情報保護學會論文誌
第 22 卷 第 6 號, 2012. 12

데이터 암호화에 따른 성능 실증 사례 연구

홍 정 화,† 정 익 래‡
고려대학교

A case study on the performance of encrypted data

Jung Hwa Hong,† Ik Rae Jeong‡
Korea University

요 약

최근에 개인정보 유·노출에 의한 프라이버시 침해로 인해 개인정보보호의 중요성은 나날이 증가하고 있으며, 이를 해소하기 위한 가장 효과적인 방안중의 하나로 데이터 암호화가 회자되고 있다. 그러나 현재의 데이터 암호화 방식은 시스템 성능 지연과 어플리케이션 변경 이슈로 인하여 실무에 보편화되지 못하고 있는 실정이다. 따라서 본 논문에서는 최근에 화두가 되고 있는 데이터 암호화 관련 기술을 살펴보고, 어플리케이션 변경이 필요치 않는 Plug-In 방식의 데이터 암호화 솔루션으로 오라클 데이터베이스를 암호화할 때 성능에 미치는 영향을 실험하였다. 아울러 데이터 암호화시 데이터베이스 성능 저하 및 어플리케이션 변경을 최소화하는 데이터 암호화 기법을 제안하고 제안된 기법의 효율성을 분석하였다.

ABSTRACT

The importance of protecting personal information is increasing day by day due to invasion of privacy, and data encryption is the most effective way to eliminate it. However, current data encryption methods tend to having problems for applying in practical fields because of critical issues such as low performances and frequent changes of applications. In order to find proper solutions for data security, this paper reviews data encryption technologies and experiments on performance of encrypted data in Oracle Database. On top of that, this paper analyses a data encryption technique not only efficiency of performance but also minimization of application changes.

Keywords: performance of encrypted data

I. 서 론

현대 사회는 인터넷을 이용한 비즈니스가 활성화됨에 따라 인터넷상에서 개인정보를 손쉽게 획득할 수 있게 되었다. 이에 따라 개인정보보호의 중요성이 더욱 강조되고 있으나 개인정보보호를 위한 각종 규제들이 개별 법률에 의한 상이한 정보보호 처리기준으로 혼란을 겪어왔다. 다행히도 '11.9월 말 시행된 개인정보보호법에서는 공공행정, 정보통신, 금융 및 신용정보 등 비즈니스 분야에 관계없이 국제수준에 부합하는 개인정보 처리 원칙을 단일화하여 제시하고 있으며, 특히 주민등록번호 등 고유식별정보가 변조·유출 또는 도용되지 않도록 암호화 등 안전성 확보에 필요한 조치를 취하여야 한다고 명시하고 있어 데이터 보호를 위한 활동이 증가할 것으로 예상된다.

그러나 데이터 암호화 방식이 시스템 성능을 저하시키며, 데이터를 효율적으로 검색하기 위해 설치된 인덱스에도 암호화가 걸리면서 프로세스가 느려지는

* 접수일(2011년 8월 16일), 수정일(1차: 2011년 12월 12일, 2차: 2012년 7월 6일), 게재확정일(2012년 12월 6일)
† 주저자, hjh9340@hotmail.com
‡ 교신저자, lrjeong@korea.ac.kr

문제가 발생[1]~[5]하여 실제 환경에 적용하고 있지 못한 것이 현실이다. 데이터 암호화시 성능 저하 이슈는 여전히 기술적으로 해결되지 않아서 개인정보보호법 준수와 비즈니스 성능 사이에서 기업의 고민은 증가할 것이다.

따라서 본 논문에서는 현재 시장에 출시되어 있는 데이터 암호화 솔루션을 이용하여 데이터 암호화시에 발생하는 성능 이슈에 대해서 검토하고, 데이터베이스 성능 저하 및 어플리케이션 변경을 최소화하는 암호화 기법을 실증하고자 한다.

II. 개인정보 데이터 암호화 관련 연구

2.1 개인정보 보호 필요성

2.1.1 개인정보 중요성

현대 정보사회에서 개인정보의 노출은 개인 프라이버시 침해와 직결된다. 개인정보가 유출되는 경우 개인은 생명 및 신체에 대한 위협, 재산상의 손실, 사회적 평가의 저하 등 다양한 불이익을 받을 수 있다. 사이버스토킹[1]에 의한 지속·반복적 욕설, 협박 등에 의한 생명·신체 위협은 재산상 이익을 취하기 위한 불건전한 의도를 가진 익명의 이용자에 의해 행해질 수 있고, 피해자는 본인이 의도하지 않게 자신의 개인 홈페이지나 블로그 등에 자신에 대한 사회적 평가에 부정적 영향을 줄 수 있는 게시글, 댓글로 피해를 받을 수 있다. 개인정보 유출로 인한 온라인에서의 피해는 일단 발생한 경우 회복이 어려울 뿐만 아니라 피해가 복구되기 전에 다른 게시판이나 서비스 등으로 글이 전달됨으로써 2차 피해가 실시간으로 발생할 수 있으며, 개인정보의 노출 근원을 확인하는데 많은 자원이 소요된다. 또한 기업은 관리 부주의로 인하여 개인정보를 유·노출하였을 경우 브랜드 가치의 하락, 평판의 악화는 물론이거니와 악의적인 단체, 개인의 협박이나 이용자의 손해배상청구 등으로 인하여 직·간접적인 재정상의 손해를 감수해야 한다[6]. 이와 같이 개인정보의 유·노출은 단순히 개인의 정보가 공개되어 프라이버시가 침해될 수 있다는 측면뿐만 아니라 개인과 기업의 정신적 피해와 직·간접적인 경제적 손실이 수반되므로 개인정보는 안전하게 보호, 관리되어야 한다.

1) 이동통신, 이메일, 대화방, 게시판 등의 정보통신망을 이용해 의도와 악의를 가지고 지속적으로 공포감, 불안감 등을 유발하는 행위

2.1.2 개인정보 유출증가

'08~'10.6월까지 건강보험관리공단의 23,468명 개인 정보 유출 사례에 관한 국회의원 보도자료와 '08.2월 옥션 1,081만명 개인회원 정보 유출, '11.4월 현대캐피탈 고객 42만명 개인정보 유출 사례에서 보는 바와 같이 내부 직원 및 해킹에 의한 개인정보 유출 사건이 해마다 증가되고 있다. 개인 정보를 관리하고 있는 기업들은 기업의 신뢰도 하락과 경제적 손실을 방지하기 위해서 개인정보 유출 사고가 발생해도 신고하지 않아 정확한 집계가 불가하나, 한국인터넷진흥원에서 발표한 개인정보 침해 건수로 유추해보면 '00년 2,035건에서 '09년 35,167건으로 최근 10년간 개인정보 유출 사고가 상당히 증가되었음을 알 수 있다. 또한 '06~'09년까지 개인정보 침해 건수를 유형별 상담건수로 상세 분류해보면 본인이 인지하지 못한 상황에서 정보가 유·노출되는 상황이 전체 상담건수의 50% 이상을 차지하고 있어서 개인정보 유·노출 상황이 심각함을 알 수 있다. 게다가 오늘날의 개인정보는 사회 각 분야에서 인터넷과 정보통신기술의 사용이 일상화되면서 과거의 단순한 신분 확인용 정보에서 오늘날에는 기업의 수익창출을 위한 자산적 가치로 높게 평가되고 있다. 따라서 RFID(Radio Frequency IDentification), 스마트폰 등 새롭게 등장한 정보통신기술에 의한 개인 정보 침해는 급속히 증가할 것으로 예상된다.

