This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2020.2975175, IEEE Transactions on Cloud Computing

Towards Secure and Efficient Equality Conjunction Search over Outsourced Databases

Weipeng Lin, Ke Wang, Zhilin Zhang, Ada Waichee Fu, Raymond Chi-Wing Wong, Cheng Long and Chunyan Miao

Abstract—Searchable symmetric encryption enables a cloud server to answer queries directly over encrypted data. Two key requirements are a strong security guarantee and a sub-linear search performance. The bucketization approach in the literature addresses these requirements at the expense of downloading false positives and requiring the local search at the client side. In this work, we propose a novel approach to meet these requirements while minimizing the clients work and communication cost. First, a relaxed notion of ciphertext indistinguishability on partitioned data is formalized, called class indistinguishability, which provides a level of ciphertext indistinguishability similar to that of bucketization but allows the server to perform search of relevant data and filter false positives. We present a construction for achieving these goals through a two-phase search algorithm. The first phase finds a candidate set through a sub-linear search. The second phase finds the exact query result using a linear search applied to the candidate set. The experiment results on large real-world data-sets show that our approach outperforms the state-of-the-art. This work focuses on the class of equality conjunction search, but it applies to the general class of Boolean queries of equalities because the latter can be reduced to several equality conjunction queries.

Index Terms—Symmetric Searchable Encryption, Equality Conjunction Search, Sub-linear Search Performance.

1 Introduction

The current trend towards cloud-based Database-as-a-Service (DaaS) as an alternative to traditional on-site relational database management systems has largely been driven by the perceived simplicity and cost-effectiveness. While moving data from local devices to the cloud server offers great convenience to the clients, outsourced data are under threat from being accessed or used by unauthorized parties (including the server) for their own benefits without the client's knowledge. Outsourcing encrypted data can preserve privacy but preclude the client from delegating query processing tasks to the server. A promising solution to this problem is *symmetric searchable encryption (SSE)* [1] that allows the server to answer queries directly over encrypted data on the client's behalf while protecting the confidentiality of plaintext data and queries.

A typical SSE system is shown in Figure 1. At the setup time, the trusted client builds an encrypted index and outsources it to the server. Later on, the client can issue a query by generating the encrypted form of the query, called the trapdoor. The server is responsible for computing the query result using the trapdoor and the encrypted index. During this interaction, the server is considered as "honest-but-curious": it will follow the protocol honestly but may passively attempt to learn the content of encrypted data and queries. For this reason, the server is semi-trusted and is treated as the adversary.

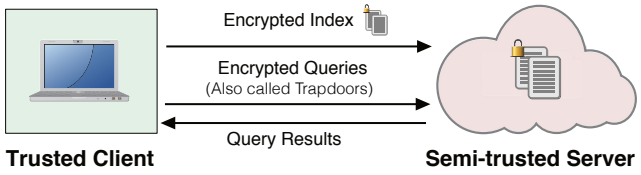


Fig. 1: A typical SSE system

1.1 Motivation

The core of SSE is to meet three design goals: a strong security guarantee, efficient search performance, and supporting a large class of queries. An important line of research (e.g. [2], [3], [4], [5]) has been to realize practical SSE schemes that achieve *ciphertext indistinguishability* [2] by restricting to single keyword queries or single equality queries. In many practical applications, however, such simple types of query is insufficient to provide a good search experience. For example, suppose an authorized doctor in the hospital outsources the following relational database of patients to the server,

```
PATIENT (name,sex,age,city,country,disease),
```

and wants to retrieve medical records for all patients who are diagnosed with HIV and live in Vancouver. In this case, simply performing search for each equality in the query and then intersecting the results of the two queries often leads to inefficient and unsecured query processing [6]. A sophisticated system should support authorized doctors to retrieve data through a single query defined by an *equality conjunction query* such as

- *W. Lin is with the School of Artificial Intelligence, Shenzhen Polytechnic China. (The work done in SFU) E-mail: weipengl@spzt.edu.cn*
- *K. Wang and Z. Zhang are with the School of Computing Science, Simon Fraser University, Canada. E-mail: {ke_wang,zhilinz}@sfu.ca*
- *A. Fu is with Chinese University of Hong Kong, Hong Kong, China.*
- *R. Wong is with Hong Kong University of Science and Technology, Hong Kong, China.*
- *C. Long and C. Miao are with Nanyang Technological University, Singapore, Singapore*

Manuscript received June 30, 2019.

2168-7161 (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: Hong Kong University of Science and Technology. Downloaded on April 17,2022 at 10:16:44 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2020.2975175, IEEE Transactions on Cloud Computing


```
SELECT * FROM PATIENT as P,
WHERE P.city = Vancouver AND P.disease = HIV.
```

The above example raises the following question:

Can we design a practical SSE scheme that meets both ciphertext indistinguishability and sub-linear search performance for equality conjunction search, or more generally, for the class of all equality Boolean search?

Challenge and perspectives. A grand challenge is that these two requirements are in conflict with each other. On the one hand, ciphertext indistinguishability requires that the adversary (i.e., the server) cannot distinguish encrypted records by observing their ciphertexts. On the other hand, a sub-linear search on the server side entails distinguishing the encrypted records that do not need to be searched from those that do, which leaks significantly more information about data and queries to the server than what is allowed to be leaked by ciphertext indistinguishability. More importantly, it is in general difficult to capture the full extent of such search related low-level disclosures in the security definition (see Section 3 for more details).

We argue that the strong security level required by ciphertext indistinguishability is not always necessary. In many practical scenarios, it suffices to maintain indistinguishability among a number of individuals instead of all. For example, widely used k-*anonymity* [7] only requires that each individual cannot be distinguished from $(k-1)$ other individuals, where k is a security parameter. Another example is the *bucketization* approach ([8], [9], [10]), which only provides indistingushiability within each bucket. In other cases, indistinguishability is needed only for those who care about it. For example, if $Alice$ and Bob care about indistinguishability between them, but Cat and Dog do not, it suffices to partition the domain into three classes $g_0 = \{Cat\}, g_1 = \{Dog\}, g_2 = \{Alice, Bob\}$ and enforce ciphertext indistinguishability *within each class*. In general, the class partitioning is over the joint domain of several attributes. The purpose of this class-level ciphertext indistinguishability is to enable sub-linear performance by pruning irrelevant classes for a given query, which is possible because there is no indistinguishability requirement on data from different classes.

The bucketization approach ([8], [9], [10]) enforces ciphertext indistinguishability in the class level by partitioning the records in the database into buckets (i.e., classes in our work) according to some specified partitioning of the domain of each attribute, and each bucket is assigned a unique bucket id. The plaintext records are encrypted using traditional techniques and stored on the server. To retrieve the data requested by a query, the records in a bucket are identified using the bucket id and the client first maps the query condition to the relevant bucket ids using a local index and submits such bucket ids to the server. The server returns encrypted data according to the received bucket ids. The client then recovers the query result by decrypting returned data and filtering false positives. Sub-linear search performance is supported by retrieving only the data in the relevant buckets for a query.

However, the bucketization approach suffers from two main limitations. One is that the client needs to search for relevant bucket ids for a query locally, referred to as query translation processing in [8], [9], [10]. This imposes the overhead on the client to store and maintain the translation information of all buckets for dynamic data, which is non-trivial when data is partitioned by multi-attributes. On the other hand, the resource-rich server is under-utilized in that it only takes charge of retrieving encrypted data according to the given bucket ids computed by the client. Another limitation is that false positives have to be communicated to the client because they can only be filtered by the client. These two limitations are indicated by grey boxes named "Search" and "Filtering" on the client side in Figure 2. A finer bucket granularity will increase client's search work due to the increased number of buckets (especially for multi-dimensional data), whereas a coarser bucket granularity will increase the number of false positives and the communication cost. With client's resources and network bandwidth typically being the bottleneck, this approach's application will be limited.

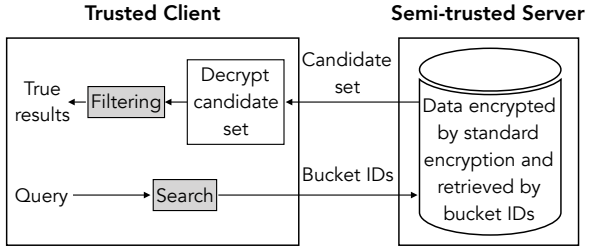


Fig. 2: Bucketization [8] [9] [10]: The client searches for relevant bucket ids, the server returns all records in the buckets, and the client filters false positives

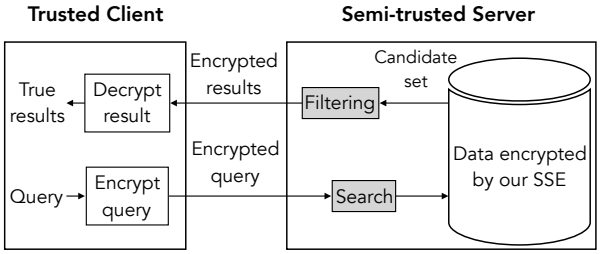


Fig. 3: Proposed scheme: The client encrypts the query predicate, the server searches for a candidate set and filters false positives, and the client decrypts the query result

1.2 Contributions

A preferred solution is pushing the "Search" and "Filtering" tasks to the server as shown in Figure 3, where the client only needs to encrypt the query and decrypt the query result. In this work, we present a novel scheme to meet these requirements. We focus on the class of equality conjunction queries, such as the query on the "PATIENT" table; the more general class of Boolean queries of equalities can be reduced to such queries. Our contributions are summarized as follows.

- The motivation of our choice of SSE schemes, we begin with an analysis of two types of SSE

2168-7161 (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: Hong Kong University of Science and Technology. Downloaded on April 17,2022 at 10:16:44 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2020.2975175, IEEE Transactions on Cloud Computing

schemes, namely plaintext-indexing schemes and ciphertext-indexing schemes. The analysis suggests that ciphertext-indexing schemes can better meet our security and performance goals. Therefore, for the rest of the paper, we consider ciphertext-indexing schemes.

- We formalize a relaxed notion of ciphertext indistinguishability, called *class indistinguishability*, that achieves a level of indistinguishability similar to that of bucketization, but enables the design in Figure 3.
- We propose a novel ciphertext-indexing SSE scheme, called CLASS, for computing equality conjunction queries while meeting class indistinguishability and supporting sub-linear search. Importantly, with CLASS we are able to push the search and filtering tasks to the server as in Figure 3, therefore, freeing the client from local search and maintenance work. Furthermore, CLASS can be implemented by plugging in existing indexing methods without designing specialized methods.
- We formally prove the class indistinguishability of CLASS and the resistance to common attacks based on auxiliary knowledge.
- We present an empirical study to evaluate the practical efficiency of CLASS on large and real life databases. The results show that CLASS outperforms the state-of-the-art.

2 PRELIMINARIES

This section introduces necessary notations and definitions.

Databases. We consider a relational database $\mathcal{D} = \{P_1, \cdots, P_{|\mathcal{D}|}\}$ containing $|\mathcal{D}|$ records with d attributes $\{A_1, \cdots, A_d\}$. Each A_t has a discrete domain $dom(A_t)$. A numeric domain can be discretized into a small number of intervals. Besides $\{A_1, \cdots, A_d\}$, the database may contain other attributes that do not occur in any query.

Queries. An *equality conjunction query* has the form $e_1 \wedge \cdots \wedge e_q$, where e_t, $1 \leq t \leq q$, is an equality $A_t = v_t$ with $v_t \in dom(A_t)$ for a distinct attribute A_t. An *equality disjunction query* has the form $E_1 \wedge \cdots \wedge E_q$, where each E_t is a disjunction of equalities on an attribute A_t. This equalty disjunction query can be rewritten into a collection of equality conjunction queries that do not overlap in results. Moreover, an inequalty $A_t <> v_t$ can be rewritten into a disjunction of equalities on A_t. Therefore, any Boolean query of equalites can be reduced to a collection of equality conjunction queries. For this reason, we shall focus on the class of equality conjunction queries in the rest of the paper.

Symmetric Searchable Encryption (SSE). We adopt the formal definition of SSE from [2] as shown in Definition 1. Like most prior work [2], [6], [11], we focus on concealing plaintext data and queries but allow the disclosure of "access pattern" (i.e., what encrypted records are retrieved by a search query) and "search pattern" (i.e., whether two search queries are identical) as a result of granting the server the search capacity. Please refer to [12], [13], [14], [15], [16] for more details on access and search patterns hiding.

Definition 1 (SSE scheme [2]). A searchable symmetric encryption (SSE) scheme is a collection of four polynomial-time algorithms $\mathsf{SSE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Trpdr}, \mathsf{Search})$ such that,

$K \leftarrow \mathsf{Gen}(1^\psi)$: A probabilistic algorithm takes as input a security parameter ψ and outputs a secret key K.

$(\mathbf{I}, \mathbf{c}) \leftarrow \mathsf{Enc}(K, \mathcal{D})$: A probabilistic algorithm takes as input a secret key K and a database $\mathcal{D}$. The output contains encrypted data $\mathbf{c}$ and an encrypted index $\mathbf{I}$.

$\mathbf{T} \leftarrow \mathsf{Trpdr}(K, Q)$: An algorithm takes as input a secret key K and a query Q. The output is a query trapdoor $\mathbf{T}$.

$\mathcal{E} \leftarrow \mathsf{Search}(\mathbf{I}, \mathbf{T})$: A deterministic algorithm takes as input an encrypted index $\mathbf{I}$ and a query trapdoor $\mathbf{T}$. The output is a set of data IDs $\mathcal{E}$.

