질의-인식 복호화를 사용한 암호화된 XML 데이타에 대한 안전한 질의 처리

(Secure Query Processing against Encrypted XML Data Using Query-Aware Decryption)

이 재 길 † 황 규 영 ††
(Jae-Gil Lee) (Kyu-Young Whang)

요 약 인터넷에서 XML 데이타를 그대로 배포할 경우 모든 사용자가 배포된 XML 데이타를 아무 제약 없이 액세스할 수 있어, XML 데이타 제공자의 프라이버시가 보장되지 않는다. 따라서, 배포된 XML 데이타에 대해서 액세스를 통제할 수 있도록 암호화 기법을 사용하는 방법들이 최근에 제안되었다. 그러나, 이들 방법에서는 배포된 XML 데이타에 대한 질의 처리 성능이 충분히 논의되지 않았다. 질의 처리기는 암호화된 XML 데이타를 복호화하기 전까지 XML 데이타의 실제 내용을 알 수 없으며, 이로 인해 질의 결과를 포함하지 않은 부분까지도 복호화해야 하는 오버헤드가 발생한다. 본 논문에서는 암호화된 XML 데이타에 대한 효율적인 질의 처리를 위해 질의-인식 복호화(query-aware decryption)라는 개념을 제안한다. 질의-인식 복호화란 암호화된 XML 데이타 중에서 질의 결과를 포함하는 부분만 복호화하는 방법이다. 이를 위해 XML 인덱스를 암호화하여 데이타와 함께 배포한다. 암호화된 XML 인덱스만을 복호화함으로써 암호화된 XML 데이타에서 질의 결과가 포함되어 있는 위치를 알아내어, 다른 암호화된 XML 데이타의 불필요한 복호화를 방지할 수 있다. 암호화된 XML 인덱스는 암호화된 XML 데이타에 비해 그 크기가 매우 작으므로 이를 복호화하는 비용은 암호화된 XML 데이타를 불필요하게 복호화하는데 낭비된 비용에 비해 매우 작다. 실험 결과는 질의-인식 복호화를 사용하는 질의 처리 방법이 기존의 방법에 비해 질의 처리 성능을 최대 6배까지 향상시킴을 보인다. 마지막으로, 암호화된 XML 인덱스로 인해 추가적인 보안 누출이 발생하지 않음을 정형적으로 증명한다.

키워드 : XML, 프라이버시, 보안, 암호화, 복호화

Abstract Dissemination of XML data on the internet could breach the privacy of data providers unless access to the disseminated XML data is carefully controlled. Recently, the methods using encryption have been proposed for such access control. However, in these methods, the performance of processing queries has not been addressed. A query processor cannot identify the contents of encrypted XML data unless the data are decrypted. This limitation incurs overhead of decrypting the parts of the XML data that would not contribute to the query result. In this paper, we propose the notion of query-aware decryption for efficient processing of queries against encrypted XML data. *Query-aware decryption* allows us to decrypt only those parts that would contribute to the query result. For this purpose, we disseminate an encrypted XML index along with the encrypted XML data. This index, when decrypted, informs us where the query results are located in the encrypted XML data, thus preventing unnecessary decryption for other parts of the data. Since the size of this index is much smaller than that of the encrypted XML data, the cost of decrypting this index is negligible compared with that for unnecessary decryption of the data itself. The experimental results show that our method improves the performance of query processing by up to 6 times compared with those of existing methods. Finally, we formally prove that dissemination of the encrypted XML index does not compromise security.

Key words : XML, Privacy, Security, Encryption, Decryption

· 본 연구는 첨단정보기술연구센터를 통하여 한국과학재단으로부터 지원을 받았음

† 정 회 원 : 한국과학기술원 전산학과 / 첨단정보기술연구센터
jglee@mozart.kaist.ac.kr
†† 종신회원 : 한국과학기술원 전산학과 교수, 첨단정보기술연구센터 소장
kywhang@mozart.kaist.ac.kr

논문접수 : 2004년 10월 11일
심사완료 : 2005년 1월 24일

1. 서 론

XML이 인터넷 상에서의 문서 교환의 표준으로 자리 잡음에 따라, 인터넷에서 XML 데이타를 배포(dis-seminate)하는 응용이 증가하고 있다. 특히 Peer to Peer (P2P)[1]와 웹 서비스(web service)[2] 응용에서 XML 데이타의 배포가 많이 발생한다. 이러한 응용에서 XML 데이타를 그대로 배포할 경우 모든 사용자가 배포된 XML 데이타를 아무 제약 없이 액세스할 수 있어, XML 데이타 제공자의 프라이버시가 보장되지 않는다. 따라서, 배포된 XML 데이타에 대한 액세스 통제 방법이 중요한 연구 이슈가 되었다[3,4].

Bertino와 Ferrari[3] 및 Miklau와 Suciu[4]는 XML 데이타 중에서 액세스를 통제해야 하는 각 부분을 서로 다른 키(key)로 암호화(encrypt)하고, 암호화된 XML 데이타를 키와 함께 배포하는 방법을 제안하였다. XML 데이타를 전달받은 사용자는 자신이 보유한 키로 복호화(decrypt) 할 수 있는 부분만 액세스할 수 있으므로, 배포된 데이타에 대해서 액세스 통제가 가능해진다.

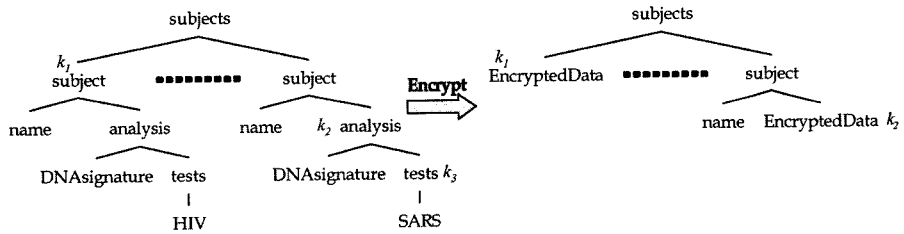
그러나, 이들 방법에서는 배포된 XML 데이타에 대한 질의 처리 성능이 충분히 논의되지 않았다. 배포된 XML 데이타가 누출되는 경우를 대비하기 위해, 이들 방법에서 배포된 XML 데이타는 복호화된 상태로 저장되어서는 안되고 암호화된 상태로 저장된다[5,6]. 그러므로, 질의 처리기는 암호화된 XML 데이타를 복호화하기 전까지 XML 데이타의 실제 내용을 알 수 없으며, 이로 인해 질의 결과를 포함하지 않은 부분까지도 복호화해야 하는 오버헤드가 발생한다.

예 1. XML 데이타를 그림 1(a)와 같이 암호화하여 배포한다고 가정한다. 암호화된 엘리먼트와 그 서브 엘리먼트들은 **EncryptedData** 엘리먼트로 대치된다[7]. 음영 처리된 k_1, k_2, k_3는 XML 엘리먼트를 암호화하는데 사용된 키를 나타낸다. 또한, 키 k_2, k_3를 가지고 있는 사용자가 그림 1(a)의 암호화된 XML 데이타에 대하여 /subjects/subject/analysis/tests/HIV 질의를 수행한다고 가정한다. 그러면, 그림 1(b)와 같이 연속적으로 두 번 복호화를 수행하여도 질의 결과가 나타나지 않는다. 이러한 불필요한 복호화로 인해 질의 처리 성능이 저하될 수 있다. □

