

교무업무시스템을 위한 데이터베이스 암호화 구현 및 성능 평가

김보선[†], 홍의경[‡]

요 약

데이터베이스 암호화시스템은 데이터를 암호화하여 저장함으로 내부 관리자나 외부 해커로부터 데이터 유출 시 내용을 보호할 수 있는 중요한 수단중의 하나이다. 그러나 암호화시스템은 질의처리 시 자주 발생하는 암복호화로 성능상의 문제가 발생할 수 있어 이를 고려하여 개발되어야 한다. 본 논문에서는 데이터베이스 암호화시스템 아키텍처와 데이터 암호화 처리기를 구축하고, 암호화 데이터 처리를 위한 SQL 질의어 확장에 관한 구현 사례를 제공한다. 성능 향상을 위해 암호기는 데이터베이스별 단일키로 제한하고 암호화 알고리즘으로는 한국표준 암호화 알고리즘들 중 성능이 빠른 ARIA를 채택하였다. 데이터베이스에 대한 성능 평가 관련 연구는 현재까지 매우 미흡한 실정이다. 본 논문에서 구축한 데이터베이스 암호화시스템을 바탕으로 다양한 동시 사용자수와 서버 환경에 대해 암호문과 평문의 처리 성능을 측정한 결과를 제시한다.

Implementation and Performance Evaluation of Database Encryption for Academic Affairs System

Bo-Seon Kim[†], Eui-Kyeong Hong[‡]

ABSTRACT

Database encryption is one of the important mechanisms for prohibiting internal malicious users and outside hackers from utilizing data. Frequent occurrences of encryption and decryption cause degradation of database performance so that many factors should be considered in implementing encryption system. In this paper, we propose an architecture of database encryption system and data encryption module. In addition we suggest extended SQL in order to manage data encryption and decryption. In implementing database encryption system, we adopt ARIA encryption algorithm which is proved to be the most fast one among Korea standardized encryption algorithm. We use a single key for each database in encrypting data rather than using several keys in order to improve performance. Research over performance evaluation of database encryption system is rare up to now. Based on our implemented system, we provide performance evaluation results over various H/W platforms and compare performance differences between plain text and encrypted data.

Key words: Database Security(데이터베이스 보안), Database Encryption(데이터베이스 암호화), Encryption Algorithm(암호 알고리즘), Performance Evaluation(성능 평가), SQL Extension(질의어 확장)

* 교신저자(Corresponding Author) : 김보선, 주소 : 서울 중구 쌍립동 22-1 KERIS 빌딩 8층(100-400), 전화 : 02) 2118-1383, FAX : 02)2265-6889,
E-mail : ysoonkim@keris.or.kr
접수일 : 2007년 9월 10일, 완료일 : 2007년 11월 19일

[†] 정회원, 한국교육학술정보원 교육행정정보센터

[‡] 정회원, 서울시립대학교 컴퓨터과학부
(E-mail : ekhong@venus.uos.ac.kr)

* 본 연구는 첨단정보기술 연구센터를 통하여 과학재단의 지원을 받았음.

1. 서 론

최근 네트워크와 인터넷의 발달로 정보공유의 장점과 함께 정보 누출, 해킹 등의 문제가 발생함에 따라 정보보호에 대한 관심과 필요성이 커지게 되었다. 또한 기업이나 조직의 운영 뿐만 아니라 국가차원의 행정업무를 처리하기 위한 방대한 자료가 데이터베이스로 구축되면서 그에 따른 역기능 중 하나인 자료의 유출이 문제가 되었다.

종래의 보안은 대부분 물리적 혹은 보안전용 프로그램을 통해 여러 단계를 거치는 접근제어에 의존하는 소극적인 보안 대책으로써 여러 형태의 공격을 받을 수 있으며 특히 비밀 데이터의 내용 자체를 그대로 저장하고 있는 취약한 문제점을 갖고 있다. 따라서 데이터 암호화는 디스크 자체가 도난당하는 물리적 사고나, 시스템관리자와 같은 내부인의 악의적 접근으로부터 데이터를 보호하는 효과적인 방법 중의 하나라 할 수 있다[1,2].

암호화 이전 본래의 메시지를 평문이라 하고, 암호화된 메시지는 암호문이라 한다. 암호화는 메시지의 내용이 불명확하도록 평문을 재구성하여 암호문을 만드는 것인데, 이 때 사용하는 메시지의 재구성 방법을 암호화 알고리즘이라 한다. 암호화 알고리즘에서는 암호화의 비밀성을 높이기 위해 키를 사용하기도 한다. 복호화란 암호화의 역과정으로 불명확한 메시지로부터 본래의 메시지를 환원하는 과정이다. 암호화 기법을 적용하는 암호화 및 복호화 과정으로 구성된 시스템을 암호시스템이라 한다.

본 논문에서 다루고자 하는 데이터베이스 수준의 암호화는 DBMS에 저장되어 있는 데이터를 보호하는 방안으로 저장 매체의 도난에서부터 악의적인 의도를 가진 DBA, 개발자 혹은 사용자로부터의 데이터의 접근 및 유출을 시도하는 위험으로부터 보호할 수 있는 최적의 수단이라 할 수 있다.

데이터베이스에 대한 암호화는 데이터베이스로부터 데이터를 쓸 때 암호화하며, 읽을 때 복호화하는 것으로, 기존 데이터베이스 스키마의 변경이나 암호화 데이터에 대한 색인 데이터 관리, 키관리, SQL 질의어 등의 기능 개선이 필요하다.

데이터베이스를 암호화하는 데 있어 암호화 데이터 단위에 따라 컬럼 혹은 4KB 크기의 페이지 단위

로 암호화 하는 방법이 있다. 컬럼 단위로 암호화하는 것이 암호화할 대상 데이터의 크기가 가장 작은 것으로 좋은 방법이나 관계 데이터베이스 서버 내에 암호화 기능을 내장하여야 한다.

암호화가 수행되는 시스템 위치에 따라 로컬 암호화시스템과 전용 암호화시스템 등이 있다. 로컬 암호화시스템은 암호화가 응용프로그램이 수행되는 시스템의 CPU에서 처리되는 것을 의미한다. 이는 암호화 알고리즘이 라이브러리로 제공되면서 응용프로그램 내에서 호출하여 사용하는 경우를 예로 들 수 있고 데이터베이스 업체가 자체 암호화 프로그램을 내장하여 제공하는 경우가 그러하다. 이 모델은 구현이 용이하고, 암호화를 이용하기 위하여 별도의 하드웨어를 고려하거나 하드웨어와 통신하기 위한 인터페이스 등을 고려할 필요가 없기 때문이 구현 비용이 비교적 싸다고 할 수 있다. 그러나 키에 대한 안정성은 상대적으로 낮다고 할 수 있다. 전용 암호화시스템의 경우 암호화를 위한 별도의 전용 CPU를 두는 등 하드웨어를 설치하는 작업이 필요하다. 그러나 키 정보가 데이터와 분리되어 안전하게 관리되며, 수많은 데이터를 암복호화 하는 작업이 응용프로그램이 있는 CPU에서는 작업이 일어나지 않으므로 상대적으로 서버의 부하가 분산되는 효과가 있다.

