論文
5-8-6

Web컨텐츠보호를위한PMI기반의해킹방지sDBMS (sDBMS for Hacking Prevention Based on PMI for Web Contents Protection)

유두규(Du-Gyu Eyoo)[1] 전문석(Mun-Suk Jun)[2]

요 약

웹 환경의 비즈니스 모델은 대부분 디지털 데이터의 제공으로 이루어진다. 사용자와 서비스 제공자 사이의 정보 교환은 암호화된 데이터로 이루어져야 한다. 암호화된 데이터는 해킹으로부터 안전하다. DB 암호와 기술의 적용은 컨텐츠를 보호하는 주요한 기술이다. 본 논문에서 DB 암호화는 접근제어 기술을 적용하여 비인가자에 의한 컨텐츠 이용을 방지하였다. 본 논문은 PMI 기반의 역할기반접근제어와 전자서명을 이용한 새로운 DB 암호화 메커니즘을 제안하고 구현하였다.

Abstract

Business model in Web environments is usually provided by digital data. Information exchange of users and service providers should be performed by encrypted data. Encrypted data is secure from hacking. Application of DB encryption is main technology for contents protection. In this paper, We have prevented using contents by users is not accessed based on RBAC. in this paper, We propose a new DB encryption scheme which use RBAC and digital signature based on PMI.

키워드 : Digital Signature, DB encryption, RBAC, PKI, PMI

1) 정회원 : 세명컴퓨터고등학교 재직

2) 정회원 : 숭실대학교 정보과학대학 정교수

논문접수 : 2004. 7. 29.
심사완료 : 2004. 8. 13.

Ⅰ.서론

공개키 기반 구조(PKI : Public Key Infrastructure)에서 사용되는 공개키인증서(PKC : Public Key Certificate)는 사용자의 신원 확인을 위한 인증에 사용되며 사용자의 신원을 보증하는 수단으로 가장 효율적인 것으로 평가받고 있다. 그러나 공개키 인증서의 경우는 DBMS와 같이 다단계 권한이 필요한 시스템에서는 적용하기 곤란한 점이 존재한다.[1-3]

최근 역할기반 접근제어(RBAC : Role Based Access Control) 방법에 대한 연구가 많이 이루어지고 있다. 기본적 개념은 개별적인 사용자 인증 보다 권한 또는 역할이 주어지는 접근권한이다. 사용자들은 서로 다른 권한에 따라 정보 시스템내의 행위가 주어진다.[4]

접근제어를 위한 새로운 기술로 권한관리기반구조(PMI : Privilege Management Infrastructure)가 사용되어 진다. PMI의 중요한 기능은 인증이 이루어진 이후에 보다 강화된 권한을 부여하는 것이다. PMI의 데이터 구조는 X.509 속성인증서(AC : Attribute Certificate)에 명세화되어 있다. 공개키인증서가 인증서 소유자의 공개키를 보증해주는 것처럼 속성인증서도 속성인증서 소유자의 속성들에 대하여 속성기관이 보증한다. 그러나 PMI가 아직까지 널리 적용되고 있지 않다.[5-6]

최근 비인가자에 의한 전산원장 부당 변경 및 조회, 비밀번호 변경등 고객 정보 유출에 의한 예금 부당 인출등 고도의 전문기술을 이용한 보안사고가 증가하고 있어, 국가적인 차원에서 인터넷 기반의 전자문서 관리 및 데이터베이스, 인터넷 뱅킹과 같은 인터넷 비지니스에 관한 취약점을 보완하기 위한 기술적인 보완이 요구되고 있다.[7]

본 논문에서 PKI, PMI의 구조와 RBAC 모델을 적용하여 AC에 저장되어 있는 권한정책과 역할에 의하여 접근제어가 이루어지도록 하는 것을 주요 개념으로 한다. 또한 X.509 PKCs와 X.509 ACs를 연결하여 인증이 이루어진 후 권한에 따른 서비스 이용이 이루어지도록 했다. 즉 공개키 인증서의 사용자인증과 전자서명의 기능을 확장하고 AC를 이용하여 응용서비스(DB)에 접근할 수 있는 권한과 자격을 부여하여 DBMS 관리자에 의한 사용자의 비밀정보가 노출되는 것을 방지하였다. 또한 Role Authority를 이용하여 다단계 및 그룹속성을 이용하여 DB에 접근하고 데이터를 암호화 할 수 있는 프로토콜을 제안한다.

본 논문의 구성은 다음과 같다. 2장에서는 적용된 기술의 동향과 구조를 살펴보고 3장에서 본 논문에서 제안한 안전한 sDBMS의 스키마에 대하여 설명하였다. 그리고 4장에서 구현을 통한 암호학적 성능평가와 결과를, 5장에서 본 논문의 결론을 맺는다.

Ⅱ.관련연구

2.1 PMI의 구조

X.509 인증서의 확장 필드를 이용하면 기존의 공개키기반구조 시스템의 큰 변경 없이 인증, 권한을 부여하는 절차가 간소화 된다. 그러나 공개키 인증서를 사용하게 되면 다음과 같은 문제가 발생한다.

첫째, 인증서 소유자의 자격과 권한이 변할 때 공개키 인증서를 폐지해서 재 발행해야 한다. 예를 들어, 반년에 한번 인사이동이 있는 기업이나 조직의 경우를 보면 반년마다 인증서의 재발행이 필요하게 된다. 인증서 폐지 목록(CRL)도 커져 버린다. 공개키 인증서의 유효기간은 통상 1년 이상이다.

