A Major Project Report On

SECURE FILE ACCESS USING ATTRIBUTE BASED ENCRYPTION WITH KEYWORD SEARCH

Submitted in the partial fulfillment of the requirements

for the award of degree of

MASTER OF TECHNOLOGY

IN

SOFTWARE ENGINEERING

By

Parul Choudhary

(2K14/SWE/13)

Under the guidance of

Ms. Divyashikha Sethia

Department of Computer Science Engineering, DTU

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

DELHI TECHNOLOGICAL UNIVERSITY

BAWANA ROAD, DELHI

2014 - 2016

Computer Science & Engineering Department
Delhi Technological University
Delhi-110042
www.dtu.ac.in

DECLARATION

I hereby want to declare that the thesis entitled "**Secure File Access Using Attribute Based Encryption with Keyword Search"** which is being submitted to the **Delhi Technological University**, in partial fulfillment of the requirements for the award of degree in **Master of Technology in Software Technology** is an authentic work carried out by me. The material contained in this thesis has not been submitted to any institution or university for the award of any degree.

_____

Parul Choudhary

Department of Computer Engineering

Delhi Technological University,

Delhi.

Computer Science & Engineering Department
Delhi Technological University
Delhi-110042
www.dtu.ac.in

ACKNOWLEDGEMENT

I feel immense pleasure to express my heartfelt gratitude to Ms. Divyashikha Sethia for her constant and consistent inspiring guidance and utmost co-operation at every stage which culminated in successful completion of my project work.

I am also grateful to Dr. O.P. Verma, Head of Department for his continuous support and directions. I would like to thank entire teaching and non-teaching staff in the Department of Computer Engineering, DTU for all their help during my tenure at DTU and also my peers for their kind advice and help from time to time.

I owe my profound gratitude to my family which has been a constant source of inspiration and support.

Parul Choudhary
M.Tech Software Engineering
2K14/SWE/13

Table of Contents

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Patient Health Records used in management of health records contains sensitive information related to every patient. These health records are retained by the patient on his mobile device and could be outsourced to servers for management or processing and these servers are generally un-trusted. This PHR contains sensitive and important health related information, so we need to ensure security and privacy of our health data. Along with confidentiality, we want the patient to have control over his health data, so we first encrypt the PHR with Attribute Based Encryption and then outsource to the server for storage and also if any search operation is required. PHR's contains different sections and in order to give selective access based on the content of the data and who is going to use it, the encryption is sectional using CPABE, which is a type of Attribute Based Encryption. CPABE fulfills the requirement of providing fine-grained access control. Different Access Policies could be used to encrypt sections of the Patient Health Record.

To reduce the overhead of decrypting whole PHR and then searching the records containing some keywords, we want search mechanism on the encrypted data. So, based on the keywords of the sections of PHR, we construct our search mechanism. Searchable Symmetric Encryption is use to achieve this searchability on encrypted data. Conjunctive keyword search query is also incorporated to provide better search query options to the medical professional.

In the improved version, patient also creates a hash table on the unique keywords contained in the PHR and outsource this also to the server. Authorized users whose attributes satisfy the access policy of different sections can only decrypt the corresponding the sections.

CHAPTER 1 - INTRODUCTION

1.1 Overview

Patient Health Record can be defined as patient's health data repository which would help the patient to maintain his past medical history by keeping a track record of medications, lab records, diagnoses and values of the important vitals. It helps in easy exchange of data between medical professionals and patient. PHR's can also be used to store information like, health insurance information and emergency information that can be used at the time of emergency situations. Keeping all these health related data in the plaintext format, incurs a threat to the security and privacy of our information.

To ensure confidentiality of our health data, we need to encrypt our health data. Various types of encryption models are present, like symmetric key encryption, public key encryption. In Symmetric key encryption, there is only single key present, and the owner of the document who has encrypted the document can only decrypt it. So, this is not suitable in our scenario, as we need to give access to medical professionals also. The other type of cryptographic model is public key encryption, in which pair of key is used. In this, the public key is known to everyone, but the private key is known to owner of the key only. The public key is used to encrypt and private key is used to decrypt. Since we want to give selective access to the medical professionals based on the content of the data, and also keeping in view that who can access that data, therefore we need Attribute based encryption model which fulfills our need of fine-grained access control. There are two types of Attribute based encryption, one is Key-Policy Attribute Based Encryption [3] and another is Ciphertext-Policy Attribute Based Encryption [4]. In KP-ABE the attributes are associated with the ciphertext and access policy is associated with the key. A user can decrypt only if the attributes of ciphertext satify the access policy associated with his/her key. Whereas, in CP-ABE, the attributes of user are associated with his key and access policy is associated with the ciphertext. If the attributes in user key satisfy the access policy of the ciphertext, then only the decryption is possible. CPABE can provide role-based access control.

The Patient Health Record contains many visits, allergies section, vaccination section, lab tests. Among these, if a medical professional wants to search for all records containing the vital "white blood cells", then it easy to do if the health record is in plaintext form. But since, for security purposes we have encrypted whole of out data, so searching on encrypted data is another problem. The naïve solution will be to decrypt whole PHR and then search for that keyword. This would mean, more search time, as decryption is costly in case of CPABE where pairings are involved.

Therefore, we need to search in the encrypted PHR itself, and then do the decryption on the retrieved records only. There are many searchable encryption techniques proposed in the past like, Symmetric Searchable Encryption [5,6,7,8], Public Key based searchable encryption[9], based on Predicate Encryption[11] and based on Homomorphic encryption[12]. Searchable symmetric encryption (SSE) allows the owner of the data to outsource his/her data which is present in encrypted form to server, at the same time having the capability to make keyword search over it [8]. Single user setting [reference] and multi user setting[reference] both have been proposed in the past. In our system, we need single user setting, as we want that patient can only search. The medical professional could submit query keywords to the patient, and patient then would generate trapdoors and would submit to the server for searching on encrypted index.

In Public key based searchable encryption [9], there are two keys present and anyone can encrypt the data using the public key of the recipient but search can be performed by owner of the document.

Using the symmetric searchable encryption, we have given search capability to the medical professionals and patient. Using Song's technique [5], we encrypt the keyword file which contains the keywords of every visits, records and section. This file would be used to search for the keywords. If the match is successful, then those records are retrieved and sent to the medical professional for decryption. Now, if the records access policy is satisfied by the medical professionals attributes, then only the decryption would be done otherwise not.

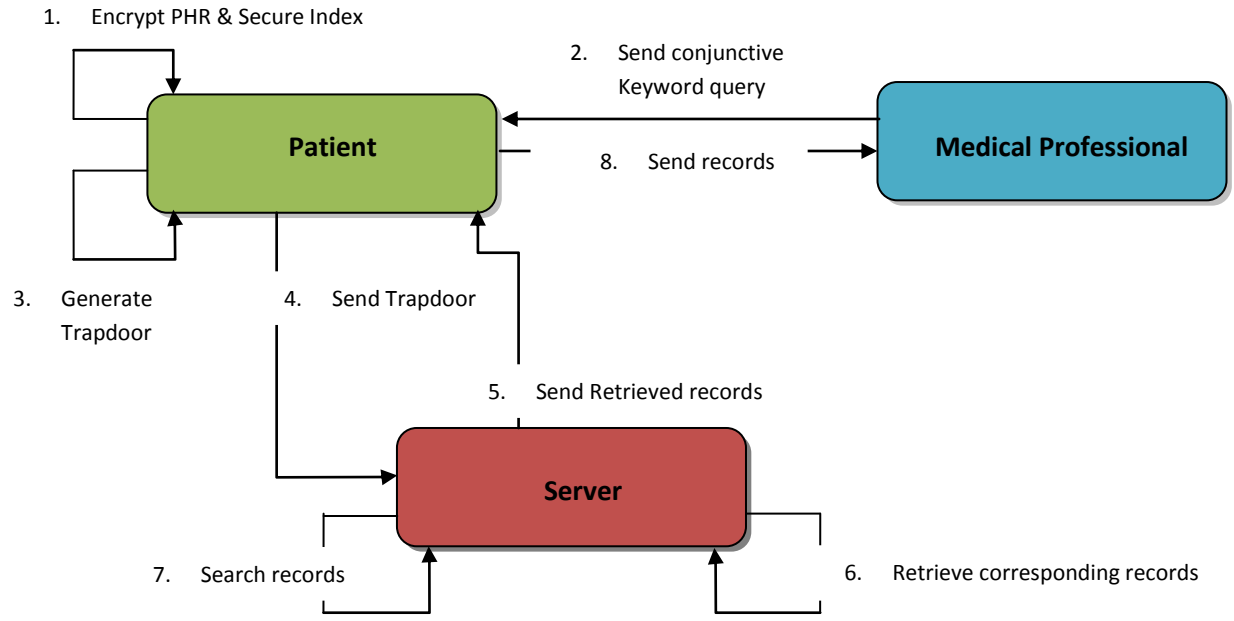


Figure 1: Flow Diagram

1.2 Motivation

PHR contains all the health related information about the patient which eases the exchange of health data of patients among the different stakeholders. PHR of patient contains sensitive health information, so security and privacy of health data is important. To achieve confidentiality, we need to encrypt Patient Health Record. For storage and for processing PHR, we keep this encrypted PHR, on servers.

Sometimes, medical professionals may need to look at the visits, tests, allergies etc. of the past, with some particular keywords. For example, all visits where "Allegra" was prescribed as the medicine or to determine whether the patient was given a particular vaccine or not. So, instead of following the naïve approach where all records and sections of PHR are decrypted and then searched for these particular keywords, we will search in the encrypted PHR itself. This will save time of unnecessary decrypting the other visits.

The medical professional may also want to search for records containing combination of keywords, for example, all records containing "Allegra or Paracetamol". This type of conjunctive keyword search would provide flexibility in search query to the medical professionals.

For confidentiality, we use Attribute based encryption to support fine grained access control. The earlier schemes encrypted the whole documents with same access policy, but in our health record we want to encrypt the different sections of the health record with the different access policy. Keyword search on encrypted data is done using SSE. The earlier SSE schemes used to encrypt the whole document which resulted in search time linear to number of words per document. We improve this by applying SSE on the keywords per section.

From the above discussion, it is evident that we need a system which encrypts the health record of patient and could perform the search. The searching is further improved by applying indexes on the keywords.

1.3 Problem Statement

Since PHR contains important health related data, so confidentiality of the data needs to be addressed. Also, the access to the PHR's section should be based on the content of the data and based on who can access that content, therefore encryption model based on selective access control should be chosen. PHR of patient contains many visits, lab tests, medications and for each of them the access policy may be different. So, sectional encryption is also proposed.

Searching on encrypted data is important since we have our encrypted data on cloud and we do not want to decrypt all records of PHR unnecessarily, if we want records which contain a particular keyword. We will decrypt only those records which contains those keywords.
A application is needed for searching on encrypted records, and looking at the retrieved records after search and display of important vitals graphically along with comparison of search times.

"*Development of application to encrypted PHR with fine grained access control and conjunctive-keyword search query and incorporating hashing on unique keyword to further improve the search time*"

1.4 Scope of Work

In this work, we have considered the confidentiality Patient Health Record and seachability on encrypted PHR. For ensuring confidentiality, we have used Attribute Based Encryption which will also helps to ensure selective access control. Unlike previous researches where the ABE is applied on the whole document in one go, in our system, we encrypt all visits and different sections (allergies, vaccination, medication) separately. So, we can different access policy for the sub-sections of a single PHR. This helps us to ensure role based access control on the sub-sections.

The medical professional only wants to retrieve the relevant documents containing some specific keywords. So, searching the keywords in encrypted keyword file is done, and only records containing those keywords are decrypted instead of whole PHR. The keyword file contains keywords corresponding to the sections and sub-sections of the PHR. Conjunctive keyword search

is also incorporated to provide more flexibility to medical professionals in the querying the PHR for specific records.

The modified version of this work, improves the search time in case of keywords specific to particular departments or some allergies or vaccination. Since in these cases, search is restricted to records of those departments or those respective sections. Search times are compared with the change in number of keywords in the keyword file and with the number of records in the PHR. Search time of the two versions are also compared.

1.5 Thesis Organization

The remaining chapters of the thesis are organized as follows:

Chapter 2 presents the literature survey of the various techniques for access control and searchable encryption. Brief overview and analysis is provided for the past techniques.

Chapter 3 presents the detailed insight over the techniques used in the proposed work, i.e., the attribute based encryption for access control and symmetric searchable encryption for giving the capability to search on ciphertexts.

Chapter 4 presents the proposed system for providing secure access in PHR with keyword search capability.

Chapter 5 gives the design and implementation details of the proposed work.

Chapter 6 presents the results regarding the search time of keywords in the two versions.

Chapter 7 concludes the thesis and presents the future work that could be done.

CHAPTER 2 – LITERATURE SURVEY

In this chapter we present a literature survey on existing techniques of access control and searchability on encrypted data. Overview is provided for each technique, with the algorithms used.

2.1 Symmetric Encryption Based Schemes

Among the various types of searchable techniques, symmetric searchable technique has only one key. Therefore, the one who is the encryptor of the data, only he has the capability to search on the encrypted data. No one else can search on the encrypted data.

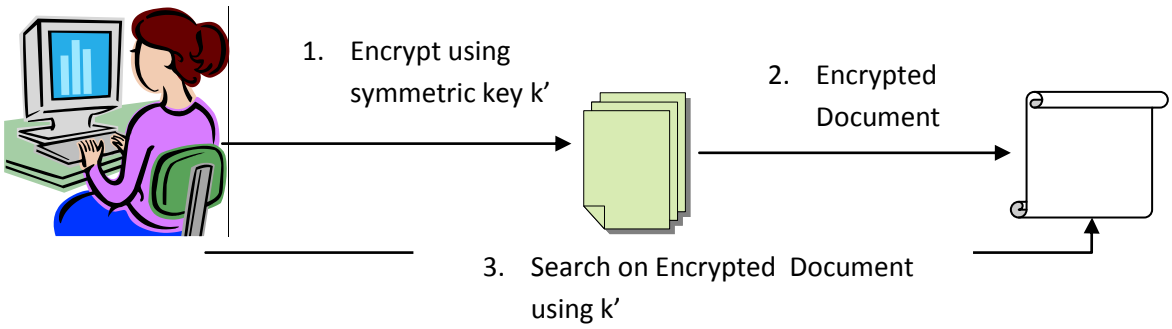


Figure 2: General Symmetric Searchable Scheme [5]

2.1.1 Song et al. [5] proposed the first practical scheme for searching on encrypted data. This technique uses a two-layered encryption construct that allows searching on the ciphertexts with a sequential scan.

2.1.1.1 Song et al. scheme's contain two main operations:

a. Encrypt each word separately and adds a hash value

b. During searching, the server extracts this hash value and check if the value is of that form which will decide if there is a match or not.

2.1.1.2 Analysis: The complexity of the encryption and search algorithms in this technique is linear to the total number of words per document (in worst case). During encryption phase one encryption, one XOR and two PRF's are performed per word. In searching phase, one XOR and one PRF is performed per word per document.

2.1.2 Goh [6] proposed a scheme Z-IDX which adds an index for each document and it is independent of the underlying encryption algorithm of the data. Goh used Bloom filter [13] as a

per-document index.

2.1.2.1 Bloom Filter: A BF is used to getting set membership queries. An array of m bits is used to represent a bloom filter which is all initially set to 0. In general, the filter uses x independent hash functions, each hash function maps a set element to one of the m array positions. For each element e (e.g., keywords) in the set S = {k_1 , . . . k_p }, the bits at positions $h_1(k_i)$, . . . , $h_r(k_i)$ are set to 1. In order to check if an element y(keyword) belongs to the set S, check if any of the bits at positions h1 (y), . . . , hx (y) are set to 1. If yes, then y is considered a member of set S.

2.1.2.2 This scheme proposed by Goh consists of following four algorithms:

a.) Keygen(s) : Given a input security parameter s, the output is a master private key Kpriv.

b.) Trapdoor (Kpriv, w) : Input is master private key and word w to be searched, it outputs a trapdoor for Tw for w.

c.) BuildIndex (D, Kpriv) : The inputs being the document and master private key, output is the index Id.

d.) SearchIndex (Tw, Id) : The inputs being the trapdoor and Index, it outputs 0 if w does not belong to D and 1 otherwise.

Disadvantage: The disadvantage of using Bloom filters is the possibility of false positives. By setting some parameters, the false-positive probability can be reduced.

2.1.2.3 Analysis: The index generation has to generate one BF per document. Thus, the algorithm is linear in the number of distinct words per document. The BF lookup is a constant time operation and has to be done per document. Thus, the time for a search is proportional to the number of documents. The size of the document index is proportional to the number of distinct words in the document.

2.1.3 Chang and Mitzanmacher [7] developed two index schemes, C-I and C-II.

2.1.3.1 In this scheme, following steps are taken

a.) prebuilt dictionary consisting of search keywords are used, so as to make an index per document.

b.) A n-bit array is used to make an index which are all initially set to 0, in which each bit position represents a keyword in the dictionary.

c.) If the document contains a keyword, its index bit is set to 1.

Both the schemes assume that the user is mobile with limited storage space and bandwidth.

Pseudo-random permutations and pseudo-random functions are used. First scheme stores the dictionary at the client and second encrypted at the server.

2.1.3.2. Analysis: The index generation is linear in the number of distinct words per document. The time for a search is proportional to the total number of documents.

2.1.4 In 2006, Curtmola et al. [8] proposed two new constructions (SSE-I, SSE-II).The unique contribution of this scheme is to add an inverted index, where an index per distinct word in the database is created instead of per document. This technique helps in reducing the search time to the number of documents that contain the keyword. This is not only sublinear but also optimal.

2.1.4.1. The index consists of

(a) an array A made of a linked list L per distinct keyword

(b) a look-up table T to identify the first node in A

Trapdoor generation : The trapdoor helps to identify and decrypt the correct node in T. Given the position and the correct decryption key for the first node, the server is able to find and decrypt all relevant nodes to obtain the document's identifiers.

2.1.4.2. Analysis : This scheme propose the first sublinear scheme. The index generation is linear in the number of distinct words per document. The computation performed by the server per search is proportional to the number of documents that contain a word 'w'.
In the second scheme proposed the search is proportional to maximum number of documents that contain a word w.

2.1.5 In 2010, Chase and Kamara [14] proposed a construction that is based on Curtmola's SSE. In this scheme, an inverted index is created in the form of a padded and permuted dictionary. Hash tables are used to implement dictionary which helps in getting optimal search time.

2.1.5.1 Analysis :Index generation requires two pseudo-random functions per distinct keyword in the database. During search, the position of the desired query keyword is searched and then decrypts the stored values, which are the document IDs of the documents which contains the keyword.

2.1.6 In 2012, Kamara et al. [15] (KPR) proposed a scheme which is further improvement of

Curtmola's Scheme [8], which allows efficient updates for eg., add, delete, and modify documents. This scheme uses only pseudo-random functions and XOR's. In this deletion arrays are added to keep information of search array positions that may be modified in case a update is made.

2.1.6.1 Analysis: This scheme efficiently handles the update and have optimal search time. Eight PRFs per keyword are computed in index generation phase and in search phase table look up(TLU) and one XOR operation for decryption of nodes.

2.2 Asymmetric Searchable Encryption

The asymmetric searchable technique has uses two keys unlike the single key in case of symmetric searchable encryption. In this scheme anyone can encrypt the data, but search is performed by owner of the document only.

