SGAC: A Patient-Centered Access Control Method

Nghi Huynh*†, Marc Frappier†, Herman Pooda†, Amel Mammar‡ and Régine Laleau*
†Université de Sherbrooke, Québec, Canada
*Université Paris-Est Créteil Val de Marne, France
‡Institut Mines-Télécom/Télécom SudParis, France

***Abstract*—This paper presents SGAC(*Solution de Gestion Automatisée du Consentement, automatised consent management solution*), a new healthcare access control model and its support tool, that manages patient wishes regarding access to their electronic health record (EHR). The development of this model has been achieved in the scope of a project with the Sherbrooke University Hospital, and thus has been adapted to take into account laws and regulations applicable in Québec and Canada, as they set bounds to patient wishes: under strictly defined contexts, patient consent can be overridden to protect his/her life. Moreover, since patient wishes and laws can be in conflict, SGAC provides a mechanism to address this problem. Besides, laws do not cover all cases where consent should be overridden to ensure patient safety. To this end, we define a formal model of SGAC which allows for property verification, making it possible to detect these cases. A performance comparison with XACML (WSO2/Balana) is presented and demonstrates the superior performances of SGAC.**

***Index Terms*—healthcare, access control method, consent management, formal model, verification.**

I. Introduction

Before being electronic, patient data were stored physically in each health centre. In Québec, access to health records is managed by specially trained staff, the archivists, who are responsible for applying the laws and regulations to access request. Laws set a frame within which patients can manage access to their health record, as long as they are not endangering themselves. Access control in healthcare knows two major contentious concerns: patient data confidentiality and patient safety. The former is about non-disclosure of data their owner would judge confidential; the latter is about the rules not being too restrictive and a burden for the health worker when requesting all necessary data to provide suitable care to the patient. Having patients specifying access rules to their records (thus expressing their consent) is a way to address the first concern. To address the second concern, laws and regulations set a frame that allows accesses to patient data without consent under strictly defined contexts. The problems this approach rises are multiple: laws generally set frame only for exceptional cases and not for everyday care, thus it does not always allow to override patient consent in order to give him/her suitable care, for instance when the patient is hiding important data like medicinal allergies. Furthermore, conflicts may arise between hospital rules, which define health workers regular access, patient rules, and break-the-glass rules which must provide full access to the physicians in strictly defined contexts.

In this paper, we present an access control method named SGAC (*Solution de gestion automatisée du consentement)/(Automated consent management solution)*, which offers a resolution mechanism to the different conflicts that may occur between rules from different sources. This method allows formal verification in order to detect cases where suitable care cannot be given.

The rest of this paper is structured as follows. Section II provides requirements for access control and consent management used at the Sherbrooke University Hospital (CHUS) and scenarios illustrating expected behaviours. Section III introduces SGAC. We illustrate how SGAC behaves in Section IV. Section V provides a complete formal model of SGAC. Section VI compares our findings with similar work on access control in healthcare. A performance comparison with XACML [1] is given in section VII. We conclude this paper with an appraisal of our work in Section VIII.

II. Access Control Requirements at CHUS

Our access control model has been designed to meet the requirements of CHUS within the context of applicable laws on privacy protection in Québec and Canada. We believe these requirements are sufficiently general to be applicable in other countries as well.

- **Req. 1**: The patient's consent must be obtained in order to provide access to his/her electronic health record (EHR).
- **Req. 2**: A patient can grant or deny access to any of his/her EHR to any person of the hospital staff.
- **Req. 3**: As required by the laws of Québec, when the patient's life is in danger, the medical staff must have access to his/her EHR, without regards to his/her consent. Other conditions, like a court order, can also override the patient's rules.
- **Req. 4**: Rules can be specified for a single person or a group of persons. Persons can be grouped according to any criteria, like functional role, work group, departments, care unit, etc.

Req. 5: Rules can be specified for a single record or a group of records. Records can be grouped according to the taxonomy commonly used for EHR.

Req. 6: When several rules are applicable for a user request, they must be ordered according to the following priority to determine which rule prevails: the rules prescribed by laws override the patient's rules; the rules of the patient override the rules of the hospital.

Req. 7: For two rules at the same level of priority, a rule which targets a group of person G_1 has precedence over a rule targeting a less specific group of persons G_2, (ie, when $G_1 \subset G_2$).

Req. 8: For two rules at the same level of priority, when neither of the two groups of persons is more specific than the other (*i.e.*, when $\neg(G_1 \subset G_2 \vee G_2 \subset G_1)$), a prohibition rule overrides a permission rule.

Req. 9: Each rule has a condition that determines its applicability. This condition can refer to any attribute that can be computed using the context of the clinical system (*e.g.*, the state of the patient, the presence of the patient in the hospital, etc).

Req. 10: The access control system shall be able to handle a very large volume of data, hundreds of thousands of patients and rules.

To illustrate some of these requirements, we provide the following scenarios. In these scenarios, Anna and Sam are patients, Alice is a nurse and Bob is a doctor. For each scenario, we refer to the requirements that it illustrates.

Scenario 1 - Group prohibition

Anna wants to deny access to her psychiatric records to the entire hospital staff.
Requirements: Req. 2, Req. 4 and Req. 5.

Scenario 2 - Record taxonomy

Sam has two laboratory results, *lab1* and *lab2*. He authorises hospital staff to access all his laboratory results. Later, Sam receives a third laboratory result, *lab3*.
Expected behaviour: all requests from hospital staff to access Sam's laboratory results, including *lab3* should be permitted.
Requirements: Req. 5.

Scenario 3 - Priority

Sam wishes to grant all hospital staff access to his blood tests, DNA tests and psychiatric records. However, there is a law that restricts access to psychiatric records to psychiatrists only.
Expected behaviour: all requests from hospital staff, other than psychiatrists, to access Sam's psychiatric records are denied; all requests of hospital staff to access Sam's blood and DNA record are permitted.
Requirements: Req. 6.

Scenario 4 - Specificity

Anna wants to deny Alice access to her laboratory results. Anna also has a rule granting nurses access to her laboratory results.
Expected behaviour: all requests from nurses, except Alice, to access Anna's laboratory results are permitted; Alice can't access Anna's laboratory results.
Requirements: Req. 7.

Scenario 5 - User group specificity

Anna specifies two rules: the first rule denies emergency staff access to her EHR; the second rule grants general practitioners access to her EHR. Bob, working in both department, requests access to Anna's EHR.
Expected behaviour: Bob's request should be denied, since the group of general practitioners is not more specific than the group of emergency staff, and vice-versa.
Requirements: Req. 8.

Scenario 6 - Condition

Sam wants to specify rules that are valid in certain contexts: he want to restrict access to his EHR when he is hospitalised; when he is not hospitalised, Sam wants to deny access to his EHR to all hospital staff.
Requirements: Req. 9.

III. SGAC Data Structures, Rules and Requests

This section presents our model SGAC and the different data structures needed to specify rules and requests. Conflict resolution is then illustrated by different examples. Notations are first introduced.

A. Notation

For the rest of the paper, we introduce the following notations drawn, in most cases, from the B notation [2].

1) Set Theory: Let A, B, C be sets.