둘째, 인증기관과 등록기관은 인증서 소유자가 갖고 있는 권한에 대해서는 관계하지 않는다. 권한을 관리하는 것은 인증기관과 별도의 부서이며 조직이다. 예를 들어 기업에서의 이용을 생각하면 사원인지를 확인하는 것은 인사부지만 각 사원에 사내의 어느 정보를 이용할 수 있는지는 각 정보의 관리 책임자가 결정하는 것이다. 때문에 인증기관과 권한 부여 자는 분리할 필요가 있다.

셋째, 공개키 인증서에 자격 등의 속성 정보

를 넣으면, 공적으로 이용하기가 어려워진다. 예를 들어 통신을 할 때에 본인인 것을 확인하기 위해서 공개키 인증서를 통신 상대에게 보내는 경우 공개키 인증서에 속성 정보가 기재되어 있으면 전할 필요가 없는 속성 정보까지 상대에게 전한다.[8]

따라서 이러한 문제들을 해결하기 위해서는 사용자의 고유 식별정보를 인증하는 공개키기반구조 이외에 권한, 역할 등의 속성에 대한 관계를 보증하는 별도의 인증구조가 필요하다. 권한관리 기반구조는 이와 같이 권한 관련 자원과 소유자간의 관계를 인증기간이 인증하고 유지하는 구조를 말한다. 그림 1은 PMI 구조를 나타낸다.

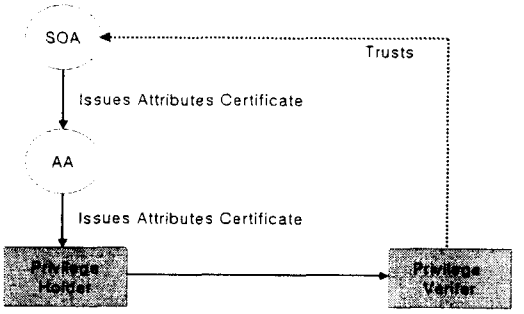

그림 1. PMI 구조

공개키 인증서는 인증기관(CA : Certification Authority)에서 사용자에 대한 신원을 확인 후 발급하는 반면, 속성인증서는 속성기관(AA : Attribute Authority)에서 발급한다. 사용자의 신원을 확인하기 위해서는 공개키인증서를 검증하고 사용자의 권한이나 역할을 확인하기 위해서는 속성인증서를 검증하면 된다. 이러한 검증과정에서 권한 검증자는 속성인증서와 공개키인증서를 연결하여 사용자가 정당한 권한을 가지고 있는지 판별하게 된다.

2.2 PMI 모델

ITU_T X.509 표준에서는 속성 인증서를 정보보호 메커니즘으로 활용할 수 있도록 일반 모델(General Model), 제어 모델(Control Model), 위임 모델(Delegation Model), 역할 모델(Roles Model)을 제시하고 있다. 여기서는 본 논문에 적용된 역할 모델에 관하여 분석한다.

2.2.1 역할 모델

역할(Roles) 모델은 사용자 개인에게 간접적으로 권한을 할당할 수 있는 기능을 제공해 준다. 이 모델에서는 개인들에게 직접 권한을 주지 않고 단지 속성 인증서의 Role 속성에 사용자에게 필요한 역할을 할당한다. Role속성을 기술한 속성 인증서는 역할할당 인증서(role assignment certificate) 또는 역할지정 인증서(role specification certificate)라 한다. 역할지정 인증서는 공개키 인증서로는 제공할 수 없으며 속성인증서를 이용해야 한다.[8]

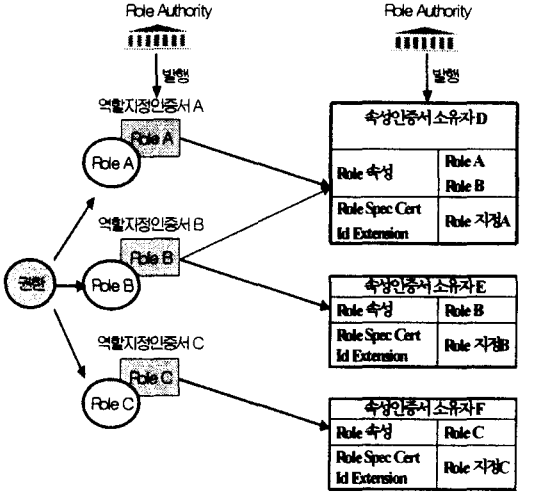

그림2. 속성인증서와 역할지정인증서

이 역할지정 인증서는 권한(역할)부여기관(role authority)이 발행한다. 그림 2는 속성인증서 소유자와 역할과의 관계를 나타낸다. 속성인증서 소유자 D는 역할지정인증서 A의 역할 Role A 와 역할지정인증서 B의 2가지 역할을 Role 속성 필드에 가지고 있다. 이 모델의 특징은 Role Based Access Control에서 큰 장점을 가진다.

