Access Control for Smarter Healthcare Using Policy Spaces[☆]

Claudio A. Ardagna[a], Sabrina De Capitani di Vimercati[a], Sara Foresti[a], Tyrone W. Grandison[b], Sushil Jajodia[*,c], Pierangela Samarati[a]

[a] *DTI - Università degli Studi di Milano, 26013 Crema, Italy*
[b] *IBM Almaden Research Center, San Jose, CA 95120-6099, USA*
[c] *CSIS - George Mason University, Fairfax, VA 22030-4444, USA*

Abstract

A fundamental requirement for the healthcare industry is that the delivery of care comes first and nothing should interfere with it. As a consequence, the access control mechanisms used in healthcare to regulate and restrict the disclosure of data are often bypassed in case of emergencies. This phenomenon, called *"break the glass"*, is a common pattern in healthcare organizations and, though quite useful and mandatory in emergency situations, from a security perspective, it represents a serious system weakness. Malicious users, in fact, can abuse the system by exploiting the break the glass principle to gain unauthorized privileges and accesses.

In this paper, we propose an access control solution aimed at better regulating break the glass exceptions that occur in healthcare systems. Our solution is based on the definition of different policy spaces, a language, and a composition algebra to regulate access to patient data and to balance the rigorous nature of traditional access control systems with the "delivery of care comes first" principle.

Key words: Access control, break the glass, policy spaces, exceptions

[☆] A preliminary version of this paper appeared under the title "Regulating Exceptions in Healthcare using Policy Spaces," in Proc. of the 22nd Annual IFIP WG 11.3 Working Conference on Data and Applications Security, London, U.K., July 2008 [1].

[*] Corresponding author.
Mail Stop 5B5, George Mason University, Fairfax, VA 22030-4444, USA. Fax +1 (703) 993-4776

Email addresses: claudio.ardagna@unimi.it (Claudio A. Ardagna), sabrina.decapitani@unimi.it (Sabrina De Capitani di Vimercati), sara.foresti@unimi.it (Sara Foresti), tyroneg@us.ibm.com (Tyrone W. Grandison), jajodia@gmu.edu (Sushil Jajodia), pierangela.samarati@unimi.it (Pierangela Samarati)

1. Introduction

Healthcare systems support interactions among patients, medical practitioners, insurance companies, and pharmacies. The very sensitive nature of the information managed by these systems requires the balance between two contrasting needs: the need for data, to guarantee proper delivery of care; and the need for keeping data secure, to properly protect the privacy of patients. Access control is the base mechanism that healthcare systems adopt for protecting medical data. Traditional access control models and policies are based on the assumption that possible access requests that will have to be obeyed are known in advance and can therefore be captured by authorizations. However, since in healthcare systems an important requirement is that "nothing interferes with the delivery of care" [2], access control restrictions may need to be bypassed in case of emergencies, especially when the patient's life is at risk. For instance, in case of emergency, a nurse may require (and should be granted) access to data that under "normal" conditions she cannot view. This phenomenon is usually referred to as "break the glass". While useful and mandatory in the name of care delivery, the break the glass can represent a weakness for the security of the system, since allowing it in an unconditional or uncontrolled manner can easily open the door to abuses [3]. To limit (or prevent) such abuses the access control system should minimize the cases in which no regulation applies and the break the glass principle is enforced. An access control system designed to operate in the healthcare scenario should also be flexible and extensible (i.e., it should not be limited to a particular model or language), should protect the privacy of the patients, and should not allow exchange of identity data, in compliance with government legislations (e.g., the Health Insurance Portability and Accountability Act [4] in the United States).

In this paper, we address the need for a flexible and powerful access control system for the healthcare scenario by proposing a model that attempts to balance, on one hand, the rigorous nature of access control models and, on the other hand, the priority of care delivery in healthcare scenarios. We introduce the concept of *policy space* and we describe how policies, which regulate access to medical data, are specified and enforced within each space and how their composition works. Our proposal is aimed at limiting accesses that break the glass, by classifying (a subset of) these access requests as abuses or planned exceptions and by defining specific policies regulating them. In [1], we presented an early version of our proposal that here is extended to the consideration of context information, to allow environment factors to influence how and when a policy is enforced. With respect to the original paper, we also introduce an algebra for combining policies within spaces.

The remainder of this paper is organized as follows. Section 2 presents the requirements of an access control system in the healthcare scenario. Section 3 introduces our assumptions and an illustrative use case. Section 4 defines policy spaces for the management of exceptions. Section 5 defines our language in terms of authorizations, policies, and composition algebra. Section 6 illustrates how the policies are defined in the different spaces. Section 7 describes the policy

evaluation and enforcement. Section 8 discusses related work. Finally, Section 9 presents our concluding remarks.

2. Requirements for Access Control in Healthcare

The design of a comprehensive solution for protecting personal health information should incorporate the specific security, privacy, and integrity requirements arising in a healthcare scenario [5]. In the following, we consider three main categories of requirements: *i)* healthcare professional and patient requirements, *ii)* policy and model requirements, and *iii)* implementation requirements.

Healthcare professional and patient requirements. Medical information is one of the most sensitive classes of information whose confidentiality and integrity is fundamental. Medical information should then be accessible only to the authorized healthcare professionals and patients. In particular, among healthcare professionals, the physician in charge of the patient's care and treatments is usually responsible for the use of the patient's information (*care stewardship*). Patients should be able to access their information and to maintain the control over who can access it by means of a proper notification mechanism, prior approval, and explicit consent. Since patient's health is more important than data confidentiality and integrity, the principle of *delivery of Care Comes First* (*CCF*) must be enforced. The main consequence of this principle is that, in case of emergencies, the medical information of a patient should be accessible to whom is able to deliver care to the patient, even if she is not explicitly authorized. In these situations, access control policies may be overridden. To prevent or discover possible abuses via a post-analysis, every access granted must be recorded in association with the identity of the subject accessing the system.

Policy and model requirements. Since in the healthcare scenario the privileges of subjects often depend on their role (e.g., nurse, doctor, patient), an access control model should support both the definition of authorizations for subjects that can be hierarchically organized in groups and the definition of a hierarchy of roles. Also, access control models and policies for healthcare systems should support the hierarchical organization of the information to facilitate the definition of authorizations.

