Cryptographically Enforced Personalized Role-Based Access Control

Milan Petković, Claudine Conrado, and Malik Hammoutène

Philips Research, Information & System Security
High Tech Campus 34 (MS 61), 5656 AE Eindhoven, The Netherlands
{milan.petkovic, claudine.conrado}@philips.com

Abstract. The present paper addresses privacy and security enhancements to a basic role-based access control system. The contribution is twofold. First, the paper presents an approach to personalized access control, i.e. a combination of role-based access control and user-managed access control. Second, the proposed access control approach is cryptographically enforced and an efficient key management method for the personalized role-based access control is described. The proposed solutions are discussed in the context of a system architecture for secure management of Electronic Health Records.

1 Introduction

Data security and privacy issues are traditionally important in the medical domain. However, recent developments such as digitization of medical records, the creation of national health record databases, and extramural applications in the personal health care domain, pose new challenges towards the protection of medical data. In contrast to other domains, such as financial, which can absorb the cost of the abuse of the system (e.g. credit card fraud), healthcare cannot. Once sensitive information about an individual's health problems is uncovered and social damage is done, there is no way to revoke the information or to restitute the individual.

In addition to this, the medical field has some other characteristics, such as data can be accessed only by authorized persons that are assigned a certain role (general practitioner, nurse, etc). To support such requirements, the access control system bases the access decisions on the role that a particular user is associated with. Access rights are grouped by roles so that an individual acting in a specific role may perform only particular sets of operations that are assigned to that role. This approach, called Role-Based Access Control (RBAC) [1], is very important in the healthcare domain, because in addition to practice where it is used, it is also addressed by legislation (e.g. the Health Insurance Portability and Accountability Act (HIPAA) [2]).

Another trend in the healthcare sector towards personal, user-focused healthcare demands more patient-involvement at all levels of healthcare. Patients are taking a more active role in their own healthcare management, which includes obtaining disease information, discussing with doctors, tracking symptoms and managing their illnesses. Consequently, patients are also more involved in keeping and managing important medical documents. This also has some backing in the legislation such as

Please use the following format when citing this chapter:
Author(s) [insert Last name, First-name initial(s)], 2006, in IFIP International Federation for Information Processing, Volume 201, Security and Privacy in Dynamic Environments, eds. Fischer-Hubner, S., Rannenberg, K., Yngstrom, L., Lindskog, S., (Boston: Springer), pp. [insert page numbers].

HIPAA [2] which gives patients more rights with respect to Electronic Health Records (EHRs)[1] [3]. For example, patients have the right to have more influence with respect to access control to their health records. Consequently, a patient could request additional restrictions on the disclosure of their records to certain individuals involved in his care that otherwise would be permitted (based on role-based access control governed by the care institution). If the care provider agrees with his request, it is required to comply and enforce the agreement. This poses additional requirements on access control. Namely, in addition to access based on roles, individual restrictions have to be taken into account. Furthermore, a patient should be able to allow his family or friends to access his personal record. Therefore, there is a clear need for a combination of role-based access control and user-managed access control, which is addressed in this paper as *personalized* role-based access control.

While access control mechanisms do play a crucial role in EHR security, they have also some limitations. One of these limitations, which is addressed in this paper, concerns the possibility of anyone bypassing the access control mechanism and having direct access via the file system to the EHR database where data is stored in plain text. A straightforward solution to this problem is to enhance access control with data encryption, providing in this way a "cryptographically-enforced" access control mechanism. This is an important feature since it also supports the secure *off-line* usage of EHR, when the data is no longer protected by the access control mechanism, thus increasing data availability. The problem of key management for access control has already been addressed in the literature. However, most of the approaches focus on hierarchical key management (see for example [4, 5]) and do not dynamically handle role-based access control policies which may include patient-defined policies. Therefore, this paper describes an efficient key management method within the context of a *cryptographically-enforced* personalized RBAC system.

