A Multi-user Shared Searchable Encryption Scheme Supporting SQL Query

Mingyue Li[1,2(✉)], Ruizhong Du[3], and Chunfu Jia[1,2]

[1] College of Cyber Science,
Nankai University, Tianjin 300350, China
15630424277@163.com
[2] Tianjin Key Laboratory of Network and Data Security Technology,
Nankai University, Tianjin 300350, China
[3] School of Cyber Security and Computer,
Hebei University, Baoding 071002, China

Abstract. Due to the tremendous benefits of cloud computing, organizations are highly motivated to store electronic records on clouds. However, outsourcing data to cloud servers separates it from physical control, resulting in data privacy disclosure. Although encryption enhances data confidentiality, it also complicates the execution of encrypted database operations. In this paper, we propose a multi-user shared searchable encryption scheme that supports multi-user selective authorization and secure access to encrypted databases. First, we apply the Diffie-Hellman protocol to a trapdoor generate algorithm to facilitate fine-grained search control without incremental conversions. Second, we utilize a private key to generate an encrypted index by bilinear mapping, which makes it impossible for an adversary to obtain trapdoor keywords by traversing the keyword space and to carry out keyword guessing attacks. Third, we use double-layered encryption to encrypt a symmetric decryption key. Only the proxies whose attributes are matched with access control list can obtain the key of decrypted data. Through theoretical security analysis and experimental verifications, we show that our scheme can provide secure and efficacious ciphertext retrieval without the support of a secure channel.

Keywords: Data privacy · Searchable encryption · Structured Query Language (SQL) · Multi-user shared

1 Introduction

With the rapid development of computer technology and internet applications, the demand for data access and the information storage capacity is increasing [1]. Cloud computing enables users to enjoy high-quality services and ubiquitous network access on demand. Outsourcing data to cloud servers separates it from

D. Wang et al. (Eds.): SPNCE 2020, LNICST 344, pp. 406–422, 2021.
https://doi.org/10.1007/978-3-030-66922-5_28

physical control, resulting in data privacy disclosure [2]. To protect user data privacy, plaintext is typically encrypted before being outsourced to clouds. While many organizations (such as governments, hospitals or companies) always store their data in relational databases, data encryption may lead to the fact that widely used structured queries language (SQL) of plaintext databases cannot be directly applied to encrypted data, e.g., the probabilistic encryption generates different ciphertext each time the same plaintext is encrypted and minimizes leaked information. However, it makes encrypted data inaccessible.

There are two ways to construct an SQL query scheme for encrypted data in a database. The first is to directly operate over encrypted database data, and the second is to use an index. The problem of performing calculation directly on encrypted data was put forward long before the application scenario of cloud computing appeared. Rivest et al. [3] presented the concept of full homomorphic encryption to solve this problem. Full homomorphism encryption supports any computation and does not disclose any information about the data itself. It can achieve ideal data confidentiality. However, the performance overhead is very large. Before the performance of full homomorphic encryption reaches practical level, we have to try other methods with lower performance overhead. Curtmola et al. [4] established an index for a whole database based on a pseudo-random function and a bloom filter to improve the query efficiency. Encryption index can quickly locate the records satisfying query conditions by making the encrypted data independent of query operations; thus, the subsequent searchable encryption schemes supporting SQL queries generally adopt this approach.

Unfortunately, existing solutions that support SQL queries have some limitations in terms of multi-user sharing, e.g., since different institutions encrypt indexes with different keys, the traditional schemes construct the token adjustment search scheme based on a key derivation algorithm, that is, for each query from user, data owner has to generate an encryption different of two tokens, which are respectively encrypted by the owner and user for keyword w, to realize the conversion between trapdoors. While the repeated encryption conversion lead to huge computations. Besides, some mechanisms that resist keyword guessing attacks require a data owner to encrypt a special keyword set by public key of authorized users. If the number of authorized users is large, such computation will leads to enormous overhead in terms of computing resources and storage space.

1.1 Our Contributions

In this paper, we propose a multi-user shared searchable encryption scheme supporting the SQL query (MSE-SQ) for secure and effective data sharing. MSE-SQ constructs a reversed index for each encrypted in its database to improve search efficiency. Moreover, the effectiveness of MSE-SQ is verified by theoretical analysis and experimentation. Our contributions are as follows:

– We propose a new trapdoor generation method based on the Diffie-Hellman key exchange protocol, which preserves the search functionality without

frequent key exchanges when a user search different indexes encrypted by different keys. It can reduce vast communication overhead of trapdoor transformation.
– We mix the private key of a data owner into the encrypted index by bilinear mapping to resist the keyword guessing attacks from malicious servers, thus an adversary cannot obtain the trapdoor keywords by traversing the keyword space.
– We design a novel double-layered encryption algorithm based on user's attributes to ensure a secure transmission of symmetric decryption key without the support of a secure channel.

1.2 Related Work

Nowadays, data stored on a server in plaintext no longer meets the requirement of privacy protection; thus, data encryption technology has received increasingly greater attention. However, the encrypted storage of private data produces data query problems; thus, the searchable encryption scheme is proposed. In recent years, searchable encryption mechanisms have been widely studied [5,6,8–10]. However, these mechanisms focus on the retrieval of text files and cannot meet certain requirements, i.e., they do not support basic SQL queries for stored and queried ciphertext data in the database.

At present, there are many researches on searchable database encryption systems, which are mainly divided into two types: direct operations on encrypted database data and index methods.