본 논문에서는 교무업무시스템에 저장되어 있는 학생 개인 신상정보를 데이터베이스에 저장할 때 암호화하여 저장하는 데이터베이스 암호화시스템 구현사례와 성능평가를 제시한다. 2장에서는 데이터베이스 암호화시스템과 관련된 사례 연구로서 암호화 알고리즘을 살펴보고 성능 비교 실험을 통해 교무업무시스템에 적합한 알고리즘을 선택한다. 또한 상용 데이터베이스 암호화시스템의 기능을 살펴보고 교무업무시스템을 위한 기능 요소들을 제시한다. 3장에서는 데이터베이스 암호화시스템의 구현 사항을 데이터베이스 암호화시스템의 구조, 암복호화 처리 모듈, 암호화된 데이터 조회를 위한 SQL 질의어 확장 등 세 가지 영역으로 구별하여 제시한다. 4장에서는 구현 시스템에 성능 측정 실험 환경을 제시하고 암호문과 평문을 처리하는데 소요된 응답시간과 처리율을 중심으로 실험 결과를 제시한다. 끝으로 결론 및 향후 연구과제 등을 제시한다.

2. 관련 연구

데이터베이스 암호화 시스템을 설계할 때 고려해야 할 요소로 다음 세 가지를 들 수 있다[3-7]. 첫째, 데이터베이스 구조를 가능한 바꾸지 않고 암호화 기능을 활용할 수 있어야 한다. 개발 및 유지 보수 상황 시 데이터 암호화 및 키 관리 등의 기술적인 사항을 다루는 것은 평문을 다루는 시스템에 비해 비용이 많이 들며, 데이터베이스 구조의 변경에 따라 응용 프로그램이 변경될 필요가 있을 수 있다. 둘째, 데이터베이스의 부하를 고려한다. 부하란 처리 성능상의 부하와 저장소의 부하 등 두 가지를 의미한다. 암복호화 작업은 질의 처리 시 매우 빈번히 일어나는 작업이므로 데이터베이스 시스템의 성능을 좌우하는 요소가 되므로 설계 시 이를 고려해야 한다. 암호화된 데이터의 결과가 평문에 비하여 지나치게 큰 크기의 데이터가 되지 않도록 암호화 모듈 설계 시 고려해야 한다. 셋째, 암호화 알고리즘의 안정성이다. 암호화 알고리즘은 깨지기 어려운 것이어야 하고 이때 사용한 키가 유출되어 데이터가 복호화되지 않도록 키의 저장, 분배 등도 함께 고려되어야 한다.

본 절에서는 관련 연구로서 암호화 모듈에 도입하기 위한 암호 알고리즘을 살펴보고, 이들의 성능 평가 실험을 통해 비교해 본다. 또한 상용 데이터베이스의 암호화 기능을 살펴보고 시스템 구축 시 고려할 사항에 대한 분석표를 제시한다.

2.1 암호화 알고리즘

데이터베이스 암호화에 이용할 수 있는 암호 알고리즘으로 키의 특성에 따라 대칭키 혹은 비밀키 알고리즘, 공개키 알고리즘, 해시 알고리즘으로 분류할 수 있다. 데이터베이스 암호화의 경우 이름이나 주민 번호와 같이 질의 처리 시마다 암복호화가 지속적으로 발생하는 경우 속도가 빠른 대칭키 알고리즘이 적당하고, 고객의 패스워드와 같이 굳이 복호화가 필요 없는 항목인 경우에는 해시 알고리즘을 사용하는 것이 적당하다[1,8,9].

현재 국내 표준 암호화 방식으로 SEED와 ARIA가 금융 및 대국민 행정 서비스용으로 사용되고 있다. 본 절에서는 데이터 암호화를 위해 DES, SEED, ARIA 블록 암호 알고리즘에 대해 살펴보고, 성능을 비교함으로써 암호화 모듈에 적합한 알고리즘을 선

택한다.

2.1.1 DES 알고리즘

DES는 1977년 미국 정부에 의해 표준으로 지정된 블록 암호 알고리즘이다. IBM에서 최초로 발표된 후 광범위하게 연구되었으며, 세계적으로 가장 잘 알려지고 꽤 넓게 사용되고 있다. DES는 64 비트의 평문을 46 비트의 암호문으로 만드는 블록 암호 시스템으로 64 비트의 키를 사용한다. 64 비트의 키 중 56 비트만 실제의 키로 사용된다. 현재 DES는 컴퓨터 성능의 발달에 따라 보안성이 약화되어 세 개의 키로 DES를 세 번 반복함으로써 암호의 강도를 높인 3DES가 사용되고 있다[1,2].

2.1.2 SEED 알고리즘

SEED는 한국정보보호센터와 한국정보보호진흥원이 국내 암호전문가들 함께 1999년에 개발한 128비트의 대칭형키 블록 암호 알고리즘이다. 이는 민간 부분인 인터넷, 전자상거래, 무선 통신 등에서 공개 시에 민감한 영향을 미칠 수 있는 정보와 개인 프라이버시 등을 보호하기 위하여 개발되었다. SEED는 1999년 9월 정보통신단체표준으로 제정되었으며, 2005년 국제 표준화 기구인 ISO/IEC와 IETF의 국제 블록 암호 알고리즘 표준으로 제정되었다[1,2].

2.1.3 ARIA 알고리즘

국내 전자정부 구현 등으로 국가기관과 민간과의 자료 소통의 필요성이 커질 것으로 예상됨에 따라 국가보안기술연구소는 이의 정보보호를 목적으로 ARIA를 개발하였다. ARIA는 국가보안기술연구소 주도로 학계, 국가정보원 등의 암호기술 전문가들이 공동으로 개발하였고, 2004년 12월 한국 산업규격 KS 표준으로 제정되었다. ARIA는 민간 암호화 알고리즘 SEED와 함께 전자정부의 대국민 행정서비스 용으로 보급되고 있으며, 스마트카드 등의 초경량 환경 및 고성능 서버 환경 등에서 우수한 성능을 나타내었다. ARIA 알고리즘의 입출력 크기는 128 비트이고, 키 길이는 128, 192, 256 비트의 가변 길이를 지원한다. 초경량 환경에 효율적이고, 하드웨어로 구현되었을 시 더욱 성능이 빠른 것으로 알려져 있다[10-14]. 표 1은 DES, SEED, ARIA 암호 알고리즘의 특징을 비교한 것이다.

