TELKOMNIKA, Vol. 11, No. 4, April 2013, pp. 2200 ~ 2208
ISSN: 2302-4046

Securing Personal Health Records in Clouds by Enforcing Sticky Policies

Chunxia Leng[1,2], Huiqun Yu[1,2*], Jingming Wang[3], Jianhua Huang[1]
[1]Department of Computer Science and Engineering, East China University of Science and Technology, Shanghai 200237, China, Telp/Fax: +86-021-64253546/64252984
[2]Shanghai Key Laboratory of Computer Software Evaluating and Testing, Shanghai 201112, China, Telp/Fax: +86-021-64251936/64253682
[3]School of Computer and Information Engineering, Chuzhou University, Anhui 239012, China, Telp/Fax: +86-0550-3510481/3510045
*Corresponding author, e-mail: yhq@ecust.edu.cn

Abstract

The personal health records (PHR) always contain much health-related privacy information in different categories. When storing the PHR data in the cloud, the PHR owner loses control to the sensitive information and is confronted with potential privacy exposure. In this paper, we propose a scheme to enable the protection of the PHR data hosted in the cloud. It not only supports that the data access can be fine-grained and base on the privacy policies specified by the PHR owner, but also affords an effective encryption mechanism and flexible key management approach to enforce the privacy policies sticky to the PHR data.

Keywords: *PHR, cloud, sticky policies, conditional proxy re-encryption, policy enforcement*

1. Introduction

Online personal health records (PHR) service allows an individual to create, store, manage, and share his personal health data in a centralized way, which is helpful for the PHR owners to obtain health care services and monitor their health status. While cloud computing becomes a promising computing paradigm in which everyone can enjoy the elastic storage and the infinite computing resources, it is attractive for the PHR service providers to shift the PHR data and applications into the cloud in order to lower their operational cost.

While the PHR cloud service provides many benefits for everyone, it brings data privacy and security risks as well. The main concern is that the PHR data always contain much health-related privacy information in different sensitive categories. For example, the first category is the individual information including age, stature, weight, family, food statistics, contact information, etc. Another category is the medical history including clinical records, family illness history, laboratory test results, allergies, chronic diseases, imaging reports, immunization records, drug reactions, etc. The third category is the health insurance information including insurance company, amount, deadline, insurance agent, etc. When storing his PHR data in the cloud, the PHR owner loses control to them and is confronted with potential privacy exposure. Consequently it is prerequisite to provide security mechanisms based upon the PHR owner's privacy requirements in the cloud.

One commonly adopted solution to protect privacy information is encryption. Basically, the PHR owner encrypts his PHR data by himself before uploading them to the cloud and stores the ciphertexts in the cloud. Only those trusted by the PHR owner can be authorized to acquire the decryption keys and then decrypt the ciphertexts. Contrarily, for the unauthorized parties who do not have the corresponding decryption keys, the PHR data remains confidential. In this case, encryption alone is not sufficient and it is required to enforce fine-grained access control on these privacy data. Such control is based on agreed policies, which should be specified by the PHR owner to meet his expectation of privacy data protection requirements, such as the category of the PHR data, the role a user playing, the privileges to read or write, the purpose of using the privacy data, the time and location conditions, obligations and so forth. Figure 1 shows the privacy-aware access control policies [1] for different categories of the PHR data. For

Received January 14, 2013; Revised February 28, 2013; Accepted March 8, 2013

example, the PHR owner may allow that the doctors can write his medical history during working period for medical treatment purpose, whereas allow that his insurance broker only can read his insurance information for insurance claim purpose.