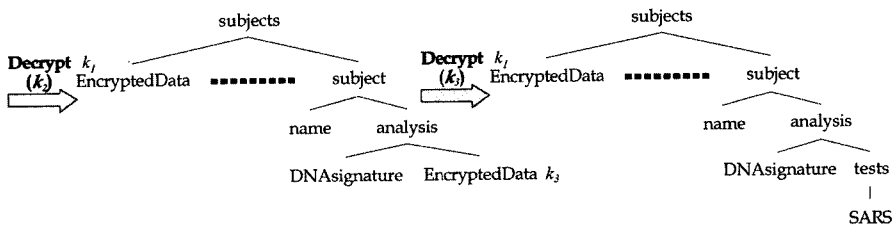
본 논문에서는 암호화된 XML 데이타에 대한 효과적인 질의 처리를 위해 질의-인식 복호화(query-aware decryption)라는 개념을 제안한다. **질의-인식 복호화**란 암호화된 XML 데이타 중에서 질의 결과를 포함하는 부분만 복호화하는 방법이다. 이를 위해 XML 인덱스를 암호화하여 데이타와 함께 배포한다. 암호화된 XML 인덱스만을 복호화함으로써 암호화된 XML 데이타에서 질의 결과가 포함되어 있는 위치를 알아내어, 다른 암호화된 XML 데이타의 불필요한 복호화를 방지할 수 있다. 암호화된 XML 인덱스는 암호화된 XML 데이타에 비해 그 크기가 매우 작으므로(통상적으로 1/20 이내), 이를 복호화하는 비용은 암호화된 XML 데이타를 불필요하게 복호화하는데 낭비된 비용에 비해 매우 작다.

본 논문의 공헌은 다음과 같이 요약된다.

- 암호화된 XML 데이타에 대한 질의 처리 시 불필요한 복호화를 방지하는 질의-인식 복호화 개념을 제안



(a) XML 데이타의 암호화 예



(b) 암호화된 XML 데이타의 복호화 예

그림 1 불필요한 복호화(decryption)가 발생하는 예

하고, 암호화된 XML 인덱스를 사용하여 질의-인식 복호화를 수행하는 방법을 제시한다.

- 암호화된 XML 인덱스로 인해 추가적인 보안 누출(security compromise)이 발생하지 않음을 정형적으로 증명한다.
- 다양한 실험을 통해 질의-인식 복호화가 암호화된 XML 데이타에 대한 질의 처리 성능을 크게 향상시킴을 입증한다.

본 논문의 구성은 다음과 같다. 제2장에서는 XML 데이타의 암호화에 대한 기존의 연구를 설명한다. 제3장에서는 질의-인식 복호화의 개념과 수행 방법을 제안한다. 제4장에서는 암호화된 XML 인덱스로 인해 추가적인 보안 누출이 없음을 증명한다. 제5장에서는 질의-인식 복호화를 사용한 질의 처리 방법의 성능 평가 결과를 제시한다. 마지막으로, 제6장에서는 결론을 내린다.

2. 관련 연구

본 장에서는 XML 암호화 방법에 대한 연구를 요약한다. 제2.1절에서는 W3C에서 제정한 XML 암호화 표준[7]을 설명한다. 제2.2절에서는 암호화를 사용하는 배포된 XML 데이타에 대한 액세스 통제 방법[3,4]을 설명한다.

2.1 XML 암호화 표준

W3C에서는 XML 데이타를 암호화하는 표준을 제정하였다[7]. 이 표준에 의하면 암호화된 XML 데이타는 XML 포맷을 따르며, 엘리먼트의 이름이나 내용이 EncryptedData 엘리먼트라는 암호화된 문자열로 대치된다. 이 엘리먼트는 EncryptionMethod, KeyInfo, CipherData, EncryptionProperties의 네개의 서브 엘리먼트를 가진다. EncryptionMethod는 암호화/복호화에 사용된 알고리즘 및 파라미터를나타낸다; KeyInfo는 암호화/복호화에 사용된 키의 이름을 나타낸다(키의 값은 포함하지 않는다.); CipherData는 서브 엘리먼트로 CipherValue를 가지며, CipherValue는 엘리먼트의 이름이나 내용이 암호화된 문자열을 나타낸다; EncryptionProperties는 EncryptedData 생성에 관련된 추가적인 정보를 제공한다. 이미 암호화된 엘리먼트를 다시 암호화하는 것을 허용하며, 이를 **수퍼-암호화**(*super-encryption*)라 부른다.

예 2. 그림 2(a)의 XML 데이타에서 첫번째 subject 엘리먼트를 암호화한 결과로 생성되는 XML 데이타는 그림 2(b)와 같다. 여기에서, 첫번째 subject 엘리먼트와 그 서브 엘리먼트들이 EncryptedData 엘리먼트로 대치된 것을 알 수 있다. □

2.2 XML 암호화를 사용한 액세스 통제 방법

Bertino와 Ferrari[3] 및 Miklau와 Suciu[4]는 암호화 기법을 사용하는 배포된 XML 데이타에 대한 액세스 통제 방법을 제안하였다. 이 두 방법은 액세스 통제 정책을 명시하는 방식에서 약간의 차이가 있을 뿐 근본적으로 동일하다. Bertino와 Ferrari는 XML 데이타에 권한을 부여하는 방식으로 액세스 통제 정책을 명시하며, Miklau와 Suciu는 XQuery[8]의 문법을 확장한 언어를 사용하여 액세스 통제 정책을 명시한다.

그림 3은 이들 방법을 개념적으로 설명한다. XML 데이타의 다른 두 부분을 각각 키 k_1과 k_2로 암호화한 후에, 사용자 A에게는 암호화된 XML 데이타와 키 k_1을 배포하고, 사용자 B에게는 암호화된 XML 데이타와 키 k_2를 배포한다. 사용자 A(혹은 사용자 B)가 액세스 할 수 있는 부분은 사용자 A(혹은 사용자 B)가 보유하고 있는 키 k_1(혹은 k_2)로 복호화할 수 있는 부분과 암호화되지 않은 부분의 합인 음영이 없는 부분이다.

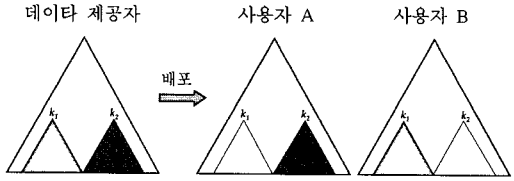
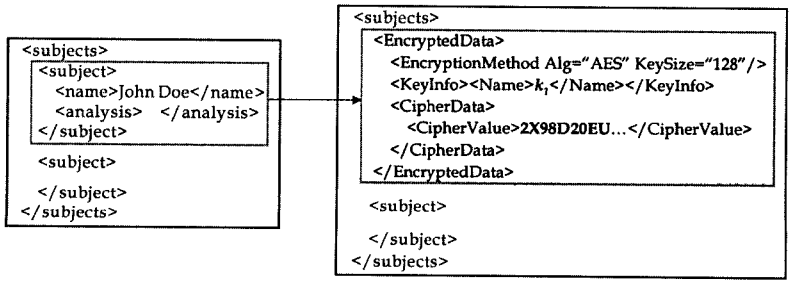


그림 3 Bertino와 Ferrari[3] 및 Miklau와 Suciu[4]의 방법



(a) 암호화 전의 XML 데이타. (b) 암호화 후의 XML 데이타.

그림 2 XML 데이타의 암호화[7]의 예

이들 방법에서는 암호화된 XML 데이타에 대한 질의 처리 성능은 충분히 다루지 않았다. Miklau와 Suciu[4]는 자신이 보유한 키로 암호화된 **EncryptedData** 엘리먼트를 일단 복호화하고, 복호화된 엘리먼트가 질의에 명시된 엘리먼트와 일치하는지 검사하는 질의 처리 방법을 제안하였다. 그러나, 이 방법은 서론에서 설명한 바와 같이 불필요한 복호화를 유발하는 문제가 있다. 본 논문에서는 질의-인식 복호화라는 개념을 사용하여 이러한 문제를 해결한다.

