Cryptographically Enforced Conditional Access for XML

Gerome Miklau Dan Suciu
University of Washington
{gerome, suciu}@cs.washington.edu

Abstract

Access control for databases is typically enforced by a trusted server responsible for permitting or denying users access to the database. This server-based protection model is increasingly becoming inconvenient for web based applications. We propose encryption techniques that allow XML documents to be distributed over the web to clients for local processing while maintaining certain access controls. In particular, we focus on conditional access controls, where a user is granted access to certain data elements conditioned on the user's existing knowledge of another part of the data. We believe such access controls are important in practice, and that enforcing them cryptographically on remote instances allows for more efficient data dissemination and processing.

1 Introduction

An access control model is used to permit or refuse access by subjects to data objects. Subjects are users, or groups of users usually defined by name, network identification or other static affiliation. For XML, objects are documents or parts of documents defined by XPath expressions. Access control in relational database systems, and most proposed XML systems, is enforced by a server that handles all data requests and strictly controls which users can access what data. While this model is sometimes also used in Web applications, it is too restrictive for a large class of applications. As the following examples show, there are a number of advantages to delivering remote copies of the data to clients if access control can be maintained:

Data processing flexibility A credit card company does not make its accounts database public. However, it is willing to provide the available credit amount to a vendor that presents a correct account number *and* a correct expiration date. The credit card company could benefit from making the entire database available to the vendor since the vendor could join it with their own customer database, mine the integrated data, and pay for the service. No information would be leaked if vendors only accessed accounts for which they know the account number and expiration date, but the credit card company clearly cannot allow unrestricted usage of the entire database.

Privacy A vendor provides valuable data for a fee, and therefore requires authentication from users to confirm that they are paying customers. However a client's queries may reveal private information they prefer not to reveal to the vendor[1]. Remote enforcement of access control will allow the vendor to deliver a copy of the data so that clients can process queries locally with privacy guaranteed.

Offline browsing A Web vendor does not make its products database available to the public, but, of course, allows users to browse and query for specific products. The vendor would benefit from making the database available publicly since users could download the data to their laptop and shop during a plane trip.

We propose an approach for publishing XML data on the Web while controlling how the data is accessed. In particular, we propose a novel and flexible language of *conditional access rules* used to define security policy. We explain how to encrypt XML data to enforce these access controls without a trusted server, and we discuss query processing over these encrypted instances.

Our notion of conditional access generalizes the static categorization of subjects into authorization classes. Subjects are not identified by user name or network identifier, but by the state of their knowledge. As a special case, access may be conditioned on knowledge of a private key or password in a conventional way. But more generally subjects qualify for access to an object by virtue of their knowledge of the data. In this case conditional access rules specify what data values need to be presented by the subject before granting access to other data values. Subject authorization is therefore flexible and dynamic in a way not possible with conventional access

[1]Private information retrieval, or the problem of allowing a subject to query a remote database without revealing information through his/her queries, was first addressed in [5].

classes. The flexibility of our conditional access rules distinguish our work from other attempts to encrypt data and manage decryption keys [2, 15].

Trusting the encryption mechanism to enforce access controls on remote replicas is a substantial departure from server-based enforcement. Some applications surely require a higher level of trust in the security mechanism and will need to rely on the server model. On the other hand, there are many applications where releasing proprietary data to customers and partners is very beneficial. Our approach targets these applications, and creates opportunities for free data dissemination and new processing models.

The paper is organized as follows. In section 2 we review encryption primitives and present a simple table encryption scheme which is the basis of our document encryption method. Section 3 presents *conditional access rules* that describe access controls in our formalism. In Section 4 we show how to generate an encrypted XML instance enforcing a set of conditional access rules, and then briefly discuss querying such an instance in Section 5. Overall security is reviewed in Section 6. We address related work and conclude in Sections 7 and 8.

2 Background

We present here known techniques for encrypting a relational table to enforce certain access controls. Later we adapt table encryption to enforce conditional access to XML data. We begin with definitions of encryption schemes and one-way functions following [14].