표 1. DES, SEED, ARIA 암호 알고리즘의 특징 비교

| 암호화 알고리즘 | DES | SEED | ARIA |
|----------|----------|--|---------------------|
| 입출력 크기 | 64 비트 | 128 비트 | 128 비트 |
| 키 크기 | 56 비트 | 128 비트 | 128, 192, 256 비트 |
| 라운드 수 | 16 | 16 | 12, 14, 16 |
| 구조 | 전치와 치환 | 전치와 치환 | 전치와 치환 |
| 개발 국가 | 미국 | 한국 | 한국 |
| 사용처 | 광범위하게 사용 | 전자상거래, 전자우편, 인터넷 뱅킹, 데이터 저장, 지적재산권 보호 | 전자정부시스템의 데이터 암호화 |

표 2. DES, SEED, ARIA 암호 알고리즘의 성능 비교

| 알고리즘 | DES | SEED | ARIA |
|---------------------|--|-------------------------------------|--|
| 키 길이(비트) | 64비트 | 128비트 | 128, 192, 256비트 |
| 데이터 길이(성적) | 8 바이트 | 16 바이트 | 16바이트 |
| 처리 성능 (단위 바이트/초) | 6.5배 암호화 : 4,587,365 복호화 : 4,290,905 | 1 암호화 : 698,036 복호화 : 719,100 | 1.32 배 암호화 : 923,773 복호화 : 921,401 |
| 보안성 | 다소 낮음 | 중간 | 가장 높음 |
| 정부 권고 | 지양 | 2순위 | 1순위 |

다음 표 2는 DES, SEED, ARIA 알고리즘을 대상으로 교무업무시스템에 저장되는 성적 데이터 8바이트를 기준으로 암호화, 복호화 성능 비교를 실시하였다. ARIA가 SEED보다 처리 성능 면에서 1.32배 더 우수한 것으로 나타났다. 이에 본 논문에서 구축하는 데이터베이스 암호화시스템 구축에 ARIA 알고리즘을 채택한다. DES는 속도는 빠르나 안전성 면에서 떨어지기 때문에 도입 대상에서 제외하였다.

2.2 데이터베이스 암호화 기능

데이터베이스 암호화는 데이터베이스 제품에서 제공하는 방법을 활용하는 것과 응용프로그램으로 구현하는 방법이 있다[15,16]. 본 절에서는 오라클과 MySQL에서 제공하는 데이터 암호화 기능에 대해 살펴보고[17], 교무업무시스템을 위한 데이터베이스 암호화 기능에 대해 살펴본다.

2.2.1 오라클 데이터베이스

오라클에서는 8i에서 최근 10g 버전에 이르기까지 암호화 패키지를 지속적으로 발전시켜왔다. 오라클 8i, 9i의 DBMS_OBFUSCATION_TOOLKIT, Java

Cryptographic Extension(JCE), DBMS_CRYPTO에 이르러, 2005년 오라클 10g TDE(Transparent Database Encryption)가 출시되면서 암호화 기능이 점차 개선되었다.

DBMS_OBFUSCATION_TOOLKIT 패키지는 raw, varchar2 타입에 대해 각각 암호화, 복호화할 수 있는 총 4개의 프로시저를 제공한다. 암호화 컬럼에 대해서 암호화된 데이터 값을 기반으로 색인 데이터를 생성한다. 그림 1은 ‘홍길동’을 key를 키 값을 하여 암호화 한 후 저장하는 입력 예시이다.

다음 예에서 나타난 바와 같이 암복호화 시 사용할 키 정보를 SQL 질의에 명시해야 한다. 따라서 암호화한 키 값을 분실한 경우 이를 복호화할 수 없고, 또 키 정보를 관리하는 테이블이나 데이터에 대해 접근 권한이 있는 사용자가 키 값을 이용하여 복호화 할 수 있는 위험이 존재한다. 응용프로그램 내에서 암호 함수를 호출하는 형태로 구현되어 있는 경우는 개발자가 어떤 데이터를 암복호화 하는지, 키는 어떻게 관리되는지 등의 응용프로그램 전체의 구성을 알고 있어야 하며 프로그램의 유지보수에 개발자들이 신경을 써야 하기 때문에 평문 시스템에 비하여 운영 비용이 비싸다고 할 수 있다.

```
insert into encrypt_table
values (1, CryptIT.encrypt('홍길동', 'key'));
```

그림 1. 오라클 DBMS CRYPTO를 이용한 데이터 암호화

이와 비교하여 오라클 10g에 적용된 TDE에서는 암호화 대상 데이터 필드만 선언문에서 명시하면 디스크에 해당 데이터를 기록하기 전에 자동으로 암호화를 하며 저장하도록 개발되었다. 따라서 SQL의 문법을 변경하지 않고 그대로 사용할 수 있고, 개발자는 응용프로그램을 구현하는데 있어 암복호화에 대한 사전 지식이 필요 없다. 암호화 키는 테이블별 구별하여 사용가능하고, 암호 알고리즘으로는 3DES, AES 등이 지원된다.

2.2.2 MySQL 데이터베이스 암호화시스템

MySQL 5.1 버전에서는 암호화 결과 데이터를 이진 문자열로 반환한다. 따라서 암호화 할 데이터는 char나 varchar를 사용하는 대신 blob 타입으로 선언한다. 암호화 알고리즘으로는 3DES와 AES를 지원한다[11].

암복호화 함수는 데이터와 키 정보를 매개변수로 하고, 키 정보는 생략할 수 있다. 암복호화 함수로는 AES와 3DES 암호 알고리즘에 대해 AES_Encrypt(data[], key[]), AES_Decrypt(data[], key[]), DES_Encrypt(data[], key[]), DES_Decrypt(data[], key[])] 등 총 4개를 제공한다. 만일 key 정보를 생략하는 경우 키 정보를 보관하고 있는 DES_KEY_FILE에 저장된 첫 번째 키 정보를 사용한다. 키 정보를 변경하거나 새로 생성할 때는 FLUSH DES_KEY_FILE 명령을 이용한다. DES_KEY_FILE에는 최고 9개의 키를 보관할 수 있다.

```
insert into user (id, password)
values ('홍길동', des_encrypt('암호정보','key'));
```

그림 2. MySQL에서 데이터 암호화관련 질의문 예시

키 정보를 생략하는 경우의 장점은 개발자나 사용자가 암호화 키 정보에 대한 지식이 없어도 프로그램 개발이 가능하다는 것이다. 그림 2는 ID와 password를 필드를 보유한 테이블 user에 대해서 홍길동의 암호인 '암호정보' 데이터를 key를 이용하여 3DES 알고리즘으로 암호화하는 SQL 예시이다.

