情 報 保 護 學 會 誌
第22卷 第7號, 2012. 11

클라우드에서의 데이터베이스 보안

유 천 영*, 김 지 홍**

요 약

클라우드 서비스란 문서, 영화, 사진, 음악 등의 다양한 파일들을 서버 컴퓨터에 저장해 두고 사용자의 PC 혹은 스마트폰 등의 다양한 매체를 이용하여 인터넷이 가능한 모든 곳에서 사용 할 수 있는 서비스이다. 이와 같은 클라우드 서비스는 사용자들에게 시간과 장소 및 다양한 기기에서 사용할 수 있다는 편리함과 유용성을 제공하는 반면에, 많은 사용자들이 함께 사용함으로서 발생될 수 있는 여러 가지 보안위험이 존재한다.

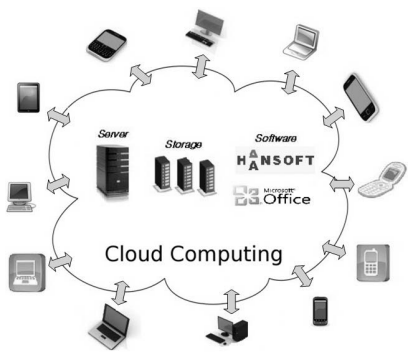
본 논문에서는 클라우드 서비스를 통하여 제공되는 데이터베이스에 관한 보안방법을 다룬다. 중요 데이터에 대한 암호화 기능을 서버에 두는 방식과 클라이언트의 응용 프로그램에 두는 방식을 비교하고, 이들 중 클라우드 서비스에 적합한 응용레벨의 암호화 방식에 대하여 최근에 소개된 여러 가지 암호화 방식을 소개한다.

Ⅰ. 서 론

1.1 클라우드 컴퓨팅 서비스 정의

클라우드 서비스란 문서, 영화, 사진, 음악 등의 다양한 파일들을 서버 컴퓨터에 저장해 두고 사용자의 PC 혹은 스마트폰/ 태블릿 PC 등의 다양한 매체를 이용하여 인터넷이 가능한 모든 곳에서 데이터, 네트워크, 콘텐츠 등을 사용할 수 있도록 하는 서비스이다. 클라우드 서비스는 그림 1과 같이 S/W, 오피스, 저장장치 등의 자원을 웹 서버에서 지원하여 따로 설치 없이 PC 및 서버, 스마트폰, PDA 등에서 자유롭게 활용할 수 있다[1]. 이러한 서비스는 사용자가 필요로 하는 만큼의 자원을 서버에서 제공받고, 사용자가 비용을 지불하는 기술로서 기업 인프라 유지보수 부담 및 사업초기 대규모 초기투자비용에 대한 부담도 줄일 수 있다.

클라우드 서비스의 분류로 IaaS(Infrastructure as a Service)는 CPU, 디스크 등 컴퓨터 시스템의 하드웨어 자원을 가상화하여 여러 사용자에게 제공하고, PaaS(Platform as a Service)는 하드웨어 자원을 추상화하고 그 위에 소프트웨어 개발과 수행환경을 제공한다.



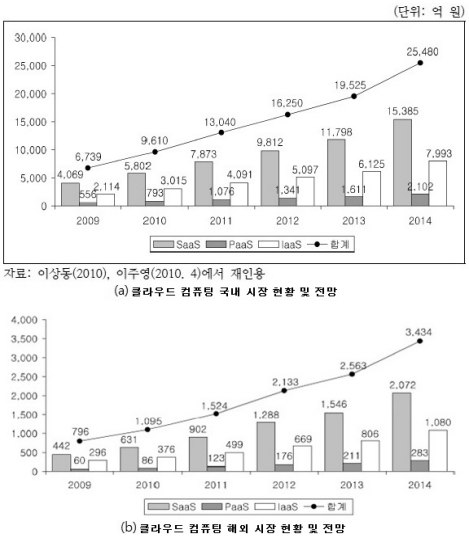
〔그림 1〕 클라우드 서비스

SaaS(Software as a Service)는 하드웨어 자원 및 응용 소프트웨어를 서비스 형태로 제공한다[2].