2.3 DBMS의 접근제어와 암호화

데이터베이스의 보안 침해 요소를 살펴보면 침해 공격, 유추 문제, 집합 문제 등을 들 수 있다. 이러한 보안 침해 요소로부터 데이터베이스를 보호하기 위하여 여러 가지 보안 요구사항이 고려되어 왔는데 기본적인 방법으로 시스템 감사, 사용자인증, 정당한 사용자의 데이터 접근 통제 등이 있다. 논리적 일관성 유지를 위한 요구사항으로는 데이터 무결성 유지, 데이터 연산의 무결성 유지 등이 있으며, 다양한 데이터베이스 응용들을 위한 강력한 보안 요구사항으로는 추론 방지, 기밀 데이터 관리 및 보호, 다단계 보호(Multilevel protection), 접근제한(Confinement)등을 들 수 있다.[9]

데이터베이스에 대한 부당접근 및 변경은 내부 관리자에 의하여 이루어지는 경우와 외부 네트워크를 통한 두 가지 경우가 있으나 어떤 경우든 중요 정보가 누출되었을 때 심각한 문제를 야기하게 된다. 기존의 DBMS 시스템에 의한 데이터의 관리는 DBMS 엔진 자체적으로 이루어지는 권한 관리에 의하여 데이터에 대한 접근제어를 제한하였으나 이 또한 내부관리자에 의하여 정보가 누출되는 경우 막대한 영향을 초래하게 된다.

데이터베이스 암호화의 경우 데이터베이스 자체 내에 제공하는 암호함수를 이용할 수 있지만 이것은 성능 면에서 비효율적이고, 키 관리 측면에서 노출이 쉽다. 즉 데이터베이스 암호화시 키관리 부분은 데이터베이스 운영자들이 관여하지 않고 키를 자동적으로 생성하고 분배해 줄 수 있는 중앙 집중적인 키 관리 프로토콜이 존재한다면 가장 이상적이지만, 현실적으로 합당한 키 관리 솔루션이 상용화 되어 사용되지 못하고 있는 실정이다. 따라서 키 생성은 사용자 스스로 생성하고 키 관리는 제3의 기관을 통해서 키의 관리 및 사용자에 대한 인증을 수행하게 하여 이러한 사용자만이 데이터베이스에 접근하고 암호화 하게 된다면 위에서 제기한 문제를 해결할 수 있다.

Ⅲ. 제안된 DB접근제어시스템(sDBMS)

2장에서 기술한 바와 같이 기존 DBMS 시스템의 문제인 DB의 무결성, 기밀성, 사용자의 속성에 따른 접근제어 시스템을 구현하기 위해서 보안관리모듈을 설계하고 보안관리 모듈에서 사용자의 서비스 정보를 감시하도록 하여 기존 시스템이 가지고 있는 보안 취약성을 해결하였다.

3.1 sDBMS 시스템의 수행 프로토콜

3.1.1 구성요소

① 사용자(User : Certification Subject) : Web 컨텐츠를 사용하는 사용자로서 시스템 사용 시 인증이 필요한 주체

② 속성인증서 소유자(User : Attribute Certification Holder) : 인증기관으로부터 인증서를 발급받은 사용자로서 Web 응용프로그램의 사용 시 권한 인증과 역할이 필요한 접근 주체

③ 인증기관(CA : Certification Authority) : Web 환경에서 사용자와 서버에게 인증서를 발급해 주거나 관리 기능을 제공해주는 신뢰기관이다.

④ 속성기관(AA : Attribute Authority) : Web 서비스 사용자에게 권한과 역할을 부여하는 기관으로 속성인증서의 발급과 폐지 기능을 제공해 주는 회사, 조직, 기관이다.

⑤ OCSP서버(OCSP : Online Certificate Service Protocol) : 인증서와 속성인증서 상태를 온라인으로 제공하는 서버.[10]

⑥ 검증서버(VS : Verification Server) : 인증서 기반의 로그온이나 전자서명을 전송할 때 서비스 사용자의 인증서와 속성인증서를 검증하는 서버로서 보안 모듈의 접근정책에 따라 응용프로그램에 대한 접근을 허가한다.

⑦ DB 서버(AS : Application Server) : DBMS 서버 또는 응용프로그램을 제공하는 서버

3.1.2 표기

ID_A : 사용자 A의 ID

KR_a : 사용자 A의 개인키(Private Key)

KU_a : 사용자 A의 공개키(Public Key)