- For a n-tuple $a = (a_1, \ldots, a_n)$ we denote by $a.a_k$ the component of a named a_k.
- $\mathcal{P}(A) = \{X \mid X \subseteq A\}$, called the power set of A, is the set of all subsets of A.
- $A \times B = \{x \mapsto y \mid x \in A \wedge y \in B\}$ is the Cartesian product; it is a set of ordered pairs $x \mapsto y$.
- A relation R from A to B is a subset of $A \times B$.
- $\mathsf{id}(A) = \{x \mapsto x \mid x \in A\}$ denotes the identity relation on A, i.e. the relation that associates each element of A to itself.
- $A \leftrightarrow B = \mathcal{P}(A \times B)$ denotes the set of relations between A and B.

- $\mathsf{dom}(R) = \{x \in A | \exists y \in B \bullet x \mapsto y \in R\}$ denotes the domain of R.
- $R[C] = \{y \mid y \in B \wedge \exists x \in C \bullet x \mapsto y \in R)\}$ denotes the image set of C by relation $R \in A \leftrightarrow B$.
- $A \nrightarrow B$ denotes the set of (partial) functions from A to B. A partial function f from A to B is a relation such that $|f[\{x\}]| \leq 1$ for $x \in A$.
- $A \rightarrow B$ denotes the set of total functions from A to B. A total function f is a partial function such that $\mathsf{dom}(f) = A$.
- $R_1 ⨾ R_2 = \{x \mapsto z \mid \exists y \in B \bullet x \mapsto y \in R_1 \wedge y \mapsto z \in R_2\}$ is the relational compositon of $R_1 \in A \leftrightarrow B$ and $R_2 \in B \leftrightarrow C$.
- Let $R \in A \leftrightarrow A$. R^n denotes the composition of R with itself n times ($n \geq 0$), with $R^{n+1} = R ⨾ R^n$ and $R^0 = \mathsf{id}(A)$.
- $R^+ = \bigcup_{n \geq 1} R^n$ denotes the transitive closure of R, *i.e.*, the smallest transitive relation which contains R.
- Let $R \in A \leftrightarrow A$. $R^* = R^+ \cup \mathsf{id}(A)$ denotes the transitive and reflexive closure of R, *i.e.*, the smallest transitive and reflexive relation which contains R.

2) Graph: A *directed graph* is an ordered pair $G = (V, E)$ where V is the set of *vertices* V and E is the set of *edges*, such that $E \in V \leftrightarrow V$. G is said *acyclic* iff $G.E^+ \cap \mathsf{id}(G.V) = \emptyset$. In an edge $x \mapsto y$, y is called a *successor* of x and x a *predecessor* of y. In an edge $x \mapsto y$ of $G.E^+$, *i.e.*, the transitive closure of $G.E$, x is an ancestor of y and y is a descendant of x. A vertex without any successor is called a *sink* and sinks reachable from a vertex v in a graph G are denoted by $sink(G, v) = G.E^*[\{v\}] - \mathsf{dom}(G.E)$. All the sinks of a graph G are denoted by $sink(G) = G.V - \mathsf{dom}(G.E)$.

B. Using graphs

In SGAC, two directed acyclic graphs are needed in order to specify rules and requests:

- the subject graph represents the hierarchy which mirrors the functional organisation chart or any grouping of users relevant for access control;
- the resource type graph represents the taxonomy of EHR and their organisation in the healthcare facilities.

Fig. 1 illustrates a subject graph. The graph includes people and subjects as vertices. A subject represents a person or a set of people. The hierarchy works as follows: a rule on subject s is inherited by all the successors of s in $G.E^+$. For instance in Fig. 1, if a permission is given to the *General Practice* department then this permission is inherited by *GP Physicist* and *GP Nurse*, *Bob* and *Alice*.

Fig. 2a illustrates the resource type graph. We distinguish between *resources types* and *documents*. Medical records are structured into a taxonomy which is represented by a graph of resource types. A document is an actual medical record of a patient. Documents are instances of sinks of the resource type graph. A document has attributes which can be given as parameters to non-sink vertices. For instance, a certain AIDS screening test can have many attributes such has: the patient it is related to, the visit when it was ordered, the ID of the screening test etc... There is a functional dependency between the document type identifier and the other attributes, making the key *document identifier* sufficient to retrieve a document, and all its attributes.

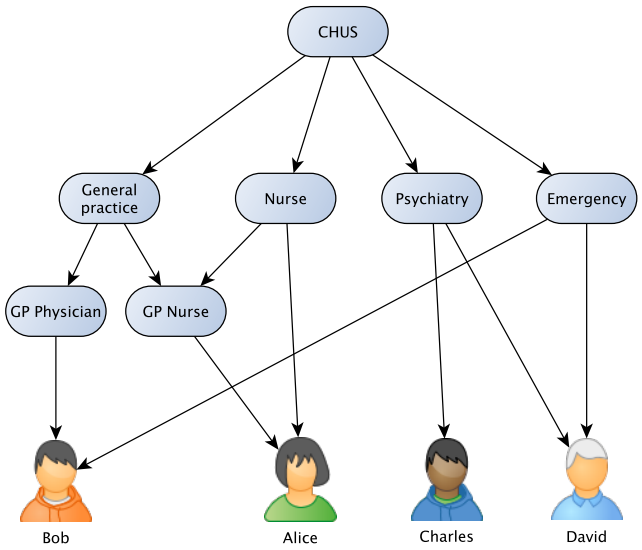


Fig. 1: Subject graph example

The resource type graph sinks are *document type*, and the non-sink vertices represents aggregations of these *document types*. For instance, the vertex *patient* represents all the data types of all patients and can be instantiated with a parameter to target the data of a particular patient.

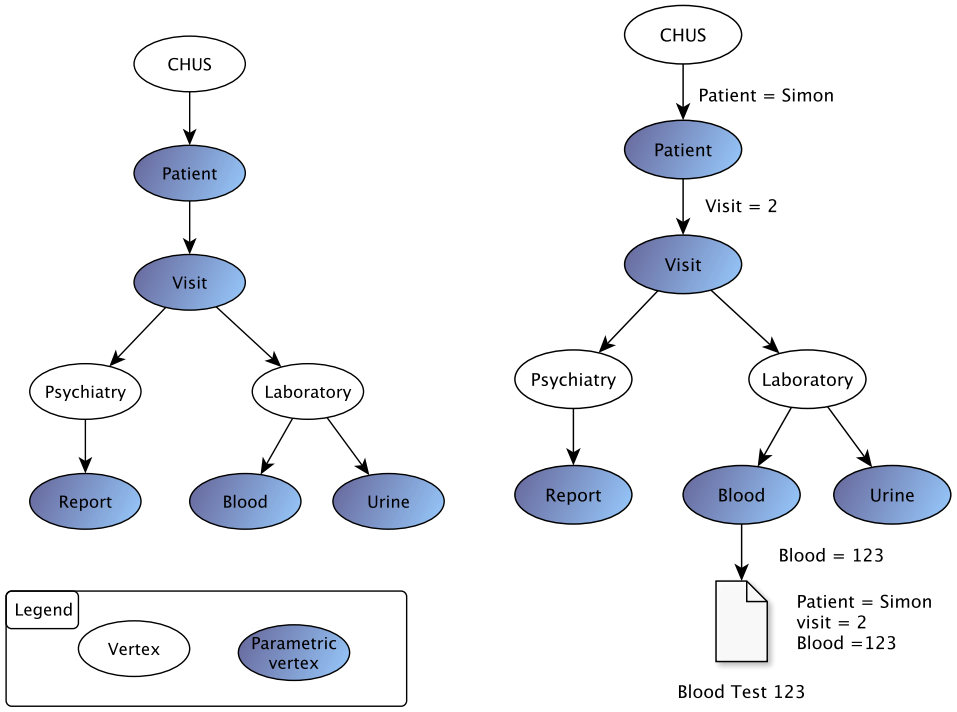
Fig. 2b illustrates the resource type graph being instantiated for the document *Blood 123* of the patient *Simon* during his visit no. *2*.

