IJREAT International Journal of Research in Engineering & Advanced Technology, Volume 2, Issue 2, Apr-May, 2014
ISSN: 2320 – 8791 (Impact Factor: 1.479)
www.ijreat.org

Fine-Grained Access Control in Cloud Computing for Personal Health Care Domain

Ms.Deepika.S[1], Ms.Keerthi Sravanthi.G[2], Mr.S.Gopalakrishnan[3]

[1, 2, 3]**Information Technology, AIHT, Chennai, Tamil Nadu, India**

Abstract

Health being the most important assert, it is to be safeguarded in today's mechanized life. The maintenance of personal health and tracking of Personal Health Records(PHR) are of great importance, as (PHR) is an emerging patient-centric model of health information exchange, which is often outsourced to be stored at a third party servers. There have been wide privacy concerns as personal health information could be exposed to those third party servers and to unauthorized parties, malicious threats, and possibility of hacking. It is necessary to assure the patient's control over access to their own PHR's, it is a promising method to encrypt the PHRs before outsourcing. In this paper, we propose a patient-centric framework and mechanisms for data access control to PHRs stored in semi-trusted servers, such as cloud servers. To achieve fine-grained and scalable data access control for PHRs, we use the concept of attribute based encryption (ABE) techniques to encrypt each patient's PHR file using Rijndael Managed algorithm.

Keywords: *Personal Health Records, cloud computing, data privacy, fine grained access control, ABE, Rijndael Managed algorithm.*

1. Introduction

The personal health record (PHR) has emerged as a patient-centric model of health information exchange in the recent years. A PHR service allows a patient to create, manage, and control his personal health data in one place through the web, which has made the storage, retrieval, and sharing of the medical information more efficient. Each patient has the full control of her medical records and can share her health data with a wide range of users, including healthcare providers, insurance providers, family members or friends. Due to the high cost of building and maintaining specialized data centers, many PHR services are outsourced to or provided by third-party service providers, for example, Microsoft Health Vault. Recently, architectures of storing PHRs in cloud computing have been proposed.

Even though the PHR services are convenient for everyone, there are many security and privacy risks which could be involved. The main concern is about whether the patients could actually control the sharing of their sensitive personal health information (PHI), especially when they are stored on a third-party server which people may not fully trust. Even though there exist healthcare regulations such as HIPAA which is recently amended to incorporate business associates, cloud providers are usually not covered entities and also due to the high value of the sensitive PHI, the third-party storage servers are often the targets of various malicious behaviors which may lead to exposure of the PHI so this leads to hacking. Hence, to ensure patient-centric privacy control over their own PHRs, it is necessary to have fine-grained data access control mechanisms for semitrusted servers.

A possible and best approach is to encrypt the data before storing on the semitrusted servers. The PHR owner himself should decide in what way he can encrypt his files and also decide as to which set of users he can allow to access each of his file. A PHR file should only be available to the users who are given the corresponding decryption key, while remaining confidential to the other users .Also, the patient should always retain the right to not only grant, but also revoke access privileges when he feels it is necessary. However, while achieving the goals, scalability conflicts may arise. The authorized users may either need to access the PHR for personal use or professional purposes. Examples of personal domain are family members, relatives, friends, and for the professional use, it can be medical doctors, nurses, pharmacists, researchers, etc. The professional domain is very wide or large. In such a case if each owner himself is directly responsible for managing all the professional users, he will have to face the key management overhead problem. Since there are many number of professional users, it is difficult for an owner to maintain a list of them and also provide proper write access to each of them .For example, a hospital may have many doctors ,

nurses and other non technical staff and assistants. In case if all of them have been provided write access to a patient's highly confidential or sensitive PHR file, then it is too difficult for them to maintain order and accuracy about a patients details. Hence it is very important to maintain and set privileges and access policies for the data contributors as to who can edit the PHR and who cannot. On the other hand, different from the single data owner scenario considered in most of the existing works in a PHR system, there are multiple owners who may encrypt according to their own ways, possibly using different sets of crypto-graphic keys. When a patient is not online and if a user want to access the patients PHR, the accessibility would be limited. An alternative is to involve a central authority (CA) which does the key management on behalf of all PHR owners, but this requires too much trust on a single authority (i.e., cause the key escrow problem).

