Multi-Authority Secure Personal Health Record System

Phuwanai Thummavet

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of

Master of Engineering in Computer Engineering

Prince of Songkla University

2015

Thesis Title	Multi-Authority Secure Personal Health Record System
Author	Mr.Phuwanai Thummavet
Major Program	Computer Engineering

Major Advisor :

.....

(Asst.Prof.Dr.Sangsuree Vasupongayya)

Examining Committee :

.....Chairperson

(Assoc.Prof.Dr.Sinchai Kamolphiwong)

.....

(Asst.Prof.Dr.Sangsuree Vasupongayya)

.....

(Prof.Dr.Verapol Chandeying)

The Graduate School, Prince of Songkla University, has approved this thesis as partial fulfillment of the requirements for the Master of Engineering Degree in Computer Engineering.

.....

(Assoc.Prof.Dr.Teerapol Srichana)

Dean of Graduate School

This is to certify that the work here submitted is the result of the candidate's own investigations. Due acknowledgement has been made of any assistance received.

..... Signature
(Asst.Prof.Dr.Sangsuree Vasupongayya)
Major Advisor

..... Signature
(Mr.Phuwanai Thummavet)
Candidate

I hereby certify that this work has not been accepted in substance for any degree, and is not being currently submitted in candidature for any degree.

..... Signature

(Mr.Phuwanai Thummavet)

Candidate

ชื่อวิทยานิพนธ์	ระบบจัดการข้อมูลสุขภาพส่วนบุคคลอย่างปลอดภัยแบบมัลติออโทริตี้
ผู้เขียน	นายภูวนัย ธรรมเวช
สาขาวิชา	วิศวกรรมคอมพิวเตอร์
ปีการศึกษา	2557

บทคัดย่อ

ข้อมูลสุขภาพส่วนบุคคล (Personal Health Record, (PHR)) เป็นแนวคิดของการจัดการและการแบ่งปันข้อมูลซึ่งเกี่ยวข้องกับสุขภาพโดยปัจเจกบุคคล ต่างจากข้อมูลการรักษาอิเล็กทรอนิกส์ (Electronic Medical Record, (EMR)) ซึ่งเป็นสิ่งที่ถูกจัดการโดยโรงพยาบาล ข้อมูลสุขภาพส่วนบุคคลอยู่ภายใต้การควบคุมโดยปัจเจกบุคคล ตามหลักการ ข้อมูลสุขภาพส่วนบุคคลสามารถบรรจุข้อมูลเช่น โรคประจำตัว ประวัติการรักษาโรค การแพ้ยาและอาหาร ข้อมูลด้านสุขภาพจิตและจิตเวช การวินิจฉัยโรค และการให้คำปรึกษาของแพทย์ เป็นต้น ยิ่งไปกว่านั้น ข้อมูลสุขภาพส่วนบุคคลยังรองรับแนวคิดของการเก็บรวบรวมข้อมูลการเฝ้าสังเกตอาการของผู้ป่วยที่บ้าน (Homed-monitored data) ซึ่งข้อมูลสัญญาณถูกรวบรวมโดยอุปกรณ์เซ็นเซอร์และถูกบันทึกลงไปยังฐานข้อมูลของระบบข้อมูลสุขภาพส่วนบุคคลผ่านทางอินเทอร์เน็ต ข้อมูลที่บรรจุในฐานข้อมูลของระบบข้อมูลสุขภาพส่วนบุคคลนี้ สามารถช่วยให้แพทย์นำไปวิเคราะห์หาสาเหตุและอาการของโรคของผู้ป่วยได้ดีขึ้น แนวคิดของข้อมูลสุขภาพส่วนบุคคลถูกนำเสนอขึ้นมาไม่ใช่แค่ลดค่าใช้จ่ายทางด้านการดูแลและรักษาสุขภาพซึ่งมีราคาสูงในทุกวันนี้เท่านั้น แต่ยังช่วยเพิ่มโอกาสในการรักษาโรคให้หายขาดด้วย

เนื่องจากความอ่อนไหวของข้อมูลสุขภาพส่วนบุคคล ประเด็นด้านความปลอดภัยและความเป็นส่วนตัวต่อข้อมูลสุขภาพส่วนบุคคลจึงกลายเป็นความกังวลขั้นพื้นฐานของเจ้าของข้อมูลสุขภาพส่วนบุคคลจำนวนมาก เพื่อปกป้องประเด็นดังกล่าว จึงมีการนำเสนอการควบคุมการเข้าถึงข้อมูล (Access control) หลากหลายรูปแบบ หนึ่งในรูปแบบที่ได้รับการนำไปใช้อย่างแพร่หลาย คือ Ciphertext-Policy Attribute-Based Encryption (CP-ABE) ภายใต้ CP-ABE เจ้าของข้อมูลสุขภาพส่วนบุคคลสามารถกำหนดนโยบายในการเข้าถึง (Access policy) ข้อมูลโดยกำหนดเซตของแอตทริบิวต์ (Attributes) ของผู้ใช้ที่ได้รับอนุญาติให้เข้าถึงข้อมูลได้ ข้อมูลสุขภาพส่วนบุคคลดังกล่าวจะถูกเข้ารหัสข้อมูลด้วย CP-ABE โดยกำหนดเซตของแอตทริบิวต์เพื่อใช้เป็นพารามิเตอร์สำหรับ

เข้ารหัสข้อมูลดังกล่าว จากนั้น ข้อมูลสุขภาพส่วนบุคคลที่ถูกเข้ารหัสดังกล่าวจะสามารถถูกถอดรหัสได้เฉพาะผู้ใช้ที่มีกุญแจส่วนบุคคล (CP-ABE private key) ซึ่งมีเซตของแอตทริบิวต์ที่สอดคล้องกับนโยบายในการเข้าถึงที่ถูกกำหนดโดยเจ้าของข้อมูลสุขภาพส่วนบุคคลในระหว่างเข้ารหัสข้อมูลเท่านั้น อีกนัยนึง CP-ABE เป็นรูปแบบการกำหนดการเข้ารหัสข้อมูลโดยใช้พื้นฐานการตรวจสอบแอตทริบิวต์ของผู้ใช้

ในสภาพแวดล้อมของระบบข้อมูลสุขภาพส่วนบุคคล ผู้ใช้อาจมีบทบาทหรือแอตทริบิวต์ที่แตกต่างกัน เช่น ผู้ป่วย สมาชิกในครอบครัวของผู้ป่วย ผู้ดูแล แพทย์ บุคลากรทางการแพทย์อื่นๆ หน่วยฉุกเฉินต่างๆ และผู้ให้ประกันสุขภาพ เป็นต้น อย่างไรก็ตาม CP-ABE ถูกออกแบบมาสำหรับสภาพแวดล้อมในการจัดการผู้ใช้แบบผู้มีอำนาจเดี่ยว (Single user authority) อีกนัยนึง ผู้ใช้และแอตทริบิวต์ทั้งหมดจะต้องถูกจัดการโดยผู้มีอำนาจแบบศูนย์กลาง (Centralized user authority) ดังนั้น ประเด็นและข้อจำกัดจำนวนมากอาจจะเกิดขึ้นเมื่อนำ CP-ABE มาประยุกต์ใช้กับระบบข้อมูลสุขภาพส่วนบุคคลในทางปฏิบัติ เช่น ปัญหาข้อจำกัดทางด้านความสามารถในการขยายตัวของระบบ (System scalability) ปัญหาคอขวด (Single point of failure) และปัญหาการจัดการผู้ใช้อย่างมีประสิทธิภาพ (Efficient user management) เป็นต้น

เพื่อที่จะจัดการกับประเด็นและข้อจำกัดดังกล่าวข้างต้น งานวิจัยชิ้นนี้นำ CP-ABE มาประยุกต์ใช้เพื่อให้สามารถรองรับสภาพแวดล้อมแบบมัลติออโทริตี้ (Multi-authority) ส่งผลให้ระบบข้อมูลสุขภาพส่วนบุคคลที่นำเสนอสามารถกระจายภาระการจัดการผู้ใช้และแอตทริบิวต์ไปยังผู้มีอำนาจการจัดการผู้ใช้ (User authority) ต่างๆ ได้ เช่น โรงพยาบาลสามารถก่อตั้งผู้มีอำนาจการจัดการผู้ใช้ของตนเอง (Healthcare authority) ขึ้นมาสำหรับรองรับบุคลากรทางการแพทย์ของตนเองได้ ขณะที่ผู้ป่วยก็สามารถก่อตั้งผู้มีอำนาจการจัดการผู้ใช้เป็นของตนเอง (Personal authority) เพื่อที่จะรองรับสมาชิกในครอบครัว ผู้ดูแล และเพื่อนๆได้ แบบแผนของระบบข้อมูลสุขภาพส่วนบุคคลแบบมัลติออโทริตี้ที่นำเสนอในงานวิจัยนี้ยังได้นำเสนอกลไกความปลอดภัยเพื่อที่จะรักษาความเป็นส่วนตัวของเจ้าของข้อมูลสุขภาพส่วนบุคคล กลไกดังกล่าวประกอบด้วย แบบแผนการปกป้องข้อมูลสองชั้น (PHR dual layer protection scheme) แบบจำลองความเชื่อใจแบบลำดับขั้น (Hierarchical trust model) และแบบแผนการแบ่งปันข้อมูลสุขภาพส่วนบุคคลอย่างปลอดภัย (End-to-end secure PHR sharing scheme) ด้วยกลไกความปลอดภัยที่นำเสนอ เจ้าของข้อมูลสุขภาพส่วนบุคคลสามารถที่จะควบคุมการเข้าถึงข้อมูลสุขภาพส่วนบุคคลของตนได้อย่างละเอียด (Fine-grained access control) นอกจากนี้ กลไกที่นำเสนอยังรับประกันว่าข้อมูลสุขภาพส่วนบุคคลจะสามารถถูกเปิดอ่านหรือถูกแก้ไขได้เฉพาะผู้ใช้ซึ่งได้รับอนุญาติจากเจ้าของข้อมูลสุขภาพส่วนบุคคลเท่านั้น

Thesis Title	Multi-Authority Secure Personal Health Record System
Author	Mr.Phuwanai Thummavet
Major Program	Computer Engineering
Academic Year	2014

ABSTRACT

Personal health record (PHR) is a concept of managing and sharing personal health related information by an individual. Unlike an electronic medical record (EMR) which is managed by a hospital, PHR is under a full control by the PHR owner. Ideally, PHR can contain any type of individual health information such as personal diseases, medical history, allergies, mental health information, diagnosis and physicians' recommendations. Beyond the EMR, PHR also supports an idea of collecting home-monitored data which are captured from body sensor devices. For example, a patient can capture his/her vital signs using body sensor devices and then record the captured data into a database of any internet-based PHR system. With the captured vital signs, a physician can make a better diagnosis of diseases and symptoms. PHR is intentionally proposed in order to not only reduce some expensive today healthcare costs but also increase an opportunity for curing diseases.

Due to the highly sensitivity of personal health information, security and privacy issues of PHR become a primary concern of many PHR owners. To prevent such issues, several access control schemes were proposed. One of the most widely adopted access control schemes for the PHR is Ciphertext-Policy Attribute-Based Encryption (CP-ABE). Under CP-ABE, a PHR owner can specify an access policy over a set of attributes for encrypting each particular PHR record. The resulting encrypted PHR can only be decrypted by the users who have the CP-ABE private key containing the set of specific attributes that satisfies the associated access

policy pre-defined by the PHR owner. In other words, CP-ABE is an attribute-based encryption scheme.

Under a PHR system environment, there can be multiple users with different roles/attributes; for example, patients, family members, caregivers, physicians, other medical practitioners, emergency responders, and health insurers. Unfortunately, CP-ABE is designed for a single user authority (UA) environment. In other words, all users and attributes must be managed by a single centralized UA. Consequently, several issues and limitations may be occurred when employing CP-ABE in practice, including a system scalability problem, a single point of failure problem, and an efficient user management problem.

To handle such issues and limitations, a multi-authority secure personal health record (MA-PHR) scheme is proposed in this research. By adding an initial setting, the CP-ABE can be used in a similar fashion to handle the users and attributes from multiple authorities. With the proposed scheme, for example, a hospital can establish a healthcare authority for supporting its medical practitioners locally as well as a patient can create his/her own personal authority for supporting his/her family members, caregivers, and relatives. The proposed MA-PHR scheme also presents several security mechanisms in order to preserve the privacy of the PHR owners including the PHR dual layer protection scheme, the hierarchical trust model, the end-to-end secure PHR sharing scheme. With the proposed security mechanisms, the PHR owner has a fine-grained access control on his/her PHR records whereas the proposed mechanisms guarantee that PHR information can be read or modified only by the owner-authorized users.

Contents

List of Figures

1. Introduction

In recent years, the technology for storing and managing healthcare information has been shifted from the paper-based record to the electronic medical record (EMR) [1]. A general concept of the EMR is to transform patients' health and medical information recorded by a hospital, which is traditionally stored in terms of physical paper-based records, to digital forms that can be managed by a computer instead. EMR gains several advantages than the prior one such as reducing costs in storing, maintaining, searching and accessing health records; providing fast searching; and the health records can be accessed by multiple users anytime anywhere.

The personal health record (PHR) [2], [3] is gaining popularity nowadays while the EMR is managed and controlled by a hospital. PHR is a concept of storing the personal health information, managing and controlling by an individual. The PHR owner can gather his/her personal health information from various hospitals or clinics and then store the information into his/her PHRs. Then, the PHR owner can selectively share each of his/her PHRs to any desired people. Furthermore, the PHR also opens to an idea of collecting the home-monitored data which are captured from body sensor devices. For example, a patient can capture his/her vital signs using body sensor devices and then record the captured signs into any internet-based PHR system. With the captured vital signs, a physician can make a better diagnosis of diseases and symptoms of the patient [2]. PHR is intentionally proposed in order to not only reduce some expensive healthcare costs but also increase an opportunity for curing diseases.

Typically, the PHRs are stored and handled by the PHR cloud-based service providers (PHR providers). The PHR providers typically provision an abundant storage capacity, a high computation processing unit and a large network bandwidth with a reasonable price to PHR owners. However, the PHR owners and the PHR providers are generally considered in different trust domains [2], [4], [5], and [6]. Therefore, security and privacy issues on the PHR information become a primary concern of many PHR owners [2], [4], [5], and [6], due to the highly sensitive information contained in the PHRs [5] such as personal diseases, medical history, allergies, mental health information, and diagnosis and physicians' recommendations.

Thus, an access control for the PHRs is necessarily required in order to enforce an access policy on who is able to access or modify any particular PHR. In other words, a PHR owner must be able to define an access policy in order to control all accesses on each of his/her PHRs.

Unfortunately, the traditional access control schemes—such as role-based access control (RBAC) [7] and attribute-based access control (ABAC) [8]—are not suitable for the PHR because those schemes typically require the system users and the storage providers to be in the same trust domain [9]. In a PHR system, however, the PHR users (i.e., PHR owners and owner-authorized users) and the PHR providers are obviously in different trust domains [2], [4], [5], and [6]. A widely adopted method is an encryption-based access control [10], [11]. That is, a PHR record will be protected from its source (i.e., encrypting at the PHR owner's client before storing on any PHR storage). Thereby, only the authorized users who possess a decryption key will be able to decrypt the encrypted PHR. Since PHR providers are not included in the list of authorized users, the providers cannot access the stored PHRs. For this reason, PHRs are securely stored.

Nowadays, several encryption schemes are available. The schemes can be classified into two types including one-to-one encryption and one-to-many encryption. The one-to-one encryption (e.g., symmetric-key encryption (SKE) [12], public-key encryption (PKE) [13] and identity-based encryption (IBE) [14]) allows only a particular user to decrypt a ciphertext. Meanwhile, the one-to-many encryption (e.g., policy-based encryption (PolBE) [15], key-policy attribute-based encryption (KP-ABE) [16] and ciphertext-policy attribute-based encryption (CP-ABE) [17]) allows a set of authorized users to decrypt a ciphertext. In reality, a PHR record can be accessed by multiple users, for example, the owner himself/herself, family members, caregivers, and physicians. Thus, the one-to-many encryption is suitable for the PHR system.

Concept of the one-to-many encryption is to empower a PHR owner to specify an access control policy for each PHR record during an encryption process. The access control policy will be expressed in terms of the roles or attributes of authorized users. For example, if the policy states that "Physician OR Caregiver". The

resulting encrypted PHR can be decrypted by the authorized users who possess the key with the “Physician” and/or the “Caregiver”. PolBE can provide an access policy described above [4]. However, a user collusion problem enables malicious users to escalate their access privilege to unauthorized PHR [17], [18] by combining their private keys together. To prevent such the problem, CP-ABE scheme mathematically links all of the attributes together to produce a private key [17]. Thereby, CP-ABE private keys cannot be combined together in order for making the user collusion anymore. Therefore, CP-ABE becomes one of the most adopted schemes for the PHR as was used in [19], [20], [21], and [22]. KP-ABE was used in [23], [24], and [25]. Under KP-ABE, an access policy will be transformed into a decryption private key instead of a ciphertext. Meanwhile, the ciphertext will be embedded a set of attributes. If the KP-ABE is adapted to the PHR system, a PHR owner must define a set of specific attributes for encrypting a PHR and then generate different private keys for each authorized user before sharing the PHR. KP-ABE not only provide an unintuitive way for protecting and sharing the PHR, but also make the PHR system more complicated in a comparison with using CP-ABE. Moreover, KP-ABE also requires the PHR owner to generate and distribute the decryption private keys to each authorized user himself/herself whereas the PHR owner is not required to do that under the CP-ABE scheme. For this reason, KP-ABE is not suitable for the PHR. In this way, the CP-ABE scheme is selected to be applied for securing the PHRs in this research.

Under the CP-ABE scheme, the user and attribute management is centralized. In other words, all users and all attributes are managed by a single centralized authority. In practical, however, a PHR system consists of multiple users with different roles/attributes [26], [27], such as, patients, family members, caregivers, physicians, other medical practitioners, emergency responders, and health insurers. To employ the original CP-ABE to the PHR system directly, several issues and limitations, such as a system scalability problem, a single point of failure problem, and an efficient user management problem [23], [24], and [26], must be considered.

This research proposes a multi-authority secure personal health record (MA-PHR) scheme. The proposed scheme can improve the issues and limitations discussed previously. The tasks of handling users and attributes are

distributed to multiple expert authorities under the proposed scheme. With the proposed scheme, a company or an individual is allowed to establish its own authority in order to support the management of users and attributes related to its expert domain. For example, a hospital can establish a healthcare authority for supporting its medical practitioners as well as a patient can create his/her own personal authority for supporting his/her family members, caregivers and relatives. Under the proposed scheme, each authority can join and take part in establishing the network of a global MA-PHR system so that the joining authorities can collaborate with one another.

Additional contribution of this research is to preserve the privacy of the PHR owners. A PHR dual layer protection is proposed in this research in order to enable the PHR owner to take a full control on his/her PHRs in terms of who can download, upload or even delete his/her (encrypted) PHR records stored on a PHR provider. The PHR owner can selectively grant one or more access permissions to each user. For example, an owner may grant the upload and the download permissions to his/her personal physicians while he/she may grant only the download permission to his/her family members and caregivers. A hierarchical trust model is applied in this research to provide a method to verify a user from a different authority. Moreover, an end-to-end secure PHR sharing scheme is also presented in order to guarantee that only the authorized users are able to access the PHR information.

2. Objectives

The aim of this research is to propose a simple approach to transform a traditional centralized-based user authority PHR system to a multi-user authority PHR system (MA-PHR system for short). The proposed MA-PHR system comes with several security mechanisms in order to protect the security and privacy of the PHRs. The proposed approach addresses issues and limitations of the centralized-based PHR systems including the system scalability, and the single point of failure problem. Moreover, the proposed system provides a simple, easy, and efficient user management. The security mechanisms used in this research assure that the PHR owner can take a full control over his/her shared PHRs. In other words, the owner can selectively share each of his/her PHRs to any desired user. The access control mechanism proposed in this research allows the PHR owner to define different access permissions, covering READ and STORE actions separately, to different group of users, or a user. To proof the concept of this research, a software prototype is developed and the security issues as well as the usability issues are also evaluated.

3. Preliminary technique

The background technique on the original CP-ABE scheme [17] is presented in this section. CP-ABE is a one-to-many asymmetric-key encryption algorithm that allows multiple authorized users to access the encrypted data. Under CP-ABE scheme, a set of attributes reflecting the user's roles is transformed into a decryption CP-ABE private key and assigned to the corresponding user. The data owner can then specify an access policy over a set of attributes of authorized users for encrypting each particular PHR. For example, the policy is expressed as follows.

Policy P = "(family_member) OR (physician AND hospital-A) OR (physician AND hospital-B)"

As a result, any user can decrypt the data encrypted using CP-ABE with the policy P if and only if he/she possesses the CP-ABE private key satisfying the policy P. In the example, the list of authorized users includes the owner's family members and the physicians of hospital-A or hospital-B. The CP-ABE scheme consists of four steps as follows.

The first step is the Setup phase. This phase generates the public parameters *PK* and the master secret key *MSK*. The *PK* consists of the generator g, g^{β}, and $e(g,g)^{\alpha}$, where e is a computable symmetric bilinear map. The *MSK* is the value β and g^{α}. The *PK* is revealed to the public, while the *MSK* must be kept secret.

The second step is the Encryption phase. This phase takes as input the public parameters *PK*, a plaintext message *M*, and an access policy tree *T* and outputs the ciphertext *CT*. The access policy tree can be generated for any policy using a set of boolean formulas. For example, the policy P above can be transformed into an access policy tree as shown in Fig. 1.

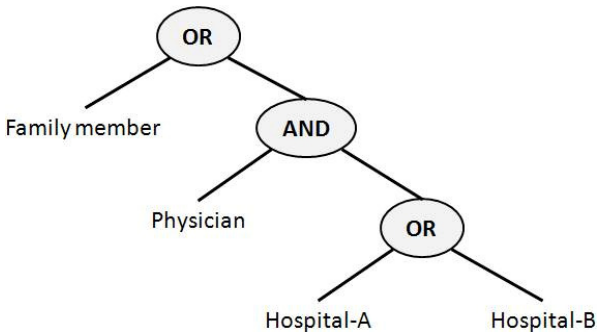

Fig. 1. An access policy tree of the policy P

The third step is the KeyGen phase. This phase requires a set of input including the public parameters *PK*, the master secret key *MSK*, and a set of user attributes *S*. The attributes *S* are mathematically incorporated and transformed into the private key *SK*. This phase generates the private key *SK*, associated with the set of attributes *S* that describes the key, as output.

The last step is the Decryption phase. This phase takes as input the public parameters *PK*, the ciphertext *CT*, and the private key *SK*. The ciphertext *CT* that was encrypted with the access policy tree *T*, will be decrypted if and only if the set of attributes *S* associated with the private key *SK* satisfies the policy tree *T*.

Typically, CP-ABE is designed for a single user authority (UA) environment. In other words, all users must be managed by the single centralized UA. In this research, a mechanism to extend the CP-ABE to deal with a multi-UA environment is proposed. Under the proposed scheme, any traditional CP-ABE-based PHR system can be modified to handle a multi-UA environment.

4. The proposed MA-PHR scheme

The detailed construction of the proposed MA-PHR scheme is presented in this section. First, the system model and assumptions are described. Second, the proposed mechanism to extend any traditional CP-ABE to a multi-UA environment is described. Third, the security mechanisms including hierarchical trust model, PHR dual layer protection, and end-to-end secure PHR sharing, are presented in order to explain the security of the proposed scheme.

4.1 System models and assumptions

Under the proposed scheme, there are five entities as shown in Fig. 2-4 including root authority (RA), user authority (UA), audit server (AS), PHR server, and user. RA verifies and certifies all UAs and AS. There can be multiple UAs owned by companies or individuals, in which each of them manages users and attributes related to its expert domain independently. AS records all requests and transactions and generates a log report for the PHR owners or the authorized users if any dispute is occurred. RA, UAs and AS are assumed to be trusted by the users in the system. The PHR server provides the users with an abundant storage space and a large network bandwidth for storing and sharing PHRs. The PHR server can be a third-party cloud storage provider and it can be considered untrusted. The users in question can be anyone with different roles such as the PHR owners (patients), the family members, the physicians. In addition, all connections under the proposed scheme are always secured by the secure sockets layer (SSL)/transport layer security (TLS) protocol [28].

4.2 Modified CP-ABE initial settings

The original CP-ABE setup phase typically generates the mathematical linked key pair, namely, public parameters *PK* and master secret key *MSK*. The *PK* is revealed to the public and required as the explicit parameter in the CP-ABE private key generation and the ciphertext encryption/decryption processes. The *MSK* must be kept secret by the trusted UA and also required as the explicit parameter in the private key generation. Thereby, the ciphertext *CT* can be decrypted with the private

key if and only if both the ciphertext and the private keys are generated from the same *PK* and *MSK* key pair.

To modify the original single-authority CP-ABE to multi-authority CP-ABE, the key idea is to generate the *PK* and *MSK* key pair at the RA and then securely distribute the generated key pair to all the trusted UAs. Next, all UAs generate the CP-ABE private keys for their users using the same key pair. Thus, the PHR user of an authority can share his/her encrypted PHR to another user of a different authority transparently, as if they were in the same global authority. The proposed MA-PHR system consists of four setup phases as follows:

4.2.1 MA-PHR core system setup

The RA is used as the root of all UAs under the proposed scheme. Therefore, the MA-PHR core system setup is initiated by the RA as shown in Fig. 2. The RA first executes the CP-ABE Setup algorithm to generate the *PK* and *MSK* key pair (denoted as 1 in Fig. 2). Next, the generated key pair is kept secret by the RA and prepared for securely distributing to all UAs. Specifically, the generated key pair will be used as the root of all CP-ABE private keys that will be issued by the UAs to all PHR users in the system.

