Privacy-Preserving Ranked Searchable Encryption Based on Differential Privacy

Yu Zhao, Chungen Xu(✉), Lin Mei, and Pan Zhang

Nanjing University of Science and Technology, Nanjing 210094, JS, China
{zy0326,xuchungen,meilin}@njust.edu.cn

Abstract. Ranked search allows the cloud server to search the top-k most relevant documents according to the relevance score between query keyword and documents, which has been recognized as the most promising way to realize secure search over encrypted database. However, recent studies show that some privacy protection methods commonly used in ranked search, like order-preserving encryption (OPE), have some security problems. In this paper, we first propose a scheme, called privacy-preserving ranked searchable encryption based on differential privacy (DP-RSE). Specifically, we add noise drawn from a Laplace distribution into the relevance score to disturb its value. In this way, no matter how much background the adversary has, he (or she) cannot obtain the true relevance score or ranked order. Moreover, our scheme ensures the correctness of search results with high probability. The experiment results show that our scheme can achieve sub-linear efficiency and the accuracy of search results can reach 94%.

Keywords: Differential privacy · Ranked search · Laplace distribution · Order-preserving encryption

1 Introduction

The huge storage capacity and economic service cost of cloud servers attract more and more clients to outsource private data to them. While in practice, cloud server is not always fully trusted. Considering the privacy of data, it is necessary to encrypt the data before uploading. But this will bring another problem, that is, some common retrieval methods such as the keyword search cannot be directly executed on ciphertexts. Ranked searchable encryption allows the cloud server to search the top-k most relevant documents according to the relevance score between query keyword and documents, which has been recognized as the most promising way to realize secure search over encrypted database. This reduces not only the computational overhead, but also the network burden.

This work was supported by the National Natural Science Foundation of China (No: 62072240) and the National Key Research and Development Program of China (No. 2020YFB1804604).

X. Yuan et al. (Eds.): QShine 2021, LNICST 402, pp. 323–339, 2021.
https://doi.org/10.1007/978-3-030-91424-0_19

Considering that relevance score may reveal the frequency information of the private data, some researchers suggest using order-preserving encryption (OPE) to hide above relevance scores. Here OPE refers to the order-preserving encryption, which can preserve the numerical order of the plaintexts even after encryption. Nevertheless, many recent studies [7,14,15,20,21,23] show that OPE is vulnerable to inference attacks which can reveal the plaintexts accurately. More specifically, the adversary can leverage order information to estimate the expected distribution of the plaintexts and then correlate it with the encrypted data based on the underlying property being preserved [12]. Therefore, it is desired to design a scheme which can disturb the ranked order while optimizing the search results. Some schemes [26,30] propose to mask the real score by adding a certain number of virtual scores to the search results under a fixed safety parameter. However, these suggested solutions are empiric and no theoretical security analysis is provided to guarantee their claims.

To this end, in this paper, our goal is to design a practical scheme to safeguard prior rank schemes while providing meaningful security guarantees. Our initial idea is to introduce the notion of differential privacy which is defined with respect to a privacy parameter ϵ. Compared to cryptographic techniques or k-anonymity, the advantage of differential privacy is that there is no need to provide special attack assumptions, i.e., even if the attacker has the greatest background knowledge, it can provide strong privacy protection [8].

In light of above observations, we summarize our methodology and contribution as follows. First, we propose a ranked search scheme over encrypted cloud data with differential privacy which masks the real relevance score through adding noise drawn from a Laplace distribution. As a result, the relevance score the server obtained is obfuscated. Thus, the server cannot combine with its known background knowledge to infer privacy information. Furthermore, to maintain the utility of the basic rank search scheme, we also investigate how to select approximate ϵ. We prove the security of the proposed scheme and conduct a series of experiments to evaluate its efficiency. The experiments show that the accuracy of search results in our scheme can reach 94% and the search efficiency of our scheme is almost the same as that of the existing schemes.

Organization. The rest of this paper is organized as follows. In Sect. 2, we introduce related work. Section 3 describes the system model, threat model, design goals and some notations used in this paper. In Sect. 4, we present some primitives used in this paper. In Sect. 5, we give the detailed structure of our scheme and give the security proof. We discuss efficiency and accuracy as well as give experimental results in Sect. 6 Finally, we give a brief conclusion in Sect. 7

2 Related Work

Searchable Encryption was first proposed by Song et al. [25], but the search efficiency of this scheme is low. In order to improve the efficiency, Goh et al. [13] proposed an index-based scheme named Z-IDX, which used bloom filter as the index. Curtmola et al. [4] proposed the inverted index, which can achieve

sublinear search time. At the same time, they standardized symmetric searchable encryption and its security, and their proposed SSE-1 and SSE-2 schemes can achieve indistinguishability in non-adaptive and adaptive attack models, separately. Unfortunately, they all have to make use of generic and relatively expensive techniques to achieve dynamization. Kamara et al. [17] first proposed a dynamic SSE scheme to achieve efficient updating in ciphertext. Noted that, all the schemes above need the client decrypt all the returned documents to obtain ones most matching his (or her) interest, which will bring huge cost to the client.

Ranked SSE allows cloud server to quantify and rank-order the relevance of documents in response to any given search query. In this way, the client just needs to decrypt a small part of ciphertexts to obtain ones most matching their interest. Wang et al. [27] proposed the first ranked search scheme in cloud computing environment. In order to improve the practicability, Cao et al. [3] proposed the first multi-keyword ranked searchable encryption scheme based on the secure KNN technique. But the search efficiency of the scheme is not high, the search complexity is linear with the number of documents. To improve the search efficiency, Sun et al. [26] proposed a scheme using the MDB-tree based on the vector space model. But it is hard to support sub-linear search time and efficient updates. Liu et al. [22] proposed a verifiable and dynamic ranked search encryption scheme, which can achieve sub-linear search time. Recently, Yang et al. [31] overcame the shortcomings of KNN technology, and proposed a scheme not require a predefined keyword set and supported keywords in arbitrary languages.

