韓國 컴퓨터情報學會 2007하계학술발표논문집&학회지
第15卷 第1號, 2007. 6.

XML 데이터 스트림 환경에서 세분화된 접근제어 방법

안 동 찬*

Fine-Grained Access Control Method in XML Data Stream

Dong-Chan An *

요 약

다양한 사용자 및 응용 프로그램들이 XML을 기반으로 정보의 분산과 공유를 요구함에 따라 안전하고 효율적으로 XML 데이터를 접근하려는 요구가 중요한 이슈로 관심을 모으고 있다. 특히, 접근제어 규칙을 XPath로 표현함으로써 문서 단위 접근 범위의 한계를 극복하고 문서의 일부분 단위의 접근을 해결하면서 접근제어를 수행할 때 발생될 수 있는 충돌에 대한 해결책을 마련하는 안전성에 초점을 둔 연구들은 많았으나 접근제어를 수행할 때의 효율성에 초점을 둔 연구는 미비하다. 본 논문은 XML 데이터 스트림의 특성을 고려하여 롤 프라임 넘버 레이블링 기법을 이용한 세분화된 접근제어 방법을 제안한다. 본 논문에서 제안한 방법은 기존 시스템과 비교하여 구현의 용이성, 낮은 실행시간, 그리고 안전하고 정확한 질의응답을 보장한다. 또한, 이와 같은 장점들을 실험을 통해 분석한다.

▸ Keyword : XML, 보안, 접근제어, 롤 프라임 넘버

Ⅰ. 서 론

XML[1]이 인터넷 환경에서 데이터 저장과 전송을 위한 표준으로서 부각되면서 XML 데이터 기반의 정보 분산과 공유에 대한 요구가 날로 증가되고, 효율적이고 안전하게 XML 데이터에 접근할 수 있는 방법이 매우 중요한 이슈로 관심을 모으고 있다. 그럼에도 불구하고, 이와 관련된 연구들은 문서의 일부분 단위의 접근을 해결하면서 접근제어를 수행할 때 발생될 수 있는 충돌에 대한 해결책을 마련하는 안전성에만 주로 초점을 두어 왔으며 질의에 기초한 효율적인 접근제어를 수행하는 연구는 미비하다.

또한, 기존의 데이터 환경에서 접근제어 방법은 유한하고 영속적인 데이터 환경과 시스템 중심 그리고 정적인 데이터 환경에서의 접근제어 정책이 주류를 이루었다. 그러나 최근에는 유비쿼터스 환경 속에서 끊임없이 계속적으로 발생되는 데이터 스트림 환경에 맞는 데이터 접근제어 정책이 필요한 것이 사실이고 기존의 접근제어 정책으로는 이러한 요구사항을 수용할 수 없음을 알 수 있다.

본 논문은 유비쿼터스 환경에서 XML 데이터 스트림의 특성을 고려하여 롤 프라임 넘버 레이블링 기법을 이용한 세분화된 접근제어 방법을 제안한다.

• 제1저자 : 안동찬
* 안산공과대학 디지털미디어과

제안 기법은 유비쿼터스 환경의 이동단말기에서 요청된 질의를 처리함에 있어 XML 데이터 스트림의 특징을 잘 반영하여, 안전하고 정확한 실시간 질의응답이 가능하게 해준다.

본 논문은 다음과 같이 구성되어 있다. 2장에서는 관련 연구들과 그들의 단점을 간략히 설명한다. 3장에서는 본 논문의 이론적 배경과 기본적인 개념을 설명한다. 4장에서는 제안 방법에 대한 알고리즘과 기법을 설명한다. 5장에서는 여러 실험들을 통해 접근 제어 정책 미적용 시스템과 제안 방법을 적용한 시스템의 성능을 비교 평가하고, 최종적으로 6장에서 결론 및 향후 연구로 논문을 맺는다.

2. 관련연구

2.1 최소 단위 XML 접근제어 모델

XML 데이터에 대한 권한부여로 XPath[2]를 이용하는 다양한 접근 단위의 객체를 보호하는 것과 관련되어 있다. 일반적으로, 현존하는 XML 접근제어 모델은 5-튜플 (주체, 객체, 접근유형, 기호, 타입)로 구성된 접근제어 규칙 명세를 가정한다[4-13]. 주체는 권한을 갖고 있는 사용자 혹은 사용자 그룹을 말한다. 객체는 XML 데이터 (혹은 일부분)와 관련되며 XPath[2]를 이용하여 표현하고 있다. 접근유형은 연산의 종류를 말하고, 기호는 접근의 허가 (긍정적 접근제어 규칙: +) 혹은 불허 (부정적 접근제어 규칙: -)를 말한다. 타입은 접근의 범위로서 전파허용(Recursive) 혹은 전파불허(Local)를 말한다. E. Damiani et al.[4-7], E. Bertino et al.[8-11],Gabilon 과 Bruno[12], 그리고 Stoica와 Farkas[13]들의 연구에서는 XML의 계층적 성질에 의해 접근 범위를 다음과 같이 분류한다.

