Secure Spontaneous Emergency Access to Personal Health Record

Sye Loong Keoh, Muhammad Asim, Sandeep S. Kumar and Peter Lenoir
Department of Information & System Security
Philips Research Europe, High Tech Campus 34
5656 AE, Eindhoven, The Netherlands
{sye.loong.keoh, muhammad.asim, sandeep.kumar, peter.lenoir}@philips.com

ABSTRACT

We propose a system which enables access to the user's Personal Health Record (PHR) in the event of emergency. The access typically occurs in an ad-hoc and spontaneous manner and the user is usually unconscious, hence rendering the unavailability of the user's password to access the PHR. The proposed system includes a smart card carried by the user at all time and it is personalized with a pseudo secret, an URL to the PHR Server, a secret key shared with the PHR Server and a number of redemption tokens generated using a hash chain. In each emergency session, a one-time use redemption token is issued by the smart card, allowing the emergency doctor to retrieve the user's PHR upon successful authentication of his credentials and validation of the redemption token. The server returns the PHR encrypted with a one-time session key which can only be decrypted by the emergency doctor. The devised interaction protocol to facilitate emergency access to the user's PHR is secure and efficient.

Categories and Subject Descriptors

C.3 [**Special-Purpose and Application-Based Systems**]: Smartcards; H.4 [**Information Systems Applications**]: Miscellaneous

General Terms

Design, Management, Security

Keywords

Authentication, access control, emergency access, patient health record

1. INTRODUCTION

There is a growing need for a systematic management of health and medical data. Diagnostic reports from different hospitals or clinics, prescriptions, medication consumption logs, etc must be kept securely and access to such medical data must be convenient to the user and her physician during treatment. The health data management application is particularly useful for patients who suffer from chronic diseases and elderly who tend to be forgetful and requires assistance in managing their health records. The concept of Personal Health Record (PHR) [5, 7] that can be solely managed by the users themselves is ideal for storing and controlling access to the users' health data. Electronic Medical Records (EMR) and Electronic Health Records (EHR) maintained by the healthcare providers can be imported into the user's PHR, allowing ubiquitous access to the user's health data whenever an Internet connection is available.

Due to the sensitivity and confidential nature of the PHR, access to the PHR is restricted to the users themselves as they typically do not want to share their medical records with others. However, the elderly may choose to delegate the task of managing their medical records by granting full (or subset of) permission to their family member who is more competent. Access control policies can be specified *a priori* to grant access to the respective entity to the PHR using Role-Based Access Control (RBAC) [15], password-based access control, specifying an access control list, etc. However, in an emergency situation, such pre-defined access control policies fall short because there is no policy defined *a priori* that would allow any emergency doctor and ambulance team to access the user's PHR when providing emergency treatment. If the user is unconscious, this implies that the password to access the PHR cannot be obtained either. Existing services [1, 7] shows that it is beneficial if some background information about the user's health condition can be provided while treating the user in the event of emergency. In the US, there were 108,000 deaths last year from preventable medical errors, and 93% of those errors could have been avoided if the doctor had needed information about the patient when first being treated [1].

One of the main security challenges in the emergency situation is to prevent the misuse of this emergency trigger as such break-the-glass access to the user's PHR can be exploited unlawfully. Essentially, it is difficult for the PHR Server to distinguish between a genuine emergency situation and a malicious attempt to access the PHR because in both scenarios, the user is deemed unavailable. In this paper, we propose a system which enables secure and efficient access to the user's PHR in an emergency situation. The user carries a smart card that has been personalized with a pseudo secret, an URL to the PHR Server, a secret key shared with the PHR Server and a number of redemption

tokens generated using a hash chain. In each emergency session, a one-time use redemption token is issued by the smart card, allowing the emergency doctor's device to retrieve the user's PHR upon successful authentication of his credentials and validation of the redemption token. The server returns the PHR encrypted with a one-time session key which can only be decrypted by the emergency doctor.

This paper is organised as follows: Section 2 discusses various related work. Section 3 presents the threat model, while Section 4 introduces our approach to provide one-time secure emergency access to PHR. Section 5 presents some discussions and we conclude the paper in Section 6.