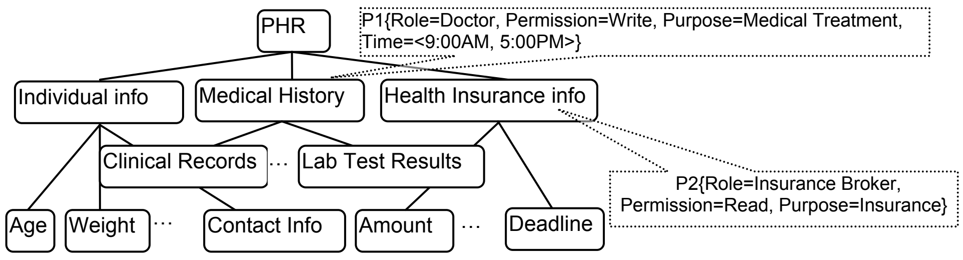
PHR
P1{Role=Doctor, Permission=Write, Purpose=Medical Treatment, Time=<9:00AM, 5:00PM>}
Individual info
Medical History
Health Insurance info
Clinical Records
Lab Test Results
P2{Role=Insurance Broker, Permission=Read, Purpose=Insurance}
Age
Weight
Contact Info
Amount
Deadline

Figure 1. Privacy-aware Access Control Policies for PHR

To make sure that the privacy-aware access control policies he specified always be enforced, the PHR owner can 'stick' these policies to his PHR data and then upload the data associated with the sticky policies to the cloud. Unfortunately, the cloud server semi-trusted by the PHR owner cannot be commissioned to strictly enforce the policies. Therefore, the trusted third parties (namely, Trusted Authorities) should be selected by the PHR owner to provide compliance checking whether the users who want to access the PHR data satisfy the requirements specified in the sticky policies. As mentioned above, the PHR data can be divided into different categories based on different sensitive level and can be specified different privacy policies for every category. Therefore, selecting different trusted third parties to enforce relevant policies can ensure that other categories of the PHR data cannot be illegitimately disclosed even if a certain category are exposed.

On the other hand, realizing fine-grained access control under encryption essentially can be transformed into a key management issue. It involves not only grant but also revoke access privileges. To avoid from high key management complexity, an effective encryption mechanism should be adopted. Thanks to C-PRE [2] mechanism, with it the read privilege and write privilege can be distinguished easily. Moreover, granting access privileges to a user or revoking access privileges from him can be subtly enforced.

In this paper, we propose a novel and practical scheme to enable the protection of the PHR data hosted in the cloud. It not only supports that the data access can be fine-grained and based on privacy policies specified by the PHR owner, but also affords an effective encryption mechanism and flexible key management scheme.

The rest of the paper is organized as follows: In section 2 we briefly review selected work related to ours. In section 3 we describe system model and system goals. In Section 4 we present our proposed scheme in detail. Section 5 analyzes system security properties. We conclude the paper in section 6.

2. Related Work

The sticky policies paradigm was originally proposed by Karjoth et al in [3]. When submitting personal data to an enterprise, the privacy preferences can be specified by the user and attach with personal data to make sure that the privacy preferences can always be enforced. This paradigm provides very useful inspiration on how to protect sensitive personal data. However, the platform suggested in [3] is rather weak in the sense that the enforcement can not prevent the modification of personal data and the privacy policies because there have no encryption mechanism adopted.

Pearson et al [4] describe a solution using sticky policy and IBE to manage privacy personal data. In their solution, a sticky policy is mapped to an IBE encryption key which describes the subject and the access condition. However, the key management is too complex. In their latest work [5], they use PKI infrastructure to enforce sticky policies and deploy the framework in the cloud. The solution also uses the trusted third parties to control the enforcement of the sticky policies. However, there has no consideration of the data categories and the distinction of read and write privilege.

Conditional proxy re-encryption (C-PRE) is a cryptographic primitive derived from proxy re-encryption (PRE) [6]. In a PRE scheme, a semi-trusted proxy is given a re-encryption key by delegator and thus can convert a ciphertext encrypted under delegator's public key into another ciphertext that can be decrypted by delegatee's private key without learning anything about the plaintext. In a C-PRE Scheme, compared with traditional PRE, the delegator can categorize his plaintexts into different portions and distribute the decryption right of each portion to different delegatee through a proxy under the same key pair with respect to a certain condition.