Order-preserving encryption is often used to encrypt the relevance score in ranked search. It was first proposed by Agrawal et al. [1]. This technology can keep the numerical order of the ciphertext consistent with that of plaintext. Boldyreva et al. [2] first proposed order-preserving symmetric encryption (OPSE). Popa et al. [24] proposed a variable order preserving coding scheme (mutable Order Preserving Encoding, mOPE). Different from conventional order-preserving encryption, mOPE encodes sensitive data sequentially. Nevertheless, recent studies show that these commonly used OPE schemes [19,24] are vulnerable to many attacks like: inference attack, frequency analysis, sorting and cumulative attack [15,21].

Differential privacy is a new privacy protection mechanism proposed by Dwork et al. [8] in 2006. In this model, the purpose of privacy protection is achieved by adding noise to data. The advantage is that it can ignore how much background knowledge the attacker has. Traditional differential privacy schemes [9,11] centralize the original data to a center, which is called centralized differential privacy. But in practical applications, it is very difficult to find a truly trusted third-party data collection platform, which greatly limits the application of centralized differential privacy technology. In view of this, in the scenario of untrusted third-party data collectors, local differential privacy [6,18] appears. Nowadays, differential privacy has been adopted in many practical settings, especially in the field of machine learning [16,29]. Some major commercial

organizations like Alibaba [28], Microsoft [5], Google [10] also use differential privacy to protect some sensitive data.

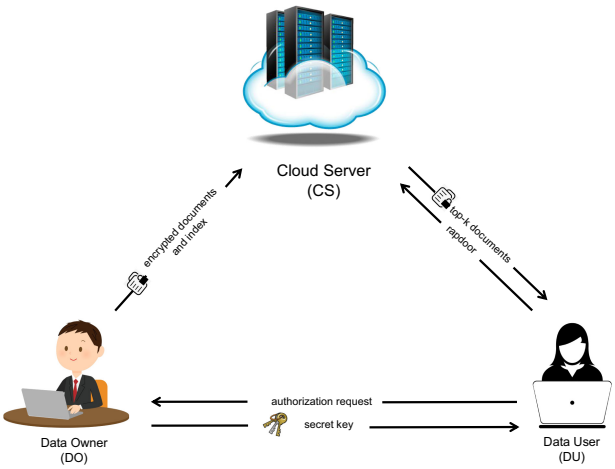

Fig. 1. System model

3 Problem Formulation

3.1 System Model

The system model involves three parties: the data owner (DO), data user (DU), and cloud server (CS), as illustrated in Fig. 1.

Data Owner: The data owner first extracts the keywords from a collection of documents $F = \{F_1, F_2, ..., F_n\}$, then calculates the relevance score between each document and its keywords. After that, he (or she) builds the secure encrypted index I and outsources I to the CS, together with the ciphertext $\mathbf{c}$ of F.

Data User: To search the document containing a specific keyword w, the data user first gets authorization from data owner and obtains the secret key, then generates the trapdoor T_w and submits it to the CS.

Cloud Server: Upon receiving the trapdoor T_w, the cloud server is responsible to search the index I and returns the top-k most relevant documents according to the relevance score between the search keyword w and each documents.

3.2 Threat Model

In this paper, we think that the data owner and user are fully trusted. We consider that the CS is honest-but-curious in our scheme. That is, the CS will

execute the cryptographic protocol correctly, but it will try to learn the privacy information from the executive process.

Known Background Model. In this model, the cloud server knows more statistical information about dataset, except the ciphertext $\mathbf{c}$ and the index I. The cloud server may infer the specific content of keyword or document based on the known trapdoor ranked information of documents.

3.3 Design Goals

- **Ranked Search:** The proposed scheme should allow the CS to sort encrypted documents according to the relevance score between the search keyword and document, then return the ranked results to the user.
- **Differential Privacy:** Our scheme can achieve differential privacy, i.e., no matter how much background knowledge the attacker has, he (or she) will not infer the true ranked information according to the disturbed relevance score.
- **Efficiency:** Above goals on functionality and privacy should be achieved with low communication and computation overhead.

4 Preliminaries

In this section, we present some primitives used in this paper. Table 1 is part of the notations we use in this paper.

Table 1. Notations and descriptions

Notation	Description
F	The collection of n plaintext documents, $F = \{F_1, F_2, ..., F_n\}$
$\mathbf{c}$	The collection of ciphertexts for F, $\mathbf{c} = (C_1, C_2, ..., C_n)$
$F(w_i)$	A sequence of documents containing keyword w_i
F_{ij}	The j-th document in $F(w_i)$
s_{ij}	The real relevance score of F_{ij} about w_i
S_{ij}	The disturbed s_{ij}
ES_{ij}	The encrypted S_{ij}
W	A dictionary of m keywords, $W = \{w_1, w_2, ..., w_m\}$
$\delta(F)$	The collection of distinct keywords in F
A	A search array
T	A search table
I	The encrypted index, $I = (A, T)$

4.1 Symmetric Encryption

Definition 1. *(**Symmetric Encryption**) A symmetric encryption scheme is a collection of three polynomial-time algorithms **SKE** = (**Gen**, **Enc**, **Dec**) such that:*

