Audit-based Compliance Control (AC2) for EHR Systems

M.A.C. Dekker[1,2], J. den Hartog[2], S. Etalle[2]
[1] Security group, TNO Information and communication technology
[2] Distributed and Embedded Systems group, Universiteit Twente

1 Introduction

Traditionally, medical data is stored and processed using paper-based files. Recently, medical facilities have started to store, access and exchange medical data in digital form. The drivers for this change are mainly demands for cost reduction, and higher quality of health care. The main concerns when dealing with medical data are *availability* and *confidentiality*. Unavailability (even temporary) of medical data is *expensive*. Physicians may not be able to diagnose patients correctly, or they may have to repeat exams, adding to the overall costs of health care. In extreme cases availability of medical data can even be a matter of life or death. On the other hand, confidentiality of medical data is also important. Legislation requires medical facilities to observe the privacy of the patients, and states that patients have a final say on whether or not their medical data can be processed or not. Moreover, if physicians, or their EHR systems, are not trusted by the patients, for instance because of frequent privacy breaches, then patients may refuse to submit (correct) information, complicating the work of the physicians greatly.

In traditional data protection systems, confidentiality and availability are conflicting requirements. The more data protection methods are applied to shield data from outsiders the more likely it becomes that authorized persons will not get access to the data in time. Consider for example, a password verification service that is temporarily not available, an access pass that someone forgot to bring, and so on. In this chapter we discuss a novel approach to data protection, *Audit-based Compliance Control (AC2)*, and we argue that it is particularly suited for application in EHR systems. In AC2, *a-priori* access control is minimized to the mere authentication of users and objects, and their basic authorizations. More complex security procedures, such as checking user compliance to policies, are performed *a-posteriori* by using a formal and automated auditing mechanism. To support our claim we discuss legislation concerning the processing of health records, and we formalize a scenario involving medical personnel and a basic EHR system to show how AC2 can be used in practice.

This chapter is based on previous work (Dekker & Etalle 2006) where we assessed the applicability of a-posteriori access control in a health care scenario. A more technically detailed article about AC2 recently appeared in the IJIS journal, where we focussed however on collaborative work environments (Cederquist, Corin, Dekker, Etalle, & Hartog, 2007). In this chapter we first provide background and related work before explaining the principal components of the AC2 framework. Moreover we model a detailed EHR case study to show its operation in practice. We conclude by discussing how this framework meets current trends in healthcare and by highlighting the main advantages and drawbacks of using an a-posteriori access control mechanism as opposed to more traditional access control mechanisms.

2 Background

To protect data confidentiality, numerous distributed access control mechanisms have been developed. Typically, these systems try to prevent illegitimate actions before their occurrence, by deciding on the fly whether or not access should be granted. We believe that in EHR systems a different approach can be very useful. Basically, EHR systems must fulfill two requirements (The European Parliament and the Council of the European Union, 2002; The US Dpt. of Health and Human Services, 2000; Nederlands Normalisatie Instituut, 2002):

- To provide high-quality health care, the EHR must be immediately available, preferably across the boundaries of the different hospitals and abroad.
- To protect the patient's privacy, the EHR must remain confidential and should be disclosed only according to applicable law and/or the patient's explicit consent.

The latter requirement is particularly important in a country such as The Netherlands, where medical insurances are privatized and the patients' medical data is worth millions.
Note that fulfilling both requirements is hard. To fulfill the first requirement, the mechanism should be relatively simple and fast. The second requirement however states that access should only be granted under precise conditions and circumstances.
Considering the complexity of the medical work flow, the large number of health records and the variety of institutions, users and systems involved, it seems likely that checking these circumstances and conditions is slow and prone to errors.
Besides the problem of designing a fast access control mechanism that at the same time takes into account complex conditions (Abadi, 2003; Becker & Sewell, 2004; Halpern & Weissman, 2003), it is considered impossible, in general, to design an access control mechanism that models every circumstance perfectly (Rissanen, Firozabadi, & Sergot, 2004); in other words, there are always exceptional, unforeseen circumstances. This is an important issue in the EHR setting, given the mobility of patients and staff, and the urgency of health care. We believe that at least it should be possible for medical staff to self-authorize exceptions to rules, while leaving the process of *justifying* the exceptions for later.

2.1 Related work

We are unaware of existing a-posteriori access control systems. We would like to mention however systems which are somehow technically related to the AC2 framework.
The *Cassandra system* (Becker & Sewell, 2004) was designed to implement an (a-priori) access control in an EHR system. The authors test the expressivity of the Cassandra policy language by expressing existing policies regarding access to medical data and activation of medical roles. The policy languages used in the Cassandra system are different flavors of *Datalog with constraints*. (In fact Cassandra builds on a line of research in distributed access control systems using Datalog to express policies, most notably Binder and Delegation Logic.) While Cassandra's policy language is in theory undecidable, it is argued that this is not problematic in practice. On the other hand, the policy language used in AC2 is not based on Datalog, but on a fragment of *first-order logic* (Cederquist et al., 2006), which is semi-decidable. Rissanen et al. (Rissanen et al., 2004) address the issue of how to override safely the decisions of a preventive access

control system called the Privilege Calculus. At each override a procedure starts to find the appropriate authority which is notified to audit the override. In AC2 there is only a minimal preventive access control mechanism, which can not be overridden. Moreover, in our approach it is up to the auditors to decide when and which users to audit.
The *justification proofs* in AC2 are based on a formal logic. A number of distributed access control models are based on formal logics (see the survey by Abadi (Abadi, 2003)). In these models an authorization request or an authentication credential corresponds to a logical formula and the authorization or authentication decision corresponds to a proof of the formula. In the PCA framework (Appel & Felten, 1999), proposed by Appel and Felten, higher-order logic proofs are generated by the clients, proving that they have the authorization to access webservers. The undecidability of proof finding in higher order logic is not a problem in their setting, because it is the user's task to find the proofs. A similar construction is used in AC2, where users justify actions to auditors by submitting formal proofs which can be checked automatically.