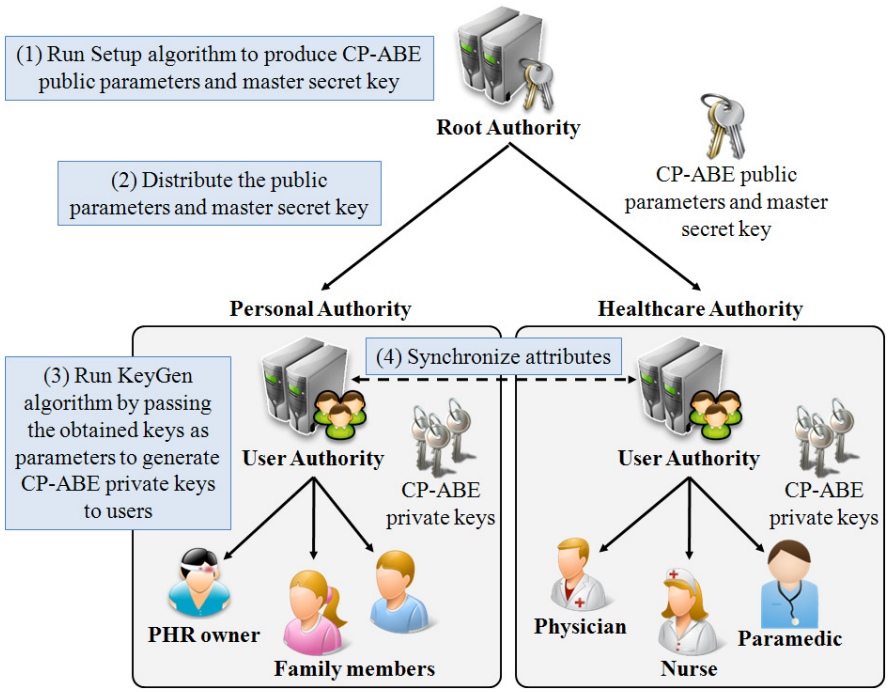

Fig. 2. Setting up the CP-ABE for a multi-authority environment

4.2.2 User authority setup

After setting up the MA-PHR core system, the RA is now ready to accept the UA request to join the system. The new UAs will send their request to join the network of the system. Upon receiving the request, the RA must first verify the authenticity of the requesting UA. After a successful verification process, the RA then securely distributes the *PK* and *MSK* key pair to the new UA (denoted as 2 in Fig. 2). The UA will use the acquired key pair for generating the CP-ABE private keys to its members during the user key generation phase described next.

4.2.3 User key generation

When a new user is registering with a certain UA, the UA will execute the CP-ABE KeyGen algorithm using the acquired *PK* and *MSK* key pair as the specific parameters together with a set of user attributes associated with the user's roles (denoted as 3 in Fig. 2). Therefore, the user will be assigned the CP-ABE private key that is able to decrypt any encrypted PHR from any authority in the system as if the decryptor and the PHR owner were in the same global authority.

4.2.4 Inter-authority synchronization

The proposed scheme allows a company or an individual to create and manage its own UA locally. For example, a hospital can create a healthcare authority that has a certain set of attributes such as physician, nurse, and paramedic, whereas a patient can create a personal authority that has a different set of attributes such as PHR owner, family member, relative and friend. As a result, each UA can have a different set of attributes. For this reason, all UAs must synchronize their attribute sets with each other periodically (denoted as 4 in Fig. 2), in order to enable a PHR user from a different authority to be able to define a set of attributes in his/her access policy for a group of users from another authorities during an encryption process.

4.3 Security mechanisms

Three security mechanisms of the proposed scheme are described in the following subsections including hierarchical trust model, PHR dual layer protection, and end-to-end secure PHR sharing.

4.3.1 Hierarchical trust model

As mentioned earlier, the proposed scheme allows a company or an individual to establish its own UA for handling users and attributes related to its expert domain. For example, a healthcare authority is created and managed by a hospital, whereas a personal authority is owned by a patient. On the one hand, the patient from the personal authority can share his/her PHRs to the physicians from the healthcare authority. On the other hand, the physicians can contribute in updating the patient's PHRs accordingly. Since the patient and the physicians are from different authorities, an inter-authority user verification mechanism is required in order to build a mutual trust relationship among them.

To initiate the inter-authority user verification mechanism, the proposed scheme applies a hierarchical trust model (as shown in Fig. 3), which is a feature provided by the SSL/TLS protocol. With the hierarchical trust model, consequently each UA can certify and issue SSL/TLS certificates to its members locally. For the example described above, the personal authority and the healthcare authority can certify and issue SSL/TLS certificates to their members locally. Later, two PHR users from different authorities can mutually verify the identity and authenticity of one another.

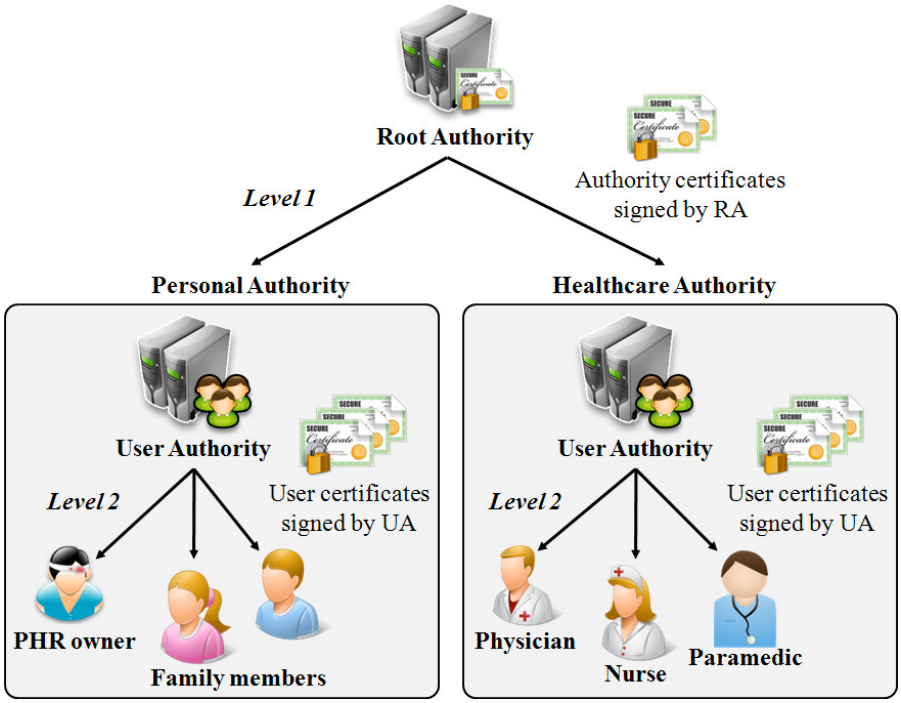

Fig. 3. A hierarchical trust model

The hierarchical trust model consists of two levels: authority level and user level. Each level is allowed to verify and certify its sub-entities. The root of the hierarchical trust model is the RA. When a new UA is created and requested to join the proposed system, the RA must verify an identity of the requesting UA and issue a unique authority certificate to the requesting UA (denoted as '*level 1*' in Fig. 3). Then, the requesting UA uses the obtained authority certificate for a chain certifying and issuing a unique user certificate to each of its PHR users locally (denoted as '*level 2*' in Fig. 3). Later, a PHR user can use the obtained user certificate for authenticating and establishing a secure channel [27], [29]. Because each UA generates the user certificates to its members from the same root certificate, the user can verify the certificate of a peer user across authorities.

4.3.2 PHR dual layer protection

To protect the PHR information from an unauthorized reading and modifying, the proposed scheme presents the PHR dual layer protection which consists of two protection layers: read protection layer and store access control layer. A PHR owner is allowed to define access policies for READ and STORE actions

separately. The read protection layer protects the PHR information from an unauthorized read action by encrypting the PHR using the modified CP-ABE at the source before securely uploading to the PHR server through a secure channel (denoted as 4 in Fig. 4). According to the CP-ABE scheme, the PHR owner can specify an access policy for each particular PHR and the policy will be transformed and embedded into the encrypted PHR. The resulting encrypted PHR can be decrypted by any user if and only if the user possesses the CP-ABE private key containing the set of attributes satisfying the associated access policy pre-specified by the owner.

The store access control layer protects the encrypted PHRs stored on the PHR server from an unauthorized access. The PHR owner can selectively assign one or more of the three access permissions—including upload, download and delete permissions—to each user. A user can perform only the authorized actions on the encrypted PHRs stored on the PHR server. With the proposed dual protection layers, thus the owner can have a full control over which users can read or modify his/her PHRs.

4.3.3 End-to-end secure PHR sharing

Once a user logs in to the MA-PHR system, the certain UA that the user has been registered will verify his/her authenticity and allow the user to obtain his/her CP-ABE private key and user certificate if his/her claim is valid (denoted as 1 in Fig. 4). The CP-ABE private key will be used as the PHR decryption key. The user certificate will be used for authenticating himself/herself and establishing a secure communication channel when the user contacts with any server. Under the proposed scheme, the PHR owner can freely grant any of the three access permissions (i.e., upload, download, and delete permissions) to any selected user. As depicted in Fig. 4, for explanation purpose, assuming that Alice would like to grant some permission on her PHRs to her personal physician Bob. Alice can achieve the permission granting task through her UA (denoted as 2 in Fig. 4). The granted access permissions will be securely recorded in the UA database and will be used when Bob requests access to any PHR record of Alice. In this example, since Alice and Bob are members of different authorities, the synchronization mechanism from Alice's authority to Bob's authority is required, in order to synchronize all parameters

between the two authorities such as the sets of attributes, the lists of granted access permissions, and the lists of users (denoted as 3 in Fig. 4). Typically, all authorities always periodically synchronize their attribute sets, user lists, and granted access permission sets with each other.

The proposed scheme offers an end-to-end PHR protection method. That is, the proposed scheme will encrypt Alice's PHR information using the modified CP-ABE at Alice's client before securely transmitting to the PHR server through a secure channel (denoted as 4 in Fig. 4). Since no decryption key is stored on the PHR server, even the PHR server itself cannot read the content of the PHR information stored. The resulting encrypted PHR would be decrypted by only the authorized users at their clients. Thus, Alice can assure that her PHR information will be accessed by her authorized users only.

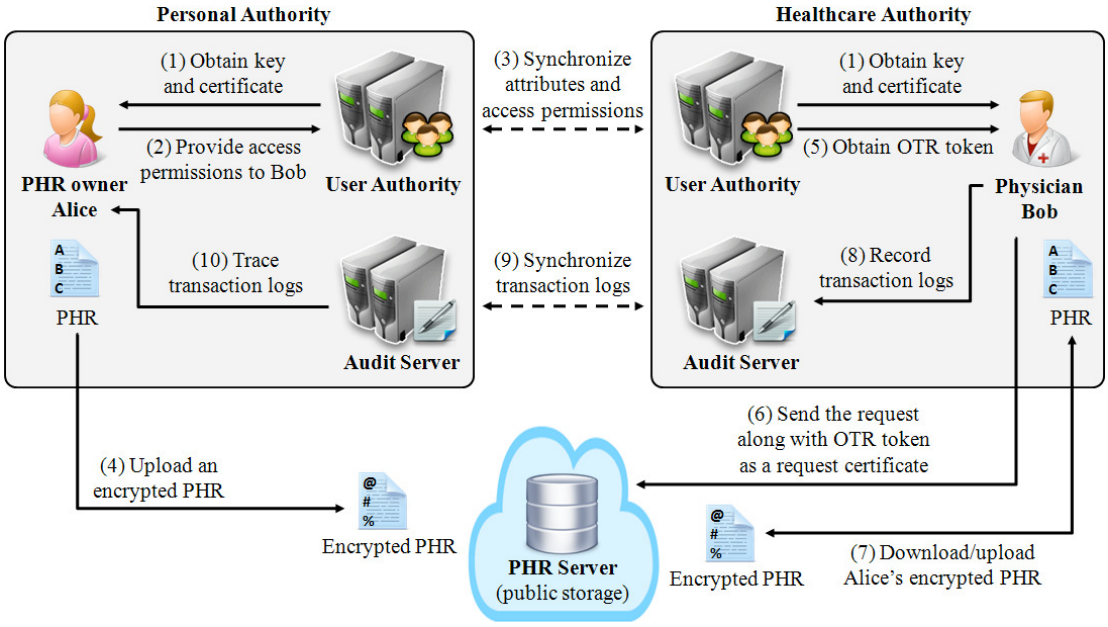

Fig. 4. End-to-end secure PHR sharing workflow

Assuming that Bob needs to access or modify Alice's PHR information, Bob must request and obtain a one-time request (OTR) token from his UA (denoted as 5 in Fig. 4). The OTR token typically contains Bob's access permissions granted by Alice and the token will be given a specific expiration date/time. Therefore, Bob can only perform actions indicated on the token during a valid time period. In other words, Bob cannot re-use the token if the token lifetime is expired. After Bob gets

the OTR token, Bob sends the request message along with the obtained OTR token to the PHR server (denoted as 6 in Fig. 4). The PHR server will verify the token and Bob's request. Next, Bob can perform any request action once his request is verified (denoted as 7 in Fig. 4). In addition to provide a non-repudiation feature [27], [30], Bob's request will be recorded as a transaction log on the AS (denoted as 8 in Fig. 4). The transaction logs will be periodically synchronized from Bob's authority to Alice's authority (denoted as 9 in Fig. 4). This way, Alice can keep track of all accesses to her PHRs from the log report generated by her AS later (denoted as 10 in Fig. 4).

5. System development

The software prototype of the MA-PHR system is developed in order to proof the concept of the research. Some detailed design of the software prototype is given. Fig. 5 shows the proposed system structure that consists of client and server modules. The client side is executed on any PC computer supporting Ubuntu OS. The underlying client side consists of the frontend and the backend. The client frontend is responsible for rendering a graphical user interface (GUI) and inputting the user commands, which is implemented using Java language. The client backend gathers several low-level modules such as encryption, security and privacy, network, and user management modules, which is implemented using C language. The client frontend and backend communicate with each other via the Java Native Interface [31]. The server side includes four servers: namely, root authority (RA), user authority (UA), audit server (AS), and PHR server, which are fully implemented using C and SQL languages. Each server has a local database for independently storing the information. The server and the client sides always securely communicate with each other via an SSL/TLS secure channel.

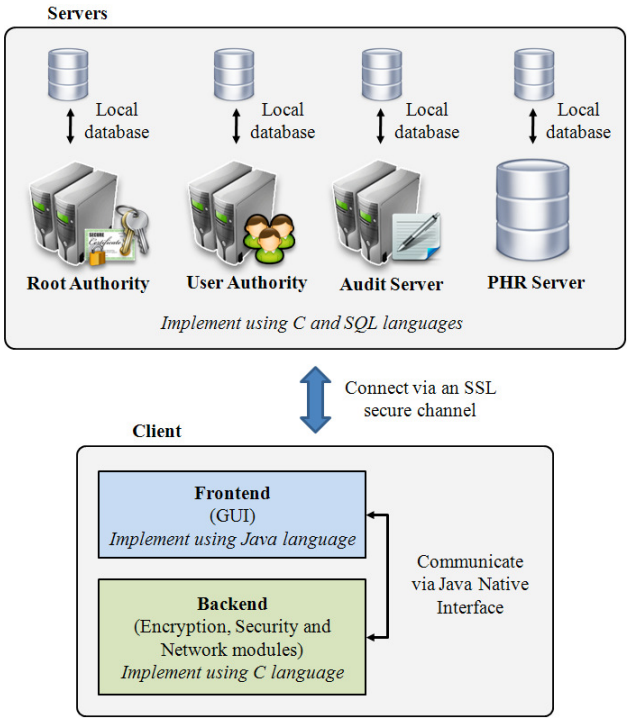

Fig. 5. The proposed system structure

6. System demonstration

The demonstration of the developed MA-PHR system is presented in this section. For the demonstration purpose, assuming that a PHR owner Alice would like to share her PHR records to some selected people. The selected people include Alice's family members: John and James, and Alice's personal physician: Bob. As the different professional roles, Alice decides to grant different sets of access permissions to each of the selected people as shown in Fig. 6. That is, John and James who are Alice's family members would be assigned only the download permission or the read permission, while the physician Bob would be assigned the upload and the download permissions or both the read and store permissions. However, none of the selected people would be assigned the delete permission. As a consequent, John and James can only download Alice's PHR records for informing and updating their knowledge about Alice's health condition, while Bob can download Alice's PHR records for using in diagnosing Alice's diseases and symptoms whereas Bob can also upload or append Alice's PHR record (e.g., medical history or recommendation). In other words, the developed system allows Alice to selectively grant any access permission to each particular person according to his/her professional roles.

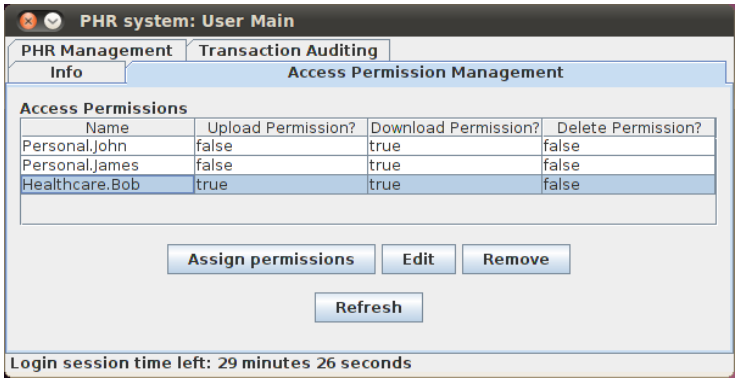

Fig. 6. Alice assigns different sets of access permissions to each of the selected people

After assigning the access permissions, Alice loads her PHR record to her client application and then specifies an access policy for that record as shown in Fig. 7. The specified access policy is expressed as follows: "Personal.family_member OR Healthcare.caregiver OR Healthcare.physician". Thus, Alice's family members who

are in Personal authority and the caregivers or the physicians who are in Healthcare authority can decrypt the resulting encrypted PHR record. Fig. 8 shows the process of encrypting the PHR record occurred on Alice's client application before the resulting encrypted PHR would be securely uploaded to the PHR server.

Once the physician Bob wants to download Alice's PHR record, Bob initializes his download request on the specific PHR-owner name (i.e., Alice) as shown in Fig. 9. Next, Bob will be informed of all Alice's encrypted PHR records stored on the PHR server as shown in Fig. 10. Bob can select and download the requested encrypted PHR from the PHR server to his client application. Then, the decryption process will take place at Bob's client. Bob is able to decrypt Alice's encrypted PHR if and only if his CP-ABE private key satisfies the associated access policy pre-specified by Alice as shown in Fig. 11. Furthermore, Bob also has permission to upload or update Alice's PHR records. Specifically, Bob can achieve this task by making the upload request on Alice's record on his GUI interface in Fig. 9. Next, Bob can load a new or an updated Alice's PHR record. The PHR record will then be encrypted before securely uploading to Alice's repository on the PHR server as shown in Fig. 8. Moreover, the developed system provides the transaction auditing mechanism that enables Alice to track all actions performed on her PHR records later. Thus, Alice can be informed of all accesses on her PHR records by Bob as shown in Fig. 12.

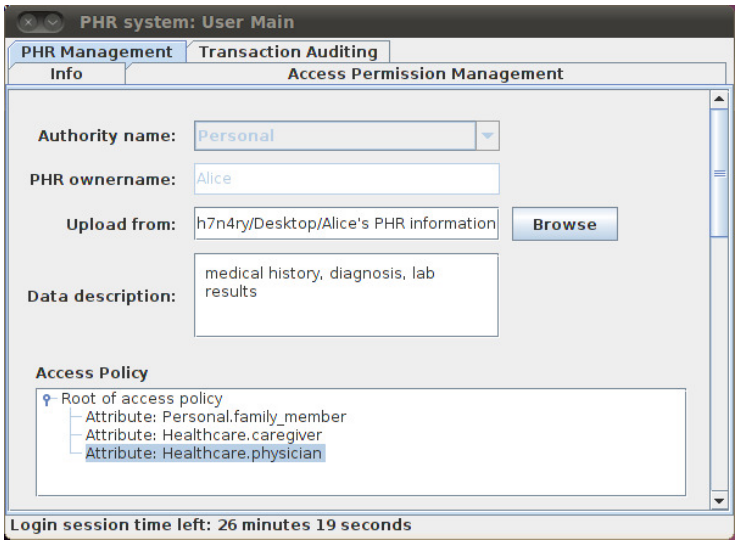

Fig. 7. Alice specifies an access policy for her PHR record

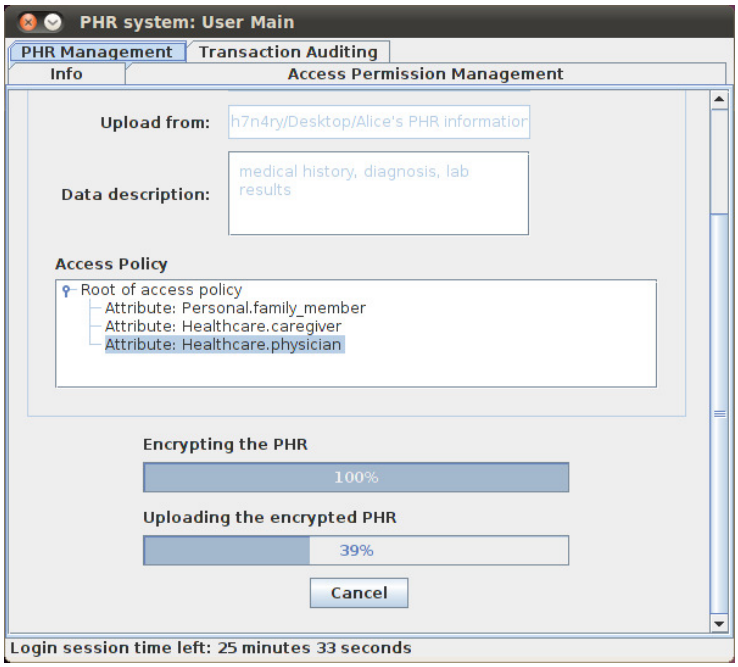

Fig. 8. Alice's encrypted PHR is uploaded to the PHR server

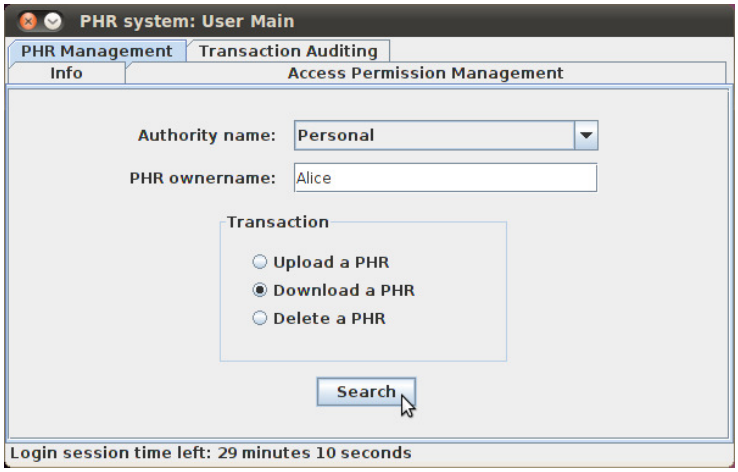

Fig. 9. Bob initializes his download request

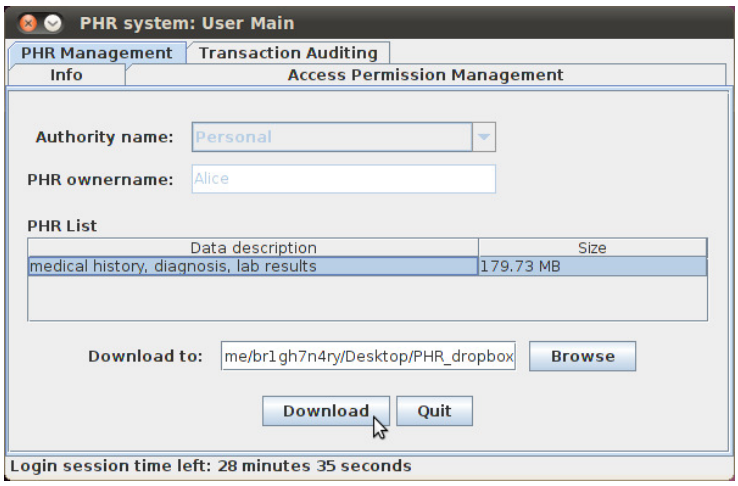

Fig. 10. Alice's encrypted PHR stored on the PHR server

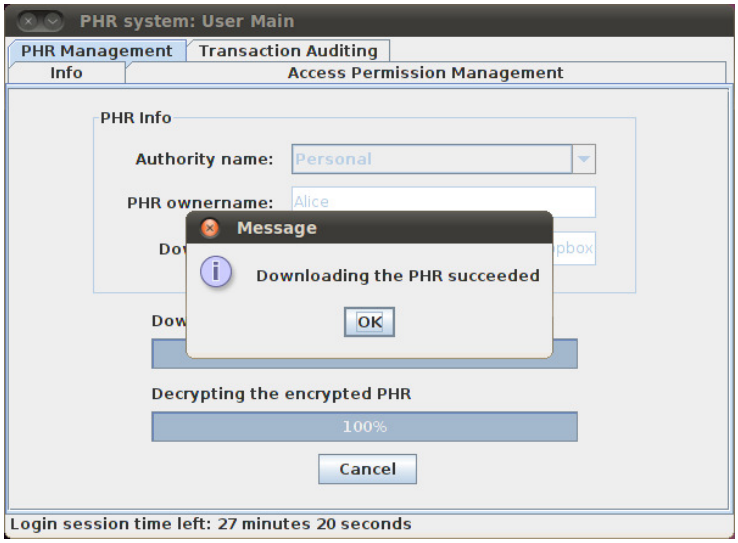

Fig. 11. Bob can access Alice's PHR as requested

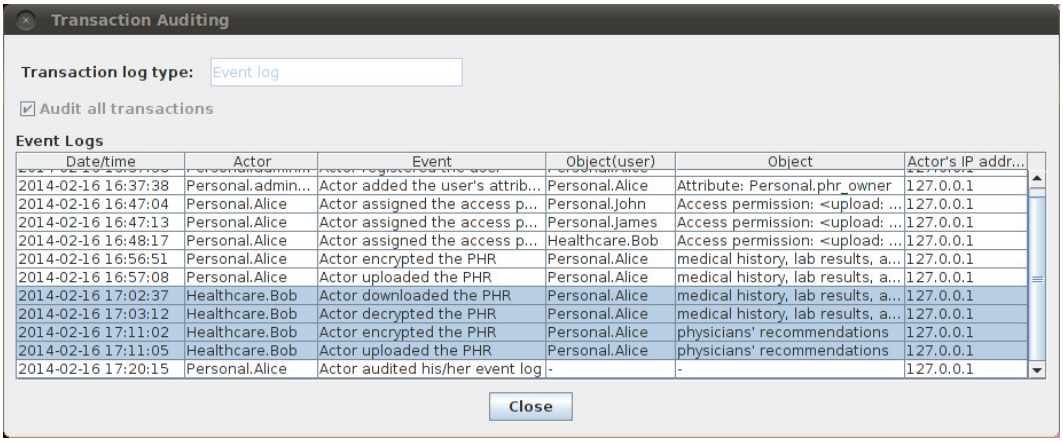

Fig. 12. Alice traces all accesses on her PHRs

7. Security and usability discussions

In this section, the security and usability issues of the proposed system is discussed. The security issues consist of five attack models including storage intruders, unauthorized actions, replay attacks, PHR client attacks, and non-repudiation cases. The usability issues consist of four usability models including efficient encryption scheme, efficient network bandwidth usage, read and store access control, and system scalability and availability.

