Privacy of Medical Records: From Law Principles to Practice

Béatrice Finance, Saïda Medjdoub, Philippe Pucheral

Privacy of Medical Records: from Law Principles to Practice

Béatrice FINANCE [1,2], Saïda MEDJDOUB[1], Philippe PUCHERAL [1]

[1] SMIS Project – INRIA Rocquencourt, [2] PRiSM Laboratory University of Versailles,
{firstname.lastname}@inria.fr

Abstract

Regulating access to electronic health records has become a major social and technical challenge. Unfortunately, existing access control models fail in translating accurately basic law principles related to the safeguard of personal information (e.g., medical folder). This paper identifies the problem and proposes a solution in the EHR context.

1. Introduction

Setting up large-scale Electronic Health Records (EHR) systems has become a primary concern for several countries, with the objective to improve the quality of care while decreasing costs. However, practitioners and patients are reluctant to use such systems due to the threat on citizen's privacy. Organizing the safe sharing of medical folders among several parties (patients, physicians, pharmacists, medical labs, Medicare and insurance companies) having different duties and objectives is indeed a real challenge.

Government's enact laws related to the safeguard of personal information [3,6,7]. Considering their high sensitivity, specific laws are dedicated to the protection of medical records, like the well recognized Health Insurance Portability and Accountability Act (HIPAA) [7]. More than ever, there is a strong need to define access control models that help translating law principles into practice. Among these principles, the basic *need-to-know* and *consent* principles are particularly difficult to deal with. The need-to-know principle limits access to information to those people who need strictly this information to carry out their duties. The consent principle means that the donor must be given some prerogative (framed by the law) to control how her information (e.g., her medical folder) is exposed and made accessible to others.

At the same time, a strong standardization effort is done to describe, store and exchange health records in XML [5]. Regulating access to XML documents has attracted a considerable attention in recent years [1,2,4]. All these works have the commonality to focus the access control on the nodes of an XML document (elements and attributes). In this context, the contribution of this paper is threefold. First, it shows that existing access control models fail in translating accurately the need-to-know and consent principles in a number of situations. Second, it tackles this issue by integrating XML relationships as first class citizen in the access control model. Third, it validates the model in the EHR context.

2. Problem statement

As pictured in Figure 1, an XML document can be represented as a tree where nodes, also called *elements* (e.g., `Folder`), are linked by edges, also called *relationships*. Relationships between elements may reveal information as sensitive as the one carried out by the elements themselves and hence, deserve to be protected as such. More precisely, disregarding XML relationships in the access control leads to two important problems.

Proceedings of the 18th IEEE Symposium on Computer-Based Medical Systems (CBMS'05)
1063-7125/05 $20.00 © 2005 **IEEE**

– *Classification disclosure*: the hierarchical structure of an XML document often reveals a classification. To illustrate this, the membership of an element (e.g., a patient folder) to a given subtree (e.g., a medical service) conveys its classification (e.g., the pathology the patient is treated for). Existing access control models for XML fail in hiding this information. Every time this information is not strictly mandatory to achieve a given purpose, this hurts the *need-to-know* principle.

– *Uniform filiation:* in existing access control models, there is no way to deliver two different views of the path leading to two different XML elements, thereby hurting the *consent* principle. For example, one patient may request to hide the medical service she is treated in while another consents disclosing this information.

3. Case Study

Our case study is built from requirements expressed by a real life medical application related to the treatment of AIDS disease. Below are examples of important authorization rules supported by our model and that cannot be managed by existing XML control models:

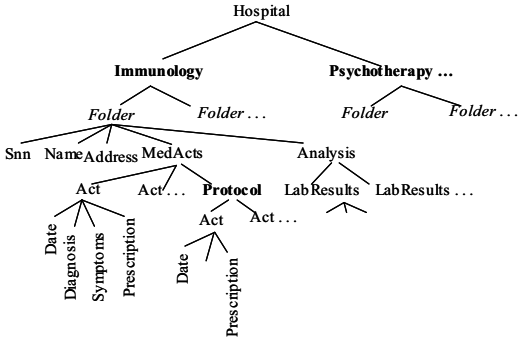

Figure 1. XML Medical Folders

- ***R1****: Hide to the hospital's directory application the name of the service where patients are treated, for those who didn't consent making this information public.*

As stated in HIPAA [7], the hospital directory is a rather sensitive information considering the inquiries made about patients by relatives, employers, media, police and members of religious groups. The effect of this authorization rule on the document pictured in Figure 1 should be to attach the `Folder` element of each patient of interest to a depersonalized medical service element (i.e., element with an anonymous label) while keeping the ancestor chain of the other folders unaffected. As pictured in Figure 2.a, this restructuration must be done in a way that prevents classification disclosure.