Direct Operations on Encrypted Database Data. Raluca et al. [11] used onion encryption to combine different encryption methods, that is, they encrypt a data field by multiple nested encryption methods. The more secure the outer layer encryption method is, the weaker the function of this layer is. Mit et al. [12] used homomorphic encryption to support secure aggregation queries, wherein operations on aggregate fields are obtained by calculating the aggregation of aggregate fields on the server side and decrypting them on the client side. However, the encryption efficiency is very low when Mit et al.'s method is used to encrypt large-scaled data. Wong et al. [13] proposed a data interoperability scheme based on secure multi-party computing that uses the RSA encryption method to change the key re-encryption; thus, their scheme supports the operations such as addition, comparison and connection, etc. However, Wong et al.'s scheme does not consider the problem of multi-user key updates. Liu et al. [14] proposed a practical and secure homomorphic order-preserving encryption (FHOPE) scheme to solve efficiency problems associated with multiple encryption combinations. FHOPE allows cloud servers to perform complex SQL queries with different operators on encrypted data without repeated encryption. FHOPE directly operates on encrypted database data. Its efficiency is low. Subsequently, the encryption index method was established wherein encrypted data are independent of query operations. This enables the records satisfying query conditions to be quickly localed by an index.

Encrypted Index. Encrypted indexes enable records that meet the query conditions to be quickly located and improve query efficiency. Golle et al. [6] first proposed a keyword ciphertext query scheme that establish an index for each data table and define a security model called IND-CKA (indistinguishability against adaptive chosen keyword attack) for the secure index. This index scheme can quickly determine matched data table and does not reveal unnecessary matching information. Curtmola et al. [4] established an index for a whole database based on a pseudo-random function and a bloom filter which improved the query efficiency. However, it cannot achieve fine-grained access control. Li et al. [15] designed an encrypted index with a tree structure. They transform ciphertext sequence relations into data structure by a cryptography method and perform a strict security analysis. Since each node in the tree includes a bloom filter, the disadvantage of the tree index is its high miscalculation rate. To improve upon this database index, Karras et al. [16] offered a self-adaptive concept of database domain and put forward the self-adaptive encrypted index suitable for the range query of a column database. However, there is no strict security analysis of the self-balanced binary tree structure of the self-adaptive scheme. Subsequently, Monir et al. [17] extend the searchable encryption index from a text to an SQL database. This SE index supports the functions of range and boolean queries. The time complexity that judges whether a keyword is contained in a data table is $O(1)$. However, the range query of their scheme is a simple multi-keyword query, which needs to specify the range of values in advance. Thus the efficiency is relatively low for a non-standard query.

2 Preliminaries

In this section, we review some basic cryptographic notions and definitions used later in the paper, such as the notation of our encryption scheme, adversary modeling and security definitions.

2.1 Notations

In a cloud database system, the data owner outsources tremendous amount of two-dimensional table set $T = \{T_1, T_2, \ldots, T_m\}$ to a database server in encrypted form $C = \{C_1, C_2, \ldots, C_m\}$, where m is the number of two-dimensional tables. $R_{i,j}$ can represent a record in the two-dimensional table T_i, where $1 \leq j \leq \#T_i$. The keyword set is extracted from T_i and denoted by $W = \{w_1, w_2, \ldots, w_n\}$, where n is the number of keywords. For each $w_i \in W$, the $A(w_i)$ represents a collection of records containing the keyword w_i. Besides, the proxy of the data owner generates a secure index set I. We use h to denote the number of users and χ to denote the number of attributes of users.

2.2 Adversary Model

To standardize the scope of the study, we present three hypotheses as follows:

- The cloud server faithfully stores the encrypted data of the data owner and responds to the data requests, but it will try to analyze the encrypted data and index of the data owner to obtain privacy information.
- The authorized users are trusted. He/She neither actively disclose the plaintext of the data owner to an unauthorized entity nor actively disclose the obtained decryption information.
- The proxy server is fully trusted, which executes all the user-side functions of our protocol.

2.3 Complexity Assumptions

Assumption 1 *(DDH, Decisional Diffie-Hellman [18]). Let G_1 is multiplicative cyclic groups of a large prime order p and g is the generators of G_1. Given $(g, g^a, g^b, g^c) \in G_1$ as the challenge input, the DDH problem is to distinguish between the distributions (g, g^a, g^b, g^{ab}) and (g, g^a, g^b, g^c), where a, b, c are random in Z_p. If (ε, k) – DDH problem assumption stands on group G_1 and a polynomial time adversary A can solve the problem with a probability $Adv^{DDH}_{A,G_1}(k)$, we say that there is no such adversary A can solve DDH problem with a non-negligible probability.*

$$Adv^{DDH}_{A,G_1}(k) = \left| \begin{matrix} Pr(A(g, g^a, g^b, g^{ab}))- \\ Pr(A(g, g^a, g^b, g^c)) \end{matrix} \right| \leq \varepsilon \tag{1}$$

2.4 Security Definition

Let A be a polynomial bounded adversary and B be a challenger. After A has issued a table and a subset $L \subseteq \{1, 2, ..., n\}$, B responds with two encrypted tables associated with L , then A cannot but distinguish the encrypted tables created by L. Therefore, this security game can indicate a secure goal which it requires that A is unable to deduce the plaintext from other tables. Moreover, the scheme should resist keyword guessing attacks. We define an indistinguishability of ciphertext from limited random (ICLR) game on the basis of traditional security model in [20].

