A Novel Privacy Preserving Search Technique for Stego Data in Untrusted Cloud

Mohammad Saidur Rahman and Ibrahim Khalil and Xun Yi and Tao Gu
School of Science, RMIT University, Melbourne, VIC 3000, Australia
mohammadsaidur.rahman@rmit.edu.au, ibrahim.khalil@rmit.edu.au, xun.yi@rmit.edu.au, tao.gu@rmit.edu.au

Abstract

We propose the first privacy preserving search technique for stego health data in untrusted cloud in this paper. The Cloud computing is a popular technology to the healthcare providers for outsourcing health data due to flexibility and cost effectiveness. However, outsourcing health data to the cloud introduces serious privacy issues to the patient. For example, dishonest personnel of the cloud provider may disclose patient sensitive information to business organizations for some financial benefits. Using steganography, patient sensitive information is hidden within health data for privacy preservation. As a result, stego health data is generated. To the best of our knowledge, no method exists for searching a particular stego data without disclosing any information to the cloud. We propose a framework for privacy preserving search over stego health data. We systematically describe each component of the proposed framework. We conduct several experiments to evaluate the performance of the framework.

1. Introduction

The healthcare providers are producing a very large volume of data everyday. Storing and managing this large volume of data is challenging for the healthcare providers due to the cost of infrastructures and human resource. The cloud computing technology provisions remote data storage and management to its customers. The flexibility and economic savings offered by cloud computing are motivating healthcare providers to outsource their local complex data management system into the cloud. However, outsourced data to the cloud are prone to privacy breach [1]. For example, a dishonest personnel of the cloud provider may disclose patient sensitive information (e.g. patient's personal information) to business organizations for some financial benefits.

In order to protect data privacy and combat unsolicited accesses in the cloud, personal health records of patients may have to be encrypted by data owners before outsourcing to the commercial public cloud [2]. However, encrypting health records obsoletes the traditional healthcare research and data utilization service based on plaintext records. To make it more clear, encrypted health data do not allow normal operation on the data without decrypting them. Hence, general analysis tasks on encrypted health cannot performed by a user who has no credential. The *steganography* [3] is another way of provisioning privacy of patients' sensitive information [4, 5]. Using steganography, a patient's sensitive information can be hidden inside the patient's health records without loosing the usability of the health record [4]. From there, the traditional healthcare research and data utilization service based on plaintext records become possible [5]. It is possible to hide patient sensitive information using steganography within different types of health data such as image [6, 7], text [8, 9], biosignal [4, 10, 11], and DNA [12, 13].

The healthcare providers convert their healthcare documents into *stego documents* by hiding patient sensitive information within the healthcare documents. Throughout this paper, we refer the healthcare provider as a *data owner*. The data owner outsorce the collection of stego documents to the cloud for storing and data management. Therefore, the cloud provider does not get the patient sensitive information. Hence, the privacy is preserved.

The key challenge of outsourcing stego documents to the cloud is the searching of a particular stego document in the cloud without revealing any sensitive information to the cloud. For instance, a data consumer (e.g. a doctor) wants to retrieve a healthcare document of a particular patient from the cloud. In order to preserve the privacy of the patient, the search keywords should not be passed to the cloud. A trivial solution is downloading all the stego documents at data consumer's side and search for desired document. The trivial solution is clearly impractical, due to the huge amount of bandwidth and computing requirements. Moreover, aside from eliminating the local storage management, storing data into the cloud serves no purpose unless they can be easily searched and utilized. Thus, exploring privacy-preserving and effective search service over stego cloud data is of paramount importance.

Considering the potentially large number of on-demand data users and huge amount of outsourced data documents in the cloud, this problem is particularly challenging as it is extremely difficult to meet also the requirements of performance, system usability and scalability. Privacy preserving search over encrypted data in the cloud is a very active research [1, 14, 15]. Nevertheless, proposed privacy preserving search techniques over encrypted data in the literature can not be applied for privacy preserving search over stego data due to fundamental differences of stego and encrypted data. To the best of our knowledge, the privacy preserving search technique for stego data does not exist till date.

In this paper, for the first time, we define and solve the problem of search over stego document on untrusted cloud while preserving strict system-wise privacy in the cloud computing paradigm. We present a model for privacy preserving search over stego data in this paper. In our model, a data owner, who is a healthcare provider, generates a stego database at first. The stego database contains stego documents. Each stego document in the stego database has hidden information of the related patient. As our main focus is developing a privacy preserving search mechanism, we do not discuss stego data generation process. The work in [4] can be considered as an example of hiding patient sensitive data in health data. The data owner builds a secure search index for the stego database at second. The secure search index contains encrypted stego file names and hashes of corresponding hidden keywords. At last, the data owner sends both stego database and secure search index to the cloud. An authorized data consumer sends a *search query* as hash string to the cloud. The search query may contain single or multiple keywords. In case of multiple keywords, keywords are concatenated and a hash string is generated. The hash string is generated by the data consumer using a *backdoor*. The backdoor is a secret information that is used during the hash generation [16]. It allows the data consumer to generate the same hash string for the search keywords that was generated by the data owner during search index construction. The cloud tries to find the given hash string in the secure search index and retrieves the corresponding encrypted file name. No information is leaked to the cloud during the search process. We have few assumptions in this model. Firstly, we assume that data transfers from one party to another party are performed using a secure channel. The secure channel uses state-of-the-art cryptography techniques to ensure communication security. Hence, how data security is provided during communication is not in the scope of this paper. Secondly, the cloud that provisions storage as a service is untrusted. Finally, the data consumer is authorized to access stego data. Hence, the data owner shares secret credentials with the data consumer to retrieve secret data from stego data. The data owner shares secret credentials with the data consumer via a trusted third-party.