The rest of the paper is organized as follows. The basic concept of personalized role-based access control is described in the next section. Section 3 enhances the security of the proposed access control method by enforcing it cryptographically. Furthermore, this section presents an efficient key management method in the context of the personalized RBAC. Section 4 presents the architecture of a secure EHR management system that deploys the proposed solutions. Finally, Section 5 draws conclusions.

2 Personalized Role-based Access Control

In a classical RBAC model, data access decisions are based on a user's role in the system. In the case of EHR, a user identified as a general practitioner (GP), for instance, may be able to see the data of his patients without restriction, as long as he stays within the scope of a GP's rights. It may occur, however, that a patient uses his rights given by legislation to deny access to a specific GP who has by default rights to

[1] According to the Healthcare Information and Management Systems Society, an Electronic Health Record (EHR) is "a secure, real-time, point-of-care, patient-centric information resource for clinicians". The EHR supports the decision-making process by providing access to patient health record information where and when needed.

access his data. A simple solution to this problem is to include this doctor in a "black list". To access the patient's record, the practitioner is checked if he belongs to the black list. However, this approach is not scalable considering the medical system with thousands of users and patients who may want to deny access to many of them. Moreover, some patients might deny access to all doctors except one. Therefore, to deal with this issue, an approach is proposed in this section that uses overriding policies, called personalized policies, and an exception list, where the rights and restrictions are re-defined per user and per concerned data.

The method proposed is based on two main components, namely, the policy repository and the exception list. The policy repository contains all policies in the system. Initially, it contains only *default* policies. A default policy defines default access rights for all existing roles in the system (e.g., general practitioner, emergency doctor, etc.). At this point, there is no patient influence on the definition of policies. When a patient wants to add some change to a default policy, an exception list is created. The exception list is defined as a list of all the users (care providers) that are in conflict with the policy which is applied. Typically, the exception list reflects patient's wishes by restricting or adding rights to any of the patient's pieces of data for each of the listed users. A piece of data is considered here as any subset of the EHR that may have its specific access rules.

When a patient creates an exception list, the system decides whether it is necessary or not to replace the default policy by a personalized policy. The idea is to keep the exception list as light as possible. If the same restrictions specified in the exception list concern the majority of the users, the default policy is overridden by a personalized policy. A personalized policy is a set of rules, an identifier for the rule-combining algorithm and (optionally) a set of obligations [6]. Rules, rule-combining algorithm and obligations are themselves combinations of resources, subject and actions that find their origin in a set of predefined atomic policy blocks. An atomic policy block is a logical unit defined as the smallest element in eXtensible Access Control Markup Language (XACML) [7] that is executed completely or not at all. By combining different atomic policy blocks we obtain complete personalized policies. When created, personalized policies are dominant over default policies. Moreover, the personalized policies themselves could be changed, as new restrictions or grants may be introduced, therefore these policies may be highly dynamic.

An example is given below for a patient P. The default atomic policy block in his case is that all GPs (A, B, C, D and E) can initially have access to the piece of his EHR data concerning sexual diseases (D_1). If patient P now wants to restrict access to D_1 to all GPs, except A, instead of listing B, C, D and E in the black list, the old default policy for data pieces D_1 is replaced by a personalized atomic policy block. This is now the policy which is default, so in order to allow GP A to access data piece D_1, an exception list is then created:

Default Atomic Policy Block:	all GPs have access to D_1 of EHR of patient P
Personalized Atomic Policy Block:	no GP has access to D_1 of EHR of patient P
Exception List:	GP A: $+D_1$

The symbol "+" above indicates that the user (GP A) has granted rights on the concerned data (D_1). If the patient wants to do the same for data piece D_2, a corresponding personalized atomic policy block will replace the old one and the exception list will be extended with a new entry (+D_2).

With the policy repository and the exception list dynamically updated as above, when a user requests data of a given patient, the system must check the policy repository for that patient as well as whether an exception list exists which includes that user/data-object. In this way, patient-defined access policies may be handled more efficiently in the system. Instead of having up to n-1entries in the list (where n is the total number of doctors), the exception list has now a maximum of n/2 entries.