Game 1: The ICLR game is defined between a polynomially bounded adversary A and a challenger B, where adversary A can request the encryption $EncIndex(r, T_i, W)$ of any two-dimensional table T and any search capabilities (trapdoors). The ICLR game process is as follows.

Setup. Adversary A generates the public/private key pairs.

Phase 1. Adversary A requests the encryption $EncIndex(r, T_i, W)$ of any two-dimensional table T and any search capabilities (trapdoors).

Challenge. Phase 1 ends. A chooses a two-dimensional table T, and then sends it to the challenger B. B creates two tables $T_0 = Rand(T, L - \{t\})$ and $T_1 = Rand(T, L)$, where $L \subseteq \{1, 2, ..., n\}$ and a value $t \in L$, and chooses a random bit $b \in \{0, 1\}$, and then gives $EncIndex(r, T_\beta, W)$ to A.

Phase 2. A continues to ask for encrypted tables and capabilities, with the restriction that A may not ask for a capability that is distinguishing for T_0 and T_1. The challenger answers as in Phase 1.

Guess. A outputs $\beta^{'} \in \{0,1\}$ and wins the game if $\beta^{'} = \beta$. The advantage of A winning is defined as $|pr[\beta^{'} = \beta] - 1/2|$.

Definition 1. *If polynomially bounded adversary A wins Game 1 with a negligible advantage, then the scheme is semantically secure in ICLR (Indistinguishability of Ciphertext from Limited Random) games.*

3 SQL Database

3.1 SQL Database

As shown in Table 1, an SQL database organizes data in a set of two-dimensional tables $(T_1, T_2, ..., T_m)$. Each row of a two-dimensional table is a record. Each column is an attribute. Each table item corresponds to a value.

Table 1. Example of an SQL database table

Per-id	Per-sex	Per-age	Per-marital
213781001	Male	20	Never-married
310034025	Female	18	Married-civ-spouse
209020132	Male	39	Widowed Mexico
243020122	Male	54	Married-civ-spouse
...	...	...	...

3.2 Dictionary Creation

There are two types of SQL data. One is insensitive plaintext data, the other is sensitive encrypted data. The former accounts for the majority of the database data. In view of the characteristics of the data, we only use algorithm 1 to create an appropriate keyword set for sensitive data.

We consider the following select operation for our data-base queries: SELECT attributes FROM table WHERE conditions. This query retrieves the records that match the conditions of a two-dimensional table (or view). Authorized users can perform the following three types of queries:

(1) Simple keyword query: Only contains one condition, $p_1 =$ <attr = val>.
(2) Conjunctive query: Contains multiple query conditions, such as SELECT attributes FROM table WHERE $p_1 \wedge p_2 \wedge \cdots \wedge p_x$, where $p_i =$ <attr = val> $(1 < i < n)$. If the SQL query statement contains two query criteria: $p_1 =$ <Per-sex = 'Male'> and $p_2 =$ <Per-age = '20'>, the proxy server generates $trapdoor_1$ = trapdoor(tk, Per-sex = 'Male') and $trapdoor_2$ = trapdoor(tk, Per-age = '20').
(3) Disjunctive query: The select operation is SELECT attributes FROM table WHERE $p_1 \vee p_2 \vee \cdots \vee p_x$. The SQL query is converted to an SE query by the same process as the connection query.

Algorithm 1. $\{W, A(w)_{w \in W}\} \leftarrow CreateDict(D)$.

Require: database , T;
Ensure: Keyword set, W; Record set ,$A(w_i)_{w_i \in W}$.

```
W = φ ;
for T_i in T do
    for attr do
        w ←< attr = val >;
        if w not in W then
            W = W ∪ {w};
        end if
    end for
end for
for w_i ∈ W do
    determine A(w_i);
end for

return W and A(w_i)_{w_i ∈ W}.
```

4 The Detailed Scheme

The system is divided into two modules encompassing multi-user ciphertext retrieval and the decryption of encrypted data.

4.1 System Model

As shown in Fig. 1, there are four entities in the system model: data user (User), proxy server (PS), database server (DBMS) and certification authority (CA). The User refers to the data owner, such as a hospital or medical institution user who is responsible for collecting, classifying data and searching encrypted data through SQL query statements. The PS is responsible for establishing the encrypted index, generating the trapdoor, sharing the secret key and decrypting the data. The DBMS is responsible for storing the encrypted database and executing specific queries according to the user's query requirements. The CA is a trusted certificate authority that provides trusted credentials for users during key sharing.

4.2 Multi-user Fine-Grained Access Control

Since different proxy servers generate different trapdoors and indexes for the same keywords. The query trapdoor is frequently converted when multi-user share data, which leads to vast computation overhead [17]. To this end, we propose a new trapdoor generation method based on the Diffie-Hellman key conversion protocol. It does not convert trapdoors through multiple encryption when multi user share encrypted data that reduces the computation overhead of

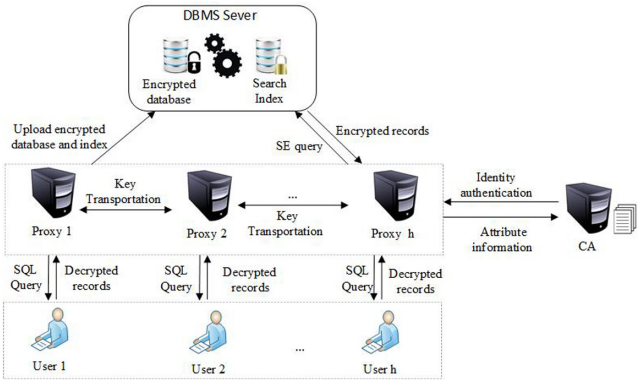

Fig. 1. System model

ciphertext retrieval. Moreover, only authorized users can search encrypted data to achieve fine-grained access control, which improves the security of the system. The specific algorithms of our scheme are as follows.

