IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received October 22, 2021, accepted November 2, 2021, date of publication November 8, 2021, date of current version November 15, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3126200

Efficient Search Over Encrypted Medical Data With Known-Plaintext/Background Models and Unlinkability

SHERIF ABDELFATTAH[1], MOHAMED BAZA[2], MAHMOUD M. BADR[1], MOHAMED M. E. A. MAHMOUD[1], (Senior Member, IEEE), GAUTAM SRIVASTAVA[3,4], (Senior Member, IEEE), FAWAZ ALSOLAMI[5], AND ABDULLAH MARISH ALI[5]

[1]Department of Electrical & Computer Engineering, Tennessee Tech University, Cookeville, TN 38505, USA
[2]Department of Computer Science, College of Charleston, Charleston, SC 29407, USA
[3]Department of Mathematics and Computer Science, Brandon University, Brandon, MB R7A 6A9, Canada
[4]Research Centre for Interneural Computing, China Medical University, Taichung 404, Taiwan
[5]Department of Computer Science, King Abdulaziz University, Jeddah 21341, Saudi Arabia

Corresponding author: Mohamed M. E. A. Mahmoud (mmahmoud@tntech.edu)

This work was supported by the Deanship of Scientific Research (DSR) at King Abdulaziz University, Jeddah, under Grant KEP-16-611-42.

ABSTRACT In advanced health care systems, patients' medical data can be outsourced to cloud servers to enable remote healthcare service providers to access and analyze patients' data from any location to provide better treatment. However, outsourcing sensitive medical data makes data owners, i.e., patients, concerned about their privacy because private companies run the cloud service and the data can be accessed by them. Therefore, it is important to encrypt the data in the form of documents before outsourcing them to the cloud in a way that enables a data user, i.e., a doctor, to search over these documents without allowing the cloud provider to learn any private information about patients. Several schemes have been proposed to enable search over encrypted medical cloud data to preserve patient privacy, but the existing schemes suffer from high communication/computation overhead because they are designed for a single-data-owner setting. Moreover, they are not secure against known-plaintext/background and linkability attacks and do not allow doctors to customize their search to avoid downloading irrelevant documents. In this paper, we develop an efficient search scheme over encrypted data for a multi-data-owner setting. To secure our scheme, the cloud server obtains noisy similarity scores and doctors de-noise them to download the most relevant documents. Our scheme enables doctors to prescribe search conditions to customize the search without revealing the conditions to the server. Our formal proof and analysis indicate that our scheme can preserve privacy and is secure against known-plaintext/background and linkability attacks, and the results of extensive experiments demonstrate the efficiency of our scheme compared to the existing works.

INDEX TERMS Security, privacy, e-health, searchable encryption schemes, cloud computing.

I. INTRODUCTION

Due to the cloud computing capability of storing large scale databases [1], the patients' medical data can be outsourced to cloud servers through a high-speed cellular network, e.g., 5G network and beyond [2], [3]. The cloud enables remote healthcare service providers to access patients' data from any location to analyze this data using data mining [4] and machine learning [5] techniques for providing better treatment [6], [7].

Well-known examples for cloud-based health systems are the national e-health infrastructures in Finland and Croatia [8]. Also, the USA is widely implementing cloud-based health services, and the market cap is expected to exceed $40 billion by 2026 [9]. However, outsourcing sensitive medical data makes *data owners*, i.e., patients, concerned about their

The associate editor coordinating the review of this manuscript and approving it for publication was Jerry Chun-Wei Lin.

privacy because private companies run the cloud service and the data can be accessed by them. For instance, over 113 million clinical records were hacked in the US in 2015 [10].

Therefore, it is essential to encrypt the data in the form of documents before outsourcing them to the cloud in a way that enables a *data user*, i.e., a doctor, to search over these documents without allowing the cloud provider to learn any private information about patients. To enable doctors to download documents of interest without revealing any information to the server, several schemes have been developed for searching over encrypted data [11]–[15]. The idea is that patients attach with each document an encrypted vector (called index) for the keywords of the document. Then, a doctor encrypts a vector (called trapdoor) that contains the keywords of the documents he/she wants to download and sends it to the cloud server. The server can compute the similarity score of an index and a trapdoor without being able to learn their keywords and returns to the doctor relevant documents.

Motivations. The existing schemes suffer from several limitations.

Firstly, these schemes suffer from high communication/computation overhead and the need for a large number of keys because they are designed for a single-data-owner setting (one patient and multiple doctors). In medical applications, a multi-data-owner setting (multiple patients and multiple doctors) is more appropriate because a doctor treats several patients, and thus he should be able to search the documents of these patients efficiently. In the existing schemes, a doctor needs to use a unique key for each patient to be able to search his/her documents, which makes key management inefficient due to using many keys at the doctor's side.

Secondly, in the existing schemes, doctors cannot customize their search scope to download only the documents that achieve certain search conditions, which may result in downloading irrelevant documents, and thus wasting communication and computation resources. An example of a search condition is laboratory reports with a certain issuance date.

Thirdly, the existing schemes are vulnerable to known-plaintext/background attacks and linkability. In the *known plaintext* attack, an adversary can decrypt encrypted data (indices and trapdoors) if he possesses a set of plaintext/ciphertext pairs. In the *known background* attacks, an adversary uses background (or statistical) information, such as the frequency of keywords, to infer the keywords of the documents by analyzing the frequency of downloading these documents, which may reveal sensitive information on the patients' health condition. The existing schemes also suffer from linkability attacks in which the server can link the trapdoors (or indices) that have the same keywords. The existing schemes try to thwart this attack by using random numbers in the encryption so that two trapdoors having the same keywords look different, but this is not enough because the server can link the trapdoors by observing that they give the same scores when they matched to all the documents.

TABLE 1. Comparison with the related works.

Schemes	$\mathbb{F}_1$	$\mathbb{F}_2$	$\mathbb{F}_3$	$\mathbb{F}_4$	$\mathbb{F}_5$	$\mathbb{F}_6$	$\mathbb{F}_7$	$\mathbb{F}_8$
[16]	$\surd^*$	$\times$	$\times$	$\surd$	$\times$	$\times$	$\surd^*$	$\times$
[12]	$\times$	$\surd$	$\times$	$\surd$	$\surd$	$\times$	$\times$	$\times$
[17]	$\surd^*$	$\times$	$\times$	$\times$	$\times$	$\surd^*$	$\times$	$\times$
EPSM	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$

$\mathbb{F}_1$: Unlinkability of indices/trapdoors; $\mathbb{F}_2$: Using unique key for each doctor; $\mathbb{F}_3$: Support multi-data-owner setting; $\mathbb{F}_4$: Support multi-data-user setting; $\mathbb{F}_5$: A patient/doctor cannot decrypt the indices/trapdoors of other patients/doctors; $\mathbb{F}_6$: Secure against *known plaintext model*; $\mathbb{F}_7$: Secure against *known background model*; $\mathbb{F}_8$: Customized search. Note: $\surd$ denote a realized feature, $\surd^*$ denote a realized feature with lower search accuracy and $\times$ denotes an unrealized feature.

Contributions. To address the aforementioned limitations, we propose **EPSM**: an **E**fficient and **P**rivacy-preserving **S**earch over **M**edical cloud data with known plaintext/background and unlinkability security. We provide a formal proof and privacy analysis for EPSM to prove that our scheme is secure and can preserve the privacy of the patients. Moreover, we conduct extensive experiments to evaluate the performance of our scheme and compare it to the existing works. Specifically, the main contributions of this paper are listed as follows:

- EPSM enables *customized search* in multi-data-owner and multi-data-user settings so that doctors can prescribe search conditions in trapdoors to limit the search scope to the documents that can satisfy the conditions, without revealing the conditions to the server. In EPSM, the cloud server computes noisy similarity scores for indices and trapdoors and doctors de-noise them to download the most relevant documents. Moreover, unlike the existing schemes, EPSM allows each doctor to use only one key to search the data of all patients he treats.
- Our security analysis proves that EPSM is secure under *known plaintext/background* models, and the cloud server cannot link two trapdoors (or indices) that have the same keywords.
- Extensive experiments are conducted, and the results indicate that EPSM requires a low overhead compared to the existing schemes.

The organization of this paper is as follows. The network and threat models and design goals are presented in Section III. In Section IV, the proposed EPSM is explained in detail. Then, we analyze the security and privacy of EPSM in Section V. In Section VI, we present the performance evaluation of EPSM. Section II provides the related works. Finally, conclusions are drawn in Section VII.

II. RELATED WORK

In this section, we review the related works and compare them to EPSM.