Access control decisions may often depend on the information stored for the involved patient, which includes different kinds of data with different protection requirements. As a consequence, a fine-grained access control that verifies whether or not to permit access to health information on an attribute/property basis is important. The support for *context information* that triggers the activation of (dynamic) policies is also of paramount importance. In fact, an access control decision may depend on a set of environment factors that influence how and when a policy is enforced. For instance, under normal circumstances the health information of patients stored in the database of a hospital can be accessed only locally, in the event of an epidemic or of a natural disaster (e.g., a hurricane), physicians working remotely can access patient information to treat

victims. Although in emergency situations policies that apply in normal circumstances can be overridden, there are some accesses that should always be prevented, since they cannot help in managing such emergency situations and represent abuses. Such accesses should never be permitted, meaning that policy override must be denied.

Implementation requirements. The policy and model requirements influence the way in which access control systems should be implemented. In particular, to guarantee the enforcement of the CCF principle, it is necessary to provide a way for bypassing the access control mechanisms. This phenomenon is usually called *break the glass* or BtG for short. Every access must be logged to take a-posteriori decisions on the accesses and eventually take countermeasures in case of misuses. To this purpose, an authentication mechanism that verifies the identity claimed by the user accessing the system (especially during an emergency) as well as an auditing mechanism that analyzes security logs must be implemented.

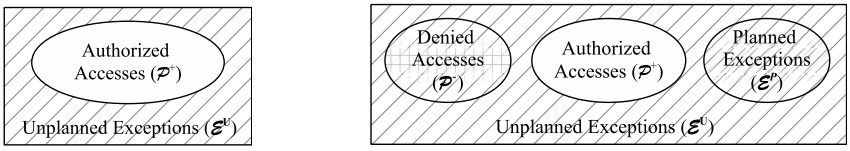
3. System Assumptions

We assume a closed world scenario involving a healthcare provider (e.g., a hospital), where system users (e.g., doctors, nurses, and patients) are known, meaning that their information is available at access request. In addition to the *user id*, which uniquely identifies each user in the system, we assume that users are associated with profiles that contain pairs of the form $\langle attribute\_name, attribute\_value \rangle$ representing their properties. Such a user-related information is both *static* (i.e., it does not change or does not change frequently such as, name, address, and date of birth) and *dynamic* (i.e., it may depend on the specific user session) in nature. For instance, in healthcare systems based on a role-based access control model [6], the roles activated during a user session are an example of dynamic information that is stored within the corresponding user's profile. Another attribute representing dynamic information associated with a patient could indicate that a hearth attack is ongoing.

We refer to the medical data to be protected (i.e., the objects of the access control policies) as *datasets*. Each dataset is characterized by a unique *object identifier*. Datasets can be organized in classes containing groups of datasets that can be collectively referred with a given name and are associated with an object profile (metadata) that provides additional information. Single attributes appearing in user and object profiles are referenced via the usual dot notation. For instance, **Alice**.*Address* denotes attribute *Address* of user with id **Alice**.

We consider an access control system that protects medical datasets based on policies that are modeled as a set of authorizations stating who can or cannot execute which action on which resource. Policies can be further combined to define complex policies regulating access to resources.

In our model, the context information needed to trigger the activation of (dynamic) policies (Section 2) is represented as a set E of pairs of the form $\langle attribute\_name, attribute\_value \rangle$, which describes the *state* of the system at a



(a) Traditional solutions

(b) Our solution

Figure 1: Access control for the healthcare scenario

given time and is subject to frequent changes. Indeed, any event could possibly cause a modification of the state of the system and therefore of the set E of attributes. By checking the values of the attributes in E, it is possible to recognize emergency situations. For instance, the presence of the pair $\langle$catastrophe,earthquake$\rangle$ in E means that an earthquake has just happened and that the hospital must be ready to give aid to survivors. It is important to note that E contains only environmental information not related to a specific patient.

In the remainder of the paper, we refer our examples to a fictitious Mount Cedar (MC) Hospital described below.

Example 3.1. *The Mount Cedar (MC) Hospital is a children's hospital and includes the following main actors:*

- the children *who need treatment;*
- the children's relatives *who are responsible for hospitalized children;*
- the doctors *who are responsible for providing care to children;*
- the nurses *who are responsible for helping the doctors;*
- the social workers *who are possibly responsible for helping the children in case of trauma or abuses;*
- the police men *who are responsible for investigating and establishing possible criminal charges and responsibilities, in the cases of trauma or abuses.*

4. Exception-Aware Policy Spaces for Healthcare

Access control systems tailored to the healthcare scenario are characterized, as illustrated in Figure 1(a), by the presence of two policy spaces, namely $\mathcal{P}^+$ and $\mathcal{E}^U$. In general, a policy space can be defined as a policy repository, whose policies regulate access to resources. Space $\mathcal{P}^+$ represents *authorized accesses* and regulates common practice requests. If a request satisfies a policy in $\mathcal{P}^+$, then it is permitted. Space $\mathcal{E}^U$ represents *unplanned exceptions* and regulates

all those requests for which policies in $\mathcal{P}^+$ are not applicable. In healthcare scenario, since nothing should interfere with the delivery of care, space $\mathcal{P}^+$ may be bypassed, especially when a patient's life is at risk. In these emergency situations, although the requester does not have the authorization to perform the action requested (i.e., no policy in $\mathcal{P}^+$ applies), the request is always permitted by the policies in $\mathcal{E}^U$, thus breaking the glass. This solution makes the system vulnerable to malicious users that may exploit the BtG principle for breaching the patient's privacy also when it is not strictly necessary.

To limit the possible abuses exploiting the BtG option, we put forward the idea of defining a solution based on the following extended set of *policy spaces* (see Figure 1(b)).

- *Authorized Accesses* ($\mathcal{P}^+$). Space $\mathcal{P}^+$ corresponds to traditional access control policies. Intuitively, $\mathcal{P}^+$ includes positive authorizations regulating *'common practice'*.

- *Denied Accesses* ($\mathcal{P}^-$). Space $\mathcal{P}^-$ corresponds to access control policies that are used to prevent abuses. Policies in this space are meant to limit exceptions that can result in unauthorized accesses exploiting the BtG option. As a consequence, they must be strictly enforced and do not allow any exception. These policies reflect actions that cannot help even in emergency situations, but can only cause privacy breaches and must be avoided. They can be specified *a priori* to eliminate accesses that should never be authorized (i.e., accesses that should not be bypassed by BtG) and/or inserted *a posteriori* because of observed abuses.

