基于客户端对 XML 文档的访问控制管理

路长胜　洪晓光
（山东大学计算机科学与技术学院　济南 250061）

摘　要　对 XML 文档的访问控制管理的研究目前是个热点问题，已有诸多文献提出了许多访问控制模型，它们大多是以静态的方式共享加密数据。然而在某些情况下，会出现临时的、动态的访问控制规则，这样静态模型就不能解决这种情况。随着客户端设备软硬件的发展，出现了基于客户端的动态访问控制方案。本文提出了一个结合静态模型和动态模型的新的混合方案，举例介绍了基于客户端的动态访问控制部分，并与纯动态方案在性能上作了比较。

关键词　访问控制，静态访问，动态访问，混合访问，跳跃索引

Client-Based Access Control Management for XML Documents

LU Chang-Sheng　HONG Xiao-Guang
(School of Compute Science and Technology, Shandong University, Jinan 250061)

Abstract　Access control for XML documents is a non-trivial topic, as can be witnessed from the number of approaches presented in the literature. Several encryption schemes can be used to serve this purpose but all suffer from a static way of sharing data. Unfortunately, there are many situations where access control rules are user specific, dynamic and then difficult to predict. However, models compiling access control policies in the data encryption cannot tackle this dynamicity. With the emergence of hardware and software security elements on client devices, dynamic client-based access control schemes can be devised. This paper proposes a new hybrid solution combining static and dynamic solutions. We compare it with dynamic solution.

Keywords　Access control, Static access control, Dynamic access control, Hybrid access control, Skip index

1　背景

随着 Internet/WWW 的应用日渐广泛和深入，促进了数据库和 Internet/WWW 两种技术的结合，使 XML 文档已成为事实上的数据交换标准[1]。目前针对 XML 文档的访问控制管理的研究是个热点问题。访问控制管理是数据库系统管理的基石之一，并且传统上在服务器端执行，但这个地方的数据安全性和保存数据的机密性，不再受用户的信任[4]，致使访问控制从服务器端向客户端转移。由于原先的客户端设备的安全性不值得信任，所有基于客户端的访问控制解决方案都依赖于数据加密。这些加密的数据保存在服务器上，客户根据自己拥有的密钥访问他们被授权的部分。根据不同的应用环境，人们设计出了这个基本模型的不同变种[3]，这些模型的共同点是通过以静态的方式共享数据来把对客户端的信任需求降到最低。并且，不管多大的共享粒度，数据集分裂成子集，反映了当前的共享情况，每一个子集用不同的或合成的密钥加密。这样访问控制规则被预加密。一旦数据集被加密，规则定义的改变会影响到子集的边界，因此会导致对部分子集的重新加密和可能重新分发密钥。

然而，有些情况访问控制规则是用户具体的、动态的并且难以预料的。例如，医疗信息的交换传统上规定严格的共享策略来保护病人隐私，但是这些规定在一些特殊情况下会有例外（如紧急情况），也可能是由于时间变化（如依赖于病人的治疗方案）并且服从于临时的授权，会产生一些动态的访问控制规则。这样在加密数据中编译访问控制策略的静态模型不能解决这种动态性（例外情况）。随着客户端设备软硬件安全因素的出现，SOE（安全操作环境）在客户端变成了现实[3]，出现了基于客户端的动态访问控制方案。