(1) 노드 그 자체
(2) 노드와 그 노드의 속성
(3) 노드와 그 노드의 텍스트 자식 노드
(4) 노드와 그 노드의 속성, 그리고 그 노드의 자손 및 자손들의 속성.

만약 접근 범위가 (1)과 (2)라면 전파불허이고 (3)과 (4)라면 전파허용이 된다.

한편, 부정적(negative) 접근제어 규칙과 긍정적(positive) 접근제어 규칙을 혼합하여 사용하는 방법에는 두 가지 유형이 있다[3]. 첫 번째 유형은 긍정적 접근제어 규칙의 범위 내의 부정적 접근제어 규칙의 조합이고 두 번째 유형은 부정적 접근제어 규칙의 범위 내의 긍정적 접근제어 규칙의 조합이다. M. Murata et al.[16]은 첫 번째 유형을 '접근불허 하향 일관성 (denial downward consistency)'이라고 정의하고 있으며 특히, 두 번째 유형은 '불법적인 판독 접근성 (invalid read accessibility)'의 문제를 앓고 있다고 지적하고 있다.

2.2 효율적인 XML 접근제어 이행 메커니즘

전통적인 XML 접근제어 이행 메커니즘 [4-13]은 뷰-기반 이행 메커니즘이다. 즉, 질의를 한 사용자와 관련된 접근제어 규칙들에 의해 생성된 문서의 특정 뷰를 기반으로 사용자의 접근을 제어한다. 트리 레이블링 기법으로 뷰를 계산할 수 있는 유용한 알고리즘을 제시하였지만, 뷰 생성의 높은 비용 및 뷰 유지비용의 문제점을 앓고 있다. 또한, 사용자 증가에 따른 확장성 문제를 갖고 있다.

뷰 기반의 문제점을 극복하기 위해 M. Murata et al.[16]은 접근 제어 정책을 만족하지 못하는 질의를 사전에 걸러낼 수 있는 필터링 방법을 제안하였다. J. M Jeon et al. [17]은 XML 문서에 대한 모든 접근제어 규칙에 대한 구조 요약 정보를 에지로 구성하고, 각 노드에 접근제어 규칙을 기록한 XML 접근제어 트리(XACT)를 두어 사용자의 질의를 XACT 접근제어 정보들을 비교하여 변형된 안전한 질의로 변형하는 방법을 제안하였다. XACT는 질의를 제출한 사용자뿐만 아니라 문서에 대한 모든 접근제어 규칙을 갖고 있기 때문에 불필요한 계산 시간을 갖고 있다. B. Luo et al. [18]은 [16]의 방법론을 한층 발전시켜 접근제어 정책의 일부분만을 만족하고 있는 질의에 대해 공유 비결정 유한 오토마타 (shared NFA) 방법을 이용해 재작성하는 기법을 제안하였다. 그러나, 공유 비결정 유한 오토마타 방법은 사용자의 질의 관점에서 불필요한 접근제어 규칙들에 대한 오토마타까지 포함하고 있어 사용자 질의의 접근허가, 접근불허, 혹은 질의 재작성에 대한 결정을 내리는데 불필요한 시간 낭비를 초래한다.

3. 배경

이번 장에서는 본 논문에서 제안하는 XML 접근제어 방법의 배경 지식과 논문에서 제안하고 있는 가정들 및 기본 용어들에 대해 설명한다.

3.1 XML

XML 문서는 엘리먼트, 속성, 그리고 텍스트 노드로 구성된다. 각 엘리먼트의 내용은 중첩된 엘리먼트들의 일련의 순서 혹은 텍스트 노드이다. 하나의 엘리먼트는 속성 집합을 가질 수 있으며 이들은 이름(name)과 값(value)로 되어 있다. 속성은 엘리먼트에 대한 추가적인 정보를 제공한다.

XML 문서는 잘 정형화된(well-formed) 문서와 유효한(valid) 문서, 두 가지 유형이 있다. 다음의 세 가지 규칙들을 준수한 문서를 잘 정형화된 문서라 한다.

(1) 문서는 적어도 하나의 엘리먼트를 포함해야 한다.
(2) 문서는 하나의 루트 엘리먼트를 포함해야 한다. 이것은 문서 전체를 둘러싼 유일한 시작 태그와 끝 태그이다.
(3) 다른 모든 엘리먼트는 엘리먼트들 간의 겹침 (overlap)이 없이 중첩되어야 한다.

한편, 유효한 문서는 엘리먼트 혹은 속성 생성의 규칙들을 표현한 Document Type Definition(DTD) 혹은 XML Schema에 정의된 허용된 타입과 순서 규칙들을 준수한 잘 정형화된 문서를 말한다. 따라서, 유효한 문서 환경에서는 XPath 기반의 질의의 경로 구조(프레디키트는 제외)는 반드시 DTD(혹은 XML Schema)의 구조를 따르게 된다.