(1) $params \leftarrow Setup(1^k)$: Given a security parameter k and output system parameters $params = <q, g, \hat{e}, P, G, H_1, H_2>$, where q is a large prime number related to security parameter k, G and G_T are cyclic groups of order q, g is the generator of group G, $\hat{e} : G \times G \rightarrow G_T$ is an efficient non-degenerate bilinear map. H_1 and H_2 are hash functions, where $H_1 : \{0,1\}^* \rightarrow G_q$, $H_2 : G_{q^2} \rightarrow \{0,1\}^n$, $H_3 : \{0,1\}^* \rightarrow Z_q$.

(2) $(SK, PK) \leftarrow KeyGen(params)$:
- The proxy server of the data owner uses system parameters to generate the private key $x \in Z_q$ and the public key g^x.
- The proxy server of the data user uses system parameters to generate private key $y \in Z_q$ and public key g^y.

(3) $I \leftarrow EncIndex(r, T_i, W)$: The index generation algorithm specifies that the proxy server of data owner creates a keyword index for each table T_i in the database.
- Randomly select $r \in Z_q$.
- For each w_i in encrypted T_i, let $I_{w_i} = \hat{e}(H_1(w_i)^x, g^r)$ and the encrypted index $I = \{I_{w_1}, \ldots, I_{w_n}\}$.

(4) $C \leftarrow Enc(T, K)$: Proxy servers encrypt data uploaded to the cloud server. For each $T_i \in T$, the data owner runs $Enc_K(T_i)$ to generate ciphertext C_i and ciphertext set $C = \{C_1, C_2, \ldots, C_m\}$.

(5) $Td \leftarrow Trapdoor(g^x, y, Qc)$: It takes public key g^x of the data owner, the private key y of the data user and the query keyword set Qc as the input and executes the flowing steps:
- The proxy server of the data user follows the Elliptic Curve Diffie-Hellman protocol and calculates $(g^x)^y$ using its private key y and public key g^x of the data owner, where $(g^x)^y$ is a point on an elliptic curve. The horizontal

coordinate a and vertical coordinate of this point b are concatenated and hashed to a point in G as $tk = H_3(a||b)$ and used as secret key for generating the trapdoor.

$$tk = H_3(a||b) \tag{2}$$

– The proxy server of the data user uses tk to generate trapdoor Td for $Qc = \{w_1^{'}, w_2^{'}, ..., w_t^{'}\}$, where $1 \leq t \leq n$.

$$Td = \{H_1(w_1^{'})^{tk}, H_1(w_2^{'})^{tk}, ..., H_1(w_t^{'})^{tk}\} \tag{3}$$

(6) $Acd \leftarrow Delegate(x, g^y, C)$: The proxy server of the data owner generates access control data Acd for the data user.
– The proxy server of the data owner uses private key x and public key of user g^y to compute $(g^y)^x$ and obtain (g^{xy}). Then, the horizontal and vertical coordinates of point g^{xy} are joined to calculate $tk = H_3(a||b)$.
– For each encrypted two-dimensional table C_j that the user is allowed access, entry (C_j, ζ_j) is created, where

$$\zeta_j = g^{rx} \tag{4}$$

$$\eta_j = g^{tk} \tag{5}$$

and list $Acd = \{(C_j, \zeta_j, \eta_j)|j \in (1, m)\}$

(7) $Dlist \leftarrow Search(Acd, Td)$: For the encrypted two-dimensional table C_j queried by the data user, the DBMS obtains the search result $Dlist$ by computing $\frac{\hat{e}(H_1(w_i^{'})^{tk}, \zeta_j)}{\hat{e}(\eta_j, g)} \stackrel{?}{=} I_{w_i^{'}}$, $1 \leq i \leq t$.

4.3 Secure Key Transmission with Dual Encryption

In the application of a public key searchable encryption scheme, considering the efficiency of plaintext encryption and decryption, another symmetric encryption scheme is typically used to encrypt and decrypt plaintext to protect data privacy. In our encryption scheme, we encrypt the tables in the database with symmetric encryption. As the security of symmetric algorithms depends on the encryption key used, we design a double-layered encryption algorithm based on attribute encryption to ensure the secure transmission of symmetric keys.

The key transfer process is shown in Fig. 2. After receiving the ciphertext encrypted by the proxy server of the data owner (PS_j) from the DBMS, the proxy server of the data user (PS_i) requests the attribute private key from the CA. The CA first confirms the data user's identity according to his/her attributes list and then generates the attribute private key ak for the proxy PS_i, where the identity list is shown in Table 2. Simultaneously, the CA sends the attribute information about the proxy PS_i to PS_j. PS_i requests the key from PS_j. The proxy PS_j uses double-layered encryption algorithm to encrypt the key requested by the PS_i and send it to PS_j. In addition, PS_j uses the private key sk and the attribute private key ak to decrypt the corresponding key. Finally, PS_i decrypts the encrypted two-dimensional tables with the decrypted keys to obtain the plaintext data. The main key sharing algorithms are as follows.