2.2.1 Boneh et al. [9] proposed the first searchable encryption scheme using a public key system. Boneh in their PEKS scheme used identity-based encryption (IBE), in which the keyword acts as the identity. Each user encrypts the data with the recipient's public key and this content could be searched over. The recipient having the private key would be able to generate a trapdoor over a keyword to search inside the encrypted data. PEKS scheme is based on IBE [16, 17].

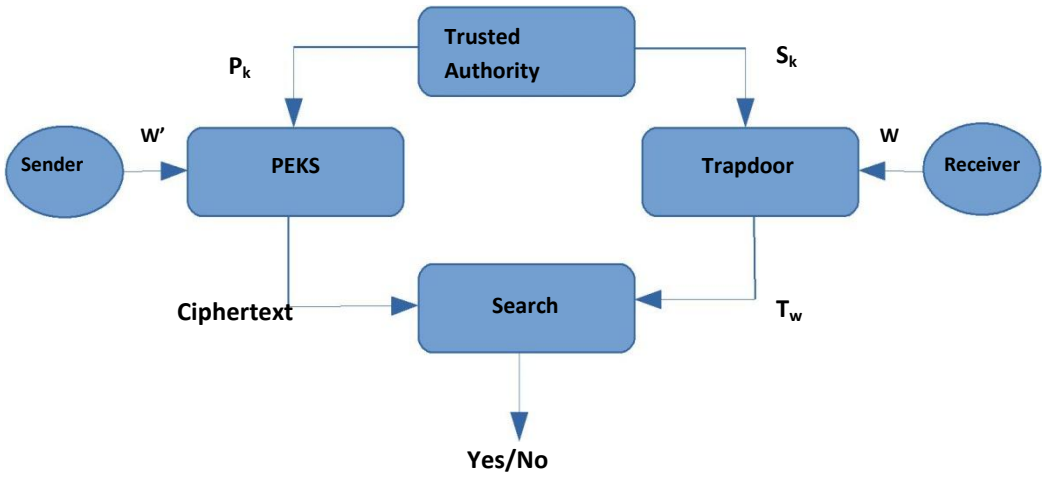


Figure 3: General asymmetric searchable encryption scheme [9]

To create a searchable ciphertext, the sender encrypts his/her message with a public key system and appends the PEKS of each keyword. The sender then sends the following ciphertext :

$$E_{Kpublic}(M)||C_1 = PEKS(K_{public}, w\ 1)|| \ldots ||C_n = PEKS(K_{public}, w_n)$$

In order to search, the receiver uses the master secret key to derive a secret key for a specific keyword it wants to search for (i.e., the keyword is the identity used for the secret key). The resulting secret key is used as the trapdoor. The server will try to decrypt all the ciphertexts. If the decryption is successful, that means the encrypted message contains the keyword which is been searched for.

The scheme uses symmetric prime order pairings. The encryption requires one pairing computation, two exponentiations, and two hashes per keyword. The search complexity is linear in the number of keywords per document.

This asymmetric searchable encryption technique is suitable to situations where the part searching over the data is different from part that is generating the data.

2.3 Data Access Control and Searchable Encryption

A relatively new cryptographic system was introduced, with respect to data access control, which is called Identity Based Encryption [16, 17]. With the use of IBE, users can communicate securely, without the exchange of keys, and having the need of maintaining the key directories.

2.3.1 After that, ABE which is a type of IBE was first proposed by Sahai and Water [1]. They proposed access control mechanism which is based on descriptive attributes. These attributes are used to describe encrypted data and user key. In ABE scheme, identity is set of attributes and if the receivers attributes satisfy the attributes in the encrypted message then only decryption is possible.

2.3.2 Goyal et al [3] formulated two types of ABE and proposed KP-ABE. They gave a more elaborative picture of the access control mechanism and in this, attributes are used to describe encrypted data and access policy is embedded in user's key which will define the data to which user is given access. In KPABE scheme, the access structure is provided in users decryption key whereas the encrypted data have the set of descriptive attributes. Each user's decryption key consists of a access structure in which the leaves represent the encrypted data's descriptive attributes. A data user can decrypt the encrypted data if the descriptive attributes in the encrypted data fits in the access policy which is defined in his decryption key.

2.3.3 Brethencourt et al. [4] proposed the first CP-ABE scheme. All the ABE schemes use the

access control trees as the access policy. This is in contrast to the KPABE scheme described by Goyal et al. [3] where the attributes describe the encrypted data and access policy is provided in the user's decryption key. In CP-ABE encryption model, the attributes are used to describe user's credentials, which describe the user's decryption key, and access policy defines who all can decrypt the data. CP-ABE provides role based access control.

2.3.4 Yang [18] proposed searchable encryption with synonym keyword search function. It uses ABE[1] for fine-grained access control. Secure index is created on keywords during encryption phase. It supports semantic keyword search. It consists of six polynomial-time algorithms –

- *Setup :* It takes as input a security parameter and outputs a global parameter and master secret key.
- *KeyGen :* It takes global parameter, user's identity and attribute set of user as input and outputs public and private key of user.
- *Encrypt :* Data owner runs the encryption algorithm. This algorithm takes as input the global parameter, Data to be encrypted, public key, keyword w and access structure. Synonym set will be generated for the keyword w. Finally the ciphertext and secure index is generated as output.
- *Trapdoor :* Taking global parameter, keyword w and private key of user as inputs, this algorithm generates trapdoor.
- *Retrieve :* Server runs this algorithm and takes global parameters, trapdoor and attribute set of user and access structure and checks if attribute set satisfies the access structure and the trapdoor matches the secure index or not.
- *Decrypt :* Data user runs this algorithm and decrypts all those documents retrieved in the above step.

2.3.5 Narayan et al. [19] proposed secure management of Electronic Health Records (EHR) and capability to search on encrypted health data. It uses broadcast CPABE which is a slight variation of ABE having a added functionality of user revocation. The health data is encrypted using symmetric key cryptography and ABE is used for making the symmetric keys accessible to authorized users. A separate key is used by the medical professionals for searching on the encrypted data. For searchability, a combination of b-ABE and PEKS is used. Each keyword is encrypted using PEKS during uploading of document. Medical professional with his decryption key generates trapdoor and then it is searched in the search index.

2.3.6 Ramu and Reddy [20] have used Searchable Symmetric Encryption and ABE for role based access. The proposed scheme has three phases – mainly setup phase, secure index phase which generates secure index. In the second phase a hash table is created containing each keywords trapdoor and a pointer pointing to linked list having the list of documents containing that keyword. The third phase is the search phase, where a trapdoor is generated with the symmetric key and then it is searched in the hash table and corresponding list of documents is retrieved.

The search time depends on the number of unique keywords in the document set.

2.3.7 Wang et al. proposed KSF-CP-ABE [10] which is a integration of PEKS and CP-ABE. In this a data owner who wants to outsource his sensitive data in the public cloud, first encrypts the sensitive data under an access policy and builds a corresponding secure index for keywords. Only authorized users whose credentials satisfy the access policy can retrieve this encrypted data through keyword search and decrypt the ciphertext. KSF-CP-ABE construction is based on bilinear pairings, which can ensure the security with fine-grained access control on shared sensitive data, and provide keyword search service for data users without leaking their privacy of queries and breaking confidentiality of data contents. The scheme uses five polynomial-time algorithms – Setup, ABE-Keygen, KSF-Keygen, Encrypt, Index and Test. In this scheme, for each keyword search three bilinear pairing operations are performed and one bilinear pairing operation is performed during decryption.

2.3.8 Han et al. [21] proposed ABEKS scheme based on the KP-ABE scheme and permits multi-users to execute a flexible search on the remote encrypted data. ABEKS employs the access control policy from KP-ABE, and search could be done by defining a search policy. The encryptor encrypts the plaintext with the keywrod set of data file, the data user construct an access policy to get a secret key as a trapdoor and server decrypts the ciphertext with the trapdoor to determine whether that file is desired.

2.4 Homomorphic Encryption and Searchable Encryption

Homomorphic encryption (HE) allows one to perform an algebraic operation on ciphertexts without decrypting them. This makes HE useful for searching over encrypted data, as many computation can be done on the encrypted data. Most HE schemes support either additions [23] or multiplications [22] on ciphertexts. After that, Fully Homomorphic Encryption (FHE) was proposed which can compute arbitrary functions over encrypted data. The issue related with FHE is the

performance, since current schemes have high computational and storage overhead. For some applications, so-called somewhat homomorphic encryption schemes can be used. These schemes are more efficient then FHE but allow only a certain amount of additions and multiplications.

The main problem when using somewhat or FHE as is is that the search schemes requires a search time linear in the length of the dataset. So the search time is quite high if we consider them for practical purposes.

Xiong et al[12] proposed a searchable encryption CPABE (SE-CP-ABE) by combining Homomorphic encryption with CPABE.

CHAPTER 3 –ATTRIBUTE BASED ENCRYPTION AND SYMMETRIC SEARCHABLE ENCRYPTION

3.1 Identity Based Encryption

Identity-based Encryption model [24, 16, 17] allows any person/party to obtain a public key from a known identity value which could be for example, a ASCII string. A trusted third party, called the **Private Key Generator** (PKG), is given the authority to generate the corresponding private keys. The PKG first generates a master public key, and helds the key corresponding **master private key** . If the master public key is known, then any person can obtain a public key corresponding to the identity *ID* by combining the master public key with the identity value. To get the private key corresponding to the public key, the person will have to contact the PKG with his/her identity *ID. PKG* uses the master private key to generate the private key for the user with identity *ID*.

This enables users to encrypt messages or verify signatures with no prior distribution of keys between individual users. This is the advantage of IBE that no prior distribution of keys is required and helpful in situations where it is inconvenient or infeasible due to technical restraints. In order to decrypt or sign messages, the users need to obtain the corresponding private key from the PKG.

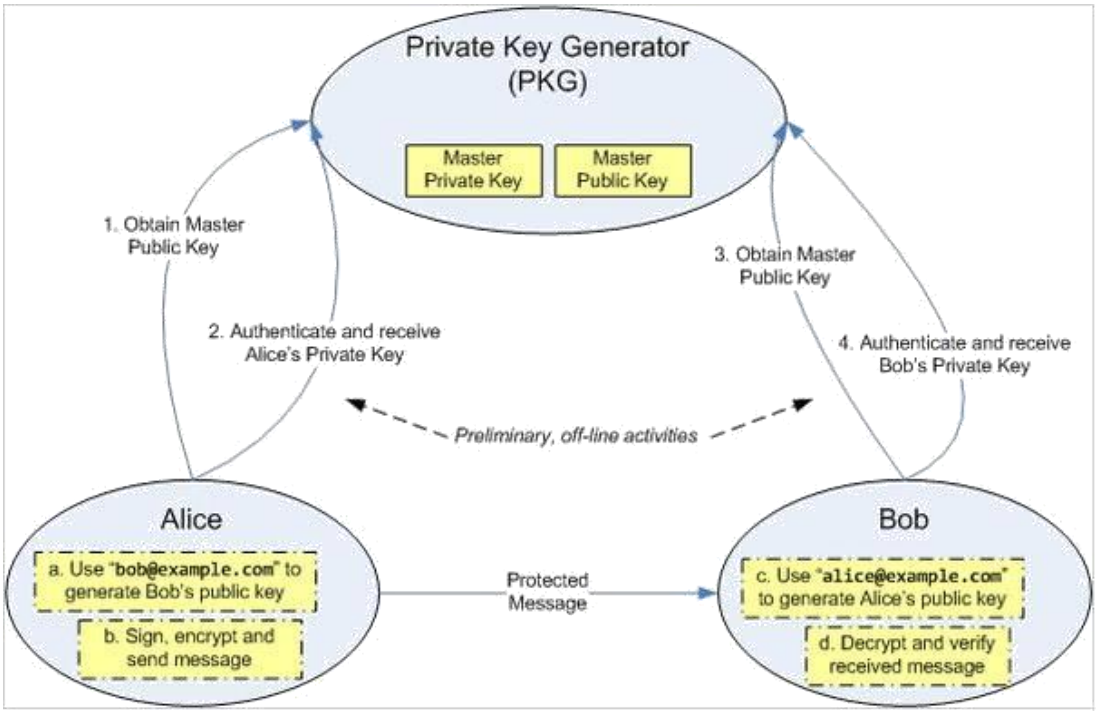


Figure 4: Identity-Based Encryption [1]

The set of four algorithms that form a complete IBE system [1]:

1. **Setup**: The setup algorithm is run by the PKG once for creating the IBE environment. The system parameters are made public .The master key is kept secret and used to derive users' private keys, while. A security parameter k is given as input and output is as follows:
   - A set P of system parameters, including the message space and ciphertext space M and C
   - A master key Km.
2. **Extract**: This algorithm is run when a user makes a request to the PKG for his private key.

- The input are P, K_m and an identifier ID ∈ {0,1}* and returns the private key d for user ID.

3. **Encrypt**: The Encrypt algorithm takes P, a message m ∈ M and ID ∈ {0,1}* as inputs and outputs the ciphertext c ∈ C.
4. **Decrypt**: The decrypt algorithm takes d, P and c ∈ C as inputs and returns m ∈ M as the output.

3.2 Attribute Based Encryption

Attribute-based encryption (ABE) is a relatively recent approach which reconsiders the concept of public-key cryptography. In traditional public-key cryptography, a message is encrypted for a specific receiver using the receiver's public-key. Identity-Based Encryption (IBE) changed the earlier understanding of public-key cryptography by allowing the public-key to be an arbitrary string, for example, the email address of the receiver. ABE(Attribute Based Encryption) goes a step forward and now the identity is not atomic but is set of attributes, e.g., roles, and messages can be encrypted with respect to subsets of attributes (key-policy ABE - KP-ABE) or policies defined over a set of attributes (ciphertext-policy ABE - CP-ABE). The key issue is, that someone should only be able to decrypt a ciphertext if the person holds a key for "matching attributes" where user keys are always issued by some trusted party.

3.3.1 Fine Grained Access Control

Fine-grained access control systems facilitate granting differential access rights to a set of users and allow flexibility in specifying the access rights of individual users [3]. Access control relies on software checks to ensure that a user can access a piece of data only if he is authorized to do so. For example, some document should be made accessible to only those who either have attribute called "Doctor" or have attribute "Head of Department", otherwise the rest of the users should not be given access.

3.3.2 Access Structure

Access structure are used where multiple people/parties need to work together to obtain resource. Group of people/parties that are granted accesses are called qualified. In set theory, they are call qualified sets. The set of all qualified sets are called access structure of system. It describes that who all need to cooperate with who all, in order to access the resource.

Only subgroups of participants contained in the access structure are able to join their part of shares to recompute the secret. Let $\mathbf{P} = \{P_1, P_2, \ldots, P_n\}$ be a set of parties and let $2^{\mathbf{P}}$ denote its power set. A collection A is monotone if for every **B** and **C**, if **B** belongs to A and **B** is subset of **C** then **C** belongs to A. Access structure are monotone in the sense that if a subset S is in the access structure, all sets that contains S as a subset should also be a part of the access structure. An access structure is monotone, if adding a participant to already qualified set of participants will not disqualify it conversely removing participants from a non-qualified set will not make it qualified. An access structure (respectively, monotone access structure) is a collection (respectively, monotone collection) A of non-empty subsets of **P**. The sets which are present in A are termed as authorized sets, and the remaining sets not present in A are termed as unauthorized sets.

3.3.3 Construction of Access Trees

In the access-tree construction, ciphertexts are labeled with a set of descriptive attributes. Private keys are identified by a tree-access structure where every interior node of the tree represents a threshold gate to be satisfied and the leaves are associated with attributes. A user can decrypt a ciphertext with a his/her key if and only if his attributes when assigned to the nodes of the tree, satisfy the tree.

Access tree *T* . Let *T* be a tree representing an access structure. Every non-leaf node represents a threshold gate, which can be described by

- its children
- a threshold value.

If no_x = number of children of a node x

- k_x is its threshold value
    - $0 < k_x \leq no_x$

When $k_x = 1$,

- the threshold gate represents an
  OR gate

When $k_x = no_x$,

- the threshold gate represents an
  AND gate

If x is leaf node,

- x described by an attribute and a threshold value $k_x = 1$

To facilitate working with the access trees, we define a few functions.

1. parent(x): The parent of the node *x* in denoted by parent(*x*).
2. att(x) : The function att(*x*) is defined only if *x* is a leaf node and denotes the attribute associated with the leaf node *x* in the tree.
   - The access tree *T* also defines an ordering between the children of every node, that is, the children of a node are numbered from 1 to *num*.

3. index(x) : The function index(*x*) returns a number associated with the node *x* in the tree, *w*here the nodes in the access structure have unique index value for a given key.

3.3.4 Satisfying An Access Tree

Let *T* be an access tree with root *r*. We denote by T_x the subtree of *T* which is rooted at the node *x*.

If γ *(*which represents the set of attributes*)* satisfies the access tree T_x, then $T_x(\gamma) = 1$. Recursive computation is is done on $Tx(\gamma)$ as follows.

If x represents a non-leaf node in the tree,

- Evaluate $T_{x0}(\gamma)$ for all children x_0 of node *x* in the tree
- $T_x(\gamma)$ would return 1 if and only if at least k_x number of children return 1

If x represents a leaf node,

- $T_x(\gamma)$ would return 1 if and only if $\text{att(x)} \in \gamma$

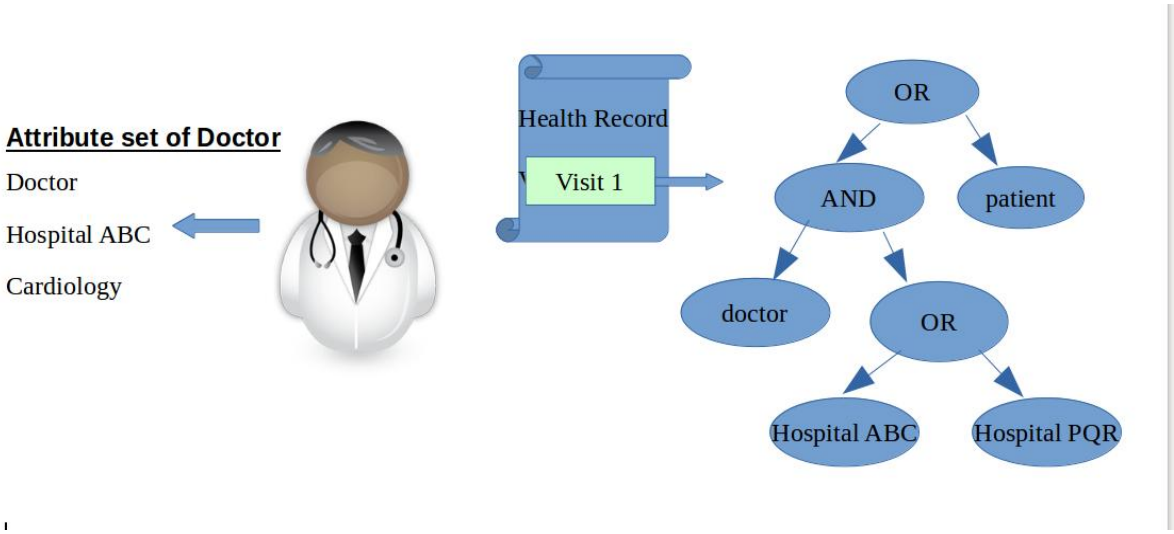


Figure 5: Attributes and Access structure

3.3.5 Secret Sharing Scheme

Secret-sharing schemes (SSS) are used to divide a secret among a number of parties. The information given to a party is called the share (of the secret) for that party. Every SSS realizes some access structure that defines the sets of parties who should be able to reconstruct the secret by using their shares.