3. 질의-인식 복호화(Query-Aware Decryption)

본 장에서는 질의-인식 복호화(query-aware decryption)라는 개념을 제안한다. 제3.1절에서는 먼저 질의-인식 복호화를 정의한다. 제3.2절에서는 질의-인식 복호화를 지원하기 위한 암호화된 XML 인덱스를 설명한다. 제3.3절에서는 질의-인식 복호화를 사용한 질의 처리 알고리즘을 설명한다.

3.1 문제 정의(Problem Definition)

먼저 정의 1에서 *RelevantEncryption*이라는 함수를 정의하고, 이를 사용하여 정의 2에서 질의-인식 복호화를 정의한다.

정의 1. *RelevantEncryption*은 질의 결과와 관련있는 암호화된 데이타인지를 반환하는 함수로 다음과 같이 정의한다. 여기에서, *ED*는 **EncryptedData** 엘리먼트이고, *KEYS*는 보유하고 있는 키의 집합이며, *Q*는 XPath[9] 표현식(expression)이다.

$$RelevantEncryption(ED, KEYS, Q) = \begin{cases} true & \text{if } ED, \text{ when decrypted using } KEYS, \\ & \text{contains part of the result of } Q \\ false & \text{otherwise} \end{cases}$$

정의 2. 질의-인식 복호화는 $RelevantEncryption(ED, KEYS, Q) = true$인 **EncryptedData** 엘리먼트만을 복호화하는 것으로 정의한다.

3.2 암호화된 XML 인덱스(Encrypted XML Index)

질의-인식 복호화를 위해서는 *RelevantEncryption*(*ED, KEYS, Q*) 함수를 효과적으로 계산할 수 있는 방법이 필요하다. *RelevantEncryption*(*ED, KEYS, Q*)의 계산을 돕기 위해, XML 데이타의 구조를 요약하고 있는 정보를 함께 배포한다. 이러한 요약 정보는 질의 결과가 XML 데이타 내에서 어느 위치에 포함되어 있는지 알아낼 수 있게 해주므로, 특정 **EncryptedData** 엘리먼트가 질의 결과를 포함하고 있는지 알아낼 수 있다. 이러한 요약 정보로 인해 XML 데이타에 대한 정보가 누출되지 않도록 요약 정보 자체도 XML 데이타와 마찬가지로 암호화되어 사용자에게 배포된다.

본 논문에서는 요약 정보를 나타내기 위한 구조로서 역 인덱스(inverted index)[10]를 채택한다. 역 인덱스를 채택하는 이유는 이미 XML 데이타에 대한 인덱스 구조로 역 인덱스가 널리 사용되고 있기 때문이다[11].

본 논문에서 사용하는 XML 인덱스의 구조는 그림 4와 같다. 엘리먼트 타입이 텍스트 문서에서의 키워드에 해당하고, 각 엘리먼트 타입의 출현 위치가 포스팅 리스트에 해당한다는 점에서 전통적인 역 인덱스의 구조와 유사하다. XML 인덱스의 각각의 엔트리는 (1) 엘리먼트 타입(Element Type), (2) (1)의 엘리먼트 타입의 출현 위치(Occurrences), (3) (2)의 출현 위치가 가리키는 엘리먼트 타입의 인스턴스들을 암호화하는데 사용된 키의 이름(Key Name)을 나타낸다. 암호화되지 않은 인스턴스들을 가리키는 인덱스 엔트리의 Key Name 필드에는 *null*이 저장된다. XML 인덱스의 각각의 엔트리별로 Key Name, Element Type, Occurrences 필드도 Key Name 필드에 명시된 키로 암호화된다. 그러나, Key Name 필드가 *null*이면 암호화되지 않는다.

본 방법에서는 출현 위치를 나타내기 위해서 듀웨이 번호(Dewey number)[12]를 사용한다. 출현 위치를 나타내는 듀웨이 번호는 특정 서브트리가 하나의 **EncryptedData** 엘리먼트로 대치되더라도 변경되지 않는 좋은 특성을 지니고 있다. 본 논문에서는 이러한 특성을 **서브트리-대치 불변성**(*subtree-replacement invariant*)이라 부른다. 듀웨이 번호는 계층 구조를 가지는 데이타에서 특정 엘리먼트 인스턴스의 위치를 가리키며, 다음과 같은 방식으로 구성된다[12]: (1) 듀웨이 번호의 자리수는 루트로부터 인스턴스가 위치하는 레벨이다; (2) 듀웨이 번호의 각 자리의 값은 시블링(sibling)들 사이에 순차적으로 부여된 값이다. 예를 들어, 듀웨이 번호 1.3.2는 루트의 세번째 자식의 두번째 자식인 레벨 3의 인스턴스를 가리킨다. 이러한 듀웨이 번호의 정의에 따라, 듀웨이 번호 $i.\cdots.j$를 루트로 하는 서브트리가 하나의 인스턴스로 대치될 경우 듀웨이 번호 $i.\cdots.j.\cdots$를 가지는 인스턴스들이 사라질 뿐, 나머지 인스턴스들의 듀웨이 번

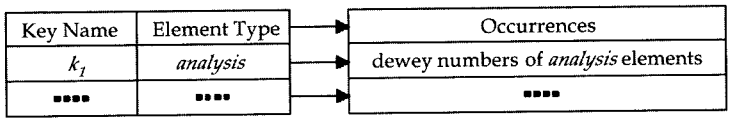


그림 4 암호화된 XML 인덱스의 구조

호에는 영향을 주지 않는다. 따라서, 듀웨이 번호는 서브트리-대치 불변성을 만족한다.

예 3. 그림 5(a)의 XML 데이타에서 듀웨이 번호가 1.1인 subject 엘리먼트 인스턴스를 키 k_1으로 암호화하고, 듀웨이 번호가 1.2.2인 analysis 엘리먼트 인스턴스를 키 k_2로 암호화하고, 듀웨이 번호가 1.2.2.2인 tests 엘리먼트 인스턴스를 키 k_3로 암호화하면, 그림 5(b)의 암호화된 XML 데이타가 생성된다. 이때, 구축되는 암호화된 XML 인덱스는 그림 5(c)와 같다. 엘리먼트 인스턴스 1.2.2.2와 1.2.2.2.1은 먼저 키 k_3로 암호화되고 다시 키 k_2로 암호화되므로, 이들 인스턴스를 가리키는 인덱스 엔트리들의 Key Name 필드는 $\{k_2, k_3\}$을 포함한다. 그림 5 (c)에서 음영 처리된 필드는 암호화된 상태임을 의미한다. □

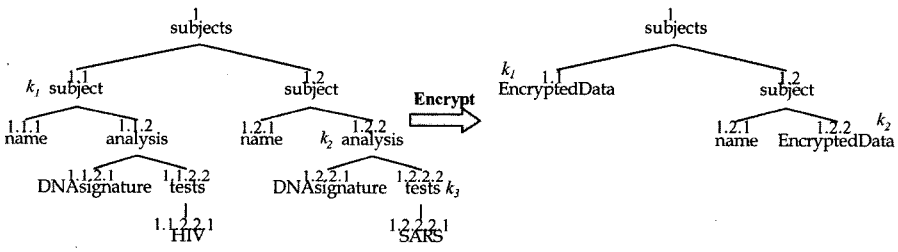
3.3 질의-인식 복호화를 사용한 질의 처리 알고리즘

보조정리 1에서 암호화된 XML 인덱스를 사용하여 *RelevantEncryption*(*ED*, *KEYS*, *Q*) 함수를 계산하는 방법을 제시한다.