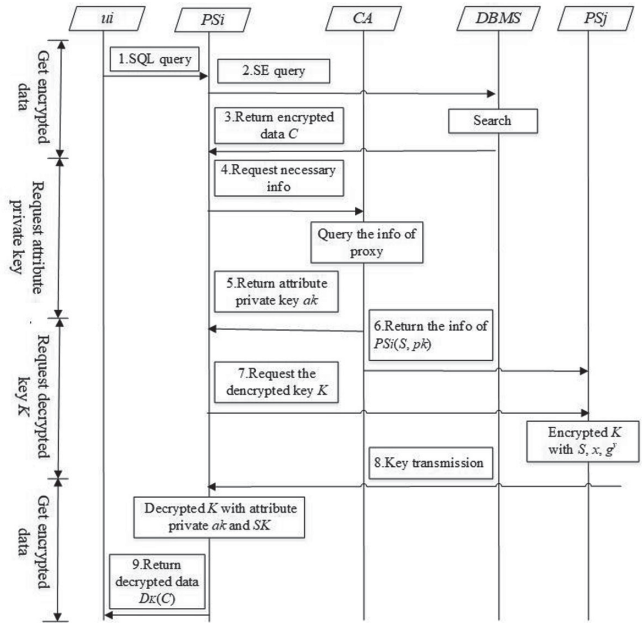

Fig. 2. The progress of decryption data

(1) $(MSK, P) \leftarrow Init(params)$: The CA randomly selects $MSK \in Z_q$ and generates $P = g^{MSK}$.

(2) $ak \leftarrow Skeygen(S, MSK, g^y)$: Take attribute set S and public key g^y of PS_j as input and complete the flowing steps:
- Map the attributes of PS_j to point $Q = H_1(attr_1 \text{——} attr_2 ... \text{——} attr_\chi)$.
- Generate attribute private key $ak = Q^{MSK}$.
- Use public key g^y to encrypt ak and send it to PS_j, The proxy then uses the private key to decrypt ak.

(3) $c \leftarrow EncryTrans(S, x, g^y, K)$: The proxy of the data owner encrypts symmetric key $K \in \{0,1\}^N$ using the private key x, attributes and the public key g^y of the data user.
- Map the attributes of the data user to point Q as done for the CA.
- Randomly select $\vartheta \in Z_q$.
- Use g^{xy} to encrypt ϑ to obtain $\vartheta^{'} = g^{xy} + \vartheta$ and let $c = <\vartheta^{'}, K \oplus H_2(\hat{e}(Q, P))>$, where $\hat{e}(Q, P) \in G_T$.

(4) $K \leftarrow DecryKey(c, d, y, g^x)$: Decrypt K using private key y of the data user and the attribute private key ak given by the CA.
- For $c = <\vartheta^{'}, K \oplus H_2(\hat{e}(Q, P))>$, decrypt $\vartheta^{'}$ with the private key y and the public key g^x of the data owner and obtain $\vartheta = \vartheta^{'} - g^{xy}$.
- Let $V = K \oplus H_2(\hat{e}(Q, P))$, decrypt V with attribute private key ak: $V \oplus H_2(\hat{e}(Q, P)) = K$.

Table 2. Attributes list of proxies

Proxy ID	$attr_1$	$attr_2$	$attr_3$	...	$attr_\chi$
ID_1	$S_{1,1}$	$S_{2,1}$	$S_{3,1}$	...	$S_{\chi,1}$
ID_2	$S_{1,2}$	$S_{2,2}$	$S_{3,2}$	...	$S_{\chi,2}$
ID_3	$S_{1,3}$	$S_{2,3}$	$S_{3,3}$	...	$S_{\chi,3}$
...	...	...	...	...	...

5 Security Analysis

5.1 Confidentiality of Queries

Theorem 1. *Our proposed MSE-SQ scheme is* $(1-1/2e^n q_T n)$ *secure under the ICLR game without random oracle if DDH assumption is intractable.*

Proof. Let A be a polynomially bounded adversary with an advantage ε in Game 1, there is an emulator B that has advantage of $\varepsilon^{'} = \varepsilon/e^n q_T n$ at minimum for solving the DDH problem, where q_T is the number of times the trapdoor is queried in $O(\text{time}(A))$.

Setup. A randomly selects element $x \in Z_q$ and set it as the private key, then he generates his public key g^x. Finally, he sents the private key $y \in Z_q$ and the public key g^y to B.

Phase 1. A forms the following queries:

- Index query. A sends an index generation request to B for the keyword set $W = \{w_1, w_2, ..., w_n\}$ in the two-dimensional table T_i. To simulate the EncIndex algorithm, B selects a random element γ_j in G for each key w_j. Furthermore, B selects random number u_i in Z_q and returns the searchable ciphertext $I = \{\hat{e}(\gamma_1^x, g^{u_i}), ..., \hat{e}(\gamma_\tau^x, g^{bu_i}), ..., \hat{e}(\gamma_n^x, g^{u_i})\}$, $\zeta_j = g^{xu_i}$ and $\eta_j = g^{tk}$, where $tk = H_3(a_x||b_y)$, a_x and b_y are the horizontal coordinate and vertical coordinate, respectively, of g^{xy} on the elliptic curve.
- Trapdoor query. In order to test the search algorithm, A sends a trapdoor generation request $Qc = \{w_1^{'}, w_2^{'}, ..., w_t^{'}\}$ to B. After receiving the trapdoor generation request, B asserts $\{w_1^{'}, w_2^{'}, ..., w_t^{'}\} \subseteq W$ and computes $Td^* = \{H_1(w_1^{'})^{tk}, H_1(w_2^{'})^{tk}, ..., H_1(w_t^{'})^{tk}\}$ for A.