These two graphs define the basis on which rules and requests are built.

C. Rule and request specification

A rule allows to specify a control over the access to a resource. It is defined by:

- a *subject*: a person or a group of people to control;
- a *resource*: the data to be protected;
- an *action*: the operation the *subject* wants to do on the *resource*;
- a *priority*: a number which defines the priority of the rule;
- a *modality*: an authorisation or a prohibition which defines the effect of the rule;
- a *condition*: a formula which determines the applicability of the rule. It can be evaluated at run time by functions checking for instance information stored in



(a) Resource type graph (b) Instantiated resource type graph

Fig. 2: Resource graph example

a database. For the rest of the paper, we describe rule conditions in natural language.

A request is the demand the *subject* issues in order to execute an *action* on a *resource*. It has then the following attributes:

- a *subject*: the request initiator;
- a *document*: a document the request initiator wants access to;
- an *action*: the operation the *subject* wants to do on the *document*.

D. Conflict resolution

When more than one rule apply to a request, and if they have different modalities, a situation, typically called a *conflict* in the literature, arises. To decide whether access is granted or denied, we define an ordering (a precedence) on rules. The rule with the "highest" precedence determines the access decision. Let r_1, r_2 be two applicable rules for a request.

1) If r_1 has a smaller priority than r_2, we say that r_1 has precedence over r_2.
2) If r_1 and r_2 have the same priority, and if the subject of r_1 is more specific than the subject of r_2 (*i.e.*, the subject of r_1 is a descendant of the subject of r_2 in the subject graph), then r_1 has precedence over r_2.
3) If r_1 and r_2 have the same priority, and neither of their subjects is more specific than the other, then prohibitions have precedence over permissions.

This ordering is not total. There may be two rules r_1, r_2 such that neither of them precedes the other. However, in such a case, r_1 and r_2 have the same modality, thus there is no conflict and the decision is the modality of these elements with highest precedence. The formal definition of this ordering in Section V shall clarify the third clause in some subtle cases, to avoid any ambiguity in its interpretation.

This conflict resolution method is absolutely autonomous and does not require the intervention of an external actor. Section IV illustrates the conflict resolution technique with three examples.

IV. Examples

This section illustrates the behaviour of SGAC with three examples. For the sake of simplicity, we illustrate *read* requests. The same approach applies for any other action.

A. Example 1: basics

Let's model scenario 1. The resource type graph must be instantiated with the parameters defining Anna's data. Modelling Anna's rule consists in prohibiting access to the documents descending from the vertex *Psychiatry* in the resource graph. The vertex *Patient* gets the unique identifier of the patient Anna. The vertex *CHUS* in the subject graph (Fig. 1) represents all the personnel from the hospital. By convention, patient rules are of priority

Rule	Resource	Subject	Pri.	Mod.	Cond.
r_1	Psychiatry (patient = Anna)	CHUS	2	−	*TRUE*

TABLE I: Scenario 1 rule (for the action *read*)

2. When no condition is specified, the rule condition is set to *TRUE*. A prohibition is represented by symbol "−", whereas a permission is represented by symbol "+". The rule is presented in Table I.

If Bob requests an access to Anna's psychiatric report no.20, then SGAC will first determine the applicable rules. Rule r_1 is applicable because: $r_1.subject$ is an ancestor of the request subject, $r_1.resource$ is an ancestor of the requested resource, the action matches, the condition is verified, and the parameter fits. If this is the only rule applicable, then the system returns *prohibition*. We only described the rule issued by Anna's consent for the sake of simplicity in this example. In the case where no rules from laws and regulations are applicable, if Anna's rule is among the other rules applicable to a request, then this request is denied.

B. Example 2: let's get started

In this example, the rule base is as follow:

- the laws and regulations allow emergency physicians to access (read and write) the data of any patient who is in a life-threatening situation;
- the hospital allows general physicians to read and write data for any patient under their care;
- the hospital allows nurses to read vitals of a patient at any time.

Rule	Resource	Subject	Pri.	Mod.	Cond.
r_1	Patient	Emergency	1	+	patient life is threatened
r_2	Patient	GP Physician	3	+	the subject is the attending physician
r_3	Vitals	Nurses	3	+	*TRUE*

TABLE II: Example 2 rules (for the action *read*)

This can be represented by the rule base presented in Table II. By convention for these examples, the priority of a rule is determined by the entity issuing the rule: if the rule is from a healthcare facility, then it is set to 3, if it is from the patient, then priority is set to 2, and if the rule is from laws and regulations, priority is set to 1. The lower the value a rule priority has, the higher precedence the rule gets. This reflects the wanted behaviour: laws and regulation have precedence over patient rules, which have precedence over healthcare facility rules.

Rule r_1 translates the fact that any physician in the *Emergency* department can access a record if its owner's life is threatened: an authorisation given to the vertex *Emergency* to read all documents from *Patient*, under the specified condition. The priority is set to 1 since the rule stems from the laws and regulations.

The rule r_2 translates the fact that a physician is allowed to read the data of the patients under his/her care, i.e. the physician has to be the patient's attending physician: an authorisation given to the vertex *GP Physician* to read all documents from *Patient*, under the condition that the physician is the attending physician of the patient. The priority of this rule is set to 3 since the rule stems from the hospital.

Finally, the rule r_3 translates the fact that a nurse is allowed to read the vitals of any patient, at any time. Since, the nurse can access the *Vitals* of any *Patient* in any condition, the condition of r_3 is set to *TRUE*. The priority is also set to 3 since the rule stems from the hospital too.

In order to have a better understanding of the rules, the subject graph, the resource type graph and the rules are presented in the same picture in Fig. 3.

Now let's say that patient Anna is treated for some light mental disorder by Charles, a psychiatrist. Since Charles is Anna's attending physician, he can access her records while others can't except Alice who can read Anna's vitals. The access rights are summed up in Table III.

Staff	Pulse	Blood Pressure	Report	Blood	Urine
Alice	√	√	×	×	×
Bob	×	×	×	×	×
Charles	√	√	√	√	√
David	×	×	×	×	×

TABLE III: Example 2: Access of the CHUS personnel to Anna's Record, wrt Fig. 3

Then comes Sam, badly hurt, unconscious in the *Emergency* department. Since, Bob and David are working in the Emergency department and that Sam's life is threatened, both have access to his records. Alice still can read Sam's vitals while Charles does not have any access to Sam's data. The resulting accesses are presented in Table IV.

Staff	Pulse	Blood Pressure	Report	Blood	Urine
Alice	√	√	×	×	×
Bob	√	√	√	√	√
Charles	×	×	×	×	×
David	√	√	√	√	√

TABLE IV: Example 2: Access of the CHUS personnel to Sam's Record, wrt Fig. 3

Finally, in the case of a patient who has no attending physician, and whose life is not threatened, the only person who can access this patient's records is Alice, who is allowed to read the vitals.

