Using Cloud Information Accountability (CIA) framework for Storing Personal Health Records in the Cloud

Bollamma C K[1] , Keerthana R M[2], Radhika H S[3], Suhas S Bhat[4], Sangeeta Uranakar[5]
Department of Information Science and Engineering,City Engineering College, Doddakalsandra, Bangalore-560061
bollamma.ck@gmail.com[1] , keerthu.rm@gmail.com[2] , radhikahs18@gmail.com[3], suhas111x@gmail.com[4]

Abstract

Personal health record (PHR) is an emerging patient-centric model of health information exchange, but it is often outsourced to be stored in remote servers, such as cloud providers. However, there have been wide privacy concerns as personal health information could be exposed to those third party servers and to unauthorized parties. , which makes it necessary for each patient to encrypt her PHR data before uploading to the cloud servers..Yet, issues such as risks of privacy exposure and flexible access have remained the most important challenges toward achieving fine-grained, cryptographically enforced data access control. Here, we propose a novel patient-centric framework and a suite of mechanisms for data access control to PHRs stored in semi-trusted servers. To achieve security and accountability for PHRs, we leverage a highly decentralized information accountability framework to keep track of the actual usage of the users' PHR in the cloud. In particular, we propose an object-centered approach that enables enclosing our logging mechanism together with users' data and policies. We use the JAR programmable capabilities to both create a dynamic and traveling object, and to ensure that any access to the PHR will trigger authentication and automated logging local to the JARs. To strengthen user's control, we also provide distributed auditing mechanisms. A high degree of patient privacy is guaranteed. We provide extensive experimental studies that demonstrate the efficiency and effectiveness of the proposed approaches.

Index Terms—*Personal health records, Cloud computing, data privacy, Accountability*.

1.Introduction

In recent years, personal health record (PHR) has emerged as a patient-centric model of health information exchange. A PHR service allows a patient to create, manage, and control her personal health data in one place through the web, which has made the storage, retrieval, and sharing of the information more efficient. Especially, each patient is promised the full control of her medical records and can share her health data with a wide range of users, including healthcare 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 HealthVault. While it is exciting to have convenient PHR services, there are many security and privacy risks which could impede its wide adoption. 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. On one hand, although there exist healthcare regulations such as HIPAA which is recently amended to incorporate business associates, cloud providers are usually not covered entities. On the other hand, 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. As a famous incident, a Department of Veterans.

Affairs database containing sensitive PHI of 26.5 million military veterans, including their social security numbers and health problems was stolen by an employee who took the data home without authorization. To ensure patient-centric privacy control over their own PHRs, it is essential to have data access control mechanisms that work with semi-trusted servers. A feasible and promising approach would be to encrypt the data before outsourcing. Basically, the PHR owner herself should decide how to encrypt her files and to allow which set of users to obtain access to each file. A PHR file should only be available to the users who are given the corresponding decryption key, while remain confidential to the rest of users.

In this paper, we endeavor to study the patient centric, secure sharing of PHRs stored on semi-trusted servers, In order to protect the PHR stored on a semi-trusted server, we propose a novel approach, Cloud Information Accountability (CIA) framework, based on the notion of information accountability. Unlike privacy protection technologies which are built on the hide-it-or-lose-it perspective, information accountability focuses on keeping the data usage transparent and trackable. Our proposed CIA framework provides end-to-end accountability in a highly distributed fashion. One of the main innovative features of the CIA framework lies in its ability of maintaining lightweight and powerful accountability that combines aspects of access control, usage control and authentication. By means of the CIA, data owners can track not only whether or not the service-level agreements are being honored, but also enforce access and usage control.

2. Framework for Patient-centric, Secure and Scalable PHR sharing

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

2.1 Problem Definition

We consider a PHR system where there are multiple PHR owners and PHR users. The proposed framework for patient-centric, secure and scalable PHR sharing on semi-trusted storage under multi-owner settings is shown in fig 1. The owners refer to patients who have full control over their own PHR data, i.e., they can create, 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 caregiver or a researcher. Users access the PHR documents through the server in order to read or write to someone's PHR, and a user can simultaneously have access to multiple owners' data. Also 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 collude with the server.

