Information Sciences 506 (2020) 234–257

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Dynamic multi-client searchable symmetric encryption with support for boolean queries

Leilei Du[a], Kenli Li[a,*], Qin Liu[a,c,*], Zhiqiang Wu[a], Shaobo Zhang[b]

[a] College of Computer Science and Electronic Engineering, Hunan University, Changsha, Hunan Province 410082, PR China
[b] School of Computer Science and Engineering, Hunan University of Science and Technology, Xiangtan, Hunan Province 411201, PR China
[c] State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, Jiangsu Province 210093, PR China

ARTICLE INFO

Article history:
Received 15 January 2019
Revised 27 July 2019
Accepted 3 August 2019
Available online 6 August 2019

Keywords:
Cloud computing
Multi-client searchable symmetric encryption
Boolean query
Non-interactivity
Dynamic

ABSTRACT

With the rapid growth of cloud computing, an increasing amount of data is being outsourced to cloud servers, in the meantime, how to search data securely and efficiently has got an unprecedented concern. Searchable symmetric encryption (SSE) that enables keyword-based searches over encrypted data provides an efficient way to this problem. However, the majority of existing SSE schemes focus on single keyword searches in the single-client setting, which limits their wide application in cloud computing. In this paper, we propose a Dynamic Multi-client SSE (DMSSE) scheme with support for boolean queries, by incorporating a client's authorization information into search tokens and indexes. Our scheme allows a data owner to authorize multiple clients to perform boolean queries over an encrypted database, and limits a client's search ability to legitimate keywords. Compared with existing MSSE schemes, our DMSSE scheme has the following merits: 1) *Non-interactivity*. After the grant of search permission, the clients can perform queries on their own without the help of the data owner. 2) *Dynamic*. The data owner can efficiently update a client's search permission without affecting other clients. Experimental evaluations conducted on a real data set demonstrate that our DMSSE scheme is practical for use in a large-scale encrypted database.

1. Introduction

Cloud computing that offers ubiquitous and on-demand services over Internet has recently become an indispensably leading computing paradigm [20]. Faced with the benefits of high flexibility, low cost, and great scalability, data owners become increasingly motivated to migrate massive data from local storage to cloud servers [26]. Unfortunately, with the wide use of cloud servers as storage platforms, we have also witnessed frequent data leakages recently, ranging from personal medical records to personal privacy data on social platforms [21,23,33–36]. Now, data owners are stuck in a dilemma. They want to enjoy the great advantages of cloud computing, yet fear using cloud services which exposes them to possible attacks and data breaches. The solution to this contradiction is letting data owners encrypt sensitive data before outsourcing [13,22].

* Corresponding authors at: 310 Software building, Hunan University, Changsha, Hunan provice, China (Kenli Li); 436 Software building, Hunan University, Changsha, Hunan provice, China (Qin Liu).
E-mail addresses: lkl@hnu.edu.cn (K. Li), gracelq628@hnu.edu.cn (Q. Liu).

https://doi.org/10.1016/j.ins.2019.08.014
0020-0255/

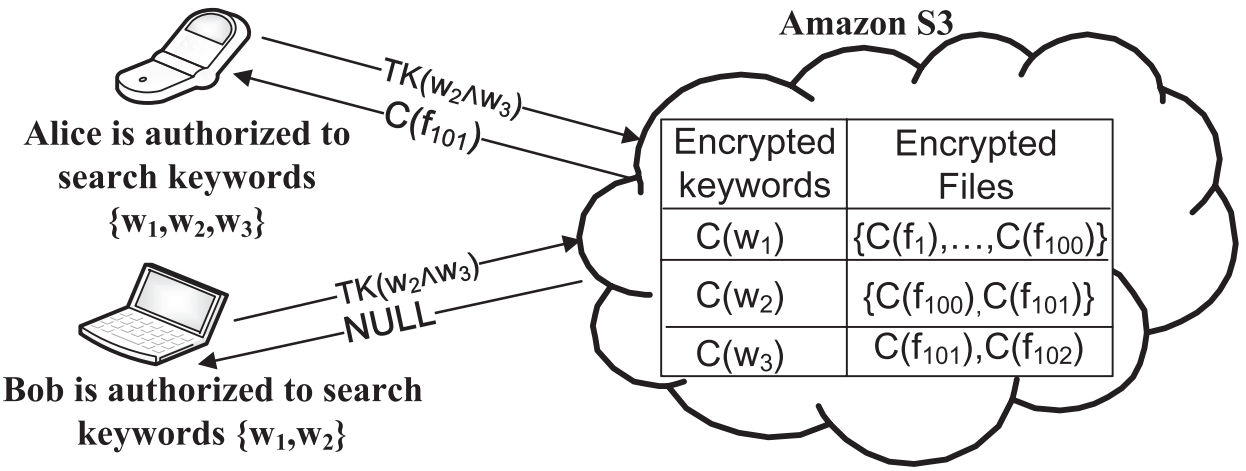

Fig. 1. Application scenario. An encrypted file system is outsourced to the cloud server, where $C(w_i)$ and $C(f_j)$ denote the encrypted versions of keyword w_i and file f_j, respectively. The search token $TK(w_2 \wedge w_3)$ as an encrypted version of query $Q = w_2 \wedge w_3$ can be used to retrieve matched files only when both w_2 and w_3 are legitimate keywords for the inquirer.

Although data encryption provides an effective security guarantee, it limits data utilization making certain ordinary practices like keyword-based searches a challenging problem.

Let us consider the following application as shown in Fig. 1. Company A as the data owner outsources its encrypted file system to Amazon S3, and authorizes its staff as clients to perform various keyword-based queries to retrieve matched files. According to post, each staff is associated with a set of legitimate keywords. For example, Alice as the manager of Company A is qualified to search all keywords including confidential keywords (e.g., w_3), but Bob as an ordinary employee is allowed to search only non-confidential keywords (e.g., w_1, w_2). For data privacy, each staff is only allowed to search the keywords within permission. For example, if Bob issues a search token $TK(w_2 \wedge w_3)$, the cloud server returns nothing because keyword w_3 is beyond the search permission of Bob.

In the above application scenario, the adopted encryption scheme should satisfy the following requirements: **(1) Supporting efficient and secure boolean queries.** A client is able to issue arbitrary boolean queries over keywords to retrieve matched files in a private and practical way. Given a set of keywords, a naive solution is letting the cloud server query the database with each individual keyword and calculate a boolean function on the resultant file sets. However, this solution falls short of efficiency or security. First, the cloud server needs to execute searches multiple times rendering the search cost to grow linearly with the number of keywords in the query. Second, the cloud server will know which files match each keyword, even if only few files match the query. For example, for query $Q = w_1 \wedge w_2$, this solution enables the cloud server to know files $f_1, \ldots, f_{100}$ match keyword w_1 and files f_{100}, f_{101} match keyword w_2, although only file f_{100} matches query Q. **(2) Supporting authorized searches in the multi-client setting.** A client is associated with a set of legitimate keywords and can search only the keywords within his permission. For a query containing an illegal keyword w, no results will be returned even if there exists a file matching w. Note that this property is virtual for data privacy. For example, the collaboration agreement, f_{101}, described with non-confidential keyword w_2 ="Company B" and confidential keyword w_3 ="Project X" can be accessed only by Alice. If Bob can retrieve $C(f_{101})$ by querying with $Q = w_2 \wedge w_3$, he can infer that Company A is cooperating with Company B on Project X from the search results returned, without needing to decrypt $C(f_{101})$.

Searchable symmetric encryption (SSE) [7,10,29] offers a potential solution to the above scenario by providing a way for the encrypted data to be searched efficiently and securely. In SSE, a data owner outsources an encrypted database to an untrusted server such that she can later query the server to retrieve matched ciphertexts while hiding information about the database and queries from the server. By using an inverted-index structure, SSE allows keyword-based searches to be performed in sublinear and also optimal time. While efficient and offering good privacy, most of SSE schemes focus on single-keyword searches in the single-client setting. The first practical SSE scheme with support for boolean queries, named the OXT scheme, is proposed by Cash et al. [6], where a set of oblivious cross-tags are constructed to achieve sublinear search time and minimal privacy leakage. Based on their work, Jarecki et al. [12] and Sun et al. [30] extended the OXT scheme to multi-client setting by expressing keywords as attribute-values pairs. However, existing Multi-client SSE (MSSE) schemes supporting boolean queries are either interactive (the data owner is required to stay online to generate search tokens for clients) or hard to support *dynamic search permission* (the update of one client's search permission will render a set of relevant clients to update their current secret keys). Therefore, how to design an effective SSE scheme satisfying all our requirements is still an open problem.

Motivated by the OXT scheme of Cash et al. [6], this paper proposes a Dynamic Multi-client Searchable Symmetric Encryption (DMSSE) scheme to achieve authorized boolean queries in a non-interactive and dynamic way. Our scheme allows a data owner to authorize multiple clients to perform boolean queries over an encrypted database, and limits a client's search

ability to legitimate keywords. The main idea of our DMSSE scheme is to incorporate a client's authorization information into search tokens and indexes, such that the client can retrieve files matching a boolean query only when those files contain keywords making the output of the boolean function be true and all keywords in the query are within his permission. The main contributions of this paper are summarized as follows:

- To the best of our knowledge, it is the first attempt to devise a non-interactive and dynamic MSSE scheme with support for boolean queries in cloud computing. Based on the construction of the OXT scheme, our DMSSE scheme can achieve sublinear search time while leaking minimal information to cloud servers.
- Compare with existing MSSE schemes, our DMSSE scheme has the following merits: 1) Non-interactivity. After authorization, the clients can perform queries on their own without the help of the data owner. 2) Dynamic. It allows the data owner to efficiently update a client's search permission without affecting other clients.
- Our scheme is proven to be secure against non-adaptive servers and malicious clients. A comprehensive analysis of the efficiency is given, and experiments on a real-world data set further verify the feasibility and practicability of our DMSSE scheme.

Paper organization. We introduce the related work in Section 2 and provide the preliminaries in Section 3. After the overview of this work in Section 4, we construct our DMSSE scheme in Section 5. We analyze and evaluate its performance in Section 6 before concluding this paper in Section 7. Finally, we provide formal security proof in Appendices.

2. Related work

Our work focuses on achieving secure and efficient boolean queries in the multi-client environment. SSE that allows an untrusted server to perform query over encrypted data can partially address our requirements as follows.

Song et al. [29] presented the first SSE scheme, where both the query and the data were encrypted under a symmetric key. The main drawback of this approach is that the search cost grows linearly with the size of the database. To improve the query efficiency, Goh [10] developed a secure searchable index scheme based on Bloom filters. As a seminal work, Curtmola et al. [7] provided rigorous security definition for SSE, and constructed an inverted index to achieve optimal search time.

The initial SSE schemes mainly focus on single-keyword searches. To enrich the search functionality, Golle et al. [11] proposed a secure SSE scheme with support for conjunctive keyword searches. Followed by their work, Ballard et al. [1] and Jin et al. [5] provided improved conjunctive SSE schemes. The main drawback of these schemes is that their search cost grows linear with database size and thus lack scalability. The first practical SSE scheme with support for boolean queries was proposed by Cash et al. [6], where an OXT scheme was constructed to support general boolean queries over super-large scale encrypted databases. Specifically, in the OXT scheme, a TSet is constructed as an inverted index for sublinear search time, and a set of oblivious cross tags are utilized to provide minimal privacy leakage. Based on their work, Lai et al. [17] proposed the HXT scheme, which achieved a higher level of security than OXT by hiding keyword pair result patterns. Pappas et al. [27] proposed a scalable private DBMS, Blind Seer, based on a Bloom filter tree index to support rich query functionalities. However, their scheme cannot resist the attack from a malicious client. In order to eliminate this deficiency, Fisch et al. [9] improved the construction of Blind Seer to achieve robust access control and data protection. Kamara et al. [14] proposed an IEX scheme to support disjunctive queries with an optimal communication complexity. Their scheme can also handle boolean queries via conjunctive normal form (CNF) in the sacrifice of security. However, the above SSE schemes with rich query functionalities only support the single-client setting.

The concept of Multi-client SSE (MSSE) was introduced by Curtmola et al. [7]. Bao et al. [2] constructed a dynamic MSSE scheme with support for user revocation. Tang [31] presented a MSSE scheme considering different server-client collusion possibilities, but the authorization was granted at the index level. Kasten et al. [15] proposed a MSSE scheme supporting search over an encrypted graph. However, their scheme requires to perform encryption multiple times for building a secure index, resulting in a large computation cost on the data owner.

The first MSSE scheme with support for boolean queries was proposed by Jarecki et al. [12]. Their scheme extends the OXT scheme to the multi-client setting by the utilization of homomorphic signature and oblivious pseudorandom functions (PRFs), which can resist malicious clients and achieve efficient retrieval. Inspired by their work, Faber et al. [8] presented an MSSE scheme that supports rich query types. However, both of the above schemes request clients to interact with the data owner during each query. Sun et al. [30] proposed a non-interactive MSSE scheme based on keyword-to-prime hash functions and employed attribute-based encryption to achieve fine-grained access control. However, their scheme is hard to support dynamic search permission. Li et al. [19] presented an adaptively secure MSSE scheme based on an indistinguishable binary tree. Their scheme can achieve a low storage and fast retrieval, but it only supports conjunctive query. Wang et al. [32] and Liu et al. [25] proposed MSSE schemes under the malicious server model. In Ref. [32], the proposed scheme allows clients to verify the correctness of search results based on accumulator, but it only supports conjunctive keyword search. In Ref. [25], a two-keyword index is constructed to reduce searching time, however, the upper limit of client number needs to be pre-determined in the initial phase. Another line of work with respect to rich query functionality in the multi-client setting can be found in authorized searches in public key searchable encryption [18,28,37]. The comparison results between our DMSSE scheme and previous work are shown in Table 1.

Table 1
Comparison with previous work.

	Query type	Multi-client	Dynamic
Golle et al. [11]	conj	×	×
Ballard et al. [1]	conj	×	×
Jin et al. [5]	conj	×	×
Cash et al. [6]	conj/bool	×	×
Lai et al. [17]	conj/bool	×	×
Pappas et al. [27]	conj/bool	×	×
Fisch et al. [9]	conj/bool	×	×
Kamara et al. [14]	disj/bool	×	√
Curtmola et al. [7]	w	√	×
Bao et al. [2]	w	√	√
Tang [31]	w	√	×
Kasten et al. [15]	w	√	√
Jarecki et al. [12]	conj/bool	√	×
Faber et al. [8]	conj/bool	√	√
Sun et al. [30]	conj/bool	√	×
Li et al. [19]	conj	√	×
Wang et al. [32]	conj	√	×
Liu et al. [25]	bool	√	√
Our DMSSE scheme	conj/bool	√	√

w: single-keyword; conj: conjunctive-keyword; disj: disjunctive-keyword; bool: boolean-keyword.

3. Preliminaries

This section first introduces our models, and then describes related notations, hardness assumptions, and cryptographic preliminaries.

3.1. System and adversary models

As illustrated in Fig. 2, our system model consists of three different parties: the data owner $\mathcal{D}$, the cloud server $\mathcal{S}$ and multiple clients $\mathcal{C}s$. $\mathcal{D}$ possesses a proprietary database DB, and decides to outsource the encrypted database EDB to $\mathcal{S}$ for resilient services and economic savings. To achieve fine-grained query control, she is responsible for the authorization and update of search permission for each client. After obtaining an authorization certificate Ω_i from $\mathcal{D}$, a client $\mathcal{C}_i$ generates a search token $TK_{i,Q}$ for a boolean query Q and sends it to the cloud server $\mathcal{S}$. Once receiving a search request, $\mathcal{S}$ evaluates the search token $TK_{i,Q}$ on the encrypted database EDB and returns search results **R**. To support dynamic search permission, $\mathcal{S}$ also follows $\mathcal{D}$'s instruction to update EDB appropriately.

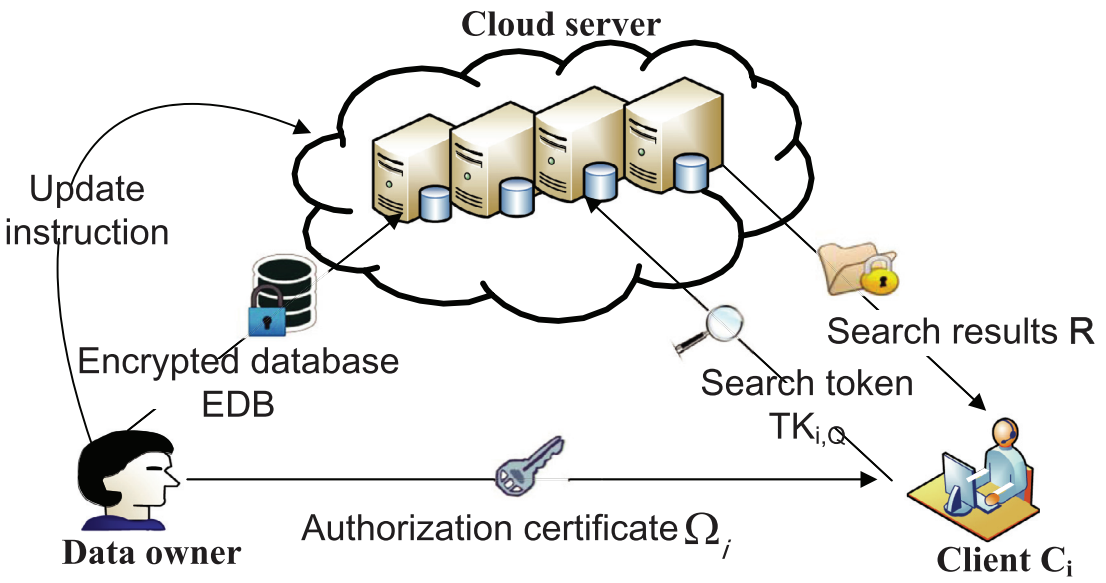

Fig. 2. System model. The authorization certificate will be transmitted through secure channels once mutual authentication has finished.

Table 2
Summary of notations.