The key contributions of this research are highlighted as follows:

1. We propose the first model for privacy preserving search technique over stego data in the cloud.

2. We propose a framework for privacy preserving search over stego database. In this framework, we develop of a secure search index to facilitate the search of stego data without any disclosure of sensitive information.

3. We design a privacy preserving query mechanism in this framework that takes hash string as input instead of plaintext, and performs search in the cloud on secured search index.

4. We conduct an experimental study that demonstrates the effectiveness of the proposed approach.

This paper is organized as follows. We review related work in privacy preserving search techniques and steganography techniques in Section 2. We discuss our proposed model, threat model and design goals in Section 3. In Section 4, we present the proposed framework for privacy preserving search over stego database. In Section 5, the results of our conducted experiments are provided. We conclude our paper and provide future research directions in Section 6.

2. Related Work

In this section, we discuss some of the works closely related to our work. As there exists no work on privacy preserving search technique for stego data, we discuss only privacy preserving search technique for encrypted data. The objective of discussion is to provide readers the view of privacy preserving search techniques.

Traditional single keyword searchable encryption schemes [17, 18, 19, 20, 21, 22, 23, 24, 25, 26] usually build an encrypted searchable index such that its content is hidden to the server unless it is given appropriate

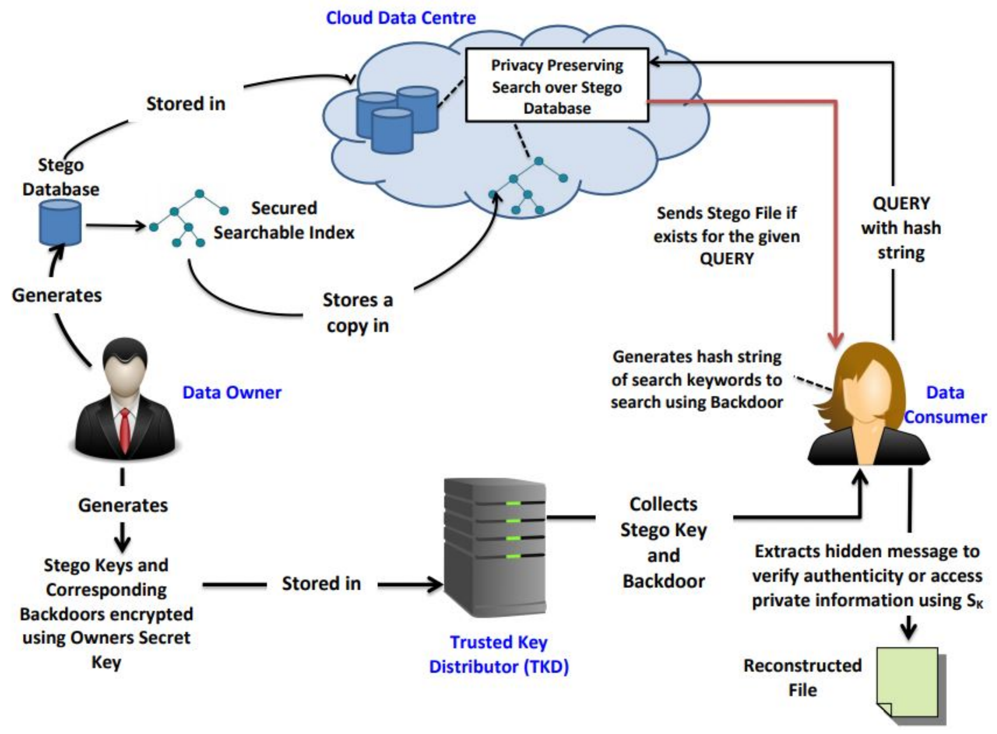
Cloud Data Centre
Privacy Preserving Search over Stego Database
Stored in
Stego Database
Secured Searchable Index
Generates
Stores a copy in
Sends Stego File if exists for the given QUERY
QUERY with hash string
Generates hash string of search keywords to search using Backdoor
Data Owner
Data Consumer
Generates
Stego Keys and Corresponding Backdoors encrypted using Owners Secret Key
Stored in
Collects Stego Key and Backdoor
Trusted Key Distributor (TKD)
Extracts hidden message to verify authenticity or access private information using S_K
Reconstructed File

Figure 1: System model of privacy preserving search over stego data in the untrusted cloud data centre

trapdoors generated via secret key(s) [2]. It is first studied by Song et al. [17] in the symmetric key setting, and improvements and advanced security definitions are given in Goh [18], Chang et al. [19] and Curtmola et al. [20]. The work [26] solves secure ranked keyword search which utilizes keyword frequency to rank results instead of returning undifferentiated results. However, it only supports single keyword search. In the public key setting, Boneh et al. [21] present the first searchable encryption construction, where anyone with public key can write to the data stored on server but only authorized users with private key can search. Public key solutions are usually very computationally expensive however. Furthermore, the keyword privacy could not be protected in the public key setting since server could encrypt any keyword with public key and then use the received trapdoor to evaluate this ciphertext.