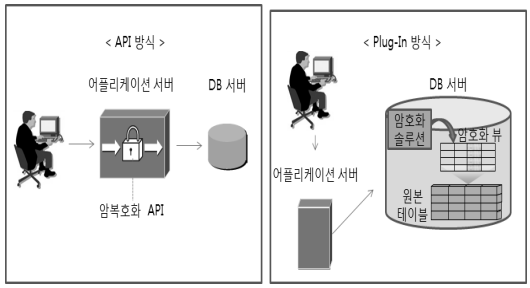
2.1.3 법적규제 준수

세계 각국은 개인정보의 중요성을 인지하고 개인정보 보호를 위한 관련 법률을 제정하여 시행중이다. 국내에서도 개인정보 보호와 관련된 개별법 간 상이한 처리기준으로 인해 개인과 기업들에게 많은 혼란을 야기하였다. 다행히도 '04년부터 논의되었던 개인정보보호법이 '11.9.30일부터 전면 시행되어 개인정보 수집 및 이용, 처리, 파기 등 단계별로 공통된 보호기준과 원칙을 제시하고 있다. 개인정보보호법은 적용대상을 공공·법인·단체 등 모든 개인정보처리자로 확대하며, 데이터 보호 관점에서는 고유식별정보의 분실·도난·유출·변조·훼손 방지를 위해 암호화 등 안전성 확보에 필요한 조치 마련이 의무화하였다. 이에 따라 개인정보처리자는 법적 규제 준수를 위해 강화된 데이터 보호 기술을 적용하고 운영하여야 한다.

2.2 데이터 암호화 기술

데이터베이스에 저장된 개인정보를 보호하는 주요 방안은 데이터베이스 사용자 계정관리, 접근제어 및 모니터링, 데이터 암호화 기술을 활용하는 것이다. 데이터베이스 사용자 계정관리는 데이터 보호를 위한 전통적인 방식으로 업무 수행에 필요한 자원만 접근할 수 있도록 최소한의 권한만 설정하여 운영하는 것이고, 접근제어 및 모니터링은 사용자 인증 및 권한을 강화하여 비인가자가 데이터베이스의 중요 테이블에 접근하여 데이터의 유출, 위·변조하는 행위를 차단하고 사후추적하기 위한 기술이다. 그러나 기업의 정보유출 사례의 80% 이상이 전·현직 내부 직원에 의한 사고[7]임을 감안할 때, 계정관리나 접근제어 기술은 권한있는 사용자에 의한 정보유출을 방지하기 어렵다는 한계점이 있다. 반면, 데이터 암호화는 데이터베이스 관리 시스템에 저장되어 있는 데이터를 암호화하여 저장하여 보호하는 방안으로서 내·외부에서 악의적인 의도로 데이터를 접근하거나 유출을 시도하는 위험으로부터 가장 안전하게 보호할 수 있는 방법이다. 데이터 암호화는 개인정보와 같은 중요 데이터를 허가받지 못한 사용자가 볼 수 없도록 암호화 기술을 이용하여 데이터베이스에 데이터를 기록할 때 암호화하여 저장하고, 권한있는 사용자가 조회 할 때 복호화하여 정보를 제공하는 기술이다. 현재 오라클이나 마이크로소프트와 같은 데이터베이스 제품들은 데이터 암호화, 접근제어, 모니터링 등과 같은 보안 기능을 제공하고 있으나, 국정원의 국가용 암호 제품 검증을 통과한 상용 암호화 솔루션에 비해 일부 데이터만 암호화하고, 암호를 푸는 복호화 키 저장방식도 안전하지 않아 데이터 유출시 손쉽게 해독될 수 있다[8].

한편, 국정원 인증을 받은 상용 데이터 암호화 솔루션은 암·복호화를 처리하는 물리적 위치에 따라 API 방식, Plug-In방식, 하드웨어 방식이 있다. [그림 1]과 같이 API방식은 외부 어플리케이션 서버에 제품이 설치되며 데이터베이스 외부에서 암·복호화가 수행된다. 이는 암·복호화시 데이터베이스 서버에 부하가 발생하지 않는 장점이 있으나, 기존 어플리케이션이나 테이블의 수정이 선행되어야 한다. Plug-In 방식은 데이터베이스 서버에 제품이 설치되며, 암·복호화가 데이터베이스 서버에서 수행되어 기존의 어플리케이션이나 테이블의 수정이 거의 발생하지 않는 장점이 있으나 데이터베이스 서버에 부하가 발생할 수 있다[9]. 하드웨어방식은 서버의 외부에 별도의 암호화 장



[그림 1] 데이터베이스 암호화 솔루션 개념도

비를 설치하여 시스템의 과부하를 감소시킨다. 이러한 솔루션들은 ARIA, DES(Data Encryption Standard), Triple DES, AES(Advanced Encryption Standard), SEED 등의 암호화 알고리즘을 적용하고 있으며, 암·복호화 처리 원리는 공통적으로 뷰(View)와 트리거(Trigger)의 기능을 활용한다. 데이터 암·복호화를 위해서는 제품별로 다소 차이가 있지만 일반적으로 비밀키 기반의 마스터키와 암호화키가 생성된다. 이러한 비밀키의 유실 또는 손상시 데이터 암·복호화가 불가능하며, 비밀키가 유출될 경우 데이터가 노출될 수 있으므로 철저한 보안 및 백업 관리가 중요하다[10].

2.3 데이터 암호화 이슈

개인정보보호법이 시행됨에 따라 주민등록번호와 같은 개인 정보의 암호화가 의무화되나, 시장에 출시되어 있는 데이터 암호화 솔루션을 이용하여 암호화시 다음과 같은 문제점이 발생할 수 있다[10]~[12].

2.3.1 객체관리의 혼란

데이터 암호화 솔루션의 원리는 뷰와 트리거를 사용하여 암·복호화를 수행하는 것이다. 이때 사용하는 뷰와 트리거는 실제 데이터 객체와 접근객체가 상이하게 되므로 객체관리에 다소 혼란을 초래할 수 있다.

2.3.2 인덱스의 제약사항 발생

데이터의 암호화는 테이블뿐만 아니라 제약조건을 정의한 PK(Primary Key)와 인덱스 등의 암호화도 적용하여야 하나 인덱스 암호화 기술은 간단하지 않다. 최근에 인덱스 색인시 데이터베이스의 전체 테이블 스캔(Full Table Scan)에 따른 운영의 효율성

저하를 최소화하고 검색 시간을 단축시킴은 물론 데이터 색인 시에 암호화된 특정 데이터의 일부에 대한 보안을 제공하는 기술을 적용한 솔루션이 출시[13]되었으나 아직 그 실효성이 검증되지 못했다.

2.3.3 심각한 성능저하

암호화 솔루션을 이용한 데이터 암호화는 암·복호화 알고리즘을 처리하는 시간보다 인덱스 제약사항으로 인해 암호화 칼럼에 인덱스가 생성되어 있어도 범위 스캔(Range Scan) 질의에 대해서 인덱스를 사용하지 못하고 전체 테이블 스캔을 함으로서 발생하는 시간 때문에 심각한 성능저하를 야기시킨다[14].

2.3.4 기타 추가적인 문제점

일부를 제외한 대부분의 솔루션이 실시간 데이터 암호화를 지원하지 못하는 것도 문제점으로 지적된다. 최근의 IT환경이 24시간/365일 항시적인 서비스 유지를 요구하고 데이터베이스가 대용량화되는 현실을 감안한다면 데이터 암호화를 위한 장시간의 서비스 중단은 사실상 허용이 불가능하다. 또한 데이터 암호화는 데이터베이스의 구조 자체를 변경시키기 때문에 암호화된 테이블의 컬럼 추가 및 변경시 복호화 후 재암호화해야 하는 제약사항이 발생한다.