- ***R2****: Hide to pharmacists the fact that some drug prescriptions participate in a protocol (i.e., a medical trial).*

The pharmacist must be aware of all prescriptions to check drug incompatibilities but giving him the knowledge that some drugs participate in a protocol discloses important information on the patient's disease and its stage. The expected effect of this authorization rule is to drop `Protocol` elements and attach `Act` elements as direct children of their `MedActs` ancestor, giving them a position similar to regular `Act` elements (see Figure 2.b). Depersonalizing Protocol is useless since that information would be obvious to infer.

- ***R3****: Hide to a medical lab the correlation between the medical acts and analysis information on one side and the identification information on the other side.*

HIPAA stipulates that the patient consent is required for any disclosure related to marketing. Let us assume that the first group of elements (`MedActs`, `Analysis`) is required wrt the need-to-know principle while the second (`Name`, `Address`) is collected under the patient consent for marketing purpose (e.g., related to new medications). The expected effect of this authorization rule is to make both groups of information available while precluding the inference of their initial sibling relationship, as shown in Figure 2.c.

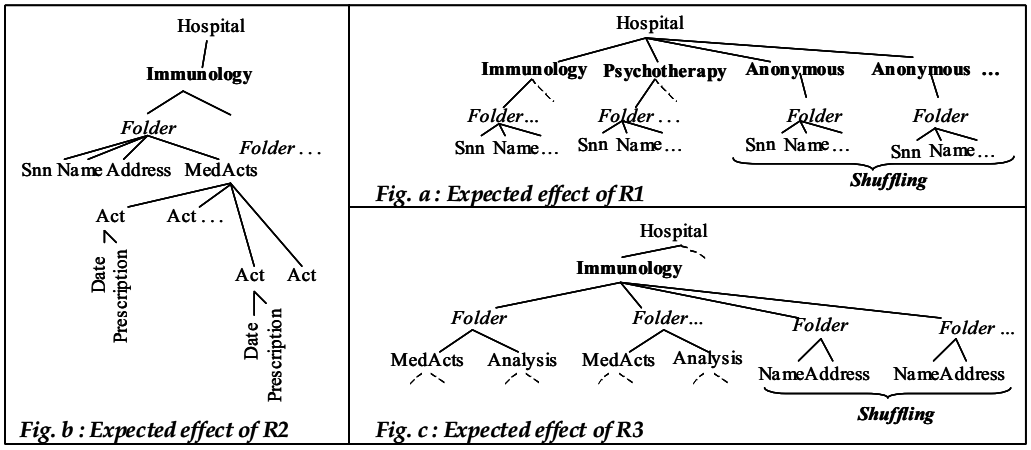

Figure 2. Authorized Views.

Existing XML access control models interpret an access control policy as a mapping between a source document (or `Source`) and an authorized view of this same document (or `View`) and rely on the assumption that `View` $\subseteq$ `Source`. More precisely, authorization rules select the subset of Source nodes that will participate in `View`. As a side effect, edges having one of their extremity node discarded by an authorization rule are in turn discarded from `View`. Considering the authorizations described above compels us to revisit this assumption since `View` may result from a more complex restructuration of `Source`.

4. Relationship-aware access control model

This section introduces briefly our access control model. A more detailed analysis of the limits of existing access control models and the foundation of our own model can be found in [8]. First, we introduce two mandatory mechanisms, namely cloning and shuffling, to translate the user's consent principle into an authorized view of an XML document. Then, we present a reference model for expressing node authorizations that captures the common foundation of existing XML access control models. Finally, we propose an extension to this reference model that supports relationship authorizations. Rather than proposing yet-another access control model for XML, we show that the proposed approach allows a seamless integration of relationship authorizations in existing access control models.

4.1. Cloning and Shuffling mechanisms

Taking into account the user's consent in access control models imposes to generate in `View` different replicas of the same `Source` nodes and paths. Basically, replicating a `Source` node *n1* is required each time two of its authorized descendants *n2* and *n3* must be reachable in `View` by a path delivering two conflicting visions of *n1* to conform to the semantics of a given authorization rule. Rule R1 of our motivating example illustrates this situation. Since an XML document is a tree, every node participating in the common subpath Path(n1,Parent(n2))$\cap$Path(n1,Parent(n3)) has in turn to be replicated.

