Received October 7, 2020, accepted November 8, 2020, date of publication November 18, 2020, date of current version January 20, 2021.

Digital Object Identifier 10.1109/ACCESS.2020.3039040

Efficient SSE With Forward ID-Privacy and Authentication in the Multi-Data-Owner Settings

HUIGE WANG[1,2], GUANGYE SUI[2], YUNLEI ZHAO[2], AND KEFEI CHEN[3,4]
[1]Department of Information Security, Anhui Science and Technology University, Bengbu 233030, China
[2]School of Computer Science, Fudan University, Shanghai 200433, China
[3]School of Science, Hangzhou Normal University, Hangzhou 310036, China
[4]Westone Cryptologic Research Center, Beijing 100070, China

Corresponding author: Yunlei Zhao (wanghuige@fudan.edu.cn)

The work of Huige Wang was supported in part by the National Key Research and Development Program of China under Grant 2017YFB0802000, and in part by the National Natural Science Foundation of China under Grant NSFC61702007 and Grant NSFC61572318, and in part by Other Foundations under Grant 2019M661360, Grant gxbjZD27,KJ2018A0533, Grant XWWD201801, Grant ahnis20178002, Grant KJ2017A519, and Grant 16ZB0140. The work of Guangye Sui was supported in part by the National Key Research and Development Program of China under Grant 2017YFB0802000, in part by the National Natural Science Foundation of China under Grant 61877011, Grant 61472084, and Grant U1536205, in part by the Shanghai Innovation Action Project under Grant 16DZ1100200, in part by the Shanghai Science and Technology Development Funds under Grant 16JC1400801, and in part by the Shandong Provincial Key Research and Development Program of China under Grant 2017CXG0701 and Grant 2018CXGC0701. The work of Kefei Chen was supported in part by the National Key Research and Development Program of China under Grant 2017YFB0802000, and in part by the National Natural Science Foundation of China under Grant U1705264.

ABSTRACT Based on Sun *et al.*'s multi-client symmetric searchable encryption (SSE) scheme (at ESORICS 2016), and combining Zhao's identity-concealed authenticated encryption (CCS 2016), a new SSE scheme with multi-data-owner functionalities is proposed. By setting two key generation centers, our scheme first implements multi-data-owner SSE. In particular, compared with Sun *et al.*'s scheme, the new scheme not only meets the same security requirements stated by them, but also further strengthens the securities of the same category relevant scheme by providing identity-concealment, authentication of data user to server and confidentiality of search token. The identity-concealment aims to provide privacy protection (Forward ID-Privacy) for data users by hiding their identity information, while the authentication is to resist the camouflage attack by applying certificate-based mechanism to our scheme. In particular, the confidentiality of the search token provides replay-attack-resistant by encrypting the plaintext search token generated by data user. While in other works, the adversary can employ the previously generated plaintext search tokens to force the server to perform the same search queries. Furthermore, by efficiency analysis, our scheme reaches almost the same level of efficiency as Sun *et al.*'s scheme.

INDEX TERMS Symmetric searchable encryption, identity-concealment, authentication, confidentiality, multi-data-owner.

I. INTRODUCTION

As people generate more and more data every day, cloud storage technology has been widely practiced and reveals its promising future. However, when the cloud server undertakes outsourcing tasks of the data, the privacy and security problem face enormous challenges. One intuitive solution is uploading the encrypted data to the cloud server, then decrypt the entire file when a data user wants to use the data. This kind of operations led to huge computational power costs in the data user side, especially when the size of file is huge. Searchable encryption(SE) is a cryptosystem that can migrate the time-consuming search task to the server side and support keyword search on the ciphertexts directly.

According to the key used in a system, searchable encryption(SE) can be classified into two categories: symmetric

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

searchable encryption (SSE) [23], [29] [1], [34] [2], [3] and asymmetric searchable encryption (ASE) [4]–[6] In SSE, the key used in creating encrypted database is the same as or highly related with that used to decrypt the encrypted database, while in ASE, the key used in creating encrypted database is the public key and the key used to decrypt is the private key and the two key is different. Since SSE is characterized by low computational overhead, simple algorithm and high speed, this article will mainly focus on the SSE cryptosystem.

According to the number of data users or owners that use or own databases, symmetric searchable encryption is typically divided into 4 flavours: single-data-owner/single-data-user (sDO-sDU), single-data-owner/multi-data-user (sDO-mDU), multi-data-owner/single-data-user (mDO-sDU), multi-data-owner/multi-data-user (mDO-mDU). The sDO-sDU SSE is the simplest where only one data owner is allowed to use a specified encryption algorithm to convert a plaintext database into a ciphertext database and outsource its storage and service. After that only allowing one data user (including the data owner) to make queries on the database. More concretely, the data user first gets a search key on an authorized keyword set decided by the data owner, and then generates a search token (which is then sent to the server) using this key and an authorized keyword subset. After receiving the token, the server sent the ciphertext set to the data user who then decrypts the result set to get the final query document indexes using his search key. In the mDO-mDU setting, a database can be divided into multiple sub-databases and the search query on these databases can be executed by multiple data users. Thanks to the powerful capabilities of the mDO-mDU SSE, in the rest of this article, we concentrate on researching it.

A. MOTIVATIONS

In [34], Sun *et al.* proposed an efficient non-interactive sDO-mDU SSE, which improved the scheme in [29] by enhancing the communication efficiency between the data owner and the data user. Concretely, the scheme in [29] requires the data user to make interaction with the data owner every time once she prepares a search on an encrypted database, while in Sun *et al.*' scheme [34], no matter how many times a data user executes a query, he only interacts with the owner once, as long as the query keywords are in the authorized keyword set. Nevertheless, despite the advantages in [34], there are still many other problems not resolved in their article. For example, their scheme only works in the sDO-sDU setting, and how to design a scheme working in the mDO-mDU settings is still an open problem. Especially, in their scheme, since the identity information of data user is transmitted publicly to the server and no authentication from the data user to the server is provided when the data user is interacting with the server, the above scheme is easily subject to disguised attacks. In addition, since the search token generated by the data user is transmitted in a clear text and authentication is also not provided to the server, their scheme is vulnerable to the replay search attack on the plaintext token. Based on the above analysis, we put forward the following questions.

Motivation 1: Can we propose an SSE scheme that supports the mDO-mDU setting and shares the same efficiency advantage as Sun et al's scheme?

The authentication protocol with identity-concealment is an efficient cryptographic tool for achieving secure end-to-end communication over the Internet. Implementing authentication and identity-concealment in searchable encryption is important for enhancing the security and privacy. However, existing searchable encryption schemes, especially the SSE scheme in [34], does not provide authentication, data protection and identity-concealment for client-sever communication, which can lead to serious security risks. For example, an adversary can pretend to be a client to initiate a large number of search queries to the server so as to result in the disguised attack and DoS attack. Moreover, if no protection is provided to search token, an adversary can use a previously generated one to launch a large-scale search query against the server thereby lead to a replay search attack. In addition, without identity concealment, the adversary can obtain more sensitive information about the data user through the identity information (through human flesh search, etc.), which will seriously compromise the privacy of the data user. Intuitively, we should use standard protocols like TLS1.3 or Google's QUIC to solve above problems. However, these protocols provide too many functions we do not need, which will deteriorate the efficiency problem of SSE and make our scheme less practical. In the public-key setting, authenticated encryption refers to signcryption [8], which is functionally equivalent to one-pass authenticated key-exchange [9], [10] [11] and has applications in asymmetrical key-exchange settings [12]. Zheng's signcrytion [8], [14] [15] and one-pass HMQV (HOMQV) [12], [13] are potential solutions for our problems, but they did not consider the ID concealment issue and their efficiency is still not satisfactory. Identity is a fundamental privacy concern and implementing identity-confidentiality is now mandated by a list of prominent standards such as TLS1.3 [17], EMV [18], QUIC [19], and the 5G telecommunication standard [20] by 3GPP, etc, and is enforced by General Data Protection Regulation (GDPR) of EU.

Higncryption is a cryptographic primitive proposed by Zhao [37], which implements identity-concealment, authentication and confidentiality simultaneously in a system. Compare to signcryption and HOMQV, higncryption has similar efficiency, but achieves higher security (such as CMIM, UKS, KCI, CNM, PFS, x-disclosure, etc. The reader can refer to [37] for details.) and provides more functionalities. Moreover, higncryption fits perfectly to TLS1.3 [17] and QUIC [19]. Based on these, we propose the following question.

Motivation 2: Can we propose a symmetric searchable encryption that also simultaneously supports identity-concealment, authentication and confidentiality?

B. CONTRIBUTIONS

Concretely, our contributions are listed below.

- **Multi-data-owner functionality:**Building on the work presented in [34], we propose an SSE scheme which supports any boolean queries and mDO-mDU setting by introducing multi-data-owner functionality. We achieve this by firstly setting up a data owner key generation center (DWK-KGC) to generate an encrypted database public/private key pair for each data owner. Then we use each data owner's key pair to convert his plaintext database into a ciphertext database.
- **Identity-concealment, authentication and confidentiality of search token:** In particular, compared with Sun's scheme [34], our scheme not only meets the same security requirements stated by them, but also further strengthens the securities of the same category relevant scheme by providing identity-concealment, authentication of data user to server and confidentiality of search token. We implement them by modifying and integrating Zhao's higncryption scheme [37] into our scheme. The identity-concealment aims to provide privacy protection for data users by hiding their identity information, while the authentication is to resist the camouflage attack by applying certificate-based mechanism to our scheme. The confidentiality of the search token provides replay-attack-resistant by encrypting the plaintext search token. While in other similar works, the adversary can employ the previously generated plaintext search tokens to force the server to perform the same search queries many times.
- **Forward ID-privacy:** Our scheme provides forward ID-privacy properties, which means one client's identity-privacy (i.e., ID-privacy) preserves even when his long-term private key is revealed. Please note that neither signcryption nor HOMQV can achieve ID concealment and forward ID-privacy.
- **Efficiency analysis:** Our scheme achieves the comparable performance with Sun's scheme except that the computation cost for each data user in our scheme is 2.5 exponent operations more than that of theirs. In addition, like Zhao's higncryption scheme [37], our scheme can also directly apply to 0-RTT protocol, showing that our scheme is compatible well with the QUIC and OPTLS-based SSE scheme.

C. RELATED WORKS

In 2000, Song *et al.* [33] gave an scheme for searching on ciphertext for the first time, but because the scheme needs to scan the full document, the algorithm is less efficient and vulnerable to statistical attacks. To improve search efficiency and security, Goh [26] proposed the concept of secure indexing and IND-CKA security model for SE. In the process of querying, the server only searches the index without directly operating on the ciphertext, which greatly improves the efficiency. In 2006, a strong security notion IND-CKA2 is proposed by Curtmola *et al.* [25]. Later, Kurosawa and Ohtaki [30] proposed the IND-CKA2 security notion in the UC framework. In order to overcome the accuracy rate problem caused by Bloom filter, Chang and Mitzenmacher [24] introduced the concept of key dictionary and proposed the first deterministic SE scheme.

Since the SE with single-keyword query is non-adaptable with the multi-keyword search in real life application. In 2004, Golle *et al.* [27] gave two SSE schemes for connection keyword search with linear time complexity. To improve search efficiency, the first SSE scheme that supports boolean queries and logarithmic time complexity was proposed by Cash *et al.* [23]. In 2003, Cash *et al.* [23] first proposed multi-client (i.e., multi-data-user) SSE with linear communication overhead in the number of queries between data user and data owner. In 2016, Sun *et al.* [34] proposed a non-interactive mDO-sDU SSE that supports boolean queries, which improved the scheme in [29] to obtain constant communication overhead and is independent of the number of queries. In 2017, Rompay *et al.* [31] proposed an mDO-sDU SSE scheme that can resist leakage attacks.

In 2007, using Attribute-Base Encryption(ABE), Identity-based encryption(IBE), Boneh and Waters [21] proposed a new SE scheme for privacy search. In 2014, Sun *et al.* [35] and Shi *et al.* [32] presented an Attribute-Base SE schemes that supports withdraw and keyword combination queries under certain conditions, respectively. In 2015, Zheng *et al.* [28] gave two searchable ABE schemes. These two schemes not only realized keyword retrieval but also provided verifiability of query results. In 2018, using fine-grained access control properties of ABE, Wan and Deng [36] gave an SE scheme with multi-keyword searchable encryption that has fine-grained access control functionalities. In 2020, Leilei Du *et al.* [38] proposed a dynamic multi-client searchable symmetric encryption with support for boolean queries which allowed multiple clients to perform boolean queries over an encrypted database. In 2019, Zarezadeh *et al.* [39] put forward a multi-keyword ranked searchable encryption scheme with access control for cloud storage which solved the problems in index construction, trapdoor generation and search procedures existed in [45] by proposing a new multi-keyword ranked search encryption scheme. In 2019, Kermanshahi *et al.* [40] gave a multi-client cloud-based symmetric searchable encryption that removed the need for a constant online presence of the data owner to provide services to the users with supporting the property of the user key-exposure-resilience. In 2019, Cong Zuo *et al.* [41] proposed a dynamic searchable symmetric encryption with forward and stronger backward privacy with one roundtrip. In 2019, Jin Li *et al.* [43] put forward an SSE scheme in a new forward search privacy security model, in which, the search operation over freshly search document does not leak sensitive information to the past queried. In 2020, Jing Chen *et al.* [44] proposed a dynamic searchable symmetric encryption with forward security and low storage overhead. However, all the above schemes are implemented in the single-data-owner setting, and do not provide identity-concealment, authentication of data user to server and confidentiality of search token.

D. ORGANIZATION

This article is structured as follows: Section II gives some important notations and definitions, Section III, IV and V present our scheme and provide a security analysis. Section VI gives a performance Analysis. Section VII shows some concluding remarks and possible future directions.

II. PRELIMINARY

In this section, we list a series of notations, assumptions, terminologies and cryptographic components that will be used in our construction.

A. NOTATIONS

λ: the security parameter; ind_i: the i-th document index; W_i: the set of keywords contained in the i-th document; $\mathsf{DB} = (\mathsf{ind}_i, W_i)$: a database consisting of a vector of document index and keyword-set pairs; $\mathsf{DB}[w] = \{\mathsf{ind}_i : w \in W_i\}$: the set of indexes that match the keyword w; $W = \bigcup_{i=1}^{d} W_i$: keyword set in database; $[T]$: $\{1, \cdots, T\}$; sterm: the least frequent term among the queried terms/keywords in a search query; xterm: other queried terms in a search query except the sterm. For $X = g^x, Y = g^y$ with respect to the basis g, we define $\mathsf{dh}(X, Y) = g^{xy}$.

B. GROUP DESCRIPTION

We use $\mathcal{G}'$ to denote group generator, which acts as follows: $(\mathbb{G}_1', N, \mathbb{G}_1, g', q') \leftarrow \mathcal{G}'(1^\lambda)$, where $\mathbb{G}_1'$ is an abelian group of order N, $\mathbb{G}_1 =< g' >$ is a unique subgroup of $\mathbb{G}_1'$ with generator g' of prime order q'. $\mathcal{G}$ denotes a random group generator, which acts as below: takes as input a security parameter λ, and generates $(\mathbb{G}_2, g, n, p, q) \leftarrow \mathcal{G}(1^\lambda)$, where $\mathbb{G}_2$ is a cyclic group with order n, $n = pq$; p, q be two security primes of λ-bit, and g is a random generator of $\mathbb{G}_2$ of order n.