In SSS, one can specify a tree-access structure where the interior nodes consist of AND and OR gates and the leaves consist of different parties. Any set of parties that satisfy the tree can come together and reconstruct the secret. Therefore in SSS, collusion among different users (or parties) is not only allowed but is necessarily required. In the construction of KPABE each user's key is associated with a tree-access structure where the leaves are associated with attributes. A user will be able to decrypt a ciphertext if the attributes associated with that ciphertext satisfy the key's access structure.

3.3.6 Types of Attribute Based Encryption

There are two types of ABE depending on which of private keys or ciphertexts that access policies are associated with:

1. Key-policy attribute-based encryption (KP-ABE)[3]
2. Ciphertext-policy attribute-based encryption (CP-ABE)[4]

Key-policy Attribute-Based Encryption

In a key-policy attribute-based encryption (KP-ABE) system, users' keys is associated with an access structure that specifies what all types of ciphertexts the key will be able to decrypt, while

ciphertexts are associated by the encryptor with a set of descriptive attributes. KP-ABE is usefule in cases where one has to set rules about who may read particular documents, but it is unable to specify policies on a per-message basis. Other important applications include secure forensic analysis and pay-TV system with package policy (called target broadcast). Goyal et al. Proposed the first construction of KPABE in [3], which was very expressive in that it allowed the access policies to be expressed by any monotonic formula over encrypted data. In KP-ABE, encrypted documents are associated with sattributes, and decryption key of user is associated with policies. A important point to notice in KP-ABE is that, the encryptor has no control over who can have access to the data he/she encrypts, except for the choice of attributes for the data. The trusted authority generating the keys is trusted that he will give the keys to the appropriate users and deny to unauthorized users. Therefore, in KP-ABE, the main role of intelligence is supposed to be with the trusted authority, and not the encryptor.

KP-ABE is the dual to CP-ABE in the sense that an access policy is encoded into the users secret key, e.g., (A∧ C)∨ D, and a ciphertext is computed with respect to a set of attributes, e.g., {A,B}. In this example the user would not be able to decrypt the ciphertext but would for instance be able to decrypt a ciphertext with respect to {A, C}.

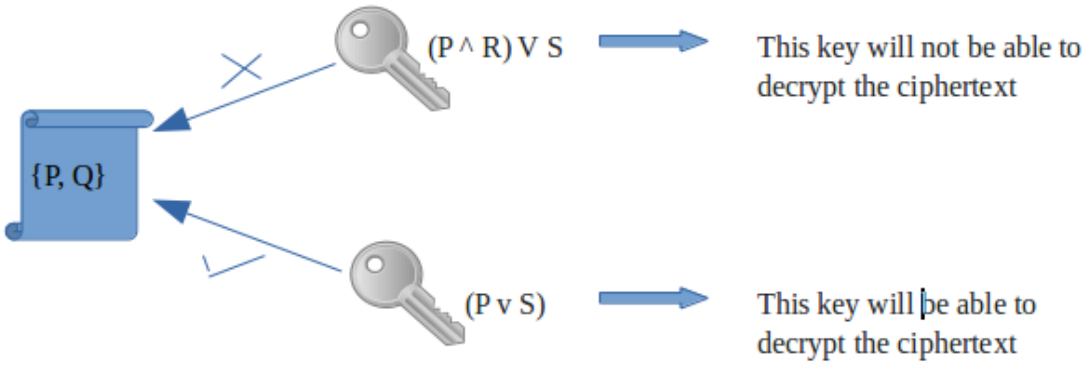


Figure 6: Key-Policy Attribute Based Encryption

An important property which has to be achieved by both, CP- and KP-ABE is called collusion resistance. This basically means that it should not be possible for distinct users to "pool" their secret keys such that they could together decrypt a ciphertext that neither of them could decrypt on their own (which is achieved by independently randomizing users' secret keys).

Ciphertext-policy Attribute-Based Encryption

In ciphertext-policy attribute-based encryption (CP-ABE) a user’s decryption key is associated with a set of attributes and ciphertext is associated with an access policy over a defined universe of

attributes within that system. A user can decrypt a ciphertext, iff his attributes satisfy the access policy of the respective ciphertext. Policies may be defined over attributes using conjunctions, disjunctions and (k,n)-threshold gates, i.e.,k out of n attributes have to be present. For example, let us say that the universe of attributes is defined to be {P, Q, R, S} and user A receives a key to attributes {P, Q} and user B to attribute {S}. If one of the ciphertext is encrypted with access policy (P ∧ R) ∨ S, then user B will be able to decrypt, while user A will not be able to decrypt.

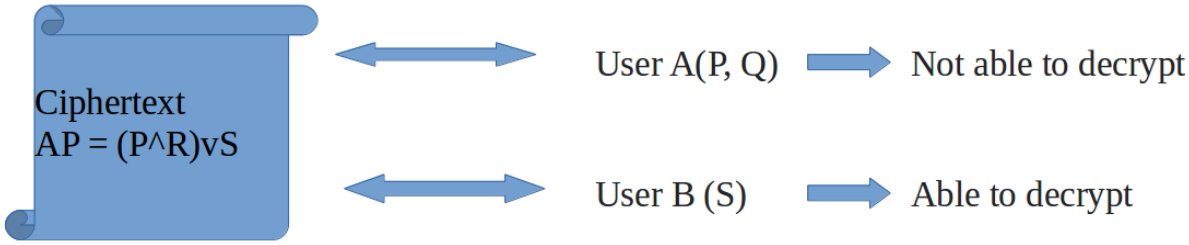


Figure 7: Ciphertext-Policy Attribute Based Encryption

CP-ABE thus helps to achieve implicit authorization, meaning thereby that authorization is included into the encrypted data and only people who satisfy the associated policy can decrypt data. Another feature is, no prior distribution of key is required, which means that users can obtain their private keys after message has been encrypted with respect to access policies. So message can be encrypted without knowing of the actual set of users that will be able to decrypt, but only specifying the policy which allows to decrypt. Any future users that will be given a key with respect to attributes such that the policy can be satisfied will then be able to decrypt the data

A CP-ABE scheme consists of four algorithms: Setup, Encrypt, KeyGen, and Decrypt [4]

1. Setup : The setup algorithm takes as input the security parameter. The output of the algorithm is the public parameters PK and a master key MK. Two random exponents $\alpha, \beta \in Zp$ are chosen:

$$PK = (G0, g, h = g^{\beta}, f = g^{1/\beta}, e(g, g)^{\alpha})$$
$$MSK = (\beta, g^{\alpha})$$

2. Encrypt (PK, M, A) **:** Inputs to the algorithm are PK, a message M, and an access structure A over the universe of attributes. The algorithm will encrypt message M and produce a ciphertext CT for the message M. The ciphertext implicitly contains A. The encryption algorithm encrypts a message M under the tree access structure T .
   The algorithm chooses a polynomial q_x for every node x in the tree. Starting from the root

node R, the polynomials are chosen in topdown fashion. For every node x in the access tree, set $d_x = k_x - 1$.

The algorithm chooses a random $s \in Z_p$ starting from root node and sets $q_R(0) = s$. After that, it randomly chooses d_R other points of the polynomial q_R to define the polynomial. For any other node x, it sets $q_x(0) = q_{parent(x)}(index(x))$ and chooses d_x other points randomly to completely define q_x. The tree access structure is given and finally the ciphertext is constructed.

$$CT = (T , \acute{C} = M\, e(g , g)^{\alpha s} , C = h^s , \forall y \in Y : Cy = g^q y^{(0)}, C'y = H(att(y))^{qy(0)})$$

3. Key Generation(MK,S) **:** The inputs to this algorithm are master key MK and S (set of attributes) that describe the key. The output of this algorithm is a private key SK. The algorithm first chooses a random $r \in Z_p$, and for each attribute, a random $r_j \in Z_p$.

$$SK = (D = g^{(\alpha+r)/\beta} ,$$
$$\forall j \in S : Dj = g^r \cdot H(j)^{rj} , D'j = g^{rj})$$

4. Decrypt(PK, CT, SK) **:** The inputs for this algorithm are the public parameters PK, a ciphertext CT containing an access policy A, and a private key SK. If S satisfies the access structure A then this algorithm will decrypt the ciphertext and return a message M.

3.3 Symmetric Searchable Encryption

The Symmetric Searchable scheme proposed by Song et al. is the first practical scheme which was proposed for searching on encrypted data. This scheme performs sequential scan on the encrypted ciphertexts.

- This is a single writer and single reader scheme. There is only one secret key, one with the owner of the document, who is reader and writer both.
- This is an equality test scheme, thereby meaning that exact keyword matching is done unlike, probabilistic as proposed by Goh [6].
- This is a sequential scan scheme which uses two layers of encryption.

The following procedure if followed to implement this scheme :

1. The data to be encrypted is first split into fixed size words, and a deterministic encryption algorithm is used to encrypt the words w_i, $X_i = E_{k''}(w_i)$.

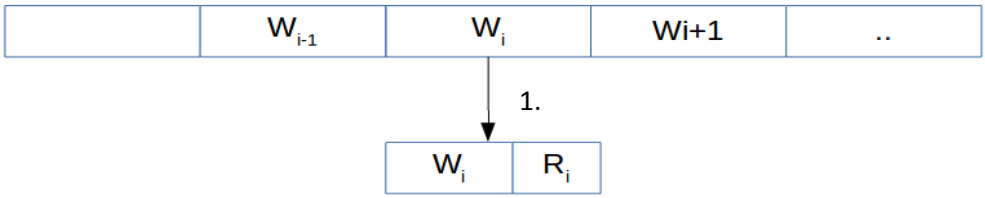


Figure 8: Deterministic encryption of word using symmetric searchable scheme [5]

2. The encrypted word obtained from deterministic algorithm is splitted into two parts left and right, i.e., Xi=<L_i, R_i>.
3. A stream cipher is used to generate a pseudo random value, S_i
4. Using a pseudo-random function, A key k_i is calculate with L_i as input to pseudo random function, $k_i=f_{k'}(L_i)$.
5. The above generated key will be used in to hash the value Si using the keyed hash function vi. With this Yi is generated which is <S_i, $F_{ki}(S_i)$>.

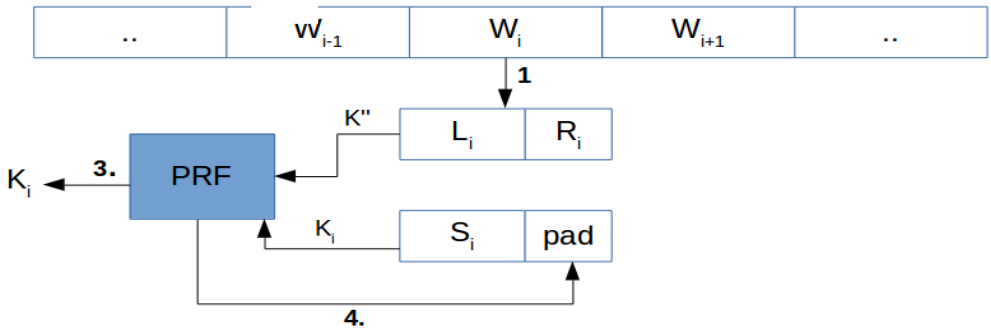


Figure 9: Intermediate steps to generate cipher using symmetric searchable scheme [5]

6. Finally, ciphertext is generated using XOR function $C_i=X_i$ XOR Y_i.

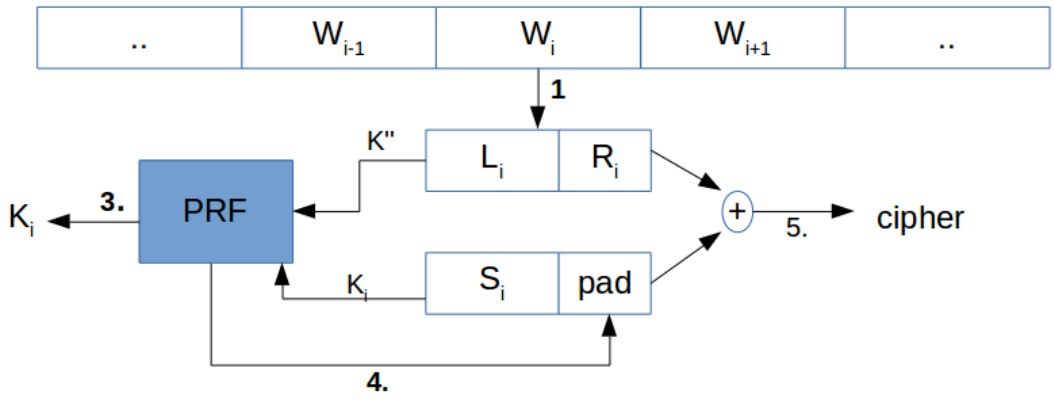


Figure 10: Final cipher for the word using symmetric searchable scheme [5]

II.Trapdoor Generation

i. The keyword is encrypted using the deterministic encryption algorithm, $X= E_{k''}(w)=<L, R>$.

ii. Corresponding key is generated using the left split of the encrypted keyword, $k = f_{k'}(L)$.

iii. Finally, $T_w=<X, k>$

III. Search Phase

i. For all stored ciphertexts C_i, server searched if C_i xor X is of form $<s, F_k(s)>$ for some value of s.

ii. If the match occurs, that means that keyword is found.

CHAPTER 4 - PROPOSED MODEL FOR ACCESS CONTROL AND SEARCHABLE ENCRYPTION

The proposed work assumes that the PHR (Patient Health Record) of a patient is retained by the owner itself in his device, like mobile phones. Health data of a patient is important, therefore, the patient keeps this data in encrypted form to secure his/her data. The work deals with construction of secure and privacy preserving PHR system where the patient can share his data with medical professionals with the fine grained access control using Attribute Based Encryption and capability to search on the encrypted health records.

Users of the system are patient and medical professionals like doctor, nurse, pharmacists and lab technicians. We require that our system ensures

- Confidentiality of patient health record. The patient has his/her own health record on this mobile device. Confidentiality will ensure that no attacker could read the patients health data.
- Privacy of health data. All the search on encrypted data are carried out on the server, so the server should not be able to infer any information on what data is searched for except that the server will be able to know the retrieved data size, the department to which that data belongs. The main content of health record and keywords to be searched should not be revealed to the server.
- Single and Conjunctive keyword search. Both single and conjunctive keyword search should be accomplished. Conjunctive keyword search allows search on each of several keywords in the encrypted data.

The proposed work deals with the use of Attribute Based Encryption (ABE) [1, 4] combined with the symmetric searchable encryption[5]. CPABE is based on public key encryption system in which set of attributes are associated with the user's key and the encrypted data is associated with an access policy. A user can decrypt a encrypted message only if user's decryption key which is associated with a set of attributes satisfy the access policy associated with the encrypted message. This type of ABE where the set of attributes are associated with key and access policy with the ciphertext is called Ciphertext Policy Attribute based encryption (CPABE) [4].

The following assumptions are made in the work which we are proposing:

(a) The Trusted Authority is responsible for generating all system level parameters used in cryptographic operations like the master secret key and public key.

(b) The trusted authority generates the keys for all the stakeholders of the system, like the medical professionals, patient, admin etc. The trusted authority generates the keys after verifying the attributes associated with the stakeholders.

(c) A user has

- a unique id
- a set of attributes verified by TA

(d) Access policies associated with different sections/departments are stored in the database. During encryption phase this database is consulted.

(e) The PHR (Patient Health Record) of a patient is retained by the patient itself in his device, like mobile phones. The PHR in encrypted form is sent to server and stored there. The search operations are performed at the server. The server is trusted in a way that it will perform the operations as specified by the patient and not by any other professional. So, any search operation to be performed by medical professional first needs to send its request to the patient.

(f)The server should not be able to perform any other operation on patient data such as reading the contents of PHR. Therefore, the data should in secure form at the server side.

Patient Health Record is a JSON file having the following structure:

i. The health record consists of various sections consisting of ORU's(which contains the all the visits of the patients and data is in HL7 format), Vaccinations, Data , Current Medication.

ii. In the Data sections, there are sub sections based on the Department to which the visit belongs. For example, the departments could be cardiology, endocrinology or oncology.

iii. In the departments there are further sub sub sections which are based on the date which may contain open and closed visits.

- Open visits are those where suppose some test results are still due, and prescription is still left. In those cases the visits are said to be *open.*
- Closed visits are those where all the prescription and lab tests have been done and the medication is said to be completed.

iv. The visits contain the values of important vitals, such as Blood pressure, temperature, weight, etc. depending on the department to which that visits belong.

4.1 Section Wise Encryption

A health record consists of various sections, like visits, medication, allergies, etc. So, a patient may want to give selective access to different medical professionals for different sections. For eg : pharmacist can access "Current Medication" and not the visits. So, the patient encrypts the different sections separately with their own access policy instead of encrypting the whole health data with single access policy.

Access policies in our system are determined by the nature of the data contained in the sections and also based on the attributes of the users who are allowed access. For example, if the visit belongs to the Cardiology department then a doctor or nurse from cardiology department should be given access. So this depends both on nature of data, along with the attributes the user who is given access.

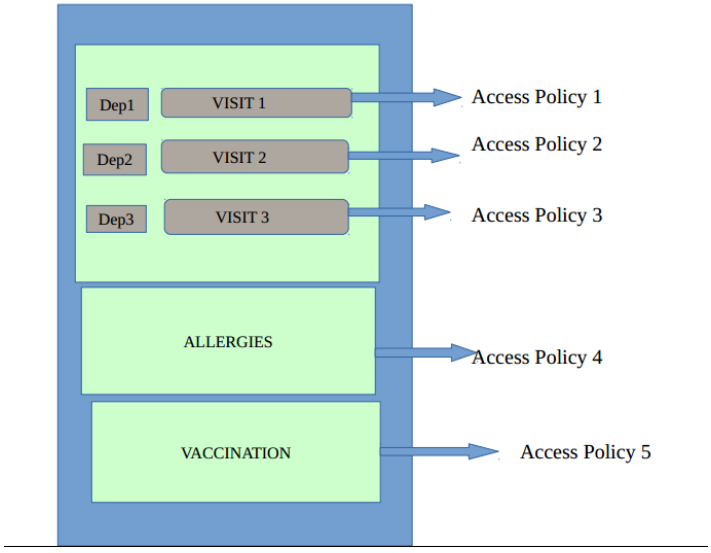


Figure 11: Section-wise encryption of PHR

The above figure shows the section wise encryption of health data which is unique in our implementation. This helps in restricting the medical professionals to access only those information for which their attributes satisfy the access structure associated with the encrypted section.

4.2 User Attributes

The user (patient or medical professional) attributes contain a unique identifier, type identifier indicating whether he/she is a doctor or nurse or pharmacist or patient. It also contain the department to which he/she belongs i.e., Cardiology, Oncology, Endocrinology. It also contains the hospital to which he/she belongs in case of medical professionals. The section access attribute defines the section to which the medical professional has a access. For example, the pharmacist can have access to the medication section. The following table summarizes the attributes possessed by different medical professionals, admin, emergency and the patient.

Stakeholder	Attributes				
	Unique identifier	Type identifier	Department	Hospital	Section Access
Admin	✓	✓		✓	
Doctor	✓	✓	✓	✓	
Nurse	✓	✓	✓	✓	
Pharmacist	✓	✓			✓
Lab Technician	✓	✓			✓
Patient	✓	✓			
Emergency	✓	✓		✓	

Table 1: Attributes of different stakeholders

Keys possessed:

- The medical professionals, admin and patient all have one public key and one decryption key, which are the keys generated by Trusted Authority and are used for access control.
- The patient also has the symmetric key which is used for searchable encryption.