Related to the problem of protection of health records, is the enforcement of copyrights in content-sharing systems (DRM). It has been argued that DRM can be used to enforce privacy policies. For instance, Conrado et al. (Conrado, Petkovic, Veen, & Velde, 2005) propose to use DRM to enable privacy in content-sharing systems and vice versa to use privacy as a driver for a wider use of DRM enabled devices. DRM however, unlike AC2, requires special (compliant) hardware or software at the application layer. This makes DRM unsuitable for EHR systems or enterprise-privacy systems. In the context of DRM, a type of a-posteriori access control was proposed by Shmatikov and Talcott (Shmatikov & Talcott, 2005). There, a reputation-based trust management (TM) model is presented, in which the reputation of individual agents is determined by the fulfillment or violation of (DRM) licenses. We believe that Trust Management (Li, Mitchell, & Winsborough, 2002), coupled with auditing, may be an interesting solution, especially in large distributed EHR systems.

The main differences of AC2 with respect to access control (AC) and digital rights management (DRM) are summarized in Table 1.

	AC	DRM	AC2
Administration of assigned rights	centralized	decentralized	decentralized
Encryption of data required	no	yes	no
Authentication	a-priori	a-priori	a-priori
Authorization	a-priori	a-priori	a-posteriori
Data control after access	no	yes	yes
Post Obligations	no	no	yes

Table 1: Comparison of characteristics and features with respect to access control (AC) and digital rights management (DRM).

Access control, as opposed to AC2 and DRM assumes a centralized access point, where access requests are evaluated against an access control list, or policy. In DRM and in AC it is possible for users to store licenses or policies offline, without having to rely on a central access point. In DRM, to protect the data, the data is exchanged in encrypted form, while only special hardware can be used to process (or render) the data. In all three systems users are authenticated before they can request data. AC2 differs from DRM and AC in that users authorizations are not checked on the fly, but only in an a-posteriori audit procedure. One of the disadvantages of AC is that once the user accesses the data, the access point has no control over further processing. In DRM, and in AC2, there is control after the data has been accessed. In DRM this is used for example to implement pay-per-view policies. Finally, policies containing post-obligations, or promises, can not be implemented in AC, nor in DRM. In AC2 it is possible to specify and enforce policies of the form *"notify doctor X within two days"*.

3 Audit-based Compliance Control

In this section we provide the details about the principal components of the architecture of the AC2 system and we show how it can be used in an e-health setting.

As an introduction we first give a brief overview of the components in the AC2 system. The overall architecture is illustrated in Figure 1. The architecture contains agents exchanging *policies* (1) and executing *actions* (1,3). Actions can be *logged* (2) and *audited* (4-6) for their compliance to policies at a later time, i.e. a-posteriori. The framework is supported by tools which enable agents to *build* compliance proofs and auditors to automatically *check* these proofs.

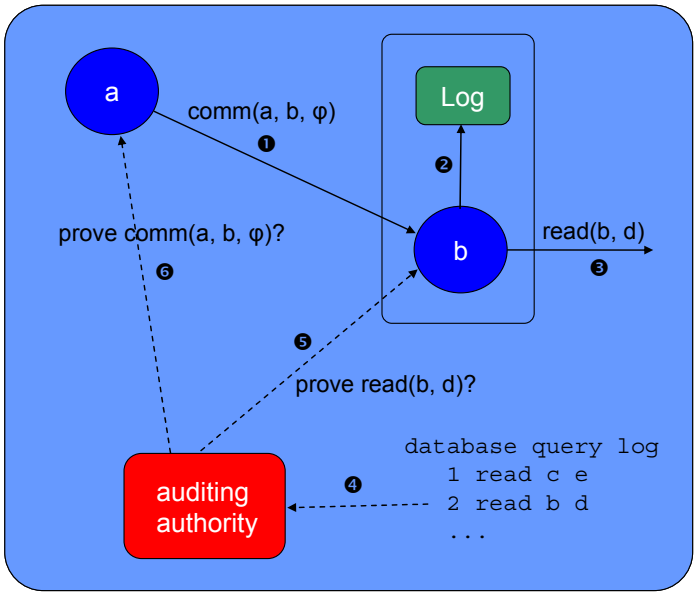

Figure 1: An overview of the architecture. Agents execute actions and

keep logs in order to respond to a possible audit later on. Dotted arrows denote interactions with an auditor. Policy ϕ equals mayRead(b, d).

3.1 Architecture of AC2

3.1.1 Policies and Actions

Actions are modeled in AC2 by a set Act containing basic actions and setting-specific actions. Actions can be performed by real users, systems, programs, and so on, and we refer to these as *agents*. The basic actions are *comm(a, b, P)* and *creates(a, d)*:

- *creates(a, d)* represents the creation of a piece of data d by agent a.
- *comm(a, b, P)* expresses the communication of policy *P* from agent *a* to agent *b*.