Challenge. Once adversary A decides that Phase 1 is over, it sends a triple $\{T_i, t, L\}$ to emulator B, in which $L \subseteq \{1, ..., n\}$ and $t \in L$. Notice that the random element τ is independent of the location t selected. B responds as follows:

- If $j \neq t$ and $j \in L$, let $h_j = k_j$, where k_j is a random element in G.
- If $j \neq t$ and $j \notin L$, let $h_j = \hat{e}(\gamma_j^x, g^a)$.
- If $j = t$:
  - $t = \tau$, let $h_t = \hat{e}(\gamma_t^x, g^c)$.
  - $t \neq \tau$, let $h_t = k_t \in Z_q$

– Send $\{h_1, ..., h_n, g^{ax}, g^{tk}\}$ to A.

Phase 2. Adversary A issues more queries, where A cannot query the keywords that have been queried. Emulator B responds as in Phase 1.

Guess. A outputs a guess $\beta^{'} \in \{0,1\}$. If $\beta^{'} = \beta$ and $\beta = 1$, then (g^a, g^b, g^c) is considered to be a DDH tuple. So when $\tau = t$, B can prove that (g^a, g^b, g^c) is a DDH tuple. The proof is as follows.

$$\begin{aligned}
&\frac{\hat{e}(\gamma_\tau^x, g^{bu_i})}{\hat{e}(g^{xu_i}, g^{tk})} = \frac{\hat{e}(\gamma_t^x, g^c)}{\hat{e}(g^{ax}, g^{tk})} \Leftrightarrow \frac{\hat{e}(\gamma_\tau, g)^{xbu_i}}{\hat{e}(g,g)^{xu_i tk}} = \frac{\hat{e}(\gamma_j, g)^{xc}}{\hat{e}(g,g)^{axtk}} \\
&\Leftrightarrow \frac{\hat{e}(\gamma_\tau, g)^b}{\hat{e}(g,g)^{tk}} = \frac{\hat{e}(\gamma_j, g)^c}{\hat{e}(g,g)^{atk}} \Leftrightarrow \frac{\hat{e}(\gamma_\tau, g)^b}{\hat{e}(g,g)^{tk}} = \frac{\hat{e}(\gamma_j, g)^c}{\hat{e}(g,g)^a \hat{e}(g,g)^{tk}} \\
&\Leftrightarrow \hat{e}(\gamma_\tau, g)^b \hat{e}(g,g)^a = \hat{e}(\gamma_t, g)^c \\
&\Leftrightarrow \hat{e}(\gamma_\tau, g^{ab}) = \hat{e}(\gamma_t, g^c) \\
&\Leftrightarrow g^{ab} = g^c
\end{aligned} \tag{6}$$

If $\beta = 0$, the ciphertext on position t is a random value. Thus, B cannot prove that (g^a, g^b, g^c) is a DDH tuple. To ensure a random encryption at position t, the challenge must not be a DDH-tuple. Whereas, the A's advantage in winning the ICLR game is the same as that of B settles the DDH challenge.

The above steps compose the simulation process of the security proof. We analyze the probability of B solving the DDH problem. Event E_1 indicates that B responds to the trapdoor query request for n keywords of adversary A. Event E_2 denotes that B gives up responding in the progress of the challenge. When q_T is large enough, $Pr(E_1) = 1/e^n$, $Pr(E_2) = 1/q_T n$. Thus, the probability for B solving the DDH problem is $\varepsilon^{'} = \varepsilon \cdot Pr[E_1 \cup E_2] \geq \varepsilon / e^n q_T n$, where $\varepsilon / e^n q_T n \in [0, 1/2e^n q_T n]$ is negligible. Therefore, the probability that our encryption scheme is secure is at least $(1 - 1/2e^n q_T n)$ in Game 1.

5.2 Security of Secret Keys

Theorem 2. *Neither the certification authority, nor any other user can get the symmetric key of a document.*

Proof. Before uploading data to the DBMS, all the database tables are encrypted by standard symmetric cryptography. To enable only the authorized users to decrypt the encrypted records, the symmetric key for each table is encrypted to be transmitted by the following formula.

$$K \oplus H_2((Q, g^{MSK})^{\vartheta}) \tag{7}$$

Even if the certification authority CA has the attribute private key ak of a proxy, it cannot recover ϑ because it does not have the private key SK of the proxy. Since ϑ is random, the CA cannot get K and $H_2(\hat{e}(Q, P))$ must be obtained. Furthermore, if a malicious attacker steals the private key SK, it means the attacker can get $\vartheta^{'}$. However, it does not have the corresponding attribute private key ak and cannot compute $H_2(\hat{e}(Q, P))$. That is to say, the malicious attacker cannot restore K. Therefore, the proposed encryption scheme effectively ensures the secure transmission of secret keys and enables only the authorized users to decrypt the encrypted records.