An SSE scheme is correct if for all $\psi \in \mathbb{N}$, for all K output by $\mathsf{Gen}(1^\psi)$, for all $\mathcal{D} \subseteq \prod_{t=1}^{d} dom(A_t)$, for all $\mathbf{I}$ output by $\mathsf{Enc}(K, \mathcal{D})$, for all queries Q, and $\mathbf{T}$ output by $\mathsf{Trpdr}(K, Q)$, the output of $\mathsf{Search}(\mathbf{I}, \mathbf{T})$ is the set of IDs for the records in $\mathcal{D}$ satisfying Q. □

Notation	Description
ψ	security parameter
$negl(\psi)$	negligible function in security parameter ψ
K	secret key
d	number of potential querying attributes
A_t	an attribute in $\{A_1, \cdots, A_d\}$
$dom(A_t)$	domain of an attribute A_t
$\mathcal{D}$	a database $\mathcal{D}$ contains $\|\mathcal{D}\|$ records
RID	unique record ID for each record in a database $\mathcal{D}$
P_i and $E(P_i)$	record and encryption
Q_j and $E(Q_j)$	query and encryption
$Att(Q_j)$	attributes where a query Q_j has an equality
$RID(\mathcal{D}, Q_j)$	IDs for the records in $\mathcal{D}$ satisfying a query Q_j
$\mathbf{I}$	encrypted index for a database $\mathcal{D}$
$\mathbf{T}$	query trapdoor for a query
κ	class size for an attribute A_t
$\{g_0^t, \cdots, g_{l-1}^t\}$	class partitioning for an attribute A_t

TABLE 1: Summary of notations

$\mathbf{c}$ can be generated by any traditional encryption method such as AES [17] because it is not involved in the search function (i.e., $\mathsf{Search}(\mathbf{I}, \mathbf{T})$). We shall focus on building the encrypted index $\mathbf{I}$ and call it the ciphertext, not to be confused with $\mathbf{c}$. An SSE scheme is dynamic if $\mathbf{I}$ can be maintained for data update by the server.

In the rest of the paper, $\mathsf{Enc}(K, P_i)$ denotes the encryption function for a record P_i and $E(P_i)$ denotes the ciphertexts generated by the encryption function. We use $x \xleftarrow{\$} X$ to denote that x is sampled uniformly at random from a finite set X, and use $|X|$ to represent the number of elements in X. $x \leftarrow \mathcal{A}$ means that x is the output of an algorithm $\mathcal{A}$. Table 1 summarizes some frequently used notations.

3 TWO TYPES OF SSE

The construction of the encrypted index $\mathbf{I}$ involves two requisite stages for supporting a sub-linear search: encryption and index building. In this section, we analyze two different orders of these stages, which motivates our choice of how to construct the encrypted index $\mathbf{I}$.

2168-7161 (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: Hong Kong University of Science and Technology. Downloaded on April 17,2022 at 10:16:44 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2020.2975175, IEEE Transactions on Cloud Computing

3.1 Plaintext-Indexing

We say that an SSE scheme is *plaintext-indexing* if $\mathbf{I}$ is generated by first building an index over plaintext database and then encrypting the index. This index must be built by the client because it requires accessing plaintext records, as shown in Figure 4. Most existing SSE schemes that support a sub-linear search are plaintext-indexing, such as PB-Tree [18] and $\hat{R}$-Tree [19] for range queries, Oblivious Cross-Tags Protocol (OXT) [6], IXE [20] and IBTree [21] for keyword search. A drawback of plaintext-indexing is that the client has to maintain the index information locally for dynamic data. Another drawback is that a sub-linear search using such $\mathbf{I}$ tends to disclose *low-level* information beyond the query result and such disclosures are hard to capture.

For concreteness, consider the OXT scheme [6], which is the first SSE scheme to support conjunctive keyword queries with sub-linear search complexity. The encrypted index $\mathbf{I}$ of OXT is essentially the inverted index built over the plaintext database but hiding document pointers using specialized structures called *TSet* and *XSet*. To compute a query, the server needs to scan the inverted list of the s-term (i.e., the search keyword with the shortest inverted list) using TSet and for each document id on the list checks if it is found on the lists of all x-terms (i.e., the remaining search keywords) using XSet. This process discloses the following information for 2-keyword queries [6]: for two queries that share the same x-term, the server learns the documents containing both s-terms of the two queries, called *conditional intersection pattern* (IP); for each query, OXT leaks the x-terms that match the initial result from the list for the s-term, called *x-term matching pattern* (XMP).

The above disclosure is due to plaintext-indexing and the sub-linear search requirement. In particular, such indexes are built using the relationships between plaintext records; even though this index is encrypted, the sub-linear performance requires that the encrypted index provides certain information useful for pruning irrelevant data during the search for a query. Therefore, such disclosures are inevitable for any encryption scheme that supports plaintext-indexing and the sub-linear requirements. This can also be seen from the leakage function in the security definition used by OXT. Moreover, since such disclosures are related to the sub-linear search process, it is difficult to capture the full extent of such low-level disclosures.

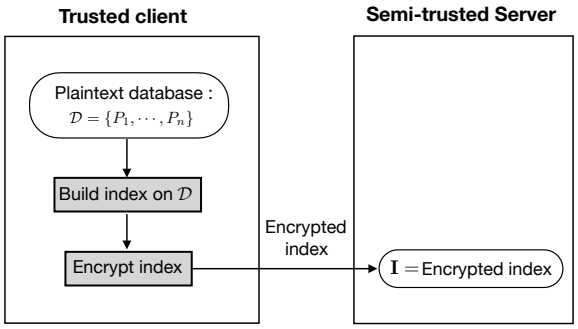


Fig. 4: Plaintext-Indexing SSE: the client first builds the index for the plaintext database $\mathcal{D}$, then encrypts the index and uploads the encrypted index as $\mathbf{I}$ to the server

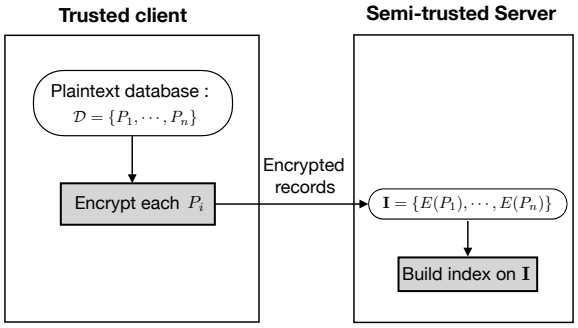


Fig. 5: Ciphertext-Indexing SSE: the client first encrypts each record P_i and then uploads the encrypted records $\{E(P_1), \cdots, E(P_n)\}$ as $\mathbf{I}$; the server can build the index using $\mathbf{I}$

3.2 Ciphertext-Indexing

We say that an SSE scheme is *ciphertext-indexing* if $\mathbf{I}$ is generated by encrypting each plaintext record in the database individually, denoted by $\mathbf{I} = \{E(P_1), \cdots, E(P_n)\}$, and then building any index structure based on $\mathbf{I}$ as shown in Figure 5. This index can be built by the server given $\mathbf{I}$. Some examples of ciphertext-indexing schemes are [11], [22], [23], [24]. To insert a record P_j, the client just needs to upload encrypted record $E(P_j)$ for the new record P_j, instead of uploading an entire updated index as in the case of Figure 4, and the server will update the index structure by inserting $E(P_j)$ into it.

Therefore, unlike a plaintext-indexing SSE, a ciphertext-indexing SSE is suitable for dynamic data, and more importantly, any sub-linear search using such indexes disclose no more information than $\{E(P_1), \cdots, E(P_n)\}$. Consequently, it suffices to focus on the disclosure of encrypted records, rather than dealing with low-level disclosures arising from a sub-linear search process, which hugely simplifies the threat model and security definition. In particular, there is no need for a leakage function to capture the search process's disclosure as in the case of OXT.

Despite the above mentioned nice properties of ciphertext-indexing SSE, existing ciphertext-indexing SSE schemes support only a linear search which are not scalable for large databases. Indeed, designing a ciphertext-indexing SSE that supports the sub-linear search is a nontrivial task due to the following dilemma: the security consideration requires that encrypted records $E(P_i)$ provide little information about P_i, whereas the sub-linear performance consideration requires that the index structure built using $E(P_i)$ enables the pruning of irrelevant data for a query, which tends to disclose some information about data and query. In the rest of the paper, we propose a relaxed notion of ciphertext indistinguishability and present a ciphertext-indexing SSE scheme to support sub-linear search. We start with our security definition.

2168-7161 (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: Hong Kong University of Science and Technology. Downloaded on April 17,2022 at 10:16:44 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2020.2975175, IEEE Transactions on Cloud Computing

4 Proposed Security

In this section, we formalize a relaxed notion of ciphertext indistinguishability, called class indistinguishability, to maintain the indistinguishability among the members in a class who care about indistinguishability. Since no indistinguishability are required for different classes, class indistinguishability allows the server to perform sub-linear search by pruning irrelevant classes. Before going ahead, we first give the definition of classes used later.

4.1 Classes

We assume that for an attribute A_t ($1 \leq t \leq d$), the domain of A_t is partitioned into disjoint *value classes*, $\{g_0^t, \cdots, g_{l_t-1}^t\}$. Typically, the class partitioning for each attribute is specified by the data owner based on indistinguishability required for the members in each class (as the example shown in Section 1.1). For this reason, we assume that the class partitioning for each attribute is given in the following definition. Section 5.3 will discuss how to construct the class partitioning if the data owner has no preference.

Definition 2 (Classes). Let $\{g_0^t, \cdots, g_{l_t-1}^t\}$ be the class partitioning for A_t, $1 \leq t \leq d$.

- A *record class* consists of all records such that for any two records P_i and P_i' in the class, $P_i[t]$ and $P_i[t]'$ are in the same value class for every attribute A_t.
- A *database class* consists of all databases $\mathcal{D}$ such that for any database $\mathcal{D}'$ in the class, there is a bijection η from $\mathcal{D}$ to $\mathcal{D}'$ such that for each record P_i in $\mathcal{D}$, P_i and $\eta(P_i)$ are in the same record class.
- A *query class* consists of all queries Q such that for any query Q' in the class, $Att(Q) = Att(Q')$ and for each $A_t \in Att(Q)$, $Q[t]$ and $Q'[t]$ are in the same value class.
- A *history class* consists of all histories $H = (\mathcal{D}, \mathcal{Q} = \{Q_1, \cdots, Q_m\})$ such that for any history $H' = (\mathcal{D}', \mathcal{Q}' = \{Q_1', \cdots, Q_m'\})$ in the class, $\mathcal{D}$ and $\mathcal{D}'$ are in the same database class, and for $1 \leq j \leq m$, Q_j and Q_j' are in the same query class. □

Intuitively, a database class consists of all databases obtained by replacing each record with a record from the same record class; a query class consists of all queries obtained by replacing each specified value with a value from the same value class; a history class consists of all histories obtained by replacing the database with a database from the same database class and replacing each query with a query from the same query class. We give the following example to better illustrate record (database, query, history) classes.

Example 1. Let $dom(A_1) = \{a_1, a_2, a_3, a_4\}$ and $dom(A_2) = \{b_1, b_2, b_3, b_4\}$. Assume that the domain of A_1 and A_2 are partitioned as follows to form four value classes,

$$A_1 : g_0^1 = \{a_1, a_4\}, g_1^1 = \{a_2, a_3\},$$

$$A_2 : g_0^2 = \{b_1, b_3\}, g_1^2 = \{b_2, b_4\}.$$

Consider two databases $\mathcal{D}$ as

$$\{P_1 = (a_1, b_1), P_2 = (a_4, b_2), P_3 = (a_2, b_3), P_4 = (a_2, b_3)\}$$

and $\mathcal{D}'$ as

$$\{P_1' = (a_4, b_3), P_2' = (a_4, b_4), P_3' = (a_3, b_1), P_4' = (a_2, b_1)\}$$

where A_1 and A_2 are in the first and second positions in each record. Let η be the bijection from $\mathcal{D}$ to $\mathcal{D}'$ such that $\eta(P_i) = P_i'$. Because $P_i[t]$ and $P_i'[t]$ are in the same value class for every attribute A_t, P_i and P_i' are in the same record class. Consequently, $\mathcal{D}$ and $\mathcal{D}'$ are in the same database class. Similarly, the queries $Q_1 = (a_1, -)$ and $Q_1' = (a_4, -)$ are in the same query class, where a dash means no condition on the attribute. Let $H_1 = \{\mathcal{D}, \{Q_1\}\}$ and $H_2 = \{\mathcal{D}', \{Q_1'\}\}$. It is clearly that H_1 and H_2 are in the same history class. □

In the rest of discussion, whenever it is clear from the context, we use the term "class" for any of record class, database class, query class, and history class.

4.2 Class Indistinguishability

Our security definition, called class indistinguishability, is the restriction of ciphertext indistinguishability to only the members from the same class of histories. A standard approach to formalize a security definition in SSE is defining a probabilistic game. We adopt the probabilistic game from [2] to enforce the ciphertext indistinguishability in each history class. First, we borrow the notion of "trace" for a history from [2].

Definition 3 (The trace of a history [2]). Given a history $H = (\mathcal{D}, \mathcal{Q} = \{Q_1, \cdots, Q_m\})$,

- *Access pattern* induced by H is the tuple $\alpha(H) = (RID(\mathcal{D}, Q_1), \cdots, RID(\mathcal{D}, Q_m))$;
- *Search pattern* induced by H is the symmetric binary matrix $\sigma(H)$ such that for $1 \leq i, j \leq m$, the element in the i-th row and j-th column is 1 if $Q_i = Q_j$, and 0, otherwise;
- *Trace* induced by H is $\tau(H) = (|\mathcal{D}|, \alpha(H), \sigma(H))$.

Two histories H_0 and H_1 have the same trace if there is a renaming ρ of RIDs such that $|\mathcal{D}_0| = |\mathcal{D}_1|, \alpha(H_0) = \rho(\alpha(H_1)), \sigma(H_0) = \sigma(H_1)$ (The renaming of RID is used to ignore any difference in the choices of RIDs for the two databases).

The trace of a history captures all the information about the history that is allowed to be leaked to the server. Then, we can formalize the class indistinguishability through a probabilistic game as follows.