By communicating policies agents delegate rights to other agents to execute certain actions. Examples of setting-specific actions are *read(a, d)*, expressing that *a* reads file *d* and *giveDrug(a, b, d)* expressing that agent a gives drug *d* to subject *b*.

AC2 uses a policy language which is based on *first-order logic*. The advantages of using first-order logic are that first that *reasoning* in first-order logic is well understood from a scientific point of view, and second, that first-order logic has an clear semantics. Note that it has been shown that first-order logic can be used to express a wide range of access control policies (Halpern, J. Y., & Weissman, V. , 2003), and note also that although we will stick with the common syntax for logics, it is straightforward to translate this syntax to XML, for example.

Policies are built using a set of atomic predicates, which can be either permissions, or conditions. A basic predicate is *owns(a, d)* expressing that agent *a* owns object *d*. The meaning of ownership is as usual in discretionary access control. A creator of data, owns that data, and can authorize other users to process the data. Additionally, one may have scenario-specific predicates such as permissions *mayRead(a, d)*, expressing that *a* is allowed to do the read action on *d*, or the condition *isNurse(a)* expressing that agent *a* is a nurse. Complex policies can be built from permissions and conditions using the usual logical connectives. The grammar of the policy language is (Cederquist et al., 2005):

$$\phi ::= \phi \wedge \phi \mid \forall o.\ \phi \mid \phi \rightarrow \phi \mid a \text{ says } \phi \text{ to } b \mid !\alpha \rightarrow \phi \mid ?\alpha \rightarrow \phi \mid$$
$$::= \mathsf{p}(o_1, ..., o_n) \mid a \text{ owns } d \mid \top.$$

Figure 2: Grammar of the policy language

Here p is an n-ary predicate symbol and o are objects or variables of the appropriate sort, and ! and ? express use once and use many obligations, respectively. For example, the policy 'Each time a nurse gives drug to a patient, agent c can bill that patient' is written as:

isNurse(x)→ isPatient(y)→ !giveDrug(x, y, c)→ mayBill(c, y).

Use-once obligations require special care in the distributed setting we are considering, to ensure that they are used only to justify a single action (Cederquist et al., 2005).

The *owns* predicate models ownership of data, which gives the permission to derive any policy concerning this data, including the permission to delegate permissions about it to other agents. The logical connectives `∧' and '∀', and are treated as usual. The *says* construct is a special connective, which is used to express the permission to *delegate* policies. For example, the policy `Any doctor can delegate the treatment of his patient to a nurse' is written as:

$$\forall x, y, z.\ isDoctorOf(x, y) \rightarrow isNurse(z) \rightarrow x\ says\ mayTreat(z, y)\ to\ z.$$

Our logic allows deriving permissions from actions that have occurred. For example, the communication action allows the recipient to derive certain permissions. We refer to earlier work for a more complete description of the underlying derivation system (Cederquist et al., 2007).

3.1.2 Logging and Accountability

In the AC2 system, similarly to what happens in proof-carrying code frameworks (Appel & Felten, 1999; Whitehead, Abadi, & Necula, 2004), when an action of an agent is audited, by an auditor, the agent has to present a *justification proof* for it. The agent may use (part of) its log as evidence in the justification proof. For example, consider the two actions executed by agents a and b in Figure 1: In step (1) agent a communicates the policy to agent b and agent b logs this communication (2), for later. Agent b reads data d (3) and at some moment (4) the Auditing Authority finds this action in some audit trail, in this case the log of queries executed at some database. The auditor decides to ask agent b for justification (5). Agent b justifies, and uses the logged evidence (in step 2) of agent a's policy communication. The auditing authority now asks agent a for a justification of having said to b. A formal proof system, denoted >>, is used to derive justification proofs from evidences in the log. Informally, an agent a can justify an action act using an excerpt E of its log, by proving that E >> P where P is the proof-obligation for the action act. For example, suppose that the proof-obligation for the action read(a, d) is the formula mayRead(a, d), then to justify the action read(a, d) to an auditor, agent a has to supply a proof of a mayRead(a, d). This proof can then be checked by the auditor. Here we are not interested in specifying how auditors collect evidence and which actions should be audited by the auditors (Sandhu & Samarati, 1996). More details regarding proof-obligations can be found in earlier work (Cederquist et al., 2005). In the event of an audit, an agent needs to find justification proofs based on its log. This involves reasoning about possibly complex policies and actions, hence this is supported by a *theorem prover*. Checking supplied proofs in turn is done by a *proof checker*. Both tools are implemented separately using different systems (Cederquist et al., 2006): The proof checker uses Twelf, the proof finder uses SWI Prolog. Shortly, the reason for this distinction is that proof finding is much harder than proof checking. Prolog is a fast application for finding proofs, but the code can be hard to verify for soundness. Proof checking on the other

hand is much easier, and can be done by type-checking. Our type-checker is coded in a few lines of Twelf code which can be checked manually for soundness.

3.2 An E-health application case study

In this section we illustrate how AC2 can be used in the e-health setting, by modeling a simple case study involving health records and medical personnel. In our example health records are processed and accessed by numerous systems in different places. To protect the patient's privacy and the privacy of medical personnel, policies describe who can access the medical record and under which circumstances. In this section, we ignore the issues of moving records from one system to the next; instead we focus on the policies that accompany the records.