1.2 클라우드 서비스 시장 현황

스마트폰 활성화에 따른 클라우드 서비스 증가로 국내 및 해외 포털사 및 통신사업자의 다양한 클라우드 서비스 출시로 경쟁이 과열되고 있다. 스마트폰이나 노트북, 태블릿PC 등 다양한 디바이스에서 파일을 손쉽게 관리 할 수 있는 개인형 클라우드 서비스가 증가되어,

본 연구는 한국연구재단의 기초연구사업의 지원으로 수행되었습니다.(과제번호 : 2010-0023648)
* 세명대학교 정보통신학부 (soliebe1@semyung.ac.kr)
** 세명대학교 정보통신학부 (jhkim@semyung.ac.kr)



자료: 이상동(2010), 이주영(2010. 4)에서 재인용
(a) 클라우드 컴퓨팅 국내 시장 현황 및 전망

(b) 클라우드 컴퓨팅 해외 시장 현황 및 전망

〔그림 2〕 국내 및 해외 클라우드 서비스 시장 비교
(http://mobilist.tistory.com/82)

비용 절감, 유연성, 민첩성 등의 다양한 이점의 제공으로 클라우드 이용자 수는 급증하는 추세이다. 그림 2와 같이 국내 및 해외에서의 클라우드 서비스 시장의 전망으로 클라우드 서비스 경쟁이 확대되고 있다.

II. 본론

2.1 데이터베이스 보안 방안

데이터베이스내의 중요 데이터에 대한 비밀성(confidentiality), 무결성(integrity), 유용성(availability)을 유지하기 위한 보안방법으로는 접근제어 방법과 암호화 방법으로 구분할 수 있다.

접근제어 방법은 DBMS(DB Management System) 내의 사용자에 대한 접근제한 정책을 설정하여 불법접근을 차단하는 방법으로서, 임의 접근제어(Discretionary Access Control - DAC), 강제 접근제어(Mandatory Access Control - DAC), 그리고 직무기반의 접근제어(Role Base Access Control - RBAC) 방식이 있다. 이러한 접근제어 방식들은 다양한 방법으로 우회접근의 위험성을 가지고 있다.

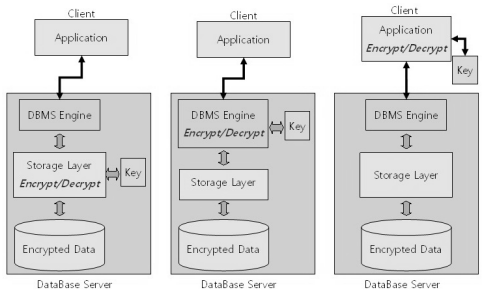
암호화 방법은 최근 클라우드 기반의 데이터베이스를 사용하는 경우에는 클라우드 서비스 제공자인 데이터베이스 관리자에 의한 중요 정보 노출 위험과 불법침입에 의한 노출 위험을 방지하기 위한 방법으로 중요 데이터에 대한 암호화가 최선의 방법으로 부각되고 있다.

2.1.1 암호화 목적

데이터베이스를 백업용으로 저장하기 위해 암호화하는 경우와 백업용이 아닌 실시간 검색 등의 용도로 활용하기 위한 중요 데이터를 암호화하는 방법으로 구분할 수 있다. 데이터베이스 백업용으로 암호화를 사용하는 경우에는 단지 안전하게 저장하기 위해 사용될 수 있으며, 삽입(Insert), 변경(Modify) 및 실시간 검색(Search) 등의 기능이 필요하지 않다. 다만 안전하게 암호키 와 복호키를 보관하고, 필요시에 백업된 데이터베이스를 복호키로 복구할 수 있으면 된다. 그러나 실시간으로 동작되고 있는 데이터베이스를 암호화하기 위해서는 중요 정보에 대한 암호화와 함께 실시간 검색이 가능하도록 보안기능과 검색성능이 함께 고려되어야 한다.

2.1.2 암호화 레벨

데이터베이스를 암호화 방식은 그림 3과 같이 DBMS 구조에 따라 다음과 같이 세 가지 형태로 구분할 수 있다.