Definition 4 (Class indistinguishability). Assume that the class partitioning $\{g_0^t, \cdots, g_{l-1}^t\}$ is given for every attribute A_t. Let $\mathsf{SSE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Trpdr}, \mathsf{Search})$. Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an adversary. Consider the following probabilistic game:

$\mathbf{Ind}_{\mathsf{SSE},\mathcal{A}}(\psi)$

1. $K \leftarrow \mathsf{Gen}(1^\psi)$
2. $(st_{\mathcal{A}}, H_0, H_1) \leftarrow \mathcal{A}_1(1^\psi)$
3. $b \xleftarrow{\$} \{0, 1\}$
4. parse H_b as $(\mathcal{D}_b, \mathcal{Q}_b)$
5. for $1 \leq i \leq n$
6. $E(P_{b,i}) \leftarrow \mathsf{Enc}(K, P_{b,i})$

2168-7161 (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: Hong Kong University of Science and Technology. Downloaded on April 17,2022 at 10:16:44 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2020.2975175, IEEE Transactions on Cloud Computing

7. let $\mathbf{I}_b = (E(P_{b,1}), \cdots, E(P_{b,n}))$
8. for $1 \leq j \leq m$
9. $\mathbf{T}_{b,j} \leftarrow \mathsf{Trpdr}(K, Q_{b,j})$
10. let $\mathbf{T}_b = (\mathbf{T}_{b,1}, \cdots, \mathbf{T}_{b,m})$
11. $b' \leftarrow \mathcal{A}_2(st_{\mathcal{A}}, \mathbf{I}_b, \mathbf{T}_b)$
12. if $b' = b$, output 1
13. otherwise output 0

subject to two restrictions: (i) H_0 and H_1 have the same trace, (ii) H_0 and H_1 are from the same class. $st_{\mathcal{A}}$ is a string that captures $\mathcal{A}_1$'s state after choosing the plaintext. We say that SSE ensures ***class indistinguishability*** if for all polynomial-size adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$,

$$|\mathsf{Pr}[\mathbf{Ind}_{\mathsf{SSE},\mathcal{A}}(\psi) = 1] - \frac{1}{2}| \leq negl(\psi) \quad (1)$$

where the probability is taken over the choice of b and the coins of Gen, Enc and Trpdr. We say that SSE ensures ***strict class indistinguishability*** if

$$\mathsf{Pr}[\mathbf{Ind}_{\mathsf{SSE},\mathcal{A}}(\psi) = 1] = \frac{1}{2}.\Box \quad (2)$$

In the above game, the adversary chooses two histories H_0 and H_1 (Line 2). Given a bit $b \in \{0, 1\}$ which is chosen uniformly at random (Line 3), the database $\mathcal{D}_b$ and queries $\mathcal{Q}_b$ in history H_b are encrypted to an encrypted index $\mathbf{I}_b$ and query trapdoors $\mathbf{T}_b$, respectively (Lines 4-10). After receiving $\mathbf{I}_b$ and $\mathbf{T}_b$, the adversary needs to guess the value of b (Line 11). The game outputs 1 if the adversary correctly guesses the value b (i.e., $b' = b$), otherwise, 0 (Lines 12-13). Let $\mathsf{Pr}[\mathbf{Ind}_{\mathsf{SSE},\mathcal{A}}(\psi) = 1]$ be the probability of the correct guess. Eqn (1) states that an SSE scheme satisfies ***class indistinguishability*** if no adversary can win the above probabilistic game with the probability significantly greater than an adversary who must guess randomly. Eqn (2) states that an SSE scheme satisfies ***strict class indistinguishability*** if the adversary's guess is a random guess.

The difference from the standard ciphertext indistinguishability in [2] is the additional condition (ii), which restricts H_0 and H_1 to be from the same history class; consequently, the indistinguishability holds only for the members from the same history class. Note that the above definition considers a non-adaptive adversary in that all queries are chosen by the adversary before receiving any encrypted data or queries. An adaptive adversary can choose the next query after receiving the encrypted records and encrypted queries for the previous queries. We will consider the adaptive adversary in Section 6.

Discussion. Class indistinguishability ensures that any two histories from the same class cannot be distinguished given their ciphertexts and the search result (captured by traces). The standard ciphertext indistinguishability is the extreme case of a single class containing all histories in the class, thus, providing the maximum level of indistinguishability. However, this single class leads to ineffective pruning in computing queries. Class indistinguishability offers a trade-off between the level of indistinguishability and the effectiveness of sub-linear search through a more general class partitioning for each attribute, because classes containing no query result will not not be searched. It is worth noting that knowing a record belonging to a class is not a privacy concern, but linking a ciphertext to an individual record is. Class indistinguishability addresses this concern by ciphertext indistinguishability between any two members from the same class. This notion requires non-singularity of a class but is independent of the size of a class because it considers two members at a time.

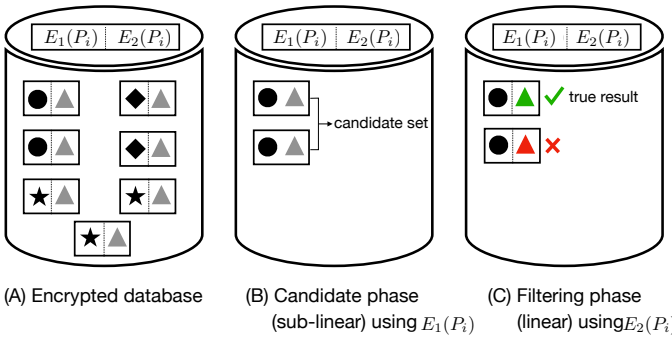


Fig. 6: Main idea of the two-phase search strategy

5 Construction

In this section, we construct a new ciphertext-indexing SSE, called Class, to meet class indistinguishability, and support a sub-linear search for equality conjunction queries, and in addition, pushing the tasks of searching for relevant data and filtering false positives to the server as in Figure 3.

5.1 Overview

At the high level, we want to achieve all these requirements through a two-phase search strategy. The ***Candidate Phase* (sub-linear phase)** focuses on pruning the sub-space not containing the query result to get a *candidate set* that may contain false positives. The ***Filtering Phase* (linear phase)** is applied to the candidate set to filter false positives. In general, the precision based search (i.e., false positive free) in the filtering phase is expensive. To reduce overhead, we want the candidate phase to be a lightweight sub-linear search that uses only standard operations (instead of crypto operations) and the candidate set must be small. Consequently, the efficiency comes from the fact that the candidate set in the filtering phase is smaller than the full database.

One solution to the above two-phase search is encrypting each record in the database by two different SSE schemes as shown in Figure 6(A). We assume that $E_1(P_i)$ is generated by a strict class indistinguishability SSE, and $E_2(P_i)$ is generated by a ciphertext indistinguishability SSE. Figure 6(B) shows that the candidate phase uses $E_1(P_i)$ can find the candidate set through a sub-linear search by pruning the classes (i.e., diamonds and stars) not containing any query result. Figure 6(C) illustrates that the filtering phase using $E_2(P_i)$ is a linear search process that only applies to the candidate set instead of the full space to find the true result.

2168-7161 (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: Hong Kong University of Science and Technology. Downloaded on April 17,2022 at 10:16:44 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2020.2975175, IEEE Transactions on Cloud Computing

Algorithm 1 $\mathsf{Search}(\mathbf{I}, \mathbf{T}_j)$

Require: The server has the encrypted index structure $\mathbf{I} = \{E(P_1), \cdots, E(P_{|\mathcal{D}|})\}$, and the query trapdoor $\mathbf{T}_j = (E_1(Q_j), E_2(Q_j))$

Candidate Phase:

$Cand \leftarrow \mathsf{Search}_1(\mathbf{I}, E_1(Q_j))$

Filtering Phase:

$Results \leftarrow \mathsf{Search}_2(Cand, E_2(Q_j))$

More formally, the proposed CLASS consists of two SSEs:

$$\mathsf{SSE}_1 = (\mathsf{Gen}_1, \mathsf{Enc}_1, \mathsf{Trpdr}_1, \mathsf{Search}_1),$$

$$\mathsf{SSE}_2 = (\mathsf{Gen}_2, \mathsf{Enc}_2, \mathsf{Trpdr}_2, \mathsf{Search}_2),$$

where we require that SSE_1 achieves strict class indistinguishability and SSE_2 satisfies ciphertext indistinguishability. The client encrypts each record P_i ($1 \le i \le |\mathcal{D}|$) in a relational database $\mathcal{D}$ into

$$E(P_i) = (E_1(P_i), E_2(P_i)),$$

where $E_b(P_i)$ ($b \in \{1, 2\}$) denotes the ciphertexts generated using the encryption function Enc_b in SSE_b. Then, the client builts the encrypted index structure $\mathbf{I}$ as

$$\mathbf{I} = (E(P_1), \cdots, E(P_{|\mathcal{D}|})).$$

At the query time the client encrypts a query Q_j into the query trapdoor

$$\mathbf{T}_j = (E_1(Q_j), E_2(Q_j)),$$

and submits $\mathbf{T}_j$ to the server where $E_b(Q_j)$ ($b \in \{1, 2\}$) denotes the trapdoor for the search query Q_j generated by the trapdoor generation function Trpdr_b in SSE_b. The search for the query answer proceeds in two phases described in Algorithm 1. These two phases correspond to the Search and Filtering in Figure 3, respectively. With a small *Cand*, any existing SSE achieving ciphertext indistinguishability with a linear search, such as [11], [22], [23], can serve as SSE_2. For this reason, our discussion in the following focuses on the construction of SSE_1.

5.2 Construction of *SSE*$_1$

We assume that for every attribute A_t, $1 \le t \le d$, the class partitioning $\{g_0^t, \cdots, g_{l-1}^t\}$ is given and the domain values in each class g_y^t are arranged in any order.

The intuition of our SSE_1 is modeling the equivalence of the domain values in the same class g_y^t by encoding each domain value into an angle and by the periodicity of circular functions sin and cos over such angles. In particular, we encode the domain value v at the x-th position in the class g_y^t by the angle ($1 \le x \le |g_y^t|$ and $0 \le y \le l-1$):

$$\alpha(v) = y\frac{\pi}{l} + (x-1)\pi. \tag{3}$$

The class label y determines the initial angle $y\frac{\pi}{l}$ for the class and each next value in the class adds an additional angle π. Note that $\alpha(v)$ depends on the assignment of class labels to classes and the order of values in a class, but any such assignment and order will do. Since any two values from the same class have the same first term $y\frac{\pi}{l}$, the next lemma follows immediately.

Algorithm 2 $\mathsf{Enc}_1(K_1, P_i)$

Require: • The client has the secret key $K_1 = (\mathbf{M})$

1) for $1 \le t \le d$

a) $\epsilon_{t,i} \xleftarrow{\$} [-U, -L] \cup [L, U]$, $(0 < L \le U)$

$$\begin{aligned} I_i[t]_1 &\leftarrow \epsilon_{t,i} \sin(\alpha_t(P_i[t])) \\ I_i[t]_2 &\leftarrow \epsilon_{t,i} \cos(\alpha_t(P_i[t])) \end{aligned} \tag{4}$$

2) $I_i \leftarrow (I_i[1]_1, I_i[1]_2, \cdots, I_i[d]_1, I_i[d]_2)$

3)

$$E_1(P_i) \leftarrow \frac{\mathbf{M}^{-1} I_i}{|\mathbf{M}^{-1} I_i|} \tag{5}$$

Algorithm 3 $\mathsf{Trpdr}_1(K_1, Q_j)$

Require: • The client has the secret key $K_1 = (\mathbf{M})$
• $Att(Q_j) \neq \varnothing$

1) for $1 \le t \le d$

a) if $A_t \in Att(Q_j)$

i) $\mu_{t,j} \xleftarrow{\$} [-U, -L] \cup [L, U]$, $(0 < L \le U)$

$$\begin{aligned} T_j[t]_1 &\leftarrow \mu_{t,j} \cos(\pi - \alpha_t(Q_j[t])) \\ T_j[t]_2 &\leftarrow \mu_{t,j} \sin(\pi - \alpha_t(Q_j[t])) \end{aligned} \tag{6}$$

b) if $A_t \notin Att(Q_j)$

i) $T_j[t]_1 = T_j[t]_2 \leftarrow 0$

2) $T_j \leftarrow (T_j[1]_1, T_j[1]_2, \cdots, T_j[d]_1, T_j[d]_2)$

3)

$$E_1(Q_j) \leftarrow \frac{\mathbf{M}^T T_j}{|\mathbf{M}^T T_j|} \tag{7}$$

Lemma 1. For any two values (v, v') in the domain of A_t, $\alpha(v) - \alpha(v')$ is a multiple of π if and only if v and v' are from the same class of A_t.

PROOF. The lemma holds because v and v' have the same class label y if and only if they are from the same value class. □

Below, we present detailed construct for each component of SSE_1, including secret key generation, encrypted index generation, query trapdoor generation and query evaluation.

5.2.1 Secret Key Generation

The key generation function $\mathsf{Gen}_1(1^{\psi_1})$ outputs the secret key $K_1 = (\mathbf{M})$, where $\mathbf{M}$ is a randomly chosen $(2d \times 2d)$ invertible matrix (i.e., $\mathbf{M}^{-1}\mathbf{M}$ is equal to the $(2d \times 2d)$ identity matrix). The key size ψ_1 is implicitly specified by the data dimensionality d. If necessary, dummy attributes can be added to increase d.

5.2.2 Encrypted Index Generation

The algorithm for generating $E_1(P_i)$ is given in Algorithm 2. Step 1 encodes each entry $P_i[t]$ into two values $I_i[t]_1$ and $I_i[t]_2$, where $\alpha(P_i[t])$ is the angle in Eqn (3) and $\epsilon_{t,i}$ is a noise randomly sampled from $[-U, -L] \cup [L, U]$

2168-7161 (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: Hong Kong University of Science and Technology. Downloaded on April 17,2022 at 10:16:44 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2020.2975175, IEEE Transactions on Cloud Computing

for t and i, $0 < L \leq U$. The effect of the parameters for randomness (i.e., L and U) will be discussed later on. Step 2 creates a randomized $2d$-dimensional vector I_i using $I_i[t]_1$ and $I_i[t]_2$ for all attributes. Step 3 "blends" all dimensions together using the secret matrix $\mathbf{M}$ and normalizes $E_1(P_i)$. Consequently, $E_1(P_i)$ is mapped as a point on the $2d$-dimensional unit sphere centered at the origin, and the location of the point is randomized by the random noises $\epsilon_{t,i}$ for each attribute A_t.

5.2.3 Query Trapdoor Generation

Algorithm 3 gives the algorithm for generating $E_1(Q_j)$. Step 1 encodes each specified $Q_j[t]$ (i.e., $A_t \in Att(Q_j)$) into two values $(T_j[t]_1, T_j[t]_2)$ using the angle $(\pi - \alpha_t(Q_j[t]))$, and encodes each unspecified $Q_j[t]$ (i.e., $A_t \notin Att(Q_j)$) into $(0, 0)$. Step 2 creates a randomized $2d$-dimensional vector T_j and Step 3 blends all dimensions together and produces $E_1(Q_j)$ as a randomized point on the $2d$-dimensional unit sphere centered at the origin. Note that T_j is not all zero because there is at least one condition in a query.