Encryption Primitives Let $\mathcal{M}$ be the message space of plain text strings and $\mathcal{C}$ the cipher text space. A *symmetric encryption scheme* consists (for each key k) of an encryption function $E_k : \mathcal{M} \to \mathcal{C}$ and a corresponding decryption function $D_k : \mathcal{C} \to \mathcal{M}$ with the property that for all $m \in \mathcal{M}$, $D_k(E_k(m)) = m$. An encryption scheme is secure if it is computationally infeasible to deduce m from $E_k(m)$ without knowledge of k. Generally, D and E are publicly known, and the security of encryption rests in the key. A *one-way hash function* is a function $h : \mathcal{M} \to \mathcal{C}$ such that given $m \in \mathcal{M}$, it is easy to compute $h(m)$, but given $c \in \mathcal{C}$, it is computationally infeasible to find an m such that $c = h(m)$. Several symmetric encryption algorithms and candidate one-way hash functions are mentioned in [16]. DES is an fast and well-tested option for E and D, while MD5 and SHA are good options for h.

Restricting Access to Relational Tables Suppose Alice has a binary relation $T[A, B]$. She wants to publish T, but also wants to restrict access so that a user needs to present an A value before being allowed to retrieve corresponding B values. In other words, she wants to allow T to be used only in the context $\sigma_{A=a}(T)$, for some constant a. We denote this access control rule, r, as $T : (A \to B)$. The solution to this problem is described in [11], expanding on a method first proposed by Needham. Using the encryption primitives described above, the *access controlled* table (with respect to rule r) is T_r^{ac}, defined as:

$$T_r^{ac} = \{(\ h(a), E_a(b)\ ) \mid (a, b) \in T\}$$

Alice publishes T_r^{ac} (along with h and D) instead of T. The tuples of T_r^{ac} consist only of cipher text. Any subject can use T_r^{ac} in order to access T in the intended manner. For example, consider the following datalog query over table T:

$$Q_1(x) \quad :- \quad T(\text{“abc”}, x)$$

This query accesses the data in the way it was intended, and it can be rewritten for T_r^{ac} as the following rule in datalog extended with encryption primitives:

$$Q_1'(x) \quad :- \quad T_r^{ac}(h(\text{“abc”}), v), x = D_{\text{“}abc\text{''}}(v)$$

The re-written query hashes “abc”, retrieves all corresponding B values v from T_r^{ac}, and then decrypts them using “abc” as the key. For a more complex example, consider Q_2 below and its re-writing Q_2':

$$\begin{aligned} Q_2(x) \quad &:- \quad T(\text{“abc”}, y), T(y, z), T(z, x) \\ Q_2'(x) \quad &:- \quad T_r^{ac}(h(\text{“abc”}), v), y = D_{\text{“}abc\text{''}}(v), T_r^{ac}(h(y), u), z = D_y(u), T_r^{ac}(h(z), w), x = D_z(w) \end{aligned}$$

Q_2 still accesses the data in the way it was intended: the user first evaluates $T(\text{“abc”}, y)$, to get a set of values for y, then uses each of them as a constant to evaluate $T(y, z)$, then uses each resulting z to evaluate $T(z, x)$. The rewritten query Q_2' shows that this is possible to express on T_r^{ac}, by a judicious sequence of applications of h and D.

On the other hand it is difficult for an adversary, Mallory, to use T_r^{ac} in order to execute on T queries that use different access patterns. For example the following queries are hard to compute without knowing values for x:

$$\begin{aligned} Q_3(x, y) &\ :-\ \ T(x, y) \\ Q_4(x) &\ :-\ \ T(x, \text{“abc”}) \end{aligned}$$

There are two known attacks to this scheme: the dictionary attack, and the guessing attack. The first applies when Mallory can generate the entire message space of h which is determined by the domain of attribute A. For instance, when A represents credit card accounts, then Mallory can enumerate all account numbers on a powerful computer and invert h by brute force. The usual protection against this attack is to increase the size of the message space: take A to be the account number *and* the expiration date. The second attack is the guessing attack: Mallory can simply probe random account numbers and dates, and discover some of the x's with some probability. We discuss the security of encryption further in Section 6.