- ***Gen***(1^λ)***:*** *It takes a security parameter λ as input, and returns a secret key SK.*
- ***Enc***(m, SK)***:*** *It takes a plaintext m and a secret key SK as inputs, then returns a ciphertext c.*
- ***Dec***(c, SK)***:*** *It takes a ciphertext c and a secret key SK as inputs, then returns plaintext m.*

Definition 2. *(**Order-Preserving Function**) For $A, B \subseteq \mathbb{R}$ with $|A| \leq |B|$, a function $f : A \to B$ is order-preserving if for any $i, j \in A$, $f(i) > f(j)$ iff $i > j$.*

Definition 3. *(**Order-Preserving Encryption**) [2] For plaintext-space $\mathcal{D}$, ciphertext-space $\mathcal{R}$ and key-space $\mathcal{K}$, we say a collection of three polynomial-time algorithms **OPE** = $(\mathcal{G}en, \mathcal{E}nc, \mathcal{D}ec)$ is order-preserving if $\mathcal{E}nc$ is an order-preserving function from $\mathcal{D}$ to $\mathcal{R}$ for all K output by $\mathcal{G}en$, which can be described as follows:*

- $\mathcal{G}en(1^\lambda)$ *: It takes a security parameter λ as input, and returns a secret key $K \in \mathcal{K}$.*
- $\mathcal{E}nc(K, m)$***:*** *It takes a secret key K and a plaintext m as inputs, then returns a ciphertext c.*
- $\mathcal{D}ec(K, c)$***:*** *It takes a secret key K and a ciphertext c as inputs, then returns plaintext m.*

4.2 Relevance Score Function

TF× IDF rule is always used to evaluate relevance score in information retrieval. Here, the term frequency (TF) refers to the number of times for a given keyword appears within a document, and the inverse document frequency (IDF) refers to the importance of keywords in the whole document collection, which can obtained through dividing the cardinality of document collection by the number of documents containing the keyword. Without losing generality, we choose the following formula to calculate the relevance score s_{ij} between a document F_i and a keyword w_j:

$$s_{ij} = Score(F_i, w_j) = \frac{1 + \ln f_{i,j}}{|F_i|} \cdot \ln(1 + \frac{n}{|F(w_j)|}) \tag{1}$$

Here F_i denotes the document with length $|F_i|$; w_j denotes a search keyword; $f_{i,j}$ denotes the TF value of w_j in document F_i; n denotes the total number of documents in the collection and $|F(w_j)|$ denotes the number of documents that contain the keyword w_j in the collection.

4.3 Differential Privacy

Definition 4. *(**ϵ-Local Differential Privacy**) A randomized mechanism Q satisfies ϵ-local differential privacy (ϵ-LDP) if and only if for any two inputs x_1, x_2 and any possible output y of Q, we have that*

$$Pr[Q(x_1) = y] \leq e^{\epsilon} \cdot Pr[Q(x_2) = y] \tag{2}$$

In practice, in order to let an algorithm meet the requirement of differential privacy, there are different methods for different problems. Among them, adding the noise which corresponds to Laplace distribution, i.e., noise drawn from a Laplace distribution, to the results is a common method to protect numerical results.

Definition 5. *(**Laplace Distribution**) The distribution with following density function is called Laplace distribution:*

$$f(x|\mu, \lambda) = \frac{1}{2\lambda} e^{\frac{(-|x-\mu|)}{\lambda}} \tag{3}$$

where μ is the positional parameter and λ is the scale parameter.

4.4 Inverted Index

Inverted index [4] is widely used in information retrieval, which is an efficient indexing structure. In this paper, we use the inverted index with the same as [4], which containing a search array A and a search table T. $A[i]$ denotes the value stored at location i in A, which refers to the information about a document containing the keyword w. All nodes in A containing the same keyword w constitute a ranked linked list L_w, which defined as follows. All head node of $L_w (w \in W)$ constitute the search table T.

Definition 6. *(**Ranked Linked List**) For a dictionary $W = \{w_1, w_2, ..., w_m\}$, L_{w_i} is a linked list about keyword w_i, which containing $|F(w_i)|$ nodes $(N_{i1}, N_{i2}, ..., N_{i|F(w_i)|})$. Each node $N_{ij} =< fid_{ij}||ES_{ij}||k_{ij}||addr(N_{i,j+1}) >$ is stored in search array A, where fid_{ij} is the identifier of the j-th document in $F(w_i)$, ES_{ij} is the encrypted disturbed score of fid_{ij} using **OPE**, k_{ij} is the secret key of next node $N_{i,j+1}$ and $addr(N_{i,j+1})$ is the address of the next node $N_{i,j+1}$ in search array A. Each N_{i1} is stored in search table T.*

5 DP-RSE Scheme

In this section, we first introduce the idea of our scheme, and then give the concrete construction. Finally, we give the security proof.

5.1 Intuition Behind Our Construction

Before building the index, we first calculate the relevance score of each document and its keywords. Then we add noise drawn from a Laplace distribution to the relevance score to realize differential privacy. Specifically, let q be the query function, which inputs a query keyword w and then outputs k scores of the top-k most relevant documents in F. For example, $q(w_j) = (s_{j1}, s_{j2}, ..., s_{jk})$ is the output of the function which inputs the keyword w_j. And $s_{j1}, s_{j2}, ..., s_{jk}$ are the k relevance scores between top-k most relevant documents and w_j. Now, we set $q(w_i) = (s_{i1}, s_{i2}, ..., s_{ik})$ is the outputs of the function which inputs another keyword w_i. We define

$$\Delta q = \max_{w_i, w_j \in W} (\sum_{g=1}^{k} |s_{jg} - s_{ig}|). \tag{4}$$

Then the disturbed relevance score of document F_{ij} is

$$S_{ij} = s_{ij} + Lap(\frac{\Delta q}{\epsilon}) \tag{5}$$

where $Lap(\Delta q/\epsilon)$ is a Laplace distribution with $\mu = 0, \lambda = \Delta q/\epsilon$.