E_{KRa} : 사용자 A의 개인키로 암호화, 전자서명

ID_V : 검증서버의 ID

KR_v : 검증서버의 개인키

KU_v : 검증서버의 공개키

KR_{auth} : 인증기관 CA의 개인키

KU_{auth} : 인증기관 CA의 공개키

E_{KRauth} : 인증기관의 개인키로 암호화, 전자서명

D_{KUauth} : 인증기관의 공개키로 복호화

C_A : 사용자 A의 공개키 인증서

C_V : 검증서버의 공개키 인증서

T : 인증서의 유효성을 나타내는 타임스탬프

KR_{aa} : 속성기관의 개인키

$KUaa$: 속성기관의 공개키

E_{KRaa} : 속성기관의 개인키로 암호화, 전자서명

D_{KUaa} : 속성기관의 공개키로 복호화

R_A : 사용자 A의 속성 인증서

$Role_{id}$: 속성기관에 의해서 인증된 역할 속성

$Service_{id}$: 서버에 대한 서비스 요청 메시지

K_S : Role Secure Session Key

$H(SQL)$: SQL 메시지를 일방향 해쉬

$Nonce$: Replay Attack 방지를 위한 비표

$E_{Ks}(Data)$: 세션키 K_S로 비밀키(대칭키) 암호화

|| : 연접

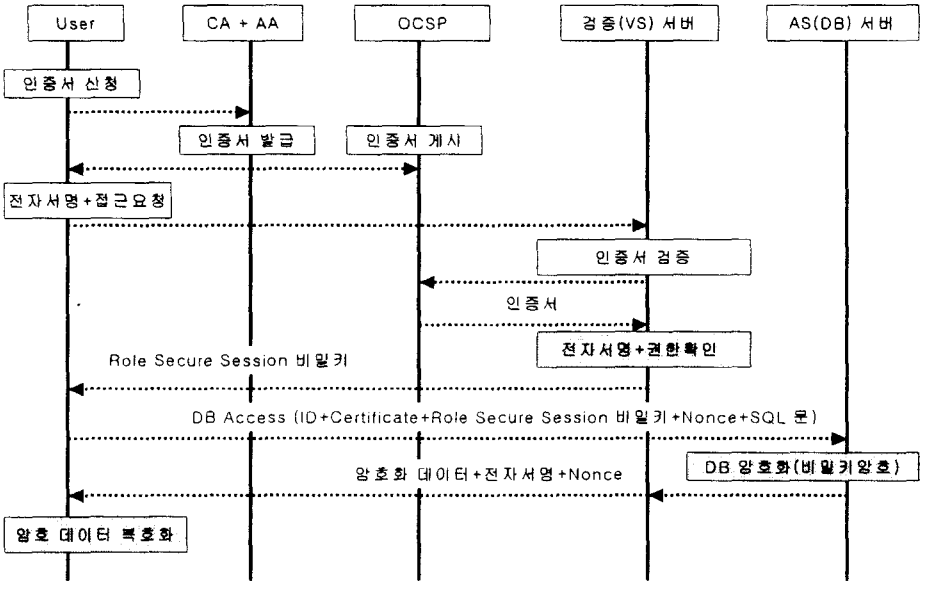

그림 3. sDBMS 시스템의 동작 수행도

3.1.2 사용자 인증

1) 사용자 A의 공개키 인증

① A -> CA : $(ID_A||KU_a)||E_{KRa}[H(ID_A||KU_a)]$

② CA -> A : $C_A||KU_{auth}$

단

$C_A = E_{KRauth}[(T_A||ID_A||KU_a)||H(T_A||ID_A||KU_a)]$

③ A : $D_{KUauth}[C_A]$

$= D_{KUauth}[E_{KRauth}(T_A||ID_A||KU_a)||H(T_A||ID_A||KU_A)]$

$= (T_A||ID_A||KU_a)$

2) 검증 서버 VS 의 공개키 인증

④ VS -> CA : $(ID_V||KU_v) = E_{KRv}[H(ID_V||KU_v)]$

⑤ CA ->AS : $C_V||KU_{auth}$

단

$C_V = E_{KRauth}[(T_V||ID_V||KU_v)||H(T_V||ID_V||KU_v)]$

⑥ VS : $D_{KUauth}[C_V]$

$= D_{KUauth}[E_{KRauth}[(T_1||ID_1||KU_v)]||H(T_1||ID_1||KU_v)]$

$= (T_V||ID_V||KU_v)$

3) 검증 서버 VS가 A의 공개키 인증서 검증

⑦ A ->VS : $(ID_A||KU_a||C_A)||E_{KRa}[H(ID_A||KU_A||C_A)]$

⑧ VS : $D_{KUauth}[C_A]$

$= D_{KUauth}[E_{KRauth}(T_A||ID_A||KU_a)]$

사용자 ID_A는 인증기관 CA에 공개키 인증서 발급을 요구하여 신청한다. 사용자 ID_A의 경우 기관은 다음 형태의 인증서를 발급한다. KR_{auth}은 인증기관에 의해 사용되어 지는 개인키이다. 사용자 ID_A는 서비스를 이용할 때 인증서를 전달하여 주며, 검증 서버는 사용자의 인증서를 읽어 다음과 같이 인증서를 확인한다. 사용자는 자신의 이름 ID_A와 공개키 KU_a를 검증 서버에게 전송한다. 타임스탬프 T_A는 인증서의 현재성이 정당함을 확인한다.[11]

인증서를 수신한 서버는 인증서를 복호하기 위하여 인증기관의 공개키 KU_{auth}를 사용한다

$D_{KUauth}[C_A] = D_{KUauth}[E_{KRauth}(T_A||ID_A||KU_a)] = (T_A||ID_A||KU_a)$.

인증서는 인증기관의 공개키를 사용하여야만 읽을 수 있기 때문에, 이것은 사용자 ID_A의 인증서 $C_A = E_{KRauth}[T_A||ID_A||KU_a]$가 인증기관이 보증하는 것임을 확인 할 수 있다.

검증 서버 ID_V의 경우도 사용자와 같은 방법으로 인증기관으로부터 서버용 인증서를 발급받는다. 검증 서버의 인증서 형태는 다음과 같다.

$C_V = E_{KRauth}[(T_V||ID_V||KU_v)||H(T_1||ID_1||KU_v)]$

인증서를 수신한 서버는 인증서를 복호하기 위하여 인증기관의 공개키 KU_{auth}를 사용한다. 인증서는 인증기관의 공개키를 사용하여야만 읽을 수 있기 때문에, 서버 ID_V에 대한 인증서가 인증기관으로부터 온 것임을 확인할 수 있다.