In this paper, we focus on addressing the complicated and challenging key management issues, security challenges and access privileges that take place in the patient centric model in which the PHR's are stored on semi trusted servers. In order to protect the personal health data stored on semitrusted servers, we adopt attribute-based encryption (ABE) as an encryption primitive. We also use the rijndael algorithm to generate the public and secret keys for the user for any access. Using ABE, access policies are expressed based on the attributes of users or data, which enables a patient to selectively share his PHR among a set of different users by encrypting the file under a set of attributes, without needing to know the entire list of users. Depending on the number of attributes, the complexities per encryption, key generation, and decryption varies. However, to integrate ABE into a large-scale PHR system, important issues such as scalability in key management, inefficient on-demand revocation are hard to solve. To end this, we make the following main contributions:

1. We propose a multi owner setting framework for patient-centric secure sharing of PHRs in cloud computing environments. For combatting the key management issues, we divide the users in the system into two types of domains, namely public domain (PUD) and personal domain (PSD). Since the majority of users are professional users, they are managed distributively by attribute authorities (AA), while each owner only needs to manage the keys of a small number of users in her personal domain. In such a way, our framework simultaneously handles the sharing of different types of PHR files, which further involves minimal key management overhead for both owners and users in the system. In addition, our framework implements write, read access control, and provides break-glass access to PHRs under emergency situations.

2. In the public domain, we use multiauthority ABE (MA-ABE) to improve the security and avoid key escrow problem. Each attribute authority (AA) in it governs a disjoint subset of user role attributes, while none of them alone is able to control the security of the whole system. We propose mechanisms for key distribution and encryption so that PHR owners can specify personalized fine-grained role-based access policies during file encryption. In addition to that we have also set permissions as to who can edit and update information in a particular PHR. In the personal domain, owners directly assign access privileges for personal users and encrypt a PHR file under its data attributes. Again, over here we have another problem of security, this is based on the assumption i,e incase if the personal users come to know about the PHR owners password's then there are chances of editing the PHR without the knowledge of the owner. Furthermore, we enhance MA-ABE by putting forward an efficient and on-demand user/ attribute revocation scheme, and prove its security under standard security assumptions. In such a way, patients have full privacy control over their PHRs.

3. We provide a thorough analysis of the complexity and scalability of our proposed secure PHR sharing solution, in terms of multiple metrics in computation, communication, storage, and key management. We also compare our scheme to several previous ones in complexity, scalability and security.

Compared with the preliminary version of this paper there are several main additional contributions:

1) We clarify and extend our usage of MA-ABE in the public domain, and formally show how and which types of user-defined file access policies are realized.

2) We have used the rijndael managed encryption standard for the benefit that it can encrypt an entire document at a time rather than as a single string.

3) We also ensure proper write access controls by providing mutable and immutable forms of the same file (word, PDF, image etc) to the users request depending upon their attribute.

4) We provided an additional layer of security for the personal domain assuming a risk of personal details being known by the owners personal users.

5) We carry out both real-world experiments and simulations to evaluate the performance of the proposed solution in this paper.

2. Framework for fine grained access in patient - centric, secure and scalable PHR sharing

In this section, we describe our patient-centric secure data sharing framework for cloud-based PHR systems.

2.1 Problem Definition

We consider a PHR system where there are multiple PHR owners and PHR users. The owners are the patients who have full control of their own PHR data, i.e., they can create, modify, manage, and delete it. There is a central server belonging to the PHR service provider that stores all the owners PHRs. The users may come from various aspects; for example, a friend, a relative, a doctor or a researcher. In order to read or write to someone's PHR, users access the PHR documents through the server and they can simultaneously have access to multiple owners data. The main problem over here with the existing system is that it demands that the key issuing owner or the organization to be online in case the users need to access the PHR details. This condition is not considered favorable as it greatly reduces the accessibility of the PHR's. To add to it in the public domain, there where problem as to which users should be granted the right to edit the owners PHR file. Secondly there is another assumption that incase if the personal domain users of the PHR come in contact to know the confidential keys or passwords of the PHR owner. A typical PHR system uses standard data formats. For example, continuity-of-care (CCR) (based on XML data structure), which is widely used in representative PHR systems including Indivo, an open-source PHR system adopted by Boston Children's Hospital. Due to the nature of XML, the PHR files are logically organized by their categories in a hierarchical way.