보조정리 1. 암호화된 XML 인덱스에서 Key Name 필드의 값이 사용자가 보유한 키의 집합인 *KEYS*의 부분집합이고, Element Type 필드의 값이 질의 *Q*의 결과의 엘리먼트 타입들 중의 하나에 해당하는 인덱스 엔트리들을 고려하자. 이들 인덱스 엔트리의 Occurrences 필드에 저장된 듀웨이 번호의 집합을 $\mathbb{R}$이라고 하자. 즉, 이 듀웨이 번호들은 질의 결과가 될 수 있는 인스턴스들을 나타낸다. **EncryptedData** 엘리먼트 *ED*의 듀웨이 번호를 *e*라고 할 때, $\{ r \mid r \in \mathbb{R}$이고, *e*가 *r*의 프리픽스 $\} \neq \emptyset \Rightarrow$ *RelevantEncryption*(*ED*, *KEYS*, *Q*) = *true* 이다.

증명: 듀웨이 번호의 정의에 따라, 엘리먼트 *a*가 엘리먼트 *d*의 조상이면, *a*의 듀웨이 번호 $dewey_a$가 *d*의 듀웨이 번호 $dewey_d$의 프리픽스이고 그 역도 성립한다. 또한, **EncryptedData** 엘리먼트는 암호화된 후손 엘리먼트들을 포함한다[7]. 따라서, **EncryptedData** 엘리먼트 *ED*의 듀웨이 번호 *e*가 듀웨이 번호 $r \in \mathbb{R}$의 프리픽스이면, *ED*는 *r*이 가리키는 인스턴스의 조상이며 암호화된 그 인스턴스를 포함한다. □

그림 6은 질의-인식 복호화를 사용한 질의 처리 알고리즘을 나타낸다. 본 논문에서는 이 알고리즘을 *Query-Aware Decryption*이라 부른다. Query-Aware Decryption은 크게 네 단계로 구성된다. 첫번째 단계에서는 암호화된 XML 인덱스의 Key Name 필드를 사용자가 보유한 키의 집합인 *KEYS*를 사용하여 복호화한다. 여기에서, *KEYS*의 부분집합을 포함하는 Key Name 필드만이 복호화된다. 두번째 단계에서는 Key Name 필드가 복호화된 인덱스 엔트리들의 Element Type 필드를 Key Name 필드에 명시된 키를 사용하여 복호화한다. 세번째 단계에서는 Element Type 필드의 값이 질의 결과의 엘리먼트 타입들 중의 하나인 인덱스 엔트리들을 가려내어, 이들 인덱스 엔트리의 Occurrences 필



(a) 암호화되지 않은 XML 데이타 (b) 암호화된 XML 데이타

Key Name	Element Type	Occurrences	
null	*subjects*	{1}	
null	*subject*	{1.2}	
null	*name*	{1.2.1}	
{k_1}	*subject*	{1.1}	k_1으로 암호화
{k_1}	*name*	{1.1.1}	
{k_1}	*analysis*	{1.1.2}	
{k_1}	*DNAsignature*	{1.1.2.1}	
{k_1}	*tests*	{1.1.2.2}	
{k_1}	*HIV*	{1.1.2.2.1}	
{k_2}	*analysis*	{1.2.2}	k_2로 암호화
{k_2}	*DNAsignature*	{1.2.2.1}	
{k_2, k_3}	*tests*	{1.2.2.2}	k_2, k_3로 암호화
{k_2, k_3}	*SARS*	{1.2.2.2.1}	

(c) 암호화된 XML 인덱스

그림 5 암호화된 XML 인덱스의 예

드를 Key Name 필드에 명시된 키를 사용하여 복호화한다. 이 단계는 질의 결과가 될 수 있는 인스턴스들의 듀웨의 번호를 알아내는 과정이다. 이들 인스턴스는 복호화되어 질의 조건을 만족하는지 검사되어야 한다. 네 번째 단계에서는 보조정리 1에 따라 *RelevantEncryption*(*ED*, *KEYS*, *Q*) = *true*인 EncryptedData 엘리먼트만을 복호화하면서 복호화된 XML 데이타에 대하여 질의 처리를 수행한다. 복호화되는 EncryptedData 엘리먼트들은 질의 결과의 일부를 포함하고 있다.

예 4. 키 k_1, k_2, k_3를 보유한 사용자가 그림 5(b)의 암호화된 XML 데이타와 그림 5(c)의 암호화된 XML 인덱스를 전달 받아 //subject//HIV 질의를 수행한다고 가정한다. 단계 1에서 질의 처리기는 Key Name 필드를 키 k_1, k_2, k_3를 사용하여 복호화한다. 단계 2에서 질의 처리기는 subject, name, analysis, DNAsignature, tests, HIV, SARS 엘리먼트 타입의 Element Type 필드들을 키의 집합 { k_1 }, { k_2 }, { k_2, k_3 }를 사용하여 복호화한다. 단계 3에서 질의 처리기는 질의 결과의 엘리먼트 타입인 HIV 엘리먼트의 Occurrences 필드를 키의 집합 { k_1 }을 사용하여 복호화한다. HIV 엘리먼트 인스턴스는 1.1.2.2.1에 위치하므로, 보조정리 1에 따라 듀웨이 번호 1.1의 EncryptedData는 *RelevantEncryption*(*ED*, *KEYS*, *Q*) = *true*이며, 듀웨이 번호 1.2.2의 EncryptedData는 *RelevantEncryption*(*ED*, *KEYS*, *Q*) = *false*이다. 따라서, 단계 4.2에서 질의 처리기는 듀웨이 번호 1.2.2의 EncryptedData는 복호화하지 않고 질의 처리에서 제외한다. 듀웨이 번호 1.1의 EncryptedData로부터 복호화된 엘리먼트 인스턴스들은 단계 4.1에서 질의 조건을 만족하는지 검사된다. □

4. 보안 누출(Security Compromise) 분석

보안 누출이란 사용자가 알고 있는 정보를 가지고 사용자가 알면 안되는 정보를 유추할 수 있는 것을 의미한다[13]. 본 논문에서 제안하는 방법은 암호화된 XML 인덱스를 배포하는데, 암호화된 XML 인덱스로 인해 이러한 인덱스 없이는 유추할 수 없었던 정보를 사용자가 추가적으로 유추할 수 있게 되는지 확인해야 한다.

본 장에서는 암호화된 XML 인덱스로 인해 추가적인 보안 누출이 없음을 정형적으로 증명한다. 제4.1절에서는 증명에 필요한 배경 지식으로서 Miklau와 Suciu가 제안한 질의-뷰 안전성(query-view security)[14] 모델을 설명한다. 제4.2절에서는 제안한 방법이 질의-뷰 안전함을 증명한다.

4.1 질의-뷰 안전성(query-view security)

Dobkin 등은 정의 3과 같이 **보안 누출**(*security compromise*)[13]을 정의하였고, Miklau와 Suciu는 정의 4와 같이 **질의-뷰 안전성**(*query-view security*)[14]을 정의하였다. 다음으로, 보안 누출과 질의-뷰 안전성과의 관계를 논한다.

정의 3. [Dobkin 등] UNKNOWN이 사용자 U가 알면 안되는 데이타 엘리먼트의 집합을 나타내고, KNOWN이 사용자 U가 알고 있는 데이타 엘리먼트의 집합을 나타낸다고 하자. 사용자 U는 연속적으로 질의 $q_1,\cdots,q_n$을 수행하여 자신의 KNOWN을 확장하려고 시도한다. KNOWN과 UNKNOWN이 서로 교차(intersect)하면 보안이 누출(*compromise*)되었다고 정의한다[13].

