This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2018.2854790, IEEE Transactions on Cloud Computing

SeSPHR: A Methodology for Secure Sharing of Personal Health Records in the Cloud

Mazhar Ali, *Member, IEEE*, Assad Abbas, *Member, IEEE,* Muhammad Usman Shahid Khan, *Member, IEEE,* and Samee U. Khan, *Senior Member, IEEE*

Abstract — The widespread acceptance of cloud based services in the healthcare sector has resulted in cost effective and convenient exchange of Personal Health Records (PHRs) among several participating entities of the e-Health systems. Nevertheless, storing the confidential health information to cloud servers is susceptible to revelation or theft and calls for the development of methodologies that ensure the privacy of the PHRs. Therefore, we propose a methodology called SeSPHR for secure sharing of the PHRs in the cloud. The SeSPHR scheme ensures patient-centric control on the PHRs and preserves the confidentiality of the PHRs. The patients store the encrypted PHRs on the un-trusted cloud servers and selectively grant access to different types of users on different portions of the PHRs. A semi-trusted proxy called Setup and Re-encryption Server (SRS) is introduced to set up the public/private key pairs and to produce the re-encryption keys. Moreover, the methodology is secure against insider threats and also enforces a forward and backward access control. Furthermore, we formally analyze and verify the working of SeSPHR methodology through the High Level Petri Nets (HLPN). Performance evaluation regarding time consumption indicates that the SeSPHR methodology has potential to be employed for securely sharing the PHRs in the cloud.

Index Terms—Access control, cloud computing, Personal Health Records, privacy

1 Introduction

CLOUD computing has emerged as an important computing paradigm to offer pervasive and on-demand availability of various resources in the form of hardware, software, infrastructure, and storage [1, 2]. Consequently, the cloud computing paradigm facilitates organizations by relieving them from the protracted job of infrastructure development and has encouraged them to trust on the third-party Information Technology (IT) services [3]. Additionally, the cloud computing model has demonstrated significant potential to increase coordination among several healthcare stakeholders and also to ensure continuous availability of health information, and scalability [4, 5]. Furthermore, the cloud computing also integrates various important entities of healthcare domains, such as patients, hospital staff including the doctors, nursing staff, pharmacies, and clinical laboratory personnel, insurance providers, and the service providers [6]. Therefore, the integration of aforementioned entities results in the evolution of a cost effective and collaborative health ecosystem where the patients can easily create and manage their Personal Health Records (PHRs) [7]. Generally, the PHRs contain information, such as: **(a)** demographic information, **(b)** patients' medical history including the diagnosis, allergies, past surgeries, and treatments, **(c)** laboratory reports, **(d)** data about health insurance claims, and **(e)** private notes of the patients about certain important observed health conditions [8]. More formally, the PHRs are managed through the Internet based tools to permit patients to create and manage their health information as lifelong records that can be made available to those who need the access [9]. Consequently, the PHRs enable the patients to effectively communicate with the doctors and other care providers to inform about the symptoms, seek advice, and keep the health records updated for accurate diagnosis and treatment.

Despite the advantages of scalable, agile, cost effective, and ubiquitous services offered by the cloud, various concerns correlated to the privacy of health data also arise. A major reason for patients' apprehensions regarding the confidentiality of PHRs is the nature of the cloud to share and store the PHRs [10]. Storing the private health information to cloud servers managed by third-parties is susceptible to unauthorized access. In particular, privacy of the PHRs stored in public clouds that are managed by commercial service providers is extremely at risk [11]. The privacy of the PHRs can be at risk in several ways, for example theft, loss, and leakage [12]. The PHRs either in cloud storage or in transit from the patient to the cloud or from cloud to any other user may be susceptible to unauthorized access because of the malicious behavior of external entities. Moreover, there are also some threats by valid insiders to the data [13]. For instance, the PHRs either in cloud storage or in transit from the patient to the cloud or from cloud to any other user may be susceptible to unauthorized access because of the malicious behavior of external entities [10]. The individuals working at the cloud service provider can behave maliciously. A popular example for that is an incident when an employee of the Department of Veterans Affairs of the U.S. carried home

M. Ali, A. Abbas, and M. U. S. Khan are with the COMSATS University Islamabad, Pakistan e-mail: {mazhar, ushahid}@ciit.net.pk}, assadabbas@comsats.edu.pk.

S. U. Khan is with the North Dakota State University, 1411 Centennial Blvd, Fargo, ND 58108-6050. E-mail: samee.khan@ndsu.edu.

xxxx-xxxx/0x/$xx.00 © 200x IEEE Published by the IEEE Computer Society

the sensitive personal health information of around 26.5 million without any authorization [14]. The Health Insurance Portability and Accountability Act (HIPAA) mandates that the integrity and confidentiality of electronic health information stored by the healthcare providers must be protected by the conditions of use and disclosure and with the permission of patients [15]. Moreover, while the PHRs are stored on the third-party cloud storage, they should be encrypted in such a way that neither the cloud server providers nor the unauthorized entities should be able to access the PHRs. Instead, only the entities or individuals with the 'right-to-know' privilege should be able to access the PHRs. Moreover, the mechanism for granting the access to PHRs should be administered by the patients themselves to avoid any unauthorized modifications or misuse of data when it is sent to the other stakeholders of the health cloud environment.

Numerous methods have been employed to ensure the privacy of the PHRs stored on the cloud servers. The privacy preserving approaches make sure confidentiality, integrity, authenticity, accountability, and audit trial. Confidentiality ensures that the health information is entirely concealed to the unsanctioned parties [14], whereas integrity deals with maintaining the originality of the data, whether in transit or in cloud storage [16]. Authenticity guarantees that the health-data is accessed by authorized entities only, whereas accountability refers to the fact that the data access policies must comply with the agreed upon procedures. Monitoring the utilization of health-data, even after access to that has been granted is called audit trial [6].

We present a methodology called Secure Sharing of PHRs in the Cloud (SeSPHR) to administer the PHR access control mechanism managed by patients themselves. The methodology preserves the confidentiality of the PHRs by restricting the unauthorized users. Generally, there are two types of PHR users in the proposed approach, namely: **(a)** the patients or PHR owners and **(b)** the users of the PHRs other than the owners, such as the family members or friends of patients, doctors and physicians, health insurance companies' representatives, pharmacists, and researchers.

The patients as the owners of the PHRs are permitted to upload the encrypted PHRs on the cloud by selectively granting the access to users over different portions of the PHRs. Each member of the group of users of later type is granted access to the PHRs by the PHR owners to a certain level depending upon the role of the user. The levels of access granted to various categories of users are defined in the Access Control List (ACL) by the PHR owner. For example, the family members or friends of the patients may be given full access over the PHRs by the owner. Similarly, the representatives of the insurance company may only be able to access the portions of PHRs containing information about the health insurance claims while the other confidential medical information, such as medical history of the patient is restricted for such users.

In contrast to the approach presented in [10] that proposes the management of multiple keys by the PHR owners, which eventually leads to overheads at the PHR owner's end, the SeSPHR methodology avoids the overhead by delegating the SRS for setting up the public/private key pairs and producing the decryption keys for the authorized users only. The methodology considers the cloud servers as the untrusted entity and therefore, introduces a semi-trusted server called the Setup and Re-encryption Server (SRS) as the proxy. Proxy Re-encryption based approach is used for the SRS to generate the re-encryption keys for secure sharing of PHRs among the users. The PHRs are encrypted by the patients or PHR owners and only the authorized users having the keys issued by the SRS can decrypt the PHRs. Moreover, the users are granted access to the specific portions of PHRs as deemed important by the PHR owner. The proposed approach is secure as compared to various other constructions used in the sense that the SRS in the proposed framework is never transmitted the PHR data. Instead, the responsibility of the SRS is to manage the keys while the encryption operations are performed by the PHR owners whereas the decryption is performed at the requesting users' end having the valid decryption keys.