C. CRYPTOGRAPHIC ASSUMPTIONS

Definition 1 (Decisional Diffie-Hellman (DDH) Assumption [16]): Let $\mathbb{G}_2$ is a cyclic group of prime order p, the DDH problem is to distinguish a DDH tuple (g, g^a, g^b, g^{ab}) from a non-DDH tuple (g, g^a, g^b, g^z), where $g \leftarrow_\$ \mathbb{G}_2$, $a, b, c \leftarrow_\$ Z_p$. Formally, the advantage that a PPT adversary $\mathcal{A}$ distinguish the DDH problem is defined as: $\mathsf{Adv}_{\mathbb{G}_2,\mathcal{A}}^{\mathsf{DDH}}(\lambda) = \Pr[\mathcal{A}(g, g^a, g^b, g^{ab})] - \Pr[\mathcal{A}(g, g^a, g^b, g^z)]$. *The DDH assumption holds if the advantage of the adversary $\mathcal{A}$ is negligible in λ.*

Definition 2 (Strong RSA Assumption [42]): Let $n = pq$, where p and q are two λ-bit security prime numbers and $p = 2p'' + 1$, $q = 2q'' + 1$ for some primes p'', q''. Sample $g \leftarrow_\$ Z_n^*$. For any PPT adversary $\mathcal{A}$, we define its advantage $\mathsf{Adv}_{\mathcal{A},n}^{\mathsf{sRSA}}(\lambda) = \Pr[\mathcal{A}(n, g) = (z, e) : z^e = g \mod n]$. *The strong RSA holds for any PPT adversary $\mathcal{A}$, if its advantage is negligible in λ.*

Definition 3 (Gap Diffie-Hellman (GDH) Assumption [22]): Define $\mathbb{G}_1$ to be a multiplicative group with prime order p'. For any PPT adversary $\mathcal{A}$, we define the advantage function as $\mathsf{Adv}_{\mathcal{A},\mathbb{G}_1}^{\mathsf{GDH}}(\lambda) = \Pr[\mathcal{A}^{\mathcal{O}_{DDH}(\cdot,\cdot,\cdot)}(g', g'^a, g'^b) = g'^{ab}]$, *with the DDH oracle $\mathcal{O}_{DDH}(\cdot, \cdot, \cdot)$ is defined as: it takes as input input $(U, V, Z) \in \mathbb{G}_1^3$ and outputs 1 if and only if $Z = \mathsf{dh}(U, V)$. For instance, the oracle outputs 1 when $(U, V, Z) = (g'^a, g'^b, g'^{ab})$, and 0 when $(U, V, Z) \neq (g'^a, g'^b, g'^{ab})$. The GDH assumption holds for any above adversary $\mathcal{A}$ with access to a DDH oracle, if its advantage* $\mathsf{Adv}_{\mathcal{A},\mathbb{G}_1}^{\mathsf{GDH}}(\lambda)$ *is negligible in λ.*

D. PSEUDORANDOM FUNCTIONS

Define $F : \{0, 1\}^\lambda \times \mathcal{X} \rightarrow \mathcal{Y}$ to be a function with the domain $\{0, 1\}^\lambda \times \mathcal{X}$ and range $\mathcal{Y}$. F is a pseudorandom function (PRF) if for any probabilistic polynomial-time (PPT) adversaries $\mathcal{A}$, the advantage specified as $\mathsf{Adv}_{F,\mathcal{A}}^{\mathsf{prf}}(\lambda) = \Pr[\mathcal{A}^{F(K,\cdot)}(1^\lambda)] - \Pr[\mathcal{A}^{f(\cdot)}(1^\lambda)]$ is negligible in λ, where$f : \mathcal{X} \rightarrow \mathcal{Y}$ and $K \leftarrow_\$ \{0, 1\}^\lambda$.

E. PARTICULAR MAP FROM KEYWORDS TO Primes[26]

Here we review a deterministic, memory-efficient mapping family $\mathbb{H}$ from a keyword set W to a 2λ-bit prime integer set $P_{2\lambda}$. In [34], the keyword-to-prime mapping $\mathbb{H}$ is designed based on a collision-resistance hash (CRH) function $\bar{\mathbb{H}}$ and a pseudorandom function $\mathcal{F}_\lambda$. Here, to save space, we omit the detailed implementation of $\mathbb{H}$ (refer to [34]) and directly apply it into our scheme. In particular, for convenience, we still use the keyword presentation instead of its prime presentation in our scheme, but by default, we assume that the keywords used in our scheme have been mapped to the corresponding primes.

F. AUTHENTICATED ENCRYPTION WITH ASSOCIATED DATA (AEAD)[29]

We review the AEAD scheme and its security definition from [37]. We hereby stress that all properties of the scheme follow that of the scheme in [37].

Roughly speaking, an AEAD scheme provides both the authentication and confidentiality in a scheme by encrypting a message M and a (public) header information H (or called associated data). Note that the H may vary as the context changes in which it works. More specifically, an AEAD scheme contains the following three PPT algorithms $\mathsf{AEAD} = (\mathsf{AEAD.Gen}, \mathsf{AEAD.Enc}, \mathsf{AEAD.Dec})$.

- Key Generation algorithm $\mathsf{AEAD.Gen}(1^\lambda)$: This algorithm (probabilistic) takes as input a security parameter 1^λ and outputs an AEAD symmetric key K_{AEAD}, where $K_{\mathsf{AEAD}} \leftarrow_\$ \mathcal{K}_{\mathsf{AEAD}}$ and $\mathcal{K}_{\mathsf{AEAD}}$ is the AEAD key space.
- Encryption algorithm $\mathsf{AEAD.Enc}(K_{\mathsf{AEAD}}, H, M)$: This algorithm (may be probabilistic) takes as input an AEAD key K_{AEAD}, an public header H, and a plaintext M.

It outputs a ciphertext c_{AE} or $\perp$. Note that for convenience, in the text, we sometimes denote the encryption algorithm as $\mathsf{AEAD.Enc}_{K_{\mathsf{AEAD}}}(H, M)$ with K_{AEAD} as the subscript (applied in Figure 1). The case is the same for the decryption algorithm as follows.

- Decryption algorithm $\mathsf{AEAD.Dec}(K_{\mathsf{AEAD}}, H, c_{AE})$: This algorithm (deterministic) takes as input an AEAD key K_{AEAD}, an public header H and a ciphertext c_{AE}. It finally outputs a message M or $\perp$.

Security. The security game for the scheme AEAD played between a PPT adversary $\mathcal{A}$ and a challenger $\mathcal{C}$ is shown in Table 1. The advantage function of $\mathcal{A}$ is defined as $\mathsf{Adv}^{ae\text{-}sec}_{\mathsf{AEAD}} = |2.\Pr[\mathsf{Exp}^{ae\text{-}sec}_{\mathsf{AEAD},\mathcal{A}} = 1]-1|$. The AEAD scheme is ae-*secure*, if the adversary $\mathcal{A}$'s advantage is negligible in λ.

G. ATTRIBUTE-BASED ENCRYPTION (ABE) [7]

An ABE scheme [7] is typically classified into two categories: the CP-ABE and KP-ABE. The CP-ABE means that the ciphertext is related to an access control policy determined by an attribute set and each decryption key is related to the attribute set; while the KP-ABE is defined oppositely. We hereby only review the CP-ABE definition. Concretely, a CP-ABE contains the following four PPT algorithms:

$$\mathsf{ABE} = (\mathsf{ABE.Setup}, \mathsf{ABE.KeyGen}, \mathsf{ABE.Enc}, \mathsf{ABE.Dec})$$

which are defined as follows.

- Setup algorithm $\mathsf{ABE.Setup}(1^\lambda)$: This algorithm takes as input 1^λ and outputs a key pair (mpk, msk) which denotes the master public key and master secret key.
- Key generation algorithm $\mathsf{ABE.KeyGen}(msk, S)$: This algorithm takes as input the master secret key msk and an attribute set S. It outputs a private key sk_S associated with the attribute set S.
- Encryption algorithm $\mathsf{ABE.Enc}(mpk, M, \mathbb{A})$: This algorithm takes as input the master public key mpk, a message M and an access control policy $\mathbb{A}$. It outputs a ciphertext e.
- Decryption algorithm $\mathsf{ABE.Dec}(sk_S, e)$: This algorithm takes as input a private key sk_S and a ciphertext c, it outputs a message M or $\perp$.

III. DEFINITION OF IDENTITY-CONCEALED MULTI-DATA-OWNER SSE WITH AUTHENTICATION

In the following, we give the definition of identity-concealed SSE with authentication in the multi-data owner settings (icSSE for short). Our icSSE scheme Π contains eight PPT algorithms $\Pi = (\mathsf{Setup}, \mathsf{LKeyGen}, \mathsf{DWKeyGen}, \mathsf{EDBGen}, \mathsf{SKeyGen}, \mathsf{ETokenGen}, \mathsf{Search}, \mathsf{Retrieve})$.

- $\mathsf{Setup}(1^\lambda)$: This algorithm is the responsibility of the data owner. It takes as input 1^λ, and outputs a long-term key generation public parameter par_L and an encrypted database identifier generation public parameter par_E.
- $\mathsf{LKeyGen}(par_L, id_U)$: This algorithm is the responsibility of a long-term key generation center LK-KGC. It takes as input a long-term key generation public parameter par_L and a user identity $id_U \in \{0, 1\}^*$. It outputs a key pair (pk_U, sk_U) which denotes the user U's long-term public key and long private key, respectively.
- $\mathsf{DWKeyGen}(1^\lambda, id_{DW})$: This algorithm is the responsibility of a data owner key generation center DWK-KGC. It takes as input a security parameter 1^λ and a data owner identity id_{DW}. It outputs an encrypted database public/secret key pair $(\text{EPK}_{DW}, \text{ESK}_{DW})$ for data owner id_{DW}.
- $\mathsf{EDBGen}(par_E, \text{EPK}_{DW}, \text{ESK}_{DW}, \mathsf{DB}, \mathbb{A})$: This algorithm is the responsibility of a data owner id_{DW}. It takes as input an encrypted database identifier generation public parameter par_E, a data owner's encrypted database public key EPK_{DW} and encrypted database secret key ESK_{DW}, a plaintext database DB, an access control structure $\mathbb{A}$. The algorithm encrypts the database DB to a ciphertext database EDB_{DW} and sends EDB_{DW} to the server.
- $\mathsf{SKeyGen}(\text{EPK}_{DW}, \text{ESK}_{DW}, S, \mathsf{w}, id_U)$: This algorithm is the responsibility of a data owner DW. It takes as input a data owner's encrypted database public key EPK_{DW} and an encrypted database secret key ESK_{DW}, an attribute set S, an authorized keyword set w and a data user identity $id_U \in \{0, 1\}^*$. It produces a search private key $SK_U = (SK_{MS,U}, SK_{S,U})$ for the data user id_U, where SK_U consists of a master search private key $SK_{MS,U}$ and a partial search private key $SK_{S,U}$.
- $\mathsf{ETokenGen}(SK_U, sk_U, pid_U, pid_V, \mathsf{Q}, id_{\mathsf{EDB}_{DW}})$: This algorithm is the responsibility of a data user id_U. It takes as input the data user's search private key SK_U, long-term private key sk_U and public identity information $pid_U = (id_U, pk_U = U = g^u, cert_U)$, and the server public identity information $pid_V = (id_V, pk_V = V = g^v, cert_V)$, a query Q and the identifier $id_{\mathsf{EDB}_{DW}}$ of an encrypted database EDB_{DW}. It outputs an encrypted search token est. Note that the token est does not contain the public identity information pid_U and pid_V, where pid_V denotes the public identity information of the server.
- $\mathsf{Search}(sk_V, pid_V, pid_U, est, \mathsf{EDB})$: This algorithm is the responsibility of the server. It takes as input the server's long-term private key sk_V and public identity information pid_V, a data user id_U's public identity information pid_U, an encrypted search token est and the current full encrypted database EDB. Note that the encrypted database EDB contains all the partial encrypted database EDB_{DW} generated by a data owner DW. It finally generates the matching results and sends R to the data user id_U.
- $\mathsf{Retrieve}(SK_U, \mathsf{R})$: This algorithm is the responsibility of the data user id_U. It takes as input a search private key SK_U and a search result set R. It produces the document indexes matching the search keywords w given by the data user id_U.

TABLE 1. AEAD security game.

main $\mathsf{Exp}^{\text{ae-}sec}_{\mathsf{AEAD},\mathcal{A}}$:	$\mathsf{Enc}(H, M_0, M_1)$:	$\mathsf{Dec}(c')$:
$K_{\mathsf{AEAD}} \leftarrow_{\$} \mathsf{AEAD.Gen}(1^\lambda)$ $b \leftarrow_{\$} \{0,1\}$ $b' \leftarrow \mathcal{A}^{\mathsf{Enc},\mathsf{Dec}}$ Return $(b' = b)$	If $\lvert M_0\rvert \neq \lvert M_1\rvert$, Return $\perp$ $c_0 \leftarrow \mathsf{AEAD.Enc}_{K_{\mathsf{AEAD}}}(H, M_0)$ $c_1 \leftarrow \mathsf{AEAD.Enc}_{K_{\mathsf{AEAD}}}(H, M_1)$ If $c_0 = \perp$ or $c_1 = \perp$ then return $\perp$ Set $C \leftarrow C \cup c_b$ Return c_b	If $b = 1 \wedge c' \notin C$ then Return $\mathsf{AEAD.Dec}_{K_{\mathsf{AEAD}}}(c')$ Return $\perp$

Compared with the primitive by Sun *et al.* [34] (short for Sun), our prototype can be used in both multi-data-user and multi-data-owner environments. The multiple-data-owner means that multiple data owners are allowed exist simultaneously in an SSE system. In addition, this primitive can also prevent data users from performing any search query on an unauthorized keywords. In particular, it also captures the authentication of a data user to the server aiming to prevent the adversary from launching a DoS attacks (i.e., denial of service attacks). Besides, both the identity-concealment and confidentiality of search token are also implemented in our primitive aiming to resist impersonation and token replay attacks respectively.

A. SECURITY DEFINITION

We give the security definition and security analysis of icSSE primitive by modifying the security definition in [23]. In particular, comparing with the security model in previous work [23], we add authentication and identity-concealment security requirements which further strengthens the security of the proposed scheme. More specifically, for the proof of the L-simulation security, we adopt a method similar [23] to simulate the transcript of the search process. While for the proof of authentication and identity-concealment security, we directly coprrespond them to the outsider unforgeability and insider confidentiality of Zhao's higncryption scheme in [37], respectively.