정의 4. [Miklau와 Suciu] 데이타베이스에서 누출되지 않아야 하는 데이타를 질의 S로 명시하고, 데이타베

Algorithm *Query Aware Decryption*
Input: (1) the encrypted XML data, (2) the encrypted XML index,
(3) a query **Q**, (4) a set of keys **K**
Output: query results
1 Decrypt "Element Type" of the encrypted XML index using the set of keys **K**.
2 Decrypt "Occurrences" of only those entries
whose "Element Type" is the same as those of the query results.
3 Execute the query **Q** against the encrypted XML data.
3.1 Search the elements satisfying the query **Q**.
3.2 If an `EncryptedData` element is encountered, evaluate *RelevantEncryption*()
by checking prefixes of the dewey numbers in the encrypted XML index.
If *RelevantEncryption*() = *true*, then decrypt the `EncryptedData` element.
Otherwise skip the `EncryptedData` element.
3.3 Repeat steps 3.1 and 3.2 until all the query results are retrieved.

그림 6 질의 처리 알고리즘 Query-Aware Decryption

이스의 뷰 $V_1,\cdots,V_n$을 공개한다고 하자. 뷰 $V_1,\cdots,V_n$만으로는 질의 S의 결과를 전혀 알아낼 수 없을 때, 질의 S는 뷰 $V_1,\cdots,V_n$에 대하여 **질의-뷰 안전**(*query-view secure*)하다고 정의한다[14].

예 5. [Miklau와 Suciu] *Employee*(*name, department, phone*) 릴레이션을 가정한다. (1) $S_2(n, p)$:- *Employee*(*n, d, p*)이고 $V_2(n, d)$:- *Employee*(*n, d, p*)와 $V_2'(d, p)$:- *Employee*(*n, d, p*)이면, V_2와 V_2'을 조인하여 name과 phone의 쌍을 알아낼 수 있으므로 질의-뷰 안전하지 않다. (2) $S_4(n)$:- *Employee*(*n*, "Shipping", *p*)이고 $V_4(n)$:- *Employee*(*n*, "Admin", *p*)이면, Admin 부서 사람의 name은 Shipping 부서 사람의 name에 관하여 아무 정보도 유출하지 않으므로 질의-뷰 안전하다. □

정의 4의 S와 $V_1,\cdots,V_n$은 각각 정의 3의 UNKNOWN과 KNOWN에 대응한다. 이때, $V_1,\cdots,V_n$만으로 S의 결과를 전혀 알아낼 수 없다면 UNKNOWN과 KNOWN은 서로 교차할 수 없으므로, 질의-뷰 안전함은 보안 누출이 발생하지 않음을 의미한다. Miklau와 Suciu[14]는 질의-뷰 안전성을 판단하기 위한 두 개의 정리를 제공한다. 즉, 정리 1과 정리 2에서 각각 선행 지식(prior knowledge)이 존재하지 않을 때와 선행 지식이 존재할 때의 질의-뷰 안전성을 판단하는 방법을 제시한다.

정리 1. [Miklau와 Suciu] 질의 S의 결과를 만들어내는데 공헌한 투플과 뷰의 집합 $\{V_i\}$의 투플을 만들어내는데 공헌한 투플이 서로 겹치지 않으면 S는 $\{V_i\}$에 대해 질의-뷰 안전하고 그 역도 성립한다.

증명: 참고문헌 [14] 참조 □

정리 2. [Miklau와 Suciu] 선행 지식 K가 주어질 때, 데이타베이스의 전체 투플의 집합 T가 다음의 두 조건을 만족하도록 T_1과 T_2로 파티션 될 수 있으면 질의 S는 뷰 V에 대해 질의-뷰 안전하고 그 역도 성립한다: (1) K는 독립적인 두 개의 지식 K_1과 K_2의 결합(conjunction)으로서 K_1은 T_1에 대한 지식이고 K_2는 T_2에 대한 지식이다 ($K = K_1 \wedge K_2$); (2) K가 주어질 때, { S의 결과를 만들어내는데 공헌한 투플 } $\subseteq T_2$ 이고 { V의 투플을 만들어내는데 공헌한 투플 } $\subseteq T_1$ 이다.

증명: 참고문헌 [14] 참조 □

그림 7은 참고문헌 [14]의 설명을 도시함으로써 정리 2의 의미를 설명한다: "데이타베이스의 전체 투플의 집합 T가 T_1과 T_2로 파티션 될 때, K도 독립적인 두 개의 지식 K_1과 K_2의 결합이 되는데, K_1은 오로지 T_1에 대한 정보만을 나타내야 하며 K_2는 오로지 T_2에 대한 정보만을 나타내야 한다. 그리고, K를 알고 있어도 S의

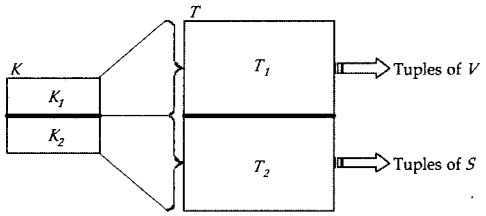


그림 7 정리 2의 의미

투플은 오직 T_2의 투플로부터 얻어진다 (T_1으로부터는 얻어지지 않는다.). 이와 유사하게, K를 알고 있어도 V의 투플은 오직 T_1의 투플로부터 얻어진다(T_2로부터는 얻어지지 않는다.)."

4.2 제안하는 방법의 질의-뷰 안전성(Query-View Security)

본 절에서는 Bertino와 Ferrari[3] 및 Miklau와 Suciu[4]의 방법에 비해 제안하는 방법이 암호화된 XML 인덱스로 인해 추가적으로 보안을 누출하지 않음을 증명한다. 이를 위해, Bertino와 Ferrari 및 Miklau와 Suciu의 방법이 질의-뷰 안전하다고 가정하면, 본 논문에서 제안하는 방법도 질의-뷰 안전함을 증명한다.

두 가지 기존의 방법이 질의-뷰 안전하다면 다음과 같은 사실이 성립해야 한다. 사용자 U가 보유하고 있는 키의 집합을 $KEY_{granted}$라고 하자. $KEY_S \not\subset KEY_{granted}$로 암호화된 엘리먼트 인스턴스들이 정의 4의 S에 해당하고, $KEY_V \subseteq KEY_{granted}$로 암호화된 엘리먼트 인스턴스들과 아예 암호화되지 않은 엘리먼트 인스턴스들의 합이 정의 4의 $V_1,\cdots,V_n$에 해당한다. 이때, 정리 1에 따라 S와 $V_1,\cdots,V_n$를 만들어내는데 공헌한 엘리먼트 인스턴스(즉, 정리 1의 투플)들은 서로 겹치지 않는다.

이제는 제안하는 방법의 질의-뷰 안전성을 나타내는 정리 3을 제시한다.

정리 3. 암호화된 XML 인덱스를 사용하지 않는 기존의 방법이 질의-뷰 안전하다면, 암호화된 XML 인덱스를 사용하는 그림 6의 Query-Aware Decryption 알고리즘은 질의-뷰 안전하다.