III. 오라클 데이터베이스에서 데이터 암호화 사례 분석

3.1 오라클 데이터베이스에서 Plug-In 방식의 데이터 암호화 구현 환경

데이터 암호화 방식은 앞서 2장에서 기술한 바와 같이 API, Plug-In, 하드웨어 세가지 방식이 있으나, 대부분의 데이터 암호화 솔루션 제품들이 Plug-In 방식 기반으로 구현되어 있다. 따라서 본 논문에서는 오라클 데이터베이스에서 Plug-In 방식의 데이터 암호화 솔루션 제품으로 데이터 암호화 수행 사례를 분석하였다.

3.1.1 비즈니스 특성

데이터베이스를 포함하여 기업의 IT시스템은 비즈니스의 운영과 새로운 가치 창출을 위해 존재하므로 기업에서 운영중인 데이터베이스의 구조와 현황을 분석하기 이전에 비즈니스에 대한 이해가 선행되어야 한다. 이번 사례의 기업은 보험회사로서, 보험에 가입한 개인 및 기업에 대한 법인번호, 사업자번호, 주민등록번호, 주소, 전화번호 등의 정보를 보유하고 있다. 보험 심사 과정에서 외부 기관의 시스템과 연계하여 개인 신용정보를 조회할 수 있으며, 기업 내부의 다른 데이터베이스의 정보를 이용하기도 한다. 일부 업무는 온라인으로 회원가입과 보험청약이 가능하며 보험가입 심사가 자동으로 수행되는 원스톱 서비스를 제공한다. 또한 온라인과 오프라인으로 가입된 모든 보험 가입내역과 보험료는 웹상에서 제공되고 있다.

3.1.2 테스트 환경

본 테스트는 HP rp3440 서버(CPU 2Ghz, 메모리 24GB)에 설치된 오라클 10g 데이터베이스에서 실행하였다. 암호화 대상 테이블 'TB_ORIGINAL'은 200만건 이상의 데이터와 100개 이상의 칼럼을 보유하고 있다. 칼럼중에는 암호화가 필요한 주민등록번호가 PK(Primary Key)가 아닌 일반 칼럼으로 구성되어 있으며, 200만건 중 주민등록번호가 없는 데이터는 'Null' 값으로 입력되어 있다. 또한 테이블 'TB_ORIGINAL'은 단순 회원 정보 이외에 보험 가입을 위한 개인 및 기업의 필수 정보를 보유하고 있으며, 보험 가입 심사를 위해서는 다른 테이블과 필수적으로 연결되어야 한다. 이때 연결되는 다른 테이블도 대용량 테이블로서 원활한 비즈니스 수행을 위해서는 성능이 가장 중요시되고 있다.

3.1.3 Plug-In 방식의 데이터 암호화 솔루션으로 주민등록번호 암호화 과정

3.1.3.1 암호화 대상 칼럼 선정

데이터 암호화는 정보 보호를 위해 필수 불가결한 것이나 시스템의 성능과 보안은 서로 상충되기 쉬우므로 암호화 적용 대상의 선정은 신중을 기해야 한다. 암호화 대상으로는 한국인터넷진흥원의 '개인정보 유형과 종류'에서 언급된 사항을 우선으로 검토해야 하나 개인의 성명, 주소, 이메일주소와 같은 정보의 암호화 필요여부는 각 기업마다 다르게 해석될 수 있으므로 정보보호법과 같은 법규준수와 기업의 비즈니스를 감안하여 실제로 암호화해야 할 정보를 취사선택해야 한다. 본 논문에서는 암호화 대상을 개인의 주민등

록번호로 한정하였다.

3.1.3.2 암호화 알고리즘 및 암호화 방식 선택

데이터 암호화 솔루션은 Triple DES, AES를 비롯하여 국내 대칭키 알고리즘인 SEED 등과 같은 암호화 알고리즘 선택이 가능하다. 암호화에 소요되는 시간은 암호화 알고리즘 종류와 키의 길이에 따라 〔표 1〕과 같은 결과를 보였으며, 본 논문에서는 상대적으로 테이블 암호화 소요시간이 짧은 Triple DES를 적용하였다. 또한, 주민등록번호가 입력되지 않은 데이터가 상당수 존재하므로 'Null' 값에 대한 정보를 감추기 위하여 'IV(Initial Vector)'를 이용하여 동일한 데이터에 다른 암호화 값이 적용되도록 하였다.

〔표 1〕 테이블 암호화 소요시간

암호화 알고리즘	Triple DES (156bit)	AES (256bit)	SEED (128bit)
소요시간 (분:초)	10:36	11:18	15:35

3.1.3.3 선택적 칼럼 암호화 적용

온라인 업무를 수행하는 비즈니스의 경우 인터넷 서비스가 365일 24시간 가동되어야 하는 제약 사항이 있다. 특별한 경우에 시스템을 중지시킬 수 있다고 가정 하여도 중지 기간이 짧을 가능성이 많다. 데이터 암호화는 시스템 운영 중에도 수행 할 수는 있으나, 데이터베이스의 안정성을 고려할 때 데이터베이스와 연동된 어플리케이션 가동을 중지하고 암호화를 수행하는 것이 권고된다.

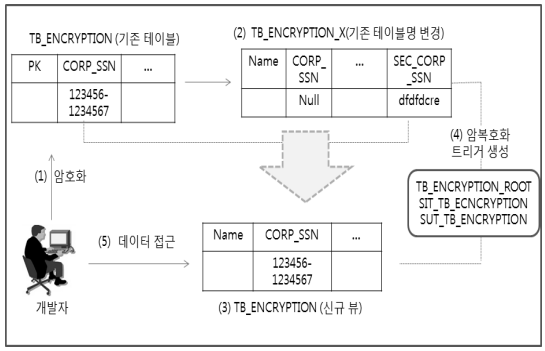
3.1.3.4 암호화 적용 후의 데이터베이스 구조

① 암호화이전

암호화 대상 칼럼을 보유하고 있는 원본 테이블명은 'TB_ORIGINAL'이며, 암호화 대상 칼럼은 주민등록번호를 갖고 있는 'CORP_SSN'이다. 또한 인덱스 'TB_ORIGINAL_IX03'은 암호화 대상 칼럼 'CORP_SSN'으로 구성되어 있다.

② 암호화 이후

암호화 전·후의 정확한 비교를 위하여 테이블 'TB_ORIGINAL'을 복사하여 테이블 'TB_ENCRYPTION'을 생성하고 인덱스를 일치시켰다. 이후 암호화 솔루션을 적용하여 테이블 'TB_ENCRYPTION'의 칼럼 'CORP_SSN'을 암호화하였다. 〔그림 2〕는 암호화 이후의 객체의 변화 내용이다.



〔그림 2〕 암호화 이후 데이터베이스 객체 변화

위의 〔그림 2〕와 같이 암호화 이후에는 기존 테이블 'TB_ENCRYPTION'은 'TB_ENCRYPTION_X'로 테이블명이 변경되어 일반 사용자가 접근할 수 없도록 보호되며, 어플리케이션과 일반 사용자를 위하여 기존 테이블과 동일 이름으로 뷰 'TB_ENCRYPTION'을 생성한다. 또한 권한 있는 사용자가 암호화 칼럼의 데이터 조회시 복호화를 위한 중간 뷰 'TB_ENCRYPTION_ROOT'에 내장된 복호화 기능을 거쳐서 뷰 'TB_ENCRYPTION'을 통해 데이터를 제공한다. 데이터를 입력할 때는 트리거 'SIT_TB_ENCRYPTION', 수정할 때는 트리거 'SUT_TB_ENCRYPTION'을 이용하여 암호화를 수행한다. 그리고 암호화시의 문제점 중의 하나로 지적되었던 인덱스 생성은 암호화 이후에도 기존 테이블과 동일하게 인덱스가 생성되었으며, 인덱스 'TB_ENCRYPTION_IX03'의 구성 칼럼인 'CORP_SSN'이 암호화된 칼럼으로 변경되었다. 또한 암호화 적용된 테이블 'TB_ENCRYPTION_X'에서 암호화 대상 칼럼 'CORP_SSN은 'Null' 값으로 업데이트 되고 칼럼 'SEC_CORP_SSN'을 추가하여 암호화된 데이터를 저장한다.