7.1 Security issues

Five attack models for the proposed scheme are discussed in this section including storage intruders, unauthorized actions, replay attacks, PHR client attacks, and non-repudiation cases. Additionally, all connections under the proposed scheme are always secured by SSL/TLS protocol. Therefore, the cases of eavesdropping on the network traffic are eliminated.

The first attack model is the storage intruders. Since the PHR server can be a third-party cloud storage provider, the proposed scheme considers the PHR server untrusted. That is, a storage intruder or even a storage administrator may try to access the PHR information stored on the PHR server. To protect the PHR information, therefore, the information is always encrypted at its source before uploading to the PHR server and the decryption keys are securely stored on the separated trusted user authority (UA). Thereby, the encrypted PHR stored on the PHR server is protected as long as the storage intruder or the storage administrator cannot break the cryptographic primitives used. And, the decryption key is not accessible by any unauthorized person.

The second attack model is the unauthorized actions. The proposed scheme offers the read and store actions. A PHR owner (e.g., Alice) can selectively grant the permission either read or store to a desired user (e.g., Eve). To read, store or modify Alice's PHR, Eve must obtain the OTR token that indicates the permission on the actions granted by Alice from Eve's UA (denoted as 3 in Fig. 13 and 4 in Fig. 14). After that, Eve must present the obtained OTR token to the PHR server for claiming her access permission on the requested PHR (denoted as 5 in Fig. 13 and 6

in Fig. 14). Nevertheless, Eve may try to modify her OTR token in order to escalate her permission to perform any unauthorized action. To prevent such action, the OTR token will be digitally signed with the UA private key. The PHR server can verify the integrity of the token with the corresponding UA public key. This method ensures that the OTR token has not been modified by any unauthorized user.

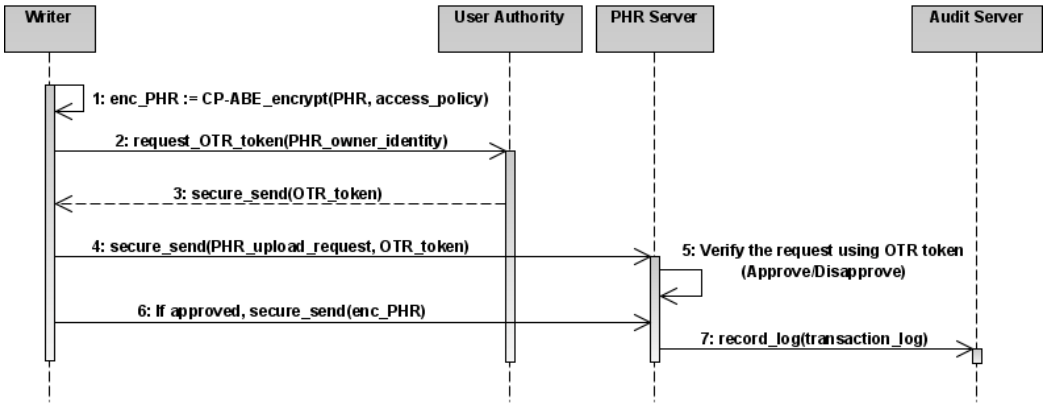

Fig. 13. The sequences of the PHR uploading transaction

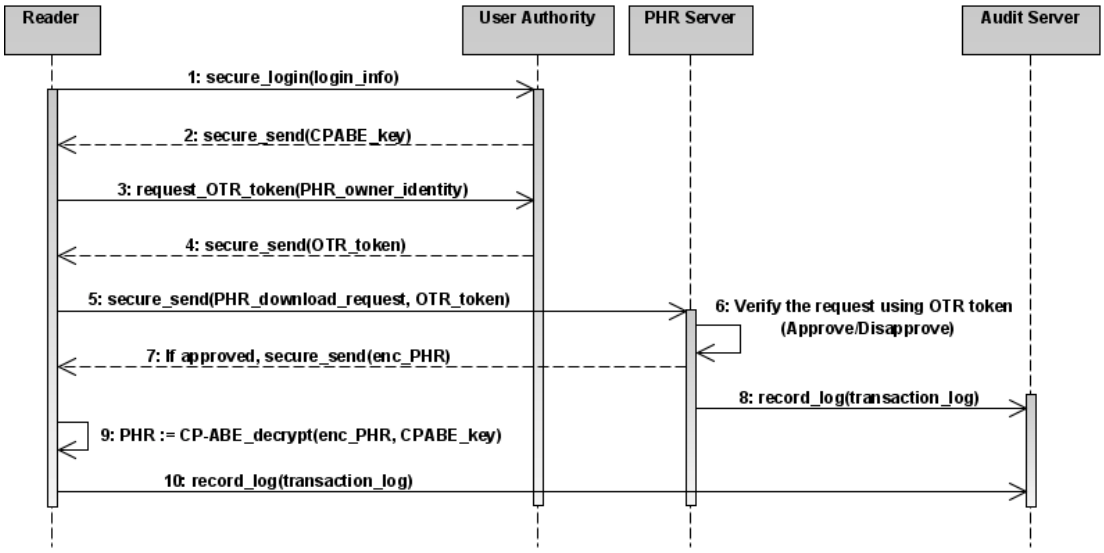

Fig. 14. The sequences of the PHR downloading transaction

The third attack model is the replay attacks. Suppose that Eve has been revoked the permission that was previously granted by Alice. In order to gain her permission back, Eve may try a replay attack by using the previous OTR token that was obtained to be used instead of the current token. However, the OTR token used in the proposed scheme is a certificate with a specific expiration date/time, Eve

will not be able to reuse the obtained OTR token once it is expired. Thus, the lifetime of the OTR token must be kept short and optimistic.

The fourth attack model is the PHR client attacks. Typically, once a user logs in to the proposed system, the user gets his/her user certificate and CP-ABE private key from his/her associated UA. The certificate and the private key will be stored on the user's client for convenience purpose. Therefore, the attacker may try to steal the user certificate or the CP-ABE private key stored on the client. To protect the key and certificate, all stored data will be encrypted using a symmetric-key encryption method with the user's password. The password will only be stored in the PHR client module's memory area for convenience purpose, assuming the attacker cannot gain his/her privileges to the superuser/root mode to dump the memory section of the PHR client module for reading the user's password. This concept can be used with any Linux-based operating system [32].

The last attack model is the non-repudiation cases. The audit server (AS) is responsible for recording all requests and transactions that occur in the system (denoted as 7 in Fig. 13 and 8, 10 in Fig. 14) and providing the transaction auditing mechanism to the PHR owners. In other words, the PHR owner can keep track of his/her PHR records by inspecting the log report generated from the AS (e.g., Fig. 12). Furthermore, the transaction auditing mechanism also enables the system administrator to monitor or detect malicious users. This mechanism is very important to guarantee a non-repudiation feature in the proposed scheme.

7.2 Usability issues

To evaluate the proposed scheme on the current environment, four usability issues are discussed. First, the practical aspect of the encryption scheme is evaluated. Second, the required network bandwidth for the proposed scheme to work efficiently is discussed. Third, the control over the read and store access permissions are investigated. Lastly, the system scalability and availability is discussed.

First, the proposed scheme selects CP-ABE as the main encryption algorithm. CP-ABE is leveraged to protect the confidentiality of the PHR information. The PHR records will be encrypted using CP-ABE at the client before uploaded to the

PHR server. Since the PHR client has several limitations such as the processing unit, the power consumption, and the storage capacity, the efficiency of the CP-ABE scheme is discussed in this section. The underlying of CP-ABE scheme is the advanced encryption standard (AES) [33] in cipher block chaining (CBC) [34] mode which is a symmetric-key encryption method. Hence, the encryption and decryption time of the AES are the same. AES has good performance in terms of processing time and power consumption [35] compared with other widely adopted symmetric encryption schemes. The results from the research thesis conducted by Hirani [36] showed that AES consumes less battery power and less encryption time than that of CAST and IDEA encryption schemes on the encryption of a 5 MB file. All the schemes use 128-bit key size. AES consumes 58% less battery and 70% less time than CAST, while AES takes 92% less battery and 110% less time than IDEA. The experiment by Nadeem and Javed also showed that AES has good performance in terms of the average time consumption than Triple-DES algorithm [37]. Thus, AES was given a choice of the encryption algorithms used in SSL/TLS protocol [38]. With CP-ABE scheme, moreover, a PHR owner can encrypt his/her PHR information for multiple authorized users at once, resulting in efficiency of storage and computation.

Network bandwidth availability is another important limitation of the PHR client side. Three objects are to be used when a PHR user uploads, downloads or deletes any PHR, the user certificate, the CP-ABE private key, and the OTR token. To reduce the communication cost between the user client and the UA, all three objects will be downloaded from the UA and stored in an encrypted form on the user client module. In order to reduce the OTR token issuing task occurred at the UA, furthermore, the OTR token is designed to have a limited lifetime. In other words, the OTR token stored on a user's client can be used multiple times until the token lifetime is expired.

Under the proposed scheme, a PHR owner has a solution to selectively grant the read (download) and store (upload and/or delete) permissions to a particular user. This method makes our scheme realistic because in the real world adoption often that a PHR owner would like to grant not only the read permission, but also the store permission to his/her contributors. For example, the

PHR owner Alice may assign the “read only” permission to her family members whereas Alice may want to assign both the read and the store permissions to her personal physicians. With the proposed scheme, Alice is freely to assign any access permission to a specific person according to his/her professional roles.

The last usability model is the system scalability and availability. A PHR system can consist of multiple user domains such as personal domain, healthcare domain, and emergency domain. Therefore, handling all user domains using a single centralized UA can lead to system scalability and system availability (single point of failure problem) problems. For this reason, the proposed scheme is intentionally designed to be a multi-authority system that allows each user domain to manage its users locally. While the proposed scheme allows each local domain to be part of a global domain when works across domain. The multi-authority feature allows the system to be scaled up. That is, the joining of a new user domain does not affect the user and attribute management tasks of another domains else significantly. Moreover, the multi-authority feature also makes the system distribute the tasks of the user and attribute management to multiple authorities. This reduces an occasion of the single point of failure problem.

8. Analysis and discussion of the proposed system and related systems

Analytical analysis and discussion of the proposed system and related systems is presented in this section. The most adopted related systems are selected to compare with the proposed system on advantages, disadvantages, and limitation issues. The related systems include Indivo health platform [39], Microsoft HealthVault [40], Google Health [41], and PCEHR system in Australia [42].

8.1 Indivo health platform

Indivo (formerly PING) [39] is an open source, open standards personally controlled health record, which provides an open standard application programming interface (API). Indivo is a web-based PHR-applications/systems platform that allows external software developers to use its platform as a health data storage backend for developing their own web-based PHR applications or systems via its open standard API calls. Indivo was greatly successful in widely adoptions [43], [44] such as Microsoft HealthVault [40], Google Health [41], Dossia [45], and Indivo X [39]. Recently, Indivo has been extended for supporting developing native PHR applications connected with Indivo backend on iOS platform [46].

Indivo was originally designed by a patient-centric model, in which a patient (or PHR owner) can collect, maintain, and share his/her medical data with desired people or applications. Indivo provides a fine-grained access control to a PHR owner, via OAuth authorization framework [47], that allows an owner to define which part of the whole data to be shared with others or applications by what actions (e.g., create, read, update or modify, and delete) are allowed. In other words, Indivo provides a PHR owner a one-to-one user-based sharing scheme. To protect the medical data, Indivo uses a database-level encryption method. That is, the plaintext data will be uploaded to store in an encrypted format on the encrypted data storage server while the encryption keys are hosted on a separated physical server managed by the Indivo server [48].

In comparison, the proposed system also offers an owner a fine-grained access control on a read action over his/her health related data. Under the proposed system, an owner can selectively share a read action on any part of his/her

whole health related data with a set of selected people. Specifically, an owner can specify a set of selected people in question by defining a read access control policy based on usernames or attributes of selected users. For example, if a policy states that “Bob OR Physician” can access the data, therefore, there can be only the user named Bob and the users who are a physician able to read that data. Interestingly with the attribute-based access control offered, furthermore, an owner can specify a set of read-authorized users according to roles of selected users by one transaction, not to perform multiple transactions for specifying multiple selected users like Indivo does. In other words, the proposed system provides a PHR owner a one-to-many attribute-based sharing scheme.

However, the proposed system does not provision a fine-grained access control on update or modify, and delete actions over any specific part of data to an owner like Indivo does. The access control for update or modify, and delete actions under the proposed system will be a repository-level access control. That is, an owner, e.g., Alice, can freely grant any set of access permissions based on create (we called “Upload permission”) and delete (we called “Delete permission”) actions to any desired user, e.g., Bob; assuming Bob was granted both the create and the delete actions by Alice; Bob is able to create some health related data to be uploaded or remove any health data stored on Alice’s repository on the PHR server. The resulting uploaded data will not overwrite any data stored previously (this feature was used by Indivo as well [49]). In case Bob wants to update or modify any certain PHR data, he can achieve by uploading a newly updated data to Alice’s repository and then delete the previous versions of that data; to provide a non-repudiation feature, the proposed system provides Alice the transaction auditing mechanism; therefore, Alice can inspect any request or transaction performed by Bob later if any concern or dispute is occurred.

Notably, the proposed system uses a client-level encryption method for protecting health related data, whereas the database-level encryption was used by Indivo. Subsequently, the proposed system provides a better data confidentiality because the plaintext data will only be encrypted and decrypted at a client side and the PHR server will not hold any encryption key. Unlike Indivo, the plaintext data will

be encrypted and decrypted at the storage server, where the Indivo server acts as the encryption keys manager and document access determiner. Furthermore, the Indivo server and the storage server are also considered in the same trust domain. Therefore, if the Indivo server is governed by malicious administrators or is being compromised by adversaries, every encrypted medical data stored on the storage server tends to be leaked. Unlike the proposed system, the PHR server is considered an untrusted external third-party server. Thus, the PHR server and the UA (the server managing the encryption keys) are intentionally located in different trust domains. In order to steal the health data stored in the proposed system, the adversaries or the malicious administrators have to compromise both the UA and the PHR server.

In the aspect of registering, verifying and managing the PHR users, Indivo was designed to be a distributed system in which a user can register to the Indivo system at the physician offices and hospitals or through well-established identity management systems [48] such as certificate-based Kerberos identity management system [50]. Similarly, the proposed MA-PHR system was designed to be a distributed e-healthcare system in which each organization (e.g., clinics and hospitals) or even an individual (e.g., patients and family members) is allowed to establish its own user authority for locally registering, verifying, and managing a group of users related to its expert domain. Section 8.4 will describe the details of this subject again.

8.2 Microsoft HealthVault

Microsoft HealthVault [40] started in October 2007, as a web-based personal health record platform. Microsoft HealthVault equips with an API, which enables external software/hardware vendors as partnership to develop external applications for exchanging medical data stored on a HealthVault account. This system was developed based on Indivo platform [43], [44], thus, it inherently derives almost all of the access control features of medical data from Indivo. That is, a PHR owner is able to share some parts of the whole data to any desired user or application with any given access permissions and with any given access expiration. The access control of Microsoft HealthVault is a one-to-one user-based sharing scheme like Indivo. By our literally investigation, there are three levels of sharing an

owner can grant to other users including view only, view and modify, and act as a custodian of the account; where the latter is a method to give a full access control over all of the HealthVault account to other users.

Similar to the case of Indivo, Microsoft HealthVault uses a database-level encryption method. On the one hand, the database-level encryption enables the HealthVault system to be able to access, maintain, and index medical data stored, resulting in several benefits such as fast data searching and retrieval, data backup, and system maintenance. On the other hand, the database-level encryption enables the HealthVault system to dominate over all of stored sensitive data of every PHR users. Let's consider if the HealthVault system is governed by malicious administrators or is being compromised by adversaries, all sensitive medical data stored can be easily stolen for trading or unauthorized exposures. In comparison, the proposed system is an alternative approach that utilizes the client-based encryption method, in which the PHR storage server and the encryption keys manager (i.e. the UA server) are considered in different trust domains. Our approach guarantees that the health related data stored on the PHR server will only be accessed by the owner-selected authorized users, even the PHR server itself is unable to access the data stored, as long as the PHR server and the UA server are not both compromised. Unlike Indivo and the proposed system, registering, verifying, and managing users in Microsoft HealthVault is centralized, in which a user can make an online registration through its website.

8.3 Google Health

Google Health [41] is another web-based PHR system developed based on Indivo platform [43] started in May 2008. Similar to Microsoft HealthVault, Google Health provides an open API based on SOAP protocol [51] for the web-services interoperability to external software/hardware vendors. In 2008, Google Health (as well as Microsoft HealthVault) was selected by the Military Health System (MHS) organization to evaluate the feasibility of delivering an interoperable PHR for its beneficiaries [52]. Unfortunately, on June 24, 2011, Google announced to discontinue Google Health [53] and thus this system has been shut down completely on January 1, 2013 [41].

Under Google Health system, an access control model for sharing medical data was a profile-based level [52], [54], [55], in which an owner only has a coarse-grained access control over the whole medical data on every single access action. In other words, an owner can only permit a write-only or a read-and-write permission for his/her whole medical data stored [54] to any desired user or application. That is, it is not possible to specify any specific permission for some parts of data. In contrast, the proposed system provides an even more fine-grained access control for sharing a read action on a portion level of health related data. More specifically under the proposed system, an owner is freely to categorize his/her health related data into several parts, each of which can be set a different read access control policy, by using CP-ABE encryption scheme. Consequently, an owner has a solution to selectively share a read action with any desired user on some parts or the whole data. In addition to protecting the medical data stored by Google Health, the data will be stored in an encrypted format using the database-level encryption method. Similar to Indivo and Microsoft HealthVault discussed previously, therefore, all medical data stored in Google Health can be leaked in case the system is governed by malicious administrators or is being compromised by adversaries. In addition to registering, verifying, and managing users; Google Health uses a centralized model by achieving through its website similar to Microsoft HealthVault.

8.4 PCEHR system in Australia

On July 1, 2014, a personally controlled electronic health record (PCEHR) system [42] was released by the Australian government, as a web-based national-scale e-healthcare system for all Australian citizens. Interestingly, the PCEHR system was designed to be a distributed e-healthcare system similar to our proposed MA-PHR system. Specifically, the PCEHR system enables each organization (e.g., hospitals, clinics, emergency departments, and research institutions) to establish its own local repository for holding clinical and medical data [56]. As an example, each hospital can independently store and maintain all patients' medical data in its local repository. Meanwhile, the PCEHR system plays the role of a central data aggregator which is responsible for indexing every medical data physically stored at different repositories, managing and controlling access privileges, and searching and retrieving

medical data [56]. That is, a PCEHR user can search and retrieve medical data of interest through the PCEHR system according to his/her access privileges. To the best of our knowledge, PCEHR [56] does not mention any support of an API (application programming interface) for external software applications.

Documents stored in the repositories of the PCEHR system are protected by a database-level encryption method. With regards to an issue of data access control; clinical and medical document creation is under controlled by a policy specification defined by Health authority agency [56]. Each PCEHR user can create different types of documents according to his/her roles, such as patients can make some notes; healthcare provider can create shared health summaries, event summaries, and discharge summaries of their patients; and Medicare agents can create Medicare history, organ donor, childhood immunizations [56]. To control document access, the PCEHR system supports a two-layer document access control [56], namely, record access control layer and limited document access control layer. A patient can hide his/her related documents from a general view by defining a record access code (RAC). Only the users whom the patient were given RAC code will be able to open and access the patient's record. The patient can further restrict access to some documents by defining a limited document access code (LDAC). Only one LDAC code can be defined, however, which must serve for all documents the patient wishes to restrict. In other words, the read access control of the PCEHR system is a one-to-one user-based sharing scheme. Nevertheless, PCEHR [56] does not state how to control an update or modify action over documents of patients. To delete any document, every user who access to a document can delete it but the document will not be physically removed from the repository exactly, only be removed from document lists and trees [56]. Thereby, the unintentionally removed documents can be recovered. All requests and transactions occurred in the system will be logged in order for later auditing.

In comparison, the proposed MA-PHR system was designed to be a distributed e-healthcare system, but it quite differs from the PCEHR system in which the health related information are stored in distributed repositories but a user management is centralized. Instead, a user management in the proposed system is

distributed to multiple user authorities whereas the health related data are stored on the public cloud-based PHR server. More specifically, under the proposed system each organization (e.g., clinics and hospitals) or even an individual (e.g., patients and family members) is able to establish its own user authority in order for registering, verifying and certifying its own related members locally. It differs from every related system discussed previously; the proposed system employs an attribute-based encryption method (i.e., CP-ABE encryption scheme) for protecting and controlling access to health related data stored on the PHR server. Once a new user asks for registering to the proposed system at a certain user authority, the user would be verified authenticity by that authority and be given a CP-ABE private key containing a set of attributes reflecting his/her roles, then the user will adopt the given CP-ABE private key as a decryption key for accessing the health related data stored on the PHR server. For this reason, the multi-authority user management is very important for the proposed system (even more than other systems) because each user in the system can have different roles, such as family member and medical staff. Thus, verifying users and managing their sets of attributes by the only user-related expert authorities is highly required under the scenario of the proposed system, in order to guarantee that each user was justified by a related expert authority and was issued only the right set of attributes. Furthermore, handling users with multiple user authorities also avoids a single point of failure problem as well as providing system scalability to the proposed system.

As mentioned earlier, the proposed system provides an owner a one-to-many attribute-based sharing scheme while the PCEHR system only provides a one-to-one user-based sharing scheme. The owner has to give RAC and/or LDAC access code to every user who is granted access to the owner's medical documents. Also, the PCEHR system protects the medical data stored using the database-level encryption method similar to other previously discussed systems. Meanwhile, the health related data stored in the proposed system are protected by encrypting at a data source client and would only be decrypted at the client of destined users specified by the data owner.

9. Missing features and future works

There are certain features still missing in the proposed MA-PHR system. This section discusses four missing features and future works of the proposed system including an interoperable platform, standard document formats, standard API for external applications, and access duration control for each part of data.

9.1 Interoperable platform

Since the software prototype of the proposed MA-PHR system was developed based on Debian-core operating system (OS), the developed software can only be run on any PC computer supporting Ubuntu or Debian OS. This makes a barrier to wide adoption of the proposed system. In order to break the adoption barrier, the proposed system has to support an interoperable platform such as a web-based application platform, similar to Indivo, Microsoft HealthVault, Google Health, and PCEHR system. To that point, consequently, the proposed system becomes independent from any OS platform. A user can access the services of the proposed system from any OS platform through any web browser.

Although the web-based application is accessible from any OS platform through a web browser, there exists plenty of arguments that the web application could not provide performance and the user experiences more than that of the native application. From the aspect of Charland and Leroux [57], the web technology today, such as WebKit and HTML5, is getting close to reach the level of performance of the native application technology. Moreover, a cost in developing a web application is cheaper than that of a native application, in which a developer has to support multiple different OS platforms such as iOS, Android, Windows, Mac, and Linux. However, there are some OS-specific features that a native application can utilize but a web-based application cannot [57] such as a push notification service and a hardware acceleration feature. Another interesting approach is to develop a hybrid application that wraps a web application in a native application [57], As a result, the cost is reduced in comparison with the cost of developing a pure native application. However, the developed application can utilize OS-specific features.