6 Experimental Analysis

6.1 Experimental Environment

Implementation. Our SE scheme is implemented based on the architecture shown in Fig. 1 and tested with three identically configuration PCs. The system environment is based on a windows 7(64-bit) system, the hardware configuration is an Intel(R)Core(TM)i7-6700 (3.40 GHz) processor equipped with 8 GB of memory and a fast Ethernet network (1 Gbps). We use Alibaba cloud storage platform, a domestic cloud storage provider, to build a storage system. Multiple virtual machines son a single PC are used to simulate the proxies. All the functions are implemented in Java. The User and CA are deployed on separate PCs. Communication between the proxy and CA is mainly realized via socket transmission.

In the key sharing scenario, the OpenSSL library is used to generate 4096-bit PKE key pairs and an elliptic curve is used to implement IBE encryption. The PBC library is called to implement bilinear pairing. The search uses the PL/Java in Java, which is an additional feature of the server-side program. The code is packed into the JAR and the JAR is further loaded into the SQL server backstage for direct searches.

Data. The data set was extracted by Barry Becker from the census database in 1994 called Adult [19]. This database lists 16,281 records and comprises eight attributes, including identity information (Per-id), age (Per-age), gender (Per-sex), monthly wage (Per-wage) and other sensitive information composition.

6.2 Query Security

The experiment makes use of the privacy protection level of the scheme of the following formula, where $0 < p(T_i) < 1, \sum_{i=1}^{m} p(T_i) = 1$. The larger the $H(T)$, the lower the possibility of privacy leakage. The value of $H(T)$ is determined if there are no other external conditions.

$$H(T) = -\sum_{i=1}^{m} p(T_i) log_2 p(T_i) \tag{8}$$

Figure 3 shows a comparison of the privacy protection of the SE schemes, wherein our scheme provides the highest and SDSE [17] provides the lowest level of privacy protection. This is because our scheme can provide secure ciphertext retrieval without the support of a secure channel.

6.3 Multi-user Keyword Query Performance

Index Construction. As shown in Fig. 4, the time required by the three SE schemes to establish an encrypted index, which increases with the number of keywords, is compared experimentally. As the most basic scheme, SDSE adopts symmetric encryption index, which has the shortest time requirement. Our scheme

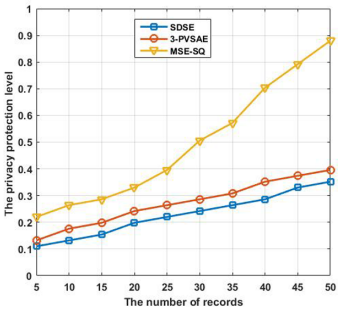

Fig. 3. Comparison of privacy protection level

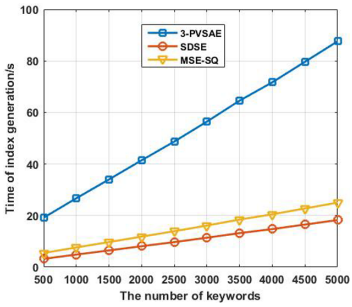

Fig. 4. Encryption index time requirements

Table 3. Compare of single keyword query time

Single keyword query	$\|A_W\|$	Search time		
		3-PVSAE	SDSE	MSE-SQ
Per-sex = 'Male'	10860	8.231 s	1.005 s	1.543 s
Per-age = '25'	354	2.503 s	0.025 s	0.031 s
Per-id = '2016126911'	1	0.392 s	0.009 s	0.012 s

Table 4. Time of conjunctive query and disjunctive query

Conjunctive query	$\|A_W\|$	Search time
Per-sex = 'Male' and Per-age = '39'	808	0.564 s
Per-age = '25' and Per-marital = 'Never-married'	247	0.215 s
Per-sex = 'Male' and Native-country = 'Peru'	11	0.015 s
Disjunctive query	$\|A_W\|$	Search time
Per-age = '25' or Per-marital = 'Never-married'	5541	0.819 s
Native-country = 'Peru' or Per-age = '39'	405	0.158 s
Per-id = '310034025' or Native-country = 'Peru'	15	0.013 s

and 3-PVSAE [7] both are public key encryption schemes, but 3-PVSAE requires the longest time to generate its encryption index due to multiple modular exponentiation.

Trapdoor Generation. Figure 5 presents an experimental comparison of time required by the three SE schemes for trapdoor generation. The time required by the 3-PVSAE scheme is nearly 3.5 times greater than our scheme. The time cost of the SDSE scheme is the lowest.

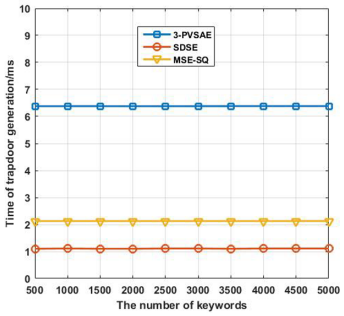

Fig. 5. Time of Trapdoor generation

Query Overhead. Table 3 shows an experimental time cost comparison of a single keyword query in each of the three SE schemes. The search time of the 3-PVSAE scheme is the longest and at nearly 7 times greater than our scheme using the same data set and index structure. The time of the SDSE scheme is the shortest. In addition, the query time overhead of the conjunctive and disjunctive queries are shown in Table 4.