Song *et al.* [18] and Boneh *et al.* [19] have proposed secure searchable symmetric encryption (SSE) schemes based on k

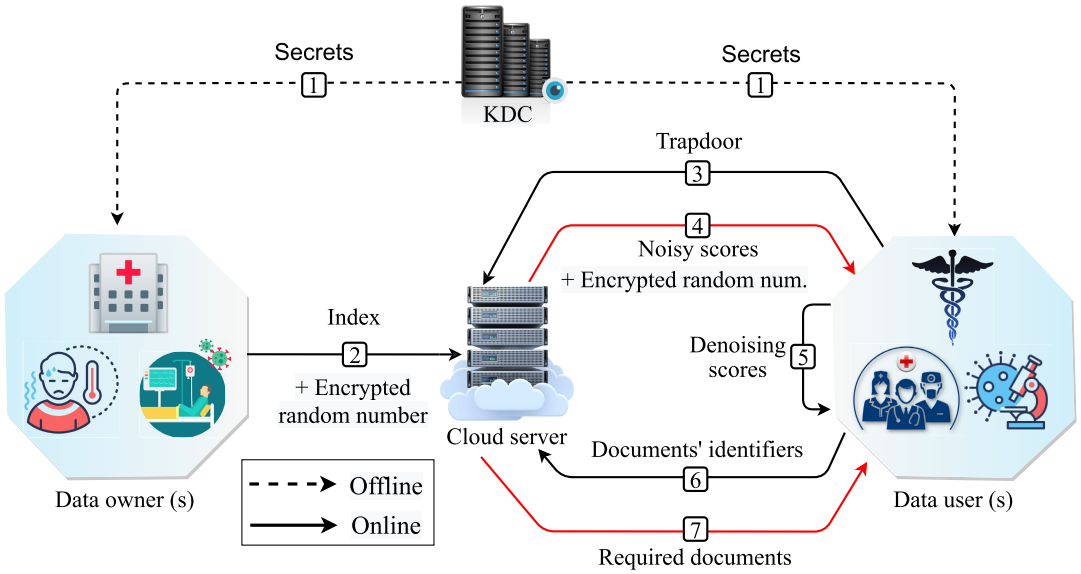


FIGURE 1. EPSM network model.

nearest neighbour (kNN) technique. However, these schemes are designed to support single keyword search over encrypted data, which is very restrictive because searching documents need multiple keywords to give accurate results. The schemes also suffer from high computation/communication overheads.

Wang *et al.* [20] have proposed a ranked search scheme. In this scheme, the cloud server executes the search process and sends back only the topmost relevant documents to the user. However, this scheme only considers the single keyword search. Then, Cao *et al.* [11] have proposed a privacy-preserving multi-keyword ranked search scheme. This scheme has a limitation in that it does not consider the keyword frequency, and this may result in inaccurate search results.

Xia *et al.* [16] have proposed a searchable encryption scheme for single-data-owner and multi-data-users setting. The scheme assumes that the server knows the term frequency of each keyword and it uses this background information to guess the keywords of a trapdoor and an index from the similarity score it computes. To thwart this attack, the server ranks the documents using inaccurate similarity scores, but this leads to inaccurate search results and downloading irrelevant documents which may cause misdiagnosis by doctors. Also, the proposed scheme is designed for a single-data-owner setting which is not suitable for medical applications where a doctor typically treats several patients and it is inefficient to use single-data-owner schemes in a multiple-data-owner setting as explained in Section VI.

Xiangya *et al.* [17] have proposed a privacy-preserving keyword search scheme for single-data-owner and single-data-user settings. This setting is not suitable for medical applications that have multiple patients and multiple doctors. To secure the scheme against the *known plaintext* model, the server ranks the documents using inaccurate similarity scores which may result in downloading irrelevant documents. Also, there is a tradeoff between accuracy and security because higher security is achieved by increasing the inaccuracy of the similarity scores, but downloading wrong documents is more likely.

Zhang *et al.* [14], have proposed a scheme for multi-keyword ranked search. The scheme uses an additive order function to retrieve the relevant search results. After receiving a trapdoor from a search user, the cloud server compares each encrypted keyword in the trapdoor with all the keywords of each data owner. Then, the cloud server adds all the document's scores with all the matched keywords. However, because of comparing the individual keywords in the trapdoor with all the keywords, this scheme requires high computation overhead. In [12], Li *et al.* have proposed a searchable encryption scheme over medical cloud data. To prevent linking the indices/trapdoors that have the same keywords, the scheme uses random numbers in the encryption so that they look different even if they have the same keywords. However, the scheme is designed for a single-data-owner setting, and the cloud server can link trapdoors (or indices) having the same keywords by observing that they give the same scores when they are matched to the documents' indices (or doctors' trapdoors).

We provide Table 1 to summarize the differences between EPSM and the aforementioned schemes. Unlike the existing schemes, EPSM supports multi-data-owner and multi-data-user settings. Also, EPSM enables a customized search feature that allows doctors to customize their search results. EPSM ensures the unlinkability of indices/trapdoors having the same keywords and ensures that the indices (or trapdoors)

TABLE 2. Main notations.

Notation	Description
m	vector size
e	vector extension size
$\mathcal{SK}_1, \mathcal{SK}_2$	KDC secret keys
S	Secret binary vector of $\mathcal{SK}_1$
$\{M_1, M_2, N_1, ..., N_8\}$	Server secret matrices of $\mathcal{SK}_1$
J	Secret binary vector of $\mathcal{SK}_2$
$\{V_1, V_2, U_1, \ldots, U_8\}$	Server secret matrices of $\mathcal{SK}_2$
$\mathcal{P}_i$	ID of data owner i
$\mathcal{SK}^1_{P_i}, \mathcal{SK}^2_{P_i}$	$\mathcal{P}_i$'s secret keys
$\{A_i, B_i, C_i, D_i\}$	Random matrices for $\mathcal{P}_i$ of $\mathcal{SK}^1_{P_i}$
$\{O_i, P_i, T_i, R_i\}$	Random matrices for $\mathcal{P}_i$ of $\mathcal{SK}^2_{P_i}$
$\mathcal{D}_x$	ID of doctor x
$\mathcal{SK}^1_{D_x}, \mathcal{SK}^2_{D_x}$	$\mathcal{D}_x$'s secret keys
$\{E_x, F_x, G_x, H_x\}$	Random matrices for $\mathcal{D}_x$ of $\mathcal{SK}^1_{D_x}$
$\{W_x, X_x, Y_x, H_x\}$	Random matrices for $\mathcal{D}_x$ of $\mathcal{SK}^2_{D_x}$
$t_{\mathcal{P}_i}$	Number of $\mathcal{P}_i$'s documents
$DC_i = \left\{d_{i,1}, d_{i,2}, \ldots, d_{i,n_{\mathcal{P}_i}}\right\}$	$\mathcal{P}_i$'s documents
$V_i = \left\{V_{i,1}, V_{i,2}, \ldots, V_{i,n_{\mathcal{P}_i}}\right\}$	$\mathcal{P}_i$'s documents' keywords vector
$a_i = \left\{a_{i,1}, a_{i,2}, \ldots, a_{i,n_{\mathcal{P}_i}}\right\}$	$\mathcal{P}_i$'s documents' random numbers
n	The length of $a_{i,j}$ in bits
$\mathcal{I}_{\mathcal{V}i} = \left\{I_{V_{i,1}}, I_{V_{i,2}} \ldots, I_{V_{i,n_{\mathcal{P}_i}}}\right\}$	$\mathcal{P}_i$'s indieces
$\mathcal{I}_{a_i} = \left\{I_{a_{i,1}}, I_{a_{i,2}} \ldots, I_{a_{i,n_{\mathcal{P}_i}}}\right\}$	$\mathcal{P}_i$'s encrypted random numbers
$t_{\mathcal{D}_x}$	Number of $\mathcal{D}_x$'s queries
$Q_x = \{Q_{x,1}, Q_{x,2}, \ldots, Q_{x,n_{\mathcal{D}_x}}\}$	$\mathcal{D}_x$'s queries
$\mathcal{I}_{\mathcal{Q}x} = \{I_{Q_{x,1}}, I_{Q_{x,2}}, \ldots, I_{Q_{i,n_{\mathcal{D}_x}}}\}$	$\mathcal{D}_x$'s trapdoors
$b_{x,y}$	$\mathcal{D}_x$'s query random number

computed by a patient (or a doctor) cannot be decrypted by other patients (or doctors). Our scheme is secure against *known plaintext* and *know background* models.

III. SYSTEM MODELS AND DESIGN GOALS

In this section, we present the network and threat models and design goals considered in this paper.

A. NETWORK MODEL

As shown in Fig. 1, the network model consists of four main entities, including an offline key distribution center (KDC), data owners (DO), data users (DU), and cloud server (CS). The role of each entity and the communication model are explained in this subsection.

- *Offline key distribution center (KDC):* The KDC is an offline entity that is not involved in the searching process. It computes and distributes the data owners' and data users' keys. The KDC can be run by the health department that is interested in the security of the system.
- *Data owners (DO):* The data owner is either a patient or a hospital, and it manages the patient's medical records. For each document, *DO* outsources to the cloud server an encrypted document, an encrypted vector containing the keywords of the document (called index), and an encrypted random number used in the index to mask the similarity scores.
- *Data users (DU):* Data users include doctors, nurses, pharmacists, researchers, etc. Each data user sends an encrypted query (called trapdoor) containing the keywords of the documents he wants to download from the cloud server. The data user receives the documents' noisy similarity scores, de-noises the scores, and sends the identifiers of the documents with the highest similarity scores to the cloud server to download them.
- *Cloud server (CS):* After receiving a trapdoor, the cloud server computes the noisy similarity scores of the trapdoor and the index of each document (that achieves the search conditions) and returns to the user the noisy scores. Then, after receiving the identifiers of the documents requested by the data user, the cloud server sends the documents.

In the rest of the paper, for simplicity, we will refer to *DO* and *DU* as patients and doctors, respectively.

B. THREAT MODEL

In EPSM, the attacker can be the cloud server and eavesdroppers. The cloud server is *honest-but-curious*, where it follows our scheme correctly but it is curious to infer sensitive information, such as the health condition of the patients, by analyzing the data it receives [16], [20]–[26]. Specifically, eavesdroppers can capture all the communications in the system and analyze them to infer sensitive information. The server should not be able to infer the keywords of the indices and the trapdoors or link two given trapdoors (or indices) if they have the same keywords or are sent from the same doctor. Moreover, EPSM should also be secure against the following attack models.