When there is a change in the subject-role assignment (e.g. a new user is added as a member of a role) related to the patient-defined exception, the patient could be informed about this change. In this case, he must explicitly requests this action from the system.

To further optimize the solution, we can define several ways of grouping the patients. Indeed, most patients could have the same policies (probably the majority of them will not even use personalized policies, but only the default ones). Therefore, they can be joined based on their personalized policies and exception list, or only based on their personalized policies. Figure 1a shows an example where patients have the same personalized policies (that can be NULL), as well as the same exception lists. Figure 1b shows an example where patients have the same personalized policies, but different exception lists.

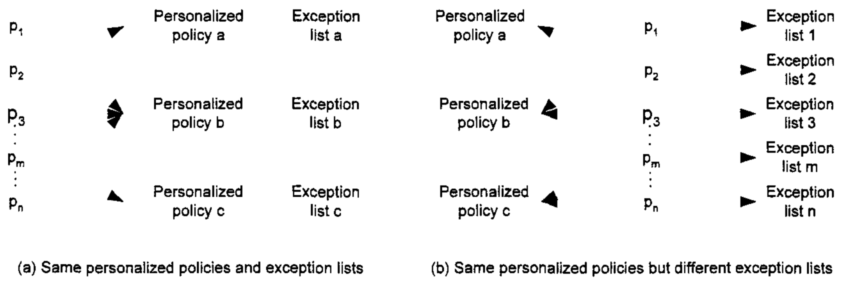

Fig. 1. Different ways of grouping.

3 Cryptographic enforcement and efficient key management for Personalized RBAC

3.1 Cryptographically-protected EHR

In order to enhance the security of the proposed RBAC system and to allow the secure off-line usage of data, access to data is cryptographically enforced. This means that the EHR is encrypted at all times and data access is achieved via access to the decryption keys. More specifically, encryption of EHR is implemented by having a different key to encrypt each data piece. Access control can then be enforced by a key hierarchy scheme where the "data key" (the fixed and unique key that encrypts a piece of data) is encrypted by "access keys". Access keys may be, for instance, the public keys of the parties that are allowed access to the data. In this way, if these parties do change, only the key encrypting the data, and not the data itself, has to be re-encrypted with the new rightful parties' access keys. It is often the case that different pieces of data have the same access policies. In this case, the number of needed keys can be significantly reduced because different data pieces with the same access policy can be grouped and encrypted with the same data key. However, in case the access policy of one or more of these data pieces changes over time, they must be re-encrypted with different data keys.

In the proposed approach, the EHR of a patient is composed not only of the encrypted health data, but also of key material needed to decrypt the health data, i.e. the data keys encrypted with the authorized users' access keys. This is depicted in Figure 2 which shows an EHR record with two fields: a data field including the data encrypted with key *sym* and a key field including the encryptions of *sym* with the access keys *a*, *b*, and *c*. The data field is typically static, whereas the key field may be frequently updated, in accordance with the system access policies.

$[\mathbf{Data}]_{sym}$	$[\mathbf{sym}]_a\,[\mathbf{sym}]_b\,[\mathbf{sym}]_c$

Fig. 2. An EHR record.

Given the structure above, when a user downloads any EHR to a portable device, he also downloads the encrypted data keys. Therefore, he is able to access the data if he has the *access keys* that decrypt the *data keys* that finally decrypt the data.

The access keys for an EHR record together with the key material enforce now the access policies for that record. Access keys may be, e.g., the public keys of the individuals who can access the data. However, because of the scalability problem, the role-based approach is more suitable. Indeed, defining the access keys as *role* keys greatly improves the system's efficiency by facilitating key management (typically there are much more individuals than the number of roles in a given system). On the other hand, the approach using role keys does not efficiently handle patient-defined restrictions, as defined in the previous section. This is due to the fact that individuals, and not only roles, must be addressable in this case. To support this requirement from the legislation as well as its efficient cryptographic enforcement, we propose an efficient key management scheme based on Broadcast Encryption.