2. RELATED WORK

The current practise for allowing emergency access in the hospital is by requiring the emergency doctor to send an access request indicating an emergency override [13]. Access is granted if the requesting entity has the appropriate credentials, i.e., the requesting entity is a certified medical doctor. Access is logged and post-access auditing is performed to determine the legitimacy of the access. However, such mechanism is ineffective as damage to the user's health data has been done if the emergency override was malicious. Essentially, auditing cannot detect malicious access to the PHR and only serve as deterrent.

Akteonline [17] is a secure data storage service and it provides flexible access management functions to EHR. The owner of the EHR is able to specify one-time access to parts of his records by using TAN which is essentially a transaction number (or secret) that grants permissions to an entity to access the EHR. In emergency situation, a read-only access can be configured to allow emergency doctors to access *emergency subset* of the patient's EHR. The web address, username and the emergency TAN (transaction number) are printed on a small wallet card to be carried at all time by the user. With the possession of this small wallet card, the emergency doctor can access the user's contact information, information about allergies, confirmed diseases and a list of actual medications when treating the patient in an emergency situation. Unfortunately, there are no guards against theft in that whoever finds or steals the card can access the PHR. Renewal of emergency TAN must be done via *Akteonline* and the card must be replaced with the new one.

AccessMyRecords [1], ICER-2-GO [6] provide a service that enables the doctors, hospitals and emergency responders to have access to the user's medical information. These typically comply with the Break-Glass approach [2]. A medical card (e.g., AccessID card) is issued to the user who has registered with the service. It is noted that the user is expected to carry this card at all times. Typically the medical card contains the user's name, a passcode and an URL to access the medical record printed on it. In emergency situation where the user is unconscious, the medical personnel can look for the user's medical card and subsequently log into the service portal, using the user's name and passcode found on the medical card. A read-only summary (or subset) of the user's medical and legal records will be granted access to the emergency personnel. Essentially, the ease of access in these services is traded off with low security, as anyone gaining possession of the medical card will gain access to the user name, passcode and eventually read-only access to the user's PHR. The fact that passcode is printed in clear on the card does not provide any security.

An alternative to storing the user's PHR in the cloud is to keep it in a small USB-based PHR device that allows the user to easily transport their personal health information. The Personal HealthKey (CapMed, Newtown, PA) and the E-HealthKEY (MedicAlert, Turlock, CA) [18] are examples of such device that use flash memory and a USB port to store a variety of health information. Both devices offers password security and encryption, and allow the user to decide which information on the USB device to share. These devices also have an emergency function that allows responders to access a subset of medical information without a password. An analysis by [18] reveals that the security protection is weak because instead of encrypting their contents with the password chosen by the user, the devices store the user's password as a string in the database, and then encrypt that database with a common password fixed by the manufacturer, which was the same across all devices. Consequently, this enables the device manufacturer to have access to all content in the devices.

Huda *et al* [12] introduces a privacy-aware protocol for handling access to the patient-controlled PHR in the emergency situations. The user carries a health IC cards and it contains an emergency access module. The emergency access module has a dedicated rewritable memory portion for storing emergency access digital pseudonyms and emergency access token (EAT). Upon successful authentication of the emergency doctor, the pseudonyms and EAT are forwarded to the P^3HR [11] server to access the user's PHR. The EAT can only be used one time by a doctor, this is implemented by logging the accessing doctor's identification information in the health IC card's dedicated memory space. This scheme relies on the information in the smart card's dedicated memory space to prevent replay attacks, it is advocated that the access log should be maintained by the P^3HR server and request to access the PHR can be denied if the doctor is found to have previously accessed the PHR. Attacks on the smart card's memory space can be launched to remove the last accessing doctor's identification information.

3. THREAT MODEL

This section describes the threat model of the proposed system:

- *Misuse of break-the-glass provision*: Break-the-glass provision is typically used to enable healthcare workers to gain access to the user's PHR in the event of emergency, even though they are not allowed access according to the patient's privacy policy. However, such provision could be misused and exploited, resulting in the violation of the user's privacy. It is important to determine the genuinity of the emergency before access can be granted.
- *Replay of PHR Redemption Token*: An attacker may try to re-use the redemption token from the previous emergency sessions in order to get access to the PHR of the user afterward. Thus, the redemption token must be of one-time use.