- *Planned Exceptions* ($\mathcal{E}^P$). Space $\mathcal{E}^P$ corresponds to policies regulating access requests that do not fall into the normal routine, as well as activities that should not be normally allowed. Policies in $\mathcal{E}^P$ are associated with, and indexed by, conditions on the context information represented by attributes in E and on dynamic information in the profiles (e.g., status of the patient), which are used to restrict their applicability. Policies in $\mathcal{E}^P$ cannot override policies in $\mathcal{P}^-$. Policies in $\mathcal{E}^P$ regulate exceptions that can be foreseen, for example, according to past observations.

- *Unplanned Exceptions* ($\mathcal{E}^U$). Space $\mathcal{E}^U$ corresponds to policies regulating all access requests not covered by the previous policy spaces ($\mathcal{P}^+, \mathcal{P}^-$, and $\mathcal{E}^P$). Space $\mathcal{E}^U$ is composed of two sub-spaces, denoted $\mathcal{E}^{U^+}$ and $\mathcal{E}^{U^-}$, respectively. The applicability of the policies in these two subspaces strictly depends on the state of the system (i.e., attributes in E) and on dynamic information in the profiles. Specifically, $\mathcal{E}^{U^-}$ enforces the *deny-all* default policy and is applicable to all requests that happen in non-emergency cases, when the enforcement of the BtG principle would be an abuse. Space $\mathcal{E}^{U^+}$ enforces the *permit-all* default policy and is applicable to all requests that happen in emergency situations, thus allowing all accesses not explicitly allowed or denied by policies in other spaces. All the ac-

cesses falling in $\mathcal{E}^U$ are inserted into an auditing log for their a posteriori analysis.

An important characteristic of these spaces is that they are not limited to a particular access control model, language, or implementation. As a consequence, our solution allows the incorporation of the policy language that better suits the requirements of each particular situation. The independence from the specific language adopted allows *backward compatibility* of the policy space definitions, meaning that our solution can immediately be integrated with existing access control models with limited effort. Another advantage of our solution is that policy spaces can be incrementally populated by analyzing accesses that are permitted or denied by policies in $\mathcal{E}^{U^+}$ and $\mathcal{E}^{U^-}$, respectively, through an auditing process. The auditing process can show access requests that: *i)* correspond to common practice and should be explicitly permitted by appropriate policies in $\mathcal{P}^+$; *ii)* should never be admitted and should be explicitly denied by defining appropriate policies in $\mathcal{P}^-$; *iii)* are frequent but not common and should be captured by appropriate exceptions in $\mathcal{E}^P$.

5. Authorization and Policy Definition

We present an access control model including authorization and policy definition, and an algebra for composing them.

5.1. Authorization

Positive authorizations establish who can execute which action on which resource [7]. In line with recent advancements, we allow the specification of authorizations with reference to generic attributes/properties of the users (e.g., name, citizenship, occupation) and metadata (e.g., owner, creation date) of the resources involved [8, 9, 10]. Authorizations are then specified over a set of subjects and a set of objects, which are identified by a boolean formula defined on their attributes [11], and are formally defined as follows.

Definition 5.1 (Authorization). *An* authorization *is a quadruple of the form* [env_cond|subject,object,action]*, where:*

- env_cond *is a boolean formula that evaluates conditions on the context E as well as on subject and object profiles;*
- subject *is a boolean formula that can refer to a set of subjects depending on whether they satisfy certain conditions evaluated on their profiles* (*e.g., user's properties, user's membership in groups, active roles*)*;*
- object *is a boolean formula that identifies the set of objects to which the authorization applies and that evaluates conditions over their profiles;*
- action *is an action, or a set thereof, that* subject *is authorized to perform on* object*.*

An authorization [*env_cond*|*subject*,*object*,*action*] states that if *env_cond* evaluates to true, *subject* is permitted to perform *action* on *object*. Conditions in *env_cond* are based on generic predicates that can evaluate attributes in E (e.g., the state of the application, the number of accesses to a given object, time/date) and attributes in the subject and object profiles (e.g., the status of the patient). To make it possible to refer to the subject or object of the request being evaluated while avoiding the introduction of variables in the language, we use keywords **user** and **object** in the definition of elements *subject* and *object* of an authorization. Specifically, keyword **user** refers to the user requesting access and keyword **object** refers to the requested object. The appearances in a conditional expression of **user** and **object** keywords are intended to be substituted with actual request parameters during the runtime evaluation of the access control policy.

Syntactically, *subject*, *object*, and *env_cond* are represented as boolean formulas, composed of terms of the form "*attribute op value*", where *op* is a generic operator defined on the domain of *attribute*. For instance, an authorization stating that "a user with role doctor can read medical documents created before year 2000 under any environment condition" can be expressed as: [*any*|**user**.*role*=*Doctor*,**object**.*creation_date*<*2000/01/01*,{*read*}].

The definition of restrictions on the environment in our model represents the most important paradigm shift with respect to access control systems operating in the healthcare scenario. Instead of being simply embedded in the definition of the subject/object, environment condition *env_cond* is defined as a separate element in the authorizations. In the following, given an authorization A, notation A.*env_cond* will be used to denote the environment condition characterizing A.

5.2. Policy

Informally, a policy is a composition of a set (possibly singleton) of authorizations. Moreover, policies and authorizations can be further combined using a composition algebra. In the following, we introduce the three operators characterizing our algebra [12].

- *Addition* (+). Binary operator + models the disjunction of two policies. Given two policies P_1 and P_2, $P_1 + P_2$ means that an access is granted if at least one of the policies is satisfied. For instance, if access to data of the patients should be authorized to the doctors *or* nurses responsible for them, authorizations A_1 = [*any*|**user**.*role*=*Doctor*, **object**.*type*=*medical_data* ∧ **object**.*doctorId*=**user**.*id*, {*read*}] and A_2 = [*any*|**user**.*role*=*Nurse*, **object**.*type*=*medical_data* ∧ **object**.*nurseId*=**user**.*id*, {*read*}] are composed as $P = A_1 + A_2$.

- *Conjunction* (&). Binary operator & models the conjunction of two policies. Given two policies P_1 and P_2, P_1 & P_2 means that an access is granted if both policies are satisfied. For instance, suppose that the hospital specifies authorization A_1 above, while patient P001 authorizes

access to her medical data only to doctors with at least 10 years of experience through A_3 = [*any*|**user**.*role*=*Doctor* $\wedge$ **user**.*experience*>10, **object**.*type*=*medical_data* $\wedge$ **object**.*patient*="P001", {*read*}], then the two authorizations are composed as $P = A_1 \& A_3$.