The proposed approach also enforces the forward and backward access control. The newly joining members of a particular user group obtain the keys from the SRS. The shared data is encrypted by the keys of the owner only. The access to the data for newly joining member is granted after the approval of the PHR owner. Similarly, a departing user is removed from the ACL and the corresponding keys for that user are deleted. The deletion of the user keys and removal from the ACL results in denial of access to the PHR for any illegitimate access attempts after the user has departed. We also performed the formal analysis of the proposed scheme by using the High Level Petri Nets (HLPN) and the *Z* language. The HLPN is used not only to mimic the system but also offers the mathematical properties that are subsequently employed to investigate the system's behavior. The verification is performed with the Satisfiability Modulo Theories Library (SMT-Lib) and the Z3 solver. The task of verification using the SMT is accomplished by first translating the petri net model into the SMT along with the specific properties and subsequently using the Z3 solver to determine if the properties hold or not.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2018.2854790, IEEE Transactions on Cloud Computing

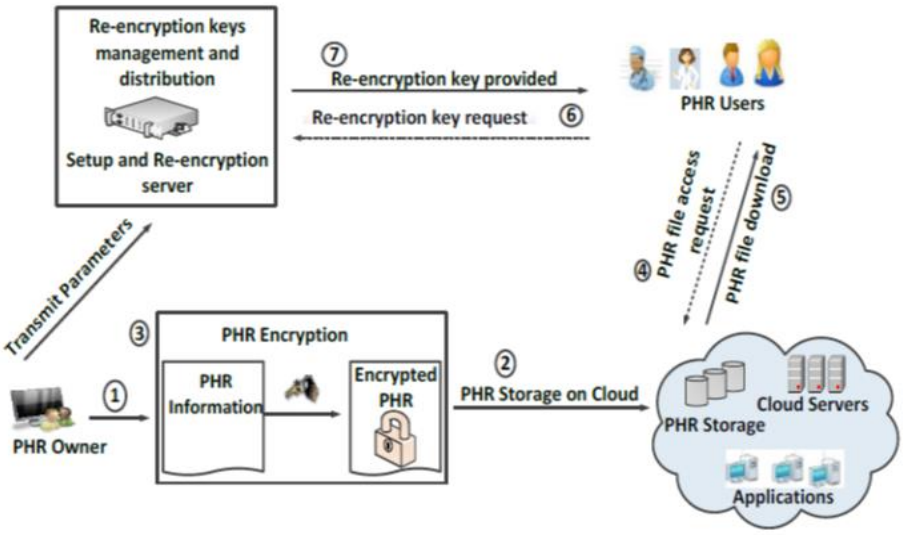

Fig. 1: Architecture of the proposed SeSPHR methodology

The key contributions of the proposed work are given below:

- We present a methodology called SeSPHR that permits patients to administer the sharing of their own PHRs in the cloud.
- The SeSPHR methodology employs the El-Gamal encryption and proxy re-encryption to ensure the PHR confidentiality.
- The methodology allows the PHR owners to selectively grant access to users over the portions of PHRs based on the access level specified in the ACL for different groups of users.
- A semi-trusted proxy called SRS is deployed to ensure the access control and to generate the re-encryption keys for different groups of users thereby eliminating the key management overhead at the PHR owner's end.
- The forward and backward access control is also implemented in the proposed methodology
- Formal analysis and verification of the proposed methodology is performed to validate its working according to the specifications.

The paper is organized as follows. Section 2 presents preliminary concepts of El-Gamal encryption and proxy re-encryption. The proposed methodology called SeSPHR is presented in Section 3 whereas the discussion on the proposed methodology is presented in Section 4. The formal analysis and the verification of the proposed methodology are presented in Section 5. Experimental results are described in Section 6 whereas Section 7 concludes the paper.

2 Preliminaries

The SeSPHR methodology enforces a fine-grained access control and permits the patients or the PHR owners to manage access over their health information. In the proposed methodology, the patients upload the encrypted PHRs by separately encrypting the partitions of PHRs, for example: (i) personal information, (ii) Medical information, (iii) insurance related information, and (iv) prescription information. Moreover, the client application for the PHR also generates the re-encryption parameters that are subsequently transmitted to the SRS. If a user wants to access any portion of the PHR, the user downloads the PHR from the cloud after authentication. It is important to state that still at this point, the user cannot decrypt the PHRs, because the user needs to obtain the corresponding decryption parameters from the SRS. The SRS checks the ACL for the requesting user and determines whether the access to the partition for which the user has requested the decryption parameters is granted by the PHR owner or not. According to the access permissions specified in the ACL, the SRS will generate the corresponding parameters and will send those to the requesting user.

It is important to mention that the scope of this paper is limited to securing the PHR itself. Moreover, it is assumed that communication between the client and SRS is secured by use of standard protocols like, IPSec or SSL. The aforesaid protocols are widely used over the Internet and are fully capable of securing communication. However, communication security is beyond the scope of this paper. The setup, key generation, and re-encryption phases are carried out at SRS. The aforesaid phases are discussed in detail in Section 3.3.

Before the detailed discussion on the proposed scheme for secure sharing of PHRs among different groups of users, we present some important preliminary concepts. Section 2.1 introduces the concept of Personal Health Records (PHRs) whereas Section 2.2 presents a brief introduction about El-Gamal encryption. The preliminary concepts related to the proxy re-encryption are highlighted in Section 2.3.

2.1 Personal Health Records (PHRs)

The PHRs can be defined as the electronic version of pa-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2018.2854790, IEEE Transactions on Cloud Computing

tients' health information, which is controlled by the patients themselves. The PHRs permit patients to manage the information, such as demographics, diagnosis, treatments, monitoring, and self-care.

The PHRs are different from the Electronic Health Records (EHRs) in the sense that the EHRs are managed by the health organizations and contain the information entered by the doctors and the hospital staff instead of patients [8].

2.2 El-Gamal Encryption

El-Gamal encryption system is a public key cryptosystem proposed by T. El-Gamal [17] that is built on Diffie-Hellman key exchange [18]. The difficulty in computing the discrete logarithms establishes El-Gamal encryption system's security. El-Gamal encryption methodology mainly comprises of the steps namely, the initialization, encryption, and decryption [19].

The preliminary details about the El-Gamal encryption are presented below:

Initialization

Given a large prime p and generator g of the multiplicative group $Z_p{}^*$. Select a random secret key x and compute $b = g^x \, mod \, p$. Moreover, (p, b, g) represents the generated public key.

Encryption

The message m is encrypted by the sender by obtaining the receiver's public key (p, b, g) as follows:

$$\gamma = g^x \, mod \, p \tag{1}$$

and,

$$\delta = m * (g^x)^k \tag{2}$$

The encrypted message $E(m) = (\gamma, \delta)$ is sent to the receiver.

Decryption

The encrypted message $E(m)$ after it is received by the receiver is decrypted by means of the private key x and the decryption factor as follows:

$$d = (\gamma^{p-1-x}) \, mod \, p \tag{3}$$

The encrypted message m is recovered as:

$$(D(E(m)) = (d) * \delta \, mod \, p \tag{4}$$

2.3 Proxy Re-encryption

The proxy re-encryption approach employs a third-party having the capability to transfigure the enciphered text that was encrypted for one of the communicating parties to be decrypted by the other user or party. The main operations in the proxy re-encryption include setup, key generation, encryption, and decryption [4].

3 The Proposed SeSPHR Methodology

The proposed scheme employs proxy re-encryption for providing confidentiality and secure sharing of PHRs through the public cloud. The architecture of the proposed SeSPHR methodology is presented in Fig. 1.

3.1 Entities

The proposed methodology to share the PHRs in the cloud environment involves three entities namely: **(a)** the cloud, **(b)** Setup and Re-encryption Server (SRS), and **(c)** the users. Brief description about each of the entities is presented below.

The Cloud: The scheme proposes the storage of the PHRs on the cloud by the PHR owners for subsequent sharing with other users in a secure manner. The cloud is assumed as un-trusted entity and the users upload or download PHRs to or from the cloud servers. As in the proposed methodology the cloud resources are utilized only to upload and download the PHRs by both types of users, therefore, no changes pertaining to the cloud are essential.

Setup and Re-encryption Server (SRS): The SRS is a semi-trusted server that is responsible for setting up public/private key pairs for the users in the system. The SRS also generates the re-encryption keys for the purpose of secure PHR sharing among different user groups. The SRS in the proposed methodology is considered as semi-trusted entity. Therefore, we assume it to be honest following the protocol generally but curious in nature. The keys are maintained by the SRS but the PHR data is never transmitted to the SRS. Encryption and decryption operations are performed at the users' ends. Besides the key management, the SRS also implements the access control on the shared data.

The SRS is independent server that cannot be deployed over a public cloud because of cloud being un-trusted entity. The SRS can be maintained by a trusted third-party organization or by a group of hospitals for convenience of the patients. It can also be maintained by a group of connected patients. However, SRS maintained by hospitals or by a group of patients will generate more trust due to involvement of health professionals and/or self-control over SRS by patients.