- *Denial-of-Service*: In the proposed system, the smart card contains limited number of tokens. An attacker could query the smart card multiple times for the redemption tokens and then use them at a later stage. This also renders the smart card unusable in the event of emergency in the future. This attack can be prevented by configuring the smart card such that it would not reveal any new redemption token if the previous emergency access session is not completed.

- *Theft*: An attacker could also steal the smart card and use it to access the user's PHR. Authentication and access control means must be in placed, and the system must be able to determine whether the user is really in the state of emergency.

4. ONE-TIME SECURE EMERGENCY ACCESS

In our system, we assume that the user carries with him a smart card. It must be provided or in possession by the medical personnel in order to trigger the emergency access. In Germany, a smart card [4] is introduced which users carry with them and it contains the important health related information; Such smart card could potentially be used for the functionality we are proposing in this paper. This smart card should be treated like a credit card in that the information it contains must be kept to the user herself. In case of emergency and that the user is unconscious, the doctor can get hold of the smart card and use the information to request a one-time emergency access to the PHR Server. In case that the smart card is stolen, the user must report loss in order to disable any emergency access to the PHR.

As a countermeasure against theft, the possession of the smart card does not necessarily mean that the requesting entity has emergency access to the user's PHR. Policies must be defined in the PHR Server to only allow certified doctors or medical personnel who can provide information from the smart card to trigger emergency access. Similarly, the doctor who does not have the information from the smart card is not authorized to initiate the emergency access.

A one-time session key for use by the emergency doctor to access the user's PHR is generated between the PHR server and the doctor. This ensures that access to the PHR is only valid for one single session. Additionally, as the PHR is encrypted using the one-time session key, this provides confidentiality to the user's PHR and it also enables the emergency doctor to ascertain the authenticity of the PHR received from the PHR server.

Figure 1 illustrates an overview of the interaction between the user, PHR Server and emergency doctor in order to facilitate secure spontaneous access to the user's PHR in an emergency situation.

4.1 Initial Personalisation of Smart Card

The user first signs up for a PHR account in which a smart card is personalized with a pseudo-secret in clear form that is used to uniquely identify the user's PHR, an URL to the PHR Server, a secret key shared with the PHR Server and a number of encrypted redemption tokens.

4.1.1 Pseudo-secret

The PHR server first chooses a random number K_1 as the first key of the hash chain [14]. A hash function is applied on the key to compute the next key on the chain. This process is repeated for $n-1$ times (i.e., the number of supported emergency sessions) to generate a hash chain as follows:

$$K_1 \rightarrow K_2{=}\mathrm{H}[K_1] \rightarrow K_3{=}\mathrm{H}[K_2] \rightarrow ... \rightarrow K_n{=}\mathrm{H}[K_{n-1}]$$

The hash chain has a one-way property in that given K_n, it is computationally infeasible for an attacker to derive K_{n-1}. The hash chain is unique to each individual user and it serves as an identifier for the PHR of the user. The key in the hash chain is used to determine the authenticity of the user's PHR. The hash chain is used in reverse order and K_n is denoted as the pseudo-secret of the user and personalised in the user's smart card.

4.1.2 Redemption Tokens

In addition to the hash chain, the PHR server generates $n-1$ redemption tokens $rk_{1..n-1}$. The server first generates $n-1$ independent identifiers, $X_{1..n-1}$ using a random number generator and then encrypts them with keys from the hash chain, K_i in the following way to create the redemption tokens:

$rk_{n-1}{=}E_{K_{n-1}}(X_{n-1})$
$rk_{n-2}{=}E_{K_{n-2}}(X_{n-2})$
...
$rk_1{=}E_{K_1}(X_1)$

The redemption tokens are used in descending order where rk_{n-1} is used first. These tokens are only known between the user's smart card and the PHR Server, the emergency doctor who uses the redemption token cannot decrypt the token, hence preventing replay prior to obtaining a permission to access the PHR. It is also assume that the PHR Server shares a secret key K_{ID} with each user's smart card, and it is stored in the secure memory location of the smart card. As each token is encrypted with a key from the hash chain and that the key is only known to the PHR Server, it is not possible for an attacker to neither create a new token nor modify the token.