2.2 Model of security

In this paper, the server is considered to be semitrusted by us i.e., honest but curious. That means the server will try to find out as much secret information in the stored PHR files as possible, but they will honestly follow the protocol in general. While some users will also try to access the files beyond their privileges. For example, a pharmacy may want to obtain the prescriptions of patients for marketing and boosting its profits. To do so, they may interact with other users, or even with the server. In addition, we assume each party in our system is preloaded with a public/private key pair, and entity authentication can be done by traditional challenge-response protocols.

2.3 Requirements

To achieve "patient-centric" PHR sharing, a core requirement is that each patient can decide as to who are authorized to access to his own PHR file. Especially, user-controlled read/write access and revocation are the two core security objectives for any electronic health record system. The security and performance requirements are summarized as follows:

2.4 Data confidentiality

Unauthorized users (including the server) who do not possess enough attributes satisfying the access policy or do not have proper key access privileges should be prevented from decrypting a PHR document, even under user collusion. Fine-grained access control should be enforced, meaning different users are authorized to read different sets of documents but write access is given only to appropriate users.

2.5 On-demand revocation

Whenever a user's attribute is no longer valid, the user should not be able to access future PHR files using that attribute. This is usually called attribute revocation, and the corresponding security property is forward secrecy. There is also user revocation, where all of a user's access privileges are revoked and so the previously used secret key cannot be

used or is no more valid to read or write information.

2.6 Write access control

We shall prevent the unauthorized contributors to gain write-access to the owners PHRs, while the legitimate contributors should access the server with accountability. On doing so only an appropriate format of file associated with that particular users attribute will be download on his/her request. The data access policies must be flexible, i.e., all dynamic changes to the predefined policies must be allowed, especially the PHRs must be accessible under emergency situations.

2.7 Scalability, efficiency, and usability

The PHR system should support users from both the personal domain and public domains. Since the set of users in the public domain may be many in number and unpredictable, the system should be highly scalable , in terms of complexity in key management, communication, and storage. Furthermore, the owner's efforts must be minimized in managing his keys to enjoy usability.

3. Outline of our proposed framework

The main goal of our framework is to provide secure patient-centric PHR for fine grained access control and efficient key management at the same time. The main idea is to divide the system into multiple security domains (namely, public domains and personal domains) according to the different user data access requirements. The PUDs consist of users who make access based on their professional roles, such as doctors, nurses, pharmacists and medical researchers. In practice, a PUD can be mapped to an independent sector in the society, such as the health care, government, or insurance sector. For each PSD, the users are personally associated with a data owner (like friends, relatives, family members etc), and they make accesses to PHRs based on access rights allotted by the owner. In both security domains, we use the concept of ABE. Especially, in a PUD multi authority ABE (MA-ABE) is used, in which there are many "attribute authorities" (AAs), each governing a disjoint subset of attributes. Role attributes are defined for PUDs, representing the professional roles. Users in PUDs obtain their attribute-based secret keys from the AAs, without directly interacting with the owners since the owners cannot maintain the complete set of users. To control access from PUD users, owners are free to specify role-based fine-grained access policies for his PHR files, when doing encryption. Since the PUDs have majority of users, it greatly reduces the key management overhead for both the owners and users. Every data owner (e.g., patient) is a trusted authority of her own PSD, who uses a KP-ABE system to manage the secret keys and access rights of users in her PSD. Since the users are personally known by the PHR owner, to realize patient- centric access, the owner is at the best position to grant user access privileges on a case-by-case basis. For PSD, data attributes are defined which refer to the intrinsic properties of the PHR data, such as the category of a PHR file. For the purpose of PSD access, each PHR file is labeled with its data attributes, while the key size is only linear with the number of file categories a user can access. Since the number of users in a PSD is often small, it reduces the burden for the owner. When encrypting the data for PSD, all that the owner needs to know is the intrinsic data properties.