- *Subtraction* ($-$). Binary operator $-$ models the case of a policy restricted by another policy (exception). Given two policies P_1 and P_2, $P_1 - P_2$ restricts policy P_1 to requests for which policy P_2 is not satisfied. Access is then granted if P_1 is satisfied and P_2 is not. For instance, if access to medical data is permitted to all nurses (A_2) except for those with less than 5 years of experience (A_4 = [*any*|**user**.*role*=*Nurse* $\wedge$ **user**.*experience*<5,**object**.*type*=*medical_data*,{*read*}]), then the two authorizations are composed as $P = A_2 - A_4$.

Based on the definitions of the algebraic operators described above, we define a *policy expression* as a composition of policies by using the following BNF syntax:

policy_expression ::= P | policy_expression + policy_expression |
| policy_expression & policy_expression |
| policy_expression − policy_expression.

where P is a policy.

We can now formally define a policy as a composition of policies, associated with an environment condition, as follows.

Definition 5.2 (Policy). *A* policy P *is of the form* [env_cond|*policy_expression*] *such that if* $P_1 \ldots P_n$ *are the policies in policy_expression, then* $\forall P_i$*, with i=1...n,* P_i.env_cond $\rightarrow$ P.env_cond.

The environment condition of a policy P, denoted $P.env\_cond$, must be less restrictive than the environment conditions $P_1.env\_cond \ldots P_n.env\_cond$ of all the policies composing it (i.e., $P_i.env\_cond \rightarrow P.env\_cond$), meaning that the satisfaction of any condition $P_i.env\_cond$, $i = 1, \ldots, n$, implies the satisfaction of $P.env\_cond$ while the vice versa is not true. For instance, policy P_1=[*catastrophe*='earthquake'|P_1 + P_2], where $P_1.env\_cond$ = (*catastrophe*='earthquake' $\wedge$ *criticality*>5) and $P_2.env\_cond$ = (*catastrophe*='earthquake' $\wedge$ *criticality*>7), is such that $P_1.env\_cond \rightarrow P.env\_cond$ and $P_2.env\_cond \rightarrow P.env\_cond$.

Note that each policy space in our system is a composition of policies, using the algebra described above. In the following, when clear from the context, we will use the terms authorization and policy interchangeably.

6. Policy Space Languages

We now show how policies are defined in the different spaces.

6.1. Policies for Spaces $\mathcal{P}^+$ and $\mathcal{P}^-$

Policies in $\mathcal{P}^+$ regulate normal accesses and correspond to positive authorizations managed by traditional access control systems. For instance, an authorization can state that a user can access the medical data of her patients when she activates the Doctor or Nurse role.

Policies in space $\mathcal{P}^-$, instead, regulate accesses to be denied. They are used to prevent abuses and provide confinement to possible exceptions, thus reducing accesses that exploit the BtG principle. Denials are strictly enforced, meaning that policies in $\mathcal{P}^+$ and exception policies in $\mathcal{E}^P$ and $\mathcal{E}^U$ can be evaluated if and only if policies in $\mathcal{P}^-$ do not apply or their evaluation is 'false'. The main goal of policies in $\mathcal{P}^-$ is then to reduce the number of access requests evaluated in the exception spaces $\mathcal{E}^P$ and $\mathcal{E}^U$, also limiting cases in which malicious users submit ad-hoc requests that break the glass. Policies in $\mathcal{P}^-$ are specified *a priori* for those unwanted accesses that can be foreseen at system setup, or inserted *a posteriori* because of abuses observed through the analysis of audit logs produced by accesses in $\mathcal{E}^U$. Note that all authorizations in $\mathcal{P}^+$ and $\mathcal{P}^-$ would have the form [*any*|*subject*,*object*,*action*] (see Definition 5.1), where environment condition *any* indicates that they need to be considered for every request.

Considering Example 3.1, Figure 2 shows a possible set of authorizations in $\mathcal{P}^+$ and $\mathcal{P}^-$. For the sake of clarity, field *env_cond* is omitted, since it is assumed to be always equal to *any*. In our example, authorizations in $\mathcal{P}^+$ ($\mathcal{P}^-$, respectively) are combined using operator + of our algebra.

6.2. Exception-Based Policies for Space $\mathcal{E}^P$

Space $\mathcal{E}^P$ allows the definition of policies used to regulate requests that cannot be considered 'normal routine'. In healthcare, *emergency requests* include all accesses necessary to preserve the health of the patients, but are inherently different from normal routine. For instance, a nurse on duty can access medical data of each patient entering the trauma ward.

In existing solutions, emergency requests fall in $\mathcal{E}^U$ and are granted by breaking the glass. The main reason for defining authorizations within space $\mathcal{E}^P$ is to limit potential abuses of the BtG principle, by adequately regulating situations that can be foreseen. Space $\mathcal{E}^P$ contains permissions that extend $\mathcal{P}^+$ and are applicable only when *env_cond* evaluates to true, meaning that an exceptional situation occurs. Like for policies in $\mathcal{P}^+$, also policies in $\mathcal{E}^P$ do not override $\mathcal{P}^-$.

An authorization in $\mathcal{E}^P$ is formally defined as follows.

Definition 6.1 (Authorization in $\mathcal{E}^P$). *An* authorization *in $\mathcal{E}^P$ has the form* [env_cond|subject,object,action] ← obligations*, where* subject*,* object*,* action*, and* env_cond *are as in Definition 5.1, and field* obligations *identifies the actions that must be performed if the access is granted.*

An authorization [*env_cond*|*subject*,*object*,*action*] ← *obligations* states that if *env_cond* evaluates to true, *subject* can perform *action* on *object*, and *obligations* must be enforced. Field *obligations* includes the actions that must

	Authorization	Description
A1	[**user**.*role*=*Nurse* ∧ **user**.*startDuty*<`time()` ∧ **user**.*endDuty*>`time()`, **object**.*type*=*medical_data* ∧ **object**.*nurseId*=**user**.*id*, {*read*}]	A nurse on duty can read the medical data of patients under her responsibility
A2	[**user**.*role*=*Doctor* ∧ **user**.*startDuty*<`time()` ∧ **user**.*endDuty*>`time()`, **object**.*type*=*health_record* ∧ **object**.*doctorId*=**user**.*id*, *any*]	A doctor on duty can do any action on the health records of patients under her responsibility
A3	[**user**.*group*=*firstAidTeam*, **object**.*type*=*health_record* ∧ **object**.*clinic*=*firstAid*, {*read*, *write*}]	Users in the first aid team can read and write the health record of patients that are in the first aid clinic