3.2.1 Architecture customization

The agents in this case study are patients, doctors, nurses and administrative employees; the users of the EHR system, while the data objects are the medical records. The actions we concentrate on are reading and updating medical records, administering medicines and billing patients for medicines.

The format of the medical records we consider is inspired by the legal directives on privacy of health records (The European Parliament and the Council of the European Union, 2002; The US Dpt. of Health and Human Services, 2000; Nederlands Normalisatie Instituut, 2002): The EHR is divided into a personal information section (PI) and a medical data section (MD). The PI records all non-medical information related to the patient, such as billing information, and information regarding the patient's family members. The MD gathers all the medical information of the patient, such as diagnoses and given prescriptions. Updates to the EHR are performed by appending new information together with the identity of the agent making the update.

Several auditors have the mandate of controlling that the medical records are used appropriately, i.e. the hospital's internal auditor, a government authority and a patient union representative. Independently, these auditors may audit different sections of the healthcare organization.

3.2.2 The Hospital policy

The hospital has defined a general policy, H, to protect the privacy of patients and allows medical personnel to access and handle the necessary health information:

H 1. *A patient may read and update the PI section of his medical record and authorize others to do so;*

H 2. *A patient may read and update the MD section of his medical record and authorize others to do so;*

H 3. *A doctor may read the PI section of the medical records of his patients;*

H 4. *A doctor may read and update the MD section of the medical record of his patients;*

H 5. *A doctor may give medicines to his patients;*

H 6. *A doctor can delegate to a nurse on his staff the administering of medicines;*

H 7. *An administration employee may bill a patient each time someone has given medicines to that patient;*

Additional customized policies may be added later to this general policy by the individual users, for example expressing explicit patient consent given to medical staff. Note that whatever is not explicitly mentioned in a policy is not permitted and that added policies can only add permissions not remove them (monotonicity). A special mechanism, such as the revocation mechanism of SPKI/SDSI, would be required to model explicit prohibitions. We do not go into these details here.

3.2.3 Scenario

As a concrete instance of the general setting we consider the patients Alice and Bob, doctors David and Diana, a nurse Natalie and an administrative employee called Charlie. Alice trusts doctor David to decide when to give her medical file to other doctors, e.g. for a second opinion, i.e. Alice has a policy Pa stating *`Dr. David can delegate the permission to read the MD section of Alice's medical record.'*.
This policy is expressed in the AC2's policy language by:

$$\forall b, d.\ isMD(alice, d) \rightarrow isDoctor(b) \rightarrow david\ says\ mayRead(b, d)\ to\ b.$$

Alice's policy adds addition permissions to those provided by the rules in the hospital's policy. The hospital's policy (H4) only allows Dr. David to read Alice's medical file. With Alice's policy Dr. David may additionally authorize other doctors to read her medical file. Consider the following sequence of actions performed by the users.

A 1. *The hospital gives its policy to Dr. David.*

A 2. *Dr. David logs this for later.*

A 3. *Alice becomes patient of Dr. David.*

A 4. *Dr. David logs this for later.*

A 5. *Alice meets Dr. David in his office.*

A 6. *Dr. David reads the PI section of Alice's record, to remind himself of Alice's personal details.*

A 7. *Alice communicates her new policy Pa to Dr. David.*

A 8. *Dr. David logs this communication for later.*

A 9. *Dr. David updates the MD section of Alice's record.*

Note that Dr. David logs the events A1, A3 and A7 because the evidence of these events may be useful for him later on. For example, Dr. David can use A8 (i.e. the log of A7) later on to prove that he was allowed to show Alice's file to another doctor. The auditors on the other hand may keep an independent (e.g. random) account of actions, in so-called audit trails. In particular, the hospital's privacy officer routinely monitors the queries to a database with medical records, both to detect anomalous behavior and to ensure that the hospital's policy is adhered to. The communication between Dr. David and patient Alice, not monitored by the hospital's privacy officer may contain policies. They do not become known to him until a user uses the policy communication in some justification proof.

Independently, an external auditor controls the financial accountability of the hospital, i.e. to ensure that only actual costs are billed to patients.
In our concrete example, after some time the hospital's privacy officer asks Dr. David for a justification for having accessed Alice's file. To give a justification Dr. David needs to show to the auditor the log entries A2 and A4 from the sequence above and a proof that:

[A2, A4] >> david mayRead(david, alice).

David's proof is automatically checked and the auditor finds that Dr. David has correctly accounted for this action.

Now, unexpectedly, Alice arrives ill at the hospital while Dr. David is off duty. Dr. Diana who is on a call with nurse Natalie, treats Alice upon arrival:

B 1. *Dr. Diana logs that Natalie is a nurse on her shift.*

Informally Alice asks for treatment.

B 2. *Dr. Diana reads the MD section of Alice's medical record.*
B 3. *Dr. Diana updates the MD section of Alice's record.*

Informally Dr. Diana tells nurse Natalie to give Alice the medicine Qurol.

B 4. *Natalie administers the medicine.*
B 5. *Natalie notifies billing that Qurol was given to Alice.*
B 6. *Charlie logs this for later.*
B 7. *Charlie bills Alice for the medicine.*
B 8. *Charlie logs this, together with a reference to Natalie's notification.*

The actions in this sequence are shown in Figure 3. It is important to note that AC2 allows both Diana and Natalie to operate without proper authorizations. Alice's assertion that she is (consents to being) Diana's patient was initially missing. Moreover, Natalie should have had Diana's explicit authorization to administer Qurol.