3.2 암호화 성능 비교

3.2.1 암·복호화 소요시간 분석

일반적으로 암호화로 인한 성능지연을 이슈화할 때는 암호화 이후의 총 처리시간을 의미하므로 순수하게 데이터 암·복호화 처리에 소요되는 시간과 데이터베이스에서 데이터를 가져오는 시간이 합해진 결과이다. 데이터 암호화 솔루션을 이용한 암호화는 트리거 및 뷰를 이용하여 암·복호화에 소요되는 시간으로 인한

성능지연뿐만 아니라 원본 테이블을 숨기고 동일 이름의 뷰로 데이터를 제공함에 따른 응답 차이도 발생할 수 있다. 따라서 순수하게 데이터 입력·수정에 따른 암호화에 소요되는 시간과 데이터 조회시 복호화에 소요되는 시간을 각각 측정해 보았다.

3.2.1.1 암호화에 소요되는 시간

데이터 암호화 솔루션을 이용한 암호화는 데이터 입력시 트리거 'SIT_TB_ENCRYPTION'과 수정시 트리거 'SUT_TB_ENCRYPTION'을 이용하여 데이터를 변경한다. 이 과정에서는 암호화에 소요되는 시간만 따로 구별하여 측정되지 않으므로 암호화된 뷰의 동일한 열에 대하여 일반 칼럼의 업데이트 질의와 암호화 칼럼의 업데이트 질의를 실행하여 두 질의의 시간차로 암호화에 소요되는 시간을 유추하였다. 동일 뷰에서 업데이트하는 칼럼만 제외하고 나머지 조건을 동일시한 후 테스트를 수행한 결과는 아래 〔표 2〕와 같다

〔표 2〕 데이터 수량별 암호화에 소요되는 시간 (단위:초)

질의 수행 시간 / 데이터건수	일반 칼럼 업데이트	암호화 칼럼 업데이트	시간차 (= 암호화 소요시간)
1,000건 미만 (993건)	0.62	0.66	0.04 (1.1배)
10,000건 미만 (9,659건)	6.69	6.65	-0.04 (1배)
100,000건 미만 (94,916 건)	1:01.16	1:01.03	-0.13 (1배)
1,000,000건 미만 (994,758 건)	9:55.88	10:59.52	9.64 (1.1배)

위의 결과에서 보면 데이터의 양에 상관없이 암호화로 인해 추가적으로 소요되는 시간은 대단히 작다. 데이터 수정시에는 트리거 'SUT_TB_ENCRYPTION'을 이용하여 암호화를 수행한 결과값을 저장하게 되므로 암호화에 소요되는 시간이 상당할 것으로 예상했으나 실제 결과를 보면 수행시간이 일반 칼럼의 업데이트 시간과 동일하거나 10% 이내로 증가되었다. 따라서 순수하게 암호화에 소요되는 시간으로 암호화에 따른 성능 지연을 논하기 어렵다.

3.2.1.2 복호화에 소요되는 시간

암호화에 소요되는 시간을 유추하는 방식과 동일하게 복호화에 소요되는 시간을 유추하였다. 우선 암호화된 뷰에 대해서 PK를 기반으로 일반 칼럼을 조회하는 시간을 측정하고, 동일한 질의에 대해서 암호화 칼럼을 조회하여 시간을 측정한 결과는 〔표 3〕과 같다.

〔표 3〕 데이터 수량별 복호화에 소요되는 시간 (단위:초)

질의 수행 시간 / 데이터건수	일반 칼럼	암호화 칼럼	시간차 (= 복호화 소요시간)
1,000건 미만 (993건)	0.01	0.01	0 (1배)
10,000건 미만 (9,659건)	0.06	0.09	0.0 (1.5배)
100,000건 미만 (94,916 건)	0.76	0.85	0.09 (1.1배)
1,000,000건 미만 (994,758 건)	16.93	24.87	7.94 (1.5배)

위의 〔표 3〕에서 보듯이 데이터가 1,000건 미만으로 소량인 경우에는 복호화하는데 소요되는 시간이 아주 짧아서 일반 칼럼을 조회하는 것과 동일한 결과가 발생하였다. 데이터가 1,000건 이상으로 많아짐에 따라 복호화에 소요되는 시간이 1.5배 이내로 증가하고 있으나 100,000건 미만인 경우에는 일반칼럼과 암호화 칼럼 조회 모두 1초 이내에서 처리되므로 복호화로 인한 성능 지연은 크게 문제되지 않는다.

또한 앞선 〔표 2〕의 '데이터 수량별 암호화에 소요되는 시간' 결과와 비교해보면 동일한 데이터 수량에 대해서 암호화에 소요되는 시간이 복호화에 소요되는 시간보다 상당히 오래 걸린다는 알 수 있다. 이로써 데이터 암호화시에는 복호화보다 암호화 작업이 데이터베이스에 미치는 영향이 더 크다고 할 수 있다.

3.2.2 암호화 적용 전·후의 SQL 성능 비교

일반적으로 암호화시의 성능을 문제시하는 것은 사전에 정형화하기 힘든 다양한 데이터 검색에 대하여 암호화 이전과 동일 또는 유사한 성능을 제공할 수 있을까 하는 것이다. 따라서 본 테스트에서는 암호화 칼럼에 대한 단일 테이블의 질의와 다른 테이블과의 조인 질의, 업데이트 질의, 암호화 칼럼이 검색조건이나 결과에 포함되지 않은 암호화된 뷰와 다른 테이블과의 조인시 등 실무에서 발생하는 사례를 기준으로 테스트하였다.

3.2.2.1 암호화된 칼럼이 select문에 사용된 단일 테이블의 단순 질의 수행시간 비교

데이터베이스에서 데이터를 추출하는 가장 간단한

〔표 4〕 데이터 수량별 단일 테이블의 단순 질의 수행시간
(단위:초)

질의 수행 시간 / 데이터건수	암호화 이전	암호화 이후	시간차
1,000건 미만 (993건)	0.01	0.01	0 (1배)
10,000건 미만(9,659건)	0.07	0.09	0.02 (1.3배)
100,000건 미만 (94,916 건)	0.66	0.85	0.19 (1.3배)
1,000,000건 미만 (994,758 건)	19.92	24.87	4.95 (1.2배)

방법으로는 단일 테이블에서 PK를 이용하여 해당 열을 조회하는 것이다. 암호화 전·후의 암호화 칼럼에 대한 단순 질의의 성능차이를 확인하기 위해서 암호화 이전의 원본 테이블과 암호화 후의 뷰에서 검색조건을 동일시하고 결과로는 PK와 암호화 칼럼을 추출하는 질의를 데이터 양을 달리하며 실행한 수행시간은 〔표 4〕와 같다.