3.2 Efficient key management

A Broadcast Encryption (BE) approach can be used to efficiently manage encryption keys in the cryptographically-enforced personalized RBAC (CEPRBAC) system. In brief, a BE scheme allows a dynamic group of entities to establish a common secret by receiving broadcasted messages sent by a central authority without the need for a return communication channel. In common schemes [8, 9], a binary tree is used to represent the entities and the key material. Each entity is assigned a leaf in the tree, and all nodes (and leaves) are associated with unique (secret) symmetric keys. Each entity knows all the symmetric keys on the path between its assigned leaf and the root of the tree, therefore the root key of the tree is known to all entities.

When the central authority wants to create a new group, it encrypts the information to be shared by group members (common secret) with keys in the tree that cover only those new group members. The encrypted value can then be safely broadcasted to the entire world, since only the intended group members will be able to decrypt it. This scheme works best when the selected entities cluster together under a tree's sub-tree. In this case, the common secret only needs to be encrypted under the key at the root of the sub-tree.

In the CEPRBAC system, the encrypted data is *not* broadcasted by a central authority, but it is stored in the EHR database accessible by all users. The central authority is a party in the healthcare organization responsible for applying the access control policies (it is part of the "control unit" discussed in Section 4). Therefore, what is relevant for our scheme from the BE approach is the fact that, while *all* users have access to the encrypted messages, these messages can *only* be decrypted by a (possibly changing) privileged subset of users.

The messages referred to above are simply the encrypted data keys that encrypt the pieces of data. The entities in the BE scheme are the individuals in the system who are able to access patient data (doctors, nurses, etc). According to the defined access policies (default or personalized), the central authority decides which individuals should learn which data keys. Therefore, it encrypts the data key for each data piece with keys in the tree that cover only the individuals with the right to access that piece of data. In this way, the encrypted data keys can be stored in the database to which *all* parties have access: only the parties that have the right to access a given piece of data will have the keys to decrypt the corresponding data key.

Regarding the construction of the broadcast tree, it can be built based on roles, in line with the RBAC approach. As depicted in Figure 3, there is a root node (to which a key K_{root} is associated) and, attached to the root node, all the role sub-trees (with associated node keys K_{R1}, K_{R2}, ..., K_{Rn}). Therefore, the leaf nodes are grouped based on individuals' roles that they represent. In order to address all individuals in a given role, the role key can be used. For instance, K_{Ri} can be used to encrypt a value to be accessible only to individuals in Role i. Note that *at the root* the tree is not necessarily binary. The binary construction is used in the role sub-trees.

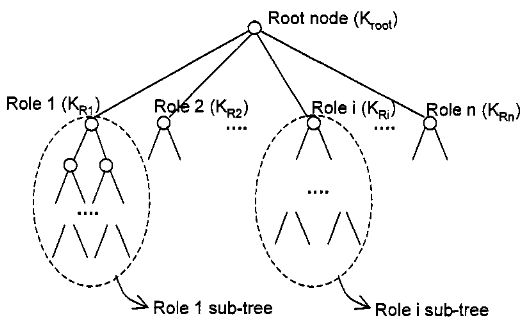

Fig. 3. Grouping nodes based on roles.

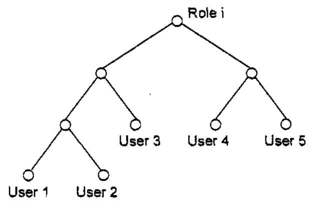

Fig. 4. An example tree for Role i.

Each role sub-tree accommodates in its leaves all the individual users in that particular role. In Figure 4, an example is shown with five users in Role i, so the sub-tree's last level is incomplete.

If there are no patient-defined policies, the system is simply a cryptographically-enforced RBAC in which the data keys are encrypted with the role keys (the keys K_{R1}, K_{R2}, etc). When a patient denies access to his data to a given user (e.g., User 4 in Figure 4), the corresponding data key must be re-encrypted with node keys in the tree that cover only the allowed users (keys K_{123} and K_5 in Figure 5). This is more efficient than having to encrypt the data key with the keys K_1, K_2, K_3 and K_5 of all allowed users.