The multi domain approach best models different user types and access requirements in a PHR system. The use of ABE makes the encrypted PHRs self-protective, i.e., they can be accessed by only authorized users even when storing on a semitrusted server, and when the owner is not online. In addition, efficient and on-demand user revocation is made possible via our ABE enhancement.

3.1 Proposed system – a detailed approach

In our proposed framework, there are multiple SDs, multiple owners, multiple AAs, multiple users. We term the users having read access as data readers and write access as data contributors.

3.2 System setup and key distribution

The system at first defines a common universe of data attributes that are shared by every PSD, such as "basic profile," "allergies," "medical history," and "prescriptions." An emergency attribute is also defined for break-glass access. Every PHR owner's client application generates its own corresponding public/master keys. These public keys can be published via user's profile in any online like health care social-network (HSN). There are two ways for distributing secret keys. At first, when using the PHR service, a PHR owner can specify the access privilege of a data reader in his PSD, and let him application generate and distribute corresponding key to the latter. Secondly, a reader in PSD could obtain the secret key by sending

him a request (indicating which types of files he wants to access) to the PHR owner via HSN, and the owner will grant him a subset of requested data types. Based on this, the policy engine of application automatically derives an access structure, and runs keygen of the algorithm to generate the user secret key. In addition, the data attributes can be organized in a hierarchical manner for efficient policy generation. When the user is granted all the file types under a category, his access privilege will be represented by his category instead. For PUDs, the system defines role attributes, and a reader in a PUD obtains secret key from AAs, which binds the user to his claimed attributes/roles. For example, a physician in it would receive "hospital B, physician, M.D., internal medicine" as his attributes from AAs. In practice, there exist multiple AAs each governing a different subset of role attributes. For instance, hospital staffs may have a different AA from pharmacy specialists. Additionally, the AAs distribute write keys which permit contributors in their PUD to write to some patient's PHR.

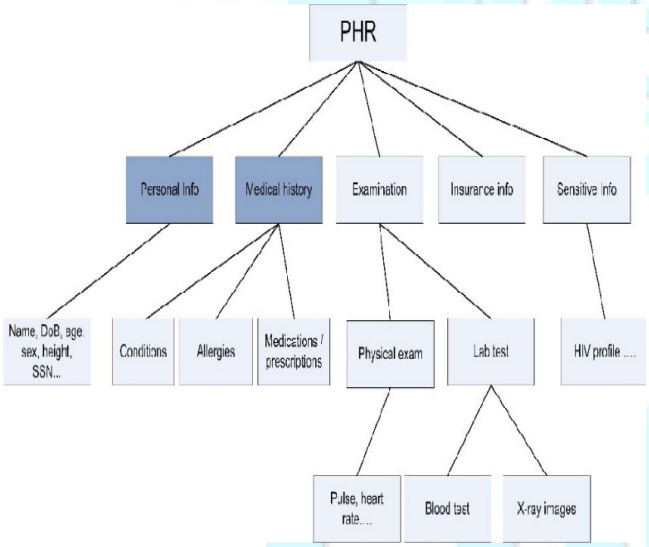

Fig.1: A sample PHR hierarchy

3.3 PHR encryption and access

The owners upload ABE encrypted PHR files to the server (3). Each owner's PHR file is encrypted both under a certain fine-grained and role based access policy for the users from the PUD to access, & under a selected set of data attributes that allow access from users in the PSD. Only authorized users can decrypt the PHR files, excluding the cloud server. For improving efficiency, the data attributes will include all the intermediate file types from a leaf node to the root. The data readers download the PHR files from the server, and they can decrypt the files only if they have suitable attribute-based keys (5). The data contributors will be granted the write access to someone's PHR, only if they present proper write keys.

3.4 User revocation

Here, we consider revocation of the data readers or his attributes/access privileges. There are several possible cases:

1. Revocation of one or more role attributes of a public domain user.
2. Revocation of a public domain user which is equivalent to revoking all of that user's attributes. These operations are done by the AA that the user belongs to, where the actual computations can be delegated to the server to improve efficiency.
3. Revocation of a personal domain user's access privileges.
4. Revocation of a personal domain user. These can be initiated through the PHR owner's client application in a similar way.

