35th Annual International Conference of the IEEE EMBS
Osaka, Japan, 3 - 7 July, 2013

Protection of Electronic Health Records (EHRs) in Cloud

Abdulatif Alabdulatif, Ibrahim Khalil, Vu Mai
School of Computer Science and Information Technology
RMIT university

***Abstract*—EHR technology has come into widespread use and has attracted attention in healthcare institutions as well as in research. Cloud services are used to build efficient EHR systems and obtain the greatest benefits of EHR implementation. Many issues relating to building an ideal EHR system in the cloud, especially the tradeoff between flexibility and security, have recently surfaced. The privacy of patient records in cloud platforms is still a point of contention. In this research, we are going to improve the management of access control by restricting participants' access through the use of distinct encrypted parameters for each participant in the cloud-based database. Also, we implement and improve an existing secure index search algorithm to enhance the efficiency of information control and flow through a cloud-based EHR system. At the final stage, we contribute to the design of reliable, flexible and secure access control, enabling quick access to EHR information.**

I. Introduction

Designing an access control model for encrypted EHRs in the cloud relies mainly on various aspects, including the encryption scheme, the key management mechanism of encrypted EHRs and the natural flow of communication between the different participants. Since EHRs systems have many participants with different security clearances who must have frequent access to the system, there is a need to apply an efficient and reliable hierarchical structure to manage the participants and to control the data flow inside the systems. Our access control design depends fully on hierarchical key management for the encryption/decryption keys.

A great deal of research has focused on hierarchical key management structure. In a first attempt Akl and Taylor [1] suggest a model based on cryptography to arrange the data in a hierarchical structure. This model has a master encryption/decryption key on which all the lower level keys depend on it. In this case, if any key in a lower level changes, all the other keys must be regenerated. One advantage of this model is that the master key holder does not need to keep all keys under his master key since all of them are derived from it. However, it costs time and overhead when the owner of the keys wishes to change any key in the hierarchy.The approach of [2] is based on top-down key generation, and Harn and Lin's [3] is based on the inverse (bottom-up) approach. All of these suggestions have a specific drawback: the relationships between the levels of the hierarchy are deep, and storage space increases depending on the depth of the hierarchy. In 2003, Lin et al. [4] suggested an access control model which has two keys in each security class: a secret key and a derived key and which describes the relationship between security levels in hierarchical structure.

Generally speaking, Kayem et al. [5] illustrate clearly that most encrypted access control in a hierarchy can only be achieved by one of two main models, a dependent key model or an independent key model. The dependent key model is described by the relationship between the master key at the top of the hierarchical structure and the lower level keys, which are explicitly derived from the master key. Thus, any change in lower level keys will lead to a change in all the keys in the structure because they are dependent on each other [5]. In an independent key model the lower level keys do not depend on the master key; they are generated separately without any relationship that links them together [5]. In addition, the lower level keys are independent of each other. The main differences between these two models include processing overhead, the amount of time consumed and storage capacity [5]. In our design we will combine both the dependent and independent key models to take advantage of both models' benefits while avoiding their disadvantages. We will show how can we apply the new key management model in our EHR system. This system as shown in Fig. 1 will provide EHR owners with full control over their records. An authorized third party will be responsible for authenticating the participants who will interact inside the system.

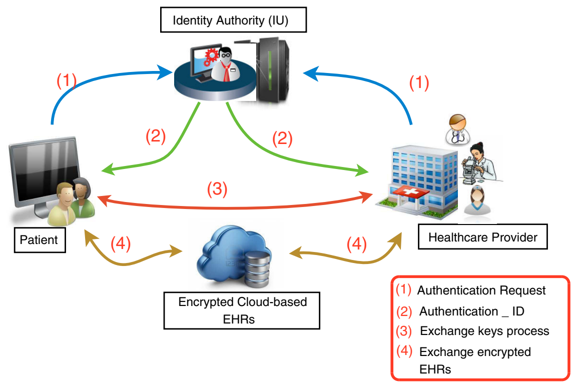
Identity Authority (IU)
(1)
(1)
(2)
(2)
(3)
Patient
(4)
(4)
Healthcare Provider
Encrypted Cloud-based EHRs
(1) Authentication Request
(2) Authentication _ ID
(3) Exchange keys process
(4) Exchange encrypted EHRs

Fig. 1. System architecture overview

In this study, we propose a new communication mechanism to serve as an integrated system between the patient and the service provider. The role of the different parts in the generation of the encryption/decryption key, the management of communications and the exchange of information will be shown in detail. We introduce the work flow and the main participants in the EHR system in Section II. The proposed key management scheme is described in Section III. We present the implementation of an existing search tool as part of the access control model for our system as well as improvement options and ways they might be applied to the proposed system in Section IV.