1) *Known ciphertext model*. In this model, the adversary only knows the encrypted indices and trapdoors [16], [27].
2) *Known plaintext model*. In this stronger model, the adversary has a set of tuples of indices (or trapdoors) and their corresponding plaintext keyword vectors. Using these plaintext-ciphertext pairs, the adversary may try to infer the keywords or the search conditions of other indices and the trapdoors [24], [28].
3) *Known background model*. In this model, the adversary possesses statistical information, such as the frequency of some keywords (or search conditions), i.e., the probability of querying documents with certain keywords. Using this information, the adversary tries to identify the keywords and the search conditions of the indices/trapdoors [16], [29].

C. DESIGN GOALS

To enable efficient and privacy-preserving search, EPSM should achieve the following design goals.

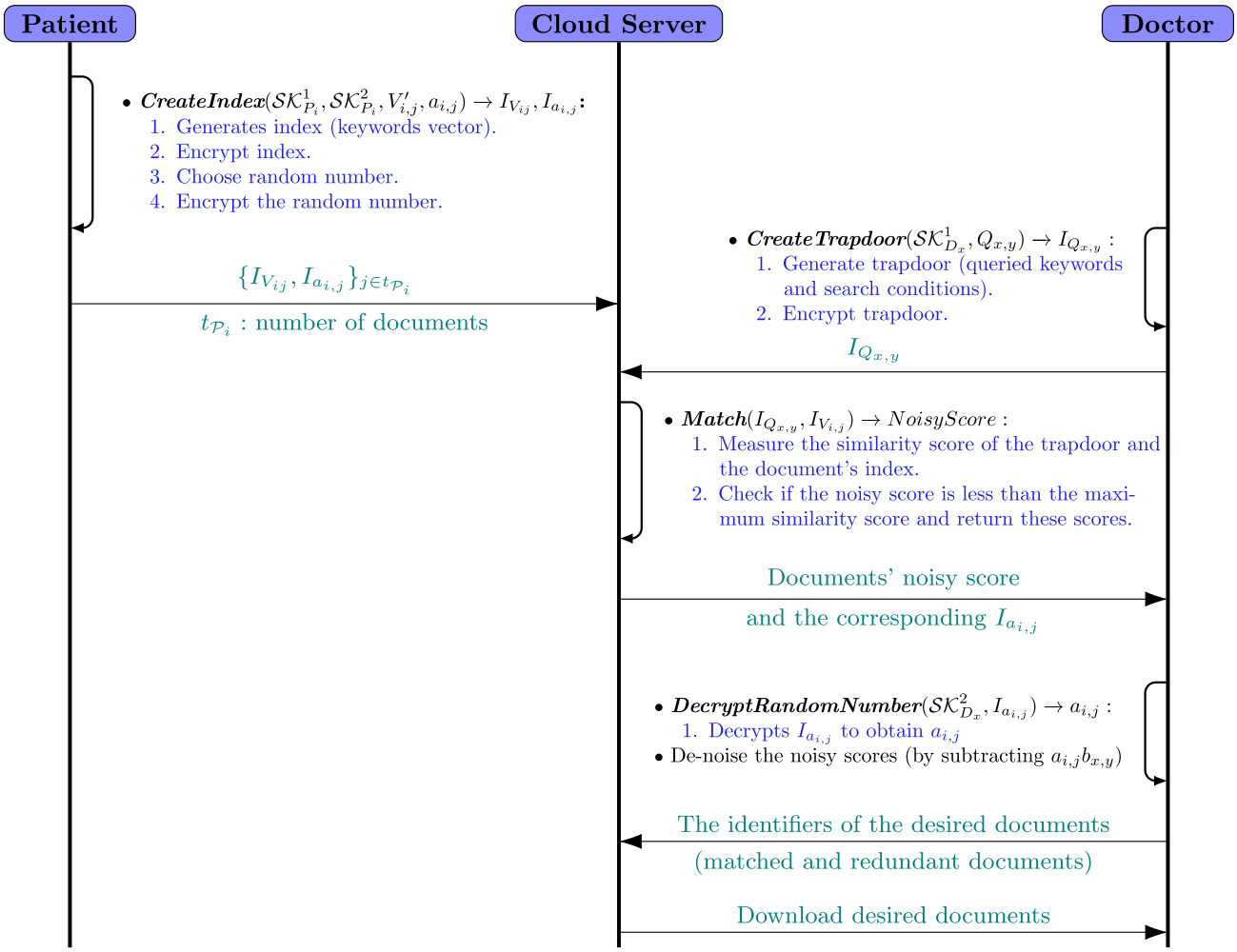


FIGURE 2. Overview for the oracles of EPSM.

(1) *Customized Search.* EPSM should enable doctors to prescribe conditions in trapdoors so that the server returns only the documents that can satisfy these conditions without being able to learn the conditions.

(2) *Security and Privacy Preservation.* EPSM should prevent the cloud server from inferring any information about the content of documents, indices, and trapdoors. EPSM should also be secure against the *Known plaintext* and *known background* models so that the server cannot identify the keywords or the search conditions of given indices/trapdoors. Also, the trapdoors (and indices) that have the same keywords and conditions or are sent from the same doctor should not be linkable. The eavesdroppers should not be able to infer any sensitive information.

(3) *Scalability and Efficiency.* EPSM should efficiently support the search for a large number of patients/doctors with a small number of keys for efficient key management. It should also need low search time and computation/communication overhead.

IV. PROPOSED SYSTEM

EPSM consists of four phases. In the *system initialization* phase, the KDC generates and distributes secret keys to patients and doctors. In the *index generation* phase, for each document, the patient composes the corresponding index and encrypts the random number used to mask the similarity score and outsources them to the cloud. In the *trapdoor generation* phase, the doctor encrypts a vector containing the keywords and search conditions of the documents he wants to download and sends the ciphertext, called trapdoor, to the cloud server. Finally, in the *query matching* phase, the server calculates the noisy similarity scores of the trapdoor and the indices of the documents that can achieve the search conditions. Then, it returns to the doctor the noisy scores to de-noise them and send to the server the identifiers of the documents he wants to download, i.e., the documents that have the highest scores. Finally, the cloud server returns to the doctor these documents. Table 2 gives the main notations used in the paper. Figure 2 shows an overview of EPSM

A. SYSTEM INITIALIZATION

The KDC runs the following algorithms to compute the secret keys of the patients and the doctors.

$Setup(1^m) \rightarrow \mathcal{SK}_1, \mathcal{SK}_2$: This algorithm takes the security parameter 1^m as an input and outputs two Keys $\mathcal{SK}_1$ and $\mathcal{SK}_2$. The first key is $\mathcal{SK}_1 = \{S, M_1, M_2, N_1, \ldots, N_8\}$, where, S is a random binary vector of length $(m + e + 2)$, and $\{M_1, M_2, N_1, \ldots, N_8\}$ are a set of random invertible matrices of size $(m + e + 2) \times (m + e + 2)$, where m and e are the sizes of the keywords and search conditions, respectively. The second key is $\mathcal{SK}_2 = \{J, V_1, V_2, U_1, \ldots, U_8\}$, where, J is a random binary vector of length n and $\{V_1, V_2, U_1, \ldots, U_8\}$ are a set of random invertible matrices of size $(n \times n)$, where n is the bit length of the random number the patient uses to mask the similarity score.

$KeyGenPatient(\mathcal{SK}_1, \mathcal{SK}_2, \mathcal{P}_i) \rightarrow \mathcal{SK}^1_{P_i}, \mathcal{SK}^2_{P_i}$: For each patient $\mathcal{P}_i$, this algorithm outputs two secret keys $\mathcal{SK}^1_{P_i}$ and $\mathcal{SK}^2_{P_i}$. $\mathcal{SK}^1_{P_i}$ is used to encrypt the keyword vectors to calculate the indices, and it is computed as follows:

$$\mathcal{SK}^1_{P_i} = \{S,\ N_1^{-1}A_i,\ N_2^{-1}B_i,\ N_3^{-1}A_i,\ N_4^{-1}B_i, \\ N_5^{-1}C_i,\ N_6^{-1}D_i,\ N_7^{-1}C_i,\ N_8^{-1}D_i\} \quad (1)$$

where, $\{A_i, B_i, C_i, D_i\}$ are $(m + e + 2) \times (m + e + 2)$ matrices of random numbers such that $A_i + B_i = M_1^{-1}$, and $C_i + D_i = M_2^{-1}$.

$\mathcal{SK}^2_{P_i}$ is used to encrypt the random number $\mathcal{P}_i$ uses to mask the similarity score, and it is computed as follows:

$$\mathcal{SK}^2_{P_i} = \{J,\ U_1^{-1}O_i,\ U_2^{-1}P_i,\ U_3^{-1}O_i,\ U_4^{-1}P_i, \\ U_5^{-1}T_i,\ U_6^{-1}R_i,\ U_7^{-1}T_i,\ U_8^{-1}R_i\} \quad (2)$$

where $\{O_i, P_i, T_i, R_i\}$ are $(n \times n)$ matrices of random numbers such that $O_i + P_i = V_1^{-1}$, and $T_i + R_i = V_2^{-1}$.

Finally, the KDC sends $\mathcal{SK}^1_{P_i}$ and $\mathcal{SK}^2_{P_i}$ to the patient $\mathcal{P}_i$.