C. Example 3: adding consent

In this example, we take the same initial rule base (Table II), and we add some consent rules. Let's say Anna personally knows Bob and does not want him to access her records (rule r_4). This rule targets directly Bob and Anna's data, and is applicable at any time. Since r_4 is directly issued by a patient, its priority is set to 2. At this

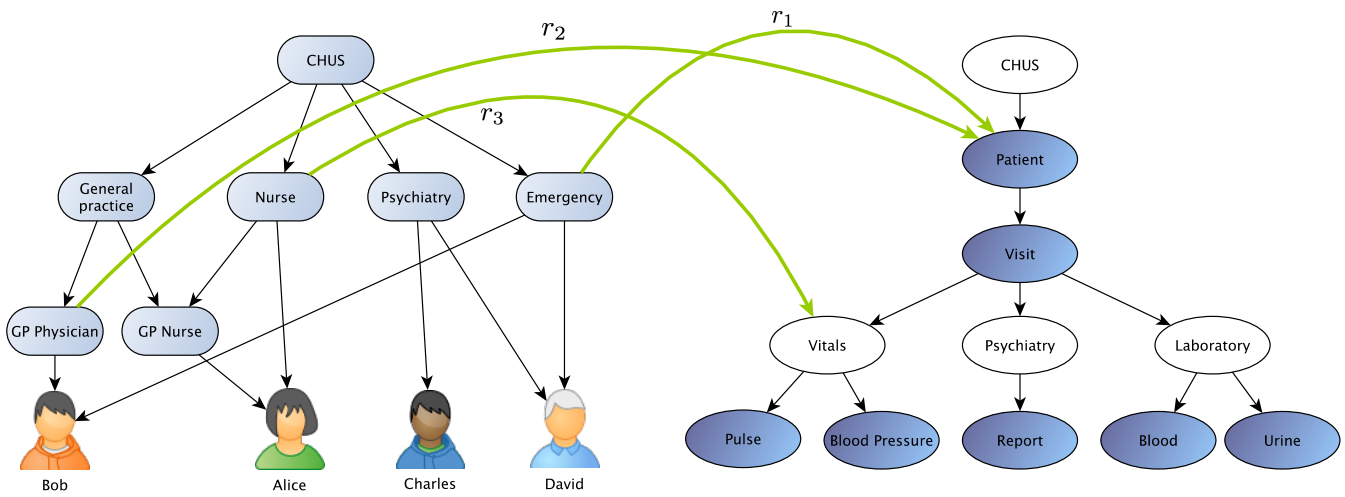


Fig. 3: Example 2 graphs with rules

Staff	Pulse	Blood Pressure	Report	Blood	Urine
Alice	✓	✓	×	×	×
Bob	×	×	×	×	×
Charles	×	×	×	×	×
David	✓	✓	×	×	×

TABLE V: Example 3: Access of the CHUS personnel to Anna's Record

Rule	Resource	Subject	Pri.	Mod.	Cond.
r_1	Patient	Emergency	1	+	patient life is threatened
r_2	Patient	GP Physician	3	+	the subject is the attending physician
r_3	Vitals	Nurses	3	+	*TRUE*
r_4	Patient = Anna	Bob	2	−	*TRUE*
r_5	Vitals	Emergency	2	+	*TRUE*
r_6	Vitals	Bob	2	+	*TRUE*

TABLE VI: Example 3 rules (for the action *read*)

point, even if Anna is under Bob's care, Bob won't have access to Anna's records because of r_4, unless there is an emergency context where Anna's life is threatened. In that case, r_1 would allow him to access the data.

Then, Anna is hospitalised and gets on with the staff of the Emergency department. When she has to undergo rehabilitation, she decides to allow the whole Emergency department to access her vitals data in order to let her new friends follow her progress (rule r_5). In this situation, there is a conflict between r_4 and r_5 when Bob wants to access Anna's vitals. Bob still can't access any data of Anna, since r_4 is considered to have precedence over r_5 since the target of r_4 is more specific than the target of r_5, but David who is also affected by r_5 can access Anna's vitals. The accesses at this point are presented in Table V.

Finally, Anna decides to share her vitals to Bob and she adds a new rule, r_6, to do so, but forgets to remove r_4. These two rules contradict each other: they have the same priority, and one is not more specific than the other. In that case, a prohibition has precedence over a permission. Bob's access is unchanged: he can't access Anna's data, unless there in an emergency context where Anna's life is threatened. The final rule base of this example is presented in Table VI and with the graphs in Fig. 4.

V. Formal model

In this section, our formalisation of SGAC is presented. This formalisation provides a way to evaluate requests for a given set of rules, and a way to verify properties.

A. Subject graph

The subject graph is denoted by S. We denote by $SUBJECT$ the set of all subjects and by $PERSON$ the set of all persons. Formally, S is described by $S = (V, E, Z)$ with:

- (V, E) is a DAG;
- $V \subseteq SUBJECT$ is the set of the subjects;
- $Z = V \cap PERSON$ represents the persons, and elements of $V - Z$ are entities which represent groups of persons;
- $Z \subseteq sink(S)$ since a person is a sink of S.

$S.E$ represents the inheritance relation: recall that a rule on a subject s is inherited by all the successors of s in $S.E^+$ There are two types of subject: persons and entities. A sink of S can be either a person or an entity, but a person is by definition a sink. A non-sink vertex is then an entity.

B. Resource Graph

As mentioned in Section III-B, data have been abstracted by types into a resource type graph. Recall that