II. System architecture

Our access control model design depends on three main participants involved in the system life cycle Fig. 1. These participants interact with each other in direct and indirect ways to perform different tasks in the cloud-based EHR system. The participants are described as follows:

A. *Administrator or Identity Authority (IU)*

We introduce the administrator as a trusted third party, such as a government institution. This participant in the system is responsible for two main functions. The administrator (1) authenticates all participants who interact with the system and (2) serves as a point-of-contact between the patients and the healthcare providers by providing the encryption/decryption keys thought a SSL connection.

B. *The End User or Patient*

The patient is the main stakeholder and participant in our system and drives our objectives. The end user (or patient) has two main responsibilities: (1) A new user must send an authentication request to the IU to get an identification number (ID) to be able to use the system services, and (2) The patient must make sure that his computer uses encryption/decryption. It must be fully secure and protected from the outside world to maintain the security of the keys, the key-generating functions or any related information that might lead to leaks of private information or threats to the security of the system.

C. *Healthcare Providers*

In our system, we assume that there are many healthcare providers who will need to interact with different parts of the system. Therefore, we also assign a user ID to the healthcare provider. This information will help the system to validate the identity of the members and grant them the appropriate permission to access specific patient records. Each healthcare provider has three main tasks: (1) Each healthcare provider must apply for an authentication request to the IU to obtain legitimate access to use the system, (2) Healthcare providers must request the key for a specific patient EHR from the IU and must provide appropriate parameters in the request and, (3) The provider may decrypt and update that information but must re-encrypt the document with the same key before sending it back to the cloud database.

In real-world systems, healthcare institutions have at least the following members: physicians, nurses, laboratory workers and other employees. Each member of these groups must have access to some part of the patient records for specific purposes. Therefore, we have developed a hierarchical structure to provide different access permission to providers according to they need to access. Fig. 2 describes a proposed structure for organizing a patient's records in the cloud.

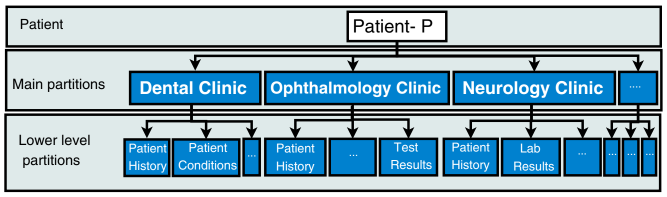

Fig. 2. Proposed structure for organizing patients records in the cloud

D. *The cloud database*

The cloud database is the backbone of our system. It is a central data Center for holding and organizing patient EHRs. Requesters must provide all information related to their access rights as well as the requested documents. Fig. 2 shows that our cloud database is designed in a hierarchical manner. This structure will help us to provide an efficient key management mechanism and enables secure distribution of encryption/decryption keys to participants. Since each participant in the system should be able to access only specific documents rather than all patient records, we will specify each participant's permission by setting different encryption/decryption keys for each different group of documents, depending on who should be able to have access to those documents.

In order for access permission to be granted, each document that is produced for any participant must include patient details, healthcare provider details, document type (e.g. personal, diagnosis or lab), category (e.g. dental) and document publisher (e.g. patient or physician). This information, together with the authentication parameters which prove the eligibility of the applicant to upload and download EHRs, will be used to grant access to different EHRs in the cloud database.

III. Document structure and key management

The key management scheme depends completely on how the documents are structured inside the cloud database. As we have many different participants who have to access to the database, we implement a hierarchical structure that limits access permissions for different participants based on the need-to-know principle. The hierarchical structure is shown in Fig. 2 keeps all records for any patient under the Patient section, Since the database is in the cloud, patients have the ability to access their records from different places and at any time.

The patient is responsible for managing the keys of his or her documents. The main partitions for each patient will be labelled as Dental clinic records, Ophthalmology clinic Records, Neurology clinic Records and other medical specialties. These are directly located under each patient's section (see Fig. 2); there will be up to eight partitions in most cases. These main partitions of patient documents will have an independent key model to manage their keys. Even though it is the patients responsibility to manage their own keys, we will use a dependent key model to manage the keys under each main partition of the patient documents because we assume that they will used frequently by different healthcare workers. Thus, all lower level keys belonging to different main partitions need to change regularly to ensure the security of records. As each patient's records have an individual hierarchical structure, they will be managed using both an independent and a dependent key model in our design.