$KeyGenDoctor(\mathcal{SK}_1, \mathcal{SK}_2, \mathcal{D}_x) \rightarrow \mathcal{SK}^1_{D_x}, \mathcal{SK}^2_{D_x}$: For each doctor $\mathcal{D}_x$, this algorithm outputs two secret keys $\mathcal{SK}^1_{D_x}$ and $\mathcal{SK}^2_{D_x}$. $\mathcal{SK}^1_{D_x}$ is used to encrypt the vectors of keywords to compose trapdoors and it is computed as follows.

$$\mathcal{SK}^1_{D_x} = \{S,\ E_xN_1,\ E_xN_2,\ F_xN_3,\ F_xN_4, \\ G_xN_5,\ G_xN_6,\ H_xN_7,\ H_xN_8\} \quad (3)$$

where, $\{E_x, F_x, G_x, H_x\}$ are $(m + e + 2) \times (m + e + 2)$ matrices of random numbers such that $E_x + F_x = M_1$, and $G_x + H_x = M_2$.

$\mathcal{SK}^2_{D_x}$ is used to decrypt the random numbers of the patients to de-noise the similarity scores, and it is computed as follows.

$$\mathcal{SK}^2_{D_x} = \{J,\ W_xU_1,\ W_xU_2,\ X_xU_3,\ X_xU_4, \\ Y_xU_5,\ Y_xU_6,\ Z_xU_7,\ Z_xU_8\} \quad (4)$$

where $\{W_x, X_x, Y_x, H_x\}$ are $(n \times n)$ matrices of random numbers such that $W_x + X_x = V_1$ and $Y_x + Z_x = V_2$.

Finally, the KDC sends $\mathcal{SK}^1_{D_x}$ and $\mathcal{SK}^2_{D_x}$ to the doctor.

B. INDEX GENERATION

To outsource a document, a patient $\mathcal{P}_i$ computes an index and an encrypted random number and sends them to the cloud server. To do so, the patient executes the following algorithm.

$CreateIndex(\mathcal{SK}^1_{P_i}, \mathcal{SK}^2_{P_i}, V'_{i,j}, a_{i,j}) \rightarrow I_{V_{ij}}, I_{a_{i,j}}$: This algorithm takes as input the patient's secret keys $\mathcal{SK}^1_{P_i}$ and $\mathcal{SK}^2_{P_i}$, a keyword vector $V'_{i,j}$ corresponding to the document, and a random number $a_{i,j}$, and outputs the index of the document ($I_{V_{ij}}$) and the encrypted random number ($I_{a_{i,j}}$).

For a document j, $\mathcal{P}_i$ chooses a keyword set $\{w_{i,j,1}, w_{i,j,2}, \ldots\}$ to generate an m-element keyword vector $V'_{i,j}$. Every element in $V'_{i,j}$ contains the relevance score of the TF-IDF (Term Frequency - Inverse Document Frequency) [30], [31], which represents the significance of keyword $w_{i,j,k}$ within the whole document collection, and it is computed as follows.

$$TF - IDF(w_{i,j,k}, d_{i,j}) = freq_{w_{i,j,k},d_{i,j}} * \log(\frac{N}{n_{w_{i,j,k}}}) \quad (5)$$

where, the frequency of the keyword $w_{i,j,k}$ is $freq_{w_{i,j,k},d_{i,j}}$, N represents the total number of keywords in the documents set, and $n_{w_{i,j,k}}$ is the total number of documents the keyword appears in. Then, $\mathcal{P}_i$ chooses a random number $a_{i,j}$ for the $(m+1)-th$ element in the vector $V'_{i,j}$. After that, $\mathcal{P}_i$ composes an $(m + e + 2)$-element vector $V_{i,j} = V'_{i,j}||EF_{i,j}$, where $EF_{i,j}$ has $(e + 1)$ elements for the search conditions. For example, assuming that there is one condition on the issuance year, an example for $V_{i,j}$ is shown in Fig. 3. The figure shows that, the element that represents the issuance year stores one (2021 in the figure) and the other elements store zeros. e elements are used to represent the years and one element stores one all the time. For simplicity, the figure shows the vector with one condition, but the idea can be extended to include multiple conditions.

Then, in order to encrypt $V_{i,j}$, $\mathcal{P}_i$ first splits it into two column vectors v'_{ij} and v''_{ij} using the secret S. So, for every element in $V_{i,j}$, $\mathcal{P}_i$ checks the value of the corresponding element in S. If it is zero, $\mathcal{P}_i$ sets the corresponding element in v'_{ij} and v''_{ij} with the same value of the element in $V_{i,j}$. Otherwise, $\mathcal{P}_i$ chooses two random numbers for this element in v'_{ij} and v''_{ij} where their summation is equal to the value of the corresponding element in $V_{i,j}$.

$$I_{V_{i,j}} = \Big[N_1^{-1}A_iv'_{ij};\ N_2^{-1}B_iv'_{ij};\ N_3^{-1}A_iv'_{ij}; \\ N_4^{-1}B_iv'_{ij};\ N_5^{-1}C_iv''_{ij};\ N_6^{-1}D_iv''_{ij}; \\ N_7^{-1}C_iv''_{ij};\ N_8^{-1}D_iv''_{ij}\Big] \quad (6)$$

where, $I_{V_{i,j}}$ is an $8(m + e + 2)$-element column vector.

Then, to encrypt the random number $a_{i,j}$, $\mathcal{P}_i$ first splits it into two column vectors $a'_{i,j}$ and $a''_{i,j}$ using the secret J. So, for every element in $a_{i,j}$, $\mathcal{P}_i$ checks the corresponding element in J. If it is zero, the corresponding elements in $a'_{i,j}$ and $a''_{i,j}$ are set to the same value of the element of $a_{i,j}$. Otherwise, two

Algorithm 1: Search Algorithm by the Cloud

```
Input: I_{Q_{x,y}}, I_{V_{i,j}} and I_{a_{i,j}}
Output: Λ and the corresponding I_{a_{i,j}}
        of each document
Λ ← Scorelist()
// Scorelist is a list to hold the
   scores of the matched documents
for j ← 1 to numberofdocuments do
    calculate the matching score
    Score(Q_{x,y} • V_{i,j}) ← Match(I_{Q_{x,y}}, I_{V_{i,j}})
    // where Score(Q_{x,y} • V_{i,j}) is the dot
       product noisy score of Q_{x,y} and
       V_{i,j}
    if Score(Q_{x,y} • V'_{i,j}) ≥ maxscore then
        ignore this index and continue
    else
        Λ ←
         Scorelist.Append(Λ, Score(Q_{x,y} • V_{i,j}))
    end
end
Output: Send Λ and the
        corresponding I_{a_{i,j}} of each
        document to D_x
```

random numbers are chosen for this element in $a'_{i,j}$ and $a''_{i,j}$ where their summation is equal to the corresponding element in $a_{i,j}$. Finally, the encryption of $a_{i,j}$ ($I_{a_{i,j}}$) is computed using $\mathcal{SK}^2_{P_i}$ as follow.

$$I_{a_{i,j}} = \Big[U_1^{-1}O_i a'_{i,j};\ U_2^{-1}P_i a'_{i,j};\ U3^{-1}O_i a'_{i,j};\ U_4^{-1}P_i a'_{i,j};\ U_5^{-1}T_i a''_{i,j};\ U_6^{-1}R_i a''_{i,j};\ U_7^{-1}T_i a''_{i,j};\ U_8^{-1}R_i a''_{i,j}\Big] \tag{7}$$

where, $I_{a_{i,j}}$ is a column vector of size $8n$.

Finally, for each document, $\mathcal{P}_i$ sends to the cloud server the corresponding index $I_{V_{i,j}}$ and the encryption of $a_{i,j}$ ($I_{a_{i,j}}$).

C. TRAPDOOR GENERATION

In this phase, to search for documents of interest, a doctor composes a query ($Q_{x,y}$) containing the keywords of interest and search conditions, and then uses the following algorithm to encrypt it and obtain the trapdoor $I_{Q_{x,y}}$.

CreateTrapdoor$(\mathcal{SK}^1_{D_x}, Q_{x,y}) \rightarrow I_{Q_{x,y}}$: This algorithm takes the doctor's secret key $\mathcal{SK}^1_{D_x}$ and the query vector $Q_{x,y}$ as input, and computes the trapdoor $I_{Q_{x,y}}$.

Firstly, the doctor $\mathcal{D}_x$ composes the $m + e + 2$-element query vector $Q_{x,y}$. The first m elements contain the keywords of interest where each element stores one or zero to indicate whether or not the corresponding keyword to the element exists in the documents of interest. Specifically, $Q_{x,y}[k] = 1$ if the doctor is interested in keyword k, and $Q_{x,y}[k] = 0$ if the doctor is not interested in the keyword. Then, a random number $b_{x,y}$ is selected for the $(m + 1) - th$ element. After that, $\mathcal{D}_x$ uses the following $e + 1$ elements to prescribe the search conditions as follows.

$$Q_{x,y}[k] = \begin{cases} -c, & \text{if } k \in \bar{F} \\ g \times c, & \text{if } k = m + e + 2 \\ 0, & \text{other} \end{cases} \tag{8}$$

where c is a random number that is greater than the maximum noisy similarity score, $\bar{F} \subset [m + 2, m + e + 1]$ is set of the elements' positions of the document issuance years that the doctor wants to search, and g is the length of $\bar{F}$.