〔그림 3〕 세 가지 방식의 암호화 방식

① 스토리지 레벨의 암호화 방식은 그림 3의 첫 번째 그림과 같이 데이터베이스내의 스토리지 영역에서 암호 및 복호를 수행하는 방식으로서, 운영체제 측면에서 파일들이나 디렉토리 단위로 암호화를 수행한다. 실제로 응용프로그램의 변경을 요구하지 않기 때문에, 비교적 간단한 방법으로 암호화할 수 있다. 사용자 측면에서 암호화 키/복호화 키가 필요하지 않으며, 사용자의 권한 정보 혹은

데이터의 민감도와 무관하게 암호화 한다. 암호화 부하를 줄이기 위해서 중요정보만을 암호화하는 선택적인 암호화 방식을 적용하기 위해서는 중요 정보만을 보관하는 파일 혹은 디렉토리 단위로 세분화하여 구성하여야 한다.

② 데이터베이스 레벨의 암호화 방식은 그림 3의 두 번째 그림과 같이 데이터베이스내의 DBMS 엔진 내에서 동작된다. 즉, 사용자와 데이터베이스간의 접속부분에서 동작하며, 사용자가 데이터를 삽입할 때 암호화하고, 반대로 사용자가 데이터베이스로부터 데이터를 복구할 때 복호화 하는 방식이다. DBMS 엔진 내에서 암호화/복호화가 수행되기 때문에 데이터베이스의 테이블(table), 컬럼(column), 열(row) 단위로 세부 항목별로 암호화가 수행될 수 있다는 장점이 있지만, 이러한 데이터를 복호화 하여 사용자에게 전달하기 위해서는 추가적으로 암호화데이터를 효율적으로 검색하기 위한 인덱스가 필요하기 때문에 부분적으로 사용자의 응용 프로그램에 대한 수정이 요구될 수 있다. 이와 같이 스토리지 레벨의 암호화 방식과 데이터베이스 레벨의 암호화 방식은 모두 데이터베이스 서버 내에서 암호화와 복호화가 수행되기 때문에 데이터베이스에 대한 관리가 클라우드 서비스 제공업체의 데이터베이스 관리자에 의해 수행되는 암호화 방식으로 클라우드 시스템에 적용하기에는 중요 정보 누출의 위험이 있다.

③ 응용레벨의 암호화 방식은 그림 3의 세 번째 그림과 같이 암호화 및 복호화 동작이 사용자의 클라이언트 응용프로그램에서 동작된다. 즉 사용자가 데이터를 삽입할 경우에는 클라이언트 모듈에서 암호화 기능이 동작되어 암호화된 데이터가 데이터베이스 서버로 전송된다. 마찬가지로 사용자가 클라이언트 모듈에서 데이터를 검색하고자 하는 경우에는 데이터의 암호화가 이루어진 상태에서 데이터베이스 서버로 전달되고, 암호화된 데이터에 대한 검색 결과가 사용자 모듈로 전달되어 복호화 된다. 암호화 및 복호화와 관련된 키는 항상 클라이언트 모듈에 보관되어 있어야 하며, 데이터베이스 서버에는 오로지 암호화된 데이터만을 보관하는 형태로 구성되기 때문에 클라우드 에서의 데이터베이스 보안에 적합한 방식이다. 중요정보에 대한 선택적인 암호화 및 복호화 기능도 응용프로그램에서 구현할 수 있기 때문에 다양한 융통성을 가지고 있으나, 가장 큰 단점은 암호화된 데이터베이스에 대한 검색 시에 실지 데이터보다 많은 암호화된 데이터 들을 사용자 모듈로 가져와서 이를 복호화 하여 찾고자하는 데이터를 얻을 수 있기 때문에 검색성능이 저하 된다는 단점이 있다. 이러한 단점을 보완하기 위하여 다양한 형태의 보안/검색 성능 향상을 위한 암호화 방안이 제안되고 있다.

