MySQL 데이터베이스에서 데이터 속성에 따른 적절한 암호화 기법의 적용에 관한 연구

신영호[0] 류재철
한국과학기술정보연구원, 충남대학교
shinyh@kisti.re.kr jcryou@home.cnu.ac.kr

Study on adoption of suitable encryption scheme according to data properties on MySQL Database

Youngho shin[0] Jae-Cheol Ryou
Korea Institute of Science and Technology Information, Chungnam National University

요 약

최근 개인정보 및 민감한 데이터에 대한 불법적인 접근 및 유출로 인하여 사회적 문제를 야기하고, 이에 따른 경제적인 파급효과와 함께 개인정보 및 민감한 데이터에 대한 보안에 대한 관심이 더욱 증가하고 있다. 또한 법령상으로도 개인의 주민등록번호, 계좌번호, 패스워드 등 개인정보가 포함된 DB에 대하여 데이터를 암호화하여 저장, 관리하도록 규정하고있다. 본 논문에서 공개 데이터베이스인 MySQL에서 이러한 개인정보 및 민감한 데이터에 대한 암호화를 통하여 데이터를 저장, 관리하는데 있어서 데이터의 속성에 따라 적절한 암호화 기법을 사용함으로써 암호화를 통한 데이터보호와 함께 속도 등의 성능상의 오버헤드와 운영, 관리상의 효율을 높이기 위하여 지원하는 암호화 기법에 대하여 알아보고, 암호화 기법별로 성능을 시험하여 데이터의 속성에 따른 최적의 암호화 방식의 적용에 대한 방안을 제시한다.

1. 서 론

최근 인터넷의 발전과 더불어 정보에 대한 가치가 더욱 증가하고 있으며, IT서비스가 점차 다양화되고 개인 맞춤형 서비스로 발전함에 따라 개인정보에 대한 수요와 가치가 증가하고 있으며, 수집된 개인 정보의 불법적인 접근 및 유출에 대한 우려가 증가하고 있다.

현재의 개인정보 이용 환경을 볼 때, 데이터 접근 시에 사용자의 질의 내용과 그에 대한 결과가 관리자 및 타인에게 그대로 노출되며, 이로 인해 발생하는 사용자의 프라이버시 침해가 반드시 해결되어야 할 문제로 지적되고 있다. 최근 인터넷상에서 무분별하게 개인정보가 도용되어 심각한 사회적 문제로 대두되고 있으며 또한 회원들의 이름, 주소, 주민등록번호, 이메일 등의 개인정보의 누출 사고가 커다란 문제가 되고 있다.

특히 주민등록번호와 같은 경우, 현재 우리나라에서는 신원을 확인하기 위한 기본적인 방법으로 광범위하게 사용되고 있으며, 여기에는 심각한 부작용이 따름에도 불구하고 기존의 신원확인절차를 임의로 변경하여 사용해서 생기는 문제 등에 대해서 법적 보호문제 및 책임문제가 발생하게 된다.

따라서, 이러한 주민등록번호 대체수단에 대한 연구가 많이 진행되고 있으며 그 결과로 가상주민번호서비스, 공인인증서를 이용한 주민번호 대체 서비스, 그린버튼 서비스, 개인ID인증 서비스, 개인 인증키를 이용한 주민번호 보호 서비스 등 5가지 대체수단이 제시되었다. 하지만, 여전히 주민등록번호가 주로 사용되고 있다.[2]

따라서, 이러한 데이터의 보호 문제를 근본적으로 해결하기 위한 방법으로는 데이터베이스내의 데이터 자체를 암호화하여 저장하는 방식이 필수적이다.[1]

데이터베이스의 데이터를 보호하기 위한 방법으로 네트워크단에서 접근통제를 수행하는 접근제어방식과 DBMS내부에서 데이터를 암호화하는 암호화방식으로 크게 나눌 수 있으며, 최근에는 암호화 방식에 접근제어를 혼용한 방식도 제안되고 있다.