For example, if the doctor wants to search for the documents issued in 2021 and download them as shown in Fig. 3c. He/she stores $-c$ in the element corresponding to 2021, c in the last element, and zeros in the other elements. Moreover, as illustrated in Fig. 3b, if the document is issued in 2021, the patient stores one in the corresponding element to 2021 and the last element and zero in the other elements of the index vector. By doing so, *if the condition is satisfied, the dot product of the elements of the conditions in the index and the trapdoor is equal to zero. Otherwise, it is c* whose value is greater than the maximum noisy similarity score. So, if the noisy similarity score obtained by the cloud server is greater than the maximum score, this indicates that the document does not satisfy the search conditions, otherwise, all the conditions are satisfied. For simplicity, Fig. 3 shows only one condition, but it can be extended to add additional conditions.

To encrypt the query vector $Q_{x,y}$ and obtain the trapdoor $I_{Q_{x,y}}$, $Q_{x,y}$ is first split into two row vectors q'_{xy} and q''_{xy} using the secret S, as follows. For every element in $Q_{x,y}$, $\mathcal{D}_x$ checks the corresponding element in S. If it is one, the corresponding element in q'_{xy} and q''_{xy} are set to the same value of the element of $Q_{x,y}$. Otherwise, two random numbers are chosen for this element in q'_{xy} and q''_{xy} where their summation is equal to the corresponding element in $Q_{x,y}$. Finally, the trapdoor $I_{Q_{x,y}}$ is computed using $\mathcal{SK}^1_{D_x}$ as follow.

$$I_{Q_{x,y}} = \Big[q'_{xy}E_xN_1,\ q'_{xy}E_xN_2,\ q'_{xy}F_xN_3,\ q'_{xy}F_xN_4,\ q''_{xy}G_xN_5,\ q''_{xy}G_xN_6,\ q''_{xy}H_xN_7,\ q''_{xy}H_xN_8\Big] \tag{9}$$

where $I_{Q_{x,y}}$ is an $8(m + e + 2)$-element row vector. Finally, $\mathcal{D}_x$ sends the trapdoor $I_{Q_{x,y}}$ to the cloud server.

D. QUERY MATCHING

In this phase, the cloud server computes the noisy similarity score of the trapdoor and the index of each document that achieves the search conditions without being able to learn the real score. Then, the server sends to the doctor the noisy scores and the encryptions of the random numbers the patients used to mask the scores as indicated in Algorithm 1. After that, the doctor de-noises the scores and sends to the cloud server the identifiers of the documents he wants to download. These documents include the ones that have high similarity scores in addition to redundant documents that are downloaded to protect against known-background attacks by

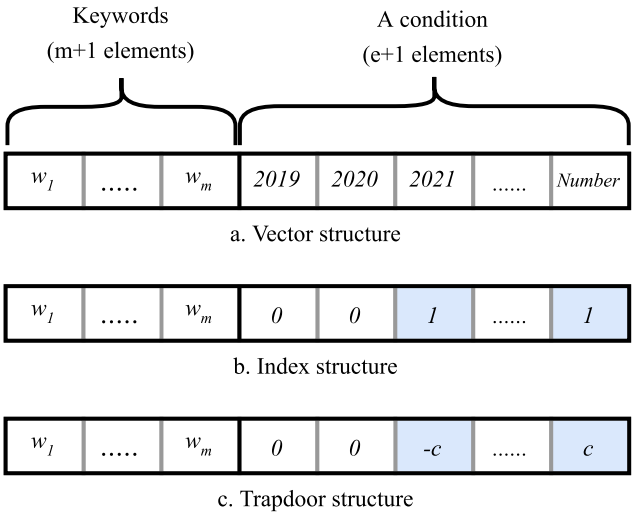


FIGURE 3. An illustration for our customized search feature.

preventing the server from learning the documents of interest and guessing the keywords of these documents. Finally, the cloud server returns to the doctor the documents he requested. The following algorithms are used in this phase.

Match$(I_{Q_{x,y}}, I_{V_{i,j}}) \rightarrow$ *NoisyScore*: This algorithm takes a trapdoor $I_{Q_{x,y}}$ and an index $I_{V_{i,j}}$ as input, and produces the noisy similarity score of $Q_{x,y}$ and $V_{i,j}$ by computing the dot product $(I_{Q_{x,y}} \cdot I_{V_{i,j}})$.

Theorem 1: The server can obtain the noisy similarity score of indices and trapdoors using dot product operation.

Proof:

$$\begin{aligned} I_{Q_{x,y}} \cdot I_{V_{i,j}} &= q'_{xy}E_xA_iv'_{ij} + q'_{xy}E_xB_iv'_{ij} + q'_{xy}F_xA_iv'_{ij} \\ &\quad + q'_{xy}F_xB_iv'_{ij} + q''_{xy}G_xC_iv''_{ij} + q''_{xy}G_xD_iv''_{ij} \\ &\quad + q''_{xy}H_xC_iv''_{ij} + q''_{xy}H_xD_iv''_{ij} \\ &= q'_{xy}(E_x + F_x).(A_i + B_i)v'_{ij} \\ &\quad + q''_{xy}(G_x + H_x).(C_i + D_i)v''_{ij} \\ &= q'_{xy}M_1M_1^{-1}v'_{ij} + q''_{xy}M_2M_2^{-1}v''_{ij} \\ &= Q_{x,y} \cdot V_{i,j} \end{aligned} \tag{10}$$

If all the search conditions prescribed in the trapdoor are satisfied, $Q_{x,y} \cdot V_{i,j} = KeywordScore + a_{i,j}b_{x,y}$, which gives the noisy similarity score that is equal to the similarity score of the keywords part in vectors $V_{i,j}$ and $Q_{x,y}$ (*KeywordScore*) masked by the random number $a_{i,j}b_{x,y}$, where $a_{i,j}$ is added by the patient in the document index and $b_{x,y}$ is added by the doctor in the trapdoor. If at least one condition is not satisfied $Q_{x,y} \cdot V_{i,j} = KeywordScore + a_{i,j}b_{x,y} + c$ and by selecting c to be greater than the maximum noisy similarity score, the server can learn that the document does not achieve at least one condition and it should discard the document. Finally, for each document that achieves the doctor's conditions, the cloud server returns to the doctor the noisy similarity score and the encryption of the random number $a_{i,j}$ $(I_{a_{i,j}})$ used by the patient to mask the similarity score.

For each document, the doctor decrypts $I_{a_{i,j}}$ to obtain $a_{i,j}$ using the algorithm *DecryptRandomNumber*(). Then, using this random number and his trapdoors' random number $b_{x,y}$, the doctor de-noises the noisy scores (by subtracting $a_{i,j}b_{x,y}$) to obtain the real scores. Then, the doctor sends to the cloud server the identifiers of the documents he wants to download, i.e., the documents that have the highest scores. The doctor should also download redundant documents to protect against known-background attacks by preventing the server from learning the documents of interest and guessing the keywords of these documents. Finally, the cloud server returns to the doctor these documents.

DecryptRandomNumber$(\mathcal{SK}^2_{D_x}, I_{a_{i,j}}) \rightarrow a_{i,j}$: This algorithm takes the doctor's secret key $\mathcal{SK}^2_{D_x}$ and the encrypted random number $I_{a_{i,j}}$, and outputs the random number $a_{i,j}$. The algorithm multiplies $\mathcal{SK}^2_{D_x}$ by $I_{a_{i,j}}$ to obtain $a'_{i,j}$ and $a''_{i,j}$, and then the splitting vector J is used to obtain $a_{i,j}$ as follows. For each element k^{th} in J, if $J[k]$ is one, $a'_{i,j}[k]$ and $a''_{i,j}[k]$ are added to obtain $a_{i,j}[k]$, while if $J[k]$ is zero, then set $a_{i,j}[k]$ is equal to $a'_{i,j}[k]$ or $a''_{i,j}[k]$.

Theorem 2: The doctor can decrypt the encrypted random number $I_{a_{i,j}}$ by multiplying it by $\mathcal{SK}^2_{D_x}$.

Proof:

$$\begin{aligned} &(W_xO_ia'_{i,j}) + (W_xP_ia'_{i,j}) + (X_xO_ia'_{i,j}) + (X_xP_ia'_{i,j}) \\ &\quad = (W_x + X_x).(O_i + P_i)a'_{i,j} \\ &\quad = V_1V_1^{-1}a'_{i,j} \\ &\quad = a'_{i,j} \end{aligned} \tag{11}$$

similarly,

$$a''_{i,j} = (Y_xT_ia''_{i,j}) + (Y_xR_ia''_{i,j}) + (Z_xT_ia''_{i,j}) + (Z_xR_ia''_{i,j}) \tag{12}$$

V. SECURITY AND PRIVACY ANALYSIS

Our formal proof of the security/privacy-preservation of our scheme follows the logic and model presented in [32]. The goal of the proof is to prove that the cloud server can compute the noisy similarity score of an index and a trapdoor without revealing their keywords and search conditions. We will also prove that external attackers cannot reveal the keywords and search conditions. The server and external attackers cannot also learn the similarity scores of the indices and trapdoors.

Preposition 1: The cloud server can calculate the noisy similarity score of an index and a trapdoor without being able to learn the keywords or the search conditions.