다음 표 3은 상용 데이터베이스의 암호화시스템 기능을 비교하고 교무업무시스템을 위한 구성 요소를 비교 분석한 것이다. 암호키 정보는 내부자의 접근 금지 및 유출이 되지 않도록 암호화하여 저장하고 있다. 따라서 데이터별 혹은 필드별 다른 암호키를 사용하는 경우 키를 복호화하는 작업이 부가적으로 필요하고, 테이블별 다른 키를 사용하는 경우 조인 질의 처리 시 데이터를 모두 복호화한 후 조인해야 하는 부하가 예상된다. 따라서 교무업무시스템에서는 응용 프로그램의 특징을 고려하여 학교별 한 개의 단일키를 사용하도록 설계하였다. 암호키는 보안을 위해 암호화하여 저장되어야 한다. 암호화한 데이터는 복호화 시 데이터 오류 등을 검증 할 수 있어야 하며, 성능 향상을 위해 암호 컬럼에 대한 색인 기능을 지원해야 한다.

3 데이터베이스 암호화시스템 구현

본 절에서는 데이터베이스 암호화시스템 아키텍처와 암호화 처리기 모듈에 대한 구현 내용을 소개하고, 암호화된 데이터의 관리를 위한 SQL 확장에 대

표 3. 데이터베이스 암호화시스템 기능 비교 및 교무업무시스템을 위한 필요 요소 검토

| 구분 | 오라클 TDE | MySQL 5.1 | 제안 시스템 |
|-----------|-----------------------|-----------|-------------------|
| 키 생성 | 난수 키 생성 | 난수 키 생성 | 난수 키 생성 |
| 키 적용 | 테이블별 암호키 구별 | 한 개의 키 사용 | 단일 키 사용 |
| 키 저장 | 암호화 하여 저장 | - | 암호화 하여 저장 |
| 암복호화오류 탐지 | MD5 | MD5 | 필요 |
| 암호화 타입 | OBJ, ADT, LOB 외 모든 타입 | BLOB | VARCHAR, VARCHAR2 |
| 암호 알고리즘 | 3DES, AES | 3DES, AES | ARIA |
| SQL 확장 | 확장 | 확장 | 필요 |
| 암호 필드 색인 | 있음 | - | 필요 |

해 소개한다.

3.1 데이터베이스 암호화시스템 구조

교무업무시스템에서 암호화하여 저장하는 데이터는 학생 이름, 주민번호, 반복호와 석차 정보이다. 데이터베이스는 별도로 개발한 암호화 모듈과의 연동이 용이하고, SQL 질의어 확장이 용이하도록 소프트웨어 저작권을 국내에서 보유한 큐브리드 데이터베이스 관리시스템을 사용하였다. 데이터 암호기는 데이터베이스별 한 개로 사용하는 것으로 개발하였다. 데이터 암복호화 처리에 시스템 및 기존 응용 프로그램의 변경을 최소화하기 위하여 SQL 질의어에 암호화 모듈인 ENCRYPT(data) 함수와 복호화 모듈인 DECRYPT(data) 함수를 추가하였다. 암호화 데이터는 VARCHAR과 VARCHAR2 타입에 대해서 지원한다. 키정보는 데이터베이스에 저장되어 내부 관리자나 해커로부터 접근 및 해독이 불가하도록 암호화 하여 저장하고 키관리자 모듈에 의해서 추출되도록 설계하였다. 암호화 알고리즘은 2.1 절에서 제시한 성능 평가 실험 결과에 따라 처리 성능이 가장 빠른 ARIA 알고리즘을 적용하였다.

다음 그림 3은 교무업무시스템에 적용한 데이터베이스 암호화시스템의 구조를 나타낸 것이다.

데이터베이스시스템에는 교무업무관리를 위한 교무업무 DB, 입진학 DB, 보건 DB를 포함하고, 사용자 접근 제어를 위해 사용자권한 DB와 암호키 관련 정보를 포함하는 보안 DB로 구성되어 있다. 암복호화 처리를 위해 제공하는 암호기능 제공 모듈(DBGuard)에는 데이터 암복호화를 위한 암호화 처리기와 복호화 처리기, 키 정보를 추출하는 키관리자, 암복호화 모듈에 접속할 때 사용자의 암복호화 처리 권한을 확인하는 DB접속정보 인증모듈과 DB 접속정보 관리모듈로 구성되어 있다.

그림 3을 바탕으로 사용자가 교무업무시스템에 접속하여 암호화된 학생 데이터를 조회하는 과정은 다음과 같다. 사용자가 교무업무시스템에 로그인하는 과정에 사용자 인증이 완료되면 웹 서버에 저장된 서버 인증서, DB 접속정보를 인증플래그라는 자료구조에 저장한 후 큐브리드 브로커를 통해 DBMS 엔진에 접속한다. DB 접속정보에는 DB접속용 암호정보, DB 계정, DB계정의 비밀번호 정보가 포함되어 있다. CAS 프로세스는 DB 계정 및 DB 계정의

비밀번호를 확인하고 사용자권한 DB에 접속하여 사용자가 학생의 정보 조회가 가능한지 확인 한다. 학생 데이터를 조회하는 과정에 학생 이름을 복호화할 경우 다음 절차가 진행된다.

키관리자는 데이터베이스에 저장된 보안 DB로부터 암호화키와 DB접속용 암호정보를 추출하여 사용자의 접속이 유지되는 동안 암복호화 성능 향상을 위해 추출한 키정보를 캐시에 저장한다. DB 암복호화 모듈의 사용 권한을 재확인하는 절차로 DB접속 정보 인증모듈이 키관리자로부터 추출한 DB접속용 암호정보와 교무업무시스템 응용 서버로부터 전달받은 DB접속용 암호정보가 일치하는지 확인하고, 일치하면 비로소 데이터 복호화 모듈을 통해 생성된 평문 형식의 학생 이름을 데이터베이스 엔진에 반환한다.

3.2 데이터베이스 암호화 처리기 구조

그림 4는 암호화 처리기의 내부구조이다. 예를 들어, DBMS에 홍길동이라는 학생의 이름을 암호화하여 저장하는 경우 Encrypt(홍길동) 모듈을 호출한다.