접근제어방식의 경우 DBMS에 변경이 없고 성능저하가 적은 것이 장점이다. 반면, 암호화 방식은 DBMS내부에서 데이터를 직접 암/복호화 함에 따라서 암/복호화 성능문제와 함께 시스템에 부하가 발생하며, 온라인 서비스에 사용되는 시스템의 경우에 있어서 암/복호화 성능은 특히 심각한 문제가 될 수 밖에 없다.[5][6]

본 논문에서는 개인정보 및 민감 데이터에 대한 법령 요구사항을 충족하면서 이러한 암/복호화 성능문제를 다소 해소하기 위하여 MySQL 데이터베이스에서 제공하는 암호화 기법을 통하여 데이터의 속성에 적합한 암호화 기법을 제안하고자 한다.

본 논문의 구성은 2장에서는 MySQL 데이터베이스 지원하는 암호화 함수에 대하여 알아보고, 3장에서는 데이터의 속성에 따른 암호화 함수의 적용 가능한 부분을 알아보기 위하여 암호화 함수별로 암호화 성능을 시험하며, 4장에서는 성능시험 결과를 통하여 데이터의 속성에 적합한 암호화 기법을 제안하며, 마지막으로 5장에서는 결론과 향후 과제에 대해서 기술하였다.

2. MySQL 데이터베이스의 암호화 함수

MySQL에서 지원하는 암호화관련 함수는 단방향 암호함수와 양방향 암호함수를 포함하여 8가지 정도가 있다.

단방향 암호화 함수(복호화 불가능)로는 MD5, PASSWORD(OLD_PASSWORD), SHA1(SHA)과 같은 암호화 함수가 있으며, 양방향 암호화 함수 (암호화, 복호화 가능)는 ENCODE(DECODE), ENCRYPT(DECRYPT), DES_ENCRYPT(DES_DECRYPT), AES_ENCRYPT(AES_DECRYPT), COMPRESS(UNCOMPRESS) 함수가 존재한다.[3]

각 암호화 함수의 개략적인 특징은 다음과 같다.

2.1 md5

스트링에 대해서 MD5 128비트 체크섬을 계산한다. 입력된 데이터의 길이에 상관없이 128비트 암호화 해시 함수로서 주로 프로그램이나 파일이 원본 그대로인지를 확인하는 무결성 검사 등에 주로 사용되는 알고리즘이다.

2.2. password(), old_password()

평범한 문장의 패스워드 str에서 패스워드 스트링을 계산해서 바이너리 스트링으로 리턴한다. MySQL의 user 그랜트 테이블의 password컬럼에 있는 패스워드를 암호화하는데 사용하는 함수이다.

2.3 sha(), sha1()

Secure Hash Algorithm으로 스트링에 대해서 SHA-1 160비트 체크섬을 계산한다. SHA1은 SHA함수 들 중에서 가장 많이 쓰이며, TLS, SSL, PGP, SSH, IPSec 등 많은 보안 프로토콜과 프로그램에서 사용되고 있으며, 이전에 많이 사용되던 MD5를 대신해서 쓰이기도 한다.

2.4 encode(str, key_str), decode(crypt_str, key_str)

스트링을 키(패스워드)스트링을 이용해서 암호화와 복호화를 수행한다.

2.5 encrypt(str[,salt]), decrypt(crypt_str[,salt])

Unix crypt() 시스템 호출을 사용해서 str 스트링을 암호화하고 바이너리 스트링을 리턴한다. salt인수는 최소한 두 문자 길이 이상의 스트링이어야 한다.

2.5 des_encrypt(str[,{key_num|key_str}]), des_decrypt(crypt_str[,key_str])

주어진 키 값을 갖는 스트링을 triple-DES알고리즘을 사용해서 암/복호화한다.

DES(Data Encryption Standard) 알고리즘은 대표적인 대칭키 알고리즘으로서 1977년도에 미연방정부의 표준으로 채택되었으며, MySQL에서 지원하는 DES는 TripleDES로서 56비트의 키 길이를 가지는 DES의 취약성에 대응하여 개발된 알고리즘으로서 수행 속도면에 있어서는 고려해야 할 요소가 있지만, 112비트의 키 길이를 통한 더욱 보안성이 강화된 알고리즘이다.[7]

2.6 aes_encrypt(str, key_str), aes_decrypt(crypt_str, key_str)