Notation	Description
Γ	Data owner $\mathcal{D}$'s master secret key
Δ	System parameter
KAL	Keyword access list maintained by data owner $\mathcal{D}$
DB	A list of d identifier-keywords pairs $(\text{ind}_i, \text{W}_i)_{i=1}^{d}$
W	A list of m keywords $(w_1, \ldots, w_m)$
W_i	A set of keywords associated with file f_i
$\text{DB}(w)$	A set of identifiers of files containing keyword w
$\text{DB}(Q)$	A set of identifiers of files matching query Q
$\overline{\text{W}}_i$	A set of legitimate keywords for client $\mathcal{C}_i$
Ω_i	An authorization certificate of client $\mathcal{C}_i$
$TK_{i,Q}$	A search token of query Q generated by client $\mathcal{C}_i$
R	Search results returned by server $\mathcal{S}$

In our adversary model, the data owner $\mathcal{D}$ is assumed to be fully trusted, and the clients $\mathcal{C}s$ and the cloud server $\mathcal{S}$ are assumed to be potential adversary. Specifically, malicious $\mathcal{C}s$ will query with keywords outside their permission and try to gain information beyond what they are authorized to access. $\mathcal{S}$ is assumed to be *honest but curious*. That is, it will always correctly execute a given protocol, but may try to learn some additional information about the stored data and the queried keywords. We assume that multiple clients acts maliciously and possibly collude with each other, but a honest-but-curious server does not collude with clients.

3.2. Notations

Let $\lambda \in \mathbb{N}$ be a secure parameter of the whole system. For $t \in \mathbb{N}$, notation $[t_1, t_2]$ is used to denote the set of integers in $\{t_1, \ldots, t_2\}$, which can be abbreviated as $[t_2]$ when $t_1 = 1$. The set of all binary strings of length t is denoted by $\{0, 1\}^t$ and the set of finite binary strings is denoted by $\{0, 1\}^*$. Given a sequence of elements $\mathbf{V}$, its i-th element is denoted by $\mathbf{V}[i]$. If S is a set then $|S|$ refers to its cardinality. If s is a string then $|s|$ refers to its bit length. The concatenation of two strings s_1 and s_2 is denoted by $s_1||s_2$. We write $x \xleftarrow{\$} X$ to represent an element x being sampled uniformly at random from a set X. The output x of an algorithm A is denoted by $x \leftarrow A$.

A database $\text{DB} = (\text{ind}_i, \text{W}_i)_{i=1}^{d}$ is a list of d identifiers and keywords pairs, where $\text{ind}_i \in \{0, 1\}^{\lambda}$ is a file identifier and $\text{W}_i \subseteq \{0, 1\}^*$ is a set of keywords in file f_i. The universal keywords is set as $\text{W} = \cup_{i=1}^{d} \text{W}_i$ where $|\text{W}| = m$. A query Q is expressed as $\psi(\overline{\mathbf{w}})$, where ψ is a boolean formula and $\overline{\mathbf{w}} \in \text{W}$ is a set of querying keywords. We write $\text{DB}(Q)$ for the set of identifiers of files that satisfy query $Q = \psi(\overline{\mathbf{w}})$. This means that $\text{ind}_i \in \text{DB}(\psi(\overline{\mathbf{w}}))$ if the formula $\psi(\overline{\mathbf{w}})$ evaluates to be true when we replace each keyword $w \in \overline{\mathbf{w}}$ with true or false depending on if $w \in \text{W}_i$ or not. We write $\text{DB}(w) = \{\text{ind}_i, s.t. w \in \text{W}_i\}$ and denote the size of $\text{DB}(w)$, i.e., $|\text{DB}(w)|$, by $\#w$. We assume that the file identifier is independent of the file content, and thus is allowed to be exposed to the cloud server. For quick reference, the most relevant notations used in our DMSSE scheme are shown in Table 2.

3.3. Pairing and hardness assumptions

Definition 1 (Bilinear pairing)**.** Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two cyclic groups of prime order p. A bilinear pairing $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ satisfies the following properties: 1) *Bilinearity*: $\hat{e}(g_1^a, g_2^b) = \hat{e}(g_1, g_2)^{ab}$ for all $g_1 \in \mathbb{G}_1$, $g_2 \in \mathbb{G}_2$ and $a, b \in \mathbb{Z}_p^*$. 2) *Non-degeneracy*: $\hat{e}(g_1, g_2) \neq 1$. 3) *Computability*: There exists an efficient algorithm to compute $\hat{e}(g_1, g_2)$ for all $g_1 \in \mathbb{G}_1$, $g_2 \in \mathbb{G}_2$.

As defined in [4], $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a type-III bilinear pairing if there is no efficiently computable isomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$.

Definition 2 (Type III pairing parameter generator)**.** A type III pairing parameter generator is an algorithm Γ that takes as input a security parameter λ and outputs $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, g_1, g_2)$, where $\mathbb{G}_1$ and $\mathbb{G}_2$ are groups of prime-order p, g_1 and g_2 are a random generator of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively, and $\hat{e}$ is a type III bilinear pairing satisfying $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$.

Definition 3. (DDH problem). Let $\mathbb{G}$ be a cyclic group of prime order p, the decisional Diffie-Hellman(DDH) problem is to distinguish between the distribution of (g, g^a, g^b, g^{ab}) and (g, g^a, g^b, g^z), where $g \in \mathbb{G}$ and $a, b, z \in \mathbb{Z}_p^*$ are chosen uniformly at random. For any probabilistic polynomial time(PPT) adversary A, its advantage is define as: $\text{Adv}_{A,\mathbb{G}}^{\text{DDH}}(\lambda) = |\Pr[A(g, g^a, g^b, g^{ab}) = 1] - \Pr[A(g, g^a, g^b, g^z) = 1]|$.

Definition 4. (SAXDH) problem. Let $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ be three cyclic groups whose order is prime p, with and $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be type-III bilinear pairing. Given random generator $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$, the symmetric augmented external Diffie-Hellman(SAXDH) problem is to distinguish between the distribution of $(g_2^{xs_1}, g_2^{s_1}, g_2^{x}, g_2^{s_2}, g_1^{J/s_1}, g_1^{J/s_2}, g_1^{J}, g_1^{J/s_2})$ and

$(g_2^y, g_2^{s_1}, g_2^x, g_2^{s_2}, g_1^{J/s_1}, g_1^{J/s_2}, g_1^J, g_1^{J/s_2})$, where x, s_1, s_2, I and J are chosen randomly from $\mathbb{Z}_p^*$. For any PPT adversary A, its advantage is define as: $\mathrm{Adv}^{\mathrm{SAXDH}}_{\mathbb{G}_1,\mathbb{G}_2,A}(\lambda) = |\Pr[A(g_2^{xs_1}, g_2^{s_1}, g_2^x, g_2^{s_2}, g_1^{J/s_1}, g_1^{J/s_2}, g_1^J, g_1^{J/s_2}) = 1] - \Pr[A(g_2^y, g_2^{s_1}, g_2^x, g_2^{s_2}, g_1^{J/s_1}, g_1^{J/s_2}, g_1^J, g_1^{J/s_2}) = 1]|$.

The (symmetric) SAXDH assumption also requires that a similar assumption hold when the roles of $\mathbb{G}_1$ and $\mathbb{G}_2$ (i.e. g_1 and g_2) are reversed.

3.4. Cryptographic preliminaries

In our DMSSE scheme, symmetric key encryption (SKE) [3] is employed to encrypt file identifiers, pseudorandom functions (PRFs) [16] are used to generate random factors, and the OXT scheme [6] of Cash et al. is used as a building block to construct our searchable indexes.

Symmetric key encryption. SKE consists of three polynomial-time algorithms $\Sigma = (\mathsf{Enc},\mathsf{Dec})$, where the encryption algorithm Enc takes a key k_e and a file identifier ind as its input and returns a ciphertext **e**. The decryption algorithm Dec takes the key k_e and a ciphertext **e** as its input, and returns the file identifier. SKE is secure under chosen-plaintext attacks (CPA-secure). That is, the ciphertexts do not reveal any information about the plaintexts to the adversary that can adaptively query an encryption oracle.

Pseudorandom function. Let F: $\{0,1\}^\lambda \times X \to Y$ be a function defined from $\{0,1\}^\lambda \times X \to Y$. We say F is a PRF if for a probabilistic polynomial time (PPT) adversary A, its advantage $\mathrm{Adv}^{\mathrm{prf}}_{A,F} = |\Pr[A^{F(K,\cdot)}(1^\lambda)] - \Pr[A^{f(\cdot)}(1^\lambda)]|$ is negligible in λ, where $K \xleftarrow{\$} \{0,1\}^\lambda$ and f is a random function from X to Y.

The OXT scheme. Let $\Sigma = (\mathsf{Enc},\mathsf{Dec})$ be a SKE scheme defined as above, and let $\mathbb{G}$ be a cyclic group of prime order p where $g \in \mathbb{G}$ is a random generator. The OXT scheme that allows arbitrary boolean queries over encrypted data consists of the following algorithm:

- $\mathbf{K} \leftarrow \mathsf{KeyGen}(\lambda)$: This algorithm takes the security parameter λ as input and outputs the secret key $\mathbf{K} = \{K_X, K_I, K_Z, K_S, K_T\}$, where K_X, K_I, K_Z and K_S, K_T are secret keys for PRF F_p (with range in $\mathbb{Z}_p^*$) and PRF F, respectively.
- $\mathsf{EDB} \leftarrow \mathsf{EDBSetup}(\mathsf{DB}, \mathbf{K})$: This algorithm first initializes the TSet **T** to empty array indexed by keywords from W and initializes the XSet **X** to an empty set. For each $w \in \mathsf{W}$, it performs as follows:
  - Initialize **t** to be an empty list and initialize a counter c to 0.
  - Set $K_e \leftarrow F(K_S, w)$ and for each $\mathrm{ind} \in \mathsf{DB}(w)$:
    - Set $\mathrm{xind} \leftarrow F_p(K_I, \mathrm{ind})$, $z \leftarrow F_p(K_Z, w||c)$, $y \leftarrow \mathrm{xind} \cdot z^{-1}$, and $\mathbf{e} \leftarrow \mathsf{Enc}(K_e, \mathrm{ind})$.
    - Append $(y, \mathbf{e})$ to **t** and set $c \leftarrow c + 1$.
    - Set $\mathsf{xtag} \leftarrow g^{F_p(K_X,w)\cdot \mathrm{xind}}$ and add xtag to **X**.
  - Set $\mathsf{stag} \leftarrow F(K_T, w)$ and $\mathbf{T}[\mathsf{stag}] \leftarrow \mathbf{t}$[1]

  The encrypted database is set as $\mathsf{EDB} = (\mathbf{T}, \mathbf{X})$.
- $TK_Q \leftarrow \mathsf{TokenGen}(\mathbf{K}, Q)$: Given query $Q = \psi(\overline{\mathbf{w}})$, where $\overline{\mathbf{w}} = (\overline{w}_1, \ldots, \overline{w}_n)$, this algorithm first chooses the keyword with lowest-frequency from $\overline{\mathbf{w}}$ as the sterm and sets the remaining keywords in $\overline{\mathbf{w}}$ as xterms. Suppose that $\overline{w}_1$ is chosen as the sterm. It then generates the search token $TK_Q = (\mathsf{stag}, \mathsf{xtoken}[2], \ldots, \mathsf{xtoken}[n])$ as follows:
- Set $\mathsf{stag} \leftarrow F_{K_T,\overline{w}_1}$. For $c = 1, 2, \ldots$ and until the server sends *stop*
  - For $i = 2, \ldots, n$, set $\mathsf{xtoken}[c,i] \leftarrow g^{F_p(K_Z,\overline{w}_1||c)\cdot F_p(K_X,\overline{w}_i)}$
  - Set $\mathsf{xtoken}[c] \leftarrow \mathsf{xtoken}[c,2], \ldots, \mathsf{xtoken}[c,n]$.

  Here, the stag and xtoken can be regarded as the encrypted versions of the sterm and xterm, respectively.
- $\mathbf{R} \leftarrow \mathsf{Search}(TK_Q, \mathsf{EDB})$: Assume that ψ is a conjunctive function. This algorithm first sets $\mathbf{t} \leftarrow \mathbf{T}[\mathsf{stag}]$. For each $(\mathbf{e}, y)$ pairs in **t**, it tests whether $\forall i = 2, \ldots, n$, s.t. $\mathrm{xtoken}[c,i]^y \in \mathbf{X}$. If so, it adds **e** into **R**. When the last tuple in **t** is reached, it sends *stop* to the client and halts, and returns **R** as search results. The implementation process of boolean queries is more complicated and will be discussed in Section 5.4.

  By utilization of oblivious cross-tags (xtag can be obtained from y and xtoken while leaking the minimal information), the OXT scheme is proven to be secure against adaptive server based on DDH assumption. Furthermore, the TSet is constructed as inverted index achieving sublinear search time.

4. Overview

This section introduces related definitions, and outlines the working process of our DMSSE scheme.

4.1. Keyword access list (KAL)

In our system, each client $\mathcal{C}_i$ is associated with a public/private key pair (pk_i, sk_i), where $sk_i \in \mathbb{Z}_p^*$ and $pk_i = g_1^{sk_i} \in \mathbb{G}_1$. Once joining the system, $\mathcal{C}_i$ will broadcast its public key so that everyone knows its existence. Suppose that $\mathcal{C}_i$ is entitled to search a set of keywords $\overline{\mathsf{W}}_i \subseteq \mathsf{W}$. Before querying, $\mathcal{C}_i$ should obtain an authorization certificate Ω_i from the data owner

[1] The construction of a secure TSet is simplified for ease of illustration. We refer readers to Ref [6] for detailed construction.

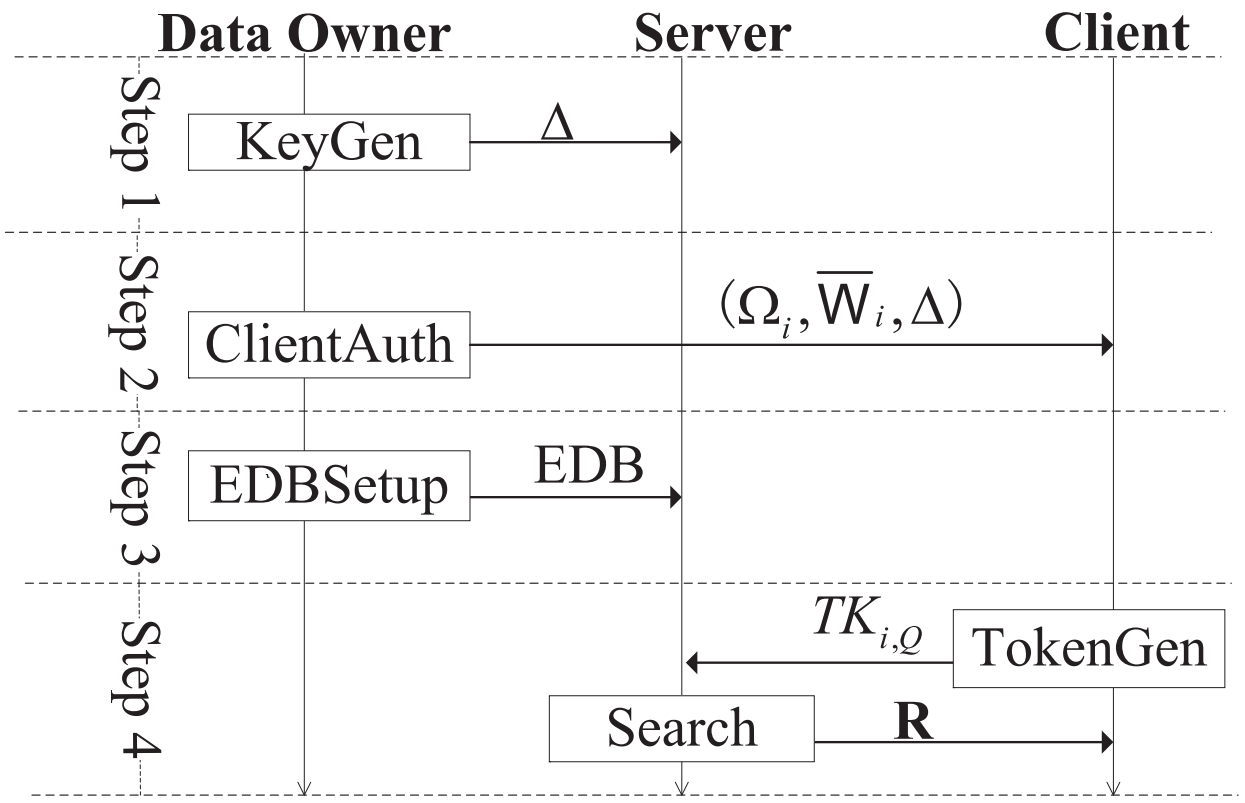

Fig. 3. The working process of DMSSE scheme.

$\mathcal{D}$. Furthermore, to enable $\mathcal{C}_i$ to correctly select a sterm for its query, $\mathcal{D}$ will send back $\overline{\mathsf{W}}_i = (\overline{w}_1, \ldots, \overline{w}_n)$ together with the authorization certificate Ω_i where the keyword appearance frequency satisfies $|\mathsf{DB}(\overline{w}_1)| \le |\mathsf{DB}(\overline{w}_2)| \le \ldots \le |\mathsf{DB}(\overline{w}_n)|$.

The keyword access list KAL is structured as a dictionary indexed by keywords and is kept by the data owner $\mathcal{D}$. For each keyword $w \in \mathsf{W}$, $\mathsf{KAL}[w]$ stores PK_w, a list of public keys of clients authorized to search keyword w. KAL is initialized as empty, and will be dynamically updated once a client's search permission is changed. For example, if client $\mathcal{C}_i$ is authorized to search keyword w, its public key pk_i will be appended at the end of PK_w.