증명: 암호화된 XML 인덱스의 전체 엔트리의 집합을 선행 지식 K로 간주하고 정리 2를 적용하여 정리 3을 증명한다. 암호화된 XML 인덱스를 사용하지 않는 기존의 방법은 질의-뷰 안전하므로, 정리 1에 따라 XML 데이타의 전체 엘리먼트 인스턴스의 집합 T는 $KEY_V \subseteq KEY_{granted}$로 암호화된 그리고 아예 암호화되지 않은 엘리먼트 인스턴스의 집합 T_1과 $KEY_S \not\subset KEY_{granted}$로 암호화된 엘리먼트 인스턴스의 집합 T_2로 파티션된다. K와 T가 정리 2의 두 조건을 만족함을 보인다. (1) K는 $KEY_V \subseteq KEY_{granted}$로 암호화된 그리고 아예 암호화되지

않은 인덱스 엔트리의 집합 K_1과 $KEYS \not\subseteq KEY_{granted}$로 암호화된 인덱스 엔트리의 집합 K_2로 파티션된다.[1] 제 3.2절에서 논의한 바와 같이 XML 데이타의 엘리먼트 인스턴스들을 암호화하는데 사용된 키는 이들 인스턴스를 가리키는 인덱스 엔트리들을 암호화하는데 사용된 키와 동일하다. 따라서, K_1의 인덱스 엔트리들은 오직 T_1의 엘리먼트 인스턴스들에 대한 정보만을 가지고 있으며, 다른 엘리먼트 인스턴스들에 대한 정보는 가지고 있지 않다. K_2도 T_2에 대해서 마찬가지이다. 따라서, K_1과 K_2는 정리 2의 첫번째 조건을 만족한다. (2) T_2에 속하는 엘리먼트 인스턴스들은 사용자에게 보여지지 않는 엘리먼트 인스턴스의 집합 S를 구성하며, T_1에 속하는 엘리먼트 인스턴스들은 사용자에게 보여지는 엘리먼트 인스턴스의 집합 V를 구성한다. 따라서, T_1과 T_2는 정리 2의 두번째 조건을 만족한다. 그림 8은 이러한 증명을 도시한다. □

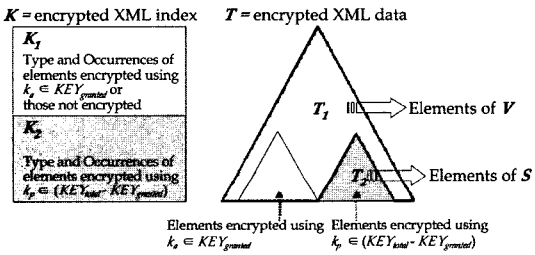


그림 8 정리 3의 증명

5. 성능 평가

본 장에서는 질의-인식 복호화를 사용한 질의 처리의 성능 평가 결과를 제시한다. 제5.1절에서는 성능 평가를 수행한 실험 데이타와 실험 환경을 소개하고, 제5.2절에서는 실험 결과를 설명한다.

5.1 실험 데이타 및 실험 환경

본 실험의 목적은 (1) 암호화된 XML 인덱스를 전송하고 복호화하는 오버헤드가 그리 크지 않음을 입증하기 위해 암호화된 XML 인덱스의 크기가 암호화된 XML 데이타의 크기에 비해 매우 작음을 보이고, (2) 질의-인식 복호화로 인해 불필요한 복호화가 방지되므로 암호화된 XML 데이타에 대한 질의 처리 성능이 크게 향상됨을 보이는 것이다. 제안하는 방법의 질의 처리 성능을 Miklau와 Suciu[4]의 방법과 비교한다.[2] Miklau와 Suciu의 방법은 자신이 보유한 키로 암호화된 EncryptedData 엘리먼트를 무조건 복호화하므로 편의상 *Unconditional Decryption*이라 부른다.

암호화된 XML 데이타에 대한 질의 처리 성능은 그림 9의 두 가지 파라미터에 의해 크게 영향을 받는다. $ratio_{encrypted}$는 얼마나 많은 엘리먼트가 암호화 되었는지를 나타내며, $ratio_{relevant}$는 얼마나 많은 EncryptedData 엘리먼트가 실제로 질의 결과를 포함하고 있는지를 나타낸다. 후손으로 EncryptedData 엘리먼트를 가지는 엘리먼트를 암호화하면 후손에 위치한 EncryptedData 엘리먼트는 숨겨지는데, 이와 같이 숨겨진 EncryptedData 엘리먼트의 개수도 파라미터의 계산에 포함한다.

본 실험에서 $ratio_{encrypted}$를 10%에서 40%까지 변화시키고, $ratio_{relevant}$를 0.1%에서 66%(주어진 $ratio_{encrypted}$ = 10%~40%에서의 최대값)까지 변화시킨다. 주어진 두 가지 파라미터 값에 따라 XML 데이타를 암호화하기 위해 다음과 같은 방식을 사용한다. 암호화되지 않은 XML 데이타에서 전체 엘리먼트의 집합을 E_{total}이라 하고, 질의 결과와 질의 결과의 조상 엘리먼트의 집합을 $E_{relevant}$라 한다. $E_{relevant}$에 포함된 엘리먼트를 암호화하면 보조정리 1에 따라 $RelevantEncryption(ED, KEYS, Q) = true$이다. 그러므로, $E_{relevant}$에 포함된 엘리먼트 중에서 암호화된 엘리먼트의 비율($ratio'_{encrypted}$)과 $E_{total} - E_{relevant}$에 포함된 엘리먼트 중에서 암호화된 엘리먼트의 비율 ($ratio''_{encrypted}$)을 조정하여 주어진 $ratio_{encrypted}$와 $ratio_{relevant}$의 값을 충족시킨다. 그림 9의 파라미터들은 수식 (1), (2)와 같이 표현된다. 수식 (1), (2)로 부터 수식 (3), (4)와 같이 $ratio'_{encrypted}$과 $ratio''_{encrypted}$을 구

$$ratio_{encrypted} = \frac{\text{the total number of EncryptedData elements in encrypted XML data}}{\text{the total number of elements in unencrypted XML data}}$$

$$ratio_{relevant} = \frac{\text{the number of EncryptedData elements satisfying } RelevantEncryption(ED, KEYS, Q)=true}{\text{the total number of EncryptedData elements}}$$

그림 9 실험에 사용되는 두 가지 파라미터

1) Miklau와 Suciu[14]는 $K = K_1 \wedge K_2$의 의미를 파티션으로 해석한다. 따라서, 본 증명에서도 K가 K_1과 K_2로 파티션됨을 보인다.

2) 본 실험에서 Bertino와 Ferrari[3]의 방법과는 비교하지 않는다. Bertino와 Ferrari는 암호화된 XML 데이타에 대한 질의 처리 방법을 제시하지 않았기 때문이다.

할 수 있다. 이때, $E_{relevant}$에 포함된 엘리먼트에서 $|E_{relevant}| \times ratio'_{encrypted}$개의 엘리먼트를 랜덤하게 선택하여 암호화하고, $E_{total} - E_{relevant}$에 포함된 엘리먼트에서 $|E_{total} - E_{relevant}| \times ratio''_{encrypted}$개의 엘리먼트를 랜덤하게 선택하여 암호화한다.

$$ratio_{encrypted} = \frac{|E_{relevant}| \times ratio'_{encrypted} + |E_{total} - E_{relevant}| \times ratio''_{encrypted}}{|E_{total}|} \quad (1)$$

$$ratio_{relevant} = \frac{|E_{relevant}| \times ratio'_{encrypted}}{|E_{total}| \times ratio_{encrypted}} \quad (2)$$

$$ratio'_{encrypted} = \frac{|E_{total}| \times ratio_{encrypted} \times ratio_{relevant}}{|E_{relevant}|} \quad (3)$$

$$ratio''_{encrypted} = \frac{|E_{total}| \times ratio_{encrypted} \times (1 - ratio_{relevant})}{|E_{total} - E_{relevant}|} \quad (4)$$

본 실험에는 합성(synthetic) 데이타인 XMark 벤치마크[15] 데이타를 사용한다. 암호화된 XML 데이타에 XPath 질의를 수행하여 질의 처리 시간을 측정한다. 이는 복호화를 위한 CPU 비용이 질의 처리의 주된 비용이기 때문이다. 수행하는 XPath 질의는 *//site//open_auctions[.//bidder]//seller*이다. 다른 다양한 질의를 사용해서도 실험을 수행하였으나, 다른 질의에 대한 실험 결과는 생략한다. 데이타 복호화 시간이 질의 처리 시간에서 매우 큰 비중을 차지하기 때문에 다른 질의를 사용한 실험 결과도 유사한 경향을 보였다.