.

3.5 Policy updates

A PHR owner can update his sharing policy for an existing PHR document by updating the attributes (or access policy) in the ciphertext. The supported operations include add/delete/modify, which can be done by the server on behalf of the user.

3.6 Break - glass

When any emergency happens, the regular access policies may no longer be applicable. To handle this situation, break-glass access is required to access the victim's PHR. In our framework, every owner's PHR access right is also delegated to an emergency department (ED) and to prevent from abuse of break-glass method, the emergency staff need to contact the ED to verify her identity and the emergency situation, and obtain the temporary read keys. After the emergency is over, the patient then can revoke the emergent access via the ED.

4. System architecture

The system architecture is depicted in the following diagram.

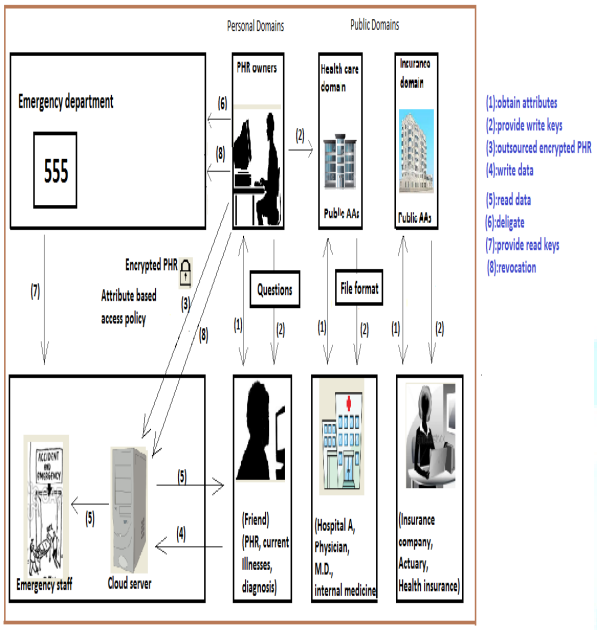

Fig.2: System Architecture

5. Algorithm and encryption standard used for encrypting PHR's- Rijndael Managed

The algorithm used for the encryption is Rijndael Managed. We use Rijndael Managed with 256 bits key and 256 bits block size. Longer key/IV lengths provide better security and they will be slow. In the world of encryption, the slower, the better and more difficult to crack. The default for Rijndael Managed is 265 bits key and 128 bits block size. The Block Size property determines the IV (Initialization Vector) size. MSDN says — the classes that derive from the Symmetric Algorithm class use a chaining mode called cipher block chaining (CBC), which requires a key and an initialization vector to perform cryptographic transformations on data. To decrypt data that was encrypted using one of the Symmetric Algorithm classes, you must set the Key property and IV property to the same values that were used for encryption. This algorithm independent of the key (i.e) it is not key dependent. We use rijndael managed algorithm because most of the other encryption standards are used only for encrypting a single string at a time whereas this algorithm standard is useful in encrypting a entire file at a time.

SETUP: Initially before commencing the encryption process the primary work is to check whether the file exist and is empty. Then it verifies whether the owner is null or not.

Inputs: file & owner

If(input !=NULL)
{
Keygen();
}

KEY GENERATION: The algorithm used over here for key generation is rijndael managed. The size of the public which is generated over here is of 256 bits. This key is generated upon the request of the user.

}
Else
Return;

KEY ISSUE: The public key is generated and issues here to the user based upon his request.

ENCRYPTION:

Inputs: (file, public key(Pk), owner name)

{

Encrypt()
{

The file before encryption can be of any form i.e word, PDF, html etc.

File →byte array;
Byte array → blob form(image format);
storeDB();
}

}

DECRYPTION: The initial step of decryption process is to validate the users request for the correct attributes.