More on the random noises. It is worth noting that random noises ϵ in Eqn (4) and μ in Eqn (6) are used to unlink the ciphertext and the class of a record and a query, respectively. The interval size of the noise does not affect the size of *Cand* or the proof of class indistinguishability, but will affect the degree of unlinking the ciphertext and the class. In other words, even for the extreme case that $L = U$, the class indistinguishability still holds because the server cannot distinguish two histories from the same class. A larger interval $[-U, -L] \cup [L, U]$ $(0 < L \leq U)$ means more randomness of encrypted records and queries, which is more effective for thwarting statistic-based attacks but affects the effectiveness of sub-linear search for computing the candidate set *Cand*. Furthermore, in the case that $\mathcal{D}$ has a single attribute or Q_j has a single equality condition, the normalization step in Eqns (5) and (7) will cancel the effect of added noises. This problem can be fixed by adding a dummy attribute A^* with a single domain value v^* to every record and adding the equality $A^* = v^*$ to every query.

5.2.4 Searching for The Candidate Set

The search function Search_1 computes the candidate set of the query Q_j, denoted by *Cand*(Q_j), as the set of $E_2(P_i)$ such that $(E_1(P_i), E_2(P_i))$ is in $\mathbf{I}$ and $P_i[t]$ is in the same class as $Q_j[t]$ for every $A_t \in Att(Q_j)$. *Cand*(Q_j) contains the query result and possibly false positives. The next lemma gives the computation of *Cand*(Q_j). By "P_i is in *Cand*(Q_j)", we mean "$E_2(P_i)$ is in *Cand*(Q_j)".

Lemma 2. If P_i is in *Cand*(Q_j),

$$E_1(Q_j)^T E_1(P_i) = 0. \tag{8}$$

If P_i is not in *Cand*(Q_j), Eqn (8) holds with an exceedingly small probability.

PROOF. From Eqns (5) and (7), we have

$$E_1(Q_j)^T E_1(P_i) = \frac{T_j^T I_i}{|\mathbf{M}^T T_j||\mathbf{M}^{-1} I_i|}, \tag{9}$$

where the superscript T denotes a transpose operation. $E_1(Q_j)^T E_1(P_i) = 0$ holds if and only if $T_j^T I_i = \sum_{t=1}^{d}(I_i[t]_1 T_j[t]_1 + I_i[t]_2 T_j[t]_2) = 0$. Since $T_j[t]_1 = T_j[t]_2 = 0$ for all A_t which are not in $Att(Q_j)$, from Eqns (4) and (6), we have

$$\begin{aligned} T_j^T I_i &= \sum_{A_t \in Att(Q_j)} (I_i[t]_1 T_j[t]_1 + I_i[t]_2 T_j[t]_2) \\ &= \sum_{A_t \in Att(Q_j)} \epsilon_{t,i}\mu_{t,j} \sin(\pi + \alpha_t(P_i[t]) - \alpha_t(Q_j[t])). \end{aligned} \tag{10}$$

If P_i is in *Cand*(Q_j), $P_i[t]$ and $Q_j[t]$ are in the same class for every $A_t \in Att(Q_j)$, so $(\alpha_t(P_i[t]) - \alpha_t(Q_j[t]))$ is a multiple of π (Lemma 1) and $\sin(\pi + \alpha_t(P_i[t]) - \alpha_t(Q_j[t]) = 0$. In this case, Eqn (8) holds. If P_i is not in *Cand*(Q_j), $P_i[t]$ and $Q_j[t]$ are not in the same class for some $A_t \in Att(Q_j)$, and $(\alpha_t(P_i[t]) - \alpha_t(Q_j[t])$ is not a multiple of π (Lemma 1), so $\sin(\pi + \alpha_t(P_i[t]) - \alpha_t(Q_j[t]) \neq 0$. In this case, the chance that $T_j^T I_i = 0$ holds is small because noises ϵ's and μ's are randomly chosen. □

Example 2. Continue with Example 1. Let $\mathcal{D} = \{P_1 = (a_1, b_1), P_2 = (a_4, b_2), P_3 = (a_2, b_3), P_4 = (a_2, b_3)\}$ and consider three queries

$$Q_1 = (a_1, -),\ Q_2 = (a_2, b_3),\ Q_3 = (a_2, b_3),$$

where Q_2 and Q_3 are repeating queries. The next table lists (x, y) and $\alpha(v)$ for all domain values v:

	A_1				A_2			
v	a_1	a_2	a_3	a_4	b_1	b_2	b_3	b_4
(x, y)	(1,0)	(1,1)	(2,1)	(2,0)	(2,0)	(2,1)	(1,0)	(1,1)
$\alpha(v) = y\frac{\pi}{2} + (x-1)\pi$	0	$\frac{\pi}{2}$	$\frac{3}{2}\pi$	π	π	$\frac{3}{2}\pi$	0	$\frac{\pi}{2}$

For example, the (x, y) for domain value a_3 is (2,1) because a_3 is the second value in g_1. Consider the secret key $\mathbf{M}$ (i.e., a (4×4) invertible matrix)

$$\mathbf{M} = \begin{bmatrix} 10.538 & 0.319 & 3.579 & 0.725 \\ 1.833 & -1.308 & 2.769 & -0.063 \\ -2.259 & -0.434 & -1.350 & 0.714 \\ 0.862 & 0.343 & 3.035 & -0.205 \end{bmatrix}.$$

Algorithm 2 encrypts P_i into

$$E_1(P_1) = (-0.359, -0.309, 0.252, -0.844),$$
$$E_1(P_2) = (-0.446, -0.822, 0.199, -0.293),$$
$$E_1(P_3) = (-0.360, -0.124, 0.159, -0.911),$$
$$E_1(P_4) = (0.554, -0.370, -0.529, 0.526),$$

and Algorithm 3 encrypts Q_j into

$$E_1(Q_1) = (0.145, 0.086, 0.966, 0.196),$$
$$E_1(Q_2) = (-0.193, -0.735, 0.588, 0.279),$$
$$E_1(Q_3) = (0.687, -0.153, 0.699, -0.128).$$

The random noises $\epsilon_{t,i}$ and $\mu_{t,j}$ for $P_i[t]$ and $Q_j[t]$ are drawn from the interval $[-1100, -1000] \cup [1000, 1100]$ (i.e., $L = 1000$ and $U = 1100$). Note that probabilistically generated $E_1(Q_2)$ and $E_1(Q_3)$ are different even though Q_2 and Q_3 are identical. The following table shows the inner product $E_1(Q_j)^T E_1(P_i)$.

Due to the limited precision, a zero inner product is represented by a small value close to zero indicated by the gray cells. *Cand*(Q_1) contains $E_1(P_1)$ and

2168-7161 (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: Hong Kong University of Science and Technology. Downloaded on April 17,2022 at 10:16:44 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2020.2975175, IEEE Transactions on Cloud Computing

$\lvert E_1(Q_j)^T E_1(P_i)\rvert$	$E_1(P_1)$	$E_1(P_2)$	$E_1(P_3)$	$E_1(P_4)$
$E_1(Q_1)$	0.0006	0.0005	0.088	0.359
$E_1(Q_2)$	0.209	0.726	0.0000	0.0007
$E_1(Q_3)$	0.085	0.004	0.0005	0.0001

$E_1(P_2)$, and $Cand(Q_2)$ and $Cand(Q_3)$ contain $E_1(P_3)$ and $E_1(P_4)$. In an actual implementation with higher precision, $E_1(Q_j)^T E_1(P_i) = 0$ can be replaced by $|E_1(Q_j)^T E_1(P_i)| \leq c$ for a "small" positive value c. □

Below, we discuss several issues related to implementation and practical use.

Numeric unstability. Let $x = E_1(P_i)$ and $y = E_1(Q_j)$ be the exact theoretical values, and let x' and y' be the finite machine representation of x and y. These are $2d$-dimensional vectors, where d is the number of attributes in the database. For the t-th entry of these vectors, let $\Delta x[t]$ and $\Delta y[t]$ be the errors due to the machine representation, i.e., $x[t] = x'[t] + \Delta x[t]$ and $y[t] = y'[t] + \Delta y[t]$. Note that, when $xy = 0$, $x'y'$ is not always 0 (due to the limited machine representation) but will be a "small" value close to 0. Therefore, we can use a tight upper bound ub, i.e., the condition $x'y' \leq ub$, to find all $E_1(P_i)$ such that $E_1(Q_j)^T E_1(P_i) = 0$.

To find such ub, let $e(x, y) = |xy - x'y'|$ be the error of using $x'y'$ to approximate $E_1(Q_j)^T E_1(P_i)$.

$$
\begin{aligned}
e(x,y) =& |\sum_t (x'[t]\Delta y[t] + y'[t]\Delta x[t] + \Delta x[t]\Delta y[t])| \\
&\leq \sum_t (|x'[t]\Delta y[t]| + |y'[t]\Delta x[t]| + |\Delta x[t]\Delta y[t]|) \\
&\leq \sum_t (|\Delta y[t]| + |\Delta x[t]| + |\Delta x[t]\Delta y[t]|)
\end{aligned}
$$

The last inequality follows from $|x'[t]| \leq 1$ and $|y'[t]| \leq 1$ because $E_1(P_i)$ and $E_1(Q_j)$ are unit vectors. For the DOUBLE data type (other types are considered similarly), $x'[t]$ and $y'[t]$ have the precision of 10^{-15}, that is, $|\Delta x[t]| \leq 10^{-15}$, $|\Delta y[t]| \leq 10^{-15}$ and $\Delta x[t]\Delta y[t] \leq 10^{-15}$. Therefore, we have $e(x, y) < 6d * 10^{-15}$ because Δx and Δy are $2d$-dimensional vectors. So $ub = 6d * 10^{-15}$ serves the above upper bound on $x'y'$.

Transforming to hyperplane queries. From Lemma 2, the server can compute $Cand(Q_j)$ by the testing condition $E_1(Q_j)^T E_1(P_i) = 0$. Alternatively, the server can compute the *hyperplane query* defined by $E_1(Q_j)^T V = 0$ for a $2d$-dimensional point V, which retrieves all ciphertexts that are on the hyperplane defined by the norm $E_1(Q_j)$ and passing through the origin. In other words, the original query Q_j is transformed into a hyperplane query in the ciphertext space, which enables any existing sub-linear methods for hyperplane queries to be deployed by the server.

Search techniques for hyperplane queries are well studied, such as R-Tree [25], M-Tree [26], half-space queries [19] and scalar product queries [27]. In this work, we adopt M-Tree for hyperplane queries processing. Figure 7 shows an example. At the system setup, the server builds an M-Tree using the $E_1(P_i)$ portion of $E(P_i)$. Each node in the tree represents a sphere and leaf nodes store the encrypted records. Given a query hyperplane for Q_j (indicated by the red line), the search starts from the root and goes to the child nodes if the current node intersects the query hyperplane. On reaching a leaf node (i.e., nodes H and I), all stored points in the leaf node are tested against the query hyperplane. If a point is on the hyperplane, it is added to $Cand(Q_j)$. This search is sub-linear in that all the leaf nodes not reached in this search are ignored (i.e., nodes A to G).

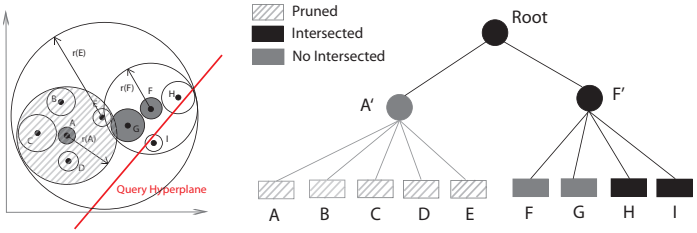


Fig. 7: Query hyperplane testing with M-Tree

Updating. The proposed SSE_1 in the above enjoys all the benefits discussed in Section 3.2 because it is ciphertext-indexing SSE (i.e., $\mathbf{I} = \{E(P_1), \cdots, E(P_n)\}$). In particular, the server can maintain the index structure on encrypted data for dynamic data easily. To insert a new record, the client only needs to encrypt the new record and outsource the encrypted data. Then the server can follow the standard insertion procedure for M-Tree to insert the encrypted data $E_1(P_i)$ into $\mathbf{I}$, by treating $E_1(P_i)$ as a regular record and treating $\mathbf{I}$ as a regular database. Deleting a record specified by RID follows the standard deletion operation on M-Tree as well. Alternatively, records can be deleted by specifying a condition on index attributes, which requires first locating the records satisfying the condition, like processing a regular query discussed above.

Comparison with bucketization. It is interesting to compare our approach with the bucketization approach [8] [9] [10]. Our candidate set is similar to the result retrieved using the bucket ids of the query in bucketization. However, there are several substantial differences. The first difference is that bucketization requires the client to perform local search of bucket ids for a query, whereas the client in our approach only needs to encrypt the query. Secondly, bucketization requires the client to filter false positives, whereas our approach filters false positives by the server (through SSE_2). Finally, bucket ids in bucketization are static, thus, directly tell what records are in the same bucket, whereas our encryption functions are probabilistic thanks to fresh random noises for each encryption.

5.3 Constructing Class Partitioning

While we expect that the class partitioning $\mathcal{X}_t = \{g_0^t, \cdots, g_{l-1}^t\}$ for an attribute A_t is specified by the data owner, the class partitioning can also be constructed to minimize a cost metric for a given class size $|g_y^t|$, for $1 \leq y \leq l-1$, which is useful if the data owner has no preference except that each class must have a minimum size. Below, we give a construction of $\mathcal{X}_t = \{g_0^t, \cdots, g_{l-1}^t\}$ to minimize the number of false positives in the candidate set, thus, the search cost of the linear time Search_2.