Users: Generally, the system has two types of users: **(a)** the patients (owners of the PHR who want to securely share the PHRs with others) and **(b)** the family members or friends of patients, doctors and physicians, health insurance companies' representatives, pharmacists, and researchers. In SeSPHR methodology, the friends or family members are considered as private domain users whereas all the other users are regarded as the public domain users. The users of both the private and public domain may be granted various levels of access to the PHRs by the PHR owners. For example, the users that belong to private domain may be given full access to the PHR, whereas the public domain users, such as physicians, researchers, and pharmacists may be granted access to some specific portions of the PHR. Moreover, the aforementioned users may be allowed full access to the PHRs if deemed essential by the PHR owner. In other words, the SeSPHR methodology allows the patients to exercise the fine-grained access control over the PHRs. All of the users in the system are required to be registered with the SRS to receive the services of the SRS. The registration is based on the roles of the users, for instance, doctor, researcher, and pharmacist.

3.2 The PHR Partitioning

The PHR is logically partitioned into the following four portions:

- Personal Information;
- Medical information;
- Insurance related information;
- Prescription information;

However, it is noteworthy that the above said partitioning is not inflexible. It is at the discretion of the user to partition the PHR into lesser or more number of partitions. The PHRs can be conveniently partitioned and can be represented in formats, for example XML. Moreover, the PHR owner may place more than one partition into same level of access control. Any particular user might not be granted a full access on the health records and some of the PHR partitions may be restricted to the user. For example, a pharmacist may be given access to prescription and insurance related information whereas personal and medical information may be restricted for a pharmacist. Likewise, family/friend may be given full access to the PHR. A researcher might only need the access to the medical records while de-identifying the personal details of the patients. The access rights over different PHR partitions are determined by the PHR owner and are delivered to the SRS at the time of data uploading to the cloud.

3.3 The Working of the Proposed Methodology

The proposed SeSPHR methodology comprises of the steps namely: **(a)** setup, **(b)** key generation, **(c)** encryption, and **(d)** decryption. Each of the steps is discussed below:

Setup

The proposed methodology works on groups G_1 and G_2 with the prime order q. The bilinear mapping of G_1 and G_2 is $G_1 \times G_1 \rightarrow G_2$. A parameter g is a random generator such that $g \in G_1$. The variable Z is another random generator such that $Z = e(g, g) \in G_2$.

Key Generation

The public/private key pairs are generated by the SRS for the set of authorized users. The keys are generated as following:

$$SK_i = x_i\,, PK_i = g^{x_i} \tag{5}$$

where $x_i \in Z_q^*$. The SK_i and PK_i represent the private and public key of user i, respectively. The keys are securely transmitted to the corresponding users.

Encryption

Suppose any patient P needs to upload his/her PHR onto the cloud. The patient client application generates random number(s) equal to the PHR partitions placed in the distinct access level groups by the user. In our case, we consider that all of the four partitions described in Section 3.2 are at different access levels. Therefore, in our case four random variables $r_1, r_2, r_3, r_4 \in Z_q^*$ are generated. The variable r_i is used to encrypt *i-th* partition of the PHR. Each partition is encrypted separately by the client application. The XML format conveniently allows the application to perform encryption/decryption on logical partitions of the PHR. The encryption of the aforesaid partitions of the PHR is performed as follows.

$$C_{per} = Z^{r_1}.PHR_{per} \tag{6}$$

where $\mathrm{PHR_{per}}$ refers only to the personal partition of the PHR and $\mathrm{C_{per}}$ is the semi-encrypted file that contains the personal partition as encrypted text.

$$C_{ins} = Z^{r_2}.PHR_{ins} \tag{7}$$

where PHR_{ins} refers only to the insurance partition of the PHR and C_{ins} is the semi-encrypted file that contains the insurance partition as encrypted text in addition to the C_{per} that was encrypted in the previous step.

$$C_{med} = Z^{r_3}.PHR_{med} \tag{8}$$

where PHR_{med} refers only to the medical information partition of the PHR and C_{med} is the semi-encrypted file that contains the insurance partition as encrypted text in addition to the $\mathrm{C_{per}}$ and $\mathrm{C_{ins}}$ that were encrypted in the previous steps.

$$C = Z^{r_4}.PHR_{pres} \tag{9}$$

where PHR_{pres} refers only to the prescription information partition of the PHR. Here, C represents the complete encrypted file that contains all the partitions in the encrypted form. Therefore, we have not used the subscript with the last step of encryption. It is noteworthy that the sequence of encryption may be changed and the above given sequence is not hard and fast.

In addition to the above stated encryptions, the client also calculates the following parameters.

$$R_{per\_P} = g^{r_1 x_p} \tag{10}$$

$$R_{ins\_P} = g^{r_2 x_p} \tag{11}$$

$$R_{med\_P} = g^{r_3 x_p} \tag{12}$$

$$R_{pres\_P} = g^{r_4 x_p} \tag{13}$$

where x_p is the private key of the patient that is uploading the PHR. The parameter R is used to produce the re-encryption key for the partition indicated in the subscript of each R. The P in the subscript shows that the parameter R is generated by the user P. The completion of the encryption phase is followed by the upload of complete encrypted file C to the public cloud. The parameters R_{per_P}, R_{ins_P}, R_{med_P}, and R_{pres_P} are transmitted to the SRS along with the file identification for which these parameters are generated.

It is noteworthy that a patient after registering with the SRS needs to send at least the following information to get the aforementioned parameters.

- Number of partitions of PHR
- Label of each partition, for instance personal information, medical information, insurance information, and prescription information
- Role that has access to any particular partition (any role may be given access to more than one partitions), like doctors may be given access to medical information
- Initial members of family/friends to give access

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2018.2854790, IEEE Transactions on Cloud Computing

- Default access (if any) in case of new member

Decryption

Suppose a user U desires to access the encrypted PHR (C) uploaded by the patient P. The user U downloads the C directly from the cloud (after the cloud authentication process). Afterwards the user U requests the SRS to compute and send the corresponding R parameters that are used for decryption. The SRS checks the ACL for the requesting user and determines whether the access to the partition for which the user has requested R, is granted by the PHR owner or not. According to the access permissions specified in the ACL, the SRS will generate the corresponding parameters and will send those to the requesting user. In the following text, we will show the generation of R for all of the partitions to clarify the process at a single place. Therefore, we assume that user U has access to all of the partitions. The SRS calculates the re-encryption key and R and transmits it to the user U. The re-encryption keys and R are calculated below:

$$RK_{P\to U} = g^{\frac{x_U}{x_P}} \tag{14}$$

where $RK_{P\to U}$ is the re-encryption key from patient P to user U, x_U and x_P are the private keys of U and P, respectively. Subsequently, the parameters R for all of the partitions corresponding to the user U are calculated according to the following equations.

$$R_{per_U} = e\left(RK_{P\to U}, R_{per\ \ _P}\right) = e\left(g^{\frac{x_U}{x_P}}, g^{r_1 x_P}\right) = e(g,g)^{r_1 x_U} = Z^{r_1\ x_U} \tag{15}$$

where R_{per_U} is the parameter used to decrypt the partition 'personal information' and is applicable for the user U. Similarly, R parameters for other partitions corresponding to user U are calculated in Eq. 16, Eq. 17, and Eq. 18.

$$R_{ins\_U} = e\left(RK_{P\to U}, R_{ins\_P}\right) = e\left(g^{\frac{x_U}{x_P}}, g^{r_2 x_P}\right) = e(g,g)^{r_2 x_U} = Z^{r_2\ x_U} \tag{16},$$

$$R_{med\_U} = e\left(RK_{P\to U}, R_{maed\_P}\right) = e\left(g^{\frac{x_U}{x_P}}, g^{r_3 x_P}\right) = e(g,g)^{r_3 x_U} = Z^{r_3\ x_U} \tag{17},$$

$$R_{pres\_U} = e\left(RK_{P\to U}, R_{pres\_P}\right) = e\left(g^{\frac{x_U}{x_P}}, g^{r_4 x_P}\right) = e(g,g)^{r_4 x_U} = Z^{r_4\ x_U} \tag{18}$$

The above given parameters are provided to the user U that decrypts each of the partitions based on the following equations.

$$PHR_{per} = \frac{C_{per}}{R_{per\_U}^{\frac{1}{x_U}}} \tag{19}$$

$$PHR_{ins} = \frac{C_{ins}}{R_{ins\_U}^{\frac{1}{x_U}}} \tag{20}$$

$$PHR_{med} = \frac{C_{med}}{R_{med\_U}^{\frac{1}{x_U}}} \tag{21}$$

$$PHR_{pres} = \frac{C_{pres}}{R_{pres\_U}^{\frac{1}{x_U}}} \tag{22}$$

The decryption of the last partition will result in complete PHR in plain form. As mentioned earlier, the user will obtain the R parameter from the SRS for only the partition(s) for which access is allowed to the requesting user.

Newly joining members

A new member can enter into the group by registering with the SRS. The new members are registered to the system by the SRS according to their roles and the approval for registering the new members is granted by the PHR owner. The SRS generates the public/private key pairs. The keys are securely transmitted to the users (new members).