3. Problem Formulation

3.1. System Model

We consider a cloud computing environment that hosts an outsourced *stego database*, based on which privacy preserving data sharing application can be built. Additionally, the system allows data consumers to verify the authenticity of data source. There are four participants in our system: *data owners*, *data consumers*, *trusted key distributor* (TKD) and *cloud data center* (CDC). We refer *data owner* as producer of the stego database. For instance, hospital or clinic can be data owner. A data owner collects health data from a patient and hides the patient sensitive information within the health data (D_i) for producing a stego health data (D_i). In the rest of this paper, we use *health stego data* and *health data* interchangeably. A set of secret information, called *stego key* (K_i) is used to produce S_i from D_i. Finally, the data owner produces a *stego database* (DB_S) of health stego data of n patients, and *secured searchable index* (SI). A set SK of stego keys for DB_S is generated as well. Here, SK can be defined as: $SK = \{SK_1, SK_2, \ldots, SK_n\}$. The data owner stores DB_S and a copy of SI in *cloud data center* (CDC). The set of stego keys SK is stored in a *trusted key distributor* (TKD). The CDC stores DB_S and SI, and performs search operation on stego data in SI for data consumers. The CDC receives a search query (Q) as a *hash string* and returns a stego health data S_i as response. The TKD is an entity that is trusted by all the participants in the system. In our system model, we assume that all the communications with TKD is secured. A *data consumer* wants to access a stego data for a specific patient, and can send a search request (*i.e.*, query) in a private manner over to CDC. The data consumer receives a stego health data S_i as response if exists in stego database. The overall system model is illustrated in Figure 1.

3.2. Threat Model

We assume that cloud data center (CDC) in our system model is "honest-but-curious" [27, 28]. In other words, a CDC satisfies following properties: (1) the CDC provides storage facility of stego data without modifying or destroying stored data, and (2) the CDC endeavours to acquire sensitive information of patients from the stored documents, data consumers' queries and search outcomes.

3.3. Design Goals

The system design of privacy preserving search over stego data in cloud data center should achieve the following main security and performance goals.

- **Index and Query Privacy**: The main security objective of this system is to prevent the CDC from learning any useful information about the patient related to stego data in stego database (DB_S), searchable index (SI) and data consumers' queries. However, the CDC can gain knowledge from the search result in order to make the search faster for future search operations. Index privacy refers to the confidentiality of SI, and query privacy refers to the protection of data consumers' queries.
- **Multi-attribute Keyword Search**: To design a search approach which allows a data consumer to insert multiple keywords. Each keyword indicates an attribute of a patient's sensitive information hidden within the patient's stego document. The search approach should be able to return the desired stego document based on the input attributes.
- **Scalability**: A new stego document should be added or an unused stego file should be removed without changing the security setup.
- **Efficiency**: Aforementioned goals should be obtained with low computation and communication overhead.

4. Proposed Framework for Privacy Preserving Search Over Stego Data Repository

We define the framework for privacy preserving search over stego data repository in this section. The framework consists of five tasks. We discuss the tasks in detail in rests of this section.

4.1. Construction of Stego Database

In this task, a data owner generates a stego database DB_S. The DB_S is a set of n number of stego data, *i.e.*, $DB_S = \{S_i | 1 \leq i \leq n\}$. A stego data S_i is generated by embedding SI-th patient's secret information M_i within the patient's medical data D_i. The embedding operation E_m takes D_i, M_i and SK_i as arguments and generates S_i. Here, SK_i is the *stego key*. The generation of stego data can be expressed as:

$$S_i = E_m(D_i, M_i, SK_i). \tag{1}$$

In this framework, the secret message M_i is a set of k attributes: $< m_{i1}, m_{i2}, \ldots, m_{ik} >$. An attribute m_{ik} denotes a sensitive data related to the SI-th patient and *searchable* by an authorized user. Giving examples, patient ID, name, date of birth, address, and contact number are patient's sensitive data. The M_i contains values of the aforementioned attributes for the SI-th patient. Figure 2 illustrates the stego data construction process. As stated in Section 1, the work in [4] can be considered as an example of hiding patient sensitive data in health data.

4.2. Building Secure Search Index

The data owner constructs a *secured search index* SI in this task. The search index SI is a matrix. A row of the matrix consists of an encrypted file name and hash of searchable attributes of a stego file. The file name is generated by encrypting a file's actual identifier with the data owner's secret key. The hash values are generated using an efficient hash function [16]. As a result, cloud can only see the encrypted file name and the hash values of searchable attributes.

The construction of SI involves the following steps:

1. ***Generating vectors of hash values*:** The data owner generates n number of vectors for n number of stego data in DB_S. A vector V_i contains hash values of all possible combinations of m_i. For k attributes in a secret message M_i, there are $2^k - 1$ possible combinations of attributes. Assume that, a secret message contains only two attributes: *patient ID* and *name*. The values of these two attributes for the first patient are: $P1001$ and $Alice$. The number of possible combinations are: $2^2 - 1 = 3$. The combinations are as follow:

$$< P1001 >, < Alice >, < P1001, Alice >$$

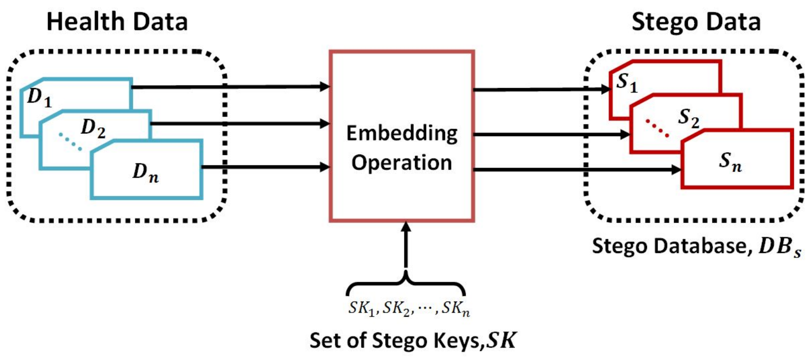

Figure 2: Illustration of the Stego Database DB_S construction process.