AES는 그동안 사용되어온 DES를 대체할 보다 강력하고 견고한 알고리즘을 공모한 끝에 'Rijndael'이라는 알고리즘으로 두 명의 벨기에 암호학자(John Daemen, Vincent Rijmen)가 제출한 알고리즘이 채택되어 2000년부터 AES(Advanced Encryption Standard)라는 표준 알고리즘으로 지정되었다. AES는 128비트 크기의 키를 지원하지만, 192비트, 256비트 크기의 키를 지원하는 가변적인 키 길이를 지원하는 알고리즘이다.[7]

2.7 compress(str), uncompress(crypt_str)

COMPRESS는 zlib과 같은 압축라이브리를 통한 압축 알고리즘으로서, 엄밀한 의미에 있어서는 암호화 함수로 보기 어려울 수 있으나, 광의의 의미에서는 압축도 암호화의 일부로 포함된다.

3. 암호화 함수별 암호화 성능 시험

본 논문에서 암호화 성능 시험을 위하여 사용한 시스템은 Dell 서버로서 구체적인 사양은 다음과 같다.

항목	스펙	비고
CPU	Intel Xeon 3.00GHz * 2	dual core
메모리	ECC DDR2 6GB	
디스크	RAID-5 280GB	
OS	RHEL AS4 64bit	
MySQL	5.0.67 community server	

MySQL 데이터베이스는 공개용으로 5.0.67엔진을 암호화함수를 사용할 수 있는 옵션을 포함한 형식으로 컴파일하였으며, 튜닝은 별도로 수행하지 않았으며 기본적인 환경 파일을 적용하였다.

성능시험에서는 쓰기 또는 갱신 시간을 제외한 순수하게 암호화하는 시간만을 측정하였다. 따라서 실제 서비스에서 사용하기 위해서는 쓰기/갱신 오버헤드를 고려하여야 한다.

사용된 테이블은 제공하는 암호화 함수가 데이터 속성별로 어떠한 성능을 보여주는가를 알아보기 위하여 UserInfo 라는 개인정보가 담긴 회원 테이블을 사용하였으며, 데이터의 레코드 수에 따른 암호화 기법의 성능 비교를 위하여 Report라는 보고서 테이블을 사용하였다.

양방향 암호화 함수의 경우에 있어서 모든 암호화 키는 “encryption"이라는 키를 사용하였다.

또한 성능시험 결과 값의 정확성을 도모하기 위하여 모든 테스트는 각각 3회 수행하여 수행결과 값의 평균을 최종 결과 값으로 사용하였으며, 사용된 테이블의 주요

정보는 다음과 같다.

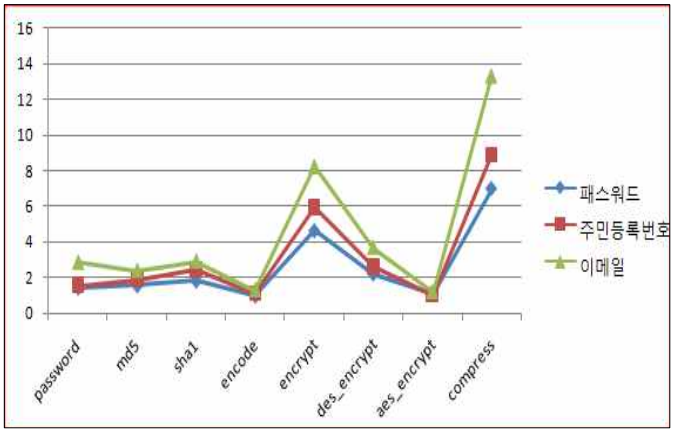
회원 테이블(UserInfo)의 주요 스키마 (레코드 수: 407,222건)

필드명	타입 및 길이	Comment
USERID	varchar(30)	회원 ID
USERNAME	varchar(50)	회원 이름
PASSWORD	varchar(30)	패스워드
SSN_1	char(6)	주민등록번호_1
SSN2_2	char(7)	주민등록번호_2
EMAIL_1	varchar(50)	이메일_
EMAIL_2	varchar(50)	이메일_2
ADDR	varchar(250)	주소
...	...	...

보고서 테이블(Report)의 주요 스키마 (레코드 수 : 106236건 vs 1,009,242건)

필드명	타입 및 길이	Comment
TI	text	보고서 제목
AU	text	저자
AB	longtext	초록
KW	text	키워드
RT	char(2)	보고서 형태
...	...	...