9.2 Standard document formats

The proposed system currently treats every data a user wants to upload as a single file or a folder of files. That is, the system will only be responsible for protecting the data and securely delivering the data to only the owner-specified authorized users. However, the proposed system does not support any standard electronic clinical and medical document formats. At present, Continuity of Care Record (CCR) [58], [59] created by ASTM and Continuity of Care Document (CCD) [60] created by HL7 are the most adopted electronic clinical and medical document formats, as CCR and CCD were fully supported by Microsoft HealthVault while Google Health partially supported a subset of CCR [52], [55]. Both CCR and CCD contain information such as patient demographics, allergies and recent medical procedures, medication lists, and insurance and health care provider information, which are expressed in XML (EXtensible Markup Language) format. Hence, both CCR and CCD can be created and read by any e-healthcare system. For this reason, if the proposed system is being extended to support such standard formats, the system would be able to collaborate with other e-healthcare systems seamlessly as well as providing a better user experience.

9.3 Standard API for external applications

An API for external applications is quite important and necessary for the proposed system. The API enables software/hardware vendors as partnership to develop external applications for exchanging the health related data with the proposed system. For example, a mobile medical device vendor that builds a sensor for monitoring the vital signs can develop its own smartphone application that facilitates its customers by automatically uploading the real-time captured vital signs data to store in the customers' PHR repositories managed by the proposed system. As presented in Appendix C, the proposed system provides a simple API for an external developer to build external applications that can connect for calling the client backend modules (e.g., encryption, network management, user management modules) of the proposed system and utilize the proposed system's servers as backend services for securely sharing health related data. However, the provided API is just a simple and non-standard API that the developer can call for modules, which

were compiled as a shared library, via the provided API functions/methods. Anyway, the provided API does not cover a remote procedure call (RPC) that the developer can directly connect to and invoke modules of the server side. One of the necessarily future works, hence, the proposed system should be developed to support a standard RPC API such as SOAP protocol [61] or JSON standard format [62] such that the external developer can connect to and invoke modules of the proposed system universally.

9.4 Access duration control for each part of data

Access duration control is another feature that the proposed system is still missing. As supported in Microsoft HealthVault [52], a PHR owner can freely define access duration of his/her specific part of medical data to each single user. This feature makes a PHR owner more convenient when he/she wants to temporarily share a user or even a group of users access to his/her PHR information, without the need to manually revoke that access permission later by himself/herself.

10. Conclusion

The key contribution of this research is to handle the problems and limitations of the centralized PHR system including system scalability, single point of failure, and efficient user management problems. Those problems and limitations are handled by distributing the user management tasks to multiple UAs instead of using only a single centralized UA. The proposed scheme enhances the original CP-ABE scheme which is designed for a single centralized UA to support the multi-UA environment. The key idea is to distribute the initialized CP-ABE parameters, the key pair: public parameters *PK* and master secret key *MSK*, to all local UAs in the system. With the corresponding *PK* and *MSK* key pair, then the local UAs can establish the multi-UA-compatible CP-ABE environment. All UAs must synchronize the attribute sets with each other periodically.

To preserve security and privacy of the PHR information as well as its owner, the proposed scheme applies a hierarchical trust model to enable an inter-authority user verification mechanism to ensure the authenticity of each PHR user. The proposed scheme offers the PHR dual layer protection, which consists of two protection layers including read protection layer and storage access control layer, making a fine-grained access control on the PHR possible. Moreover, the end-to-end secure PHR sharing scheme is also presented to make sure that the PHR records would be accessed only by the owner-authorized users. Furthermore, the proposed scheme also provides the transaction auditing mechanism that allows the PHR owners to keep track of all accesses performed on their PHR records.

References

[1] Healthit.gov. 2014. What is an electronic medical record (EMR)? Healthit.gov. http://www.healthit.gov/providers-professionals/ electronic-medical-records-emr. (accessed November 1, 2014).

[2] P. C. Tang, J. S. Ash, D. W. Bates, J. M. Overhage, and D. Z. Sands. 2006. Personal health records: definitions, benefits, and strategies for overcoming barriers to adoption. *Journal of the American Medical Informatics Association (JAMIA)* 13(2): 121-126.

[3] AHIM Foundation. 2014. What is a personal health record (PHR)? AHIM Foundation. https://www.myphr.com/StartaPHR/what_is_a _phr.aspx. (accessed November 1, 2014).

[4] K. Garson, and C. Adams. 2008. Security and privacy system architecture for an e-hospital environment. *Proceedings of the 7th Symposium on Identity and Trust on the Internet (IDtrust), Gaithersburg, MD*. 122-130.

[5] B. A. Malin, K. El Emam, and C. M. O'Keefe. 2013. Biomedical data privacy: problems, perspectives, and recent advances. *Journal of the American Medical Informatics Association (JAMIA)* 20(1): 2-6.

[6] K. Caine, and R. Hanania. 2013. Patients want granular privacy control over health information in electronic medical records. *Journal of the American Medical Informatics Association (JAMIA)* 20(1): 7-15.

[7] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman. 1996. Role-based access control models. *Computer* 29(2): 38-47.

[8] D. R. Kuhn, E. J. Coyne, and T. R. Weil. 2010. Adding attributes to role-based access control. *Computer* 43(6): 79-81.

[9] E. E. Mon, and T. T. Naing. 2011. The privacy-aware access control system using attribute-and role-based access control in private cloud. *Proceedings of the 4th IEEE International Conference on Broadband Network and Multimedia Technology (IC-BNMT)*, Shenzhen. 447-451.

[10] R. Singh, V. Gupta, and K. Mohan. 2013. Dynamic federation in identity management for securing and sharing personal health records in a patient-

centric model in cloud. *International Journal of Engineering and Technology (IJET)* 5(3): 2201-2209.

[11] Y. Ding, and K. Klein. 2010. Model-driven application-level encryption for the privacy of e-health data. *Proceedings of the 10th International Conference on Availability, Reliability, and Security (ARES)*, Krakow. 341-346.

[12] B. A. Forouzan. 2008. *Cryptography and Network Security*. McGRAW-HILL. 56-61.

[13] B. A. Forouzan. 2008. *Cryptography and Network Security*. McGRAW-HILL. 294-296.

[14] N. P. Smart. 2003. Access control using pairing based cryptography. *RSA conference on the cryptographers' track (CT-RSA)*. 111-121.

[15] W. Bagga, and R. Molva. 2005. Policy-based cryptography and applications. *Proceedings of the 9th International Conference on Financial Cryptography and Data Security (FC)*, Roseau. 72-87.

[16] V. Goyal, O. Pandey, A. Sahai, and B. Waters. 2006. Attribute-based encryption for fine-grained access control of encrypted data. *Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS)*, Alexandria, Virginia, USA, 89-98.

[17] J. Bethencourt, A. Sahai, and B. Waters. 2007. Ciphertext-policy attribute-based encryption. *Proceedings of the 28th IEEE Symposium on Security and Privacy (SP)*, Berkeley, CA, USA. 321-334.

[18] L. Cheung, and C. Newport. 2007. Provably secure ciphertext policy ABE. *Proceedings of the 14th ACM conference on Computer and communications security (CCS)*. 456-465.

[19] L. Ibraimi, M. Asim, and M. Petkovic. 2009. Secure management of personal health records by applying attribute-based encryption. *Proceedings of the 6th International Workshop on Wearable Micro and Nano Technologies for Personalized Health (pHealth)*, Oslo. 71-74.

[20] C. Wang, X. Liu, and W. Li. 2012. Implementing a personal health record cloud platform using ciphertext-policy attribute-based encryption. *Proceedings of*

the International Conference on Intelligent Networking and Collaborative Systems (INCoS), Bucharest. 8-14.

[21] J. Li, G. Zhao, X. Chen, D. Xie, C. Rong, W. Li, and Y. Tang. 2010. Fine-grained data access control systems with user accountability in cloud computing. *Proceedings of the 2th International Conference on Cloud Computing Technology and Science (CloudCom)*, Indianapolis, IN, USA. 89-96.

[22] P. Thummavet, and S. Vasupongayya. 2013. A novel personal health record system for handling emergency situations. *Proceedings of the 17th International Computer Science and Engineering Conference (ICSEC)*, Nakorn Pathom, Thailand. 266–271.

[23] M. Li, S. Yu, Y. Zheng, K. Ren, and W. Lou. 2013. Scalable and secure sharing of personal health records in cloud computing using attribute-based encryption. *IEEE Transactions on Parallel and Distributed Systems (TPDS)* 24(1): 131-143.

[24] M. Li, S. Yu, K. Ren, and W. Lou. 2010. Securing personal health records in cloud computing: patient-centric and fine-grained data access control in multi-owner settings. *Proceedings of the 6th International Conference on Security and Privacy in Communication Networks (SecureComm)*, Singapore. 89-106.

[25] J. Huang, M. Sharaf, and C. T. Huang. 2012. A hierarchical framework for secure and scalable EHR sharing and access control in multi-cloud. *Proceedings of the 41st International Conference on Parallel Processing Workshops (ICPPW)*, Pittsburgh, PA, USA. 279-287.

[26] T. Parameswaran, S. Vanitha, and K. S. Arvind. 2013. An efficient sharing of personal health records using DABE in secure cloud environment. *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)* 2(3): 925-932.

[27] J. L. F. Alemán, I. C. Señor, P. Á. O. Lozoya, and A. Toval. 2013. Security and privacy in electronic health records: A systematic literature review. *Journal of biomedical informatics* 46(3): 541-562.

[28] H. L. McKinley. 2003. SSL and TLS: A beginners guide. SANS Institute. http://www.sans.org/reading-room/whitepapers/protocols/ssl-tls-beginners-guide-1029. (accessed September 4, 2014).

[29] D. Weerasinghe, and M. Rajarajan. 2011. Secure trust delegation for sharing patient medical records in a mobile environment. *Proceedings of the 7th International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM)*, Wuhan. 1–4.

[30] J. Li, G. Zhao, X. Chen, D. Xie, C. Rong, W. Li, L. Tang, and Y. Tang. 2010. Fine-grained data access control systems with user accountability in cloud computing. *Proceedings of the 2nd IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, Indianapolis, IN. 89-96.

[31] C. Neil. 2009. Java Native Interface (JNI). Javamex UK. http://www.javamex.com/tutorials/jni/. (accessed October 13, 2014).

[32] S. Leppert. 2012. Android memory dump analysis. Student research paper. Department of Computer Science, Friedrich-Alexander-University Erlangen-Nuremberg, Germany.

[33] National Institute of Standards and Technology. 2001. Advanced encryption standard (AES). National Institute of Standards and Technology. http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf. (accessed April 29, 2014).

[34] National Institute of Standards and Technology. 2001. Recommendation for block cipher modes of operation. National Institute of Standards and Technology. http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf. (accessed April 29, 2014).

[35] D. S. A. Elminaam, H. M. Abdual-Kader, and M. M. Hadhoud. 2010. Evaluating the performance of symmetric encryption algorithms. *International Journal of Network Security* 8(8): 213-219.

[36] S. Hirani. 2003. Energy consumption of encryption schemes in wireless devices. Doctoral dissertation, University of Pittsburgh.

[37] A. Nadeem, and M. Y. Javed. 2005. A performance comparison of data encryption algorithms. *Proceedings of the 1st IEEE international conference on Information and communication technologies (ICICT)*. 84-89.

[38] P. Chown. 2014. Advanced encryption standard (AES) ciphersuites for transport layer security (TLS). IETF. http://www.ietf.org/rfc/rfc3268.txt. (accessed May 12, 2014)

[39] The Indivo Personally Controlled Health Record, http://indivohealth.org/.

[40] Microsoft HealthVault, https://www.healthvault.com/.

[41] Google Health, http://www.google.com/intl/en_us/health/about/.

[42] PCEHR System in Australia, http://www.ehealth.gov.au/.

[43] Indivo Research and History, http://indivohealth.org/research/.

[44] D. Haas. 2011. Children's Hospital Boston. http://www.oscon.com/oscon2011/public/schedule/detail/19713. (accessed July 13, 2015).

[45] Dossia Healthcare System, http://www.dossia.org/.

[46] P. B. Pfiffner, and K. D. Mandl. 2013. An iOS framework for the Indivo X personally controlled health record. American Medical Informatics Association *Summits on Translational Science Proceedings (AMIA)*. 196-200.

[47] Ed. D. Hardt. 2012. The OAuth 2.0 authorization framework. IETF. https://tools.ietf.org/html/rfc6749. (accessed July 13, 2015)

[48] K. D. Mandl, W. W. Simons, W. C. Crawford, and J. M. Abbett. 2007. Indivo: a personally controlled health record for health information exchange and communication. *BMC Medical Informatics and Decision Making* 25(7): 1-10.

[49] B. Adida, A. Sanyal, S. Zabak, I. S. Kohane, and K. D. Mandl. 2010. Indivo X: Developing a fully substitutable personally controlled health record platform. *American Medical Informatics Association Annual Symposium Proceedings (AMIA)*. 6-10.

[50] Massachusetts Institute of Technology. Kerberos: The network authentication protocol. http://web.mit.edu/kerberos/. (accessed July 21, 2015)

[51] Simple Object Access Protocol. SOAPClient. http://soapclient.com/standards.html. (accessed July 13, 2015)

[52] N. V. Do, R. Barnhill, K. A. Heermann-Do, K. L. Salzman, and R. W. Gimbel. 2011. The military health system's personal health record pilot with Microsoft HealthVault and Google Health. *Journal of the American Medical Informatics Association (JAMIA)* 18(2): 118-124.

[53] B. Dolan. 2011. Official: Google Health shuts down because it couldn't scale adoption. http://mobihealthnews.com/11453/official-google-health-shuts-down-because-it-couldnt-scale/. (accessed July 13, 2015)

[54] A. Sunyaev, A. Kaletsch, and H. Krcmar. 2010. Comparative evaluation of Google Health API vs. Microsoft HealthVault API. *Proceedings of the 3rd International Conference on Health Informatics (HEALTHINF)*, Valencia, Spain. 195-201.

[55] A. Sunyaev, D. Chornyi, C. Mauro, and H. Krcmar. 2010. Evaluation framework for personal health records: Microsoft HealthVault vs. Google Health. *Proceeding of the 43rd Hawaii International Conference on System Sciences (HICSS)*, Hawaii, USA. 1-10.

[56] C. Pearce, and M. Bainbridge. 2014. A personally controlled electronic health record for Australia. *Journal of the American Medical Informatics Association (JAMIA)* 21(4): 707-713.

[57] A. Charland, and B. Leroux. 2011. Mobile application development: web vs. native. *Communications of the ACM* 54(5): 49-53.

[58] ASTM International. Standard specification for Continuity of Care Record (CCR). http://www.astm.org/Standards/E2369.htm/. (accessed July 17, 2015)

[59] D. Ranjan. 2013. Introduction to Continuity of Care Record (CCR). http://www.codeproject.com/Articles/564505/Introduction-to-Continuity-of-Care-Record-CCR/. (accessed July 17, 2015)

[60] HL7 International. HL7/ASTM Implementation Guide for CDA® R2 -Continuity of Care Document (CCD®) Release 1. http://www.hl7.org/implement/standards/product_brief.cfm?product_id=6/. (accessed July 17, 2015)

[61] R. Cover. 2003. Simple object access protocol (SOAP). http://xml.coverpages.org/soap.html/. (accessed July 17, 2015)

[62] Json.org. Introducing JSON. http://json.org/. (accessed July 17, 2015)

[63] N. Coffey. Getting started with JNI. http://www.javamex.com/tutorials/jni/getting_started.shtml. (accessed January 2, 2015)

[64] M. Mead. Programmming in C/C++ with the Java native interface. http://home.pacifier.com/~mmead/jni/cs510ajp/index.html. (accessed January 2, 2015)

[65] Codingfreak blog. Creating and using shared libraries in Linux. http://codingfreak.blogspot.com/2009/12/creating-and-using-shared-libraries-in.html. (accessed January 2, 2015)

[66] Codingfreak blog. Creating shared libraries in Linux - part 2. http://codingfreak.blogspot.com/2010/11/creating-shared-libraries-in-linux-part.html. (accessed January 2, 2015)

[67] Python Software Foundation. Extending Python with C or C++. https://docs.python.org/2/extending/extending.html. (accessed January 2, 2015)

[68] Python Software Foundation. ctypes — A foreign function library for Python. https://docs.python.org/2/library/ctypes.html. (accessed January 2, 2015)

Appendix A

The publications of the thesis

Appendix A1

Conference Paper

P. Thummavet, and S. Vasupongayya, “A novel personal health record system for handling emergency situations,” Proceedings of the 17th IEEE International Computer Science and Engineering Conference (ICSEC), Nakorn Pathom, Thailand, September 2013, pp. 266–271.

2013 International Computer Science and Engineering Conference (ICSEC): ICSEC 2013 English Track Full Papers

A novel personal health record system for handling emergency situations

P. Thummavet[1], S. Vasupongayya[2]
Department of Computer Engineering, Faculty of Engineering, Prince of Songkla University,
Hat Yai, Songkhla 90112, Thailand
E-mail: 5310120127@email.psu.ac.th[1], vsangsur@coe.psu.ac.th[2]

Abstract—**Personal health record (PHR) becomes a popular research topic nowadays. Many research works have proposed several concepts in managing and organizing a PHR. However, there are several uncertain issues left such as the role of a PHR in emergency situations. In this paper, a solution to handle a PHR information management in emergency situations is proposed. Because a PHR is controlled by its owner, the critical challenge in handling the PHR in emergency situations is how emergency staffs can access PHR information, even when the PHR owner is unable to give his/her consent. The proposed scheme allows each PHR to be classified into several categories. Each category presents a different restriction. And, the emergency staffs can access each category according to the policy defined by the PHR owner. The threshold cryptosystem is adapted in this work to allow the selected set of PHR-owner-delegates to grant permission to the emergency staffs when the PHR owner is unconscious.**

Keywords-personal health record; privacy; security; emergency situation; threshold cryptosystem

I. INTRODUCTION

Recently, a personal health record (PHR) becomes a hot issue in researches and adoptions. PHR presents the concept of individual health information sharing that is controlled by its owner [1]. In other words, the owner can selectively share his/her health information to selected users through a PHR system. Typical PHR information includes congenital diseases, medical history, allergy, disease risks, lab results, physicians' recommendations, exercise patterns and results. Thus, information stored in a PHR is usually considered to be highly sensitive [1], [2]. Therefore, a PHR management system must protect the data and allow only authorized users to access the data. Usually, a PHR is protected by encrypting before sharing.

There are a lot of research works regarding the security of PHR [2], [3], [4], [5], [6] and [7]. Many research works propose important and useful concepts of the PHR security. However, there are several uncertain issues still. One of those missing issues is how to manage PHR information in emergency situations. Typically, emergency units do not directly involve with the PHR owner. Thus, these personals are not in the PHR system. Therefore, there is no access right for these groups of people. Thus, the emergency staffs are not able to access a PHR of the patients even in emergency situations, even though these staffs are the first responders who will reach the patients. Therefore, allowing an emergency staff to access information stored in PHRs in emergency situations is essential. Information stored in the patient's PHR may help an emergency staff make better decisions.

In this paper, we propose a novel scheme for handling accesses to PHR information in emergency situations. A critical part of our work is how emergency staffs can access PHR information, even when an owner stays unconscious or inconvenient to give his/her consent. The proposed system allow external units (i.e., the emergency staffs) to access an individual PHR according to the PHR owner's policy. Three levels of confidentiality of PHR information including secure, restricted and exclusive levels are proposed. A PHR owner can define any of the three levels to each PHR. The secure level is defined for freely accessing by the emergency staffs without any consent in emergency situations. The restricted level is considered to be sensitive and the emergency staffs must have an access right to access it. However, the PHR owner may not be able to give his/her consent. Thus, the consent can be granted by at least t out of n trusted users, who are defined by the PHR owner. Novelty, we adopt a threshold cryptosystem [8] to originate an access granting mechanism for the restricted-level information. The threshold cryptosystem is an important key to create the access granting mechanism to emergency staffs, in which an owner can specify pre-determined threshold value (t) to the restricted-level information. Consequently, the threshold cryptosystem allows emergency staffs to access the restricted-level information only if they are granted the permission by at least t (pre-determined threshold) out of n trusted users. The exclusive level is considered to be top secret. That is, the emergency staffs cannot access it even in emergency situations. With our proposed scheme, an owner can share PHR information to emergency staffs but still be able to control privacy and confidentiality of highly sensitive information.

The proposed scheme can be implemented as an add-on to any existing PHR management system in order to support emergency staffs under emergency situations. The remaining of this document is organized as follows. Section 2 describes related work. The traditional personal health record management system is discussed in Section 3. Section 4 presents the proposed scheme. Finally, conclusions are given in Section 5.

This work is supported in part by the National Research University Funding no. MED540548S

II. Related Work

There are a few interesting PHR management systems that have features to handle emergency situations [3], [4], [9], [11], [12].

Weerasinghe and Muttukrishnan [9] proposed a PHR system that can exchange medical information in emergency situations. Medical information is protected by the database-level encryption concept [10]. Therefore, a PHR service provider can be defined to serve medical information to an emergency staff during emergency situations. In their work, a trusted granting server (third party) is used to establish the connection between two unknown parties (i.e., emergency staff and PHR service provider). Therefore, each party can assure that the peer party is genuine. The main drawback of their work is that medical information is encrypted by the database key. The database key may introduce privacy risks of PHR owners because the key can allow multiple accesses without any need to request for an authorization from the PHR owners. Furthermore, the access right is binary. That is, the whole database can be accessed, thus an emergency staff can access any information on the database once he/she has the key.

Huda, Yamada and Sonehara [11] proposed a PHR system that has strong authentication using health smart cards. A PHR owner's smart card keeps a digital pseudonym that is indexing the PHRs stored on a database. In emergency situations, after authentication, an emergency staff can decrypt the digital pseudonym and use it to retrieve the PHR information. They claim that an owner's name (field in a database) is replaced with a pseudonym in order to protect privacy of an owner. Therefore, even if records are exposed to unauthorized parities, the owner's privacy is still preserved from those unauthorized parties. However, the same problem occurs that is the access is still binary. Thus, an emergency staff can access any information even though that information is highly sensitive.

The HCPP (Healthcare system for Patient Privacy) [12] introduced a solution for handling emergency situations with backup mechanisms. The backup mechanisms work by letting an owner defines his/her emergency information. The emergency information is stored at a trusted server controlled by the government offices. In emergency situations, an emergency staff can access the emergency information without compromising the secret information. HCPP allows the owner to control which information can be accessed or cannot be accessed in emergency situations. However, HCPP does not support any information in the case when the physicians need more information other than the defined emergency information. In contrast, the proposed scheme in this work defines three levels of protections. In emergency situations, an emergency staff can access secure-level information immediately but if an emergency staff needs further information beyond the secure level. An emergency staff can request to access the restricted-level information but he must be granted the permission by at least t out of n trusted users, who are defined by the PHR owner.

The break-glass access was proposed by Li, Yu, Zheng, Ren and Lou [3], [4]. Typically, a PHR is encrypted by the key-policy attribute-based encryption (KP-ABE) [13]. KP-ABE enables an owner to specify a set of attributes embedded into an encrypted PHR file. For information selected by an owner to be emergency information, it will be encrypted with a set of desired attributes plus the attribute "emergency". Then, an owner generates the emergency key containing the attribute "emergency" as an access policy. The emergency key is delegated to an emergency department (ED). In emergency situations, an emergency staff authenticates himself to ED, then requests and obtains the emergency key for decrypting the emergency PHR files. Similar to the HCPP, the break-glass access mechanism does not satisfy an emergency staff if he needs further information other than the selected emergency information. Unlike our work, the proposed scheme can satisfy that requirement by the access granting mechanism for the restricted-level information.

III. Traditional Personal Health Record System

The most widely used PHR model [2], [5], [6], [7] due to its simplicity, is illustrating in Figure 1. The model consists of three entities: user authority (UA), PHR server and users. According to the model, UA manages all tasks concerning a user in the PHR system, such as certifying users, generating keys and certificates, authenticating users, distributing keys and certificates, and revoking users. The PHR server acts as a warehouse of the PHR in which the owners can share their PHRs to selected users securely. To protect the PHR information, a PHR must be encrypted at a client before uploading to the PHR server. Typically, a PHR contains health information related to an individual but is often accessed by multiple users, such as the owner, family members, physicians, and caregivers. Therefore, the one-to-many encryption scheme is employed by several PHR management systems [2], [3], [4], [5], [6] and [7]. The most common encryption scheme used to protect PHRs [5], [6], [7] is Ciphertext-policy attribute-based encryption (CP-ABE) [14]. According to the CP-ABE, the owner can specify an access policy and embed the policy into the encrypted PHR file. Only the user who has the CP-ABE private key satisfying the required access policy can decrypt the protected PHR file.

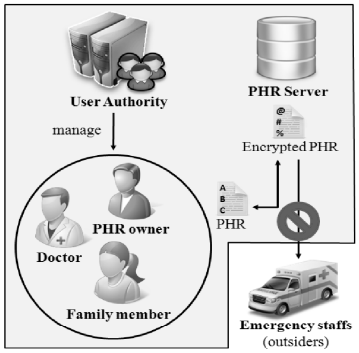

Figure 1. Traditional personal health record system.