Next, *hash values* are generated for all combinations of m_i using an efficient *hash function* $\mathcal{H}$. The efficient *hash function* $\mathcal{H}$ can informally be defined as an irreversible and efficiently computable function which compresses a message of any length to a message of fixed length [16]. The generation of V_i from M_i using a secure hash function h is illustrated in Figure 3.

Definition 4.1 *Hash Function.* A *hash function* is a pair of efficient algorithms $\mathcal{H} = (KGen, \mathcal{F})$. The hash function $\mathcal{H}$ associates with *key space* $\mathcal{K}$, *message space* $\mathcal{M}$, and *hash space* $\mathcal{G}$, such that:

- $\delta \leftarrow KGen(\lambda)$: $KGen$ is an algorithm that generates and outputs a *hash key* $\delta \in \mathcal{K}$ on the input of a security parameter λ.
- $H \leftarrow \mathcal{F}(\delta, m)$: F is an algorithm that outputs a hash value $H \in \mathcal{G}$ on the input of a hash key $\delta \in \mathcal{K}$ and a message $m \in \mathcal{M}$.

The hash value of j-th combination of M_i can be represented as:

$$\delta \leftarrow KGen(\lambda),$$
$$H \leftarrow \mathcal{F}(\delta, c_{i,j}),$$

$$H_{i,j} = \mathcal{H}(KGen, \mathcal{F}), \tag{2}$$

where:

- $\mathcal{H}$ is the cryptographic hash function
- $c_{i,j}$ is the j-th combination of M_i
- $H_{i,j}$ is the generated hash value of $c_{i,j}$

Hash values are generated for all combinations of secret attributes using the *hash function* in (2). The vector V_i is denoted as:

$$V_i = \{H_{i,j} | 1 \leq j \leq 2^k - 1\}. \tag{3}$$

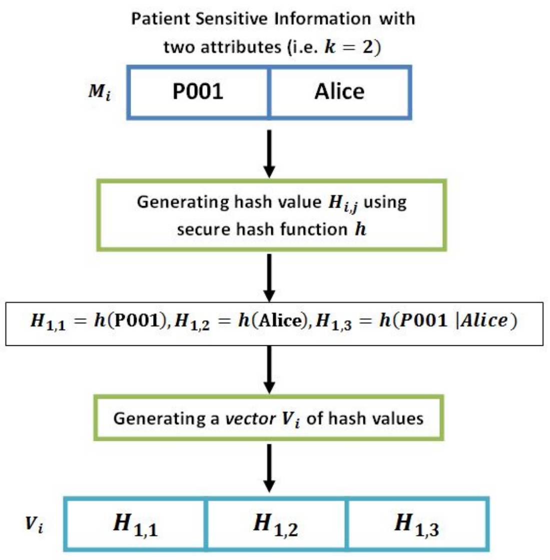
Patient Sensitive Information with two attributes (i.e. $k = 2$)
M_i
P001
Alice
Generating hash value $H_{i,j}$ using secure hash function h
$H_{1,1} = h(\mathrm{P001}), H_{1,2} = h(\mathrm{Alice}), H_{1,3} = h(P001 \mid Alice)$
Generating a vector V_i of hash values
V_i
$H_{1,1}$
$H_{1,2}$
$H_{1,3}$

Figure 3: An illustration of generating V_i from M_i using a secure hash function h

Hence, the vectors for all stego documents are presented as a matrix V as below:

$$V = \begin{bmatrix} V_1 \\ V_2 \\ \vdots \\ V_n \end{bmatrix} = \begin{bmatrix} H_{1,1} & H_{1,2} & \dots & H_{1,2^k-1} \\ H_{2,1} & H_{2,2} & \dots & H_{2,2^k-1} \\ \vdots & \vdots & \ddots & \vdots \\ H_{n,1} & H_{n,2} & \dots & H_{n,2^k-1} \end{bmatrix} \tag{4}$$

An illustration of generating V_i from M_i is provided in Fig. 3.

2. ***Secure naming of stego Data*:** The objective of this step is naming each stego file in a secure way. The secure naming makes stego data names

non modifiable. To make it clear, the name of a stego data is generated by the data owner using data owners secret key. Hence, only a data owner can generate the names of his/her stego data. The data owner can verify the names of outsourced stego data anytime by regenerating the name and matching. The data owner uses a secret key based encryption function $Enc()$ for generating the secure name of SI-th stego data in the stego database. The generation of the secure can be depicted as follows:

$$C_{P_l,i} = Enc(P_l, M_i, K_{P_l}), \tag{5}$$

where:

- P_l is the data owner's unique ID
- M_i is the secret message of SI-th stego data
- K_{P_l} is the the data owner's secret key
- $C_{P_l,i}$ is the secure name of SI-th stego data of the data owner.

All of the secure stego data names of the data owner having ID P_l are stored in a $n \times 1$ vector C stego documents are presented as a vector C as follows:

$$C = \begin{bmatrix} C_{P_l,1} \\ C_{P_l,2} \\ \vdots \\ C_{P_l,n} \end{bmatrix} \tag{6}$$

The SI-th stego document is named as $C_{P_l,i}$ in the stego database DB_S. In order to add a new stego document, the encrypted name of the stego document needs to be computed according to 5. The name should be appended in C. A row of the vector C is deleted and rows are re-organized for removing a stego document from the DB_S. Any stego document that has no reference in C is deleted from the DB_S at a later time. In this way, the scalability is achieved.

