i^*})^{-1}$ can be calculated.

  When $\mathcal{A}$ queries for $\mathcal{O}_{Encrypt}$, $\mathcal{B}$ randomly generates $t_{i,l} \in \mathbb{Z}_p^*$, calculates $c_{i,l} = (c_{1,i,l}, c_{2,i,l}, c_{3,i,l})$ and responds to $\mathcal{A}$ ($c_{1,i,l} = h^{t_{i,l}}, c_{2,i,l} = (v' \cdot g_i)^{t_{i,l}}$, $c_{3,i,l} = \frac{e(H(w_l),h)^{t_{i,l}}}{e(g_1,h_n)^{t_{i,l}}}$).
- Forge: $\mathcal{A}$ outputs S^*, k_{agg}^* and sends them to $\mathcal{B}$.
  - If $i^* \notin S^*$, abort
  - If $i^* \in S^*$, $k_{agg}^* = (\Pi_{j \in S, j \neq i^*} g_{n+1-j})^{\beta - \alpha^{i^*}} \cdot (g_{n+1-i^*})^{\beta - \alpha^{i^*}}$. By using this k_{agg}^*, $\mathcal{B}$ calculates $(\Pi_{j \in S^*, j \neq i^*} g_{n+1-j})^{\beta} \cdot (\Pi_{j \in S^*, j \neq i^*} g_{n+1-j+i^*})^{-1} \cdot (g_{n+1-i^*})^{\beta} / k_{agg}^* = g^{\alpha+1}$ and outputs results.

The result in the above (ϵ', n)-aggregate key unforgeability game is identical to the answer of the (ϵ, n)-DHE problem in $(\mathbb{G}, \mathbb{H})$. That is, the advantage of the aggregate key unforgeability game is equal to the advantage of the (ϵ, n)-DHE problem in $(\mathbb{G}, \mathbb{H})$. Thus, if the (ϵ, n)-DHE assumption holds, the main construction satisfies the (ϵ', n)-aggregate key unforgeability.