Cloning is the principle by which `Source` elements and paths are replicated in `View`. The ordering of clones in `View` has to be carefully managed to avoid basic inference. To illustrate this, let us consider Rule R3 of our motivating example and assume that the `View` ordering is such that all instances of the two groups (`MedActs`, `Analysis`) and (`Name`, `Address`) keep the same relative order as in `Source`. In this case, their initial sibling relationship, which should be obfuscated by the cloning mechanism, is patently disclosed by the element ordering (i.e., the i^{th} instance of `MedActs`, `Analysis`) corresponds to the i^{th} instance of (`Name`, `Address`)). A similar problem exists with Rule R1 if the clones of a medical service element are placed in close proximity to their original (e.g., direct right or left sibling). Thus, cloning does not make sense without *node shuffling*.

Node shuffling is a recursive process that applies at each node of `View` containing clone children. All clones, children of a given node, are shuffled together to prevent ordering-based inference. For a given node, the clone children are grouped after the original ones (by convention), and then shuffled. The relative order of the original children must however be preserved in `View` since node ordering is significant in XML.

4.2. Reference model for node authorizations

While existing XML access control models introduce subtleties on the way node authorizations propagate down through the hierarchy and conflicts are solved, they share strong commonalities. Basically, an authorization rule takes the form of a tuple <*Subject*, *Object*, *Operation*, *Sign*>. Depending on the models, *Subject* can take many forms (a user, a group of users, a role, etc). *Object* characterizes the part of the XML document targeted by the rule by means of an XPath expression (i.e., a regular expression on trees). *Operation* denotes the operation (read, update, delete, append) the Subject may perform on the Object. Finally, *Sign* denotes either a permission (grant rule) or a prohibition (deny rule) for that operation. In the sequel, we do not make any assumption on the way subjects are managed and, since the focus is on data confidentiality, read is the only operation of interest. Hence, the node authorization rules considered below, denoted by NA, are simply defined by <*Subject*, *Object*, *Sign*> where *Subject* is an abstract entity, *Object* is an XPath expression applied on `Source`, and *Sign* $\in$ {+, -}.

To match the well accepted least privilege principle, we consider a closed policy, meaning that an implicit negative authorization applies to the whole document. In other words, the access to every object that is not explicitly authorized is forbidden. We assume that both positive and negative authorizations propagate implicitly down through the XML hierarchy. This mode of propagation corresponds to the cascading option present in well-known models [1,2,4]. Conflicts between direct and/or propagated rules are managed as follows. Let us assume two rules R1 and R2 of opposite sign. These rules may conflict because they are defined either on the same node, or on two different nodes n1 and n2, linked by an ancestor relationship (i.e., n1$\in$Anc(n2)). In the former situation, the Denial-Takes-Precedence policy favors the negative rule according to the least privilege principle. In the latter situation, the Most-Specific-Object-Takes-Precedence policy favors the rule that applies directly to a node against the inherited one (i.e., R2 takes precedence over R1 on n2). In other words, authorizations propagate until overridden by an opposite authorization on a descendant node.

4.3 Relationship authorization rules

A relationship authorization rule, denoted by RA, is defined by a tuple <*Subject*, *Object*>, where *Object* is in turn defined by a 4-tuples: <*Anc, Desc, Path-visibility, Sibling*>

- *Anc* and *Desc* characterize the relationship(s) to be protected among a (set of) descendant(s) and one of its (their) ancestor. Anc and Desc are the common denominator of all relationship authorizations. They are both defined as XPath expressions.
- *Path-visibility* characterizes the vision of the path *u* linking each descendant node to its ancestor. For each node *n* participating in *u*, Path-visibility states whether the node is preserved or not in the path clone of *u* and, in the positive case, whether *n*'s label is preserved or not. Implicitly, hiding an ancestor relationship hides the relationship between a descendant node and its siblings.
- *Sibling* characterizes the list of siblings a descendant must keep its relationships with, to allow for a selective sibling decorrelation.

The RA definition deserves two important remarks. First, regarding conciseness and manageability, RA captures gracefully and in a rather simple way the different forms of relationship authorizations. By defining Anc and Desc as XPath expressions, it allows to sump up ancestor/descendant relationships in a single statement. Second, unlike NA, RA does not integrate a Sign parameter. The reason for this is that RA characterizes only negative authorizations.

The global semantics of the model is as follows. NA rules are defined according to a closed policy and deliver an authorized view `View'` $\subseteq$ `Source` in the usual way (i.e., edges having one of their extremity node discarded by a NA rule are in turn discarded from `View'`). RA rules are defined on `View'` according to an open policy and deliver the final authorized view `View`. Consequently, if no RA rule is defined, the semantics of the model complies with the one of the existing XML access control models. Hence, a seamless integration of relationship authorizations in these models can be reached. Table 1 (resp. Table 2) summarizes the possible choices for the Path-visibility (resp. Sibling) parameter along with their associated semantics. The first row of each table gives an extensive syntax for the corresponding parameter while the next rows propose shortcuts to express a monotonic policy along the path.

