Journal of Parallel and Distributed Computing 135 (2020) 127–139

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

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

Privacy-preserving range query over multi-source electronic health records in public clouds

Jinwen Liang[a], Zheng Qin[a,*], Sheng Xiao[a], Jixin Zhang[a], Hui Yin[a,b], Keqin Li[a,c]

[a] College of Computer Science and Electronic Engineering, Hunan University, Changsha, 410082, China
[b] College of Applied Mathematics and Computer Engineering, Changsha University, Changsha, 410022, China
[c] Department of Computer Science, State University of New York, New Paltz, NY, 12561, USA

ARTICLE INFO

Article history:
Received 19 January 2019
Received in revised form 3 July 2019
Accepted 28 August 2019
Available online 25 September 2019

Keywords:
Electronic health record
Order-preserving encryption
Privacy-preserving range query

ABSTRACT

Range query is an important data search technique in cloud-based electronic healthcare (eHealth) systems. It enables authorized doctors to retrieve target electronic health records (EHRs) that are generated and outsourced by patients from the cloud server. In reality, patients always encrypt their EHRs before outsourcing, making the range query impossible. In this paper, we identify three threats in real cloud-based eHealth systems, i.e., privacy leakage, frequency analysis, and identical data inference. To capture the security properties that resist these threats, we define a security notion of indistinguishability under multi-source ordered chosen plaintext attack (IND-MSOCPA). Then, we propose a multi-source order-preserving encryption (MSOPE) scheme for cloud-based eHealth systems to enable range queries over encrypted EHRs from multiple patients. Security analysis proves that the MSOPE scheme is IND-MSOCPA secure. We also conduct comprehensive performance evaluations, which demonstrate the high efficiency of the MSOPE scheme.

1. Introduction

Characterized by convenient storage and management, electronic health records (EHRs), which have shown great potential in digitization and visualization in electronic healthcare (eHealth) systems [11], are capable to provide efficient care coordination and enhanced healthcare quality. By utilizing online health record management systems such as Microsoft HealthVault[1] and Zebra-Health,[2] patients can outsource their EHRs to a remote cloud and manage their digital health records anytime anywhere via the Internet [38]. In cloud-based eHealth systems, multiple patients can easily share their EHRs with doctors to enhance the healthcare quality [27]. Meanwhile, authorized doctors can perform queries on EHRs provided by hundreds of thousands of patients, and later collect qualified biomedical data to build a diagnosis model [12,16].

As an important technique for collecting qualified biomedical data, range query returns a set of interesting EHRs between an upper bound and a lower bound, helping doctors to investigate specific disease. As shown in Table 1, heart disease EHRs provided by multiple patients are collected to a remote cloud-based eHealth system. When a doctor wants to investigate adolescent heart disease, he/she will perform a range query at the age column with $0 \leq \text{Age} \leq 16$ and obtain qualified EHR rows with No. 2, 4, 5, 6. Meanwhile, when another doctor wants to investigate the relationship between gender and heart disease, he/she will query EHR whose gender column is "male", i.e., the gender column equals '1', and therefore performs a range query on EHRs via setting the upper bound and the lower bound equal to '1', namely, $1 \leq \text{Gender} \leq 1$. Thus, the range query is a powerful technique to query qualified biomedical data on EHRs.

However, the risk of patients' EHRs breach weakens the desire of using the cloud-based eHealth system. A cloud server is always considered as a semi-trusted party and therefore patients will concern about the unauthorized use for their EHRs stored in the public cloud [18]. At the same time, doctors will also worry about the privacy leakage when searching the EHRs on a semi-trusted cloud server, because searching operations may leak the content of EHRs. Therefore, the content of EHRs should be protected when using the cloud-based eHealth system. With such security constraints, a significant amount of regulations and laws such as Health Insurance Portability and Accountability Act (HIPAA) and the European Data Protection Directive 95/46/EC have been proposed for managing and sharing EHRs [11]. Apart from the aforementioned regulations, it is desirable to design efficient and privacy-preserving range query schemes to protect the privacy of EHR contents as well as to enable range query over EHRs collected from multiple patients. We consider two requirements for

* Corresponding author.
E-mail address: zqin@hnu.edu.cn (Z. Qin).
1 http://www.healthvault.com.
2 https://www.zebrahealth.com.

https://doi.org/10.1016/j.jpdc.2019.08.011
0743-7315/© 2019 Elsevier Inc. All rights reserved.

Table 1
EHRs that stored in a cloud-based eHealth system.

No.	Name	Gender (0/1)	Age	Heart Disease (0/1)
1	Alice	0 (female)	26	0 (No)
2	Bob	1 (male)	16	1 (Yes)
3	Cathy	0 (female)	23	0 (No)
4	David	1 (male)	14	1 (Yes)
5	Eve	0 (female)	15	0 (No)
6	Frank	1 (male)	7	0 (No)

privacy-preserving range query schemes in cloud-based eHealth systems: (1) privacy preservation, i.e., unauthorized users cannot obtain the content of EHRs; and (2) functionality and efficiency, i.e., authorized users can perform efficient range query over the protected EHRs.

Order-preserving encryption (OPE) enables efficient range query on encrypted data, which balances the efficiency and security [2,3]. With an important characteristic that the ciphertext and plaintext are in the same order, order-preserving encryption has been widely used in encrypted databases [26,35]. Recently, plenty of OPE schemes have been proposed for a single data source scenario, which involves a data provider and a cloud server [13–15,25,29]. Most of the works are more of theoretic attempts to push the security notions to the limit, such as *indistinguishability under ordered chosen plaintext attack* (IND-OCPA). However, the single data source schemes are not applicable in cloud-based eHealth systems because such schemes cannot support privacy-preserving range queries over EHRs from multiple patients, which is an essential functionality in cloud-based eHealth systems.

To support multiple patients cases, multi-source (or abbreviated as multi-provider or multi-user) order-preserving encryption schemes have been used in cloud-based eHealth systems [35,36]. Multi-source order-preserving encryption schemes not only protect the privacy of EHR contents but also enable efficient range query over multi-source encrypted EHRs. Most of the existing multi-source schemes [32,35,36] focus on the practicality but leak both the value of plaintext and distance between any two plaintexts [4]. Meanwhile, these schemes suffer from frequency analysis threat. As a comparison, the security feature of frequency hiding has been implemented in Kerschbaum's single-source scheme [13] but not in most of existing multi-source schemes.

In this paper, we first identify three threats from real cloud-based eHealth systems, i.e., *privacy leakage*, *frequency analysis*, and *identical data inference*. To capture the security that thwarts these threats, we define a security notion called *indistinguishability under multi-source ordered chosen plaintext attack* (IND-MSOCPA). Then, we propose a multi-source order-preserving encryption (MSOPE) scheme for cloud-based eHealth systems which enables doctors to perform privacy-preserving range queries over encrypted EHRs from multiple patients. This work extends our previous research in [17] by improving the encryption efficiency and enhancing the functionality for achieving privacy-preserving range query.

We summarize our contributions as follows.

- We define a security notion of IND-MSOCPA, which captures the security against threats in real cloud-based eHealth systems, i.e., *privacy leakage*, *frequency analysis*, and *identical data inference*.
- We propose the MSOPE scheme for cloud-based eHealth systems, which enables doctors to perform range queries over outsourced ciphertexts (i.e., encrypted EHRs). The MSOPE scheme is built on a secure comparing protocol with a minimal number of homomorphic encryption operations, which significantly improves its computational efficiency.

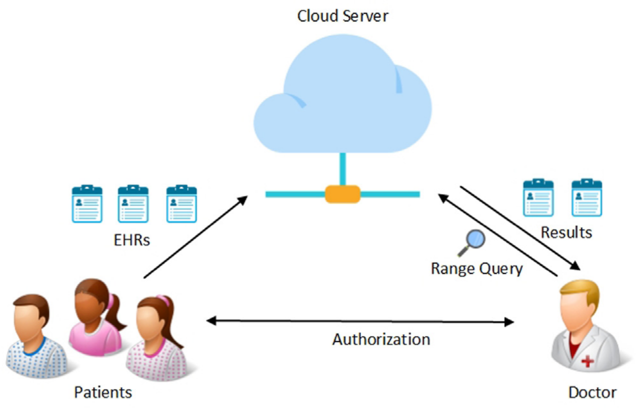


Fig. 1. Cloud-based eHealth system.

- We present a formal security proof to demonstrate that the MSOPE scheme is IND-MSOCPA secure. Extensive experiments show that the MSOPE scheme is more efficient than the primary version of this work in terms of computational overhead [17].

The remainder of this paper is organized as follows: Section 2 presents the system model, threat model, and design goals. Section 3 presents the preliminaries and definitions. Section 4 elaborates the proposed scheme. Section 5 provides performance analysis and evaluations. Section 6 shows related works. Section 7 concludes this paper. We also provide Appendices A and B to verify the correctness of the proposed secure comparing protocol and provide formal security proof for the proposed MSOPE scheme, respectively.

2. Models and design goals

2.1. System model

We consider a cloud-based eHealth system in Fig. 1, which involves three different parties, i.e., patients, a cloud server, and a doctor.

Cloud Server ($\mathcal{CS}$): The cloud server stores the EHRs collected from multiple patients and provides range query service for an authorized doctor.

Patients ($\mathcal{P} = \{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_k\}$): Patients are owners of EHRs, who outsource their EHRs to the cloud server, and authorize some doctors to access their EHRs. We assume that there are k patients, and each patient $\mathcal{P}_i \in \mathcal{P}$ is an EHR data source.

Doctor ($\mathcal{D}$): The doctor $\mathcal{D}$ is an authorized data user, who submits a range query request for EHRs and then obtains the corresponding results.

In our system model, EHRs are considered as private data to the owner of it. Therefore, the content of each EHR can only be exposed to the data owner or the authorized data user, i.e., the corresponding patient $\mathcal{P}_i$ and the doctor $\mathcal{D}$. Namely, $\mathcal{P}_i$ can only manage his/her EHRs stored in $\mathcal{CS}$, but should not access to other patients' EHRs. Therefore, $\mathcal{P}_i$ is an authorized user to his/her EHRs but an unauthorized user to other patients' EHRs. $\mathcal{CS}$ needs to provide range query service to $\mathcal{D}$ but it should not access to any EHRs, namely, $\mathcal{CS}$ is an unauthorized user to EHRs. The doctor $\mathcal{D}$ can perform range queries to EHRs stored in $\mathcal{CS}$ and obtain the corresponding results. Namely, $\mathcal{D}$ is an authorized user to EHRs stored in $\mathcal{CS}$.

2.2. Threat model

We adopt the Honest-but-Curious (HbC) model, which has been widely adopted in privacy-preserving search work in cloud computing. The HbC adversary will follow the protocol honestly but be curious about the data content. We assume that each patient in our scheme is HbC adversary, who is only authorized to access his/her EHRs but curious about the content of other patients' EHRs. Meanwhile, the cloud server is considered as a HbC adversary, who is curious about the content of patients' EHRs and the doctor's query. The doctor is considered as an honest user because he/she is authorized to perform range query to the patients' EHRs.

To enable efficient and privacy-preserving range queries on EHRs, we design multi-source order-preserving encryption, whose ciphertexts preserve the order information of EHRs. Without leakage of auxiliary information, the order information of EHRs cannot recover the encrypted EHRs [8], and therefore we assume that the order information of EHRs is not private information for patients. We identify three threats in cloud-based eHealth systems as follows.