4.3 Proposed system working for first version

The system proposed works as follows:

1. Setup: Trusted Authority runs this Setup algorithm, which takes as input a implicit security parameter. It outputs the master secret key and public parameters.
   It can be described as

   Setup (1^K) -> (master secret key, public parameters)

2. ABE-Keygen: This algorithm is used to generate the attribute based decryption key of the users. In this the users (patient, medical professionals, admin) send their attributes to the Trusted Authority and TA uses these to generate the decryption key. Trusted Authority, in order to generate the key, takes as input the public parameters and attribute set of user as the public input. And the private input is the master secret key. At the end of this algorithm, the user gets his/her decryption key.

   ABE-Keygen(*params, w , msk*) -> d_w

3. Encrypt-Rec-File :

This algorithm of encryption of Health Record File is run by the owner of the PHR, i.e., the patient. It parses the ORU section, Data section, Allergies section etc, and takes the individual visits and sections as *data*, an access policy *A* over the universe of attributes and the system public parameters *params* as input to the algorithm. The algorithm will encrypt the individual visits and produce the corresponding ciphertext *ct*. Patient deletes the plaintext health record file after the encryption is done, so that no attacker could read the health data intentionally or unintentionally.

Encrypt-Rec-File algorithm will encrypt the PHR section-wise with the appropriate access policy, and create another encrypted PHR, in the same format as the original PHR.

Encrypt-Rec-File(*params, A, data*) -> *ct*

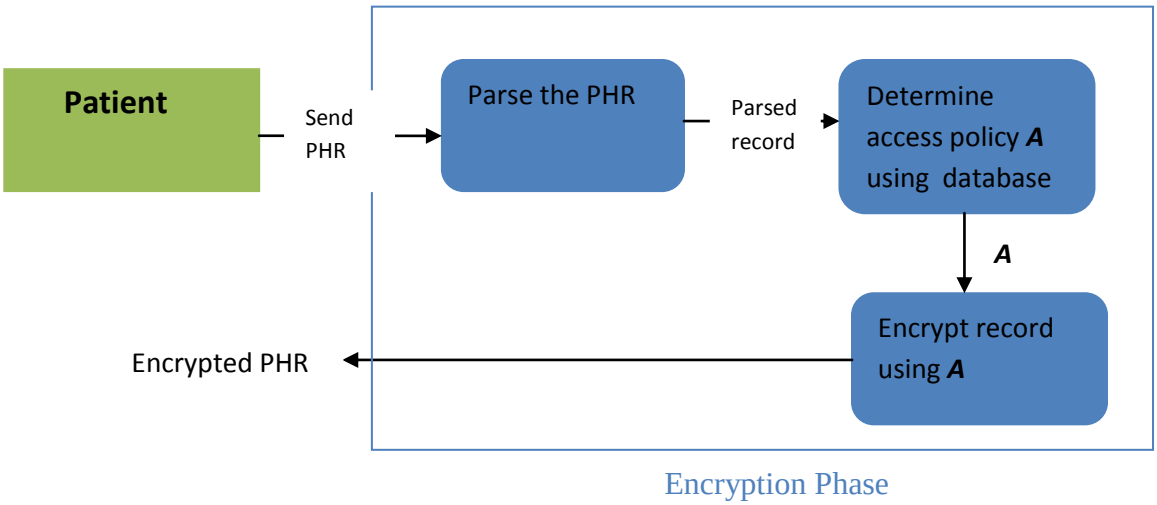


Figure 12: Flow diagram for encryption phase

4. Secure-Index:

The encrypted secure index generation algorithm is run by the owner of the PHR, i.e., the patient. There is a keyword file which has corresponding keywords for all the visits and sections of the PHR. The file is in same format as the original health record.

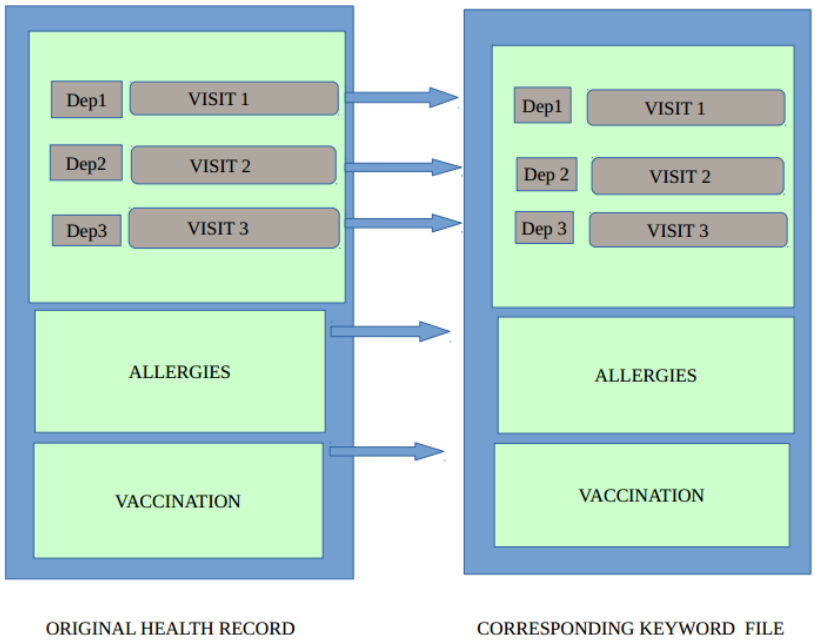


Figure 13: Correspondence between PHR and keyword file of PHR

For example:

Visit 1 : { “Blood_Pressure Weight Temperature Red_blood_cells Paracetamol Cardiology”}

Visit 2 : {“Platelets Weight Height Temperature White_blood_cells Paracetamol Cardiology”}

Suppose these are two visits of a patient. The corresponding are the keywords in the respective visits. The correspondence between the encrypted keyword file and original plaintext keyword file is maintained by keeping both in the same sequence with respectect to sections and subsections.

This algorithm takes as input the symmetric key *k''* of the patient, and a set of keywords kw = $\{k_{wi}\}^{l}i=1$corresponding to a visit/section. The algorithm outputs a secure index SI(**kw**) for keyword set **kw**, which will be associated with the ciphertext ct(a visit or a section), After running this algorithm on all the keyword sets of all visits and sections, the output is a corresponding encrypted keyword file containing corresponding SI(kw) for each visit and sections.

Secure-Index (*k''*,**kw**) -> SI(**kw**)

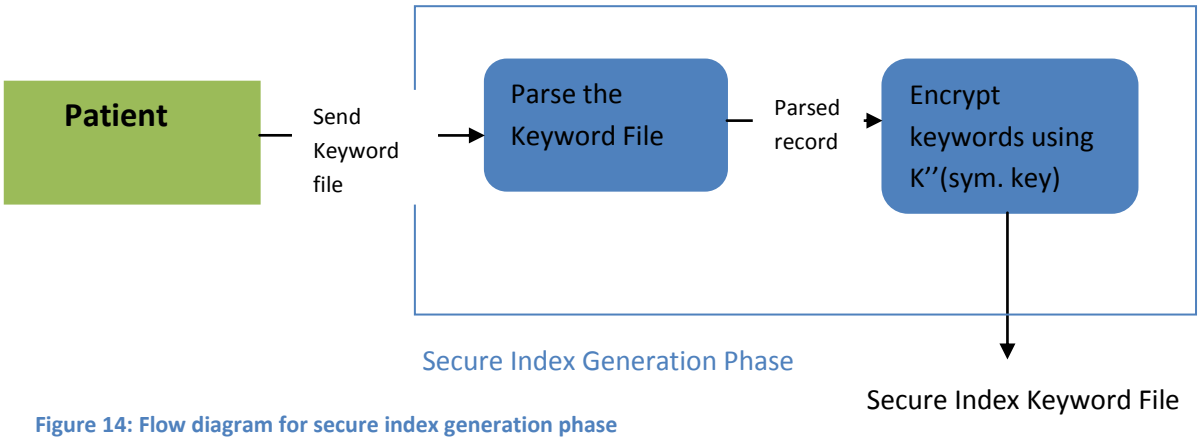


Figure 14: Flow diagram for secure index generation phase

5. Trapdoor:

The trapdoor generation algorithm is also run by the owner of the PHR, i.e., the patient. The keywords to be queried are sent to the patient by the medical professionals and the patient then creates the trapdoor on those keywords and sends it to the server. This algorithm takes as input the symmetric key k" of the patient and set of keywords **kw'** = $\{kw_i\}^{l}i=1$. This algorithm outputs the set of trapdoors $T_{\mathbf{kw'}}$ for the corresponding keyword set **kw'**. In case of conjunctive keyword search, for all the keywords the trapdoor are generated and sent to the server for searching.

Trapdoor (k",**kw'**)-> $T_{\mathbf{kw'}}$

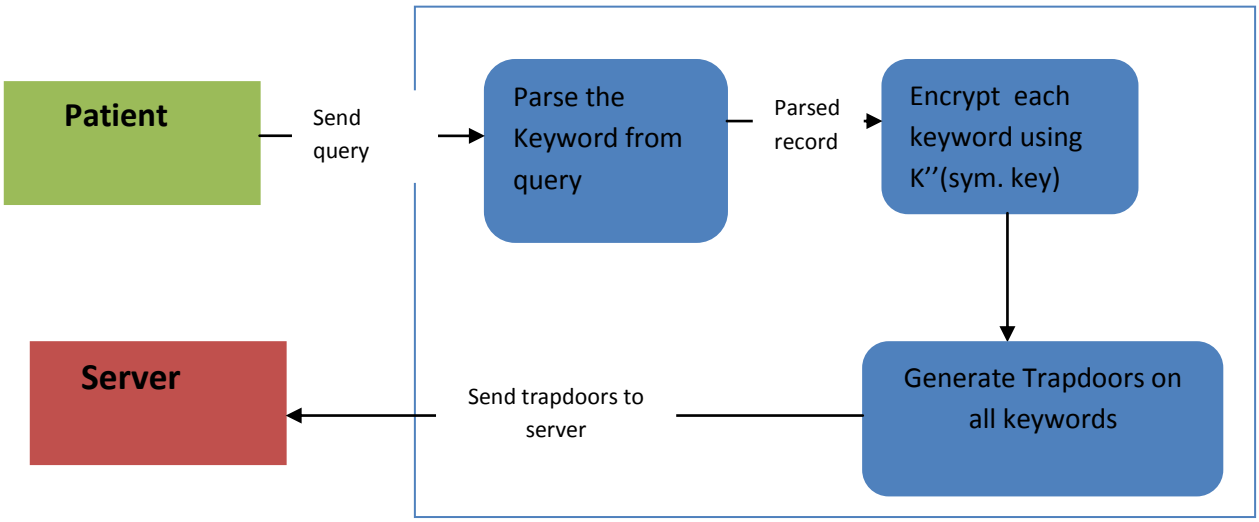


Figure 15: Flow diagram for Trapdoor generation phase

6. Search:

The trapdoor which is sent by the client is used to search on the encrypted keyword file. In Song et al. Scheme, the search time is proportional to number of keywords in the document. Therefore, we have searching on reduced number of words by performing search only on the keywords instead of the whole record.

The keywords search algorithm is run by the server where the encrypted PHR and Secure Index keyword file is stored. This algorithm is run for all the Secure Indexes for each visit and sections of the PHR present in the Secure Index keyword file. This search algorithm takes as input the secure index SI(**kw**) for the keyword set **kw** of a visit or section, the set of trapdoors $T_{kw'}$ sent by the patient for the keyword set **kw'** and the encrypted PHR. It outputs 1, if there is keyword match from the keyword set kw' in the tapdoor set $T_{kw'}$. This search procedure is followed for each keyword in the keyword set kw' in the tapdoor set $T_{kw'}$ and for all the Secure Indexes present in the encrypted secure index keyword file. For all the matching cases, the corresponding visit or section from encrypted PHR is copied and placed in the new PHR of retrieved records. In case match does not occur, it outputs 0.

Search ($T_{kw'}$, SI(**kw**)) -> 0 or 1

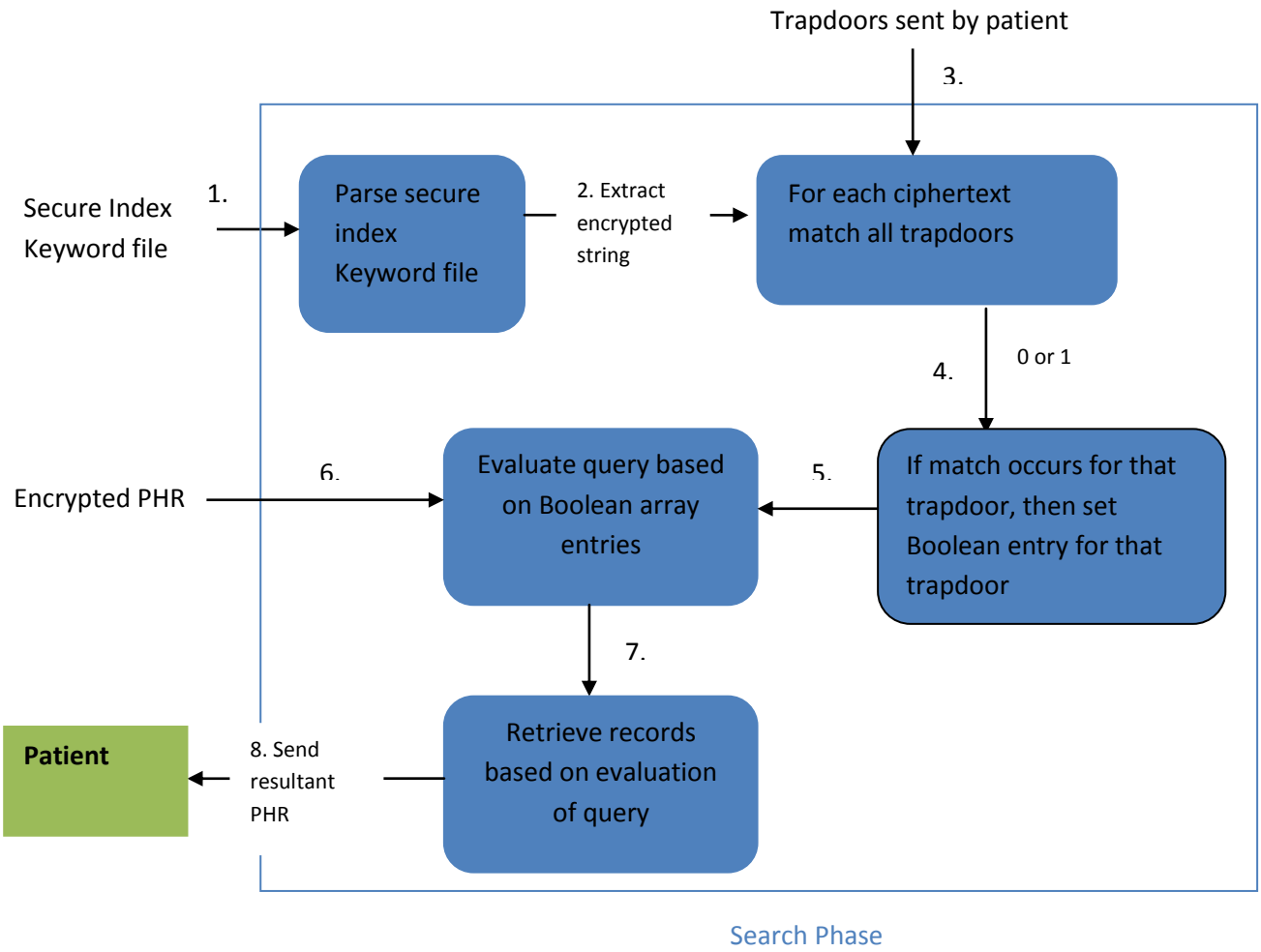


Figure 16: Flow diagram for search phase

7. Decrypt:

The decryption algorithm can be run by the medical professional who has initiated the search query or by the patient himself. This algorithm takes as input the system parameters *params*, the searched ciphertexts ct, and users decryption key dw. It outputs plaintext visits/sections if the set w of attributes in the decryption key of the user satisfies the access policy associated with the corresponding encrypted records. Otherwise, it outputs NULL.

Decrypt(*params*, *ct*, *dw*) -> *data* or NULL

The above mentioned steps are taken in our first implementation. A modified improved version of this is also proposed. In the modified version the search time reduces for the uncommon keywords like, “Allegra”, “Paracetamol”, “Serum_Glucose”, etc.

4.4 Proposed system working for second version

A second version is implemented to improve on the search time on keywords which are specific to a department or section. For the keywords which are common to all the departments, the search time is almost the same as in the first version.

The algorithms used in the second version are as follows:

1. Setup: Trusted Authority runs this Setup algorithm, which takes as input a implicit security parameter. It outputs the master secret key and public parameters.

It can be described as

$$\text{setup}(1^K) \rightarrow (\text{master secret key, public parameters})$$

2. ABE-Keygen: This algorithm is used to generate the attribute based decryption key of the users. In this the users(patient, medical professionals, admin) send their attributes to the Trusted Authority and TA uses these to generate the decryption key. Trusted Authority, in order to generate the key, takes the public parameters and attribute set of user as the public input. And the private input is the master secret key. At the end of this algorithm, the user gets his/her decryption key

$$\text{ABE-Keygen (params, w, msk)} \rightarrow d_w$$

3. Encrypt-Rec-File:

This algorithm of encryption of Health Record File is run by the owner of the PHR, i.e., the patient. It parses the ORU section, Data Section, Allergies section etc, and takes the individual visits and sections as *data*, an access policy *A* over the universe of attributes and the system public parameters *params* as input to the algorithm. The algorithm will encrypt the individual visits and produce the corresponding ciphertext *ct*. While encrypting the individual visits it will also create a linked list for each department and save the index of the visit in the respective departments linked list. These linked list will be helpful to restrict the search space during searching.

Encrypt-Rec-File algorithm will encrypt the PHR section-wise with the appropriate access policy, and create another encrypted PHR, in the same format as the original PHR.

$$\text{Encrypt-Rec-File}(\textit{params, A, data}) \rightarrow \textit{ct}$$

4. Hash-Create: This Hash-Create algorithm is run by the owner of the PHR, i.e., the patient. This

algorithm is used to create a hash table on the unique keywords of every department and all sections. There could be some keywords which are common to all departments, then they come under the category "COMMON". The Hash-Create algorithm takes symmetric key **k''** of patient and the keyword **kw'** as inputs to the algorithm. It then hashes the value of XOR between these two inputs. The output of this algorithm is the hash table ***H*** with the key as the hash created after the XOR operation and value is the department or section to which it belongs.

Hash-Create (**k'', kw'**) -> ***H***

5. Secure-Index: The encrypted secure index generation algorithm is run by the owner of the PHR, i.e., the patient. In the second modified version, this is run after the Hash-Create algorithm. There is a keyword file which has corresponding keywords for all the visits and sections of the PHR. This algorithm takes as input the symmetric key *k''* of the patient, and a set of keywords kw = {kwi}^{l}i=1corresponding to a visit/section. The algorithm outputs a secure index SI(**kw**) for keyword set **kw**, which will be associated with the ciphertext ct(a visit or a section), After running this algorithm on all the keyword sets of all visits and sections, the output is a corresponding encrypted keyword file containing corresponding SI(kw) for each visit and sections.