(a) $\mathcal{P}^+$

	Authorization	Description
N1	[**user**.*group*=*medicalStaff*, **object**.*type*=*payment_data*, {*read*}]	Users in the medical staff cannot read the payment data of patients
N2	[**user**.*group*=*administrativeStaff*, **object**.*type*=*medical_data*, {write,update}]	Users in the administrative staff cannot write or modify the medical data of patients
N3	[**user**.*group*=*medicalStaff*, **object**.*type*=*health_record* ∧ **user**.*id*∈**object**.*parents*, {*write*, *update*}]	Users in the medical staff cannot write or modify the health records of their own children

(b) $\mathcal{P}^-$

Figure 2: An example of authorizations in $\mathcal{P}^+$ (a) and $\mathcal{P}^-$ (b)

be enforced after an access is granted [13]. Syntactically, a single obligation is a term of the form `predicate_name`(*arguments*), where arguments is a list, possibly empty, of constant values or attributes. Examples of obligations are: logging, auditing, and patient/supervisor notification. Similarly, policies in $\mathcal{E}^P$ extend Definition 5.2 with obligations.

At an abstract level, $\mathcal{E}^P$ can be seen as space $\mathcal{P}^+$ for situations involving emergencies or in general exceptional situations. The key difference to be considered in the specification of $\mathcal{E}^P$ is the definition of environment conditions. The applicability of policies/authorizations in $\mathcal{E}^P$ is therefore determined at run time during the evaluation of each request and depends on the satisfaction of *env_cond*. Figure 3 shows, with reference to Example 3.1, three authorizations composing policy space $\mathcal{E}^P$.

6.3. Exception-Based Policies for Space $\mathcal{E}^U$

The space of unplanned exceptions $\mathcal{E}^U$ regulates those access requests that are not evaluated against any policy in $\mathcal{P}^+$, $\mathcal{P}^-$, and $\mathcal{E}^P$. In general, policies in $\mathcal{E}^U$ are assumed to be simple and always grant access according to the BtG principle, since the promptness in reacting against exceptions is fundamental for proper care delivery. In this scenario, a posteriori countermeasures (e.g., logging, auditing) are used to identify potential abuses of the BtG principle. However, they are only palliative, since the confidentiality/integrity of the information

	env_cond	**Rule** (*subject*,*object*,*action*)	**Description**
E1	*state*=*emergency*	[**user**.*role*=*Nurse* ∧ **user**.*startDuty* < `time()` ∧ **user**.*endDuty* > `time()`, **object**.*type*=*medical_data* ∧ **object**.*nurseId*≠**user**.*id*, {*read*}]← `fill_in_form`(privacyform)	A nurse on duty can read the medical data of patients not under her responsibility in case of emergencies after filling in a privacy form
E2	*state*=*emergency*	[**user**.*role*=*Doctor* ∧ **user**.*startDuty* > `time()` ∧ **user**.*endDuty* < `time()`, **object**.*type*=*medical_data* ∧ **object**.*doctorId*≠**user**.*id*, *any*]← -	A doctor on duty can do any action on medical data of patients not under her responsibility in case of emergencies
E3	*purpose*=*investigation*	[**user**.*role*=*PoliceMan*, **object**.*type*=*medical data*, {*read*}]← `notify`(**object**.*dataCollector*)	A police man can read the medical data of each patient in case of criminal investigation, notifying the data collector (i.e., the hospital)

Figure 3: An example of authorizations in $\mathcal{E}^P$

can be violated before taking countermeasures. For instance, if $\mathcal{E}^U$ space is composed of a single policy that grants all accesses it can be breached by simply requesting an access that is neither granted by $\mathcal{P}^+$ or $\mathcal{E}^P$, nor denied by $\mathcal{P}^-$.

Our solution is based on the idea that policies in $\mathcal{E}^U$ should be adopted to react to requests that happen in case of unplanned exceptions only. To limit cases in which malicious users gain unauthorized accesses by presenting ad-hoc requests that break the glass, we split $\mathcal{E}^U$ in two policy sub-spaces: $\mathcal{E}^{U^-}$ that forbids BtG under some specific environment conditions *env_cond*, and $\mathcal{E}^{U^+}$ that implements BtG. $\mathcal{E}^{U^-}$ contains a set of authorizations of the form [*env_cond*|*subject*,*object*,any] ← *obligations* that are checked first and deny all requests for which there is at least one authorization that applies to the request. $\mathcal{E}^{U^+}$ contains a set of authorizations of the form [*any*|*subject*,*object*,any] ← *obligations* that enforce the BtG principle. Although $\mathcal{E}^{U^-}$ and $\mathcal{E}^{U^+}$ could be arbitrary sets of policies that may be activated according to specific conditions on the environment state, for concreteness and fix ideas we populate them as follows: $\mathcal{E}^{U^-}$ = [*state*≠"emergency"|any,any,any] ← *obligations* and $\mathcal{E}^{U^+}$ = [*any*|any,any,any] ← *obligations*.

Obligations in spaces $\mathcal{E}^{U^+}$ and $\mathcal{E}^{U^-}$ provide post-incident capabilities (i.e., auditing and logging [14]) that can be used a posteriori both for overseeing the access requests in a given domain (e.g., a hospital department) and for better redistributing policies among spaces.

All the access requests that fall in $\mathcal{E}^{U^+}$ and $\mathcal{E}^{U^-}$ and directed to a set of objects in a given domain are notified to a supervisor. Based on log files, the supervisor takes countermeasures for misbehaving subjects or formalizes repeated observed behaviors by defining additional policies in spaces $\mathcal{P}^+$, $\mathcal{P}^-$, or $\mathcal{E}^P$. Cross-domain activities should be managed in collaboration by different supervisors, so to mitigate the risk of abuses by individual supervisors and of incorrect policy specifications. To better clarify audit functionality, suppose that an em-

ployee of the hospital responsible for cleaning the surgical equipment reads the type of patient disease to prepare the suitable cleaning protocol (as each infectious disease has a different cleaning protocol). For each request submitted by this employee and allowed in $\mathcal{E}^U$, the access is logged and an auditing process must be performed. Since the request is admittable and should be always allowed, a policy should be defined in $\mathcal{P}^+$ by the supervisor to regulate this scenario. By contrast, suppose that a malicious employee, in addition to the type of patient disease, also accesses the personal data of the patient to sell them to an insurance company. In this case, the supervisor is able to apply remedies, for example, initiating termination procedures and defining additional policies in $\mathcal{P}^-$ to avoid, in the future, similar unauthorized accesses.