Privacy leakage. Privacy leakage means that a cloud server may recover the encrypted EHRs to the plaintext form if the encrypted EHRs leak additional information other than the order information. To understand how the cloud server can recover the encrypted EHRs by using additional information, consider the scheme in [35], whose core idea is described as follows. For a data provider i, $l_{i,d}$ and $u_{i,d}$ are pre-defined as the lower limit and the upper limit of the OPE ciphertext for d respectively, where $l_{i,d} < u_{i,d}$. The encryption of plaintext d for data provider i is $l_{i,d} + (C_d \ mod \ (u_{i,d} - l_{i,d}))$, where C_d is the AES encryption of d. Namely, for data provider i, the OPE encryption scheme in [35] maps a plaintext d to a pre-defined interval $[l_{i,d}, u_{i,d}]$. Since $l_{i,d}$ and $u_{i,d}$ are reused many times, the fixed interval $[l_{i,d}, u_{i,d}]$ will be leaked because the AES encryption is a pseudo-random function. Suppose an attacker obtains two ciphertexts c_1 and c_2 of d, which are encrypted by data provider i. Namely, $c_1 = l_{i,d}+(C_d \ mod \ (u_{i,d}- l_{i,d}))$, $c_2 = l_{i,d} + (C'_d \ mod \ (u_{i,d} - l_{i,d}))$. Since AES encryption is a pseudo-random function, $C_d \neq C'_d$. Assume that $c_1 \leq c_2$. Then the fixed interval $[l_{i,d}, u_{i,d}]$ is partially leaked, because $[c_1, c_2] \subseteq [l_{i,d}, u_{i,d}]$. Note that the fixed interval would be fully leaked when $c_1 = l_{i,d}$ and $c_2 = u_{i,d}$. The adversary then could recover the ciphertext c^* to the plaintexts form d if $c^* \in [c_1, c_2]$, because $l_{i,d}$ and $u_{i,d}$ would be reused many times. The later scheme in [36] suffers from the same threat, which would leak more information other than the order information. Consider that the field "age" of EHRs are encrypted by methods with additional leakage, the encrypted values may be recovered by the cloud server and the content of "age" will be leaked.

Frequency analysis. Frequency analysis means that a cloud server may analyze the distribution of encrypted EHRs and further recover the encrypted EHRs to the plaintext form because some order-preserving encryption schemes reveal the distribution information of EHRs and the content of EHRs are not distributed uniformly. To understand how the cloud server can learn the content of EHRs from data distributions, we provide an example of encrypting the field "gender" of EHRs from a gynaecology hospital. It is normal that the field "gender" of EHRs contain only "male" or "female". In a gynaecology hospital, the registered women would much more than men, and therefore gender "female" will appear much more frequently than "male". Some existing order-preserving encryption schemes generate the same ciphertexts for the same plaintexts, such as schemes in [25] and [14]. The encrypted field "gender" of EHRs may be recovered by the cloud server because the encrypted EHRs reveals the distribution of "gender" and the cloud server learns that the keyword "female" appears more frequently than "male" in a gynaecology hospital.

Identical data inference. Identical data inference means that unauthorized patients may try to recover the content of EHRs by searching the identical encrypted EHRs. Some existing order-preserving encryption schemes in a single data source scenario generate the same ciphertexts for repeated plaintexts, such as the schemes in [25] and [14]. This characteristic would be used by honest-but-curious patients for recovering others' encrypted EHRs. Take the "age" field of EHRs as an example. Assume that both Alice and Bob are 38 years old, and the OPE encryption of "38" is "357". Thus, the encrypted "age" field of both Alice and Bob are "357". The work-flow of identical data inference is described as follows. Alice obtains "357" as the OPE encryption of her age "38". Alice then searches the encrypted "age" field of EHRs, and finds that Bob's encrypted age is also "357". Thus, Alice can infer that Bob is also "38" years old because she owns the same OPE ciphertexts of "38".

2.3. Design goals

Our goal is to design a privacy-preserving range query scheme to protect the data content of EHRs from the adversaries and provide efficient range query services over EHRs in terms of computational overhead. Specifically, the following objectives should be achieved in our proposed scheme.

Privacy preservation. The first design goal is to protect the content of EHRs from the adversaries under the above three threats, i.e., privacy leakage, frequency analysis, and identical data inference.

Functionality and Efficiency. The second design goal is to achieve range queries over multi-source EHRs for the doctor and to improve the computational efficiency than the primary version of this work [17].

3. Preliminaries and definitions

3.1. Preliminaries

We outline some cryptographic preliminaries, which serve as the building blocks of the MSOPE scheme. Let κ be the security parameter.

Symmetric key encryption (SKE). A symmetric key encryption contains three polynomial-time algorithms, i.e., SKE = (SKE. Gen, SKE.Enc, SKE.Dec). We use $w_{i,*}$ to denote an EHR collected from $\mathcal{P}_i$, and $\widehat{w_{i,*}}$ to denote the corresponding encrypted EHR. Each patient $\mathcal{P}_i \in \mathcal{P}$ generates a secret key sk_i. Then, the symmetric key encryption is described as follows.

- **Key Generation:** $sk_i \leftarrow \text{SKE.Gen}(1^\kappa, \mathcal{P}_i)$.
- **Encryption:** $\widehat{w_{i,*}} \leftarrow \text{SKE.Enc}(sk_i, w_{i,*})$.
- **Decryption:** $w_{i,*} \leftarrow \text{SKE.Dec}(sk_i, \widehat{w_{i,*}})$.

Homomorphic encryption (HOM). A homomorphic encryption contains three polynomial-time algorithms, i.e., HOM = (HOM.Gen, HOM.Enc, HOM.Dec). We use $w_{i,*}$ to denote an EHR collected from $\mathcal{P}_i$, and $[\![w_{i,*}]\!]$ to denote the corresponding encrypted EHR. Each patient $\mathcal{P}_i \in \mathcal{P}$ generates a public key PK_i and a private key SK_i of HOM. Then, the homomorphic encryption is described as follows.

- **Key Generation:** $PK_i, SK_i \leftarrow \text{HOM.Gen}(1^\kappa, \mathcal{P}_i)$.
- **Encryption:** $[\![w_{i,*}]\!] \leftarrow \text{HOM.Enc}(PK_i, w_{i,*})$.
- **Decryption:** $w_{i,*} \leftarrow \text{HOM.Dec}(SK_i, [\![w_{i,*}]\!])$.

In this paper, we use the Paillier homomorphic encryption [24] to construct the MSOPE scheme, which is additively homomorphic encryption, satisfying

$$[\![w_1]\!] \cdot [\![w_2]\!] = [\![w_1 + w_2]\!], \tag{1}$$

where w_1 and w_2 are plaintexts in W.

3.2. Definitions

We define that the EHRs collected from multiple patients are merged as a sequence $W = \{w_{*,1}, w_{*,2}, \ldots, w_{*,n}\}$ with n not necessarily distinct numerical data, where the subscript $*$ is a wildcard denoting a patient in $\mathcal{P}$. Namely, if the jth EHR is collected from patient $\mathcal{P}_i$, then $w_{*,j}$ should be expressed as $w_{i,j}$. We assume that each EHR has been transformed to numerical data, because each column of EHRs can be coded to numerical data. For example, name "Bob" can be coded via ASCII code and transformed to "0x426F62". We use n_i to denote the number of EHRs provided by patient $\mathcal{P}_i$, i.e.,

$$n = \sum_{i=1}^{k} n_i. \tag{2}$$

We use W_i to denote an EHR sequence provided by $\mathcal{P}_i$, which involves n_i EHRs. We define that the domain of EHRs is D, namely,

$$1 \le w_{i,j} \le D, \tag{3}$$

where $i = 1, 2, \ldots, k$ and $j = 1, 2, \ldots, n$.

The MSOPE scheme is constructed based on secret states, which could be implemented by a self-balancing binary search tree (AVL tree) [1], because the in-order traversal for an AVL tree reveals the ascending order of data stored in the AVL tree. We use T to denote the AVL tree with N nodes, which is also the secret state of our MSOPE scheme. According to the previous work [17], for a uniformly chosen EHR sequence of size n, the expected number of N is

$$E[N] = \sum_{i=1}^{k} D\left(1 - \left(\frac{D-1}{D}\right)^{n_i}\right), \tag{4}$$

where

$$n = \sum_{i=1}^{k} n_i. \tag{5}$$

The AVL tree T is stored in the cloud and can be accessed by $\mathcal{P}$, $\mathcal{D}$, and $\mathcal{CS}$. In T, each node stores a patient's ID and a SKE ciphertext of an EHR. For instance, for an EHR $w_{i,x}$, the corresponding node of $w_{i,x}$ in T is $(\mathcal{P}_i, \widehat{w_{i,x}})$.

We use $C = \{c_{*,1}, c_{*,2}, \ldots, c_{*,n}\}$ to denote the corresponding MSOPE ciphertext of EHR sequence W, where the subscript $*$ is a wildcard that denotes a patient in $\mathcal{P}$. For example, we use $c_{i,j}$ to denote the corresponding MSOPE ciphertext of $w_{i,j}$. We use M to denote the ciphertext domain of MSOPE, i.e.,

$$0 \le c_{i,j} \le M. \tag{6}$$

The ciphertexts of our scheme are generated by the secret state (an AVL tree) with N values. Let H be the minimum height of an AVL tree, then $M = 2^H$. According to the primary version of this work [17], H satisfies:

$$H = \left\lceil \frac{3}{2}\log_2(N+1) - 1 \right\rceil. \tag{7}$$

In order to store N EHRs on the AVL tree, M should not be less than 2^H. For simplicity, we define M as follows.

$$M = 2^H. \tag{8}$$

Table 2
Important notations.

Notation	Descriptions
$\mathcal{CS}$	The cloud server.
$\mathcal{P}$	The set of k different patients
$\mathcal{P}_i$	The ith patient in $\mathcal{P}$.
$\mathcal{D}$	The doctor.
W	The EHRs sequence.
$w_{i,j}$	The jth EHR in W, which is collected from $\mathcal{P}_i$.
n	The number of values in W.
n_i	The number of EHRs provided by $\mathcal{P}_i$.
D	The domain of W.
T	The AVL tree, which is the secret state of MSOPE.
N	The number of nodes in T.
C	The corresponding MSOPE ciphertext of W.
M	The domain of C.
d	The subscript of data provided by $\mathcal{D}$.
$w_{d,ub}$	The upper limit of $\mathcal{D}$'s query range.
$w_{d,lb}$	The lower limit of $\mathcal{D}$'s query range.
$c_{d,ub}$	The corresponding MSOPE ciphertext of $w_{d,ub}$.
$c_{d,lb}$	The corresponding MSOPE ciphertext of $w_{d,lb}$.

The MSOPE scheme involves a privacy-preserving range query scheme, which enables the doctor to perform range query over the encrypted EHRs. We use a subscript d to denote the doctor $\mathcal{D}$. Furthermore, we use $w_{d,ub}$ and $w_{d,lb}$ to denote the upper bound and lower bound of the doctor's query range, respectively, and $c_{d,ub}$ and $c_{d,lb}$ to denote the corresponding MSOPE ciphertexts generated. Important notations are summarized in Table 2.

The MSOPE scheme contains four polynomial-time algorithms, i.e., **Secret State Generation**, **Encryption**, **Decryption**, and **Range Query**. We define these four algorithms as follows.

- **Secret State Generation:** The secret state generation algorithm is an initialization scheme, which generates an empty AVL tree T and fixes the ciphertext domain M.
- **Encryption:** For an input EHR $w_{i,x}$ provided by patient $\mathcal{P}_i$, the encryption algorithm generates the corresponding MSOPE ciphertext $c_{i,x}$ by inserting a new node t to T and then updates the secret state T.
- **Decryption:** For an encrypted EHR $c_{i,x}$, the decryption algorithm finds the location of $\widehat{w_{i,x}}$ in T, and returns $w_{i,x}$ by decrypting $\widehat{w_{i,x}}$.
- **Range Query:** For an upper bound $w_{d,ub}$ and a lower bound $w_{d,lb}$, the range query algorithm generates $c_{d,ub}$ and $c_{d,lb}$ and returns the desired EHRs $w_{i,x}$ that satisfies $c_{d,lb} \le c_{i,x} \le c_{d,ub}$.