To avoid the cloud server recovering the original scores according to some statistics of disturbed relevance score, we use ***OPE*** to encrypt the disturbed score S_{ij} to ES_{ij} before outsourcing to cloud server. We define Q is a function, which inputs the search keyword w_i and then outputs k encrypted relevance scores after being disturbed of the top-k most relevant documents, i.e. $Q(w_i) = (ES_{i1}, ES_{i2}, ..., ES_{ik})$.

5.2 Construction

Now we give the detail of our scheme.

Let **SKE = (Gen,Enc,Dec)** be a secure symmetric encryption scheme. Let $W = (w_1, w_2, ..., w_m)$ be a dictionary of m keywords. $\delta(F)$ is the collection of distinct keywords in the document collection F. Let H be a pseudo-random function and ψ, π be two pseudo-random permutations with the following parameters:

- $H : \{0,1\}^{\lambda} \times \{0,1\}^{l} \rightarrow \{0,1\}^{\lambda + \log_2(r)}$
- $\pi : \{0,1\}^{\lambda} \times \{0,1\}^{l} \rightarrow \{0,1\}^{l}$
- $\psi : \{0,1\}^{\lambda} \times \{0,1\}^{\log_2(r)} \rightarrow \{0,1\}^{\log_2(r)}$

where l means that the keyword can be represented using at most l bits, r is the total size of the encrypted document collection in *min-units* (e.g., one byte).

As can be seen from Fig. 2, our scheme contains four algorithms, which can be described as follows:

- **KeyGen** (1^{λ}): The data owner simply chooses five random λ-bit strings k_e, k_o, k_1, k_2, k_3 separately. Set secret key $SK = \{k_e, k_o, k_1, k_2, k_3\}$, where k_e is the key to encrypt the document and k_o is the key used in **OPE**.

KeyGen(1^λ)

$k_e, k_o, k_1, k_2, k_3 \xleftarrow{R} \{0,1\}^\lambda$
$SK = \{k_e, k_o, k_1, k_2, k_3\}$

BuildIndex(W, F, SK)

Initialize $ctr = 1$
Calculate Δq
for $1 \le i \le \delta(F)$ **do**
  $k_{i0} \xleftarrow{R} \{0,1\}^\lambda$
  for $1 \le j \le |F(w_i)| - 1$ **do**
    $S_{ij} \leftarrow s_{ij} + Lap(\frac{\Delta q}{\epsilon})$
    $ES_{ij} \leftarrow \mathbf{OPE}.\mathcal{E}nc(k_o, S_{ij})$
    $k_{ij} \xleftarrow{R} \{0,1\}^\lambda$
    $N_{ij} =< fid_{ij}||ES_{ij}||k_{ij}||\psi_{k_1}(ctr+1) >$
    $A[\psi_{k_1}(ctr)] \leftarrow \mathbf{SKE}.Enc(k_{i,j-1}, N_{ij})$
    $ctr = ctr + 1$
  end for
  $N_{i,|F(w_i)|} =< fid_{i|F(w_i)|}||ES_{i,|F(w_i)|}||0^\lambda||NULL >$
  $A[\psi_{k_1}(ctr)] \leftarrow \mathbf{SKE}.Enc(k_{i,|F(w_i)|-1}, N_{i,|F(w_i)|})$
  $T[\pi_{k2}(w_i)] \leftarrow < addr(N_{i1})||k_{i0} > \oplus H_{k_3}(w_i)$
end for
Padding A and T, and set $I = (A, T)$
for each F_i in F **do**
  Set $C_k = \mathbf{SKE}.Enc(k_e, F_i)$ and $\mathbf{c} = \mathbf{c} \cup C_k$
end for
Output $(I, \mathbf{c})$

Trapdoor(SK, w)

Search token $T_w = (\pi_{k_2}(w), H_{k_3}(w))$.

Search(I, T_w)

Let $(\alpha, \beta) = T_w$
$\gamma \leftarrow T[\alpha]$
$< \theta||k > \leftarrow \gamma \oplus \beta$
while $\theta \neq NULL$ **do**
  $N_i \leftarrow \mathbf{SKE}.Dec(A[\theta], k)$
  $< fid_i||ES_i||k^{'}||\theta^{'} > \leftarrow N_i$
  $\theta \leftarrow \theta^{'}$, $k \leftarrow k^{'}$
end while
Sort documents according to their scores ES_i
Return the top-k relevant documents