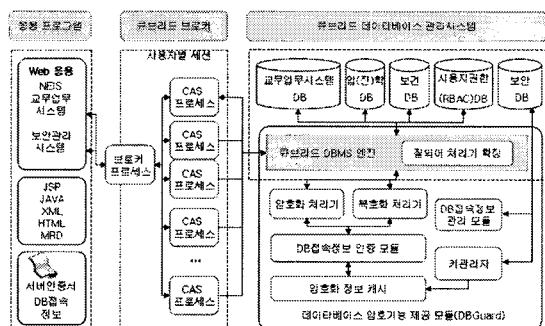


그림 3. 데이터베이스 암호화시스템 구조

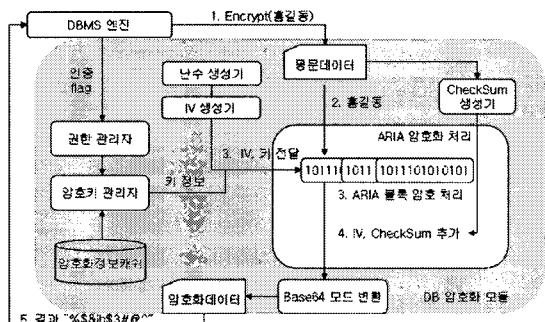


그림 4. 암호화 처리기 구조

암호화 모듈에 사용한 ARIA 알고리즘은 128비트(16바이트) 블록 단위로 평문을 나누어 128비트 키를 적용하여 데이터를 암호화 한다. 홍길동이라는 이름이 ARIA 암호화 처리 모듈로 보내지면 이진 문자열로 표현하여 블록 단위로 암호화할 준비를 하고, 키 관리자와 초기화 벡터(IV, Initial Vector) 생성기로부터 키 값과 IV 값을 전달 받는다. 블록암호 알고리즘에서는 보안성을 강화하기 위한 방법 중 하나로 이전 블록의 암호결과를 다음 블록의 입력 값으로 사용하는 CBC(Cipher Block Chaining) 모드를 사용한다. 이 때 첫 번째 블록에 사용하는 값을 입력 값으로 IV라 한다. ARIA 블록 암호 알고리즘을 통해 생성된 암호 결과에 추후 복호화 시 데이터 오류를 확인하기 위한 검증 값으로 CheckSum을 추가하고, CBC 모드에 사용한 IV 값을 추출하기 위해 암호화 결과에 IV 값을 추가한다. 이렇게 생성된 암호 데이터를 Base64 모드로 변환한다. 이는 이진 문자열을 문자열로 변환하는 것이다. 이 과정을 통해 최종 암호화 데이터가 생성되고 데이터베이스 관리시스템에 반환된다. 위 순서에 따라 암호화한 결과의 데이터 구조는 표 4와 같다.

표 5는 ARIA 알고리즘 산출 방식에 따라 암호화 전후의 데이터 크기를 나타낸 표이다. ARIA는 블록 단위로 암호화가 되므로 암호화 이후 크기는 $[16 \times (\text{TRUNC}(\text{평문 문자열 길이}/16)+1)]$ 로 산출되며 이는 128 비트의 배수가 된다. 최종 암호화 데이터의 크기는 Base64 모드를 적용하면 표 4의 암호화 결과 데이터 크기에 4/3배를 곱한 값이 된다. 따라서 $[(\text{ARIA 암호결과 크기} + \text{IV 크기} + \text{Checksum 크기}) \times 4/3] + \text{여유 바이트}(2\text{바이트})$ 를 포함하여 산출된다.

표 4. 암호화 결과 데이터 구조

| ARIA 암호화 데이터 | IV | CheckSum |
|---|------|----------|
| $16 \times (\text{TRUNC}(\text{평문 데이터 크기}/16)+1)$ | 16비트 | 16비트 |

표 5. ARIA 암호화 적용 전, 후 데이터 크기 산출

| 평문 데이터 | | ARIA 암호 적용 후 | |
|--------|----------------|--------------|----------------|
| 타입 | 크기 (단위:바이트) | 타입 | 크기 (단위:바이트) |
| CHAR | 15 | VARCHAR | 29 |
| | 16 | | 50 |
| | 31 | | 50 |

표 6. 교무업무시스템의 khkhkhshj01tt 테이블의 암호화 대상 데이터 및 크기 비교

| 한글 필드명 | 영문 필드명 | 평문 타입(크기) | 암호화 후 타입(크기) |
|-------------|--------------|--------------|-----------------|
| 학생성명_ 한글 | hak_nm | varchar(24) | varchar(50) |
| 학생성명_ 영문 | eng_hak_nm | varchar(32) | varchar(50) |
| 학생성명_ 한자 | hanja_hak_nm | varchar(24) | varchar(50) |
| 주민번호 | jumin_no | char(13) | varchar(30) |
| 반번호 | class_no | numeric(2) | varchar(30) |
| 전체 석차 | total_rank | numeric(4) | varchar(30) |

표 6은 교무업무시스템에서 암호화하는 데이터에 대해 암호화 전후 데이터 크기를 산정한 것이다. 암호화 이후 데이터 크기의 산출 방식은 16바이트보다 적으면 30바이트로 16바이트보다 크고 32바이트보다 적으면 50바이트로 설정하였다.

위 표 6의 필드 내용에 따라 데이터 타입은 모두 VARCHAR로 설정하여 khkhkhshj01tt 테이블을 생성 한 후 생성된 테이블을 바탕으로 홍길동 학생의 한글이름과 주민번호를 암호화하여 저장하는 질의는 그림 5와 같다.

그림 5의 질의가 실행되기 위해서는 그림 3의 환경 하에서 그림 4의 암호화 모듈 구조를 거친 이후 결과가 데이터베이스에 저장된다. 그림 5와 같은 질의를 통해 암호화되어 저장되어 있는 데이터를 복호화하지 않고 출력하는 예시는 그림 6과 같고, 그림 7은 해당 출력 화면을 나타낸 것이다.

```
insert into khkhkhshj01tt
values (encrypt('홍길동'), , ,
        encrypt('123458-1111111'), );
```

그림 5. 학생 데이터의 암호화 처리 저장문

```
select hak_nm, eng_hak_nm,
       hanja_hak_nm, jumin_no
from khkhkhshj01tt;
```

그림 6. 학생 데이터 조회

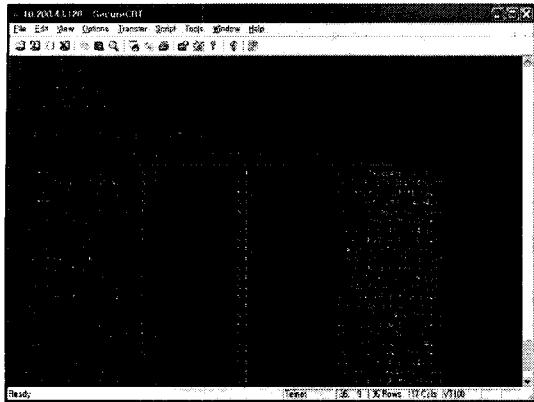


그림 7. 암호문 데이터 조회 결과

3.3 질의어 확장

암호화된 데이터의 데이터베이스 저장, 조회, 생성 등을 위하여 다음과 같이 SQL 질의어를 확장하였다. 또한 빠른 검색을 위하여 암호화된 컬럼에 대한 색인 데이터의 생성을 허용하도록 개선하였다. 이하 제시하는 질의 예는 표 6에 제시된 테이블을 기준으로 한다.

select 문을 이용하여 암호화 데이터를 조회하기 위해서는 그림 8과 같이 해당 데이터의 복호화가 먼저 선행되어야 한다. 또한 암호화 데이터는 varchar 형태로 저장되어 있으므로 성적 데이터는 복호화한 후 데이터 특성에 맞는 연산이나 데이터의 표현을 위해 to_number 함수를 이용하여 numeric 타입으로 변경한다.