3. ***Construction of privacy preserving search index*:** A *privacy preserving search index* SI_{P_l} of the data owner having ID P_l is a $n \times 2^k$ matrix. The SI-th row of the matrix SI_{P_l} contains the secure name $C_{P_l,i}$ and vector of hash values V_i of the SI-th stego data. SI_{P_l} is represented as follows:

$$SI_{P_l} = \begin{bmatrix} C & V \end{bmatrix} = \begin{bmatrix} C_{P_l,1} & V_1 \\ C_{P_l,2} & V_2 \\ \vdots & \vdots \\ C_{P_l,n} & V_n \end{bmatrix}$$

$$\therefore SI_{P_l} = \begin{bmatrix} C_{P_l,1} & H_{1,1} & H_{1,2} & \dots & H_{1,2^k-1} \\ C_{P_l,2} & H_{2,1} & H_{2,2} & \dots & H_{2,2^k-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ C_{P_l,n} & H_{n,1} & H_{n,2} & \dots & H_{n,2^k-1} \end{bmatrix} \tag{7}$$

Change Management. If the value of any of the attributes in the stego document S_i is to be changed, the data owner regenerate S_i buy embedding new set of secret attributes within D_i original document. Additionally, the new encrypted file name $C_{P_l,i}$ for S_i is produced. Moreover, the hash vector V_i is generated using the same hash key δ. At the end, i-th row of SI is changed by the $[C_{P_l,i} \;\; V_i]$ in the copy of the cloud.

If any new stego document is to be added, then the data owner generate a new stego document. Let, the new stego document is denoted as S_{n+1} , where n is the number of current stego documents. Additionally, the new encrypted file name $C_{P_l,n+1}$ for S_{n+1} is produced. Moreover, the hash vector V_{n+1} is generated using the hash key δ. At the end, $(n+1)$-th row of SI is added with $[C_{P_l,n+1} \;\; V_{n+1}]$ in the copy of the cloud, and S_{n+1} is added in DB_S.

The i-row of SI of the copy of the cloud and S_i from DB_S is simply removed to delete S_i.

4.3. Construction of the Backdoored Hash Generation Function

A *backdoored hash function* is a function $\overline{\mathcal{H}}$ that takes a secret key as input and generates a hash value similar to the hash value generated by the hash function $\mathcal{H}$ [16]. The data owner builds the backdoored hash generation function $\overline{\mathcal{H}}$ associated with the hash function $\mathcal{H}$. The backdoored hash function is used to generate a hash of the search keywords at the end user's side.

Definition 4.2 *Backdoored Hash Generator.* The function $\overline{\mathcal{H}} = (BDHGen, \overline{\mathcal{F}})$ is called a backdoored hash generator function associated with key sapce $\mathcal{K}$, message space $\mathcal{M}$, and a hash space $\mathcal{G}$, such that:

- $\mathcal{B} \leftarrow BDHKGen(\lambda)$: $BDHKGen$ is called a backdoor generator function associated that generates a backdoor $\mathcal{B} \in \mathcal{K}$ on input of a security parameter λ.

- $\overline{\mathcal{H}} \leftarrow \overline{\mathcal{F}}(\mathcal{B}, m)$: $\overline{\mathcal{F}}$ is an algorithm that outputs a hash value $\overline{H} \in \mathcal{G}$ on the input of a backdoor $\mathcal{B} \in \mathcal{K}$ and a message $m \in \mathcal{M}$ such that $\overline{H} = H$. Here, H is the hash value generated by $\mathcal{H} = (KGen, \mathcal{F})$.

4.4. Privacy Preserving Query for Searching Stego Data

A privacy preserving query $PPQ(\overline{H})$ is an operation that is initiated by an end user and executed by the cloud. The query $PPQ(\overline{H})$ takes a hashvalue as an input and returns a stego data D_{out} as output. The $PPQ(\overline{H})$ has the following steps:

- **Step-1**: An end user generates a hash value $\overline{H}$ for a search keyword m using a backdoor $\mathcal{B}$. For k number of attributes, there can be maximum $2^k - 1$ combinations of search keywords. The set of all possible combinations A is denoted as: $A = \{a_1, a_2, \ldots, a_{2^k-1}\}$. Therefore, a search keyword m is an element of A (*i.e.*, $m \in A$).

- **Step-2**: The end user initiates a query $PPQ(\overline{H})$. The query is executed by the cloud. During the execution, the cloud matches the hash value $\overline{H}$ in the search index SI_{P_l}. Assume that the cloud finds an element $H_{i,j}(1 \leq i \leq n, 2 \leq j \leq 2^k)$ in SI_{P_l}, where SI_{P_l} is a $n \times 2^k$ matrix, and SI and j denotes the row and column index of $H_{i,j}$ in SI_{P_l}, respectively. Therefore, there exists a stego data for the search keyword.

- **Step-3**: The cloud considers $C_{P_l,i}$ as the searched stego document's name and sets $D_{out} = DB_S[C_{P_l,i}]$, where $DB_S[C_{P_l,i}]$ is the stego document with a name $C_{P_l,i}$.

- **Step-4**: The cloud sends D_{out} to the end user as output of the $PPT(\overline{H})$.

- **Step-5**: If the cloud does not find any match ($\overline{H} \notin SI_{P_l}$), then returns $NULL$.

4.5. Performance Analysis

We analyze the performance of our proposed framework in this section. We analyze the performance from the aspects of semantic privacy of both search index and query operation, functionality and efficiency, and scalibility.