*1) **Independent key model**:* The model [5] depends on different distributed keys for all main partitions of the patient documents. The patient (P) has a group of main partitions of his/her records, (S1,..,Sn).

$$P = \{S1, S2, S3, ..., Sn\} \qquad where \quad n \approx 8 \tag{1}$$

All partitions of group (P) will have independent keys (S1k,..,Sik) under patient control (P) So.

$$P = \{S1k, S2k, S3k, ..., Sik\} \qquad where \quad i \approx 8 \tag{2}$$

which are fully independent from each other, and no one can derive any key depending on another key from the same level. The structure for this model is shown in Fig. 3.

The idea behind using an independent key model for the main partitions is that each key will be independent and capable of being regenerated for any partition (Sn) without affecting any other partition under the main partition (P). Moreover, applying this model has additional advantages and disadvantages. First, the main advantage is the independence of the main partitions keys (S1k,..,Sik). Second, the security level is enhanced because there is no relationship between partition keys. Finally, the time and overhead required to regenerate one of the keys is reduced because they are fully independent from each other. However, a disadvantage of using an independent key model in this case is that the patient has to keep all partition keys (S1k...Sik) secure, which is accomplished at a cost.

*2) **Dependent key model**:* This model [5] generates related keys in hierarchical form depending on the master key. We will use this model to generate lower level keys under each main partition. For each main partition (Sn), there are lower levels uniquely identified by using identifiers (C1,.., Cm) where:

$$Sn = \{C1, C2, C3, ..., Cm\} \qquad where \quad m \approx 5 \tag{3}$$

Each lower level key (Cik) is derived from its main partition key by using a one-way hash function to combine its identifier with the main partition master key (Sik). For example, we can generate a lower level key C1k that belongs to the main partition S1 as follows:

$$C1k = f(S1k, C1) \tag{4}$$

Under each main partition (Sn) are lower levels (C1, Cm) which are derived from one master key (Sik). Each lower level has different documents and the need-to-know principle is applied to determine each lower level permission. The generated keys will be granted depending on the request type and participant permissions. The relationship between each main partition key and its lower levels' keys is a partial relation:

$$Cik \leq Sik \tag{5}$$

This shows any participant with security permission Sik is able to access all documents under that main partition which have lower security permission than Sik (i.e., Cik). Using a dependent key model at this level of our system has some advantages and disadvantages. In practical situations, the main partitions and lower levels belonging to them are accessed frequently by many participants. Thus, it is beneficial to regenerate the whole partition group's keys regularly since they are dependent on each other. The main partition (Sn) has the ability to check and manipulate all lower levels (C1,..,Cm) under it because the dependent key model applies the partial relation between the master Sik key of lower level keys (C1k,..,Cik). The final key management structure is shown in Fig. 3.

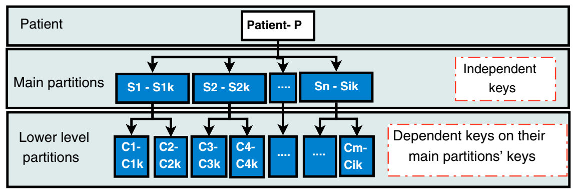

Fig. 3. Dependent and independent key management structure

The subposet key management (SPKM) scheme [5] is used as a part of our key management scheme to derive lower level keys from their main partition keys. The SPKM scheme is extended from the Akl and Taylor scheme [1] and it decreases key update cost by reducing the number of keys that need to be replaced and by eliminating collusion.

IV. Search Algorithm Design (Search Tool)

To address cloud-based EHR privacy and security issues, we must consider how we can encrypt and maintain EHRs in the cloud. Moreover, we must take into account that there will be millions of records in the cloud and that many stake-holders will need access to these records frequently for many purposes. Therefore, an EHR encryption algorithm must not only enhance privacy in the cloud, but also be fast, reliable and enable efficient searching. In addition to that, the impact of any untrusted server in the cloud participating in the system must be considered: how much information should that server know about the system and the encrypted records? Solutions provide both secure encryption and encrypted search algorithms. One of these, proposed by Eu-Jin Goh [6], uses a secure pre-processed index for a given set of documents and keeps some credentials in a specific bloom filter [7] as a part of the search process for the encrypted documents.

*A. **Algorithm design***

Goh's secure index scheme [6] depends on the bloom filter structure [7] to store different random hash values for the words in any given document; these values are used to check the availability of the words using the generated bloom filter and the actual document is not needed. This helps to encrypt the plain-text documents and upload them to the cloud with the bloom filter. Both the encrypted documents and the bloom filter will be meaningless should any unauthorized user be able to access them in the cloud. On the other hand, authorized participants will be able to search and retrieve documents in a secure way.

B. Secure index implementation

The secure index is implemented in three stages. First, the bloom filter is generated and contains random hash values corresponding to the words in a given document. Second, there is a trapdoor for each word in any given document that is handed over later to the untrusted server to help find possible matches without providing the plain-text data. Third, the actual plain text is encrypted using the selected encryption algorithm and is passed together with the bloom filter to an untrusted server to be available in the cloud.