Initially, at the time of registration, the new members are given the default access right as specified by the PHR owner depending upon the type of group in which the newly joining member is registered. However, if a certain user needs the extended access rights over the PHRs, then such rights are granted after the approval of the PHR owner. Moreover, a user in the family/friend category can only be added by the approval of the PHR owner. The ACL is updated after the registration of the new user along with the date of joining. The joining user is granted access to the files from the date of joining unless specified otherwise by the PHR owner.

To determine the role of doctor/researcher/pharmacist, X.509 role based certificates, also known as X.509 attribute certificates, can be used. X.509 attribute certificates bind the role of the individual with the identity, such as doctor, engineer, and pharmacist to carry out authorization function after successful execution of authentication mechanism. Therefore, in context of SeSPHR X.509 attribute certificates can effectively identify role of aforesaid users in open connectivity system. However, as described earlier family and friends can join the system and can gain access after approval of the PHR owner.

Departing User

If due to any reason any of the users of the PHR is required to depart, then the PHR owner notifies the SRS to revoke the granted access. The SRS deletes the keys corresponding to the departing user and removes the user from the ACL. The system does not need to change the keys for every user and also it does not require the re-encryption of entire data.

4 Discussion On The SeSPHR Methodology

The proposed methodology provides the following services for the PHRs shared over the public cloud.

- Confidentiality;
- Secure PHR sharing among the groups of authorized users;
- Securing PHRs from unauthorized access of valid insiders;
- Backward and forward access control;

In the proposed methodology, the cloud is not considered a trusted entity. The features of cloud computing paradigm, such as shared pool of resources, multi tenancy, and virtualization might generate many sorts of insid-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2018.2854790, IEEE Transactions on Cloud Computing

er and outsider threats to the PHRs that are shared over the cloud. Therefore, it is important that the PHRs should be encrypted before storing at the third-party cloud server. The PHR is first encrypted at the PHR owner's end and is subsequently uploaded to the cloud. The cloud merely acts as a storage service in the proposed methodology. The encryption keys and other control data are never stored on the cloud. Therefore, at the cloud's end the confidentiality of the data is well achieved. Even if the unauthorized user at the cloud by some means obtains the encrypted PHR file, the file cannot be decrypted because the control data does not reside at the cloud and the confidentiality of the PHR is ensured.

The uploaded PHRs are encrypted by the owner and the rest of the users obtain the plain data by utilizing the re-encryption key that is computed by the SRS. The SRS generates the re-encryption parameters only for the allowed partitions corresponding to the requesting user. Therefore, the privacy of the entire system is not disturbed by a compromised legitimate group member.

The ACL specifies all the rights pertaining to each of the users and are specified by the PHR owner. The rights are specified based on the categories of the users and are extended/limited by the approval of the PHR owner. The SRS calculates and sends the re-encryption parameters based on the specified rights on the partitions. Therefore, even the legitimate users cannot access the unauthorized partition.

The newly joining member obtains the keys from the SRS. The shared data is encrypted by the keys of the owner only. The access to the data for newly joining member is granted by the approval of the SRS. Moreover, introducing a new key in the system does not require re-encryption of the whole data. Similarly, a departing user is removed from the ACL and the corresponding keys are deleted. The deletion of the user keys and removal from the ACL results in denial of access to the PHR for any illegitimate access attempts afterwards. Therefore, the proposed methodology is effectively secure because it restricts the access of departing users (forward access control) and permits the new users to access the past data (backward access control).

The SRS is considered a semi-trusted authority that is honest but curious. In general, the SRS is assumed to follow the protocol honestly. Although the SRS generates and stores the key pair for each of the users, the data whether encrypted or plain is never transmitted to the SRS. The SRS is only responsible for key management and re-encryption parameters generation. Moreover, the access control is also enforced by the SRS. However, maintenance of the SRS is the limitation and challenge of the proposed methodology.

5 Formal Analysis and Verification

Before presenting the formal analysis of the SeSPHR methodology, brief introduction about the HLPN, SMT-Lib, and Z3 are presented. Section 5.1 presents preliminaries about the HLPN, whereas the basics about the SMT-Lib and Z3 solver are presented in Section 5.2. The formal analysis of the SeSPHR methodology is presented in Section 5.3.

5.1 High Level Petri Nets (HLPNs)

The petri nets are the tools that are employed to graphically and mathematically model the systems [20]. The petri nets can model a variety of systems that can be characterized as the parallel, concurrent, distributed, non-deterministic, asynchronous, and stochastic [21]. To model the working of the SeSPHR methodology, we used the HLPN, which is a variation of the traditional petri nets. The HLPN is a structure comprising of 7-tuples and is characterized as $N = (P, T, F, \varphi, R, L, M_0)$ [21]. Each of the tuples is defined below:

- P represents the set of places;
- T characterizes the transitions set such that $P \cap T = \emptyset$;
- F is used to represent the flow relation and is given by $F \subseteq (P \times T) \cup (T \cup P)$.
- The data type mapping of a particular place P is given by the mapping function φ such that $\varphi: P \rightarrow Type$;
- R states the rules that are used to map the transitions T;
- L represents the label used for mapping F to the L;
- The initial marking is given by M_0.

The variables (P, T, F) provide details regarding the petri net's structure whereas the variables (φ, R, L) represent the static information. In other words, the semantics of the information never change all through the system.

Each of the places in HLPN has different types of tokens. The enabling transitions in the HLPN only occur when the pre-conditions for that transition hold. In addition, to enable a certain transition the variables from the inward flows are utilized. Similarly, to fire the transitions, the variables from outgoing flows are used by the post-conditions.

5.2 The Z3 Solver and SMT-Lib

Satisfiability Modulo Theory (SMT) is employed to validate the satisfiability of formulas applied on various theories of interest. Originated from the theory of Boolean Satisfiability Solvers (SAT), the SMT-Lib offers an input platform and benchmarking framework for system evaluation [22]. The SMT has also been employed in various application areas, for example deductive software verification [20]. Along with the SMIT-Lib, we also used Z3 solver. The Z3 solver is theorem prover and an automated satisfiability checker that is developed at the Microsoft Research. Having support for a diverse range of theories, the Z3 solver focuses on unraveling the problems that rise in software verification. Moreover, the Z3 solver determines the satisfiability of certain set of formulas for SMT-Lib in-built theories [23].

5.3 Formal Analysis and Verification

Formal verification is the procedure that is used to determine the precision and correctness of a particular sys-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2018.2854790, IEEE Transactions on Cloud Computing

tem. We employed the bounded model checking [24] technique for verifying the scheme and used the SMT-Lib and Z3 solver. A Boolean formula is said to be satisfiable only if any of the system inputs that are acceptable drive the underlying state transition system to the state that terminates after finite sequence of state transitions [20]. The bounded checking process includes various tasks namely: **(a)** the specification, **(b)** model representation, and **(c)** verification [20]. Specification is the system's description stating the rules that the system must satisfy whereas the model representation refers to the mathematical modeling of the entire system. Likewise, the verification of the model involves the utilization of a tool to determine whether a specification is satisfied by the system or not.

Fig. 2 presents the HLPN model for the SeSPHR. Table I and Table II present the data types and mappings, respectively. In HLPN model presented in Fig. 2, all the transitions belonging to set T are represented by the rectangular black boxes whereas the circles represent the palaces belonging to set P.