4.2. Scheme definition

As shown in Fig. 3, our scheme consists of the following algorithms:

- $(\Delta, \Upsilon) \leftarrow \mathsf{KeyGen}(1^\lambda)$: The data owner $\mathcal{D}$ takes the security parameter λ as input and outputs the system parameter Δ and master secret key Υ.
- $(\mathsf{KAL}', \Omega_i) \leftarrow \mathsf{ClientAuth}(\Delta, \Upsilon, pk_i, \overline{\mathsf{W}}_i, \mathsf{KAL})$: To authorize client $\mathcal{C}_i$ to search over keywords $\overline{\mathsf{W}}_i$, the data owner $\mathcal{D}$ takes the system parameter Δ, her master secret key Υ, $\mathcal{C}_i$'s public key pk_i, a set of keyword $\overline{\mathsf{W}}_i$, and the current keyword access list KAL as input, and outputs an updated keyword access list KAL' and an authorization certificate Ω_i.
- $\mathsf{EDB} \leftarrow \mathsf{EDBSetup}(\mathsf{DB}, \Delta, \Upsilon, \mathsf{KAL})$: The data owner $\mathcal{D}$ takes the database DB, the system parameter Δ, her master secret key Υ, and the current keyword access list KAL as input, and outputs the encrypted database EDB.
- $TK_{i,Q} \leftarrow \mathsf{TokenGen}(Q, sk_i, \Omega_i)$: The client $\mathcal{C}_i$ takes his authorization certificate Ω_i and his private key sk_i as input to generate a search token $TK_{i,Q}$ for query Q.
- $\mathbf{R} \leftarrow \mathsf{Search}(\mathsf{EDB}, TK_{i,Q})$: The server $\mathcal{S}$ takes the encrypted database EDB and the search token $TK_{i,Q}$ as input and outputs the search results **R**.

From the systematic point of view, our scheme works as follows:

1. Initialization. The data owner $\mathcal{D}$ runs the KeyGen algorithm to generate the system parameter Δ and the master secret key Υ. She then shares Δ with all participants in the system, but keeps Υ private.

2. User grant. When a new client $\mathcal{C}_i$ joins the system, the data owner $\mathcal{D}$ first determines $\overline{\mathsf{W}}_i$ that stipulates the search permission of $\mathcal{C}_i$. She then invokes the ClientAuth algorithm to generate the updated keyword access list KAL' and the authorization certificate Ω_i. Finally, she sends $\overline{\mathsf{W}}_i$ and Ω_i along with the system parameter to $\mathcal{C}_i$.

3. Data creation. The data owner $\mathcal{D}$ executes algorithm EDBSetup to create an encrypted database EDB, which will be outsourced to the server $\mathcal{S}$.

4. Data access. If client $\mathcal{C}_i$ wants to retrieve files matching a boolean query Q, he executes the TokenGen algorithm to generate a search token $TK_{i,Q}$, which will be sent to the server $\mathcal{S}$. On receiving the data access request, $\mathcal{S}$ invokes the Search algorithm to get the result $\mathbf{R}$, and then sends $\mathbf{R}$ back to $\mathcal{C}_i$. Suppose that the client $\mathcal{C}_i$ is authorized to search over keywords W_i. For a query $Q = \varphi(\overline{\mathbf{w}})$ issued by client $\mathcal{C}_i$, our DMSSE scheme ensures that for each file identifier $\mathrm{ind} \in \mathbf{R}$, it satisfies: $\mathrm{ind} \in \mathsf{DB}(\varphi(\overline{\mathbf{w}}))$ and $\mathbf{w} \subseteq \mathsf{W}_i$.

4.3. Security definition

We consider the security against the adversarial server and malicious clients, separately.

Security against adversarial server. In this paper, we only consider the non-adaptive server where the challenging queries are predetermined before simulation. Let $\Pi = (\mathsf{KeyGen}, \mathsf{ClientAuth}, \mathsf{EDBSetup}, \mathsf{TokenGen}, \mathsf{Search})$ be our DMSSE scheme and let $\mathcal{L}$ be an algorithm. For efficient algorithms Adv and Sim, we define experiments $\mathsf{Real}^{\Pi}_{\mathsf{Adv}}(\lambda)$ and $\mathsf{Ideal}^{\Pi}_{\mathsf{Adv,Sim}}(\lambda)$ as follows:

- $\mathsf{Real}^{\Pi}_{\mathsf{Adv}}(\lambda)$: $\mathsf{Adv}(\lambda)$ chooses a database DB and a list of T queries $\mathbf{q}$. The experiment then runs algorithms KeyGen and EDBSetup to obtain $(\mathsf{EDB}, \Delta, \Upsilon)$. For each $t \in [T]$, it runs algorithms ClientAuth, TokenGen and Search and stores the transcript and the client's output in $\mathbf{t}[t]$. Finally, the game gives EDB and $\mathbf{t}$ to Adv, which returns a bit as the game's output.
- $\mathsf{Ideal}^{\Pi}_{\mathsf{Adv,Sim}}(\lambda)$: $\mathsf{Adv}(\lambda)$ chooses a database DB and a list of T queries $\mathbf{q}$. The experiment then runs $\mathsf{Sim}(\mathcal{L}(\mathsf{DB}, \mathbf{q}))$ and gives its output to Adv, which returns a bit as the game's output.

Definition 5. A function f is negligible if for every polynomial $p(\cdot)$, there exists an N such that for all integers $n > N$ it holds that $f(n) < \frac{1}{p(n)}$. We typically denote an arbitrary negligible function by neg.

Definition 6. The DMSSE scheme Π is called L-semantically secure against non-adaptive attacks if for all PPT adversaries Adv there exists an efficient simulator Sim such that $|\mathsf{Real}^{\Pi}_{\mathsf{Adv}}(\lambda) - \mathsf{Ideal}^{\Pi}_{\mathsf{Adv,Sim}}(\lambda)| \leq neg(\lambda)$.

Security against malicious clients. Whenever a client registers to the system, the data owner associates $\mathcal{C}_i$ with a set of legitimate keywords and generates the authorization certificate. In our DMSSE scheme, each client is only permitted to proceed queries for the authorized keywords. The security requires that it is impossible for the malicious client to forge a valid search token for a query containing some non-authorized keywords, even if he collude with other clients. It is the same case for an adaptive client that can select the authorized keywords by himself. Formally, the security is defined via the following game between a challenger C and an adversary A:

- **Initialization**: The challenger runs algorithm $(\Delta, \Upsilon) \leftarrow \mathsf{KeyGen}(1^{\lambda})$: and returns the system parameter Δ to adversary A.
- **Client key extraction**: When receiving an authorization request for keywords W_i and pk_i, the challenger C runs algorithm ClientAuth and sends back Ω_i to A.
- **Output**: Eventually, the adversary outputs a search token $TK_{i,Q}$ for a new query Q containing some keyword $w \notin \mathsf{W}_i$, and the adversary **wins** if $TK_{i,Q}$ is valid.

Definition 7. The search token in Π is said to be unforgeable against adaptive clients if for a PPT adversary A, its advantage $Pr(A \textbf{ wins}) \leq neg(\lambda)$.

5. Dynamic multi-client SSE

Let H_i: $\{0,1\}^* \to \{0,1\}^{\lambda}$, for $i = 1, 2, 4$ and H_3: $\{0,1\}^* \to \{0,1\}^{2\lambda}$ be hash functions. To compute $H_i(x_1, \ldots, x_n)$, we first transform each input x_i into a bit string $\hat{x}_i$ and then take the concatenation of $\hat{x}_1 || \ldots || \hat{x}_n$ as the inputs of H_i. Let F: $\{0,1\}^{\lambda} \times \{0,1\}^{\lambda} \to \{0,1\}^{\lambda}$ and $F_p : \{0,1\}^{\lambda} \times \{0,1\}^{\lambda} \to \mathbb{Z}_p^*$ be PRFs. Our DMSSE scheme $\Pi = (\mathsf{KeyGen}, \mathsf{ClientAuth}, \mathsf{EDBSetup}, \mathsf{TokenGen}, \mathsf{Search})$ is constructed as follows.

5.1. Our construction

- $(\Delta, \Upsilon) \leftarrow \mathsf{KeyGen}(1^{\lambda})$: The data owner first runs Type-III pairing parameter generator Γ on input λ to generate the system parameter $\Delta = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, g_1, g_2)$. Then, she sets the master key as $\Upsilon = (K_T, K_S, K_X, K_I, K_Z, K_C, K_R)$ where K_T, K_S, $K_C \in \{0,1\}^{\lambda}$ are random keys for PRF F and K_X, K_I, K_Z, $K_R \in \{0,1\}^{\lambda}$ are random keys for PRF F_p.
- $(\mathsf{KAL}', \Omega_i) \leftarrow \mathsf{ClientAuth}(\Delta, \Upsilon, pk_i, \overline{\mathsf{W}}_i, \mathsf{KAL})$: This algorithm (Algorithm 1) run by the data owner $\mathcal{D}$ serves two functions: 1) The generation of an authorization certificate Ω_i for a legitimate client $\mathcal{C}_i$; 2) The update of the keyword access list KAL.

The date owner $\mathcal{D}$ chooses a random number r_i, with which she calculates uk_i for client $\mathcal{C}_i$. For each keyword w authorized to $\mathcal{C}_i$, $\mathcal{D}$ first computes a secret key K_w, with which she calculates $\mathrm{csk}_{i,w}$ and then adds $\mathrm{csk}_{i,w}$ to $\mathbf{CSK}_i$. Besides, she updates KAL by adding pk_i to PK_w. In the end, she outputs the new keyword access list KAL' and the authorization certificate Ω_i.

Algorithm 1 ClientAuth.

Input: $\Delta, \Upsilon, pk_i, \overline{\mathsf{W}}_i, \mathsf{KAL}$
Output: KAL', Ω_i
1: $\mathsf{KAL}' \leftarrow \mathsf{KAL}$
2: $r_i \leftarrow F_p(K_R, pk_i)$; $uk_i \leftarrow g_1^{r_i}$; $\mathbf{CSK}_i \leftarrow \emptyset$
3: **for** $w \in \overline{\mathsf{W}}_i$ **do**
4: $K_w \leftarrow F(K_C, w)$; $\mathrm{csk}_{i,w} \leftarrow H_1(K_w, pk_i)$; $\mathbf{CSK}_i \leftarrow \mathbf{CSK}_i \cup \mathrm{csk}_{i,w}$
5: $\mathsf{PK}_w \leftarrow \mathsf{KAL}'[w]$; $\mathsf{PK}_w \leftarrow \mathsf{PK}_w \cup pk_i$; $\mathsf{KAL}'[w] \leftarrow \mathsf{PK}_w$
6: **end for**
7: $\Omega_i \leftarrow (K_S, K_X, K_Z, \mathbf{CSK}_i, uk_i)$
8: **return** $(\mathsf{KAL}', \Omega_i)$

- $\mathsf{EDB} \leftarrow \mathsf{EDBSetup}(\mathsf{DB}, \Delta, \Upsilon, \mathsf{KAL})$: Given a database $\mathsf{DB} = (\mathrm{ind}_i, \mathsf{W}_i)_{i=1}^d$ and corresponding KAL, the data owner $\mathcal{D}$ generates an encrypted database $\mathsf{EDB} = (\mathbf{T}, \mathbf{X}, \mathbf{C})$ by running Algorithm 2. EDB consists of a CSet **C**, a TSet **T**, and an XSet **X**.

Algorithm 2 EDBSetup.

Input: $\mathsf{DB}, \Delta, \Upsilon, \mathsf{KAL}$
Output: EDB
1: $\mathbf{T} \leftarrow \emptyset$; $\mathbf{X} \leftarrow \emptyset$; $\mathbf{C} \leftarrow \emptyset$
2: **for** $w \in \mathsf{W}$ **do**
3: $K_e \leftarrow F(K_S, w)$; $\mathrm{stag}_w \leftarrow F(K_T, w)$; $\mathrm{xtrap}_w \leftarrow F_p(K_X, w)$
4: $K_w \leftarrow F(K_C, w)$; $\mathsf{PK}_w \leftarrow \mathsf{KAL}[w]$;$\mathbf{K} \leftarrow \emptyset$
5: **for** $pk_i \in \mathsf{PK}_w$ **do**
6: $r_i \leftarrow F_p(K_R, pk_i)$; $\mathbf{K} \leftarrow \mathbf{K} \cup pk_i^{r_i}$
7: $\mathrm{csk}_{i,w} \leftarrow H_1(K_w, pk_i)$; $\mathrm{ctag}_{i,w} \leftarrow H_2(\mathrm{csk}_{i,w}, g_1^{r_i}||0)$
8: $\mathsf{e}_1 \leftarrow \mathrm{stag}_w||0^\lambda \oplus H_3(\mathrm{csk}_{i,w}, g_1^{r_i}||1)$; $\mathbf{C}[\mathrm{ctag}_{i,w}] \leftarrow \mathsf{e}_1$
9: **end for**
10: $cnt \leftarrow 1$; $z \leftarrow F_p(K_Z, w)$
11: **for** $\mathrm{ind} \in \mathsf{DB}(w)$ **do**
12: $\mathrm{xind} \leftarrow F_p(K_I, \mathrm{ind})$; $\mathsf{e}_2 \leftarrow \mathbf{Enc}(K_e, \mathrm{ind})$
13: $y \leftarrow g_2^{\mathrm{xind} \cdot z^{-1}}$; $l \leftarrow H_4(\mathrm{stag}_w, cnt)$; $\mathbf{T}[l] \leftarrow (\mathsf{e}_2, y)$; $cnt \leftarrow cnt + 1$
14: **for** $pk_i^{r_i} \in \mathbf{K}$ **do**
15: $\mathrm{xtag} \leftarrow \hat{e}(pk_i^{r_i}, g_2^{\mathrm{xtrap}_w \cdot \mathrm{xind}})$; $\mathbf{X} \leftarrow \mathbf{X} \cup \mathrm{xtag}$
16: **end for**
17: **end for**
18: **end for**
19: $\mathsf{EDB} \leftarrow (\mathbf{T}, \mathbf{X}, \mathbf{C})$
20: **return** (EDB)

X.

Compared with the OXT scheme, the CSet **C** is a newly added structure playing the role of query permission authentication. Specifically, **C** is indexed by ctags and stores the encrypted forms of stags. Here, a ctag is related to a client's keys and a keyword (e.g, $\mathrm{ctag}_{i,w}$ is determined by client $\mathcal{C}$'s keys $csk_{i,w}$, uk_i, pk_i and keyword w), and a stag is calculated in the same way as that in the OXT scheme and it will be used to locate keyword-associated information (i.e., (e_2, y) pairs) in the TSet. If $\mathcal{C}_i$ is authorized to search keyword w, an encrypted version of stag_w will be stored at $\mathbf{C}[\mathrm{ctag}_{i,w}]$. Specifically, stag_w is first concatenated with λ-bit 0s, and then XORed with $H_3(\mathrm{csk}_{i,w}, g_1^{r_i}||1)$ to get e_1. A client cannot calculate a stag directly. Instead, he sends a search token to the server which recovers the stag from the CSet.

Our TSet **T** and XSet **X** have the similar data structures as that in the OXT scheme, but they are constructed in a completely different way due to the adoption of different types of groups [2] The OXT scheme is constructed based on DH-type operations over a Diffie-Hellman group, but our construction adopts bilinear groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ with a pairing $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$, where $\hat{e}$ is Type-III with no efficiently computable isomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$. Such groups satisfy an extended version of DDH assumption, the SAXDH assumption.

[2] Note that our TSet **T** is modeled as an inverted index for sublinear search time. However, unlike the OXT scheme indexing **T** merely by stags, our scheme utilizes a *counter* together with a stag to determine the location of keyword-related information. For each keyword w, its counter cnt is initialized to 1 and will increase by 1 for each $\mathrm{ind} \in \mathsf{DB}(w)$.

Specifically, for each pair (e_2, y) in the TSet, the OXT scheme sets $y = \mathrm{xind} \cdot z^{-1}$ as an element in $\mathbb{Z}_p^*$, where $z \in \mathbb{Z}_p^*$ is derived from keyword w and a pair counter cnt to ensure the independency of z and the randomness of y. In contrast, our scheme sets $y = g_2^{\mathrm{xind} \cdot z^{-1}}$ as an element in $\mathbb{G}_2$, where $z \in \mathbb{Z}_p^*$ is merely derived from keyword w due to the SAXDH assumption. For the XSet construction, in the OXT scheme, each xtag is an element of DH group in the form of $g_2^{F_p(K_X,w)\cdot F_p(K_I,\mathrm{ind})}$, the exponent of which is derived from keyword w and file identifier ind. In our scheme, each xtrap is an element of bilinear group $\mathbb{G}_T$ in the form of $\hat{e}(pk_i^{r_i}, g_2^{F_p(K_X,w)\cdot F_p(K_I,\mathrm{ind})})$, where the former part $pk_i^{r_i} \in \mathbb{G}_1$ is related to a client's identity information, and the latter part $g_2^{F_p(K_X,w)\cdot F_p(K_I,\mathrm{ind})} \in \mathbb{G}_2$ is related to keyword w and file identifier ind.

For the sake of distinction, codes that are the same as the OXT construction are underlined and colored as blue. Note that, as the OXT scheme, our scheme encrypts $\mathrm{ind} \in \mathsf{DB}(w)$ with SKE under the same key $K_e = F(K_S, w)$. Given K_S, the client is able to recover all file identifiers. However, we assume that the server never colludes with any client. That is, the client can only obtain the encrypted file identifiers matching the keywords within his search permission. For fine-grained access control over file identifiers, an ABE scheme supporting user revocation [24] can be applied for encryption.

- $TK_{i,Q} \leftarrow \mathsf{TokenGen}(Q, sk_i, \Omega_i)$: When a client $\mathcal{C}_i$ wants to issue a boolean query Q with a set of keywords $\overline{\mathsf{w}} \subseteq \overline{\mathsf{W}}_i$, he first chooses a sterm from $\overline{\mathsf{w}}$. For easy of illustration, we assume that Q is a conjunctive query in the forms of $\overline{w}_1 \wedge \overline{w}_2 \wedge \ldots \wedge \overline{w}_n$ and that $\overline{w}_1$ is the sterm. The construction of the TokenGen algorithm is shown in Algorithm 3. Note that this process

Algorithm 3 TokenGen.