There has been an increasing interest in using encryption mechanisms to secure Personal Health Records recently. Li et al proposed a framework using MA-ABE and KP-ABE mechanism to achieve fine-grained access control for securing PHR in cloud computing [7], where selecting multiple attribute authorities to manage users and attributes. In [8], an attribute-based infrastructure for EHR systems was proposed, where EHR files are encrypted using a broadcast variant of CP-ABE that allows direct revocation. In [9], a scheme applying CP-ABE to manage the sharing of PHRs was described. However, in CP-ABE and KP-ABE mechanisms, it describes access policies with access structure, which is not fit to express the complex privacy-aware access policies.

3. System Model and Goals

3.1. System Model

We assume a PHR system in the cloud consists of the following entities, as shown in Figure 2.

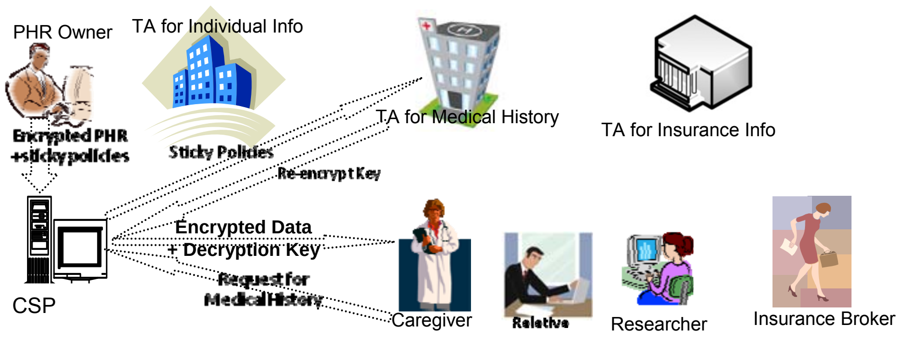

Figure 2. The PHR System Model in the Cloud

Multiple PHR Owners: The PHR owners refer to the individual who wants to upload his PHR data to the cloud after dividing them into different categories, encrypting them and sticking the privacy-aware access control policies to them. Absolutely, the PHR owners can reselect a new key to encrypt his PHR data or delete his PHR data when necessarily. He also can update the privacy policies.

Cloud Server Provider (CSP): The PHR data can be organized by their categories and stored in a central server belonging to the CSP. The CSP is semi-trusted by the PHR owners who presume that CSP can storage the encrypted PHR data and faithfully follow the protocol in general but may be interested in the privacy data and try to find out as much secret as possible.

Multiple Trusted Authorities (TAs): Similar to [7], we consider that there exists multiple TAs in the PHR system. TAs are the independent entities fully trusted by the PHR owner and provide compliance checking capabilities to enforce the sticky policies of the PHR data and authorize the users to acquire the decryption key to read or write. Unlike [1], where TAs can acquire the decryption key, we prohibit TAs from knowing about the decryption key of the PHR data.

Multiple Users: The users may come from various domains such as the relatives, the researchers, the caregivers, the insurance brokers etc. The users can be authorized to read or write the PHR data based on the sticky policies.

3.2. System Goals

The design goals of our system are:

It is easier to meet the PHR owners' requirements on the privacy data protection by allowing the PHR owners specifying their own private-aware access control policies of the PHR data and strictly enforcing the policies.

There is no need for a central policy repository because the policies are always sticky to the data.

It prevents the CSP from knowing both the PHR data and the privacy policies interrelated user's access privilege. It keeps the decryption key confidential to TAs.

It supports to authorize different users to gain read-access or write-access to the PHR data through enforcing fine-grained access control.

It should be efficient whenever the user's access privileges are revoked or the keys are updated.

4. Securing PHR by Using C-PRE to Enforce Sticky Policies

In this section, we firstly illustrate C-PRE encryption mechanism used in our scheme, and then elaborate on how to achieve fine-grained access control by using C-PRE to enforce sticky policies. Other several key design issues are also addressed in this section.