3.3. Security definitions

The security design goal of our MSOPE scheme is to protect the content of EHRs against *privacy leakage*, *frequency analysis*, and *identical data inference* threats. We define a strong security notion for multi-source order-preserving encryption, i.e., *indistinguishability under multi-source ordered chosen plaintext attack* (IND-MSOCPA), which thwarts the aforementioned three threats in cloud-based eHealth systems. Before we define the IND-MSOCPA security notion, we define a multi-source randomized order, which permutates the order of identical EHRs provided by different patients. We define the multi-source randomized order as follows.

Definition 1 (*Multi-source Randomized Order*)**.** Consider a multi-source EHR sequence $W = \{w_{*,1}, w_{*,2}, \ldots, w_{*,n}\}$, a multi-source randomized order $\Pi = \{\pi_{*,1}, \pi_{*,2}, \ldots, \pi_{*,n}\}$ of W satisfies

$$\forall i, j, w_{*,i} < w_{*,j} \Rightarrow \pi_{*,i} < \pi_{*,j},$$

and

$\forall i, j, \pi_{*,i} < \pi_{*,j} \Rightarrow w_{*,i} \leq w_{*,j}$,

where $\forall i \in [1, n], \pi_{*,i} \in [1, n]$ and $\forall i, j \in [1, n], i \neq j \Rightarrow \pi_{*,i} \neq \pi_{*,j}$.

The multi-source randomized order is a permutation of the order of unnecessarily distinct EHRs collected from different patients. Namely, the multi-source randomized order not only reveals the order of distinct EHRs but also randomizes the order of identical EHRs collected from different patients. For example, $\mathcal{P}_i$ provides $W_i = \{1, 2, 4, 7\}$ and $\mathcal{P}_j$ provides $W_j = \{1, 3, 5, 7\}$. The integrating EHR sequences is $W = \{1, 1, 2, 3, 4, 5, 7, 7\}$. Possible multi-source randomized orders for W are $\Pi_1 = \{1, 2, 3, 4, 5, 6, 7, 8\}$, $\Pi_2 = \{2, 1, 3, 4, 5, 6, 7, 8\}$, $\Pi_3 = \{1, 2, 3, 4, 5, 6, 8, 7\}$, and $\Pi_4 = \{2, 1, 3, 4, 5, 6, 8, 7\}$. We can find that the order of EHR '7' is different because it is an identical EHR provided by different patients. In addition, an example with different frequency of Π_1 is $W' = \{1, 2, 3, 3, 5, 5, 7, 8\}$.

Therefore, the leakage of multi-source randomized order from the protected EHRs will not leak the content of EHRs under the above three threats, because:

1. The multi-source randomized order denotes the order information of an EHR sequence, and hence only leaks the order information of EHR sequence, which protects the EHR content from privacy leakage.
2. The multi-source randomized order randomizes the order of identical EHRs provided by different patients and therefore hides the frequency information of EHR contents.
3. The multi-source randomized order randomizes the order of distinct EHRs provided by different patients. Then, the order of identical EHRs provided by different patients will be distinct, and therefore protects the content of EHRs from identical data inference.

The IND-MSOCPA security notion guarantees that MSOPE ciphertexts only leaks the multi-source randomized order, and therefore MSOPE ciphertexts are resistant to privacy leakage, frequency analysis, and identical data inference threats. Namely, the corresponding ciphertexts of two plaintext sequences with the same multi-source randomized order should be indistinguishable.

The IND-MSOCPA security game involves an adversary $\mathcal{A}$, a challenger $\mathcal{C}$, and k patients in $\mathcal{P}$. $\mathcal{A}$ generates two sequences $W^0 = \{w^0_{*,1}, w^0_{*,2}, \ldots, w^0_{*,n}\}$ and $W^1 = \{w^1_{*,1}, w^1_{*,2}, \ldots, w^1_{*,n}\}$ with n EHRs, which have the same multi-source randomized order relation. Namely, when $1 \leq i, j \leq n$, $w^0_{*,i} < w^0_{*,j} \Leftrightarrow w^1_{*,i} < w^1_{*,j}$. Note that the subscript $*$ denotes a patient $\mathcal{P}_* \in \mathcal{P}$, which is defined by $\mathcal{A}$. Namely, when $\mathcal{A}$ defined that the ith EHR is provided by $\mathcal{P}_x$, then $w^0_{*,i} = w^0_{x,i}$ and $w^1_{*,i} = w^1_{x,i}$. We define the IND-MSOCPA security game as follows.

IND-MSOCPA security game.

1 $\mathcal{A}$ sends W^0 and W^1 to the challenger.
2 $\mathcal{C}$ chooses a random bit $b \in \{0, 1\}$.
3 $\mathcal{C}$ and $\mathcal{P}$ engage in n rounds. At round x, where $x = 1, 2, \ldots, n$:
 (1) $\mathcal{C}$ sends $w^b_{i,x}$ to $\mathcal{P}_i$, where $\mathcal{P}_i$ denotes a patient who provides the ith EHR and is defined by $\mathcal{A}$.
 (2) $\mathcal{P}_i$ returns $c_{i,x}$ to $\mathcal{C}$.
4 $\mathcal{C}$ returns the corresponding OPE ciphertext sequence $C = \{c_{*,1}, c_{*,2}, \ldots, c_{*,n}\}$ to $\mathcal{A}$, where the subscript $*$ is a wildcard denoting a patient in P.
5 $\mathcal{A}$ outputs b', its guess for b. ■

We say that $\mathcal{A}$ wins the game if his guess for b is correct, i.e., $b' = b$. Let $win_{\mathcal{A}}$ be the probability that indicates the success of the adversary wins the above game. We define the indistinguishability under a multi-source ordered chosen plaintext attack (IND-MSOCPA) notion as follows.

Definition 2. IND-MSOCPA: indistinguishability under multi-source ordered chosen plaintext attack. A multi-source order-preserving encryption scheme is IND-MSOCPA secure if for all p.p.t. adversaries, $Pr[win_{\mathcal{A}}] \leq \frac{1}{2}$.

When an MSOPE scheme is IND-MSOCPA secure, then the leakage of the MSOPE scheme is a multi-source randomized order of the plaintexts. With such characteristics, an IND-MSOCPA secure scheme thwarts three threats, i.e., *privacy leakage*, *frequency analysis*, and *identical data inference* threats.

4. The MSOPE scheme

4.1. Overview of the MSOPE scheme

The MSOPE scheme is designed for privacy-preserving range query on encrypted outsourced EHRs. The work-flow of privacy-preserving range query involves two phases, i.e., EHRs outsourcing and EHRs querying, which are described as follows.

EHRs outsourcing. In this phase, $\mathcal{CS}$ invokes the **Secret State Generation** algorithm to generate an empty AVL tree, and fix the ciphertext domain. Then, each patient $\mathcal{P}_i \in \mathcal{P}$ invokes the **Encryption** algorithm to encrypt his/her EHRs, and later uploads the encrypted EHRs to $\mathcal{CS}$.

EHRs querying. In this phase, there are two different data users. On the one hand, $\mathcal{D}$ invokes the **Range Query** to encrypt the upper limit and lower limit of EHRs and then obtains the required encrypted EHRs. Later $\mathcal{D}$ invokes the **Decryption** algorithm to decrypt the required EHRs. On the other hand, P_i could query his/her EHRs and invokes the **Decryption** algorithm to decrypt the outsourced encrypted EHRs.

4.2. Construction of the MSOPE scheme

The MSOPE scheme is constructed based on the multi-source randomized order, which permutates the order of identical data from different patients randomly. We implement the multi-source randomized order by constructing the secret state T secretly, and later generate the order-preserving ciphertexts for EHRs collected from multiple patients based on T. The MSOPE scheme contains four polynomial-time algorithms, i.e., **Secret State Generation**, **Encryption**, **Decryption**, and **Range Query**.

Before we illustrate these four algorithms, we propose a secure comparing protocol, which compares the EHRs provided by different patients secretly. The secure comparing protocol is the key technique to generate the secret state T with EHRs provided by multiple patients. The output of the secure comparing protocol not only reflects the numerical order of the input EHRs but also randomizes the numerical order of two identical EHRs.

Secure Comparing Protocol. The secure comparing protocol is a secure two-party computation protocol involving two patients, i.e., $\mathcal{P}_i$ and $\mathcal{P}_j$. We utilize Paillier cryptosystem [24] (an additive homomorphic encryption) to construct it. The input of the secure comparing protocol is two EHR ciphertexts encrypted by SKE, i.e., $\widehat{w_{i,x}}$ and $\widehat{w_{j,y}}$, and the output of the secure comparing protocol is the comparison result R, which is one bit size denoting whether $w_{i,x}$ is less than $w_{j,y}$. The secure comparing protocol is described in Algorithm 1.

In our secure comparing protocol, $\mathcal{P}_i$ uses b_i to randomize the comparison result R. We show the relation between b_i and R in

Algorithm 1 Secure Comparing Protocol

```
Input: $\widehat{w_{i,x}}$ and $\widehat{w_{j,y}}$.
Output: $R$.
1: $\mathcal{P}_j$ runs HOM.Gen($1^\kappa$, $\mathcal{P}_j$), generates a public key $PK_j$ and a private key $SK_j$, and sends $PK_j$ to $\mathcal{P}_i$. Then $\mathcal{P}_i$ and $\mathcal{P}_j$ decrypts $\widehat{w_{i,x}}$ and $\widehat{w_{j,y}}$ to obtain $w_{i,x}$ and $w_{j,y}$, respectively.
2: $\mathcal{P}_j$ uses $PK_j$ to encrypt $(-w_{j,y})$ and sends $[\![-w_{j,y}]\!]$ to $\mathcal{P}_i$.
3: $\mathcal{P}_i$ computes $[\![w_{i,x}]\!]$ and flips a random coin $b_i \in \{0, 1\}$. Next, $\mathcal{P}_i$ randomly chooses two large random numbers $r_i$ and $r_i'$, with $r_i > r_i'$. Then $\mathcal{P}_i$ calculates:
       $V = ([\![w_{i,x}]\!] \cdot [\![-w_{j,y}]\!])^{(-1)^{b_i} \cdot r_i} \cdot [\![-r_i']\!] = [\![r_i \cdot (-1)^{b_i} \cdot (w_{i,x} - w_{j,y}) - r_i']\!]$.
   Finally, $\mathcal{P}_i$ sends $V$ to $\mathcal{P}_j$.
4: $\mathcal{P}_j$ decrypts $V$. If HOM.Dec($SK_j$, $V$) < 0, $\mathcal{P}_j$ sends $result = 0$ to $\mathcal{P}_i$; Otherwise, $\mathcal{P}_j$ sends $result = 1$ to $\mathcal{P}_i$.
5: $\mathcal{P}_i$ calculates $R = result \oplus b_i$.
```

Table 3. We use $R = 0$ to denote $w_{i,x} < w_{j,y}$ and $R = 1$ to denote $w_{i,x} > w_{j,y}$. Since $\mathcal{P}_i$ chooses b_i randomly, the compare result R of two identical EHRs is randomized. In our two-party comparing protocol, $\mathcal{P}_i$ uses r_i and r_i' to randomize $[\![(-1)^{b_i} \cdot (w_{i,x} - w_{j,y})]\!]$. Thus, $\mathcal{P}_j$ cannot recover $(-1)^{b_i} \cdot (w_{i,x} - w_{j,y})$ by decrypting V. Therefore, our secure comparing protocol not only reflects the numerical order of the input EHRs, but also randomizes the numerical order of two identical EHRs.