실험은 450 MHz CPU와 512 MB 메모리를 가진 SUN Ultra 60 워크스테이션에서 수행한다. 실험 프로그램의 구현을 위해서는 XML 데이타를 암호화하고 복호화하는 기능과 XML 데이타에 대한 XPath 질의를 수행하는 기능이 필요하다. 첫번째 기능을 위해서는 Apache XML-Security[16]를 사용하고, 두번째 기능을 위해서는 Apache XALAN[17]을 사용한다. Apache XML-Security는 W3C의 XML 암호화 표준[7]을 지원하는 라이브러리이다. Apache XALAN은 메모리 상에 로드된 XML 데이타에 대한 XPath 질의를 지원하는 XSLT 처리기이다.

5.2 실험 결과

암호화된 XML 인덱스의 크기

그림 10은 1 MB의 XMark 벤치마크 데이타를 암호화하여 생성되는 암호화된 XML 데이타와 암호화된 XML 인덱스의 크기를 나타낸다. 여기서 암호화된 XML 인덱스의 크기가 암호화된 XML 데이타의 크기의 약 1/20 이내임을 알 수 있다. $ratio_{encrypted}$가 증가할수록 암호화된 XML 데이타의 크기는 증가하는 반면, 암호화된 XML 인덱스의 크기는 거의 증가하지 않는다. 이는 XML 데이타와 XML 인덱스를 암호화하는 방식의 차이에서 기인한다. XML 인덱스를 암호화하면 Key Name, Element Type, Occurrences 필드의 값이 단순히 암호화된 문자열로 대치된다. 반면, XML 데이타를 암호화하면 엘리먼트의 이름이나 내용이 암호화된 문자열로 대치될 뿐만 아니라, 암호화된 엘리먼트마다 XML 암호화 표준[7]에 따라 **EncryptedData, EncryptionMethod, KeyInfo, CipherData, EncryptionProperties** 엘리먼트의 태그가 삽입된다. 이와 같이 암호화된 엘리먼트마다 삽입되는 태그들이 크기 증가의 주요 원인이다. 그 밖에, XML 인덱스의 암호화 시간은 XML 데이타의 암호화 시간의 12.6% 이내가 소요되는 것으로 나타났다.

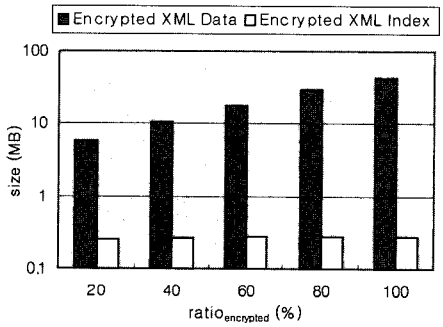


그림 10 암호화된 XML 데이타와 암호화된 XML 인덱스의 크기

질의 처리 시간

그림 11은 1 MB[3]의 XMark 벤치마크 데이타를 암호화하여 생성된 XML 데이타에 대한 *//site//open_auctions[.//bidder]//seller* 질의의 처리 시간을 나타낸다. Query-Aware Decryption의 질의 처리 시간 중에서 암호화된 XML 인덱스를 복호화하는데 소요되는 시간은 15 ms 이내로 아주 작은 비중을 차지한다. 그림 11(a), (b), (c)에서는 $ratio_{encrypted}$를 각각 10%, 20%, 40%로 고정시키고, $ratio_{relevant}$를 0.1%부터 각각 66%, 33%, 16%까지 변화시킨다. 설정 가능한 $ratio_{relevant}$의 최대값은 $|E_{total}|$ = 17,132, $|E_{relevant}|$= 1,145를 수식 (2)에 대입하여 얻을 수 있다. 또한, 암호화된 XML 데이타의 크기는 그림 11(a), (b), (c)에서 각각 3.2 MB, 5.6 MB, 11.5 MB이다.

그림 11은 $ratio_{relevant}$가 감소할수록 Query-Aware Decryption의 질의 처리 시간이 감소함을 보인다. 이는 Query-Aware Decryption이 *RelevantEncryption*(*ED*, *KEYS*, *Q*) = *true*인 **EncryptedData** 엘리먼트만을 복

3) 1 MB외에도 10 KB, 100 KB의 XMark 벤치마크 데이타를 사용하여 실험을 수행하였으며, 실험 결과의 경향은 데이타의 크기에 큰 영향을 받지 않는 것으로 나타났다. 따라서, 여기에서는 1 MB의 실험 결과만 제시하도록 한다.

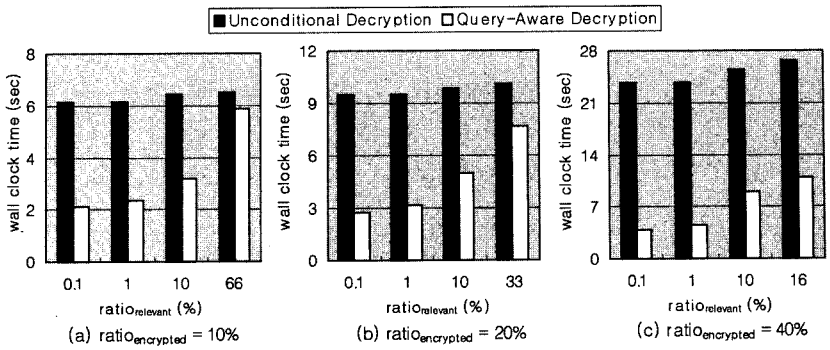


그림 11 암호화된 XML 데이타에 대한 질의 처리 시간

호화하므로 당연하다. 반면, Unconditional Decryption은 모든 **EncryptedData** 엘리먼트를 복호화하므로 질의 처리 시간이 $ratio_{relevant}$에 거의 영향을 받지 않는다. Unconditional Decryption의 약간의 질의 처리 시간 증가는 암호화된 엘리먼트의 분포에 기인한다. 높은 $ratio_{relevant}$ 값에서는 질의 결과의 조상 엘리먼트들이 대부분 암호화되는데, 이와 같이 동일한 패스의 엘리먼트들이 중첩되어 암호화되면 복호화 비용이 다소 증가한다[4]. 만약 $ratio_{relevant}$=100%이면 Query-Aware Decryption의 장점은 사라진다. 이를 확인하기 위해 $ratio_{encrypted}$를 5%로 낮추고 $ratio_{relevant}$를 66% 이상으로 증가시켜본 결과, 실제로는 *RelevantEncryption* (*ED*, *KEYS*, *Q*)의 값을 계산하는 오버헤드로 인해 $ratio_{relevant}$>90% 이면, Query-Aware Decryption이 Unconditional Decryption에 비해 좋지 않은 성능을 보인다. 하지만, 질의 결과의 개수는 XML 데이타에서의 전체 엘리먼트의 개수에 비해 일반적으로 매우 적기 때문에 $ratio_{relevant}$는 실제 환경에서 그리 크지 않으며, 따라서 Query-Aware Decryption이 효과적으로 동작한다.

그림 11은 $ratio_{encrypted}$가 증가할수록 Query-Aware Decryption과 Unconditional Decryption의 성능 차이가 증가함을 보인다. Unconditional Decryption에 비하여 Query-Aware Decryption은 그림 11(a)에서 최대 2.9배, 그림 11(b)에서 최대 3.5배, 그림 11(c)에서 최대 6.1배의 성능 향상을 보인다. 이는 $ratio_{encrypted}$가 증가할수록 Unconditional Decryption은 그에 비례하여 복호화를 더 많이 수행하는 반면, Query-Aware Decryption은 불필요한 복호화를 효과적으로 더 많이 제거하기 때문이다.