考虑到现实生活中，一个事物相对稳定的属性是主要的（否则它就不是原先那个事物了），动态变化的属性（临时的，例外的）总是占少数的。如上面的例子中，按正常的已制定好的静态共享策略进行的访问占到总访问量的 85%，而例外的访问只占 15%。再如，当今 Internet 网上的内容五花八门，有些内容是不宜于未成年儿童访问的，家长或老师为保护儿童，对他们可访问的内容要做出一些限制。由于不同的家长或老师对要限制的内容的敏感性不同，他们给出的限制规则是不可能一样的，但形成共识的限制是主要的（如只宜成人访问的网站），这类限制可占总限制数的 80%，而每个家长或老师根据自己的敏感性不同给出的不在共识之内的限制数（如某些游戏网站或娱乐网站）只占总限制数的 20%。受内存管理模型的启发，采用以空间换时间的策略[2]。本文中，我们设计了一个基于客户端的混合访问控制方案：把静态模式和动态模式二者结合起来，将访问控制分成两步：1）对一访问控制规则，首先检查是不是属于已有的静态访问控制策略，如果是，则采用静态模型直接到服务器上读取已做好的授权视图。2）如果不属于已有的静态访问控制策略，再采用动态分析方式，将 XML 文档读入到客户端，在客户端进行动态规则的分析，找出授权（需要）部分。由于静态访问的效率要远远高于动态访问，因此我们的混合模型的效率要比纯动态模型的效率高出很多。

路长胜　研究生，研究方向：数据库应用。

本文的第2部分给出了我们提出的混合访问控制方案的目标结构，并用动机例子来说明。第3部分，通过一个例子描述了在客户端进行动态规则分析，并使用了高效的跳跃索引结构。第4部分将我们的方案与纯动态分析方案进行了比较。最后是总结。

2 混合访问控制模型

2.1 动机例子

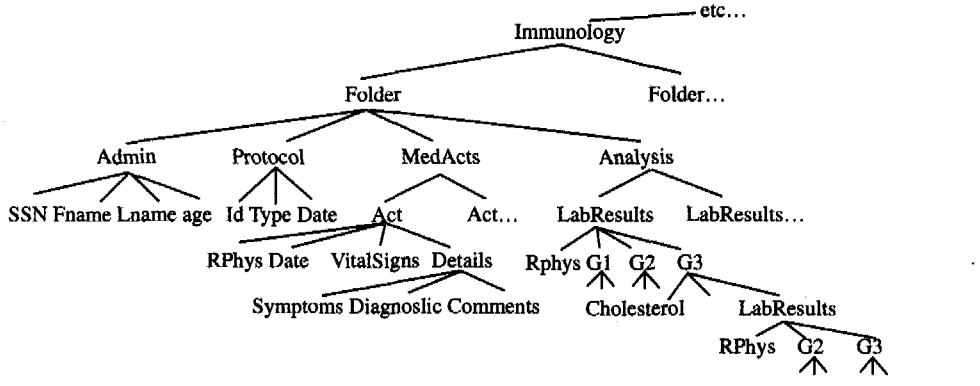

图1

我们用一个表示医学文卷的XML文档来说明混合访问控制模型的语义并作为动机例子。图1画出了文档例子，连同三种用户的静态访问控制策略(用Xpath表达式表示)，秘书、医生、医学研究员(见图1)。秘书：只能访问病人的管理文卷部分。医生：可以访问病人的管理文卷和他的病人的所有医疗行为和分析，不是他做的医疗行动的细节除外。研究员：只准访问鉴署测试协议且协议类型为G3的病人的年龄和实验结果，且胆固醇含量不超过250mg/dl。对这些静态控制策略的访问是经常的，为了提高效率，采用以空间换时间的策略，我们把这些静态访问控制策略的授权视图事先做好，经加密后放在服务器上，随用随取。

医生的静态访问控制策略
D1：+，//Folder/Admin
D2：+，//MedActs[//Rphys=user]
D3：−，//Act[Rphys! =user]/Details
D4：+，//Folder[MedActs//Rphys=user]/Analysis
研究员的静态访问控制策略
R1：+，//Folder[Protocol/Type=G3]//Age
R2：+，//Folder[Protocol/Type=G3]//LabResults//G3
R3：−，//G3[Cholesterol>250]
{G1…G10}10组中每组都有一对规则R2和R3
秘书的静态访问控制策略
S1：+，//Admin
研究员的动态访问控制
R4：+，// Folder[//Cholesterol>300]

图2 访问控制策略

医学应用可作为有时需要动态规则的例子。例如：一个研究员可以例外地并有时间限制地准予访问所有医疗文卷的片段，查阅胆固醇超过300mg/dl的比例。对这种动态性的(例外的、临时的)访问控制规则需要拿到客户端进行分析。