1) SIMULATION MODEL

Let Π = (Setup, $\mathsf{LKeyGen}$, $\mathsf{DWKeyGen}$, EDBGen, $\mathsf{SKeyGen}$, $\mathsf{ETokenGen}$, Search, $\mathsf{Retrieve}$) be an identity-concealed symmetric SSE scheme with authentication in the multi-data-owner settings, EDB be an encrypted database and Sim be a simulator. Let EPK_{DW} and ESK_{DW} be the encrypted database public/secret key of a data owner id_{DW}, SK_U be the search private key for a data user id_U. The simulation security with respect to the server id_V is defined via the following two probabilistic games played among a PPT adversary $\mathcal{A}$, a simulator Sim and a challenger $\mathcal{C}$:

- Real game: In this game, $\mathcal{A}$ first selects a database DB, then $\mathcal{C}$ performs $\mathsf{Setup}(1^\lambda)$, $\mathsf{LKeyGen}(par_L, id_U)$, $\mathsf{DWKeyGen}(1^\lambda, id_{DW})$ and $\mathsf{EDBGen}(par_E, \mathrm{EPK}_{DW}, \mathrm{ESK}_{DW}, \mathsf{DB}, \mathbb{A})$ to generate a ciphertext database EDB_{DW} and sends it to $\mathcal{A}$. $\mathcal{A}$ chooses an authorized keyword set w for a data user id_U and makes multiple queries q, where the keywords with respect to q are assumed always falls over the authorized keyword set w. Next, the challenger $\mathcal{C}$ continues to run the rest algorithms (i.e., $\mathsf{SKeyGen}$, $\mathsf{ETokenGen}$, Search and $\mathsf{Retrieve}$) to obtain the transcript. It then sends the transcript and data user output to $\mathcal{A}$. We stress that in this game $\mathcal{A}$ may get partial search private key $SK_{S,U}$ generated by the algorithm $\mathsf{SKeyGen}$. Finally, the challenger outputs that $\mathcal{A}$ outputs. For simplicity, the advantage function of $\mathcal{A}$ in the real game is defined as $\mathsf{Real}^{\Pi}_{\mathcal{A}}(\lambda)$.
- Ideal game: This game first sets an empty list q and a counter $i = 0$. $\mathcal{A}$ picks a database DB_{DW} associated with a data owner DW, and the game runs $Sim(\mathcal{L}(\mathsf{DB}))$ and returns the encrypted database EDB_{DW}. Next the game records the i-th query as $\mathsf{q}[i]$, and runs $Sim(\mathcal{L}(\mathsf{DB}, \mathsf{q}))$ and outputs the generated transcript to $\mathcal{A}$. Note that the simulation here includes the simulation of the partial search private key $SK_{S,U}$ for some data user id_U. $\mathcal{A}$ returns a bit $b \in \{0, 1\}$ at the end of the game. For simplicity, the advantage of $\mathcal{A}$ in the ideal game is denoted as $\mathsf{Ideal}^{\Pi}_{\mathcal{A},Sim}(\lambda)$.

Definition 4: The scheme Π is said to be $\mathcal{L}$-simulation secure (SS) if for any PPT adversary $\mathcal{A}$ there exists a PPT simulator Sim such that the following advantage is negligible in λ, namely $\mathsf{Adv}^{SS}_{\Pi,\mathcal{A},Sim}(\lambda) = \Pr[\mathsf{Real}^{\Pi}_{\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{Ideal}^{\Pi}_{\mathcal{A},Sim}(\lambda) = 1] \leq negl(\lambda)$

2) LEAKAGE FUNCTION

The SS security of the identity-concealed SSE with authentication in the multi-data-owner settings against adversarial server is to efficiently perform search on the encrypted database while revealing little information about private data. Following with [23] and [34], the SS security of the icSSE primitive and the corresponding leakage function is defined as follows.

Let $\mathsf{q} = (\mathsf{s}, \mathsf{x})$ denote a query list, where s denotes the *sterm* component of q and x denotes the *xterm* component of q. $\mathsf{q}[i] = (\mathsf{s}[i], \mathsf{x}[i])$ denotes the i-th query. Taking as input DB and q, the leaking information of the leakage function $\mathcal{L}$ are listed as follows:

- $K = \cup_{i=1}^{T} \mathrm{W}_i$ means the total number of keywords in the database DB.
- $N = \sum_{i=1}^{T} \lvert\mathrm{W}_i\rvert$ denotes the total number of keyword/document identifier pairs in DB.
- $\bar{s} \in \mathbb{N}^{\lvert q\rvert}$ denotes the equality pattern of s, where each distinct *sterm* in s is assigned to an integer on the basis of

its order that lies in s. For instance, if $\mathsf{s} = (a, b, c, b, a)$, then $\bar{\mathsf{s}} = (1, 2, 3, 2, 1)$.

- SP denotes the size pattern of the queries, i.e., $\text{SP}[i] = |\mathsf{DB}[\mathsf{s}[i]]|$. In addition, we define $\text{SRP}[i] = \mathsf{DB}[\mathsf{s}[i]]$ to be the search result w.r.t. the *sterm* of the i-th query.
- RP is the result pattern, namely, the common items of *sterm* and *xterm* in the same query.
- IP is the conditional intersection pattern such that

$$\text{IP}[i, j, \alpha, \beta] = \begin{cases} \mathsf{DB}[\mathsf{s}[i]] \cap \mathsf{DB}[\mathsf{s}[j]] & \mathsf{s}[i] \neq \mathsf{s}[j] \textit{ and } \\ & \mathsf{x}[i, \alpha] = \mathsf{x}[j, \beta] \\ \emptyset & \textit{otherwise} \end{cases}$$

3) AUTHENTICATION

In fact, the authentication of an icSSE scheme corresponds to the outer unforgeability of Zhao's higncryption scheme. Informally, the goal of an authentication adversary $\mathcal{A}_{aut}$ is to fabricate a valid ciphertext generated by an uncorrupted honest data user id_{U^*} for the uncorrupted server id_{V^*}. Towards this goal, the adversary $\mathcal{A}_{aut}$ may make some queries to oracles HO, UHO, EXO and Corrupt. Finally, $\mathcal{A}_{aut}$ returns (pid_{V^*}, C^*) as its forgery, where pid_{V^*} is required honest, while H^* is contained in clear text in C^*. The advantage of $\mathcal{A}_{aut}$ in the authentication game is denoted by $\mathsf{Adv}^{\mathsf{AUT}}_{\mathcal{A}_{aut},\Pi}$ if the following conditions hold.

- $\mathsf{Unhigncrypt}(sk_{V^*}, pid_{V^*}, C^*) = (pid_{U^*}, M^*)$ where pid_{U^*} is honest.
- $\mathcal{A}_{aut}$ has not queried $\mathsf{Corrupt}(pid_{U^*})$ or $\mathsf{Corrupt}(pid_{V^*})$. But $\mathcal{A}_{aut}$ may query $\mathsf{EXO}(C^*)$ to get the randomness used in creating C^*.
- C^* was not allowed to be the value that $\mathcal{A}_{aut}$ got by querying the oracle $\mathsf{HO}(pid_{U^*}, pid_{V^*}, H^*, M^*)$.

Definition 5: An identity-concealed multi-data-owner symmetric searchable encryption with authentication (icSSE) meets authentication if for all PPT adversary $\mathcal{A}_{aut}$, the function value $\mathsf{Adv}^{\mathsf{AUT}}_{\Pi,\mathcal{A}_{aut}}(\lambda)$ is negligible in λ.

The purpose of the identity-concealment is to protect the privacy of data user identity. The confidentiality of search token aims to prevent replay search attacks. These properties exactly correspond to the insider confidentiality of Zhao's higncryption scheme. In other words, the goal of the adversary $\mathcal{A}_{con}$ in the identity-concealment and confidentiality is to break the private identity information of the challenge data user and the confidentiality of the challenge search token encrypted to an uncorrupted honest server, though the sender is allowed to expose the intermediate randomness used for creating the ciphertexts. Formally, the adversary breaks the identity-concealment and confidentiality by the following experiment.

- Query 1: The adversary $\mathcal{A}_{con}$ may make queries to the oracles HO, UHO, EXO and Corrupt.
- Challenge: The adversary $\mathcal{A}_{con}$ submits a search token pair (st_0, st_1), a challenge public header H^*, and two identity information pairs $(pid_{U_0^*}, pid_{V^*})$ and $(pid_{U_1^*}, pid_{V^*})$ where $pid_{U_0^*}, pid_{U_1^*}$ and pid_{V^*} are honest. It then submits them to the challenger. The challenger $\mathcal{C}$ picks a bit $b \leftarrow_{\$} \{0, 1\}$, and generates a cipheretext $C^* = Higncrypt(sk_{U_b^*}, pid_{U_b^*}, pid_{V^*}, H^*, st_b)$.
- Query 2: $\mathcal{A}_{con}$ continues the same query as in Query 1, but with the exception that it is not allowed to query $\mathsf{UHO}(pid_{V^*}, C^*)$ or $\mathsf{EXO}(C^*)$ or $\mathsf{Corrupt}(pid_{V^*})$, since these queries will cause the adversary $\mathcal{A}_{con}$ to trivially win the experiment. In addition, this security allows $\mathcal{A}_{con}$ to make $\mathsf{Corrupt}(pid_{U_0^*})$ and $\mathsf{Corrupt}(pid_{U_1^*})$ queries.
- Guess: Finally, $\mathcal{A}_{con}$ returns a guess b' of the challenge bit b. The experiment outputs 1, if $b' = b$; outputs 0 if $b' \neq b$.

We define the advantage of a PPT adversary $\mathcal{A}_{con}$ as $\mathsf{Adv}^{\mathsf{Ic\text{-}Con}}_{\Pi,\mathcal{A}_{con}}(\lambda) = |\Pr[b' = b] - 1/2|$.

Definition 6: An icSSE scheme has identity-concealment and confidentiality security, if for any PPT adversary $\mathcal{A}_{con}$, the adversary's advantage $\mathsf{Adv}^{\mathsf{Ic\text{-}Con}}_{\Pi,\mathcal{A}_{con}}(\lambda)$ defined above is negligible in λ.

IV. CONSTRUCTION

We give a concrete construction of identity-concealed SSE with authentication in the multi-data-owner settings. First we list some basic cipher suites used in constructing this scheme.

- A simulated CPA secure ABE scheme $\mathsf{ABE} = (\mathsf{ABE.Setup}, \mathsf{ABE.KeyGen}, \mathsf{ABE.Enc}, \mathsf{ABE.Dec})$.
- $h : \{0, 1\}^* \rightarrow \{0, 1\}^l \cap Z^*_{q'}$ with $l = \lceil |q'|/2 \rceil$.
- $\mathsf{KDF} : \mathbb{G}_1 \times \{0, 1\}^* \rightarrow \{0, 1\}^*$ is a key derivation function.
- $\mathsf{AEAD} = (\mathsf{AEAD.Gen}, \mathsf{AEAD.Enc}, \mathsf{AEAD.Dec})$ is a secure authentication encryption with associated data. We assume $\mathcal{K}_{\mathsf{AEAD}}$ is the key space of the AEAD scheme where the algorithm AEAD.Gen generates the secret key K_{AEAD} by sampling $K_{\mathsf{AEAD}} \leftarrow_{\$} \mathcal{K}_{\mathsf{AEAD}}$.
- F and F_p are pseudorandom functions.

• $\mathsf{Setup}(1^\lambda)$: On input 1^λ, this algorithm generates a long-term key generation public parameter $par_L = (\mathbb{G}'_1, N, \mathbb{G}_1, g', q') \leftarrow \mathcal{G}'(1^\lambda)$ used in producing long-term public/private key which specifies the underlying group over which GDH assumption holds (as defined in Section II-B and Definition 3), and an encrypted database identifier generation public parameter $par_E = \mathsf{H}$, where H is a compression function. The algorithm finally outputs par_L and par_E.

• $\mathsf{LKeyGen}(par_L, id_U)$: This algorithm is run by LK-KGC. It takes as input a long-term key generation public parameter par_L. For each honest user $id_U \in \{0, 1\}^*$, this algorithm samples $u \leftarrow_{\$} Z^*_{q'}$, sets $pk_U = U = g'^u$ and $sk_U = u$ and outputs the key pair (pk_U, sk_U). The realtion between a user identity id_U and its public-key U is certified by the $cert_A$ issued by CA. In addition, the CA also performs a checking $pk_U \in \mathbb{G}_1 \backslash 1_{\mathbb{G}_1}$.

• $\mathsf{DWKeyGen}(1^\lambda, id_{DW})$: This algorithm is run by DWK-KGC. It takes as input 1^λ and a data owner identity id_{DW}.

It generates $(\mathbb{G}_2, g, n, p, q) \leftarrow_\$ \mathcal{G}(1^\lambda)$, where $\mathcal{G}$ is a random group generator, $\mathbb{G}_2$ is a multiplicative cyclic group, and $g \leftarrow_\$ \mathbb{G}_2$ is a random generator of order n and $n = pq$, p and q are two large primes. Then it checks whether (p, q) has been in table Tab, if it does, DWK-KGC reruns $\mathcal{G}(1^\lambda)$ until (p, q) is not in Tab and stores (p, q) into Tab. Next, it selects $K_X, K_I, K_Z, K_E \leftarrow_\$ \mathcal{K}$, $g_1, g_2, g_3 \leftarrow_\$ \mathbb{G}_2$, and computes $(mpk, msk) \leftarrow_\$ \mathsf{ABE.Setup}(1^\lambda)$. The encrypted database secret key and public key for the data owner DW are set as $\mathrm{ESK}_{DW} \leftarrow (K_X, K_I, K_Z, K_E, p, q, g_1, g_2, g_3, msk)$ and $\mathrm{EPK}_{DW} = (n, mpk)$ respectively.

- $\mathsf{EDBGen}(par_E, \mathrm{EPK}_{DW}, \mathrm{ESK}_{DW}, \mathsf{DB}, \mathbb{A})$: This algorithm is run by a data owner id_{DW}. It inputs a database identifier generation parameter $par_E = \mathsf{H}$, an encrypted database public key $\mathrm{EPK}_{DW} = (n, mpk)$, secret key $\mathrm{ESK}_{DW} = (K_X, K_I, K_Z, K_E, p, q, g_1, g_2, g_3, msk)$, a database DB, and an access control structure $\mathbb{A}$. It returns an encrypted database $\mathsf{EDB}_{DW} = (\mathsf{TSet}_{DW}, \mathsf{XSet}_{DW}, id_{\mathsf{EDB}_{DW}}, \mathcal{L}_{DW})$, where $id_{\mathsf{EDB}_{DW}}$ denotes the identifier of the encrypted database EDB_{DW}. The detailed description is demonstrated in algorithm 1.