Say half an hour later, when Alice is a bit better, she authorizes Diana, and when the shift is over, Diana authorizes Natalie:

B 9. *Alice becomes patient of Dr. Diana.*
B 10. *Dr. Diana logs this for later.*
B 11. *Dr. Diana tells nurse Natalie to give Alice the medicine Qurol.*
B 12. *Natalie logs this for later.*

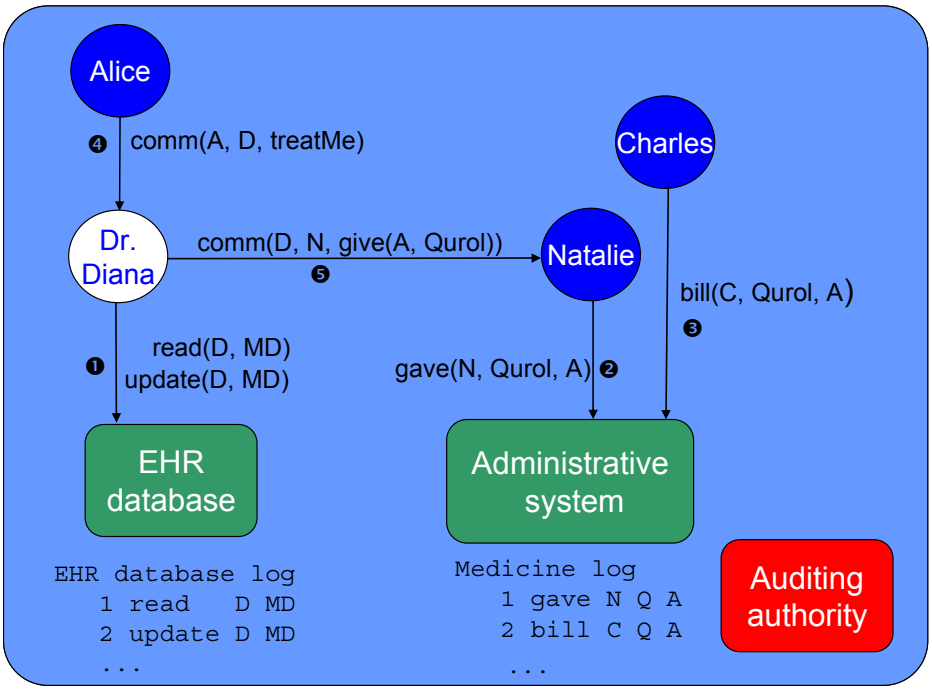

Figure 3: Medical treatment of Alice, where authorizations (4,5) are given a-posteriori to the treatment (2) and the data access (1).

In this example the medical staff first treats Alice and then records the necessary details for administration and accountability. Although initially not all the authorizations were available, *a-posteriori*, the operators *can* account for their actions.

For example, when Alice is asked to account for giving Qurol to Alice, she can send the log entry B12 above together with a proof of

[B12] >> natalie mayGiveDrug(natalie, alice, qurol).

To illustrate a more complex proof, Diana's proof for the delegation to Natalie (B11 above) is more involved. To account for it, Diana has to prove that:

[H5, B1, B9] >> Diana says mayGiveDrug(natalie, alice, qurol) to alice.

$$\dfrac{\dfrac{[H5]}{(\text{isDoctorOf}(diana, alice) \wedge \text{isStaffOf}(natalie, diana)) \rightarrow diana \text{ says mayGiveDrug}(natalie, alice, qurol) \text{ to } c.}\,\forall_e \quad \dfrac{[B1] \quad [B9]}{\text{isDoctorOf}(diana, alice) \wedge \text{isStaffOf}(natalie, diana)}\,\wedge_i}{diana \text{ says mayGiveDrug}(natalie, alice, qurol) \text{ to } natalie}\,\rightarrow_e$$

Figure 4: A Sketch of an accountability proof

The actual proof involves a number of steps. Diana can use her proof finder to generate the exact proof for her. In Figure 3 we sketch the proof. The auditor can check this proof

automatically, using the proof checker. We refer the interested reader to earlier work for more details about the implementation of the tools (Cederquist et al., 2007). Finally, to illustrate the use of use-once obligations, the next day Natalie gives another dose of Qurol, in line with what Dr. Diana told her.

C 1. *Natalie administers the medicine.*
C 2. *Natalie notifies billing that Qurol was given to Alice.*
C 3. *Charlie logs this for later.*
C 4. *Charlie bills Alice for the medicine.*
C 5. *Charlie logs this, together with a reference to (the log of) Natalie's notification (C3).*

Notice that the policy (H7) allowing Charlie to bill Alice for the medicines contains a use-once obligation: Each time someone gives drugs to Alice, Charlie can add to Alice's bill. Charlie needs to log the billing action, together with a reference to the corresponding notification by Alice (Cederquist et al., 2007). In other words, when billing Alice, Charlie has to state, in his log, to which notification it belongs. This allows the external auditor described earlier, to check that each item on the bill corresponds to a dose of Qurol administered by Natalie.

4 Applicability and Challenges

Currently 'healthcare is one of the least developed industries in regards to the application of ICT solutions'[1]. However, two main trends drive the need for innovation in the area of electronic health care. First of all we have the need for higher efficiency of the healthcare process. With the aging population and increasing lifespans, improving the efficiency of the healthcare process is essential to counter rising costs and to maintain availability of care (e.g. waiting lists). Secondly, a move to patient-central healthcare is taking place. The care process is becoming more personalized and patients are becoming more actively involved in treatment decisions. To support these processes, ICT solutions addressing the specific requirements of healthcare are needed.