3 Conditional Access Rules for XML

We describe a flexible language of rules defining conditional access to an XML document, hence *conditional access rules* (CARs). For a path expression P, and an XML tree instance D, we denote by $eval(P, D)$ the node set resulting from evaluating P starting from the root. A CAR r has the following form:

$$r ::= C : (\{\bar{B}\} \to \{\bar{F}\})$$

Here C is a single XPath expression specifying a *context*, $\bar{B}$ and $\bar{F}$ are sets of XPath expression defining, respectively, a set of required *bindings*, and a set of retrieved, or *free values*. C is an absolute XPath expression (i.e. it starts from the document root); the expressions in $\bar{B}$ and $\bar{F}$ are relative, starting from a context node $c \in eval(C, D)$. Users specify several rules to control access to an XML document.

The intuition behind a rule is that the subtree rooted at a context node $c \in eval(C, D)$ will be hidden from the user: access is allowed only by specifying values for $\bar{B}$ (a tuple of values), in which case corresponding subtree(s) $\bar{F}$ will be accessible. All XPath expressions in B are required to return atomic values, i.e. they have to end in `text()` or `data()` or in an attribute, and we assume the set $\{\bar{B}\}$ to be ordered. Figure **??** illustrates the effect of a CAR.

Semantics Let $\bar{R}$ be a set of conditional access rules, and let D be an XML document tree. We denote with $x \preceq y$ the fact that node y is a descendant-or-self of node x. In a related manner we define the "allowed" descendants to be those descendants that are not beneath any other contained context node of another rule: allowed-desc$(z) = \{x \mid z \preceq x, \neg(\exists r = c'(\bar{B}' \to \bar{F}') \in \bar{R}, y \in eval(c', D), z \prec y \preceq x)\}$. We also identify a leaf node with its value.

The semantics for CARs specifies a function $access_{\bar{R}} : \mathcal{P}(Dom) \to \mathcal{P}(nodes(D))$, where $\mathcal{P}(S)$ denotes the set of finite subsets of S, with the following meaning: if the user provides values $v_1, \ldots, v_k$, then the user is allowed to see node x if $x \in access_{\bar{R}}(\{v_1, \ldots, v_k\})$. Then $access_{\bar{R}}$ is defined below:

$$\begin{aligned} access_{\bar{R}}^{(0)}(V) &= V \cup \text{allowed-desc}(D_{root}) \\ access_{\bar{R}}^{(n+1)}(V) &= access_{\bar{R}}^{(n)}(V) \ \cup \\ &\quad \{x \mid x \in \text{allowed-desc}(z),\ z \in eval(c[B_1 = v_1, \ldots, B_k = v_k]/f_j, D), \\ &\quad\ c : (\{B_1, \ldots, B_k\} \to \{\ldots F_j \ldots\}) \in \bar{R},\ v_1, \ldots, v_k \in access_{\bar{R}}^{(n)}(V)\ \} \\ access_{\bar{R}}(V) &= \bigcup_{n \geq 0} access_{\bar{R}}^{(n)}(V) \end{aligned}$$

The function $access_{\bar{R}}^{(n)}(V)$ computes the nodes that the user can obtain after n probes to the encrypted data: to compute some node x in $access_{\bar{R}}^{(n+1)}(V)$ the user starts by picking some values $v_1, \ldots, v_k$ already in $access_{\bar{R}}^{(n)}$ (recall that we blur the distinction between leaf nodes and their values), uses them to bind the expressions $B_1, \ldots, B_k$ in the context, then picks some node z in the result of some F_j: the node x is any descendant of z, except if it is hidden from the user because some other context c' disallows access to it. Notice that $access_{\bar{R}}^{(0)}$ already includes all nodes in the document that are not explicitly hidden by a context.

We now provide a number of examples, based on the data pictured in Figure 1 to illustrate the flexibility of conditional access rules:

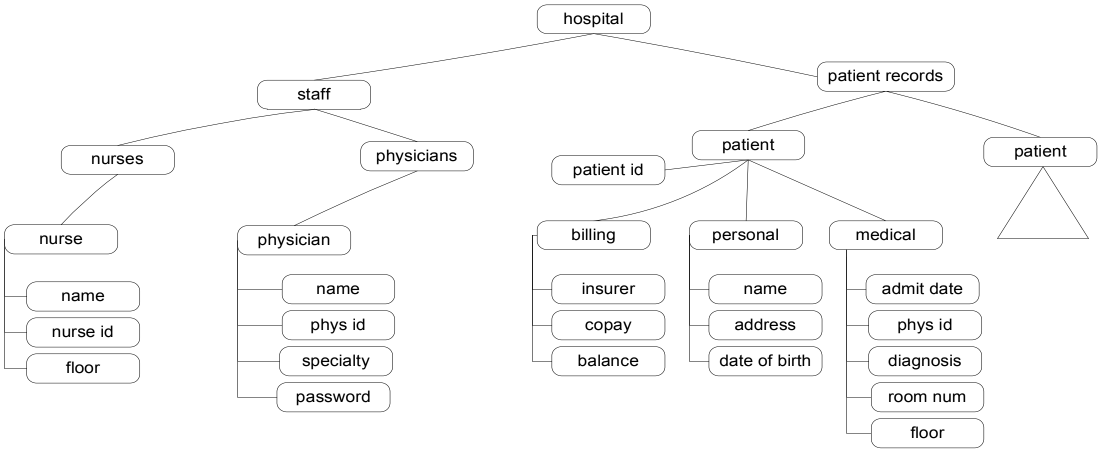

Figure 1: Tree view of XML data describing hospital staff and patients (data values omitted).

Example 3.1 Consider the set of CARs $\bar{R}$ containing rules:

$$\begin{array}{rcl} /hospital/patientrecords/patient & : & (\{patientid\} \rightarrow \{personal/name, personal/address\}) \\ /hospital/patientrecords/patient & : & (\{personal/name\} \rightarrow \{medical/room, medical/floor\}) \end{array}$$

The first rule says that within each patient context, name and address are accessible conditioned on knowledge of the corresponding patientid. The second rule says that within each patient context, the room and floor are accessible conditioned on knowledge of the corresponding patientid. Together these imply that a patientid is sufficient to access name, address, room, and floor. All data not within a patient context is accessible.

Example 3.2 Suppose the set of CARs $\bar{R}$ contains just the rule $/ : (\{\} \rightarrow \{.\})$. Here $C = /$, so D_{root} is the only context node for this set of CARs. B is the empty-set, which requires no knowledge for access to $F = \texttt{self}$ evaluated from the context, which is D_{root}. $access_{\bar{R}}(\{\})$ contains all descendants of D_{root} since there are no other contexts to consider. Thus this rule makes the entire document accessible with no conditions.

Example 3.3 Consider the set of CARs $\bar{R}$ containing rules:

$$\begin{array}{rcl} / & : & (\{\} \rightarrow \{.\}) \\ /hospital/patientrecords/patient & : & (\{patientid, medical/diagnosis\} \rightarrow \{.\}) \\ //password & : & (\{\} \rightarrow \{\}) \end{array}$$

These rules make the entire XML file accessible except (1) access to patient data is conditioned on knowing a patient's id and diagnosis, and (2) no access is given to any passwords. The last rule makes every password element a context, within which the set of free values is empty, thus protecting all passwords, regardless of other CARs in $\bar{R}$.

Example 3.4 Consider the set of CARs $\bar{R}$ containing the single rule:

$$\begin{array}{rcl} /hospital & : & (\, staff/physicians/physician/\{name, password, physid = 1234\} \rightarrow \\ & & \{patientrecords/patient[medical/physid = 1234]\}\,) \end{array}$$

This example shows how conventional password-authenticated access can be expressed using CARs when a password, private key, or other identifying information is included in the data. When such private information occurs in the data as a bound values, it will be automatically encrypted. The rule says that given a physician name, password, and id, the entire patient element is accessible for patients under the physician's care. A full treatment of conditional access rules would include a shorthand for describing the set of such rules for each physician id.