7. Policy Evaluation and Enforcement

We now discuss how policies in policy spaces are evaluated and enforced.

Access requests are of the form $\langle$*user_id*, *action*, *object*, *purposes*$\rangle$, where *user_id* is the identifier characterizing the requester, *action* is the action that is being requested, *object* is the object on which the requester wishes to perform the action, and *purposes* is the purpose (or set thereof) for which the access is requested. We assume that the personal information of patients is collected for a given *purpose* (e.g., providing patient care). In normal scenarios, data cannot be used for any other purpose without the specific informed consent of the patient it concerns, while in exception scenarios, restrictions to the purpose can be expressed in *env_cond* and used to evaluate the applicability of the policies. The purpose of a request is also stored in log files, to possibly identify fraudulent use of data and take adequate countermeasures.

When an access request is received, the sets of applicable policies in $\mathcal{P}^+$, $\mathcal{P}^-$, $\mathcal{E}^P$, and $\mathcal{E}^{U^-}$ are selected by evaluating environmental conditions *env_cond* using context information E and the information stored in the subject and object profiles. Authorization in $\mathcal{E}^{U^+}$ is instead always applicable as a default policy (i.e., permit all).

Figure 4 shows the policy evaluation flow, where each policy space is represented with a box that receives as input an access request and returns as output an evaluation response. We assume that, for each of the spaces introduced, the policy evaluation can result in three outcomes: *i) true,* positive evaluation; *ii) false,* negative evaluation; *iii) unknown,* no applicable policy has been found. Based on the response, the access request is granted, denied, or forwarded to the next policy space.

The evaluation process works as follows. First, policies in $\mathcal{P}^-$ are evaluated against the access request. If the evaluation result is 'true', the access is denied. Otherwise, the request is redirected and evaluated against the set of applicable policies in $\mathcal{P}^+$. If the evaluation result of policies in $\mathcal{P}^+$ is 'true', the access is granted. Otherwise, the request is redirected and evaluated in space $\mathcal{E}^P$ of planned exceptions. Like for policies in $\mathcal{P}^+$, if the evaluation is 'true', the access is granted, otherwise, the request falls in $\mathcal{E}^{U^-}$. Note that the evaluation of applicable policies must take into consideration complex policies and their

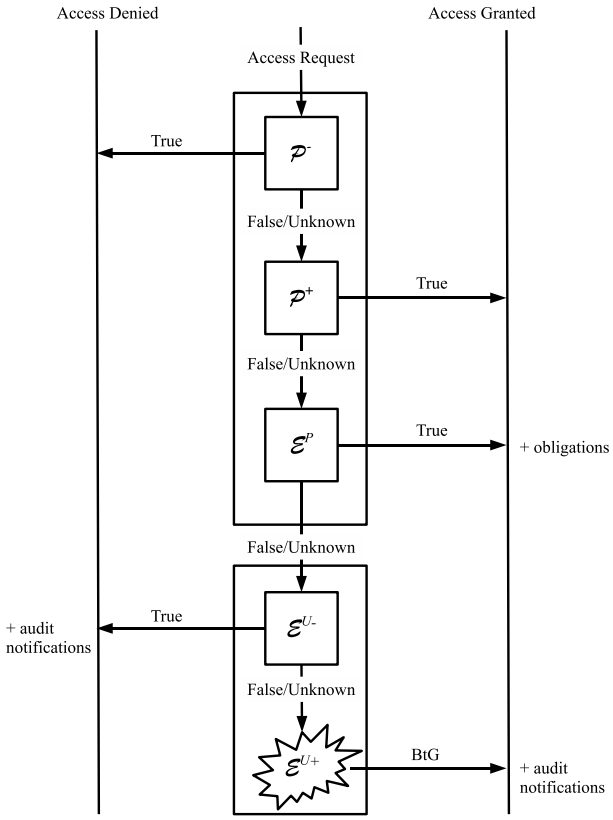


Figure 4: Policy evaluation flow

composition operators. As an example, let $P = P_1$ & P_2 be an applicable policy, since the environment condition $P.env\_cond$ is satisfied at request time. The evaluation of P requires to examine the policies contained in it and to apply the composition operator. In our example, P_1 and P_2 can be applicable or not based on environment conditions $P_1.env\_cond$ and $P_2.env\_cond$, respectively. According to operator &, in case at least one of them has no effect ('false') or is not applicable ('unknown'), the evaluation of P is 'false'.

When a request is redirected to $\mathcal{E}^{U^-}$, if the environment state of the request is not critical, the access is denied. Otherwise, the access is granted in $\mathcal{E}^{U^+}$ by BtG, and the request is inserted into a log file. In both cases, the supervisor receives a notification of the request and the result of the evaluation. The supervisor is then able to perform a subsequent analysis to possibly individuate abuses or access requests that should be regulated by the defining a proper set of policies in spaces $\mathcal{P}^+$, $\mathcal{P}^-$, or $\mathcal{E}^P$.

Example 7.1. *Consider the MC hospital described in Example 3.1 and the poli-*

cies in Figures 2 and 3 that regulate the accesses to the MC's computer system. Suppose now that Timothy, who is four-year old, is currently examined at the MC hospital. Timothy was brought into MC's first aid clinic by his mother, Eva, late Wednesday evening. The admitting staff observed that Timothy suffered from several contusions all over his body, a fractured rib, and a distorted shoulder.

Let us walk through the events that would occur in this all too common situation.