2.2 目标结构

图3给出了动机例子的目标结构的抽象表示。DSP提供服务器存储客户加密的XML文档以及加密的静态访问控制策略的视图，数据加密是为了保证数据在服务器端的安全性和私密性。用户在客户端提出访问请求，经过转换提交到服务器端[4]，由服务器判断访问控制规则是否属于静态访问控制策略，如果是，直接返回已做好的授权视图，在客户端完成解密和后续查询。如果不是(是动态访问控制规则)，则返回XML文档，在客户端完成解密、完整性检查、访问控制规则分析及查询。访问控制策略和解密需要的密钥能永久地被SOE经由安全渠道从不同的源头更新或下载而获得(受信任的第三方、安全服务器，父母或老师等)。

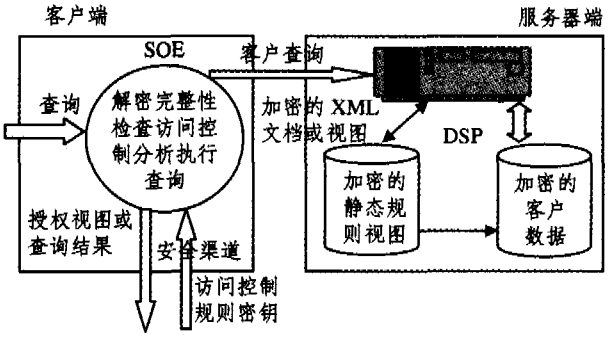

图3 混合模型目标结构

2.3 准备工作

访问控制规则简称访问规则(或规则)，用三元组(sign，subject，object)表示。sign：符号，表示读操作允许(+)或禁止(−)。subject：是主题。object：对应XML文档中的元素或子树，用XPath表达式标识。

本文中，我们考虑了XPath的一个子集，用XP{[ ]，*，//}表示。这个子集在实践中广泛应用，包含结点测试，孩子轴(/)，后离轴(//)，通配符(*)和谓词([ ])。冲突解决使用两种策略：否定优先(Denial-Takes-precedence)和最具体对象优先(most-specific-object-Takes-precedence)。

给定文档上与给定主题相关的规则集称为访问控制策略。这种策略定义了文档的授权视图，依赖于具体应用，这个视图是可查询的。查询结果是从查询文档的授权视图中计算得到的。

用一个基于事件的解析器(如SAX)，对输入文档的每一opening，text和closing tag分别产生open，value和close事件。用非确定有穷自动机(NFA)表示每一条访问规则，称为访问规则自动机ARA(Access Rules Automata)。○圆圈表示一个状态，◎双圆圈表示终结状态，用唯一的状态号标识。有向边表示转换，用事件标签标记。为了标记后裔轴(//)，添加一个自转换，用*匹配任意open事件。一个ARA包含一个导航路径和可选的一个或几个谓词路径(图7中灰色表示)。为了管理表示给定访问控制策略的ARA集，我们介绍下面的数据结构。

令牌和令牌栈：对 ARA 上的路径每一转换，创建一个令牌(token)，区别导航令牌 NT(navigational token)和谓词令牌 PT(predicate token)。令牌栈 TS(Token stack)用来记住转进程，栈顶包含所有活动的 NT 和 PT 令牌，这些活动令牌在下一个事件到来时可触发一个新的转换。通过触发转换创建一个新的令牌并进栈，每关闭一个事件出栈。

规则状态和授权栈 AS(Authorization Stack)，规则状态有活动的和未知的两种。授权栈 AS 登记那些已经到达导航路径终结态的规则状态。栈中规则的状态有四种：正活动(+)，正未知(+?)，负活动(-)，负未知(-?)。栈底默认是负活动(-)。

考虑后裔轴(//)，在同一文档的不同深度可遇到同一个元素名，在初始谓词创建的时候，用深度标记导航令牌和谓词令牌。一个令牌必须包含四方面的信息：RuleID(R，S，…..)，导航/谓词状态(n 或 p)，状态号和深度。