4.1. Preliminary-Conditional Proxy Re-Encryption

In our proposed system, C-PRE is used to achieve flexible access privilege granting and revoking. C-PRE scheme is mainly composed of seven algorithms defined as follows:

Setup(1^{κ}) The setup algorithm takes as input a security parameter 1^{κ} and determines (q, G, G_T, e), where q is a κ-bit prime, G and G_T are two cyclic groups with prime order q, and e is the bilinear pairing, $e: G\times G\rightarrow G_T$. Then it chooses $g\in_R G$, and five hash functions H_1, H_2, H_3, H_4 and H_5. $H_1:\{0,1\}^*\rightarrow Z_q$, $H_2:\{0,1\}^*\rightarrow G$, $H_3:G\rightarrow\{0,1\}^n$, $H_4:\{0,1\}^*\rightarrow G, H_5:G\rightarrow Z_q$, where n is polynomial in κ and the message space is $M = \{0,1\}^n$. The global parameter is:

$$param= ((q, G, G_T, e), g, n, H_1, H_2, H_3, H_4, H_5) \quad (1)$$

KeyGen(1^{κ}) The key generation algorithm takes a security parameter 1^{κ} as input and then picks $x_i\in_R Z_q$. It generations the public/private key pairs for user U_i as follows.

$$(pk_i,sk_i)=(g^{x_i},x_i) \quad (2)$$

ReKeyGen(sk_i,w,pk_j) The re-encryption key generation algorithm takes a private key sk_i, a condition w, a public key pk_j as input and randomly picks $s\in_R Z_q$, then outputs the re-encryption key as follows.

$$rk_{i\xrightarrow{w}j}=(rk_1,rk_2)=((H_2(pk_i,w)pk_j^{s\cdot H_5(pk_j^{s\cdot sk_i})})^{-sk_i}, pk_i^s) \quad (3)$$

Where condition w can represent keyword, attribute and even any meaningful bit-string.

Enc$_2$(pk,w,m) On input a public key pk, a condition w and a message $m\in M$, the second encryption algorithm picks $R\in_R G_T$. Then it computes $r = H_1(m, R)$, and outputs the ciphertext CT as follows. CT can be re-encrypted using the suitable re-encryption key.

$$CT=(C_1,C_2,C_3,C_4)=(g^r,R\cdot e(pk,H_2(pk,w))^r,m\oplus H_3(R),H_4(C_1,C_2,C_3)^r) \quad (4)$$

Enc$_1$(pk,m) On input a public key pk and a message $m\in M$, the first encryption algorithm picks $R\in_R G_T$ and $\bar{s}\in_R Z_q^*$. Then it computes $r = H_1(m, R)$, and outputs the ciphertext CT as follows, which cannot be re-encrypted.

$$CT=(\overline{C}_1,\overline{C}_2,\overline{C}_3,\overline{C}_4)=(g^r{}_1,R\cdot e(g,pk)^{-r\cdot\bar{s}\cdot H_5(pk^{\bar{s}})},m\oplus H_3(R),g^{\bar{s}}) \quad (5)$$

ReEnc(CT**$_i$**, $rk_{i \xrightarrow{w} j}$***)*** This re-encryption algorithm takes the ciphertext $CT_i=(C_1,C_2,C_3,C_4)$ and the re-encryption key $rk_{i \xrightarrow{w} j}=(rk_1,rk_2)$ as input. Firstly, it checks whether the following equality holds:

$$e(C_1,H_4(C_1,C_2,C_3))=e(g,C_4) \tag{6}$$

If not, outputs $\perp$; else outputs the ciphertext CT_j as follows.

$$CT_j=(\overline{C}_1,\overline{C}_2,\overline{C}_3,\overline{C}_4)=(C_1,C_2\cdot e(C_1,rk_1),C_3,rk_2) \tag{7}$$

Dec(sk,CT) On input the private key *sk* and the ciphertext $CT=(\overline{C}_1,\overline{C}_2,\overline{C}_3,\overline{C}_4)$, the decryption algorithm firstly computes:

$$R=\overline{C}_2\cdot e(\overline{C}_1,\overline{C}_4)^{sk\cdot H_5(\overline{C}_4^{sk})} \tag{8}$$

$$m=\overline{C}_3\oplus H_3(R) \tag{9}$$

Then it checks whether the following equality holds:

$$g^{H_1(m,R)}=\overline{C}_1 \tag{10}$$

If not, outputs $\perp$; else outputs the plaintext *m*.