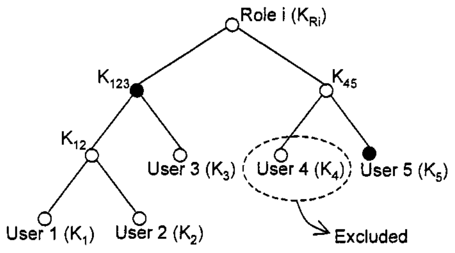

Fig. 5. Excluding a user.

When only a single user is excluded by a patient, there is a clear gain in efficiency in the system regarding the number of re-encryptions of the data key (and therefore the number of *stored* messages): all users who do not belong to the excluded user's largest sub-tree can be addressed with a single key (the key K_{123} in the example above). When more users are excluded, say *m* users, still the number of re-encryptions of the data key is smaller than *(n-m)*, where *n* is the total number of users in the system. For instance, the elimination of a single user from a role in a complete binary tree implies that the number of needed keys is of order log_2n rather than *n*. The worst possible case in the broadcast encryption scheme is in fact when the leaf keys of all non-excluded users must be used to encrypt the data key.

Some of the keys in the system need to be updated if they are compromised. The disclosure of an access key has typically much more serious consequences than the disclosure of a data key. In the former case, not only the disclosed access key needs to be updated, but also all data keys that are encrypted with that access key. This is especially troublesome when the access key is a role key. Then a large number of data keys will probably be involved as pure role-based policies are expected to be more common in the system.

3.3 Adding and deleting users

When a new user enters the system or a user leaves the system, the broadcast tree has to be updated. These two situations are discussed below.

When a new user enters the system, he is positioned in the tree according to his role (or roles) and his similarities concerning access rules with other users in the tree. For instance, if there is already in the tree a (first) user whose access rights are very similar to the right of the new user, a new level is added from the first user: his leaf becomes then an internal node from which two new leaves emerge, one assigned to the first user and the other to the new user. In case there are no users in the tree with any similarity to the new user, a new branch for the user is created from the root node (for a more detailed description see [10]). The new user must now learn his new (leaf) key and all the keys on his path to the root-node. Once the tree is updated, the database itself must be updated to take the new user parameters into account and the affected data keys must be encrypted with new access keys.

When a user leaves the system, his leaf in the tree is deleted and the sibling node must go one level up in the tree. Since the user knows all keys on his path to the root-node, all keys in that path must be renewed to ensure the system's security. To avoid involving a number of users in the node-key update process, the system can use smartcards for storing the access keys in a secure way. Then the user will not be able to learn the keys he possesses while using the system. In the case of leaving the system, the user is required to return the smartcard so that the keys are not compromised. Consequently, the tree could keep some free nodes reserved for new users.

3.4 Discussion: positioning users in the tree

The issue of how to position users in the leaves of the broadcast tree is an important one since an optimal positioning may give significant savings on the number of node keys necessary to address a certain group of users. In other words, key management can be made much more efficient if the keys which are higher in the tree are used as much as possible. Pure RBAC is a straightforward case, with an optimal tree that is simply based on roles.

For the role-based broadcast tree, however, additional regrouping may further optimise the system. It may be possible that the roles themselves are grouped together: if the roles are different but have similar access rights, they can be put under the same sub-tree (and therefore most often addressed with a single node key). This is the case, e.g., of anaesthesiologists and emergency doctors who may need to access very similar types of data. Furthermore, there are also degrees of freedom regarding the positioning of individuals within a role sub-tree. In case, e.g., personalized policies exist which are similar for a number of individuals within a role, these individuals may be also regrouped under the same sub-tree.

Another important issue concerning user positioning in the tree is the fact that some users may have multiple roles within the institution, which they may assume according to a daily-defined schedule (e.g., a user may be a GP one day and an Emergency Doctor (ED) the next day). For such users, as against the *single-role* users, the positioning in the tree requires more thought, as discussed below.