Fig. 2. Our DP-RSE scheme construction

- **BuildIndex** (W, F, SK): The data owner first calculates Δq corresponding to the document set F, and then for each keyword $w_i \in \delta(F)$, randomly selects λ-bit string k_{i0} as the encryption key of N_{i1}. For each document containing w_i, the data owner generates an encrypted disturbance relevance score ES_{ij} by adding noise drawn from a Laplace distribution $Lap(\frac{\Delta q}{\epsilon})$ and using OPE to encrypt it. Next, N_{ij} is generated according to definition 6 and stored in A. Meanwhile, the secret key and the location in A of N_{ij} are stored in T. To avoid revealing the number of distinct keyword in F, we pad the remaining $|W| - |\delta(F)|$ entries in T with the random strings. Meanwhile, we set the size of A to r, and pad remaining entries in A with some random strings. Finally, the data owner generates a sequence of ciphertext $\mathbf{c}$ using the secure symmetric encryption algorithm, then sends $(I, \mathbf{c})$ to the cloud server.
- **Trapdoor** (SK, w): When a data user wants to search some documents containing the keyword w, he (or she) first gets authorization from data owner and obtains the secret key, then generates the trapdoor $T_w = (\pi_{k_2}(w), H_{k_3}(w))$ about the keyword w and sends it to the cloud server.
- **Search** (I, T_w): When receiving the trapdoor T_w, the CS first parses T_w as (α, β), then gets the address θ and decrypted key k of the head node in L_w through $T[\alpha] \oplus \beta$. Using the k, the CS can recover the head node in L_w, then get the address and decrypted key of next node. Repeating the same operation, the CS can obtain the information of all documents containing w. Finally, the CS sorts documents according to their scores ES_i, and returns the top-k documents.

5.3 Security

In this section, we analyze the security of our scheme.

Theorem 1. *The DP-RSE scheme satisfies ϵ-local differential privacy.*

Proof. We inject noise drawn from a Laplace distribution into the relevance score of each document to hide its true value before outsourcing to CS. When performing a search operation, the CS will get the encrypted disturbed relevance scores. According to formula (3), (4) and (5), we get that:

$$\begin{aligned}
\frac{Pr[\mathcal{D}ec(Q(w_i)) = Y]}{Pr[\mathcal{D}ec(Q(w_j)) = Y]} &= \frac{Pr[(S_{i1}, S_{i2}, ..., S_{ik}) = (y_1, y_2, ..., y_k)]}{Pr[(S_{j1}, S_{j2}, ..., S_{jk}) = (y_1, y_2, ..., y_k)]} \\
&= \frac{Pr[(s_{i1} + n_{i1}, s_{i2} + n_{i2}, ..., s_{ik} + n_{ik}) = (y_1, y_2, ..., y_k)]}{Pr[(s_{j1} + n_{j1}, s_{j2} + n_{j2}, ..., s_{jk} + n_{jk}) = (y_1, y_2, ..., y_k)]} \\
&= \frac{Pr[(s_{i1}, s_{i2}, ..., s_{ik}) + (n_{i1}, n_{i2}, ..., n_{ik}) = (y_1, y_2, ..., y_k)]}{Pr[(s_{j1}, s_{j2}, ..., s_{jk}) + (n_{j1}, n_{j2}, ..., n_{jk}) = (y_1, y_2, ..., y_k)]} \\
&= \frac{Pr[(n_{i1}, n_{i2}, ..., n_{ik}) = (y_1, y_2, ..., y_k) - (s_{i1}, s_{i2}, ..., s_{ik})]}{Pr[(n_{j1}, n_{j2}, ..., n_{jk}) = (y_1, y_2, ..., y_k) - (s_{j1}, s_{j2}, ..., s_{jk})]}
\end{aligned}$$

$$= \frac{\prod_{l=1}^{k} e^{\frac{-|y_l - s_{il}|\epsilon}{\Delta q}}}{\prod_{l=1}^{k} e^{\frac{-|y_l - s_{jl}|\epsilon}{\Delta q}}} = e^{\frac{\epsilon}{\Delta q} \sum_{l=1}^{k} (|y_l - s_{jl}| - |y_l - s_{il}|)}$$

$$\leq e^{\frac{\epsilon}{\Delta q} \sum_{l=1}^{k} |s_{il} - s_{jl}|} \leq e^{\frac{\epsilon}{\Delta q} \cdot \Delta q} = e^{\epsilon}$$

According to definition 4, we know that the DP-RSE scheme satisfies ϵ-local differential privacy.

6 Performance Analysis

In this section, we mainly test the efficiency and the accuracy of the scheme. Meanwhile, we compare our scheme with two schemes proposed in [22] and [27] that have the same index construction as our scheme. We implement the scheme on a personal computer with Windows 10 64-bit operating system, 3 GHz AMD R5 CPU, and 16 GB RAM. We first generate a set of data and then use the Cryptography library in JAVA to test the efficiency and the accuracy.

6.1 Efficiency

In this subsection, we mainly discuss the efficiency of our scheme from two algorithms, **Buildindex** and **Search**. We choose SHA256 as the pseudo-random

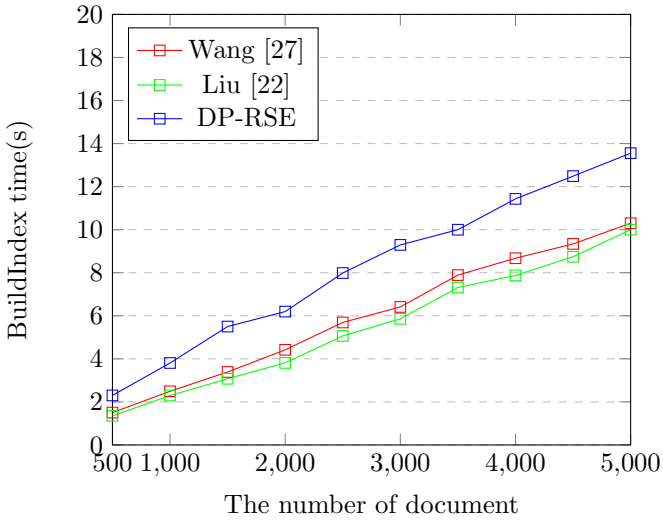

Fig. 3. Set $k = 20$, $\delta(F) = 100$, the time cost of **BuildIndex** towards different number of document for each keyword.

function and pseudo-random permutation in our scheme. Besides, we choose SM4 as the secure symmetric encryption to replace **SKE**. Similar to [27], we use the **OPE** proposed in [2], and set $|\mathcal{R}| = 2^{46}$.