1) ***Bloom filter:*** The bloom filter consists of an array of bits which are initially all set to '0. Any new element is added to the bloom filter set after the number of hashes are performed. The output of each hash operation is added to the index by setting the position of that hash value to '1.

2) ***Encryption:*** The implementation of the encryption process relies on generating a bloom filter [7] as an index for a given document by using a Build Index function BuildIndex (Di, K master). The index will be used to generate the bloom filter for a document (Di) with the given master key. Thus, by getting the document and the master key, the trusted client splits the document into separate words. Then, for each word (Wi) in that document we follow these steps:

- Generate a trapdoor for Wi using the master key and suitable hash function:

$$Tw = \{fk1(Wi), fk2(Wi), fk3(Wi), fk4(Wi)\} \quad (6)$$

  where (i) is the word number in a document.
- Generate a codeword (Ci) based on the trapdoor (Ti) by taking every value of the trapdoor and hashing it with Di and then assigning the resulting values to the Bloom filter as values between 0 and 2047. We had specified the Bloom filter array size in our implementation to 2048.

$$Tw = \{Ci = fDi(Tw), fDi(Tw), fDi(Tw), fDi(Tw)\} \quad (7)$$

- Add code word values to the bloom filter to represent the index of the document. Then encrypt the plain-text document using Data Encryption Standard (DES).

3) ***Search:*** When a trusted client (e.g. a patient) wants to search for a specific word or multiple words, he/she has to generate a trapdoor for each word separately as previously explained and hand it to an untrusted server with the document identifier Di. Then, the untrusted server implements an iterative operation for all the documents it stores. It will generate the codeword for a given word; the bloom filter corresponding to a given document identifier will be checked to see if the generated codeword is a member in that bloom filter or not, and the results will be shown. Fig. 5 shows a general overview of secure index algorithm implementation.

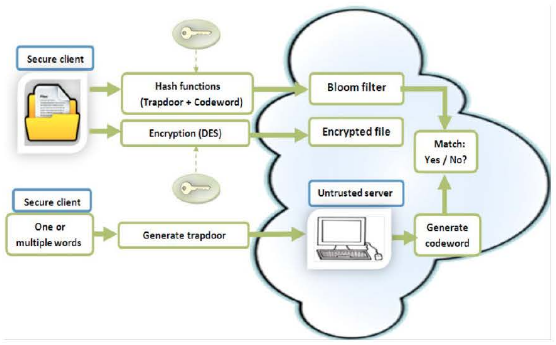

Fig. 4. Secure index algorithm implementation

We have used Java programming language to implement our proposed architecture. Java supports a wide range of security libraries that made our implementation easier. Java security libraries are used for initial user authentication, and also the encryption mechanism used to generate word trapdoors. In order to generate word trapdoors, we have used a strong hashing algorithm SHA-256, which provides unique hash values. A database stores all hashed words information that can be used for searching the encrypted document with the help of bloom filters.

V. Conclusion

This research has illustrated how our key management scheme can enhance the efficiency and the security of EHRs without the need for a third-party interaction during the exchange of encryption/decryption keys between the participants within the system. Moreover, it has demonstrated how an existing search tool can be used efficiently to search inside encrypted EHRs in the cloud. In the future, we are planning to design an access control model for EHRs which will be adaptable with our key management scheme and the searching mechanism.

References

[1] S. Akl and P. Taylor, "Cryptographic solution to a problem of access control in a hierarchy," *ACM Transactions on Computer Systems (TOCS)*, vol. 1, no. 3, pp. 239–248, 1983.
[2] S. MacKinnon, P. Taylor, H. Meijer, and S. Akl, "An optimal algorithm for assigning cryptographic keys to control access in a hierarchy," *Computers, IEEE Transactions on*, vol. 100, no. 9, pp. 797–802, 1985.
[3] L. Harn and H. Lin, "A cryptographic key generation scheme for multilevel data security," *Computers & Security*, vol. 9, no. 6, pp. 539–546, 1990.
[4] I. Lin, M. Hwang, and C. Chang, "A new key assignment scheme for enforcing complicated access control policies in hierarchy," *Future Generation Computer Systems*, vol. 19, no. 4, pp. 457–462, 2003.
[5] A. Kayem, S. Akl, and P. Martin, *Adaptive cryptographic access control.* Springer, 2010, vol. 48.
[6] E. Goh *et al.*, "Secure indexes," *An early version of this paper first appeared on the Cryptology ePrint Archive on October 7th*, 2003.
[7] B. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, 1970.