In essence, the smart card contains the pseudo-secret key, K_n in clear, an URL to access the PHR Server and a list of (encrypted) redemption tokens and a secret key shared with the PHR Server:

Pseudo-secret: K_n
URL: *www.personalPHR.com/emergency*
Redemption Tokens: rk_{n-1}, rk_{n-2}, ..., rk_2, rk_1
Secret-key: K_{ID}

The user is assumed to carry the smart card at all times. This personalisation process is shown as *Step 1* in Figure 1.

4.2 Policy Specification

The PHR Server is responsible for managing access to the user's PHR. There are two types of policies that can be specified at the PHR Server, namely the *access control* policies

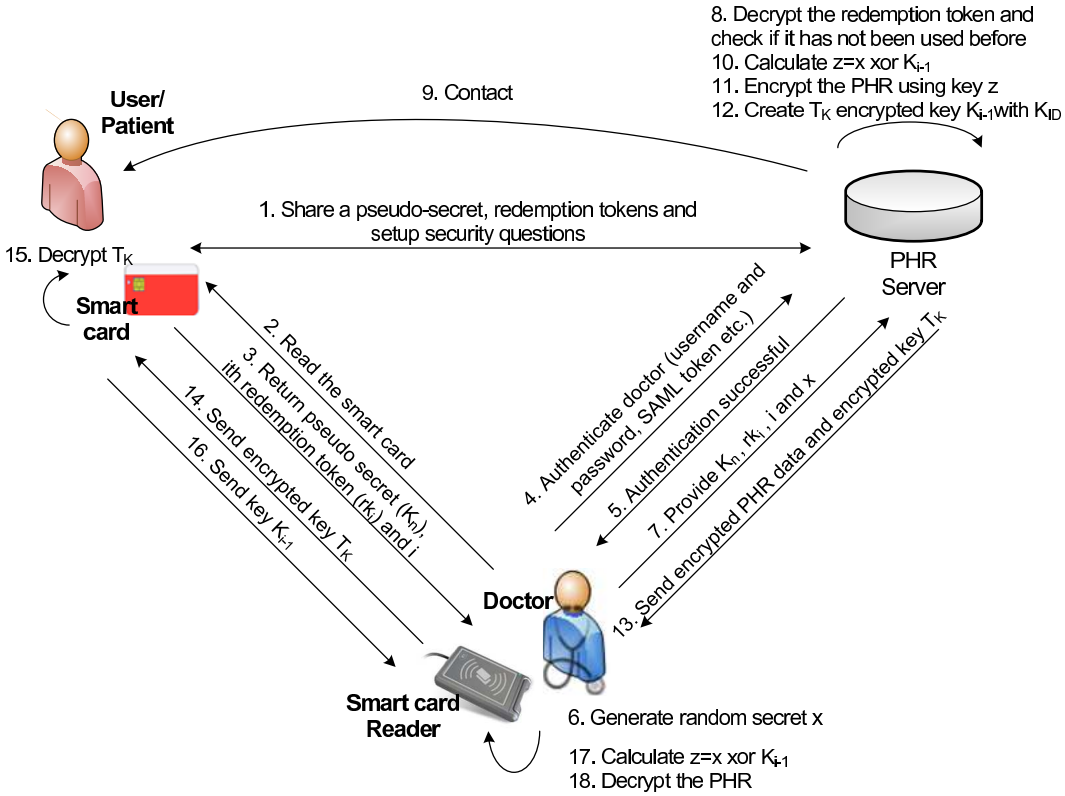

Figure 1: Interaction protocol to facilitate secure emergency access to PHR.

and *obligation* policies. These policies are typically defined to facilitate access to the user's PHR in the normal day-to-day operation, e.g., granting access to family members to access the user's PHR, allowing the user to update her health record, import medical histories from EMR and EHR, etc. It is conceivable that additional policies can be specified to adapt to the emergency situation. In this section, we provide a few examples of policies that could be deployed to deal with emergency access.