Secure-Index(*k''*,**kw**) -> SI(**kw**)

6. Trapdoor Generation:

Phase 1: The Phase 1 trapdoor generation algorithm is run by the owner of the PHR, i.e., the patient. Medical professional sends the keywords to be queried to the patient and patient creates the phase 1 trapdoor which will be used to check the entry in the hash table ***H.*** This algorithm takes the symmetric key of the patient and set of keywords **kw'** = {kwi}^{l}i=1 to be searched. The output of this algorithm is $T1_{\mathbf{kw'}}$ **,** that means trapdoor of phase 1.

Trapdoor-phase1(k'',**kw'**)-> $T1_{\mathbf{kw'}}$

Phase 2: The Phase 2 trapdoor generation algorithm is also run by the owner of the PHR, i.e., the patient. The keywords to be queried are sent to the patient by the medical professionals and the patient then creates the trapdoor for phase 2 on those keywords and sends it to the server. This

algorithm takes as input the symmetric key k'' of the patient and set of keywords **kw'** = $\{kwi\}^{l}i=1$. This algorithm outputs the set of trapdoors $T2_{\mathbf{kw'}}$ for the corresponding keyword set **kw'**.

Trapdoor (k'', **kw'**)-> $T2_{\mathbf{kw'}}$

7. Search: The search algorithm is divided into two phases :

Phase 1: The keywords search algorithm phase 1 is run by the server. This algorithm takes the hash Table ***H*** and the T1kw' as the inputs to the algorithm. It takes a trapdoor from the set of trapdoors and searches that in the hash table which was created by the server. If the match is found then, the value corresponding to that hash is written to another file for reference in next phase of Search algorithm. The benefit of this algorithm is to restrict the search space to lesser sections, which in earlier version was the whole PHR file.

Search-Phase1(H,T1kw') -> temporary file 'sections'

Phase 2: The keywords search algorithm is run by the server where the encrypted PHR and Secure Index keyword file is stored. This algorithm is run for all the Secure Indexes for each visit and sections of the PHR present in the Secure Index keyword file. This search algorithm takes as input the secure index SI(**kw**) for the keyword set **kw** of a visit or section, the set of trapdoors $T_{\mathbf{kw'}}$ sent by the patient for the keyword set **kw'** and the encrypted PHR. It outputs 1, if there is keyword match from the keyword set kw' in the trapdoor set $T_{\mathbf{kw'}}$. This search procedure is followed for each keyword in the keyword set kw' in the trapdoor set $T_{\mathbf{kw'}}$ and for all the Secure Indexes present in the encrypted secure index keyword file. For all the matching cases, the corresponding visit or section from encrypted PHR is copied and placed in the new PHR of retrieved records. In case match does not occur, it outputs 0.

Search(T**kw'**, SI(**kw**)) -> 0 or 1

CHAPTER 5 – ARCHITECTURE AND DESIGN

This work assumes that the PHR of the patient is held by the patient in his/her device. PHR of patient are encrypted using CPABE which gives selective access control and requires no prior distribution of keys among the users (medical professionals, admin, patient). But for searchability, patient is the single reader and writer, meaning encryption can be done by patient himself and search operation can also be performed by patient only.

In this architecture, we provide overview of how records are encrypted section-wise and about the search operations which also allow conjunctive keyword search. This sections describes the overall architecture, different components and their work and the implementation details.

5.1 System Architecture

Our system supports both fine-grained access control with keyword search which is achieved by symmetric searchable encryption.

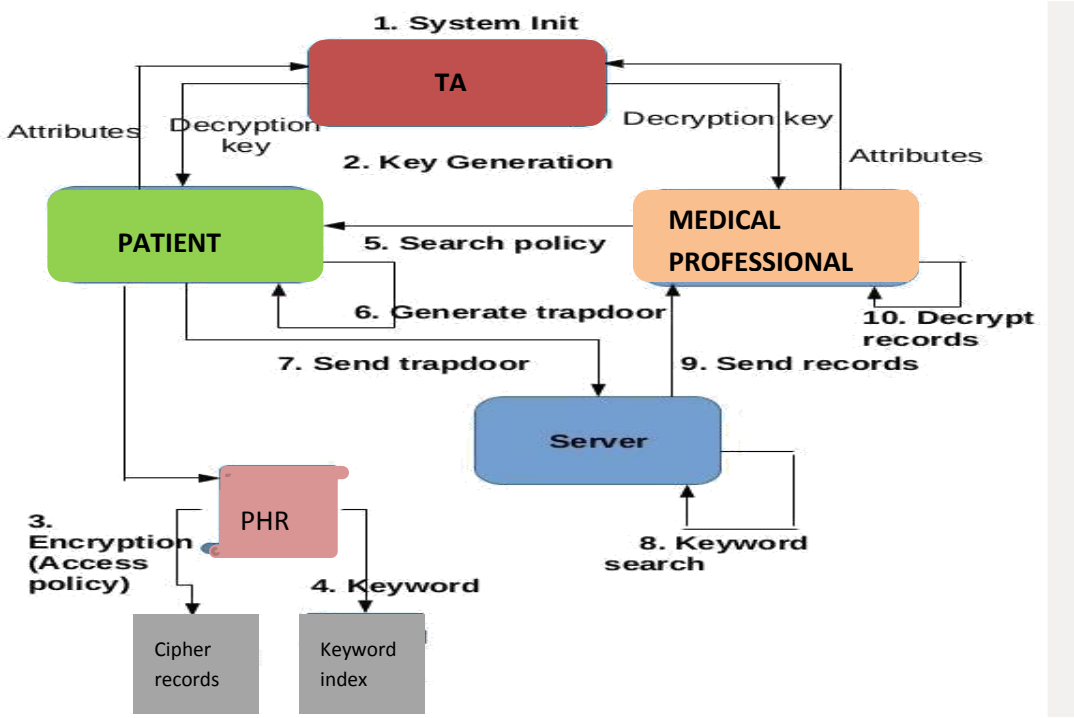


Figure 17: System architecture

5.1.1 Participants of the system

The entire system, shown in figure 6.1 below, has four main participants:

1. Trusted Authority (TA): It is the key generation center which is trusted by all the users of the system. Trusted Authority is responsible for
    - initializing the system level parameters
    - generating decryption keys of users

    It is assumed that the TA verifies the attributes submitted by the users of the system (medical professionals, patient, and admin) and then only generates the corresponding decryption keys.
2. Patient: Patient is the owner of the PHR. Patient encrypts his PHR and then stores it on the server. This encrypted data can be shared with the medical professionals, if their credentials satisfy the access policy as specified by the patient. The responsibility of the Patient is to -

    1. Encrypt the health record.
    2. Choose keywords to build secure index.
    3. Encrypt the keyword file.
    4. In the modified version, create hash table on the unique keywords of the PHR.

3. Medical Professional: Medical professionals are also part of this system who queries the encrypted PHR stored at the server. Medical Professional submits the query to the patient and patient further creates the trapdoor and sends it to the server. For the retrieved records, if the medical professional attributes satisfy the access policy of the encrypted records, then only the restoration of original records is possible. Medical professional's responsibility is to initiate the search query and send it to the patient for creating the trapdoor.

4. Server: This entity in the system provides storage of the PHR and searching on the encrypted PHR and retrieval service. It stores the encrypted PHR sent by the patient. The encrypted PHR has corresponding encrypted keyword file, which is searchable and server do the searching in this file. The retrieved records are sent to the medical professional via patient. Responsibility of server -
    1. To search in the encrypted keyword file.

2. Send the retrieved records back to the medical professional via patient.

5.1.2 Health Record Storage

1. PHR: Patient Health Record contains medical data of the patient. The medical data may encompass current vitals, medication history, prescriptions, diagnoses information, lab tests result etc. All these information is present in plaintext format. This PHR is a single JSON object which contains various key-value pairs corresponding to various sections.

   Following is the format of the PHR present in the JSON file:

HL7 Portion	"ORU" : [
	{
	Date, oru, Hospital, Department, Doctor},
	{
	Date, oru, Hospital, Department, Doctor
	}
	]

Non-HL7 Portion	"DATA":{
	Department1:{
	Date 1:{
	Date1Time1:{"concept id, observation value, timestamp"
	}
	}
	},
	Department 2:{
	Date 1:{
	Date1Time1:{"concept id, observation value, timestamp"
	}
	}
	}
	}

	"CURRENT MEDICATION":[ ]
	"VACCINATION":["vaccination1 vaccination2 "]
	"ALLERGIES":["Allergy1 Allergy2"]

Table 2: Format of PHR

The different sections of this health record are described below:

- ORU: ORU section is a array of JSON objects. The individual JSON objects present inside this array represents single visit of the patient to a doctor. The visit contains details like date of the visit, department to which this visit belong, which hospital, billing status, whether the visit is closed or not and HL7 string which is the observation message.

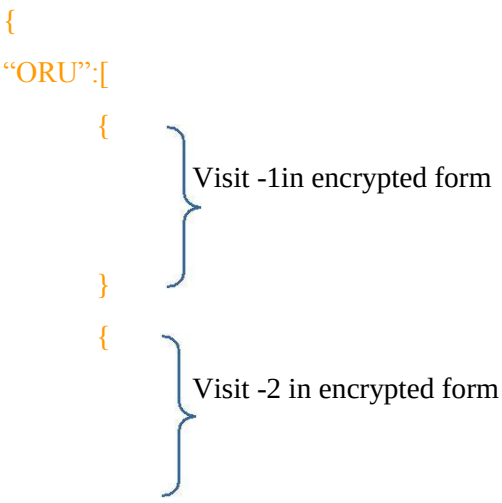
- Data Section: In the visits in the "ORU" section, the observations were in the HL7 format, which is not easily understandable by medical professional. So, that HL7 string is parsed and the observations are placed in the "DATA" section. All the visits are placed in their respective departments. For example, all "cardiology" related visits are placed under the key "CARDIOLOGY". Here each vital is given a id which is called concept id. All the observation values, who examined it and what was prescribed is given here.

- Vaccination Section: Vaccination is a JSON Array which conatains all the vaccines administered to the patient till now. This array contains names of the vaccines.

- Current Medication: This is also a JSON Array which contains the medication which is been followed by the patient.

- Allergies: The Allergies section is a JSON array which describes about what all allergies the patient is suffering from.

2. Encrypted PHR: The encrypted PHR has the same format as the original PHR, except that the among the key-value pairs, only the value part is encrypted and the keys are present in plaintext.



```
        }
],
"DATA":{
"DEPARTMENT-
1":{
        "DATE 1":{
        "DATE 1 TIME 1":[ encrypted Concept ID's ],
        "DATE 1 TIME 2":[ encrypted Concept ID's ]
        }
},
"DEPARTMENT-
        2":{  "DATE
        2":{
        "DATE 2 TIME 1":[ encrypted Concept ID's],
        "DATE 2 TIME 2":[ encrypted Concept ID's]
        }
},
"VACCINATION" : [encrypted vaccines administered],
"CURRENT_MEDICATION" :[ encrypted medication],
"ALLERGIES" : [ encrypted allergies ]

}
```

Figure 18: Encrypted PHR

3. Keyword File: The keyword file contains the keywords of every visit present in the "ORU" and "DATA" section and also keywords for other sections. The same key-value pairs are present in the keyword file as they were present in the PHR, with the modification that in the values part, keywords are present instead of the data. All the keywords in a section/visit are present in a single string form, with a single space present between each keyword. Figure below shows the keyword file in the plaintext format.

```
{
"ORU":[
        "key1 key2",  ⟶ keywords corresponding to visit 1
        "key1 key2"  ⟶ keywords corresponding to visit 2
],
"DATA":{
"DEPARTMENT1":
{
        "DATE 1" :{
        "DATE 1 TIME 1":[ "key1 key2" ],
        "DATE 1 TIME 2":[ "key1 key2" ]
        }
},
"DEPARTMENT-
        2":{ "DATE
        2":{
        "DATE 2 TIME 1" : ["key1 key2"],
        "DATE 2 TIME 2":[ "key1 key2"]
        }
},
"VACCINATION": ["keyword1 keyword 2"], ⟶Keywords corresponding to vaccination section
"CURRENT_MEDICATION":["keyword1 keyword2"], ⟶ Keywords in medication
section
 "ALLERGIES" : [ "keyword1 keyword2" ] ⟶ Keywords in allergies section

}
```

Figure 19: Keyword file for PHR

4. Secure Index Keyword File : This secure index file is created by encrypting the keyword file using symmetric key of the patient. This will be used to search for keywords corresponding to records in the PHR. The keywords string for every visit or section is parsed and submitted to the secure index creation algorithm. The secure index obtained is placed in the new secure index keyword file, which is in same format as the keyword file, having same "keys" for the key-value

pairs. Only the values part is changed, where instead of plaintext keywords, now we have encrypted keywords which could be searched upon. Figure below shows the secure index keyword file:

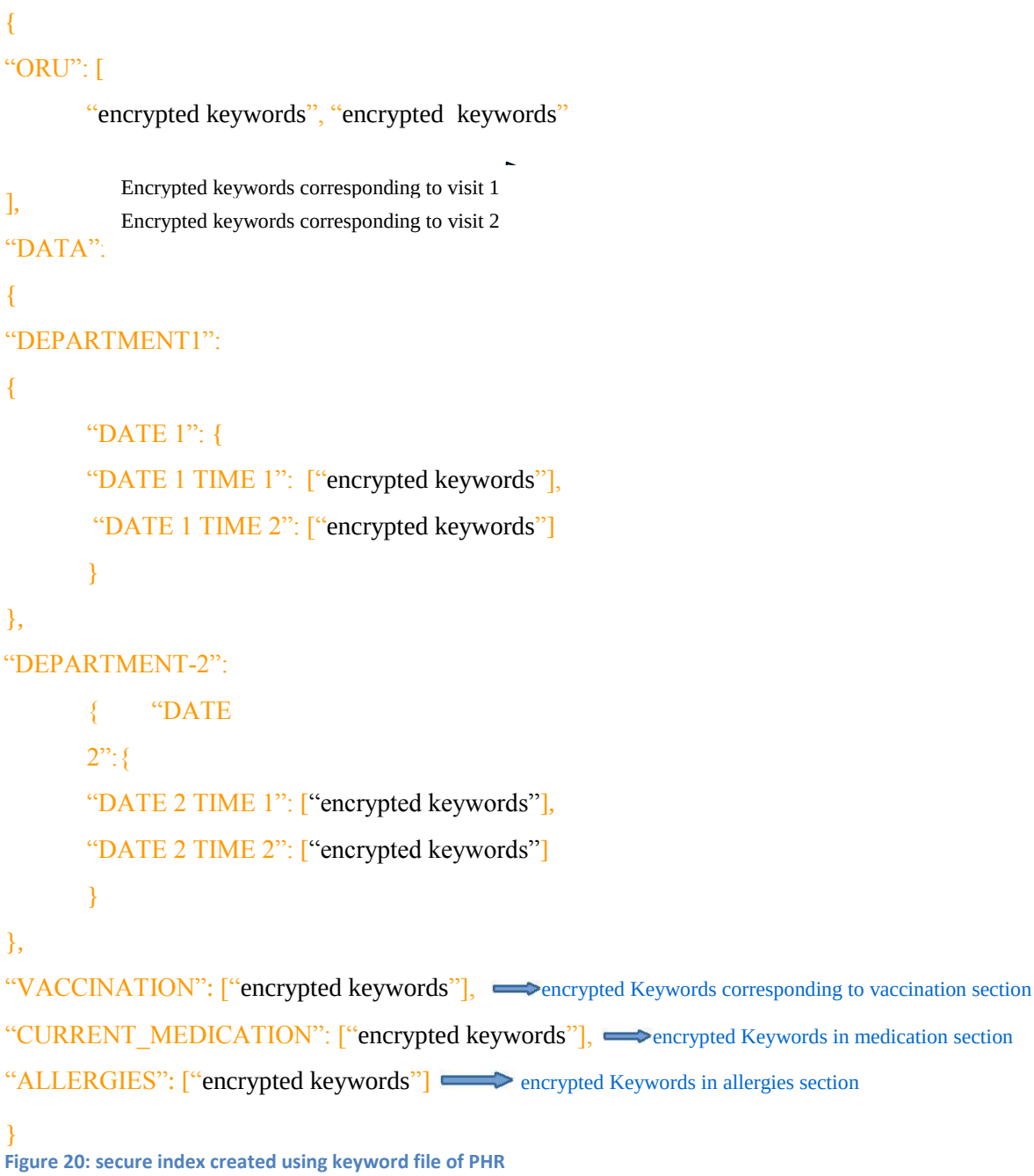


Figure 20: secure index created using keyword file of PHR

5.2 Algorithms

A user (patient, medical professional, admin) is recognized by pair (*id*, *w*), where w denotes a subset of attributes possessed by the user and id is the unique id.

For the two schemes proposed in our system, below are the algorithms used:

1. Setup**:** In the Setup algorithm, the Trusted Authority two random elements alpha and beta belongs to Zp. This algorithm generates the public key and master secret key.

$$\text{Public key} = \text{Go}, g, h=g^{\beta}, f=g^{1/\beta}, e(g,g)^{\alpha}$$

$$\text{Master secret key} = (\beta, g^{\alpha})$$

2. ABE-Keygen: To generate decryption for a user (medical professional, patient, admin)with the set of attributes w, the following protocol will be executed between the user and the TA :

User sends a request for decryption key along with his credentials which corresponds to the set of attributes to the TA.

It is assumed that the TA validates the credentials given by the user and does not generate the key if the credentials fail. Otherwise, it randomly chooses r belongs to Z_p, then r_j belongs to Z_p for each attribute j belongs to S.

3. Encrypt Rec-File :

Step 1: Read the PHR which is a JSON Object

Step 2: Iterate over the JSON Objects for the key value "ORU"

Step 2.1: For each JSON Object, that represents a visit, read the value of department ***dep*** for the key "Department"

Step 2.2: Based on the value of ***dep,*** read the access policy **A1** to be chosen from the database, so as to encrypt the visit.

Step 2.3: In the second version, based on the value of ***dep***, add the index number to the respective department linked list.

Step 2.4: Encrypt the JSON Object which represents a visit with access policy ***A1*** using

CPABE Encryption

Step 3: Read the JSON Object for the key value "DATA" Step 3.1: Iterate over the department keys ***dep***

Step 3.2: Read the JSON Object for that department key Step i: Iterate over the Date keys

Step ii: Read the JSON Object for that Date key

Step a: Iterate over the date keys where the value is JSON Array

Step b: Read the JSON Array for the above date key

Step c: Based on the ***dep*** of the JSON Array in which we are reading**,** the access policy is chosen from the database.

Step d: The JSON Array is encrypted using CPABE encryption with the above determined access policy A2

Step 4: Read the JSON Array for the key value "VACCINATION"

Step 4.1: Based on the key value, i.e, "vaccination" the access policy ***A3*** is determined from the database with which this array is going to be encrypted.

Step 4.2: Encrypt the array with the access policy ***A3*** using CPABE based encryption. Step 5: Read the JSON Array for the key value "ALLERGIES"

Step 5.1: Based on the key value, i.e, "allergies" the access policy ***A5*** is determined from the database with which this array is going to be encrypted.

Step 5.2: Encrypt the array with the access policy ***A5*** using CPABE based encryption.