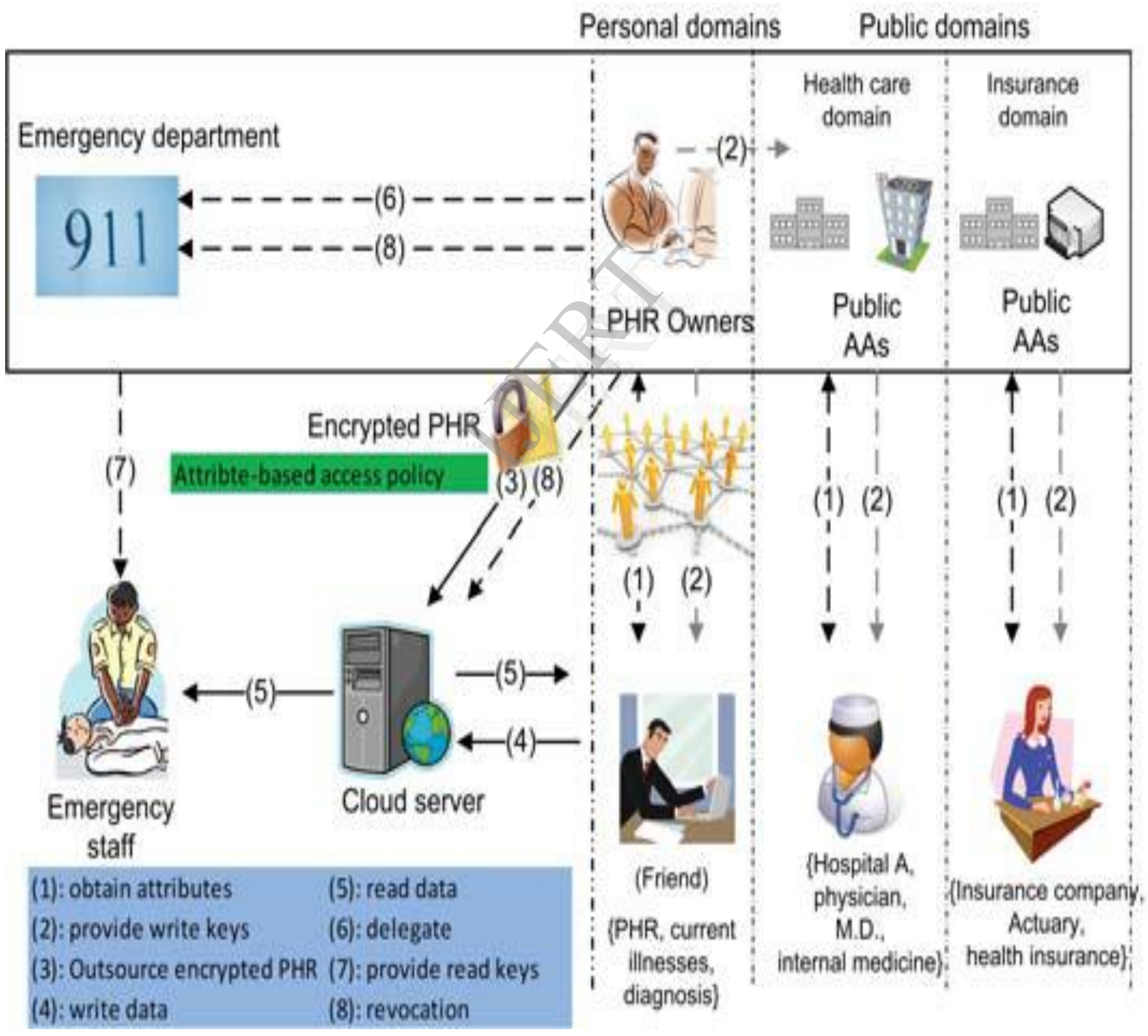

Fig.1 The proposed framework for PHR sharing on semi-trusted storage under multi-owner settings