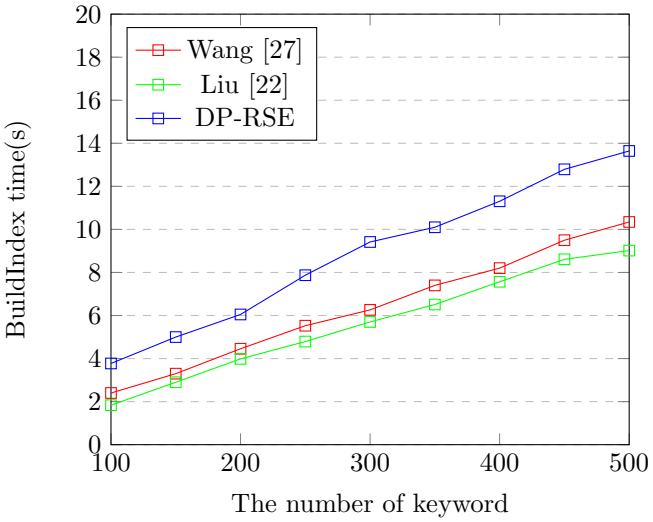

Fig. 4. Fix $k = 20$, $|F(w_i)| = 1000(w_i \in W)$, the time cost of **BuildIndex** towards different number of keyword

In the proposed scheme, the process of **BuildIndex** includes two main steps, i.e.,1) building the search array A and 2) building the search table T. The time cost of **BuildIndex** mainly comes from the number of document and keyword. So we test the time cost for these two factors. On the one hand, we set $k = 20$, the number of distinct keyword in the collection of documents be 100, then test the cost time of **BuildIndex** algorithm by changing the number of document for each keyword. As can be seen from Fig. 3, the result reveals that the time cost of **BuildIndex** algorithm in our scheme is linear with the number of document. The reason is that for each **BuildIndex** operation, there is a tuple inserted to index. On the other hand, we first set $k = 20$, and for each keyword w_i, we set $|F(w_i)| = 1000$. Then we test the cost time of **BuildIndex** algorithm by changing the number of keyword. As can be seen from Fig. 4, the result reveals that the time cost of **BuildIndex** algorithm in our scheme is linear with the number of keyword. Compared with [22] and [27], our scheme costs more time in **BuildIndex**. The difference is mainly caused by adding noise drawn from a Laplace distribution to the relevance score.

In the **Search** algorithm, we only consider the single keyword search. When a client wants to search the top-k documents containing a specified keyword, he (or she) just needs to send the trapdoor to the cloud server. Then the server search for the corresponding entries through A and T. Finally, returning the

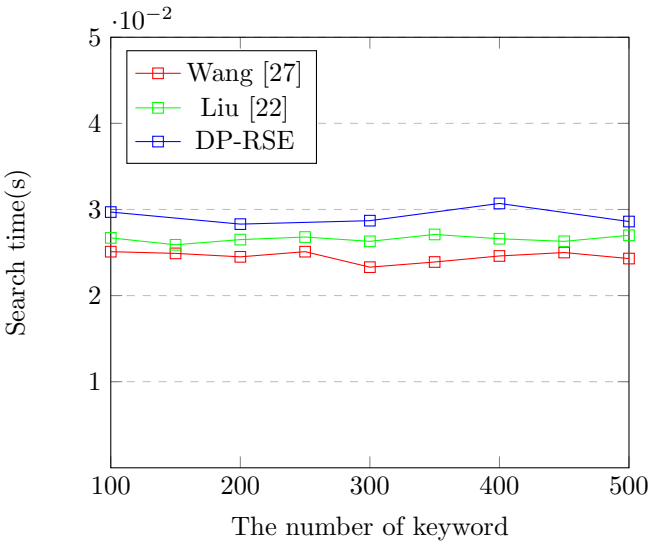

Fig. 5. Set $k = 20$, $|F(w_i)| = 1000(w_i \in W)$, the time cost of **Search** towards different number of keyword

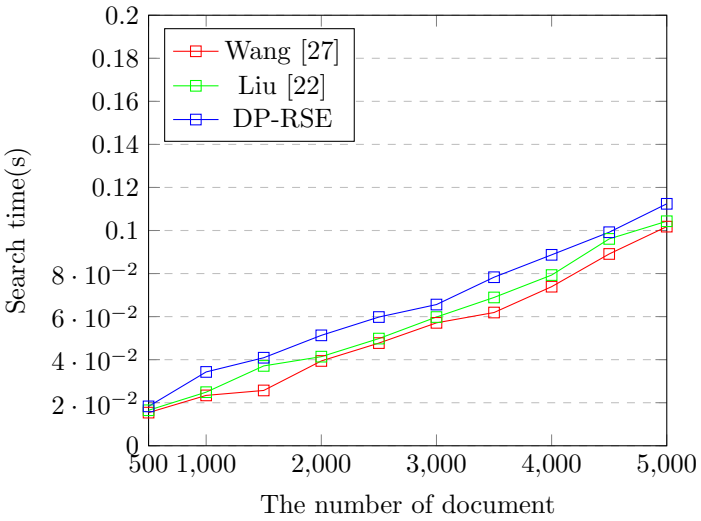

Fig. 6. Fix $k = 20, \delta(F) = 100$, the time cost of **Search** towards different number of document of each keyword.

top-k documents to the client. As can be seen from the Fig. 5 and Fig. 6, the time cost is mainly caused by the number of document containing the specified keyword. In Fig. 6, we set $k = 20$, $\delta(F) = 100$, and test the time cost of search for different number of document containing the keyword. We get that the time cost of **Search** algorithm in our scheme is linear with the number of document containing the specified keyword. Meanwhile, by comparison, the result reveals that the cost time of **Search** algorithm in our scheme is almost the same as [22] and [27].