Proof:

History. The history consists of two sets, including a set of n indices corresponding to the documents of patients $(I_V = \{I_{V_{i,1}}, I_{V_{i,2}} \ldots, I_{V_{i,n}}\}$, for each patient $\mathcal{P}_i$ generated by encrypting a set of keywords vectors $V = \{V_{i,1}, V_{i,2} \ldots, V_{i,n}\})$ and a set of u trapdoors corresponding to the doctors' queries $(I_Q = \{I_{Q_{x,1}}, I_{Q_{x,2}}, \ldots, I_{Q_{x,u}}\}$, for each doctor $\mathcal{D}_x$ generated by encrypting a set of queries vectors $Q = \{Q_{x,1}, Q_{x,2}, \ldots, Q_{x,u}\})$.

Trace. A trace $Trace(H)$ represents the information of the history H that is deduced by the cloud server, e.g., from the search patterns.

View. The view $W(I_V, I_Q, Trace(H))$ has the encrypted history and its trace and it is the observation of the server.

A simulator $\mathcal{S}$ can produce a fake view W' that is indistinguishable from the original view W by executing these steps.

Step 1: $\mathcal{S}$ generates the secret key $sk' = \mathcal{SK}'$.

Step 2: $\mathcal{S}$ generates a set of random documents $D' = \{d'_1, \ldots, d'_n\}$ such that $|d_i| = |d'_i|$, $1 \leq i \leq n$, $d'_i = \{w_1, w_2, \ldots\}$, where $|d'_i|$ is the number of keywords in d'_i.

Step 3: $\mathcal{S}$ generates a set of queries as $Q' = \left\{Q'_{x,1}, Q'_{x,2}, \ldots, Q'_{x,u}\right\}$, where Q' is a random copy of Q.

Step 4: $\mathcal{S}$ generates a set of keyword vectors (V') which is a random copy of V, where $V' = \left\{V'_{i,1}, V'_{i,2} \ldots, V'_{i,n}\right\}$.

Step 5: S generates indices I'_V and trapdoors I'_Q using the secret sk'.

From the previous construction, EPSM is indistinguishable and secure if S has a trace $Trace(H')$ of the history $H' = (I'_V, I'_Q)$ that is similar to the original trace $Trace(H)$ such that in no probabilistic polynomial time, an adversary can differentiate between the original view $W(I_V, I_Q)$ and the fake view $W(I'_V, I'_Q)$ with non-negligible advantage, where the correctness of the construction implies this conclusion.

Preposition 2: EPSM ensures that adversaries can not reveal any keyword or search condition from trapdoors and/or indices, i.e., EPSM is secure in the known-ciphertext model.

Proof: In EPSM, the *confidentiality* of the indices and trapdoors is protected using encryption. For each patient/doctor, the matrix M is split into two randomly chosen matrices which are multiplied by another matrix N, and thus, no patient/doctor can reconstruct the matrix M. This is important because by knowing M, adversaries can compute the keywords or the search conditions from the indices or trapdoors. This means that the keywords and the conditions are protected in the *known-ciphertext* model because no information can be leaked about them.

Preposition 3: EPSM ensures that the indices (or trapdoors) computed by a patient (or a doctor) cannot be decrypted by other patients (or doctors).

Proof: If all patients (or doctors) share the same key, then the indices (or the trapdoors) computed by a patient (or a doctor) can be decrypted by other patients (or doctors). Thus, patients' sensitive information, e.g., their health condition, can be revealed by other patients. To avoid this problem in EPSM, each patient/doctor has a unique key, and, despite using different keys to encrypt the indices/trapdoors, the cloud server is still able to obtain the dot product of the keyword and query vectors and obtain the noisy similarity score.

Preposition 4: EPSM is secure against known-plaintext model if the random numbers $a_{i,j}$ and $b_{x,y}$ of each index and trapdoor are not known by the adversary.

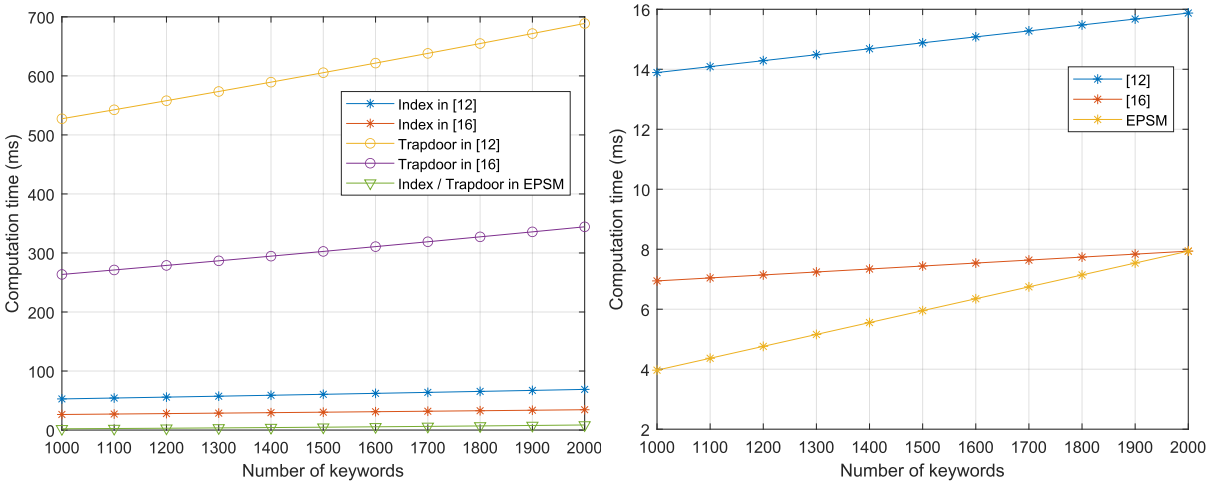
Proof: Under the *known-plaintext model*, the adversary possesses a set of plaintexts (keyword vector and queries) and their ciphertexts (indices and trapdoors). The adversary tries to use this set to attack the encryption scheme, e.g., by decrypting a new ciphertext. Most of the existing schemes are not secure against the known-plaintext model because the server can learn the similarity score, by calculating the dot product of an index and trapdoor. Therefore, if an index has n elements (i.e., n unknowns), the server needs n trapdoors (with known plaintexts) to create n linear equations and solve them to compute the n elements of the index. To protect against this attack in our scheme, the server does not know the similarity score. It only knows noisy similarity score (*real score* $+ a_{i,j}b_{x,y}$). The random numbers $a_{i,j}$ and $b_{x,y}$ are known only to the patient and doctor. The patients should use a different $a_{i,j}$ for each index and doctors should use a different $b_{x,y}$ in each trapdoor, so that $a_{i,j}b_{x,y}$ is always different even if the same query is used multiple times. By reusing $a_{i,j}$ and $b_{x,y}$, the server can subtract two equations to cancel the term $a_{i,j}b_{x,y}$ and obtain the difference between the two scores, and thus the server can create enough number of equations to obtain the keyword vector of an index. Therefore, by changing $a_{i,j}b_{x,y}$ continuously, the server cannot have enough number of equations to solve because $a_{i,j}b_{x,y}$ introduces a new unknown. Similarly, for the same reasons explained, the server cannot create equations to decrypt trapdoors.

Preposition 5: EPSM ensures that the cloud server or an external eavesdropper cannot identify the keywords and the conditions of the documents/trapdoors under the known background model.

Proof: In the *known background* attacks, an adversary uses background (or statistical) information, such as the frequency of keywords, to infer the keywords of the documents by analyzing the frequency of downloading these documents, which may reveal sensitive information on the patients such as their diseases. To protect against this attack in our scheme, the server should not know the real frequency of downloading documents, and this is done by downloading redundant documents (that do not have the highest similarity scores) by the doctors, and because of hiding the similarity scores of the documents in our scheme, the server cannot identify these redundant documents.

Preposition 6: EPSM ensures unlinkabilty of indices/trapdoors sent from the same patient/doctor or having the same keywords and search conditions.

Proof: The existing schemes suffer from linkability attacks in which the server can link the trapdoors (or indices) that have the same keywords. They try to thwart this attack by using random numbers in the encryption so that two trapdoors (or indices) having the same keywords look different. However, this is not enough because the server can link two trapdoors (or indices) by observing that they give the same similarity scores when they are matched to a set of indices (or trapdoors). EPSM ensures that the encrypted indices/trapdoors that have the same keywords or are sent from the same patients/doctors look different because of



(a) Index/trapdoor generation.

(b) Query matching.

FIGURE 4. Computation overhead.

using random numbers in splitting the vectors $V_{i,j}$ and $Q_{x,y}$. Moreover, our scheme also hides the similarity score from the server using the random numbers $a_{i,j}$ and $b_{x,y}$. Therefore, by computing the noisy similarity scores by a server for two trapdoors with the same keywords, the scores look different due to using different $b_{x,y}$ in the two trapdoors.

VI. PERFORMANCE EVALUATION

In this section, we compare the performance of EPSM with the existing schemes.

A. EXPERIMENT SETUP AND METRICS

1) EXPERIMENT SETUP

To evaluate the communication and computation overheads of EPSM, we have performed our experiments using python running on an Intel ® Core i7-8700 CPU @3.20GHz and 16 GB RAM. The computation and communication overheads of EPSM are compared to the proposed schemes in [12] and [16] after using them in a multi-data-owners setting. All the results presented in this section are averaged over 1000 trials for 2,000 documents, 10 patients, 10 doctors, and 2 bytes for each element in the ciphertext vector.

2) PERFORMANCE METRICS

Three performance metrics are used for the comparison and assessment of our scheme.