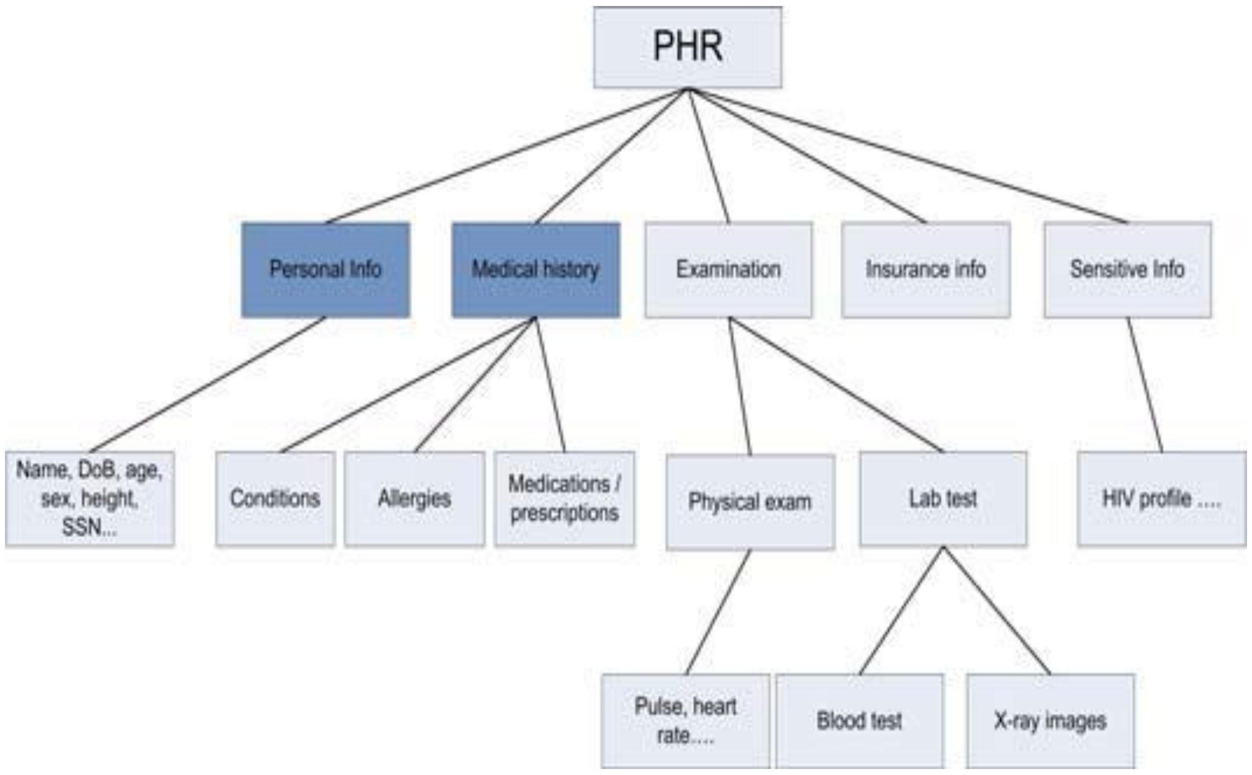

Fig. 2.The attribute hierarchy of files – leaf nodes is atomic file categories while internal nodes are compound categories. Dark boxes are the categories that a PSD's data reader has access to

2.1.2 Requirements
To achieve "*patient-centric*" PHR sharing, a core requirement is that each patient can control who are authorized to access to her own PHR documents. 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:

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. Owner should come to know who is trying to access the data.
Write access control. We shall prevent the unauthorized contributors to gain write-access to owners' PHRs, while the legitimate contributors should access the server with accountability. The data access policies should be flexible, i.e. dynamic changes to the predefined policies shall be allowed, and especially the PHRs should be accessible under emergency scenarios.

Scalability, efficiency and usability. The PHR system should support users from both the personal domain and public domains. Since the set of users from the public domain may be large in size and unpredictable, the system should be highly scalable, in terms of complexity in key management, communication, computation and storage. Additionally, the owners' efforts in managing users and keys should be minimized to enjoy usability.

2.2 Overview of Our Framework

In this section, we present an overview of the Cloud Information Accountability framework and discuss how the CIA framework meets the design requirements discussed in the previous section. The Cloud Information Accountability framework proposed in this work conducts automated logging and distributed auditing of relevant access performed by any entity, carried out at any point of time at any cloud service provider. It has two major components: logger and log harmonizer.

2.2.1 Major Components

There are two major components of the CIA, the first being the logger, and the second being the log harmonizer. The logger is the component which is strongly coupled with the owners PHR, so that it is downloaded when the data are accessed, and is copied handles a particular instance or copy of the owner's data and is responsible for logging access to that instance or copy. The log harmonizer forms the central component which allows the owner access to the log files. The logger is strongly coupled with user's data (either single or multiple data items). Its main tasks include automatically logging access to data items that it contains, encrypting the log record using the public key of the content owner, and periodically sending them to the log harmonizer. It may also be configured to ensure that access and usage control policies associated with the data are honored. For example, a data owner can specify that user X is only allowed to view but not to modify the data. The logger will control the data access even after it is downloaded by user X. The logger requires only minimal support from the server (e.g., a valid Java virtual machine installed) in order to be deployed. The tight coupling between data and logger, results in a highly distributed logging system, therefore meeting our first design requirement. Furthermore, since the logger does not need to be installed on any system or require any special support from the server, it is not very intrusive in its actions, thus satisfying our fifth requirement. Finally, the logger is also responsible for generating the error correction information for each log record and send the same to the log harmonizer. The error correction information combined with the encryption and authentication mechanism provides a robust and reliable recovery mechanism, therefore meeting the third requirement. The log harmonizer is responsible for auditing. Being the trusted component, the log harmonizer generates the master key. It holds on to the decryption key for the IBE key pair, as it is responsible for decrypting the logs. Alternatively, the decryption can be carried out on the client end if the path between the log harmonizer and the client is not trusted. In this case, the harmonizer sends the key to the client in a secure key exchange. It supports two auditing strategies: push and pull. Under the push strategy, the log file is pushed back to the data owner periodically in an automated fashion. The pull mode is an on-demand approach, whereby the log file is obtained by the data owner as often as requested. In case there exist multiple loggers for the same set of data items, the log harmonizer will merge log records from them before sending back to the data owner. The log harmonizer is also responsible for handling log file corruption. In addition, the log harmonizer can itself carry out logging in addition to auditing. Separating the logging and auditing functions improves the performance. The logger and the log harmonizer are both implemented as lightweight and portable JAR files. The JAR file implementation provides automatic logging functions.