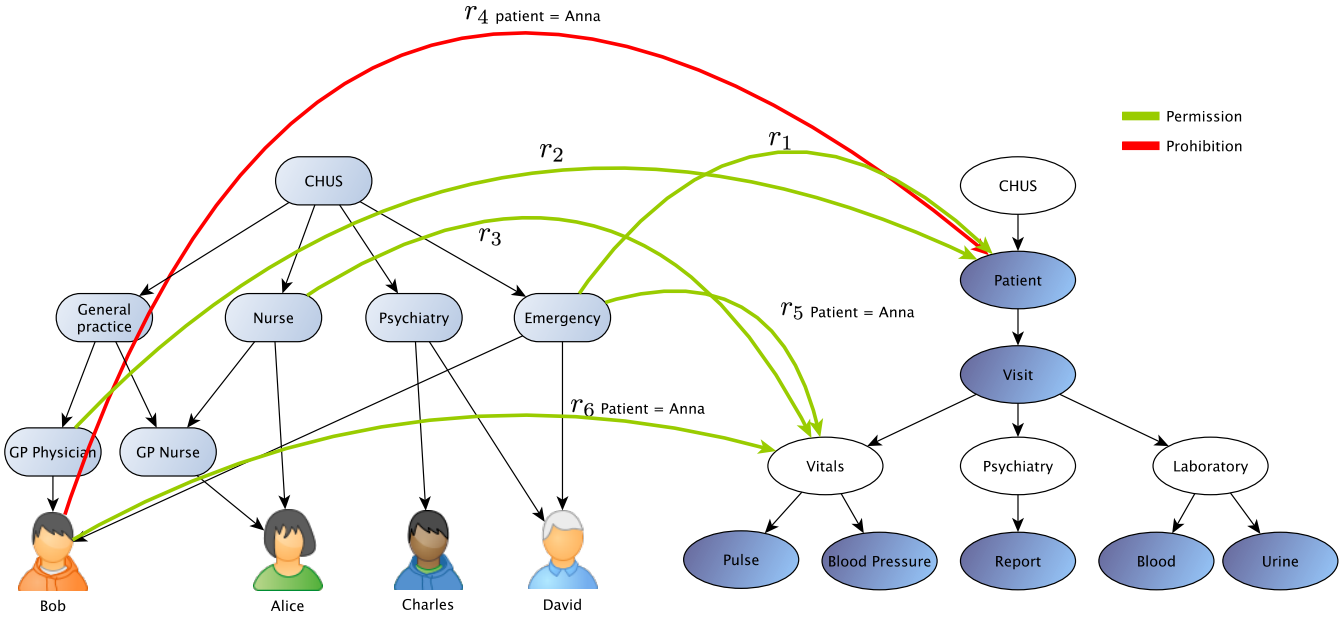


Fig. 4: Example 3 graphs with rules

an atomic element of data is called a *document* and can be for instance a prescription, a radiography, etc... We introduce the notion of parametric directed acyclic graph (PDAG) as follows: $R = (G, K)$ where G is a DAG and $H = (T, D, U, W)$ denotes constraints linking the DAG G to the documents. More precisely, $G = (V, E, P)$ where V is the set of the vertices, E the edges and P the set of the parametric vertices. We denote by $DOCUMENT$ the set of all documents.

- $G.P$ denotes parametric vertices that are called *parametric groups* and the elements of $G.V - G.P$ are called *groups*. Parametric groups introduce exactly one parameter, like *patient*, *visit* etc...

$$G.P \subseteq G.V$$

- Sinks of R are document types, so they are parametric groups since a type of document requires an identifier.

$$sink(R) = sink(G.V, G.E) \subseteq G.P;$$

- D denotes the set of all the documents, and U the type of a document. Each document has exactly one type, so

$$U \in D \to sink(R);$$

$$D \subseteq DOCUMENT$$

- W denotes a valuation of parameters of the documents. The parameter valuation W is defined for each document and associates a document with a (partial) function between *parametric groups* and parameter values. It is a partial function since a document does not use all the parameters of the graph, but only those of its ancestors. Since each document has unique attributes, valuation of parameters is defined for all documents, thus we have

$$W \in D \to (P \nrightarrow T);$$

- W is defined for each parameter inherited by a document,

$$\forall d \in D \bullet \mathsf{dom}(W(d)) = G.E^{-1*}[U(d)] \cap G.P$$

C. Rule

A rule l is a septuplet which contains:

- a modality *mod*;
- a resource *res* with the valuation *val* of its parameters;
- an action *act*;
- a subject *sub*;
- a priority *pri*;
- a condition *con*.

We denote by $ACTION$ and $RULE$ the sets of all actions, and of all rules. Since a rule depends on a subject graph and a resource type graph, we introduce the object *Policy*, composed of a subject graph, a resource type graph, and a set of rules. Each rule of the policy targets elements of the graphs of the policy. Formally, we denote by $P = (S, R, L)$ a policy and we have:

- $S = (V, E, P)$ a subject DAG;
- $R = (G, H)$ a resource type PDAG;
- $L \subseteq RULE$ the set of rules of the policy.

We have to link the rules of a policy to the graphs by the following constraints:

- the subject is a person of S:

$$\forall l \in L \bullet l.sub \in sink(S)$$

- the action belongs to $ACTION$ and the priority is a positive real:

$$\forall l \in L \bullet l.act \in ACTION$$

$$\forall l \in L \bullet l.pri \in \mathbb{R}^+$$

- $l.res$ is a vertex of $R.G.V$ and $l.val$ is a valuation of parametric groups of $R.G.P$ with adequate values:

$$\forall l \in L \bullet l.res \in R.G.V \wedge l.val \subseteq R.G.P \times R.H.T$$

- the only possible modalities are permission, and prohibition;

$$\forall l \in L \bullet l.mod \in \{permission, prohibition\}$$

We also introduce the function *documents*:

$$documents(R, v, K) = \{d | d \in R.H.D \wedge R.H.U(d) \in sink(R, v) \wedge K \subseteq R.H.W(d)\}.$$

The function $documents(R, v, K)$ returns all documents reachable from vertex v in PDAG R with document parameter valuation K.

For example, $documents(R, Visit, \{patient \mapsto Simon\})$ denotes the set of all documents issued during any visit of patient $Simon$. The blood test of the example of Fig. 2b denoted by bt has the following associated parameters:

$$\begin{array}{ll} & R.H.W(bt) \\ = & \{patient \mapsto Simon, visit \mapsto 2, id \mapsto 123\} \\ \supseteq & \{patient \mapsto Simon\} \\ = & K, \end{array}$$

thus $bt \in documents(R.H, Visit, \{patient \mapsto Simon\})$.

D. Request

We define a request q as a triplet (sub, act, doc) where:

- $sub \in PERSON$ is the person initiator of the request;
- $act \in ACTION$ is the action sub wants to do;
- $doc \in DOCUMENT$ is the document targeted by the action act.

Do note that a request is made by one person and only targets one document at a time.

E. Request evaluation

The approach to evaluate a request is the following:

- extract all rules applicable to the request;
- sort extracted rules and represent them by a rule DAG;
- evaluate the request from the rule DAG.

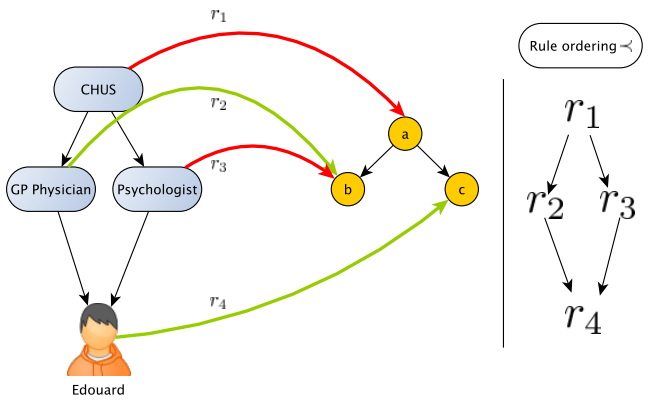


Fig. 5: Illustration of the partial order relation $\prec$

1) Rule extraction: to this end, we introduce the function $Rules(P, q)$ for a policy P and a request q:

$$Rules(P, q) = \{l \mid l \in P.L \wedge sub \wedge act\_con \wedge doc\}$$

where:

- $sub := q.sub \in (P.S).E^*[\{l.sub\}]$;
- $act\_con := (l.act = q.act) \wedge eval_f(l.con)$;
- $doc :=$

$$\begin{array}{ll} & (P.R.U)(q.doc) \in P.S.E^*[l.res] \\ \wedge & l.val \subseteq P.R.W(q.doc) \end{array}$$

- the function $eval_f(f)$ evaluates the formula f, taking into account values of variables occurring in f.

Then for a policy P and a request q, $Rules(P, q)$ designates all rules of $P.L$ of which:

- action corresponds to $q.act$;
- subject is $q.sub$ or an ancestor of $q.sub$;
- condition is evaluated to $TRUE$;
- reachable documents contains $q.doc$.