Initially, Timothy's doctor in the first aid clinic, Dr Murthy, takes a history, fills in the electronic patient record, assigns the patient to a care team, and orders a series of examinations. These actions are allowed since no policy in $\mathcal{P}^-$ is satisfied, while authorization A3 in Figure 2(a) evaluates to true. When the examination results return, Dr Murthy suspects child abuse and initiates the corresponding protocol setting the state *as "critical".*

As a consequence, the police and social services are informed of Timothy's situation. Now, the police officer responsible for the criminal investigation, Lt. Starke, requires access to Timothy's medical information for purpose investigation. First, policies in $\mathcal{P}^-$ are evaluated and, since the request is not an access abuse*, it evaluates to 'false'. Then, the request is evaluated against policies in $\mathcal{P}^+$ and $\mathcal{E}^P$ (see Figure 2(a) and Figure 3). The request is not a* normal practice *and then policies in $\mathcal{P}^+$ evaluate to false. Exception E3, instead, is applicable based on the environment and evaluates to true. According to the discussion in Section 6, access is allowed and Lt. Starke fulfills the obligations by informing MC of the access.*

At the same time, the social worker responsible for helping abused kids reported by MC's staff, Miss Woodrow, requests access to Timothy's health record. Like for the case with Lt. Starke, the $\mathcal{P}^-$ policies evaluate to 'false' and the request is evaluated against applicable policies in $\mathcal{P}^+$ first and $\mathcal{E}^P$ then. However, both the $\mathcal{P}^+$ and $\mathcal{E}^P$ policy evaluation results in 'unknown', since no authorization is applicable. The request is then redirected to space $\mathcal{E}^{U^-}$. As the state of the patient is critical, access is forwarded to $\mathcal{E}^{U^+}$ and granted to Miss Woodrow who breaks the glass*. The supervisor is then awakened and the audit process starts. Since Miss Woodrow's access is common in case of child abuse, the supervisor may define an additional policy in $\mathcal{E}^P$ to manage such a request in the future and avoid further* break the glass *accesses.*

Finally, suppose that Timothy's health deteriorates (and thus state *is set to "emergency") and Dr Murthy is not on duty. If the current doctor on duty, Dr Wright, submits an access request to read Timothy's medical data, under "emergency" situation, then the request falls in $\mathcal{E}^P$. Considering policies in Figure 3, Dr Wright satisfies authorization E2 and accesses Timothy's medical data.*

8. Related Work

Although a number of projects and research works about access control models and languages have been presented in the last few years [8, 10, 11,

15], only few proposals have attempted to provide a comprehensive framework specifically targeted to the healthcare scenario (e.g., [16]) and, in particular, to the management of exceptions (e.g., [3, 14, 17, 18, 19, 20, 21, 22, 23]).

A work that shares our goal of assigning users higher privileges in case of emergencies or critical situations is that of Gupta et al. [19], where the authors present a criticality-aware access control model for pervasive applications and smart environments. The proposed solution is based on the definition of roles, ordered on the basis of their privileges, and on two access control modes (normal and criticality-aware) that are activated on the basis of a criticality level. This solution assigns a *system role* to each user on the basis of her responsibilities. It then associates a *space-role* on the basis of the system roles and context information. A *promoterole* function assigns space-roles with higher privileges to the users when a critical event happens and the criticality-aware mode is entered. A limitation of this solution with respect to our proposal is that it does not provide a fine-grain control on when and who can exercise the "extra-privileges" needed for taking care of an emergency situation. For instance, suppose that in case of an emergency involving a patient affected by tuberculosis, only nurses with a specialization in respiratory disorders should be permitted to read the medical data of the patient to take the appropriate action. This situation cannot be managed by the promoterole solution, where only the role of the user (i.e., "nurse") and the context information (i.e., "health emergency") are considered to determine the promoted role (i.e., the role with 'higher privileges'). It could only be managed by creating ad-hoc roles, but this solution may result in an unrestricted proliferation of roles.

Another important line of contribution focused on extending eXtensible Access Control Markup Language (XACML) [10] with a new security and privacy authorization profile [23], which fosters interoperability in the healthcare context and introduces mechanisms to enforce authorization policies controlling access to information, possibly stored across enterprise boundaries. This profile provides interoperability by offering common vocabularies and semantics for policy request/response, policy lifecycle, and policy enforcement. It also includes definition and evaluation of patient consent directives expressed as HL7 (Health Level Seven International) confidentiality codes, and integrates HL7 permission codes that are associated with the users according to their roles. Several examples of use of this new profile show its suitability for the definition of policies in healthcare scenarios, including the definition of policies in our spaces [24]. However, the proposed model mainly focuses on the definition of an infrastructure for an interoperable, secure, privacy-oriented, and fine-grained access control, but it does not consider the management of exceptions in emergency situations. Although policies can be defined for emergency situations (and requests can be tagged with an *emergency purpose*), they cannot regulate exceptions, BtG scenarios, and security threats caused by unexpected events.

A different but related line of research is represented by Hippocratic Databases (HDBs) [25]. HDBs are founded on ten principles: *1)* purpose specification; *2)* respondents consent; *3)* collection limitation; *4)* use limited to specified purposes; *5)* limited disclosure; *6)* limited retention; *7)* accuracy;

8) safety; *9)* openness; and *10)* compliance. Hippocratic Databases are used in different contexts. First of all, they have been studied to limit the disclosure of sensitive information and provide fine-grained access control [26]. Also, they provide specific mechanisms allowing data owners to audit (a posteriori) if the releases of their information are compliant with the declared purposes [27]. Finally, Hippocratic Databases have been adopted in the healthcare scenario (e.g., [28, 29, 30]). These solutions address the problem of defining a system compliant with privacy and security laws, providing audit functionality. Both the proposals in [28, 29] address this problem by combining query re-writing, for access control enforcement, and HDB compliance auditing systems. These proposals adopting HDBs in the healthcare scenario are aimed to the protection of medical data when sensitive information is released to third parties, with a purpose different from "delivery of care". As a consequence, these solutions do not support the BtG principle and exception management that we capture through the definition of different policy spaces.

9. Conclusions

In healthcare, nothing should interfere with the delivery of care. Solutions based on the break the glass principle are usually adopted to subvert access control decisions in emergency situations. The break the glass scenario, however, represents a backdoor for malicious users that try to gain unauthorized accesses. In this paper, we presented an exception-based access control solution whose main goal is to better control the break the glass attempts in healthcare systems, to reduce possible breaches in the patients' privacy. To this aim, we introduced the definition of policy spaces that balance the rigorous nature of traditional access control systems with the delivery of care comes first principle. We illustrated how policies are specified and enforced within each space, and how these policy spaces are composed by means of an algebra.

Acknowledgements

This work was supported in part by the EU (project "PrimeLife", 216483); Italian MIUR (project "PEPPER" 2008SY2PH4); and NSF (CT-20013A, CT-0716567, CT-0716323, CT-0627493, and CCF-1037987).