In C-PRE, the following two equations always hold:

$$Dec(Enc_1(pk_i,m),sk_i)=m \tag{11}$$

$$Dec(ReEnc(Enc_2(pk_i,m,w),ReKeyGen(sk_i,w,pk_j)),sk_j)=m \tag{12}$$

4.2. Privacy Policies Sticky to the PHR Data

In our scheme, it allows the PHR owner to specify the privacy-aware access control policies which embody the PHR owner's privacy requirements of access his PHR data. The policy may be represented in any convenient format, such as an XML format, and may contain some elements as the following:

1. Types of the PHR data, such as the individual information, the medical history, etc.
2. Purposes of using the data, such as for research, for medical treatment, etc.
3. Operations including read and write.
4. Roles allowed accessing the data.
5. Conditions that specify use of the data only within a given set of platforms with certain period of time, a given network, or a subset of the enterprise.
6. Obligations that required the user to perform certain operations, such as deletion of data after a certain time, notification the PHR owner, and so on.

Then the PHR owner should adopt the algorithm particularized in section 4.4 to strongly binding these privacy policies to the PHR data they are associated with. By specifying and enforcing sticky policies, it is easier for the PHR owner to control accesses to his sensitive PHR data in cloud. In addition, sticky policies provide another means of data protect besides encryption, since the PHR data which the privacy policies is sticky to cannot be accessed unless these policies are complied with.

4.3. Read Privilege and Write Privilege

As proposed in [10], only encrypting the PHR data with a symmetric encryption key K_{data} can not distinguish the read privilege and write privilege. So the PHR owner should not only choose a symmetric encryption key K_{data} but also a pair of public/private key K_{verify}/K_{sign}. He uses K_{data} to encrypt his PHR data and then sign the ciphertext with the private key K_{sign}.

The symmetric encryption key K_{data} associated with the public key K_{verify} are assigned to the reader. The reader can firstly verify the signature with K_{verify} and then decrypt the cyphertext with K_{data}.

The writer should acquire the private key K_{sign} besides K_{data} and K_{verify}. After modifying the PHR data, the writer should encrypt them again with K_{data} and then sign with K_{sign}.

4.4. Creating Data Uploaded to CSP

In our scheme, we presume that every TA and every user has the public/private key pairs generated by the key generation algorithm in C-PRE. TA$_i$'s public key pk_{TAi} has been broadcasted to the PHD owners but the private key sk_{TAi} is kept safely by TA$_i$. A user's public key pk_{user} has been broadcasted to all TAs but the private key sk_{user} is kept safely too.

Before upload his PHR data to the cloud, the PHR owner firstly divides the data into different categories and encrypt them. Then he specifies privacy-aware access control policies for every category and selects different Trusted Authorities (TAs) to enforce relevant policies. After strongly 'sticking' the policies to the data, the PHR owner finally uploads them to CSP and then stays offline.

The following algorithm elaborates the process carried out by the PHR owner:

Step1. Randomly select a symmetric AES encryption key K_{data} and a pair of RSA public/private key K_{verify} /$K_{sign.}$

Step2. Classify the PHR data into different categories, i.e. $\{<t_1, D_1>, <t_2, D_2>, ..., <t_n, D_n>\}$, where t_i is the string represents the categories, such as "IndividualInfo", "MedicalHistory", "InsuranceInfo", etc. D_i is the subset of the PHR data in t_i category.

Step3. For every D_i in t_i category:

(a). Encrypt D_i with K_{data} by using AES and then sign with K_{sign} by using RSA to get the ciphertext C_i.

$$C_i=E_{RSA}(K_{sign}, E_{AES}(K_{data}, D_i)) \tag{13}$$

(b). Select a TA_i to authorize the users who want to access D_i。

(c). By using the second encryption algorithm in C-PRE scheme, encrypt K_{data} and K_{verify} with selected TA$_i$'s public key pk_{TAi} to denote the read privilege R_i.

$$R_i=Enc_2(pk_{TAi}, 'r'||t_i, K_{data}||K_{verify}) \tag{14}$$

Where, $'r'||t_i$ means connecting the string $'r'$ with t_i, such as "rMedicalHistory", and it is the input parameter of C-PRE to represent condition w.