2.2 키 관리 방법

암호화 방법은 암호화 대상에 따라 달라질 수 있다. 암호화 대상으로 중요 속성정보에 대해서만 암호화, 컬럼 단위의 암호화, 튜플(열) 단위의 암호화, 테이블 단위의 암호화 등의 방법으로 적용할 수 있다. 또한 암호화 속성이 문자형과 숫자형인 경우에도 다른 암호방식을 적용할 수 있다. 문자형인 경우에는 문자형 데이터의 검색에 적합한 블룸필터를 이용할 수 있고, 숫자형인 경우에는 숫자의 특성상 순서를 유지한 암호화 방식이 적용될 수 있으며, 동일값에 대하여 암호화 결과가 같이 나타나는 ECB 모드와 동일값에 대한 분포를 노출을 방지하기 위해서는 CBC 모드를 사용할 수 있다.

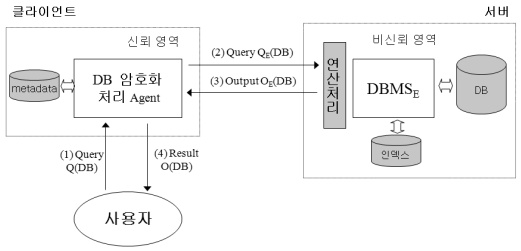
데이터베이스 암호화 방식에서의 키 관리 방법은 크게 HSM (Hardware Security Module) 방식과 보안서버(Security Server)를 이용한 방식으로 구분할 수 있다.

① HSM방식은 클라이언트 모듈에 HSM 모듈을 삽입시키는 방법으로 동작된다 (스토리지 레벨 방식과 데이터베이스 레벨 방식에서는 데이터베이스 서버에 HSM 모듈을 삽입). 암호화 키는 HSM 모듈에 저장된 마스터 키에 의해 암호화되어 저장된다.

② 보안서버를 이용한 방식은 별도로 구성된 보안서버내의 보안엔진을 통해 키를 보관하거나 복구하는 방법이다. 보안서버에는 사용자의 역할, 권한 정보와 함께, 암호화 정책들과 암호화키가 저장된다. 특히 모바일 클라우드 시스템의 경우에는 응용프로그램을 이용하여 보안서버와 접속하여 사용자의 권한정보를 확인한 후, 안전하게 암호화키와 복호화키를 전달받는 방법이 사용될 수 있다.

2.3 클라우드에서의 데이터베이스 암호화 방안

전 장에서 설명된 바와 같이 응용계층 레벨의 암호화 방식을 클라우드 환경에 적용하기 위해서는 클라이언트에서 암호화한 데이터가 데이터베이스에 저장된 후, 이를 효율적으로 검색하기 위한 인덱스 데이터가 서버에 포함되어야 하며, 이러한 구조는 다음 그림과 같다.



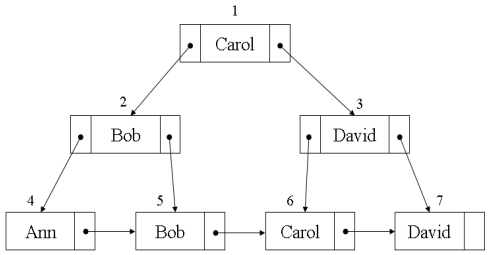
〔그림 4〕 클라우드에 적합한 보안구조

사용자의 평문 질의어(1)는 암호화된 데이터베이스를 검색하기에 적합한 형태로 클라이언트 모듈에 내장된 암호화키에 의해 암호화와 함께 변형된 질의어가 데이터베이스 서버로 전달된다(2). 데이터베이스 서버에서는 암호화된 데이터를 검색하기 위한 인덱스 데이터를 사용하여 암호화된 데이터의 검색을 효율적으로 수행하고, 검색된 결과를 클라이언트 모듈로 전달하고(3), 이를 복호화하고 평문형식의 질의 응답으로 변형되어 사용자에게 전달한다(4).

다음은 그림 4와 같은 구조를 이용하여 데이터베이스에 저장된 중요 정보에 대한 암호화하는 방법으로 B+ 트리 인덱스방식, 버킷인덱스 방식, 호모모피즘 방식 등에 대해 살펴본다.