References

[1] C. Ardagna, S. De Capitani di Vimercati, T. Grandison, S. Jajodia, P. Samarati, Regulating exceptions in healthcare using policy spaces, in: Proc. of the 22nd Annual IFIP WG 11.3 Working Conference on Data and Applications Security, London, U.K., 2008.

[2] T. Grandison, J. Davis, The impact of industry constraints on model-driven data disclosure controls, in: Proc. of the 1st International Workshop on Model-Based Trustworthy Health Information Systems, Nashville, Tennessee, USA, 2007.

[3] L. Rostad, O. Edsberg, A study of access control requirements for healthcare systems based on audit trails from access logs, in: Proc. of the 22nd Annual Computer Security Applications Conference, Miami Beach, Florida, USA, 2006.

[4] Health Insurance Portability and Accountability Act, http://www.dhhs.gov/ocr/hipaa/.

[5] B. Blobel, P. Hoepner, R. Joop, S. Karnouskos, G. Kleinhuis, G. I. Stassinopoulos, Using a privilege management infrastructure for secure web-based e-health applications, Computer Communications 26 (16) (2003) 1863–1872.

[6] R. Sandhu, D. Ferraiolo, D. Kuhn, The NIST model for role based access control: Towards a unified standard, in: Proc. of the 5th ACM Workshop on Role Based Access Control, Berlin, Germany, 2000.

[7] P. Samarati, S. De Capitani di Vimercati, Access control: Policies, models, and mechanisms, in: R. Focardi, R. Gorrieri (Eds.), Foundations of Security Analysis and Design, LNCS 2171, Springer-Verlag, 2001.

[8] C. Ardagna, M. Cremonini, S. De Capitani di Vimercati, P. Samarati, A privacy-aware access control system, Journal of Computer Security 16 (4) (2008) 369–392.

[9] S. De Capitani di Vimercati, S. Foresti, P. Samarati, Recent advances in access control, in: M. Gertz, S. Jajodia (Eds.), Handbook of Database Security: Applications and Trends, Springer-Verlag, 2008.

[10] eXtensible Access Control Markup Language (XACML) Version 2.0, http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf (February 2005).

[11] P. Bonatti, P. Samarati, A uniform framework for regulating service access and information release on the Web, Journal of Computer Security 10 (3) (2002) 241–272.

[12] P. Bonatti, S. De Capitani di Vimercati, P. Samarati, An algebra for composing access control policies, ACM Transactions on Information and System Security 5 (1) (2002) 1–35.

[13] C. Bettini, S. Jajodia, X. Wang, D. Wijesekera, Provisions and obligations in policy management and security applications, in: Proc. of the 28th International Conference on Very Large Data Bases, Hong Kong, China, 2002.

[14] R. Bhatti, T. Grandison, Towards improved privacy policy coverage in healthcare using policy refinement, in: Proc. of the 4th VLDB Workshop on Secure Data Management, Vienna, Austria, 2007.

[15] S. Jajodia, P. Samarati, M. Sapino, V. Subrahmanian, Flexible support for multiple access control policies, ACM Transaction On Database Systems 26 (2) (2001) 214–260.

[16] L. Røstad, O. Nytrø, Personalized access control for a personally controlled health record, in: Proc. of the 2nd ACM Workshop on Computer Security Architectures, Alexandria, Virginia, USA, 2008.

[17] D. Keppler, V. Swarup, S. Jajodia, Redirection policies for mission-based information sharing, in: Proc. of the ACM Symposium on Access control Models and Technologies, Lake Tahoe, California, USA, 2006.

[18] M. Han, T. Thiery, X. Song, Managing exceptions in the medical workflow systems, in: Proc. of the 28th International Conference on Software Engineering, Shanghai, China, 2006.

[19] S. Gupta, T. Mukherjee, K. Venkatasubramanian, Criticality aware access control model for pervasive applications, in: Proc. of the 4th Annual IEEE International Conference on Pervasive Computing and Communications, Pisa, Italy, 2006.

[20] J. Longstaff, M. Lockyer, J. Nicholas, The tees confidentiality model: an authorisation model for identities and roles, in: Proc. of the 8th ACM Symposium on Access Control Models and Technologies, Como, Italy, 2003.

[21] A. Brucker, H. Petritsch, Extending access control models with break-glass, in: Proc. of the 14th ACM Symposium on Access Control Models and Technologies, Stresa, Italy, 2009.

[22] A. Ferreira, R. Cruz-Correia, L. Antunes, P. Farinha, E. Oliveira-Palhares, D. Chadwick, A. Costa-Pereira, How to break access control in a controlled manner, in: Proc. of the 19th IEEE Symposium on Computer-Based Medical Systems, Salt Lake City, Utah, USA, 2006.

[23] Cross-Enterprise Security and Privacy Authorization (XSPA) Profile of XACML v2.0 for Healthcare Version 1.0, `http://www.oasis-open.org/committees/document.php?document_id=34164&wg_abbrev=xacml` (August 2009).

[24] Cross-Enterprise Security and Privacy Authorization (XSPA) profile of XACML v2.0 for Healthcare, Implementation Examples, `http://www.oasis-open.org/committees/document.php?document_id=30430` (September 2008).

[25] R. Agrawal, J. Kiernan, R. Srikant, Y. Xu, Hippocratic databases, in: Proc. of the 28th International Conference on Very Large Data Bases, Hong Kong, China, 2002.

[26] K. LeFevre, R. Agrawal, V. Ercegovac, R. Ramakrishnan, Y. Xu, D. DeWitt, Limiting disclosure in hippocratic databases, in: Proc. of the 30th International Conference on Very Large Data Bases, Toronto, Canada, 2004.

[27] R. Agrawal, R. Bayardo, C. Faloutsos, J. Kiernan, R. Rantzau, R. Srikant, Auditing compliance with a hippocratic database, in: Proc. of the 30th International Conference on Very Large Data Bases, Toronto, Canada, 2004.

[28] R. Agrawal, C. Johnson, Securing electronic health records without impeding the flow of information, International Journal of Medical Informatics 76 (5-6) (2007) 471–479.

[29] C. Johnson, T. Grandison, Compliance with data protection laws using hippocratic database active enforcement and auditing, IBM Systems Journal 46 (2) (2007) 255–264.

[30] R. Agrawal, A. Kini, K. LeFevre, A. Wang, Y. Xu, D. Zhou, Managing healthcare data hippocratically, in: Proc. of the 2004 ACM SIGMOD International Conference on Management of Data, Paris, France, 2004.