Algorithm 1 $\mathsf{EDBGen}(par_E, \mathrm{EPK}_{DW}, \mathrm{ESK}_{DW}, \mathsf{DB}, \mathbb{A})$

```
Input: $par_E \leftarrow \mathsf{H}$, $\mathrm{EPK}_{DW} \leftarrow (n, mpk)$,
    $\mathrm{ESK}_{DW} \leftarrow (K_X, K_I, K_Z, K_E, p, q, g_1, g_2, g_3, msk)$,
    $\mathsf{DB} = (\mathrm{ind}_i, \mathsf{W}_i)_{i=1}^{T}$, $\mathbb{A}$
Output: $\mathsf{EDB}_{DW}$
1:  $\mathsf{stag}, \mathsf{TSet}_{DW}, \mathsf{XSet}_{DW}, \mathsf{EDB}_{DW}, \mathcal{L}_{DW} \leftarrow \emptyset$,
2:  for $w \in \mathrm{W}$ do
3:    $c \leftarrow 1$
4:    $\mathrm{stag}_w \leftarrow F(K_E, g_1^{1/w} \bmod n)$
5:    $\mathsf{stag} \leftarrow \mathsf{stag} \cup \mathrm{stag}_w$
6:    for $\mathrm{ind} \in \mathsf{DB}[w]$ do
7:      $\mathrm{xind} \leftarrow F_p(K_I, \mathrm{ind})$;
8:      $z \leftarrow F_p(K_Z, g_2^{1/w} \bmod n||c)$;
9:      $l \leftarrow F(\mathrm{stag}_w, c)$; $\mathcal{L}_{DW} \leftarrow \mathcal{L}_{DW} \cup \{l\}$;
10:     $e \leftarrow \mathsf{ABE.Enc}(mpk, \mathrm{ind}, \mathbb{A})$; $y \leftarrow \mathrm{xind} \cdot z^{-1}$ //where
        $\mathbb{A}$ refers to the access control structure.
11:     $\mathsf{TSet}_{DW}[l] = (e, y)$
12:     $\mathrm{xtag} \leftarrow g^{F_p(K_X, g_3^{1/w} \bmod n) \cdot \mathrm{xind}}$;
13:     $\mathsf{XSet}_{DW} \leftarrow \mathsf{XSet} \cup \{\mathrm{xtag}\}$
14:     $c \leftarrow c + 1$
15:   endfor
16: endfor
17: $id_{\mathsf{EDB}_{DW}} \leftarrow \mathsf{H}(\mathsf{TSet}_{DW}, \mathsf{XSet}_{DW})$
18: $\mathsf{EDB}_{DW} \leftarrow (\mathsf{TSet}_{DW}, \mathsf{XSet}_{DW}, id_{\mathsf{EDB}_{DW}}, \mathcal{L}_{DW})$
19: return $\{\mathsf{EDB}_{DW}\}$
```

- $\mathsf{SKeyGen}(\mathrm{EPK}_{DW}, \mathrm{ESK}_{DW}, \mathsf{w}, id_{DW}, id_U, id_{\mathsf{EDB}_{DW}})$: This algorithm is performed by a data owner id_{DW}. It inputs an encrypted database public key $\mathrm{EPK}_{DW} = (n, mpk)$ and secret key $\mathrm{ESK}_{DW} = (K_X, K_I, K_Z, K_E, p, q, g_1, g_2, g_3, msk)$, a set of keywords w, a data owner identity id_{DW}, a data user identity id_U and an encrypted database identifier $id_{\mathsf{DB}_{DW}}$ (assume that once a data owner generates an encrypted database, it immediately records the corresponding identifier). Assuming that the data user id_U may run a search over the authorized keyword set $\mathsf{w} = \{w_1, \cdots, w_N\}$. The data owner id_{DW} first computes $sk_{\mathsf{w}}^{(i)} = (g_i^{1/\prod_{j=1}^{N} w_j} \bmod n)$ for $i = \{1, 2, 3\}$ and an attribute key $sk_S \leftarrow \mathsf{ABE.KeyGen}(msk, S)$, where $S \in \mathcal{U}$ is the attribute set of authorized data users and $\mathcal{U}$ is the attribute universe. Next, the data owner id_{DW} sends the search private key $\mathrm{SK}_U = (\mathrm{SK}_{MS,U}, \mathrm{SK}_{S,U})$ to the data user id_U, where $\mathrm{SK}_{MS,U} = (K_E, K_X, K_Z, sk_{\mathsf{w}}, id_{\mathsf{EDB}_{DW}})$ and $\mathrm{SK}_{S,U} = sk_S$ respectively denotes the master and partial search private key for the data user id_U, and $sk_{\mathsf{w}} = (sk_{\mathsf{w}}^{(1)}, sk_{\mathsf{w}}^{(2)}, sk_{\mathsf{w}}^{(3)})$. In particular, note that the identifier $id_{\mathsf{EDB}_{DW}}$ of the encrypted database EDB_{DW} is also included in the master search private key $\mathrm{SK}_{MS,U}$ so that the data user id_U easily locates the encrypted database with the help of $id_{\mathsf{EDB}_{DW}}$.
- $\mathsf{ETokenGen}(\mathrm{SK}_U, sk_U, pid_U, pid_V, \bar{\mathsf{w}}, id_{\mathsf{EDB}_{DW}})$: This algorithm inputs a search private key $\mathrm{SK}_U = (K_E, K_X, K_Z, sk_{\mathsf{w}}, id_{\mathsf{EDB}_{DW}}, sk_S)$, a long-term private key $sk_U = u$, public identity information $pid_U = (id_U, U, cert_U)$ and $pid_V = (id_V, V, cert_V)$, a set of authorized keywords $\bar{\mathsf{w}}$, and an encrypted database identifier $id_{\mathsf{EDB}_{DW}}$. When a data user id_U intends to carry out a query $\bar{\mathsf{w}} \subseteq \mathsf{w}$, he first determines the s-terms $\bar{\mathsf{s}} \subseteq \bar{\mathsf{w}}$. Assuming that $\bar{\mathsf{w}} = (w'_1, \cdots, w'_d)$ and w'_1 is the chosen s-term, then the detailed encrypted search token (i.e., $est = (H, \overline{X}, c_{AE})$) code for this query refers to Algorithm 2.
- $\mathsf{Search}(sk_V, pid_V, pid_U, est, \mathsf{EDB})$: This algorithm inputs the long-term private key sk_V, a public identity information pid_V of server id_V, the identity information pid_U of a data user U, an encrypted search token $est = (H, \overline{X}, c_{AE})$ and the current full encrypted database $\mathsf{EDB} = \{\mathsf{EDB}_{DW} = (\mathsf{TSet}_{DW}, \mathsf{XSet}_{DW}, id_{\mathsf{EDB}_{DW}}, \mathcal{L}_{DW})\}_{DW \in \mathcal{DW}}$, where $\mathcal{DW}$ denotes the set of data owners registered on the cloud server. With the long-term private key sk_V and the public identity information pid_V and pid_U, the algorithm first performs an authentication process, and then recovers the search token st and the database identifier $id_{\mathsf{EDB}_{DW}}$ from the encrypted search token est. Next, with the identifier $id_{\mathsf{EDB}_{DW}}$, the encrypted database EDB_{DW} is screened. Then, the server uses the search token $st = (\mathrm{stag}_{w'_1}, \mathsf{xtoken})$ to carry out a single keyword search, and gets an encrypted document index set R that match the search criteria. The detailed search procedure is described in Algorithm 3.
- $\mathsf{Retrieve}(\mathrm{SK}_U, \mathsf{R})$: This algorithm inputs a search private key SK_U and an encrypted document index set R. It first uses the partial private key sk_S (contained in the secret key SK_U) to decrypt each element in R to get the authorized part of the document indexes. Concretely, for each $e \in \mathsf{R}$, compute $\mathrm{ind} = \mathsf{ABE.Dec}(sk_S, e)$ if $S \in \mathcal{U}$ matches $\mathbb{A}$ associated with the ciphertext e that encrypts the index ind matching the query $\bar{\mathsf{w}}$.

Remark 1: Note that in our data model, we assume that each data owner id_{DW} has only one encrypted database

Algorithm 2 $\mathsf{ETokenGen}(\mathrm{SK}_U, sk_U, pid_U, pid_V, \overline{\mathsf{w}}, id_{\mathsf{EDB}_{DW}})$

```
Input: $\mathrm{SK}_U = (\mathrm{SK}_{MS,U}, \mathrm{SK}_{S,U})$, $sk_U = u$, $pid_U = (id_U, pk_U = U = g'^{u}, cert_U)$, $pid_V = (id_V, pk_V = V = g'^{v}, cert_V)$, $\overline{\mathsf{w}}$, $id_{\mathsf{EDB}_{DW}}$
Output: $est$
 1: $\mathsf{xtoken} \leftarrow \{\}, st, est, \bar{\mathsf{s}} \leftarrow \phi$, parse $\mathrm{SK}_U$ into $\mathrm{SK}_U = (K_E, K_X, K_Z, sk_{\mathsf{w}}, id_{\mathsf{EDB}_{DW}}, sk_S)$
 2: $\bar{\mathsf{s}} \leftarrow \bar{\mathsf{s}} \cup \{w'_1\}$
 3: $\mathsf{x} \leftarrow \bar{\mathsf{w}} \backslash \bar{\mathsf{s}}$
 4: $\mathsf{stag}_{w_1} \leftarrow F(K_E, (sk^{(1)}_{\overline{\mathsf{w}}})^{\prod_{w \in \overline{\mathsf{w}} \backslash w'_1} w} \bmod n)$
    $= F(K_E, g_1^{1/w'_1} \bmod n)$
 5: for $c = 1, 2, \cdots$ until the server stops do
 6:     for $i = 2, \cdots, d$ do
 7:     $\mathsf{xtoken}[c, i] \leftarrow g^{F_p(K_Z, (sk^{(2)}_{\overline{\mathsf{w}}})^{\prod_{w \in \overline{\mathsf{w}} \backslash w'_1} w} \bmod n || c) \cdot F_p(K_X, (sk^{(3)}_{\overline{\mathsf{w}}})^{\prod_{w \in \overline{\mathsf{w}} \backslash w'_i} w} \bmod n)} = g^{F_p(K_Z, g_2^{1/w'_1} \bmod n || c) \cdot F_p(K_X, g_3^{1/w'_i} \bmod n)}$
 8:     endfor
 9:     $\mathsf{xtoken}[c] = \mathsf{xtoken}[c, 2], \cdots, \mathsf{xtoken}[c, d]$
10: endfor
11: $st \leftarrow (\mathsf{stag}_{w_1}, \mathsf{xtoken})$
    Authenticationphase :
12: $\boxed{cert_U, u \leftarrow_{\$} Z^*_{q'}, U = g'^{u}}$  // Assume that these values have been precomputed.
13: $x \leftarrow Z^*_{q'}, X = g'^{x}$,
    $d = h(X, cert_U, cert_V, st, id_{\mathsf{EDB}_{DW}})$  // where $u$ is the data user $id_U$'s long-term private key and $U$ corresponds to the long-term public key; $cert_U$ and $cert_V$ are respectively the data user $id_U$'s and the server $id_V$'s certificates generated by the CA by signing on the public keys $U$ and $V$ with CA's signing key.
14: $\overline{X} = UX^d, PS = V^{u+xd}$
15: $K_{\mathsf{AEAD}} = \mathsf{KDF}(PS, \overline{X} || cert_V)$
16: $M \leftarrow st || id_{\mathsf{EDB}_{DW}}$  //where $id_{\mathsf{EDB}_{DW}}$ denotes the the database identifier associated with the query $\overline{\mathsf{w}}$.
17: $c_{AE} \leftarrow \mathsf{AEAD.Enc}_{K_{\mathsf{AEAD}}}(H, cert_U || X || M, ts)$
    //where $H$ denotes the associated data of the encryption algorithm for AEAD and $ts$ denotes the current time.
18: $est \leftarrow (H, \overline{X}, c_{AE})$
19: return $est$
```

Algorithm 3 $\mathsf{Search}(sk_V, pid_V, pid_U, est, \mathsf{EDB})$

```
Input: $sk_V = v$, $pid_V = (id_V, pk_V = V = g'^{v})$,
    $pid_U = (id_U, pk_U = U = g'^{u})$,
    $est = (H, \overline{X}, c_{AE})$, $\mathsf{EDB} =$
    $\{\mathsf{EDB}_{DW} = (\mathsf{TSet}_{DW}, \mathsf{XSet}_{DW}, id_{\mathsf{EDB}_{DW}})\}_{DW \in \mathcal{DW}}$
Output: $\mathsf{R}$
 1: $\mathsf{R} \leftarrow \phi$
    Authenticationphase :
 2: $\boxed{cert_V, v \leftarrow Z^*_{q'}, B = g'^{v}}$  // where $v$ is the server's private key, $V$ corresponds to its public key, and $cert_V$ is its certificate generated by the CA (certificate authority) by signing on the public key $V$ with CA's signing key. Also assume that these values have been recomputed beforehand.
 3: $PS = \overline{X}^{v}$
 4: $K_{\mathsf{AEAD}} \leftarrow \mathsf{KDF}(PS, \overline{X} || cert_V)$
 5: $(cert_U || X || M, ts) \leftarrow \mathsf{AEAD.Dec}_{K_{\mathsf{AEAD}}}(H, c_{AE})$
 6: if $ts' - ts > \delta$ reject and abort  // $ts'$ is the current time and $\delta$ is a constant
 7: $(st || id_{\mathsf{EDB}_{DW}}) \leftarrow M$
 8: $d = h(X, cert_U, cert_V, st, id_{\mathsf{EDB}_{DW}})$
 9: if $cert_U$ is valid and $\overline{X} = UX^d$ then accept
10: Otherwise, reject and abort
    SearchPhase :
11: use $id_{\mathsf{EDB}_{DW}}$ to search the target encrypted database $\mathsf{EDB}_{DW}$ from the full encrypted database $\mathsf{EDB}$.
12: Parse $\mathsf{EDB}_{DW} = (\mathsf{TSet}_{DW}, \mathsf{XSet}_{DW}, id_{\mathsf{EDB}_{DW}}, \mathcal{L}_{DW})$
13: $(\mathsf{stag}_{w_1}, \mathsf{xtoken}[1], \mathsf{xtoken}[2], \cdots) \leftarrow st$
14: $c \leftarrow 1; l \leftarrow F(\mathsf{stag}_{w_1}, c)$
15:   while $l \in \mathcal{L}_{DW}$ do
16:     $(e, y) \leftarrow \mathsf{TSet}_{DW}[l]$
17:     if $\mathsf{xtoken}[c, i]^{y} \in \mathsf{XSet}_{DW}$ for all $i$ then
18:       $\mathsf{R} \leftarrow \mathsf{R} \cup \{e\}$
19:     endif
20:   endwhile
21: return $\mathsf{R}$
```

EDB_{DW}*, and the encrypted database* EDB_{DW} *for the data owner is indexed by a unique and public identifier* $id_{\mathsf{EDB}_{DW}}$ *which makes our solution easily applied to a multi-database environment or a multi-data-owner scenario.*

V. SECURITY PROOF

We first prove the security against attacks with respect to server, which aims to break the simulation security of the SSE scheme, and then give the security against attacks associated with an adversarial data user where it aims to compute a valid private key. Finally, we give the authentication, identity-concealment, and confidentiality of the scheme against the impersonation attacks of data user to server, the privacy of data user identity information, and the replay search attacks for previously generated search token.

Theorem 1: Assuming that the DDH problem holds in $\mathbb{G}_2$. Let F and F_p be secure, and ABE *is CPA secure, then Π is $\mathcal{L}$-semantically secure, where $\mathcal{L}$ follows the definition in Section III-A2.*

Proof: The theorem is proved via a series of games from G_0 to G_{11} and three simulated games: simulated TSet, XSet and transcript t. We stress that in the first 12 games, i.e., G_0-G_{11}, the adversary provides an unencrypted database DB and a search query set q at the beginning of the game. Note that G_0 is equal to the real game (assuming no false positives), except a slightly modification but with the exactly identical distribution as in the real settings. G_{11} have the same distribution as the simulation game so that it can be easily modeled by a simulator *Sim*. We show that the simulator is designed correctly and satisfies the theorem by proving the

TABLE 2. Changes in each game.