谓词集合 PS(Predicate Set)：记录 PT 令牌已经到达的谓词路径的终结状态。

为加快对文档的分析，通过对文档的结构建立一高度简洁的跳跃索引(图 4)，用递归的方式记录每一元素的父标签字典，每一元素的后裔标签集，子树大小，及对元素进行编码，并以流方式嵌入文档，便于分析。RemainingLabels(t)函数，用来决定令牌 t 在它的 ARA 中从当前状态转换到终结状态过程中遇到的标签集。

分析筛选原理：每到来一个事件，对每一规则按它的 ARA 产生令牌并进栈，对栈顶每一个活动令牌 t，通过调用函数 RemainingLabels(t)来判断，如果其结果不包含在当前元素的后裔标签集中，说明令牌 t 的 ARA 在该子树中不能到达终结状态，不必继续分析，可从 AS 栈顶删除，若栈顶的令牌全删除了，说明以当前元素为根的子树是未授权的或无关的，分析时可跳过，从而加快分析进程。

3 SOE 上的动态访问控制分析

下面用一个抽象的例子(如图 4、5 所示，给出了编码文档及数据结构)介绍在客户端进行动态访问控制分析的过程。

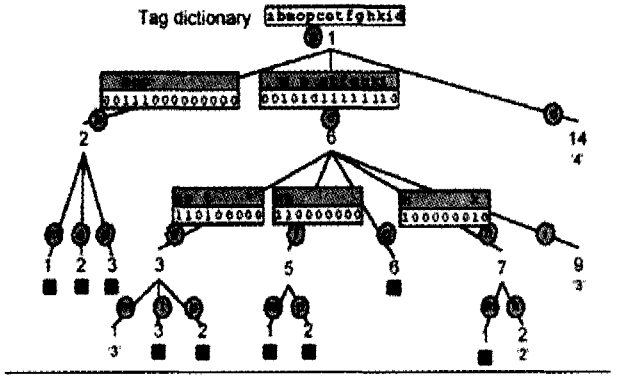

图 4 编码的 XML 文档

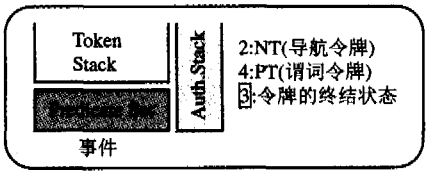

图 5 数据结构

图 6 给出了分析的过程。

初始化：初始状态 1，6，11，16 进 TS 栈，AS 栈中默认状态为'一'。

1)事件 a 触发产生令牌 Rn21 和 Rp41(图中简记为 2 和 4，下文同)进栈，此时不允许筛选。

2)事件 b 只触发自转换，这时栈顶的令牌可筛选。调用函数 RemainingLabels(6)={c，e，m}不真包含在当前元素 b 的后裔标签集{m，o，p}中，令牌 6 可删除，令牌 11 和 12 同样也可删除，不必继续分析，这样栈顶为空，没有活动令牌，相当于关闭事件 b(第三步/b)，即跳过了以 b 为根的子树。

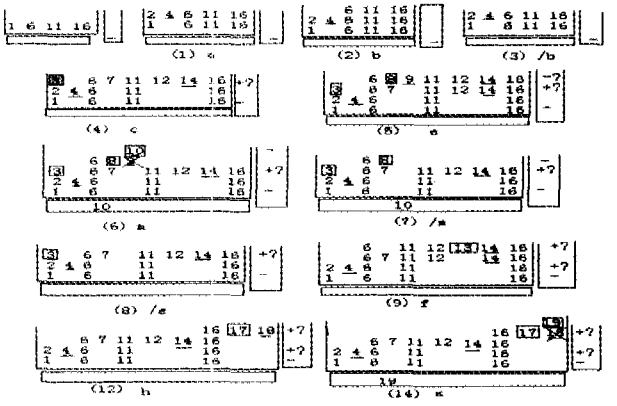

图 6 分析过程

4)遇到事件 c，规则 R 中令牌 2 到达终结状态 3，由于谓词 d=4，还没到达，现在规则 R 的状态是正未知的(+?)进 AS 栈，元素 C 还不能输出，等待谓词条件为真。规则 S 转换到状态 7，进 TS 栈，规则 T 转换到状态 12 和 14 进 TS 栈。此时不允许筛选。