REFERENCES

[1] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proc. IEEE Symp. Secur. Privacy S&P*, Nov. 2002, pp. 44–55.
[2] R. A. Popa and N. Zeldovich, "Multi-key searchable encryption," *IACR Cryptol. ePrint Arch.*, vol. 2013, p. 508, Aug. 2013.
[3] A. Kiayias, O. Oksuz, A. Russell, Q. Tang, and B. Wang, "Efficient encrypted keyword search for multi-user data sharing," in *Proc. Eur. Symp. Res. Comput. Secur.* Cham, Switzerland: Springer, 2016, pp. 173–195.
[4] A. Hamlin, A. Shelat, M. Weiss, and D. Wichs, "Multi-key searchable encryption, revisited," in *Proc. IACR Int. Workshop Public Key Cryptogr.* Cham, Switzerland: Springer, 2018, pp. 95–124.
[5] F. Bao, R. H. Deng, X. Ding, and Y. Yang, "Private query on encrypted data in multi-user settings," in *Proc. Int. Conf. Inf. Secur. Pract. Exper.* Berlin, Germany: Springer, 2008, pp. 71–85.
[6] Y. H. Hwang and P. J. Lee, "Public key encryption with conjunctive keyword search and its extension to a multi-user system," in *Proc. Int. Conf. Pairing-Based Cryptogr.* Berlin, Germany: Springer, 2007, pp. 2–22.
[7] C. V. Rompay, R. Molva, and M. Önen, "Multi-user searchable encryption in the cloud," in *Proc. Int. Inf. Secur. Conf.* Berlin, Germany: Springer, 2015, pp. 299–316.
[8] F. Zhao, T. Nishide, and K. Sakurai, "Multi-user keyword search scheme for secure data sharing with fine-grained access control," in *Proc. Int. Conf. Inf. Secur. Cryptol.* Berlin, Germany: Springer, 2011, pp. 406–418.
[9] B. Cui, Z. Liu, and L. Wang, "Key-aggregate searchable encryption (KASE) for group data sharing via cloud storage," *IEEE Trans. Comput.*, vol. 65, no. 8, pp. 2374–2385, Aug. 2016.
[10] T. Li, Z. Liu, C. Jia, Z. Fu, and J. Li, "Key-aggregate searchable encryption under multi-owner setting for group data sharing in the cloud," *Int. J. Web Grid Services*, vol. 14, no. 1, pp. 21–43, 2018.
[11] T. Li, Z. Liu, P. Li, C. Jia, Z. L. Jiang, and J. Li, "Verifiable searchable encryption with aggregate keys for data sharing in outsourcing storage," in *Proc. Australas. Conf. Inf. Secur. Privacy*. Berlin, Germany: Springer, 2016, pp. 153–169.
[12] D. Boneh, C. Gentry, and B. Waters, "Collusion resistant broadcast encryption with short ciphertexts and private keys," in *Proc. Annu. Int. Cryptol. Conf.* Berlin, Germany: Springer, 2005, pp. 258–275.
[13] A. Fiat and M. Naor, "Broadcast encryption," in *Proc. Annu. Int. Cryptol. Conf.* Berlin, Germany: Springer, 1993, pp. 480–491.
[14] D. Boneh, C. Gentry, B. Lynn, and H. Shacham, "Aggregate and verifiably encrypted signatures from bilinear maps," in *Proc. Int. Conf. Theory Appl. Cryptograph. Techn.* Berlin, Germany: Springer, 2003, pp. 416–432.
[15] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, 1979.
[16] Y. Wu, Z. Wei, and R. Deng, "Attribute-based access to scalable media in cloud-assisted content sharing networks," *IEEE Trans. Multimedia*, vol. 15, no. 4, pp. 778–788, Jun. 2013.
[17] G. S. Poh, P. Gope, and J. Ning, "PrivHome: Privacy-preserving authenticated communication in smart home environment," *IEEE Trans. Depend. Sec. Comput.*, to be published.
[18] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2007, pp. 321–334.
[19] Y. Yao, Z. Zhai, J. Liu, and Z. Li, "Lattice-based key-aggregate (searchable) encryption in cloud storage," *IEEE Access*, vol. 7, pp. 164544–164555, 2019.
[20] H. Wang, X. Dong, Z. Cao, D. Li, and N. Cao, "Secure key-aggregation authorized searchable encryption," *Sci. China Inf. Sci.*, vol. 62, no. 3, 2019, Art. no. 39111.
[21] M. Padhya and D. C. Jinwala, "MULKASE: A novel approach for key-aggregate searchable encryption formulti-owner data," in *Proc. Frontiers Inf. Technol. Electron. Eng.*, 2018, pp. 1–32.
[22] R. Zhou, X. Zhang, X. Du, X. Wang, G. Yang, and M. Guizani, "File-centric multi-key aggregate keyword searchable encryption for industrial Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 14, no. 8, pp. 3648–3658, Aug. 2018.
[23] S. Patranabis and D. Mukhopadhyay, "Key-aggregate searchable encryption with constant-size trapdoors for fine-grained access control in the cloud," *IACR Cryptol. ePrint Arch.*, vol. 2017, p. 318, Apr. 2017.
[24] J. Zhang, C. Song, Z. Wang, T. Yang, and W. Ma, "Efficient and provable security searchable asymmetric encryption in the cloud," *IEEE Access*, vol. 6, pp. 68384–68393, 2018.
[25] D. N. Wu, Q. Q. Gan, and X. M. Wang, "Verifiable public key encryption with keyword search based on homomorphic encryption in multi-user setting," *IEEE Access*, vol. 6, pp. 42445–42453, 2018.
[26] J. Ye, J. Wang, J. Zhao, J. Shen, and K.-C. Li, "Fine-grained searchable encryption in multi-user setting," *Soft Comput.*, vol. 21, no. 20, pp. 6201–6212, Oct. 2017.

