Succinct Access Control Policies for Published XML Datasets

Tomasz Müldner[1], Jan Krzysztof Miziołek[2], Gregory Leighton[3]
[1]Jodrey School of Computer Science, Acadia University, Wolfville, N.S. Canada
tomasz.muldner@acadiau.ca

[2]IBI AL, University of Warsaw, Warsaw, Poland
jkm@ibi.uw.edu.pl

[3]Department of Computer Science, University of Calgary, Calgary, Canada
gleighto@cpsc.ucalgary.ca

Keywords: Access control, XML, parameterized roles.

Abstract: We consider the setting of secure publishing of XML documents, in which read-only access control policies (ACPs) over static XML datasets are enforced using cryptographic keys. The role-based access control (RBAC) model provides a flexible method for specifying such policies. Extending the RBAC model to include role parameterization addresses the problem of role proliferation which can occur in large scale systems. In this paper, we describe the complete design of a parameterized RBAC (PRBAC) model for XML documents. We also describe algorithms for generating the minimum number of keys required to enforce an arbitrary PRBAC policy; for distributing to each user only keys needed for decrypting accessible nodes; and for applying the minimal number of encryption operations to an XML document required to satisfy the protection requirements of the policy. The time complexity of our approach is linear w.r.t. document size and the number of roles.

1 INTRODUCTION

There is growing interest in providing controlled and secure access to XML documents [(Bertino et al, 2002), (Bertino et al., 2004), (Damiani et al, 2002), (Devanbu et al, 2001)]. In this context, *controlled* access allows the owner of data to specify permission policies indicating which users can access specific documents, or parts thereof. *Secure* access to data means that data is confidential, i.e., visible only to authorized parties. Since XML data has become a de facto standard for many applications, in particular for Web applications, much of the research done on controlled access in recent years deals with data in this format. In addition to proposed standards for encrypting portions of an XML document, such as the W3C XML Encryption Syntax and Processing recommendation (W3C XML Encryption, 2001), secure publishing approaches have been proposed in the research community that illustrate how multiple users holding varying permissions over an XML document can gain controlled access to the same version of that document, through the selective application of cryptographic keys [(Bertino et al, 2002), (Mikalu et al, 2003), (Müldner et al, 2006)]. Using this technique, a single version of an XML document is published (e.g., posted to an HTTP server). Each user is provided with a set of keys via a secure channel; by applying her available decryption keys, a user gains access only to the portions of the document authorized by the designated ACP. For example, the contents of a statistical or scientific database may be periodically sanitized (to remove sensitive information), exported as XML, and published. In such a setting, it is important to note that each published document is static, and hence access control policies only allocate read permissions. The *role-based access control (RBAC)* model provides a powerful and flexible method for specifying such policies. However, the RBAC model is susceptible to role proliferation. For example, thousands of scientists may be granted access to various parts of an XML dataset; the access permissions accorded to each scientist may vary according to their specialization and their affiliation. In the worst case, it is possible that a role-based policy must assign a unique role to each scientist. The concept of *role parameterization* has been shown to be an effective solution to role proliferation (Ge et al, 2004); instead of defining a separate role for each scientist, one can use a much smaller number of roles and parameterize each role with variables representing area of specialization and affiliation. With a smaller number of roles, it becomes easier to formulate and enforce the desired access control restrictions.

In this paper, we introduce techniques that can be used to implement controlled and secure access to published XML documents. Specifically, we define *parameterized role-based ACPs* (*PRBAC policies*); each such policy consists of a set of rules associating a role with one or more views (or fragments) defined over an XML document. Policy rules may contain role parameters and/or system variables. Once a user is authenticated to play role R, any role parameters and system variables are instantiated and the user can access only those views which are associated with role R. We have designed a key assignment algorithm which, given a PRBAC policy and an XML document (or dataset), generates the minimal number of keys required to enforce the stated policy. Generated keys are used to *multi-encrypt* a document so that each element of the document is encrypted with at most one key. Since each view may be encrypted with multiple keys, users playing a specific role R are provided with an *R-accessible keyring*, consisting of the set of keys needed to decrypt and access *exactly the document portions* they are allowed to see.