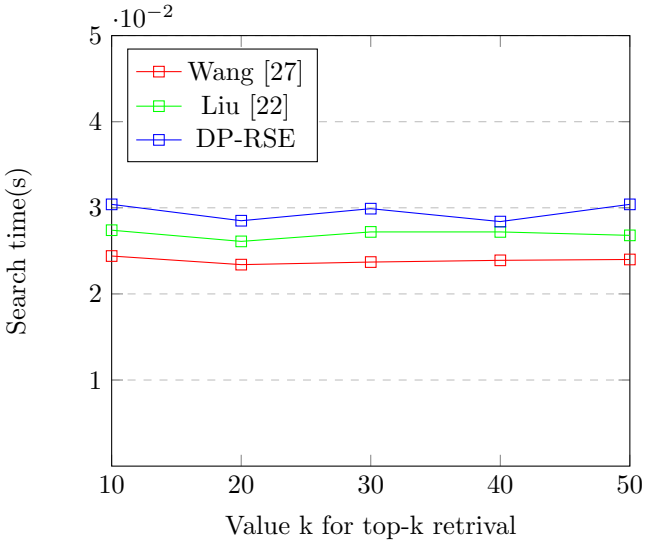

Fig. 7. Set $|F(w_i)| = 1000(w_i \in W)$, $\delta(F) = 100$, the time cost of **Search** towards different value of k

Finally, we set $|\delta(F)| = 100, |F(w_i)| = 1000(w_i \in W)$, and change the value of k for top-k search. As can be seen from Fig. 7, the search time is about 0.03s for different value of k in our scheme. The result reveals that our scheme is practical.

6.2 Accuracy

In order to measure the accuracy of the output results in our scheme, we define a measure as accuracy $P = k^{'}/k$, where the $k^{'}$ is the number of real top-k documents that are returned by the CS. We set the value of k be 20 and evaluate the accuracy of the scheme by changing ϵ. As can be seen in Fig. 8, we change ϵ from 0.01 to 1.2, and when $\epsilon = 1.2$, the accuracy can reach 94%. As shown in Fig. 8, the higher value of ϵ, the higher accuracy. But from the notion of differential privacy, we can see that the higher value of ϵ, the worse privacy. So we should weigh the value of ϵ according to the demand of privacy and accuracy.

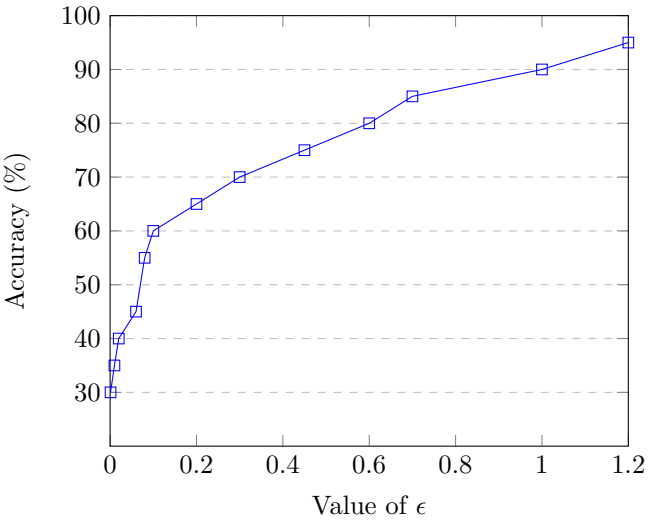

Fig. 8. Set $|F(w_i)| = 1000(w_i \in W)$, $\delta(F) = 100$, the **Accuracy** for different value of ϵ

7 Conclusion

OPE is a common means to protect privacy in ranked searchable encryption, but some studies show that **OPE** scheme will bring privacy leakage. To solve this problem, in this paper, we introduce the notion of differential privacy to the ranked searchable encryption and propose a privacy-preserving ranked searchable encryption scheme. In our scheme, we add noise drawn from a Laplace distribution to the relevance score between the search keyword and document to realize the differential privacy. In this way, no matter how much background knowledge the attacker has, he (or she) will not obtain the information of plaintext according to the relevance score. The experiments show that our scheme is efficient and practical. In the future work, we will be committed to exploring multi-keyword ranked search scheme over encrypted cloud data based on differential privacy.

References

1. Agrawal, R., Kiernan, J., Srikant, R., Xu, Y.: Order-preserving encryption for numeric data. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, pp. 563–574. ACM, New York (2004)
2. Boldyreva, A., Chenette, N., Lee, Y., O'Neill, A.: Order-preserving symmetric encryption. In: Joux, A. (ed.) Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, 2009. LNCS, vol. 5479, pp. 224–241. Springer, Heidelberg (2009)
3. Cao, N., Wang, C., Li, M., Ren, K., Lou, W.: Privacy-preserving multi-keyword ranked search over encrypted cloud data. In: 30th IEEE International Conference