4. <u>Secure-Index :</u>

The steps followed in this algorithm are:

Step 1: Read the keyword file which is a JSON Object Step 2: Read the JSON Array for the key value "ORU"

Step 2.1: Iterate over the strings in the JSON Array

Step 2.1.1: Parse the string for words which are separated by single space

Step 2.1.2: Encrypt each word with deterministic encryption algorithms

Ek"(w)

Step 2.1.3: Split Ek"(w) into two parts <L,R>

Step 2.1.4: generate pseudo random value Si using stream cipher

Step 2.1.5: generate key ki using pseudo random function f(.) and and left part of encrypted keyword, ki=f(Li)

Step 2.1.6: Use ki in the keyed hash function to hash the pseudo random value Si Yi=<Si, Fki (Si)>

Step 2.1.7: Encrypt Xi using Yi and XOR operation, Ci = (Xi XOR Yi)

Step 2.2: Store Ci under the same the same index from where the plaintext keyword string was extracted.

Step 3: Read the JSON Object for the key value “DATA”

Step 3.1: Iterate over the department keys ***dep***

Step 3.2: Read the JSON Object for that department key

Step 3.2.1: Iterate over the Date keys

Step 3.2.2: Read the JSON Object for that Date key

Step i: Iterate over the date keys where the value is JSON Array

Step ii: Read the JSON Array for the above date key

Step a: Iterate over the strings in the JSON Array

Step b: Parse the string for words which are separated by single space

Step c: Encrypt each word with deterministic encryption algorithms Ek"(w)

Step d: Split Ek"(w) into two parts <L,R>

Step e: generate pseudo random value Si using stream cipher

Step f: generate key ki using pseudo random function f(.) and and left part of encrypted keyword, ki=f(Li)

Step g: Use ki in the keyed hash function to hash the pseudo random value Si , Yi=<Si, Fki (Si)>

Step h: Encrypt Xi using Yi and XOR operation, Ci = (Xi XOR Yi)

Step iii: Store Ci under the same the same index from where the

plaintext keyword string was extracted.

Step 4: Read the JSON Array for the key value VACCINATION" Step 4.1: Iterate over the strings in the JSON Array

Step 4.1.1: Parse the string for words which are separated by single space

Step 4.1.2: Encrypt each word with deterministic encryption algorithms Ek"(w)

Step 4.1.3: Split Ek"(w) into two parts <L,R>

Step 4.1.4: generate pseudo random value Si using stream cipher

Step 4.1.5: generate key ki using pseudo random function f(.) and and left part of encrypted keyword, ki=f(Li)

Step 4.1.6: Use ki in the keyed hash function to hash the pseudo random value Si Yi=<Si, Fki (Si)>

Step 4.1.7: Encrypt Xi using Yi and XOR operation, Ci = (Xi XOR Yi)

Step 4.2: Store Ci under the same the same index from where the plaintext keyword string was extracted.

Step 5: Read the JSON Array for the key value ALLERGIES" Step 5.1: Iterate over the strings in the JSON Array

Step 5.1.1: Parse the string for words which are separated by single space

Step 5.1.2: Encrypt each word with deterministic encryption algorithms Ek"(w) Step 5.1.3: Split Ek"(w) into two parts <L,R>

Step 5.1.4: generate pseudo random value Si using stream cipher

Step 5.1.5: generate key ki using pseudo random function f(.) and and left part of encrypted keyword, ki=f(Li)

Step 5.1.6: Use ki in the keyed hash function to hash the pseudo random value Si Yi=<Si, Fki (Si)>

Step 5.1.7: Encrypt Xi using Yi and XOR operation, Ci = (Xi XOR Yi)

Step 5.2: Store Ci under the same the same index from where the plaintext keyword string was extracted.

Step 6: Send the secure index keyword file to the server.

5. Trapdoor generation :

The steps followed in the trapdoor generation algorithm are :

Step 1: Read the search query

Step 2 Copy the query into another string and remove the “or” and “and” from the query

Step 3: Parse the search query to store keywords in a array

Step 4: Create trapdoor for phase 1 search

Step 4a: For every keyword in the array, perform:

Step 4.1: Take XOR of the keyword and the symmetric key k" of the patient Step 4b: Store these in temporary trapdoor_phase 1 file

Step 5: Create trapdoor for phase 2 search

Step 5a: For every keyword in the array, perform:

Step 5.1: Encrypt each keyword using deterministic encryption algorithms Step 5b: Store the trapdoors in new temporary trapdoor_phase 2 file

Step 6: Send temporary trapdoor file to server

6. Search Algorithm:

For first version:

Step 1: Read the secure index keyword file which is a JSON Object

Step 3: Read the JSON Array for the key value “ORU”

Step 3.1: Create a boolean array of Size equal to number of trapdoors

Step 3.2: Read array 1 to n, representing each visit

Step I: Create a boolean array of Size equal to number of trapdoors

Step II: Read the fixed size ciphertext

Step i: Read the trapdoors from trapdoor_phase2 file

Step ii: Compare each trapdoor with the fixed sized keyword.

Step iii: If match occurs, make the entry of boolean array as 1 for that respective trapdoor.

Step III: Read the boolean array and place the corresponding boolean entry at the place of that keyword

Step IV: Evaluate the boolean expression, if final answer is 1, that means a match occurs, otherwise match does not occur.

Step V: If result equals to 1

Step i: Place the corresponding visit from the encrypted PHR into another file which will be sent to the patient after all searching is done.

Step 4: Read the JSON Array for the key value "ALLERGIES"

Step 4.1: Create a boolean array of size equal to number of trapdoors

Step 4.2: Read the fixed size ciphertext

Step i: Read the trapdoors from trapdoor_phase2 file

Step ii: Compare each trapdoor with the fixed sized keyword.

Step iii: If match occurs, make the entry of boolean array as 1 for that respective trapdoor.

Step III: Read the boolean array and place the corresponding boolean entry at the place of that keyword

Step IV: Evaluate the boolean expression, if final answer is 1, that means a match occurs, otherwise match does not occur.

Step V: If result equals to 1

Step i: Place the corresponding visit from the encrypted PHR into another file which will be sent to the patient after all searching is done.

Step 5: Read the JSON Array for the key value "VACCINATION"

Step 5.1: Create a boolean array of size equal to number of trapdoors

Step 5.2: Read the fixed size ciphertext

Step i: Read the trapdoors from trapdoor_phase2 file

Step ii: Compare each trapdoor with the fixed sized keyword.

Step iii: If match occurs, make the entry of boolean array as 1 for that respective trapdoor.

Step III: Read the boolean array and place the corresponding boolean entry at the place of that keyword

Step IV: Evaluate the boolean expression, if final answer is 1, that means a match occurs, otherwise match does not occur.

Step V: If result equals to 1

Step i: Place the corresponding visit from the encrypted PHR into another file which will be sent to the patient after all searching is done.

Step 6: Read the JSON Array for the key value "DATA"

Step 6.1:Create a boolean array of size equal to number of trapdoors

Step 6.2: Iterate over the department keys ***dep***

Step 6.3: Read the JSON Object for that department key

Step I: Iterate over the Date keys

Step II: Read the JSON Object for that Date key

Step i: Iterate over the date keys where the value is JSON Array

Step ii: Read the fixed size ciphertext for the above date key

Step a: Read the trapdoors from trapdoor_phase2 file

Step b: Compare each trapdoor with the fixed sized keyword.

Step c: If match occurs, make the entry of boolean array as

1 for that respective trapdoor.

Step iv: Read the boolean array and place the corresponding boolean entry at the place of that keyword

Step v: Evaluate the boolean expression, if final answer is 1, that means a match occurs, otherwise match does not occur.

Step vi: If result equals to 1

Step a: Place the corresponding visit from the encrypted PHR into another file which will be sent to the patient after all searching is done.

For second version

The following steps are performed in the search algorithm:

Step 1: Read the secure index keyword file which is a JSON Object

Step I: Read the temporary_phase1 file

Step i: Read the department values or section values

Step ii: If any trapdoor belongs to "COMMON" section, set flag for COMMON else set flags for other departments or sections

Step 2: Check flag for each department, i.e., Cardiology, Oncology and Cardiology

Step 2a: If flag set for department

Step I: Read linked list of that department, j = 1 to size of ll

Step II: Read encrypted string (represents a visit) for indexes, i.e., ll[j]

Step III: Create a boolean array of Size equal to number of trapdoors

Step IV: Read the fixed size ciphertext

Step i: Read the trapdoors from trapdoor_phase2 file

Step ii: Compare each trapdoor with the fixed sized keyword.

Step iii: If match occurs, make the entry of boolean array as 1 for that respective trapdoor.

Step V: Read the boolean array and place the corresponding boolean entry at the place of that keyword

Step VI: Evaluate the boolean expression, if final answer is 1, that means a match occurs, otherwise match does not occur.

Step VII: If result equals to 1

Step i: Place the corresponding visit from the encrypted PHR into another file which will be sent to the patient after all searching is done.

Step 3: Read the flag for "VACCINATION"

Step 3.1: If flag value set:

Step 3a: Iterate over the JSON Array, for each encrypted string(represents a visit)

Step I: Create a boolean array of size equal to number of trapdoors

Step II: Read the fixed size ciphertext

Step i: Read the trapdoors from trapdoor_phase2 file

Step ii: Compare each trapdoor with the fixed sized keyword.

Step iii: If match occurs, make the entry of boolean array as 1 for that respective trapdoor.

Step III: Read the boolean array and place the correponding boolean entry at the place of that keyword

Step IV: Evaluate the boolean expression, if final answer is 1, that means a match occurs, otherwise match does not occur.

Step V: If result equals to 1

Step i: Place the corresponding visit from the encrypted PHR into another file which will be sent to the patient after all searching is done.

Step 3.2: If flag value not set, skip this section

Step 4: Read the flag for "ALLERGIES"

Step 4.1: If flag value set :

Step 4a: Iterate over the JSON Array, for each encrypted string(represents a visit) :

Step I: Create a boolean array of size equal to number of trapdoors

Step II: Read the fixed size ciphertext

Step i: Read the trapdoors from trapdoor_phase2 file

Step ii: Compare each trapdoor with the fixed sized keyword.

Step iii: If match occurs, make the entry of boolean array as 1 for that respective trapdoor.

Step III: Read the boolean array and place the corresponding boolean entry at the place of that keyword

Step IV: Evaluate the boolean expression, if final answer is 1, that means a match occurs, otherwise match does not occur.

Step V: If result equals to 1

Step i: Place the corresponding visit from the encrypted PHR into another file which will be sent to the patient after all searching is done.

Step 4.2 : If flag value not set, skip this section

CHAPTER 6 – IMPLEMENTATION

This chapter provides the implementation details of the proposed sectional access control on the PHR and efficient seachability on encrypted health record. The detailed explanation can be divided into three sections. The first section provides a brief description of the libraries and platforms used and the second section discusses about the implementation of the application developed for searching in the encrypted PHR by the medical professional.

6.1 Brief Description

The proposed application for sectional access control and efficient search on encrypted PHR with conjunctive keyword search is implemented using JavaScript, C, JAVA and JAVA EE.

JavaScript is used in the searchable encryption library which is used to implement the symmetric searchable scheme proposed by Song et al. Modifications are done in this library to support conjunctive keyword search and to provide searchability on the keywords of the document instead of the original document.

C is used in the Ciphertext-Policy Attribute Based Encryption library, the scheme which is proposed by Brethencourt et al. [4]. This library is used to provide fine-grained access control.

JAVA is used to parse the Patient Health Record (PHR) and pass the visits, sections and their corresponding access policies to the encryptor which encrypts it using CP-ABE. It is also used to parse the keyword file containing the keywords for the corresponding visits and sections. The keywords for the visits/section are encrypted to create the secure-index. Java is also used to create the hash table on the unique keywords per department/section using the symmetric key of the patient.

Eclipse IDE is used for:

1. Designing the interface of the proposed application
2. Dispatching the keyword query to the server
3. Downloading the retrieved documents

4. Displaying the final Retrieved records
5. comparing the search time for different keywords queried
6. comparing the value of vitals in the retrieved records

The JAR (Java Archive) files used in the development of the proposed application are:

(f) *JSON Simple 1.1.1* which helps to read and parse the JSON file.

(g) *Org Apache Commons FileUpload* to allow file to be uploaded to the server and downloaded from the server.

(h) *Nodejs* used to run the JavaScript files.

6.2 Implementation

The Patient, who is the owner of the PHR uses his/her symmetric key to encrypt the keyword file to generate the secure-index and also to create the hash on the unique keywords.

The Medical Professional sends the keyword search query to the patient. He presents the profession he has, along with the keyword query. Figure 21 shows the interface in which the medical professional enters the query.

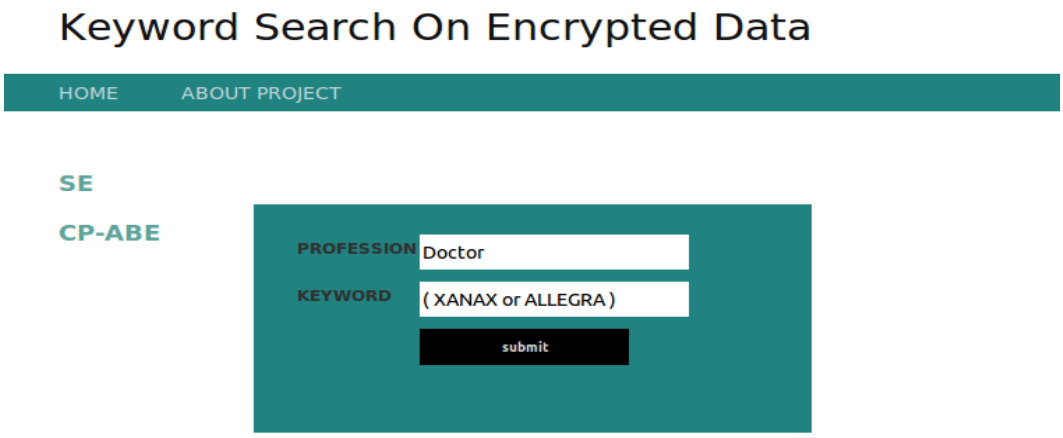


Figure 21: Query Submission Phase

After the retrieval of search records the medical professional can download the encrypted retrieved records and decrypt with the decryption key generated by the Trusted Authority based on his/her attributes.

Figure 22: Download retrieved records after search

Only those records will be decrypted for which the medical professional's attributes satisfy the access policy of the encrypted records. After decryption, the medical professional can see the ORU containing the HL7 data and other information by clicking on the ORU button. Also, the Department wise visits can be seen in the Data Section, where each visit contains the concept id's, depicting the value of vital, who was the observer, etc.

ORU Button:

After clicking the ORU button, the list of dates contained in the retrieved records is displayed. The Medical Professional can click the required date to see the contents of that date

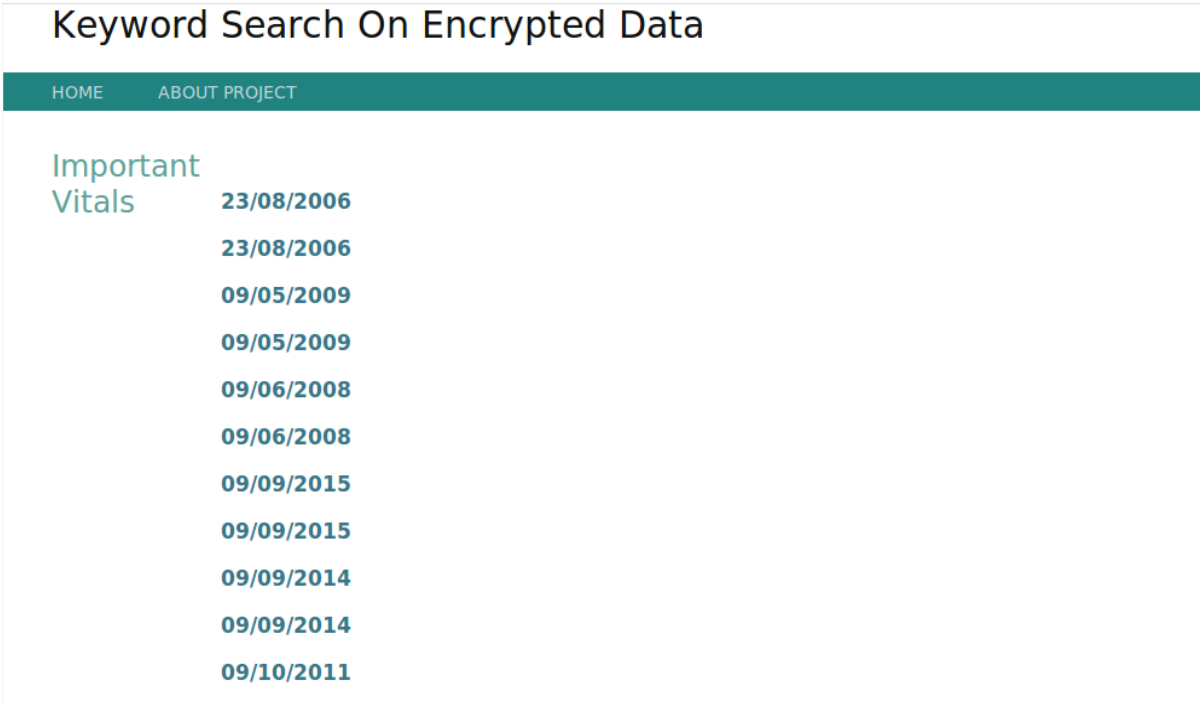


Figure 23: Display all retrieved ORU's

On Clicking a certain date, suppose, we click on the first date, i.e., 23/08/2006, the following information will be displayed.