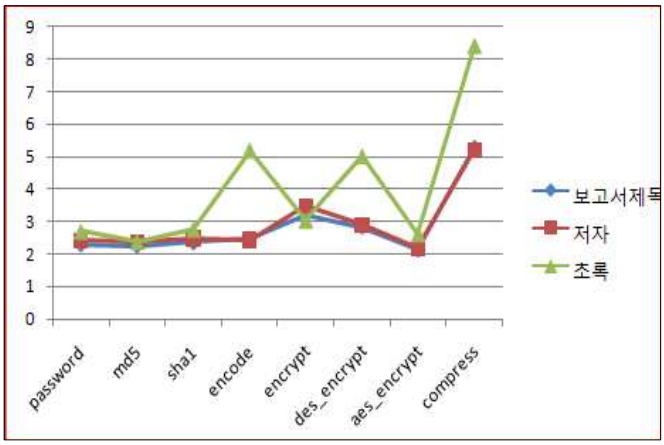
<그림 1>의 성능시험 결과는 개인정보에 해당하는 회원테이블을 대상으로 단방향 함수와 양방향 함수의 성능시험을 수행한 결과이다.



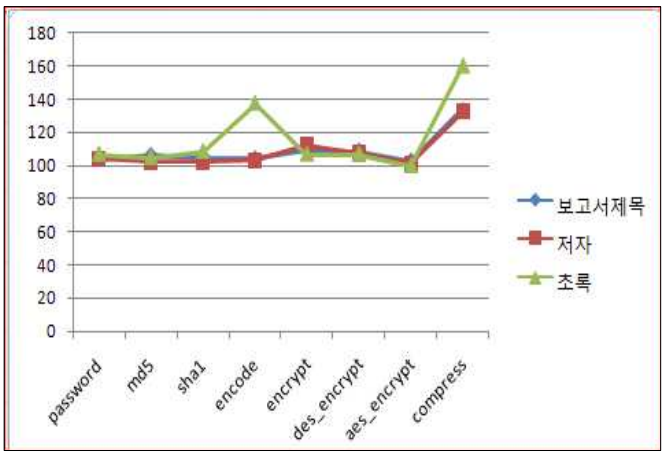
<그림 1> 개인정보(UserInfo)테이블 암호화 성능시험

성능시험 결과 값에 따르면, 단방향함수인 경우에 있어서 차이는 근소하지만 PASSWORD > MD5 > SHA1의 암호화 알고리즘 순서로 수행속도가 나타났으며, 양방향 함수인 경우, ENCODE > AES > DES > ENCRYPT > COMPRESS의 순서로 암호화 수행속도를 보여 주었다.

<그림 2><그림 3>은 보고서 테이블을 대상으로 단방향 함수와 양방향 함수의 성능시험을 수행한 결과이다.



<그림 2> 보고서(Report)테이블 암호화 성능시험 1



<그림 3> 보고서(Report)테이블 암호화 성능시험 2

보고서 테이블의 경우, 암호화 대상 레코드 수에 있어서 <그림 3>(약 100만 건)의 결과 성능이 <그림 2>(약 10만 건)의 결과 성능이 레코드 수에 비례관계가 아니라 레코드 수가 증가함에 따라 기하급수적으로 암호화 성능이 현저하게 떨어졌으며, 성능시험의 대상 필드의 데이터 속성 및 길이가 크게 차이가 없는 구조이지만, 데이터속성이 longtext(길이가 최대 4GB)인 초록(AB)필드의 경우에 DES와 COMPRESS함수의 경우 현저하게 암호화 수행 속도가 오래 걸렸다.

4. 데이터속성에 적합한 암호화 기법의 제안

암호화 적용에 있어서 법령의 요구사항으로써는 첫째, 비밀번호 및 지문, 홍채, 정맥, 음성, 필적 등의 개인을 식별할 수 있는 신체적 행동적 특징에 관한 정보인 바이오정보는 복호화 되지 않은 일 방향 암호 알고리즘을 통하여 저장해야 한다. 둘째, 주민등록번호, 신용카드번호 및 계좌번호에 대해서는 안전한 암호화 알고리즘으로 암호화하여 저장해야 한다. [8]

이러한 데이터 암호화에 대한 법령 요구조건과 데이터 속성 및 해당 데이터의 보안성의 요구정도에 따른 암호화 기법으로 본 논문에서는 성능 시험 결과 값 정보를 통하여 암호화 대상 필드와 적합한 암호화 알고리즘을

다음과 같이 제안한다.