The cost metric is minimized with respect to a chosen query workload. For simplicity, we consider only queries with a single equality. For each attribute A_t, the query workload is denoted by $\{Q_1, \cdots, Q_{|A_t|}\}$ where Q_j, $1 \leq j \leq |A_t|$, denotes the query with the single equality $A_t = v_j$. We

2168-7161 (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: Hong Kong University of Science and Technology. Downloaded on April 17,2022 at 10:16:44 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2020.2975175, IEEE Transactions on Cloud Computing

assume that the frequency for Q_j, $1 \le j \le |A_t|$, denoted by f_j, is known. Let O_j, $1 \le j \le |A_t|$, be the number of records in the database $\mathcal{D}$ having $A_t = v_j$. Consider a value class $g_y^t = \{v_1, \cdots, v_\kappa\}$ for A_t. For a query Q_j, all records having a value $v_k \in (g_y^t - \{v_j\})$ are false positives, so the cost of false positives is $Cost(g_y^t, Q_j) = \Sigma_{v_k \in g_y^t - \{v_j\}} O_k f_j$ (recall that each false positive is returned f_j times). The cost of false positives related to g_y^t for all queries is $Cost(g_y^t) = \sum_{v_j \in g_y^t} Cost(g_y^t, Q_j)$, and the communication cost of all false positives is $Cost(\mathcal{X}_t) = \Sigma_{y=0}^{l-1} Cost(g_y^t)$.

Definition 5 (Optimal κ-sized class partitioning). Given a class size $\kappa > 1$ such that $|A_t|$ is divisible by κ and $l = \frac{|A_t|}{\kappa}$, $(O_1, \cdots, O_{|A_t|})$ and $(f_1, \cdots, f_{|A_t|})$ specified above, find a class partitioning for A_t, $\mathcal{X}_t = \{g_0^t, \cdots, g_{l-1}^t\}$, such that $Cost(\mathcal{X}_t)$ is minimized and all g_y^t have the size κ. □

This problem can be solved as the following *r-way equipartition problem* for which a branch-and-cut algorithm exists [28]: divide the vertices of a weighted graph $G = (V, E)$ into r equally sized sets, so as to minimize the total weight of edges that have both endpoints in the same set. To solve our optimal class partitioning problem, we can define the graph $G = (V, E)$ as follows: $V = \{1, \cdots, |A_t|\}$ and $E = \{(i, j) \mid 1 \le i < j \le |A_t|\}$, where for each edge $(i, j) \in E$, the weight $(w_{(i,j)} = O_i f_j + O_j f_i)$. Let $r = l = \frac{|A_t|}{\kappa}$. Intuitively, $w_{(i,j)}$ is the communication cost for returning false positives if i and j are grouped into the same class. Then $\mathcal{X}_t = \{g_0^t, \cdots, g_{l-1}^t\}$ is an optimal κ-sized class partitioning if and only if $\mathcal{X}_t$ is an optimal solution to the r-way equipartition problem for $G = (V, E)$.

6 Security Analysis

We show that $\mathsf{SSE} = (\mathsf{SSE}_1, \mathsf{SSE}_2)$ presented in Section 5 achieves class indistinguishability (Definition 4). We first consider SSE_1 and then SSE. The next lemma shows that all records or queries from the same class cannot be distinguished from their ciphertexts.

Lemma 3. Let $\mathsf{SSE}_1 = (\mathsf{Gen}_1, \mathsf{Enc}_1, \mathsf{Trpdr}_1, \mathsf{Search}_1)$ be the SSE scheme constructed in Section 5 and let K_1 be the secret key generated by Gen_1. For any $2d$-dimensional vector V,

(i) $\Pr[\mathsf{Enc}_1(K_1, P_i) = V] = \Pr[\mathsf{Enc}_1(K_1, P_i') = V]$ holds for any records P_i and P_i' from the same record class.

(ii) $\Pr[\mathsf{Trpdr}_1(K_1, Q_j) = V] = \Pr[\mathsf{Trpdr}_1(K_1, Q_j') = V]$ holds for any queries Q_j and Q_j' from the same query class.

PROOF. We prove (i) first. Recall from Eqn (4) that

$$(I_i[t]_1, I_i[t]_2) = (\epsilon_{t,i} \sin(\alpha_t(P_i[t])), \epsilon_{t,i} \cos(\alpha_t(P_i[t]))),$$

$$(I_i'[t]_1, I_i'[t]_2) = (\epsilon_{t,i}' \sin(\alpha_t(P_i'[t])), \epsilon_{t,i}' \cos(\alpha_t(P_i'[t]))),$$

where $\epsilon_{t,i}$ and $\epsilon_{t,i}'$ are drawn from $[-U, -L] \cup [L, U]$ uniformly at random. Since P_i and P_i' are from the same class, from Lemma 1, $\alpha(P_i[t])$ and $\alpha(P_i'[t])$ differ by a multiple of π, which means

$$\sin(\alpha_t(P_i'[t])) = \theta \sin(\alpha_t(P_i[t])),$$

$$\cos(\alpha_t(P_i'[t])) = \theta \cos(\alpha_t(P_i[t])),$$

where θ is either + or - sign. Since $\epsilon_{t,i}$ and $\epsilon_{t,i}'$ are drawn from $[-U, -L] \cup [L, U]$ uniformly at random, for any (v_1, v_2), $\Pr[(I_i[t]_1, I_i[t]_2) = (v_1, v_2)] = \Pr[(I_i'[t]_1, I_i'[t]_2) = (v_1, v_2)]$. This implies $\Pr[I_i = V] = \Pr[I_i' = V]$ for any $2d$-dimensional vector V. Finally, the proof follows from the fact that the matrix multiplication in the last step of Enc_1 does not affect the probability.

The proof of (ii) is similar. According to Eqn (6), for each attribute $A_t (t \in [1, d])$, $Q_j[t]$ is encoded as:

$$(T_j[t]_1, T_j[t]_2) = (\mu_{t,j} \cos(\pi - \alpha_t(Q_j[t])), \mu_{t,j} \sin(\pi - \alpha_t(Q_j[t]))),$$

and $Q_j'[t]$ is encoded as:

$$(T_j'[t]_1, T_j'[t]_2) = (\mu_{t,j}' \cos(\pi - \alpha_t(Q_j'[t])), \mu_{t,j}' \sin(\pi - \alpha_t(Q_j'[t]))),$$

Recall from Lemma 1 that $(\alpha(Q_j[t]) - \alpha(Q_j'[t]))$ is a multiple of π, because Q_j and Q_j' are from the same query class. This implies that

$$\cos(\pi - \alpha_t(Q_j'[t])) = \theta \cos(\pi - \alpha_t(Q_j[t])),$$

$$\sin(\pi - \alpha_t(Q_j'[t])) = \theta \sin(\pi - \alpha_t(Q_j[t])),$$

where θ is either + or - sign. Since $\mu_{t,j}$ and $\mu_{t,j}'$ are drawn from $[-U, -L] \cup [L, U]$ uniformly at random, for any (v_1, v_2),

$$\Pr[(Q_j[t]_1, Q_j[t]_2) = (v_1, v_2)] = \Pr[(Q_j'[t]_1, Q_j'[t]_2) = (v_1, v_2)],$$

and for any $2d$-dimensional vector V,

$$\Pr[Q_j = V] = \Pr[Q_j'] = V.$$

Finally, the matrix multiplication in the last step of Trpdr_1 does not affect the above probability. □

Theorem 1. SSE_1 constructed in Section 5 meets strict class indistinguishability, i.e.,

$$\Pr[\mathbf{Ind}_{\mathsf{SSE}_1, \mathcal{A}}(\psi_1) = 1] = \frac{1}{2}. \tag{11}$$

PROOF. Consider two same trace histories $H_0 = (\mathcal{D}_0 = \{P_{0,1}, \cdots, P_{0,n}\}, \mathcal{Q}_0 = \{Q_{0,1}, \cdots, Q_{0,m}\})$ and $H_1 = (\mathcal{D}_1 = \{P_{1,1}, \cdots, P_{1,n}\}, \mathcal{Q}_1 = \{Q_{1,1}, \cdots, Q_{1,m}\})$ from the same history class chosen by the adversary in Definition 4. This means that $\mathcal{D}_0$ and $\mathcal{D}_1$ are from the same class, and for $1 \le j \le m$, $Q_{0,j}$ and $Q_{1,j}$ are from the same class. For any $b \in \{0, 1\}$ chosen randomly, the adversary receives the ciphertext for $(\mathcal{D}_b, \mathcal{Q}_b)$ generated by Enc_1 and Trpdr_1. From Lemma 3, given $E_1(P_{b,i})$ and $E_1(Q_{b,j})$, where $1 \le i \le n$ and $1 \le j \le m$, H_0 and H_1 are equally likely to be the underlying history based on the observed ciphertexts. This remains true even if the adversary computes the candidate set $Cand(Q_{b,j})$ because $Cand(Q_{0,j})$ and $Cand(Q_{1,j})$ have the same size, $1 \le j \le m$. Finally, any index structure $\mathbf{I}$ constructed using $E_1(P_{b,i})$, $1 \le i \le n$, discloses no more information than $E_1(P_{b,i})$ does. So the adversary gains no advantage in guessing the value of b from accessing $E_1(P_{b,i})$ and $E_1(Q_{b,j})$, where $1 \le i \le n$ and $1 \le j \le m$, computing the queries. □

Theorem 2. Let SSE_1 be constructed in Section 5 and let SSE_2 be any scheme meeting ciphertext indistinguishability (say [2]). Then $\mathsf{SSE} = (\mathsf{SSE}_1, \mathsf{SSE}_2)$ meets class indistinguishability, that is,

$$|\Pr[\mathbf{Ind}_{\mathsf{SSE}, \mathcal{A}}(\psi_1, \psi_2) = 1] - \frac{1}{2}| \le negl(\psi_2),$$

2168-7161 (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: Hong Kong University of Science and Technology. Downloaded on April 17,2022 at 10:16:44 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2020.2975175, IEEE Transactions on Cloud Computing

where ψ_2 is the security parameter of SSE_2.

PROOF. To show class indistinguishability, we consider two histories $H_0 = (\mathcal{D}_0 = \{P_{0,1}, \cdots, P_{0,n}\}, \mathcal{Q}_0 = \{Q_{0,1}, \cdots, Q_{0,m}\})$ and $H_1 = (\mathcal{D}_1 = \{P_{1,1}, \cdots, P_{1,n}\}, \mathcal{Q}_1 = \{Q_{1,1}, \cdots, Q_{1,m}\})$ chosen by the adversary of SSE_1. H_0 and H_1 have the same trace and belong to the class. (H_0, H_1) is also a valid pair chosen by the adversary in the game of SSE_2 because the latter only requires H_0 and H_1 having same trace.

Recall that

$$E(P_{b,i}) = (E_1(P_{b,i}), E_2(P_{b,i})),$$
$$E(Q_{b,j}) = (E_1(Q_{b,j}), E_2(Q_{b,j})).$$

In the game for SSE_1, the adversary has access to $E_1(P_{b,i})$ and $E_1(Q_{b,j})$, and in the game for SSE_2, the adversary has access to $E_2(P_{b,i})$ and $E_2(Q_{b,j})$, as well as the candidate sets $Cand(Q_{b,j})$ computed by SSE_1, for $1 \le j \le m$.

First, the ciphertext indistinguishability of SSE_2 implies that the adversary's advantage in guessing the value of b from accessing $E_2(P_{b,i})$ and $E_2(Q_{b,j})$ is negligibly different from the probability $\frac{1}{2}$.

In addition, the history formed by the candidate sets $Cand(Q_{1,j})$ plus queries $Q_{1,j}$, $1 \le j \le m$, and the history formed by $Cand(Q_{0,j})$ plus queries $Q_{0,j}$, $1 \le j \le m$, have the same trace. Therefore, the adversary's advantage remains unchanged even if the adversary also has access to the candidate sets computed by SSE_1.

Finally, from Theorem 1, this advantage is unaffected by accessing $E_1(P_{b,i})$ and $E_1(Q_{b,j})$ in the game of SSE_1 because the adversary gains *no advantage* in the game of SSE_1 thanks to the strict class indistinguishability provided by SSE_1. □

6.1 Adaptive Adversaries

So far, we considered a non-adaptive adversary in Definition 4. Our approach achieves class indistinguishability for an adaptive adversary as well. We first define class indsitinguishability for an adaptive adversary.

Definition 6 (Class indistinguishability for an adaptive adversary). Assume that the class partitioning $\{g_0^t, \cdots, g_{l-1}^t\}$ is given for every attribute A_t. Let $\mathsf{SSE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Trpdr}, \mathsf{Search})$. Let $\mathcal{A} = (\mathcal{A}_0, \cdots, \mathcal{A}_{q+1})$ be an adversary such that $q \in \mathbb{N}$. Consider the following probabilistic game:

$\mathbf{Ind}_{\mathsf{SSE},\mathcal{A}}(\psi)$

1. $K \leftarrow \mathsf{Gen}(1^\psi)$
2. $(st_{\mathcal{A}}, \mathcal{D}_0, \mathcal{D}_1) \leftarrow \mathcal{A}_0(1^\psi)$
3. $b \xleftarrow{\$} \{0,1\}$
4. for $1 \le i \le n$
5. $E(P_{b,i}) \leftarrow \mathsf{Enc}(K, P_{b,i})$
6. let $\mathbf{I}_b = (E(P_{b,1}), \cdots, E(P_{b,n}))$
7. $(st_{\mathcal{A}}, Q_{0,1}, Q_{1,1}) \leftarrow \mathcal{A}_1(1^\psi, \mathbf{I}_b)$
8. $\mathbf{T}_{b,1} \leftarrow \mathsf{Trpdr}(K, Q_{b,1})$
9. for $2 \le j \le q$
10. $(st_{\mathcal{A}}, Q_{0,j}, Q_{1,j}) \leftarrow \mathcal{A}_j(st_{\mathcal{A}}, \mathbf{I}_b, \mathbf{T}_{b,1}, \cdots, \mathbf{T}_{b,j-1})$
11. $\mathbf{T}_{b,j} \leftarrow \mathsf{Trpdr}(K, Q_{b,j})$
12. let $\mathbf{T}_b = (\mathbf{T}_{b,1}, \cdots, \mathbf{T}_{b,q})$
13. $b' \leftarrow \mathcal{A}_{q+1}(st_{\mathcal{A}}, \mathbf{I}_b, \mathbf{T}_b)$
14. if $b' = b$, output 1
15. otherwise output 0

subject to two restrictions: (i) H_0 and H_1 have the same trace, (ii) H_0 and H_1 are from the same class. $st_{\mathcal{A}}$ is a string that captures $\mathcal{A}$'s state. We say that SSE ensures ***class indistinguishability for an adaptive adversary*** if for all polynomial-size adversaries $\mathcal{A} = (\mathcal{A}_0, \cdots, \mathcal{A}_{q+1})$ such that $q = \mathsf{poly}(\psi)$,

$$|\mathsf{Pr}[\mathbf{Ind}_{\mathsf{SSE},\mathcal{A}}(\psi) = 1] - \frac{1}{2}| \le negl(\psi) \tag{12}$$

where the probability is taken over the choice of b and the coins of $\mathsf{Gen}, \mathsf{Enc}, \mathsf{Trpdr}$. We say that SSE ensures ***strict class indistinguishability for an adaptive adversary*** if

$$\mathsf{Pr}[\mathbf{Ind}_{\mathsf{SSE},\mathcal{A}}(\psi) = 1] = \frac{1}{2}.\square \tag{13}$$

Unlike a non-adaptive adversary, an adaptive adversary can choose adaptively the next query pair $(Q_{0,j}, Q_{1,j})$ *after* receiving the encrypted records and encrypted queries for the previous queries (Lines 9-11). The next theorem shows that our SSE construction also meets strict class indistinguishability for an adaptive adversary.