Games	Changes	States in last game
Real Game	see construction IV	
G_0	XSet is computed by running a single function XSetSetup on inputs $(n, K_X, K_I, \mathsf{DB})$	Real game: see the construction for XSet in Algorithm 1
G_1	$\text{stag}_w \leftarrow \text{stag}$, $\text{query\_stag} \leftarrow \text{stags}[\mathsf{s}_t]$	G_0: $\text{stag} \leftarrow F(K_E, g_1^{1/w} \mod n)$; $\text{query\_stag} \leftarrow$ $F(K_E, (sk_{\mathsf{w}}^{(1)})^{\prod_{w \in \mathsf{w} \setminus \{s_t\}} w} \mod n)$
G_2	$f_E, f_X, f_I, f_Z \leftarrow_\$ \text{Fun}(\{0,1\}^\lambda, Z_p^*)$, $\text{stag} \leftarrow \{0,1\}^\lambda$	G_1: $K_E, K_X, K_I, K_Z \leftarrow_\$ \{0,1\}^\lambda$
G_3	$e \leftarrow \mathsf{ABE.Enc}(mpk, 0^\lambda, \mathbb{A})$	G_2: $e \leftarrow \mathsf{ABE.Enc}(mpk, \text{ind}_{\sigma[c]}, \mathbb{A})$
G_4	$\mathcal{H}[\text{ind}_i, w] \leftarrow X[w]^{\text{xind}}$, $Y[w,u,c] \leftarrow X[u]^{f_Z(g_2^{1/w} \mod n\|\|c)}$	
G_5	$y \leftarrow_\$ Z_p^*$	G_4: $y \leftarrow \text{xind} \cdot z^{-1}$
G_6	$\mathcal{H}[\text{ind}_i, w] \overset{\$}{\leftarrow} \mathbb{G}_2$, $Y[w,u,c] \overset{\$}{\leftarrow} \mathbb{G}_2$	G_5: $\mathcal{H}[\text{ind}_i, w] \leftarrow X[w]^{\text{xind}}$, $Y[w,u,c] \leftarrow X[u]^{f_Z(g_2^{1/w} \mod n\|\|c)}$
G_7	if $\exists t,\ \alpha$ s.t. $\text{ind} \in \mathsf{DB}_{DW}[\mathsf{s}[t]] \wedge \mathsf{x}[t,\alpha] = w$ then $\mathsf{XSet}_{DW} \leftarrow \mathsf{XSet}_{DW} \cup \{\mathcal{H}[\text{ind}, w]\}$ else $h \overset{\$}{\leftarrow} \mathbb{G}_2$; $\mathsf{XSet} \leftarrow \mathsf{XSet} \cup \{h\}$	G_6: $\text{xind} \leftarrow f_I(\text{ind})$; $\text{xtag} \leftarrow g^{f_X(g_3^{1/w} \mod n)\cdot\text{xind}}$; $\mathsf{XSet}_{DW} \leftarrow \mathsf{XSet}_{DW} \cup \{\mathcal{H}[\text{ind}, w]\}$
G_8	if $\exists t' \neq t,$ s.t. $\text{ind}_{\sigma[c]} \in \mathsf{DB}_{DW}[\mathsf{s}_{t'}] \wedge \mathsf{x}_t[\alpha] \in \mathsf{x}_{t'}$ then $\text{xtoken}[\alpha, c] \leftarrow \mathcal{H}[\text{id}_{\sigma[c]}, \mathsf{x}_t[\alpha]]^{1/y}$ else $\text{xtoken}[\alpha, c] \overset{\$}{\leftarrow} \mathbb{G}_2$	G_7: $\text{xtoken}[\alpha, c] \leftarrow \mathcal{H}[\text{id}_{\sigma[c]}, \mathsf{x}_t[\alpha]]^{1/y}$
G_9	$\text{Res} \leftarrow \text{SearchRes}(\mathsf{EDB}, \text{query\_stag}, \text{xtoken}, id_{\mathsf{EDB}_{DW}})$	G_8: $\text{Res} \leftarrow$ $\text{Search}(sk_V, pid_V, pid_U, est_t, \mathsf{EDB})$
G_{10}	$K_{\mathsf{AEAD}} \leftarrow_\$ \mathcal{K}_{\mathsf{AEAD}}$	G_9: $K_{\mathsf{AEAD}} = \mathsf{KDF}(PS, \overline{X}\|\|cert_V)$
G_{11}	$c_{AE} \leftarrow \mathsf{AEAD.Enc}_{K_{\mathsf{AEAD}}}(H, 0^{l_{\mathsf{AEAD}}}, ts)$	G_{10}: $c_{AE} \leftarrow$ $\mathsf{AEAD.Enc}_{K_{\mathsf{AEAD}}}(H, cert_U\|\|X\|\|M, ts)$

distribution indistinguishability of any two adjacent games. For clarity, the key changes between any two adjacent games are described in TABLE 2. The detailed proof is as below.

G_0 : This game is almost the same as the real game but with the only difference is that we make some minor changes to make the analysis easier to deal with, the detailed description is demonstrated in Algorithm 4. With $(1^\lambda, par_L, par_E, \mathsf{EDB}, \mathsf{DB}_{DW}, \mathsf{w}, \mathsf{s}, \mathsf{x}, id_U, id_{DW})$ as input, we compute the encrypted database components TSet and XSet by performing Initialize algorithm, which shares the EDBGen code in Algorithms 1 with the only difference is that we compute XSet as a single function XSetSetup to demonstrate changes between adjacent games. Then generate the transcript t by running the Initialize function and transGen function. Concretely, the Initialize function first employs par_L to compute two long-term public/private key pairs (pk_U, sk_U) and (pk_V, sk_V) for data user id_U and server id_V which are used for implementing authentication from id_U to id_V. To construct a transcript array t, for $\mathsf{t} \in [T]$ (where T is the total number of queries), it sets $\mathsf{t}[t]$ to be the returned value by the function transGen on inputs $(\mathsf{DB}_{DW}, \mathsf{EDB}_{DW}, \mathrm{SK}_U, sk_U, sk_V, pid_U, pid_V, id_{\mathsf{EDB}_{DW}}, \mathsf{s}[t], \mathsf{x}[t,\cdot], query\_stag)$, which generates a transcript that is identical to the real settings but with the difference is that it calculates ResInds in another way: it directly locates the answers for the query from the plaintext database rather than decrypting the ciphertexts Res returned by the server, i.e., it screens out the document identifiers in both sets $\mathsf{DB}_{DW}[\mathsf{s}_t]$ and $\mathsf{DB}_{DW}[\mathsf{x}_t]$, where $\mathsf{s}_t = \mathsf{s}[t]$, $\mathsf{x}_t = \mathsf{x}[t,\cdot]$ and $\mathsf{DB}_{DW}[\mathsf{x}_t] = (\mathsf{DB}_{DW}[\mathsf{x}[t,2]], \mathsf{DB}_{DW}[\mathsf{x}[t,3]], \cdots)$ may be alternatively defined for $t \in [T]$.

Beyond that, we also made other changes: the order of the document indexes for each keyword w are saved in a list WPerms$[w]$. For the sake of keeping consistent with the real game, the order is chosen at random and uniformly from a random permutation family.

Conditioned on no false positives, this game is equally distributed with the real experiment $\mathsf{Real}_{\mathcal{A}}^{\Pi}(\lambda)$. So the following holds

$$\Pr[\mathsf{G}_0 = 1] - \Pr[\mathsf{Real}_{\mathcal{A}}^{\Pi}(\lambda) = 1] \leq negl(\lambda).$$

G_1 : Same as G_0 but with the exception that some changes were made in G_1 line of TABLE 2. More concretely, in G_1, the stag values are stored after they are first calculated rather than recomputing them, then look up them again in the future. Scilicet, for any $t \in [T]$, set query_stag $\leftarrow$ stag$[\mathsf{s}[t]]$, and for each $w \in W$, set stag$[w]$ $\leftarrow$ stag such that stag $\leftarrow F(K_E, g_1^{1/w} \mod n)$. By calculation of the stags in $\mathsf{ETokenGen}$ algorithm, this game is equally distributed with game G_0. So, the following equation holds

$$\Pr[\mathsf{G}_1 = 1] = \Pr[\mathsf{G}_0 = 1].$$

G_2 : Identical to the game G_1 except that we use random functions to replace the PRFs F and F_p, where the changes are given in the G_2 line of TABLE 2. Note that the evaluation

Algorithm 4 G_0

```
1: function INITIALIZE ($1^\lambda$, $par_L$, $par_E$, $\mathsf{EDB}$, $\mathsf{DB}_{DW}$, $\mathsf{w}$, $\mathsf{s}$, $\mathsf{x}$, $id_U$, $id_{DW}$)
   //For each $t \in [T]$, the keywords associated with this query satisfy that $\mathsf{s}[t] \cup \mathsf{x}[t,.] \subseteq \mathsf{w}$, $id_U$ is a data user's identity and $id_{DW}$ is a data owner's identity.
2: $(\mathbb{G}_1', N, \mathbb{G}_1, g', q') \leftarrow par_L$; $\mathsf{H} \leftarrow par_E$; $(\mathrm{ind}_i, W_i)_{i=1}^d \leftarrow \mathsf{DB}_{DW}$; $W \leftarrow \bigcup_{i=1}^d W_i$
3: $(mpk, msk) \leftarrow_\$ \mathsf{ABE.Setup}(1^\lambda)$
4: $(\mathbb{G}_2, g, n, p, q) \leftarrow_\$ \mathcal{G}(1^\lambda)$
5: $g_1, g_2, g_3, g \leftarrow_\$ \mathbb{G}_2$
6: $K_E, K_X, K_I, K_Z \leftarrow_\$ \{0,1\}^\lambda$
7: $\mathrm{EPK}_{DW} \leftarrow (n, g, mpk)$
8: $\mathrm{ESK}_{DW} \leftarrow (K_X, K_I, K_Z, K_E, p, q, g_1, g_2, g_3, msk)$
9: $\mathsf{EDB}_{DW} \leftarrow \{\}$; $\mathsf{TSet}_{DW} \leftarrow \emptyset$; $\mathsf{XSet}_{DW} \leftarrow \emptyset$; $\mathcal{L}_{DW} \leftarrow \emptyset$; $sk_{\mathsf{w}} \leftarrow \emptyset$
10:
11: for $w \in W$ do
12:     $(\mathrm{ind}_1, \ldots, \mathrm{ind}_{T_w}) \leftarrow \mathsf{DB}[w]$
13:     $\sigma \xleftarrow{\$} \mathrm{Perm}([T_w])$
14:     $\mathrm{WPerm}[w] \xleftarrow{\$} \sigma$
15:     $\mathrm{stag} \leftarrow F(K_E, g_1^{1/w} \mod n)$
16:     for $c \in T_w$ do
17:         $l \leftarrow F(\mathrm{stag}, c)$
18:         $\mathcal{L}_{DW} \leftarrow \mathcal{L}_{DW} \cup \{l\}$
19:         $e \leftarrow \mathsf{ABE.Enc}(mpk, \mathrm{ind}_{\sigma[c]}, \mathbb{A})$
20:         $\mathrm{xind} \leftarrow F_p(K_I, \mathrm{id}_{\sigma[c]})$;
21:         $z \leftarrow F_p(K_Z, g_2^{1/w} \mod n || c)$
22:         $y \leftarrow \mathrm{xind} \cdot z^{-1}$
23:         $\mathsf{TSet}_{DW}[l] = (e, y)$
24:     end for
25: end for
26: $(pk_U, sk_U) \leftarrow \mathrm{LKeyGen}(par_L, id_U)$
27: $(pk_V, sk_V) \leftarrow \mathrm{LKeyGen}(par_L, id_V)$
28: $(\mathrm{PK}_{DW}, \mathrm{MK}_{DW}) \leftarrow \mathrm{EDBKeyGen}(par, \mathrm{MK}, id_{DW})$
29: $\mathsf{XSet}_{DW} \leftarrow \mathrm{XSetSetup}(n, K_X, K_I, \mathsf{DB})$
30: $id_{\mathsf{EDB}_{DW}} \leftarrow \mathsf{H}(\mathsf{TSet}_{DW}, \mathsf{XSet}_{DW})$
31: $\mathrm{SK}_U \leftarrow \mathrm{SKeyGen}(\mathrm{EPK}_{DW}, \mathrm{ESK}_{DW}, S, \mathsf{w}, id_U)$
32: $\mathsf{EDB}_{DW} \leftarrow (\mathsf{TSet}_{DW}, \mathsf{XSet}_{DW}, id_{\mathsf{EDB}_{DW}}, \mathcal{L}_{DW})$
33: $\mathsf{EDB} \leftarrow \mathsf{EDB} \cup \{\mathsf{EDB}_{DW}\}$
34:
35: for $t \in [T]$ do
36:     $\mathrm{query\_stag} \leftarrow F(K_E, (sk_{\mathsf{w}}^{(1)})^{\prod_{w \in \mathsf{w} \backslash \{s_t\}} w} \mod n)$
37:     $\mathbf{t}[t] \leftarrow \mathrm{TransGen}(\mathsf{DB}_{DW}, \mathsf{EDB}_{DW}, \mathrm{SK}_U, sk_U, sk_V, pid_U, pid_V, id_{\mathsf{EDB}_{DW}}, \mathsf{s}[t], \mathsf{x}[t,.], \mathrm{query\_stag})$
38: end for
39: return $(\mathsf{EDB}_{DW}, \mathbf{t})$
40: end function
41: function XSetSetup($n$, $K_X$, $K_I$, $\mathsf{DB}_{DW}$)
42: $(\mathrm{ind}_i, W_i)_{i=1}^d \leftarrow \mathsf{DB}_{DW}$; $W \leftarrow \bigcup_{i=1}^d W_i$;
43: for $w \in W$ and $\mathrm{ind} \in \mathsf{DB}_{DW}[w]$ do
44:     $\mathrm{xind} \leftarrow F_p(K_I, \mathrm{ind})$
45:     $\mathrm{xtag} \leftarrow g^{F_p(K_X, g_3^{1/w} \mod n) \cdot \mathrm{xind}}$
46:     $\mathsf{XSet}_{DW} \leftarrow \mathsf{XSet}_{DW} \cup \{\mathrm{xtag}\}$
47: end for
48: return $\mathsf{XSet}$
49: end function
50:
51: function SKeyGen($\mathrm{EPK}_{DW}$, $\mathrm{ESK}_{DW}$, $S$, $\mathsf{w}$, $id_U$)
52: $(n, g, mpk) \leftarrow \mathrm{EPK}_{DW}$
53: $(K_X, K_I, K_Z, K_E, p, q, g_1, g_2, g_3, msk) \leftarrow \mathrm{ESK}_{DW}$
54: for $i \in [3]$ do
55:     $sk_{\mathsf{w}}^{(i)} \leftarrow g_i^{1/\prod_{j=1}^n w_j}$
56: end for
57: $sk_{\mathsf{w}} \leftarrow (sk_{\mathsf{w}}^{(1)}, sk_{\mathsf{w}}^{(2)}, sk_{\mathsf{w}}^{(3)})$
58: $sk_S \leftarrow \mathsf{ABE.KeyGen}(msk, S)$
59: $\mathrm{SK}_{MS,U} \leftarrow (K_E, K_X, K_Z, sk_{\mathsf{w}}, id_{\mathsf{EDB}_{DW}})$
60: $\mathrm{SK}_{S,U} \leftarrow sk_S$
61: $\mathrm{SK}_U \leftarrow (\mathrm{SK}_{MS,U}, \mathrm{SK}_{S,U})$
62: return $\mathrm{SK}_U \leftarrow (\mathrm{SK}_{MS,U}, \mathrm{SK}_{S,U})$
63: end function
64:
65: function TransGen($\mathsf{DB}_{DW}$, $\mathsf{EDB}_{DW}$, $\mathrm{SK}_U$, $sk_U$, $sk_V$, $pid_U$, $pid_V$, $id_{\mathsf{EDB}_{DW}}$, $\mathsf{s}[t]$, $\mathsf{x}[t,.]$, query_stag)
66: $(\mathrm{ind}_i, W_i)_{i=1}^d \leftarrow \mathsf{DB}_{DW}$
67: $(\mathsf{TSet}_{DW}, \mathsf{XSet}_{DW}, id_{\mathsf{EDB}_{DW}}, \mathcal{L}_{DW}) \leftarrow \mathsf{EDB}_{DW}$
68: $(K_E, K_X, K_Z, sk_{\mathsf{w}}, id_{\mathsf{EDB}_{DW}}, sk_S) \leftarrow \mathrm{SK}_U$
69: $u \leftarrow sk_U$; $v \leftarrow sk_V$; $(id_U, U = g'^u, cert_U) \leftarrow pid_U$; $(id_V, V = g'^v, cert_V) \leftarrow pid_V$
70: for $\alpha \in [|\mathsf{x}_t|]$ do
71:     for $c \in [T_c]$ do
72:         $\mathrm{xtoken}[\alpha, c] \leftarrow g^{F_p(K_Z, (sk_{\mathsf{w}}^{(2)})^{\prod_{w \in \mathsf{w} \backslash \{s_t\}} w} \mod n || c) \cdot}$
73:             $F_p(K_X, (sk_{\mathsf{w}}^{(3)})^{\prod_{w \in \mathsf{w} \backslash \{\mathsf{x}_t[\alpha]\}} w} \mod n)$
74:     end for
75: end for
76: $st_t \leftarrow (\mathrm{query\_stag}, \mathrm{xtoken})$
    Authenticationphase :
77: $x \leftarrow Z_{q'}^*$, $X = g'^x$, $d = h(X, cert_U, cert_V, st_t, id_{\mathsf{EDB}_{DW}})$
78: $\overline{X} = UX^d$, $PS = V^{u+xd}$
79: $K_{\mathsf{AEAD}} = \mathsf{KDF}(PS, \overline{X} || cert_V)$
80: $M \leftarrow st_t || id_{\mathsf{EDB}_{DW}}$
81: $c_{AE} \leftarrow \mathsf{AEAD.Enc}_{K_{\mathsf{AEAD}}}(H, cert_U || X || M, ts)$
82: $est_t \leftarrow (H, \overline{X}, c_{AE})$
83: $\mathrm{Res} \leftarrow \mathrm{Search}(sk_V, pid_V, pid_U, est_t, \mathsf{EDB})$
84: $\mathrm{ResInds} \leftarrow \mathsf{DB}_{DW}[\mathsf{s}_t, \mathsf{x}_t]$
85: return $((H, \overline{X}, c_{AE}), \mathrm{Res}, \mathrm{ResInds})$
86: end function
```