Thus, only the users selected by an owner can access information stored in the PHRs. During emergency situations,

however, the owner may be unconscious or inconvenient to grant any access right to the emergency staffs, who are typically not in the PHR system [3], [4], [9], [11], [12]. Therefore, in this paper we will improve the PHR system to support emergency staffs in order to access the PHR information under emergency situations. The proposed scheme can improve the missing important requirement of traditional PHR systems.

IV. Proposed Scheme

In this section, the proposed improvement scheme to the referred traditional PHR system in order to support accessing information stored in PHRs during emergency situations is described. The primary concern of our work is similar to the traditional system that is the privacy and confidentiality of individual information must be protected. Therefore, the emergency staffs referred in this paper are assumed to be trustworthy. In addition, the emergency units, who will certify the emergency staffs, are assumed to be trusted by the PHR system.

In order to protect the privacy and confidentiality of the PHR information, the proposed scheme defines three levels of confidentiality of PHR information including **secure**, **restricted** and **exclusive** levels. The secure level refers to the information that must be protected during normal situations. However, the emergency staffs can access the secure-level information immediately during emergency situations. The restricted level also refers to the information that must be protected in normal situations. However, the emergency staffs can access it only if they are granted the permission by at least t out of n trusted users, who are defined by the PHR owner. The exclusive level refers to the information that cannot be accessed by emergency staffs even in emergency situations. In the proposed scheme, an owner can define the confidentiality level to his/her PHR. Thus, the PHR owner has a solution to define a fine-grained access control on his/her PHR.

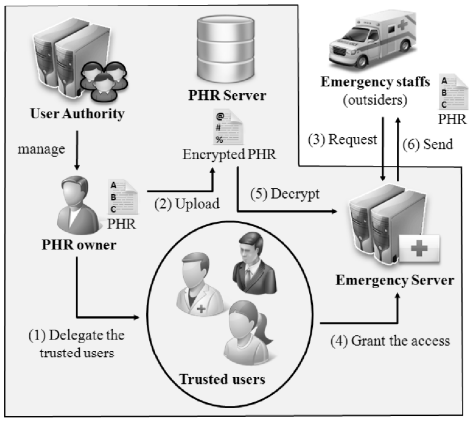

Figure 2. The proposed personal health record system.

The proposed scheme is presented in Figure 2. An emergency server (EmS) is added to the original model (Figure 1) to handle an emergency situation. EmS acts as a service provider that serves emergency staffs (the outsiders) if emergency situations occur. In other words, emergency staffs can request an access to the information stored in the PHR system through the EmS. EmS will strictly perform actions on PHRs according to their confidentiality level defined by the PHR owner. That is, emergency staffs can access information only if they have an access right on that information. Note that, EmS should be a high computational processing unit and have a large network bandwidth in order to support any disaster situation in which a huge number of requests are generated, such as during tsunami, earth quake or great flood.

A. Delegating Trusted Users

In the proposed scheme, an owner can define a set of trusted users (denoted as 1 in Figure 2) to make a decision on granting access rights on the restricted-level information on behalf of the PHR owner during emergency situations. A contact list of trusted users is securely recorded into an EmS database and will be used if emergency staffs request to access any restricted-level information stored in the PHR system. When the emergency staffs request to access the restricted-level information (denoted as 3 in Figure 2), EmS will broadcast the request message to all selected trusted users. If any trusted user approves the request, he/she sends an approval message to EmS (denoted as 4 in Figure 2). If approval messages collected by the EmS are more than or equal to the pre-determined threshold (t) defined by the PHR owner, the EmS can decrypt the protected PHR (denoted as 5 in Figure 2) and the emergency staffs can access the information (denoted as 6 in Figure 2).

B. Preparing PHRs for Emergency Situations

Since information stored in a PHR is considered to be sensitive, a PHR will be encrypted by the CP-ABE at a client before uploading to the PHR server (denoted as 2 in Figure 2). During encryption, the PHR owner can define a confidentiality level (i.e., secure, restricted or exclusive level) to the particular PHR. The following sections describe how the proposed scheme prepares the PHR system for emergency situations.

1) Secure-level information: According to the definition, the information stored in a PHR defined at this level must be protected during normal situations. However, the emergency staffs can access secure-level PHR information immediately. That is, the EmS can decrypt a secure-level PHR instantly and send the decrypted information to the emergency staffs during emergency situations. In other words, a secure-level PHR must be decrypted by the EmS key, namely the emergency key. To do so, an access policy specified by the owner must be modified to support the decryption using the EmS key. The sequence of transactions is shown below.

a) An owner specifies the access policy for encrypting his/her PHR.

b) The PHR client module adds the identity attribute of the EmS key to the access policy.

c) The PHR client module encrypts the PHR using the CP-ABE with the access policy as a parameter.

d) The PHR client module uploads the resulting encrypted PHR to the PHR server through a secure channel.

e) The PHR client module inserts a metadata for this PHR into the EmS database and sets the PHR to the secure-level information.

2) Restricted-level information: According to the definition, the information stored in a PHR defined at this level must be protected during normal situations. The emergency staffs can access the restricted-level PHR information if and only if they are granted the permission by at least t out of n trusted users, pre-selected by the PHR owner. That is, when the emergency staffs request to access a PHR, the EmS will broadcast the request message to all trusted users pre-defined by the PHR owner. If there are at least t approvals, the EmS can derypt the PHR and send the decrypted information to the emergency staffs. The pre-determined threshold (t) is also pre-defined by the PHR owner during the PHR encryption. Thus, the threshold cryptosystem idea is adapted in this work in order to handle such situations.

Typically, a PHR is encrypted using the CP-ABE before uploading to the PHR server. However, the access policy, which is embedded into the resulting encrypted PHR file, must be modified. By adding the identity attribute of the unique emergency key, which is generated to be a key for this particular PHR, the resulting encrypted PHR file can be decrypted by the unique emergency key as well. However, this unique emergency key is also encrypted using the threshold cryptosystem. Once the PHR owner selects a set of trusted users, the PHR system will assign a secret key for each of these trusted users. The secret key is a random string. Each secret key is encrypted with the associated trusted user's public key. Thus, only the pre-defined users can decrypt the secret keys. The encrypted unique emergency key and the encrypted secret key of each trusted user are then uploaded to the EmS database. The encrypted restricted-level PHR is however uploaded to the PHR server. The sequence of transactions is shown below.

a) An owner specifies the access policy for encrypting his/her PHR.

b) The PHR client module contacts the user authority (UA) in order to generate the secret keys and the unique emergency key.

c) The UA encrypts the unique emergency key by the threshold cryptosystem with a list of secret keys and the pre-determined threshold as parameters. Then, the UA maps each secret key to each trusted user and encrypts each secret key with the corresponding trusted user's public key.

d) The UA sends the encrypted unique emergency key and the encrypted secret keys to the PHR client module through a secure channel.

e) The PHR client module adds the identity attribute of the unique emergency key to the access policy.

f) The PHR client module encrypts the PHR using the CP-ABE with the access policy as a parameter.

g) The PHR client module uploads the resulting encrypted PHR to the PHR server through a secure channel.

h) The PHR client module uploads the encrypted unique emergency key and a set of encrypted secret keys to the EmS database through a secure channel.

i) The PHR client module inserts a metadata for this PHR into the EmS database and sets the PHR to the restricted-level information.

3) Exclusive-level information: According to the definition, the information stored in a PHR defined at this level is not accessible by emergency staffs even in emergency situations. Therefore, the EmS cannot decrypt this kind of PHR. Thus, the PHR client module will encrypt a PHR without the need to add any identity attribute of the emergency keys. The sequence of transactions is shown below.

a) An owner specifies the access policy for encrypting his/her PHR.

b) The PHR client module encrypts the PHR using the CP-ABE with the access policy as a parameter.

c) The PHR client module uploads the resulting encrypted PHR to the PHR server through a secure channel.

C. Accessing PHR Information by Emergency Staffs

Under the proposed scheme, the emergency staffs can access information stored in the PHR system through the EmS. Two levels of PHR confidentiality are defined for emergency staffs including secure and restricted levels. If the secure-level information is requested, the emergency staffs can access it instantly. However, if the restricted-level information is requested, the emergency staffs must be granted an access by a set of trusted users pre-selected by the PHR owner. At least t out of n pre-selected trusted users must approve the access right.

The tasks of authenticating the emergency staffs, however, are left for the emergency units that are trusted by the PHR system. Typically, an emergency unit issues some identity information (such as certificate or token) to its emergency staffs. When an emergency staff requests to access PHR information, the EmS will verify the requestor by using such information. Thus, the proposed system does not restrict any authentication protocol for verifying the users. That is, the authentication protocol of the proposed scheme can be applied with any protocol, such as the secure socket layer protocol [15], the challenge and response protocol [11], or the token-based protocol [9]. In addition, the EmS should have a transaction log that records all transactions. This way, the PHR owners can trace all accesses to their PHR information. The following paragraphs describe how emergency staffs can access the PHR information during emergency situations.

1) Secure-level information: To access the secure-level information stored in a requested PHR, an emergency staff sends a request message to the EmS (denoted as 3 in Figure 2).

Then, the EmS and an emergency staff authentication process occurs. Once, the authentication process succeeds, the EmS downloads the requested encrypted PHR from the PHR server. Then, the EmS decrypts the encrypted PHR with its emergency key (denoted as 5 in Figure 2). Finally, the EmS sends the requested PHR information to the emergency staff through a secure channel (denoted as 6 in Figure 2). More concretely, Figure 3 illustrates the sequence of actions occuring during the secure-level information accessing process.

2) Restricted-level information: To access the restricted-level information stored in a requested PHR, an emergency staff sends a request message to the EmS (denoted as 3 in Figure 2). Then, the authentication process between the EmS and the emergency staff occurs. Once the authentication succeeds, the EmS broadcasts the request message along with the encrypted secret key to each corresponding pre-selected trusted user. If a trusted user approves the request, he/she decrypts the encrypted secret key with his/her private key and sends the secret key to the EmS through a secure channel (denoted as 4 in Figure 2). If the number of approvals is more than or equal to the pre-determined threshold (variable t), the EmS can decrypt the encrypted unique emergency key, which is encrypted using the threshold cryptosystem. Then, the EmS downloads the requested encrypted PHR from the PHR server and uses the unique emergency key to decrypt the encrypted PHR (denoted as 5 in Figure 2). Finally, the requested PHR information is sent to the emergency staff through a secure channel (denoted as 6 in Figure 2). More concretely, the process of accessing the restricted-level information is shown in Figure 4.

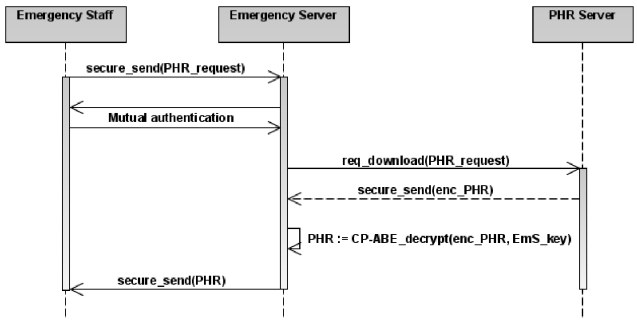

Figure 3. The secure-level PHR information accessing sequence.

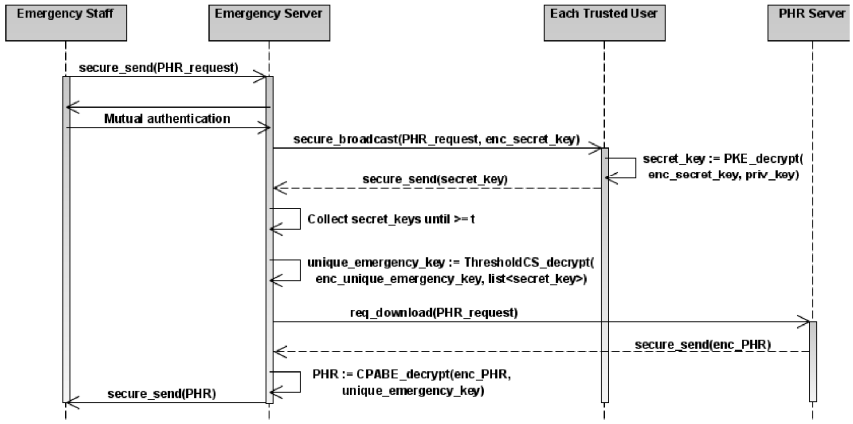

Figure 4. The restricted-level PHR information accessing sequence.

V. CONCLUSIONS

A scheme for managing the PHR information during emergency situations is proposed in this paper. Because the PHR information is usually sensitive, the critical challenge of this work is to manage the PHR access for emergency staffs when the PHR owner is not able to grant his/her consent. Under the proposed scheme, the PHR owner is allowed to define a fine grain access control over his/her PHR information during emergency situations. Three levels of confidentiality including secure, restricted and exclusive are defined. The PHR owner can define one of the three levels to his/her PHR information. The secure-level PHR information is considered open to emergency staffs during the emergency situations. Meanwhile, the restricted-level PHR information can be opened to emergency staffs if and only if there are enough approvals from the pre-selected trusted users. The exclusive-level PHR information, however, is not opened to the emergency staffs.

Thus, the secure-level and restricted-level information enable the owner to specify fine-grained access control on his/her information to emergency staffs. Furthermore, if the owner does not agree to give any access right on any of his/her PHR information, he/she can define the information as exclusive-level PHR information. All access controls are achieved through the use of an additional emergency management server. The emergency management server acts as a connection between the emergency unit staffs and the traditional PHR management system. The emergency units must have a method to authenticate their emergency staffs. Thus, the tasks of authenticating the emergency staffs are done according to the method used by the emergency units. Furthermore, the emergency units in this work are assumed to be trustworthy.

The contribution of this work is to enable the emergency staffs to gain an access to the PHR management system according to the policy defined by the PHR owner. Under the proposed system, the PHR owner can specify a fine grain access control policy during emergency situations. To the best of our knowledge, existing methods still do not allow fine grain access control like what has been proposed in this paper.

REFERENCES

[1] P. C. Tang, J. S. Ash, D. W. Bates, J. M. Overhage, and D. Z. Sands, "Personal health records: definitions, benefits, and strategies for overcoming barriers to adoption," Journal of the American Medical Informatics Association, JAMIA2006., in press.

[2] K. Garson, and C. Adams, "Security and privacy system architecture for an e-hospital environment," Identity and trust on the Internet, IDtrust2008., in press.

[3] M. Li, S. Yu, K. Ren, and W. Lou, "Securing personal health records in cloud computing: patient-centric and fine-grained data access control in multi-owner settings," Security and Privacy in Communication Networks, SecureComm2010., in press.

[4] 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," Parallel and Distributed Systems, PDS2013., in press.

[5] J. Huang, M. Sharaf, and C. T. Huang, "A hierarchical framework for secure and scalable EHR sharing and access control in multi-cloud," Parallel Processing Workshops, ICPPW2012., in press.

[6] L. Ibraimi, M. Asim, and M. Petkovic, "Secure management of personal health records by applying attribute-based encryption," Wearable Micro and Nano Technologies for Personalized Health, pHealth2009., in press.

[7] C. Wang, X. Liu, and W. Li, "Implementing a personal health record cloud platform using ciphertext-policy attribute-based encryption," Intelligent Networking and Collaborative Systems, INCoS2012., in press.

[8] T. P. Pedersen, "A threshold cryptosystem without a trusted party," Advances in Cryptology. EUROCRYPT1991., in press.

[9] D. Weerasinghe, and M. Rajarajan, "Secure trust delegation for sharing patient medical records in a mobile environment," Wireless Communications, Networking and Mobile Computing, WiCOM2011., in press.

[10] Y. Ding, and K. Klein, "Model-driven application-level encryption for the privacy of e-health data," Availability, Reliability, and Security, ARES2010., in press.

[11] M. N. Huda, S. Yamada, and N. Sonehara, "Privacy-aware access to patient-controlled personal health records in emergency situations," Pervasive Computing Technologies for Healthcare, PervasiveHealth2009., in press.

[12] J. Sun, X. Zhu, C. Zhang, and Y. Fang, "HCPP: Cryptography based secure EHR system for patient privacy and emergency healthcare," Distributed Computing Systems, ICDCS2011., in press.

[13] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," Computer and Communications Security, CCS2006., in press.

[14] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," Security and Privacy, SP2007., in press.

[15] J. Viega, M. Messier, and P. Chandra, Network Security with OpenSSL: Cryptography for Secure Communications, O'Reilly Media, 2002, pp.93-142.

Appendix A2

Journal Paper

P. Thummavet, and S. Vasupongayya, “A simple ciphertext-policy attribute-based encryption extension to provide multi-authority personal health record systems,” Journal of Biomedical and Health Informatics (J-BHI), X(X), 2015, pp. X-X. (under the reviewing process)

A Simple Ciphertext-Policy Attribute-based Encryption Extension to Provide Multi-Authority Personal Health Record Systems

Phuwanai Thummavet and Sangsuree Vasupongayya

***Abstract*—A PHR concept enables an individual to create, manage, and share his/her health information to a group of selected people through an online PHR system. The majority of current PHR systems use a single user authority to manage all participants. However, a PHR system usually involves several user domains. Thus, a simple multi-authority secure personal health record scheme is proposed in this work. The proposed scheme allows an individual to create, verify, and manage a group of selected PHR users locally. The main idea is to distribute the key pair to all authorities for generating the user CP-ABE private keys. Thus, the PHR user of some authority can share his/her encrypted PHR to a user of other authority transparently as if they are in the same single global authority. The hierarchical trust model is applied to provide a method to verify a user from other authority. SSL protocol is also used for establishing a secure end-to-end communication channel for the proposed scheme.**

***Index Terms*—ciphertext-policy attribute-based encryption, multi-user domain, personal health record, privacy.**

I. Introduction

PREVENTIVE healthcare is an alternative healthcare idea that emphasizes on preventing the disease instead of curing the disease afterward. However, the majority of today healthcare styles are curative healthcare at a severe stage of the symptom because the disease typically cannot be easily detected at the early stage [1]. As a result, the late detection of a disease can increase the healthcare cost. Moreover, the late detection of a disease may cause a difficulty to cure a disease, or may be too late for a dangerous disease such as heart disease, stroke, cancer, chronic respiratory diseases, and diabetes [2]. A preventive healthcare, on the other hand, emphasizes on monitoring a person condition, detecting a difference or symptom, and avoiding a disease. For example, an elder can capture his/her physiological responses from some wearable sensor devices [3]. The monitored data can be of various types depending on the elder's interest such as body temperature, blood pressure, electrocardiogram (ECG) signal, dietary details, exercise patterns, sleeping patterns, and transient symptoms. This information can be used later by the elder's physician. This way, any early symptom or threat of a disease can be detected early. However, the monitored data must be stored and shared with the physician.

A concept of personal health record (PHR) [4] enables an individual to create, manage, and share his/her health information to a group of selected people through an online PHR management system. In addition to the home-monitored data, PHRs can contain other health related information such as medical history, personal diseases, past allergic reactions, laboratory results, mental health information, physician diagnostic results and recommendations, Therefore, a PHR is usually considered to be highly sensitive which raises a concern on the security of such data [5],[6,],[7],[8]. Typical PHR system requires an encryption method to provide security by encrypting the data at the source before transferring to a PHR storage system.

However, only encryption is not enough to manage a PHR system because the selected encryption method must also allow multiple people to access the same encrypted data since the PHR information is often accessed by multiple potential users. A list of potential users includes the PHR owner, his/her family members, the medical staffs, and the caregivers. Therefore, an encryption method must also enforce an access control of each user on the defined PHR. As a result, the one-to-many encryption schemes such as Policy-based encryption (PolBE) [10], Key-policy attribute-based encryption (KP-ABE) [11], and Ciphertext-policy attribute-based encryption (CP-ABE) [12] are usually applied in order to enforce an access control on the PHR.

The one-to-many encryption scheme allows the PHR owner to define a specific access policy on his/her PHR. Then, the PHR will be encrypted such that only the user with a specific set of attributes defined in the policy can access the encrypted PHR. This way, the owner has a full control on who can access his/her PHR information. For example, a PHR owner can define a policy to allow his/her family member, the physician at Hospital A and the physician at Hospital B to access his/her PHR information. Thus, the person who has the attribute defined in the policy can access the PHR information. In other words, only the user who has the attribute defined in the policy (i.e., a family member, a physician at Hospital A and a physician at Hospital B) will be able to decrypt the

This work was supported in part by the National Research University Funding no. MED540548S. The authors thank Centre for Network Research (CNR) lab at Prince of Songkla University for providing equipments and facilities.

The authors are with the Department of Computer Engineering, Faculty of Engineering Prince of Songkla University, Hat Yai, Songkhla 90112, Thailand (e-mail: vsangsur@fivedots.coe.psu.ac.th).

encrypted PHR. This way, multiple users can access the same information as long as they process the attribute(s) satisfied the access policy of the encrypted PHR. The one-to-many encryption schemes—PolBE, KP-ABE, and CP-ABE—are applied in many traditional PHR systems. PolBE was used in [5]. KP-ABE was used in [13][15]. CP-ABE was used in [16][18].

A limitation to these PHR systems are the requirement of a single user authority or a centralized user control system, in which all users must be pre-registered with the single user authority in the system. In the real environment, however, the PHR may be involved several user domains [20] such as a personal domain (e.g., patient/PHR owner and family members), a healthcare domain (e.g., medical staffs), a health insurance domain (e.g., insurers), and a medical research domain (e.g., researchers). Each of these domains may already establish an authentication and verification method. However, the traditional one-to-many encryption schemes may not be able to utilize such features. Moreover, handling multiple user domains on a centralized user authority environment can create several issues and limitations [13], [20] such as the system scalability, a single point of failure, and a trust management problem.

To solve such issues and limitations, [13] and [14] proposed a multi-user domain PHR system, which is divided into two security domains: personal domain and public domain. The personal domain is managed by the PHR owner directly. The PHR owner can add users such as his/her family members and friends to the personal domain. A user in the personal domain can access the PHRs according to the access permissions given by the PHR owner. On the other hand, the public domain consists of multiple types of PHR users such as physicians, nurses, and paramedics. A user in the public domain can access the PHRs according to the access permissions defined by his/her roles. In their work, the KP-ABE and multi-authority attribute-based encryption (MA-ABE) [22] were applied to protect the PHR information in the personal domain and the public domain respectively. With the MA-ABE, their system can distribute the user verification and key management tasks to multiple expert authorities. However, there are only two types of domains. In addition, [23] proposed a multi-authority cloud storage system which applied CP-ABE for multi-authority ABE where the tasks of user and attribute management are distributed to multiple attribute authorities.

In this paper, a simple multi-authority secure personal health record (MA-PHR) scheme is proposed to allow an individual to create, verify, and manage a group of selected PHR users locally. The idea is to modify the CP-ABE initial setting to handle a multi-authority environment. This way, any existing single-authority personal health record system can utilize the proposed method in order to handle multi-authority environment. Furthermore, a dual layer protection is also proposed to enable the user to control which users can download, upload or even delete his/her (encrypted) PHR stored on the PHR server. The end-to-end secure PHR sharing scheme is also ensured in order to guarantee that only the authorized users is able to access the PHR information. With the proposed scheme, the traditional centralized user authority PHR systems can be easily transformed to multi-user authority PHR systems.

The remaining of this document is organized as follows. The original CP-ABE scheme is described in Section II. The proposed MA-PHR scheme is presented in Section III. Section IV provides related works. Finally, the conclusion is given in Section V.

II. Ciphertext-Policy Attribute-Based Encryption

The original CP-ABE scheme [12] is presented in this section in order to layout the background information required for understanding the proposed scheme presented in the next section. CP-ABE is a one-to-many encryption algorithm that allows multiple users to access the encrypted data. Under CP-ABE scheme, a set of attributes is used to describe the user's decryption key (called the CP-ABE private key). The data owner can specify an access policy using a list of attributes associated with the authorized person. The policy is then embedded into the encrypted data. For example, the policy in the previous example can be expressed as follow.

Policy P = "(family_member) OR (physician, hospital-A) OR (physician, hospital-B)"

This way, the encrypted data can be decrypted if the user processes the defined attributes. There is no method to make a decision of who can access the data because the data can be decrypted if the user processes the attributes family_member or a physician at either hospital-A or hospital-B. Otherwise, the data cannot be decrypted. The CP-ABE scheme consists of four steps as follows.

The first step is the setup phase. The setup phase generates a public parameter *PK* and a master secret key *MSK*. The *PK* consists of the generator g, g^{β}, and $e(g,g)^{\alpha}$, where e is a efficiently computable symmetric bilinear map. The *MSK* is the value β and g^{α}. The *PK* can be reveal publicly, while the *MSK* must be kept secret.

The second step is the encryption phase. The encryption phase generates the ciphertext *CT* from a set of input including the public parameters *PK*, a plaintext message *M*, and an access policy tree *T*. The access policy tree can be generated for any policy using a set of boolean formulas. For example the policy P above can be transformed into an access policy tree as shown in Figure 1.

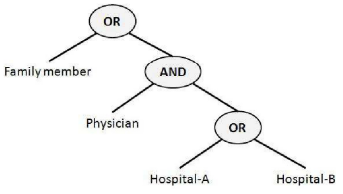

Fig. 1. An access policy tree of the policy P.

The third step is the keygen phase. The keygen phase generates the private key *SK* associated with the set of attribute *S* that described the key. This process takes in the public parameters *PK*, the master secret key *MSK*, and a set of user attributes *S* as a set of input. That is, the attributes are mathematically incorporated into the key.

The last step is the decryption phase. At this phase, the ciphertext *CT* will be decrypted if and only if the set of attributes associated with the private key *SK* satisfies the access policy tree *T*.