Input: Q, sk_i, Ω_i
Output: $TK_{i,Q}$
1: $\mathrm{ctag}_{i,\overline{w}_1} \leftarrow H_2(\mathrm{csk}_{i,\overline{w}_1}, g_1^{r_i}||0)$; $dk_{i,\overline{w}_1} \leftarrow H_3(\mathrm{csk}_{i,\overline{w}_1}, g_1^{r_i}||1)$
2: **for** $j = 2, \ldots, n$ **do**
3: $\quad \mathrm{xtoken}_i[j] \leftarrow g_1^{r_i \cdot F_p(K_Z,\overline{w}_1)\cdot F_p(K_X,\overline{w}_j)\cdot sk_i}$
4: **end for**
5: $TK_{i,Q} \leftarrow (\mathrm{ctag}_{i,\overline{w}_1}, dk_{i,\overline{w}_1}, \mathrm{xtoken}_i[2], \ldots, \mathrm{xtoken}_i[n])$
6: **return** $TK_{i,Q}$

is totally non-interactive, not even requires the data owner to participate.

Unlike the OXT scheme enabling the client $\mathcal{C}_i$ to directly send $\mathrm{stag}_{\overline{w}_1}$ to the server, in our scheme, $\mathcal{C}_i$ cannot independently generate $\mathrm{stag}_{\overline{w}_1}$ since he does not know K_T. Instead, $\mathcal{C}_i$ generates $\mathrm{ctag}_{i,\overline{w}_1}$ and $dk_{i,\overline{w}_1}$ with his authorization certificate, so that the server can recover $\mathrm{stag}_{\overline{w}_1}$ in the Search algorithm. Furthermore, in our scheme, each xtoken contains client identification information $g^{r_i \cdot sk_i}$ in addition to keyword information $F_p(K_X, \overline{w}_j)$ and blinding factor $F_p(K_Z, \overline{w}_1)$. Note that the counter is omitted in the calculation of the blinding factor $F_p(K_Z, \overline{w}_1)$. Therefore, the number of xtokens is only determined by the number of keywords in query Q.

- $\mathbf{R} \leftarrow \mathsf{Search}(\mathsf{EDB}, TK_{i,Q})$: As shown in Algorithm 4, the server $\mathcal{S}$ evaluates a search token $TK_{i,Q}$ over an encrypted

Algorithm 4 Search.

Input: $\mathsf{EDB}, TK_{i,Q}$
Output: **R**
1: $\mathsf{e}_1 \leftarrow \mathbf{C}[\mathrm{ctag}_{i,\overline{w}_1}]$; $\mathbf{p}_1 \leftarrow \mathsf{e}_1 \oplus dk_{i,\overline{w}_1}$; $\mathbf{R} \leftarrow \emptyset$
2: **if** the last λ bits of $\mathbf{p}_1$ are all zeros **then**
3: $\quad \mathrm{stag}_{\overline{w}_1} \leftarrow$ the first λ bits of $\mathbf{p}_1$
4: **else~return** $\emptyset$
5: **end if**
6: $cnt \leftarrow 1$
7: **while** true **do**
8: $\quad l \leftarrow H_4(\mathrm{stag}_{\overline{w}_1}, cnt)$
9: $\quad$ **if** $\mathbf{T}[l] = null$ **then**
10: $\quad\quad$ break
11: $\quad$ **end if**
12: $\quad (\mathsf{e}_2, y) \leftarrow \mathbf{T}[l]$
13: $\quad$ **if** $\hat{e}(\mathrm{xtoken}_i[j], y) \in \mathbf{X}$ for $j = 2 \ldots, n$ **then**
14: $\quad\quad \mathbf{R} \leftarrow \mathbf{R} \cup \mathsf{e}_2$
15: $\quad$ **end if**
16: $\quad cnt \leftarrow cnt + 1$
17: **end while**
18: **return R**

database EDB, and returns the search results **R** to the client $\mathcal{C}_i$.

Different from the OXT scheme, $\mathcal{S}$ first visits the CSet to check whether $TK_{i,Q}$ is valid or not. If so, $\mathcal{S}$ checks the TSet and XSet in the similar way as the OXT scheme to find search results. Specifically, $\mathcal{S}$ first locates e_1 in the CSet indexed by $\mathrm{ctag}_{i,\overline{w}_1}$. After decrypting e_1, $\mathcal{S}$ checks whether the corresponding plaintext is ended with λ-bit 0s or not. If not, $\mathcal{S}$ returns an empty set to the client. Otherwise, $\mathcal{S}$ obtains $\mathrm{stag}_{\overline{w}_1}$ and locates each pair (e_2, y) in the TSet keyed by $H_4(\mathrm{stag}_{\overline{w}_1}, cnt)$ where cnt starts at 1 and increases by 1 until $\mathbf{T}[H_4(\mathrm{stag}_{\overline{w}_1}, cnt)]$ equals null. For each pair (e_2, y), it adds e_2 into the result set **R** if $\hat{e}(\mathrm{xtoken}_i[j], y)$ exists in the XSet.

5.2. Authorization update

One of the greatest merits of our DMSSE scheme is that it allows efficient update of clients' search permission. From the above construction, we know that a client's search permission on a keyword is expressed as an *independent* entry in both CSet and XSet. For an independent entry, its existence has nothing to do with other entries, and vice versa. Therefore, authorization update can be transformed to the addition/deletion of corresponding entries in the CSet and XSet. In the whole process, the data owner $\mathcal{D}$ interacts with the server $\mathcal{S}$ to perform update without requiring any client intervention.

To illustrate, let us consider the following example. Suppose that the client $\mathcal{C}_i$ with (sk_i, pk_i) joins the system, and is authorized to search keyword w. The data owner $\mathcal{D}$ first runs the ClientAuth algorithm to generate Ω_i and updates KAL by appending pk_i to PK_w. Then, $\mathcal{D}$ interacts with the server $\mathcal{S}$ as follows: 1) $\mathcal{D}$ calculates $\mathrm{ctag}_{i,w}$ and e_1 the encrypted version of stag_w by executing line 7–8 of the EDBSetup algorithm, and asks $\mathcal{S}$ to add $(\mathrm{ctag}_{i,w}, e_1)$ into the CSet. 2) She sends stag_w to the server, which executes lines 6–12 of the Search algorithm and return all the encrypted file identifiers matching w. 3) After recovering the plaintext file identifers, she generates a set of xtags by executing line 15 of the EDBSetup algorithm, and asks the server to adding these xtags to the XSet. The process of permission revocation is similar except that the "addition" operation is replaced by the "deletion" operation. We omit details of this part due to the space limitation.

5.3. Supporting for boolean queries

As the OXT scheme, our DMSSE scheme can be extended to handle boolean queries. Given a query Q over keywords $(w_1, \ldots, w_n)$, we first transform Q into the searchable normal form (SNF): $w_1 \wedge \varphi(w_2, \ldots, w_n)$, where φ is an arbitrary boolean formula (e.g., $\varphi(w_2, w_3, w_4) = w_2 \wedge w_3 \vee w_4$). Then, we generate a modified boolean formula $\hat{\varphi}$ by replacing keyword w_j with a boolean variable v_j. When a client $\mathcal{C}_i$ wants to perform query Q, he sends search token $TK_{i,Q}$ together with $\hat{\varphi}(v_2, \ldots, v_n)$ to the server. The server executes the Search algorithm and decides whether return the encrypted file identifier or not as follows: For $j = 2, \ldots, n$, it sets the boolean variable v_j to **true** if $\mathrm{xtoken}_i[j]$ can be found in the XSet. Then, it evaluates the expression $\hat{\varphi}$ based on the values of $v_2, \ldots, v_n$ and puts e_2 in **R** if the result if **true**.

5.4. Illustrative example

(***Initialization and user grant***) Suppose that the data owner $\mathcal{D}$ holds a database as shown in Fig. 4(a), and that the client $\mathcal{C}_1$ with key pair $(sk_1, pk_1 = g_1^{sk_1})$ can search keywords $\{w_1, w_2\}$. Given (Δ, Υ), $\mathcal{D}$ generates an authorization certificate $\Omega_1 = (K_S, K_X, K_Z, \mathbf{CSK}_1 = \{\mathrm{csk}_{1,w_1}, \mathrm{csk}_{1,w_2}\}, uk_1 = g_1^{r_1})$ for $\mathcal{C}_1$ and produces the original KAL as shown in Fig. 4(b).

(Data creation) Given $(\Delta, \Upsilon, \mathrm{KAL})$, the data owner $\mathcal{D}$ constructs the encrypted database $\mathsf{EDB} = (\mathsf{C}, \mathsf{T}, \mathsf{X})$ as shown in Fig. 5.

(Search) To retrieve files satisfying query $Q = w_1 \wedge w_2$, the client $\mathcal{C}_1$ sends the server $\mathcal{S}$ a search token $TK_{1,Q} = (\mathrm{ctag}_{1,w_1}, dk_{1,w_1}, \mathrm{xtoken}_1[2])$, where $\mathrm{xtoken}_1[2] = g_1^{r_1 \cdot F_p(K_Z, w_1) \cdot F_p(K_X, w_2) \cdot sk_1}$. Given $TK_{1,Q}$, $\mathcal{S}$ first evaluates the CSet as follows: It

File ID	Keywords
ind_1	w_1
ind_2	w_1, w_2
ind_3	w_2

(a) Database

Keyword	PK_w
w_1	pk_1
w_2	pk_1

(b) The original KAL

Keyword	PK_w
w_1	pk_1, pk_2
w_2	pk_1

(c) The updated KAL'

Fig. 4. The example database and KAL.

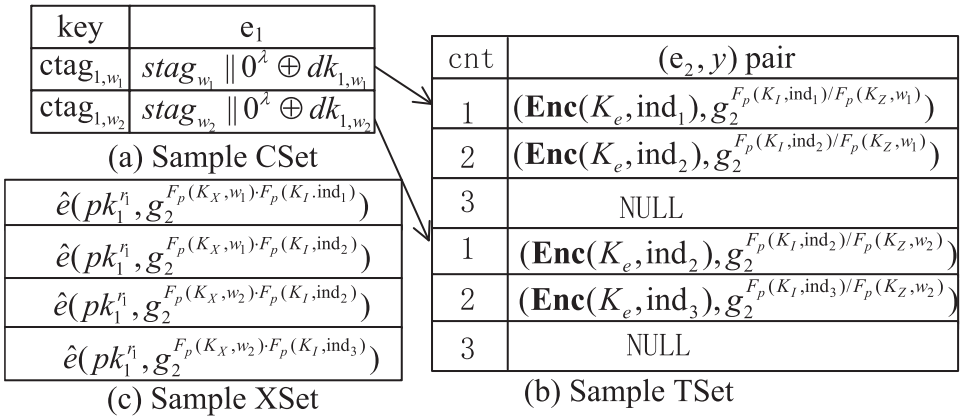

Fig. 5. The example encrypted database.

fetches $\mathtt{e}_1$ from $\mathbf{C}[\mathrm{ctag}_{1,w_1}]$, and computes $\mathtt{e}_1 \oplus dk_{1,w_1}$ to recover $\mathbf{p}_1 = \mathrm{stag}_{w_1}||0^{\lambda}$. Since $\mathbf{p}_1$ is ended with λ-bits 0s, $\mathcal{S}$ knows that $\mathcal{C}_1$ is authorized to search keyword w_1, and can recover stag_{w_1} by truncating $\mathbf{p}_1$ to its first λ bits.

Then, $\mathcal{S}$ operates the TSet as follows: It calculates $l \leftarrow H_4(\mathrm{stag}_{w_1}, cnt)$ and $(\mathbf{e}_2,\ y) \leftarrow \mathbf{T}[l]$, where cnt is initialized with 1 and increased by 1 until $\mathbf{T}[l]$ equals NULL. In this step, $\mathcal{S}$ obtains $(\mathbf{Enc}(K_e, \mathrm{ind}_1), g_2^{F_p(K_I,\mathrm{ind}_1)/F_p(K_Z,w_1)})$ and $(\mathbf{Enc}(K_e, \mathrm{ind}_2), g_2^{F_p(K_I,\mathrm{ind}_2)/F_p(K_Z,w_1)})$.

Finally, $\mathcal{S}$ tests the XSet as follows: For each $(\mathbf{e}_2,\ y)$ pair obtained in the last step, it computes $\mathrm{xtag} \leftarrow \hat{e}(\mathrm{xtoken}_1[2], y)$ and tests whether xtag exists in the XSet or not. If so, it puts $\mathbf{e}_2$ into results $\mathbf{R}$. In this step, $\mathbf{R} = \{\mathbf{Enc}(K_e, \mathrm{ind}_2)\}$ will be returned since only xtag_2 can be found in the XSet. The calculation process is as follows:

$$\begin{aligned}
\mathrm{xtag}_1 &= \hat{e}(\mathrm{xtoken}_1[2], g_2^{F_p(K_I,\mathrm{ind}_1)/F_p(K_Z,w_1)}) \\
&= \hat{e}(g_1^{r_1\cdot F_p(K_Z,w_1)\cdot F_p(K_X,w_2)\cdots k_1}, g_2^{F_p(K_I,\mathrm{ind}_1)/F_p(K_Z,w_1)}) \\
&= \hat{e}(g_1^{r_1\cdot F_p(K_X,w_2)\cdots k_1}, g_2^{F_p(K_I,\mathrm{ind}_1)}) \\
&= \hat{e}(pk_1^{r_1}, g_2^{F_p(K_X,w_2)\cdot F_p(K_I,\mathrm{ind}_1)}) \\
\mathrm{xtag}_2 &= \hat{e}(\mathrm{xtoken}_1[2], g_2^{F_p(K_I,\mathrm{ind}_2)/F_p(K_Z,w_1)}) \\
&= \hat{e}(g_1^{r_1\cdot F_p(K_Z,w_1)\cdot F_p(K_X,w_2)\cdots k_1}, g_2^{F_p(K_I,\mathrm{ind}_2)/F_p(K_Z,w_1)}) \\
&= \hat{e}(g_1^{r_1\cdot F_p(K_X,w_2)\cdots k_1}, g_2^{F_p(K_I,\mathrm{ind}_2)}) \\
&= \hat{e}(pk_1^{r_1}, g_2^{F_p(K_X,w_2)\cdot F_p(K_I,\mathrm{ind}_2)})
\end{aligned}$$

(Update) Suppose that the client $\mathcal{C}_2$ with key pair $(sk_2,\ pk_2 = g_1^{sk_2})$ joins in the system, and he is entitled to search keywords w_1. The data owner first creates an authorization certificate $\Omega_2 = (K_S, K_X, K_Z, \mathbf{CSK}_2 = \{\mathrm{csk}_{2,w_1}\}, uk_1 = g_1^{r_2})$ for $\mathcal{C}_2$ and generates an updated KAL as shown in Fig. 4(c). Then, she updates the CSet and XSet as shown in Fig. 6. Note that, in the update phase, the data owner only needs to interact with the server for generating an updated database without affecting any client.

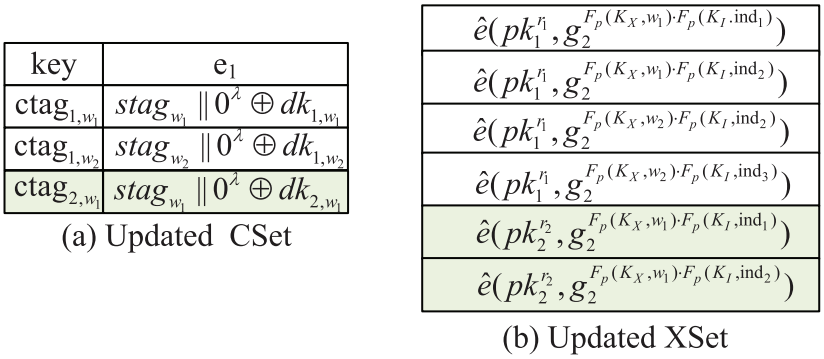

Fig. 6. The updated encrypted database.

Table 3
Comparison of functionality features.

	Query type	Multi-client	Non-interactive	Dynamic
OXT	Boolean	No	-	-
MSSE	Boolean	Yes	No	No
DMSSE	Boolean	Yes	Yes	Yes

Table 4
Comparison of computation cost.

	OXT	MSSE	DMSSE
EDBSetup	$O(N \cdot \mathsf{exp})$	$O(N \cdot \mathsf{exp})$	$O(N \cdot \mathsf{exp} + N \cdot U \cdot \mathsf{bp})$
TokenGen	$O(n \cdot P \cdot \mathsf{exp})$	$O(n \cdot P \cdot \mathsf{exp})$	$O(n \cdot \mathsf{exp})$
Search	$O(P \cdot \mathsf{exp})$	$O(P \cdot \mathsf{exp})$	$O(P \cdot \mathsf{bp})$

Table 5
Comparison of communication cost.

	Token size	Database size
OXT	$O(n \cdot P \cdot \lvert\mathbb{G}_1\rvert)$	$O(N \cdot (\lambda + \lvert\mathbb{Z}_p^*\rvert + \lvert\mathbb{G}_1\rvert))$
MSSE	$O(n \cdot P \cdot \lvert\mathbb{G}_1\rvert)$	$O(N \cdot (\lambda + \lvert\mathbb{Z}_p^*\rvert + \lvert\mathbb{G}_1\rvert))$
DMSSE	$O(n \cdot \lvert\mathbb{G}_1\rvert)$	$O(N \cdot (\lambda + \lvert\mathbb{G}_2\rvert) + N \cdot U \cdot \lvert\mathbb{G}_T\rvert)$

6. Evaluation

In this section, we evaluate the performance of our DMSSE scheme in terms of computation and communication costs, and we compare it with Cash et al. [6] (denoted by OXT) and Jarecki et al. [12] (denoted by MSSE).

6.1. Performance analysis

Before performance analysis, we first provide a comparison of functionality features in Table 3. Both our work and Jarecki et al. [12] extend the OXT scheme by Cash et al. [6] to a multi-client setting. However, our DMSSE scheme aims to achieve non-interactive and dynamic authorization, and thus is more practical.