Now we illustrate the four polynomial-time algorithms as follows, i.e., **Secret State Generation**, **Encryption**, **Decryption**, and **Range Query**.

Secret State Generation. The secret state generation algorithm generates the secret state T and the ciphertext domain M. We present the secret state generation algorithm in Algorithm 2.

Algorithm 2 Secret State Generation

```
Input: N/A.
Output: $T$ and $M$.
1: $\mathcal{CS}$ generates an empty AVL tree $T$ as the secret state.
2: $\mathcal{CS}$ fixes $M$, which can be estimated by equation (7) and (8).
```

Encryption. The MSOPE encryption algorithm generates a MSOPE ciphertext $c_{i,x}$ for EHR $w_{i,x}$ that provided by $\mathcal{P}_i$. The MSOPE encryption algorithm involves three algorithms, i.e., *Tree Node Construction*, *Secret State Update*, and *Ciphertext Generation*. We firstly present the encryption algorithm in Algorithm 3, and later describe these three algorithms.

Algorithm 3 Encryption

```
Input: $w_{i,x}$.
Output: $c_{i,x}$.
1: $\mathcal{P}_i$ invokes the Tree Node Construction algorithm to encrypt $w_{i,x}$ and generate the message $t_{i,x}$ as and AVL tree node.
2: $\mathcal{P}_i$ invokes the Secret State Update algorithm to insert the tree node $t_{i,x}$ to the AVL tree $T$.
3: $\mathcal{CS}$ invokes the Ciphertext Generation algorithm to generate order-preserving ciphertexts according to $T$.
```

Tree Node Construction. In the tree node construction algorithm, the patient uses his SKE key to encrypt $w_{i,x}$, and later generates the tree node message $t_{i,x}$. We present the tree node construction algorithm in Algorithm 4.

Table 3
A description of our secure comparing protocol.

Case	b_i	*result*	$R = result \oplus b_i$
$w_{i,x} < w_{j,y}$	0	0	**0**
$w_{i,x} < w_{j,y}$	1	1	**0**
$w_{i,x} = w_{j,y}$	0	0	**0**
$w_{i,x} = w_{j,y}$	1	0	**1**
$w_{i,x} > w_{j,y}$	0	1	**1**
$w_{i,x} > w_{j,y}$	1	0	**1**

Algorithm 4 Tree Node Construction

```
Input: $w_{i,x}$.
Output: $t_{i,x}$.
1: $\mathcal{P}_i$ uses his SKE key $sk_i$ to encrypt the EHR $w_{i,x}$, and obtains the corresponding ciphertext $\widehat{w_{i,x}}$.
2: $\mathcal{P}_i$ generates the message $t_{i,x} = (\mathcal{P}_i, \widehat{w_{i,x}})$ as an AVL tree node.
```

Secret State Update. In the secret state update algorithm, $\mathcal{P}_i$ updates the secret state by inserting the message $(\mathcal{P}_i, \widehat{w_{i,x}})$ to the AVL tree T, if $(\mathcal{P}_i, \widehat{w_{i,x}})$ has not been in T. The secret state update algorithm can be described in Algorithm 5.

Algorithm 5 Secret State Update

```
Input: $t_{i,x} = (\mathcal{P}_i, \widehat{w_{i,x}})$
Output: $T$
1: If the node $(\mathcal{P}_i, \widehat{w_{i,x}})$ has already in $T$, the secret state will not be updated. The secret state update algorithm outputs the prior secret state $T$ and ends this algorithm. Otherwise, goes to the next step.
2: $\mathcal{P}_i$ asks $\mathcal{CS}$ for the root node of $T$.
3: $\mathcal{CS}$ returns a node $t$ to $\mathcal{P}_i$.
4: If the node $t$ is provided by $\mathcal{P}_i$, i.e., $t = (\mathcal{P}_i, \widehat{w_{i,r}})$, then $\mathcal{P}_i$ decrypts $\widehat{w_{i,r}}$ and compares $w_{i,x}$ with $w_{i,r}$. If the node $t$ was not provided by $\mathcal{P}_i$, i.e., $t = (\mathcal{P}_j, \widehat{w_{j,r}})$, then $\mathcal{P}_i$ invokes the secure comparing protocol to compares $w_{i,x}$ with $w_{j,r}$ secretly.
5: If the compared result shows that $w_{i,x} < w_{i,r}$ or $w_{i,x} < w_{j,r}$, then $\mathcal{P}_i$ asks $\mathcal{CS}$ for the left child node of $t$ in $T$. Otherwise, $\mathcal{P}_i$ asks $\mathcal{CS}$ for the right child node of $t$ in $T$.
6: If $\mathcal{CS}$ does not arrive at an empty spot of $T$, $\mathcal{CS}$ returns the next qualified child node to $\mathcal{P}_i$ and goes back to step 3; otherwise, it goes to the next step.
7: $\mathcal{CS}$ inserts a new node $(\mathcal{P}_i, \widehat{w_{i,x}})$ to the AVL tree $T$ and balances the AVL tree $T$. Then secret state $T$ is updated after the new node insertion and the balance operation.
```

Ciphertext Generation. In the ciphertext generation algorithm, the cloud server generates order-preserving ciphertexts for EHRs stored in T. The ciphertexts are generated from T and the ciphertext domain M. We use recursion to generate the order-preserving ciphertexts for EHRs in T. We use *Min* and *Max* to denote the lower bound and the upper bound of recursion respectively. The ciphertext generation algorithm returns the corresponding order-preserving ciphertext of $w_{i,x}$ for P_i. The ciphertext generation algorithm can be described in Algorithm 6.

Decryption. The MSOPE decryption algorithm is used for $\mathcal{P}_i$ to decrypt his encrypted EHR $c_{i,x}$. The decryption algorithm can be described in Algorithm 7.

Range Query. The range query algorithm is a variant of the **Encryption** algorithm, which is used for $\mathcal{D}$ to perform a range query over encrypted multi-source EHRs. When $\mathcal{D}$ wants to search the desired EHRs that lie between the upper bound $w_{d,ub}$ and

Algorithm 6 Ciphertext Generation

```
Input: $Min$, $Max$, and $T$.
Output: $c_{i,x}$.
1: $\mathcal{CS}$ initializes $Min$ to be $-1$, and $Max$ to be $M$. Then $\mathcal{CS}$ starts ciphertext generation phase from the root node in $T$.
2: For a node $t = (\mathcal{P}_*, \widehat{w_{*,r}})$ in $T$, $\mathcal{CS}$ calculates
       $$c_{*,r} = \left\lceil \frac{Max + Min}{2} \right\rceil.$$
   Then $\mathcal{CS}$ stores $(\mathcal{P}_*, \widehat{w_{*,r}}, c_{*,r})$.
3: If the left subtree of $t$ is not empty, $\mathcal{CS}$ resets $Max$ to $c_{*,r}$, and generates the order-preserving ciphertext for left child of $t$ by using the same procedure in Step 2.
4: If the right subtree of $t$ is not empty, $\mathcal{CS}$ resets $Min$ to $c_{*,r}$, and generates the order-preserving ciphertext for left child of $t$ by using the same procedure in Step 2.
5: Finally, $\mathcal{CS}$ generates order-preserving ciphertexts for EHRs in $T$. Then $\mathcal{CS}$ returns $c_{i,x}$ to $\mathcal{P}_i$.
```

Algorithm 7 Decryption

```
Input: $c_{i,x}$.
Output: $w_{i,x}$.
1: $\mathcal{P}_i$ sends $c_{i,x}$ to $\mathcal{CS}$.
2: $\mathcal{CS}$ searches $c_{i,x}$ in $T$, and returns the corresponding SKE ciphertext $\widehat{w_{i,x}}$ to $\mathcal{P}_i$.
3: $\mathcal{P}_i$ decrypts $\widehat{w_{i,x}}$ by using $sk_i$, and obtains the corresponding EHR $w_{i,x}$.
```

the lower bound $w_{d,lb}$, he firstly encrypts $w_{d,ub}$ and $w_{d,lb}$, and generates $c_{d,ub}$ and $c_{d,lb}$ by using the **Encryption** algorithm. Note that when applying the **secure comparing protocol**, the random coin b_d chosen by $\mathcal{D}$ should be fixed, i.e., when encrypting the upper bound $w_{d,ub}$, b_d should always be 1, and when encrypting the lower bound $w_{d,lb}$, b_d should always be 0. By doing so, for each EHR $w_{i,x}$ from $\mathcal{P}_i$, if $w_{i,x} = w_{d,ub}$, then $c_{i,x} < c_{d,ub}$, else if $w_{i,x} = w_{d,lb}$, then $c_{i,x} > c_{d,lb}$. Then, $\mathcal{CS}$ returns a set of encrypted nodes $\{t\}$ in T whose OPE ciphertext c satisfies $c_{d,lb} \le c \le c_{d,ub}$. Note that each node $t_{i,x} = (\mathcal{P}_i, \widehat{w_{i,x}})$. Finally, $\mathcal{D}$ uses the SKE key to decrypt the corresponding SKE encrypted data. If $t_{i,x}$ satisfies $c_{d,lb} \le c_{i,x} \le c_{d,ub}$, then $\mathcal{D}$ uses sk_i to decrypt $\widehat{w_{i,x}}$ and obtains the corresponding EHR $w_{i,x}$. We present the **Range Query** algorithm in Algorithm 8.

Algorithm 8 Range Query

```
Input: $T$, $w_{d,lb}$ and $w_{d,ub}$.
Output: any EHR $w_{i,x}$ in $T$ with $w_{d,lb} \le w_{i,x} \le w_{d,ub}$.
1: $\mathcal{D}$ uses encrypts $w_{d,ub}$ and $w_{d,lb}$, and generates $c_{d,ub}$ and $c_{d,lb}$.
2: $\mathcal{CS}$ returns a set of encrypted nodes $\{t\}$ in $T$ whose OPE ciphertext $c$ satisfies $c_{d,lb} \le c \le c_{d,ub}$.
3: $\mathcal{D}$ uses the SKE key to decrypt the corresponding SKE encrypted data. Namely, if $t_{i,x}$ satisfies $c_{d,lb} \le c_{i,x} \le c_{d,ub}$, then $\mathcal{D}$ uses $sk_i$ to decrypt $\widehat{w_{i,x}}$ and obtains the corresponding EHR $w_{i,x}$.
```

4.3. Examples of the MSOPE scheme

We provide two examples to describe our scheme, i.e., one for the **Encryption** algorithm, and the other for the **Range Query** algorithm.

Fig. 2 is an example for our **Encryption** algorithm, including three patients $\mathcal{P}_1$, $\mathcal{P}_2$, $\mathcal{P}_3$, and a cloud server $\mathcal{CS}$, which is described as follows.

(1) $\mathcal{P}_1$, $\mathcal{P}_2$, and $\mathcal{P}_3$ want to outsource their EHRs $\{15, 19, 81\}$, $\{3, 1, 14\}$, and $\{91, 15, 15\}$ to $\mathcal{CS}$, respectively.
(2) $\mathcal{P}_1$, $\mathcal{P}_2$, and $\mathcal{P}_3$ use SKE encryption to encrypt their EHRs and construct AVL tree nodes.
(3) $\mathcal{P}_1$, $\mathcal{P}_2$, and $\mathcal{P}_3$ help $\mathcal{CS}$ to construct the secret state (the AVL tree) by inserting the tree node to the AVL tree (invoking the secret state update algorithm). Note that $\mathcal{P}_3$ only inserts $\{91, 15\}$ to the secret state because repeated EHR 15 only inserts once. In the secret state, we can find that identical EHR 15 provided by $\mathcal{P}_1$ and $\mathcal{P}_3$ have different positions in the AVL tree because our comparing protocol randomizes the compared result of 15 provided by different patients.
(4) $\mathcal{CS}$ invokes the ciphertext generation algorithm to generate the MSOPE ciphertexts. Finally, we can find that the corresponding ciphertexts of EHRs $\{15, 19, 81, 3, 1, 14, 91, 15, 15\}$ are $\{6, 10, 12, 2, 1, 4, 14, 8, 8\}$.