[가정 1] 본 논문에서는 DTD 기반의 유효한 XML을 이용한다.

비록 설명되는 내용은 DTD 기반의 유효한 XML 환경이지만, XML Schema 기반의 유효한 문서 환경뿐만 아니라, 잘 정형화된 XML 환경에서도 같은 방법으로 적용될 수 있다. 잘 정형화된 XML 환경에서는 일시적인 유연한 DTD를 생성하여 할 수도 있고, 아니면 직접 XML 문서에 적용할 수 있다.

3.2 XPath와 접근제어

XPath는 XML 문서를 트리 형태로 표현하고 특정 부분을 나타내는 경로 표현 언어이다. 전형적인 경로 표현은 연속적인 스텝(location step)들로 이루어진다. 각 스텝은 문서 내의 노드 사이의 자식(child), 속성(attribute), 조상(ancestor), 혹은 자손(descendant) 등의 관계를 나타낸다. 경로 상의 어떤 스텝은 또한 선택되는 노드를 걸러주는 필터 역할을 하는 프레디키트(predicate)를 포함하기도 한다. 이와 같은 XPath의 특성을 이용하여 접근 단위의 최소화 요구사항을 만족시키기 위해 XML 문서에 대한 접근을 이행하는 권한부여 모델은 XPath를 이용한다[2].

일반적인 접근제어는 추가적인 계산 시간이 생긴다. 그러나 XML 문서 기반의 접근제어 규칙이 XPath로 표현되고, 사용자는 XPath 기반의 질의를 하는 환경에서는 이와 같은 불필요한 접근제어 규칙들에 대한 추가 시간의 낭비를 줄일 수 있음을 간과하고 있다. 즉, XML 문서는 사용자의 질의를 기반으로 조상(Ancestor, 부모 포함), 후손(Descendant, 자식 포함), 그리고, 형제(Sibling, following-sibling과 preceding-sibling), 이렇게 세 부분으로 나뉠 수 있다. 결국, 사용자의 질의를 기반으로 필요한 접근제어 규칙들은 조상 혹은 후손 영역에 기술된 접근제어 규칙들 만이다. 형제 영역에 기술된 접근제어 규칙들은 사용자 질의 관점에서는 불필요한 접근제어 규칙이 된다.

본 논문에서는 DTD에 대한 메타데이터인 롤 프라임 넘버 테이블을 통해 사용자의 질의가 들어오면 그 사용자의 접근제어 규칙들만을 추출하는 효과적인 알고리즘을 제시한다.

3.3 접근제어 정책

계층적 데이터 모델(예를 들어, Object-Oriented Data Model, XML Data Model 등) [3]에서는 보안 관리자가 명시한 권한부여는 명시적(explicit)이라고 부르며, 명시적 권한부여를 기반으로 시스템이 파생한 권한부여를 묵시적(implicit)이라고 한다. 저장의 이점을 얻고자 묵시적 권한부여 방법을 사용하면서 적절하게 '전파 정책[2] (propagation policy)' 을 만든다. 여러 환경에 따라 최적의 전파 정책은 다르겠지만 일반적으로 '가장 특정화된 것을 우선으로 하는 정책(*most specific precedence*)'을 사용한다. 이와 같은 묵시적 권한부여에 의한 전파 정책에 의해 발생될 수 있는 '충돌(conflict) 충돌이라 함은 어떤 노드가 '접근허용' 및 '접근불가' 규칙이 동시에 정의되어 있는 경우를 말한다.' 문제를 해결하는 정책으로는 '거절 우선 정책(denial take precedence)'을 일반적으로 사용한다. 또한, 긍정적(positive) 권한부여와 부정적(negative) 권한부여를 혼합하여 사용하기 때문에 명시적인 권한부여가 없는 노드에 대한 '결정 정책(decision policy)'로 명시적 권한부여가 없는 노드는 접근을 허용하는 '개방(open) 정책'과 접근을 불허하는 '폐쇄(closed) 정책'을 사용한다.

2) 한 노드에 대한 접근제어 결과가 그 노드의 후손에 전파되는지 여부를 결정하는 정책을 말한다.

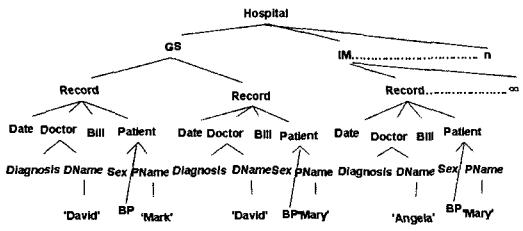
일반적으로, 엄격한 데이터 보안을 위해 '거절 우선 정책'과 '폐쇄 정책'을 사용한다.