(d). By using the second encryption algorithm in C-PRE scheme, encrypt K_{data}, K_{verify} and K_{sign} with with selected TA's public key pk_{TAi} to denote the write privilege W_i.

$$W_i=Enc_2(pk_{TAi}, 'w'||t_i, K_{data} || K_{verify} || K_{sign}) \tag{15}$$

Where, $'w'||t_i$ means connecting the string $'w'$ and t_i, such as "wIndividualInfo", and it is the input parameter of C-PRE to represent condition w.

(e). Specify the privacy-aware access control policy $Policy_i$ for D_i.

(f). Encrypt $Policy_i$ and t_i with TA$_i$'s public key pk_{TAi} by using the first encryption algorithm in C-PRE scheme and get P_i.

$$P_i= Enc_1(pk_{TAi}, Policy_i || t_i)) \tag{16}$$

(g). Upload the following data to CSP.

$$\{ t_i || TA_i || C_i || R_i || W_i || P_i\} \tag{17}$$

4.5. Enforcement of Access Control

When a user requests for read or write access privileges of t_i category PHR data in the cloud server, CSP and TA perform the protocol as the following steps shown.

Step1. CSP searches for the related data according to t_i and get *{ t_i||TA_i ||C_i ||R_i ||W_i ||P_i}*. Then CSP sends the user's request and P_i to TA_i.

Step2. TA_i decrypts P_i with it's own private key sk_{TAi} by using the decryption algorithm in C-PRE scheme. According to equation (11), TA_i can obtain the privacy policy $Policy_i$ and the category t_i.

$$Policy_i \,||\, t_i = Dec(sk_{TAi},\ P_i)) \tag{18}$$

Step3. TA_i carries out policy checking whether the user satisfy the access requirements defined in the privacy policy $Policy_i$ and then decides whether the user can be authorized to read or write t_i category PHR data.

Step4. TA_i generates the re-encryption key and send it to CSP. If the use is not allowed to access t_i category PHR data, TA_i will send nothing to CSP. If the user can be granted the read privilege, TA_i will execute the re-encryption key generation algorithm in C-PRE scheme with the input parameter: TA_i's private key sk_{TAi}, condition *'r'*||t_i and the user's public key pk_{user} to generate the re-encryption key as shown in formula (19). If the user can be granted the write privilege, TA_i will generate the re-encryption key with the input parameter sk_{TAi}, *'w'*||t_i and pk_{user}, as formula (20) shown.

$$rk_{TA_i \xrightarrow{'r'||t_i} user} = \text{ReKeyGen}\ (sk_{TA_i}, 'r'||t_i, pk_{user}) \tag{19}$$

$$rk_{TA_i \xrightarrow{'w'||t_i} user} = \text{ReKeyGen}\ (sk_{TA_i}, 'w'||\, t_i, pk_{user}) \tag{20}$$

Step5. By using the received re-encryption key from TA_i as input parameter, CSP executes the re-encryption algorithm in C-PRE scheme to re-encrypt R_i and W_i and obtains R'_i and W'_i. If the received re-encryption key is $rk_{TA_i \xrightarrow{'r'||t_i} user}$, the re-encryption result will be:

$$(R_i^{'}, W_i^{'}) = (\text{ReEnc}(rk_{TA_i \xrightarrow{'r'||t_i} user}, R_i), \text{ReEnc}(rk_{TA_i \xrightarrow{'r'||t_i} user}, W_i)) \tag{21}$$

Undoubtedly, R'_i is valid but W'_i is invalid in formula (21). It means that the user can only decrypt R'_i with his privacy key sk_{user} correctly but cannot decrypt W'_i with sk_{user}. In this way, the user is granted read privilege.