2) Rule ordering: once we have all applicable rules, we need to sort them. We therefore introduce a partial order relation $\prec$ defined as follows :

$$\begin{array}{l} \forall x, y \in RULE \bullet \\ \quad x \prec y \\ \Leftrightarrow \\ \qquad y.pri < x.pri \\ \quad \vee \quad (x.pri = y.pri \wedge y.sub \in S.E^+[\{x.sub\}]) \end{array}$$

This relation $\prec$ captures the fact that precedence is given to the rule with a lower priority or, at equal priority, to the rule targeting the lower subject (inclusion-wise). This order does not take into account the resources targeted by a rule. For instance, in Fig. 5, we suppose that r_1, r_2, r_3 and r_4 share the same priority. We have then $r_1 \prec r_2$, $r_1 \prec r_3$, $r_2 \prec r_4$, $r_3 \prec r_4$ and finally $r_1 \prec r_4$. Note that r_2 and r_3 can't be compared with $\prec$.

If r_1, r_2, r_3 and r_4 are the only applicable rules then precedence over the other rule would be given to r_4 since it is the only maximal element (there is no other rule r' such

that $r_4 \prec r'$). But what happens when there are more than one maximal element? Let's take the previous example, and remove r_4. We have $r_1 \prec r_2$ and $r_1 \prec r_3$, but r_2 and r_3 still can't be compared. We then define another partial order on rules, noted "$<$". The set of maximal elements of the set $Rules(P,q)$ with the relation $\prec$ is denoted by $max_{\prec}$. Formally,

$$max_{\prec} = \{x \in Rules(P,q) \mid \nexists y \in Rules(P,q) \bullet x \prec y\}$$

We define $<$ on rules as follows:

$$\begin{array}{l} \forall x, y \in Rg.V \bullet \\ \quad x < y \\ \Leftrightarrow \\ \qquad x \prec y \\ \quad \vee \quad (\ x, y \in max_{\prec} \\ \qquad \wedge\, y.mod = prohibition \\ \qquad \wedge\, x.mod \neq y.mod \\ \qquad) \end{array}$$

The partial order $<$ extends $\prec$: in the case there are more than one maximal element, precedence over the permissions are given to the prohibitions. Thus ordered rules can be represented in the DAG $Rg_{P,q}$ which is calculated from a request q in the policy P. $Rg_{P,q}$ is defined by:

- $Rg_{P,q}.V = Rules(P,q)$;
- $Rg_{P,q}.E$ is the covering relation of $<$, which is in our case equal to the transitive reduction of $<$, in order to find the immediate successor precedence-wise.

3) Rule graph analysis: The rule graph $Rg_{P,q}$ contains all rules applicable to a request q in a policy P ordered by precedence. Thus the rules from $sink(Rg_{P,q})$ have precedence over the other. We denote by the function $eval(P,q)$ the evaluation of the request q in the policy P; $eval(P,q)$ returns *TRUE* if q is approved. In order to determine *eval*:

1) we determine first all applicable rules by calculating $Rules(P,q)$;
2) we create the DAG $Rg_{P,q}$;
3) we verify the following property:

$$\begin{array}{l} Prop(P,q) := sink(Rg_{P,q}) \neq \emptyset \\ \qquad\qquad \wedge\, \forall l \in sink(Rg_{P,q}) \bullet l.mod = permission. \end{array}$$

We define *eval(P,q)* as follows:

$$eval(P,q) := eval_f(Prop(P,q))$$

If all sinks of Rg are permissions, then a permission is returned and *eval* returns *TRUE*. If $Rg.V$ is empty (*i.e.*, no rules are applicable), a *prohibition* is returned and *eval* returns *FALSE*.

As noted before, $<$ ensures that all sinks of Rg have the same modality. To see this, let $r_1, r_2 \in sink(Rg)$ with $r_1.mod \neq r_2.mod$. There are two cases:

- $r_1.pri = r_2.pri$: according to the definition of $<$, we have $r_1 < r_2$ or $r_2 < r_1$, which is absurd since $r_1, r_2 \in sink(Rg)$.
- $r_1.pri \neq r_2.pri$ then we have $r_1 < r_2$ or $r_2 < r_1$, which is absurd.

Thus we have the following properties for *eval*:

$$\begin{array}{l} eval(P,q) \Leftrightarrow \\ \quad \wedge \quad \exists l \in sink(Rg_{P,q}) \bullet l.mod = permission \end{array}$$

and

$$\begin{array}{l} eval(P,q) \Leftrightarrow \\ \qquad Rules(P,q) \neq \emptyset \\ \quad \wedge \quad \nexists x \in max_{\prec} \bullet \\ \qquad\qquad x.mod = prohibition \end{array}$$

The first says that if a sink of $Rg_{P,q}$ is a permission, then access is granted. The second says that if there is at least one applicable rule and if there is no prohibition in the maximal elements of $Rules(P,q)$ wrt $\prec$, then access is granted.

F. Example

Let's say that the patient Anna is to be hospitalised in the CHUS. She did work there when she was a nurse and had befriended most of her former colleagues, but also had some rivals like Alice. Anna decided to share her laboratory data to her nurse friends except for Alice and general practice physicians. She is aware that in the emergency department, physicians can access the all records of the patient, while that patient is under their care. Since she knows personally some of these physicians, she decides to prevent the department from accessing her records. But in the case her life is threatened, regulations and laws permit emergency physicians to access her records in order to provide faster and better medical care. We denote by P_1 the policy containing all the previous rules, the subjects and resources. We have the following rules presented in the Tab. VII as $P_1.L$. We use Fig. 1 as the subject graph $P_1.S$ and Fig. 2a as the resource graph $P_1.R$

We suppose that Anna's EHR only contain two blood tests bt_1, bt_2 and a psychiatry report pr_1. bt_1 has been issued during the first visit, and bt_2 and pr_1 during the second visit. For this example, $P_1.R$ only contains Anna's documents. Formally, we introduce the documents in $P_1.R.H$, which we simply denote by H in the sequel:

- $H.D = \{bt_1, bt_2, pr_1\}$;
- $H.U = \{bt_1 \mapsto Blood, bt_2 \mapsto Blood, pr_1 \mapsto Report\}$;
- $H.W =$
  $\{bt_1 \mapsto \{Patient \mapsto Anna, Visit \mapsto 1, Blood \mapsto 1\},$
  $bt_2 \mapsto \{Patient \mapsto Anna, Visit \mapsto 2, Blood \mapsto 2\},$
  $pt_1 \mapsto \{Patient \mapsto Anna, Visit \mapsto 2, Report \mapsto 1\}\}$.

We then have:

$$documents(P_1.R, Patient, Patient \mapsto Anna) = \{bt_1, bt_2, pt_1\}.$$

Let's assume that Alice wants to access bt_1. Let's denote by q_1 the request $(Alice, read, bt_1)$.

$$Rules(P_1, q_1) = \{r_1, r_2\}$$

Rule	Resource	Subject	Pri.	Mod.	Cond.
r_1	Laboratory (Patient = Anna)	Nurse	2	+	*TRUE*
r_2	Laboratory (Patient = Anna)	Alice	2	−	*TRUE*
r_3	Laboratory (Patient = Anna)	GP Physician	2	+	*TRUE*
r_4	Patient	Emergency	3	+	*the subject is the attending physician of the data owner*
r_5	Patient = Anna	Emergency	2	−	*TRUE*
r_6	Patient	Emergency	1	+	*data owner's life is threatened*

TABLE VII: Rule Base Example, for the action *read*

We then calculate $Rg_{P_1,q_1}.E = \{r_1 \mapsto r_2\}$ since Alice belongs to the subject *Nurse*. We have $sink(Rg_{P_1,q_1}) = \{r_2\}$. Thus: $eval(P_1, q_1) = false$. Moreover, in all possible contexts, Alice can't access Anna's data, since $RulesRg_{P_1,q_1}$ will not vary with contexts.