4.2.1 Obligation policy

The *obligation policy* is an *event-condition-action* rule that defines the actions that need to be performed as part of the system management when certain event occurs and the contextual conditions such as time, location, etc are true.

Figure 2 shows an example of obligation policy to manage the access to the PHR in the emergency situation. When the PHR Server is notified that an event indicating an emergency request to access the user's PHR has occurred, the PHR server *must* place an automated call to the user to confirm whether this is an emergency situation and subsequently invokes an action to verify the access request.

4.2.2 Access Control Policy

The *access control policy* grants permission to entities requesting access to the resources in the PHR Server based on the possessed credentials or attributes such as the roles assigned to the entity.

As an example, Figure 3 shows an access control policy that

```
On emergencyEvent(credential, token, identifier)
Do
    user = retrieve(identifier)
    If (call(user.telephone) == EMERGENCY)
        requestAccess(credential, token, user)
    Else
        abort()
```

Figure 2: Example of *obligation* policy

verifies the access request and determines whether the emergency doctor (*subject*) is allowed to access the subset of the user's PHR (*target*). The *verify(credential)* method in the policy triggers an action on the PHR Server to verify the credential of the emergency doctor, e.g., verifying the attribute certificate presented by the doctor to prove that he is a certified doctor [10]. The *verify(token, user.identifier)* method ensures that the redemption token received is fresh and have not been tampered with.

```
auth+ requestAccess(credential, token, user)
    subject doctor = verify(credential)
    target PHR = user.retrieveEmergencyPHR()
    action PHR.send(doctor)
    when verify(token, user.identifier) == SUCCESS
```

Figure 3: Example of *access control* policy

Both the *access control* and *obligation* policies can be specified and deployed using any of the suitable policy specifi-

cation languages and its enforcement architecture such as XACML [3], SAML [8], or Ponder2 [16], etc.

4.3 Triggering Emergency Access

In the event of emergency in which the user is unconscious and requires urgent medical treatment, the emergency doctor and his *Emergency Medical Team* (EMT) first locate the user's smart card and use a portable device to retrieve information from the smart card.

As shown in *Step 3* in Figure 1, the smart card returns the pseudo-secret K_n and the ith redemption token, $rk_i = E_{K_{i-1}}(X_i)$. The pseudo-secret uniquely identifies the user's PHR, while the redemption token grants emergency access to the doctor. In the next emergency session, a new redemption token, $rk_{i-1} = E_{K_{i-2}}(X_{i-1})$ is issued along with the user's pseudo-secret.

Step 4 and 5 — Authentication: Once the doctor's portable device has successfully obtained the K_n, rk_i and i (which indicates the position of the current key in the hash chain) from the smart card, it sends a request to the PHR server indicating an emergency access to the user's PHR is needed. Together with the access request, an authentication means must be provided, e.g., an SPKI attribute certificate, X.509v3 Certificate, SAML Token, etc. The PHR Server authenticates the requesting entity and returns either authentication failure or successful message.

Step 6: Upon successful authentication, the doctor generates a random number, x locally to be used to compute a one-time session key to encrypt the PHR.

Step 7 - 9 — Evaluation of policies: An emergency access request containing the pseudo-secret K_n, redemption token rk_i, iterator i, random number x, signed using the doctor's credential and encrypted using the PHR Server's public-key is sent to the PHR Server. This request is captured as an event, thus triggering the execution of the *obligation policy* to initiate an automated call to contact the user to determine whether the user is in emergency situation. As indicated in the policy, the emergency access request will be immediately aborted if the user indicates otherwise. Subsequently the evaluation of the *access control policy* is triggered to determine whether permission can be granted. Based on the pseudo-secret K_n, the PHR server locates the user's corresponding hash chain and advances the hash chain backward to obtain K_{i-1}. The redemption token is then decrypted using K_{i-1}. It is also important to ensure that the token has never been used before, otherwise the PHR server aborts the session and will not release the user's PHR.