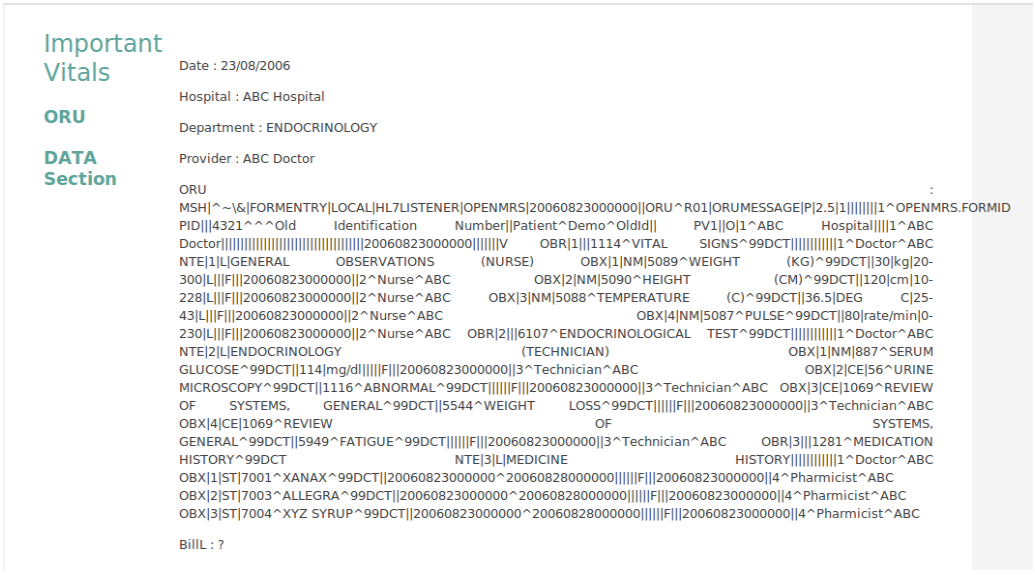


Figure 24: Display details of ORU for a date

Data Section Button:

Figure 6.5: Display of different departments in the Data Section

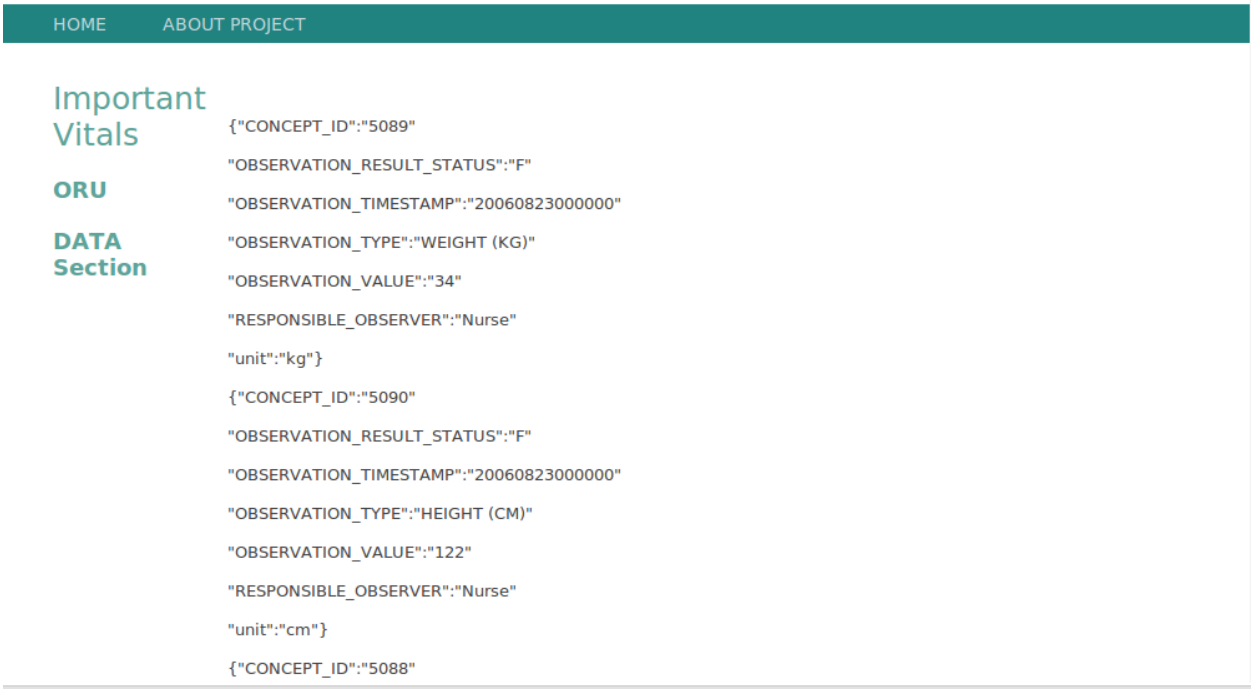


Figure 25: Content of particular date in Data Section

Vitals Display

On clicking other vitals, on can compare the observation values of that vitals, in the retrieved visits. For example, if I want to the pulse observation value for the retrieved records for that patient, then medical professional can click on the Pulse Button to display the values. Following figure shows how the values are depicted graphically, to make the visualization easier for the medical professional.

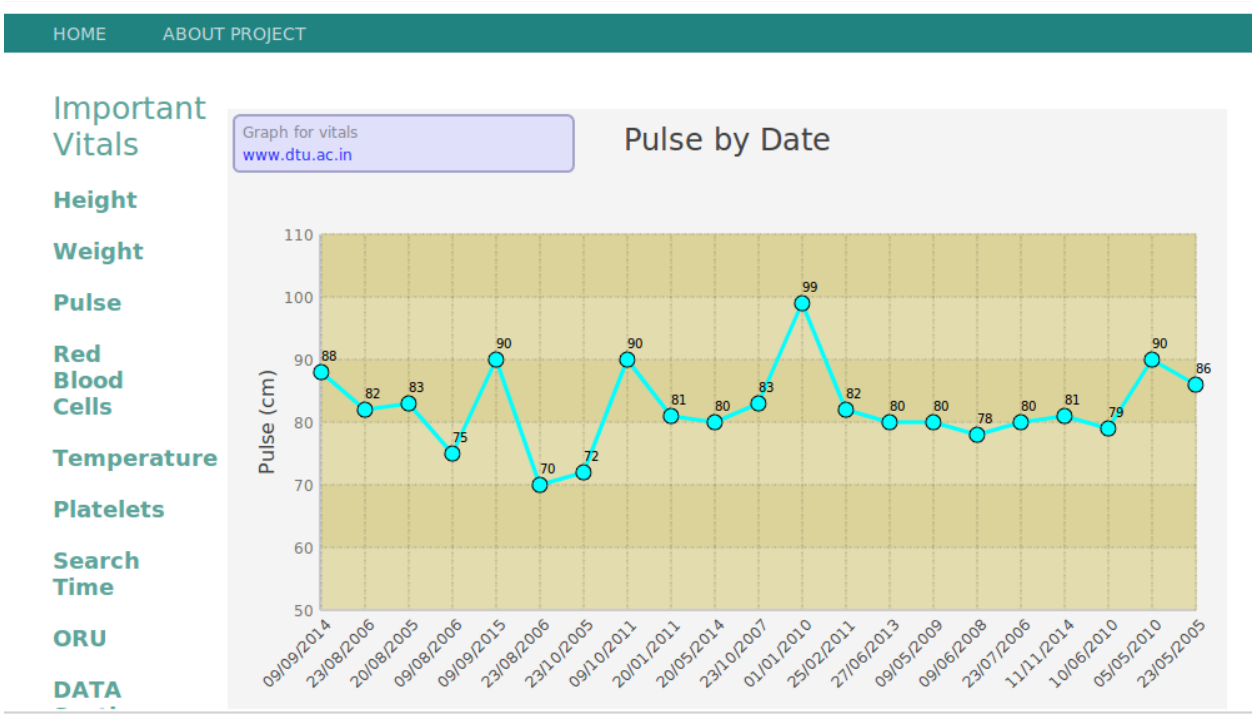


Figure 26: Graph for vitals' observation value and date

CHAPTER 7 – RESULTS

The results of the implementation of the proposed application indicates that the two techniques, CPABE combined with symmetric searchable encryption provides fine-grained access control, along with this, efficiently retrieves encrypted records based on a search query. It is also capable of searching conjunctive queries (containing boolean '*and'* and '*or'*). The second version of implementation also gives improved results compared to the first version on the keywords which are specific to some department or to keywords which are specific to some section like, vaccinations, allergies etc.

The below table 3 shows the comparison in search time in the case when we first decrypt the whole PHR and then do the keyword search versus our technique where we first search on encrypted PHR and then do the decryption. Since decryption of whole PHR takes more time, because of the pairing operations involved, our schemes saves time to decrypt whole PHR and will decrypt only those records which contain the keyword.

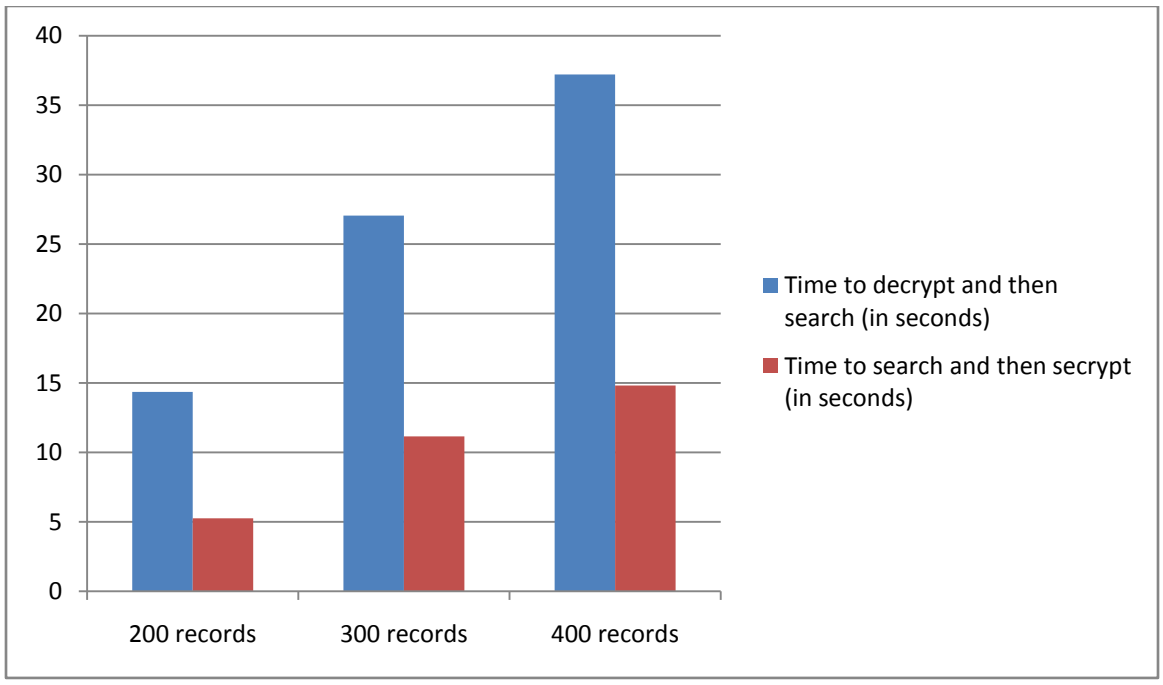


Figure 27: Comparison of decrypt then search records vs search then decrypt records

No. of Records	Keyword Searched	Time to decrypt and then search (seconds)	Time to search and then decrypt (seconds)
200	ALLEGRA	14.35	5.25
300	ALLEGRA	27.05	11.16
400	ALLEGRA	37.21	14.81

Table 3: Comparison of decrypt then search records vs search then decrypt records

The search time is compared for different queries by increasing the size of the query, i.e., more number of keywords in query. This comparison is done using the first version of the implementation which does linear scan on all the ciphertext. Figure 28 this shows the query and search time to retrieve records based on the query.

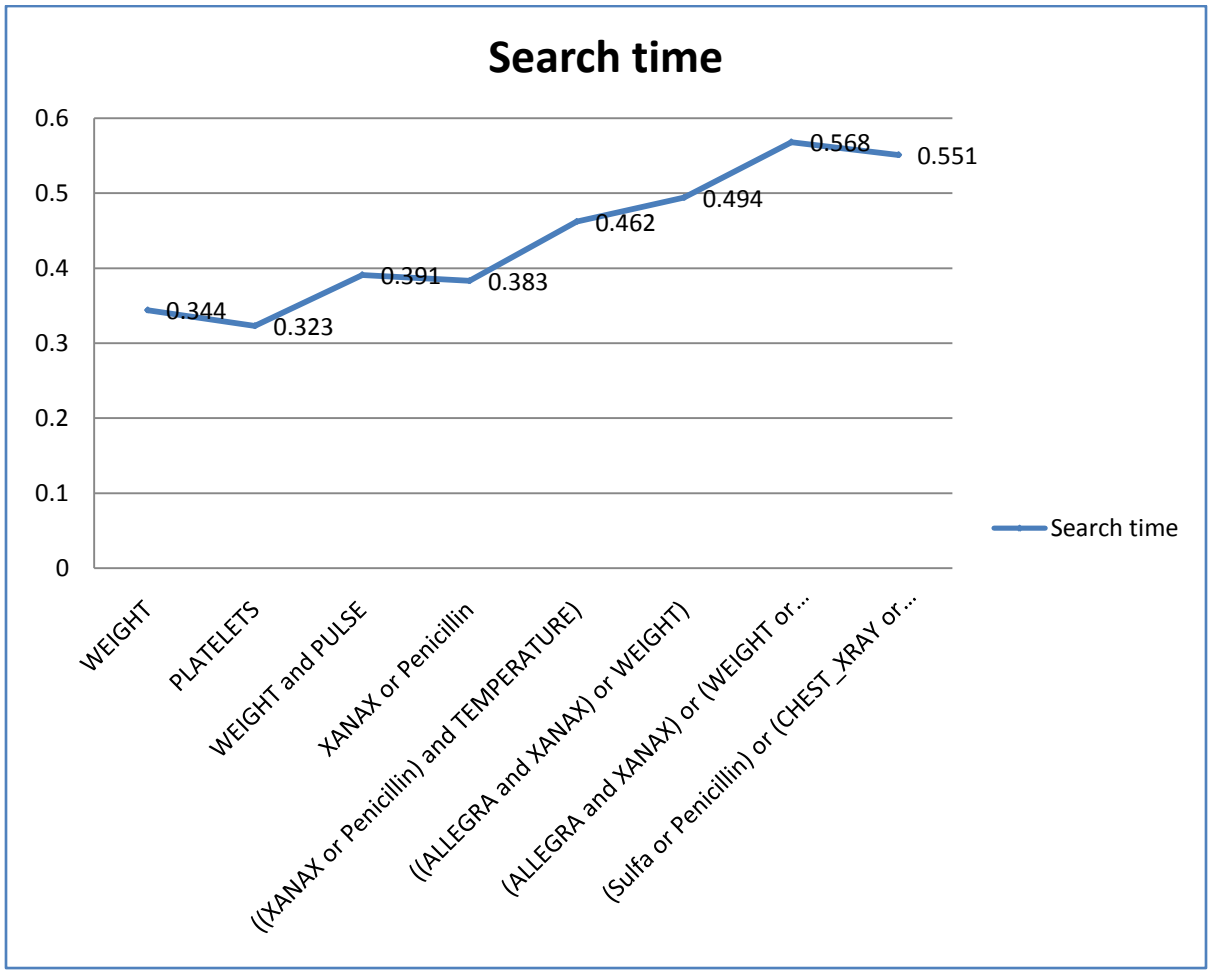


Figure 28: comparison on search time based on size of query

The search time and decryption time is shown in the same graph for some keyword queries. This graph is also displayed in the proposed application. It store the keywords searched and stores

their search time and decryption time. The decryption time is different because different number of records and sections are retrieved based on the query.

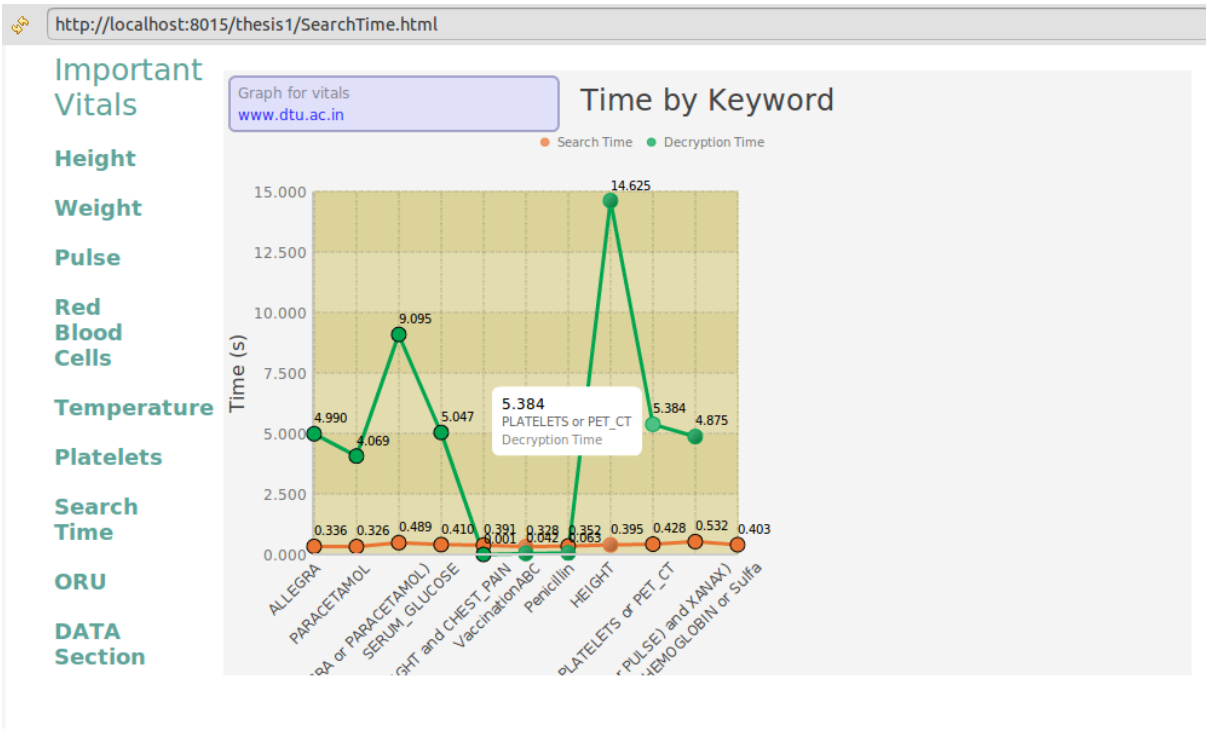


Figure 29: Comparison of search time and decryption time for different keyword query

The graph depicts that by increasing the number of keywords, the search time increases, as we have to match every cipher word with the all the keyword trapdoor. Therefore, the search time increases with increasing the size of query.

The search time is also compared by changing the number of keywords per record. The results show that as we increase the number of keywords per record the search time increases.

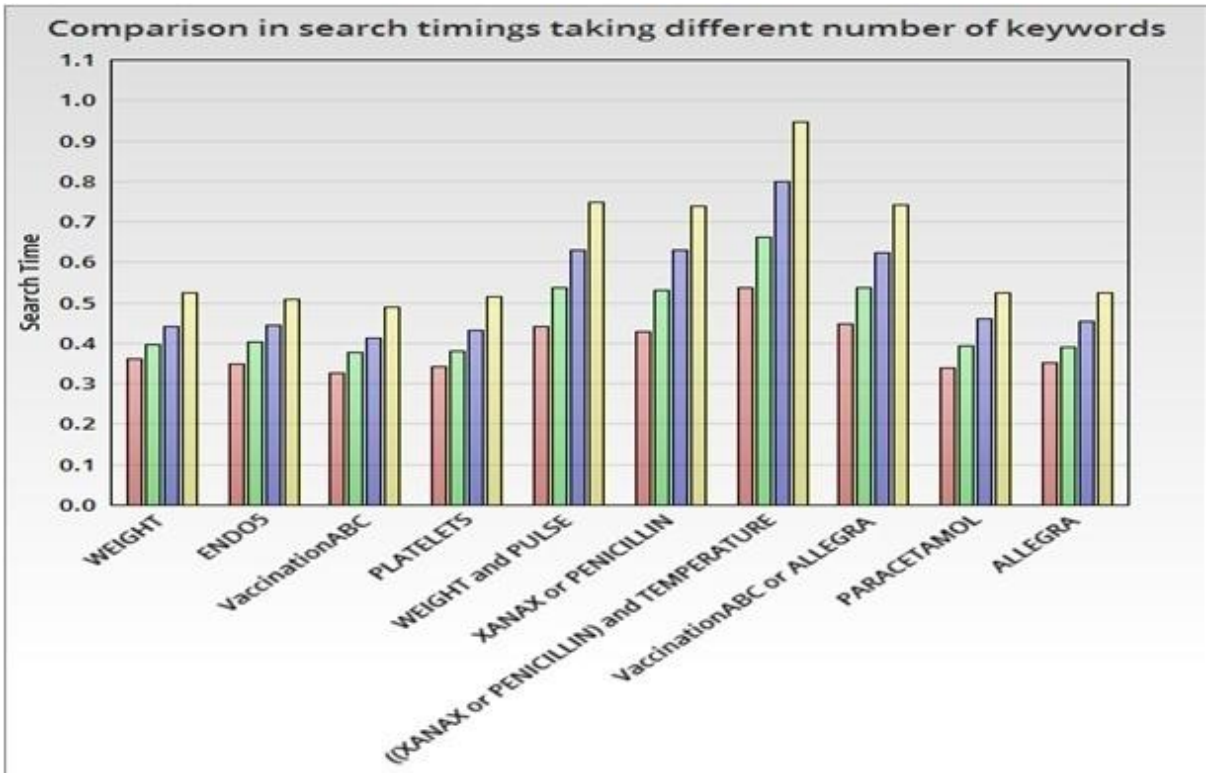


Figure 30: Comparison in search with different number of keywords

The comparison between the timings of the two versions is depicted in the figure 6.3. The graph shows that the search time is reduced for the second version for the unique keywords and remains almost same for the common keywords which exist in all the departments and other sections.