Path-visibility	**Semantics of Path visibility**
[$label_1$?,., $label_n$?]	gives the list of nodes to be discarded (?=†) or depersonalized (?=Φ).
[]	all nodes are kept on the path (i.e., all nodes are cloned) and their original label is inherited. This option is the default one.
[Φ]	all nodes are kept on the path and are depersonalized (i.e., the label of their respective clone is set to "anonymous").
[†]	All nodes are discarded from the path.

Table 1. Path-visibility semantics

Sibling	**Semantics of Sibling**
[$label_1$,… $label_n$]	Nodes those label belongs to this list must keep their sibling relationship with the descendant node of interest.
[⊥]	The descendant node is disconnected from all its siblings. This is the default option.
[ψ]	The descendant node preserves its sibling relationships with all siblings targeted by the same authorization rule as him.
[≡]	The descendant node preserves all the sibling relationships it is involved in.

Table 2. Sibling semantics

Figure 3 illustrates the use of a relationship-aware access control model for expressing the access control rules introduced in our motivating example. Each of these rules actually mix NA and RA rules. Some NA and RA rules reference the user's consent. We do the assumption that the user's consent is materialized by a `Consent` element present in each folder. The `Consent` element is in turn composed of sub-elements (e.g., `directory`,

marketing) expressing each dimension of the user's consent. For expressing R1, three NA rules are required which capture the information strictly required by the hospital's directory group to accomplish their duty (typically, MedActs and Analysis are withdrawn). RA1 depersonalizes ([Φ]) the medical service ancestor (/* targets all medical service elements) of each folder owned by a patient who didn't consent disclosing that information and disconnects that folder from its siblings ([⊥]). For rule R2, RA2 alone expresses a path reduction discarding the parent Protocol ([†]) of Act elements. For expressing R3, two NA rules deny to the medical lab access to the name and address of patients who didn't consent disclosing this information for marketing purpose. For patients giving their consent, RA3 precludes the inference between the identification information (Name, Address) and the rest of the folder.

***Rule R1*:**

NA1: < DirectoryGroup, /Hospital, + >

NA2: < DirectoryGroup, //MedActs, - >

NA3: <DirectoryGroup, //Analysis, - >

RA1: <DirectoryGroup,/Hospital/*,/Folder[./Consent/Directory/Service='no visible'],[Φ],[⊥] >

Rule R2:

RA2: < Pharmacist, //MedActs/Protocol, /Act, [†],[⊥]>

Rule R3:

NA5: < Medical lab, //Folder[./Consent/Marketing/PersonalInfo='no visible']/name, - >

NA6: < Medical lab, //Folder[./Consent/Marketing/PersonalInfo ='no visible ']/Address, - >

RA3: < Medical lab, //Folder, /Name, [], [Address]>

Figure 3 : Motivating example's NA & RA rules.

5. Conclusion

Regulating access to electronic health records has become a major concern for governments, practitioners and citizens. This paper shows that existing access control models are unable to handle accurately the sophisticated authorizations required to preserve medical data privacy. To cope with this issue, we introduced an extended access control model for XML [8] better capturing the basic user's consent and need-to-know principles. An experimentation is going on with hospitals, clinics and general practitioners from the Yvelines region in France to assess the benefit of our access model in the context of a regional EHR system.

6. References

[1] Bertino, E., Castano, S., Ferrari, E., Mesiti, M. Specifying and Enforcing Access Control Policies for XML Document Sources. WWW Journal 3(3), 2000.

[2] Damiani, E., De Capitani di Vimercati, S., Paraboschi, S. , Samarati, P. A Fine-Grained Access Control System for XML Documents, ACM TISSEC 5(2), 2002.

[3] European Directive 95/46/EC, "Protection of individuals with regard the processing of personal data",Official Journal L 281, 1985.

[4] Gabillon, A., Bruno, E. Regulating access to XML documents. IFIP Conf. on Database and Application Security, 2001.

[5] HL7: Health Level 7 (http://www.hl7.org)

[6] The Privacy Act, 5 U.S.C. § 552a, 1974. http://www.usdoj.gov/04foia/ privstat.htm.

[7] United States Department of Health and Human Services, "HIPAA : Health Insurance Portability and Accountability Act", Public Law 104-191, 104th Congress, 1996. http://www.hhs.gov/ocr/hipaa/

[8] B. Finance, S. Medjdoub, P. Pucheral, 'The Case for Access Control on XML Relationships', INRIA internal report, n°5446, 2005.