Step 10 and 11 — One-time session key establishment: When access has been granted by the policy, and the emergency situation has been ascertained, the PHR server generates a one-time secret key $z = x \oplus K_{i-1}$. The PHR server then uses the secret key, z to encrypt the user's PHR.

Step 12 - 13 — Access to the PHR: The encrypted PHR is sent to the doctor's portable device, along with the key $T_k = (K_{i-1})_{K_{ID}}$. The key K_{i-1} is encrypted with K_{ID}, in which only the user's smart card has the capability of decrypting it. This means that the doctor's device has access to neither the K_{i-1} nor K_{ID} and hence it is not yet able to compute the one-time secret.

Step 14 - 18: An additional step to interact with the smart card is necessary by sending the received T_k to the smart card for decryption. If successful, the key K_{i-1} decrypted from T_k is revealed to the doctor's device, thus enabling it to compute the one-time secret, z. However, prior to that, the smart card must first ensure that K_{i-1} is authentic and it corresponds to the identity of the user, thus ensuring that the correct PHR is retrieved. This is realized by applying the hash function to K_{i-1} and the resulting hash values repeatedly until it arrives at K_n which is essentially the pseudo-secret of the user. This serves as a way to authenticate the PHR Server as only the PHR Server knows the entire hash chain. The doctor's device can also perform the routine in order to determine the authenticity of the PHR.

$$\mathrm{H}[K_{i-1}] \rightarrow ... \rightarrow \mathrm{H}[K_{n-1}] = K_n$$

Finally, the doctor decrypts the PHR using the one time secret z. In this case, the user's original password or secret to access the PHR is not revealed to anyone due to the emergency needs.

4.4 Audit Log

Since the emergency access was performed without getting direct user consent, it is important that the PHR server logs all the emergency access information and report to the user as soon as possible. Such audit log not only guarantees non-repudiation in which the emergency doctor cannot deny requesting access to the user's PHR in emergency situations, it also serves as an important piece of information to the intrusion detection system to detect any malicious attempt by attackers to exploit the emergency access trigger.

5. DISCUSSION

It is the aim of this paper to ensure that the emergency doctor only has one-time access to user's PHR in an emergency situation, hence preventing any replay attacks. The pseudo-secret and the redemption token obtained from the user's smart card cannot be replayed because the PHR Server keeps track of the used tokens. In order for the same emergency doctor to access the user's PHR again, the doctor must retrieve a new redemption token from the smart card. Additionally, the PHR Server which keeps an audit log can detect whether a doctor is attempting to access the user's PHR record multiple times and can therefore deny the access request.

The proposed protocol is designed to prevent *multiple times querying attacks* in which the attackers attempt to query the smart card multiple times, while keeping the retrieved redemption tokens for use at a later time. As the protocol has two interactions with the user's smart card, the intermediary (i.e., the doctor's device) cannot gain access to the user's PHR without possessing the smart card in hand. Although the attacker could send the unused redemption token to retrieve the PHR, it does not have the capability to decrypt it because in order to compute the session key z, it must obtain the key share K_{i-1} from the user's smart card

as shown in Step 12 - 18. Additionally, the smart card is configured not to reveal any new redemption token if the previous handshake is not completed, thus preventing the attacker to collect redemption tokens for later use. This also serves as a guard against potential Denial-of-Service attacks to render the smart card unusable in the emergency situation.

Key escrow [9] and key recovery mechanisms that are used by the government agencies to get access to encrypted communication could be adapted for use in the context of emergency situation as key escrow is typically used for key backup, enabling the users to recover lost secret-keys. In general, the key escrow scheme could be used to allow emergency doctor to recover the user's password for accessing the PHR in case of emergency. However, a trusted third party (TTP) must be nominated to act as the escrow agent, which in the case of emergency can reveal the password or key for accessing the patient's PHR to the requesting entity. This introduces an additional entity that the user and the doctor have to trust. Additionally, it has the disadvantage in that the user's secret is broken for use in the emergency situation and this does not provide backward secrecy to the information encrypted using the same secret-key in the past. Furthermore, users generally re-use the same password for many different services in the cloud, and when this secret is broken or revealed, this also means that access to the user's other protected information is made possible using the broken password.