[27] Q. Wang, Y. Zhu, and X. Luo, "Multi-user searchable encryption with coarser-grained access control without key sharing," in *Proc. Int. Conf. Cloud Comput. Big Data*, Nov. 2014, pp. 119–125.
[28] G. Wang, C. Liu, Y. Dong, P. Han, H. Pan, and B. Fang, "IDCrypt: A multi-user searchable symmetric encryption scheme for cloud applications," *IEEE Access*, vol. 6, pp. 2908–2921, 2018.
[29] 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.
[30] Q. Zheng, S. Xu, and G. Ateniese, "VABKS: Verifiable attribute-based keyword search over outsourced encrypted data," in *Proc. IEEE INFOCOM-IEEE Conf. Comput. Commun.*, Apr. 2014, pp. 522–530.
[31] H. Yin, J. Zhang, Y. Xiong, L. Ou, F. Li, S. Liao, and K. Li, "CP-ABSE: A ciphertext-policy attribute-based searchable encryption scheme," *IEEE Access*, vol. 7, pp. 5682–5694, 2019.
[32] A. Alrawais, A. Alhothaily, C. Hu, X. Xing, and X. Cheng, "An attribute-based encryption scheme to secure fog communications," *IEEE Access*, vol. 5, pp. 9131–9138, 2017.
[33] S. Wang, D. Zhang, Y. Zhang, and L. Liu, "Efficiently revocable and searchable attribute-based encryption scheme for mobile cloud storage," *IEEE Access*, vol. 6, pp. 30444–30457, 2018.
[34] C. Guo, N. Luo, M. Z. A. Bhuiyan, Y. Jie, Y. Chen, B. Feng, and M. Alam, "Key-aggregate authentication cryptosystem for data sharing in dynamic cloud storage," *Future Gener. Comput. Syst.*, vol. 84, pp. 190–199, Jul. 2018.
[35] C.-K. Chu, S. S. M. Chow, W.-G. Tzeng, J. Zhou, and R. H. Deng, "Key-aggregate cryptosystem for scalable data sharing in cloud storage," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 2, pp. 468–477, Feb. 2014.
[36] S. Patranabis, Y. Shrivastava, and D. Mukhopadhyay, "Provably secure key-aggregate cryptosystems with broadcast aggregate keys for online data sharing on the cloud," *IEEE Trans. Comput.*, vol. 66, no. 5, pp. 891–904, May 2017.
[37] Y. Zhang, J. Katz, and C. Papamanthou, "All your queries are belong to us: The power of file-injection attacks on searchable encryption," in *Proc. 25th USENIX Secur. Symp. (USENIX Secur.)*, 2016, pp. 707–720.
[38] J. Ning, J. Xu, K. Liang, F. Zhang, and E.-C. Chang, "Passive attacks against searchable encryption," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 3, pp. 789–802, Mar. 2019.
[39] A. Joux, "A one round protocol for tripartite Diffie–Hellman," in *Proc. Int. Algorithmic Number Theory Symp.* Berlin, Germany: Springer, 2000, pp. 385–393.
[40] J. Herranz, F. Laguillaumie, B. Libert, and C. Ràfols, "Short attribute based signatures for threshold predicates," in *Proc. Cryptographers' Track RSA Conf.* Berlin, Germany: Springer, 2012, pp. 51–67.
[41] G. Ateniese, J. Camenisch, and B. De Medeiros, "Untraceable RFID tags via insubvertible encryption," in *Proc. 12th ACM Conf. Comput. Commun. Secur.-CCS*, 2005, pp. 92–101.
[42] L. Ballard, M. Green, B. D. Medeiros, and F. Monrose, "Correlation-resistant storage via keyword-searchable encryption," *IACR Cryptol. ePrint Arch.*, vol. 2005, p. 417, Nov. 2005.
[43] 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. 2014.
[44] A. Arriaga, Q. Tang, and P. Ryan, "Trapdoor privacy in asymmetric searchable encryption schemes," in *Proc. Int. Conf. Cryptol. Afr.* Cham, Switzerland: Springer, 2014, pp. 31–50.
[45] 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, Nov. 2011.
[46] S. Kamara and C. Papamanthou, "Parallel and dynamic searchable symmetric encryption," in *Proc. Int. Conf. Financial Cryptogr. Data Secur.* Berlin, Germany: Springer, 2013, pp. 258–274.
[47] Y. G. Desmedt, "Threshold cryptography," *Eur. Trans. Telecommun.*, vol. 5, no. 4, pp. 449–458, 1994.
[48] D. Boneh, X. Boyen, and S. Halevi, "Chosen ciphertext secure public key threshold encryption without random oracles," in *Proc. Cryptographers' Track RSA Conf.* Berlin, Germany: Springer, 2006, pp. 226–243.
[49] X. Boyen, "The uber-assumption family," in *Proc. Int. Conf. Pairing-Based Cryptogr.* Berlin, Germany: Springer, 2008, pp. 39–56.

MASAHIRO KAMIMURA received the B.Eng. degree in engineering science from Osaka University, Japan, in 2018. He is currently pursuing the M.S. degree with the Graduate School of Information Science and Technology, Osaka University, Japan. His research interest includes information security.

NAOTO YANAI (Member, IEEE) received the B.Eng. degree from the National Institution of Academic Degrees and University Evaluation, Japan, in 2009, the M.S.Eng. from the Graduate School of Systems and Information Engineering, University of Tsukuba, Japan, in 2011, and the Dr.E. degree from the Graduate School of Systems and Information Engineering, University of Tsukuba, in 2014. He is currently an Assistant Professor with Osaka University, Japan. His research areas are cryptography and information security.

SHINGO OKAMURA (Member, IEEE) received the B.E., M.E., and Ph.D. degrees in information science and technology from Osaka University, in 2000, 2002, and 2005, respectively. Since 2005, he has worked for Osaka University. In 2008, he joined the National Institute of Technology, Nara College, where he is currently an Associate Professor. His research interests include cryptographic protocols and cyber security. He is a member of the IEICE, IEEJ, ACM, and IACR.

JASON PAUL CRUZ (Member, IEEE) received the B.S. degree in electronics and communications engineering and the M.S. degree in electronics engineering from the Ateneo de Manila University, Quezon City, Philippines, in 2009 and 2011, respectively, and the Ph.D. degree in engineering from the Graduate School of Information Science, Nara Institute of Science and Technology, Nara, Japan, in 2017. He is currently a Specially Appointed Assistant Professor with Osaka University, Osaka, Japan. His current research interests include role-based access control, blockchain technology, hash functions and algorithms, privacy-preserving cryptography, and Android programming.