In order to handle multiple roles, a tree can be built still based on roles. After all single-role users have been assigned to leaves on their role sub-tree, multiple-role users can be assigned to leaves with use of extra information (e.g., the statistics of assignment of users to one or another role and users' similarities). Some cases are:

- If the user has *predominantly* a given role (e.g., he works mostly as a GP, but sometimes as an ED), then he is assigned to a leaf under the sub-tree of the predominant role.
- If the user has no predominant role, but normally his role schedule strongly *correlates* with that of a second user (e.g., very often they both work together as EDs), then he may be grouped together with that second user.
- If the user has no predominant role, and there are no role correlations with other users, then he may be assigned to a leaf under the sub-tree of any of the possible roles he may have.

Moreover, the addition of the multiple-role users to a role sub-tree is done preferably by keeping the single-role users (who have been allocated firstly) isolated as much as possible under some sub-tree.

Given above is a set of *guidelines* that can be used to build the broadcast tree when users may have multiple roles in the system. If enough new information is gained with the use of the system in time (e.g., information that help group together users with the same role at the same time), then it can be used to re-build the tree at any time.

4 An EHR Management System

This section places the above-proposed solutions in the architectural context of a system for secure management of EHR. First, the system components are described followed by a description of the general system architecture as well as the system dynamics. The two different usages of the system, on-line in a secure environment as well as off-line in an untrusted environment, are addressed.

4.1 System Components

The CEPRBAC system consists of five main components: the control unit, the EHR database, the trust center, the access device, and the repository of default and personalized policies.

The **control unit** is the core component of the system. It links the architecture components together performing two main tasks. First, it manages access to the data based on roles and personalized polices. Second, it identifies the users and handles their queries.

The **EHR database** is a collection of XML documents (each patient's EHR is considered by the system as an XML document). If the system security is based on data encryption, then the EHR database will actually store encrypted XML documents as well as "data keys" encrypted with "access keys", as defined in section 3.1. In addition, the EHR database will also provide an HL7 interface for data exchange.

The **trust center** is a trusted server that keeps sensitive information. It stores the important keys of the system. For key management the trusted center uses XML Key Management Specification (XKMS) [11], which defines a Web services interface to a public key infrastructure. The role of the Trust Center is twofold. First, it generates a public key pair for each user. The secret key is stored on the user smartcard and is used for identification and to sign data. Second, it stores all the necessary information to retrieve patients' data, including the symmetric keys used to protect the data.

The **access device** is a device that the user will use to view or update the EHR data. It could be an integral part of the system (the server depicted in Figure 6). However, it could be also a client device from which the data is accessed remotely. We assume compliance, especially for off-line scenarios where this device is used in an untrusted environment. With compliance we mean among the rest that device will not export in the clear the data or the keys which are used to decrypt EHR data.

The **repository of policies** stores default and personalized policies as well as exception lists. This component is not needed for the off-line use as the policies are stored together with the data in the form of encrypted data keys.

4.2 General Architecture

Having described the main components of the system, let us look at how they interoperate and support secure management of the EHR. Depending on the security context in which the system is used, the components can be differently combined and deployed. For example, in a secure environment of a hospital it might be decided to rely on traditional security mechanisms and not to encrypt the data. Let us look how the EHR could be managed in this case (Figure 6).

A user logs into the system using the user-name password procedure. Once the user is authenticated, the medical control unit consults the default policies as well as policies specific for this user and keeps them in memory (steps 1.1 to 1.4 in Figure 6).

From this point, the user can make two types of queries: writing or reading. "Writing" is used to add data and "reading" to view data. Deleting and updating data is strictly forbidden by the system. Those two actions have to be considered as particular kinds of writing: if a user wants to update or delete data, he enters new data with a comment explaining that the previous data is deprecated or not to be used any more. For each operation the user performs, the control unit checks first using the policies if the user is allowed to perform the operation based on the data provided in the query (operation, data piece and patient identifier). It checks which personalized policies are applicable (steps 2.2 – 2.3 in Figure 6) and if allowed performs the operation.

A patient can influence the personalized policies if he wishes to add or restrict rights to a specific user. However, the final approval of new policies comes from the administrator who checks if the new policy is compliant with the hospital policies. This is depicted in Figure 6 in steps 3.1 till 3.4.