2.3.1 B+ 트리 인덱스 방식

Damiani et al.[3]가 제안한 암호 방식은 평문 데이터베이스에 적용되고 있는 B+ 트리 인덱스 방식을 암호문에 적용한 방식이다. 각 노드 요소(element)는 이전 요소(previous element)와 다음 요소(next element)를 함께 암호화하여 보관하는 요소 단위의 암호화 방식이다. 이와 같은 B+ 트리 인덱스 방식은 최종 검색하고자 하는 요소를 찾기 위해서는 DB 서버와 DB 클라이언트 간의 B+ 인덱스 트리의 깊이(depth) 만큼의 통신을 하여야 하므로, 서버와 클라이언트간의 통신량이 많아진다는 단점과 요소 단위의 암호화 방식이므로 DB 내의 메모리를 많이 소모한다는 단점을 가지고 있다. B+ 트리 인덱스 방식은 데이터 암호화와 함께 평문데이터의 순서를 유지시킴으로서, 범위 질의어의 특성을 만족시킬 수 있는 인덱스 방식이다.



〔그림 5〕 B+ 트리 인덱싱 구조의 예

그림 5는 Name 필드 속성에 대한 B+트리 인덱스의 구성을 예로 든 것이며, 표 1은 평문으로 구성한 경우와 표 2는 암호화 방식을 적용한 경우의 암호화된 데이터 테이블이다.

〔표 1〕 노드 내용 테이블

ID	Node Content
1	2, Carol, 3
2	4, Bob, 5
3	6, David, 7
4	Ann, 5
5	Bob, 6
6	Carol, 7
7	David, NULL

〔표 2〕 암호화 데이터 테이블

ID	C
1	gtem945/*
2	8dq59wq*d'
3	ue63/)jw
4	8/*5sym,p
5	mw39wio
6	=wco21!ps
7	oie5(p8*

이와 같이 B+ 트리 인덱스 방법은 WHERE 절에 표현될 수 있는 동일값 검색과 범위검색을 만족시킬 수 있다. 게다가 순서를 유지하고 있기 때문에 암호화된 데이터에 대한 "ORDER BY"와 "GROUP BY" 절 뿐만 아니라 대부분의 집계함수도 처리할 수 있다.

2.3.2 버킷인덱스 (bucket Index) 방식

hacigumus[4] 등이 제안한 버킷기반 인덱스(bucket based Index) 방식은 튜플 전체를 암호화하고, 각 필드의 속성 도메인을 버킷 단위로 분류하여 저장하는 기법이다. 즉, 임의의 구간을 버킷이라는 중복되지 않은 연속적인 값을 가진 부분집합으로 나누는 과정을 버킷화 과정(bucketization process)이라 부르며, 모두 동일한 크기의 버킷들로 생성한다. 버킷기반 방식에서 범위 검색 쿼리의 경우, 버킷단위로 서버의 1차 쿼리 결과를 클라이언트에서 복호화하고 다시 2차 분석으로 버킷 내에서 최종 원하는 결과 값을 얻을 수 있는 방법이다. 결국, 버킷의 범위를 넓게 하면, 비도는 높아지지만 검색 성능은 저하되고, 반면에 버킷의 범위를 좁게 하면, 검색 성능은 향상되지만, 버킷 분포를 이용하여 사용자의 데이터 분포가 노출된다는 단점이 있다.

α	β	γ	δ	ε	ζ	η	θ	λ	μ
0~9	10~19	20~29	30~39	40~49	50~59	60~69	70~79	80~89	90~99

〔그림 6〕 범위별 버킷 적용 예시

일반적으로 레코드의 튜플 단위로 암호화하고, 각 필드에 버킷값을 부여하는 방식이다. 그림 6와 같이, 필드가 가지고 있는 전체 구간(0-99)을 동일한 크기의 버킷으로 생성되는 값으로 저장한다. 예로는 0~9까지의 버킷은 “α”이고, 10~ 19까지의 버킷은 “β”이며, 이러한 버킷 값들을 해당 필드 저장된다. 검색 에는 숫자 데이터에 대하여 구간별 버킷값을 사용하여 해당 구간에 암호화된 데이터를 클라이언트로 가져와서 이를 복호화하여 사용자가 원하는 데이터를 추출할 수 있다. 이외에도 데이터의 분포를 고려하여 버킷의 크기를 가변적으로 구성하여 버킷을 구성하는 방법도 제안되었다[5].