This paper is organized as follows. Section 2 describes related work. Section 3 introduces preliminary notation and concepts and Section 4 defines the language of parameterized roles and the PRBAC model. Section 5 describes key generation and multi-encryption of documents. Section 6 describes areas for future work. Because of space restrictions, proofs are omitted.

Contributions

Our contributions include the complete description of a PRBAC model tailored for static, published XML datasets. To our knowledge, this is the first formulation of such a model. We also detail an approach that, for a given *uninstantiated* PRBAC policy (i.e. even before values of parameters and system variables are known) and XML document D (1) generates the minimum number of keys needed to multi-encrypt D; (2) applies the minimal number of encryption operations on D needed to enforce the PRBAC policy; and (3) for each role R in the PRBAC, generates the R-accessible keyring. All of these steps can be carried out using *two SAX-based traversals* of D.

2 RELATED WORK

Motivated by the increasing use of XML as a data representation format, several access control models specifically tailored for XML have been proposed in recent years. Such approaches permit the formulation of fine-grained access control policies at the schema, document, and/or element level. At a high level, it is possible to distinguish between *materialized view-oriented* approaches, in which client queries are answered over a sub-document (view) generated by the database management system, containing only the accessible regions of an XML database.[(Bertino et al, 2002), (Damiani et al, 2002)], and *secure publishing* approaches (Miklau et al 2003) and (Müldner et al, 2006), in which a *single*, partially-encrypted version of a document is distributed and access control policies are enforced using public-key cryptography. While materialized view-oriented approaches hide the original document from the client, a very large number of materialized views may be required in applications dealing with large, complex documents and/or several users. Secure publishing approaches are designed for cases in which it is unnecessary, and even undesirable, to allow users direct access to a database, and instead provide to users a published, static “snapshot” of the database contents. Our approach follows the secure publishing paradigm.

Role based ACPs have been extensively researched [(Ferraiolo et al, 2001), (Osborn et al, 2000), (RBAC, 2008), (Wang et al, 2004)]. (Ge, 2004) describes an extension to the role-based access control model in which parameterized roles are used to deal with scenarios in which data access is dependent on certain characteristics held by an individual user. In applications with a small number of users, it is feasible to define a separate role for each individual user, yet this approach clearly becomes unmanageable if the user base is moderately large. Rather than defining several thousand roles with a membership of one, an administrator can define a single, parameterized role, and specify an access control rule which dictates access to specific data. Our approach applies the notion of parameterized roles to ACPs for XML documents, allowing them to be used for expressing access control policies.

3 PRELIMINARIES

3.1 Introduction to Roles

In most systems, access rights are defined using access control policies (ACPs) which state which subjects have specific rights. In the *role-based access control model (RBAC),* users are identified through roles, and access rights are associated with each role. When a user is assigned to a role, they acquire the role's permissions.

In general, XML access control policy rules are five-tuples (subject, resource, action, permission, propagation), with actions such as read, write, or modify; permissions such as add or remove; and a propagation policy that allows one to limit the rule to a local scope or apply it to a more global scope (Fundulaki et al, 2004). In this paper, we consider only *read access* (since we focus on secure publishing scenarios, writes are not applicable), and we do not consider various permissions. We fix a *no propagation* policy: specifying an element *e* in an access control rule applies the rule only to *e*, and not to other elements within the sub-tree rooted at *e*. Therefore, role-based access control rules for XML documents in this paper will consist of pairs of the form (role, resource), where a resource is a document fragment specified using an XPath expression (XPath, 2008).

3.2 Documents and Views

The focus of our work is to define ACPs for fragments of XML documents, which we call *views*. In our approach, we publish a single multi-encrypted XML document (or dataset).