Typically, CP-ABE is designed for a single user authority (UA) environment. In other words, all users must be managed by the centralized UA. In this work, a specific initial setting of CP-ABE in order to handle the multi-UA environment is proposed. By using our proposed scheme, any traditional CP-ABE-based PHR system can be modified to handle the multi-UA environment.

III. The Proposed Scheme

The proposed scheme extends the original CP-ABE to handle a multi-user authority environment by modifying an initial setting of the original CP-ABE. Under the proposed scheme, each user authority or domain is allowed to manage its members locally. However, a method to verify a user from other authority is needed. To solve such problem, the hierarchical trust model [24] which is a feature provided by the SSL protocol [25] is applied in this work to provide this mechanism. Furthermore, the SSL protocol is also used for establishing a secure end-to-end channel for secure communication in the proposed scheme. In addition, the PHR dual layer protection is also proposed in this scheme in order to provide a security on READ and STORE actions separately.

The details of all mechanisms proposed are given in the next two sections. The last section presents the resulting prototype MA-PHR developed in this work.

A. Modified CP-ABE Initial Settings

Traditionally, CP-ABE setup step is performed during the system installation process. The setup phase generates the key pair: public parameters *PK* and master secret key *MSK*. This key pair is mathematically linked. Whereas *PK* can be reveal publicly, *MSK* must be privately stored by the trusted UA. *PK* is required as the explicit parameter for generating the CP-ABE private key and encrypting/decrypting the ciphertext *CT*. *MSK* is required as the explicit parameter when generating any private key. The ciphertext *CT* can be decrypted with the private key if and only if both the ciphertext and private key are produced from the corresponding *PK* and *MSK* key pair.

The following subsections describe how to set up the original CP-ABE for a multi-authority environment. The main idea is to distribute the *PK* and *MSK* key pair to all authorities and let them generate the CP-ABE private keys for their users by using the corresponding key pair, a PHR user of some authority can share his/her encrypted PHR to another user of some authority transparently, as if they were in the same global authority. The setup transactions consist of four phases as follows:

1) MA-PHR Core System Setup

This phase is used for setting up the MA-PHR core system that is controlled by the root authority (RA) as shown in Fig. 2. The RA runs CP-ABE Setup algorithm to generate the key pair: public parameters *PK* and master secret key *MSK* (denoted as 1 in Fig. 2). Then, this key pair is privately stored by RA and will be used as the root of all children CP-ABE private keys in the system.

2) User Authority Setup

When a new user authority (UA) requests to join the MA-PHR core system, the RA must first verify the new UA. After a successful verification process, the RA is then securely distributing the *PK* and *MSK* key pair to the UA (denoted as 2 in Fig.). This key pair will be used as the specific parameters for generating the CP-ABE private keys in the user key generation phase. While *PK* can be reveal publicly, *MSK* must be privately stored by trusted UA. Thus, the *MSK* must be distributed securely to the UA.

3) User Key Generation

When a user at a certain UA is requesting for his/her private key, the UA executes the CP-ABE KeyGen algorithm using the obtained *PK* and *MSK* key pair as the specific parameters together with a set of user attributes (denoted as 3 in Fig. 2). This way, the user will receive the CP-ABE private key that is able to decrypt the encrypted data from any authority in the system as if they were in the same single global authority.

4) Inter-Authority Synchronization

Since each user authority (UA) manages a different user domain, thus, each UA can have a different set of attributes. For example, a personal authority has a set of attributes: PHR owner, family member, relative and friend, whereas a healthcare authority has a set of attributes: physician, nurse, paramedic and other medical staff attributes. Therefore, all UAs must synchronize their attribute sets with each other periodically (denoted as 4 in Fig. 2). This way, the PHR user of some authority will be able to define a set of attributes in his/her access policy for a group of users from other authorities during an encryption process.

B. Security Mechanisms of the Proposed Scheme

This section provides the details of all security mechanisms that make the proposed scheme secure under the multi-authority environment. First, the hierarchical trust model to build a mutual trust relationship among PHR users is given. Next, the PHR dual layer protection is offered to make a fine-grained access control on the PHR. Finally, the end-to-end secure PHR sharing scheme is also provided.

1) Hierarchical Trust Model

The proposed scheme allows each user authority to create, verify, and manage a group of selected PHR users locally. Basically, a PHR user can collaborate with other users across the authority. For example, a PHR owner namely Alice wants to share her PHR record with Matt, her boyfriend, who is in Alice's personal authority and Bob, her physician, who is in the healthcare authority. Because Bob and Matt are from different authority, an inter-authority user verification mechanism is required.

In the proposed scheme, the hierarchical trust model (as shown in Fig. 3), which is a feature provided by the SSL

protocol to provide the inter-authority user verification mechanism, is applied. The hierarchical trust model allows each user authority to certify and issue a SSL certificate to its members locally. For example, a personal authority and a healthcare authority can certify and issue a SSL certificate to their users locally and then a PHR user under a personal authority can verify an identity of a peer user from a healthcare authority using the peer user's certificate, and vice versa.

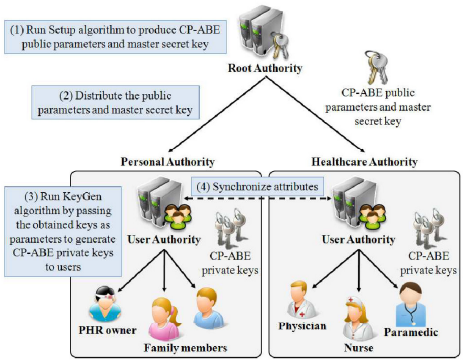

Fig. 2. Setting up CP-ABE for multi-authority environment.

The hierarchical trust model consists of two levels: authority and user levels. Each level is allowed to verify and certify its sub-entities. The root of the hierarchical trust model is called the root authority (RA). When a new user authority is created and requested to join the proposed MA-PHR system, the RA must verify an identity of the requesting authority and issue the authority certificate to the requesting authority (denoted as '*level 1*' in Fig. 3). Then, the requesting authority uses the obtained authority certificate for chain certifying and issuing the user certificates to its PHR users locally (denoted as '*level 2*' in Fig. 3). Later, a PHR user can use the obtained user certificate for authenticating and establishing a secure channel [21], [26]. Since each authority generates the user certificates to its members using the same root certificate, a user can verify the certificate of a peer user across authorities. Therefore, in case of Alice discussed previously, she can make sure that she shares her PHR information with her physician via this hierarchical trust model, because Bob will be verified and authenticated by the hospital (i.e., the healthcare authority).

2) PHR Dual Layer Protection

The proposed PHR dual layer protection consists of two protection layers: read protection layer and store access control layer. The read protection layer protects the PHR information from unauthorized read actions. As such, the PHR information will be encrypted at a source before securely uploaded to a PHR server. The read protection layer uses the CP-ABE to encrypt the PHR information at a client before securely upload to the PHR server through a secure channel (denoted as 4 in Fig. 4). According to the CP-ABE scheme, the PHR owner is allowed to specify an access policy for each PHR and the access policy will be embedded into the encrypted PHR. The user can decrypt the encrypted PHR if and only if his/her CP-ABE private key contains the set of attributes satisfying the associated access policy embedded in the encrypted PHR.

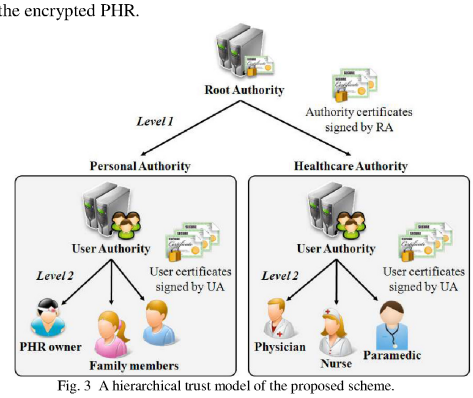

Fig. 3 A hierarchical trust model of the proposed scheme.

On the other hand, the store access control layer protects the encrypted PHRs stored on the PHR server from unauthorized accesses. The PHR owner can selectively assign any permission to each user. The permission can be upload, download and delete. The user can perform any action on the encrypted PHRs stored on a PHR server according to the access permissions granted by the PHR owner. With the proposed dual protection layers, thus, a PHR owner can have a full control over which users can read or modify his/her PHRs.

3) End-to-End Secure PHR Sharing

When a user enters the MA-PHE system, the UA that he/she is a member will allow the user to obtain his/her CP-ABE private key and the user certificate (denoted as 1 in Fig. 4). The CP-ABE private key will be used as the PHR decryption key and the user certificate will be used for authenticating himself/herself and establishing a secure channel when the user contacts with any server. In the proposed scheme, the PHR owner (e.g., Alice) can selectively grant any of the access permissions (i.e., upload, download, and delete permissions) to a selected user through the UA (denoted as 2 in Fig 4). This way, Alice can control accesses on her encrypted PHRs stored on the PHR server for each selected user. The granted access permissions will be securely recorded in the UA database and will be used when a user requests an access to any record of Alice's PHRs. If Alice and the user (e.g., Bob) are in different authorities as shown in Fig 4, the granted access permissions have to be synchronized from Alice's authority to Bob's authority (denoted as 3 in Fig 4). In the proposed scheme, all authorities always periodically synchronize their attribute sets, user lists, and granted access permission sets with each other.

The proposed scheme offers an end-to-end PHR protection method. That is, the PHR information will be encrypted using

the modified CP-ABE at the client before securely uploading to the PHR server through a secure channel (denoted as 4 in Fig 4) and the resulting encrypted PHR would be decrypted by only the authorized users at their clients. Thus, Alice can make sure that her PHR information will be protected and her information will only be exposed to the authorized users. The PHR server storing the encrypted PHR also cannot learn the plaintext information, because it does not have the decryption key [5], [13], [14], [16].

Assuming that Bob needs to access Alice's PHR information after the PHR uploading process at Alice's client side, Bob will have to request and obtain a one-time request (OTR) token from his UA (denoted as 5 in Fig. 4). The OTR token contains Bob's access permissions granted by Alice and this token will be used as a request certificate for accessing Alice's encrypted PHRs stored on the PHR server. Thus, Bob can only perform actions indicated on this OTR token. Since the OTR token is a certificate with a specific expiration date/time, Bob cannot re-use the token if the token lifetime is expired. After Bob gets the OTR token, Bob sends a request message along with the OTR token to the PHR server (denoted as 6 in Fig 4). The PHR server then uses the OTR token for verifying Bob's request. Next, Bob can download, upload or delete the requested PHR (denoted as 7 in Fig 4). In addition to provide a non-repudiation feature [21], Bob's request will be recorded as a transaction log on the audit server (denoted as 8 in Fig 4). The transaction logs will be periodically synchronized from Bob's authority to Alice's authority (denoted as 9 in Fig 4). This way, Alice can trace all accesses on her PHRs later (denoted as 10 in Fig 4).

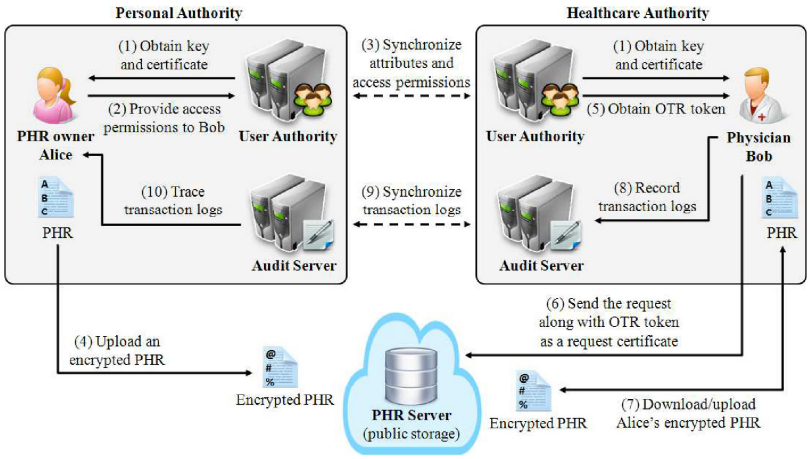

Fig. 4 End-to-end secure PHR sharing workflow.

C. *The Proposed MA-PHR System Demonstration*

The proposed MA-PHR system is developed and demonstrated in this section. The case scenario used in this section consists of four players including Alice—the PHR owner, John—Alice's family member, James—Alice's family member, and Bob—Alice's physician. Alice is a subscriber of the proposed MA-PHR system. John and James are in Alice's personal authority which is created and managed by Alice herself. Bob is a physician at a hospital which is part of the healthcare authority in the proposed system. Alice can define a list of access permissions on her PHRs to anyone in the system as shown in Fig 5. According to the access permission defined in the figure, Alice allows her family members to only download her PHRs meaning a read permission. However, Alice allows her physician, Bob, to be able to download and upload her PHRs meaning both read and store permissions. Therefore, Bob can update Alice's PHRs by himself. Note that, each player will be identified by two identities. First is the domain such as Personal or Healthcare. Second is the person such as Bob, John and James. This is part of the modification made in this project to be able to identify an authority under a multi-user authority environment in the access policy.

Next, Alice loads her PHR information containing her medical history, diagnosis information, and lab results on her PHR client module. Alice can specify an access policy for this record before the record is being encrypted as shown in Fig 6. Next, the record will be encrypted at a client using the CP-ABE scheme with the access policy as a parameter and the resulting encrypted PHR will be uploaded to the PHR server through a secure channel as shown in Fig 7.

Assuming that Bob wants to access Alice's PHR, Bob must log in to the system with his identity specified by the healthcare authority. Once Bob is authenticated and in the system, Bob can search for Alice's PHR and request to download the record as shown in Fig. 8. Then, the decryption process will be performed at Bob's client if and only if his CP-ABE private key satisfies the associated access policy embedded in the encrypted PHR as shown in Fig 9. The

process of decryption is either success or fail at this stage because of the CP-ABE scheme. There is no extra step to make a decision whether the action is allowed or disallowed.

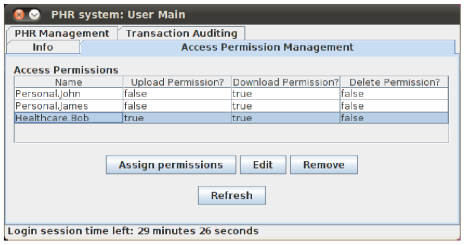

Fig. 5. Alice assigns access permissions to each player.

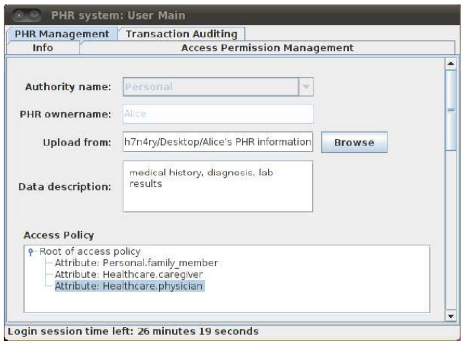

Fig 6. Alice specifies an access policy for her PHR.

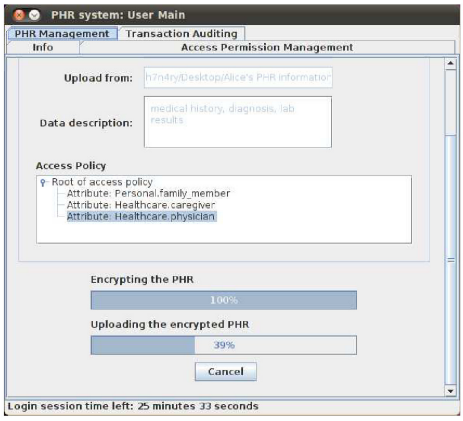

Fig 7. Alice's encrypted PHR is uploaded to the PHR server.

With the developed PHR system, users from different user authorities can collaborate with each other transparently. From the user's viewpoint, therefore, users can share their PHRs with each other with the same experience like using the traditional centralized PHR system. From the PHR administrator's viewpoint, the developed system allows the administrator to identify, verify, and manage a group of selected users related to his/her individual or organization locally. However, the proposed MA-PHR system allows the users from various authorities to participate in the system without an need to re-register with the system if they are a member of an authority in the system. This way, various organizations with well establishing verification and authentication methods can be included in the system.

However, the drawback of the proposed system is that the trusts are relaying on the security mechanisms of these various organizations. From the personal healthcare standpoint, most healthcare authorities or health service providers such as hospitals, emergency units, and rescuer units, are well established in the verification and authentication their personals. The only concern will be any unconventional unit that is requesting to join the system. Thus, the administrator of the system must pay close attention on the verification such units before adding such authority to the proposed system.

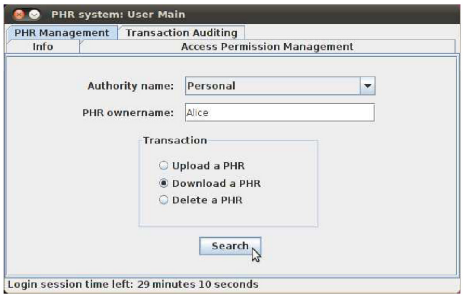

Fig 8 Bob initializes his download request.

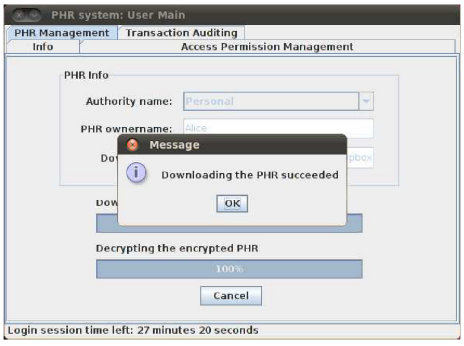

Fig 9 Bob can access Alice's PHR as requested.

IV. Related Work

Other than e-Healthcare system domain like in this work, the multi-authority concept was applied in many practical adoptions/research domains such as emergency situation process modeling [27], traffic engineering [28], and anonymous authentication [29].

In [27], a concept of multi-authority was applied for managing multiple emergency response units such as rescue, healthcare, and police units. The plan for emergency situations was presented in order to prepare each authority for react in case of a particular emergency situation efficiently.

In traffic engineering [28], an integrated network composing of TV camera sensors, databases, and physical networks is applied for intelligent transportation system. There can be multiple authorities in the traffic way such as drivers of vehicles, traffic polices, and ambulances. Each authority can request for monitoring a traffic status in real time, but there are limited resources such as network bandwidth and processing units. In order for efficient response, when clients compete for a resource, the client who has a higher priority gets the resource.

The multi-authority concept was also applied for making a digital signature scheme. In [29], a multi-authority attribute-based signature scheme was proposed. A signer can sign a message with his/her attributes, and the verifier can check whether the signer owns attributes satisfying his/her policy. A signer can be a member of multiple attribute authorities. Thus, this signature scheme is suitable for the applications in the real world because often a user can affiliate to multiple organizations. This signature scheme can be applied for an anonymous authentication or attribute-based messaging systems.

V. Conclusion

A scheme for distributing the user management tasks of a PHR system is proposed in this paper. The proposed multi-authority secure PHR scheme allows each user domain to organize and manage a group of selected PHR users locally. This way, the well-establish organization can join the system easily without any need to re-register with the PHR system. The original CP-ABE is modified in order to handle multi-user authority environment. The key idea is to distribute the initialized CP-ABE parameters (i.e., the key pair: public parameters *PK* and master secret key *MSK*) to all local UAs in the system. With the corresponding *PK* and *MSK* key pair, the local UAs can establish the multi-UA-compatible CP-ABE environment. All UAs must synchronize the attribute sets with each other periodically. A hierarchical trust model is applied to make an inter-authority user verification mechanism. PHR dual layer protection which consists of two protection layers: read protection layer and storage access control layer is offered to make a fine-grained access control on the PHR. Moreover, the end-to-end secure PHR sharing scheme is also provided.

The contribution of this work is to handle the system scalability, single point of failure, and efficient and trustable user management problems occurred in the centralized PHR system by distributing user management tasks to multiple user authorities while preserving security and privacy of the PHR. The future improvement of this work is to make a more flexible solution for access permission assignment of the PHR dual layer protection's storage access control layer.

References

[1] A. Aridarma, T. L. Mengko, and S. Soegijoko, "Personal medical assistant: Future exploration," *Proceedings of the International Conference on Electrical Engineering and Informatics (ICEEI)*, Bandung, Indonesia, July 2011, pp. 1-6.

[2] M. Kuroda, and Y. Nohara, "IEEE802. 15.6 NB portable BAN clinic and M2M international standardization," *Proceedings of the 35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, July 2013, pp. 1660-1663.

[3] Z. Pang, Q. Chen, and L. Zheng, "A pervasive and preventive healthcare solution for medication noncompliance and daily monitoring," *Proceedings of the 2nd International Symposium on Applied Sciences in Biomedical and Communication Technologies (ISABEL)*, Bratislava, November 2009, pp. 1-6.

[4] P. C. Tang, J. S. Ash, D. W. Bates, J. M. Overhage, and D. Z. Sands, "Personal health records: definitions, benefits, and strategies for overcoming barriers to adoption," *Journal of the American Medical Informatics Association (JAMIA)*, 13(2), 2006, pp. 121-126.

[5] K. Garson, and C. Adams, "Security and privacy system architecture for an e-hospital environment," *Proceedings of the 7th Symposium on Identity and Trust on the Internet (IDtrust)*, Gaithersburg, MD, March 2008, pp. 122-130.

[6] B. A. Malin, K. El Emam, and C. M. O'Keefe, "Biomedical data privacy: problems, perspectives, and recent advances," *Journal of the American Medical Informatics Association (JAMIA)*, 20(1), 2013, pp. 2-6.

[7] K. Caine, and R. Hanania, "Patients want granular privacy control over health information in electronic medical records," *Journal of the American Medical Informatics Association (JAMIA)*, 20(1), 2013, pp. 7-15.

[8] R. Singh, V. Gupta, and K. Mohan, "Dynamic federation in identity management for securing and sharing personal health records in a patient-centric model in cloud," *International Journal of Engineering and Technology (IJET)*, 5(3), 2013, pp. 2201-2209.

[9] Y. Ding, and K. Klein, "Model-driven application-level encryption for the privacy of e-health data," *Proceedings of the 10th International Conference on Availability, Reliability, and Security (ARES)*, Krakow, February 2010, pp. 341-346.

[10] W. Bagga, and R. Molva, "Policy-based cryptography and applications," *Proceedings of the 9th International Conference on Financial Cryptography and Data Security (FC)*, Roseau, March 2005, pp. 72-87.

[11] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," *Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS)*, Alexandria, Virginia, USA, October 2006, pp. 89-98.

[12] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," *Proceedings of the 28th IEEE Symposium on Security and Privacy (SP)*, Berkeley, CA, USA, May 2007, pp. 321-334.

[13] 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," *Proceedings of the IEEE Transactions on Parallel and Distributed Systems (TPDS)*, 24(1), 2013, pp. 131-143.

[14] M. Li, S. Yu, K. Ren, and W. Lou, "Securing personal health records in cloud computing: patient-centric and fine-grained data access control in multi-owner settings," *Proceedings of the 6th International Conference on Security and Privacy in Communication Networks (SecureComm)*, Singapore, September 2010, pp. 89-106.

[15] J. Huang, M. Sharaf, and C. T. Huang, "A hierarchical framework for secure and scalable EHR sharing and access control in multi-cloud," *Proceedings of the 41st International Conference on Parallel Processing Workshops (ICPPW)*, Pittsburgh, PA, USA, September 2012, pp. 279-287.

[16] L. Ibraimi, M. Asim, and M. Petkovic, "Secure management of personal health records by applying attribute-based encryption," *Proceedings of the 6th International Workshop on Wearable Micro and Nano Technologies for Personalized Health (pHealth)*, Oslo, June 2009, pp. 71-74.

[17] C. Wang, X. Liu, and W. Li, "Implementing a personal health record cloud platform using ciphertext-policy attribute-based encryption,"

Proceedings of the International Conference on Intelligent Networking and Collaborative Systems (INCoS), Bucharest, September 2012, pp. 8-14.

[18] J. Li, G. Zhao, X. Chen, D. Xie, C. Rong, W. Li, and Y. Tang, "Fine-grained data access control systems with user accountability in cloud computing," *Proceedings of the 2th International Conference on Cloud Computing Technology and Science (CloudCom)*, Indianapolis, IN, USA, December 2010, pp. 89-96.

[19] P. Thummavet, and S. Vasupongayya, "A novel personal health record system for handling emergency situations," *Proceedings of the 17th International Computer Science and Engineering Conference (ICSEC)*, Nakorn Pathom, Thailand, September 2013, pp. 266–271.

[20] T. Parameswaran, S. Vanitha, and K. S. Arvind, "An efficient sharing of personal health records using DABE in secure cloud environment," *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)*, 2(3), 2013, pp. 925-932.

[21] J. L. F. Alemán, I. C. Señor, P. Á. O. Lozoya, and A. Toval, "Security and privacy in electronic health records: A systematic literature review," *Journal of biomedical informatics*, 46(3), 2013, pp. 541-562.

[22] M. Chase, and S. S. Chow, "Improving privacy and security in multi-authority attribute-based encryption," *Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS)*, Chicago, IL, USA, November 2009, pp. 121-130.

[23] K. Yang, X. Jia, K. Ren, and B. Zhang, "Dac-macs: Effective data access control for multi-authority cloud storage systems," *Proceedings of the IEEE INFOCOM*, Turin, April 2013, pp. 2895 – 2903.