If the received re-encryption key is $rk_{TA_i \xrightarrow{'w'||t_i} user}$, the re-encryption result will be:

$$(R_i^{'}, W_i^{'}) = (\text{ReEnc}(rk_{TA_i \xrightarrow{'w'||t_i} user}, R_i), \text{ReEnc}(rk_{TA_i \xrightarrow{'w'||t_i} user}, W_i)) \tag{22}$$

Similarly, W'_i is valid but R'_i is invalid in formula (22). That means, the user is granted write privilege.

Step6. CSP sends *{C_i || R'_i || W'_i}* to the user.

Step7. The user firstly executes the decryption algorithm in C-PRE scheme to decrypt R'_i and W'_i with his private key sk_{user}. As described above, if the user is granted read privilege, he only can decrypt R'_i correctly and gain K_{data}||K_{verify}, as shown in formula (23).

$$K_{data}||K_{verify} = Dec(sk_{user},\ R_i^{'})) \tag{23}$$

If the user is granted write privilege, he can decrypt W'_i correctly and gain K_{data}||K_{verify}||K_{sign}, as shown in formula (24).

$$K_{data}||K_{verify}||K_{sign} = Dec(sk_{user},\ W_i^{'})) \tag{24}$$

For the reader, he firstly uses K_{verify} to verify the sign of C_i, and then decrypts with K_{data} to get D_i. For the writer, after modifying the PHR data, he can encrypt them again with K_{data} and then sign with K_{sign}.

4.6. User Revocation

When the PHR owner’s privacy requirements have changed to revoke the privileges on a certain category data from some users, such as revocation the write privilege on medical history from the nurse role, the PHR owner needs to update the privacy policy $Policy_i$. Furthermore, in order to avoid that the revoked users collude with CSP to violate the privacy policies and can read or write the data with the former keys K_{data} and K_{verify}/K_{sign}, the PHR owner can select the new keys K'_{data} and K'_{verify}/K'_{sign} to encrypt data.

When carrying out policy checking, TA will decide whether the user can be authorized to read or write the PHR data according to the updated policy.

4.7. Key Updating

There are four key updating problems in our scheme to discuss:

1. Updating the user’s key: If the user’s private key sk_{user} is expired or compromised, he only needs to broadcast his new public key pk'_{user} to TAs. In this case, the PHR owner should do nothing, and TA simply generates a new re-encryption key with pk'_{user} and send it to CSP.

2. Updating re-encryption key: If a re-encryption key is compromised, such as $rk_{TA_i \xrightarrow{'w'\|t_i} user}$, other categories except t_i category data will not be affected because of the condition $'w'\|t_i$. To revoke the compromised re-encryption key, the PHR owner needs to create a new category t_i' and redefines R_i, W_i and P_i. TA generates the new re-encryption key $rk_{TA_i \xrightarrow{'w'\|t_i'} user}$.

3. Updating TA’s key: If a TA’s private key sk_{TA} is compromised, TA needs to broadcast the new public key pk'_{TA} to the PHR owner. Then the PHR owner recomputes R_i, W_i and P_i.

4. Updating K_{data}, K_{verify} and K_{sign}: If the symmetric encryption key K_{data} and the public/private key K_{verify} /K_{sign} are compromised, the PHR owner have to select the new keys K'_{data} and K'_{verify}/K'_{sign}. Then he recomputes C_i, R_i, W_i and P_i.

5. System Analysis

In this section, we will analyze security properties of our proposed scheme.

5.1. Fine-grainedness of Access Control

It adopts the sticky policies paradigm to achieve fine-grained access control in our proposed scheme. The PHR owner is able to specify the privacy-aware access control policies which embody his privacy requirements of access his PHR data. Moreover, the PHR owner can recommend the trusted third parties to enforce fine-grained access control according to the sticky policies. By using C-PRE mechanism, it is easy to distinguish read privilege and write privilege.