$F(K_E, \cdot)$ can be chosen from the range of the PRF F instead of calculating it on its inputs since its output is only executed on the same input once. For $F_p(K_X, \cdot)$, $F_p(K_I, \cdot)$ and $F_p(K_Z, \cdot)$, they are replaced with random functions f_X, f_I and f_Z respectively. By a standard hybrid argument, we can see that there exists two efficient adversaries $\mathcal{A}_{1,1}$ and $\mathcal{A}_{1,2}$ such that

$$\Pr[\mathsf{G}_2 = 1] - \Pr[\mathsf{G}_1 = 1] \leq \mathsf{Adv}_{F,\mathcal{A}_{1,1}}^{prf}(\lambda) + 3\mathsf{Adv}_{F_p,\mathcal{A}_{1,2}}^{prf}(\lambda).$$

G_3 : This game is the same as G_2 except that we encrypt 0^λ instead of document indexes, which is demonstrated in G_3 line of TABLE 2. By the CPA security of the ABE scheme and a standard hybrid argument, we can observe that there exists an efficient adversary $\mathcal{B}_2$ satisfying the following inequality

$$\Pr[\mathsf{G}_3 = 1] - \Pr[\mathsf{G}_2 = 1] \leq poly(\lambda).\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{ABE},\mathcal{B}_2}(\lambda).$$

The reduction is easily got from the CPA security poof of the ABE scheme. To save space, we hereby omit the proof details.

G_4 : Same as G_3 but with the exception is that the XSet and xtoken functions are defined in a different way which are demonstrated in the G_4 line of TABLE 2. Roughly speaking, we recompute each possible values in set XSet as $\mathcal{H}(\mathrm{ind}_i, w) = g^{f_X(g_3^{1/w} \bmod n).f_I(\mathrm{ind}_i)}$ for each index ind_i and keyword $w \in W$ and place them in the array $\mathcal{H}$. In addition, for xtoken not matching w are saved in another array Y.

In G_4, XSetSetup assigns the values in $\mathcal{H}$ to the elements in the set XSet. Specifically, for a specified $w \in W$ and ind $\in \mathsf{DB}_{DW}[w]$, we set the result $\mathcal{H}(\mathrm{ind}, w)$ as $g^{f_X(g_3^{1/w} \bmod n).f_I(\mathrm{ind})}$ during the INITIALIZE procedure. It is easy to see that this presentation is the same as that in game G_3. Moreover, note that the outputs of TransGen have the same value as game G_3 if both games have the same values in xtoken. Hence, we only pay attention to how to generate xtoken.

In game G_3, we compute xtoken as in the real game. Concretely, the value $\mathsf{xtoken}[\alpha, c]$ for each $\mathsf{x_t}[\alpha]$ and $c \in [T_c]$ is assigned to $\mathcal{H}[\mathrm{ind}_{\sigma[c]}, \mathsf{x}_t[\alpha]]^{1/y}$, which is equal to $g^{f_Z(g_2^{1/\mathsf{s}_t} \bmod n||c).f_X(g_3^{1/\mathsf{x}_t[\alpha]} \bmod n)}$, where T_c denotes the number of the document indexes that match the sterms s_t. In the game G_4, xtoken is constructed as as follows. On input DB_{DW}, EDB_{DW}, SK_U, sk_U, sk_V, pid_U, pid_V, $id_{\mathsf{EDB}_{DW}}$, $\mathsf{s}[t]$, $\mathsf{x}[t, \cdot]$, and query_stag, the algorithm TransGen first parses $(\mathrm{ind}_1, \cdots, \mathrm{ind}_{T_s}) \leftarrow \mathsf{DB}_{DW}[\mathsf{s}_t]$ and $\sigma \leftarrow \mathrm{WPerms}[\mathsf{s}_t]$ and then for each $\mathsf{x}_t[\alpha]$ and $c \in [T_c]$, it uses query_stage to compute $l = F(\text{query\_stag}, c)$ and l to locate $\mathsf{TSet}[l] = (e, y)$, where $y = f_I(\mathrm{ind}_{\sigma[c]}).(f_Z(g_2^{1/\mathsf{s}_t}) \bmod n||c)^{-1}$. The value $\mathsf{xtoken}[\alpha, c]$ is computed as:

- if $c \in [T_s]$ then $\mathsf{xtoken}[\alpha, c] = \mathcal{H}[\mathrm{ind}_{\sigma[c]}, \mathsf{x}_t[\alpha]]^{1/y}$
- if $c \in [T_c] \backslash [T_s]$ then $\mathsf{xtoken}[\alpha, c] = Y[\mathsf{s}_t, \mathsf{x}_t[\alpha], c]$

By above, we can see that $\mathsf{xtoken}[\alpha, c] = g^{f}_{Z}(g_2^{1/\mathsf{s}_t} \bmod n||c).f_X(g_3^{1/\mathsf{x}_t[\alpha]} \bmod n)$ for any $\mathsf{x}_t[\alpha]$ and $c \in [T_c]$ implies that the xtoken values in G_4 and G_3 are completely identical. So the following equations holds

$$\Pr[\mathsf{G}_4 = 1] = \Pr[\mathsf{G}_3 = 1].$$

G_5 : Same as G_4 but with the only difference is that we set $y \leftarrow_\$ Z_p^*$, which is demonstrated in the G_5 line of TABLE 2. Observe that in game G_4, the value $f_Z(g_2^{1/w} \bmod n||c)$ is used only one time during the INITIALIZE phase. Hence, the equation $\Pr[\mathsf{G}_5 = 1] = \Pr[\mathsf{G}_4 = 1]$ holds.

G_6 : Identical with G_5 but with a little difference that all the values in $\mathcal{H}$ and Y are randomly selected from $\mathbb{G}_2$ (see for G_6 line of TABLE 2). Therefore, we conclude that there exists an efficient adversary $\mathcal{A}_{\mathsf{DDH}}$ satisfying

$$\Pr[\mathsf{G}_6 = 1] - \Pr[\mathsf{G}_5 = 1] \leq \mathsf{Aadv}^{\mathsf{DDH}}_{\mathbb{G}_2, \mathcal{A}_{\mathsf{DDH}}}$$

To prove this conclusion, we can construct a reduction in which an adversary $\mathcal{A}$ could break the indistinguishability of G_5 and G_6, then a reduction algorithm $\mathcal{A}_{\mathsf{DDH}}$ can be built using $\mathcal{A}$ to break the DDH assumptions. Roughly speaking, we set the values of X array in game G_5 as g^a, and xind as b of DDH tuple. Therefore, $\mathcal{H}$ and Y in G_5 have the value of the form g^{ab}, while in G_6, they are randomly and uniformly. Thus, the indistinguishability between the two games is easy to reduce to the DDH assumption.

G_7 : Same as G_6 but with the exception that we only include the elements of $\mathcal{H}$ in XSetSetup and the difference is demonstrated in the G_7 line of TABLE 2. Obviously, the indistinguishability directly comes from the fact that the generated array $\mathcal{H}$ is only used in the functions XSetSetup and TransGen. Since the detailed proof is similar to that of Sun *et al.*'s scheme [34], we omit it here. Thus, we have $\Pr[\mathsf{G}_7] = \Pr[\mathsf{G}_6]$.

G_8 : Same as G_7 but with the exception that the way accessing $\mathcal{H}$ in TransGen is modified. In this game it only uses the elements of $\mathcal{H}$ to compute xtoken. The difference is demonstrated in the G_8 line of TABLE 2. To check whether an element (e.g., indexed by (ind, w)) in $\mathcal{H}$ is reused, we must test if XSetSetup has used this index, or whether TransGen will reuse it. In the last game, XSetSetup was changed so as to it only uses the element $\mathcal{H}(\mathrm{ind}, w)$ if both the conditions $\mathsf{DB}[\mathsf{s}_t] \cap \mathsf{DB}[\mathsf{x}_t[\alpha]]$ and $w = \mathsf{x}_t[\alpha]$ are true, which is precisely seized by the first "if" statement in the TransGen function in game G_8. Whereas, it maybe have another possibility that TransGen accesses the same element twice. Observe that this case only appears when TransGen is invoked for two distinct queries since one running of the function only visits a single element of $\mathcal{H}$. For simplicity, we set the current t as an input argument of this function. In more details, for an element of index (ind, w) that will be visited two times, it requires that both conditions ind $\in \mathsf{DB}[\mathsf{s}_t] \cap \mathsf{DB}[\mathsf{s}_{t'}]$ and $w = \mathsf{x}_t[\alpha] \in \mathsf{x}_{t'}$ are satisfied simultaneously for some $t' \neq t$, which is precisely captured by the second "if" statement. When both do not hold, we set the xtoken as a random value from its range. By a simple argument, we have $\Pr[\mathsf{G}_8 = 1] = \Pr[\mathsf{G}_7 = 1]$.

G_9 : Almost the same as G_8 but with the difference is that the Search algorithm is substituted with the algorithm SearchRes, which is shown in the G_9 line of TABLE 2. In particular, the function SearchRes does not need any private key of the server, which can be defined by taking (EDB, query_stag, xtoken, $id_{\mathsf{EDB}_{DW}}$) as input and running the codes in the lines 11-21 in Algorithm 3. It is easy to see that the two functions have the same outputs in both games G_9 and G_8. Thus we have $\Pr[\mathsf{G}_9] = \Pr[\mathsf{G}_8]$.

G_{10} : This game is the same as G_9 except that the secret key of the AEAD scheme $K_{\mathsf{AEAD}} \leftarrow_\$ \mathcal{K}_{\mathsf{AEAD}}$ is selected randomly and uniformly from its key space $\mathcal{K}_{\mathsf{AEAD}}$ (see G_{10} line of TABLE 2), while in game G_{10}, K_{AEAD} is computed as $K_{\mathsf{AEAD}} \leftarrow \mathsf{KDF}(PS, X||cert_V)$. In particular, since the KDF function can be seen as a random oracle, so its output is random and uniform. By a simple argument, the following equations hold

$$\Pr[\mathsf{G}_{10}] = \Pr[\mathsf{G}_9].$$

G_{11} : Almost the same as G_{10} with the only difference is that the AEAD ciphertext c_{AE} is substituted with the encryption of a constant message $0^{l_{\mathsf{AEAD}}}$, where l_{AEAD} denotes the message length of the scheme AEAD. We show the differences in the G_{11} line of TABLE 2. By the security of the scheme AEAD and a simple reduction, we conclude that the current generated ciphertext is indistinguishable from the encryption of the message $cert_U||X||M$. More specifically, we assume a PPT adversary $\mathcal{A}$ can distinguish the two games, then another PPT adversary $\mathcal{A}_{\mathsf{AEAD}}$ can be constructed to break the AEAD security of the scheme AEAD via employing the former. Particularly, except the AEAD ciphertext are generated from the challenger of the adversary $\mathcal{A}_{\mathsf{AEAD}}$, the rest values can be produced by the adversary itself. Thus, we have the following inequality hold

$$\Pr[\mathsf{G}_{11}] - \Pr[\mathsf{G}_{10}] \leq \mathsf{Adv}^{sec}_{\mathsf{AEAD},\mathcal{A}_{\mathsf{AEAD}}}(\lambda).$$

Simulator. The simulator is first constructed by taking a leakage $\mathcal{L}_{leak}$ $(\mathsf{DB}, \mathsf{s}, \mathsf{x}) = (DW, N, \bar{s}, \mathrm{SP}, \mathrm{RP}, \mathrm{SRP}, \mathrm{IP}, \mathrm{XT})$ as input and a simulated encrypted database EDB and transcript t as output. Then we prove that the simulator generates the identical distribution as G_8. By the indistinguishability between any two adjacent games, the simulator is proved structurally correct according to the requirements in the theorem. The simulator starts by generating a restricted equality pattern $\hat{\mathsf{x}}$ of x as in [34]. For completeness, we describe it again in details. Concretely, in the simulation model, we construct the simulated TSet, XSet and transcript t respectively in Algorithms 5.

Given an encrypted database EDB, a restricted equality pattern can be inferred as the server can decide which xterms are equal according to the elements in the set XSet with a overwhelming probability. This leakage can be represented by the IP structure. When there exists a document index ind and two different queries t_1 and t_2 with ind $\in \mathsf{DB}[\mathsf{s}[t_1]] \cap \mathsf{DB}[\mathsf{s}[t_2]]$, then the server can infer that the two xterms $\mathsf{x}[t_1, \alpha]$ and $\mathsf{x}[t_2, \beta]$ might be equal by observing if there exists $\mathcal{H}(\mathsf{x}[t_1, \alpha], \mathrm{ind}) = \mathcal{H}(\mathsf{x}[t_2, \beta], \mathrm{ind})$. This can be constructed according to the structure IP by defining a $T \times A$ table $\hat{\mathsf{x}}[t, \alpha]$ s.t. $\hat{\mathsf{x}}[t_1, \alpha] = \hat{\mathsf{x}}[t_2, \beta]$, if and only if $\mathrm{IP}[t_1, t_2, \alpha, \beta] \neq \phi$, where $\hat{\mathsf{x}}$ is an array of integers. The table $\hat{\mathsf{x}}$ demonstrates that the server knows which xterms are equal. Concretely, we have

(1) $\hat{\mathsf{x}}[t_1, \alpha] = \hat{\mathsf{x}}[t_2, \beta] \Longrightarrow \mathsf{x}[t_1, \alpha] = \mathsf{x}[t_2, \beta]$,

(2) $(\mathsf{x}[t_1, \alpha] = \mathsf{x}[t_2, \beta]) \wedge (\mathsf{DB}[\mathsf{s}[t_1]] \cap \mathsf{DB}[\mathsf{s}[t_2]] \neq \phi) \Longrightarrow \hat{\mathsf{x}}[t_1, \alpha] = \hat{\mathsf{x}}[t_2, \beta]$.

Note that the element $\hat{\mathsf{x}}[t_2, \beta]$ is defined as $\hat{\mathsf{x}}[t_1, \alpha]$ if both conditions $(t_1, \alpha) < (t_2, \beta)$ and $\mathrm{IP}[t_1, t_2, \alpha, \beta] \neq \phi$ hold. The simulated encrypted database component is constructed as in TSet in Algorithm 5. The main difference between the Game G_{11} and simulated TSet code is that game G_{11} fills out TSet with $w \in W$, but the simulator fills out only with $i \in \bar{\mathsf{s}}$. By the structure of $\bar{\mathsf{s}}$, we conclude $|\bar{\mathsf{s}}| < |W|$. Because N is the number of the elements in the set TSet, the simulator fills out the other $N - |\bar{\mathsf{s}}|$ positions with random entries. In both cases, the keys in TSet are indistinguishable, and each of its corresponding values is the tuple (e, y) consisting of the ABE ciphertext $e \leftarrow \mathsf{ABE.Enc}(mpk, 0^\lambda, \mathbb{A})$ and the random value $y \leftarrow_\$ \mathbb{G}_2$. So it is easy to see that the distributions of TSet in game G_{11} and the simulator are indistinguishable.