[24] How CA Certificates Establish Trust. (2014). [Online]. Available: https://access.redhat.com/site/documentation/en-US/Red_Hat_Certificate_System/8.0/html/Deployment_Guide/Introduction_to_Public_Key_Cryptography-Certificates_and_Authentication.html#Certificates_and_Authentication-How_CA_Certificates_Establish_Trust

[25] J. Viega, M. Messier, and P. Chandra, "Network Security with OpenSSL: Cryptography for Secure Communications," O'Reilly Media, Incorporated, 2002, pp. 93-142.

[26] D. Weerasinghe, and M. Rajarajan, "Secure trust delegation for sharing patient medical records in a mobile environment," *Proceedings of the 7th International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM)*, Wuhan, September 2011, pp. 1–4.

[27] P. Linna, J. Leppaniemi, J. Soini, and H. Jaakkola, "Harmonizing emergency management knowledge representation," *Proceedings of the Portland International Conference Management of Engineering & Technology (PICMET)*,Portland, OR, August 2009, pp. 1047-1051.

[28] I. Mizunuma, and I. Masaki, "Multi-authority virtual network for intelligent transportation systems," *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*, Dearborn, MI, 2000, pp. 430-435.

[29] D. Cao, B. Zhao, X. Wang, J. Su, and G. Ji, "Multi-authority attribute-based signature," *Proceedings of the 3rd International Conference on Intelligent Networking and Collaborative Systems (INCoS)*, Fukuoka, November 2011, pp. 668-672.

Appendix A3

Journal Paper

P. Thummavet, and S. Vasupongayya, “Privacy-preserving emergency access control for personal health records,” Maejo International Journal of Science and Technology (MIJST), 9(01), 2015, pp. 108-120.

Maejo Int. J. Sci. Technol. **2015**, *9*(01), 108-120; doi: 10.14456/mijst.2015.7

Maejo International Journal of Science and Technology

ISSN 1905-7873

Available online at www.mijst.mju.ac.th

Technical Note

Privacy-preserving emergency access control for personal health records

Phuwanai Thummavet and Sangsuree Vasupongayya*

Department of Computer Engineering, Faculty of Engineering, Prince of Songkla University, P.O. Box 2 Kohong, Hatyai, Songkhla, 90112, Thailand

* Corresponding author, e-mail: vsangsur@coe.psu.ac.th

Received: 19 June 2014 / Accepted: 2 April 2015 / Published: 9 April 2015

Abstract: Recently, a flexible scheme for handling personal health records (PHRs) in emergency situations has been proposed. Under such a scheme, each PHR is classified as secure, restricted, or exclusive information. Secure PHRs are immediately available to the emergency response unit (ERU) staff. Restricted PHRs require additional approvals from a set of authorised people who are pre-selected by the PHR owner. Exclusive PHRs are only accessible by the owner. Previous work assumed that all ERU staff is trustworthy. To be practical, this work eliminates such an assumption. Several mechanisms are applied to ensure the usability and security of the newly proposed scheme. For example, an access-request authentication mechanism is applied to enhance the trustworthiness of the requests that are invoked by the ERU staff. Moreover, a transaction auditing mechanism is applied to provide a non-repudiation feature. This paper discusses the usability and security issues of the proposed scheme in practice and suggests how to classify a PHR considering the above-mentioned privacy levels.

Keywords: personal health record, privacy, security, ciphertext-policy attribute-based encryption, threshold cryptosystem

INTRODUCTION

Today, people are more aware of information concerning their health because of the rising cost of healthcare. Recently, alternative medicines such as dietary supplements and herbal products have gained popularity [1]. In addition, personal health record (PHR) system is emerging as a preventive healthcare method [2]. The PHR system allows an individual to collect, store, analyse, and share his/her personal health data with a group of trusted people such as family members, family doctors and caretakers [3]. The PHR system usually contains highly sensitive information

Maejo Int. J. Sci. Technol. **2015**, *9*(01), 108-120; doi: 10.14456/mijst.2015.7

[4]; it can include information related to the PHR owner's health such as his/her mental health, disease risks and laboratory test results. Therefore, the PHR system must ensure the security and privacy of the PHR owner's information, and the actual PHRs must be protected from an unauthorised access or modification. Moreover, the PHR owner must be able to manage and control all authorised access to his/her PHRs. To achieve such features, the PHR system should allow the PHR owner to define an access control policy on his/her PHRs, which must be enforced by the PHR system. Thus, an individual can access a PHR if and only if that individual has been granted the authority by the PHR owner via an access control policy. For example, John can grant access to his family doctor, Jason, by defining a policy such as "Jason, who is a doctor, can access my records." Hence the PHR system will allow only Jason, who is a doctor, to access John's PHRs.

An interesting PHR management issue arises during an emergency [5, 6]. Generally, an emergency response unit (ERU) staff member is the first care provider to reach the victim. Providing correct and useful health information (e.g. personal diseases) about the victim in the emergency situation can increase the opportunity to provide proper treatment to save the victim's life or alleviate his/her critical conditions. Therefore, it is vital to allow the ERU staff to access the necessary PHR information of the victim in an emergency situation [7]. According to the above example, John can allow Dr. Jason to access his PHRs because John knows Dr. Jason. In an emergency, John may not know any ERU staff. Thus, John will not be able to define a policy to allow a specific ERU staff member to access his PHRs. During an emergency situation, John may be unconscious and may not be able to grant any permission to the ERU staff at the scene. Moreover, ERU staff should not access John's PHRs unless they are necessary to save his life. Thus, the question is how to allow ERU staff to access the victim's PHRs during an emergency situation.

A scheme to manage and handle PHRs during an emergency situation has been proposed in our previous work [8], which allows different access restrictions. Under such a scheme, each PHR is classified into secure, restricted, and exclusive categories. Different categories provide different access permissions to ERU staff for the victim's PHRs. Secure PHRs are freely available to ERU staff during an emergency situation. Restricted PHRs are accessible to an ERU staff member if and only if he/she was granted access permission by at least t out of n trusted people who are pre-selected by the PHR owner, where t is an acceptable threshold, pre-defined by the PHR owner and n is the total number of trusted people on the PHR owner's list. Exclusive PHRs are not accessible even during an emergency situation. With the proposed scheme, the PHR owners can selectively share their PHRs with the ERU staff while additional data can be requested if needed.

This work extends the previous scheme to cover external ERUs that were not included in the original design [8]. The ERU staff authentication process was not considered in the original design because the ERU staff was assumed to be trustworthy and part of the PHR system. However, in the real world ERU staff can be from various external sources such as public organisations, medical care institutions, private organisations, non-profit organisations or a group of volunteers. Therefore, the ERU staff must be verified by their authorised commander/manager. To guarantee the reliability of the verification process, an access request authentication (ARA) mechanism is proposed in this work as an extension of the previous scheme. Hence the PHR access request from any ERU staff can be performed if and only if the staff member is granted access permission by his/her authorised ERU commander/manager. In addition, a transaction auditing mechanism is used in this study to provide a non-repudiation feature. Furthermore, the security of all connections in the proposed scheme is provided by means of secure sockets layer protocols and secure shell protocols.

Maejo Int. J. Sci. Technol. **2015**, *9*(01), 108-120; doi: 10.14456/mijst.2015.7

RELATED WORK

A database-level encryption was employed by Weerasinghe and Muttukrishnan [9] to provide an information exchange scheme between the ERU staff and the PHR service providers via a trusted third party. Under such a scheme, the actual PHRs are encrypted and stored by a PHR service provider. During emergency situations, the PHR service provider delivers the requested PHR to the ERU staff on behalf of the PHR owner. The authentication of each party should be done by the trusted third party. However, the use of a database key becomes an issue because such a technique can introduce a privacy risk for the PHR owner [10]. To access the information, the ERU staff has access to the key through which multiple access can be performed. Typically, the access permission under such a scheme is binary and the ERU staff can access the entire database. In other words, the ERU staff can access all records stored in a particular database even though some records may not be related to their tasks.

To solve the privacy concern of the database-level encryption technique, a digital pseudonym was introduced by Huda et al [11]. The pseudonym indexes the PHRs for each PHR owner, whose name (i.e. a field in the database) is replaced by a random pseudonym. Then the pseudonym is encrypted and stored on the PHR owner's health smart card. The ERU staff uses the pseudonym to retrieve the victim's PHRs during an emergency situation. Using the pseudonym, even though the database records are exposed to unauthorised users, the PHR owner's privacy is still preserved. However, the scope of the information accessed by the ERU staff cannot be limited because the ERU staff can access all records indexed by a particular pseudonym.

A backup mechanism at a trust centre for the PHR owner was proposed by the healthcare system for patient privacy [12]. Under such a scheme, the PHR owner can define the information that will be available during emergency situations and selects his/her PHRs to be stored at a trusted server. The information stored at the trust centre is freely available to the ERU staff during an emergency situation. Hence the privacy of the PHR owner and the secrecy of the PHR can be preserved. However, only static information pre-selected by the PHR owner is available. In our proposed scheme both static and additional information is available to the ERU staff. The static pre-selected information is the secure PHRs and the additional information is the restricted PHRs, which are available upon request.

A key-policy attribute-based encryption (KP-ABE) [13] was employed in the break-glass access [14, 15] to protect the PHR information. The KP-ABE enables a PHR owner to specify a set of attributes embedded in the encrypted PHR. The PHR owner selects a set of PHRs to be freely available to the ERU staff during emergency situations. Then a special 'emergency' attribute is added during the PHR encryption process. A set of PHRs can only be decrypted by a key that contains the 'emergency' attribute and is distributed to the ERUs during an emergency situation. However, only static pre-selected information is available.

The KP-ABE was also employed by Huang et al. [6] to offer the PHR information according to the severity level of the situation. The PHR owner can assign any of the three severity levels (mild, moderate and severe) to each PHR. Then each PHR is encrypted using the KP-ABE technique with a set of owner-desired attributes and the severity level. The KP-ABE private keys that are based on different severity levels can access the PHR information with different scopes. Under such a scheme, the availability of the pre-defined information is an issue. For example, if the ERU staff is allowed to access mild-level and moderate-level information, then the information is always available even when it is not required.

Maejo Int. J. Sci. Technol. **2015**, *9*(01), 108-120; doi: 10.14456/mijst.2015.7

OUR PROPOSED PRIVACY-PRESERVING EMERGENCY ACCESS CONTROL SCHEME

Our proposed privacy-preserving emergency access control scheme for PHRs is illustrated in Figure 1. The scheme consists of five modules and three players. The modules comprises the user authority (UA), the emergency server (EmS), the PHR server, the audit server and the emergency authority (EA). The players include the PHR owner, the PHR trusted users and the ERU staff.

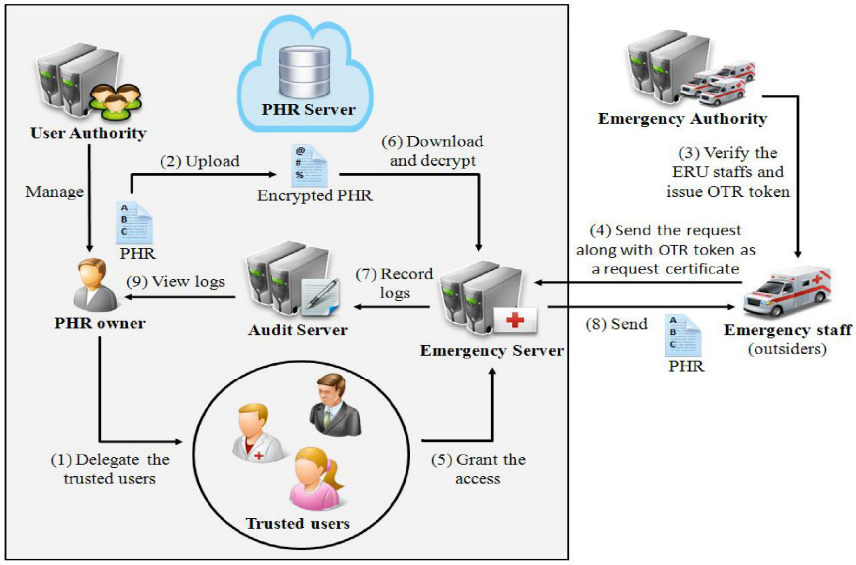

Figure 1. Proposed privacy-preserving emergency access control scheme

The UA is responsible for performing all PHR user management tasks such as creating a user, generating a user key, distributing the user key and revoking the user. The PHR server is an actual PHR storage, which can be internal or public storage. The PHRs are encrypted and uploaded to the storage. The EmS handles all tasks related to an emergency situation. These three modules were presented in the original design [8]. The next two modules are added in this work. The audit server records all activities performed by the ERU staff during emergency situations and produces reports for the PHR owner. The EA is responsible for managing tasks related to the ERU staff, such as verifying ERU staff identity, authorising ERU staff, revoking ERU staff access and generating a one-time request (OTR) token for ERU staff. The EA can be either distributed or centralised as long as it is registered with the proposed scheme.

The PHR owners can manage and track all activities conducted on their PHRs. Trusted users can be partially granted restricted access on behalf of the PHR owner during emergency situations. When the total number of approvals received from the pre-selected trusted users is equal to or greater than the pre-defined threshold, restricted PHR access permission is granted. The ERU staff must be verified and authorised by their corresponding EA.

Maejo Int. J. Sci. Technol. **2015**, *9*(01), 108-120; doi: 10.14456/mijst.2015.7

The ARA mechanism ensures that only authorised ERU staff is allowed to access the PHR system. Each PHR access request invoked by the ERU staff must be verified by its commander/ manager (denoted as EA in Figure 1). Once the ERU staff member is verified, the OTR token, which is a certificate with a specific expiration time, is generated. The request with a valid OTR token is processed by the EmS. The ERU staff cannot reuse the token once it is expired. To expedite this process, the ERU staff can be verified and issued with the OTR token in an emergency vehicle before they arrive at the emergency location. Hence the delay time to access any secure PHR is eliminated because the secure PHRs can be immediately accessed once the valid OTR token is presented. In addition, a transaction auditing mechanism is employed to guarantee a non-repudiation feature of all access conducted by the ERU staff.

Under our proposed scheme, the PHRs are encrypted at the data origin using the ciphertext-policy attribute-based encryption (CP-ABE) technique [16]. Then the encrypted PHR is uploaded to the PHR server (denoted as (2) in Figure 1). Using the CP-ABE scheme, the access policy of each PHR is embedded during the PHR encryption process. The policy is defined by the PHR owner. Only the user who has the CP-ABE private key that satisfies the access policy can decrypt the encrypted PHR. In addition, the secure sockets layer protocol and the secure shell protocol are employed to provide a secure communication among the modules and players under the proposed scheme. The information collected by the network traffic eavesdropping technique remains protected. In the following section the assignment of a privacy level to each PHR is presented. Then the PHR pre-processing and accessing methods are described.

Defining Privacy Level

This section provides a guideline for the PHR owners in order to classify their PHRs into one of the three privacy levels: secure, restricted and exclusive. The guideline is created according to the sensitivity of the PHR information. Typically, health related information of an individual stored in a PHR system has different sensitivity levels. For example, some information such as mental health, domestic abuse/violence, drug abuse, disorders and disabilities is considered to be sensitive for some people. A person usually does not disclose such information to others. Such information can result in disgrace or even unfair job opportunities to its owner [4, 5]. However, other information such as congenital diseases, allergies and disease risks can help the ERU staff make a better decision in treating the victim during emergency situations [7].

The secure PHRs are available to the ERU staff during emergency situations. Therefore, the basic information of a person's health that is necessary to treat the person must be provided. Some people are allergic to simple medicine such as Paracetamol. Such information is important during a life-threatening condition. Thus, a list of information such as congenital diseases, allergies and disease risks is suggested to be under the secure-level category [17]. In addition, a list of emergency contact people for the victim is classified under this category so the people who know the victim can be informed about the situation.

The ERU staff is allowed to access the restricted-level information if an access permission is granted by a certain number of the victim's delegates. Unlike the secure-level information, the restricted-level information will not be available immediately. Thus, the information under this category can only be used by the physicians that are away from the emergency scene, in a fully equipped emergency vehicle or an intensive care unit at a hospital. This set of information should include the person's medical history, laboratory test results, physicians' recommendations, his/her

Maejo Int. J. Sci. Technol. **2015**, *9*(01), 108-120; doi: 10.14456/mijst.2015.7

physicians' contact information, and some relevant health-monitoring data. Such information will help physicians make a better judgment on the next actions.

Finally, the exclusive-level information is considered to be highly sensitive according to the PHR owner judgment. Usually, this set of information includes mental health, domestic abuse/violence, drug abuse, disorders and disabilities [4, 5]. Note that this guideline is provided as a suggestion and the proposed scheme is not limited to it.

PHR Pre-processing

The proposed scheme uses the EmS to perform all PHR retrieving and decrypting tasks during emergency situations. The EmS attribute must be defined in the access policy of the secure and restricted PHRs. The exclusive PHR access policy does not include any attribute of the EmS. As a result, the ERU staff is not allowed to access exclusive PHRs even during emergency situations while the secure PHR is accessible by the authorised ERU staff during emergency situations. To assign a secure PHR, the EmS attribute must be added to the access policy of that particular PHR. Then the PHR is encrypted using the CP-ABE with a defined access policy and the encrypted PHR is securely uploaded to the PHR server. By adding the EmS attribute to the access policy, the EmS is able to decrypt a particular PHR. During an emergency situation, the authorised ERU staff can access the secure PHRs instantly via the EmS.

The restricted PHR is accessible by the authorised ERU staff if and only if they are granted an access permission by at least t (pre-determined threshold) out of n trusted users who are pre-selected by the PHR owner. To assign a restricted PHR, the PHR owner must first select a set of trusted users (denoted as (1) in Figure 1). These trusted users are asked to make a decision to grant access permission on any restricted PHR on behalf of the PHR owner. A restricted emergency key (REK) attribute is added to the access policy of that particular PHR. Then the PHR is encrypted using the CP-ABE with defined access policy. Next, a set of random secret keys associated with the number of trusted users are generated. The REK attribute is encrypted using threshold cryptosystem [18] with a set of random secret keys and a pre-determined threshold (t) as encryption parameters. A random secret key is assigned to each trusted user. Each secret key is encrypted using a corresponding trusted user's public key. The encrypted PHR is securely uploaded to the PHR server while the encrypted REK attribute and the set of encrypted secret keys are securely uploaded to the EmS. Thus, the EmS can decrypt any restricted PHR if and only if at least t trusted users provide their approval. Because each secret key is encrypted with a trusted user's public key, only the trusted user's private key can decrypt the secret key. With the threshold cryptosystem, the REK attribute can be decrypted if at least t secret keys are provided. Using the REK attribute, the EmS can decrypt the restricted PHRs.

PHR Accessing

In this section both secure and restricted PHR accessing sequences are explained. Figure 2 and Figure 3 show sequences of transactions occurring when accessing secure PHRs and restricted PHRs respectively. To enhance the trustworthiness of a request invoked by the ERU staff, the ARA mechanism guarantees that each request must be verified by his/her EA (steps 1–4 in Figure 2 and Figure 3). The OTR token ensures that the ERU staff is verified by his/her EA (step 3 in Figure 2 and Figure 3). The verification process can be expedited by issuing an OTR token to the ERU staff once the emergency case is assigned. Because the OTR token is a certificate with a specific expiration time, the token cannot be reused when its lifetime expires. Therefore, this mechanism

Maejo Int. J. Sci. Technol. **2015**, *9*(01), 108-120; doi: 10.14456/mijst.2015.7

assures that only authorised ERU staff can access the PHRs. All requests and transactions are recorded. Figure 4 shows the transactions collected by the auditing system using our prototype software.

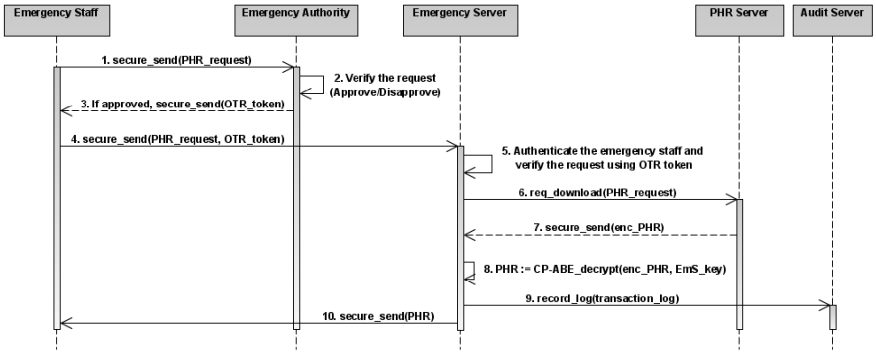

Figure 2. Secure PHR access sequences.

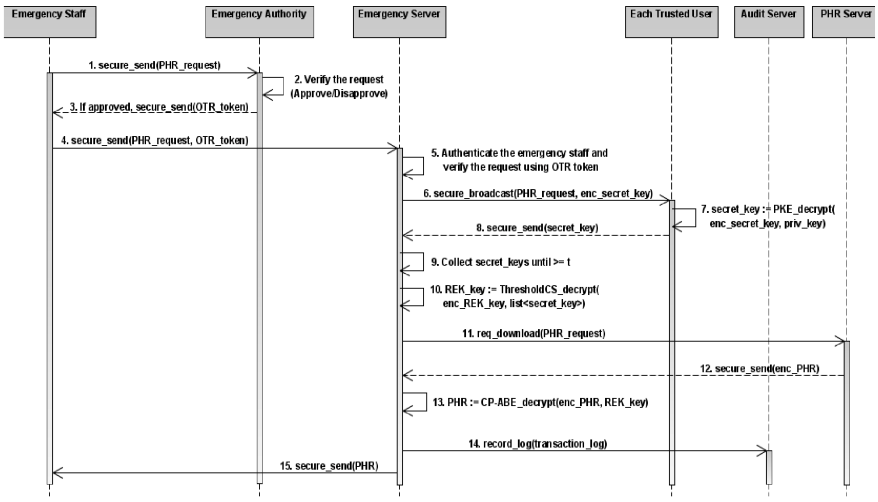

Figure 3. Restricted PHR access sequences

Maejo Int. J. Sci. Technol. **2015**, *9*(01), 108-120; doi: 10.14456/mijst.2015.7

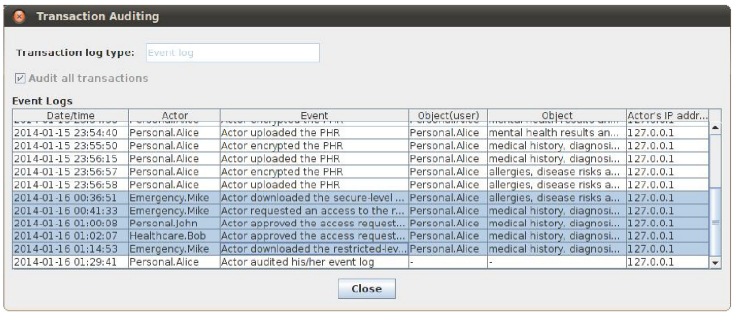

Figure 4. List of transactions collected by the audit server

To access secure PHRs, the ERU staff must send a PHR access request along with the OTR token to the EmS (step 4 in Figure 2). The OTR token is sent to the ERU staff if he/she is verified by his/her corresponding EA. Once the EmS successfully verifies the OTR token, the requested secure PHRs are downloaded and decrypted using the EmS attribute (steps 5–8 in Figure 2). Then the EmS stores the transaction information on the audit server (step 9 in Figure 2) and securely sends the requested secure PHRs to the ERU staff (step 10 in Figure 2).

To access restricted PHRs, the ERU staff must send a PHR access request along with the OTR token to the EmS (step 4 in Figure 3). The OTR token is sent to the ERU staff if he/she is verified by his/her corresponding EA. Once the EmS successfully verifies the OTR token (step 5 in Figure 3), the EmS securely broadcasts the request message to each corresponding trusted user for approval (step 6 in Figure 3). If the trusted user approves the request, the corresponding encrypted random secret key will be decrypted by the trusted user's private key (step 7 in Figure 3). The random secret key is sent to the EmS through a secure channel (step 8 in Figure 3). If the total number of random secret keys collected by the EmS is equal to the pre-determined threshold (t), the EmS decrypts the encrypted REK attribute value using the threshold cryptosystem (steps 9-10 in Figure 3). Next, the EmS downloads the requested PHRs from the PHR server and uses the REK attribute to decrypt the PHRs (steps 11-13 in Figure 3). Finally, the ERU staff receives the requested PHRs from the EmS via a secure channel (step 15 in Figure 3). In addition, the EmS records a transaction log on the audit server (step 14 in Figure 3).

USABILITY AND SECURITY DISCUSSIONS

Usability Issues

Figure 5 shows a typical flow of events during an emergency situation. First, an emergency situation occurs. Second, the call is made to the emergency hotline centre. Third, the emergency location and victim's current conditions are provided to the assigned ERU staff. Fourth, the ERU staff reach the victim. Fifth, the victim is transferred to a hospital or a medical facility. The commonly accepted standard response time from the first call (step 2 in Figure 5) until the ERU staff reach the victim (step 4 in Figure 5) is 8 minutes [19]. This section will cover only secure and restricted PHRs because the proposed scheme allows the ERU staff to access only these types of PHRs.