Fig. 3 is an example for the **Range Query** algorithm, which involves $\mathcal{D}$ and $\mathcal{CS}$, and is described as follows.

(1) $\mathcal{D}$ sets the lower bound and upper bound of range query domain to be 15 and 18, respectively, i.e., $w_{d,lb} = 15$, and $w_{d,ub} = 18$.
(2) $\mathcal{D}$ uses his SKE key to encrypt $w_{d,lb}$ and $w_{d,ub}$, and later constructs AVL tree nodes for these two encrypted EHRs.
(3) $\mathcal{D}$ inserts the upper bound and the lower bound of his query to the secret states. Note that when inserting $\widehat{w_{d,lb}}$ to the secret state, the random coin b_d chosen by $\mathcal{D}$ in **secure comparing protocol** is 0, i.e., $b_d = 0$. When inserting $\widehat{w_{d,ub}}$, then $b_d = 1$. Afterwards, $w_{d,lb}$ and $w_{d,ub}$ will be inserted to the secret state at the positions shown in Fig. 3. Then $\mathcal{CS}$ updates the MSOPE ciphertexts for nodes in secret state.
(4) $\mathcal{CS}$ returns '$0x97bcd7$' and '$0xe652af$' to $\mathcal{D}$ because the corresponding MSOPE ciphertext of these two encrypted EHRs are less than $c_{d,ub}$ and larger than $c_{d,lb}$.
(5) $\mathcal{D}$ decrypts '$0x97bcd7$' and '$0xe652af$' by using the SKE key provided by the owner of these two EHRs.
(6) $\mathcal{D}$ obtains the corresponding EHRs, i.e., '15' and '15'.

4.4. Security analysis

The security design goal of the MSOPE scheme is to protect the content of EHRs against *privacy leakage*, *frequency analysis*, and *identical data inference* threats. As we have defined in Section 3.3, the IND-MSOCPA security definition guarantees that schemes only leak the multi-source randomized order of EHRs, and thus protect EHRs against the aforementioned threats. We assume that SKE encryptions are computationally indistinguishable from random values, HOM encryption is IND-CPA secure [24]. We state the security properties of the MSOPE scheme in Theorem 1.

Theorem 1. *The multi-provider order-preserving encryption scheme is secure against multi-source ordered chosen plaintext attack. Namely, our scheme is IND-MSOCPA secure.*

We provide formal security proof for Theorem 1 in Appendix B.

5. Performance analysis and evaluations

5.1. Performance analysis

We analyze four polynomial-time algorithms in MSOPE in terms of computational complexity, i.e., **Secret State Generation**, **Encryption**, **Decryption**, and **Range Query**.

The **Secret State Generation** algorithm creates an empty AVL tree as secret states, and thus the computational complexity is $\mathbf{O}(1)$.

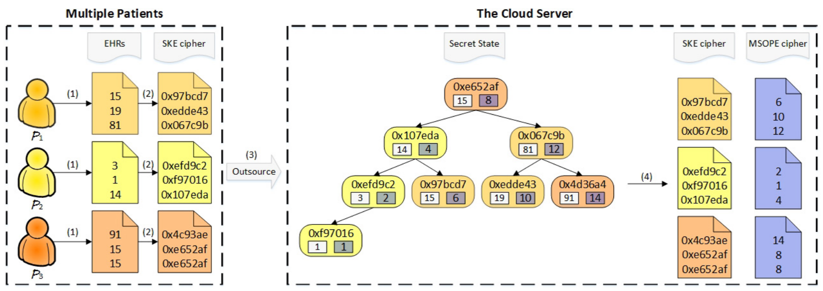


Fig. 2. An example of the **Encryption** algorithm.

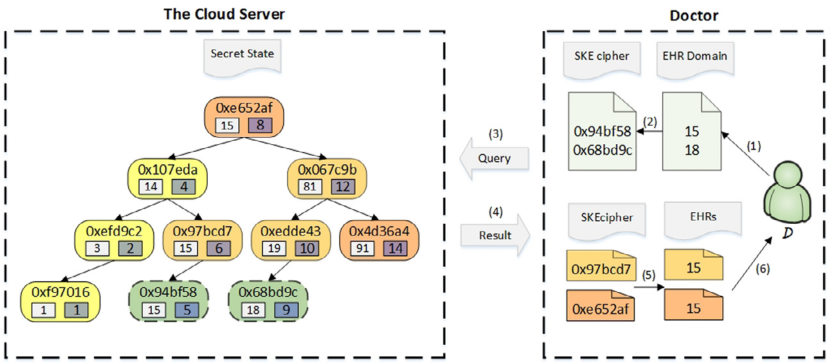


Fig. 3. An example of the **Range Query** algorithm.

The **Encryption** algorithm involves three algorithms. The *Tree Node Construction* algorithm encrypts an EHR and generates a secret state $t_{i,x}$, whose computational complexity is $\mathbf{O}(1)$. The *Secret State Update* algorithm invokes the secure comparing protocol to compare the node $t_{i,x}$ with $\mathbf{O}(\log N)$ nodes in the AVL tree T, which requires complex computation such as modular exponentiation computation. Thus, the computational complexity of the *Secret State Update* algorithm is $\mathbf{O}(\log N)$. The *Ciphertext Generation* algorithm is a pre-order traversal of the AVL tree, whose computational complexity is $\mathbf{O}(N \log N)$. Since the *Secret State Update* algorithm requires complex computation, which requires more time than the pre-order traversal in *Ciphertext Generation* phase, the computational complexity of **Encryption** algorithm is $\mathbf{O}(\log N)$.

The **Decryption** algorithm finds the corresponding SKE ciphertext of an MSOPE ciphertext in the AVL tree and decrypts the SKE ciphertext. Since the depth of the AVL tree is $\mathbf{O}(\log N)$, searching a qualified node in the AVL tree requires $\mathbf{O}(\log N)$ comparison operations. Since ciphertexts encrypted by order-preserving encryption preserve the order information of the plaintexts, the comparison operations could be performed in the ciphertexts instead of plaintexts. Therefore, the computational complexity of **Decryption** is $\mathbf{O}(\log N)$.

The **Range Query** algorithm encrypts the upper bound and the lower bound of EHRs, searches for the desired encrypted EHRs, and decrypts encrypted EHRs. The encryption phase requires $\mathbf{O}(\log N)$ complex computations. The search phase requires $\mathbf{O}(N)$ operations. The decryption phase requires $\mathbf{O}(N)$ operations. Since the encryption phase requires complex computations, which requires more time than other phases. Therefore, the computational complexity of **Range Query** is $\mathbf{O}(\log N)$. We summarized the computational complexity results in Table 4.

5.2. Performance evaluations

The experiments are conducted on a 64-Bit workstation with an Intel Xeon E-1226 CPU with 3.30 GHz and 32 GB RAM. We use the Paillier cryptosystem with 1024 bit key length to implement our comparing protocol. We implement our scheme (MSOPE), the conference version in [17] (MPOPE), the scheme in [25] (MOPE), and the scheme in [13] (FHOPE) in Java 1.8. In our experiments, we assume that each patient encrypts the same number of EHRs. Namely, when the number of EHRs is n and the number of patients is k, then each patient encrypts n/k EHR items. Our evaluations target on the performance of the **Encryption** algorithm and the **Range Query** algorithm, because the **Secret State Generation** algorithm and the **Decryption** algorithm are very lightweight.

Table 4
Computational complexity.

Algorithm	Computational Complexity
Secret State Generation	$\mathbf{O}(1)$
Encryption	$\mathbf{O}(\log N)$
Decryption	$\mathbf{O}(\log N)$
Range Query	$\mathbf{O}(\log N)$

5.2.1. Time cost of **Encryption** algorithm in MSOPE

To evaluate the performance of **Encryption** algorithm in MSOPE, we first evaluate the time delay of **Encryption** algorithm with respect to the number of patients k and the number of EHRs n. Fig. 4 presents the average time cost of **Encryption** algorithm with respect to k. In this experiment, we evaluate the average running time when $k = 2, 4, 8, 16, 32$ and $n = 4000, 16000, 64000$. The experiment results show that when k increases exponentially, the average time cost grows logarithmically. It demonstrates that when k grows, the total number of EHRs in the AVL tree grows, and therefore the depth of AVL tree grows (according to Eqs. (4) and (7)), which requires more time to update the secret state.

Fig. 5 shows the time cost of **Encryption** algorithm with respect to n. In this experiment, we evaluate the average running time when $k = 2, 8, 32$ and $n = 4000, 8000, 16000, 32000$,

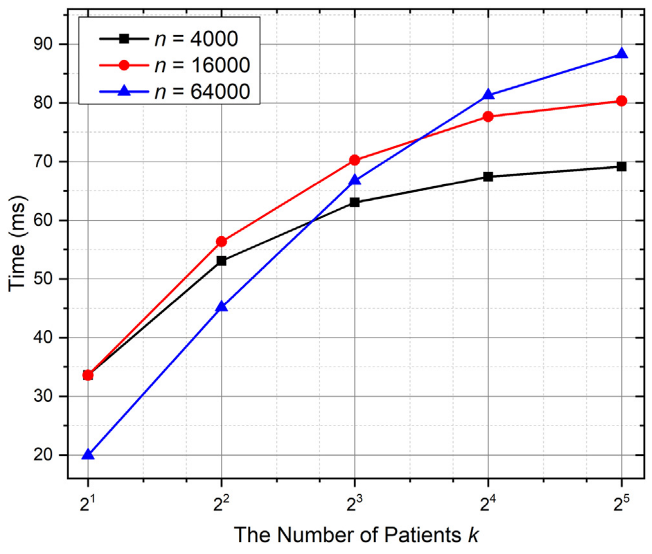


Fig. 4. The time cost of **Encryption** algorithm in respect of *k*.

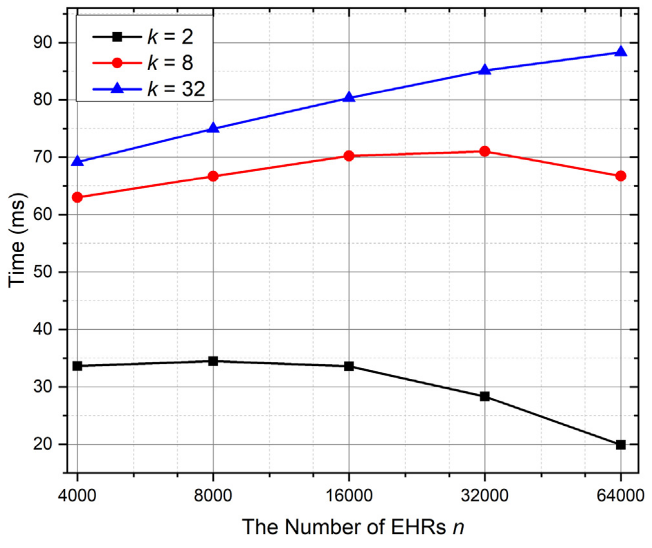


Fig. 5. The time cost of **Encryption** algorithm in respect of *n*.

64000. We can observe that the average running time firstly increases and then decreases when n grows. Due to the increased depth of the secret states, the secret state update time increases and therefore the average running time increases firstly. According to Eq. (4), the expected number of N is $k * D$. When n is larger than $k * D$, the **Encryption** algorithm does not need to update the secret state because these identical EHRs from the same patients do not need to be inserted to the secret state, leading to the decreased average running time.