$D_{KUauth}[C_V] = D_{KUauth}[E_{KRauth}(T_V||ID_V||KU_v)] = (T_V||ID_V||KU_v)$

위의 경우에 사용자 ID_A 및 서버의 공개키 인증서는 각각 사용자와 서버만이 그에 해당하는 개인키를 가지고 있음을 인증기관이 증명하고 있으므로 강력한 사용자 인증이 된다.

3.1.3 권한 인증

1) 속성 인증서 신청

① A -> AA : $(ID_A||KU_a||C_A||Role_{id})||E_{KRa}[H(ID_A||KU_a||C_A||Role_{id})]$

단

$C_A = E_{KRauth}[(T_A||ID_A||KU_a)||H(T_A||ID_A||KU_a)]$

2) 속성 권한 부여와 인증

② AA ->A : $R_A||KU_{aa}$

$R_A = E_{KRaa}[(T_r||ID_A||KU_a||Role_{id})]||H(T_r||ID_A||KU_a||Role_{id})]$

3) 속성 인증서 복호화

③ A : $D_{KUaa}[R_a]$

$= D_{KUaa}[E_{KRaa}(T_r||ID_A||KU_a||Role_{id})||H(T_r||ID_A||KU_a||Role_{id})]$

$= (T_r||ID_A||KU_a||Role_{id})$

속성기관 AA는 인증기관 CA로부터 인증서를 발급받은 사용자에 한해서 속성인증서 소유자의 권한과 역할 및 응용프로그램의 서비스 종류에 따라 속성기관의 개인키 KR_{aa}로 서명하여 발행한다. 속성 인증서는 소유자의 공개키 인증서의 시리얼 번호 등으로 공개키 인증서와 연결되어 되어 있다. 즉 속성인증서는 소유자의 공개키인증서의 시리얼 번호와 발행자가 기술되고 속성인증서를 검증할 때는 공개키 인증서와 조합해서 이용한다.

$R_A = E_{KRaa}[T_r||ID_A||KU_a||Role_{id}]$

여기서 $Role_{id}$는 속성인증서 소유자의 권한 및 역할이 되며 서버들의 응용프로그램의 사용 권한이나 역할이 된다. 타임스탬프 T_r은 속성 인증서의 현재성이 정당함을 증명한다.

$D_{KUaa}[R_A] = D_{KUaa}[E_{KRaa}(T_r||ID_A||KU_a||Role_{id})]$

$= (T_r||ID_A||KU_a||Role_{id})$

속성 인증서를 수신한 검증서버는 속성인증서를 복호화하기 위하여 속성기관의 공개키 KU_{aa}를 사용한다. 속성 인증서는 속성기관의 공개키를 사용하여 읽을 수 있기 때문에 이것은 사용자 ID_A의 권한 및 역할을 속성기관이 보증하게 된다.

3.1.4 서버에 대한 접근 요청 절차

① A -> VS :

$(ID_A||KU_a||R_A||C_A||Role_{id}||Service_{id})||E_{KRa}[H(ID_A||KU_a||R_A||C_A||Role_{id}||Service_{id})]$

② VS -> A :

$M_s = (ID_A||Role_{id}||Service_{id}||ID_V||C_V)||E_{KRv}(K_S)||E_{KRv}[H(K_S)]$

단

$C_V = E_{KRauth}[(T_V||ID_V||KU_v)||H(T_V||ID_V||KU_v)]$

③ A : 메시지 수신

$M_s = (ID_A||Role_{id}||Service_{id}||ID_V||C_V)||E_{KRv}(K_S)||E_{KRv}[H(K_S)]$

$D_{KUv}[E_{KRv}(K_S)]$로 복호화, 세션키 공유

④ A -> VS :

$M_{sql} = [(ID_A||KU_A||Role_{id}||Service_{id}||C_A||R_a)||E_{KRa}(K_s||H(SQL)||Nonce||SQL)]$

서비스 신청자는 사용자 ID_A와 서비스 요청 메시지 $Service_{id}$ 공개키 인증서 C_A와 속성인증서 R_A로 이루어진 요청메시지 $(ID_A||KU_a||R_A||C_A||Role_{id}||Service_{id})$를 사용자의 개인키 KR_a로 서명 및 암호화하고 $E_{KRa}[H(ID_A||KU_a||R_A||C_A||Role_{id}||Service_{id})]$와 함께 검증서버로 전송한다.

검증 서버는 사용자의 공개키 인증서와 속성인증서를 검증하고 사용자의 공개키를 이용하여 사용자 인증은 물론 권한과 역할을 확인한다. 인증서와 속성인증서를 검증한 서버는 사용자에게 응용프로그램의 암호화에 사용할 Role Secure Session Key K_S를 서버의 개인키 KR_v로 서명하고 암호화 한 다음 서버의 공개키 인증서

$C_V = E_{KRauth}[(T_V||ID_V||KU_v)||H(T_V||ID_V||KU_v)]$

와 메시지 M_s를 함께 전송한다.

단 서버로부터의 메시지

$M_s = (ID_A||Role_{id}||Service_{id}||ID_V||C_V)||E_{KRv}(K_S)||E_{KRv}[H(K_S)]$

이다.