조회 시는 decrypt()함수를 이용하여 해당 컬럼의 데이터를 복호화해서 사용하며, varchar 타입으로 결과가 제시되는 셋차 데이터를 활용할 때는 'select to_number(decrypt(total_rank)) from...' 예제와 같이 to_number() 함수를 사용한다. 이때 복호화 적용 컬럼에 대해 원 컬럼 명으로 별칭을 선언해야 한다.

where 조건 절에서 사용 시 주의할 점은 암호화 컬럼에 대해서 like, between <, >, <=, >= 등과 같은 비교 연산자는 사용할 수 없다는 것이다. 이와 같은 연산자는 암호화 데이터에 적용되는 것으로 처리가 불가하다. 이 경우 동등비교로 변환하여 질의를 다시 작성하거나 혹은 데이터를 복호화한 후 비교 연산이 수행되도록 질의문을 수정해야 한다. 그림 9와 같이 데이터베이스가 처리할 수 없는 오류 문은 그림 10과 같이 수정해야 한다.

```
select decrypt(hak_nm) from khhkhshj01tt;

select to_number(decrypt(total_rank)) total_rank
from khhkhshj01tt;
```

그림 8. 데이터 복호화 관련 select 예시

```
select * from khhkhshj01tt where total_rank>70 ;
```

그림 9. 데이터베이스에서 처리할 수 없는 질의

```
select * from khhkhshj01tt
where to_number(decrypt (total_rank)) > 70 ;
```

그림 10. 암호화 데이터의 비교를 위한 수정된 질의

```
insert into khhkhshj01tt(hak_nm)
values (encrypt(hak_nm));
```

그림 11. 데이터 암호화 후 저장하는 insert 질의문

insert나 update와 같은 입력 및 수정 질의에서 암호화 컬럼에 대해 데이터를 입력하거나 수정할 경우 그림 11과 같이 encrypt() 함수를 이용하여 해당 컬럼의 데이터를 암호화 한 후 처리해야 한다. 또한 암호화된 데이터를 조회한 후 다른 테이블에 입력하고자 할 경우 암호화된 데이터 자체를 복호화하고 다시 암호화 과정을 거치지 않고 암호화된 데이터를 직접 입력하는 것이 가능하다.

Order by에서 암호화된 컬럼을 사용할 때 주의할 점은 순서가 보장되지 않는 점이다. 암호화 컬럼을 order by하기 위해서는 그림 12와 같이 해당 컬럼을 select 절에 명시하여야 한다. 예문에서 'order by 2'의 '2'는 select 절의 2번째 값인 복호화한 total_rank 정보를 순서대로 결과를 제시한다는 의미이다.

```
select decrypt(hak_nm) name,
       to_number(decrypt(total_rank)) total_rank
  from khhkhshj01tt
 where class_no = 'class_01' order by 2;
```

그림 12. 암호화 데이터에 대한 order by 사용 예시

```
select * from khhkhshj01tt
where decrypt(hak_nm) = '홍길동';
hak_nm에 대한 색인 사용 불가

select * from khhkhshj01tt
where hak_nm = encrypt('홍길동');
hak_nm에 대한 색인 사용 가능
```

그림 13. 색인 사용 불가능 및 색인 사용 가능 질의 예시

빠른 검색을 위해서 암호화한 데이터에 대하여 색인정보 생성을 지원한다. 암호화 데이터에 대한 색인도 빠른 검색을 위해 활용하는데 일반 평문의 색인과 동일하다. 그림 13은 색인 사용이 불가능한 예시와 색인 사용이 가능한 예시 문이다.

4 성능 평가 및 분석

성능검증을 위한 시스템 환경은 그림 14와 같다. 인위적인 부하발생 환경 하에서 서버, 암호화 모듈의 성능을 점검하였다. 서버간의 네트워크는 테스트 대상에 포함되었으나 사용자 단말의 네트워크는 테스트 대상에서 제외되었다. 응답시간은 에이전트의 네트워크를 포함한 전체 시스템의 응답시간이며 사용자 화면에서의 화면처리를 위한 처리시간은 제외하였다. 교무업무시스템의 학기 초 업무 중 최대 부하를 유발하는 학생생활기록부 조회 프로그램을 대상으로 가상 동시 사용자 수를 48, 96, 192명으로 설정한 환경에서 테스트를 실시하였다. 동시 사용자 수는 실제 운영 환경에서 학교서버에 접속하여 이용할 수 있는 사용자 수를 고려하여 설정하였다. 부하 생성 및 성능 테스트에 Mercury Load Runner 8.0을 활용하였고 가상 부하 발생을 위해서 에이전트 서버에 Rational Suite Enterprise를 설치하였다. 큐브리드 DBMS 서버의 경우 SUN 서버에는 AMD Opteron 2.2Ghz×2, 4G 메모리를 설치한 경우와, AMD Opteron 2.2Ghz×4, 8G 메모리로 변경하면서 실험하였고, HP 서버를 사용한 경우 Intel Itanium2 1.3Ghz×4, 8G 메모리를 설치하여 실험하였다. WEB/WAS에는 WebtoB 3.0과 Jeus 4.2버전을, 큐브리드 DBMS 서버에는 UniSQL 6.3버전을 실험에 이용하였다.

다음 표 7은 성능 측정 실험을 위한 시나리오이다.

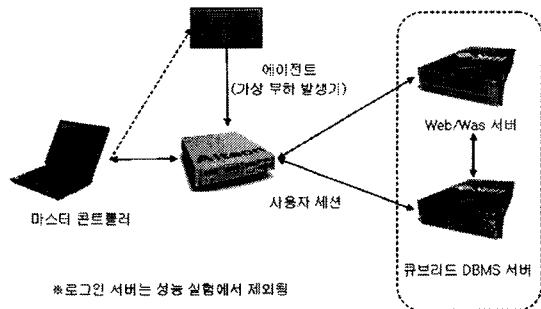


그림 14. 성능 평가를 위한 실험 환경

표 7. 성능 측정 실험을 위한 시나리오

| 메뉴 명 | 페이지 명 | 단계 | 실험 방법 |
|-----------------|-----------------------|---------------------------------------|-----------|
| 생활 기록부 조회 | 로그인 메뉴 이동 학생 선택 | 초기화 | |
| 생활 기록부 조회 | 생활기록부 조회 | 반복 수행 동시 사용자 수: 48, 96, 192명 | 5분간 부하 발생 |

교무업무시스템의 기능 중 부하가 큰 생활기록부 조회 업무를 600명의 학생 정보를 순차적으로 조회하여 데이터베이스의 캐시에 의한 성능향상을 최소화하도록 실험 환경을 구성하였다.