The SeSPHR methodology was discussed in detail in Section 3. The system starts with the setup and key generation phase. The setup and key generation process is represented by transition *Gen_Keys* and the following equation maps to it.

$$R\ (Gen\_keys) = \forall\, x_1\ \in X_1 |$$
$$x_1[4] \coloneqq Gen\_\, g(x_1[1])\ \wedge x_1[5] \coloneqq Gen\ \_{Z_q}^{*}(x_1[1]\ \wedge x_1[2] \coloneqq Gen\_SK_i(x_1[1] \wedge x_1[3] \coloneqq Gen\_PK_i(x_1[1]) \wedge {X_1}' = X_1 \cup \{x_1\} \quad (23)$$

The transition *send_keys* represents the process of delivering the keys to the users in the system. The following rule maps to the transition.

$$R\ (send\_keys) = \forall\, x_2 \in X_2, \forall\, x_3\ \in X_3 |$$
$$x_3[1] \coloneqq x_2[1] \wedge x_3[2] \coloneqq x_2[2] \wedge x_3[3] \coloneqq x_2[3] \wedge x_3[6] \coloneqq x_2[6] \wedge x_3[8] \coloneqq x_2[4] \wedge {X_3}' = X_3 \cup \{x_3\} \quad (24)$$

Whenever the encryption of the PHRs before uploading to the cloud is required, a random number is generated by the PHR owner according to the number of partitions in the PHR. The transition *Gen_r_i* and the associated rule are given as below.

$$R\ (Gen\_r_i) = \forall x_4 \in X4 |$$
$$x_4[5] \coloneqq Gen\_r_i(x_4[4]) \wedge {X_4}' = X_4 \cup \{x_4\} \quad (25)$$

After the generation of the random number the encryption performed as following.

$$R\ (Encrypt\ \_P_i) = \forall x_5 \in X5 |$$
$$x_5[7] \coloneqq encrypt\ (x_5[4], x_5[5], x_5[6]) \wedge {X_5}' = X_5 \cup \{x_5\} \quad (26)$$

The R parameters are calculated by the PHR owner used for generating re-encryption keys according to the process described in Section 3. The transition $Compt\_R$ represents the process and maps to the following rule.

$$\mathrm{R}(Compt\_R) = \forall x_6 \in X_6 |$$
$$x_6[9] \coloneqq comp\_\, R_i\ (x_6[2], x_6[8], x_6[5]) \wedge\ {X_6}' = X_6 \cup \{x_6\} \quad (27)$$

TABLE I: DATA TYPES AND DESCRIPTION

Data Type	Description
G	A number belonging to group G_1
Zq^*	A random number generator
Z	Number $e(g,g)$ that belongs to G_2
Ui	The number representing user i
Pi	A number representing *i-th* partition of PHR
SK_i	Secret key of a certain user i
PK_i	Public key of a certain user i
r_i	*i-th* random number used to secure *i-th* PHR partition
C	Encrypted PHR
R_i	Parameter R for decrypting *i-th* PHR partition

TABLE II: MAPPINGS AND PLACES

Place	Mapping
φ *(SRS)*	$\mathbb{P}\ ({U_i}^1 \times {SK_i}^2 \times {PK_i}^3 \times g^4 \times {Z_q^*}^5 \times Z^6 \times {R_i}^7 \times {P_i}^8)$
φ *(User)*	$\mathbb{P}\ ({U_i}^1 \times {SK_i}^2 \times {PK_i}^3 \times {P_i}^4 \times {r_i}^5 \times Z^6 \times C^7 \times g^8 \times {R_i}^9)$
φ *(Cloud)*	$\mathbb{P}(C)$

After the completion of encryption process, the encrypted data is transmitted to the cloud server. The following transition and equation represents the process.

$$R(send\_C) = \forall x_7 \in X_7, \forall x_8 \in X_8$$
$$x_8[1] \coloneqq x_7[7] \wedge {X_8}' = X_8 \cup \{x_8\} \quad (28)$$

The calculated R parameters are sent to the SRS. The transition $send\_R_i$ shows the associated rule in the following.

$$R(send\_R_i) = \forall\, x_9 \in X_9, \forall\ x_{10} \in X_{10} |$$
$$x_{10}[7] \coloneqq x_9[9] \wedge x_{10}[8] \coloneqq x_9[4] \wedge {X_{10}}' = X_{10} \cup \{x_{10}\} \quad (29)$$

The encrypted PHR is downloaded by the requesting user from the cloud according to the below transition and associated rule:

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2018.2854790, IEEE Transactions on Cloud Computing

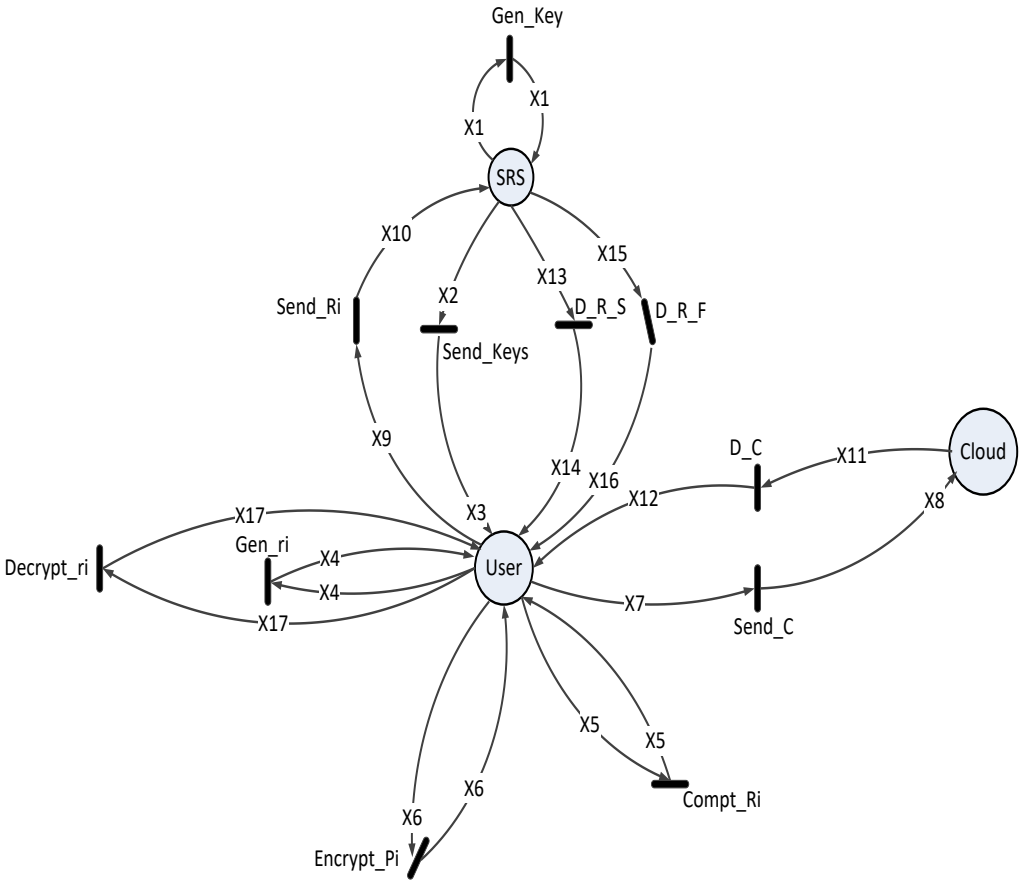

Fig. 2. The HLPN model of the proposed SeSPHR methodology

$$R(D\_C) = \forall x_{11} \in X_{11}, \forall x_{12} \in X_{12} | x_{12}[7] := x_{11}[1] \wedge X_{12}' = X_{12} \cup \{x_{12}\} \quad (30)$$

For decryption, the requesting user needs re-encrypted parameter. The user requests SRS for the re-encryption parameter. The SRS after checking the ACL for the requesting user determines whether the user has been granted access to uploaded message, the manager computes the re-encryption parameters and sends to the requesting user. This is done in the following rule:

$$R(D\_R\_S) = \forall x_{13} \in X_{13}, \forall x_{14} \in X_{14} | x_{13}[1] = x_{14}[1] \wedge x_{13}[8] = x_{14}[4] \wedge x_{14}[9] := x_{13}[7] \wedge X_{13}' = X_{13} \cup \{x_{13}\} \wedge X_{14}' = X_{14} \cup \{x_{14}\} \quad (31)$$

If the user requesting the access does not belong to the access list, then the request for re-encryption parameters fails and is shown is the rule below:

$$R(D\_R\_F) = \forall x_{15} \in X_{15}, \forall x_{16} \in X_{16} | x_{15}[1] \neq x_{16} \vee x_{15}[8] \neq x_{16}[4] \wedge X_{15}' = X_{15} \wedge X_{16}' = X_{16} \quad (32)$$

After receiving the required parameters, the user decrypts the PHR as per following equation.

$$R(Decrypt\_C) = \forall x_{17} \in X_{17} | x_{17}[4] = decrypt(x_{17}[7], x_{17}[9]) \wedge X_{17}' = X_{17} \cup \{x_{17}\} \quad (33)$$

Verification of Properties

To determine whether the presented SeSPHR scheme operates according to the specifications, we performed verification of the properties. The following properties pertinent to the working of SeSPHR methodology are verified:

- A valid system user cannot obtain the re-encryption parameters for a PHR partition for which access is not granted to the user.
- The encryption and decryption is performed correctly as specified by the system.
- Any unauthorized user is not able to generate the re-encryption parameters and decrypt the PHR.