For the computation cost, we only consider the most expensive operations, i.e., exponentiations (denoted by exp) and bilinear pairing (denoted by bp). In algorithms KeyGen and ClientAuth, neither exp operations nor bp operations are involved, and we only consider the performance of algorithms EDBSetup, TokenGen, and Search. Let $U = \max|\mathsf{PK}_w|$ denote the maximal number of clients being authorized to search keyword $w \in \mathsf{W}$, and let $P = \max|\mathsf{DB}(w)|$ denote the maximal number of files matching keyword $w \in \mathsf{W}$. The comparison results are shown in Table 4, where notations N and n are the total number of keyword-identifier pairs and the maximal number of keywords in queries, respectively.

For the communication cost, we mainly consider the size of a search token from the client to the server and the size of the encrypted database from the data owner to the server. For all schemes, a file identifier is encrypted with SKE, and the ciphertext size is related to the security parameter λ. Let $|\mathbb{Z}_p^*|$, $|\mathbb{G}_1|$, $|\mathbb{G}_2|$, and $|\mathbb{G}_T|$ denote the size of an element in groups $\mathbb{Z}_p^*$, $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$, respectively. The comparison results are shown in Table 5, where notations n, N, U, and P have the same meaning as Table 4.

6.2. Parameter setting

Our experiments are conducted on a local machine running the Microsoft Windows 10 Ultimate operating system and Intel(R) Core(TM) i7-7700M CPU with 3.60 GHz and 16 GB memory. Our programs are implemented in Java compiled using IntelliJ IDEA 15.0.2. We leverage the javax.crypto and java.security library for cryptographic primitives, and use the jPBC library for group operation and bilinear pair calculation. We choose symmetric elliptic curve groups $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ with order of prime number $p = 33,546,239$, satisfying the equation $y^2 = x^3 + x$. The security parameter λ is set to 128. We use HMAC-MD5 for PRFs and hash functions $H_i (i = 1, 2, 4)$, HMAC-SHA256 for hash function H_3, and AES for the Enc algortihm.

Enron Email Dataset[3] is chosen to validate the feasibility of our scheme in practice. This dataset includes more than 30,000 email files, sent by about 150 different users. Each email is composed of a set of message heads and a message body. The elements in the head in the form of attribute-value pairs (e.g., (sender, Alice)) are chosen as keywords directly. For words in the message body, only those appear in more than 10 files are chosen as keyword values, and their attributes are all set to "content". After the pre-processing, we extract $m = 21,623$ keywords with 15 district attributes.

[3] http://www.cs.cmu.edu/./enron/.

Table 6
Experiment parameters.

Parameters	Description
$U \in [5, 30]$	The maximal number of clients
$N \in [581817, 2962756]$	The total number of keyword-identifer pairs
$n \in [10, 500]$	The number of keywords in a query
$\|DB(w)\| \in [10, 30000]$	The number of files matching a keyword
$m \in [7709, 21623]$	The total number of keywords
$d \in [6000, 30000]$	The total number of files

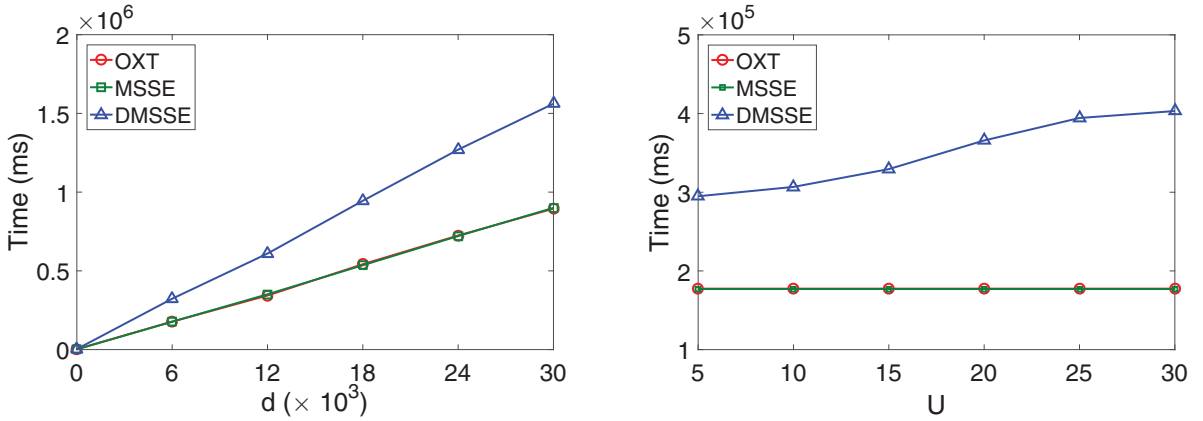

Fig. 7. Comparison results of the execution time for generating an encrypted database. (a) A database of d files and fixed $U = 10$ clients. (b) A database of fixed $d = 6,000$ files and U clients.

In the experiments, there are $U \in [5, 30]$ clients in the system, and we assign them to search $n \in [10, 500]$ random keywords. In terms of types of queries, we only consider the conjunctive query. We show the experiment parameters in Table 6.

6.3. Experiment results

The computation costs of algorithms KenGen and ClientAuth are related inexpensive. Therefore, we omit the related comparison results.

Fig. 7 shows the comparison result for generating an encrypted database. The results in this figure are consistent with the analysis results. The running time of the EDBSetup algorithm in three schemes is linearly with the dataset size and it is also impacted by the number of clients in our scheme. Our DMSSE scheme takes the most time for the following reasons. First, bilinear map operations used in our scheme are more expensive than exponential operations used in OXT and MSSE. Second, the number of bilinear map operations is related to both the number of clients and the keyword-identifier pairs. However, this is a one-time cost and can be done once for all.

Fig. 8 shows the comparison result for generating a search token. This cost grows linearly with the number of keywords n in the query in all three schemes, and is impacted by the number of files matching a keyword w, denoted by $|DB(w)|$, in OXT and MSSE. For example, with fixed $|DB(w)| = 200$, as n increases from 1 to 500, the execution time increases from 1ms to 52ms in our scheme, and increases from 10ms to 5800ms in OXT and MSSE. Therefore, our scheme is the most efficient among three scheme. Fig. 9 shows the comparison result of the search process. For all three schemes, this cost grows linearly with the number of files matching a keyword w, denoted by $|DB(w)|$, but is slightly impacted by the number of keywords n in the query.

Summary. Our DMSSE scheme consumes the most execution time in generating an encrypted database. The reason is that the performance of algorithm EDBSetup is also impacted by the maximal number of clients authorized to search a keyword in addition to the total number of keyword-identifier pairs in the system. However, this is a one-time cost and related process can be executed just once for all. Furthermore, our DMSSE scheme performs the best in generation of a search token. Therefore, the TokenGen algorithm incurs the least computation overhead on the client. Finally, our DMSSE scheme costs the most execution time in the searching process. The main reason is that the server needs to evaluate the search token over an additional CSet in the Search algorithm. However, from the experiment results, we know that this extra overhead is within acceptable limits.

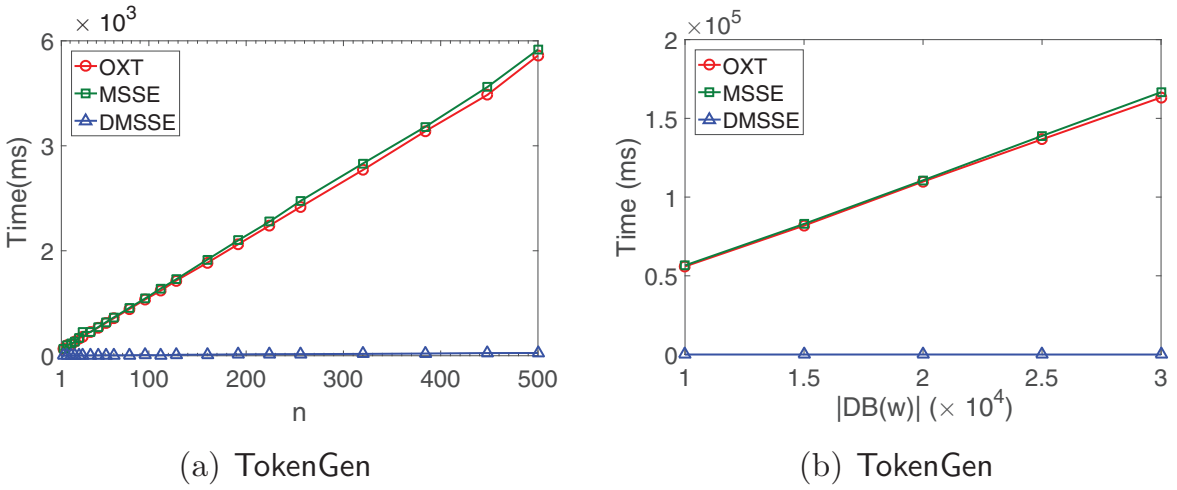

Fig. 8. Comparison results of search token generation time. (a)Fixed size $|\mathrm{DB}(w)| = 200$ and n keywords in the query. (b)$|\mathrm{DB}(w)|$ varies from 10,000 to 30,000 and fixed $n = 100$ keywords in the query.

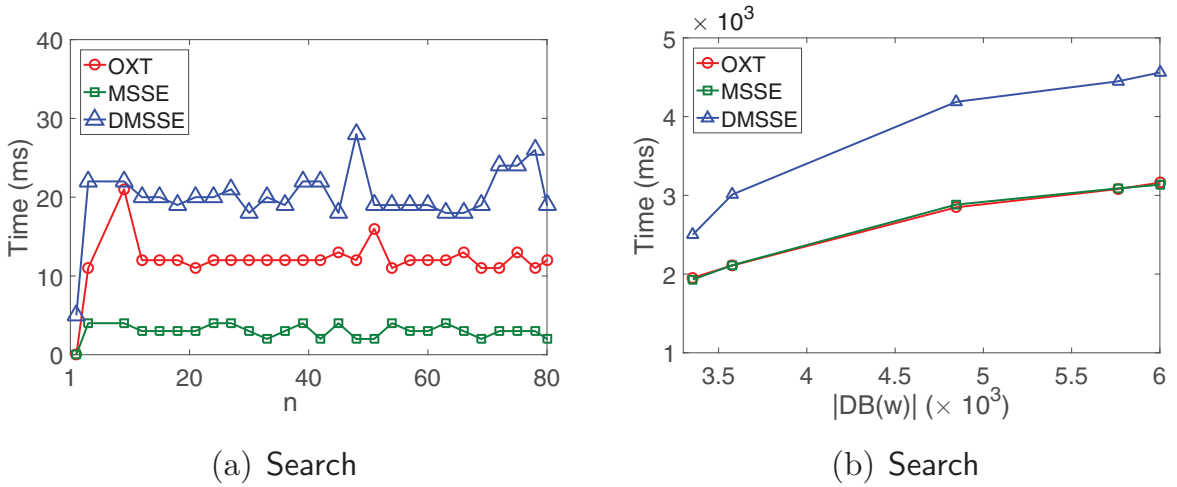

Fig. 9. Comparison results of the search time. (a)Fixed size $|\mathrm{DB}(w)| = 200$ and n keywords in the query. (b)$|\mathrm{DB}(w)|$ varies from 3300 to 6000 and fixed $n = 100$ keywords in the query.

7. Conclusion

In this paper, we study the problem of authorized boolean queries over encrypted data in the multi-client setting. The proposed DMSSE scheme allows a data owner to authorize multiple clients to perform boolean queries with their permission in a non-interactive and dynamic way. Experiment results demonstrate that our scheme is effective in a large-scale encrypted dataset. However, the main insufficient of our scheme is that the size of the XSet grows as the number of clients. As part of our future work, we will try to improve the efficiency of our scheme by reducing the size of the XSet.

Declaration of Competing Interest

We declare that we have no financial and personal relationships with other people or organizations that can inappropriately influence our work, there is no professional or other personal interest of any nature or kind in any product, service and/or company that could be construed as influencing the position presented in, or the review of the manuscript entitled “Dynamic Multi-Client Searchable Symmetric Encryption with Support for Boolean Queries”.

Acknowledgments

This work was supported in part by NSFC grants 61632009, 61872133, 61872130, and 61572181.

Appendix A. Security against adversarial server

Before proving the security of our DMSSE scheme, we first provide the description of leakage function $\mathcal{L}$. For ease of illustration, we consider a simple scenario where a client issues only conjunctive queries with fixed number of xterms. The security of arbitrary boolean queries can be derived in the same way as in Ref. [6].

A sequence of T non-adaptive queries is represented by $\mathbf{q} = (\mathbf{s}, \mathbf{x}, \mathsf{CPK})$ and each individual query is represented as $\mathbf{q}[t] = (\mathbf{s}[t], \mathbf{x}[t, \cdot], \mathsf{CPK}[t])$, where $\mathbf{s}[t]$ and $\mathbf{x}[t, \cdot] = \mathbf{x}[t, 1], \ldots, \mathbf{x}[t, n]$ denote the sterm and n xterms in the t-th query, respectively, and $\mathsf{CPK}[t]$ is the public key of the client issuing the t-th query, for $t \in [T]$. Taking $\mathsf{DB} = (\mathrm{ind}_i, \mathsf{W}_i)_{i=1}^{d}$, $\mathsf{KAL} = (\mathsf{PK}_w)_{w \in \mathsf{W}}$, and $\mathbf{q} = (\mathbf{s}, \mathbf{x}, \mathsf{CPK})$ as input, the leakage function $\mathcal{L} = (N, M, \bar{\mathbf{s}}, \mathsf{SP}, \mathsf{RP}, \mathsf{IP}, \mathsf{CPK})$ is defined as below.

- $N = \sum_{i=1}^{d} |\mathsf{W}_i|$ is the total number of (ind, w) pairs.
- $N' = \sum_{i=1}^{d} \sum_{w \in \mathsf{W}_i} |\mathsf{PK}_{w_i}|$ is the total number of $(\mathrm{ind}, w, \mathcal{C})$ pairs.
- $M = \sum_{i=1}^{m} |\mathsf{PK}_{w_i}|$ is the total number of $(\mathcal{C}, w)$ pairs.
- $\bar{\mathbf{s}} \in [m]^T$ is the equality pattern of $\mathbf{s} \in \mathsf{W}^T$ indicating which queries have the same sterms.
- SP is the *size pattern* of the queries, which is the number of files matching the sterm in each query. Formally, $\mathsf{SP}[t] = |\mathsf{DB}(\mathbf{s}[t])|$ for $t \in [T]$.
- RP is the *result pattern* of the queries, which consists of the identifiers of files matching the sterm and any other xterm in the query. Formally, $\mathsf{RP}[t, \alpha] = \mathsf{DB}(\mathbf{s}[t]) \cap \mathsf{DB}(\mathbf{x}[t, \alpha])$ for $t \in [T]$ and $\alpha \in [n]$.
- IP is the *conditional intersection pattern* defined as follows:

$$\mathsf{IP}[t_i, t_j; \alpha, \beta] = \begin{cases} \mathsf{DB}(\mathbf{s}[t_i]) \cap \mathsf{DB}(\mathbf{s}[t_j]), & \text{if} \quad i \neq j \quad \text{and} \quad \mathbf{x}[t_i, \alpha] = \mathbf{x}[t_j, \beta] \\ \emptyset \end{cases}$$

- CPK consists of the public keys of clients issuing queries. If the t-th query is issued by client $\mathcal{C}_i$, then $\mathsf{CPK}[t] \leftarrow pk_i$.

Theorem 1. *Let $\mathcal{L}$ be the leakage function defined above. Our DMSSE scheme Π is $\mathcal{L}$-semantically secure against non-adaptive attacks in the random oracle model, where all queries are n-conjunctions, assuming that the SAXDH assumption holds in groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$, that F and F_p are secure PRFs and that $\Sigma = (\mathsf{Enc}, \mathsf{Dec})$ is an IND-CPA SKE scheme.*

Proof of Theorem 1.. The hash functions $H_1, \ldots, H_4$ are modeled as random oracles with different output lengths. For hash function H_i with output length l, its oracle maintains a mapping $\mathbb{M}_i$ that stores input/output pairs (x, y) where $x \in \{0, 1\}^*$ and $y \in \{0, 1\}^l$. Given an input string x, the oracle first checks whether there is an entry x in mapping $\mathbb{M}_i$ or not. If so, it sets y as the value associated with x (denoted by $y \leftarrow \mathbb{M}_i[x]$) and returns y. Otherwise, the oracle randomly picks a string y from $\{0, 1\}^l$ and stores (x, y) in $\mathbb{M}_i$ (denoted by $\mathbb{M}_i[x] \leftarrow y$) and returns y.

We structure our proof using several games $\mathsf{G}_0, \ldots, \mathsf{G}_5$, where the differences between G_i and G_{i+1} are underlined and marked as blue color in the algorithm of G_{i+1}. In each game, Adv starts by supplying $(\mathsf{DB}, \mathbf{q})$, which is then given to an Initialize function, the outputs of which are given to Adv who then outputs a bit as the game's output. Game G_0 is designed to generate exactly the same distribution as $\mathsf{Real}^{\Pi}_{\mathsf{Adv}}$ except that G_0 assumes no false positives occur. The final game G_5 is structured so that it is easy to simulate with given leakage profile. By relating the games, we argue that the final simulator satisfies Definition 6, completing the proof. For easy of illustration, we assume a client only queries keywords within his permission.

Game G_0. This game shown in Algorithm 5 is an implementation of the real game with some minor changes that makes the following analysis easier. G_0 starts by running Initialize on input of $(\mathsf{DB}, \mathsf{KAL}, \mathbf{s}, \mathbf{x}, \mathsf{CPK})$ to simulate EDBSetup. Initialize is identical to EDBSetup except that the construction of the XSet is separated as a subfunction XSetSetup.