5.2. Privacy Policies Confidentiality

Before uploading the privacy policy $Policy_i$ to CSP, the PHR owner encrypts it with selected TA’s public key pk_{TAi}. Therefore, it is impossible for CSP to know about the privacy-aware access control policies and thus derive the user’s access privileges.

5.3. Data Confidentiality

In our scheme, the PHR data uploaded to CSP are encrypted with a symmetric AES encryption key K_{data} and then K_{data} is encrypted with TA’s public key pk_{TAi}. So the PHR data is secret for CSP. When the user requires accessing the PHR data, CSP only transfers the privacy policies to TA. By this means, TA can not know anything about the PHR data or K_{data}. After the re-encryption key is generated by TA and the re-encryption algorithm is processed by CSP, K_{data} is encrypted with the user’s public key pk_{user}. Only possessing the private key sk_{user} can decrypt K_{data} and thus decrypt the PHR data.

6. Conclusions

The scheme proposed in this paper achieves securing the personal health records in the cloud. Before uploading their PHR data to the cloud, the PHR owner can define the privacy-aware access control policies and strongly bind them with associated data by using C-PRE encryption mechanisms. TAs provide compliance checking capabilities to enforce the sticky policies on the PHR data and then authorize the users to acquire the decryption key to read or write. In this way, it not only supports fine-grained access control based on the PHR owner's privacy preferences, but also prevents the CSP from learning both the PHR data and the privacy policies. Through implementation and simulation, it shows that the scheme is both efficient and scalable.

Acknowledgement

The work was supported by the NSF of China under grants No. 60773094, 60473055 and 61173048, Shanghai Shuguang Program under grant No. 07SG32.

References

[1] Ni Q, Trombetta A, Bertino E, Lobo J. *Privcy-Aware Role Based Access Control*. Proceedings of the 12th ACM Symposium on Access Control Models and Technologies. New York. 2007: 41–50.
[2] Weng J, Yang J, Tang Q, Deng RH, Bao F. *Efficient Conditional Proxy Re-Encryption with Chosen-Ciphertext Security*. Proceedings of the 12th International Conference on Information Security. Sydney. 2009: 151–166.
[3] Karjoth G, Schunter M, Waidner M. *Platform for Enterprise Privacy Practices: Privacy-Enables Management of Customer Data*. Proceedings of the 2nd Workshop on Privacy Enhancing Technologies. San Francisco. 2002; 2482: 69–84.
[4] Pearson S, Mont MC, Chen LQ, Reed A. *End-to-End Policy-Based Encryption and Management of Data in the Cloud*. Proceedings of the 3rd IEEE International Conference on Cloud Computing Technology and Science. Athens. 2011: 764-771.
[5] Mont MC, Pearson S, Bramhall P. *Towards Accountable Management of Identity and Privacy: Sticky Policies and Enforceable Tracing Services*. Proceedings of 14th International Workshop on Database and Expert Systems Applications. Washington. 2003: 377-382.
[6] Blaze M, Bleumer G, Strauss M. *Divertible Protocols and Atomic Proxy Cryptography*. International Conference on the Theory and Application of Cryptographic Techniques. Espoo. 1998: 127–144.
[7] Li M, Yu SC, Ren K, Lou WJ. *Securing Personal Health Records in Cloud Computing: Patient-Centric and Fine-Grained Data Access Control in Multi-Owner Settings*. Proceedings of 6th Iternational ICST Conference, SecureComm. Singapore. 2010: 89–106.
[8] Narayan S, Gagne M, Safavi-Naini R. *Privacy Preserving EHR System using Attribute-Based Infrastructure*. Proceedings of the 2nd ACM Cloud Computing Security Workshop. Chicago. 2010: 47–52.
[9] Ibraimi L, Asim M, Petkovic M. *Secure Management of Personal Health Records by Applying Attribute-Based Encryption*. University of Twente. Technical Report. 2009.
[10] Kallahalla M, Riedel E, Swaminathan R, Wang Q, Fu K. *Plutus: Scalable Secure File Sharing on Untrusted Storage*. Proceedings of the 2nd USENIX Conf on File and Storage Technologies. Berkeley. 2003: 29-42.