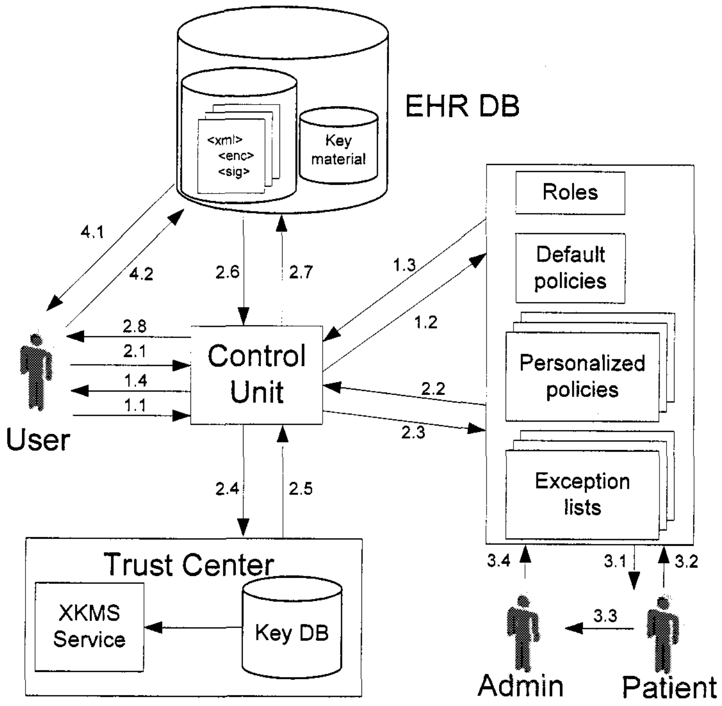

Fig. 6. Architecture of the EHR management system.

The hospital could also decide to enhance the security of a traditional access control mechanism using cryptography. If the data in the EHR database is encrypted that would prevent an attacker bypassing the access control mechanisms of being able to get the data in the clear. However, this will influence the architecture of the system in a way that the control unit will get additional responsibilities and the trusted center will start playing a role. For example, during the add operation, the control unit, in addition to consulting the policies, extracts also the metadata which is later used in the retrieval process. This metadata is stored in the trust center (steps 2.4 – 2.5 in Figure 6). Indeed, because the data is stored encrypted it furnish no information about its content, the system needs this metadata. To encrypt the data, a symmetric key is generated in the trusted center for each data piece. The data is encrypted with this key, signed with the creator's private (secret) key and together with patient ID, data piece ID and the creator ID stored in the database (steps 2.4 – 2.7).

The introduction of encryption and the method proposed in this paper, which actually builds in the data itself the data access policies, allows the off-line use of the portions or complete EHR database. The required parts of the database only have to be copied on a CD or portable device that will be used off-line. In this way an authorized user, such as a doctor visiting a patient or any of a group of emergency doctors in an ambulance car, will be able to access related EHRs offline. This can be accomplished by using a smartcard and a compliant software installed on a portable device without the complete hospital infrastructure (4.1 – 4.2 in Figure 6). This is a first step towards Digital Rights Management (DRM) functionality [12, 13] for the EHR management system.

5 Conclusions

This paper presents and discusses a system for personalized role-based access control that combines traditional role-based access control with user-managed access control. As required by legislation in the healthcare domain (e.g. HIPAA), the system allows a user to define individual access rights restrictions (or grants) on top of the default role-based policies. Furthermore, the proposed methods supports efficient evaluation of this combination in rum-time. This is achieved by means of dynamic revisions of personalized policies and exception lists that always list the minority of the users who are in conflict with the personalized policy.

Furthermore, security is enhanced by cryptographic enforcement of the proposed access control method. This prevents both an authorized insider as well as an attacker from outside to bypass the access control mechanism and have direct access to the EHR database where all data is in the clear[2]. Even more importantly, the proposed approach increases the data availability (the most important requirement in healthcare), supporting a simple offline usage of the data by incorporating access policies within the data. The paper discusses also important issues relating to key management, which appear in this approach, and proposes an efficient solution based on broadcast encryption. It is found that broadcast encryption is highly applicable to the method of personalized role-based access control proposed in this paper, not only in the healthcare domain, but in any domain that needs to cope with role-based access control and individual restrictions.