We assume that at the system level, there are pre-defined system variables (such as user ID), and we use identifiers starting with $ to denote these variables (e.g., $ID). System variables are typed using XML Schema types (XML Schema, 2008) (e.g., $ID: xs:integer), and they must remain static during the course of a user session (i.e., we do not consider dynamic system variables representing values such as the current time or the number of users currently in the system). In other words, there is a fixed set of static variables, and for each user values of all these variables are initialized and do not change during the session, i.e., until this user logs off.

Views are specified using a subset of XPath expressions referred to as *document paths*. Document paths may have conditions, and in XPaths these conditions use element names and attributes. Here, we extend XPaths and allow *free variables* to appear in conditions. There are two kinds of free variables: those that represent *system variables* and whose names start with $, and those that represent *formal parameters* and whose names start with % (for more on formal parameters of roles, see Section 4.1). We assume that a single comparison in the path can use *at most one* free variable; e.g. we don't allow conditions of the form [$Id = %Id].

Definition 3.1

A *local* document path is a document path with no free variables. A *global* document path is a document path which is not local. A global document path is called *instantiated* when each occurrence of free variables is replaced by some value.□

For a document D, by $P_{D,\ loc}$ we denote the set of local paths in D. Each local document path defines a fragment of the document D, which we call a *view* of D (and we frequently refer to the path P as the view P). Similarly, by $P_{D,\ glob}$ we denote the set of global paths in D, and the set of all document paths is denoted by $P_D = P_{D,\ loc} \cup P_{D,\ glob}$. A local path and an instantiated global path each define a fragment of the document D.

3.2 Multi-encryption

In this section, by a *key* we refer to a symmetric cryptographic key (e.g., an AES key). A *keyring* is a set of keys. An ACP may define multiple views for a single document. By the *multi-encrypted* document *m-document*, we denote a document with an associated keyring $\mathcal{K}$, in which various views may be encrypted with different sets of keys from $\mathcal{K}$. However, no node in the m-document is super-encrypted, i.e., encrypted more than once. Based on permissions defined in the ACP, users will be provided a subset of $\mathcal{K}$ enabling them to decrypt exactly those views they are entitled to see.

4 ROLE BASED ACCESS CONTROL

In this section, we first describe roles and then use roles to define the document-level PRBAC.

4.1 Language of Roles

In our access control model, roles can be parameterized or parameter-less. *Parameterized* roles contain typed parameters. They are useful because they help to avoid hard-coding multiple roles which provide permissions depending on external values known when the role is being assigned; for example:

- There are many departments with different names.
- There is a "security level" attribute and permissions depend on the given security level.

A parameterized role is called *instantiated* if all its formal parameters are replaced by actual values.

Let N be the set of role names, P the set of parameter names, and T the set of parameter type names (we assume that these three sets are mutually disjoint). We define the alphabet A as the union $N \cup P \cup T \cup \{(,), ;\}$. We write role names in upper case (e.g., STUDENT). Parameter names start with %.

Definition 4.1
A *role grammar* Γ over the alphabet A is defined as follows:

role := roleName | parameterizedRole
parameterizedRole := roleName '(' rolePars ')'
rolePars := formalPar | formalPar ';' rolePars
formalPar := parName':' parTypeName

where roleName$\in$N, parName$\in$P, and parTypeName$\in$T. □

Example 4.1
An example of a parameterized role is:

USER(%id : xs:integer; %name : xs:string). □

4.2 Simple Roles

In Section 4.1 we defined the role grammar Γ, and by Λ_Γ we denote the language of all roles.