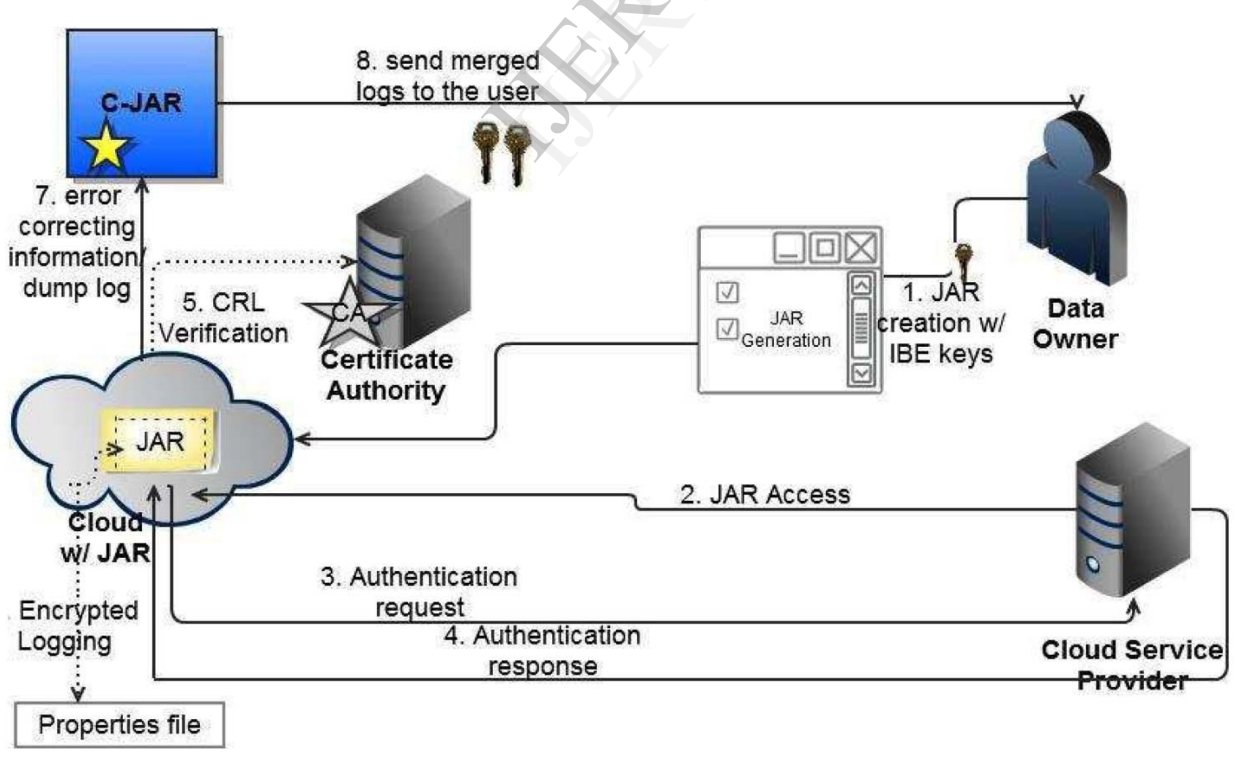

Fig 3: Overall CIA framework combining data, users, loggers and harmonize

The overall CIA framework, combining data, users, logger and harmonizer is sketched in Fig. 3. At the beginning, each owner of the PHR creates a pair of public and private keys based on Identity-Based Encryption (step 1 in Fig. 3). This IBE scheme is a Weil-pairing-based IBE scheme, which protects us
against one of the most prevalent attacks to our architecture. Using the generated key, the owner will create a logger component which is a JAR file, to store its data items. The JAR file includes a set of simple access control rules specifying whether and how the cloud servers and possibly other users are authorized to access the content itself. Then, he sends the JAR file to the cloud service provider that he subscribes to. To authenticate the CSP to the JAR (steps 3-5 in Fig. 3), we use OpenSSLbased certificates, wherein a trusted certificate authority certifies the CSP. In the event that the access is requested by a user, we employ SAML-based authentication, wherein a trusted identity provider issues certificates verifying the user's identity based on his username. Once the authentication succeeds, the service provider (or the user) will be allowed to access the data enclosed in the JAR. Depending on the configuration settings defined at the time of creation, the JAR will provide usage control associated with logging, or will provide only logging functionality. As for the logging, each time there is an access to the data, the JAR will automatically generate a log record encrypt it using the public key distributed by the data owner, and store it along with the data (step 6 in Fig. 3). The encryption of the log file prevents unauthorized changes to the file by attackers. The data owner could opt to reuse the same key pair for all JARs or create different key pairs for separate JARs. Using separate keys can enhance the security without introducing any overhead except in the initialization phase. In addition, some error correction information will be sent to the log harmonizer to handle possible log file corruption (step 7 in Fig.3. To ensure trustworthiness of the logs, each record is signed by the entity accessing the content. Further, individual records are hashed together to create a chain structure, able to quickly detect possible errors or missing records. The encrypted log files can later be decrypted and their integrity verified. They can be accessed by the data owner or other authorized stakeholders at any time for auditing purposes with the aid of the log harmonizer (step 8 in Fig. 3). Note that our work is different from traditional logging methods which use encryption to protect log files. With only encryption, their logging mechanisms are neither automatic nor distributed. They require the data to stay within the boundaries of the centralized system for the logging to be possible, which is however not suitable in the cloud.