사용자는 서버로부터 받은 메시지 M_s를 서버의 공개키 KU_v를 이용하여 복호화한 다음 SQL 메시지를 일방향 해쉬함수를 이용하여 $H(SQL)$의 메시지 인증코드를 생성한다. 그리고 SQL 문과 Replay Attack을 방지하기 위한

Nonce를 함께 전송한다. 이때 전송메시지는 M_{sql}이 다.

$M_{sql}=(ID_A||Role_{id}||Service_{id}||C_A||R_a)||E_{KRa}[K_S||H(SQL)||Nonce||SQL)]$

3.1.5. 서버 응용프로그램 접근 및 DB 암호화

① A -> VS : $M_{sql}=(ID_A||Role_{id}||Service_{id}||C_A||R_a)||E_{KRa}[K_S||H(SQL)||Nonce||SQL)]$ $D_{KUa}[E_{KRa}(K_S)]$ 세션키 복호화, 해쉬를 통한 SQL 문 인증, SQL Injection 방지

② A -> DB : SQL Query 실행

③ DB ->VS ->A : $M_{Data}=(ID_A||Role_{id}||Service_{id}||C_V)||E_{K_S}(Data)||E_{KRv}[H(Data)||(Nonce)]$

④ A : $D_{KUv}[E_{KRv}[H(Data)||(Nonce)]]$ 데이터 복호화

접근제어의 정책에 따라 서버의 응용프로그램의 프로세스를 사용하게 되면 사용자는 웹을 통하여 고객관리서버 시스템의 DB에 접근할 수 있게 된다. DB에 접근하게 될 때는 사용자의 속성 인증서에서 명세화되어 있는 권한과 역할에 따라 DB의 사용이 허가되고 해당 프로세스를 실행할 수 있다. 컨텐츠를 관리하는 관리자는 DB에 접근하여 성적 관련 자료를 Insert, Delete, Select, Append 할 수 있다. 일반 서비스 이용자의 경우는 DB의 조회 기능에 대해서만 권한을 부여 할 수 있다.

사용자로부터 다음과 같은 메시지 $M_{sql}=(ID_A||Role_{id}||Service_{id}||C_A||R_a)||E_{KRa}[K_S||H(SQL)||Nonce||SQL)]$를 수신한 서버는 사용자의 공개키를 이용하여 세션키와 Nonce를 복호화하고 SQL문의 인증을 위하여 평문으로 전송된 SQL Query를 해쉬하여 메시지 인증코드로 전송된 $H(SQL)$와 비교하여 신뢰된 메시지인 경우 SQL Query를 DB에 전달하게 된다.

DB의 SQL Query 실행 이후에 서버는 DB에 저장된 데이터 파일을 암호화한 다음 해쉬함수를 이용하여 Data에 대한 해쉬값 $H(Data)$를 구하고 서버의 개인키 KR_v로 서명하여 전달한다.

$M_{Data}=(ID_A||Role_{id}||Service_{id}||C_V)||E_{K_S}(Data)||E_{KRv}[H(Data)||(Nonce)]$

가 전송되는 데이터로서 데이터에 대한 암호화는 이미 서버와 사용자 사이에 공유하고 있는 Role Secure Session Key 인 K_S를 이용하여 비밀키 암호 방식으로 실행한다. DB에 저장되는 파일의 크기가 큰 경우는 파일 암호화시 암호학적 연산에 많은 시간이 걸린다. 따라서 데이터의 암호화에는 암호화 속도가 빠른 비밀키 암호 방식을 사용하고 인증서, ID, 세션키, 메시지 인증코드 등의 암호화 또는 서명에는 공개키 암호를 사용하여 사용자의 인증, 전자서명, 기밀성, 무결성 등을 확보한다.

서버로부터 데이터 M_{Data}를 수신한 사용자는 서버의 공개키 KU_v를 이용하여 $Nonce$와 메시지 인증 값 $H(Data)$를 복호화하고 세션키 K_S를 이용하여 암호화된 데이터를 복호화한다. 또한 데이터를 해쉬하여 메시지 인증 값과 비교하여 신뢰된 메시지인지를 확인한다.

Ⅳ. 제안시스템의 DB암호화 구현과 분석

4.1 DB암호화 구현

① 그림 4는 개인키와 공개키를 생성을 나타낸다. DBMS에 대한 접근허가를 받게 되면 개인키와 공개키를 생성하는 과정이 실행된다. 사용자는 자신의 개인키를 이용하여 DB 데이터에 대한 전자서명을 수행할 수 있는 키를 가진다. 여기서 사용하는 개인키와 공개키는 속성인증서에 저장된 키를 불러서 사용한다.

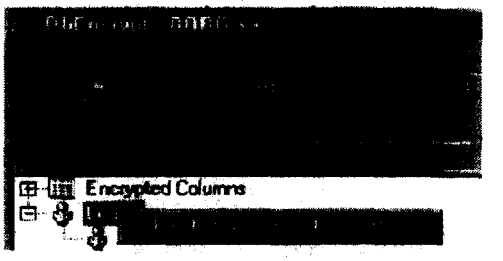

그림 4. 개인키와 공개키 쌍 생성화면

② 그림 5는 DB에 접속하여 암호하기 전의 컬럼 내용을 나타낸다. 평문의 데이터가 그대로 보인다.

fname	lname	cc	exp	
TOM	MILLS	1234123412341234		0101
BARRY	BROWN	4321432143214321		0904
SARA	JAMESON	1111111111111111		0303
JANE	DOE	2222222222222222		0501
MARK	BLOGGS	3333333333333333		0203

Transact-SQL procedure successfully completed.