4 Encrypting an XML Document

Given an XML document D and a set $\bar{R}$ of conditional access rules, we generate an encrypted document $D^{ac}_{\bar{R}}$ that enforces access according to $\bar{R}$. Our strategy is to apply table encryption, as described in Section 2, to binary tables that are implied by each CAR within a context. If $r = C : (\bar{B} \to \bar{F})$ is a CAR in $\bar{R}$, and x is a context node in $eval(C, D)$, then let $T_r(x)$ be the binary table which pairs the concatenation of bound data values with the concatenation of free data values. That is, $T_r(x)$ is defined by:

$$\{(b, f) \mid b = b_1.b_2 \ldots b_k, f = f_1.f_2 \ldots f_l, b_i \in eval(C/B_i, D), i \in [1..k], f_i \in eval(C/F_j, D), j \in [1..l]\}$$

To construct $D^{ac}_{\bar{R}}$ we begin with D. Nodes that are not a context for any rule appear unchanged in $D^{ac}_{\bar{R}}$. But for other nodes we consider each rule r for which x is a context, compute $T_r(x)$ and replace x and its subtree with the collection of tables $T_r(x)$ for each $r \in \bar{R}$. The tables are represented as XML in a standard way as a list of row elements each containing two column elements (since the $T_r(x)$ are binary). These replaced elements also need to contain some metadata describing the paths to the encoded bound and free values. A similar construction can be recursively applied to nested contexts; we omit the details from this abstract.

Nevertheless, this is a rather naive encryption scheme. The resulting $D^{ac}_{\bar{R}}$ may be large compared with D, and grows with the number of conditional access rules in $\bar{R}$. Furthermore, there is an opportunity for optimizing the set of CARs to produce smaller encrypted instances. Given a set of CARs $\bar{R}$, we would like to find an equivalent[2] set $\bar{R}'$ that is minimal.

Surprisingly, we can use the theory of functional dependencies to aid minimization of a set of access rules, at least in the case of relational data. For example, given a ternary table $T[A, B, C]$ with the set of relational access rules (as in Section 2) $T : (A \to B), T : (B \to C), T : (A \to C)$ the naive encrypted version will represent consist of three binary relations, encoding $\Pi_{AB}(R), \Pi_{BC}(R), \Pi_{AC}(R)$. The first two relations, however, are sufficient. This happens because the given set of access rules is equivalent to the following: $T : (A \to B), T : (B \to C)$. For a set of relational access rules $\bar{S}$, denote $\bar{S}^{fd}$ the functional dependencies obtained from interpreting $\to$ as being a functional dependency rather than access control. For relational schema, $T : A \to B$ has the standard meaning: A functionally determines B in T.

Theorem 4.1 *Two sets of relational access rules $\bar{S}_1, \bar{S}_2$ are equivalent iff $\bar{S}_1^{fd}$ is equivalent to $\bar{S}_2^{fd}$.*

For XML, we can interpret a CAR $C : (B \to F)$ as a functional dependency according to the definition given in [3]. However, the above theorem does not hold for XML documents. We leave the interesting connection between functional dependencies and conditional access rules, along with the general CAR minimization problem, as a compelling direction for future work.

5 Query Processing for Encrypted XML

Since an encrypted, access-controlled document D^{ac} may contain large portions of the original document unencrypted, it makes sense to use existing tools and technologies to query D^{ac}. We consider XQuery[4] a suitable general-purpose query language. For a query q and an XML document D we denote by $q(D)$ the answer of q evaluated on D. The query rewriting problem is to find a new query q' such that $q(D) = q'(D^{ac})$. The new query q' will consist of XQuery syntax augmented with operators for the hash function h and the decryption function D needed to process the data. In particular, the parts of the query that attempt to access protected contexts need to be replaced with routines that extract data from the encrypted tables represented as XML beneath the context. These routines are closely related to the rewriting examples provided in Section 2. A full treatment of the re-writing problem is omitted from this abstract.

6 Security

As we mentioned in Section 2, there are two explicit attacks on table encryption (dictionary and guessing), and these vulnerabilities are inherited by our encrypted instances. Both attacks depend on the size of the message space, determined by the product of the domain sizes of the bound attributes in a CAR. The main protection against a guessing attack is to decrease the probability of success by increasing the domain. It is worth noting that a secure server allowing similar access is also vulnerable to a guessing attack, however this can be dealt with by delaying the response between unsuccessful probes. In a similar manner, the computation speed of the hash

[2]Two CAR sets $\bar{R}_1, \bar{R}_2$ are equivalent if for any XML document D, $access_{R_1}(D) = access_{R_2}(D)$.

function will slow down probes to the encrypted instance but also impede query processing. Other strengthening possibilities include "salting" the input to the hash function to discourage dictionary attacks, the use of private keys in the data, and keyed hash functions.