The translation of the described model to SMT-Lib was performed and verification was done through the Z3 solver. The solver exhibited the practicality of the model in accordance with the stated properties. After encryption, the Z3 solver in total consumed 0.07 seconds to upload

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2018.2854790, IEEE Transactions on Cloud Computing

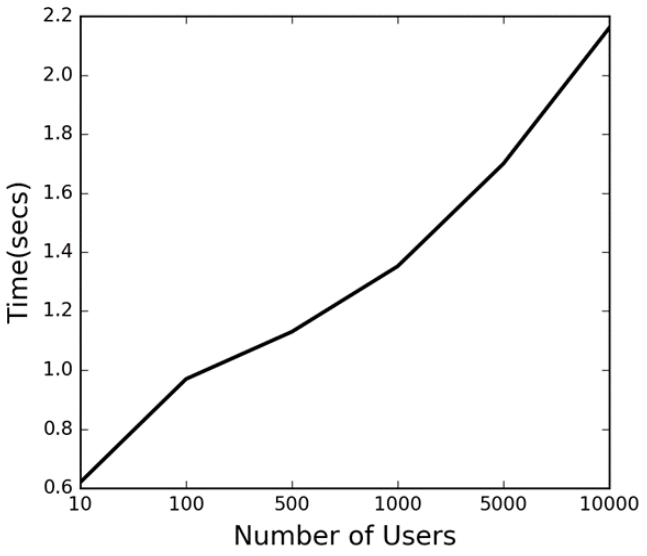

Fig. 3. Time consumption for key generation

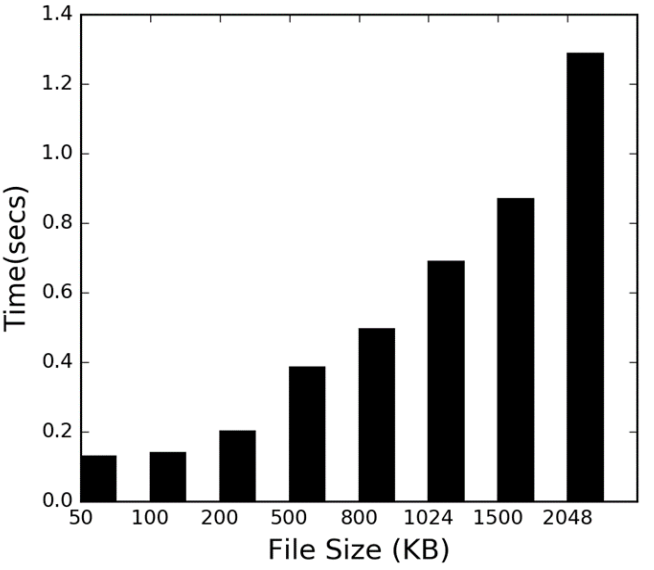

Fig. 4. Time consumption for encryption

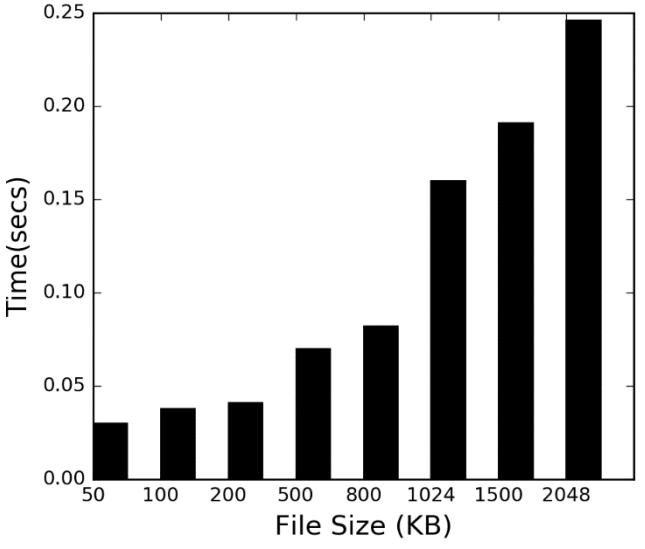

Fig. 5. Time consumption for decryption

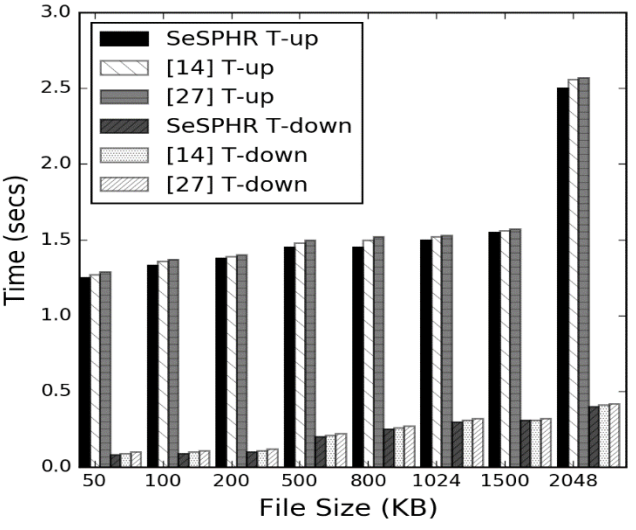

Fig. 6. Turnaround time comparison of SeSPHR with [14] and [27]

user data and followed by a subsequent download and decrypt operation for a different user in the group.

6 Performance Evaluation

We evaluated the performance of the SeSPHR methodology from multiple perspectives, such as key generation times, encryption and decryption time, and turnaround time. Also, we compared the complexity of the SeSPHR methodology with other methodologies. The details of the experimental setup and results are presented in subsequent sections.

6.1 Experimental Setup

The performance of the SeSPHR methodology to securely share the PHRs among different types of users was evaluated by developing a client application in Java. The entities of the proposed SeSPHR methodology include the cloud, SRS, and the users. We used Amazon Simple Storage Services (Amazon S3) [25] as our cloud storage. The Amazon Web Services SDK (AWS) for Java was used to obtain the Java APIs for AWS services. The SRS that actually is responsible for producing the public/private key pairs and the re-encryption keys is implemented as a third-party server. Java Pairing Based Cryptography (JPBC) library was used for the encryption of PHR data [26]. From the JPBC library we used Type A pairing that is constructed on the curve $y^2 = x^3 + x$ on the prime field F_q. The prime number q is set to be of 64 bytes or 512 bits. Due to the fixed size of the prime number, the encryption and decryption process was carried out in the

chunks of 64 bytes. The experiments were conducted on the computer having Intel® Core i7-2600 CPU @ 3.40 GHz with 8 GB memory.

6.2 Results

The performance of the SeSPHR methodology was evaluated regarding generation, encryption, decryption, and turnaround time. The results for each of the above evaluation criteria are discussed below.

6.2.1 Key Generation

As stated earlier in Section 3 that the responsibility of the SRS is to generate the private/public key pairs for the users belonging to the set of authorized users. However, the key generation time for the systems with large numbers of users may affect the overall performance of the system. Therefore, we appraised the performance of the SeSPHR in terms of the time consumed for the key generation step for different number of user. The time consumption for generating keys for 10, 100, 500, 1000, 5000, and 10,000 users in presented in Fig. 3.

Contrary to the general trend of increased key generation time when the number of users increases, it can be seen in Fig. 3 that with the increased number of users, the corresponding increase in the key generation time is not uniform. For example, the time consumption to generate keys for 10 users is 0.6 second whereas for 100 users, the key generation time increases to 0.97 second. Likewise, the key generation time for 10,000 users is observed 2.16 seconds, which is also very reasonable considering the high number of users. The key generation time for newly joining members is also minimal because such members join occasionally and generating keys for a single user is indeed an efficient process.

6.2.2 Encryption and Decryption

The time consumption of the SeSPHR methodology to encrypt and decrypt the data files of varying sizes is also evaluated. The file sizes used for the experimentation are 50 KB, 100 KB, 200 KB, 500 KB, 800 KB, 1024 KB, 1500 KB, and 2048 KB. The time consumption for both the encryption and decryption operations for the files of aforementioned sizes is shown in Fig. 4 and Fig. 5, respectively. From Fig. 4 we can see that with the increase in PHR file size, the encryption time also increases. For example, the encryption time for the file of size 50 KB is 0.13 second whereas the encryption time for the 2 MB file is 1.289 seconds. On the contrary, the time required for decryption of the PHR files was considerably less than the encryption time. An average decrease of 24.38% in decryption time was observed as compared to the encryption time.