Maejo Int. J. Sci. Technol. **2015**, *9*(01), 108-120; doi: 10.14456/mijst.2015.7

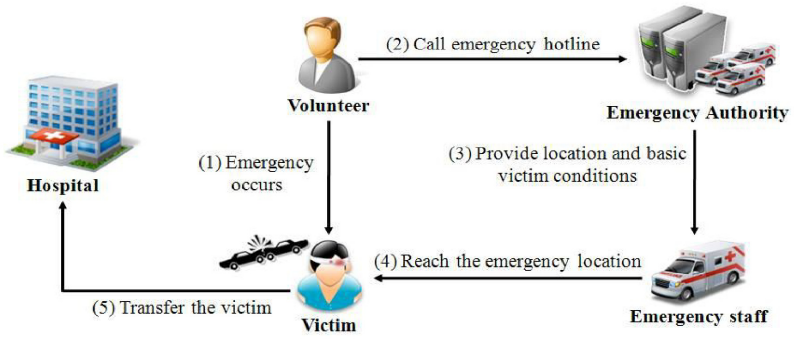

Figure 5. Typical flow of events during an emergency situation

According to the secure PHR access sequences shown in Figure 2, the total processing time to retrieve the secure PHRs includes the required time for: (1) the ERU staff to request an OTR token, (2) the EA to authorise and issue the OTR token, (3) the OTR token to be sent to the ERU staff, (4) the ERU staff to send the PHR request along with their OTR token to the EmS, (5) the EmS to download and decrypt the requested PHRs, and (6) the EmS to send the PHRs to the ERU staff. According to the cellular standard for the third generation [20], the data transmission rate in a moving vehicle is 348 kbps, meaning that 348,000 bits or 43.5 KB of data can be transferred each second. This amount of data can contain a text of approximately 10 novel-size pages. Therefore, the amount of time to transmit a request, an OTR token and a secure PHR is negligible. The EmS processing time depends on the PHR storage and the decryption process. The underlying encryption scheme of the CP-ABE is an advanced encryption standard (AES) [21] in cipher block chaining (CBC) mode [22], which takes less than 3.25 s to encrypt an image with the size of 468 KB [23]. Because the encryption and decryption time for AES-CBC is the same, the decryption processing time can be negligible. Using the current data storage technology, 50,000 records can be searched in 2.5 s [24]. Thus, the PHR storage processing time is not a problem. The only external factor to the total processing time is the EA processing time, which will be discussed later. Using the above supporting evidence, the secure PHR accessing time is reasonable in practical situations.

As described previously, the restricted PHRs are designed for the medical staff at the hospital to treat the victim when he/she is no longer at the emergency location. Therefore, there is a period of time between the call to the hotline and the victim arrival at the hospital during which the medical staff can obtain the necessary approval to access the necessary restricted PHRs. The processing time to retrieve the restricted PHRs includes the required time for: (1) the ERU staff to receive the OTR token, (2) the ERU staff to send the PHR request along with the OTR token, (3) the EmS to send the request to each trusted user, (4) each trusted user to respond to the request, (5) the partial secret key of each trusted user to be sent to the EmS, (6) the EmS to download and decrypt the requested PHRs, and (7) the EmS to send the PHRs to the ERU staff. The above factors discussed for the secure PHRs can also be applied to the restricted PHRs, the only difference being the response time of the trusted user. According to a study [25], the average response time of a person to an incoming text messaging is 431.28 s during simultaneous conversations and 391.88 s during non-simultaneous conversations. Therefore, it can take up to 7 min. for the trusted users to respond to a restricted PHR access approval request. Considering the 8-min. response time standard,

Maejo Int. J. Sci. Technol. **2015**, *9*(01), 108-120; doi: 10.14456/mijst.2015.7

the medical staff at the hospital are able to access the restricted PHRs of the victim before the victim reaches the hospital.

To provide an additional assurance that the trusted users will respond to the request, the PHR owner should include at least one of the trusted users on the emergency contact list. Because the list is classified as a secure PHR, the ERU staff can contact the trusted user directly. The restricted PHRs are designed for the medical staff at the hospital; therefore, the medical staff can be added as an attribute in the access policy during the CP-ABE encryption of the PHRs and can thus access the PHRs. However, they must be a member of an authority that is recognised by the victim's PHR system.

Under the proposed scheme, the EA acts as a trusted agent to verify all of its ERU staff. Because the EA can be from various sources, the process of adding a new EA to the proposed scheme must be done carefully and the new EA must be verified. To ensure the performance of the ERU staff, each EA must be periodically evaluated. The request approval processing time and the OTR token generation time must be used as the key performance indicators to evaluate the EA. In addition, the OTR token lifetime may allow the ERU staff to perform a replay attack on the PHR system. Therefore, the ERU staff misconducts and performances can be used as another key performance indicator to evaluate the EA. The EA with poor performance must be removed.

Security Issues

Four attack models are discussed to account for possible security threats. The first model involves a database intruder. Under the proposed scheme, the actual PHR storage can be a public storage. Therefore, the database intruder or the storage administrator may try to access the information. However, the PHR is encrypted and the decryption keys are securely stored on separate trusted servers (i.e. the UA and the EmS). Hence the encrypted PHR stored on the PHR storage is protected with the assumption that the cryptographic primitives are not broken and the decryption key is not accessible.

The second attack model concerns an unauthorised access. The ERU staff may try to access the PHRs. However, the ARA mechanism prevents such access by allowing only the ERU staff with an approval from their corresponding EA to access the data. The approval is in the form of a valid OTR token. The ERU staff uses the received OTR token as a request certificate to access the requested PHRs through the EmS. An unauthorised access is prevented at the EmS and any attempt from the ERU staff is recorded by the audit server. Because the conduct of each ERU staff is used as a key performance indicator during the EA evaluation process, any misconduct by the ERU staff affects its EA performance. The EA evaluation process and the transaction auditing mechanism can indirectly prevent unauthorised access.

The third attack model is a replay attack. The ERU staff with a valid OTR token may conduct a replay attack. However, the OTR token is a certificate with a specific expiration time. Therefore, the ERU staff will not be able to reuse the OTR token once it is expired. However, this does not cover the period of time that the OTR token remains valid. Therefore, the lifetime of the OTR token must be short. The auditing information can show any misconduct of the ERU staff, which affects the performance of the corresponding EA.

The last attack model is a non-repudiation case. The audit server records all transactions invoked by the ERU staff and all activities can be tracked by the PHR owner. The transaction auditing mechanism is a very important mechanism to provide a non-repudiation feature.

Maejo Int. J. Sci. Technol. **2015**, *9*(01), 108-120; doi: 10.14456/mijst.2015.7

CONCLUSIONS

This work has extended the original design of a PHR system for handling emergency situations to support a more practical scenario. In the original design all players were assumed to be trustworthy. In this work the ERU staff is considered an outsider and unknown to the system. Two mechanisms have been proposed to enhance the trustworthiness of the PHR access requests from an ERU staff member during an emergency situation. First, an ARA mechanism is designed to ensure the verification of the ERU staff by an on-duty emergency unit commander/manager. Any request invoked by an ERU staff member must be approved by his/her corresponding on-duty EA commander. Second, the transaction auditing mechanism is added to allow the PHR owners to track all transactions related to their PHRs. In addition, the auditing mechanism serves as a method for providing a non-repudiation feature for all PHR access performed by the ERU staff. Using the proposed extension, the trustworthiness of the requests invoked by the ERU staff is enhanced and the limitation of the previous work is eliminated.

The current data transmission rate and storage technology allows the proposed scheme to provide the requested PHRs within a commonly acceptable time and there is only one security limitation in the proposed scheme, which is the period of time that the OTR token remains valid. The suggested solution is to keep the lifetime of the OTR token short and to evaluate the EA based on its ERU staff performance and misconduct. Furthermore, a guideline on defining a proper privacy level for each PHR has been presented. The idea of collecting and storing transactions by an audit server is demonstrated using our developed prototype.

ACKNOWLEDGEMENTS

This work was supported by the Higher Education Research Promotion and National Research University Project of Thailand, Office of the Higher Education Commission (under the funding no. MED540548S at Prince of Songkla University).

REFERENCES

1. W. Wangcharoen, D. Amornlerdpison and K. Mengumphan, "Factors influencing dietary supplement consumption: A case study in Chiang Mai, Thailand", *Maejo Int. J. Sci. Technol.*, **2013**, *7*, 155-165.
2. A. A. Ozok, H. Wu, M. Garrido, P. J. Pronovost and A. P. Gurses, "Usability and perceived usefulness of personal health records for preventive health care: A case study focusing on patients' and primary care providers' perspectives", *Appl. Ergonom.*, **2014**, *45*, 613-628.
3. P. C. Tang, J. S. Ash, D. W. Bates, J. M. Overhage and D. Z. Sands, "Personal health records: Definitions, benefits, and strategies for overcoming barriers to adoption", *J. Am. Med. Inform. Assoc.*, **2006**, *13*, 121-126.
4. B. A. Malin, K. El Emam and C. M. O'Keefe, "Biomedical data privacy: Problems, perspectives, and recent advances", *J. Am. Med. Inform. Assoc.*, **2013**, *20*, 2-6.
5. K. Caine and R. Hanania, "Patients want granular privacy control over health information in electronic medical records", *J. Am. Med. Inform. Assoc.*, **2013**, *20*, 7-15.
6. J. Huang, M. Sharaf and C. T. Huang, "A hierarchical framework for secure and scalable EHR sharing and access control in multi-cloud", Proceedings of 41st International Conference on Parallel Processing Workshops, **2012**, Pittsburgh, USA, pp. 279-287.

Maejo Int. J. Sci. Technol. **2015**, *9*(01), 108-120; doi: 10.14456/mijst.2015.7

7. J. L. Fernández-Alemán, I. C. Señor, P. Á. Lozoya and A. Toval, "Security and privacy in electronic health records: A systematic literature review", *J. Biomed. Inform.*, **2013**, *46*, 541-562.
8. P. Thummavet and S. Vasupongayya, "A novel personal health record system for handling emergency situations", Proceedings of 17th International Computer Science and Engineering Conference, **2013**, Nakorn Pathom, Thailand, pp.266-271.
9. D. Weerasinghe and R. Muttukrishnan, "Secure trust delegation for sharing patient medical records in a mobile environment", Proceedings of 7th International Conference on Wireless Communications, Networking and Mobile Computing, **2011**, Wuhan, China, pp.1-4.
10. Y. Ding and K. Klein, "Model-driven application-level encryption for the privacy of e-health data", Proceedings of 5th International Conference on Availability, Reliability and Security, **2010**, Krakow, Poland, pp.341-346.
11. M. N. Huda, S. Yamada and N. Sonehara, "Privacy-aware access to patient-controlled personal health records in emergency situations", Proceedings of 3rd International Conference on Pervasive Computing Technologies for Healthcare, **2009**, London, UK, pp.1-6.
12. J. Sun, X. Zhu, C. Zhang and Y. Fang, "HCPP: Cryptography based secure EHR system for patient privacy and emergency healthcare", Proceedings of 31st International Conference on Distributed Computing Systems, **2011**, Minneapolis, USA, pp.373-382.
13. M. Li, S. Yu, K. Ren and W. Lou, "Securing personal health records in cloud computing: Patient-centric and fine-grained data access control in multi-owner settings", Proceedings of 6th International Conference on Security and Privacy in Communication Networks, **2010**, Singapore, pp.89-106.
14. V. Goyal, O. Pandey, A. Sahai and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data", Proceedings of 13th ACM Conference on Computer and Communications Security, **2006**, Alexandria, USA, pp.89-98.
15. 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 Trans. Parallel Distr. Syst.*, **2013**, *24*, 131-143.
16. J. Bethencourt, A. Sahai and B. Waters, "Ciphertext-policy attribute-based encryption", Proceedings of IEEE Symposium on Security and Privacy, **2007**, Berkeley, USA, pp.321-334.
17. S. Aguinaga and C. Poellabauer, "Method for privacy-protecting display and exchange of emergency information on mobile devices", Proceedings of International Conference on Collaboration Technologies and Systems, **2012**, Denver, USA, pp.596-599.
18. T. P. Pedersen, "A threshold cryptosystem without a trusted party", Proceedings of Workshop on Theory and Application of Cryptographic Techniques, **1991**, Brighton, UK, pp. 522-526.
19. P. T. Pons and V. J. Markovchick, "Eight minutes or less: Does the ambulance response time guideline impact trauma patient outcome?", *J. Emerg. Med.*, **2002**, *23*, 43-48.
20. International Telecommunication Union, "About mobile technology and IMT-2000", **2011**, http://www.itu.int/osg/spu/imt-2000/technology.html#Cellular Standards for the Third Generation (Accessed: April 2014).
21. National Institute of Standards and Technology, "Advanced encryption standard (AES)", **2001**, http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf (Accessed: April 2014).
22. National Institute of Standards and Technology, "Recommendation for block cipher modes of operation", **2001**, http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf (Accessed: April 2014).

Maejo Int. J. Sci. Technol. **2015**, *9*(01), 108-120; doi: 10.14456/mijst.2015.7

23. R. Doomun, J. Doma and S. Tengur, "AES-CBC software execution optimization", Proceedings of International Symposium on Information Technology, **2008**, Kuala Lumpur, Malaysia, pp. 1-8.
24. K. K. Lee, W. Tang and K. Choi, "Alternatives to relational database: Comparison of NoSQL and XML approaches for clinical data storage", *Comput. Meth. Programs Biomed.*, **2013**, *110*, 99-109.
25. A. Battestini, V. Setlur and T. Sohn, "A large scale study of text-messaging use", Proceeding of 12th International Conference on Human Computer Interaction with Mobile Devices and Services, **2010**, Lisbon, Portugal, pp.229-238.

Appendix B

Development tools

- Development tools
  - Ubuntu 10.04 LTS (Lucid Lynx)
  - TextEdit and Eclipse
  - Java SDK 6
  - OpenSSL 1.0.1e
  - MySQL 14.14 Distrib 5.1.62
  - GCC compiler 4.4.3
  - m4-1.4.16, gmp-5.0.2, pbc-0.5.12, cpabe-0.11, libpbc-0.5.12, and libbswabe-0.9
  - simclist-1.6 and curl-7.27.0
  - paillier.jar
  - Java, C, and SQL languages

Appendix C

API For the Proposed MA-PHR Framework

This section provides a guideline to developing an external application based on the proposed MA-PHR framework to an external software developer. That is, the external developer can develop his/her PHR client application using the proposed MA-PHR system as a secure data storage backend through provided application programming interface (API) functions. With the proposed system, the developer does not need to develop security and privacy mechanisms, user and attribute management mechanisms or transaction auditing mechanisms himself/herself.

As presented in the system development section, the proposed system consists of two sides: client and server sides. The client side consists of the backend and the frontend. The client backend gathers several low-level modules such as encryption, security and privacy, network, and user management modules, which were implemented using C language. The client frontend is the place where the external developer can develop and customize his/her application (using any language such as C, Java, JavaScript, and Python) and the application can utilize the features of the client backend via the provided API functions which will be introduced later. In addition, the server side of the proposed system includes four servers, namely, root authority (RA), user authority (UA), audit server (AS), and PHR server, which were fully implemented using C and SQL languages. Each server has a local database for independently storing the information. The server and the client sides always securely communicate with each other via an SSL/TLS secure channel.

Under the proposed framework, the client backend modules will be compiled and built as the shared library (.so file) which allows the external developer to load it onto his/her client application (the client frontend) at the runtime. As an example, the following shows how the client frontend application that is implementing using any language (e.g., Java, C, and Python) to collaborate with the proposed MA-PHR framework's client backend modules.

Calling the client backend modules from Java code

Basically, the client backend was designed and developed to support a call from Java code. That is, the Java code can invoke the client backend modules

using the Java Native Interface (JNI) [31] without modifying any backend code. For example, assuming that a client frontend application wants to invoke the PHR encryption module, Fig. 15 demonstrates an example Java snippet code.

```
1  public class PHRclient{
2
3      // Declare the native C function
4      private native boolean encrypt_phr_main(String phr_path, String access_policy);
5
6      public encryptPHR(String phr_path, String access_policy){
7
8          // Load the shared library "PHRapp_User_JNI.so"
9          System.loadLibrary("PHRapp_User_JNI");
10
11         // Call the native C function
12         if(encrypt_phr_main(phr_path, access_policy))
13             System.out.println("encryption succeeded");
14         else
15             System.out.println("encryption failed");
16     }
17 }
```

Fig. 15. An example of calling the client backend module by Java code

According to the example above, if the application calls the method encryptPHR(), the method will load the shared library "libPHRapp_User_JNI.so" into the application's memory section (line 9 in Fig. 15), and then the application will initiate the backend encryption module by calling the JNI-to-C mapping function encrypt_phr_main() (line 12 in Fig. 15) that is provided by the loaded library. Note that, every shared library always has the "lib" prefix and the ".so" suffix as extension. But, when calling the library in the Java code, we have to exclude them as shown at line 9 in Fig. 15. Furthermore, we have to declare the native JNI-to-C mapping function prototype at line 4 in Fig. 15 so that the Java compiler can know the being of the function. Please refer to the tutorials [63], [64] for more information about the JNI.

At this point, if the developer goes to the source code file named "client_user_main_jni.c" (the client backend source code), the developer will found the JNI-to-C mapping function named Java_UserMain_encrypt_1phr_1main() as illustrated in Fig. 16. This mapping function will be called when the frontend application is calling the encrypt_phr_main(). Next, the mapping function will call the real encryption function encrypt_phr() (line 31-32 in Fig. 16) to encrypt the PHR record. Additionally, if the developer goes to the file named "client_common.h", the

developer will found the list of all functions for all client backend modules available for the client frontend application to invoke.

```
/*
 * Class:     UserMain
 * Method:    encrypt_phr_main
 * Signature: (Ljava/lang/String;Ljava/lang/String;)Z
 */
JNIEXPORT jboolean JNICALL Java_UserMain_encrypt_1phr_1main(
    JNIEnv *env, jobject obj, jstring j_phr_path, jstring j_access_policy)
{
    const char *phr_path;
    const char *access_policy;
    jclass     cls;
    boolean    encrypting_flag;

    // Get variables from Java
    phr_upload_from_path = (*env)->GetStringUTFChars(env, j_phr_path, 0);
    access_policy        = (*env)->GetStringUTFChars(env, j_access_policy, 0);

    Java_env    = env;
    Java_object = obj;

    // Get the method id for returning output to Java
    cls = (*env)->GetObjectClass(env, obj);
    Java_backend_alert_msg_callback_handler_id = (*env)->GetMethodID(env, cls,
        "backend_alert_msg_callback_handler", "(Ljava/lang/String;)V");

    if(Java_backend_alert_msg_callback_handler_id == 0)
        int_error("Could not find the method \"backend_alert_msg_callback_handler\"");

    // Encrypt the PHR
    set_phr_encrypting_working_flag(true);
    encrypting_flag = encrypt_phr((char *)phr_path,
        (char *)access_policy, backend_alert_msg_callback_handler);
    set_phr_encrypting_working_flag(false);

    // Free up the Java string arguments
    (*env)->ReleaseStringUTFChars(env, j_phr_path, phr_path);
    (*env)->ReleaseStringUTFChars(env, j_access_policy, access_policy);

    return encrypting_flag;
}
```

Fig. 16. An example of a JNI-to-C mapping function of the client backend module

Calling the client backend modules from C code

To call the backend modules from C code, since the backend modules were developed based on C language, the client frontend application can call the backend modules directly, without the need to communicate with any JNI-to-C mapping function. Fig. 17 shows an example C snippet code. For more information, please refer to the tutorials [65], [66]. Note that, when compiling the frontend C code, use the following command:

“gcc –g –o application/name

list/of/source/files

–lshared/library/name/excluding/prefix/and/suffix

–Lshared/library/directory

–Wl,–rpath=shared/library/directory”

```c
#include <stdio.h>

int main(int argc, char **argv)
{
    char *phr_path;
    char *access_policy;

    if(argc != 3)
        printf("failed to get the correct arguments\n");

    phr_path      = argv[1];
    access_policy = argv[2];

    // Call the backend module
    if(encrypt_phr(phr_path, access_policy))
        printf("encryption succeeded\n");
    else
        printf("encryption failed\n");

    return 0;
}
```

Fig. 17. An example of calling the client backend module by C code

Calling the client backend modules from Python code

There are two approaches that Python code can call the C-implemented functions, the proposed client backend modules. The first is to extend Python with C [67]. The second is to use the foreign function library for Python called ctypes [68]. In this document, we present the latter approach since it enable us call the native C functions in a shared library directly, no need to communicate with any JNI-to-C mapping function. Fig. 18 shows an example Python snippet code calling the proposed client backend module.

```
def encryptPHR(phr_path, access_policy):
    "Example of Python code calls the native C function"

    # Load the shared library and define its symbolic name as "backend"
    cdll.LoadLibrary("libPHRapp_User.so")
    backend = CDLL("libPHRapp_User.so")

    # Call the backend module
    if backend.encrypt_phr(phr_path, access_policy) == 1:
        print "encryption succeeded"
    else
        print "encryption failed"

    return
```

Fig. 18. An example of calling the client backend module by Python code

Appendix D

Dependency packages/libraries installation, and software compilation, configuration, and execution

Appendix D1

Dependency packages/libraries installation

- Dependency packages/libraries installation
  - On Ubuntu OS, open Applications>Accessories>Terminal
  - Type a command “sudo su” then press Enter button and type the root password
  - Type a command “apt-get update”
  - Type a command “apt-get install build-essential” and press “Y” to confirm the installation
  - Type a command “apt-get install openjdk-6-jdk” and press “Y” to confirm the installation
  - Type a command “apt-get install mysql-server” and press “Y” to confirm the installation
    - Then the package configuration will be prompted to ask you for defining the password for the MySQL root user; type “bright” as the root password
  - Type a command “apt-get install libmysqlclient-dev” and press “Y” to confirm the installation
  - Download the following packages to the current directory: openssl-1.0.1e.tar.gz, curl-7.27.0.orig.tar.gz, m4-1.4.16.tar.gz, gmp-5.0.2.tar.gz, libpbc_0.5.12.tar.gz, pbc-0.5.12.tar.gz, libbswabe-0.9.tar.gz, and cpabe-0.11.tar.gz
  - Type a command “tar –zxvf openssl-1.0.1e.tar.gz”
  - Type a command “cd openssl-1.0.1e”
  - Type a command “./config”
  - Type a command “make install”
  - Type a command “apt-get install libssl-dev” and press “Y” to confirm the installation

- Type a command “cd ..”
- Type a command “tar –zxvf curl-7.27.0.orig.tar.gz“
- Type a command “cd curl-7.27.0”
- Type a command “./configure --with-ssl=/usr/local/openssl --enable-smtp”
- Type a command “make install”
- Type a command “cd ..”
- Type a command “tar –zxvf m4-1.4.16.tar.gz”
- Type a command “cd m4-1.4.16”
- Type a command “./configure”
- Type a command “make”
- Type a command “make install”
- Type a command “cd ..”
- Type a command “tar –zxvf gmp-5.0.2.tar.gz”
- Type a command “cd gmp-5.0.2”
- Type a command “./configure”
- Type a command “make”
- Type a command “make install”
- Type a command “cd ..”
- Type a command “tar –zxvf libpbc_0.5.12.tar.gz“
- Type a command “cd libpbc”
- Type a command “./configure”
- Type a command “make”
- Type a command “make install”
- Type a command “cd ..”

- Type a command “tar –zxvf pbc-0.5.12.tar.gz“
- Type a command “cd pbc-0.5.12”
- Type a command “./configure”
- Type a command “make”
- Type a command “make install”
- Type a command “cd ..”
- Type a command “tar –zxvf libbswabe-0.9.tar.gz“
- Type a command “cd libbswabe-0.9”
- Type a command “./configure”
- Type a command “make”
- Type a command “make install”
- Type a command “cd ..”
- Type a command “tar –zxvf cpabe-0.11.tar.gz“
- Type a command “cd cpabe-0.11”
- Type a command “./configure”
- Type a command “make”
- Type a command “make install”

Appendix D2

Software compilation, configuration, and execution

- Software compilation, configuration, and execution
  - Download PHRapp-0.30.tar.gz package to the current directory
  - Type a command “tar –zxvf PHRapp-0.30.tar.gz”
  - Configuring the OpenSSL certification
    - Type a command “cd PHRapp-0.30/Certs”
    - Type a command “./cert_gen.sh”; the script will process for a while and then ask you for entering the root password
    - Type a command “cd ..”
  - Creating the PHRapp database
    - Type a command “cd database_scripts”
    - Type a command “./db_setup.sh”
    - Type a command “cd ..”
  - Generating Diffie-Hellman key exchange parameters
    - Type a command “cd DH_params”
    - Type a command “./dh_params_gen.sh”
    - Type a command “cd ..”
  - Generating the Emergency Server key
    - Type a command “cd EmS_key”
    - Type a command “./key_gen.sh”
    - Type a command “cd ..”
  - Compiling the software for both server and client sides (Option 1)
    - Type a command “cd bin”
    - Type a command “make”
  - Compiling the software for server side only (Option 2)

- Type a command “cd bin”
- Type a command “make server_main”

- Compiling the software for client side only (Option 3)
  - Type a command “cd bin”
  - Type a command “make client_main”
- Executing the server applications
  - In the “PHRapp-0.30” directory
  - Execute the User Authority application: type a command “./UA_start.sh”
  - Execute the Audit Server application: type a command “./AS_start.sh”
  - Execute the PHR Server application: type a command “./PHR_server_start.sh”
  - Execute the Emergency Server application: type a command “./EmS_start.sh” (Only need when you would like to enable the software to support the emergency access control feature)
  - Execute the Emergency-Staff Authority application: type a command “./ESA_start.sh” (Only need when you would like to enable the software to support the emergency access control feature)
- Executing the client application
  - Execute the PHR client application for a general PHR user/admin: type a command “./PHR_client_start.sh”
  - Execute the PHR client application for an emergency staff/admin: type a command “./EmU_client_start.sh”
  - Default administrator (username: “admin” and password: “bright23”)