그림 15의 결과에 따라 암호화 처리는 평문과 비교하여 응답시간과 시스템 성능처리율의 성능에 큰 영향을 미치지 않음을 알 수 있다. 시스템의 성능에는 CPU 개수와 메모리 크기가 성능에 영향을 미치는 것으로 나타났다. 48, 96, 192명의 동시 사용자에 대해서 2개의 CPU가 장착된 SUN 서버에서는 평문과 암호 데이터 처리의 평균 응답 시간 차이가 0.22로 나타났고, 4개의 CPU가 장착된 SUN 서버에서는 0.08로 나타났다.

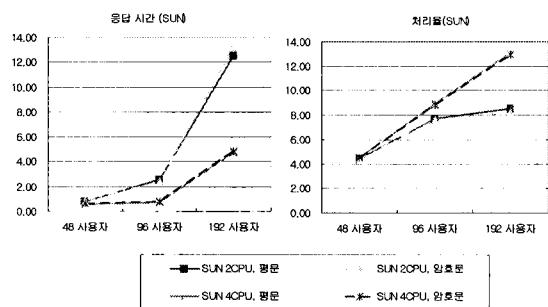


그림 15. SUN 서버에서의 암호화 및 평문에 대한 성능 평가

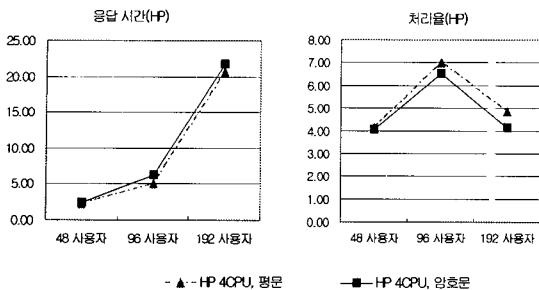


그림 16. HP 서버에서의 암호문 및 평문에 대한 성능 평가

그림 16의 4개 CPU와 8G 메모리를 장착한 HP 서버의 경우 암호문과 평문에 대한 처리 시간 차이가 48, 96, 192명의 동시 사용자에 대해 평균 0.763의 차이를 보였다. SUN 서버의 경우 4개의 CPU 설치 서버에 대해서는 192명의 동시 사용자에 대해서도 처리율이 계속 증가함을 보였으나 HP 서버에서는 96명의 동시 사용자가 적절한 처리 수준임을 실험 결과에서 알 수 있다. 그림 15의 4개 CPU가 설치된 SUN 서버에서는 192명의 동시 사용자에 대해서 평문인 경우 13.10의 처리율을 보였으나 암호문에 대해서는 12.93의 처리율을 보인다. 그림 16의 HP 서버의 경우 동시 사용자 수가 96명일 때가 응답시간 및 처리율에 있어서 적정한 수준을 나타내고 있고 평문에 대해 7.03, 암호문에 대해서는 6.51의 처리율을 나타낸다.

그림 17, 18은 리눅스와 유닉스 운영체제에서 각각의 질의에 대해 평문과 암호문의 처리 시간 차이를 나타낸 것이다. 실험 환경은 4개의 CPU와 8G의 메모리를 설치한 서버를 기준으로 실행한 것이다. 동일한 질의 실행에 대해서 리눅스와 유닉스 상에서의 처리 시간에 대한 패턴은 크게 다르지 않음을 알 수 있다.

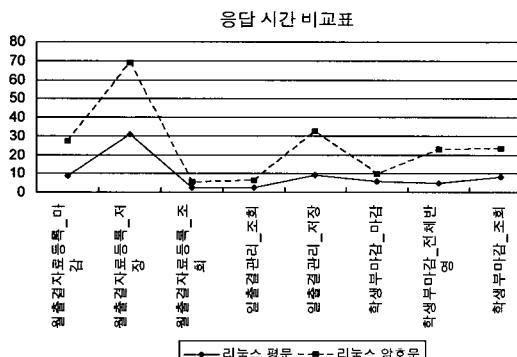


그림 17. 리눅스 상에서의 암호문 및 평문 처리 성능

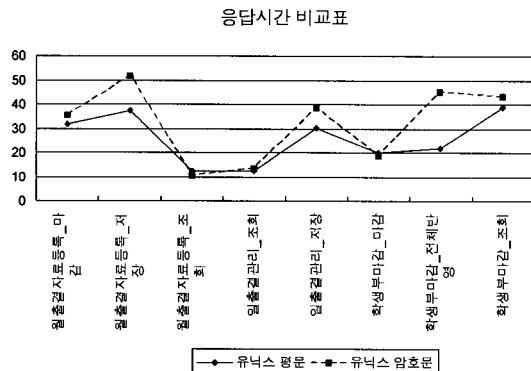


그림 18. 유닉스 상에서의 암호문 및 평문 처리 성능

월출결자료의 마감, 저장, 조회 기능과 일출결 자료의 조회, 저장, 학생부 마감자료의 마감, 전체 반영, 조회에 대한 질의 처리 결과 저장과 관련된 질의 어처리에서 평문보다 암호문의 응답시간이 더 길고, 조회와 관련된 질의 어처리는 인덱스 활용 등을 통해 암호문이나 평문에 대해서 큰 차이가 없는 것으로 나타났다.

큐브리드 데이터베이스 운영을 통한 교무업무 시스템에 적합한 환경을 찾기 위해서 스레드의 개수와 큐브리드 CAS 프로세스 풀의 개수에 변화를 주고 성능을 측정하였다. CAS 프로세스는 사용자 세션을 DBMS 엔진을 연계해 주는 모듈이다. 그림 19에서와 같이 HP 서버에 대해서 스레드는 30개 일 때 CAS 프로세스 풀이 60개 일 때가 가장 효율적인 성능을 나타낼 수 있다. SUN 서버에서는 CAS 프로세스의 개수에 대해서 커다란 영향을 받지 않는 것으로 보인다. 그 외 여러 차례의 실험을 통하여 가장 적정 성능을 나타내는 서비스 환경으로는 스레드가 15~30개 사이이고, CAS 프로세스 풀은 50개 정도인 것으로 나타났다. HP와 SUN 서버에 4개의 CPU와 8G 메모리를 설치한 경우 96명의 동시 사용

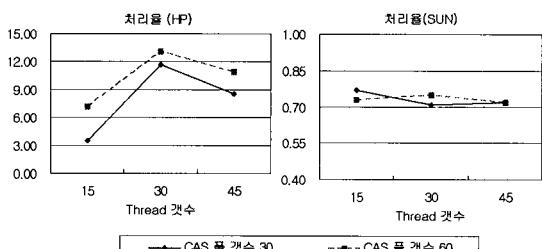


그림 19. 스레드와 큐브리드 CAS 풀의 개수에 따른 성능 변화

자에 대해서 교육행정정보시스템은 사용자에게 만족할 만한 서비스를 제공할 수 있다고 판단한다.