In existing healthcare systems data use is typically static, data is not shared and remains within a single administrative domain and no or few privacy protection mechanisms are used. With personalized patient central healthcare medical data that travels with the patient through the care process and across different administrative domains. Protecting the user's privacy and enabling user control implies the need for user selectable policies. Several pilot projects aim to provide personalized care for prevention, rehabilitation and chronic diseases. Use of wireless body area networks allows continuous and remote monitoring of patients. For example, the MyoFeedback project measures muscle tension and provides feedback to the user to train the user to regularly relax the muscles, to help prevent RSI. The Awareness project[2] addresses tele-treatement of patients with chronic

[1] ICT Innovation conference presentation, P. Smit, Senior VP Strategy and Business Development, Philips Medical Systems
[2] Freeband Awareness project, www.freeband.nl/project.cfm?id=494&language=en

pain and tele-monitoring of epileptic seizures and uncontrolled movement in spasticity. In these applications medical information is gathered, processed in a context-aware manner and possibly sent to a care provider who may also need to provide feedback. A secure and flexible framework for exchanging and managing the monitored data and providing feedback to the patient is needed.

Sharing of medical data is addressed by several projects in the Netherlands. The Dutch Nationaal ICT Instituut in de Zorg (NICTIZ) works on national standards for healthcare electronic communication and has realized a platform for electronic medication files and general practitioner observation dossiers. The prime requirements for further deployment of e-health systems identified by the NICTIZ are additional privacy protection methods, protection of the rights of both patient and care providers and adherence to legislation and security norms (e.g. NEN-norm 7510 and 7511). Along with physical protection of data, user control is identified as a key ingredient to privacy protection. The eNIK (electronic Dutch identity card) will be used to allow a patient to access their data. The need to change access conditions in emergency situation is also seen as an important requirement.

Interoperability standards such as DICOM (Digital Imaging and Communication in Medicine) for viewing any kind of medical image regardless of the origin and HL7 (Health Level 7, electronic health record functional model and standard) provide interoperability on a data level. Initiatives such as IHE (Integrating the Healthcare Enterprice) promote the coordinated use of such standards. These and other initiatives will enable the physical sharing of medical data.

A health specific guide to the general ISO/IEC 17799 standard on handling sensitive and personal data has been developed by ISO in co-operation with CEN (ISO/ DIS 27799) `Health informatics – Security management in health using ISO/IEC 17799'. This will facilitate the formulation of common security polices across healthcare, and should help promote the adoption of interoperable security components and services. However, it is also foreseen that the security requirements and policies will vary between countries and clinical settings in ways that cannot and should not be standardized.

Summarizing we see the need for a flexible policy framework which can be adapted to the requirements of individual users and different healthcare settings and which allows checking adherence to (privacy) policies over multiple administrative domains. As medical data travels with the patient the policies need to capture delegation of rights and need to move with the data.
AC2 seems to be a suitable basis for such a system; it includes delegation and allows data and policies to move easily. The mechanisms of the AC2 system seem suitable to be combined with different access control and trust management techniques to address specific security requirements. By using a-posteriori access control it provides flexibility and availability of data in emergency or unexpected situations.
The case-study in the previous section shows how AC2 can be used in the EHR setting. Below we elaborate on the main advantages and the main drawbacks of this approach.

4.1 A-posteriori versus a-priori

The advantage of a-posteriori access control is that it allows the medical staff to go ahead with their duties, without worrying about problems like expiration of certificates, passwords or failing network connectivity to some authorization server. These issues can be dealt with at a more convenient time. By definition a preventive access control system does not provide this kind of flexibility. In practice, an a-priori access control system has to be extended with mechanisms by which users can override the a-priori decisions of the access control mechanism (Rissanen et al., 2004). These overrides must be reviewed later on for legitimacy, and for this purpose users must record the circumstances under which they needed to override. This logging is a central part of AC2. Useful events or performed actions have to be logged by the medical staff, for the purpose of accountability. By keeping logs of such events and actions, doctors and nurses can account to multiple auditors that come at different times. In a traditional a-priori access control system, on the other hand, the authorizations are only checked once by a single authority, i.e. at the moment access is requested. A-posteriori access control has a characteristic drawback: it does not prevent misbehavior; hence it does not give the robust security guarantees that are required in e.g. military information systems.

Many countries have adopted legislation that explicitly describes the requirements for EHR systems. Consider for example the summary of the US act HIPAA (The US Dpt. of Health and Human Services, 2000). The principal rule is as follows:

A covered entity may not use or disclose protected health information, except either: (1) as the Privacy Rule permits or requires; or (2) as the individual who is the subject of the information authorizes in writing.

The HIPAA act stresses that patients have the right to justifications of past disclosures of their medical records. In HIPAA it is called Disclosure Accounting, being *the subject's right to get an accounting of disclosures of its EHR in the past six years*. The a-posteriori access control mechanism, described in this paper, provides a formal definition of this accounting and the tools to automate such accounting. Although not required in the HIPAA act, automation is convenient to be able to run audit tools without human supervision, enhancing both the privacy and the efficiency of the audits. Finally, about the implementation of the law, the HIPAA act states that The Privacy Rule does not require *that every risk of an incidental use or disclosure of protected health information be eliminated*. Therefore, the main drawback of our approach, i.e. that with a-posteriori access control we can not completely exclude misuse, is in principle allowed for by the law.