Privacy of search index. The search index SI consists of two things: *encrypted file names* and *hashes of search keywords*. Traditional symmetric key encryption schemes, such as advanced encryption schem (AES) [29] and data encryption standard (DES) [30], can be used to produce encrypted file name. The security strength of the aforementioned symmetric key encryption is very high. In general, the security of a symmetric key encryption lies on the length of the secret key. We assume that a secured key would be used for the generation of the encrypted file names. The data owner generates all of the encrypted file names before sending the stego database and search index. Moreover, the data owner never shares the symmetric key with the cloud. Hence, it is provably impossible for the cloud to determine the relation between a encrypted file name and a stego file.

Privacy of query operation. We assume that the hash values are generated using a secured hash generation function such as SHA-256. In each row of SI, there are $2^k - 1$ hash values for k number of searchable keywords. Cloud has no clue about the keywords hidden inside a stego document. In the privacy preserving query operation, the data consumer generates a hash string of the query word(s). The hash value is generated using a backdoor $\mathcal{B} \subset \mathcal{K}$. The consumer collects $\mathcal{B}$ from the trusted key distributor TKD using a secure channel. The generated hash value is send as query to the cloud for searh execution. Plaintext search keyword is never send to the cloud. The cloud searches received keyword within the set of hash values V in SI. Hence, our proposed query operation is executed in a private manner. Overall, the privacy is preserved in our proposed framework.

Functionality and efficiency. Enabling multi-attribute keyword search is one of our design goals. In our model, we stated that each stego document has k hidden attributes and possible number of combinations is $2^k - 1$. The SI contains the hash of $2^k - 1$ combinations. Therefore, it is guaranteed that a valid search keyword, either single or multi-attribute, will always be found in our model. The build time of SI depends on the number of secret attribute k. Hence, the runtime complexity of building SI is $\mathcal{O}(k)$.

Scalibility. The scalability depends on the number of operations required for making any changes in the SI and DB_S. Our proposed framework requires 2^k number of operations in SI to change single or multiple secret attributes. Here, $2^k - 1$ operations are required for hash generation and single operation for updating encrypted name of the stego file. For adding a new stego file, the proposed framework requires equal number of operations in SI as of changing a secret attribute value. Therefore, the number of operations to make any change is always 2^k.

5. Experimental Results and Discussion

The key contribution of this paper is building a search mechanism for stego database that preserves the privacy during search. We conduct a set of experiments for evaluating performance of our proposed privacy preserving search technique for stego data in untrusted cloud. As we discuss the semantic privacy strength and scalibility of our proposed search technique in Section 4.5, we focus on the efficiency in our experiments. We consider three evaluation criteria in our experiments: (i) times required for generating stego database, (ii) times required for building search index, and (iii) times required for searching a stego document in the search index.

5.1. Experimental Setup

In order to setup the environment of our experiments, we build a stego database containing stego biomedical signals. We collect biosignal (e.g. ECG, PPG and EEG signal) dataset from *Physionet* repository. The Physionet repository is funded by the *National Health Institute (U.S.)* [31]. PhysioNet provides free access through their website to their large collections of recorded biomedical signals. We implement the method stated in [4] of hiding patient sensitive information within biomedical signal. We generate $5,000$ stego files for our stego database. Each file is 10 seconds long with 250 Hz frequency. In other words, there are 2500 records in a biosignal file. We randomly hide different number of secret attributes in sample biomedical signals for generating stego biomedical signals. We take different number of stego biomedical signals for conducting each experiment. For generating encrypted names of stego files, we use advanced encryption standard (AES) with 128-bit key. We use SHA-256 hash algorithm for generating the hash values in our experiment. We develop JAVA based programs to run our experiment. The JAVA Development Kit (JDK) 1.8 is used to write JAVA codes. We run our experiments on 3.40 GHZ Intel Core i7 processor and 8 GB RAM operated under Windows 7 operating System.

5.2. Evaluation

We evaluate the performance of our proposed framework in terms of *time cost* of different operations: *stego database generation*, *building search index* and *average search time*. In order to observe stego database generation time, we build databases of different sizes for different number of secret attributes ($k = 2, 3, 4, 5$). Figure 4 demonstrates the comparison of time for generating stego databases. The number of stego

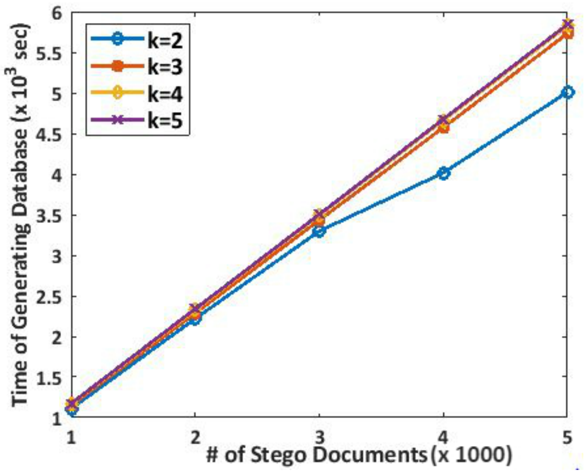

Figure 4: Comparison of times for generating stego database (DB_S) in kilo seconds for different number of stego documents in thousand.

documents are generated in thousands (i.e. $\times 1000$). Execution times are shown in kilo seconds. All of the files in our stego database are of equal size. Therefore, results show that times for generating stego databases increase linearly.