The simulated XSet is constructed according to Algorithm 5, we prove that its distribution including the set TSet and xtokens is identical to that in game G_{11}. It is easy to see that all entries in XSet in both cases are random values with the only difference is that in game G_{11}, it is done by traversing every pair of $(w \in W$, ind $\in \mathsf{DB}[w])$ and completely producing $\Sigma_{w\in W}\mathsf{DB}[w]$, while in the simulation game, it is processed by keeping track of every element in XSet with a counter j until N elements are added in. Next, we prove that the elements in TSet and xtokens have the same distributions.

To prove the above claim, we will demonstrate how the xtokens are generated. We describe the construction of the simulated transcript t and xotokes in Algorithm 5. First, it is easy to see that the values y and σ in G_{11} and the simulated t (see Algorithm 5) have the same distributions since these values are all chosen uniformly and randomly. In addition, in both G_{11} and the simulation games, the re-usage of the permutation, σ, is the same, both according to the values that are reused in $\bar{\mathsf{s}}$.

Furthermore, we show that both G_{11} and the simulator use the same $\mathcal{H}$ array when generating xtoken. In game G_{11}, $\mathcal{H}$ is used in the following two cases: first, ind matches a conjunction within the query that uses w conditioned on $\exists\alpha$ with ind $\in \mathsf{DB}[\mathsf{s}_t] \cap \mathsf{x}_t[\alpha] = w$; second, the value is reused in the the subsequent queries. The same operation is done in the simulation mode.

Next, we show that when some elements at the same locations in $\mathcal{H}$ are used for multiple times, the process is the same in both G_{11} and the simulation algorithm. Concretely, when two tuples satisfy $(\mathrm{ind}_1, \mathsf{x}_{t_1}(\alpha)) = (\mathrm{ind}_2, \mathsf{x}_{t_2}(\beta))$ in game G_{11}, then there exists two tuples satisfying $(\mathrm{ind}_1, \bar{\mathsf{x}}_{t_1}(\alpha)) = (\mathrm{ind}_2, \bar{\mathsf{x}}_{t_2}(\beta))$ in the simulation algorithm. More formally, we have $(\mathrm{ind}_1, \mathsf{x}_{t_1}(\alpha)) = (\mathrm{ind}_2, \mathsf{x}_{t_2}(\beta)) \Longleftrightarrow (\mathrm{ind}_1, \bar{\mathsf{x}}_{t_1}(\alpha)) = (\mathrm{ind}_2, \bar{\mathsf{x}}_{t_2}(\beta))$

Now we show the above equivalent formula. First, by the construction of the table $\bar{\mathsf{x}}$, it is easy to see that the equation $(\mathrm{ind}_1, \mathsf{x}_{t_1}(\alpha)) = (\mathrm{ind}_2, \mathsf{x}_{t_2}(\beta)) \Longleftarrow (\mathrm{ind}_1, \bar{\mathsf{x}}_{t_1}(\alpha)) = (\mathrm{ind}_2, \bar{\mathsf{x}}_{t_2}(\beta))$ holds, so the $\Longleftarrow$ relation holds. For the $\Longrightarrow$ relation, by the previous description, we know that the statement $(\mathsf{x}_{t_1}[\alpha] = \mathsf{x}_{t_2}[\beta]) \wedge (\mathsf{DB}[\mathsf{s}_{t_1}] \cap \mathsf{DB}[\mathsf{s}_{t_2}] \neq \phi) \Longrightarrow (\bar{\mathsf{x}}_{t_1}[\alpha] = \bar{\mathsf{x}}_{t_2}[\beta])$. holds

Therefore, we have that if $\mathsf{DB}[\mathsf{s}_{t_1}] \cap \mathsf{DB}[\mathsf{s}_{t_2}] \neq \phi$, then $\Longrightarrow$ relation holds. Again if the condition $(\mathrm{ind}_1, \mathsf{x}_{t_1}(\alpha)) = (\mathrm{ind}_2, \mathsf{x}_{t_2}(\beta))$ holds, then we have $\mathrm{ind}_1 = \mathrm{ind}_2$. Thus the intersection $\mathsf{DB}[\mathsf{s}_{t_1}] \cap \mathsf{DB}[\mathsf{s}_{t_2}]$ contains at least this ind (for ind $= \mathrm{ind}_1 = \mathrm{ind}_2$), and is nonempty.

Algorithm 5 Simulation Algorithms

```
Algorithm: Simulated TSet
$mpk, msk$
$\mathcal{L} \leftarrow \{\}$; $\mathsf{TSet} \leftarrow \emptyset$
for $i \in \bar{\mathsf{s}}$ do
    $\mathrm{stags}[i] \leftarrow \{0,1\}^\lambda; j \leftarrow 0$
    for $c \in [\mathrm{SP}[i]]$ do
        $l \leftarrow F(\mathrm{stag}, c)$
        $\mathcal{L} \leftarrow \mathcal{L} \cup \{l\}$
        $e \leftarrow \mathsf{ABE.Enc}(mpk, 0^\lambda, \mathbb{A})$
        $y \xleftarrow{\$} Z_p^*$
        $\mathsf{TSet}[l] \leftarrow (e, y)$
        $j = j + 1$
    end for
end for
for $i = j+1, \ldots, N$ do
    $l \leftarrow \{0,1\}^\lambda$
    $\mathcal{L} \leftarrow \mathcal{L} \cup \{l\}$
    $e \leftarrow \mathsf{ABE.Enc}(mpk, 0^\lambda, \mathbb{A})$
    $y \xleftarrow{\$} Z_p^*$
    $\mathsf{TSet}[l] \leftarrow (e, y)$
end for

Algorithm: Simulated XSet
for $w \in \hat{\mathsf{x}}$ and ind $\in \cup_{t\in[T],\alpha\in[A]}\mathrm{RP}[t,\alpha]$ do
    $\mathcal{H}[\mathrm{ind}, w] \xleftarrow{\$} \mathbb{G}_2$
end for
$\mathsf{XSet} \leftarrow \emptyset; j \leftarrow 0$
for $w \in \hat{\mathsf{x}}$ and ind $\in \cup_{\{(t,\alpha):\ \hat{\mathsf{x}}[t,\alpha]=w\}}\mathrm{RP}[t,\alpha]$ do
    $\mathsf{XSet} \leftarrow \mathsf{XSet} \cup \{\mathcal{H}[\mathrm{ind}, w]\}$
    $j \leftarrow j + 1$
end for
for $i = j+1, \ldots, N$ do
    $h \xleftarrow{\$} \mathbb{G}_2$; $\mathsf{XSet} \leftarrow \mathsf{XSet} \cup \{h\}$
end for

Algorithm: Simulated t
for $w \in \bar{\mathsf{s}}$ do
    $\mathrm{WPerms}[w] \xleftarrow{\$} \mathrm{Perm}([\mathrm{SP}[w]])$
end for
for $\tau \in [T]$ do
    $\mathrm{query\_stag} \leftarrow \mathrm{stags}[\bar{\mathsf{s}}[\tau]]$
    for $w_x \in [\mathrm{XT}[\tau]]$ do
        $R \leftarrow \mathrm{RP}[\tau, w_x] \cup \bigcup_{t'\in[T],\beta\in[\mathrm{XT}[\tau]]} \mathrm{IP}[\tau, t', w_x, \beta]$
        $c \leftarrow 1$
        for ind $\in \mathrm{WPerms}[\bar{\mathsf{s}}[\tau]]$ do
            $(\mathrm{ind}_1, \ldots, \mathrm{ind}_{T_s}) \leftarrow \mathrm{SRP}[\tau]$; $\sigma \leftarrow \mathrm{WPerms}[\bar{\mathsf{S}}[\tau]]$

            for $c \in [T_s]$ do
                if $\mathrm{ind}_{\sigma[c]} \in R$ then
                    $l \leftarrow F(\mathrm{query\_stag}, c)$
                    $\mathcal{L} \leftarrow \mathcal{L} \cup \{l\}$
                    $(e, y) \leftarrow \mathsf{TSet}[l]$
                    $\mathrm{xtoken}[\tau, w_x, c] \leftarrow \mathcal{H}[\mathrm{ind}_{\sigma[c]}, \hat{\mathsf{x}}[\tau, w_x]]^{1/y}$
                else
                    $\mathrm{xtoken}[\tau, w_x, c] \xleftarrow{\$} \mathbb{G}_2$
                end if
                $c \leftarrow c + 1$
            end for
        end for
        for $c = \mathrm{SP}[\bar{\mathsf{s}}[\tau]] + 1, \ldots, T_c$ do $\mathrm{xtoken}[\tau, w_x, c] \xleftarrow{\$} \mathbb{G}_2$ endfor
    end for
    $st_\tau \leftarrow (\mathrm{query\_stag}, \mathrm{xtoken})$
    Authentication phase :
    $x \leftarrow_\$ Z_{q'}^*, X = g'^x$
    $d = h(X, cert_U, cert_V, st_\tau, id_{\mathsf{EDB}_{DW}})$ //where $cert_U$ and $cert_V$ are respectively the data user and server's certificates, $id_{\mathsf{EDB}_{DW}}$ is an encrypted database public identifier
    $\overline{X} = UX^d$ //$U$ is the data user's public key
    $K_{\mathsf{AEAD}} \leftarrow_\$ \mathcal{K}_{\mathsf{AEAD}}$ //$\mathcal{K}_{\mathsf{AEAD}}$ is the key space of the scheme AEAD
    $c_{AE} \leftarrow \mathsf{AEAD.Enc}_{K_{\mathsf{AEAD}}}(H, 0^{l_{\mathsf{AEAD}}}, ts)$ // where $l_{\mathsf{AEAD}} = |cert_U||X||M|$
    $est_\tau \leftarrow (H, \overline{X}, c_{AE})$
    Res$\leftarrow$ SearchRes($\mathsf{EDB}$,(query_stag,xtoken),$id_{\mathsf{EDB}_{DW}}$)// where $\mathsf{EDB} = \{(\mathsf{TSet}_{DW}, \mathsf{XSet}_{DW}, id_{\mathsf{EDB}_{DW}}, \mathcal{L}_{DW})\}$ is the current full encrypted database
    ResInds $\leftarrow$ Real_Result$(\tau, \mathrm{RP})$
    $\mathsf{t}[\tau] = (est_\tau, \mathrm{Res}, \mathrm{ResInds})$
end for
```

The values Res in both G_{11} and the simulated t (in Algorithm 5) are distributed exactly identical. Concretely, in both cases, the Res values are obtained by running the function SearchRes on input (EDB, query_stag, stag, $id_{\mathsf{EDB}_{DW}}$), which is defined by directly performing the codes in the lines 11-21 of the Search Algorithm 3 which returns the same results as the real game, where $id_{\mathsf{EDB}_{DW}}$ denotes the identifier of encrypted database.

Now we show that the value ResInds in both G_{11} and the simulated t (in Algorithm 5) is also distributed exactly identical. In the simulation algorithm, the ResInds is obtained by running the function Real_Result on the inputs (τ, RP), which is defined by taking the document index that matches the conjunction keywords in the query and returns the matched document indexes. In game G_{11}, the value ResInds is obtained by taking the document indexes that match the conjunction keywords from the unencrypted database. Obviously, the values obtained in both cases are the same.

Theorem 2 (Authentication): If the GDH assumption in $\mathbb{G}_1$ *holds and the* AEAD *scheme is ae-sec, then the* *scheme Π proposed in section IV satisfies authentication in the RO model.*

Proof: The detailed proof about this Theorem is omitted here. We could guide the reader to the outsider unforgeability part of Theorem 4 in Section 6 in [37]. In the following, we give an overview.

Overview. Note that the authentication here includes the unforgeability of a plaintext st for some non-authorized w' and the unforgeability of the signcryption ciphertext est (i.e., encrypted search token). In particular, it also includes the authentication from data user to the server. First, for the unforgeability of plaintext search token, assume that a PPT adversary can forge a st' for some non-authorized w', then the value ($g_j^{1/w'}$ mod n) for $j \in [3]$ could be correctly guessed. If so, a PPT adversary $\mathcal{A}_{sRSA}$ could be constructed via $\mathcal{A}$ to solve the sRSA problem (i.e., strong RSA problem). We omit the detailed reduction here and the reader can refer to [34] for further description.

For the unforgeability of a signcryption ciphertext (i.e., an encrypted search token), we can reduce it to the GDH assumption on the cyclic group $\mathbb{G}_1$. At the beginning of the reduction, we first assume that there exists an honest data user id_{U^*} and server id_{V^*} participating in this game, and they could be correctly predicted with probability at least $1/n^2$ where n is the total number of users (which contains data owners, data users and server). Then, assume a PPT adversary $\mathcal{A}$ could successfully forge a tuple $(pid_{V^*}, (H, \bar{X}, c_{AE}))$ and $(H, \bar{X}, c_{AE})$ denotes the signcryption ciphertext on the plaintext search token st^*, public identity information pid_{U^*} and id_{V^*}. Then using $\mathcal{A}$, an efficient algorithm $\mathcal{A}_{GDH}$ could be constructed to break the GDH assumption with the aid of the DDH oracle. For the detailed description about the reduction, the reader can refer to the proof of outsider unforgeability of Theorem 4 in Section 6 in [37].

The authentication of data user to server is proved by combining certificate-based mechanism and the outside unforgeability of signcryption ciphertext. In other words, as long as the certificate generated by the CA is unforgeable and signcryption ciphertext generated by the data user has outside unforgeability security, then our scheme provides the authentication of data user to server. Please refer to [37] for the detailed reduction.

Theorem 3 (Identity-Concealment and Confidentiality): Assuming the GDH in $\mathbb{G}_1$ holds and the AEAD *scheme is ae-sec, then the scheme Π proposed in section IV satisfies identity-concealment (forward ID-privacy) and confidentiality in the RO model.*

Proof. Since the proof is similar to that of the insider confidentiality part of Theorem 4 in Section 6 in [37], we ignore the details.

Overview. The proof includes the identity-concealment and the confidentiality of search tokens. According to Theorem 4 in [37], the identity-concealment is implicitly included in the insider confidentiality. Thus, we only prove the insider confidentiality here. The reduction is outlined below. At the beginning, we first choose a random pair (id_{U^*}, id_{V^*}) as the challenge identities, where id_{U^*} and id_{V^*} are used for simulating the challenge data user and server respectively. Then we construct a reduction that reduce the security of the scheme to the GDH assumption. The detailed reduction is omitted here.

Remark 2: In fact, our scheme also achieves adaptive security according to Theorem 3 in Section 5 in [34] under the assumption of the DDH over the cyclic group $\mathbb{G}_2$ and the CPA secure ABE *scheme.*

VI. PERFORMANCE ANALYSIS

Besides our scheme has all advantages as Sun *et al.*'s scheme (short for Sun's scheme), our scheme also strengthens the privacy of the data users which additionally provides identity-concealment, authentication of the data users to the server, and confidentiality of the search token besides hiding the exact queried values from the data owners, (see TABLE 3 for the comparison). Furthermore, in our solusion the ABE scheme only encrypts the document index instead of the document identifier and the retrieval key. Fortunately, this difference does not result in a lack of efficiency and on the contrary, our scheme is more efficient at this point. In TABLE 3, we list the comparison results in security among our scheme and those in [23], [29], [34].