7 Related Work

There are a number of recently proposed access control models for XML. In [6, 7, 12] XPath expressions identify fine-grained authorization objects and a server negotiates access and delivers a pruned document. A similar framework is proposed in [2] and then extended to offer remote enforcement by encrypting elements and transferring encrypted keys to users. Our work generalizes this by making access conditioned on encrypted keys, or on other knowledge of the data. The authors of [13] describe a model where provisional access policies can be defined. Such a policy grants a subject access to data provided a further action or qualification is satisfied (e.g., a subject's activity may be logged, or a password is provided). Their provisions are related to our notion of conditional access control although they work within the framework of a trusted security processor. A framework for element-level encryption of XML documents has been proposed by the W3C [9]. The IBM Security Suite[1] implements the W3C XML encryption recommendation.

8 Conclusions and Future Work

We have proposed techniques for enforcing access control without relying on a secure server, with many practical applications. Removing the server from access control enables offline query data processing, anonymous evaluation of sensitive queries, and new data sharing possibilities. In addition, remotely controlled access can still be useful in a setting where data is served to clients: a data provider who must contract for server space may provide an encrypted instance to an untrusted contractor, with assurance that data will be accessed in a controlled manner. Further, our conditional access rules would make it easy for individuals to publish sensitive data in the absence of sophisticated security systems. Finally, we identify the following key problems that require further attention: (1) minimizing a set of conditional access rules to improve the efficiency of encryption and aid users in policy definition; (2) re-writing queries over encrypted instances; and (3) quantifying the security of encrypted instances in terms of the attribute domains and properties of the encryption primitives.

References

[1] AlphaWorks. Xml security suite. www.alphaworks.ibm.com/tech/xmlsecuritysuite, November 2001.

[2] E. Bertino, S. Castano, and E. Ferrari. Securing xml documents with author-x. *IEEE Internet Computing*, May/June 2001.

[3] P. Buneman, S. Davidson, W. Fan, C. Hara, and W. Tan. Reasoning about keys for xml. *Workshop of Databases and Programming Languages (DBLP 2001)*, 2001.

[4] D. Chamberlin, J. Clark, D. Florescu, J. Robie, J. Simeon, and M. Stefanascu. XQuery 1.0: An XML query language. http://www.w3.org/TR/xquery/, 07 June 2001. W3C working draft.

[5] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. Private information retrieval. In *Foundations of Computer Science (FOCS)*, 1995.

[6] E. Damiani, S. D. C. di Vimercati, S. Paraboschi, and P. Samarati. Controlling access to xml documents. *IEEE Internet Computing*, 5(6):18–28, November/December 2001.

[7] E. Damiani, S. D. C. di Vimercati, S. Paraboschi, and P. Samarati. A fine-grained access control system for xml documents. *ACM Transactions on Information and System Security (TISSEC)*, 2002. To appear.

[8] D. Denning. *Cryptography and Data Security.* Addison-Wesley, 1982.

[9] D. Eastlake and J. Reagle. Xml encryption syntax and processing. http://www.w3.org/TR/xmlenc-core, 4 March 2002. World Wide Web Consortium (W3C) Working Draft.

[10] J. Feigenbaum, E. Grosse, and J. A. Reeds. Cryptographic protection of membership lists. *Newsletter of the International Association for Cryptologic Research*, 9(1):16–20, 1992.

[11] J. Feigenbaum, M. Y. Liberman, and R. N. Wright. Cryptographic protection of databases and software. In *Distributed Computing and Cryptography*, pages 161–172, 1991.

[12] A. Gabillon and E. Bruno. Regulating access to xml documents. *Proc. of the 15th Annual IFIP WG 11.3 Working Conference on Database and Application Security*, July 2001.

[13] M. Kudo and S. Hada. Xml document security based on provisional authorization. *Computer and Communication Security (CCS)*, November 2000.

[14] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography.* CRC Press, 1997.

[15] J. Reagle. Xml encryption requirements. http://www.w3.org/TR/xml-encryption-req, 4 March 2002. World Wide Web Consortium (W3C) Note.

[16] B. Schneier. *Applied Cryptography, Second Edition.* John Wiley and Sons, Inc., 1996.