The work presented in this paper suggests some interesting directions for future research, which are discussed below.

An interesting and important issue already brought up in section 3.4, is how to arrange an optimal distribution of users in the broadcast tree given that similarities (other than role-based) may exist in the access rights of the users. These similarities may reflect patient-defined policies, for instance, but they may be also based on similarities and correlations between users if they have multiple roles.

A related topic to the construction of the tree concerns role hierarchy (role to sub-role relations), which this paper does not address. Therefore, an important question is how to map the role hierarchy on to the broadcast tree.

Furthermore, with the basic framework in place, it would be worthwhile to investigate if the proposed approach could be scaled up. That means, assuming the proposed framework would apply, e.g., to different health institutions within a

[2] Note that an insider has access to the EHR database and is authorized to view specific parts of it. However, he can bypass the database access control mechanisms and access all the data through the file system. The same holds for an attacker from outside who has managed to break into the system.

country, issues regarding the integration of such frameworks on a national level could be investigated.

Finally, the secure off-line usage of data requires the integration of functionality into the system which resembles that of DRM. This is an important direction for our future work.

Acknowledgements

We are grateful to our colleagues Joop Talstra, Toine Staring, Paul Koster, and Jordan Chong for fruitful discussions and valuable comments. Furthermore, we would like to thank the anonymous referees for their constructive remarks.

References

1. D.F. Ferraiolo and D.R. Kuhn "Role Based Access Control" 15th National Computer Security Conference, 1992.
2. The Health Insurance Portability and Accountability Act of 1996 (HIPAA), HIPAA Administrative Simplification - Regulations & Standards, http://www.cms.hhs.gov/hipaa/hipaa2/regulations/default.asp.
3. HIMSS Electronic Health Record Definitional Model Version 1.0, HIMSS Electronic Health Record Committee, http://www.himss.org/content/files/EHRAttributes.pdf.
4. Selim G. Akl and Peter D. Taylor, "Cryptographic solution to a problem of access control in a hierarchy", ACM Trans. Computer System, 1(3):239-248, 1983.
5. R. S. Sandhu. On some cryptographic solutions for access control in a tree hierarchy. In ACM '87: Proceedings of the 1987 Fall Joint Computer Conference on Exploring technology: today and tomorrow, IEEE Computer Society Press, pp: 405-410, 1987.
6. Organization for the Advancement of Structured Information Standards, http://www.oasis-open.org/home/index.php.
7. Tim Moses, eXtensible Access Control Markup Language (XACML) Version 2.03, http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf.
8. Kou, L., G. Markowsky, L.Berman, "A Fast Algorithm for Steiner Trees", Acta Informatica, Springer-Verlag, 1981: vol. 15, pp.141-145.
9. A Survey of Broadcast Encryption, Jeremy Horwitz, Manuscript, 2003, http://math.scu.edu/~jhorwitz/pubs/b5oadcast.pdf.
10. Malik Hammoutene, "Secure Management of Medical Data", EPFL Master Thesis Series, 2005.
11. W3C Recommendation, XML Key Management Specification (XKMS 2.0), http://www.w3.org/TR/xkms2/.
12. M. Petkovic, M. Hammoutène, C. Conrado and W. Jonker, "Securing Electronic Health Records using Digital Rights Management", In Proceedings of the 10th International Symposium for Health Information Management Research (iSHIMIR), Greece 2005.
13. C. Conrado, M. Petkovic, M. vd Veen, W. vd Velde, "Controlled Sharing of Personal Content using Digital Rights Management", In Security in Information Systems, Eduardo Fernandez-Medina, Julio Hernandez, Javier Garcia (Eds), 3rd International Workshop on Security in Information Systems (WOSIS), Miami, USA, 2005, pp. 173-185.