첫째, 회원테이블의 패스워드와 같은 일 방향성 데이터의 경우 : SHA1

둘째, 회원테이블의 주민등록번호, 이메일, 보고서테이블의 보고서 제목, 저자, 초록과 같은 데이터의 경우 : AES

마지막으로, 암호화된 데이터에 대한 보안성을 유지하기 위하여 키의 관리는 무엇보다도 중요한 요소로서 키의 영속성문제에 있어서 하나의 키가 많이 그리고 오래 사용될수록 보안성은 약해지게 된다. 따라서 보안성의 정도가 높은 데이터일수록 키의 라이프사이클을 짧게 해야 하며, 보안성의 정도가 높은 데이터일수록 사용되는 필드별로 서로 다른 키를 사용하여 암호화해야 한다.

5. 결론 및 향후 과제

본 논문에서는 공개 데이터베이스로서 특히 광범위하게 사용되고 있는 MySQL 데이터베이스에서 암호화를 위하여 지원하는 함수에 대한 성능 시험을 통하여 데이터 속성에 적합한 알고리즘을 사용하여 법령 요구사항을 충족시키고, 각각의 비즈니스 용도에 적합한 암호화 기법을 적용하기 위한 기법을 제안하였다.

실제로 이용자테이블(UserInfo)과 보고서(Report) 테이블을 통하여 본 논문에서 제안한 데이터 속성에 따른 적합한 암호화 알고리즘을 통한 암호화를 적용하여 본 결과, 시스템에서 과도한 부하나 속도상의 문제로 인한 문제는 발생하지 않았으며 개인정보 및 민감한 데이터의 보호를 위한 방법론으로 충분히 활용할 수 있는 가치가 있었다.

또한, 본 논문에서는 데이터베이스 보안의 두 가지 분류인 접근제어와 암호화 중에서 암호화 부분만을 대상으로 하였으며, 완벽한 데이터보호를 위해서는 암호화 외에도 접근제어뿐만 아니라, 네트워크상에서의 전송정보 조차도 암호화를 통한 통신을 수행해야 한다.

향후 과제로서 사용자별로 암호화에 사용되는 키를 안전하게 관리하는 방안 및 키의 손, 망실 등의 사고발생에 대한 키 관리 대책에 관한 연구에 대한 연구가 필요하며, 수천만 건 이상의 대용량 데이터베이스에서의 데이터베이스 자체의 암호화방법과 응용프로그램에서의 암호화 방법의 혼용을 통한 암호화에 따른 성능 저하에 대비한 부하 분산에 관한 연구 등이 필요하다. 또한, 데이터에 대한 암호화를 수행한 후에 암호화 필드에 대한 검색연산의 속도향상을 위한 방안 즉 암호화된 인덱스 및 정렬 순서가 유지되는 암호화 기술 등에 대한 연구 등이 필요하다.

참고문헌

[1] 이호균외 2명, “데이터베이스 암호화 기술과 제품 동향”, 전자통신동향분석 제22권 제1호, p105, 2007.2

[2] 전웅렬외 3명, “접근제어형 데이터베이스 보안 시스템의 보호프로파일”, 정보보호학회지 제 17권 1호, 2007.2

[3] MySQL 홈페이지, http://dev.mysql.com/

[4] 위키피디아, http://www.wikipedia.org/

[5] Paul Dubois저 장준영역, “(한국어판)MySQL: MySQL의 사용, 관리, 프로그래밍을 위한 완벽 가이드”, 사이텍미디어, 2007

[6] Elmasri & Navathe , “Fundamentals of database systems fifth edition”, Addison-Wesley, 2007

[7] William Stallings, "Cryptography and network security principles and practices fourth edition", Prentice Hall, 2006

[8] 정보통신망 이용촉진 및 정보보호 등에 관한 법률