그림 5. DB를 암호화 하기전의 컬럼 내용

③ 그림 6은 DB를 암호화한 후의 컬럼의 내용을 보여준다. 컬럼의 내용이 사용자의 개인키로 암호화 되어 있어 사용자의 공개키를 공개키 저장소에서 확인할 수 있으며 해당 공개키를 가진 다른 사용자들은 파일의 내용을 조회하여 볼 수 있다. 즉 속성인증서의 role 속성을 이용하여 같은 속성 그룹에 속한 사용자들은 공개키를 서로 찾을 수 있도록 하여 파일의 복호화가 가능하다.

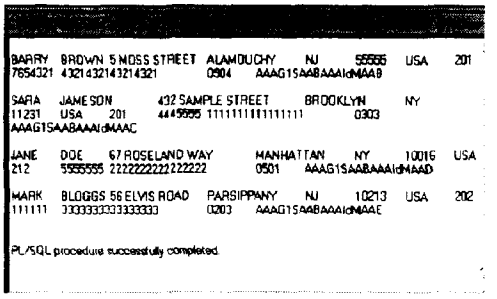

그림 6. DB 암호화 이후의 컬럼 내용

④ 그림 7은 공개키 저장트리 화면으로 같은 role 속성이 같은 그룹이 공유하는 공개키 저장소를 보여준다. 같은 업무를 수행하는 사용자들은 DB 파일에 접근할 때 다른 사용자에 의하여 전자 서명된 파일의 내용은 작성자의 공개키를 얻어서 파일의 내용을 볼 수 있다.

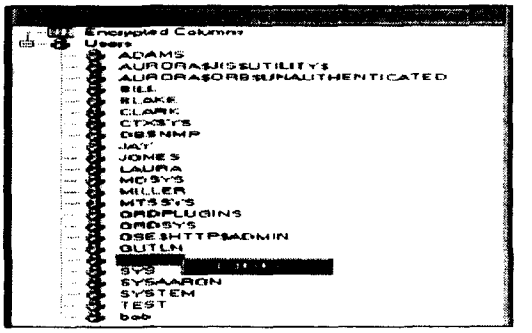

그림 7. 공개키 저장 트리 화면

4.2 제안시스템 분석

4.2.1 수행시간 비교

DB 암호화에 따른 프로그램의 수행 시간을 비교하여 제안 시스템의 성능을 분석하였다. 제안시스템의 적용결과 암호화 적용 이후 약간의 처리속도의 저하가 있음을 알 수 있다. 그러나 암호화에 따른 응답속도의 저하는 새로운 보안모듈의 도입으로 불가피한 경우라고 할 수 있다. 중요한 고객의 정보나 컨텐츠의 해킹에 따른 손실을 감안하면 미미한 정도의 성능저하라고 볼 수 있다.

표 27. DB암호화에 따른 프로그램 수행 시간 비교

구분	암호화 적용전	암호화 적용후	적용 필드
ORACLE	2.3[sec]	2.5[sec]	사용자성,사용자이름,신용카드번호
msSQL	2.4[sec]	2.6[sec]	사용자성,사용자이름,신용카드번호
mySQL	2.6[sec]	2.7[sec]	사용자성,사용자이름,신용카드번호

4.2.1 DB암호화에 따른 보안평가 분석

제안한 Secure DBMS는 기존 Web 서비스의 DB가 가진 문제를 최대한 해결할 수 있도

록 개선하였다.

① 속성인증서에 의한 역할기반 접근제어로 정당한 사용자만이 시스템에 접근할 수 있도록 하였으며, 비인가자 뿐만 아니라 시스템 관리자에 의한 DB 불법 침입을 방지하였다.

② 속성인증서의 role 속성 필드를 사용하여 인증서 발급 프로세스를 감소시켰다. 기존의 방식에서는 PKI 인증 기능만을 사용하여 서비스 이용 업무 프로세스가 발생할 때 인증서를 폐지하여 인증서폐지목록이 증가하는 경향이 많았다. 제안된 시스템에서는 속성인증서를 이용하여 인증서가 유효할 경우 폐지하지 않는다.

③ 일반적인 경우 웹 환경에서 서비스 요청 메시지 또는 응답 메시지를 평문으로 전송하여 해킹에 의한 정보의 누출 위험이 존재하였으나 제안된 시스템은 SQL Query를 서비스 요청자의 전자서명으로 암호하고 해쉬된 SQL Query와 함께 전송하여 해킹에 의한 메시지 변조가 일어나지 않도록 하였다.

④ 모든 메시지 전송에는 전자서명 프로토콜을 이용하여 메시지에 대한 부인방지 기능과, 데이터 무결성, 데이터 기밀성을 보증한다.