Before generating the transcripts, G_0 runs the ClientAuth function that simulates the ClientAuth algorithm shown in Algorithm 1 to generate the secret key Ω_i for client $\mathcal{C}_i$. For $t \in [T]$, G_0 runs function $\mathsf{TransGen}(\mathsf{EDB}, \Omega_i, \Delta, \mathbf{s}[t], \mathbf{x}[t, \cdot], \mathsf{CPK}[t])$ and sets transcript $\mathbf{t}[t]$ to its outputs $(\mathsf{stag}, \mathbf{xtoken}, \mathsf{Res}, \mathsf{ResInds})$. The TransGen function simulates the TokenGen algorithm shown in Algorithm 3 to generate search tokens and runs the Search algorithm shown in Algorithm 4 as its subfunction to generate search result Res. However, instead of decrypting the returned ciphertexts Res, it computes $\mathsf{DB}(\mathbf{s}[t]) \cap \mathsf{DB}(\mathbf{x}[t, 1]) \cap \ldots \cap \mathsf{DB}(\mathbf{x}[t, n])$ to obtain ResInds.

Now G_0 and $\mathsf{Real}^{\Pi}_{\mathsf{Adv}}$ behave exactly the same except that the false positives are assumed to never happen. By Theorem 1, assuming that F_p is a secure PRF, we have:

$$\Pr[\mathsf{G}_0 = 1] - \Pr[\mathsf{Real}^{\Pi}_{\mathsf{Adv}} = 1] \leq \mathrm{neg}(\lambda)$$

Game G_1. This game as shown in Algorithm 6 is identical to G_0 except that we replace the evaluation of PRFs F_p and F with random functions, and that we record the values of stag, r, and K_w in arrays **Stag, R, KW**, respectively, after they are first computed and look them up on later, instead of recomputing them. Furthermore, G_1 takes **R, KW** and the input in TransGen function so that the ClientAuth function can be omitted in the simulation.

Note that since functions $F(K_T, \cdot)$, $F(K_S, \cdot)$, $F_p(K_R, \cdot)$, and $F(K_C, \cdot)$ are never evaluated on the same input twice, we can equivalently replace their evaluations with random selections from given ranges. By a standard hybrid argument, it is easy to see that there exist efficient adversaries $\mathcal{B}_{1,1}$ and $\mathcal{B}_{1,2}$ such that:

$$\Pr[\mathsf{G}_1 = 1] - \Pr[\mathsf{G}_0 = 1] \leq 3\mathsf{Adv}^{\mathrm{prf}}_{F, \mathcal{B}_{1,1}}(\lambda) + 4\mathsf{Adv}^{\mathrm{prf}}_{F_p, \mathcal{B}_{1,2}}(\lambda)$$

Algorithm 5 G_0.

function INITIALIZE(DB, KAL, $\mathbf{s}$, $\mathbf{x}$, CPK)
  $\Delta = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, g_1, g_2) \xleftarrow{\$} \Gamma(\lambda)$
  $\Upsilon = (K_T, K_S, K_X, K_I, K_Z, K_C, K_R) \xleftarrow{\$} \{0,1\}^\lambda$
  $(\mathrm{ind}_i, \mathsf{W}_i)_{i=1}^d \leftarrow \mathsf{DB}$; $(\mathsf{PK}_w)_{w\in\mathsf{W}} \leftarrow \mathsf{KAL}$; $(\mathbf{C}, \mathbf{T}, \mathbf{X}) \leftarrow (\emptyset, \emptyset, \emptyset)$
  for $w \in \mathsf{W}$ **do**
    $\mathrm{stag}_w \leftarrow F(K_T, w)$; $K_w \leftarrow F(K_C, w)$; $z \leftarrow F_p(K_Z, w)$; $K_e \leftarrow F(K_S, w)$
    for $pk_i \in \mathsf{PK}_w$ **do**
      $r_i \leftarrow F(K_R, pk_i)$; $\mathrm{csk}_{i,w} \leftarrow \mathbb{M}_1(K_w, pk_i)$
      $\mathsf{e}_1 \leftarrow \mathrm{stag}_w || 0^\lambda \oplus \mathbb{M}_3(\mathrm{csk}_{i,w}, g_1^{r_i} || 1)$; $\mathbf{C}[\mathbb{M}_2(\mathrm{csk}_{i,w}, g_1^{r_i} || 0)] \leftarrow \mathsf{e}_1$
    end for
    $(\overline{\mathrm{ind}}_1, \ldots, \overline{\mathrm{ind}}_{\#w}) \leftarrow \mathsf{DB}(w)$
    for $cnt \in [\#w]$ **do**
      $l \leftarrow \mathbb{M}_4(\mathrm{stag}_w, cnt)$; $\mathrm{xind} \leftarrow F_p(K_I, \overline{\mathrm{ind}}_{cnt})$
      $\mathsf{e}_2 \leftarrow \mathsf{Enc}(K_e, \overline{\mathrm{ind}}_{cnt})$; $y \leftarrow g_2^{\mathrm{xind}\cdot z^{-1}}$; $\mathbf{T}[l] \leftarrow (\mathsf{e}_2, y)$
    end for
  end for
  $\mathbf{X} \leftarrow \mathsf{XSetSetup}(\Delta, \Upsilon, \mathsf{DB}, \mathsf{KAL})$; $\Omega_i \leftarrow \mathsf{ClientAuth}(\Delta, \Upsilon, pk_i, \mathsf{KAL})$
  $\mathsf{EDB} \leftarrow (\mathbf{C}, \mathbf{T}, \mathbf{X})$
  for $t \in [T]$ **do**
    $\mathbf{t}[t] \leftarrow \mathsf{TransGen}(\mathsf{EDB}, \Omega_i, \Delta, \mathbf{s}[t], \mathbf{x}[t, \cdot], \mathsf{CPK}[t])$
  end for
  return $(\mathsf{EDB}, \mathbf{t})$
end function

function XSETSETUP(Δ, Υ, DB, KAL)
  $(\mathrm{ind}_i, \mathsf{W}_i)_{i=1}^d \leftarrow \mathsf{DB}$; $(\mathsf{PK}_w)_{w\in\mathsf{W}} \leftarrow \mathsf{KAL}$
  for $w \in \mathsf{W}$ and $\mathrm{ind} \in \mathsf{DB}(w)$ **do**
    $\mathrm{xtrap}_w \leftarrow F_p(K_X, w)$; $\mathrm{xind} \leftarrow F_p(K_I, \mathrm{ind})$
    for $pk_i \in \mathsf{PK}_w$ **do**
      $r_i \leftarrow F(K_R, pk_i)$; $\mathrm{xtag} \leftarrow \hat{e}(pk_i^{r_i}, g_2^{\mathrm{xtrap}_w \cdot \mathrm{xind}})$; $\mathbf{X} \leftarrow \mathbf{X} \cup \mathrm{xtag}$
    end for
  end for
  return $\mathbf{X}$
end function

function CLIENTAUTH(Δ, Υ, pk_i, KAL)
  generate $\overline{\mathsf{W}}_i$ from KAL; $r_i \leftarrow F(K_R, pk_i)$; $uk_i \leftarrow g_1^{r_i}$; $\mathbf{CSK}_i \leftarrow \emptyset$
  for $w \in \overline{\mathsf{W}}_i$ **do**
    $K_w \leftarrow F(K_C, w)$; $\mathrm{csk}_{i,w} \leftarrow \mathbb{M}_1(K_w, pk_i)$; $\mathbf{CSK}_i \leftarrow \mathbf{CSK}_i \cup \mathrm{csk}_{i,w}$
  end for
  $\Omega_i \leftarrow (K_S, K_X, K_Z, \mathbf{CSK}_i, uk_i)$
  return Ω_i
end function

function TRANSGEN(EDB, Ω_i, Δ, $\mathbf{s}[t]$, $\mathbf{x}[t, \cdot]$, CPK[t]) ▷ The keywords in query $\mathbf{q}[t]$ satisfy that $\mathbf{s}[t] \cup \mathbf{x}[t, \cdot] \subseteq \overline{\mathsf{W}}_i$
  $w \leftarrow \mathbf{s}[t]$; $z \leftarrow F_p(K_Z, w)$; $\mathrm{ctag}_{i,w} \leftarrow \mathbb{M}_2(\mathrm{csk}_{i,w}, g_1^{r_i} || 0)$; $dk_{i,w} \leftarrow \mathbb{M}_3(\mathrm{csk}_{i,w}, g_1^{r_i} || 1)$
  for $\alpha \in [n]$ **do**
    $\mathrm{xtrap} \leftarrow F_p(K_X, \mathbf{x}[t, \alpha])$; $\mathbf{xtoken}_i[\alpha] \leftarrow \mathsf{CPK}[t]^{r_i \cdot \mathrm{xtrap} \cdot z}$
  end for
  $(\mathrm{stag}, \mathsf{Res}) \leftarrow \mathsf{Search}(\mathsf{EDB}, \mathrm{ctag}_{i,w}, dk_{i,w}, \mathbf{xtoken}_i)$
  $\mathsf{ResInds} \leftarrow \mathsf{DB}(\mathbf{s}[t]) \cap \mathsf{DB}(\mathbf{x}[t, 1]) \cap \ldots \cap \mathsf{DB}(\mathbf{x}[t, n])$
  return $(\mathrm{stag}, \mathbf{xtoken}, \mathsf{Res}, \mathsf{ResInds})$
end function

Algorithm 6 G_1, $\boxed{\mathsf{G}_2}$.

function INITIALIZE($\mathsf{DB}, \mathsf{KAL}, \mathbf{s}, \mathbf{x}, \mathsf{CPK}$)
  $\Delta = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, g_1, g_2) \xleftarrow{\$} \Gamma(\lambda)$
  $f_I, f_X, f_Z \leftarrow \mathsf{Fun}(\{0,1\}^\lambda, \mathbb{Z}_p^*)$
  $(\mathrm{ind}_i, \mathsf{W}_i)_{i=1}^d \leftarrow \mathsf{DB}$; $(\mathsf{PK}_w)_{w\in\mathsf{W}} \leftarrow \mathsf{KAL}$; $(\mathbf{C}, \mathbf{T}, \mathbf{X}) \leftarrow (\emptyset, \emptyset, \emptyset)$
  for $w \in \mathsf{W}$ **do**
    $\mathrm{stag}_w \xleftarrow{\$} \{0,1\}^\lambda$; $K_e \xleftarrow{\$} \{0,1\}^\lambda$; $K_w \xleftarrow{\$} \{0,1\}^\lambda$; $z \leftarrow f_Z(w)$
    $\mathbf{KW}[w] \leftarrow K_w$; $\mathbf{Stag}[w] \leftarrow \mathrm{stag}_w$
    for $pk_i \in \mathsf{PK}_w$ **do**
      if $\mathbf{R}[i] = \perp$ **then**
        $r_i \xleftarrow{\$} \mathbb{Z}_p^*$; $\mathbf{R}[i] \leftarrow r_i$
      end if
      $\mathrm{csk}_{i,w} \leftarrow \mathbb{M}_1(K_w, pk_i)$; $\mathsf{e}_1 \leftarrow \mathrm{stag}_w || 0^\lambda \oplus \mathbb{M}_3(\mathrm{csk}_{i,w}, g_1^{\mathbf{R}[i]} || 1)$
      $\mathbf{C}[\mathbb{M}_2(\mathrm{csk}_{i,w}, g_1^{\mathbf{R}[i]} || 0)] \leftarrow \mathsf{e}_1$
    end for
    $(\overline{\mathrm{ind}}_1, \ldots, \overline{\mathrm{ind}}_{\#w}) \leftarrow \mathsf{DB}(w)$
    for $cnt \in [\#w]$ **do**
      $l \leftarrow \mathbb{M}_4(\mathrm{stag}_w, cnt)$; $\mathrm{xind} \leftarrow f_I(\overline{\mathrm{ind}}_{cnt})$
      $\mathsf{e}_2 \leftarrow \mathsf{Enc}(K_e, \overline{\mathrm{ind}}_{cnt})$; $\boxed{\mathsf{e}_2 \leftarrow \mathsf{Enc}(K_e, 0^\lambda)}$
      $y \leftarrow g_2^{\mathrm{xind}\cdot z^{-1}}$; $\mathbf{T}[l] \leftarrow (\mathsf{e}_2, y)$
    end for
  end for
  $\mathbf{X} \leftarrow \mathsf{XSetSetup}(\Delta, \mathsf{DB}, f_X, f_I, \mathbf{R}, \mathsf{KAL})$
  $\mathsf{EDB} \leftarrow (\mathbf{C}, \mathbf{T}, \mathbf{X})$
  for $t \in [T]$ **do**
    $\mathbf{t}[t] \leftarrow \mathsf{TransGen}(\mathsf{EDB}, \Delta, \mathbf{R}, \mathbf{KW}, f_Z, f_X, \mathbf{s}[t], \mathbf{x}[t, \cdot], \mathsf{CPK}[t])$
  end for
  return $(\mathsf{EDB}, \mathbf{t})$
end function

function XSETSETUP($\Delta, \mathsf{DB}, f_X, f_I, \mathbf{R}, \mathsf{KAL}$)
  $(\mathrm{ind}_i, \mathsf{W}_i)_{i=1}^d \leftarrow \mathsf{DB}$; $(\mathsf{PK}_w)_{w\in\mathsf{W}} \leftarrow \mathsf{KAL}$
  for $w \in \mathsf{W}$ and $\mathrm{ind} \in \mathsf{DB}(w)$ **do**
    $\mathrm{xtrap}_w \leftarrow f_X(w)$; $\mathrm{xind} \leftarrow f_I(\mathrm{ind})$
    for $pk_i \in \mathsf{PK}_w$ **do**
      $r_i \leftarrow \mathbf{R}[i]$; $\mathrm{xtag} \leftarrow \hat{e}(pk_i^{r_i}, g_2^{\mathrm{xtrap}_w \cdot \mathrm{xind}})$; $\mathbf{X} \leftarrow \mathbf{X} \cup \mathrm{xtag}$
    end for
  end for
  return X
end function

function TRANSGEN($\mathsf{EDB}, \Delta, \mathbf{R}, \mathbf{KW}, f_Z, f_X, \mathbf{s}[t], \mathbf{x}[t, \cdot], \mathsf{CPK}[t]$)
  $pk_i \leftarrow \mathsf{CPK}[t]$; $r_i \leftarrow \mathbf{R}[i]$; $w \leftarrow \mathbf{s}[t]$; $z \leftarrow f_Z(w)$; $\mathrm{csk}_{i,w} \leftarrow \mathbb{M}_1(\mathbf{KW}[w], pk_i)$
  $\mathrm{ctag}_{i,w} \leftarrow \mathbb{M}_2(\mathrm{csk}_{i,w}, g_1^{r_i} || 0)$; $dk_{i,w} \leftarrow \mathbb{M}_3(\mathrm{csk}_{i,w}, g_1^{r_i} || 1)$
  for $\alpha \in [n]$ **do**
    $\mathrm{xtrap} \leftarrow f_X(\mathbf{x}[t, \alpha])$; $\mathbf{xtoken}_i[\alpha] \leftarrow pk_i^{r_i \cdot \mathrm{xtrap} \cdot z}$
  end for
  $(\mathrm{stag}, \mathsf{Res}) \leftarrow \mathsf{Search}(\mathsf{EDB}, \mathrm{ctag}_{i,w}, dk_{i,w}, \mathbf{xtoken}_i)$
  $\mathsf{ResInds} \leftarrow \mathsf{DB}(\mathbf{s}[t]) \cap \mathsf{DB}(\mathbf{x}[t, 1]) \cap \ldots \cap \mathsf{DB}(\mathbf{x}[t, n])$
  return $(\mathrm{stag}, \mathbf{xtoken}, \mathsf{Res}, \mathsf{ResInds})$
end function

Game G_2. The difference from G_1 is marked as the boxed code in G_2. This means that the encryption of a file identifier is replaced with an encryption of a constant string 0^λ. Since m encryption keys are used in building the TSet, we claim that there exist an efficient adversary $\mathcal{B}_2$ such that:

$$\Pr[G_2 = 1] - \Pr[G_1 = 1] \leq m \cdot \mathsf{Adv}^{\mathrm{ind-cpa}}_{\Sigma,\mathcal{B}_2}(\lambda)$$

Game G_3. As shown in Algorithm 7, this game alters the way of constructing the TSet, XSet and xtoken without changing their distributions. Intuitively, it precomputes $f_X(w)$ and $f_Z(w)$ for each $w \in W$ and precomputes $f_I(\mathrm{ind})$ for each file identifier ind, and then records them in arrays **H, Z**, and **I**, respectively.

In construction of the TSet, for each (w, ind) pair, we use the values from arrays **I** and **Z** and set y to $g_2^{\mathbf{I}[\mathrm{ind}]\cdot\mathbf{Z}[w]^{-1}}$. Similarly, functions XSetSetup and TransGen use the values from arrays to calculate xtags and xtokens. Since **H, Z**, and **I** are calculated in the same way as G_2, the outputs of two games are the same, and we have:

$$\Pr[G_3 = 1] = \Pr[G_2 = 1]$$

Game G_4. This game is almost identical to G_3 except that the boxed code is included in Algorithm 7. Instead computing the values of arrays **H, Z, I** as before, G_4 selects them at random from $\mathbb{Z}_p^*$. We claim that there exists an efficient adversary $\mathcal{B}_4$ such that:

$$\Pr[G_4 = 1] - \Pr[G_3 = 1] \leq \mathsf{Adv}^{\mathrm{SAXDH}}_{\mathbb{G}_1,\mathbb{G}_2,\mathcal{B}_4}(\lambda)$$

Intuitively, the distribution of $y = g_2^{f_I(\mathrm{ind})\cdot f_Z(w)^{-1}}$ is indistinguishable from a random element in $\mathbb{G}_2$ if we replace $f_I(\mathrm{ind})$ with x, replace $f_Z(w)^{-1}$ with s_1, replace $f_Z(w')^{-1}$ with s_2, replace $sk_i \cdot r_i \cdot f_X(w')$ with I, and replace $sk_i \cdot r_i \cdot f_X(w)$ with J. With the above replacement, the xtag can be calculated as $\hat{e}(g_1^{j/s_1}, g_2^{xs_1}) = \hat{e}(g_1, g_2)^{x\cdot j}$. Similarly, the distribution of xtokens is indistinguishable from a random element in $\mathbb{G}_1$.