3.4 XFrag

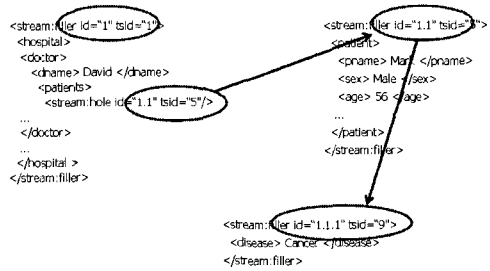
유비쿼터스 컴퓨팅의 실현을 위해서는 이동 단말기의 자원 및 컴퓨팅 파워의 효율적 사용이 필수적이다. 특히, 이동 단말기에 내장된 소프트웨어의 수행에 있어 메모리 효율성, 에너지 효율성, 그리고 처리 효율성이 요구된다. 여기서 XML 데이터는 계층적 구조를 가지고 있으며 용량이 매우 커질 수 있다. 따라서 이동 단말기의 적은 메모리로 대량의 XML 데이터를 처리하려면, XML 데이터를 적절한 조각으로 분할(fragmentation)하여 조각 단위로 처리하는 기술이 요구된다[19,20]. XML 분할에 의해 얻어지는 이점은 다음과 같다. 센서 기반의 환경 등에서 생성되는 XML 스트림 데이터를 구조적으로 분할하여 XML 조각 스트림으로 전송하고 처리하면, 이동 단말기의 메모리 효율성 및 처리 효율성을 재고할 수 있다. 또한, XML 스트림에서 어떤 특정 정보가 갱신될 경우, 전체 XML 데이터를 재전송하지 않고 변화된 조각만 전송함으로써 송수신 비용에서 이득을 얻을 수 있다.

최근에 제시된 홀-필러(hole-filler) 모델[21,22]은 XML 데이터를 구조적으로 분할하는 기법을 제시하고 있다. XFrag[21]나 XFPro[22]는 홀-필러 모델을 사용한 XML 조각 처리 기법을 제시하고 있는데, 홀-필러 모델이 필요로 하는 부가 정보로 인하여 낮은 처리 속도 및 메모리 공간 낭비 등의 문제점이 있다. XFPro는 XFrag의 파이프라인(pipeline)을 개선하여 속도를 향상시켰으나, 홀-필러 모델이 가지고 있는 문제점에 대해서는 해결 방안을 제시하지 않았다.

<그림1>과 같은 의료정보 XML 문서를 가정할 때 <그림2>와 같이 홀-필러 모델을 이용한 분할된 XML 문서를 확인 할 수 있다.



〈그림1〉 의료정보 XML 문서



〈그림2〉 홀-필러 모델을 이용한 분할된 XML문서

3.5 프라임 넘버 레이블링 기법(Prime Number Labeling Method)

동적인 XML 문서의 질의 처리를 위해서는 문서 내의 엘리먼트 들의 삽입, 삭제 시에도 이를 쉽게 반영할 수 있는 레이블링 기법이 필요하다. 문서의 갱신에 대한 고려를 하지 않은 기존의 레이블링 기법들은 문서의 갱신이 일어날 때마다 변화된 레이블 정보를 반영하기 위해서 전체 XML 트리를 재탐색하여 전체 노드의 레이블을 다시 계산하는 비용이 들게 된다[23].

동적인(dynamic) XML 문서의 등장으로 변경된 정보를 잘 반영하고 다른 부분의 레이블에 큰 영향을 주지 않는 레이블링 기법이 등장하게 되었는데 가장 대표적인 것이 프라임 넘버(prime number) 즉, 소수로 레이블링을 하는 프라임 넘버 레이블링 기법(prime number labeling scheme)[24]이다. 이 기법은 XML 트리에서 각 노드의 레이블을 소수로 부여하여 조상-후손관계를 나타낼 수 있는 특성을 가지며 문서를 갱신할 때 다른 노드의 레이블에 영향을 주지 않도록 설계되어 있다. 그러나 문서가 갱신될 때 엘리먼트 간의 순서를 나타내는 순서정보 값(order number)이 갱신되는 비용이 XML 트리의 상당히 많은 부분을 재탐색하고 갱신된 순서정보를 재 기록하는 등 많은 갱신비용이 들게 된다.

3.6 문제 정의

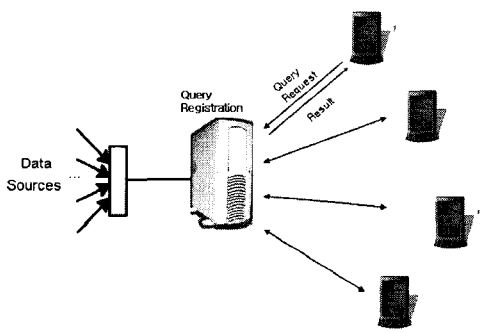
3.4절과 3.5절에서 지적하였듯이 기존의 접근제어 정책으로 구현하기 어려운 XML 데이터 스트림 환경에서 기존 방법을 개선한 본 논문에서 제안하고 있는 '롤 프라임 넘버 레이블링 기법을 이용한 세분화된 접근제어 방법'은 질

의자가 임의의 XML 문서에 대해 질의를 하였을 때, XML 조각 나누기 특징과 롤 프라임 넘버 레이블링 방법을 이용하여 효율적이면서도 안전한 질의 처리를 가능하게 해준다.