If (attribute==TRUE)
{
Secretkeygen();

Inputs:(public key(Pk),secret key(Sk), owners name, blob name)
Decrypt()
{
Blob form → byte array;
Byte array → string(file);
[reconstruction of code]
}

Downloadfile();

5.1 Steps used in encryption process

The inputs needed to begin the encryption process of the PHR file are, the file, owner's name and its corresponding public key. This public key is generated in the key generation function. Initially it checks whether the file exists and is empty. The commencing step of encryption is to convert the file into byte array .Byte array is a hexadecimal machine understandable form.(we do this conversion because the byte array form is easy for transfer of information over the network).But byte array as such cannot be stored in the database so we convert the byte array to its image format called BLOB(binary large object)Then it is stored in the database on the cloud server along with the secret key which is generated. We prefer the byte array format because since it is in the hexadecimal form even though the information is hacked and obtain from the semitrusted server it cannot be read or understood as it is in the intangible form. Thus we ensure security at its root level.

5.2 Steps used in decrypting process

The inputs needed for commencing the decrypting process are the users public key, corresponding secret key's, owners name and the image file name. The secret key's which they provide here are issued by the appropriate attribute authority. The owner itself acts are an attribute authority in care of personal domain, where as there are other attributes in case of public domain.

The process is the file is initially retrieved from the database. Then the retrieved file from the database is in the image form of byte array. It is then converted back from byte array to its corresponding form i.e. either a word or PDF or a html file. This conversion is necessary because the byte array is not a human understandable format. The file is then downloaded for the users need.

6. Conclusion

In this paper, we have proposed a promising framework of secure sharing of personal health records in cloud computing. Considering partially trustworthy cloud servers, we conclude that to fully realize the patient-centric concept, patients shall have complete control of their own privacy through encrypting their PHR files to allow fine-grained access in a tight secure manner. The framework addresses the unique challenges brought by multiple PHR owners and users, in that we greatly reduce the complexity of key management while enhance the privacy guarantees in both the public and private domains on compared with previous works. We utilize the concept of ABE and rijndael managed encryption standard to encrypt the PHR data, so that patients can allow access not only to personal users, but also various users from public domains with different professional roles, qualifications, and affiliations. Furthermore, we enhance an existing MA-ABE scheme to handle efficient and on-demand user revocation, and prove its security. Through implementation and simulation, we propose that our solution is inevitably secure and efficient.

References

[1] "Scalable and Secure Sharing of Personal Health Records in Cloud Computing Using Attribute-Based Encryption" Ming Li, Member, IEEE, Shucheng Yu, Member, IEEE, Yao Zheng, Student Member, IEEE, Kui Ren, Senior Member, IEEE, and Wenjing Lou, Senior Member.jan 2013.

[2] S. Yu, C. Wang, K. Ren, and W. Lou, "Achieving Secure, Scalable,and Fine-Grained Data Access Control in Cloud Computing,"Proc. IEEE INFOCOM '10, 2010.

[3] "The Health Insurance Portability and AccountabilityAct,"http://www.cms.hhs.gov/HIPAAGenInfo/01_Overview.asp, 2012.

[4] J. Benaloh, M. Chase, E. Horvitz, and K. Lauter, "Patient Controlled Encryption: Ensuring Privacy of Electronic Medical Records," Proc. ACM Workshop Cloud Computing Security(CCSW '09), pp. 103-114, 2009.

[5] H. Lo¨ hr, A.-R. Sadeghi, and M. Winandy, "Securing the E-Health Cloud," Proc. First ACM Int'l Health Informatics Symp. (IHI '10), pp. 220-229, 2010.

[6] "Google, Microsoft Say Hipaa Stimulus Rule Doesn'tApplytoThem,http://www.ihealthbeat.org/Articles/2009/4/8/, 2012. (CCSW '09), pp. 103-114, 2009.

[7] C. Dong, G. Russello, and N. Dulay, "Shared and Searchable Encrypted Data for Untrusted Servers," J. Computer Security,vol. 19, pp. 367-397, 2010.

[8] M. Li, W. Lou, and K. Ren, "Data Security and PrivacyinWirelessBodyAreaNetworks,"IEEEWireless Comm .Magazine ,vol.17,no. 1, pp. 51-58, Feb. 2010.

[9] S. Jahid, P. Mittal, and N. Borisov, "Easier: Encryption-Based Access Control in Social Networks with EfficientRevocation," Proc. ACM Symp. Information, Computer and Comm. Security (ASIACCS), Mar.2011.