Game G_5. As shown in Algorithm 8, G_5 changes the way to access arrays and random oracles to enable the final simulator to work with its given leakage. The case for random oracles is simple. Whenever G_5 access a random oracle $\mathbb{M}_i$, it first checks whether the input is related to a keyword $w \in \mathbf{s}$ or not. If so, it runs random oracle to get the output; otherwise, it chooses a random value from a given range as the output. Furthermore, we replace the inputs of $\mathbb{M}_2$ and $\mathbb{M}_3$ with a client's public key and a keyword disabling the use of random oracle $\mathbb{M}_1$.

Now, we describe how arrays **H, I, Z**, and **R** are accessed. Intuitively, no array will be accessed independently. To calculate the value of y in the TSet, G_5 will use access $\mathbf{I}[\mathrm{ind}]$ and $\mathbf{Z}[w]$ simultaneously in two cases: 1) For $t \in [T]$ and $\alpha \in [n]$, keyword w is the sterm of the t-th query, and file ind contains keyword w and the α-th xterm in the t-th query. 2) For $t_1, t_2 \in [T]$ and $\alpha, \beta \in [n]$, keyword w is the sterm of the t_1-th query, file ind contains both w and the sterm of the t_2-th query, and two queries have different sterms and common xterms. In other cases, y is set as a random element in $\mathbb{G}_2$.

To calculate the xtag in the XSet, $\mathbf{H}[w]$, $\mathbf{I}[\mathrm{ind}]$, and $\mathbf{R}[i]$ will be accessed simultaneously only when keyword w is an xterm of the t-th query, ind contains the sterm of the t-th query, and client $\mathcal{C}_i$ issues the t-th query, for $t \in [T]$ and $\alpha \in [n]$. The modifications will not change the distribution of the game, and thus we have:

$$\Pr[G_5 = 1] - \Pr[G_4 = 1] \leq neg(\lambda)$$

Simulator. We complete the proof by giving a simulator Sim that takes the leakage $\mathcal{L}(\mathrm{DB}, \mathbf{s}, \mathbf{x}) = (N, M, \bar{\mathbf{s}}, \mathrm{SP}, \mathrm{RP}, \mathrm{IP}, \mathrm{CPK})$ as input and outputs a simulated EDB and a simulated transcript array **t**. By combining the relations between the games, we show that $(\mathrm{EDB}, \mathbf{t})$ simulated by Sim have the same distributions as $\mathsf{Initialize}(\mathrm{DB}, \mathbf{s}, \mathbf{x})$ in G_5.

Before simulation, Sim uses the conditional intersection pattern IP to calculate a *restricted equality pattern* of **x**, denoted by $\hat{\mathbf{x}} \in [m]^{T\times n}$, as follows. First, Sim computes a relation $\equiv$ on $[T, n]$ by defining $[t_1, \alpha] \equiv [t_2, \beta]$ if $\mathrm{IP}[t_1, t_2; \alpha, \beta] \neq \emptyset$ and makes $\equiv$ an equivalence relation by taking its transitive closure. Then, Sim gives each partition an index number, and sets $\hat{\mathbf{x}}[t, \alpha]$ to the index number of partition containing $[t, \alpha]$.

As in the OXT scheme, $\hat{\mathbf{x}}$ can be used as the *equality pattern* of **x**. For all $t_1, t_2 \in [T]$ and $\alpha, \beta \in [n]$, it has the following properties:

$$\hat{\mathbf{x}}[t_1, \alpha] = \hat{\mathbf{x}}[t_2, \beta] \Rightarrow \mathbf{x}[t_1, \alpha] = \mathbf{x}[t_2, \beta]$$

and

$$(\mathbf{x}[t_1, \alpha] = \mathbf{x}[t_2, \beta]) \wedge (\mathrm{DB}(\mathbf{s}[t_1]) \cap \mathrm{DB}(\mathbf{s}[t_2]) \neq \emptyset) \Rightarrow \hat{\mathbf{x}}[t_1, \alpha] = \hat{\mathbf{x}}[t_2, \beta]$$

Algorithm 9 shows the simulator codes to generate arrays **H, I, Z**, and **R**. Unlike G_5 that fills arrays **H, Z** with every $w \in W$ and fills array **I** with every $\mathrm{ind} \in \mathrm{DB}$, Sim only does it with leakage, where all keywords w are integers from 1 to T. The simulated EDB and transcript **t** are shown in Algorithm 10 and Algorithm 11, respectively.

When computing the CSet, Sim first creates an entry for each $(\mathbf{s}[t], \mathrm{CPK}[t])$ pair. It tracks how many entries are added this way with the counter c, and adds random strings of 2λ bits to random positions of **C** until a total of M entries are added. Since both Sim and G_5 produce CSet by randomly placing M uniform and independent strings, they have the same distribution. In the construction of the TSet, the same logical is used in both games except that identifiers that do not appear are replaced by dummy symbols, and thus they are clearly identically distributed.

Algorithm 7 G_3, $\boxed{\mathsf{G}_4}$.

function INITIALIZE($\mathsf{DB}, \mathsf{KAL}, \mathbf{s}, \mathbf{x}, \mathsf{CPK}$)
$\Delta = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, g_1, g_2) \xleftarrow{\$} \Gamma(\lambda)$
$f_I, f_X, f_Z \leftarrow \mathsf{Fun}(\{0,1\}^\lambda, \mathbb{Z}_p^*)$
$(\mathrm{ind}_i, \mathsf{W}_i)_{i=1}^d \leftarrow \mathsf{DB}$; $(\mathsf{PK}_w)_{w\in\mathsf{W}} \leftarrow \mathsf{KAL}$; $(\mathbf{C}, \mathbf{T}, \mathbf{X}) \leftarrow (\emptyset, \emptyset, \emptyset)$
for $w \in \mathsf{W}$ **do**
$\underline{\mathbf{H}[w] \leftarrow f_X(w)}$; $\mathbf{Z}[w] \leftarrow f_Z(w)$ $\boxed{\mathbf{H}[w] \xleftarrow{\$} \mathbb{Z}_p^*; \mathbf{Z}[w] \xleftarrow{\$} \mathbb{Z}_p^*}$
end for
for $\mathrm{ind} \in \mathsf{DB}$ **do**
$\underline{\mathbf{I}[\mathrm{ind}] \leftarrow f_I(\mathrm{ind})}$ $\boxed{\mathbf{I}[\mathrm{ind}] \xleftarrow{\$} \mathbb{Z}_p^*}$
end for
for $w \in \mathsf{W}$ **do**
$\mathrm{stag}_w \xleftarrow{\$} \{0,1\}^\lambda$; $K_e \xleftarrow{\$} \{0,1\}^\lambda$; $K_w \xleftarrow{\$} \{0,1\}^\lambda$
$\mathbf{KW}[w] \leftarrow K_w$; $\mathbf{Stag}[w] \leftarrow \mathrm{stag}_w$
for $pk_i \in \mathsf{PK}_w$ **do**
if $\mathbf{R}[i] = \perp$ **then**
$r_i \xleftarrow{\$} \mathbb{Z}_p^*$; $\mathbf{R}[i] \leftarrow r_i$
end if
$\mathrm{csk}_{i,w} \leftarrow \mathbb{M}_1(K_w, pk_i)$; $\mathsf{e}_1 \leftarrow \mathrm{stag}_w || 0^\lambda \oplus \mathbb{M}_3(\mathrm{csk}_{i,w}, g_1^{\mathbf{R}[i]} || 1)$
$\mathbf{C}[\mathbb{M}_2(\mathrm{csk}_{i,w}, g_1^{\mathbf{R}[i]} || 0)] \leftarrow \mathsf{e}_1$
end for
$(\overline{\mathrm{ind}}_1, \ldots, \overline{\mathrm{ind}}_{\#w}) \leftarrow \mathsf{DB}(w)$;$\underline{z \leftarrow \mathbf{Z}[w]}$
for $cnt \in [\#w]$ **do**
$l \leftarrow \mathbb{M}_4(\mathrm{stag}_w, cnt)$; $\underline{\mathrm{xind} \leftarrow \mathbf{I}[\overline{\mathrm{ind}}_{cnt}]}$
$\mathsf{e}_2 \leftarrow \mathsf{Enc}(K_e, 0^\lambda)$; $y \leftarrow g_2^{\mathrm{xind}\cdot z^{-1}}$; $\mathbf{T}[l] \leftarrow (\mathsf{e}_2, y)$
end for
end for
$\mathbf{X} \leftarrow \mathsf{XSetSetup}(\Delta, \mathsf{DB}, \mathbf{H}, \mathbf{I}, \mathbf{R}, \mathsf{KAL})$
$\mathsf{EDB} \leftarrow (\mathbf{C}, \mathbf{T}, \mathbf{X})$
for $t \in [T]$ **do**
$\mathbf{t}[t] \leftarrow \mathsf{TransGen}(\mathsf{EDB}, \Delta, \mathbf{R}, \mathbf{KW}, \mathbf{Z}, \mathbf{H}, \mathbf{s}[t], \mathbf{x}[t, \cdot], \mathsf{CPK}[t])$
end for
return $(\mathsf{EDB}, \mathbf{t})$
end function

function XSETSETUP($\Delta, \mathsf{DB}, \mathbf{H}, \mathbf{I}, \mathbf{R}, \mathsf{KAL}$)
$(\mathrm{ind}_i, \mathsf{W}_i)_{i=1}^d \leftarrow \mathsf{DB}$; $(\mathsf{PK}_w)_{w\in\mathsf{W}} \leftarrow \mathsf{KAL}$
for $w \in \mathsf{W}$ and $\mathrm{ind} \in \mathsf{DB}(w)$ **do**
$\underline{\mathrm{xtrap}_w \leftarrow \mathbf{H}[w]; \mathrm{xind} \leftarrow \mathbf{I}[\mathrm{ind}]}$
for $pk_i \in \mathsf{PK}_w$ **do**
$r_i \leftarrow \mathbf{R}[i]$;$\mathrm{xtag} \leftarrow \hat{e}(pk_i^{r_i}, g_2^{\mathrm{xtrap}_w\cdot\mathrm{xind}})$; $\mathbf{X} \leftarrow \mathbf{X} \cup \mathrm{xtag}$
end for
end for
return X
end function

function TRANSGEN($\mathsf{EDB}, \Delta, \mathbf{R}, \mathbf{KW}, \mathbf{Z}, \mathbf{H}, \mathbf{s}[t], \mathbf{x}[t, \cdot], \mathsf{CPK}[t]$))
$pk_i \leftarrow \mathsf{CPK}[t]$; $r_i \leftarrow \mathbf{R}[i]$; $w \leftarrow \mathbf{s}[t]$; $\underline{z \leftarrow \mathbf{Z}[w]}$; $\mathrm{csk}_{i,w} \leftarrow \mathbb{M}_1(\mathbf{KW}[w], pk_i)$
$\mathrm{ctag}_{i,w} \leftarrow \mathbb{M}_2(\mathrm{csk}_{i,w}, g_1^{r_i} || 0)$; $\mathrm{dk}_{i,w} \leftarrow \mathbb{M}_3(\mathrm{csk}_{i,w}, g_1^{r_i} || 1)$
for $\alpha \in [n]$ **do**
$\underline{\mathrm{xtrap} \leftarrow \mathbf{H}(\mathbf{x}[t, \alpha])}$;$\mathbf{xtoken}_i[\alpha] \leftarrow pk_i^{r_i\cdot\mathrm{xtrap}\cdot z}$
end for
$(\mathrm{stag}, \mathsf{Res}) \leftarrow \mathsf{Search}(\mathsf{EDB}, \mathrm{csk}_{i,w}, \mathrm{dk}_{i,w}, \mathbf{xtoken}_i)$
$\mathsf{ResInds} \leftarrow \mathsf{DB}(\mathbf{s}[t]) \cap \mathsf{DB}(\mathbf{x}[t, 1]) \cap \ldots \cap \mathsf{DB}(\mathbf{x}[t, n])$
return $(\mathrm{stag}, \mathbf{xtoken}, \mathsf{Res}, \mathsf{ResInds})$
end function

In construction of the XSet, we consider any two index/keyword pairs $(\mathrm{ind}_1, \mathbf{x}[t_1, \alpha])$ and $(\mathrm{ind}_2, \mathbf{x}[t_2, \beta])$ read from **I** and **Z** by G_5. The simulator Sim will read $(\mathrm{ind}_1, \hat{\mathbf{x}}[t_1, \alpha])$ and $(\mathrm{ind}_2, \hat{\mathbf{x}}[t_2, \beta])$ with the same indices from either RP or IP instead of from **I** and **Z**. We show the simulation is correct by claiming that:

$$(\mathrm{ind}_1, \mathbf{x}[t_1, \alpha]) = (\mathrm{ind}_2, \mathbf{x}[t_2, \beta]) \Leftrightarrow (\mathrm{ind}_1, \hat{\mathbf{x}}[t_1, \alpha]) = (\mathrm{ind}_2, \hat{\mathbf{x}}[t_2, \beta]) \tag{A.1}$$

Algorithm 8 G_5.

function INITIALIZE(DB, KAL, **s**, **x**, CPK)
  $\Delta = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, g_1, g_2) \xleftarrow{\$} \Gamma(\lambda)$
  $(\mathrm{ind}_i, \mathsf{W}_i)_{i=1}^{d} \leftarrow \mathsf{DB}$; $(\mathsf{PK}_w)_{w \in \mathsf{W}} \leftarrow \mathsf{KAL}$; $(\mathbf{C}, \mathbf{T}, \mathbf{X}) \leftarrow (\emptyset, \emptyset, \emptyset)$
  for $w \in \mathsf{W}$ **do**
    $\mathbf{H}[w] \xleftarrow{\$} \mathbb{Z}_p^*$; $\mathbf{Z}[w] \xleftarrow{\$} \mathbb{Z}_p^*$
  end for
  for $\mathrm{ind} \in \mathsf{DB}$ **do**
    $\mathbf{I}[\mathrm{ind}] \xleftarrow{\$} \mathbb{Z}_p^*$
  end for
  for $pk_i \in \mathsf{CPK}$ **do**
    $\mathbf{R}[i] \leftarrow \mathbb{Z}_p^*$
  end for
  for $w \in \mathsf{W}$ **do**
    $\mathrm{stag}_w \xleftarrow{\$} \{0,1\}^\lambda$; $K_e \xleftarrow{\$} \{0,1\}^\lambda$; $\mathbf{Stag}[w] \leftarrow \mathrm{stag}_w$
    for $pk_i \in \mathsf{PK}_w$ **do**
      if $\exists t : \mathbf{s}[t] = w \wedge \mathsf{CPK}[t] = pk_i$ **then**
        $\mathrm{ctag}_{i,w} \leftarrow \mathbb{M}_2(\mathsf{CPK}[t], w)$; $\mathbf{C}[\mathrm{ctag}_{i,w}] \leftarrow \mathrm{stag}_w || 0^\lambda \oplus \mathbb{M}_3(\mathsf{CPK}[t], w)$
      else
        $\mathrm{ctag}_{i,w} \xleftarrow{\$} \{0,1\}^\lambda$; $\mathbf{C}[\mathrm{ctag}_{i,w}] \xleftarrow{\$} \{0,1\}^{2\lambda}$
      end if
    end for
    $(\overline{\mathrm{ind}}_1, \ldots, \overline{\mathrm{ind}}_{\#w}) \leftarrow \mathsf{DB}(w)$
    for $cnt \in [\#w]$ **do**
      if $\exists t_1 : \mathbf{s}[t_1] = w$ **then**
        if $\exists \alpha : (\overline{\mathrm{ind}}_{cnt} \subseteq \mathsf{DB}(\mathbf{s}[t_1]) \cap \mathsf{DB}(\mathbf{x}[t_1, \alpha]))$ **then**
          $y \leftarrow g_2^{\mathbf{I}[\overline{\mathrm{ind}}_{cnt}] \cdot \mathbf{Z}[w]^{-1}}$
        else
          if $\exists t_2, \alpha, \beta : (\mathbf{s}[t_1] \neq \mathbf{s}[t_2]) \wedge (\overline{\mathrm{ind}}_{cnt} \in (\mathsf{DB}(\mathbf{s}[t_1]) \cap \mathsf{DB}(\mathbf{s}[t_2]))) \wedge \mathbf{x}[t_1, \alpha] = \mathbf{x}[t_2, \beta]$ **then**
            $y \leftarrow g_2^{\mathbf{I}[\overline{\mathrm{ind}}_{cnt}] \cdot \mathbf{Z}[w]^{-1}}$
          else
            $y \xleftarrow{\$} \mathbb{G}_2$;
          end if
        end if
        $l \leftarrow \mathbb{M}_4(\mathrm{stag}_w, cnt)$
      else
        $l \xleftarrow{\$} \{0,1\}^\lambda$
      end if
      $\mathsf{e}_2 \leftarrow \mathsf{Enc}(K_e, 0^\lambda)$; $\mathbf{T}[l] \leftarrow (\mathsf{e}_2, y)$
    end for
  end for
  $\mathbf{X} \leftarrow \mathsf{XSetSetup}(\Delta, \mathsf{DB}, \mathsf{KAL}, \mathbf{s}, \mathbf{x}, \mathsf{CPK}, \mathbf{H}, \mathbf{I}, \mathbf{R})$
  $\mathsf{EDB} \leftarrow (\mathbf{C}, \mathbf{T}, \mathbf{X})$
  for $t \in [T]$ **do**
    $\mathrm{stag} \leftarrow \mathbf{Stag}[\mathbf{s}[t]]$; $pk_i \leftarrow \mathsf{CPK}[t]$; $r_i \leftarrow \mathbf{R}[i]$
    $\mathbf{t}[t] \leftarrow \mathsf{TransGen}(\mathsf{EDB}, \mathbf{H}, \mathbf{Z}, \Delta, \mathbf{s}[t], \mathbf{x}[t, \cdot], r_i, pk_i)$
  end for
  return $(\mathsf{EDB}, \mathbf{t})$
end function