4. 제안 방법

4.1 제안 환경

제안하려고 하는 '롤 프라임 넘버 레이블링 기법'의 환경은 <그림3>에서처럼 유비쿼터스 환경에서 이동 단말기로부터 질의요청이 들어 왔을 때 접근제어 정책에 의해 유효한 사용자에게 권한과 역할을 체크해서 실시간으로 정확한 질의응답을 필요로 하는 의료정보 서비스를 가정한다.

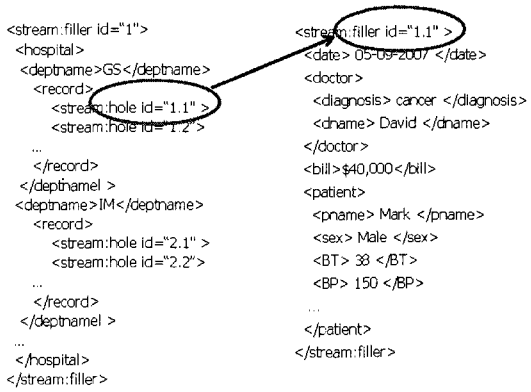


<그림3> 이동단말기의 XML 데이터 스트림 서버 접속 예

4.2 롤 프라임 넘버 레이블링 기법

(Role Prime Number Labeling Method)

<그림3>과 같은 환경에서 '롤 라임 넘버 레이블링 기법'을 설명한다. 우선, 제안 환경의 특징을 고려하여 <그림1>의 XML 문서에 대해 부분적인 단편화(fragmentation)를 <그림4>와 같이 시행한다. <그림4>에서 보듯이 기존 XFrag[19]의 홀-필러 모델이 필요로 하는 부가 정보로 인하여 낮은 처리 속도 및 메모리 공간 낭비 등의 문제점을 최소화 하였다. 즉, 단편화된 XML 문서의 순서를 고려할 필요가 없기 때문에 태그 스트럭쳐 등의 정보가 더 이상 불필요하게 되었다.



<그림4> XML 데이터 스트림의 부분적 단편화

이렇게 XML 데이터 스트림의 부분적인 단편화 과정을 거친 뒤, <그림1>의 의료정보 XML 문서의 노드에 롤(role)에 기초한 고유 프라임 넘버를 <표1>의 롤 프라임 넘버 테이블(Role Prime Number Table)과 같이 부여한다. 어떤 조직에서 롤은 한정적이기 때문에 프라임 넘버로도 충분히 표현 가능하며 확장이 용이하다.

<표1> 롤 프라임 넘버 테이블

Roles	Prime Number
Patient	2
Doctor	3
Researcher	5
Insurer	7

<표1>을 참조해 다음과 같은 과정을 거쳐 '롤 프라임 넘버 레이블링' 과정을 <알고리즘1>과 같이 수행한다.

(1) Level 1 : department에 integer number 할당(1~n)
  - GS : 1
  - IM : 2
(2) Level 2 : department 하위 노드인 record에 각각
  - GS 하위 Record 노드 : (1.1), (1.2)...
  - IM 하위 Record 노드 : (2.1), (2.2)...
(3) Level 3 : Record 노드 하위의 각 노드에 Role Prime Number Labeling
  [department].[record].[each node]
  - Date(*.*.210) : 2,3,5,7의 곱
  - Doctor(*.*.30) : 2,3,5 의 곱
  - Diagnosis(*.*.30) : 2,3,5 의 곱
  - Dname(*.*.6) : 2,3의 곱

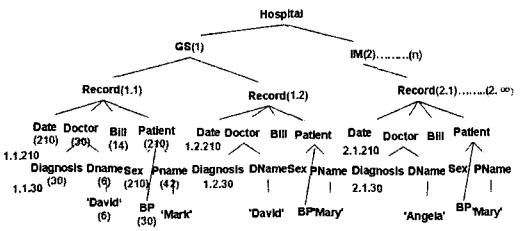
  - 단말 노드의 속성 값은 상위 노드 RPNL 상속
  - Patient(*.*.210) : 2,3,5,7의 곱

- Sex(*.*.210) : 2,3,5,7의 곱
- Panem(*.*.42) : 2,3,7의 곱
- BP(*.*.30) : 2,3,5의 곱

(4) Order 고려하지 않음

〈알고리즘1〉 롤 프라임 넘버 레이블링

<알고리즘1>을 이용해 얻은 의료정보 XML 문서는 <그림5>와 같다.