⑤ 전자서명 인증에 사용하는 해쉬함수는 응답

표 28. 보안평가 분석

구분	제안 시스템(sDBMS)	기존 시스템(DBMS)
인증	•전자서명에 의한 PKI 인증	•DBMS에 의한 사용자 인증
접근제어	•전자서명에 의한 PMI 인증 : 속성인증서의 Role 속성 필드 사용 •RBAC에 의한 접근제어로 시스템 관리자 및 비인가자 제어 •속성인증서의 사용으로 인증서는 폐지되지 않으므로 인증서 유효함	•ID와 패스워드에 의한 접근제어 •시스템 관리자의 접근제어 불가능 •권한 및 역할 변경시 시스템 관리자에 의한 접근 변경 되므로 고객 정보 노출됨
부인 방지	•전자 서명에 의한 부인 방지	•없음
데이터 무결성	•전자 서명과 해쉬에 의한 데이터 암호화로 데이터 무결성 확보	•없음
데이터 기밀성	•DB의 컬럼 암호화(비밀키 암호)에 의한 Secure DBMS 구현	•없음
SQL Injection	•SQL 암호화에 의한 안전한 SQL Query 전송으로 해킹으로부터 보호	•평문 전송으로 해킹에 취약
데이터 전송	•암호화된 데이터 전송	•평문 데이터 전송
검증서버	•응용서버나 서비스에 대한 권한을 검증서버에서 인증서를 통하여 검증	•Web서버에서 사용자 ID 와 패스워드만을 확인하여 사용자 검증

속도가 빠른 SHA1을 사용하여 전자서명의 무결성을 확보하였다.

⑥ 파일암호화에 사용하는 비밀키 알고리즘은 대용량 데이터의 경우에도 암호화 속도가 빠른 RC4 알고리즘을 사용하였다. 표 2는 제안 시스템(sDBMS)와 기존 시스템(DBMS)을 비교 분석했다.

V.결론

본 연구에서 구현한 sDBMS는 X.509 인증서와 속성인증서를 사용하여 기존의 공인인증 시스템을 변경하지 않고 역할기반접근제어(RBAC)를 실현하였다. 고객관리서버의 사용자 인증과 사용자의 권한을 구분하여 응용서비스에 접근토록 하여 비인가자에 의한 파일의 접근이나 변경, 삭제등과 같은 불법행위가 발생하지 않도록 하였다. 현재 많이 이용하고 있는 PKI 인증은 많은 상용 프로그램이 사용되고 있으나 속성인증서에 기반을 둔 접근제어 시스템은 활성화되지 못하고 있다. 또한 속성인증서에 의한 전자서명의 구현으로 역할에 따른 응용서비스의 사용하기 위한 권한 부여와 파일 암호화를 실행함으로써 인터넷상의 데이터 전송에서 무결성과 기밀성을 보장하였다. 또한 고객의 정보나 컨텐츠 관련 데이터의 암호화된 전송은 프로그램의 수행속도에 영향을 미치고 있으나 이것은 보안관련 모듈의 추가로 불가피한 경우라고 할 수 있다. 향후 프로그램 응답속도를 개선하는 것이 새로운 과제라고 본다.

참고문헌

[1] R. Housely, W. Ford, W. Polk, D. Solo, "Internet X.509 Public Key Infraxtructure Certificate and CRL Profile", IETF RFC2459, January 1999.

[2] C. Adams, S. Farrell, "Internet X.509 Public Key Infrastructure Certificate Protocols", IETF RFC2510, March 1999.

[3] M. Myers, C. Adams, D. Solo, D. Kemp, "Internet X.509 Certificate Request Message Format", IETF RFC2511, March 1999.

[4] D. W. Chadwick, A. Otenko, E. Ball. "Implementing Role Based Access Controls Using X.509 Attribute Certificates", IEEE Internet Computing, March-April 2003, pp. 62-69

[5] 이덕규, 이임영, "PMI를 이용한 확장 권한 위임에 관한 연구", 정보처리학회 춘계학술발표논문집, 제9권 제1호, pp947-950, March, 2002.

[6] 이승훈, 송주석, "PMI 인증서 검증 위임 및 검증 프로토콜", 정보보호학회논문지, 제13권 제1호, pp59-67, February 2003.

[7] 박지숙, "고객정보보호를 위한 DB 암호화 구현 사례", 삼성SDS, IT ERVIEW, 2003.

[8] 전문석, 유두규, 문주영, 문봉근, 엄기원, 고명선, 강정호, "정보이론 및 PKI", 미래컴, October 2003.

[9] 문봉근, 홍성식, 유황빈, "RSA 방식을 이용한 데이터베이스 암호화 구현", 통신정보보호학회논문지, 제3권 제2호, pp53-62. December 1993.

[10] M. Myers, R. Ankney, A. Malpani, S. Galperin, C. Adams, "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol", IETF RFC2560, June 1999.

[11] Adams, et al, "Internet X.509 Public Key Infrastructure Time-Stamp Protocol", IETF RFC3161, August 2001

유두규

1984년 2월 : 숭실대학교 전기공학과 학사
2001년 2월 : 숭실대학교 컴퓨터교육과 석사
2001년 3월 ~ 현재 : 숭실대학교 대학원 컴퓨터학과 박사과정
1984년 3월 ~ 현재 : 세명컴퓨터고등학교 인터넷영상 부장
관심분야 : 네트워크 보안, 정보보안, DB보안, DRM, 암호학

전문석

1980년 2월 : 숭실대학교 전자계산학과 학사
1986년 2월 : University of Maryland 전산과 석사
1989년 2월 : University of Maryland 전산과 박사
1989년 : Morgen State University 전산수학과 조교수
1989~1991년 : New Mexico State University 부설 Physical Science Lab. 책임연구원
1991년 ~ 현재 : 숭실대학교 정보과학대학 정교수
관심분야 : 네트워크 보안, 컴퓨터 알고리즘, 병렬처리, VLSI 설계, 암호학