2.3.3 프라이버시 호모모피즘

IBM에 근무하는 Hakan Hacigumus과 Bala Iyer와 캘리포니아 대학에 근무하는 Sharad Mehrotra에 의해 제안된 논문에서는 암호문에 대한 집계연산(aggregation query)이 가능한 프라이버시 호모모피즘에 대한 정의가 소개되었다[6].

준동형성(Homomorphism)이란 암호문 상에서 가산, 곱셈을 적용하고 이를 복호한 결과가, 평문 상에서 가산, 곱셈을 한 결과와 동일한 함수를 말한다. 가산특성을 만족하는 가산 호모모피즘(additive homomorphism)과 곱셈특성을 만족하는 곱셈 호모모피즘(multiplicable homomorphism)이라 부른다. 이와 더불어, 암호문에 대해서 기본적인 연산(+, -, ×)을 만족하는 암호화 함수가 프라이버시 호모모피즘(Privacy Homomorphism)이다

(정의) Privacy Homomorphism
평문 도메인 : M, 암호문 도메인 : C
암호화 함수 : E_k, 복호화함수 : D_K
$\forall a \in M,\ D_K(E_K(a)) = a$

$\overline{\alpha} = \{\alpha_1, \alpha_2, .., \alpha_n\}$: 평문 도메인에서의 함수 , $\overline{\beta} = \{\beta_1, \beta_2, .., \beta_n\}$: 암호문 도메인에서의 함수일 때, 만일 $1 \le i \le n$ 에서, $D_K(\beta_i(E_K(a_1), E_K(a_2), .., E_K(a_m))) = \alpha_i(a_1, a_2, .., a_m)$이 만족된다면, 이 때 $(E_K, D_K, \overline{\alpha}, \overline{\beta})$를 Privacy Homomorphism 함수라 한다.

데이터 소유자인 클라이언트에 의해 선택된 두 개의 소수(prime number) p, q를 이용하여 $k = (p,q)$인 키가 선택한다. 이때, $n = p \times q$ 로서, p와 q는 암호화와 복호화 과정에 사용되므로 비밀리에 간직하고, n은 데이터베이스 서버관리자에게 공개된다. 이러한 방법은 암호학적으로 소수분해(prime factorization)의 어려움을 이용한다.

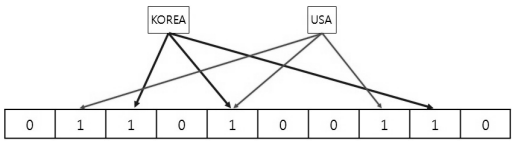
암호화 함수는 $E_k(a) = (a \bmod p,\ a \bmod q)$로서, 이때, $a \in Z_n$이며, $a \bmod p$과 $a \bmod q$를 각각 암호문의 p요소와 q 요소라고 한다.

복호화함수는 $D_K(d_1, d_2) = d_1 q q^{-1} + d_2 p p^{-1} \pmod n$으로써, 여기서 $d_1 = a \pmod p$, $d_2 = a \pmod q$이다. 또한 q^{-1}와 p^{-1}는 $qq^{-1} = 1 \pmod p$과 $pp^{-1} = 1 \pmod q$를 만족하는 역함수들이다. 여기서 $\overline{\alpha} = \{+_n, -_n, \times_n\}$은 modulo n 하에서의 덧셈, 뺄셈, 곱셈을 의미하며, $\overline{\beta} = \{+, -, \times\}$는 일반적인 덧셈, 뺄셈, 곱셈을 의미한다.