5)遇到事件 e，规则 S 中令牌 7 到达终结状态 8，并产生谓词令牌 9 还没到达终结状态，元素 e 还不能输出，等待谓词条件为真。此时规则 S 的状态为负未知的(-?)，进 AS 栈。此时可以筛选掉规则 R，T 和 U。

6)遇到事件 m，谓词令牌9到达终结状态 10，m=3，谓词为真，谓词令牌9没必要继续分析，可从 TS 栈中丢弃，此时规则 S 的状态变为已知'一'，谓词终结状态 10 进入 PS(并在第八步从 PS 中丢弃)。

7)关闭事件 m(/m)，栈顶出栈。这时栈顶活动令牌 6 可筛选掉，8 为终结态，且 AS 栈顶状态为'一'，这时以元素 e 为根结点的子树可跳过(第八步/e)。

9)遇到事件 f，规则 T 到达终结状态 13，谓词令牌 14 还没到达终结状态，此时规则 T 的状态为正未知的(+?)，进 AS 栈，元素 f 还不能输出，等待谓词条件 i=3 为真。此时栈顶活动令牌都可删除(相当于关闭事件 f，第十步)。可跳过以 f 为根结点的子树，元素 f 进入缓冲区，等待谓词条件为真时输出。

11)遇到事件 g，可跳过。

12)遇到事件 h，规则 U 到达终结状态 17，谓词令牌18还没到达终结状态，此时规则 U 的状态为正未知的(+?)，进 AS 栈，元素 h 还不能输出，等待谓词条件 k=2 为真。此时可筛选掉规则 R，S，T。

13)遇到事件 m，可跳过。

14)遇到事件 k，k=2，第 12 步谓词为真，规则 U 的状态为正，输出元素 h。

15)遇到事件 i,i=3,第 9 步谓词为真,规则 T 的状态为正,输出元素 f。

16)遇到事件 d,d=4,第 4 步谓词为真,规则 R 的状态为正,输出元素 c。

分析完毕,分析结果见图 8。

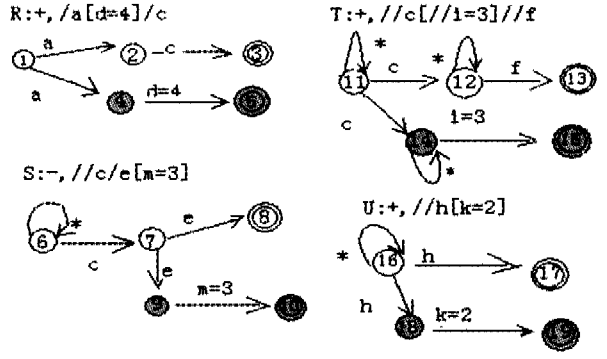

图 7 访问控制规则及 ARA

图 8 分析结果

4 分析比较

下面我们给出混合方案与纯动态方案性能比较。将静态访问控制策略的授权视图经加密存放在服务器端,对静态访问控制规则的查询将直接读取视图,免去规则分析,它的花费接近于常数(主要花费为视图的通信花费和解密花费,这二者都与视图的规模大小有关)。而动态访问控制规则的查询花费包括:加密的 XML 文档(内含索引)的通信花费和解密花费以及访问控制花费和完整性检查花费,其中通信花费和解密花费主要与文档的规模及文档的特点(文档的结构特点、是否有递归等)有关,并且与访问规则是否复杂有密切的关系,比如规则中出现的谓词在文档结束前都是未知的,则需要读入和解密整个文档。访问控制的花费(从 2%~15%)主要受解密花费(53%~60%)和通信花费(30%~38%)决定(不考虑完整性检查[3]),与规则的复杂性相关。