Definition 4.2
For an XML document D and a finite set of simple roles $\Psi \subset \Lambda_\Gamma$, the *document-level ACP* is a mapping π_D: $\Psi \rightarrow P_D$ such that $\pi_D(\Psi)$ covers the set D; i.e. each element of D belongs to at least one document path that occurs in the policy. Often, the π_D mapping is *tabulated* and shown as a tuple $[(R_1,P_1), (R_2, P_2),...,(R_n,P_n)]$. □

For a simple role $R \in \Psi$, if $\pi_D(R)$ is a *local* document path then it defines a view of D. If $\pi_D(R)$ is a *global* document path that contains free variables (system variables or formal parameters for a parameterized role R), then once this path is instantiated, it defines a view of D. The designer of a PRBAC policy for an XML document D may decide to leave some parts of D unencrypted (accessible to all users) or to make them inaccessible to all users (i.e., to encrypt them, but not to provide the keys used for encryption of these nodes to any user). For the former case, the symbol ○ is used, while for the latter case we use the symbol ●. Therefore, the actual definition of the document-level ACP is that it is pair (π_D, ◊), where ◊ is either ○ or ●. For simplicity (unless specified otherwise), in the sequel, we omit the second element of this pair, and assume that by default it is always ○ (i.e. by default, elements of D not covered by π_D are unencrypted).

Definition 4.3
The *document-level protection requirement* is said to be satisfied under the following conditions. For an XML document D, a finite set of roles $\Psi \subset \Lambda_\Gamma$, and the document-level ACP π_D: $\Psi \rightarrow P_D$ a user in role R can access *precisely* the set $\pi_D(R)$, and those nodes in D which are not covered by any path.□

5 KEY GENERATION AND MULTI-ENCRYPTION

Definition 5.1
A *keyring* $\mathcal{K}$ is a finite set of keys, where each key is a 2-tuple <key name, symmetric key>. By $\mathcal{K}_{D,\pi_D}$ we denote a document-level keyring for the document D and D's policy π_D.

5.1 Creating Local Paths from Global Paths

In this section, we describe keyring generation at the document level, corresponding to a *document-level policy* π_D: $\Psi \rightarrow P_D$. If all paths from the set P_D are *local* then keyring generation can take advantage of the fact that data in the document D can be used to evaluate all conditions in these paths, and therefore each path uniquely identifies a fragment of D. The situation is more complicated if the policy has some *global* paths; for example, it has a parameter-less role and an associated path with conditions using system variables, or it has a parameterized role and an associated path with conditions using formal parameters of this role. In such cases, conditions in paths can not be evaluated until values of corresponding *free variables* are known. However, postponing keyring generation until such time would mean that, for a parameterized role, a keyring would be generated for each user that can play this role (using specific actual parameters), possibly resulting in unnecessary duplication of keys. Therefore, we employ a different technique and deal with global paths in a special way.

Recall that a global path P contains free variables representing role parameters and/or system variables. We require an algorithm that finds all values in P which are being compared with free variables, and replaces P with local paths in which these variables are replaced with values relevant for evaluating conditions. In the current version, there is a restriction on the type T of any free variable that may appear in a global path; specifically, we assume that T has a linear ordering $<$ and $\leq$, here called a *linear order* that satisfies the following condition: for any three elements x, y and z of type T; if $x<y$, then exactly one of the following three inequalities holds: $z<x$, $x\leq z<y$, or $y\leq z$. For example, integer or real free variables may be used. We define intervals of the form$(v_1, v_2) = \{v: v_1 < v < v_2\}$, $[v_1, v_1] = \{v: v = v_1\}$, $(-\infty, v_1) = \{v: v < v_1\}$ and $(v_1, +\infty) = \{v: v > v_1\}$. Then, the following property holds: given *n* arbitrary values $v_1,v_2,...,v_n$ of type T, and intervals of the form $(-\infty, v_1)$, $[v_1, v_1]$, (v_1, v_2), $[v_2, v_2]$, … $[v_n, v_n]$, $(v_n, +\infty)$, the value of variable x of type T belongs to exactly one of these intervals.

The auxiliary Algorithm 5.1 given below considers a role R and a global path P, and splits P into one or more local paths. More specifically, it builds a set of intervals that partitions the set of values of type T, based on values from the document D used in comparisons with free variables in P, and a set of local paths corresponding to all instantiations of P using values of the set of intervals.