(a) 의료정보 XML 문서의 RPN

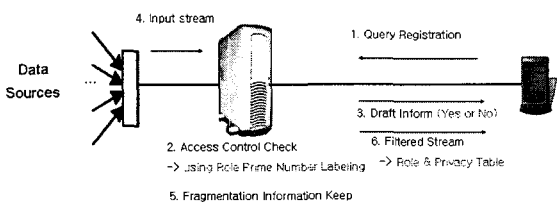
Node	Prime Number Label
Patient	1.1.210
Doctor	1.1.30
Pname	1.1.42
BP(Blood Pressure)	1.1.30

(b) //GS/Record(1.1)의 RPN

〈그림5〉 RPNL을 적용한 의료정보 XML 문서

4.3 RPN에 의한 질의 처리 과정

<그림1>의 환경에서 질의 처리 과정을 <그림6>으로 표현해 보았다.



〈그림6〉 질의 처리 흐름도

<그림6>의 질의 처리 과정은 2단계로 나누어 생각해 볼 수 있다. 1단계에서는 '롤 프라임 넘버 테이블'을 이용해서 롤 체크를 하고, 2단계에서는 '롤 프라임 테이블'을 이용해 최종 접근 권한을 체크한다. 우선 <그림6>에서 처럼 이동 단말기로부터 질의가 등록되면 1단계에서 XPath 질의 단말 노드의 프라임 넘버를 체크하여 접근제어 권한을 체크한다. 즉, 그 단말 노드의 프라임 넘버를 질의자의 롤로 나누어 나머지가 '0'이면 1단계 접근을 허용하는 것으로 본다. 그리고 2단계에서 <표2>의 롤 프라이버시 테이블(Role Privacy Table)을 참조하여 접근 가능 여부를 최종 확인한다. 또한, 2.1절에서 지적한 바와 같이 각각 1단계와 2단계에서 접근제어 정책이 상호 상반되는 경우, 부정적 접근제어와 긍정적 접근제어가 상존하는 경우 '부정적 접근제어 우선 정책[16]'에 의하여 질의 접근을 거절하게 된다. 자세한 사항은 다음의 예제를 통해 확인해 보기로 한다.

〈표2〉 롤 프라이버시 테이블

dept.	records	roles			
		patient	doctor	insurer	researcher
1	1.1	Mark	David	ING	x
	1.2	Mark	David	AIG	x
	...	...			x
	1.∞		...		x
2	2.1	Mary	Angela	AIG	x
	...	...			x
	2.∞		...		x
...	...			...	x
n	n.∞				x

[예제 1] 프레디키트 + 긍정적 접근제어 + 긍정적 접근제어

(1) //record/patient[pname="Mark"]

(2) doctor의 role을 가진 "David"가 질의요청

- 1단계, 단말노드 pname="Mark"의 레이블 확인 : 1.1.42
- david's role = doctor : 3
- 42%3=0이므로 접근허용
- 2단계, 롤 프라이버시 테이블에서 David은 1.1.*와 1.2.*만 허용
- 최종적으로 1.1.42(*//record/patient[pname="Mark"]*) 접근 허용
- 질의 요청에 대한 응답

[예제 2] 긍정적 접근제어 + 긍정적 접근제어

(1) //record/bill

(2) insurer의 롤을 가진 "AIG"가 질의요청

- 1단계, 단말노드 bill의 레이블 확인 : *.*.14
- insurer's role : 3
- 14%7=0이므로 접근허용
- 2단계, 롤 프라이버시 테이블에서 AIG는 1.2.*와 2.1.* 만 허용
- 최종적으로 1.1.14와 2.1.14 접근 허용
- 질의 요청에 대한 응답

[예제 3] 프레디키트 + 긍정적 접근제어 + 부정적 접근제어

(1) //record/patient[pname="Mark"]

(2) doctor의 롤을 가진 "Angela"가 질의요청

- 1단계, 단말노드 pname="Mark"의 레이블 확인 : 1.1.42
- angela's role = doctor : 3
- 42%3=0이므로 접근허용
- 2단계, 롤 프라이버시 테이블에서 Angela는 2.1.*만 허용 [pname="Mark"]는 1.1.42 이므로 접근 불허
- 1단계 접근 허용, 2단계 접근 불허 따라서 최종적으로 질의접근 불허

[예제 4] 부정적 접근제어

(1) //record/patient/pname

(2) researcher의 롤을 가진 자가 질의 요청

- 1단계, 단말노드 pname의 레이블 확인 : *.*.42
- researcher's role : 5
- 42%5≠0이므로 접근불허
- 최종적으로 질의 거절

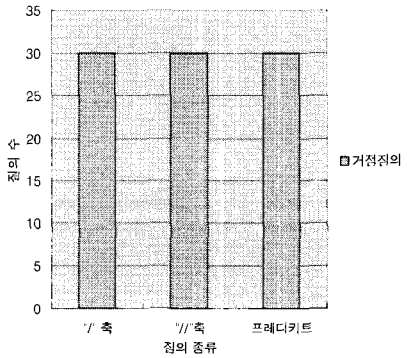
위의 [예제4]에서도 살펴 보았 듯이 제안하는 방법의 특징은 거절질의를 신속하게 처리할 수 있다는 또 하나의 장점이 있다.