Theorem 3. SSE_1 constructed in Section 5 meets strict class indistinguishability for an adaptive adversary, i.e.,

$$\mathsf{Pr}[\mathbf{Ind}_{\mathsf{SSE}_1,\mathcal{A}}(\psi_1) = 1] = \frac{1}{2}. \tag{14}$$

PROOF. A key observation from Theorem 1 is that the adversary gains no advantage of guessing the value of b from receiving encrypted records and encrypted queries. In particular, this holds after the adversary receives the encrypted queries chosen before choosing the next query pair in the adaptive process. □

Theorem 2 is also generalized to an adaptive adversary as follows.

Theorem 4. Let SSE_1 be constructed in Section 5 and let SSE_2 be any scheme meeting ciphertext indistinguishability for an adaptive adversary (say [2]). Then $\mathsf{SSE} = (\mathsf{SSE}_1, \mathsf{SSE}_2)$ meets class indistinguishability for an adaptive adversary, that is,

$$|\mathsf{Pr}[\mathbf{Ind}_{\mathsf{SSE},\mathcal{A}}(\psi_1, \psi_2) = 1] - \frac{1}{2}| \le negl(\psi_2),$$

where ψ_2 is the security parameter of SSE_2.

PROOF. The proof is essentially same as Theorem 2. The difference is that here SSE_1 has strict class indistinguishability for an adaptive adversary (Theorem 3) and SSE_2 has ciphertext indistinguishability for an adaptive adversary as assumed. □

2168-7161 (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: Hong Kong University of Science and Technology. Downloaded on April 17,2022 at 10:16:44 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2020.2975175, IEEE Transactions on Cloud Computing

6.2 Adversaries with Auxiliary Knowledge

We now consider the case where the adversary may acquire auxiliary knowledge from other sources. One type of auxiliary knowledge is the correspondence $(E_1(P_i), P_i)$ of ciphertext and plaintext for some records P_i in the database; in other words, the adversary can link $E_1(P_i)$ to the known plaintext P_i for some records P_i in $\mathcal{D}$.

Recall the equation

$$E_1(Q_j)^T E_1(P_i) = \frac{T_j^T I_i}{|\mathbf{M}^T T_j||\mathbf{M}^{-1} I_i|}, \tag{15}$$

used in our approach, where I_i or T_j are intermediate vectors generated from P_i and Q_j before being transformed by the secret matrix $\mathbf{M}$. A question is whether knowing the correspondence $(E_1(P_i), P_i)$ allows the adversary to learn I_i or T_j using the above equation, therefore, further learn the secret key $\mathbf{M}$. This attack can be prevented by modify our key generation. Recall $\alpha(v) = y\frac{\pi}{l} + (x-1)\pi$ (Eqn (3)), where x and y encode the in-class position and the class label of a domain value v. There are many different (x, y) encodings because each pair of permutation of values in a class and permutation of class labels will define a different (x, y) encoding. Therefore, we can hide the the (x, y) encoding by making the pair of permutations used for generating (x, y) a part of the secret key. In particular, we modify $\mathsf{Gen}_1(1^{\psi_1})$ to output the secret key of the form $K_1 = (\mathbf{M}, \Psi)$, where $\mathbf{M}$ is a $(2d \times 2d)$ invertible matrix as before, and Ψ is the permutation defining the (x, y) encoding for all domain values of all attributes. Now we show that with this minor modification SSE_1 is resilient to the above attack.

Theorem 5. Let d be the number of querying attributes, and l_t be the number of classes for the attribute A_t. For SSE_1 constructed in Section 5 with the modified $\mathsf{Gen}_1(1^{k_1})$ above, even if the adversary acquires the auxiliary knowledge in the form of pairs $(E_1(P_i), P_i)$ for any set of P_i in the database $\mathcal{D}$, the adversary cannot distinguish the true secret key $K_1 = (\mathbf{M}, \Psi)$ from the other $(\prod_{1\le t\le d}(2l_t) - 1)$ possible secret keys $K_1' = (\mathbf{M}', \Psi')$.

PROOF. We show that, given $E_1(Q_j)$ and $E_1(\mathcal{D})$ and the observed pairs $(E_1(P_i), P_i)$ for a set $\{P_i\} \subseteq \mathcal{D}$, "many" (I_i', T_j') cannot be distinguished from (I_i, T_j) because (I_i', T_j') is equally likely to be generated by another secret key $K_1' = (\mathbf{M}', \Psi')$ and because the adversary does not know the secret key picked. In the following, we construct such (I_i', T_j') and $K_1' = (\mathbf{M}', \Psi')$. From Eqn (4), for $1 \le t \le d$,

$$\begin{aligned} I_i[t]_1 &= \epsilon_{t,i} \sin(\alpha^t), \\ I_i[t]_2 &= \epsilon_{t,i} \cos(\alpha^t), \end{aligned} \tag{16}$$

where $\alpha^t = y_t\frac{\pi}{l_t} + (x_t - 1)\pi$ and (x_t, y_t) encodes the in-class position and class label of $P_i[t]$ according to Ψ. Let $\theta^t = \Delta_2\frac{\pi}{l_t} + \Delta_1\pi$ for integers Δ_1 and Δ_2. Replacing α^t with $(\alpha^t - \theta^t)$ corresponds to replacing the encoding (x_t, y_t) with the new encoding $(x_t - \Delta_1, y_t - \Delta_2)$, where $-$ is modular the class size for $P_i[t]$ and the number of classes for A_t, respectively. This corresponds to rotating $(I_i[t]_1, I_i[t]_2)$ clockwise by θ^t degree, i.e., $(I_i'[t]_1, I_i'[t]_2) = R_t(I_i[t]_1, I_i[t]_2)$ for the rotation matrix

$$R_t = \begin{pmatrix} cos\theta^t & -sin\theta^t \\ sin\theta^t & cos\theta^t \end{pmatrix}. \tag{17}$$

Let

$$I_i' = (I_i'[1]_1, I_i'[1]_2, \cdots, I_i'[d]_1, I_i'[d]_2).$$

Note $I_i' = RI_i$, where

$$R = \begin{bmatrix} R_1 & & & & \\ & R_2 & & & \\ & & \cdots & & \\ & & & \cdots & \\ & & & & R_d \end{bmatrix}. \tag{18}$$

Similarly, let $(T_j'[t]_1, T_j'[t]_2) = R_t(T_j[t]_1, T_j[t]_2)$ and let

$$T_j' = (T_j'[1]_1, T_j'[1]_2, \cdots, T_j'[d]_1, T_j'[d]_2).$$

$T_j' = RT_j$.

Define another secret key $K_1' = (\mathbf{M}', \Psi')$, where $\mathbf{M}' = R\mathbf{M}$ and Ψ' specifies the above new $(x_t - \Delta_1, y_t - \Delta_2)$ encoding scheme for all attributes. $\mathbf{M}'$ is an invertible matrix because R is an orthogonal matrix and $\mathbf{M}$ is an invertible matrix, i.e., $\mathbf{M}'^{-1} = \mathbf{M}^{-1}R^T$. From the above discussion, we have

$$E_1(Q_j) = \mathbf{M}^T T_j = (R\mathbf{M})^T (RT_j) = \mathbf{M}'^T T_j',$$

$$E_1(P_i) = \mathbf{M}^{-1} I_i = (\mathbf{M}^{-1}R^T)(RI_i) = \mathbf{M}'^{-1} I_i'.$$

These equalities imply that, given the observed $E_1(Q_j)$ and $E_1(P_i)$, the following two cases are possible. Case 1: I_i' and T_j' were generated from P_i and Q_j using the secret key $K_1' = (\mathbf{M}', \Psi')$, or Case 2: I_i and T_j were generated from P_i and Q_j using the secret key $K_1 = (\mathbf{M}, \Psi)$. These cases are equally likely because the secret key is chosen uniformly at random.

It remains to show that the number of distinct $K_1' = (\mathbf{M}', \Psi')$, thus, the number of distinct associated (I_i', T_j'), is large. The matrix R_t has a periodicity of 2π over θ^t, so, for each Δ_2 from $\{0, \cdots, l_t - 1\}$, R_t has two different outcomes, one for all even Δ_1 and one for all odd Δ_1. Therefore, the number of distinct R_t is $2l_t$, where l_t is the number of classes for A_t, and the number of distinct R is $\prod_{1\le t\le d}(2l_t)$. Consequently, there are $\prod_{1\le t\le d}(2l_t)$ different $K_1' = (\mathbf{M}', \Psi')$ and the same number of associated (I_i', T_j'). Even for the smallest $l_t = 1$, $\prod_{1\le t\le d}(2l_t)$ is exponential in d. □

7 Empirical Evaluation

In this section, we summarize our study on the performance of CLASS presented in Section 5.

Data sets. We consider two data sets.The **US Census data set**[1] was collected from 2006 to 2011 with $d = 3$ categorical attributes: Race (237), PlaceOfBirth (531) and City (1134), with the domain size indicated in the bracket. $\mathcal{D}_{1M}$, $\mathcal{D}_{10M}$, $\mathcal{D}_{50M}$ and $\mathcal{D}_{100M}$, denote four samples containing the first 1, 10 , 50, 100 million records, respectively. The **US Flight data-set**[2] contains 28 million records and $d = 8$ categorical attributes. We used the second data set to evaluate the impact of data dimensionality. These data sets are much larger than those in the literature for evaluating searchable encryption techniques [9], [11], [19], [29], [30].

1. IPUMS US Census data set. https://www.ipums.org
2. US Flight data-set. http://stat-computing.org/dataexpo/2009/the-data.html

2168-7161 (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: Hong Kong University of Science and Technology. Downloaded on April 17,2022 at 10:16:44 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2020.2975175, IEEE Transactions on Cloud Computing

Queries. For the US Census data, we generated a query pool $QW = \{Q^1 \cup \cdots \cup Q^d\}$ using $\mathcal{D}_{1M}$. For each integer $q \in [1, d]$, Q^q contains 100 q-equality queries generated as follows. Let Q^q_* contain all q-equality queries that have a non-empty result in $\mathcal{D}_{1M}$. Let Sel_Q denote the *selectivity* of a query Q, defined as the percentage of records in the data that satisfy the query. We picked 100 queries Q from Q^q_*. The probability of picking a query Q is modeled by the beta distribution $Beta(\alpha, \beta)$ of the selectivity sel_Q [31]. In general, with a fixed β a smaller α leads to a higher probability for a query with a smaller selectivity. This case assigns a higher probability to a query having a smaller selectivity, modeling the typical scenario that more queries retrieve more specific information. The query generation for the US Flight data set is described in Section 7.3.

Competing methods. For CLASS, we implemented the sub-linear method Search_1 for hyperplane queries by M-Tree [26] and the linear method Search_2 by Secure Index [11]. The M-Tree and the database are stored on disk and the candidate set computed by Search_1 is kept in memory. Since [11] deals with only single-keyword search, we convert equality conjunction queries to single-keyword search by treating each conjunction up to the maximum number of equalities in a query as a new keyword. We used the method in Section 5.3 to construct the class partitioning for each attribute for a given class size κ with the single equality queries Q^1_* as the input. ***By default***, we set the class size as $\kappa = 6$, the bounds for the noise interval as $(L = 1000, U = 1100)$, and the parameters of beta distribution $Beta(\alpha, \beta)$ for class parition construction as $(\alpha = 0.5, \beta = 3)$.

We consider two baselines. The **first baseline** is OXT, the state-of-the-art sub-linear search for conjunctive keywords queries [6]. OXT uses a disk-resident data structure *TSet* to locate the documents containing the least frequent keyword in the query, called s-term, and uses a RAM-resident data structure *XSet* to filter the result using the remaining keywords in the query, called x-terms. The **second baseline**, denoted by SI, is Secure Index [11] applied to the full database following the same strategy of converting equality conjunction queries to single-keyword search as described above for Search_2 in CLASS.

We wrote all codes in C++ and leveraged OpenSSL library to implement cryptographic primitives. Experiments were deployed on a Linux machine with a single Intel Core i7 CPU with 2.3 GHz and 16 GB RAM.

7.1 Setup Cost

At system initialization, there is a one-time *setup cost*, which includes storage overhead for the server, and data encryption time for the client. Note that the setup cost of CLASS is divided into the setup cost of Enc_1 and the setup cost of SI. Table 2 shows the overall storage overhead of SI, CLASS and OXT, highlighted in gray color, for various data sample sizes. These structures were stored on the server, thus, there is no client storage overhead. Since these structures were generated by the client, they also represent the upload communication cost at setup. OXT uses most storage and SI uses least storage. In fact, we can only include the result for samples up to $\mathcal{D}_{50M}$ for OXT due to the excessively long data encryption time.