7 Conclusion and Future Work

In this paper, we present a multi-user shared ciphertext retrieval scheme for SQL databases, which supports multi user using SQL statements to perform boolean query. To reduce the enormous computing burden of repeated encryption conversions of trapdoors, we use the new trapdoor generation method based on the Diffie-Hellman key exchange protocol to implement queries without converting the trapdoors and transmitting the trapdoor keys of different users. Furthermore, the double-layered encryption method provides secure retrieval without the support of a secure channel. Through theoretical security analysis, our SE scheme is statistically secure in ICLR game. We present a framework that embeds the search algorithm into SQL sever. This framework is evaluated in terms of practicality using a census database.

Due to the variety of cloud applications available nowadays, our future work consists in implementing automatic search matches for a greater number of protocols.

Acknowledgments. This work is supported by National Key R&D Program of China (2018YFA0704703); National Natural Science Foundation of China (61972215, 61702399, 61972073); Natural Science Foundation of TianJin (17JCZDJC30500).

References

1. Xu, L., Yuan, X., Wang, C., Wang, Q., Xu, C.: Hardening database padding for searchable encryption. In: INFOCOM, pp. 2503–2511 (2019)

2. Kasra Kermanshahi, S., Liu, J.K., Steinfeld, R., Nepal, S.: Generic multi-keyword ranked search on encrypted cloud data. In: Sako, K., Schneider, S., Ryan, P.Y.A. (eds.) ESORICS 2019. LNCS, vol. 11736, pp. 322–343. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-29962-0_16
3. Ronald, R.L., Len, A., Michael, D.L.: On data banks and privacy homomorphisms. Found. Secure Comput. **4**, 169–180 (1978)
4. Curtmola, R., Garay, J.A., Kamara, S., Ostrovsky, R.: Searchable symmetric encryption improved definitions and efficient constructions. In: Proceedings of the 13th ACM Conference on Computer and Communications Security, pp. 79–88 (2006)
5. Song, D.X., Wagner, D., Perrig, A.: Practical techniques for searches on encrypted data. In: Proceedings of the 21st IEEE Symposium on Security and Privacy 2000, pp. 44–55 (2000)
6. Golle, P., Staddon, J., Waters, B.: Secure conjunctive keyword search over encrypted data. In: Jakobsson, M., Yung, M., Zhou, J. (eds.) ACNS 2004. LNCS, vol. 3089, pp. 31–45. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24852-1_3
7. Zhang, R., Xue, R., Yu, T., Liu, L.: PVSAE: a public verifiable searchable encryption service framework for outsourced encrypted data. In: 2016 IEEE International Conference on Web Services (ICWS)
8. Xia, Z., Wang, X., Sun, X., Wang, Q.: A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data. IEEE Trans. Parallel Distrib. Syst. **27**, 951–963 (2016)
9. Patel, S., Persiano, G., Yeo, K.: Symmetric searchable encryption with sharing and unsharing. In: Lopez, J., Zhou, J., Soriano, M. (eds.) ESORICS 2018. LNCS, vol. 11099, pp. 207–227. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-98989-1_11
10. Zhang, Z., Wang, J., Wang, Y., Su, Y., Chen, X.: Towards efficient verifiable forward secure searchable symmetric encryption. In: Sako, K., Schneider, S., Ryan, P.Y.A. (eds.) ESORICS 2019. LNCS, vol. 11736, pp. 304–321. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-29962-0_15
11. Popa, R.A., Redfield, C.M.S., Zeldovich, N., Balakrishnan, H.: CryptDB: protecting confidentiality with encrypted query processing. In: Proceedings of the 23rd ACM Symposium on Operating Systems Principles, pp. 85–100 (2011)
12. Catherine, M.S., Nickolai, Z.: CryptDB: protecting confidentiality with encrypted query processing. In: Proceedings of the 23rd ACM Symposium on Operating Systems Principles, pp. 85–100 (2011)
13. Wong, W.K., Kao, B., Cheung, D.W.-L., Li, R., Yiu, S.-M.: Secure query processing with data interoperability in a cloud database environment. In: ACM SIGMOD Conference 2014, pp. 1395–1406 (2014). https://doi.org/10.1145/2588555.2588572
14. Liu, G., Yang, G., Wang, H., Xiang, Y., Dai, H.: A novel secure scheme for supporting complex SQL queries over encrypted databases in cloud computing. Secur. Commun. Netw. **2018**, 7383514:1–7383514:15 (2018). https://doi.org/10.1155/2018/7383514
15. Li, R., Liu, A.X., Wang, A.L., Bruhadeshwar, B.: Fast and scalable range query processing with strong privacy protection for cloud computing. IEEE/ACM Trans. Netw. **24**, 2305–2318 (2016). https://doi.org/10.1109/TNET.2015.2457493
16. Karras, P., Nikitin, A., Saad, M., Bhatt, R., Antyukhov, D., Idreos, S.: Adaptive indexing over encrypted numeric data. In: Proceedings of the 2016 International Conference on Management of Data, pp. 171–183 (2016)

17. Azraoui, M., Önen, M., Molva, R.: Framework for searchable encryption with SQL databases. In: CLOSER, pp. 57–67 (2018)
18. Ning, J., Xu, J., Liang, K., Zhang, F., Chang, E.: Passive attacks against searchable encryption. IEEE Trans. Inf. Forensics Secur. **14**(3), 789–802 (2019)
19. Dua, D., Graff, C.: UCI Machine Learning Repository. http://archive.ics.uci.edu/ml
20. Jiang, S., Zhu, X., Guo, L., Liu, J.: Publicly verifiable Boolean query over outsourced encrypted data. IEEE Trans. Cloud Comput. **7**(3), 799–813 (2019)