6. CONCLUSIONS AND FUTURE WORK

Without any security provisioning, emergency access can be easily exploited and thus allowing non-authorized parties to have access to the user's PHR using this trigger. In this paper, we have proposed a novel scheme to ensure the user's privacy and safety, while enabling a doctor to obtain the user's medical history and allergies information when providing medical treatment to the user in an emergency situation.

Specifically, we have introduced: (i) secret binding between the user's smart card and the PHR server using keys from a hash chain to encrypt the redemption tokens. This guarantees one-time use of the redemption token in an emergency session and enables the doctor's device to authenticate the PHR Server. (ii) Use of policies to provide security management in the emergency situation.

As the emergency doctor has access to the PHR (i.e., the PHR can be decrypted using the one-time session key), nothing prevents him from further disseminating the PHR to another third party, although it is advocated that he must not do so without user's consent. In some cases, it is important to further restrict the access to the PHR based on time in an emergency session. In this respect, consent management and digital rights management (DRM) technologies can be employed in the future to enable the users to have a greater control over the accessibility of their PHR.

7. REFERENCES

[1] AccessMyRecord - In Case of Emergency (AMR-ICE), available at http://www.accessmyrecords.com.

[2] Break Glass - An approach to Granting Emergency Access to Healthcare System, available at http://www.nema.org/medical/spc.

[3] eXtensible Access Control Markup Language (XACML), version 2.0, available at http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf.

[4] German Health Card, available at http://www.smartcardalliance.org/resources/pdf/German_Health_Card.pdf.

[5] Google health, available at http://www.google.com/heath.

[6] ICER-2-GO, available at http://www.icer-2-go.com.

[7] Microsoft healthvault, available at http://www.healthvault.com.

[8] Security Assertion Markut Language (SAML), version 2.0, available at http://saml.xml.org/saml-specifications.

[9] H. Abelson, R. Anderson, S. Bellovin, J. Benaloh, M. Blaze, W. Diffie, J. Gilmore, P. Neumann, R. Rivest, J. Schiller, and B. Schneier. The risks of key recovery, key escrow, and trusted third party encryption. *World Wide Web Journal*, 2(3), 1998.

[10] A. Herzberg, Y. Mass, J. Mihaeli, D. Naor, and Y. Ravid. Access control meets public key infrastructure, or: Assigning roles to strangers. In *IEEE Symposium on Security and Privacy*, pages 2–14, 2000.

[11] M. N. Huda, N. Sonehara, and S. Yamada. A Privacy Management Architecture for Patient-Controlled Personal Health Record System. In *Proc. International Conference on Network Applications, Protocols and Services (NetApps), Nov 2008*, 2008.

[12] M. N. Huda, S. Yamada, and N. Sonehara. Privacy-aware access to Patient-controlled Personal Health Records in Emergency Situations. In *Proc. 3rd International Conference on Pervasive Computing Technologies for Healthcare, London, April 2009.* IEEE, 2009.

[13] J. Kunzi, P. Koster, and M. Petkovic. Emergency Access to Protected Health Records. In *Proc. 22nd International Conference of the European Federation for Medical Informatics (MIE)*, September 2009.

[14] L. Lamport. Password Authentication with Insecure Communication. *Communications of the ACM*, 24(11):770–772, 1981.

[15] R. Sandhu and E. Coyne. Role-based Access Control Models. *IEEE Computer*, 29(8):38–47, 1996.

[16] K. Twidle, E. Lupu, M. Sloman, and N. Dulay. Ponder2: A Policy System for Autonomous Pervasive Environments. In *Proc. 5th International Conference on Autonomic and Autonomous Systems (ICAS), April 2009.* IEEE, 2009.

[17] F. K. Uckert and H.-U. Prokosch. Implementing Security and Access Control Mechanisms for An Electronic Healthcare Record. In *Proc. Annual Symposium of American Medical Informatics Association (AMIA)*, pages 825 – 829, 2002.

[18] A. Wright and D. F. Sittig. Encryption Characteristics of Two USB-based Personal Record Devices. *Journal of the American Medical Informatics Association*, 14(4):397–399, 2007.