TABLE 3. Security analysis.

reference	identity -concealment	authentication	against replay attacks? (confidentiality of search token)
[23]	×	×	×
[29]	×	×	×
[34]	×	×	×
our scheme	√	√	√

Since our scheme is designed based on Sun's scheme [34], in functionality, like their scheme, our scheme also supports boolean queries, non-interaction, multi-data-user and access control, but beyond that, our solution also supports multi-data-owner functionality. For the sake of intuition, we give a comparison on the functionalities among our scheme and those in [23], [29] [34] in TABLE 4, Note that in this table, the notations "N", "Y" and "−" mean "No", "Yes" and "non-comparability", respectively.

A. EFFICIENCY ANALYSIS

Note that in our and Sun *et al.*'s scheme, the storage costs and computational costs incurred by the ABE scheme are also the same. Furthermore, although one Setup algorithm and two KGC algorithms are additionally included in our scheme, luckily, as each output of the three algorithms for each participant appears only once and forever, it does not affect the total communication and computation costs. Due to one time of the storage capacity for each data owner to the server, so, like Sun's scheme, we only consider the communication overhead and the computation cost between the data owner and the data user, and we only focus on the main communication overheads and omit the less contributed part,

TABLE 4. Functionality analysis.

reference	query-type	multi-user	interaction	access control	multi-data-owner
[23]	boolean	N	-	N	N
[29]	boolean	Y	Y	N	N
[34]	boolean	Y	N	Y	N
our scheme	boolean	Y	N	Y	Y

TABLE 5. Communication overhead between data owner and data user & their computation cost.

reference	comm overhead	data owner's comp cost	data user's comp cost
[23]	-	$\|\mathsf{DB}[w_1]\|.(m-1)$.exp	-
[29]	$(m-1)\mathbb{G}_2$	$(m-1)$.exp	$\|\mathsf{DB}[w_1]\|.(m-1)$.exp
[34]	$3log\|\mathbb{Z}_n^*\|$	3.exp	$(\|\mathsf{DB}[w_1]\|.(m-1)+(m+1))$.exp
our scheme	$3log\|\mathbb{Z}_n^*\|+\|\mathrm{H}\|$	3.exp	$(\|\mathsf{DB}[w_1]\|.(m-1)+(m+3.5))$.exp

such as data generated by attribute-based schemes. We also assume that for each conjunctive query executed by a data user, there are m authorized keywords. For more intuitive, we summarize the total communication overheads between the data owner and the data user and their computation costs for each query in TABLE 5.

- **Communication overheads:** The main communication overheads between the data owner and the data user are generated by SKeyGen. Like Sun's scheme, the size of sk_{W} is $3log|\mathbb{Z}_n^*|$, but our scheme has an additional $id_{\mathsf{EDB}_{DW}}$, which is generated by a hash function H. So the total communication overheads of our scheme is $3log|\mathbb{Z}_n^*|+|\mathrm{H}|$.
- **computation costs:** In Sun's scheme, each data owner needs to compute 3 exponents and an attribute-based secret key, while our scheme requires the same calculation costs as them.
  We conclude that the performances in our and Sun's scheme are almost identical except that in our scheme, the computation cost for each data user is 2.5 exponent operations more than that of Sun's scheme. This extra overhead is mainly resulted by the authentication procedure between data user and server, which is not provided in sun's solution. In additon, since in Zhao's higncryption scheme [37], the calculation of hash function output d in authentication phase is $|q'|/2$ bits, it only takes 0.5 exponent operation to compute $\overline{X}$. In this way a sender only needs 2.5 exponent operations totally to finish the protocol. In particular, our extra overhead is independent from the number of authorized keywords m, the additional cost does not affect the efficiency of the scheme largely.

VII. CONCLUSION

In this article, we propose a new symmetric searchable encryption (SSE) scheme in the multi-data-user and multi-data-owner settings. Compare to Sun *et al.*'s multi-client SSE scheme proposed at ESORICS 2016, our solution not only supports multi-data-owner functionalities, but also further strengthens the securities by implementing extra securities such as identity-concealment, authentication and confidentiality. The final results suggest that our scheme reaches almost the same level of efficiency as Sun *et al.*'s scheme. The next work we will do is how to design symmetric searchable encryption schemes that can resist post-quantum attacks.

APPENDIX

See (Algorithm 4 and Algorithm 5)

REFERENCES

[1] C. Guo, X. Chen, Y. Jie, F. Zhangjie, M. Li, and B. Feng, "Dynamic multi-phrase ranked search over encrypted data with symmetric searchable encryption," *IEEE Trans. Services Comput.*, early access, Oct. 30, 2017, doi: 10.1109/TSC.2017.2768045.

[2] W. Zhang, Y. Lin, S. Xiao, J. Wu, and S. Zhou, "Privacy preserving ranked multi-keyword search for multiple data owners in cloud computing," *IEEE Trans. Comput.*, vol. 65, no. 5, pp. 1566–1577, May 2016, doi: 10.1109/TC.2015.2448099.

[3] C. Guo, W. Liu, X. Liu, and Y. Zhang, "Secure similarity search over encrypted non-uniform datasets," *IEEE Trans. Cloud Comput.*, early access, Jun. 5, 2020, doi: 10.1109/TCC.2020.3000233.

[4] Z. Lv, C. Hong, M. Zhang, and D. Feng, "Expressive and secure searchable encryption in the public key setting," in *Proc. Int. Conf. Inf. Secur.*, 2014, pp. 364–376.

[5] Q. Liu, G. Wang, and J. Wu, "Secure and privacy preserving keyword searching for cloud storage services," *J. Netw. Comput. Appl.*, vol. 35, no. 3, pp. 927–933, May 2012.

[6] F. Zare and H. Mala, "Cryptanalysis of an asymmetric searchable encryption scheme," in *Proc. 14th Int. ISC (Iranian Soc. Cryptol.) Conf. Inf. Secur. Cryptol. (ISCISC)*, Shiraz, Iran, Sep. 2017, pp. 82–85, doi: 10.1109/ISCISC.2017.8488369.

[7] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2007, pp. 321–334.

[8] Y. Zheng, "Digital signcryption or how to achieve cost(signature & encryption) $\ll$ cost(signature) + cost(encryption)," *Proc. CRYPTO*, 1997, pp. 165–179.

[9] A. W. Dent, "Hybrid cryptography," Cryptol. ePrint Arch., Int. Assoc. Cryptol. Res., Tech. Rep. 210, 2004. [Online]. Available: https://eprint.iacr.org/2004/210.pdf

[10] A. Menezes, M. Qu, and S. Vanstone, "Some new key agreement protocols providing implicit authentication," in *Proc. SAC*, 1995, pp. 70–88.

[11] M. Gorantla, C. Boyd, and J. M. G. Nieto, "On the connection between signcryption and one-pass key establishment," in *Cryptography and Coding* (Lecture Notes in Computer Science), vol. 4887, S. D. Galbraith, Ed. Berlin, Germany: Springer, 2007.

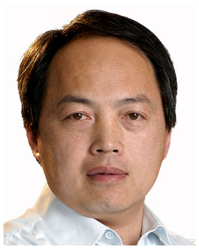
[12] S. Halevi and H. Krawczyk, "One-pass HMQV and asymmetric key-wrapping," in *Proc. PKC*, 2011, pp. 317–334.

[13] H. Krawczyk, "The order of encryption and authentication for protecting communications (or: How secure is SSL?)" in *Proc. CRYPTO*, 2001, pp. 310–331.
[14] J. Baek, R. Steinfeld, and Y. Zheng, "Formal proofs for the security of signcryption," *J. Cryptol.*, vol. 20, no. 2, pp. 203–235, Apr. 2007.
[15] J. Fan, Y. Zheng, and X. Tang, "A single key pair is adequate for the Zheng signcryption," in *Proc. ACISP*, 2011, pp. 371–388.
[16] B. Dan "The decision Dife–Hellman problem," in *Algorithmic Number Theory. ANTS* (Lecture Notes in Computer Science), vol. 1423, J. P. Buhler, Ed. Berlin, Germany: Springer, 1998, doi: 10.1007/BFb0054851.
[17] *The Transport Layer Security (TLS) Protocol Version 1.3*. Accessed: Aug. 2018. [Online]. Available: https://tools.ietf.org/html/rfc8446
[18] C. Brzuska, N. P. Smart, B. Warinschi, and G. J. Watson, "An analysis of the EMV channel establishment protocol," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, 2013, pp. 373–386.
[19] J. Roskind. *Quick UDP Internet Connections: Multiplexed Stream Transport Over UDP*. Accessed: Nov. 2020. [Online]. Available: https://www.chromium.org/quic
[20] *3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Security of the mission critical service; (Release 15)*, 3GPP, document TS 33.180 v15.3.0, 2018.
[21] D. Boneh and B. Waters, "Conjunctive, subset, and range queries on encrypted data," in *Advance TCC*, vol. 4392, S. P. Vadhan, Ed. Berlin, Germany: Springer, 2007, pp. 535–554.
[22] T. Okamoto and D. Pointcheval, "The gap-problems: A new class of problems for the security of cryptographic schemes," in *Public Key Cryptography. PKC* (Lecture Notes in Computer Science), vol. 1992, K. Kim, Ed. Berlin, Germany: Springer, 2001, doi: 10.1007/3-540-44586-2_8.
[23] D. Cash, S. Jarecki, C. Jutla, H. Krawczyk, M. C. Rosu, and M. Steiner, "Highly-scalable searchable symmetric encryption with support for Boolean queries," in *Proc. CRYPTO*, 2013, pp. 353–373.
[24] Y. Chang and M. Mitzenmacher, "Privacy preserving keyword searches on remote encrypted data," in *Proc. ACNS*. Berlin, Germany: Springer, 2005, pp. 442–455.
[25] R. Curtmola, J. A. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: Improved denitions and ecient constructions," in *Proc. CCS*, 2006, pp. 79–88.
[26] E. Goh, "Secure indexes," Int. Assoc. Cryptol. Res. (IACR Cryptol). ePrint Arch., Tech. Rep., 2003. [Online]. Available: https://eprint.iacr.org/2003/216.pdf
[27] P. Golle, J. Staddon, and B. Waters, "Secure conjunctive keyword search over encrypted data," in *Proc. ACNS*, Yellow Mountain, China, 2004, pp. 31–45.
[28] Q. Zheng, S. Xu, and G. Ateniese, "VABKS: Verifiable attribute-based keyword search over outsourced encrypted data," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2014, pp. 522–530.
[29] S. Jarecki, C. Jutla, H. Krawczyk, M. Rosu, and M. Steiner, "Outsourced symmetric private information retrieval," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, 2013, pp. 875–909.
[30] K. Kurosawa and Y. Ohtaki, "UC-secure searchable symmetric encryption," in *Proc. Int. Conf. Financial Cryptogr. Data Secur.*, 2012, pp. 285–298.
[31] V. Rompay, R. M. Cedric, and M. Onen, "A leakage-abuse attack against multi-user searchable encryption," in *Proc. 17th Privacy Enhancing Technol. Symp.*, Minneapolis, MN, USA, 2017, pp. 1–13.
[32] J. Shi, J. Z. Lai, and Y. J. Li, "Authorized keyword search on encrypted data," in *Proc. ESORICS*. Berlin, Germany: Springer, 2014, pp. 419–435.
[33] D. Xiaoding Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proc. IEEE Symp. Secur. Privacy. S&P*, Berkeley, CA, USA, May 2000, pp. 44–55.
[34] S. F. Sun, J. K. Liu, A. Sakzad, R. Steinfeld, and T. H. Yuen, "An efficient non-interactive multi-client searchable encryption with support for Boolean queries," in *Proc. ESORICS*, 2016, pp. 154–172.
[35] W. Sun, S. Yu, W. Lou, Y. T. Hou, and H. Li, "Protecting your right: Attribute-based keyword search with fine-grained owner-enforced search authorization in the cloud," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2014, pp. 226–234.
[36] Z. Wan and R. H. Deng, "VPSearch: Achieving verifiability for privacy-preserving multi-keyword search over encrypted cloud data," *IEEE Trans. Dependable Secure Comput.*, vol. 15, no. 6, pp. 1083–1095, Nov./Dec. 2018.
[37] Y. L. Zhao, "Identity-concealed authenticated encryption and key exchange," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2016, pp. 1464–1479.
[38] L. Du, K. Li, Q. Liu, Z. Wu, and S. Zhang, "Dynamic multi-client searchable symmetric encryption with support for Boolean queries," *Inf. Sci.*, vol. 506, pp. 234–257, Jan. 2020.
[39] M. Zarezadeh, H. Mala, and M. A. Talouki, "Multi-keyword ranked searchable encryption scheme with access control for cloud storage," *Peer–Peer Netw. Appl.*, vol. 13, pp. 207–218, Apr. 2019.
[40] S. K. Kermanshahi, J. K. Liu, R. Steinfeld, S. Nepal, S. Q. Lai, R. Loh, and C. Zuo, "Multi-client cloud-based symmetric searchable encryption," *IEEE Trans. Dependable Secure Comput.*, early access, Nov. 10, 2019, doi: 10.1109/TDSC.2019.2950934.
[41] C. Zuo, S. F. Sun, J. K. Liu, J. Shao, and J. Pieprzyk, "Dynamic searchable symmetric encryption with forward and stronger backward privacy," in *Proc. ESORICS*, 2019, pp. 283–303.
[42] R. Cramer and V. Shoup, "Signature schemes based on the strong RSA assumption," in *Proc. 6th ACM Conf. Comput. Commun. Secur.*, New York, NY, USA, 1999, pp. 46–51, doi: 10.1145/319709.319716.
[43] J. Li, Y. Y. Huang, Y. Wei, S. Y. Lv, Z. L. Liu, C. Y. Dong, and W. J. Lou, "Searchable symmetric encryption with forward search privacy," *IEEE Trans. Dependable Secure Comput.*, early access, Jan. 22, 2019, doi: 10.1109/TDSC.2019.2894411.
[44] J. Chen, Z. Cao, J. Shen, X. Dong, and X. Wang, "Forward secure dynamic searchable symmetric encryption with lighter storage," in *Proc. 4th Int. Conf. Cryptogr., Secur. Privacy*, Jan. 2020, pp. 24–30.
[45] S. K. Pasupuleti, S. Ramalingam, and R. Buyya, "An efficient and secure privacy-preserving approach for outsourced data of resource constrained mobile devices in cloud computing," *J. Netw. Comput. Appl.*, vol. 64, pp. 12–22, Apr. 2016.

HUIGE WANG received the Ph.D. degree from the Shanghai Jiaotong University, China. She is currently an Associate Professor with the Department of Computer, Anhui Science and Technology University, China. She is also a Post-Doctoral Researcher with the School of Computer Science, Fudan University. Her research interests are information security and cryptography.

GUANGYE SUI received the Ph.D. degree from Laval University, Canada, in 2015. He is currently a Researcher with the School of Computer Science, Fudan University, China. His research interests are cryptography, formal methods, and blockchain technology.

YUNLEI ZHAO received the Ph.D. degree from Fudan University, Shanghai, China. He is currently a Professor with the School of Computer Science, Fudan University. His research interest include in theory and applications of cryptography.



KEFEI CHEN received the B.S. and M.S. degrees in applied mathematics from Xidian University, Xian, in 1982 and 1985, respectively, and the Ph.D. degree from Justus-Liebig University Giessen, Germany, in 1994. From 1996 to 2012, he was a Professor with the Department of Computer Science and Engineering, Shanghai Jiaotong University. He is currently a Professor with the School of Science, Hangzhou Normal University. His main research areas are cryptography and theory and technology of network security.