5. 실험

실험 환경은 CPU 인텔 펜티엄 IV 3.0GHz, 메인 메모리 512MB이고 운영체제는 MS Windows XP Professional인 PC에서 수행 하였다. 구현 언어는 JAVA(JDK1.5.0)를 사용하였다. 실험데이터는 임의의 의료정보 XML 문서를 작성하여 사용하였으며, 크게 두 가지 관점에서 성능을 비교하였다.

5.1 실험 I : 거절 질의에 대한 정확한 탐지

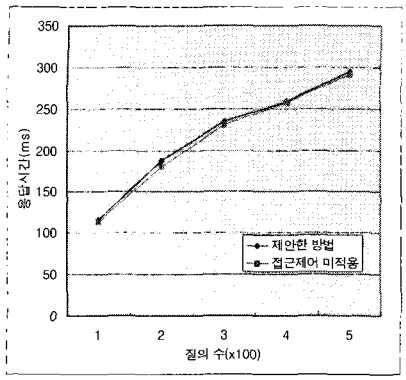
거절 질의(Rejection Query)라 함은 항상 거절이 되어야 하는 사용자 질의를 말한다. 접근제어 정책에 의해 각각의 질의 유형에 맞는 30개의 의도된 거절 질의를 만들고 실제 시스템에서 거절 질의의 탐지 개수를 비교해 보았다. 그 결과는 <그림7>과 같다. 여기서 "/"축 , "//"축 그리고 단말노드에 프레디키트가 있는 경우 세 가지로 실험을 하였다. 결과는 <그림7>과 같이 의도된 30개의 거절질의에 대해 100% 탐지함을 확인하였다.



〈그림7〉 거절 질의 탐지에 대한 결과

5.2 실험 II : 질의 응답시간

기존 시스템에서 접근제어를 적용하지 않은 질의 처리와 본 논문에서 제안한 접근제어 방법을 적용한 경우를 평균 질의 응답시간으로 비교하였다. 무작위로 추출한 XPath 질의 수(50, 100, 200, 300, 그리고 500)에 따른 평균 응답시간을 측정하였다. 응답시간 측정의 오차를 최소화 하기위해 본 실험에서는 각각 100번을 수행하여 그 응답시간의 평균값으로 응답시간을 측정하였다. 제안한 방법에서 롤 프라임 넘버 레이블링 시간은 의료정보 XML 문서 노드의 삽입, 삭제와 같은 갱신이 발생하는 경우에 재구성 되므로 응답시간 측정에 고려하지 않았으며, 질의 처리 과정 전에 이미 생성된 롤 프라임 넘버 레이블링을 참조하여 순수하게 접근 권한을 체크하는 과정을 포함한 응답속도를 측정하였다. 그럼에도 불구하고 접근제어 정보를 이용하는 것이 시스템의 성능에 큰 영향을 미치지 못함을 확인할 수 있었다. <그림8>에서 그 결과를 보여주고 있다.



〈그림8〉 질의에 대한 접근제어 처리 수행 시간

6. 결론 및 향후연구

유비쿼터스 환경에서 자주 이용되는 이동단말기에 내장된 소프트웨어의 수행에 있어 메모리 효율성, 에너지 효율성, 그리고 처리 효율성을 고려해 볼 때, 대용량의 XML 데이터 스트림 처리는 불가능하다. 또한, 유비쿼터스 환경에서 이동단말기에 접근제어를 구현한다는 것은 더욱 불가능 한 일 일 것이다. 하지만 사용자의 증가, 사용 데이터량의 증가 등으로 인해 보안의 적용이 중요한 이슈이다. 본 논문에서 이러한 두 가지 문제해결을 위해 '롤 프라임 넘버 레이블링 기법을 이용한 세밀한 접근제어 방법'을 제안하였다. 또한, 논문에서 제안한 방법의 환경으로 정의한 의료정보 XML 문서의 경우는 환자 등의 증가로 인해 깊이(depth) 보다는 폭(width)이 무한히 증가 할 수 있는 특징을 가지고 있다. 이렇게 볼 때 '롤 프라임 넘버 레이블링 기법'은 무한히 증가하는 문서의 사이즈를 충분히 감당해 낼 수 있으며, 동적인 변화에도 최소한의 비용으로 XML 문서의 유지가 가능해 유지비용을 최소화 할 수 있다. 또한, 질의 응답시 보안적용을 위해 기존의 연구에서 XML 문서의 트리구조를 2회 이상 검색해야 했으나 제안 방법은 XML 문서의 단편화와 레이블링 방법을 이용해 1회 검색으로 실시간 처리가 가능하며, 문서의 단편화로 인해 서버에서 이동단말기로의 전송비용을 최소화 할 수 있다. 보안 측면은 2단계 보안 적용을 통해 시스템 부하를 최소화 하며 완벽한 접근제어를 구현하였다. 우선, 프라임 넘버의 특징을 이용해 접근제어 규칙에 맞지 않는 롤을 가진 질의자의 질의를 간단하게 거절할 수 있으며, 2단계 보안 적용으로 한층 강화된 접근제어 규칙을 적용 할 수 있다. 다만, '롤 프라이버시 테이블'을 유지해야하는 약간의 부담이 있을 뿐이다. 그러나 실험을 통해 제안하는 방법의 우수성을 보였다.