We build search indexes for different number of secret attributes ($k = 2, 3, 4, 5$) and database sizes. Figure 5 illustrates the comparison of time for building search indexes. The number of stego documents are generated in thousands (i.e. $\times 1000$). The results show that times for generating stego databases increase linearly. According to the results, time for building search indexes increase in a linear fashion.

Figure 6 illustrates the comparison of time for searching a stego document in different search indexs SIs. We search keywords for different number of secret attributes ($k = 2, 3, 4, 5$) and database sizes. The number of stego documents are generated in thousands (i.e. $\times 1000$). We execute the experiments 100 times for each search index and same number of keywords. We take the average time of the executions. The results show that times for searching a stego document in a search index is dominated by the size of the secret attributes in the stego database. As the number of secret attributes increases, the combination of search keyword increases. Hence, the search time increases. The number of elements in each subindex (i.e. each row of the search index) is fixed for a value of k, and the required time for building each subindex is fixed as well. Therefore, the search time increases almost linearly.

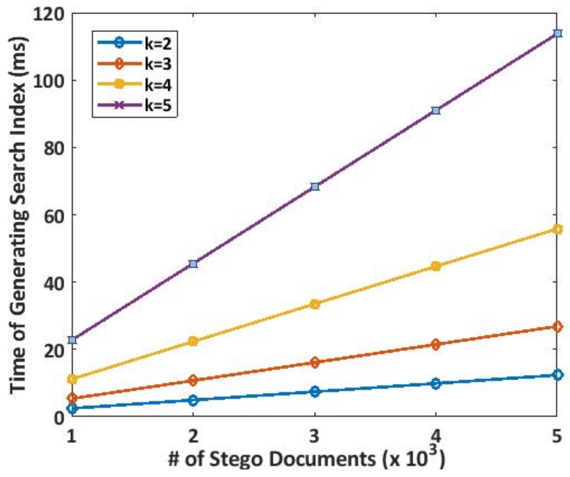

Figure 5: Comparison of times for building search index (SI) in milliseconds for different number of stego documents in thousand.

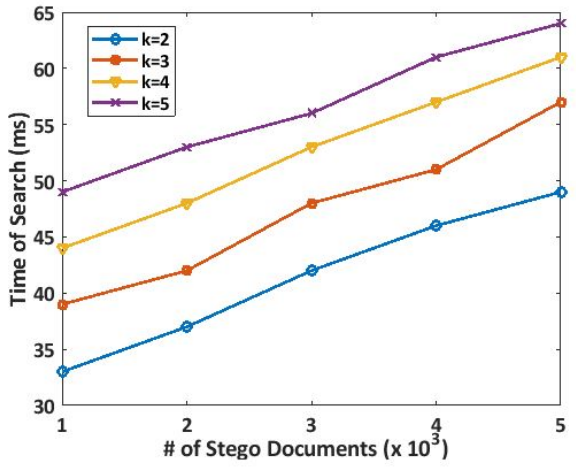

Figure 6: Comparison of times in milliseconds for searching a stego document in the search index SI for different number of stego documents in thousand.

5.3. Discussion

Our proposed framework is the first attempt of developing a privacy preserving search technique for stego data in the untrusted cloud data center. Hence, we cannot compare the performance of our work with any other work. However, our framework takes reasonable time for generating stego databases, search indexes, and query operation. We assume that the size of files are equal and number of search attributes are same for each file. Overall, the performance of the proposed framework mostly depends on the size of the stego database and number of secret attributes in a linear fashion. We consider only stego databases in our proposed search mechanism. Therefore, we do not show any performance comparison between our proposed search techniques on stego database and existing search techniques on encrypted database.

6. Conclusion

In this paper, for the first time, we propose a privacy preserving search framework for stego health data in untrusted cloud. We discuss a system model of our framework in this paper. Additionally, we briefly discuss a threat model and outline specific design goals. In the framework, we show how to construct stego database of health data by hiding patient sensitive information within health data. The key contribution of this framework is the construction of a secure search index for stego database. We build the search index by generating hash values of patient sensitive attributes. The hash function supports a backdoor that allows a data consumer to generate same hash value for some keywords to that is generated by the data owner. Our search index contains $2^k - 1$ hash values for each stego document. Therefore, the time complexity of building the search index is $\mathcal{O}(k)$ which is very efficient. We establish a private link between a stego document and a row of the search index by generating an encrypted file name. A secure symmetric key based encryption scheme, such as AES, is used. The encrypted file name is stored in the search index as well. A data consumer generates the hash string of query string at their side using the backdoor and sends the hash string as query string. Therefore, the cloud neither knows the query string nor the patient sensitive information. Moreover, it is not possible for the cloud to determine the relationship between search string and the stego file in the stego database. Our experimental results demonstrate the efficiency of our framework. The execution time linearly increases with the increment of stego database size and number of secret attributes of patient sensitive information.

The limitation of our proposed method is that it only supports multi-attribute search of exact query parameters. We do not consider ranged search query in this work. In our future work, we plan to develop a privacy preserving search mechanism that takes one or more range values to perform the search operations on stego database in untrusted cloud.

References

[1] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," *IEEE Transactions on parallel*

and distributed systems, vol. 25, no. 1, pp. 222–233, 2014.

[2] S. Kamara and K. Lauter, "Cryptographic cloud storage," in *International Conference on Financial Cryptography and Data Security*, pp. 136–149, Springer, 2010.

[3] I. Cox, M. Miller, J. Bloom, J. Fridrich, and T. Kalker, *Digital watermarking and steganography*. Morgan Kaufmann, 2007.