위의 〔표 4〕에서 단일 테이블의 암호화 칼럼에 대한 단순 질의는 〔표 3〕의 '데이터 수량별 복호화에 소요되는 시간' 결과를 동일하게 사용하였다. 데이터가 1,000건 미만인 경우는 암호화 이전과 암호화 이후의 성능 차이는 존재하지 않는다. 1,000건 이상의 경우에도 암호화 이전과 암호화 이후의 성능 차이는 약 1.3배 내외로 심각한 수준이 아니다. 또한 100,000건 미만인 경우 암호화 전·후 모두 1초 이내의 성능을 보이고 있어 사용자들이 속도 차이를 실감할 수 없을 것이다. 또한 〔표 3〕의 결과와 비교해보면, 암호화 이전의 테이블에서 칼럼 조회, 암호화 이후의 뷰에서 일반 칼럼 조회, 암호화 이후의 뷰에서 암호화 칼럼 조회 세가지의 케이스는 데이터의 양과 상관없이 응답시간 차이가 심하지 않음을 알 수 있다. 따라서 검색조건이 간단한 단순 질의에서는 테이블과 뷰의 성능 차이, 복호화로 인한 성능 차이 두 부분 모두 미미하므로 온라인상에서 회원 정보 조회와 같은 단순 업무의 처리는 암호화에 따른 성능이 문제시 되지 않을 것으로 판단된다.

3.2.2.2 암호화된 칼럼이 select문에 사용된 단일 테이블의 복잡한 질의 수행시간 비교

앞서 살펴본 3.2.2.1의 경우는 검색조건이 인덱스가 적용된 칼럼만 사용한 단순한 경우이나, 이번에 테스트한 케이스는 단일 테이블을 대상으로 Sub query[1)]를 수행하는 복잡한 질의이다. 동일한 질의를 암호화 이전과 암호화 이후로 나누어 수행해보면, 암호화 이전에는 4.67초 걸렸던 수행시간이 암호화 이후에는 13.10초로 2.8배 지연되었다. 질의 수행 결과는 1개의 row로서 추출하는 데이터의 양은 적다. 그러나 두 질의의 실행계획(Explain Plan)을 비교해보면 암호화 이전에는 테이블의 인덱스를 이용하여 Rowid별로 데이터를 가져 왔으나, 암호화 후에는 Bitmap을 활용하는 것으로 실행계획이 바뀌었다. 테이블의 칼럼 및 데이터 수, PK와 인덱스 구성 등 물리적인 형태가 동일한 테이블이 암호화 적용 여부에 따라서 실행계획이 변하며 그 결과는 약 3배정도의 성능 차이를 보이고 있다.

3.2.2.3 암호화된 칼럼이 select문과 where조건에 사용된 다른 테이블과의 조인 질의 비교

암호화시 성능 이슈가 가장 문제시되는 것은 앞의 3.2.2.1과 3.2.2.2의 경우처럼 단일 테이블에서의 질의보다 다른 테이블과의 조인 질의의 경우일 것이다. 비즈니스가 복잡할수록 연계되어야 하는 테이블의 수도 많아지고 대용량의 테이블과 조인하여 정보를 추출해야 하므로, 질의 조건도 다양해진다. 따라서 암호화된 칼럼을 검색조건과 조회 결과로 사용하면서 대용량 테이블과 조인하는 복잡한 질의를 실행한 결과, 암호화 이전에는 6.11초에 결과가 도출되었으나 암호화 이후에는 18.81초로 약 3.1배의 지연이 발생하였다. 두 질의의 실행결과를 비교해보면 3.2.2.2와 동일하게 암호화된 후에는 실행계획이 변경되면서 성능이 지연되었다.

3.2.2.4 암호화된 칼럼의 업데이트 수행시간 비교

앞서 3.2.1 '암·복호화 소요시간 분석'에서는 암호화 칼럼을 갖고 있는 테이블에 대해서 순수하게 암호화 알고리즘을 적용하는데 소요되는 시간을 유추해보았다. 이번 테스트에서는 암호화 전·후에 암호화 칼럼을 업데이트 하는데 소요되는 시간을 비교하였다. 질의문은 3.2.1 '암·복호화 소요시간 분석'에서 사용한 것으로서 검색조건으로 PK를 사용하여 주민등록번호 칼럼 'CORP_SSN'만 업데이트 하는 단순 형태이다. 이와 같은 질의문을 암호화 이전의 원본 테이블과 암호화된 뷰에 데이터 수를 달리하여 수행한 결과는 〔표

2) 다른 하나의 SQL 문장 절에 Nesteded된 select 문장을 의미이다.

〔표 5〕 데이터 수량별 암호화 칼럼 업데이트 수행시간
(단위:초)

질의 수행 시간 / 데이터건수	암호화 이전	암호화 이후	시간차
1,000건 미만 (993건)	0.02	0.66	0.64 (33배)
10,000건 미만 (9,659건)	0.21	6.65	6.44 (32배)
100,000건 미만 (94,916 건)	2.21	1:01.03	58.82 (28배)
1,000,000건 미만 (994,758건)	31.52	10:59.52	88 (21배)

5〕와 같다.

암호화에 따른 성능 테스트에서 가장 확연하게 전·후 속도 차이가 발생한 것은 데이터 업데이트이다. 1,000건 미만의 소량의 암호화 칼럼 업데이트 결과에서 보여주는 0.66초의 실행시간은 아주 작은 시간이긴 하나, 암호화 이전의 0.02초의 실행시간과 비교하면 상당한 성능 차이가 있음을 알 수 있다. 데이터 건수가 증가함에 따라 암호화에 소요되는 시간이 30배 이상에서 20배 수준으로 줄어들기는 했으나 앞선 테스트와 비교해보면 가장 심각하다고 할 수 있다. 다만, 온라인에서 소량의 데이터를 처리하는 어플리케이션 위주로 서비스를 제공한다면 업데이트 수행 시간으로 인한 업무 처리 지연은 문제되지 않을 것으로 사료된다.

3.2.2.5 암호화 이전 테이블과 암호화된 뷰의 성능 비교

암호화 솔루션의 특징은 일부 칼럼을 암호화하기 위해서 원본 테이블의 이름을 바꾸어 일반 사용자에게는 감추고, 원본 테이블과 동일한 이름의 뷰를 만들어 사용자들이 접근할 수 있는 객체로 제공한다는 것이다. 이번 테스트에서는 암호화된 칼럼이 검색조건이나 조회 결과로 사용되지 않으나 동일 테이블 또는 뷰의 다른 칼럼을 정보로써 사용해야 할 경우의 성능을 살펴보았다. 테스트를 위해 배치성으로 사용되는 복잡한 질의를 암호화 전·후에 각각 실행한 결과, 암호화 이전에는 1분08초, 암호화 후에는 1분15초로 약 7초간의 차이가 발생하였다. 그러나 테스트용 질의가 실무에서도 튜닝후에도 1분이 넘게 수행되던 질의이므로 7초간의 차이는 미미하여 성능 차이가 없다고 말할 수 있다. 따라서 데이터 암호화 솔루션이 암호화를 위해 뷰를 만들어 사용하는 구조로 인하여 발생하는 성능지연은 없다고 판단된다. 다만, 오라클의 옵티마이저가 뷰를 사용할 경우 실행계획을 정확하게 계산하지 못하는 경우도 발생 할 수 있으므로, 실제 업무에서는 실행계획 분석을 통한 성능 최적화 작업을 항상 염두해 두어야 한다.