Now Bob wants to access bt_2, $q_2 = (Bob, read, bt_2)$. We suppose that Anna is fine, and that Bob is her attending physician.

$$Rules(P_1, q_2) = \{r_3, r_4, r_5\}$$

Since, Bob belongs to *GP Physician* and *Emergency*, he is affected by any rules targeting one of the two entities. The calculus of $Rg_{P_1,q_2}.E$ is a bit trickier than before: we have $r_4 < r_3$ and $r_4 < r_5$ because $r_4 \prec r_3$ and $r_4 \prec r_5$ but r_3 and r_5 are incomparable with $\prec$. In fact, $r_3 < r_5$ since they are both maximal elements and r_5 is a prohibition and r_3 is a permission. Then we take the transitive reduction of $<$, thus $Rg_{P_1,q_2}.E = \{r_4 \mapsto r_3, r_3 \mapsto r_5\}$. We have then $sink(Rg) = \{r_5\}$. Bob's request is thus denied.

But in an emergency context, where Anna's life would be threatened, Bob would have access to this data, more precisely, to all Anna's data, since $sink(Rg_{P_1,q_2}) = \{r_6\}$ in this context for any data requested by Bob.

G. Potential danger detection

We are working on the formalisation of SGAC in B [2] and in Alloy [3]. This allows for the detection of potential dangerous situations, for instance when the patient hides important data from the medical staff. In that case, the following property must hold for the patient p within the policy P:

$$\forall d \in documents(P.R, Patient, \{Patient \mapsto p\}), \\ \exists i \in (P.S).Z \bullet eval(P, (i, read, d))$$

Model checking this property allows for counter-example exhibition, thus identify a patient who has concealed all his/her data, and warn him/her about a potential danger. This verification can be done for all patient:

$$\forall p \in \mathsf{ran}(P.R.H.W)[Patient], \\ \forall d \in documents(P.R, Patient, \{Patient \mapsto p\}), \\ \exists i \in (P.S).Z \bullet eval(P, (i, read, d))$$

This property can be simplified into:

$$\forall d \in documents(P.R, Patient, \{\}), \\ \exists i \in (P.S).Z \bullet eval(P, (i, read, d))$$

Finding a patient who has all of his data hidden is the same as finding a document which is completely hidden. Moreover, our formalisation of SGAC allows for access verification.

- *Determination of necessary conditions for a subject to access a resource*: it is possible to determine a formula which must hold in order to authorise a request.
- *Redundant rule detection*; a rule is said redundant within a policy if the requests accepted by the policy is the same with and without the rule.
- *Determination of the data accessible by a subject*: since we can determine the result of a request, we can determine all accessible documents for a given subject.

VI. Related Work

RBAC (*Role Based Access control*) [4], [5], is a classic access control model which uses the notions of user, role, operation, object and session. In order to gain privileges, which are represented by a pair (operation, object), the user must have activated one of his roles in a session that has the privileges needed. There are two additional features: role hierarchy allows for privilege inheritance among roles, and separation of duty constraints prevent a user from activating/being assigned to specified combinations of roles. Formalisation of RBAC has been done in Z [6] and in B [7]. Verified properties on those formalisation are:

- role activation: a role can be activated only if it is assigned to the user;
- role hierarchy: a role properly passes assigned privileges to its children, and the role hierarchy is acyclic;
- separation of duty all constraints of separation of duty hold.

RBAC allows privilege grouping, thanks to roles and role inheritance, but it does not support prohibition, conditions, priority, and resource inheritance. This makes the management of complex fine-grain policies quite difficult. Thus, RBAC does not satisfy the requirements of SGAC.

OrBAC [8] (*Organisation-Based Access Control*), is a logic-based access control model which takes into account RBAC weaknesses and fixes some of them. It reuses the notions of role, user, action, object, and adds some new concepts: i) activity, an abstraction of actions, ii) view, an abstraction of objects, iii) contexts, which allow for the expression of complex rule conditions, iv) prohibition, v) priority in order to manage conflicts, and vi) organisation.

The concept of organisation is used to parameterise assignment of roles to users, of views to objects, and of subjects to roles. It supports two kinds of rules: organisational rules that use abstract notions, and concrete rules that use concrete notions. Conflicts are detectable by static checking with the Prolog-based tool MotOrBAC [9]. If two organisational rules with different modalities are applicable to the same abstract concepts, then a potential conflict is detected. This conflict is only potential since there may not exist a common concrete entity (subject, action or object) for which the two organisational rules apply. The user can solve a potential conflict by modifying the priority or the rules, by adding separation constraints, or by just ignoring the conflict when the user knows that there is no concrete entity for which the two organisational rules simultaneously apply. Inheritance among roles or views can be specified by using logic rules. OrBAC is powerful enough to satisfy the SGAC requirements, but its logic-based approach may suffer from performance problem for a very large number of rules, since its execution engine is based on a prolog-like language. Conflict management also requires manual intervention, whereas SGAC uses an ordering that forbids conflicts.

Ponder [10] has a domain hierarchy which contains resources and subjects in the same graph. A rule in Ponder has a subject, a resource, an action, a modality and a condition. It can also be marked as *final* to have precedence over another rule not marked as such. In case both are/are not marked *final*, if their subjects are comparable, then precedence is given to the rule with the more specific subject, and if their subjects are the same, then precedence is given to the rule with the more specific resource. Finally, if their subject are not comparable, rules marked as *final* become normal and if there still is a modal conflict then Ponder returns a *prohibition*. Ponder does not include a rule priority attribute, and it uses a single graph to represent both subjects and resources, which cannot be used in our case where there is a huge number of resources and subjects. Moreover, its conflict management is not adapted to the SGAC requirements.

XACML [1] (*eXtensible Access Control Markup Language*) is an attribute-based access control language. A rule has a target defined by a subject, an action, a resource, a condition, an effect which can be either *permit* or *deny*. There is no native inheritance among subjects or resources. Tree-like inheritance can be simulated by using paths for resources and subjects identifiers. Precedence among rules is managed by using a rule combination algorithm. The basic rule combination algorithms are:

- permit-overrides: it returns permit if at least one applicable rule returns permit;
- deny-overrides: it returns deny if at least one applicable rule returns deny;
- first applicable: it returns the effect of the first applicable rule.

XACML satisfies most of the SGAC requirements, but its weak support of inheritance and its management of conflicts make it difficult to manage large security policies. It also suffers from poor performance when a large number of rules are used. Brians [11] formalises XACML with CSP in order to simulate policies. Using CSP has some drawbacks: conditions are not handled, properties can not be always specified in CSP and our own combining algorithms can't be added easily.

Table VIII summarises the difference between the different models for which a formal model exists.

VII. Performance comparison

Since XACML is an industrial standard and that it is very close to satisfying SGAC requirements, we tried to simulate SGAC policies in XACML using paths and the rule combining algorithm *first-applicable*. The other rule combination algorithms do not fit the SGAC requirements.