향후연구로는 '롤 프라임 넘버 레이블링'과 다양한 접근제어 모델의 적용을 통해 더욱 세분화 되고 효율적인 접근제어 방안을 모색 해보고자 한다.

참고문헌

[1] T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, F. Yergeau. Extensible Markup Language (XML) 1.0, World Wide Web Consortium (W3C), 2004. (http://www.w3.org/TR/REC-xml)

[2] A. Berglund, S. Boag, D. Chamberlin, M. F. Fernández, M. Kay, J. Robie, and J. Siméon. XPath 2.0, World Wide Web Consortium (W3C), 2005. (http://www.w3.org/TR/xpath20/)

[3] F. Rabitti, E. Bertino, W. Kim and D. Woelk.A Model of Authorization for Next-Generation Database Systems. ACM Transaction on Database Systems, Vol 126, No 1. March 1991, PP. 88-131.

[4] E. Damiani, S.Vimercati, S.Paraboschi, and P.Samarati. Securing xml document. In Proc. of the 2000 International Conf.on Extending Database Technology (EDBT2000), Konstan, Germany, March, 2000, pp.121-135.

[5] E. Damiani, S. Vimercati, S. Paraboachk and P.Samarati.XML Access Control Systems: A Component-Based Approach. In Proc. IFIP WG11.3 Working Conference on Database Security, The Netherlands, 2000. 8.

[6] E. Damiani, S. Vimercati, S. Paraboachk and P.Samarati. Design and Implementation of Access Control Processor for XML Documents. Computer Network, 2000.

[7] E. Damiani, S. Vimercati, S. Paraboachk and P.Samarati.A Fine-grained Access Control System for XML Documents. ACM Trans. Information and System Sec., Vol.5, No.2, May 2002.

[8] E. Bertino, S. Castano, E. Ferrari, and M. Mesiti. Specifying and Enforcing Access Control Policies for XML Document Sources. WWW Journal, Baltzer Science Publishers, Vol.3, N.3, 2000.

[9] E. Bertino, S. Castano, and E. Ferrai. Securing XML documents with Author-x.IEEE Internet Computing, May.June, pp.21-31, 2001.

[10] E. Bertino and E. Ferrari. Secure and Selective Dissemination of XML Documents. TISSEC, 5(3), pp. 237-260, 2002.

[11] E. Bertino, M. Braun, S. Castano, E. Ferrari, and M. Mesiti. Author-X: A Java-Based System for XML Data Protection.In Proc. IFIP WG11.3 Working Conference on Database Security, Netherlands, 2002. 8.

[12] A. Gabillon and E. Bruno. Regulating Access to XML Documents. In Proc. IFIP WG11.3 Working Conference on Database Security, 2001.

[13] A. Stoica and C. Farkas. Secure XML Views. In Proc. IFIP WG11.3 Working Conference on Database and Application Security, 2002.

[16] M. Murata, A. Tozawa, and M. Kudo. XML Access Control using Static Analysis. In ACM CCS, Washington D.C., 2003.

[17] Jae-Myeong Jeon, Yon Dohn Chung, Myoung Ho Kim, and Yoon Joon Lee, Filtering XPath expressions for XML access control. Computers & Security, 23, pp.591-605, 2004.

[18] B. Luo, D. W. Lee, W. C. Lee, and P. Liu. Qfilter: Fine-grained Run-Time XML Access Control via NFA-based Query Rewriting. In Proc. of the Thirteenth ACM Conference on Information and Knowledge Management 2004(CIKM'04), November 8, 2004, Washington, USA.

[19] "XML Fragment Interchange," W3C Candidate Recommendation 2001.

[20] Sujoe Bose, Leonidas Fegaras, David Levine and Vamsi Chaluvadi, "A Query Algebra for Fragmented XML Stream Data," DBLP 2003.

[21] Leonidas Fegaras, David Levine, Sujoe bose and Vamsi Chaluvadi, "Query Processing of Streamed XML Data," CIKM 2002, pp. 126-133.

[22] Sujoe Bose and Leonidas Fegaras, "XFrag: A Query Processing Framework for Fragmented XML Data," Web and Databases 2005.

[22] Huan Huo, Guoren Wang, Xiaoyun Hui, Rui Zhou, BoNing, and Chuan Xiao, "Effiecient Query Processing for Streamed XML Fragments," DASFAA 2006.

[23] Masatoshi Yoshikawa, Toshiyuki Amagasa, et al.,XRel: A Path-Based Approach to Storage and Retrieval of XML Documents Using Relational Databases, ACM Transaction on Internet Technology, 2001.

[24] Xiaodong Wu, Mong Li, Lee Wynne Hsu, A Prime Number Labeling Scheme for Dynamic Ordered XML Trees, ICDE, 2004.