3.2.3 암호화된 데이터의 SQL 적용범위 분석

2.3.3 '데이터 암호화의 이슈'와 관련하여 암호화시 SQL의 Where조건문에서 흔히 사용하는 'Like, Between, 〉, 〈, %' 등의 조건검색을 정상적으로 사용할 수 없고 단순히 일대일 일치검색과 부분적인 전방일치 검색만이 허용되는 인덱스 제약사항이 발생하게 된다고 한다〔9〕. 이와 같은 문제가 여전히 존재한다면 성능 이슈가 해결되었다고 하더라도 검색조건 제약으로 인해 데이터의 암호화 적용은 쉽지 않다. 이번 테스트에서는 Where 조건문에서 암호화 칼럼에 'Like, Between,〉, 〈'와 같은 조건을 적용하여 정상적으로 수행되는지 확인한 결과 암호화된 칼럼에서도 다양한 검색조건이 문제없이 적용되며, 실행계획에서도 암호화 칼럼의 인덱스를 이용하여 질의를 수행한 것으로 보아 암호화 칼럼의 검색조건 다양화와 인덱스 적용은 문제가 발생하지 않는 것으로 보인다.

3.3 데이터 암호화에 따른 어플리케이션 영향도 분석

Plug-In 방식의 암호화 솔루션은 암호화시 암호화 대상 테이블과 동일한 이름으로 뷰가 생성되어, 적절한 권한으로 암호화 칼럼을 조회하면 트리거에 의해서 복호화하여 데이터를 보여주기 때문에 어플리케이션 입장에서는 테이블과 뷰가 동일한 명칭으로 관리되므로 기본적으로는 변경의 필요성이 없다. 그러나 앞서 테스트한 바와 같이 암호화된 뷰가 다른 대용량 테이블과 조인 또는 Subquery 등 복잡한 질의로 수행될 때 발생하는 성능 저하 문제를 해결하기 위해서는 힌트를 사용하여 SQL의 실행계획을 조정하는 튜닝 과정이 필요하며, 이는 곧 어플리케이션의 수정을 의미한다. 문제는 속도 지연을 해결하기 위하여 튜닝해야 할 SQL이 얼마나 될지 알 수가 없다는 것이다. 암호화된 뷰가 비즈니스에서 어떤 용도로 다른 테이블과 어떻게 연결되는지에 따라 다르므로 각 기업에서는 암호화 적용시 사전에 충분한 검토가 있어야 한다. 또한 튜닝해야 할 SQL의 대상과 중요도에 따라서 필요 인력의 등급과 기간이 결정되므로 경우에 따라서는 상당

한 비용이 소요될 수 있다.

3.4 사례 분석을 통한 데이터 암호화시의 이슈 요약

앞서 암호화 솔루션을 활용하여 데이터 암호화시의 전·후 사례 분석을 통해 도출된 결과를 살펴보면 다음과 같은 사실을 알 수 있다.

① 수행되는 데이터의 양에 따라 다소 차이가 있으나, 암·복호화 수행시간과 단일 테이블의 단순한 질의의 경우에는 암호화가 성능 지연에 미치는 영향은 미미하다

② 단일 테이블에서 Subquery 실행 등과 같은 복잡한 질의나 다른 대용량 테이블과의 조인 질의인 경우에는 암호화 전·후의 실행계획이 변경되어 성능 지연이 발생할 수 있다.

③ 다량으로 암호화 칼럼 업데이트를 수행할 경우 상당한 성능 지연이 발생한다.

④ SQL에서 암호화된 칼럼의 검색조건 적용범위는 'like, between, in, 〉, 〈' 등 주요 검색조건이 적용가능하므로 인덱스 적용 등 질의의 제약이 발생하지 않는다.

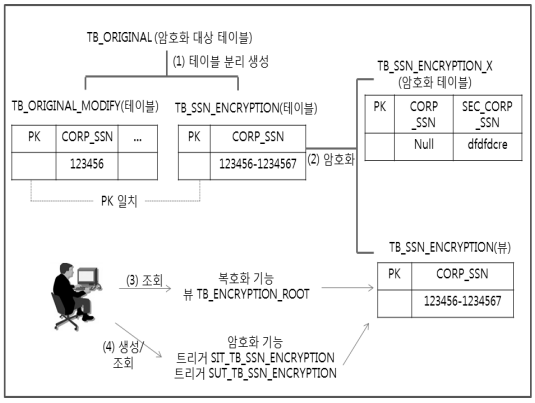
위와 같은 사실로 볼 때 데이터의 수가 적고 비즈니스가 복잡하지 않은 중소기업이나 기업 임·직원의 개인정보와 같은 소량의 데이터는 암호화 솔루션을 이용해서 데이터 암호화를 수행해도 성능면에서 크게 문제되지 않을 것으로 판단된다. 다만, 대용량의 고객 개인정보를 보유하고 있는 기업에서는 위의 ②의 실행계획 변경으로 인해 상당한 성능 지연이 발생할 수 있고, 이러한 문제를 해결하기 위해 튜닝 결과를 반영하기 위한 어플리케이션 변경이 필요할 수도 있다.

IV. 오라클 데이터베이스에서 데이터 암호화 사례 분석

4.1 데이터 암호화시 성능 개선을 위한 데이터베이스 모델 변경

3장에서 테스트 한 결과에 의하면 주민등록번호 등 개인정보 암호화에 따른 가장 큰 문제점은 복잡한 질의나 다른 대용량 테이블과의 조인시 처리 속도가 약 3배정도 지연되어 온라인 기반의 업무 처리에 적합하지 않다는 것이다. 따라서 이번 장에서는 주민등록번호 등 암호화 대상 칼럼들을 별도의 테이블로 분리한 후, 암호화를 적용한 기법에 대하여 분석하였다. 이와 같은 방법은 데이터 암호화 후에도 전체적으로 성능 차이가 미미해야 하며, 데이터베이스 구조 변경에 의한 어플리케이션에 미치는 영향을 최소화해야 한다.

앞서 3장에서와 동일하게 원본 테이블인 'TB_ORIGINAL'과 동일한 테이블 'TB_ORIGINAL_MODIFY'를 생성하고 주민등록번호를 보유하고 있는 칼럼 'CORP_SSN'을 질의 결과의 범위를 줄이기 위한 조건으로 사용하기 위해 주민등록번호 앞자리 6자리만 갖도록 업데이트 하였다. 또한 테이블 'TB_ORIGINAL'에서 암호화 대상인 주민등록번호를 갖고 있는 칼럼 'CORP_SSN'과 테이블 'TB_ORIGINAL_MODIFY'와 연계를 위해서 사용될 PK 칼럼을 추출하여 새로운 테이블 'TB_SSN_ENCRYPTION'을 구성한 후 칼럼 'CORP_SSN' 에 대해 암호화를 적용하였다. 아래 〔그림 3〕은 데이터베이스 모델 변경 사항이다.



〔그림 3〕 데이터 암호화시 성능 개선을 위한 데이터베이스 모델 변경 사항

이와 같이 데이터베이스 모델이 변경됨에 따라 암호화 이전에는 개인의 성명과 주민등록번호 확인을 위해 테이블 'TB_ORIGINAL'에만 접근하면 되었으나, 이번 테스트에서는 테이블 'TB_ORIGINAL_MODIFY'와 뷰 'TB_SSN_ENCRYPTION' 두 객체를 접근해야 한다.