The main goal of our framework is to provide secure patient-centric PHR access, accountability and efficient key management at the same time. The key idea is to divide the system into multiple security domains (namely, *public domains* (PUDs) and *personal domains* (PSDs)) according to the different users' data access requirements. The PUDs consist of users who make access based on their professional roles, such as doctors, nurses 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, its users are personally associated with a data owner (such as family members or close friends), and they make accesses to PHRs based on access rights assigned by the owner.

4 Result analyses

4.1 Security analysis

In this section, we analyze the security of the proposed PHR sharing solution. First we show it achieves data confidentiality (i.e., preventing unauthorized read accesses), by proving the enhanced IBE algorithm. Accountability is achieved through the logger and log harmonizer components. In addition, our framework achieves forward secrecy and security of write access control.

4.2 Performance analysis

First, we evaluate the scalability and efficiency of our solution in terms of storage, communication and computation costs. We compare with previous schemes in terms of cipher text size, user secret key size, and public key/information size. Our analysis is based on the worst case where each user may potentially access part of every owners' data. In the experiments, we first examine the time taken to create a log file and then measure the overhead in the system. With respect to time, the overhead can occur at three points: during the authentication, during encryption of a log record, and during the merging of the logs. Also, with respect to storage overhead, we notice that our architecture is very lightweight, in that the only data to be stored are given by the actual files and the associated logs. Further, JAR act as a compressor of the files that it handles. In particular, multiple files can be handled by the same logger component. To this extent, we investigate whether a single logger component, used to handle more than one file, results in storage overhead.

5 Conclusion

In this paper, we have proposed a novel framework of secure sharing of personal health records in cloud computing. Considering partially trustworthy cloud servers, we argue 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. 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 compared with previous works. We utilize IBE to encrypt the PHR data, so that patients can allow access not only by personal users, but also various users from public domains with different professional roles, qualifications and affiliations. Furthermore, through implementation and simulation, we show that our solution is both scalable and efficient.

References

[1] Smitha Sundareswaran, Anna C. Squicciarini, Member, IEEE, and Dan *Lin* ***"****Ensuring Distributed Accountability for Data Sharing in the Cloud"* IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING VOL.9 NO.4 YEAR 2012

[2] *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, IEEE*

[3] M. Li, S. Yu, K. Ren, and W. Lou, "*Securing personal health records in cloud computing: Patient-centric and fine-grained data access control in multi-owner settings,*" in *SecureComm'10*, Sept. 2010, pp. 89–106

[4] M. Li, S. Yu, N. Cao, and W. Lou, "*Authorized private keyword search over.*