下面以在第 2 部分中介绍的医疗文档的秘书、医生和研究者三种访问策略为例,来比较混合模型与纯动态模型的性能。这里不考虑完整性检查,压缩的原文档为 2.5MB,已做好的三授权视图大小为秘书(135KB),医生(575KB),研究者(95KB)。

为了便于比较,我们用相同的查询分别在混合模型与纯动态模型上取得实验数据,并以纯静态模型上的实验数据为参照。

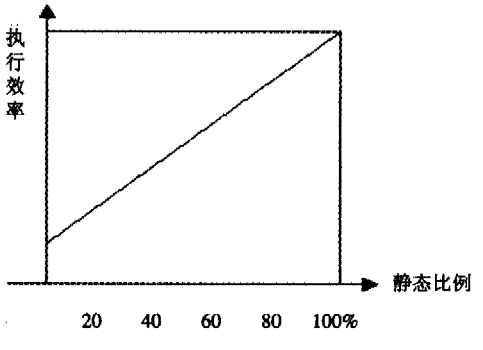

图 9

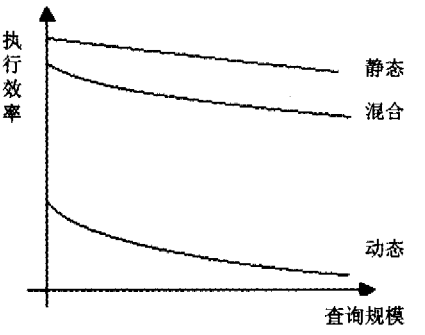

图 10

图 9 给出了在秘书的访问控制策略中,我们用同一条查询分别作为静态规则和动态规则并计算出混合模型的情况。图中 X 轴表示总访问量中静态规则占的比例。Y 轴表示每条规则的平均执行效率。由图中可以看出当混合模型中静态访问规则的比例占到 80%时的效率比纯动态模型的效率要高出很多。

据我们统计,在一些比较严格或相对固定的工作中,有 85%以上是按规定(静态规则)进行的(比如医疗,银行等),有些相对自由的地方,动态的(临时的、例外的)情况能占到总体的 20%或者以上(如前述家长对孩子上网的控制)。下面为表示简洁,我们不妨假定混合模型中静态访问占总访问数的 80%。

图 10 给出了在医生的访问控制策略中,随着查询规模的增长,两个模型的执行效率变化情况。图中 X 轴为查询规模,Y 轴为执行效率。显然,随着查询规模的增长对混合模型的执行效率的影响要比纯动态模型小很多。

图 11 给出了在研究者的访问控制策略中,两个模型的执行效率受访问规则复杂性(未知谓词情况)的影响。

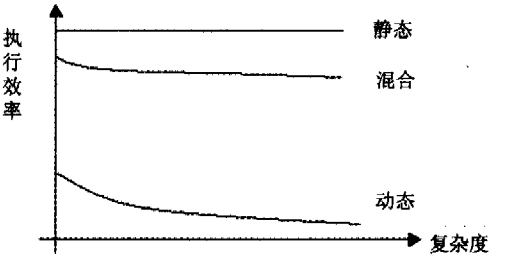

图 11

总结 基于客户端的访问控制,多是以静态的方式共享加密数据,这种方式访问效率高,而有些情况访问控制规则是用户具体的、动态的并且难以预料,从而出现了动态的访问控制,它通过跳跃索引用不大的存储花费极大地加快了规则分析。本文综合二者优点提出了一个混合模型,通过上部分的比较,很明显大大提高了访问效率。

参 考 文 献

1 万常选著. XML 数据库技术. 北京:清华大学出版社,2005

2 孟静著. 操作系统原理教程. 北京:清华大学出版社,2004

3 Bouganim L, Ngoc D. Client-Based Access Control Management for XML Documents. June 2004

4 Hakan H, Lyer B. Executing SQL over Encrypted Data in the Database-Service-Provider Model. June 2002