6.2.3 Complexity Analysis

We also compared the SeSPHR methodology in terms of key distribution, public and private key sizes, and decryption complexity with the approaches presented in [14] and [27]. Table IV shows the comparison of the SeSPHR with the abovementioned schemes. The definitions of the notations used in Table IV are presented in Table III. The owners are responsible for encrypting the data for both the users of personal/private domain and the public domain. Typically, the users in the personal/private domain are fewer than the public domain users because the personal domain only contains the families or friend of the patients whereas the public domain users include doctors, researchers, pharmacist and any other users authorized by the PHR owner. The key distribution complexity of the

TABLE III: SYMBOLS AND DEFINITIONS

Symbol	Description
PG	Private group
PuG	Public group
PSD	Personal domain
PUD	Public domain
M	Plain text length
$\mathbb{A}$	Universe of role attributes
$\mathcal{A}$	Data attribute universe
$\mathbb{A}_u$	User u's set of data attributes
P	Number of processors
N	Number of bits in the keys
M	Number of blocks in the text
$\mathcal{A}^C$	Set of role attributes associated with the ciphertext C
$\mathbb{A}^C$	Set of data attributes associated with the ciphertext C
N_i	number of PAAs (public attribute authorities (PAA)) in the *i-th* PUD
$\mathcal{A}_u$	User data attributes set of user u.

SeSPHR for users of personal domain is equal to the other comparison approaches i.e. $O(1)$ whereas for public domain users it is *O(PuG/p)*. The public and private key sizes used in SeSPHR are fixed whereas in the approaches presented in [14] and [27], the key sizes are dependent upon the universe of role attributes and da ta attributes for different users. Decryption complexity of the SeSPHR depends upon the product of text size (number of 64 bytes blocks) and square of bits in the keys. The complexity of the scheme presented in [14] is $O(1)$ as only one bilinear pairing occurs at the server in that technique during decryption phase. However, for the scheme presented in [27] [29], the decryption time complexity is dependent on the intersection of the role attributes in the user set and the universal set of the role attributes.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2018.2854790, IEEE Transactions on Cloud Computing

TABLE IV: COMPARISON OF SESPHR WITH OTHER APPROACHES

	SeSPHR			[14]			[27]		
Key Distribution	*O(PG/P)* (private group)	*O(1)* (patient)	*O(PuG/p)* (Public group)	*O(PSD)* (Owner)	*O(1)* (User)	*O(PUD)* (Public group)	*O(PSD)* (Owner group)	*O(1)* (User)	$O(\sum_{i=1}^{m} PUD_i)$ (Public group)
Public Key size	1024 bits			$\|\mathbb{A}\|_k + N_i$ (PUDk)	$\|\mathcal{A}\| + 1$ (Owner)		$\cup \|\mathbb{A}\|_k$ PUD	$\|\mathcal{A}\|$ (Owner)	
Private Key size	512 bits			$\|\mathbb{A}_u\| + 1$ (Public User)	$\|\mathcal{A}_u\| + 1$ (personal user)		$\mathbb{A}_u$ (Public user)	$\|\mathcal{A}_u\|$ (Personal user)	
Decryption complexity	$O(n^2 \times m)$			*O(1)* (w/delegation)			O($\mathcal{A}_u \cap \mathcal{A}^C$) or O($\mathbb{A}_u \cap \mathbb{A}^C$)		

6.2.4 Turnaround Time

The performance of the SeSPHR scheme was also evaluated with schemes in [14] and [27] in terms of its turnaround time for both the encryption and decryption operations. The turnaround time is the performance evaluation metric to evaluate encryption schemes for the cloud [34]. The turnaround time for encryption is given as:

$$T_{T-up} = t_{Enc} + t_{up} \quad (34)$$

where t_{Enc} and t_{up} respectively are the times for encryption and upload of the PHRs onto the cloud. Similarly, the turnaround time for decryption operation is calculated as:

$$T_{T-down} = t_{Dec} + t_{down} \quad (35)$$

where t_{Dec} and t_{down} represent the decryption time and the download time, respectively. The turnaround time for both the $T_{T-up}$and $T_{T-down}$are presented in Fig 6. It can be observed from Fig. 6 that the turnaround time $T_{T-down}$for a file of certain size is far less time than the $T_{T-up}$of the corresponding file. The reason for the T_{T-up} being significantly higher than the T_{T-down} is that T_{T-up} includes t_{up}, the time to upload the PHRs on the cloud that by itself requires more time. Therefore, the upload time significantly affects the turnaround time T_{T-up} for the encryption operation. One of the key observation we had made during our study that SeSPHR and schemes in [14] and [27] have negligible difference in turnaround times of encryption and decryption. The scheme in [14] differs from the SeSPHR as the scheme uses proxy re-encryption technique to re-encrypt the PHRs after the revocation of the users. Therefore, the SeSPHR scheme provide the turnaround time slightly smaller than the scheme in [14] and [27]. The results have shown that that actual barrier in the turnaround time are encryption and decryption procedures and uploading and downloading time have minimum effect on the turnaround time.

7 RELATED WORK

In this section, the existing works that relate to the proposed work are presented. The authors in [28] used public key encryption based approach to uphold the anonymity and unlinkability of health information in semi-trusted cloud by separately submitting the Personally Identifiable Information (PII). The patients encrypt the PHRs by the patients through the public key of the Cloud Service Provider (CSP) and the CSP decrypts the record using the private key, stores the health record and the location of the file (index), and subsequently encrypts them through the symmetric key encryption. The administrative control of the patient on the PHRs is maintained by pairing the location and the master key. However, a limitation of the approach is that it allows the CSP to decrypt the PHRs that in turn may act maliciously. On the other hand, we introduced a semi-trusted authority called the SRS that re-encrypts the ciphertext generated by the PHR owner and issues keys to the users that request access to the PHRs.

Chen *et al*. [12] introduced a method to exercise the access control dynamically on the PHRs in the multi-user cloud environment through the Lagrange Multiplier using the SKE. Automatic user revocation is the key characteristics of the approach. To overcome the complexities of

the key management, a partial order relationship among the users is maintained. However, the scheme requires the PHR owners to be online when the access is to be granted or revoked. Contrary to the scheme presented in [12], our proposed approach does not require the PHR owners to be online to grant the access over PHRs. Instead the semi-trusted authority determines the access privileges for users and after successful authorization, calculates the re-encryption keys for the users requesting the access.

The authors in [29] used a Digital Right Management (DRM) based approach to offer patient-centric access control. The authors employed the Content Key Encryption (CKE) for encryption and the users with the lawful license are permitted to access the health-data. First proxy re-encryption methodology was proposed in [33]. The policy in [33] is based on ciphertext and the size of the ciphertext increases linearly with multi-use use whereas our policy of our technique is based on keys and it doesn't affect the size of the ciphertext. This is due to the fact that the [33] requires the re-encryption step that is lacking in our methodology. An approach to securely share the PHRs in multi-owner setting, which is divided into diverse domains using the Attribute Based Encryption (ABE) is presented by Li *et al.* [14]. The proposed methodology is based on the methodology originally presented in [33]. The approach uses proxy re-encryption technique to re-encrypt the PHRs after the revocation of certain user(s). In the approach, the intricacies and cost of key management have been effectively minimized and the phenomenon of on-demand user revocation has been improved. Despite its scalability, the approach is unable to efficiently handle the circumstances that require granting the access rights on the basis of users' identities. Xhafa *et al*. [30] also used Ciphertext Policy ABE (CP-ABE) to ensure the user accountability. Besides protecting the privacy of the users, the proposed approach is also capable of identifying the users that malfunction and distribute the decryption keys to other users illegitimately.

An approach to concurrently ensure the fine-grained access and confidentiality of the healthcare data subcontracted to the cloud servers is presented in [10]. The expensive tasks of data files re-encryption, update of secret keys, and restricting the revoked users to learn the data contents are addressed through the proxy re-encryption, Key Policy ABE (KP-ABE), and lazy re-encryption. The cloud servers are delegated the tasks of re-encryption of data files and subsequent storage to the cloud environment. However, in the proposed framework the data owner is also assumed as a trusted authority that manages the keys for multiple owners and multiple users. Therefore, the inefficiencies would occur at the PHR owners' end to manage multiple keys for different attributes for multiple owners. Our approach avoids the overhead because the tasks of key generation and key distribution to different types of users are performed by the semi-trusted authority. The authors in [31] and [32] also used the proxy re-encryption based approaches to offer fine-grained access control. Our proposed framework permits the PHR encryption by the owners before storing at the cloud and introduces a semi-trusted authority that re-encrypts the ciphertext without learning about the contents of the PHRs. Only the authorized users having the decryption keys issues by the semi- trusted authority can decrypt the PHRs.