2.3.4 블룸필터(bloom filter) 방식

블룸필터 방식은 문자형 키워드 검색에 적합한 방식

으로서, m 비트로 구성된 필터를 사용하며, n개의 키워드 요소에 대하여 k 개의 해쉬 함수 결과에 따라 해당 비트를 “1”로 설정하는 방식이다[7]. 한 예로 “David” 이라는 데이터에 대한 3개의 해쉬 함수 결과 값이 {17, 56, 162}일 때에 블룸필드의 17번째 비트, 56번째 비트, 162번째 비트 값을 “1”로 세트하여 저장한다. 검색 시에는 검색하려는 데이터의 해쉬 결과에 따라, 블룸필드의 비트들이 모두 “1”로 설정되어 있다면 “David”이라는 데이터를 존재함을 알 수 있다. 그림 7은 “KOREA"에 대한 해쉬함수 결과값이 {2,4,8}인 경우와, "USA"에 대한 해쉬함수 결과값이 {1,4,7}인 경우의 예를 보여준다.



〔그림 7〕 블룸필터 방식 적용 예

블룸필터 방식은 검색하려는 키워드의 해쉬값으로 데이터베이스 내의 데이터의 존재여부 확인하기 때문에, 여러 개의 요소에 의해 해쉬 값이 동일하게 설정될 수 있으므로 긍정 오류(false positive)를 가질 수 있다. 만약, “Ann” 과 “Bob” 과 “David"에 대한 해쉬 결과값이 각각 {34, 94, 201}, {29, 82, 188}, {17, 25, 155}로 설정되었다고 가정한다. 만일 검색하려는 데이터가 “Carol”이며, 이에 대한 해쉬 결과값이 {17, 94, 188}일 때에는 데이터베이스에 “Carol” 이란 데이터가 존재하지 않더라도 해쉬 결과값이 모두 “1”로 설정되어 있기 때문에 존재하고 있다는 것으로 인식된다. 이러한 결과를 긍정 오류라고 한다. 블룸필터방식은 긍정 오류의 확률이 있기 때문에 1차 검색 후, 2차 복호화 과정을 거쳐서 올바른 결과 값을 찾을 수 있는 방식이다. 이외에도 단순히 평문값을 두 개의 데이터베이스 서버에 분리하여 저장하는 단편화 방식과 범위검색을 수월하게 하기 위하여 숫자의 순서유지 특성을 암호문에도 적용하여 순서를 유지할 수 있게 하는 OPES(Order Preserving Encryption Schema) 암호화 방식 등이 있다[8].

표 3에서와 같이, 버킷인덱스 방식과 블룸필터방식은 튜플 단위로 행 전체를 암호하고, 속성 별로, 버킷과 블룸필터라고 하는 인덱스를 사용하는 형태이며, 나머지 방식은 속성단위의 암호화를 사용하며, 추가적으로 인덱스를 사용한다.

〔표 3〕 기존 제안된 방식에 대한 질의어 처리 비교
(O : 지원, Δ : 부분지원, - : 지원하지 않음)

Index	암호 방식	문자검색		숫자 연산검색		
		문자형	문장형	동일값	범위	집계 연산
버킷인덱스방식	튜플	O	-	O	Δ	Δ
B+트리 방식	속성	O	-	O	O	O
PH 방식	속성	-		O	-	O
OPES 방식	속성	-		O	O	Δ
블룸필터방식	튜플	O	O	O	-	-

숫자 연산검색 방식으로는 B+ 트리 방식, 문자 연산검색 방식으로는 블룸필터 방식이 적합하다. 그러나 B+ 트리방식은 클라이언트와 서버간의 통신 횟수가 증가된다는 단점이 있다. 반면에 블룸필터 방식은 숫자연산 검색에서 동일값 검색에는 우수하지만, 범위 검색과 집계연산을 수행할 수 없다는 단점이 있다.

III. 결론

클라우드 서비스는 최근 IT기술의 급격한 성장으로 모바일 클라우드 서비스로 발전하고 있다. 지금까지 개인적으로 보관했던 S/W, 오피스, 저장 장치 등의 자원을 클라우드 서버에 보관하여 시간적, 공간적 제한 없이 사용할 수 있는 시스템으로 발전하고 있지만, 클라우드 시스템은 많은 사용자들이 공유하기 때문에, 데이터베이스에 보관된 중요 자료의 유출, 원본 데이터 손실 등의 문제가 발생할 위험이 존재한다.