4.2 데이터베이스 모델 변경을 통한 데이터 암호화 성능 이슈 비교

4.2.1 암호화된 칼럼이 Select문에 사용된 단일 테이블의 복잡한 질의 수행시간 비교

3장의 '암호화 이전의 암호화 대상 칼럼에 대한 단일 테이블의 복잡한 질의 수행시간' 테스트 결과와 비교하기 위해 동일한 질의를 테이블 'TB_ORIGINAL_MODIFY'와 뷰 'TB_SSN_ENCRYPTION'을 사용하는 것으로 변경 하여 수행하였다. 암호화 이전과 데이터베이스 모델 변경 후 암호화 적용한 질의를 수행한 결과, 1개 row에 대해 〔표 6〕과 같이 4.67초에서 5.24초로 1.1배 다소 증가되었으나 원본 테이블의 암호화를 적용한 뷰 'TB_ENCRYPTION'에서 소요된 13.10초의 수행시간과 비교하면 확연하게 개선되었음을 알 수 있다. 이와 같은 결과는 단일 테이블 'TB_ORIGINAL'을 분리하여 테이블 'TB_ORIGINAL_MODIFY'와 뷰 'TB_SSN_ENCRYPTION'으로 두개의 객체를 참조해야 하나, 두 객체가 PK로 연결되어 성능 차이가 거의 발생하지 않았기 때문이다.

〔표 6〕 모델변경 후 암호화에 따른 단일 테이블의 복잡한 질의 수행시간 (단위:초)

암호화 이전 (원본 테이블)	암호화 이후	
	원본 테이블 암호화	모델 변경 후 암호화
4.67	13.10	5.24

4.2.2 암호화된 칼럼이 Select문과 Where조건에 사용된 대용량 테이블과의 조인 질의 비교

이번 테스트에서는 개인정보를 별도의 테이블에 분리하고 칼럼 암호화를 수행한 후 대용량 테이블과 조인하는 질의의 수행시간을 비교하였다. 〔표 7〕과 같이 암호화 이전의 대용량 테이블과 조인한 결과는 6.11초, 분리된 테이블에서 칼럼 암호화를 수행한 후 대용량 테이블과 조인한 결과는 5.23초로 테이블을 분리하여 조인한 것이 약 1초 정도 빠르게 나왔다.

〔표 7〕 모델변경 후 암호화에 따른 대용량 테이블과의 조인 질의 수행시간 (단위:초)

암호화 이전 (원본 테이블)	암호화 이후	
	원본 테이블 암호화	모델 변경 후 암호화
6.11	18.81	5.23

두 질의의 실행결과는 동일함에도 불구하고 마지막 Select 구문에서 칼럼 'CORP_SSN'을 대용량 테이블인 'TB_ORIGINAL' 보다 데이터 양이 작은 뷰 'TB_SSN_ENCRYPTION'에서 갖고 오는 것이 블록을 더 적게 읽기 때문이다. 만약에 두 질의에 주민등록번호인 칼럼 'CORP_SSN' 이외 성명, 주소 등의 암호화 되지 않은 데이터를 같이 보여줘야 한다면 암호화 이전에는 테이블 'TB_ORIGINAL'에서 데이터를 갖고 오나 암호화 이후에는 테이블 'TB_ORIGINAL_MODIFY'와 뷰 'TB_SSN_ENCRYPTION' 두 개를 참조해야 하므로 성능이 동일 하거나 다소 늦을 것이다. 중요한 것은 두 개의 질의의 실행계획이 동일하게 작성되므로 성능에 미치는 원인은 테이블 'TB_ORIGINAL' 의 데이터 블록 수와 테이블 'TB_ORIGINAL_MODIFY'와 뷰 'TB_SSN_ENCRYPTION'을 PK로 조인하여 갖고오는 데이터 블록 수의 차이밖에 없다. 따라서 개인정보를 포함한 테이블에 대해서 대상 테이블 자체를 암호화하는 것보다 개인정보를 별도의 테이블로 분리한 후 암호화를 적용하는 것이 성능면에서 우수함을 알 수 있다. 다만, 암호화된 칼럼이 Select 구문이나 Where조건에서 사용하는 것 이외에 Group By나 Order By와 같이 정렬 조건으로 사용할 경우에는 암호화 칼럼을 복호화한 후 정렬을 함에 따라 상당한 속도 지연이 발생하므로 유의해서 사용해야 한다.

4.2.3 암호화 칼럼의 업데이트 수행시간 비교

데이터베이스 모델 변경후 암호화 적용에 대한 업데이트는 원본 테이블과 주민등록번호 데이터를 암호화하여 보유하고 있는 별도의 테이블 모두 변경해야 한다. 테이블 'TB_ORIGINAL_MODIFY'에서 암화해야 할 칼럼을 분리하여 별도의 테이블로 구성할 때 주민등록번호 검색 성능 향상을 위하여 칼럼 'CORP_SSN'을 주민등록번호 앞자리 6자리로 업데이트하였다. 이러한 방법은 주민등록번호를 검색조건으로 사용시 검색범위를 줄이므로 빠른 성능을 유지할 수 있으나, 테이블 'TB_ORIGINAL_MODIFY'에서는 주민등록번호 앞의 6자리, 뷰 'TB_SSN_ENCRYPTION'에서는 주민등록번호 13자리의 암호화된 데이터를 모두 업데이트해야하므로 데이터 업데이

〔표 8〕 모델변경 후 데이터 수량별 업데이트 수행시간
(단위:초)

질의 수행 시간 / 데이터건수	암호화 이전	원본 테이블 암호화	모델변경 후 암호화
1,000건 미만 (993건)	0.02	0.66	1:25.02
10,000건 미만 (9,659건)	0.21	6.65	1:27.98
100,000건 미만 (94,916 건)	2.21	1:01.03	2:05.88
1,000,000건 미만 (994,758건)	31.52	10:59.52	5:44.55

트 측면에서는 오히려 제약조건이 될 수 있다. 〔표 8〕은 앞서 〔표 5〕의 '데이터 수량별 암호화 칼럼 업데이트 수행시간'과 모델변경 후 칼럼 업데이트 수행시간을 비교한 것으로서 데이터가 소량일 때 업데이트 수행시간이 더 소요되는 것으로 나타났다. 이는 뷰 'TB_SSN_ENCRYPTION'의 암호화 칼럼 업데이트시 대상건 검색조건으로 인덱스를 사용하지 않고 전체 테이블 스캔으로 테이블의 전체 데이터를 검색한 후 업데이트를 수행하므로 소량의 데이터와 대용량의 데이터 수행 차이가 크지 않다. 따라서 온라인 어플리케이션에서는 변경해야 할 데이터가 소량일지라도 업데이트 쿼리문의 검색조건으로 인덱스를 사용할 수 있도록 테이블 설계시 주의하여야 한다.

4.3. 데이터베이스 모델 변경으로 인한 어플리케이션 영향도 분석

실무에서 암호화 적용시의 두가지 큰 이슈는 성능과 어플리케이션에 미치는 영향도이다. 암호화 적용시 성능의 문제가 발생하지 않더라도 안정적으로 운영되는 어플리케이션을 전반적으로 수정해야 한다면 데이터 암호화가 쉽게 이루어질 수 없다. 이번 장에서는 암호화시 성능 증대를 위하여 데이터베이스 모델 변경을 할 경우 Java 기반의 어플리케이션에 미치는 영향도가 어떠한지 확인해보았다.

4.3.1 데이터베이스 모델 변경에 따른 Java 기반의 어플리케이션 변경대상 추출

금번 테스트 대상의 회사는 온라인 어플리케이션을 Java기반으로 개발하여 운영하고 있다. 기존 암호화 이전의 원본 테이블 'TB_ORIGINAL'은 Java 프로그램에서 데이터 관리를 위하여 ValueObject로서 'CustomerInfo' 클래스로 매핑되었다. 그러나 데이터베이스 모델을 변경하게 되면 테이블 'TB_ORIGINAL_MODIFY'를 'CustomerInfo'로 매핑하고, 암호화된 뷰를 표현하기 위하여 'TB_SSN_ENCRYPTION'을 'PrivacyInfo' 라는 클래스를 추가하여 매핑하였다. 이 때 테이블 'TB_ORIGINAL_MODIFY'는 칼럼 'CORP_SSN' 에서 주민등록번호 앞 6자리로 데이터 업데이트가 발생하기는 하나 테이블 구조 자체가 변경되지 않으므로 매핑되는 'CustomerInfo'도 변하지 않는다. 그러나 주민등록번호 데이터가 기존에는 'CustomerInfo'를 통해 표현되었으나, 데이터베이스 모델 변경후에는 'PrivacyInfo'를 통해서 표시되야 하므로 어플리케이션 소스에서 주민등록번호를 보여주었던 ValueObject를 'CustomerInfo'에서 'PrivacyInfo'로 변경하여야 한다.