(1) *Computation overhead.* The time needed by patients/doctors to generate indices/trapdoors to be sent to the server. Also, the time needed to calculate the similarity score by the cloud server to search the documents.
(2) *Communication overhead.* The amount of data transmitted during the communication between the patients/doctors and the server.
(3) *Key management.* The number of a doctor's keys that is used to search all the documents of all patients.

B. EXPERIMENT RESULTS

1) COMPUTATION OVERHEAD

Fig. 4a gives the computation overhead of generating indices/trapdoors versus the number of keywords. The figure shows that the computation overhead increases as the number of keywords increases due to increasing the size of the matrices and vectors. In EPSM, because of supporting the multi-data-owners setting, each patient generates one index for each document and the doctor needs to generate only one trapdoor to search over the documents of all patients. Also, the same computation time is needed to generate the indices and the trapdoors because their vectors have the same size. It can also be seen from the figure that EPSM is more efficient compared to [12] and [16], because to use these schemes in a multi-data-owners setting, the doctor needs to calculate one trapdoor for each patient to be able to search their documents.

Fig. 4b gives the time needed to calculate the similarity score by the cloud server versus the number of keywords. As shown in the figure, the computation overhead needed to calculate the similarity score increases as the number of keywords increases because the vector size increases. The figure also shows that EPSM needs less time than [12] and [16]. Although EPSM increases by a higher rate because it needs eight dot product operations to support a multi-data-owner setting, the computation time is low (in ms) even with a high number of keywords (2000).

2) COMMUNICATION OVERHEAD

In EPSM, each patient sends an index ($I_{V_{i,j}}$) for each document. The overhead is $|I_{V_{i,j}}|$, where $|I_{V_{i,j}}|$ is the size of the index. If each element in the ciphertext is represented by 2 bytes, the ciphertext size in our scheme becomes $16(m + e + 2)$ bytes. Similarly, the trapdoor vector size is $16(m + e + 2)$ bytes. Fig. 5 gives the index/trapdoor

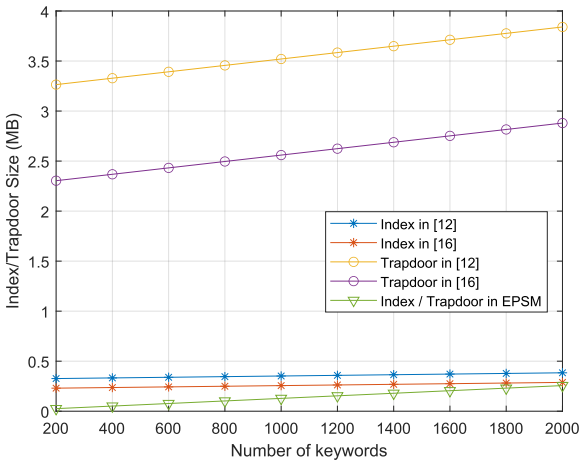


FIGURE 5. Communication overhead.

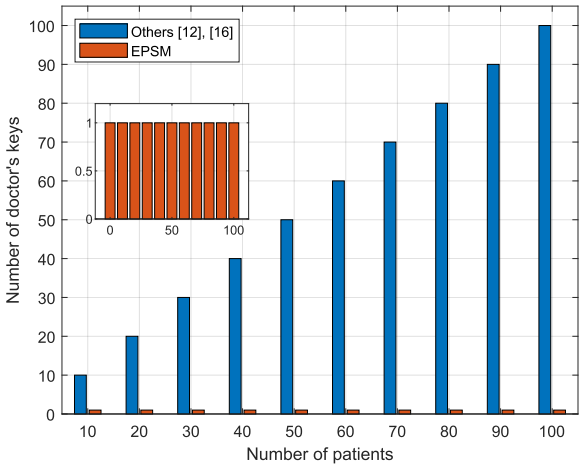


FIGURE 6. Number of doctor's keys versus the number of patients.

communication overhead versus the number of keywords. It can be seen that the communication overhead increases linearly with the number of keywords due to increasing the vector size. Moreover, the schemes [16] and [12] need more overhead compared to EPSM because they need to extend the vectors by the maximum possible number of documents before encrypting them, and the doctor needs to send multiple trapdoors that are equal to the number of patients to search their documents but in our scheme, only one trapdoor is sent to search the data of all patients.

3) KEY MANAGEMENT

Fig. 6 gives the number of a doctor's keys versus the number of patients. As shown in the figure, in EPSM, each doctor has only one key that is used to search all the documents of all patients. However, in [16] and [12], because the schemes are designed for a single-data-owner setting, each doctor needs to share a key with each patient. In most E-health applications, a doctor typically treats several patients, so multi-data-owner is a proper setting. The figure shows that the number of keys of a doctor increases linearly with the number of patients. Using many keys in the system makes key management inefficient.

VII. CONCLUSION

In this paper, we have proposed, EPSM, an efficient and secure search scheme over encrypted medical cloud data in a multi-data-owner setting. To secure EPSM, the could server cannot learn the similarity scores of indices and trapdoors, but it computes noisy scores and sends them to the doctor to de-noise them. Moreover, EPSM enables a new feature that allows doctors to customize their search results by expressing search conditions in the trapdoors. Our formal proof and security analysis demonstrates that EPSM can preserve patient privacy and is secure against *known plaintext* and *know background* models. Also, EPSM ensures the unlinkability of indices/trapdoors having the same keywords. Finally, our extensive experiments demonstrate that EPSM requires low computation and communication overheads and a small number of keys because it is designed for a multi-data-owner setting which is more suitable for medical applications. For future work, we will investigate denial of service (DoS) attacks against the centralized server. Specifically, we will try to replace the central server with a blockchain network. Also, we will investigate the use of machine learning technology to diagnose diseases in e-health systems.

ACKNOWLEDGMENT

The authors, therefore, acknowledge with thanks DSR for technical support.

REFERENCES

[1] J. C.-W. Lin, Y. Djenouri, and G. Srivastava, "Efficient closed high-utility pattern fusion model in large-scale databases," *Inf. Fusion*, vol. 76, pp. 122–132, Dec. 2021.

[2] M. M. Badr, M. M. Fouda, and A. S. T. Eldien, "A novel vision to mitigate pilot contamination in massive MIMO-based 5G networks," in *Proc. 11st Int. Conf. Comput. Eng. Syst. (ICCES)*, Dec. 2016, pp. 366–371.

[3] J. C.-W. Lin, G. Srivastava, Y. Zhang, Y. Djenouri, and M. Aloqaily, "Privacy-preserving multiobjective sanitization model in 6G IoT environments," *IEEE Internet Things J.*, vol. 8, no. 7, pp. 5340–5349, Apr. 2020.

[4] J. C.-W. Lin, Y. Djenouri, G. Srivastava, U. Yun, and P. Fournier-Viger, "A predictive GA-based model for closed high-utility itemset mining," *Appl. Soft Comput.*, vol. 108, Sep. 2021, Art. no. 107422.

[5] Y. Shao, J. C.-W. Lin, G. Srivastava, D. Guo, H. Zhang, H. Yi, and A. Jolfaei, "Multi-objective neural evolutionary algorithm for combinatorial optimization problems," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Sep. 2, 2021, doi: 10.1109/TNNLS.2021.3105937.

[6] A. M.-H. Kuo, "Opportunities and challenges of cloud computing to improve health care services," *J. Med. Internet Res.*, vol. 13, no. 3, p. e67, Sep. 2011.

[7] S. A. Alansari, M. M. Badr, M. Mahmoud, and W. Alasmary, "Efficient and privacy-preserving contact tracing system for COVID-19 using blockchain," in *Proc. IEEE Int. Conf. Commun. Workshops (ICC Workshops)*, Jun. 2021, pp. 1–6.

[8] P. D. Raeve. *The World of Cloud-Based Services: Storing Health Data in the Cloud*. Accessed: Aug. 2021. [Online]. Available: https://www.healtheuropa.eu/cloud-basedservices-storing-health-data-in-the-cloud/93053/

[9] *Healthcare Cloud Computing Market to Hit U.S.$ 40 Bn by 2026*. Accessed: Aug. 2021. [Online]. Available: https://www.globenewswire.com/news-release/2019/05/08/1819458/0/en/Healthcare-Cloud-Computing-Market-to-Hit-U.S.-40-Bn-by-2026.html