Algorithm 5.1

Input: A document D, a simple role R (different from ○), and a global document path P with *n* free variables $A_1,A_2,...,A_n$ of respective types $T_1,T_2,...,T_n$; each type T_i has a linear order.
Note that some free variables may represent parameters of a parameterized role R and others may represent system variables; in particular R may be parameter-less and all free variables in P may be system variables, or vice-versa.

Output: A vector Locals[R] consisting of pairs of the form (local path, keyring), and a table Intervals(D, R, P) containing pairs (*k*-tuples of *intervals*, index into Locals). The complexity of this algorithm is linear in terms of the size of D.

5.3 Generating Keyrings and Encrypting

Consider an XML document D and its document-level access control policy $\pi_D: \Psi \rightarrow P_D$, where $\Psi \subset \Lambda_\Gamma$ is a finite set of roles. In this section, we define a keyring $\mathcal{K}_{D, \pi_D}$, and show how for each role $R \in \Psi$ this keyring defines the set $\mathcal{K}_{D,\pi_D}(R)$ of *R-accessible keys*. A user in role R will be provided with the R-accessible keys, i.e. the keyring $\mathcal{K}_{D,\pi_D}(R)$ allowing her to decrypt the view $\pi_D(R)$.

We now consider various ways keys may be assigned to documents. Since views may be overlapping, we can not assign keys *per view*. Indeed, if we did so, then for two overlapping views V_1 and V_2 we would have two corresponding keys κ_1 and κ_2, and the intersection of the two views would have to be super-encrypted using both κ_1 and κ_2 (assigning keys per view would mean that the user who has access to the view V_i would be given a key κ_i and so other options such as encrypting V_1 with one key and V_2-V_1 with the other key would have given access to the part that is not allowed). What we need to do is to partition the set D (and at the same time each view) into *disjoint* sets, and then assign one key for each set in this partition.

Below, we will assume that all views in $\pi_D(\Psi)$ are different. If this is not the case, then we proceed as follows. For the policy $\underline{\pi_{\underline{D}}}$ obtained by removing duplicate views from π_D (and corresponding roles), we generate a keyring $\mathcal{K}_{D,\underline{\pi_{\underline{D}}}}$ (which also defines $\mathcal{K}_{D,\underline{\pi_{\underline{D}}}}(R)$). Then, we define $\mathcal{K}_{D,\pi_D} = \mathcal{K}_{D,\underline{\pi_{\underline{D}}}}$, $\mathcal{K}_{D,\pi_D}(R) = \mathcal{K}_{D,\underline{\pi_{\underline{D}}}}(R)$ if R was not a duplicate role, and $\mathcal{K}_{D,\pi_D}(R) = \mathcal{K}_{D,\underline{\pi_{\underline{D}}}}(R_1)$ if R was removed and R_1 is its duplicate (i.e., R and R_1 were associated with the same view).

We assume that conditions that appear in paths can be evaluated during a single SAX-traversal of the document D.

Algorithm 5.2

Input: XML document D, and policy $\pi_D: \Psi \rightarrow P_D$ such that all views in $\pi_D(\Psi)$ are different.
Output: Keyring $\mathcal{K}_{D,\pi_D}$, multi-encrypted XML document M_D, for each $R \in \Psi$, and two vectors Global[] and Local[] (which will be used to identify R-accessible keyrings $\mathcal{K}_{D,\underline{\pi_{\underline{D}}}}(R)$, as described below). □

Theorem 5.1

If the user in role R is provided only with R-accessible keys, then the document-level protection requirement (cf. Def. 4.3) is satisfied. □

5.4 Decrypting m-documents

Assume that a user U in role R has obtained the keyring $\mathcal{K}_{D,\underline{\pi_D}}(R)$ of R-accessible keys and wants to decrypt an m-document M_D. U will traverse M_D, and use *names* of keys from $\mathcal{K}_{D,\underline{\pi_D}}(R)$ to extract the appropriate key to decrypt R-accessible nodes.