8 Conclusions

We proposed a methodology to securely store and transmission of the PHRs to the authorized entities in the cloud. The methodology preserves the confidentiality of the PHRs and enforces a patient-centric access control to different portions of the PHRs based on the access provided by the patients. We implemented a fine-grained access control method in such a way that even the valid system users cannot access those portions of the PHR for which they are not authorized. The PHR owners store the encrypted data on the cloud and only the authorized users possessing valid re-encryption keys issued by a semi-trusted proxy are able to decrypt the PHRs. The role of the semi-trusted proxy is to generate and store the public/private key pairs for the users in the system. In addition to preserving the confidentiality and ensuring patient-centric access control over the PHRs, the methodology also administers the forward and backward access control for departing and the newly joining users, respectively. Moreover, we formally analyzed and verified the working of SeSPHR methodology through the HLPN, SMT-Lib, and the Z3 solver. The performance evaluation was done on the on the basis of time consumed to generate keys, encryption and decryption operations, and turnaround time. The experimental results exhibit the viability of the SeSPHR methodology to securely share the PHRs in the cloud environment.

Acknowledgements

Samee U. Khan's work was supported by (while serving at) the National Science Foundation (NSF). Any opinion, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

References

[1] K. Gai, M. Qiu, Z. Xiong, and M. Liu, "Privacy-preserving multi-channel communication in Edge-of-Things," Future Generation Computer Systems, 85, 2018, pp. 190-200.

[2] K. Gai, M. Qiu, and X. Sun, "A survey on FinTech," *Journal of Network and Computer Applications*, 2017, pp. 1-12.

[3] A. Abbas, K. Bilal, L. Zhang, and S. U. Khan, "A cloud based health insurance plan recommendation system: A user centered approach, "*Future Generation Computer Systems*, vols. 43-44, pp. 99-109, 2015.

[4] A. N. Khan, ML M. Kiah, S. A. Madani, M. Ali, and S. Shamshirband, "Incremental proxy re-encryption scheme for mobile cloud computing environment," *The Journal of Supercomputing*, Vol. 68, No. 2, 2014, pp. 624-651.

[5] R. Wu, G.-J. Ahn, and H. Hu, "Secure sharing of electronic health records in clouds," In *8th IEEE International Conference on Collaborative Computing: Networking, Applications and Work-*

sharing (CollaborateCom), 2012, pp. 711-718.

[6] A. Abbas and S. U. Khan, "A Review on the State-of-the-Art Privacy Preserving Approaches in E-Health Clouds," *IEEE Journal of Biomedical and Health Informatics,* vol. 18, no. 4, pp. 1431-1441, 2014.

[7] M. H. Au, T. H. Yuen, J. K. Liu, W. Susilo, X. Huang, Y. Xiang, and Z. L. Jiang, "A general framework for secure sharing of personal health records in cloud system," Journal of Computer and System Sciences, vol. 90, pp, 46-62, 2017.

[8] J. Li, "Electronic personal health records and the question of privacy," *Computers*, 2013, DOI: 10.1109/MC.2013.225.

[9] D. C. Kaelber, A. K. Jha, D. Johnston, B. Middleton, and D. W. Bates, "A research agenda for personal health records (PHRs)," *Journal of the American Medical Informatics Association*, vol. 15, no. 6, 2008, pp. 729-736.

[10] S. Yu, C. Wang, K. Ren, and W. Lou, "Achieving secure, scalable and fine-grained data access control in cloud computing," in *Proceedings of the IEEE INFOCOM*, March 2010, pp. 1-9.

[11] S. Kamara and K. Lauter, "Cryptographic cloud storage," *Financial Cryptography and Data Security*, vol. 6054, pp. 136–149, 2010.

[12] T. S. Chen, C. H. Liu, T. L. Chen, C. S. Chen, J. G. Bau, and T.C. Lin, "Secure Dynamic access control scheme of PHR in cloud computing," *Journal of Medical Systems*, vol. 36, no. 6, pp. 4005–4020, 2012.

[13] K. Gai, M. Qiu, "Blend arithmetic operations on tensor-based fully homomorphic encryption over real numbers," *IEEE Transactions on Industrial Informatics*, 2017, DOI: 10.1109/TII.2017.2780885..

[14] M. Li, S. Yu, Y. Zheng, K. Ren, and W. Lou, "Scalable and secure sharing of personal health records in cloud computing using attribute-based encryption," *IEEE Transactions on Parallel and Distributed Systems*, 2013, vol. 24, no. 1, pp. 131–143.

[15] "Health Insurance Portability and Accountability," http://www.hhs.gov/ocr/privacy/hipaa/administrative/privacyrule/, accessed on October 20, 2014.

[16] Z. Xiao and Y. Xiao, "Security and privacy in cloud computing," *IEEE Communications Surveys and Tutorials*, vol. 15, no. 2, pp. 1–17, Jul. 2012.

[17] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," in *Proceedings of CRYPTO 84 on Advances Cryptology*, 1985, pp. 10-18.

[18] W. Diffie, and M. E. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory,* vol. 22, no. 6, 1976, pp. 644-654.

[19] D. Thilakanathan, S. Chen, S. Nepal, R. Calvo, and L. Alem, "A platform for secure monitoring and sharing of generic health data in the Cloud," *Future Generation Computer Systems*, vol. 35, 2014, pp. 102-113.

[20] S. U. R. Malik, S. U. Khan, and S. K. Srinivasan, "Modeling and Analysis of State-of-the-art VM-based Cloud Management Platforms," *IEEE Transactions on Cloud Computing,* vol. 1, no. 1, pp. 50-63, 2013.

[21] T. Murata, "Petri Nets: Properties, Analysis and Applications, "*Proceedings of the IEEE*, vol. 77, no. 4, pp. 541-580, Apr. 1989.

[22] L. D. Moura and N. Bjørner. "Satisfiability modulo theories: An appetizer." In Formal Methods: Foundations and Applications, Springer Berlin Heidelberg, 2009, pp. 23-36.

[23] S. U. R. Malik, S. K. Srinivasan, S. U. Khan, and L. Wang, "A Methodology for OSPF Routing Protocol Verification," in *12th International Conference on Scalable Computing and Communications (ScalCom)*, Changzhou, China, Dec. 2012.

[24] A. Biere, A. Cimatti, E. Clarke, O. Strichman, and Y. Zhu, "Bounded Model Checking," *Advances in Computers*, vol. 58, 2003, pp. 117-148.

[25] "Amazon S3," http://aws.amazon.com/s3/, accessed on November 14, 2014.

[26] A. D. Caro, and V. Iovino, "jPBC: Java pairing based cryptography," in *IEEE Symposium on Computers and Communications (ISCC)*, 2011, pp. 850-855.

[27] L. Ibraimi, M. Asim, and M. Petkovic, *Secure management of personal health records by applying attribute-based encryption*, Technical Report, University of Twente, 2009.

[28] J. Pecarina, S. Pu, and J.-C. Liu, "SAPPHIRE: Anonymity for enhanced control and private collaboration in healthcare clouds," in *Proceedings of the 4th IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, 2012, pp. 99–106.

[29] M. Jafari, R. S. Naini, and N. P. Sheppard, "A rights management approach to protection of privacy in a cloud of electronic health records," in *11th annual ACM workshop on Digital rights management*, October 2011, pp. 23-30.

[30] F. Xhafa, Fatos, J. Feng, Y. Zhang, X. Chen, and J. Li, "Privacy-aware attribute-based PHR sharing with user accountability in cloud computing," *The Journal of Supercomputing*, 2014, pp. 1-13.

[31] C. Leng, H. Yu, J.Wang, and J. Huang, "Securing personal health records in the cloud by enforcing sticky policies," *Telkomnika Indonesian Journal of Electrical Engineering*, vol. 11, no. 4, pp. 2200–2208, 2013.

[32] D. H Tran, N. H.-Long, Z. Wei, N. W. Keong, "Towards security in sharing data on cloud-based social networks," in *8th International Conference on Information, Communications and Signal Processing (ICICS)*, 2011, pp. 1-5.

[33] X. Liang, Zhenfu Cao, Huang Lin, and Jun Shao. "Attribute based proxy re-encryption with delegating capabilities." In Proceedings of the 4th International Symposium on Information, Computer, and Communications Security, pp. 276-286. ACM, 2009.

[34] A. N. Khan, M.L. M. Kiah, S. U. Khan, Sajjad A. Madani, and Atta R. Khan. "A study of incremental cryptography for security schemes in mobile cloud computing environments." In Wireless Technology and Applications (ISWTA), 2013 IEEE Symposium on, pp. 62-67. IEEE, 2013.

Mazhar Ali is an Assistant Professor of Computer Science at COMSATS University Islamabad, Abbottabad Campus, Pakistan.

Assad Abbas is an Assistant Professor of Computer Science at COMSATS University Islamabad, Islamabad Campus, Pakistan.

Muhammad Usman Shahid Khan is an Assistant Professor of Computer Science at COMSATS University Islamabad, Abbottabad Campus, Pakistan.

Samee U. Khan is a faculty at North Dakota State University, Fargo ND, USA.