5. 결론 및 제언

본 논문에서는 교무업무시스템에 저장되어 있는 학생 개인정보를 암호화하기 위하여 구축한 데이터베이스 암호화시스템 구현 사례를 제시하고, 평문과 암호문의 처리 성능을 측정하고 그 결과를 제시하였다. 데이터베이스 암호화시스템은 질의 처리 시 매번 암복호화가 필수적으로 발생하므로 성능을 특히 고려하여 개발하여야 한다. 암호키를 테이블 별로 혹은 필드 단위로 하거나 레코드 단위로 한다면 성능상의 큰 부하를 가져올 것이다. 따라서 본 시스템을 구축하는데 있어 암호키는 학교별 한 개로 설정하고, 키를 캐시에 저장하여 성능상의 차이를 최소화 하였다. 암호 알고리즘의 선택 시 국내표준 암호 알고리즘 중 성능 평가 실험결과에서 가장 빠른 ARIA 알고리즘을 선택하였다.

생활기록부 조회와 같이 여러 개의 테이블을 조인하는 질의 처리 성능을 측정한 결과 192명 동시 사용자에 대해 CPU 2개인 경우 평문과 암호문이 12.53과 13.23으로, CPU 4개인 경우 4.70과 4.83의 결과를 얻었다. 따라서 암호문과 평문의 성능차이가 그다지 크지 않음을 알 수 있다. 그 외에 수차례의 실험을 통해 스레드는 15개에서 30개로 설정하고, CAS 프로세스는 50개 정도를 유지하며, DB 서버는 CPU를 4개로 하고, 8G의 메모리를 장착한 경우, 90여 명의 동시 사용자에 대해 안정적인 서비스를 제공할 수 있다고 판단한다.

향후 암호화 데이터의 보안 향상을 위해 레코드 별, 혹은 테이블별 키를 구별하여 사용하는 방법을 연구하고, 이에 대한 성능 향상에 관한 후속 연구가 필요하다. 또한 보안성 향상을 위해 암호화 키를 변경할 수 있는 기능과 암호화 알고리즘을 변경 할 수 있는 부가 기능 등에 대한 추가 연구가 필요하다.

참 고 문 헌

- [1] 남길현, “암호 시스템의 특성과 활용,” 정보과학 회지, 제7권 제5호, pp. 55-64, 1989.
- [2] 정진욱, “암호학 개요,” 통신정보보호학회지, 창

간호 pp. 29-44, 1991.

- [3] A. Ceselli, E. Damiani, S. D. C. Vimercati, S. Jajodia, S. Paraboschi, and P. Samarati, “Modeling and Assessing Inference Exposure in Encrypted Databases,” *ACM Transactions on Information and System Security (TISSEC)*, Vol.8, Issue 1. pp. 119–152, Feb. 2005.
- [4] H. Hacigum, B. Iyer, C. Li, and S. Mehrotra, “Research Sessions: Potpourri: Executing SQL over Encrypted Data in the Database-Service-Provider Model,” *Proc. of ACM SIGMOD*, Intl. Conf. on Mgmt. of Data SIGMOD, pp. 216-227, June 2002.
- [5] Y.S. Kim and E.K. Hong, “A Study of UniSQL Encryption System: Case Study of Developing SAMS,” *Proc. 9th ICACT Conf.*, Vol.1, pp. 577-582, Phoenix Park, Korea, Feb. 2007.
- [6] U.T. Mattsson, *Best Practice for Database Encryption Solutions*, Pipeline News Letter, Quest Software, Nov. 2005.
- [7] G.I. Davida, D.L. Wells, and J.B. Kam, “A Database Encryption System with Subkeys,” *ACM Transactions on Database Systems*, Vol.6, No.2, pp. 312-328, June 2005.
- [8] A.J. Menezes, P.C. Oorschot, and S.A. Vanstone, *Handbook of Applied Cryptograph*, CRC Press, 1999.
- [9] C.P. Pfleeger, *Security in Computing Second Edition*, Prentice Hall, USA, 1997.
- [10] 128bit block Encryption Algorithm ARIA, 국 가표준정보센터, 서울, 2004.
- [11] ARIA Specification Version 1.0, NSRI, Korea, May 2004.
- [12] D.G. Kwon, Kim, S.W. Park, S.H. Sung, Y.K. Sohn, J.H. Song, Y.G. Yeom, E.J. Yoon, S.J. Lee, J.W. Lee2, S.T. Chee, D.W. Han1, and J. Hong, “New Block Cipher: ARIA,” 2003, <http://www.nsri.re.kr/ARIA/doc/ARIA-ICIS C2003.pdf>.
- [13] 노경호, “ARIA 알고리즘용 암호 프로세서의 설계,” 한양대학교 석사학위논문, pp. 9-20, 2006.

- [14] A. Biryukov, C.D. Canni'ere, J. Lano, S.B. Ors, and B. Preneel, *Security and Performance Analysis of Aria*, Dept. Electrical Engineering-ESAT/SCD-COSIC, Katholieke Universiteit, 2003.
- [15] *An Oracle Technical White Paper Database Security in Oracle8i*, Oracle Corp., USA, 1999.
- [16] P. Needham, *Oracle Advanced Security Technical White Paper*, Oracle Corp., USA, June 2007.
- [17] *MySQL 5.1 Reference Manual*, MySQL AB, USA, 2005., <http://mysql.com/>.



홍 경

1981년 서울대학교 사범대학 수학교육과 졸업(B.S.)
1983년 한국과학기술원 전산학과 졸업(M.S.)
1991년 한국과학기술원 전산학과 졸업(Ph.D.)
1984년 ~ 현재 서울시립대학교 컴퓨터과학부 교수
2006년 9월 ~ 현재 데이터베이스 소사이어티 회장
2007년 1월 ~ 현재 한국정보과학회 총무 부회장
관심분야 : 데이터베이스, XML, 데이터 마이닝, 분산데이터베이스



김 보 선

1996년 덕성여자대학교 전산학과 졸업(B.S.)
1998년 서울시립대학교 컴퓨터통계학과 졸업(M.S.)
1998년 ~ 1999년 한국학술진흥재단부속 첨단학술정보센터 연구원
2000년 일본 NIME 연구소 연구원
1999년 ~ 현재 한국교육학술정보원 선임연구원
관심분야 : 데이터베이스 암호화, 사용자 접근제어