on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, pp. 829–837. IEEE, New York (2011)
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. ACM, New York (2006)
5. Ding, B., Kulkarni, J., Yekhanin, S.: Collecting telemetry data privately. In: Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems, pp. 3571–3580. Neural Information Processing System (2017)
6. Duchi, J.C., Jordan, M.I., Wainwright, M.J.: Local privacy and statistical minimax rates. CoRR abs/1302.3203 (2013). http://arxiv.org/abs/1302.3203
7. Durak, F.B., DuBuisson, T.M., Cash, D.: What else is revealed by order-revealing encryption? In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, pp. 1155–1166. ACM, New York (2016)
8. Dwork, C.: Differential Privacy. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP 2006. LNCS, vol. 4052, pp. 1–12. Springer, Heidelberg (2006). https://doi.org/10.1007/11787006_1
9. Dwork, C., Roth, A.: The algorithmic foundations of differential privacy. Found. Trends Theor. Comput. Sci. **9**(3–4), 211–407 (2014)
10. Erlingsson, Ú., Pihur, V., Korolova, A.: RAPPOR: randomized aggregable privacy-preserving ordinal response. In: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, pp. 1054–1067. ACM, New York (2014)
11. Friedman, A., Schuster, A.: Data mining with differential privacy. In: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 493–502. ACM, New York (2010)
12. Fuller, B., Varia, M., Yerukhimovich, A., et al.: Sok: cryptographically protected database search. In: 2017 IEEE Symposium on Security and Privacy, pp. 172–191. IEEE Computer Society, Washington (2017)
13. Goh, E.: Secure indexes. IACR Cryptol. ePrint Arch. 2003, 216 (2003). http://eprint.iacr.org/2003/216
14. Grubbs, P., Lacharité, M., Minaud, B., Paterson, K.G.: Learning to reconstruct: statistical learning theory and encrypted database attacks. In: 2019 IEEE Symposium on Security and Privacy, pp. 1067–1083. IEEE Computer Society (2019)
15. Grubbs, P., Sekniqi, K., Bindschaedler, V., Naveed, M., Ristenpart, T.: Leakage-abuse attacks against order-revealing encryption. In: 2017 IEEE Symposium on Security and Privacy, pp. 655–672. IEEE Computer Society, Washington (2017)
16. Ji, Z., Lipton, Z.C., Elkan, C.: Differential privacy and machine learning: a survey and review. CoRR abs/1412.7584 (2014). http://arxiv.org/abs/1412.7584
17. Kamara, S., Papamanthou, C., Roeder, T.: Dynamic searchable symmetric encryption. In: Proceedings of the 2012 ACM Conference on Computer and Communications Security, pp. 965–976. ACM, New York (2012)
18. Kasiviswanathan, S.P., Lee, H.K., Nissim, K., Raskhodnikova, S., Smith, A.D.: What can we learn privately? CoRR abs/0803.0924 (2008). http://arxiv.org/abs/0803.0924
19. Kerschbaum, F., Schröpfer, A.: Optimal average-complexity ideal-security order-preserving encryption. In: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, pp. 275–286. ACM, New York (2014)

20. Lacharité, M., Minaud, B., Paterson, K.G.: Improved reconstruction attacks on encrypted data using range query leakage. In: 2018 IEEE Symposium on Security and Privacy, pp. 297–314. IEEE Computer Society, Washington (2018)
21. Li, K., Zhang, W., Yang, C., Yu, N.: Security analysis on one-to-many order preserving encryption-based cloud data search. IEEE Trans. Inf. Forensics Secur. **10**(9), 1918–1926 (2015)
22. Liu, Q., Tian, Y., Wu, J., Peng, T., Wang, G.: Enabling verifiable and dynamic ranked search over outsourced data. IEEE Trans. Serv. Comput. (2019). https://doi.org/10.1109/TSC.2019.2922177
23. Onozawa, S., Kunihiro, N., Yoshino, M., Naganuma, K.: Inference attacks on encrypted databases based on order preserving assignment problem. In: Inomata, A., Yasuda, K. (eds.) IWSEC 2018. LNCS, vol. 11049, pp. 35–47. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-97916-8_3
24. Popa, R.A., Li, F.H., Zeldovich, N.: An ideal-security protocol for order-preserving encoding. In: 2013 IEEE Symposium on Security and Privacy, pp. 463–477. IEEE Computer Society, Washington (2013)
25. Song, D.X., Wagner, D.A., Perrig, A.: Practical techniques for searches on encrypted data. In: 2000 IEEE Symposium on Security and Privacy, pp. 44–55. IEEE Computer Society, Washington (2000)
26. Sun, W., Wang, B., Cao, N., et al.: Verifiable privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking. IEEE Trans. Parallel Distrib. Syst. **25**(11), 3025–3035 (2014)
27. Wang, C., Cao, N., Li, J., Ren, K., Lou, W.: Secure ranked keyword search over encrypted cloud data. In: 2010 International Conference on Distributed Computing Systems, pp. 253–262. IEEE Computer Society, Washington (2010)
28. Wang, T., Ding, B., Zhou, J., et al.: Answering multi-dimensional analytical queries under local differential privacy. In: Proceedings of the 2019 International Conference on Management of Data, pp. 159–176. ACM, New York (2019)
29. Wei, K., et al.: Federated learning with differential privacy: algorithms and performance analysis. IEEE Trans. Inf. Forensics Secur. **15**, 3454–3469 (2020)
30. Xia, Z., Zhu, Y., Sun, X., Chen, L.: Secure semantic expansion based search over encrypted cloud data supporting similarity ranking. J. Cloud Comput. **3**, 8 (2014)
31. Yang, Y., Liu, X., Deng, R.H.: Multi-user multi-keyword rank search over encrypted data in arbitrary language. IEEE Trans. Depend. Secur. Comput. **17**(2), 320–334 (2020)