	SI	CLASS			OXT
		Enc_1	SI	**Overall**	
$\mathcal{D}_{1M}$	0.015	0.12	0.015	0.135	1.21
$\mathcal{D}_{10M}$	0.156	1.28	0.156	1.436	11.98
$\mathcal{D}_{50M}$	0.781	6.55	0.781	7.331	120.35
$\mathcal{D}_{100M}$	1.562	13.98	1.562	15.542	-

TABLE 2: Storage Overhead (GB)

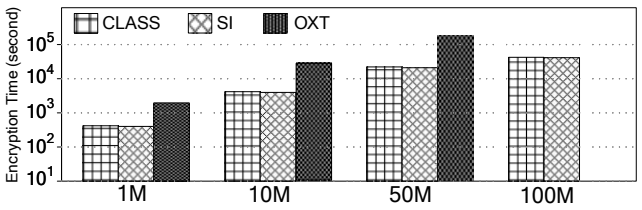


Fig. 8: Data encryption time

Figure 8 compares data encryption time at setup in *log scale*. CLASS's encryption time is the sum of encryption times of SSE_1 and SSE_2 where SSE_2 is SI. The small difference between CLASS and SI suggests that the additional encryption time caused by SSE_1 is very small. OXT's encryption time is one order of magnitude longer than CLASS and SI. The index structure in CLASS is constructed by the server by applying an existing indexing algorithm (e.g., M-Tree in our experiments), we do not further discuss its performance.

7.2 Query Cost

For each query, there is a *query cost* in the form of query encryption time, computing time and result delivery time. We focus on query computing time (averaged over the queries in QW) that dominates the query time. Figure 9 reports query time in log scale vs four different data cardinality. For $\mathcal{D}_{100M}$, we could not get OXT's query time due to long database encryption time. In fact, OXT hides the entries on an inverted list by storing them in random locations on disk, which results in a large number of random I/O accesses during index construction and query process. As expected, the query time of SI grows linearly with data cardinality. By estimation, SI needs to take about 1000 seconds on $\mathcal{D}_{100M}$ which is too slow for large databases. It is clear that CLASS outperforms SI and OXT.

The efficiency of CLASS relies on the sub-linear Candidate Phase to reduce the search space of the linear Filtering Phase to a small candidate set. We measure this effectiveness

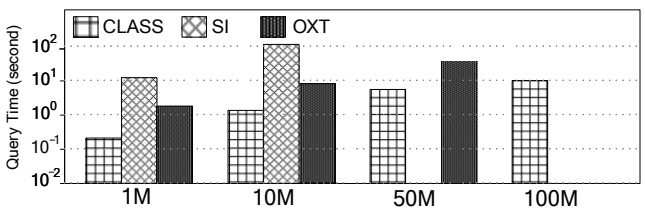


Fig. 9: Query time vs data cardinality

2168-7161 (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: Hong Kong University of Science and Technology. Downloaded on April 17,2022 at 10:16:44 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2020.2975175, IEEE Transactions on Cloud Computing

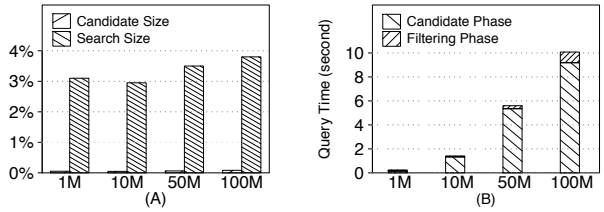


Fig. 10: Efficiency of CLASS vs data cardinality

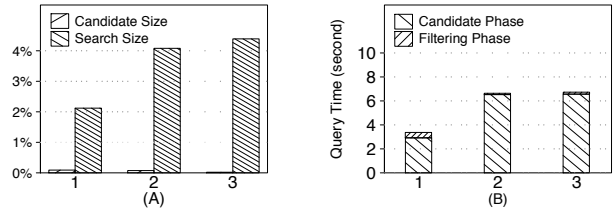


Fig. 11: Efficiency of CLASS vs query dimensionality q ($\mathcal{D}_{50M}$)

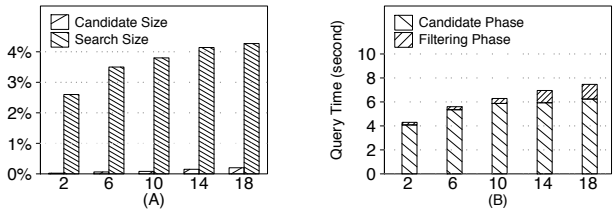


Fig. 12: Efficiency of CLASS vs value class size κ ($\mathcal{D}_{50M}$)

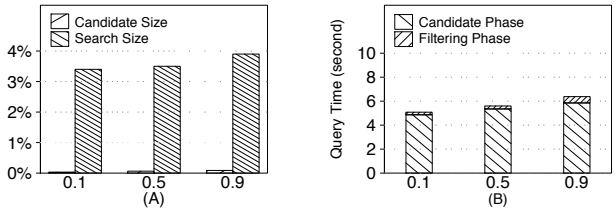


Fig. 13: Efficiency of CLASS vs query distribution α ($\mathcal{D}_{50M}$)

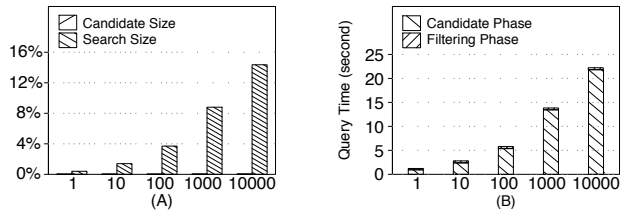


Fig. 14: Efficiency of CLASS vs random noise interval size $|U-L|$ ($\mathcal{D}_{50M}$, $L=1000$)

by two metrics:

$$\textit{candidate\_size} = \frac{|\textit{Cand}|}{|\mathcal{D}|}, \qquad \textit{search\_size} = \frac{|\textit{Test}|}{|\mathcal{D}|},$$

where *Cand* denotes the candidate set computed by Candidate Phase and *Test* denotes the set of records examined in Candidate Phase.

Figure 10(A) shows *candidate_size* and *search_size*, and Figure 10(B) shows the times spent in the two phases. *candidate_size* is no more than 0.1% and *search_size* is no more than 4%. In other words, with Candidate Phase that searches 4% of the data, the search space of Filtering Phase is reduced to a candidate set of size less than 0.1% of the full data. On query time, for $\mathcal{D}_{100M}$, Filtering Phase that runs SI on the candidate set took no more than 1 second, and Candidate Phase took 9 seconds to generate this candidate set. The total query time is much less than the 1000 seconds if SI was run on the full data as shown in Figure 9. In the following, we study the effect of other factors w.r.t the efficiency of CLASS based on $\mathcal{D}_{50M}$.

Effect of Query Dimensionality. Figure 11 reports the search space (A) and query time (B) for varied query dimensionality $q \in \{1,2,3\}$ of q-equality queries (x-axis). Queries with more equalities tend to require a larger *search_size* and a longer query time in Candidate Phase. In contrast, single-equality queries have a larger selectivity and a larger *candidate_size*, leading to a longer query time in Filtering Phase. For all q tested, *candidate_size* and *search_size* are small, verifying the effectiveness of reducing the search space in Filtering Phase and the performance of the sub-linear search in Candidate Phase.

Effect of Class Size. The class size κ of a class partitioning plays a role in balancing the level of indistinguishability and the sub-linear search performance. We studies the effect of the class size $\kappa \in \{2,6,10,14,18\}$ (x-axis) on query time. As shown in Figure 12, a larger κ leads to larger *search_size* and *candidate_size* due to more data tested in Candidate Phase and more false positives in the candidate set *Cand*. Despite this trend, even for $\kappa = 18$, *Cand* is 0.2% of the full data set. This significantly reduces the time of Filtering Phase that is applied to the candidate set, as shown in Figure 12(B). In all cases, the total query time of the two phases is no more than 10 seconds. This study clearly shows that the sub-linear Candidate Phase is highly effective in pruning the search space.

Effect of Class Construction. Our performance based the construction of class partitioning which takes a query pool and query frequencies as input to minimize false positives in the candidate set. This experiment studies the effectiveness of this construction for different query pools and query frequencies modeled by different setting of beta distribution $Beta(\alpha,\beta)$. In particular, we fixed $\beta = 3$ and set $\alpha \in \{0.1, 0.5, 0.9\}$ (x-axis) to model different slopes of the increasing sampling probability as the query selectivity decreases. Figure 13(A) shows that *candidate_size* is no more than 0.1% in all cases. In comparison, if the class partitioning were randomly generated, *candidate_size* is 19%, 20%, and 21% at $\alpha = 0.1, 0.5, 0.9$. This clearly supports that our class partitioning construction achieves the goal of reducing false positives in the candidate set.

Effect of Random Noises. Figure 14 examines the impact of the interval $[-U,-L]\cup[L,U]$ for drawing random noises ϵ and μ in functions Enc_1 and Trpdr_1, respectively. We fixed the lower limit $L = 1000$ and varied the size $(U-L)$ (x-axis). A larger $(U-L)$ leads to more random noises injected, thus less effective indexed search in Candidate Phase as shown by the larger *search_size*. However, even with the maximum $(U-L) = 10000$, *candidate_size* remains very small, which suggests that restricting Filtering Phase to the candidate set is highly effective. In general, Filtering Phase employs crypto primitives for producing the exact query result, therefore, it is more important to reduce the search

2168-7161 (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: Hong Kong University of Science and Technology. Downloaded on April 17,2022 at 10:16:44 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2020.2975175, IEEE Transactions on Cloud Computing

space in this phase. Our two phase search exactly achieves this goal.

7.3 Effect of Data Dimensionality

In this experiment, we switched to the US Flight data set that has more attributes to study the effect of data dimensionality on query time. This data set has 8 attributes: TailNum (12157), FlightNum (8130), Carrier (1492), Destination (336), Origin (331), DayOfMonth (31), Month (12), and DayOfWeek (7). For $4 \leq d \leq 8$, $\mathcal{D}^d_{20M}$ denotes the data set containing the first 20 million records with the first d categorical attributes, and the query pool QW^d contains 100 q-equality queries for each $q \in \{1, 2, 3, 4\}$ generated following the same query generation as before. The dimensionality of encrypted records and queries is $2d$. We set the class size at $\kappa = 2$ if $2 < |A_t| < 10$ and $\kappa = 6$ if $|A_t| \geq 10$.

Table 3 shows query time for $\mathcal{D}^d_{20M}$ for $4 \leq d \leq 8$. Due to OXT's long database encryption time we were able to collect only OXT's query times for $d = 4$. SI denotes the linear search applied to the full data set, Candidate Phase and Filtering Phase are the two phases of CLASS. Candidate Phase* denotes the linear version of Candidate Phase by testing the condition in Eqn (8) against every record. Note that Candidate Phase* produces the same candidate set as Candidate Phase.

Dimensionality d	4	5	6	7	8
OXT	11.2	-	-	-	-
SI	220	220	230	235	240
Candidate Phase	3.5	5.2	10.5	17.7	26.4
Filtering Phase	0.5	1.2	1.6	2.4	3.5
Candidate Phase*	25	25	26	27	28

TABLE 3: Query time (seconds) vs dimensionality d on US Flight data (20M records)

As d increases from 4 to 8 (i.e., 8 to 16 after encryption), the M-Tree based Candidate Phase's query time increases quickly, which is consistent with the well known "curse of dimensionality" principle. However, the total query time of Candidate Phase and Filtering Phase is still significantly smaller than SI's query time. Candidate Phase* is almost unaffected by the increase of d because it does not use any index. Therefore, for a high dimensionality d, Candidate Phase* followed by Filtering Phase would be the winner because their total time does not change much as d increases. In other words, even testing the condition in Eqn (8) for every record in the database is justified to reduce the search space of the heavyweight Search_2. This is because this condition uses only standard numeric operations instead of cryptographic primitives.

8 RELATED WORK

This work is at the intersection of cryptography (for strong security guarantee) and database (for high performance query processing).

8.1 Cryptographic Solutions

Most existing SSE schemes focus on conjunctive keyword queries requiring a linear search performance [32], [33], [34], [35]. Conjunctive keyword search is also studied based on Hidden Vector Encryption (HVE) [29] using bilinear pairings over elliptic curve groups which suffers from prohibitive computation and communication costs [36].

A few recent work considers sub-linear search for conjunctive keyword search [6], [20], [36]. These schemes relax the notion of ciphertext indistinguishability by capturing certain disclosures (using a leakage function) caused by a sub-linear search process. One problem with these approaches is that it is a daunting task to capture the full extent of such low-level disclosures that are specific to the design of the index structure and the sub-linear search algorithm. The real-world consequences of such low-level disclosures are poorly understood, which was highlighted as an important open question [37] [2].

Fully homomorphic encryption [38], [39], [40] currently is not suitable for data outsourcing applications due to prohibitive communication and computation costs. Differential privacy [41], [42] addresses a different scenario where changing a single record in the data does not alter significantly the output of a query. In the data outsourcing setting, the security notion prevents the cloud server from learning the plaintext information of data as well as query output.

8.2 Database Solutions

The research in database traditionally focused on efficiency for large databases by adopting ad hoc security definitions. Examples are order preserving encryption [43] and distance preserving encryption [44], which makes indexing easy but discloses order and proximity information of plaintext. CryptDB [45] enables the DBMS server to execute SQL queries on encrypted data, using deterministic encryption for equality checks, group by, and equality-joins, and order preserving encryption for order checks. It is well known that deterministic encryption does not provide sufficient protection in practice. Bucketization [8] [9] [10] provides a trade-off solution to security and sub-linear search performance. We already mentioned in Section 1.1 that this approach requires either significant client work or a high communication cost.

Asymmetric scalar product preserving encryption (ASPE) [46] is another popular ad hoc secure SSE schemes. The basic idea of APSE is representing each data and query as vectors and encrypting them by invertable matrices. Thanks to the inner product preserving property, a lot of recent work adopted ASPE to support a variety of important similarity-based queries efficiently, such as multi-keyword ranked queries [47], [48] and fuzzy multi-keyword match queries [49]. However, as proved in [50], ASPE and its vairants cannot provide sufficient security guarantee in practice.