본 논문에서는 데이터베이스를 암호화하기 위한 방법으로 여러 가지 방법을 살펴보았다. 버킷 인덱스 방식은 검색 효율화를 위하여 버킷방식을 사용하고 있으나, 버킷의 수가 증가할수록 데이터의 노출 가능성이 높아지는 단점을 가지고 있으며, B+ 트리방식은 대부분의 숫자 연산검색에 적합하지만, 데이터베이스 서버와 클라이언트간의 통신회수가 증가함으로서 통신량이 증가하다는 단점을 가지고 있다. 또한 PH 방식은 집계연산을 필요로 하는 속성에 적합하지만, 범위연산에는 적합하지 않다는 단점을 가지고 있으며, OPES 방식은 순서유지 특성을 가진 암호화된 데이터를 사용함으로서, 각종 연산에 적합하지만 원문 데이터의 크기가 노출된다는 단점을 가지고 있다. 마지막으로 문자검색에 적합한 블룸필터 방식은 문자검색에는 적합하지만 범위 및 집

계연산에는 효율성이 없다는 단점을 가지고 있다. 그러므로 각 방식의 장점을 이용하여 보다 강한 비도와 효율적인 검색성능을 가진 실시간 검색 알고리즘을 찾고자 하는 노력이 필요하다.

참고문헌

[1] “클라우드 서비스 현황과 전망”, 광고정보센터 리포트 7. 2011.

[2] 은성경, 조남수, 김영호, 최대선, “클라우드 컴퓨팅 보안기술”, 전자통신동향분석, 2009

[3] Ernesto Damiani, S.De Capitani di Vimercati, Sushi Jajordia, Balancing confidentiality and efficiency in untrusted relational DBMSs. In Jajodia, S., Atluri, V., Eds., Proc. of the 10th ACM Conference on Computer and Communications Security(CCS03), Washington, DC, USA, ACM, 2003, 93.

[4] Hakan Hacigümüs, Bala Iyer, Chen Li, and Shrad Mehrotra, Executing SQL over encrypted data in the database service provider model. In Proc. of the ACM SIGMOD 2002, Madison, Wisconsin, USA. ACM Press, 2002,

[5] Bijit Hore, Sharad Mehrotra, Gene Tsudik, "A Privacy Preserving Index for Range Queries", Proceedings of the 30th VLDB. Conference, Toronto, Canada, 2004.

[6] Hakan Hacigümüs, Bala Iyer, and Shrad Mehrotra, Efficient exeurtion of aggregation queries over encrypted relational databases, In Lee, J., li, J. Wudhang, K., and Lee, D., Eds., Proc. of the 9th International Conference on Database Systems for Advanced Applications, Volume 2973 of Lecture Notes in Computer Science, Jeju Island, Korea, Springer, 2004, 125.

[7] Andrei Broder, Michael Mitzenmacher, "Network Applications of Bloom Filters:", A Survey. Internet Math. Volume 1, Number 4, pp,89-95, 2003.

[8] Aggarwal, Gagan and Bawa, Mayank and Ganesan, Prasanna, "Two can keep a secret: a distributed architecture for secure database services", In Proc. of the Second Biannual Conference on Innovative Data Systems Research (CIDR 2005), Asilomar, CA, pp. 34-42, 2005.

〈著者紹介〉

유 천 영 (Choun-young Yu)
정회원

2005년 2월 : 세명대학교 컴퓨터과학과 학사
2007년 8월 : 세명대학교 대학원 전자계산 교육대학원 석사
2008년 2월~현재 : 세명대학교 대학원 전산정보 박사수료
<관심분야> 정보보호, DB 보안, 네트워크 보안

김 지 홍 (Ji-hong Kim)
정회원

1982년 2월 : 한양대학교 전자공학과 학사
1984년 2월 : 한양대학교 대학원 전자통신공학과 석사
1996년 2월 : 한양대학교 대학원 전자통신공학과 박사
1991년 2월~현재 : 세명대학교 정보통신학부 교수
<관심분야> 정보보호 응용, 인터넷 보안, DB 보안