Since MOPE [25], FHOPE [13], and MPOPE [17] schemes achieve similar security properties as compared to the MSOPE scheme, we compare the time cost of **Encryption** of the MSOPE scheme with existing works in Table 5. Table 5 presents a comparison of the MSOPE scheme with MOPE [25], FHOPE [13], and MPOPE [17] in single patient scenario, i.e., $k = 1$, because existing order-preserving encryption schemes in [25] and [13] mainly consider a scenario with only one data source. In the single data source scenario, both the MPOPE scheme and the MSOPE scheme are not required to invoke the secure comparing protocol, and therefore achieve better efficiency than that of the multiple data source scenario.

Table 5
A comparison of encryption time of MSOPE with several existing methods.

	$N = 4000$	$N = 8000$	$N = 16000$	$N = 32000$	$N = 64000$
MOPE [25]	~ 1067 μs	~ 2065 μs	~ 3545 μs	~ 5106 μs	~ 5996 μs
FHOPE [13]	~ 1153 μs	~ 2737 μs	~ 5838 μs	~ 12543 μs	~ 26825 μs
MPOPE [17]	~ 1133 μs	~ 2093 μs	~ 3679 μs	~ 5126 μs	~ 6118 μs
MSOPE	~ 834 μs	~ 1687 μs	~ 2968 μs	~ 4168 μs	~ 4811 μs

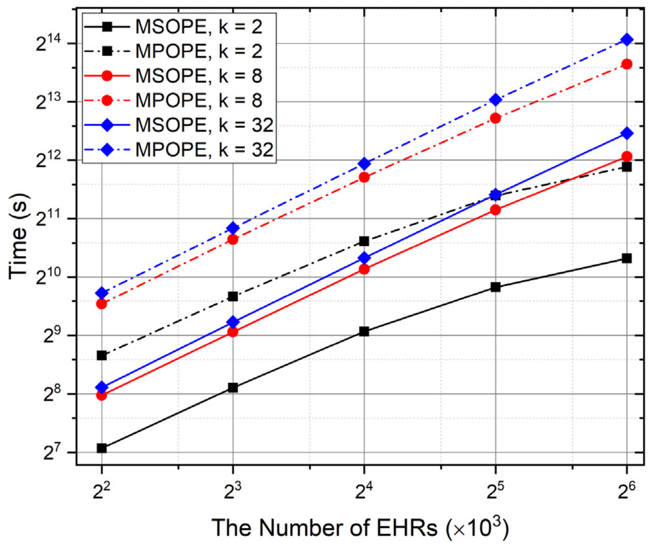


Fig. 6. A comparison of encryption time of MSOPE with MPOPE when $k = 2, 8, 32$. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

To further explore the practicality of the MSOPE scheme, we provide comparisons of MSOPE with MPOPE, which achieves the same security properties with the MSOPE scheme and is appropriate for multiple data source scenario. Fig. 6 presents total time cost of the **Encryption** algorithm of MSOPE with MPOPE when the number of patients $k = 2, 8, 32$ and the total number of EHRs $n = 4000, 8000, 16000, 32000, 64000$. The black solid line with square markers shows that when $k = 2$, the MSOPE scheme requires 134.5 s and 1275.2 s to encrypt 4000 and 64 000 EHRs, respectively. The black dash–dot line with square markers presents that when $k = 2$, the MPOPE scheme requires 403.4 s and 3786.9 s to encrypt 4000 and 64 000 EHRs, respectively. Meanwhile, the red solid line with roundness markers shows that when $k = 8$, the MSOPE scheme requires 252.1 s and 4271.0 s to encrypt 4000 and 64 000 EHRs respectively, while the red dash–dot line with roundness markers shows that the MPOPE scheme requires 744.7 s and 12787.6 s, respectively. Last but not the least, the blue solid line with rhombic markers shows that when $k = 32$, the MSOPE scheme requires 276.7 s and 5650.4 s to encrypt 4000 and 64 000 EHRs respectively, while the blue dash–dot line with rhombic markers demonstrates that the MPOPE scheme requires 848.2 s and 17167.0 s, respectively.

Fig. 6 shows that when n increases exponentially, the running time of **Encryption** algorithm in MSOPE and MPOPE increases linearly. Both the MSOPE scheme and the MPOPE scheme achieve the same security property but uses different secure comparing protocols. The comparison results of running time of **Encryption** algorithm show that our MSOPE scheme is about five times faster than the MPOPE scheme, which demonstrates that our secure comparing protocol is more efficient than the secure comparing protocol in [17], and therefore our MSOPE is more efficient than MPOPE in terms of computational overhead of the **Encryption** algorithm.

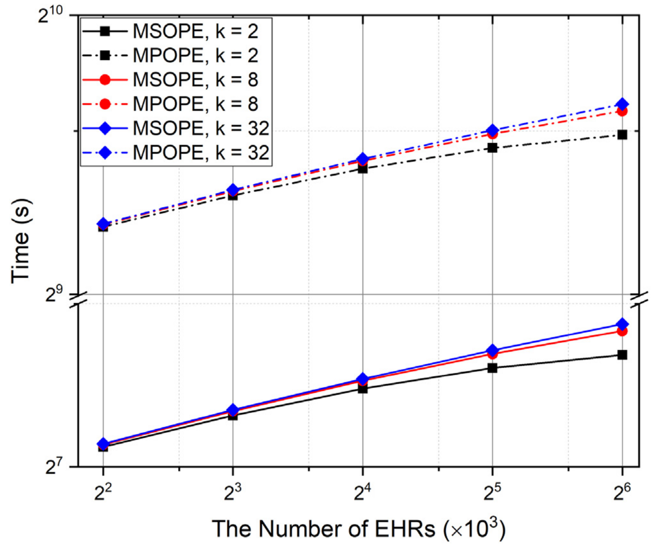


Fig. 7. A comparison of range query time of MSOPE with MPOPE, when $k = 2, 8, 32$. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

5.2.2. Time cost of ***Range Query*** *algorithm in MSOPE*

We also evaluate the time cost of **Range Query** algorithm, which returns qualified EHRs to $\mathcal{D}$, and compare the total time cost of MSOPE with MPOPE. Note that the **Range Query** algorithm is a variant of **Encryption** algorithm, because $\mathcal{D}$ is required to encrypt the upper bound and the lower bound of query range and obtain the corresponding ciphertexts that less than the encrypted upper bound but larger than the lower bound. Fig. 7 presents the comparison results when the number of patients $k = 2, 8, 32$ and the total number of EHR item in the public cloud $n = 4000, 8000, 16000, 32000, 64000$. The black solid line with square markers shows that when $k = 2$, the MSOPE scheme requires 134.5 ms and 169.1 ms to query qualified EHRs when 4000 and 64 000 encrypted EHRs stored in the public cloud respectively, while the black dash–dot line with square markers demonstrates that the MPOPE scheme requires 605.4 ms and 760.8 ms for performing range query, respectively. The red solid line with roundness markers shows that when $k = 8$, the MSOPE scheme requires 135.3 ms and 179.4 ms to query qualified EHRs when 4000 and 64 000 encrypted EHRs stored in the public cloud, respectively, while the red dash–dot line with roundness markers demonstrates that the MPOPE scheme requires 609.0 ms and 807.5 ms for performing range query, respectively. The blue solid line with rhombic markers shows that when $k = 32$, the MSOPE scheme requires 135.5 ms and 182.5 ms to query qualified EHRs when 4000 and 64 000 encrypted EHRs stored in the public cloud, respectively, while the blue dash–dot line with rhombic markers demonstrates that the MPOPE scheme requires 609.9 ms and 821.3 ms for performing range query, respectively.

Fig. 7 shows that when n increases exponentially, the running time of **Range Query** algorithm in MSOPE and MPOPE increases linearly. Both the MSOPE scheme and the MPOPE scheme achieve privacy-preserving range query by encrypting the upper limit and lower limit of query and comparing the encrypted EHRs with the encrypted upper limit and encrypted lower limit. Therefore, the running time of **Range Query** algorithm of both MSOPE and MPOPE are dependent on the running time of **Encryption** algorithm. The comparison results of running time of **Range Query** algorithm show that our MSOPE scheme is about five times faster than MPOPE, which demonstrates that our secure comparing protocol is more efficient than the secure comparing protocol in [17], and therefore our MSOPE is more efficient than MPOPE in terms of time overhead of the **Range Query** algorithm.

6. Related works

With the development of cloud computing, sensitive personal data has been outsourced to the public cloud [33], such as electronic health records [11,18], social network data [20,23], and location data [6,10,22,34]. However, the risk of outsourced data breach becomes a roadblock of using the cloud-based applications [19], especially in cloud-based eHealth systems [7,30,31]. The main reason is that once the sensitive EHRs is outsourced to a public cloud, the content of EHRs will be exposed to the cloud server directly, which may leak the patients' data privacy [37,39]. Therefore, privacy issues have become one of the most important issues in cloud-based eHealth systems [16,40]. To extract the value behind EHRs while protecting the privacy, privacy-preserving range query has been regarded as an important issue in cloud-based eHealth systems [9,21].

Order-preserving encryption is an efficient and privacy-preserving method for range query, which jointly considers efficiency and security [28]. Agrawal et al. proposed the first order-preserving encryption scheme in [2]. Boldyreva et al. [3] proposed the ideal security notion for order-preserving encryption: *indistinguishability under ordered chosen plaintext attack* (IND-OCPA), which leaks no information about data contents except the order of data (the minimum requirement for the order-preserving property). Order-preserving encryption schemes with IND-OCPA security only leaks the order information of plaintexts, and thus is secure against *privacy leakage*. Popa et al. proposed the first IND-OCPA secure order-preserving encryption scheme by constructing a stateful order-preserving encoding method via binary search tree [25]. Considering the high communication overhead of the scheme in [25], Kerschbaum and Schropfer proposed an efficient and IND-OCPA secure order-preserving encryption scheme [14]. Since the schemes in [25] and [14] leak the frequency information of plaintexts, Kerschbaum proposed a stronger security notion for order-preserving encryption: *indistinguishability under frequency analyzing ordered chosen plaintext attack* (IND-FAOCPA), which implies IND-OCPA and is secure against frequency analysis threats [13]. Then, Kerschbaum proposed an efficient and IND-FAOCPA secure scheme, which is secure against *privacy leakage* and *frequency analysis* threats. However, the scheme in [13] incurs a high storage space requirement. Inspired by the obfuscation technique, Boneh et al. proposed an order-revealing encryption scheme via multi-linear map [5]. Roche et al. proposed a partial order-preserving encryption scheme which achieves IND-FAOCPA security notion while providing extremely fast insertion and efficient search [29]. Lewi and Wu provided a novel order-revealing encryption against inference attack [15]. Aforementioned order-preserving encryption schemes consider a system model with a data provider and a cloud server, which are more of theoretic attempts to push the security notions to the limit. Yet, these schemes are not applicable in cloud-based eHealth systems, because they cannot support range queries on EHRs from multiple patients.