4.2 Infrastructure

We outline the requirements with respect to the underlying infrastructure. Actions are executed at the session layer. Here authentication and non-repudiation is required, to be able to determine which users were responsible for which actions. Once an action is executed, the evidence of its occurrence should be safely stored for later, this is called an audit trail. Also, time-stamping of actions is required, to be able to demonstrate that an action was not executed before another, or before a required policy was received. Audit trails should provide auditors with a transcript of all (or most) user actions, to detect misbehavior. Consider the example in Figure 1. Here, a database system provides a

secure log of past queries to some auditor. On the other hand, the policy communications between the two agents are ignored by the auditor. For example, the users may be using some private email system to communicate (signed) authorizations. The auditors do not monitor these exchanges. Additionally, besides using the logging facilities of electronic systems such as databases and computers, audit trails can also be established using cameras, key-loggers, RFID sensors, database query logs, etc. The audit trails need protection from tampering (Sandhu & Samarati, 1996), but this is already required for accounting and security purposes (Jajodia, Gadia & Barghava, 1995). Moving up to the application layer, the main requirement here is the presence of a secure device for users to log the actions and events useful to them. Moreover, both auditors and user may use the tools shown in Section to find and check accountability proofs automatically. These evidences are used by the users in the event of an audit. For example, suppose a doctor changes a medical file and a policy states that in that case the corresponding patient should be notified. To justify his actions later on, he should log the action of notifying the patient. The notification is executed at the session layer, where it may be caught in an audit trail. At the same time the logging device must create a tuple containing the logged action, its time-stamp and possibly other parameters, which can be used as evidence in an audit later on (Cederquist et al., 2005). To prevent that users forge their logs, some secure device is needed that takes care of this logging. At the application layer, AC2 has little impact, which makes it rather suitable to implement across different institutions, where many different kinds of legacy systems are used. For example, the databases schemas of the involved organizations can remain the same and the ICT infrastructures would not have to be re-designed from scratch.

4.3 Privacy and Trust

An important requirement for a-posteriori access control is that there must be some mechanism in place to ensure that users can be held accountable for their actions, i.e. that a user will not vanish after executing his (illegal) actions. This is necessary for the trust of users in each other, and in the EHR system. In real life, this is often done using a bail sum or by some legally binding agreement, such as an employment contract. In our setting this is particularly important, given the fact that users can give extra permissions to others. A malicious user can set off a cascade of actions by other users. A potential solution is the incorporation of trust management. We assumed for simplicity that all users were equally trusted to utter security statements. However, consider the case in which some unknown doctor made changes to the drug prescriptions for Alice. Strictly speaking, Dr. David can trust another doctor, however, Dr. David's employer may require him to make a more complex trust decision that involves checking the foreign doctor's reputation and expertise area. Making such decisions can be supported by using trust management (TM) systems (Li et al., 2002). In practice, also in a-priori access control systems some authority checks the system for flaws or abuse by logging and auditing user behavior. In fact this is considered to be essential in conventional access control (Sandhu & Samarati, 1994). Often, these audits are conducted without using formal or public procedures. In the auditing approach we have removed the a-priori access control, and provided a formal and automated auditing procedure. Automation allows for fast routine

audits, and is more privacy-friendly than auditing by hand. It remains a question if the user's perception of privacy differs between these approaches.

Deleted:
¶
Conclusions¶
In this chapter we show how the Audit Logic (Cederquist et al., 2005; Corin et al., 2004) can be used in the setting of Electronic Health Records. We outlined the full architecture and we discussed the advantages and drawbacks of this approach, regarding ICT infrastructure and the users' privacy and trust. ¶
We show that our approach requires minimal changes to the infrastructure. At the lower layers, secure audit trails, with all (or most) of the user actions, are already required (Jajodia et al., 1995). By minimizing the a-priori access control and relying on a-posteriori access control we get a more flexible system to adapt to the medical information flow. Yet the audit mechanism provides the necessary assurance, later on, that the staff complied to the relevant policies. A-posteriori access control is convenient when authorizations are not available on the moment that access to medical files is needed.
Moreover, as mentioned before, this type of auditing is required by legislation concerning EHR systems (The US Dpt. of Health and Human Services, n.d.). In the Audit Logic framework such audits are performed by a formal and automated auditing procedure.
In the EHR setting, privacy is an important issue for both patients and medical staff. A-posteriori access control is in theory more intrusive to the user's privacy than a-priori access control. A-priori access control however is also coupled with audits of logs and user actions (Sandhu & Samarati, 1994, 1996). An automatic audit procedure not only enhances the privacy of the patients, but also that of the medical staff, wary perhaps of human auditors that go through the logs by hand. Formal and public auditing procedures make the privacy protection mechanism also more transparent to the patients as well as to the medical staff. In the future we wish to investigate how we can control the actions of the auditors in turn and how we can achieve maximal privacy of the (honest) users with resp
... [1]

References

Abadi, M. (2003, June). Logic in access control. In P. G. Kolaitis (Ed.), Proc. of the 18th Symposium on Logic in Computer Science (LICS) (p. 228-233). IEEE Computer Society Press.

Appel, A. W., & Felten, E. W. (1999). Proof-carrying authentication. In G. Tsudik (Ed.), Proc. of the 6th Conference on Computer and Communications Security (CCS) (p. 52-62). ACM Press.