5.5 Obtaining Keyrings

U may request a keyring that she will use to decrypt a fragment of the m-document D. From the list of roles U is currently playing, she selects a certain role R, and then proves that she plays R by presenting the certificate obtained when U was granted R. Being in role R, U will specifically request the keyring $\mathcal{K}$ of *R-accessible keys* for the document D. Let P = π_D(R). If R is a simple role and the path P = π_D(R) is a *local* path, then $\mathcal{K}$ is retrieved from the appropriate keyring Local[]. If π_D(R) is a global path, then the set Intervals(D, R, P) is consulted. The set of values of actual parameters (if any) of R and the values of system variables are used to determine the specific cube that contains all these values. If there is no such cube, U has *no* permissions granted; otherwise, the keyring at the index associated with this cube is returned.

6 CONCLUSIONS AND FUTURE WORK

In this paper, we described a parameterized role-based access control (PRBAC) model allowing permissions to be specified over fragments of published XML datasets. Our future work will focus on implementing the PRBAC model, and designing and implementing algorithms allowing existing keyrings to be incrementally altered in response to update operations performed on the document and/or access control policy.

REFERENCES

Bertino, E., Ferrari, E. Secure and Selective Dissemination of XML Documents. *ACM Transactions on Information and System Security* (TISSEC), 5(3):290–331, (2002).

Bertino, E., Carminati, B., Ferrari, E., Thuraisingham, B., and Gupta A. Selective and Authentic Third-Party Distribution of XML Documents. *IEEE Transactions on Knowledge and Data Engineering* (TKDE), 16(10):1263-1278, (2004).

Damiani, E., De Capitani di Vimercati, S., Paraboschi, S. and Samarati, P. A Fine-grained Access Control System for XML Documents. *ACM Transactions on Information and System Security*, 5(2): 169-202, (2002).

Devanbu, P., Gertz, M., Kwong, A., Martel, C., Nuckolls, G. and S.G. Stubblebine. Flexible Authentication of XML documents. *In Proc. of the 8th ACM Conf. on Computer and Communications Security*, ACM Press, (2001).

Ferraiolo, D.F., Sandhu, R., Gavrila, S., Kuhn, D.S. and Chandramouli, R. Proposed NIST Standard for Role-Based Access Control. *ACM Transactions on Information and System Security*, Vol. 4, No. 3, (2001), 224–274.

Fundulaki, I. and Marx, M. Specifying access control policies for XML documents. *Proceedings of the ninth ACM symposium on Access control models and technologies* (2004) 61 – 69.

Ge, M. and Osborn., S.L. A Design for Parameterized Roles. *DBSec* (2004), 251-264.

Miklau, G. and Suciu, D. Controlling Access to Published Data Using Cryptography, In *Proc. of the 29th VLDB Conference*, Berlin, Germany, (2003).

Müldner, T., Leighton, G. and Miziołek, J.K. Using Multi-Encryption to Provide Secure and Controlled Access to XML Documents. *Extreme Markup Languages 2006*, (2006), Montreal, Canada.

Osborn, S., Sandhu, R., Munawer, Q. Configuring Role-Based Access Control to Enforce Mandatory and Discretionary Access Control Policies. *ACM Trans. on Information and System Security*, 3:2, (2000), 85–106.

Role Based Access Control. http://csrc.nist.gov/rbac/.

Wang, J. and Osborn, AS. A role-based approach to access control for XML databases. *Proceedings of the ninth ACM symposium on Access control models and technologies* Yorktown Heights, US (2004): 70 – 77.

W3C XML Encryption http://w3.org/Encryption/2001.

XML Path Language. http://www.w3.org/TR/xpath.

XML Schema http://www.w3.org/TR/xmlschema-0/