4.3.2 Java기반의 어플리케이션 변경 영향도 분석

4.3.1에서 언급한 바와 같이 'CustomerInfo'의 변수 'CORP_SSN'에는 주민등록번호 앞 6자리, 'PrivacyInfo' 의 변수 'CORP_SSN'에는 주민등록번호의 복호화 값이 표현된다. 따라서 사용자의 화면에 주민등록번호가 표시되어 있다면 어플리케이션 소스에서 'CustomerInfo'의 'CORP_SSN'에서 'PrivacyInfo'의 'CORP_SSN' 변수로 변경하여야 한다. 또한 어플리케이션 내부에서 사용된 질의 중 주민등록번호가 검색조건으로 사용되었다면 취급하는 데이터의 양을 감안하여 소량인 경우는 뷰 'TB_SSN_ENCRYPTION'을 직접 접근하여 데이터를 추출하고, 대용량인 경우는 테이블 'TB_ORIGINAL_MODIFY'에서 주민등록번호 앞 6자리를 이용하여 대상 데이터 범위를 줄인 후 뷰 'TB_SSN_ENCRYPTION'과 조인하여 데이터를 추출하는 것이 바람직하다. 이러한 내용들은 모두 어플리케이션 변경이 필수적으로 수반되어야 하나, 실제 비즈니스에선 주민등록번호와 같은 개인정보는 항상 표시되는 데이터가 아니므로 변경해야 할 어플리케이션이 제한적일 것이다.

아울러 원본 테이블의 암호화시에 동일한 이름의 뷰를 사용하므로 어플리케이션 변경이 없을 것으로 생각될 수 있으나, 앞의 3장 테스트에서 보인 바와 같이 암호화 이후 뷰의 사용에서는 SQL 실행계획이 변경되어 성능지연이 발생할 수 있으므로 성능향상을 위한

질의의 튜닝과 이에 따른 어플리케이션 변경이 예상된다. 따라서 단순하게 원본 테이블의 암호화 방식이 데이터베이스 모델 변경 후 암호화 적용하는 방식보다 어플리케이션에 미치는 영향이 적다고 할 수 없다.

V. 결 론

본 논문에서는 시장에서 널리 보급된 Plug-In 방식의 암호화 솔루션을 적용하여 주민등록번호를 암호화하고, 암호화 전·후에 실제로 발생하는 성능 차이를 테스트한 결과 데이터 조회시에는 약 3배, 업데이트시에는 데이터 양에 따라 많은 성능 지연이 발생함을 증명하였다. 따라서 성능 지연을 최소화하기 위하여 개인정보와 관련된 데이터를 별도의 테이블로 분리하여 암호화를 수행함으로서 원본 테이블에 미치는 영향을 최소화한 기법의 성능을 확인하였다. 이 때에도 업데이트 부분은 암호화가 동일하게 진행되므로 성능의 효과는 없으나, 데이터 조회는 원본 테이블과 구조가 동일하고, 암호화된 뷰는 PK를 통하여 조인되므로 암호화 이전과 유사한 성능을 보여준다.

그러나 여기서 테스트 한 사례가 반드시 옳은 것은 아니다. 오라클의 실행계획은 데이터의 양과 질의문, 통계 작성에 따라 달라질 수 있으며, 무엇보다도 어플리케이션 변경이 필수적으로 수반되기 때문이다. 다만 온라인 업무를 수행하는 기업에서는 어플리케이션에서의 응답시간이 무엇보다 중요하므로 성능을 유지한 채 어플리케이션 변경을 최소화하는 방법 중에 하나일 것으로 생각된다. 또한 중소기업의 매출현황이나 사원정보 등과 같은 다소 소규모의 데이터베이스를 기반으로 오프라인 방식의 비즈니스를 운영하는 기업이라면 기존 테이블의 암호화 적용시에 암호화 전·후의 성능 차이가 1~2배 이내이므로 데이터 암호화 솔루션을 적용하여 직접 암호화를 수행하여도 문제시 되지 않을 것으로 사료된다.

참고문헌

〔1〕 D. X. Song, D. Wanger and A. Perrig. "Practical techniques for searches on encrypted data," IEEE Symposium on Security and Privacy, pp. 44-55, S&P 2000

〔2〕 Z, Wang, J. Dai, W Wang and B. Shi, "Fast Query Over Encrypted Character Data in Database," Communications in Information and Systems, Vol. 4, No. 4, pp. 1027-1044, 2004

〔3〕 Z. Yang, S. Zhong and R. N. Wright, "Privacy-Preserving Queries on Encrypted Data," Proceedings of the 11th European Symposium On Research In Computer Security, pp. 479-495, 2006

〔4〕 김천식, 김형중, 홍유식, "암호화 데이터베이스에서 영역 질의를 위한 기술," 전자공학회 논문지 제45권 CI편 제3호, pp. 22-30, 2008년 5월

〔5〕 조은애, 문창주, 박대하, 홍성진, 백두권, "상세 접근 통제와 안전한 데이터 관리를 위한 데이터베이스 보안 시스템," 정보과학회논문지 데이터베이스 제36권 제5호. pp. 352-365, 2009년 10월

〔6〕 개인정보보호 포털, 개인정보의 중요성, www.i-privacy.kr

〔7〕 박명동, "데이터 보호를 위한 보안관리 정책 수립 및 적용 사례 연구," 서울시립대학교, pp. 4, 2007년 8월

〔8〕 금융권, DB 암호화 허술, 전자신문, 2011.4.12.

〔9〕 DB암호제품보안요구사항, 국정원, 2010.4

〔10〕 김정상, "개인정보 침해사례를 통한 데이터베이스 보안에 관한 연구," 건국대학교, pp. 40-49, 2007년 8월

〔11〕 백중일, "데이터베이스 보안을 위한 가용성 확장 연구," 숭실대학교, pp. 26-30, 2009년 2월

〔12〕 백중일, 박대우, "DB보안의 문제점 개선을 위한 보안등급별 Masking 연구," 한국컴퓨터정보학회 논문지 제14권 제4호, pp. 101-109, 2009년 4월

〔13〕 DB 패터별 암호화 인덱스 검색 특허 획득, 지디넷 코리아, 20112년 9월

〔14〕 데이터베이스 보안 솔루션 도입 및 운영의 허와 실, Oracle Korea, 2010년 9월

〈著者紹介〉

홍 정 화 (Jung Hwa Hong) 정회원
1997년 2월: 숙명여자대학교 통계학과 졸업
2011년 8월: 고려대학교 정보보호대학원 석사
〈관심분야〉 데이터베이스 암호

정 익 래 (Ik Rae Jeong) 정회원
1998년 2월: 고려대학교 전산학과 학사 졸업
2004년 8월: 고려대학교 정보보호학과 박사 졸업
2006년 6월~2008년 2월: 한국전자통신연구원 암호기술연구팀 선임연구원
2008년 3월~현재: 고려대학교 정보보호대학원 부교수
〈관심분야〉 프라이버시향상기술(PET), 데이터베이스 암호, 암호 이론