Becker, M. Y., & Sewell, P. (2004). Cassandra: Flexible trust management, applied to electronic health records. In R. Focardi (Ed.), Proc. of the 17th Computer Security Foundations Workshop (CSFW) (p. 139-154). IEEE Computer Society Press.

Cederquist, J. G., Corin, R., Dekker, M. A. C., Etalle, S., & Hartog, J. I. den. (2005). An audit logic for accountability. In W. Winsborough & A. Sahai (Eds.), Proc. of the 6th International Workshop on Policies for Distributed Systems and Networks (POLICY) (p. 34-43). IEEE Computer Society Press.

Cederquist, J. G., Corin, R., Dekker, M. A. C., Etalle, S., Hartog, J. I. den, & Lenzini, G. (2007). Audit-based Compliance Control, In T. Dimitrakos, F. Martinelli, P. Ryan, S. Schneider (Ed.), the International Journal of Information Security (IJIS), Springer, Berlin.

Conrado, C., Petkovic, M., Veen, M. van der, & Velde, W. van der. (2005). Controlled sharing of personal content using digital rights management. In E. Fern´andez-Medina (Ed.), Proc. of the 4th International Workshop on Security in Information Systems (WOSIS) (p. 173-185).

Corin, R., Etalle, S., Hartog, J. I. den, Lenzini, G., & Staicu, I. (2004). A logic for auditing accountability in decentralized systems. In T. Dimitrakos & F. Martinelli (Eds.), Proc. of the 2nd IFIP Workshop on Formal Aspects in Security and Trust (FAST) (Vol. 173, p. 187-202). Springer, Berlin.

Halpern, J. Y., & Weissman, V. (2003). Using first-order logic to reason about policies. In R. Focardi (Ed.), Proc. of the 16th Computer Security Foundations Workshop (csfw) (p. 187-201). IEEE Computer Society Press.

Jajodia, S., Gadia, S., , & Barghava, G. (1995). Information security: An integrated collection of essays. In M. D. Abrams, S. Jajodia, & H. J. Podell (Eds.), (chap.

Logical design of audit information in relational databases). IEEE Computer Society Press.

Li, N., Mitchell, J., & Winsborough, W. (2002). Design of a role-based trust-management framework. In M. Abadi & S. M. Bellovin (Eds.), Proc. of the Symposium on Research in Security and Privacy (S&P) (pp. 114130). IEEE Computer Society Press.

Rissanen, E., Firozabadi, B. S., & Sergot, M. J. (2004). Discretionary overriding of access control in the privilege calculus. In T. Dimitrakos & F. Martinelli (Eds.), Proc. of the 2nd IFIP workshop on Formal Aspects in Security and Trust (FAST) (p. 219-232). Springer

.

Sandhu, R., & Samarati, P. (1994). Access control: Principles and practice. IEEE Communications Magazine, 32 (9), 4048.

Sandhu, R., & Samarati, P. (1996). Authentication, Access control, and Audit. ACM Computing Survey, 28 (1), 241-243.

Shmatikov, V., & Talcott, C. L. (2005). Reputation-based trust management. Journal of Computer Security, 13 (1), 167-190.

The European Parliament and the Council of the European Union. (2002). UE DIRECTIVE 2002/58/EC on privacy and electronic communications.

The US Dpt. of Health and Human Services. (2000). Summary of the HIPAA Privacy Rule.

Nederlands Normalisatie-Instituut. (2002). NEN 7510:2002 Informatiebeveiliging in de zorg. (in Dutch, Data protection rule for the Dutch health sector).

Whitehead, N., Abadi, M., & Necula, G. C. (2004). By reason and authority: A system for authorization of proof-carrying code. In R. Focardi (Ed.), Proc. of the 17th Computer Security Foundations Workshop (CSFW) (pp. 236 - 250). IEEE Computer Society Press.

Conclusions

In this chapter we show how the Audit Logic (Cederquist et al., 2005; Corin et al., 2004) can be used in the setting of Electronic Health Records. We outlined the full architecture and we discussed the advantages and drawbacks of this approach, regarding ICT infrastructure and the users' privacy and trust.

We show that our approach requires minimal changes to the infrastructure. At the lower layers, secure audit trails, with all (or most) of the user actions, are already required (Jajodia et al., 1995). By minimizing the a-priori access control and relying on a-posteriori access control we get a more flexible system to adapt to the medical information flow. Yet the audit mechanism provides the necessary assurance, later on, that the staff complied to the relevant policies. A-posteriori access control is convenient when authorizations are not available on the moment that access to medical files is needed.

Moreover, as mentioned before, this type of auditing is required by legislation concerning EHR systems (The US Dpt. of Health and Human Services, n.d.). In the Audit Logic framework such audits are performed by a formal and automated auditing procedure. In the EHR setting, privacy is an important issue for both patients and medical staff. A-posteriori access control is in theory more intrusive to the user's privacy than a-priori access control. A-priori access control however is also coupled with audits of logs and user actions (Sandhu & Samarati, 1994, 1996). An automatic audit procedure not only enhances the privacy of the patients, but also that of the medical staff, wary perhaps of human auditors that go through the logs by hand. Formal and public auditing procedures make the privacy protection mechanism also more transparent to the patients as well as to the medical staff. In the future we wish to investigate how we can control the actions of the auditors in turn and how we can achieve maximal privacy of the (honest) users with respect the auditors.