8.3 Differences from Previous Publication

A preliminary work was reported in [51], which was extended by the current paper in the following ways. First of all, to deal with dynamic data, we analyze two types of SSE: plaintext-indexing and ciphertext-indexing (Section 3). This analysis serves as a major motivation for the choice of ciphertext-indexing SSE presented in the current work. Second, we consider two more powerful adversaries for security models: the adversary that adaptively chooses the next

2168-7161 (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: Hong Kong University of Science and Technology. Downloaded on April 17,2022 at 10:16:44 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2020.2975175, IEEE Transactions on Cloud Computing

query after receiving the results of previous queries, and the adversary that may acquire the plaintext of some ciphertext records (Section 6.1 and Section 6.2, respectively). Third, we add more examples and implementation details, including dealing with numeric unstability, leveraging implementation of hyperplane queries, and dealing with dynamic data (Section 5.2.4). Furthermore, we provide a more comprehensive evaluation of the proposed SSE scheme CLASS, including evaluating data encryption time for setup cost (Section 7.1) and examining the effect of data dimensionality using a new real-world data set (Section 7.3).

9 CONCLUSION

A key challenge of data outsourcing in cloud computing is providing a strong security guarantee while supporting a sub-linear search performance for dealing with large databases. The existing bucketization approach partially addresses this requirement at the cost of client performing local search and increased communication cost of transmitting false positives. This is unsatisfactory because client computation and communication bandwidth are typically the bottleneck. We addressed these limitations by proposing a novel dynamic SSE scheme, called CLASS, that provides a similar level of security to that of bucketization and pushes the work of search and false positive filtering tasks to the computationally powerful server. The key is a carefully chosen security definition and a carefully designed encryption scheme so that such computation can be performed by the server while providing a strong security guarantee. A distinct feature of CLASS is enabling existing plaintext based index methods for the search over encrypted data and queries.

REFERENCES

[1] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proceeding of IEEE 2000 Symposium on Security and Privacy*. IEEE, 2000, pp. 44–55.

[2] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: improved definitions and efficient constructions," vol. 19, no. 5. IOS Press, 2011, pp. 895–934.

[3] R. Bost, "$\sigma o \varphi o \varsigma$–forward secure searchable encryption," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016, pp. 1143–1154.

[4] K. S. Kim, M. Kim, D. Lee, J. H. Park, and W.-H. Kim, "Forward secure dynamic searchable symmetric encryption with efficient updates," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2017, pp. 1449–1463.

[5] J. Ghareh Chamani, D. Papadopoulos, C. Papamanthou, and R. Jalili, "New constructions for forward and backward private symmetric searchable encryption," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2018, pp. 1038–1055.

[6] D. Cash, S. Jarecki, C. Jutla, H. Krawczyk, M.-C. Roşu, and M. Steiner, "Highly-scalable searchable symmetric encryption with support for boolean queries," in *Annual Cryptology Conference*, 2013, pp. 353–373.

[7] L. Sweeney, "k-anonymity: A model for protecting privacy," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 10, no. 05, pp. 557–570, 2002.

[8] H. Hacigümüş, B. Iyer, C. Li, and S. Mehrotra, "Executing sql over encrypted data in the database-service-provider model," in *Proceedings of the 2002 ACM SIGMOD international conference on Management of data*. ACM, 2002, pp. 216–227.

[9] B. Hore, S. Mehrotra, M. Canim, and M. Kantarcioglu, "Secure multidimensional range queries over outsourced data," *The VLDB Journal*, vol. 21, no. 3, pp. 333–358, 2012.

[10] B. Hore, S. Mehrotra, and G. Tsudik, "A privacy-preserving index for range queries," in *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*. VLDB Endowment, 2004, pp. 720–731.

[11] E.-J. Goh *et al.*, "Secure indexes." *IACR Cryptology ePrint Archive*, vol. 2003, p. 216, 2003.

[12] Z. Zhang, K. Wang, W. Lin, A. W.-C. Fu, and R. C.-W. Wong, "Repeatable oblivious shuffling of large outsourced data blocks," in *10th ACM Symposium on Cloud Computing*. ACM, 2019, pp. 287–298.

[13] X. Yi, M. G. Kaosar, R. Paulet, and E. Bertino, "Single-database private information retrieval from fully homomorphic encryption," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 5, pp. 1125–1134, 2012.

[14] E. Stefanov, M. Van Dijk, E. Shi, C. Fletcher, L. Ren, X. Yu, and S. Devadas, "Path oram: an extremely simple oblivious ram protocol," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. ACM, 2013, pp. 299–310.

[15] C. Liu, L. Zhu, M. Wang, and Y.-A. Tan, "Search pattern leakage in searchable encryption: Attacks and new construction," *Information Sciences*, vol. 265, pp. 176–188, 2014.

[16] Z. Zhang, K. Wang, W. Lin, A. W.-C. Fu, and R. C.-W. Wong, "Practical access pattern privacy by combining pir and oblivious shuffle," in *28th ACM International Conference on Information and Knowledge Management*. ACM, 2019, pp. 1331–1340.

[17] "Advanced encryption standard." http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf.

[18] R. Li, A. X. Liu, A. L. Wang, and B. Bruhadeshwar, "Fast range query processing with strong privacy protection for cloud computing," vol. 7, no. 14. VLDB Endowment, 2014, pp. 1953–1964.

[19] P. Wang and C. V. Ravishankar, "Secure and efficient range queries on outsourced databases using rp-trees," in *2013 IEEE 29th International Conference on Data Engineering (ICDE)*. IEEE, 2013, pp. 314–325.

[20] S. Kamara and T. Moataz, "Boolean searchable symmetric encryption with worst-case sub-linear complexity," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2017, pp. 94–124.

[21] R. Li and A. X. Liu, "Adaptively secure conjunctive query processing over encrypted data for cloud computing," in *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*. IEEE, 2017, pp. 697–708.

[22] P. Golle, J. Staddon, and B. Waters, "Secure conjunctive keyword search over encrypted data," in *ACNS*, 2004.

[23] P. Wang, H. Wang, and J. Pieprzyk, "Keyword field-free conjunctive keyword searches on encrypted data and extension for dynamic groups," in *International conference on cryptology and network security*. Springer, 2008, pp. 178–195.

[24] J. W. Byun, D. H. Lee, and J. Lim, "Efficient conjunctive keyword search on encrypted data storage system," in *European Public Key Infrastructure Workshop*. Springer, 2006, pp. 184–196.

[25] A. Guttman, *R-trees: A dynamic index structure for spatial searching*. ACM, 1984, vol. 14, no. 2.

[26] P. Ciaccia, M. Patella, and P. Zezula, "M-tree: An efficient access method for similarity search in metric spaces," in *VLDB*, 1997.

[27] A. Khan, P. Yanki, B. Dimcheva, and D. Kossmann, "Towards indexing functions: Answering scalar product queries," in *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*. ACM, 2014, pp. 241–252.

[28] X. Ji and J. E. Mitchell, "Branch-and-price-and-cut on the clique partitioning problem with minimum clique size requirement," *Discrete Optimization*, vol. 4, no. 1, pp. 87–102, 2007.

[29] M. Li, S. Yu, N. Cao, and W. Lou, "Authorized private keyword search over encrypted data in cloud computing," in *2011 31st International Conference on Distributed Computing Systems*. IEEE, 2011, pp. 383–392.

[30] B. Wang, W. Song, W. Lou, and Y. T. Hou, "Inverted index based multi-keyword public-key searchable encryption with strong privacy guarantee," in *2015 IEEE Conference on Computer Communications (INFOCOM)*. IEEE, 2015, pp. 2092–2100.

[31] L. W. Falls, "The beta distribution: a statistical model for world cloud cover," *Journal of Geophysical Research*, vol. 79, no. 9, pp. 1261–1264, 1974.

[32] D. Boneh and B. Waters, "Conjunctive, subset, and range queries on encrypted data," in *Theory of Cryptography Conference*. Springer, 2007, pp. 535–554.

2168-7161 (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: Hong Kong University of Science and Technology. Downloaded on April 17,2022 at 10:16:44 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2020.2975175, IEEE Transactions on Cloud Computing

[33] P. Golle, J. Staddon, and B. Waters, "Secure conjunctive keyword search over encrypted data," in *International Conference on Applied Cryptography and Network Security*. Springer, 2004, pp. 31–45.

[34] C. Bösch, P. Hartel, W. Jonker, and A. Peter, "A survey of provably secure searchable encryption," *ACM Computing Surveys (CSUR)*, vol. 47, no. 2, p. 18, 2015.

[35] M. Li, S. Yu, N. Cao, and W. Lou, "Authorized private keyword search over encrypted data in cloud computing," in *2011 31st International Conference on Distributed Computing Systems*. IEEE, 2011, pp. 383–392.

[36] S. Lai, S. Patranabis, A. Sakzad, J. K. Liu, D. Mukhopadhyay, R. Steinfeld, S.-F. Sun, D. Liu, and C. Zuo, "Result pattern hiding searchable encryption for conjunctive queries," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2018, pp. 745–762.

[37] Y. Zhang, J. Katz, and C. Papamanthou, "All your queries are belong to us: The power of file-injection attacks on searchable encryption," pp. 707–720, 2016.

[38] C. Gentry and D. Boneh, *A fully homomorphic encryption scheme*. Stanford University Stanford, 2009, vol. 20, no. 09.

[39] M. Van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan, "Fully homomorphic encryption over the integers," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2010, pp. 24–43.

[40] Z. Brakerski and V. Vaikuntanathan, "Efficient fully homomorphic encryption from (standard) lwe," *SIAM Journal on Computing*, vol. 43, no. 2, pp. 831–871, 2014.

[41] C. Dwork, "Differential privacy: A survey of results," in *International Conference on Theory and Applications of Models of Computation*. Springer, 2008, pp. 1–19.

[42] X. Ding, W. Yang, K.-K. R. Choo, X. Wang, and H. Jin, "Privacy preserving similarity joins using mapreduce," *Information Sciences*, vol. 493, pp. 20–33, 2019.

[43] A. Boldyreva, N. Chenette, Y. Lee, and A. Oneill, "Order-preserving symmetric encryption," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, 2009, pp. 224–241.

[44] S. R. Oliveira and O. R. Zaiane, "Privacy preserving clustering by data transformation." in *Embrapa Informática Agropecuária-Artigo em anais de congresso (ALICE)*. In: SIMPÓSIO BRASILEIRO DE BANCO DE DADOS, 18., 2003, Manaus. Anais, 2003.

[45] R. A. Popa, C. Redfield, N. Zeldovich, and H. Balakrishnan, "Cryptdb: protecting confidentiality with encrypted query processing," in *SOSP*, 2011.

[46] W. K. Wong, D. W.-l. Cheung, B. Kao, and N. Mamoulis, "Secure knn computation on encrypted databases," in *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*. ACM, 2009, pp. 139–152.

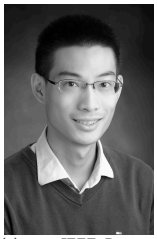
[47] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," *IEEE Transactions on parallel and distributed systems*, vol. 25, no. 1, pp. 222–233, 2014.

[48] X. Ding, P. Liu, and H. Jin, "Privacy-preserving multi-keyword top-k similarity search over encrypted data," *IEEE Transactions on Dependable and Secure Computing*, vol. 16, no. 2, pp. 344–357, 2017.

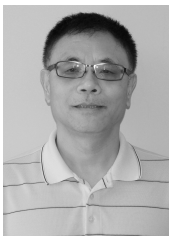
[49] B. Wang, S. Yu, W. Lou, and Y. T. Hou, "Privacy-preserving multi-keyword fuzzy search over encrypted data in the cloud," in *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*. IEEE, 2014, pp. 2112–2120.

[50] W. Lin, K. Wang, Z. Zhang, and H. Chen, "Revisiting security risks of asymmetric scalar product preserving encryption and its variants," in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2017, pp. 1116–1125.

[51] W. Lin, K. Wang, Z. Zhang, A. W. Fu, R. C.-W. Wong, and C. Long, "Class indistinguishability for outsourcing equality conjunction search," in *International Conference on Cloud Computing*. Springer, 2019, pp. 253–270.



Weipeng Lin received the BSc degree in software engineering from the Xiamen University in 2012, and the PhD degree in computing science from Simon Fraser University in 2019. He is currently a lecture at the School of Artificial Intelligence, Shenzhen Polytechnic. His research interests include cloud security, data privacy and big data.

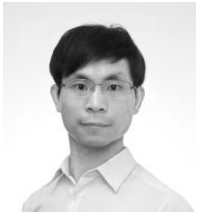


Ke Wang received the PhD degree from the Georgia Institute of Technology. He is currently a professor at Simon Fraser University. His research interests include database technology, data mining, and knowledge discovery, with emphasis on massive data sets, graph and network data, and data privacy. He has published more than 100 research papers in database, information retrieval, and data mining conferences.

Zhilin Zhang received his PhD in Computing Science at Simon Fraser University. He is currently a software develop engineer in Amazon AWS Security. His research widely lies in the security and privacy aspect of the cloud computing, such as secure searching, private retrieval and oblivious shuffling of outsourced data.

Ada Waichee Fu received the BSc degree in computer science from the Chinese University of Hong Kong in 1983, and both the MSc and PhD degrees in computer science from the Simon Fraser University in 1986 and 1990, respectively. She is an associate professor of the Department of Computer Science and Engineering, the Chinese University of Hong Kong. Her research interests include database, data mining, and graph data analysis.

Raymong Chi-Wing Wong received the BSc, MPhil and PhD degrees in computer science and engineering from the Chinese University of Hong Kong in 2002, 2004, and 2008, respectively. He is an associate professor of the Department of Computer Science and Engineering, the Hong Kong University of Science and Technology. His research interests include database and data mining.



Chong Long received the PhD degree from the Hong Kong University of Science and Technology in 2015. He is currently an assistant professor at the School of Computer Science and Engineering, Nanyang Technological University. His research interests include database and data mining.

Chunyan Miao is a Professor in the School of Computer Engineering at Nanyang Technological University. Her research focus is on infusing intelligent agents into interactive new media to create novel experiences and dimensions in game design, interactive narrative and other real world agent systems. She has done significant research work in her research areas and published many top quality international conference and journal papers.

2168-7161 (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: Hong Kong University of Science and Technology. Downloaded on April 17,2022 at 10:16:44 UTC from IEEE Xplore. Restrictions apply.