[10] *Record Compromised*. Accessed: Aug. 2021. [Online]. Available: https://dashboard.healthit.gov/quickstats/pages/breaches-protected-health-information.php
[11] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 1, pp. 222–233, Jan. 2014.
[12] H. Li, Y. Yang, Y. Dai, S. Yu, and Y. Xiang, "Achieving secure and efficient dynamic searchable symmetric encryption over medical cloud data," *IEEE Trans. Cloud Comput.*, vol. 8, no. 2, pp. 484–494, Apr. 2020.
[13] Y. Yang, X. Liu, R. H. Deng, and Y. Li, "Lightweight sharable and traceable secure mobile health system," *IEEE Trans. Dependable Secure Comput.*, vol. 17, no. 1, pp. 78–91, Jan. 2017.
[14] W. Zhang, S. Xiao, Y. Lin, T. Zhou, and S. Zhou, "Secure ranked multi-keyword search for multiple data owners in cloud computing," in *Proc. 44th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw.*, Jun. 2014, pp. 276–286.
[15] W. Zhang, Y. Lin, S. Xiao, J. Wu, and S. Zhou, "Privacy preserving ranked multi-keyword search for multiple data owners in cloud computing," *IEEE Trans. Comput.*, vol. 65, no. 5, pp. 1566–1577, May 2016.
[16] Z. Xia, X. Wang, X. Sun, and Q. Wang, "A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 2, pp. 340–352, Jan. 2015.
[17] W. Xiangyu, J. Ma, M. Yinbin, X. Liu, and Y. Ruikang, "Privacy-preserving diverse keyword search and online pre-diagnosis in cloud computing," *IEEE Trans. Services Comput.*, early access, Dec. 16, 2020, doi: 10.1109/TSC.2019.2959775.
[18] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proc. IEEE Symp. Secur. Privacy (S P)*, May 2000, pp. 44–55.
[19] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *Eurocrypt*, vol. 3027. Cham, Switzerland: Springer, May 2004, pp. 506–522.
[20] C. Wang, N. Cao, K. Ren, and W. Lou, "Enabling secure and efficient ranked keyword search over outsourced cloud data," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 8, pp. 1467–1479, Aug. 2011.
[21] C. Yang, W. Zhang, J. Xu, J. Xu, and N. Yu, "A fast privacy-preserving multi-keyword search scheme on cloud data," in *Proc. Int. Conf. Cloud Service Comput.*, Nov. 2012, pp. 104–110.
[22] W. Sun, B. Wang, N. Cao, M. Li, W. Lou, Y. T. Hou, and H. Li, "Privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking," in *Proc. 8th ACM SIGSAC Symp. Inf., Comput. Commun. Secur. (ASIA CCS)*, 2013, pp. 71–82.
[23] R. Li, Z. Xu, W. Kang, K. C. Yow, and C.-Z. Xu, "Efficient multi-keyword ranked query over encrypted data in cloud computing," *Future Gener. Comput. Syst.*, vol. 30, no. 1, pp. 179–190, Jan. 2014.
[24] Z. Fu, F. Huang, K. Ren, J. Weng, and C. Wang, "Privacy-preserving smart semantic search based on conceptual graphs over encrypted outsourced data," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 8, pp. 1874–1884, Aug. 2017.
[25] H. Li, Y. Yang, T. H. Luan, X. Liang, L. Zhou, and X. S. Shen, "Enabling fine-grained multi-keyword search supporting classified sub-dictionaries over encrypted cloud data," *IEEE Trans. Depend. Sec. Comput.*, vol. 13, no. 3, pp. 312–325, May/Jun. 2015.
[26] J. Yu, P. Lu, Y. Zhu, G. Xue, and M. Li, "Toward secure multikeyword top-k retrieval over encrypted cloud data," *IEEE Trans. Dependable Secure Comput.*, vol. 10, no. 4, pp. 239–250, Jul./Aug. 2013.
[27] W. K. Wong, D. W.-L. Cheung, B. Kao, and N. Mamoulis, "Secure KNN computation on encrypted databases," in *Proc. Int. Conf. Manage. Data*, 2009, pp. 139–152.
[28] Z. Fu, K. Ren, J. Shu, X. Sun, and F. Huang, "Enabling personalized search over encrypted outsourced data with efficiency improvement," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 9, pp. 2546–2559, Sep. 2016.
[29] S. Zerr, D. Olmedilla, W. Nejdl, and W. Siberski, "Zerber^{+R}: Top-k retrieval from a confidential index," in *Proc. 12nd Int. Conf. Extending Database Technol. Adv. Database Technol. (EDBT)*, 2009, pp. 439–449.
[30] J. Beel, B. Gipp, S. Langer, and C. Breitinger, "Paper recommender systems: A literature survey," *Int. J. Digit. Libraries*, vol. 17, no. 4, pp. 305–338, Nov. 2016.
[31] J. Zobel and A. Moffat, "Exploring the similarity space," in *ACM SIGIR Forum*, vol. 32, no. 1. New York, NY, USA: ACM, 1998, pp. 18–34.
[32] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: Improved definitions and efficient constructions," *J. Comput. Secur.*, vol. 19, no. 5, pp. 895–934, Jan. 2011.

SHERIF ABDELFATTAH received the B.S. and M.S. degrees in electronics and communications engineering from Arab Academy for Science, Technology and Maritime Transport (AASTMT), Alexandria, Egypt, in 2012 and 2016, respectively. He is currently pursuing the Ph.D. degree with the Department of Electrical & Computer Engineering, Tennessee Tech University, USA. From March 2018 to August 2019, he worked as a Researcher with the Polytechnic University of Turin, Italy. He is also a Graduate Research Assistant with the Department of Electrical & Computer Engineering, Tennessee Tech University. His research interests include machine learning, cryptography and network security, and privacy-preserving schemes for smart healthcare systems.

MOHAMED BAZA received the B.S. and M.S. degrees in electrical and computer engineering from Benha University, Egypt, in 2012 and 2017, respectively, and the Ph.D. degree in electrical and computer engineering from Tennessee Tech University, Cookeville, TN, USA, in December 2020. From August 2020 to May 2021, he worked as a Visiting Assistant Professor with the Department of Computer Science, Sam Houston State University, Huntsville, TX, USA. He is currently an Assistant Professor with the Department of Computer Science, College of Charleston, SC, USA. He also has more than two years of industry experience in information security at Apache-Khalda Petroleum Company, Egypt. He is the author of numerous papers published in major IEEE conferences and journals, such as IEEE Wireless Communications and Networking Conference (IEEE WCNC), IEEE International Conference on Communications (IEEE ICC), IEEE Vehicular Technology Conference (IEEE VTC), IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY (TVT), IEEE TRANSACTIONS ON NETWORK SCIENCE AND ENGINEERING, and IEEE SYSTEMS JOURNAL. He served as a Reviewer for several journals and conferences, such as IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, IEEE INTERNET OF THINGS JOURNAL, and the journal of *Peer-to-Peer Networking and Applications*. His research interests include blockchains, cyber-security, machine learning, smart-grid, and vehicular ad-hoc networks. He was also a recipient of the Best IEEE Paper Award in the International Conference on Smart Applications, Communications and Networking (SmartNets 2020).

MAHMOUD M. BADR received the B.S. and M.S. degrees in electrical engineering from Benha University, Cairo, Egypt, in 2013 and 2018, respectively. He is currently pursuing the Ph.D. degree with the Department of Electrical & Computer Engineering, Tennessee Tech University, TN, USA. He is holding the position of Lecturer Assistant at the Faculty of Engineering at Shoubra, Benha University. He is also a Graduate Research Assistant with the Department of Electrical & Computer Engineering, Tennessee Tech University. He has been selected as the Poster Winner in Tennessee Tech University's Annual Research and Creative Inquiry Day, in 2021. His research interests include machine learning, blockchain, cryptography, 5G networks, network security, and smart grids.

MOHAMED M. E. A. MAHMOUD (Senior Member, IEEE) received the Ph.D. degree from the University of Waterloo, in April 2011. From May 2011 to May 2012, he worked as a Postdoctoral Fellow with the Broadband Communications Research Group, University of Waterloo. From August 2012 to July 2013, he worked as a Visiting Scholar with the University of Waterloo, and a Postdoctoral Fellow at Ryerson University. He is currently an Associate Professor with the Department of Electrical and Computer Engineering, Tennessee Tech University, USA. He is the author of more than 120 papers published in leading IEEE conferences and journals. His research interests include security and privacy-preserving schemes for smart grid communication networks, blockchain, machine learning, cloud security, and e-health. He has received the NSERC-PDF Award. He won the Best Paper Award from several IEEE conferences. He serves as an Associate Editor for IEEE INTERNET OF THINGS Journal and *Peer-to-Peer Networking and Applications* journal (Springer).

GAUTAM SRIVASTAVA (Senior Member, IEEE) received the B.Sc. degree from Briar Cliff University, USA, in 2004, and the M.Sc. and Ph.D. degrees from the University of Victoria, Victoria, BC, Canada, in 2006 and 2012, respectively. He then taught for three years at the Department of Computer Science, University of Victoria, where he was regarded as one of the top undergraduate professors in the computer science course instruction at the university. In 2014, he joined a tenure-track position at Brandon University, Brandon, MB, Canada, where he is active in various professional and scholarly activities. He was promoted to the rank of Associate Professor, in January 2018. He is active in research in the field of data mining and big data. In eight years of academic career, he has published a total of 200 papers in high-impact conferences in many countries and high-status journals (SCI, SCIE) and has also delivered invited keynote guest lectures on big data, cloud computing, the Internet of Things, and cryptography at many International universities. His research is funded by federal grants from the Natural Sciences and Engineering Research Council of Canada (NSERC) and Mathematics of Information Technology and Complex Systems (MITACS).

FAWAZ ALSOLAMI received the M.A.Sc. degree in electrical and computer engineering from the University of Waterloo, Canada, in 2008, and the Ph.D. degree in computer science from KAUST, Thuwal, Saudi Arabia, in 2016. He joined the Computer Science Department, King Abdulaziz University, as an Assistant Professor of computer science. He has been the Chairperson of the Computer Science Department, King Abdulaziz University, since 2018. His research interests include artificial intelligence, machine learning and data mining, and combinatorial optimization.

ABDULLAH MARISH ALI received the B.Sc. degree in computer science from Al-Mustansiriya University, Baghdad, Iraq, the M.Sc. degree in computer science from King Abdulaziz University (KAU), Jeddah, Saudi Arabia, and the Ph.D. degree in computer science from Universiti Teknologi Malaysia (UTM), Malaysia. He is currently an Assistant Professor with the Department of Computer Science, Faculty of Computing and Information Technology, KAU. His research interests include cloud computing, software agents, data mining, information retrieval, machine learning, and cyber security.