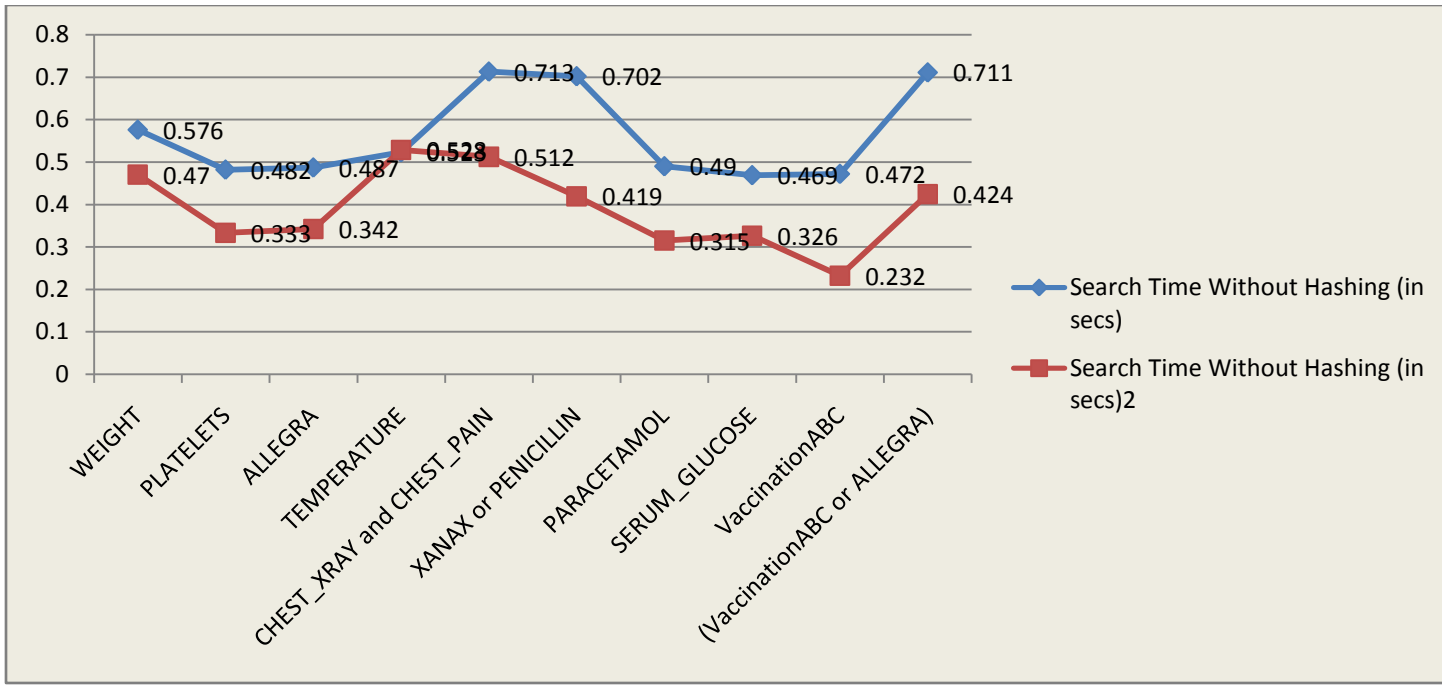


Figure 31: Comparison of search timings for two different versions

If the number of records are more, the search time can be reduced significantly using the second version. The comparison in search time is calculated by increasing the number of records from 200, 300 and 400 records.

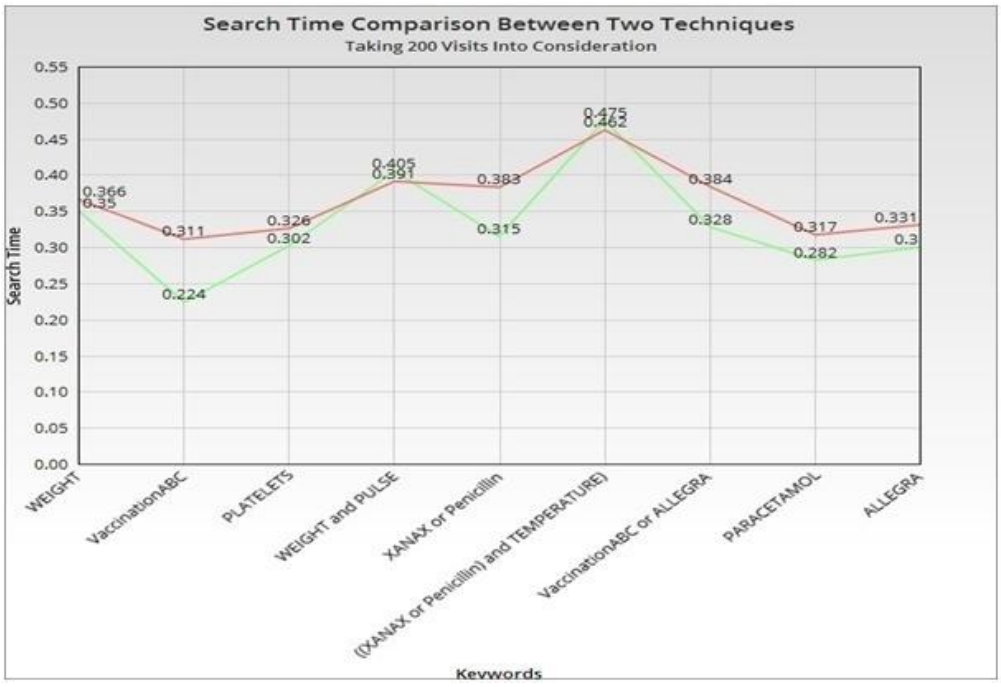


Figure 32: Search time comparison with 200 records in the PHR

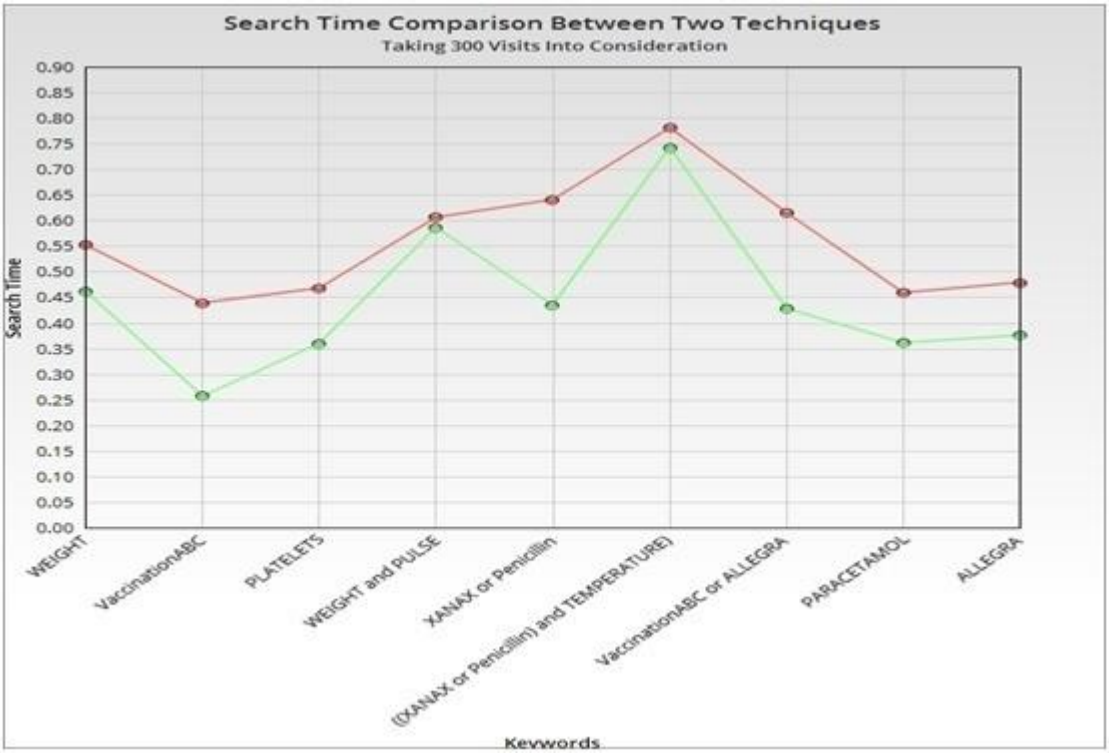


Figure 33: Search time comparsion with 300 records in the PHR

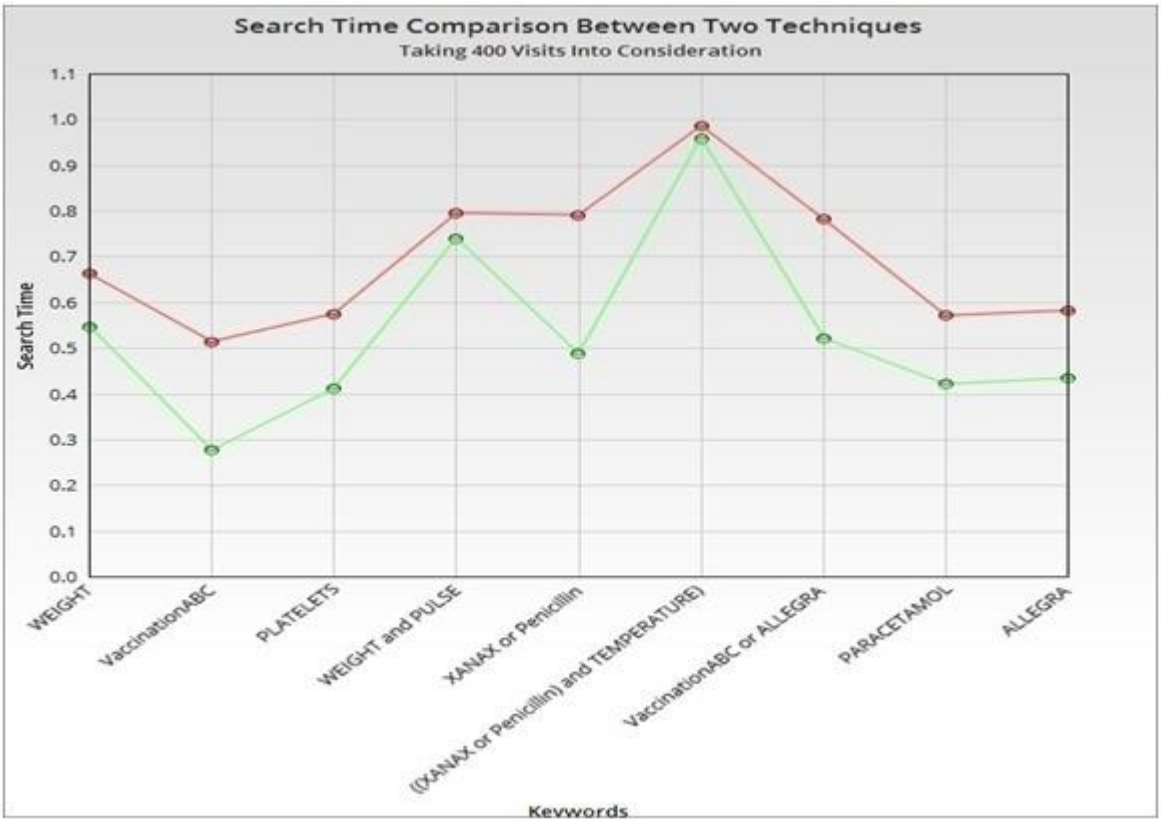


Figure 34: Comparsion of search time with 400 records in the PHR

CHAPTER 8 – CONCLUSION AND FUTURE WORK

A secure access control strategy is very helpful in protecting the user's PHR that is threatened from attackers. A searchable ciphertext access control scheme is needed where people possessing attributes can search on the ciphertext for keywords and decrypt them. A combination of CP-ABE and symmetric searchable encryption scheme is proposed for the PHR's of the patients, where the patients encrypts his PHR and medical professional can search on encrypted PHR and download only required records instead of downloading, decrypting whole PHR and then performing the search on the PHR for required records and sections. A section-wise encryption is done to provide access on records based on their role and data contained in the records. This would help flexible access control per record. The conjunctive keyword search also provides the medical professionals to make a wide range of query on the keywords.

Through implementation and testing we can see that the proposed schemes successfully retrieves the records after searching. The second version of the proposed scheme further decreases the search time for unique keywords belonging to the specific departments and sections. Finally, we provide the comparison of the search time for two versions and our proposed scheme ensures security using CP-ABE on sections/records and encrypted records retrieval using conjunctive keyword search which in turn, reduces the overall retrieval time for required records.

We plan to perform the encryption and searchability on the mobile device, i.e., mobile phone, of the user. Since the PHR could be of use anytime, so there's a necessity that the patient carries his encrypted records within his mobile phone. Using NFC, the keyword query of the medical professional could be sent to the patient and the patient's device would perform the search operation. And with taping of the medical professional's device and patient's device, the retrieved encrypted records would be transferred to the medical professional's device. Further the decryption would take place on medical professional's device, if his attributes satisfy the access structure of ciphertext.

REFERENCES

[1] Sahai, Amit, and Brent Waters. "Fuzzy identity-based encryption." In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 457-473. Springer Berlin Heidelberg, 2005..

[2] Boneh, Dan, and Matt Franklin. "Identity-based encryption from the Weil pairing." In *Annual International Cryptology Conference*, pp. 213-229. Springer Berlin Heidelberg, 2001.

[3] Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute Based Encryption for Fine-Grained Access Control of Encrypted Data. In: ACM Conference on Computer and Communications Security, pp. 89-98 (2006)

[4] Bethencourt, J., Sahai, A., Waters, B.: Ciphertext Policy Attribute Based Encryption. In: IEEE Symposium on Security and Privacy, pp. 321-334 (2007)

[5] Song D., Wagner D., Perrig A.: Pratical Techniques for Searches on Encrypted Data. In: Proceeding IEEE Symposium Security and Privacy, 2000

[6] Goh E-J: Secure Indexes. In: Technical Report 2003/216, IACR ePrint Cryptography Archive 2003

[7] Chang, Yan-Cheng, and Michael Mitzenmacher. "Privacy preserving keyword searches on remote encrypted data." In *International Conference on Applied Cryptography and Network Security*, pp. 442-455. Springer Berlin Heidelberg, 2005.

[8] Curtmola, Reza, Juan Garay, Seny Kamara, and Rafail Ostrovsky. "Searchable symmetric encryption: improved definitions and efficient constructions." *Journal of Computer Security* 19, no. 5 (2011): 895-934.

[9] Boneh, Dan, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. "Public key encryption with keyword search." In *International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 506-522. Springer Berlin Heidelberg, 2004.

[10] Li, Ming, Shucheng Yu, Ning Cao, and Wenjing Lou. "Authorized private keyword search over encrypted data in cloud computing." In *Distributed Computing Systems (ICDCS), 2011 31st International Conference on*, pp. 383-392. IEEE, 2011.

[9] Li J., Zhang L.: Attribute-based keyword search and data access control in cloud. In Proceedings of International Conference on Computational Intelligence and Security, pp. 382-386, 2014

[10] Wang C., Li W., Li Y., Xu X.: A ciphertext policy attribute-based encryption scheme supporting keyword search function. In proceedings of Cyberspace Safety and Security, LNCS 8300, Springer, pp. 377-386, 2013

[11] Jonathan Katz, Amit Sahai, and Brent Waters. 2008. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In EUROCRYPT (LNCS), Vol. 4965. 146–162

[12] Xiong, A. P., Gan, Q. X., He, X. X., & Zhao, Q. (2013, December). A searchable encryption of CP-ABE scheme in cloud storage. In *Wavelet Active Media Technology and Information Processing (ICCWAMTIP), 2013 10th International Computer Conference on* (pp. 345-349). IEEE.

[13] Burton H. Bloom. 1970. Space/time trade-offs in hash coding with allowable errors. Communications of the ACM 13, 7 (1970), 422–426

[14] Melissa Chase and Seny Kamara. 2010. Structured encryption and controlled disclosure. In ASIACRYPT (LNCS), Vol. 6477. Springer, 577–594. Available at http://dblp.uni-trier.de/db/conf/asiacrypt/ asiacrypt2010.html#ChaseK10

[15] Seny Kamara, Charalampos Papamanthou, and Tom Roeder. 2012. Dynamic searchable symmetric encryption. In CCS. ACM, 965–976

[16] Boneh, D., & Franklin, M. (2001, August). Identity-based encryption from the Weil pairing. In *Annual International Cryptology Conference* (pp. 213-229). Springer Berlin Heidelberg.

[17] Boneh, D., & Franklin, M. (2003). Identity-based encryption from the Weil pairing. *SIAM journal on computing*, *32*(3), 586-615.

[18] Yang, Y. (2015). Attribute-based data retrieval with semantic keyword search for e-health cloud. *Journal of Cloud Computing*, *4*(1), 1.

[19] Narayan, Shivaramakrishnan, Martin Gagné, and Reihaneh Safavi-Naini. "Privacy preserving EHR system using attribute-based infrastructure." In*Proceedings of the 2010 ACM workshop on Cloud computing security workshop*, pp. 47-52. ACM, 2010.

[20] Ramu, Gandikota, and B. Eswara Reddy. "Secure architecture to manage EHR's in cloud using SSE and ABE." *Health and Technology* 5, no. 3-4 (2015): 195-205.

[21] Han, Fei, Jing Qin, Huawei Zhao, and Jiankun Hu. "A general transformation from KP-ABE to searchable encryption." *Future Generation Computer Systems* 30 (2014): 107-115.

[22] Taher ElGamal. 1985. A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Transactions on Information Theory 31, 4 (1985), 469–472.

[23] Pascal Paillier. 1999. Public-Key cryptosystems based on composite degree residuosity classes. In EUROCRYPT (LNCS), Vol. 1592. Springer, 223–238

[24] Adi Shamir. 1985. Identity-Based cryptosystems and signature schemes. In CRYPTO (LNCS), Vol. 196. Springer, 47–53

[25] Li, Jiazhi, and Lei Zhang. "Attribute-based keyword search and data access control in cloud." In *Computational Intelligence and Security (CIS), 2014 Tenth International Conference on*, pp. 382-386. IEEE, 2014.

[26] Cao, Ning, Cong Wang, Ming Li, Kui Ren, and Wenjing Lou. "Privacy-preserving multi-keyword ranked search over encrypted cloud data." *IEEE Transactions on parallel and distributed systems* 25, no. 1 (2014): 222-233.

[27] Sun, Wenhai, Shucheng Yu, Wenjing Lou, Y. Thomas Hou, and Hui Li. "Protecting your right: Attribute-based keyword search with fine-grained owner-enforced search authorization in the cloud." In *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*, pp. 226-234. IEEE, 2014.