Algorithm 8 (*Continued*)

function XSetSetup(Δ, DB, KAL, $\mathbf{s}$, $\mathbf{x}$, CPK, $\mathbf{H}$, $\mathbf{I}$, $\mathbf{R}$)
  $(\mathrm{ind}_i, \mathsf{W}_i)_{i=1}^{d} \leftarrow \mathsf{DB}$; $(\mathsf{PK}_w)_{w\in\mathsf{W}} \leftarrow \mathsf{KAL}$
  for $w \in \mathsf{W}$ and $\mathrm{ind} \in \mathsf{DB}(w)$ **do**
    for $pk_i \in \mathsf{PK}_w$ **do**
      if $\underline{\exists t, \alpha : (\mathrm{ind} \in \mathsf{DB}(\mathbf{s}[t])) \wedge (\mathbf{x}[t,\alpha] = w) \wedge (\mathsf{CPK}[t] = pk_i)}$ **then**
        $\underline{\mathrm{xtag} \leftarrow \hat{e}(pk_i^{\mathbf{R}[i]}, g_2^{\mathbf{H}[w]\cdot\mathbf{I}[\mathrm{ind}]})}$
      else
        $\underline{\mathrm{xtag} \xleftarrow{\$} \mathbb{G}_T}$
      end if
      $\mathbf{X} \leftarrow \mathbf{X} \cup \mathrm{xtag}$
    end for
  end for
  return X
end function

function TransGen(EDB, $\mathbf{H}$, $\mathbf{Z}$, Δ, $\mathbf{s}[t]$, $\mathbf{x}[t,\cdot]$, r_i, pk_i)
  $\mathrm{ctag}_{i,w} \leftarrow \mathbb{M}_2(pk_i, \mathbf{s}[t])$; $\mathrm{dk}_{i,w} \leftarrow \mathbb{M}_3(pk_i, \mathbf{s}[t])$
  for $\alpha \in [n]$ **do**
    $\mathbf{xtoken}_i[\alpha] \leftarrow pk_i^{r_i\cdot\mathbf{H}(\mathbf{x}[t,\alpha])\cdot\mathbf{Z}(\mathbf{s}[t])}$
  end for
  $\mathsf{Res} \leftarrow \mathsf{Search}(\mathsf{EDB}, \mathrm{ctag}_{i,w}, \mathrm{dk}_{i,w}, \mathbf{xtoken}_i)$
  $\mathsf{ResInds} \leftarrow \mathsf{DB}(\mathbf{s}[t]) \cap \mathsf{DB}(\mathbf{x}[t,1]) \cap \ldots \cap \mathsf{DB}(\mathbf{x}[t,n])$
  return (stag, **xtoken**, Res, ResInds)
end function

Algorithm 9 Simulator codes to generate **H, Z, I, R**.

for $j \in \hat{\mathbf{x}}$ **do**
  $\mathbf{H}[j] \xleftarrow{\$} \mathbb{Z}_p^*$
end for
for $\mathrm{ind} \in \cup_{t\in[T],\alpha\in[n]}\mathsf{RP}[t,\alpha] \cup_{t_1,t_2\in[T],\alpha,\beta\in[n]} \mathsf{IP}[t_1,t_2;\alpha,\beta]$ **do**
  $\mathbf{I}[\mathrm{ind}] \xleftarrow{\$} \mathbb{Z}_p^*$
end for
for $j \in \bar{\mathbf{s}}$ **do**
  $\mathbf{Z}[j] \xleftarrow{\$} \mathbb{Z}_p^*$
end for
for $pk_i \in \mathsf{CPK}$ **do**
  $\mathbf{R}[i] \xleftarrow{\$} \mathbb{Z}_p^*$
end for

The $\Leftarrow$ direction is easy to prove by the properties of $\hat{\mathbf{x}}$. For the $\Rightarrow$ direction, suppose that $(\mathrm{ind}_1, \mathbf{x}[t_1,\alpha]) = (\mathrm{ind}_2, \mathbf{x}[t_2,\beta])$. Then we have $\mathrm{ind}_1 = \mathrm{ind}_2$, meaning that $\mathsf{DB}(\mathbf{s}[t_1]) \cap \mathsf{DB}(\mathbf{s}[t_2]) \neq \emptyset$, and thus we have $\hat{\mathbf{x}}[t_1] = \hat{\mathbf{x}}[t_2]$ by the property of $\hat{\mathbf{x}}$. Similarly, the distribution of the transcript $\mathbf{t}$ in both games is identical. □

Appendix B. Security against malicious clients

Theorem 2. *Our DMSSE scheme* Π *secure against malicious client, i.e., search token in* Π *is unforgeable against adaptive attacks in random oracle model, assuming that discrete logarithm problem is intractable in group* $\mathbb{G}_1$*, and that F and* F_p *are secure PRFs.*

Proof of Theorem. Suppose that there exists an adversarial client $\mathcal{C}_i$ can generate a valid search token for an unauthorized keyword w. This means that, the client can either guess $(\mathrm{ctag}_{j,w}, \mathrm{dk}_{j,w})$ of an authorized client $\mathcal{C}_j$ or generate a valid xtoken $g_1^{sk_j\cdot r_j\cdot F_p(K_Z,w)\cdot F_p(K_X,w)}$ because our adversary model assumes that the server $\mathcal{S}$ never colludes with any client.

For the first case, the probability of a successful guess is negligible due to the security of PRFs. For the latter case, the adversary needs to guess $g_1^{sk_j\cdot r_j}$. The security of r_j is guaranteed by the security of PRF. Given $\mathcal{C}_j$'s public key $g_1^{sk_j}$, it is hard

Algorithm 10 Simulator code to generate EDB.

for $j \in \bar{\mathbf{s}}$ **do**
  $\mathbf{Stag}[j] \xleftarrow{\$} \{0,1\}^\lambda$;
end for
$c \leftarrow 0$
for $t \in [T]$ **do**
  $j \leftarrow \bar{\mathbf{s}}[t]$; $pk_i \leftarrow \mathsf{CPK}[t]$
  if $\mathbf{C}[\mathbb{M}_2[pk_i, j]] = \perp$ **then**
    $\mathbf{C}[\mathbb{M}_2[pk_i, j]] \leftarrow \mathbf{Stag}[j] || 0^\lambda \oplus \mathbb{M}_3[pk_i, j]$; $c \leftarrow c+1$
  end if
end for
for $k = c+1, \ldots, M$ **do**
  $l \xleftarrow{\$} \{0,1\}^\lambda$; $\mathbf{C}[l] \xleftarrow{\$} \{0,1\}^{2\lambda}$
end for

$c \leftarrow 0$
for $j \in \bar{\mathbf{s}}$ **do**
  $\mathbf{IND} \leftarrow \emptyset$; $K_e \xleftarrow{\$} \{0,1\}^\lambda$
  for $t \in [T]$ and $\alpha \in [n]$ **do**
    if $\bar{\mathbf{s}}[t] = j$ **then**
      $\mathbf{IND} \leftarrow \mathbf{IND} \cup \mathsf{RP}[t, \alpha] \cup_{t' \in [T], \beta \in [n]} \mathsf{IP}[t, t'; \alpha, \beta]$
    end if
  end for
  ▷ sort indices in **IND** in canonical order and pad $|\mathbf{IND}|$ up to size $\mathsf{SP}[j]$ by adding $\perp$ indices such that $\mathbf{IND} = (\overline{\mathrm{ind}}_1, \ldots, \overline{\mathrm{ind}}_{\mathsf{SP}[j]})$
  for $cnt \in \mathsf{SP}[j]$ **do**
    if $\overline{\mathrm{ind}}_{cnt} \neq \perp$ **then**
      $y \leftarrow g_2^{\mathbf{I}[\overline{\mathrm{ind}}_{\mathrm{cnt}}] \cdot \mathbf{Z}[j]^{-1}}$
    else
      $y \leftarrow \mathbb{G}_2$
    end if
    $l \leftarrow \mathbb{M}_4[j, cnt]$; $\mathsf{e}_2 \leftarrow \mathsf{Enc}(K_e, 0^\lambda)$; $\mathbf{T}[l] \leftarrow (\mathsf{e}_2, y)$; $c \leftarrow c+1$
  end for
end for
for $k = c+1, \ldots, N$ **do**
  $l \xleftarrow{\$} \{0,1\}^\lambda$; $K_e \xleftarrow{\$} \{0,1\}^\lambda$; $\mathsf{e}_2 \leftarrow \mathsf{Enc}(K_e, 0^\lambda)$; $y \xleftarrow{\$} \mathbb{G}_2$; $\mathbf{T}[l] \leftarrow (\mathsf{e}_2, y)$
end for
for $pk_i \in \mathsf{CPK}$ **do**
  $r_i \leftarrow \mathbf{R}[i]$; $\mathbf{T}' \leftarrow \emptyset$
  for $t \in [T]$ **do**
    if $\mathsf{CPK}[t] = pk_i$ **then**
      $\mathbf{T}' \leftarrow \mathbf{T}' \cup t$
    end if
  end for
  for $j \in \hat{\mathbf{x}} \wedge \mathrm{ind} \in \cup_{t \in \mathbf{T}', \alpha \in [n]: \hat{\mathbf{x}}[t,\alpha] = j} \mathsf{RP}[t, \alpha]$ **do**
    $\mathsf{xtag} \leftarrow \hat{e}(pk_i^{r_i}, g_2^{\mathbf{H}[j] \cdot \mathbf{I}[\mathrm{ind}]})$; $\mathbf{X} \leftarrow \mathbf{X} \cup \mathsf{xtag}$; $c \leftarrow c+1$
  end for
end for
for $k = c+1, \ldots, N'$ **do**
  $\mathsf{xtag} \xleftarrow{\$} \mathbb{G}_T$; $\mathbf{X} \leftarrow \mathbf{X} \cup \mathsf{xtag}$
end for

for the adversary to guess sk_j due to the intractability of the discrete logarithm problem. Note that, even if the adversary knows both $g_1^{sk_j}$ and $g_1^{r_j}$, it cannot guess $g_1^{sk_j \cdot r_j}$ due the DDH assumption. Therefore, our DMSSE scheme is secure against malicious clients.

Algorithm 11 Simulator code to generate **t**.

for $t \in [T]$ **do**
  $pk_i \leftarrow \mathsf{CPK}[t]$; $r \leftarrow \mathbf{R}[i]$
  for $\alpha \in [n]$ **do**
    $\mathrm{xtoken}_i[\alpha] \leftarrow pk_i^{r_i \cdot \mathbf{Z}[\bar{\mathbf{s}}[t]] \cdot H[\tilde{\mathbf{x}}[t,\alpha]]}$
  end for
end for

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.ins.2019.08.014.

References

[1] L. Ballard, S. Kamara, F. Monrose, Achieving efficient conjunctive keyword searches over encrypted data, in: Proc. of ICICS, Springer, 2005, pp. 414–426.
[2] F. Bao, R.H. Deng, X. Ding, Y. Yang, Private query on encrypted data in multi-user settings, in: Proc. of ISPEC, 2008.
[3] M. Bellare, A. Desai, E. Jokipii, P. Rogaway, A concrete security treatment of symmetric encryption, in: Proc. of FOCS, 1997, pp. 394–403.
[4] X. Boyen, The uber-assumption family, in: Proc. of Pairing, 2008, pp. 39–56.
[5] J.W. Byun, D.H. Lee, J. Lim, Efficient conjunctive keyword search on encrypted data storage system, in: Proc. of EuroPKI, 2006, pp. 184–196.
[6] D. Cash, S. Jarecki, C. Jutla, H. Krawczyk, M.-C. Roşu, M. Steiner, Highly-scalable searchable symmetric encryption with support for Boolean queries, in: Proc. of CRYPTO, Springer, 2013, pp. 353–373.
[7] R. Curtmola, J. Garay, S. Kamara, R. Ostrovsky, Searchable symmetric encryption: improved definitions and efficient constructions, in: Proc. of CCS, 2006, pp. 79–88.
[8] S. Faber, S. Jarecki, H. Krawczyk, Q. Nguyen, M. Rosu, M. Steiner, Rich queries on encrypted data: beyond exact matches, in: Proc. of ESORICS, 2015, pp. 123–145.
[9] B.A. Fisch, B. Vo, F. Krell, A. Kumarasubramanian, V. Kolesnikov, T. Malkin, S.M. Bellovin, Malicious-client security in blind seer: a scalable private DBMS, in: Proc. of S&P, 2015, pp. 395–410.
[10] E.-J. Goh, et al., Secure indexes, IACR Cryptol. ePrint Arch. 2003 (2003) 216.
[11] P. Golle, J. Staddon, B. Waters, Secure conjunctive keyword search over encrypted data, in: Proc. of ACNS, 2004, pp. 31–45.
[12] S. Jarecki, C. Jutla, H. Krawczyk, M. Rosu, M. Steiner, Outsourced symmetric private information retrieval, in: Proc. of CCS, 2013, pp. 875–888.
[13] X. Jiang, J. Yu, J. Yan, R. Hao, Enabling efficient and verifiable multi-keyword ranked search over encrypted cloud data, Inf. Sci. (Ny) 403–404 (2017) 22–41.
[14] S. Kamara, T. Moataz, Boolean searchable symmetric encryption with worst-case sub-linear complexity, in: Proc. of EUROCRYPT, 2017, pp. 94–124.
[15] A. Kasten, A. Scherp, F. Armknecht, M. Krause, Towards search on encrypted graph data, in: Proc. of ISWC, 2013.
[16] J. Katz, Y. Lindell, Introduction to modern cryptography, CRC Press, 2014.
[17] S. Lai, S. Patranabis, A. Sakzad, J.K. Liu, D. Mukhopadhyay, R. Steinfeld, S. Sun, D. Liu, C. Zuo, Result pattern hiding searchable encryption for conjunctive queries, Proc. CCS (2018) 745–762.
[18] M. Li, S. Yu, N. Cao, W. Lou, Authorized private keyword search over encrypted personal health records in cloud computing, in: Proc. of ICDCS, 2011, pp. 383–392.
[19] R. Li, A.X. Liu, Adaptively secure conjunctive query processing over encrypted data for cloud computing, in: Proc. of ICDE, IEEE, 2017, pp. 697–708.
[20] Q. Liu, Y. Guo, J. Wu, G. Wang, Effective query grouping strategy in clouds, J. Comput. Sci. Technol. 32 (6) (2017) 1231–1249.
[21] Q. Liu, P. Hou, G. Wang, T. Peng, S. Zhang, Intelligent route planning on large road networks with efficiency and privacy, J. Parallel Distrib. Comput. (2019). DOI: 10.1016/j.jpdc.2019.06.012
[22] Q. Liu, Y. Tian, J. Wu, T. Peng, G. Wang, Enabling verifiable and dynamic ranked search over outsourced data, IEEE Trans. Serv. Comput. (2019). DOI: 10.1109/TSC.2019.2922177
[23] Q. Liu, G. Wang, F. Li, S. Yang, J. Wu, Preserving privacy with probabilistic indistinguishability in weighted social networks, IEEE Trans. Parallel Distrib. Syst. 28 (5) (2017) 1417–1429.
[24] Q. Liu, G. Wang, X. Liu, T. Peng, J. Wu, Achieving reliable and secure services in cloud computing environments, Comp. Electric. Eng. 59 (2017) 153–164.
[25] X. Liu, G. Yang, Y. Mu, R. Deng, Multi-user verifiable searchable symmetric encryption for cloud storage, IEEE Trans. Dependable Secure Comput. (2018), doi:10.1109/TDSC.2018.2876831.
[26] X. Liu, Q. Liu, T. Peng, J. Wu, Dynamic access policy in cloud-based personal health record (PHR) systems, Inf. Sci. (Ny) 379 (2017) 62–81.
[27] V. Pappas, F. Krell, B. Vo, V. Kolesnikov, T. Malkin, S.G. Choi, W. George, A. Keromytis, S. Bellovin, Blind seer: a scalable private dbms, in: S&P, 2014, pp. 359–374.
[28] T. Peng, Q. Liu, B. Hu, J. Liu, J. Zhu, Dynamic keyword search with hierarchical attributes in cloud computing, IEEE Access 6 (2018) 68948–68960.
[29] D.X. Song, D. Wagner, A. Perrig, Practical techniques for searches on encrypted data, in: Proc. of S&P, 2000, pp. 44–55.
[30] S.-F. Sun, J.K. Liu, A. Sakzad, R. Steinfeld, T.H. Yuen, An efficient non-interactive multi-client searchable encryption with support for Boolean queries, in: Proc. of ESORICS, 2016, pp. 154–172.
[31] Q. Tang, Nothing is for free: security in searching shared and encrypted data, IEEE Trans. Inf. Forensics Secur. 9 (11) (2014) 1943–1952.
[32] J. Wang, X. Chen, S.-F. Sun, J.K. Liu, M.H. Au, Z.-H. Zhan, Towards efficient verifiable conjunctive keyword search for large encrypted database, in: Proc. of ESORICS, 2018, pp. 83–100.
[33] Y. Wu, H. Huang, Q. Wu, A. Liu, T. Wang, A risk defense method based on microscopic state prediction with partial information observations in social networks, J. Parallel Distrib. Comput. 131 (2019) 189–199.
[34] M. Xiao, J. Wu, L. Huang, Community-aware opportunistic routing in mobile social networks, IEEE Trans. Comp. 63 (7) (2014) 1682–1695.
[35] S. Zhang, X. Mao, K.-K.R. Choo, T. Peng, G. Wang, A trajectory privacy-preserving scheme based on a dual-k mechanism for continuous location-based services, Inf. Sci. (Ny) (2019), doi:10.1016/j.ins.2019.05.054.
[36] S. Zhang, Q. Liu, Y. Lin, Anonymizing popularity in online social network with full utility, Future Generat. Comp. Syst. 72 (7) (2017) 227–238.
[37] Q. Zheng, S. Xu, G. Ateniese, VABKS: verifiable attribute-based keyword search over outsourced encrypted data, in: Proc. of INFOCOM, 2014, pp. 522–530.