To simulate SGAC policies in XACML, we proceed as follows. We define three policies, one for each level of priority (law, patient and hospital). We use first-applicable as the policy combination algorithm. Within a policy, we order rules according to the subject hierarchy and modality, enumerating the subject graph in a post-order fashion (*i.e.*. bottom-up). We use first-applicable as the rule combination algorithm of a policy. SGAC rule subjects are translated as a regular expression of the form "$*s*$". A request $q = (s_i, a, r)$ is rewritten using the XACML context handler as $q = (s_1/\ldots/s_i, a, r)$, where $s_1/\ldots/s_i$ is the path from the root of the subject graph to the vertex s_i targeted by the request. Of course, this only works when the subject graph is a tree, in which case there is a single path from the root to s_i. A request can then match any rule that applies to any ancestor subject of s_i, since rule subjects are expressed as regular expression matching any path that contains the rule subject.

To compare the performance of XACML with SGAC, we have used Balana [12], an open-source implementation of XACML based on Sun's XACML implementation. The tests were performed on a server running a virtual machine (Intel(R) CPU 2.67 GHz, 4.00 GB RAM). Balana is written in Java. SGAC is written in NodeJS.

We have generated SGAC policies in a random fashion using a program that generates a subject tree and a resource tree with depth h and node branching factor b, which gives a tree of size $(b^h - 1)/(b - 1)$. Rules are randomly generated. The size of the trees ranged from 1093 to 21845 vertices. Fig. 6 shows the average request processing time versus the number of rules given in thousands. Here are some of the conclusions we drew from these results.

- Request processing time with XACML is significantly longer than SGAC's to evaluate the same request. When the number of rules is important (*e.g.*, 100 000 rules), SGAC is in average 300 times faster.
- Request processing time with XACML increases linearly with the number of rules whereas SGAC's is

Model	Native Subject Hierarchy	Native Resource Hierarchy	Dynamic Rules	Explicit prohibition	Autonomous Conflict Management	Ease of Rule Expression
RBAC	√	×	×	×	√	√
OrBAC	√	√	√	√	×	√
XACML	×	×	√	√	√	×
SGAC	√	√	√	√	√	√

TABLE VIII: Comparison between access control models having a formal model

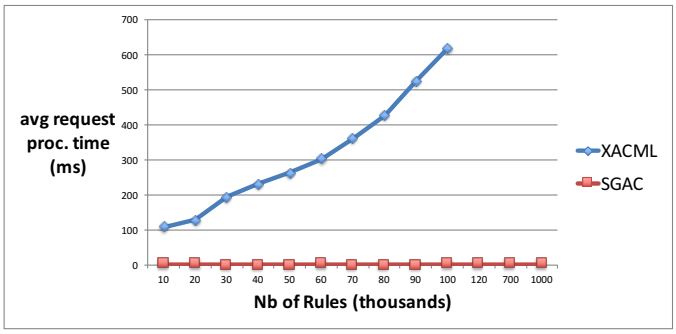


Fig. 6: Performances summary

near constant ($\leq$ 2 ms in average for up to 1M rules, and a maximum of 7 ms). SGAC uses an $n \log n$ algorithm for indexing rules at system initialisation, where n is the size of the subject graph, and a hash table provides a near constant time for fetching rules applicable to a request.

- When the number of rules is high (200 000 rules for instance), XACML cannot load the file containing the policies: the error returned refers to insufficient Java heap space, which remained even after increasing the memory to 12 GB on a 64-bit architecture. SGAC could process all tests on a 4GB virtual machine with a 32-bit architecture. XACML policies are written in XML, and they are quite verbose.

VIII. Conclusion

We have proposed SGAC, an innovative access control method, to meet the EHR access control and consent management requirements of a large hospital in Canada (CHUS). SGAC uses an intuitive ordering on rules to manage rule conflicts. This ordering uses priority to manage the different providers of rules and their precedence according to the applicable laws. Subject specificity and modalities are used to order rules of the same priority. SGAC's implementation can manage large policies (at least 1M rules) and large subject and resource graphs. Its implementation performs significantly better than Balana, an open-source implementation of XACML. SGAC's access control model offers flexibility in managing policies and in satisfying various laws on privacy in Canada. It should be applicable to other legislations in other countries, and to other application domains, like banking, insurance, social networks, government services, etc. In order to ensure patient safety, we have proposed a formal model of SGAC policies to enable automated analysis of policy properties. In future work, we plan to explore tools like Alloy [3], ProB [13] and Yices [14], to automatically analyse SGAC policies.

Acknowledgments

This research was funded by the Agence de la santé et des services sociaux de l'Estrie (ASSS). In particular, the authors would like to thank Hassan Diab, of ASSS-CHUS, and Mohammed Ouenzar, of Université de Sherbrooke (UdeS), for their contribution in defining SGAC, and all the SGAC development team at UdeS and ASSS-CHUS.

References

[1] E. Rissanen, *eXtensible Access Control Markup Language (XACML) Version 3.0*. OASIS, 2010.
[2] J. Abrial, *The B-book - assigning programs to meanings*. Cambridge University Press, 2005.
[3] D. Jackson, *Software Abstractions: Logic, Language and Analysis*. MIT Press, 2012.
[4] D. F. Ferraiolo, *Role-Based Access Control, Second Edition*. Artech House, 2006.
[5] R. S. Sandhu, E. J. Coynek, H. L. Feinsteink, and C. E. Youmank, "Role-based access control model," *IEEE Computer*, vol. 29, no. 2, p. 38–47, 1996.
[6] D. J. Power, M. Slaymaker, and A. Simpson, "On Formalizing and Normalizing Role-Based Access Control Systems," *The Computer Journal*, vol. 52, no. 3, pp. 305–325, 2009.
[7] N. Huynh, M. Frappier, A. Mammar, R. Laleau, and J. Desharnais, "Validating the rbac ansi 2012 standard using b," in *ABZ*, ser. Lecture Notes in Computer Science, Y. A. Ameur and K.-D. Schewe, Eds., vol. 8477. Springer, 2014, pp. 255–270.
[8] A. Kalam, R. Baida, P. Balbiani, S. Benferhat, F. Cuppens, Y. Deswarte, A. Miege, C. Saurel, and G. Trouessin, "Organization based access control," in *Policies for Distributed Systems and Networks, 2003. Proceedings. POLICY 2003. IEEE 4th International Workshop on*, June 2003, pp. 120–131.
[9] F. Cuppens, N. Cuppens-Boulahia, and C. Coma, "MotOrBAC: un outil d'administration et de simulation de politiques de sécurité," in *Security in Network Architectures (SAR) and Security of Information Systems (SSI), First Joint Conference*, 2006, p. 6–9.
[10] G. Russello, C. Dong, and N. Dulay, "Authorisation and conflict resolution for hierarchical domains," in *POLICY*. IEEE Computer Society, 2007, pp. 201–210.
[11] J. Bryans, "Reasoning about XACML policies using CSP," in *In SWS '05: Proceedings of the 2005 workshop on Secure web services*. ACM Press, 2005, p. 28–35.
[12] Balana. [Online]. Available: https://github.com/wso2/balana
[13] M. Leuschel and M. J. Butler, "ProB: an automated analysis toolset for the B method," *STTT*, vol. 10, no. 2, pp. 185–203, 2008. [Online]. Available: http://dx.doi.org/10.1007/s10009-007-0063-9
[14] B. Dutertre, "Yices 2.2," in *Computer-Aided Verification (CAV'2014)*, ser. Lecture Notes in Computer Science, A. Biere and R. Bloem, Eds., vol. 8559. Springer, July 2014, pp. 737–744.