In this paper, we mainly focus on a multi-source scenario, i.e., cloud-based eHealth systems. In this scenario, an authorized doctor can perform privacy-preserving range queries on EHRs collected from multiple patients. To achieve the functionality goal and security goal, some multi-source order-preserving encryption schemes have been proposed. Xiao et al. proposed the first multi-source order-preserving encryption [32]. Their scheme achieves the multi-source property by using a group of key agents to enable distributed encryption. Yet, their scheme cannot resist threats such as *privacy leakage* and *frequency analysis*. Yao et al.

proposed a multi-source order-preserving encryption scheme for cloud-based eHealth applications by using monotone minimal perfect hash function [36]. The scheme in [36] is a one-wayness order-preserving encryption scheme whose security is guaranteed by the random order-preserving function (ROPF). However, Boldyreva et al. proved that one-wayness order-preserving encryption schemes leak the value of any plaintext as well as the distance between any two plaintexts [4], and therefore their scheme cannot resist threats such as *privacy leakage* and *frequency analysis*. Our primary work in [17] improved the security property of order-preserving encryption, which is secure against threats such as *privacy leakage*, *frequency analysis*, and *identical data inference*. Different from [17], we develop an efficient and secure comparing protocol, which significantly improves the efficiency of encryption and range query in terms of computational overhead. The experimental evaluations show that our proposed scheme is about five times faster than the primary version of this work [17]. Meanwhile, compared with order-preserving encryption schemes for cloud-based eHealth systems [32,35,36], our proposed scheme significantly improves the security property, and resists threats such as *privacy leakage*, *frequency analysis*, and *identical data inference*.

7. Conclusions

In this paper, we have identified three threats, i.e., *privacy leakage*, *frequency analysis*, and *identical data inference* in cloud-based eHealth systems. Besides, we have defined the security notion of IND-MSOCPA for multi-source order-preserving encryption to capture the security properties that resist the threats. Furthermore, we have proposed the MSOPE scheme to enable doctors to perform privacy-preserving range queries over outsourced EHRs from multiple patients in cloud-based eHealth systems. We have provided a formal security proof to show that the MSOPE scheme is IND-MSOCPA secure. Extensive performance experiments demonstrate that the MSOPE scheme is more efficient than the primary version of this work.

Regarding the future works, we will investigate how to construct privacy-preserving range query schemes on other cryptographic primitives for cloud-based eHealth systems. Since existing order-preserving encryption techniques cannot protect the order information of EHR contents against cloud servers, which may enable adversaries to extract information about EHRs from the leaked order. Meanwhile, we will try to investigate secure comparing protocols, which could further reduce the time delay of order-preserving encryption based privacy-preserving range query schemes. Last but not least, we will try to further design privacy-preserving range query schemes for multi-attribute EHRs.

Declaration of competing interest

No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work. For full disclosure statements refer to https://doi.org/10.1016/j.jpdc.2019.08.011.

Acknowledgments

A primary version of this paper had been published in the proceedings of the 13th EAI International Conference on Security and Privacy in Communication Networks (SecureComm 2017) [17]. This work is supported by the National Natural Science Foundation of China (61772191, 61300217, 61872130, 61972058), Science and Technology Key Projects of Hunan Province (2015TP1004, 2016JC2012, 2018TP1009), Science and Technology Key Projects of Changsha City (kq1801008, kq1804008), Hunan Province Cooperative Innovation Center of Dongting Lake Ecological Economic Zone, Hunan University Young Scholar Funds (01OS), and China Scholarship Council (201806130132).

Appendix A. Correctness of the secure comparing protocol

The goal of Algorithm 1 is to compare $w_{i,x}$ with $w_{j,y}$ secretly, and outputs one bit size compare result R. We find 6 cases for the compare result R. When $w_{i,x} < w_{j,y}$, then $R = 0$. When $w_{i,x} < w_{j,y}$, then $R = 1$. When $w_{i,x} = w_{j,y}$, then the value of R depends on a random bit b_i. Namely, when $b_i = 0$ and $w_{i,x} = w_{j,y}$, then $R = 0$; when $b_i = 1$ and $w_{i,x} = w_{j,y}$, then $R = 1$. We analyze the correctness of the secure comparing protocol in Algorithm 1 as follows.

1. When $w_{i,x} < w_{j,y}$ and $b_i = 0$, then $\mathcal{P}_i$ calculates

   $V = [\![r_i \cdot (-1)^{b_i} \cdot (w_{i,x} - w_{j,y}) - r'_i]\!] = [\![r_i \cdot (w_{i,x} - w_{j,y}) - r'_i]\!]$,

   and sends V to $\mathcal{P}_j$. After decrypting V, since $w_{i,x} < w_{j,y}$ and $r_i > r'_i > 0$,

   $\text{HOM.Dec}(SK_j, V) < 0$.

   Thus, $\mathcal{P}_j$ sends $result = 0$ to $\mathcal{P}_i$. Since $b_i = 0$ and $result = 0$,

   $R = result \oplus b_i = 0 \oplus 0 = 0$.

2. When $w_{i,x} < w_{j,y}$ and $b_i = 1$, then $\mathcal{P}_i$ calculates

   $V = [\![r_i \cdot (-1)^{b_i} \cdot (w_{i,x} - w_{j,y}) - r'_i]\!] = [\![r_i \cdot (w_{j,y} - w_{i,x}) - r'_i]\!]$,

   and sends V to $\mathcal{P}_j$. After decrypting V, since $w_{i,x} < w_{j,y}$ and $r_i > r'_i > 0$,

   $\text{HOM.Dec}(SK_j, V) > 0$.

   Thus, $\mathcal{P}_j$ sends $result = 1$ to $\mathcal{P}_i$. Since $b_i = 1$ and $result = 1$,

   $R = result \oplus b_i = 1 \oplus 1 = 0$.

3. When $w_{i,x} = w_{j,y}$ and $b_i = 0$, then $\mathcal{P}_i$ calculates

   $V = [\![r_i \cdot (-1)^{b_i} \cdot (w_{i,x} - w_{j,y}) - r'_i]\!] = [\![-r'_i]\!]$,

   and sends V to $\mathcal{P}_j$. After decrypting V, since $r'_i > 0$,

   $\text{HOM.Dec}(SK_j, V) < 0$.

   Thus, $\mathcal{P}_j$ sends $result = 0$ to $\mathcal{P}_i$. Since $b_i = 0$ and $result = 0$,

   $R = result \oplus b_i = 0 \oplus 0 = 0$.

4. When $w_{i,x} = w_{j,y}$ and $b_i = 1$, then $\mathcal{P}_i$ calculates

   $V = [\![r_i \cdot (-1)^{b_i} \cdot (w_{i,x} - w_{j,y}) - r'_i]\!] = [\![-r'_i]\!]$,

   and sends V to $\mathcal{P}_j$. After decrypting V, since $r'_i > 0$,

   $\text{HOM.Dec}(SK_j, V) < 0$.

   Thus, $\mathcal{P}_j$ sends $result = 0$ to $\mathcal{P}_i$. Since $b_i = 1$ and $result = 0$,

   $R = result \oplus b_i = 0 \oplus 1 = 1$.

5. When $w_{i,x} > w_{j,y}$ and $b_i = 0$, then $\mathcal{P}_i$ calculates

   $V = [\![r_i \cdot (-1)^{b_i} \cdot (w_{i,x} - w_{j,y}) - r'_i]\!] = [\![r_i \cdot (w_{i,x} - w_{j,y}) - r'_i]\!]$,

   and sends V to $\mathcal{P}_j$. After decrypting V, since $w_{i,x} > w_{j,y}$ and $r_i > r'_i > 0$,

   $\text{HOM.Dec}(SK_j, V) > 0$.

   Thus, $\mathcal{P}_j$ sends $result = 1$ to $\mathcal{P}_i$. Since $b_i = 0$ and $result = 1$,

   $R = result \oplus b_i = 1 \oplus 0 = 1$.

6. When $w_{i,x} > w_{j,y}$ and $b_i = 1$, then $\mathcal{P}_i$ calculates

   $V = [\![r_i \cdot (-1)^{b_i} \cdot (w_{i,x} - w_{j,y}) - r'_i]\!] = [\![r_i \cdot (w_{j,y} - w_{i,x}) - r'_i]\!]$, and sends V to $\mathcal{P}_j$. After decrypting V, since $w_{i,x} < w_{j,y}$ and $r_i > r_i' > 0$,

HOM.Dec$(SK_j, V) < 0$.

Thus, $\mathcal{P}_j$ sends $result = 0$ to $\mathcal{P}_i$. Since $b_i = 1$ and $result = 0$,

$R = result \oplus b_i = 0 \oplus 1 = 1$.

Therefore, the correct of the secure comparing protocol has been verified.

Appendix B. Security proof

We provide a formal security proof for Theorem 1 as follows.

Proof. We prove the security goal of our scheme by **induction**. Consider that when no EHR is encrypted, our scheme starts with the same initial state which is independent of the bit b. Namely, the corresponding MSOPE ciphertexts of $w^0_{*,1}$ and $w^1_{*,1}$ are identical. Then, we assume that it holds for i rounds, i.e., both the EHR sequences $w^0_{*,1}, w^0_{*,2}, \ldots, w^0_{*,i}$ and $w^1_{*,1}, w^1_{*,2}, \ldots, w^1_{*,i}$ generate two identical secret states, which leads to an identical MSOPE ciphertext sequences $c_{*,1}, c_{*,2}, \ldots, c_{*,i}$.

In the $(i+1)$ round, we assume that both $w^0_{*,i+1}$ and $w^1_{*,i+1}$ are provided by $\mathcal{P}_x$. Therefore, the $(i+1)$th EHR is encrypted by $\mathcal{P}_x$, i.e., $c_{*,i+1} = c_{x,i+1}$. We have three cases.

Case 1. $w^b_{x,i+1} = w^b_{x,j}$ and $j < i+1$. Namely, $\mathcal{P}_x$ has encrypted another EHR $w^b_{x,j}$ whose value is equal to $w^b_{x,i+1}$. According to the secret state update phase in our **Encryption** algorithm, the secret state of both sequences will not change, and the MSOPE ciphertext of $w^b_{x,i+1}$ will be equal to $w^b_{x,j}$. Namely, $c_{x,i+1} = c_{x,j}$. Since $c_{x,j}$ is independent of b, $c_{x,i+1}$ is independent of b.

Case 2. $w^b_{x,i+1} = w^b_{y,j}$, and $j < i+1$. Namely, another patient $\mathcal{P}_y$ has encrypted an EHR $w^b_{y,j}$, whose value is equal to $w^b_{x,i+1}$. In this case, the secret state will be updated according to the secret state update phase in our **Encryption** algorithm, and the result of update is dependent on a random coin b_x in **Secure Comparing Protocol**. Since b_x of both $w^0_{x,i+1}$ and $w^1_{x,i+1}$ are the same, the corresponding MSOPE ciphertexts of both $w^0_{x,i+1}$ and $w^1_{x,i+1}$ are the same. Since b_x is randomly chosen by $\mathcal{P}_x$ and is independent of b, $c_{x,i+1}$ is independent of b.

Case 3. $w^b_{x,i+1}$ has not been encrypted. $\mathcal{P}_x$ interacts with $\mathcal{CS}$ and updates the secret state. Since W^0 and W^1 have the same order relation, the secret state of both plaintexts are the same. Hence, the MSOPE ciphertexts of both plaintexts must be the same. Therefore, $c_{x,i+1}$ is independent of b.