[4] A. Ibaida and I. Khalil, "Wavelet-based ecg steganography for protecting patient confidential information in point-of-care systems," *IEEE Transactions on Biomedical Engineering*, vol. 60, no. 12, pp. 3322–3330, 2013.

[5] A. Ibaida, I. Khalil, and D. Al-Shammary, "Embedding patients confidential data in ecg signal for healthcare information systems," in *2010 Annual International Conference of the IEEE Engineering in Medicine and Biology*, pp. 3891–3894, IEEE, 2010.

[6] K. Muhammad, J. Ahmad, N. U. Rehman, Z. Jan, and M. Sajjad, "Cisska-lsb: color image steganography using stego key-directed adaptive lsb substitution method," *Multimedia Tools and Applications*, vol. 76, no. 6, pp. 8597–8626, 2017.

[7] K. Muhammad, J. Ahmad, S. Rho, and S. W. Baik, "Image steganography for authenticity of visual contents in social networks," *Multimedia Tools and Applications*, vol. 76, no. 18, pp. 18985–19004, 2017.

[8] M. S. Rahman, I. Khalil, X. Yi, and H. Dong, "Highly imperceptible and reversible text steganography using invisible character based codeword," in *PACIS 2017: Twenty First Pacific Asia Conference on Information Systems*, pp. 1–13, AIS Electronic Library (AISeL), 2017.

[9] E. Satir and H. Isik, "A huffman compression based text steganography method," *Multimedia tools and applications*, vol. 70, no. 3, pp. 2085–2110, 2014.

[10] A. Abuadbba and I. Khalil, "Walsh–hadamard-based 3-d steganography for protecting sensitive information in point-of-care," *IEEE Transactions on Biomedical Engineering*, vol. 64, no. 9, pp. 2186–2195, 2017.

[11] A. Abuadbba and I. Khalil, "Wavelet based steganographic technique to protect household confidential information and seal the transmitted smart grid readings," *Information Systems*, vol. 53, pp. 224–236, 2015.

[12] A. Khalifa and A. Atito, "High-capacity dna-based steganography," in *Informatics and Systems (INFOS), 2012 8th International Conference on*, pp. BIO–76, IEEE, 2012.

[13] B. A. Mitras and A. K. Aboo, "Proposed steganography approach using dna properties," *International Journal of Information Technology and Business Management*, vol. 14, no. 1, pp. 96–102, 2013.

[14] 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 *Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security*, pp. 71–82, ACM, 2013.

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

[16] M. Fischlin, C. Janson, and S. Mazaheri, "Backdoored hash functions: Immunizing hmac and hkdf," in *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*, pp. 105–118, July 2018.

[17] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Security and Privacy, 2000. S&P 2000. Proceedings. 2000 IEEE Symposium on*, pp. 44–55, IEEE, 2000.

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

[19] Y.-C. Chang and M. Mitzenmacher, "Privacy preserving keyword searches on remote encrypted data," in *International Conference on Applied Cryptography and Network Security*, pp. 442–455, Springer, 2005.

[20] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: improved definitions and efficient constructions," *Journal of Computer Security*, vol. 19, no. 5, pp. 895–934, 2011.

[21] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *International conference on the theory and applications of cryptographic techniques*, pp. 506–522, Springer, 2004.

[22] M. Bellare, A. Boldyreva, and A. ONeill, "Deterministic and efficiently searchable encryption," in *Annual International Cryptology Conference*, pp. 535–552, Springer, 2007.

[23] M. Abdalla, M. Bellare, D. Catalano, E. Kiltz, T. Kohno, T. Lange, J. Malone-Lee, G. Neven, P. Paillier, and H. Shi, "Searchable encryption revisited: Consistency properties, relation to anonymous ibe, and extensions," *Journal of Cryptology*, vol. 21, no. 3, pp. 350–391, 2008.

[24] J. Li, Q. Wang, C. Wang, N. Cao, K. Ren, and W. Lou, "Fuzzy keyword search over encrypted data in cloud computing," in *INFOCOM, 2010 Proceedings IEEE*, pp. 1–5, IEEE, 2010.

[25] D. Boneh, E. Kushilevitz, R. Ostrovsky, and W. E. Skeith, "Public key encryption that allows pir queries," in *Annual International Cryptology Conference*, pp. 50–67, Springer, 2007.

[26] C. Wang, N. Cao, J. Li, K. Ren, and W. Lou, "Secure ranked keyword search over encrypted cloud data," in *Distributed Computing Systems (ICDCS), 2010 IEEE 30th International Conference on*, pp. 253–262, IEEE, 2010.

[27] J. Benaloh, M. Chase, E. Horvitz, and K. Lauter, "Patient controlled encryption: ensuring privacy of electronic medical records," in *Proceedings of the 2009 ACM workshop on Cloud computing security*, pp. 103–114, ACM, 2009.

[28] N. Cao, Z. Yang, C. Wang, K. Ren, and W. Lou, "Privacy-preserving query over encrypted graph-structured data in cloud computing," in *Distributed Computing Systems (ICDCS), 2011 31st International Conference on*, pp. 393–402, IEEE, 2011.

[29] F. P. Miller, A. F. Vandome, and J. McBrewster, "Advanced encryption standard," 2009.

[30] D. E. Standard, "Data encryption standard," *Federal Information Processing Standards Publication*, 1999.

[31] A. L. Goldberger, L. A. N. Amaral, L. Glass, J. M. Hausdorff, P. C. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng, and H. E. Stanley, "Physiobank, physiotoolkit, and physionet," *Circulation*, vol. 101, no. 23, pp. e215–e220, 2000.