6. 결 론

본 논문에서는 질의-인식 복호화라는 개념을 제안하였다. **질의-인식 복호화**는 실제로 질의 결과를 포함하고 있는 암호화된 데이타만 가려내어 복호화하는 방법이다. 이를 위해 XML 데이타의 구조 정보를 요약한 XML 인덱스를 암호화하여 배포한다. 암호화된 XML 인덱스로 인해 암호화된 XML 데이타의 어느 위치에 질의 결과가 포함되어 있는지 알아낼 수 있다. 암호화된 XML 인덱스를 사용하여 질의 결과를 포함하고 있는 암호화된 데이타인지를 결정하는 방법을 보조정리 1에서 제시하였다. 그리고, 질의-인식 복호화를 사용하는 질의 처리 방법을 알고리즘 Query-Aware Decryption으로 구현하였다.

암호화된 XML 인덱스로 인해 추가적인 보안 누출이 없음을 정리 3에서 증명하였다. 즉, 암호화된 XML 인덱스가 존재하지 않는 기존의 방법에서 보안 누출이 없다고 가정하고, 암호화된 XML 인덱스가 존재하는 제안하는 방법에서도 여전히 보안 누출이 발생하지 않음을 증명하였다. 질의-뷰 안전성(query-view security)[14]을 보안 누출이 발생하지 않음을 판단하는 기준으로 사용하였다. Miklau와 Suciu[14]는 선행 지식이 존재할 때 질의-뷰 안전성을 판단하는 정리 2를 제시하였으며, 본 논문에서는 암호화된 XML 인덱스를 선행 지식으로 간주하고 정리 2를 적용하여 암호화된 XML 인덱스가 존재하여도 질의-뷰 안전함을 증명하였다.

Query-Aware Decryption의 질의 처리 성능에 대하여 다양한 실험을 수행하였다. 우선, 암호화된 XML 인덱스의 크기가 암호화된 XML 데이타의 크기의 약 1/20 이내로 매우 작음을 보였다. 또한, Query-Aware Decryption이 Unconditional Decryption에 비해 최대 6배까지 질의 처리 성능을 크게 향상시킴을 보였다. Query-Aware Decryption의 성능 향상은 암호화된 엘리먼트가 많을수록, 암호화된 엘리먼트에 포함된 질의 결과가 적을수록 더 커진다.

본 논문의 주요 공헌은 질의-인식 복호화를 사용하는 암호화된 XML 데이타에 대한 효율적인 질의 처리 방

법을 제안한 것이다. 본 연구는 암호화된 XML 데이타를 전송해야 하는 P2P 응용에서 유용하게 사용할 수 있을 것으로 사료된다.

참 고 문 헌

[1] Parameswaran, M., Susarla, A., and Whinston, A. B., "P2P Networking: An Information-Sharing Alternative," *IEEE Computer*, Vol. 34, No. 7, pp. 31~38, July 2001.

[2] Booth, D., Haas, H., McCabe, F., Newcomer, E., Champion, M., Ferris, C., and Orchard, D., Web Services Architecture, W3C Working Group Note, Feb. 2004.

[3] Bertino, E. and Ferrari, E., "Secure and Selective Dissemination of XML Documents," *ACM Trans. on Information and System Security*, Vol. 5, No. 3, pp. 290~331, Aug. 2002.

[4] Miklau, G. and Suciu, D., "Controlling Access to Published Data Using Cryptography," In *Proc. 29th Int'l Conf. on Very Large Data Bases*, Berlin, Germany, pp. 898~909, Sept. 2003.

[5] Ahituv, N., Lapid, Y., and Neumann, S., "Processing Encrypted Data," *Comm. of the ACM*, Vol. 30, No. 9, pp. 777~780, Sept. 1987.

[6] Song, D. X., Wagner, D., and Perrig, A., "Practical Techniques for Searches on Encrypted Data," In *Proc. 2000 IEEE Symposium on Security and Privacy*, Oakland, California, pp. 44~55, May 2000.

[7] Imamura, T., Dillaway, B., and Simon, E., XML Encryption Syntax and Processing, W3C Recommendation, Dec. 2002.

[8] Boag, S., Chamberlin, D., Fernández, M. F., Florescu, D., Robie, J., and Siméon, J., XQuery 1.0: An XML Query Language, W3C Working Draft, Nov. 2003.

[9] Clark, J. and DeRose, S., XML Path Language (XPath) Version 1.0, W3C Recommendation, Nov. 1999.

[10] Faloutsos, C., "Access Methods for Text," *ACM Computing Surveys*, Vol. 17, No. 1, pp. 49~74, Mar. 1985.

[11] Zhang, C., Naughton, J. F., DeWitt, D. J., Luo, Q., and Lohman, G. M., "On Supporting Containment Queries in Relational Database Management Systems," In *Proc. 2001 ACM SIGMOD Int'l Conf. on Management of Data*, ACM SIGMOD, Santa Barbara, California, pp. 425~436, May 2001.

[12] Chan, L. M., Comaromi, J. P., Mitchell, J. S., and Satija, M. P., *Dewey Decimal Classification: A Practical Guide*, 2nd ed., OCLC Forest Press, 1996.

[13] Dobkin, D., Jones, A. K., and Lipton, R. J., "Secure Databases: Protection Against User Influence," *ACM Trans. on Database Systems*, Vol. 4, No. 1, pp. 97~106, Mar. 1979.

[14] Miklau, G. and Suciu, D., "A Formal Analysis of Information Disclosure in Data Exchange," In *Proc. 2004 ACM SIGMOD Int'l Conf. on Management of Data*, ACM SIGMOD, Paris, France, pp. 575~586, June 2004.

[15] Schmidt, A. R., Waas, F., Kersten, M. L., Carey, M. J., Manolescu, I., and Busse, R., "XMark: A Benchmark for XML Data Management," In *Proc. 28th Int'l Conf. on Very Large Data Bases*, Hong Kong, China, pp. 974~985, Aug. 2002.

[16] Apache Software Foundation, XML-Security version 1.1, http://xml.apache.org/security/c/, 2004.

[17] Apache Software Foundation, Xalan-C++ version 1.8, http://xml.apache.org/xalan-c/, 2004.

이 재 길

1993년 3월~1997년 2월 한국과학기술원 전산학과 학사. 1997년 3월~1999년 2월 한국과학기술원 전산학과 석사 1999년 3월~2005년 2월 한국과학기술원 전자전산학과 전산학전공 박사 2001년 7월~2001년 8월 Visiting Scholar, 미국 HP Labs. 2005년 3월~현재 한국과학기술원 BK21 박사후연구원. 관심분야는 데이타베이스 프라이버시, 정보 검색, XML 데이타베이스 등

황 규 영

1973년 서울대학교 전자공학과 졸업(B.S). 1975년 한국과학기술원 전기및전자학과 졸업(M.S.). 1982년 Stanford University(M.S.). 1983년 Stanford University(Ph.D.). 1975년~1978년 국방과학연구소(ADD), 선임연구원. 1983년~1990년 IBM T.J. Watson Research Center, Research Staff Member. 1992년~1994년 한국정보과학회 데이타베이스 연구회(SIGDB) 운영위원장. 1995년 한국정보과학회 이사 겸 논문지 편집위원장. 1999년~2000년 한국정보과학회 부회장. Editor-in-Chief: The VLDB Journal, 2003년~현재 Editor: IEEE Transactions on Knowledge and Data Engineering, 2002년~현재 Editor: The VLDB Journal, 1990년~2003년 Editor: Distributed and Parallel Databases: An International Journal, 1991년~1995년 Editor: International Journal of Geographical Information Science, 1994년~현재 Associate Editor: The IEEE Data Engineering Bulletin, 1990년~1993년 1998년~2004년 Trustee, The VLDB Endowment. 1999년~2005년 Steering Committee Member, DASFAA. 1999년~현재 한국과학기술원 전자전산학과 전산학전공 교수. 1990년~현재 첨단정보기술연구센터(과학재단 우수연구센터) 소장. 관심분야는 데이타베이스 시스템, 멀티미디어, GIS