Therefore, our encryption algorithm produces the same MSOPE ciphertext sequence for W^0 and W^1, and hence $\mathcal{A}$ cannot distinguish between them. Note that if different patients provide identical plaintexts, the corresponding MSOPE ciphertexts are randomized (like the second case). Therefore, our scheme is IND-MSOCPA secure. □

References

[1] G.M. Adelson-Velskii, E.M. Landis, An information organization algorithm, in: Doklady Akademia Nauk SSSR, Vol. 146, 1962, pp. 263–266.
[2] R. Agrawal, J. Kiernan, R. Srikant, Y. Xu, Order preserving encryption for numeric data, in: Proceedings of ACM SIGMOD, ACM, 2004, pp. 563–574.
[3] A. Boldyreva, N. Chenette, Y. Lee, A. O'Neill, Order-preserving symmetric encryption, in: Proceedings of EUROCRYPT, Springer, 2009, pp. 224–241.
[4] A. Boldyreva, N. Chenette, A. O'Neill, Order-preserving encryption revisited: Improved security analysis and alternative solutions, in: Proceedings of CRYPTO, Springer, 2011, pp. 578–595.
[5] D. Boneh, K. Lewi, M. Raykova, A. Sahai, M. Zhandry, J. Zimmerman, Semantically secure order-revealing encryption: Multi-input functional encryption without obfuscation, in: Proceedings of ASIACRYPT, Springer, 2015, pp. 563–594.
[6] N. Cheng, F. Lyu, J. Chen, W. Xu, H. Zhou, S. Zhang, X. Shen, Big data driven vehicular networks, IEEE Network 32 (6) (2018) 160–167.
[7] M. Du, Q. Wang, M. He, J. Weng, Privacy-preserving indexing and query processing for secure dynamic cloud storage, IEEE Trans. Inf. Forensics Secur. 13 (2018) 2320–2332.
[8] P. Grubbs, K. Sekniqi, V. Bindschaedler, M. Naveed, T. Ristenpart, Leakage-abuse attacks against order-revealing encryption, in: Proceedings of IEEE Security and Privacy (SP), IEEE, 2017, pp. 655–672.
[9] F. Han, J. Qin, J. Hu, Secure searches in the cloud: A survey, Future Gener. Comput. Syst. 62 (2016) 66–75.
[10] C. Huang, R. Lu, X. Lin, X. Shen, Secure automated valet parking: A privacy-preserving reservation scheme for autonomous vehicles, IEEE Trans. Veh. Technol. 67 (11) (2018) 11169–11180.
[11] C. Huang, R. Lu, H. Zhu, J. Shao, X. Lin, FSSR: Fine-grained EHRs sharing via similarity-based recommendation in cloud-assisted ehealthcare system, in: Proceedings of ACM ASIACCS, ACM, 2016, pp. 95–106.
[12] K.A. Jagadeesh, D.J. Wu, J.A. Birgmeier, D. Boneh, G. Bejerano, Deriving genomic diagnoses without revealing patient genomes, Science 357 (6352) (2017) 692–695.
[13] F. Kerschbaum, Frequency-hiding order-preserving encryption, in: Proceedings of ACM SIGSAC, ACM, 2015, pp. 656–667.
[14] F. Kerschbaum, A. Schroepfer, Optimal average-complexity ideal-security order-preserving encryption, in: Proceedings of ACM SIGSAC, ACM, 2014, pp. 275–286.
[15] K. Lewi, D.J. Wu, Order-revealing encryption: New constructions, applications, and lower bounds, in: Proceedings of ACM SIGSAC, ACM, 2016, pp. 1167–1178.
[16] J. Liang, Z. Qin, S. Xiao, L. Ou, X. Lin, Efficient and secure decision tree classification for cloud-assisted online diagnosis services, IEEE Trans. Dependable Secure Comput. (2019) 1–13, http://dx.doi.org/10.1109/TDSC.2019.2922958, (in press).
[17] J. Liang, Z. Qin, S. Xiao, J. Zhang, H. Yin, K. Li, MPOPE: multi-provider order-preserving encryption for cloud data privacy, in: Proceedings of SecureComm, 2017, pp. 808–822.
[18] X. Liu, Q. Liu, T. Peng, J. Wu, Dynamic access policy in cloud-based personal health record (PHR) systems, Inform. Sci. 379 (2017) 62–81.
[19] Q. Liu, Y. Tian, J. Wu, T. Peng, G. Wang, Enabling verifiable and dynamic ranked search over outsourced data, IEEE Trans. Serv. Comput. (2019) 1–14, http://dx.doi.org/10.1109/TSC.2019.2922177, (in press).
[20] Q. Liu, G. Wang, F. Li, S. Yang, J. Wu, Preserving privacy with probabilistic indistinguishability in weighted social networks, IEEE Trans. Parallel Distrib. Syst. 28 (5) (2017) 1417–1429.
[21] Q. Liu, S. Wu, S. Pei, J. Wu, T. Peng, G. Wang, Secure and efficient multi-attribute range queries based on comparable inner product encoding, in: Proceedings of IEEE CNS, IEEE, 2018, pp. 1–9.
[22] F. Lyu, H. Zhu, N. Cheng, H. Zhou, W. Xu, M. Li, X. Shen, Characterizing urban vehicle-to-vehicle communications for reliable safety applications, IEEE Trans. Intell. Transp. Syst. (2018) 1–17, http://dx.doi.org/10.1109/TITS.2019.2920813, (in press).
[23] L. Ou, Z. Qin, S. Liao, Y. Hong, X. Jia, Releasing correlated trajectories: Towards high utility and optimal differential privacy, IEEE Trans. Dependable Secure Comput. (2018) 1–13, http://dx.doi.org/10.1109/TDSC.2018.2853105, (in press).
[24] P. Paillier, Public-key cryptosystems based on composite degree residuosity classes, in: Proceedings of EUROCRYPT, Springer, 1999, pp. 223–238.
[25] R.A. Popa, F.H. Li, N. Zeldovich, An ideal-security protocol for order-preserving encoding, in: Proceedings of IEEE Security and Privacy (SP), IEEE, 2013, pp. 463–477.
[26] R.A. Popa, C.M.S. Redfield, N. Zeldovich, H. Balakrishnan, CryptDB: Protecting confidentiality with encrypted query processing, in: Proceedings of the ACM SOSP, ACM, ISBN: 978-1-4503-0977-6, 2011, pp. 85–100.
[27] T. Rabesandratana, E.U. privacy protection bill would hamper research, scientists warn, Science 343 (6174) (2014) 959–960.
[28] H. Ren, H. Li, Y. Dai, K. Yang, X. Lin, Querying in internet of things with privacy preserving: Challenges, solutions and opportunities, IEEE Network 32 (6) (2018) 144–151.
[29] D.S. Roche, D. Apon, S.G. Choi, A. Yerukhimovich, POPE: Partial order preserving encoding, in: Proceedings of ACM SIGSAC, ACM, 2016, pp. 1131–1142.
[30] Q. Wang, M. Du, X. Chen, Y. Chen, P. Zhou, X. Chen, X. Huang, Privacy-preserving collaborative model learning: The case of word vector training, IEEE Trans. Knowl. Data Eng. 30 (2018) 2381–2393.
[31] Q. Wang, M. He, M. Du, S.S.M. Chow, R.W.F. Lai, Q. Zou, Searchable encryption over feature-rich data, IEEE Trans. Dependable Secure Comput. 15 (3) (2018) 496–510.

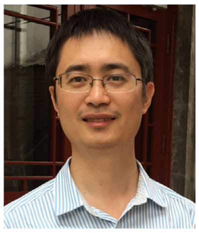
[32] L. Xiao, I.-L. Yen, D.T. Huynh, Extending order preserving encryption for multi-user systems., IACR Cryptology ePrint Archive 2012 (2012) 192.
[33] A. Yang, J. Xu, J. Weng, J. Zhou, D.S. Wong, Lightweight and privacy-preserving delegatable proofs of storage with data dynamics in cloud storage, IEEE Trans. Cloud Comput. (2018) 1–14, http://dx.doi.org/10.1109/TCC.2018.2851256, (in press).
[34] H. Yang, Q. Zhou, M. Yao, R. Lu, H. Li, X. Zhang, A practical and compatible cryptographic solution to ads-b security, IEEE Internet Things J. 6 (2) (2018) 3322–3334.
[35] X. Yao, Y. Lin, Q. Liu, S. Long, Efficient and privacy-preserving search in multi-source personal health record clouds, in: Proceedings of IEEE ISCC, IEEE, 2015, pp. 803–808.
[36] X. Yao, Y. Lin, Q. Liu, J. Zhang, Privacy-preserving search over encrypted personal health record in multi-source cloud, IEEE Access 6 (2018) 3809–3823.
[37] H. Yin, Z. Qin, J. Zhang, L. Ou, K. Li, Achieving secure, universal, and fine-grained query results verification for secure search scheme over encrypted cloud data, IEEE Trans. Cloud Comput. (2017) 1–14, http://dx.doi.org/10.1109/TCC.2017.2709318, (in press).
[38] Y. Zhang, C. Xu, H. Li, K. Yang, J. Zhou, X. Lin, Healthdep: An efficient and secure deduplication scheme for cloud-assisted ehealth systems, IEEE Trans. Ind. Inf. 14 (9) (2018) 4101–4112.
[39] Y. Zhang, C. Xu, X. Lin, X. Shen, Blockchain-based public integrity verification for cloud storage against procrastinating auditors, IEEE Trans. Cloud Comput. (2019) 1–15, http://dx.doi.org/10.1109/TCC.2019.2908400, (in press).
[40] Y. Zhang, C. Xu, J. Ni, H. Li, X. Shen, Blockchain-assisted public-key encryption with keyword search against keyword guessing attacks for cloud storage, IEEE Trans. Cloud Comput. (2019) 1–14, http://dx.doi.org/10.1109/TCC.2019.2923222, (in press).

Jinwen Liang received the B.S. degree from College of Computer Science and Electronic Engineering, Hunan University, in 2015, where he is currently pursuing the Ph.D. degree. His research interests include privacy-preserving machine learning, property-preserving encryption, and searchable symmetric encryption.

Zheng Qin received the Ph.D. degree in computer science from Chongqing University, P. R. China, in 2001. He is a professor in the College of Computer Science and Electronic Engineering at Hunan University, China. His research interests include machine learning, applied cryptography, big data, and cloud computing.

Sheng Xiao earned his B.E. (2002) from Department of Electronic Engineering, Tsinghua University, M.Sc. (2003) from National University of Singapore and Massachusetts Institute of Technology (Singapore-MIT Alliance). In 2003, He joined a startup company to lead a team for algorithmic design and verification of mobile co-processor. After successful tape-out of the chip, he moved to earn his Ph.D. (2013) from Department of Electrical and Computer Engineering, University of Massachusetts, Amherst. He is now serving as a faculty member in College of Computer Science and Electronic Engineering, Hunan University. His research interests include self-secure communication and network, data analysis and visualization, HPC applications.

Jixin Zhang received the B.S. degree in mathematics from the Hubei University of Technology and the M.S. degree in computer science and technology from the Wuhan University of Technology in 2011 and 2014, respectively. He is currently pursuing the Ph.D. degree with the College of Information Science and Engineering, Hunan University, and doing research with the Department of Information Engineering, The Chinese University of Hong Kong. His primary researches focus on security, machine learning.



Hui Yin received his B.S. in Computer Science from Hunan Normal University, China, in 2002; M.S. in Computer Software and Theory from Central South University, China, in 2008; and his Ph.D. degree from the College of Information Science and Engineering at Hunan University, China, in 2018. He is currently an assistant professor at the College of Applied Mathematics and Computer Engineering, Changsha University, China. His interests are information security, privacy protection, and applied cryptography.

Keqin Li is a SUNY Distinguished Professor of computer science with the State University of New York. He is also a Distinguished Professor at Hunan University, China. His current research interests include cloud computing, fog computing and mobile edge computing, energy-efficient computing and communication, embedded systems and cyber–physical systems, heterogeneous computing systems, big data computing, high-performance computing, CPU–GPU hybrid and cooperative computing, computer architectures and systems, computer networking, machine learning, intelligent and soft computing. He has published over 680 journal articles, book chapters, and refereed conference papers, and has received several best paper awards. He currently serves or has served on the editorial boards of the IEEE Transactions on Parallel and Distributed Systems, the IEEE Transactions on Computers, the IEEE Transactions on Cloud Computing, the IEEE Transactions on Services Computing, and the IEEE Transactions on Sustainable Computing. He is an IEEE Fellow.