버킷인덱스와 블룸필터를 이용한 범위형 의료정보 암호화기법

김창규* · 김정태** · 유천영*** · 김지홍****

A Mechanism of Medical Data Encryption Method Using Bucket Index and Bloom filter with the range property.

Chang-kyu Kim* · Jung-tae Kim** · Choun-young Yu*** · Ji-hong Kim****

이 논문은 2009학년도 동의대학교 교내연구비에 의해 연구되었음(과제번호 2009AA171)

요 약

최근 데이터베이스내의 개인정보의 유출이 사회적으로 이슈가 되고 있다. 개인의 민감한 정보를 보호하기 위한 최선의 방법은 데이터 암호화이다. 그러나 데이터를 암호화하면 질의어 처리가 어렵게 된다. 그러므로 데이터베이스를 보호하고 질의어 처리를 효율적으로 하기 위한 많은 방법들이 제안되고 있다. 본 논문에서는 기존의 연구에 대한 방법을 분석하고, 의료정보 데이터베이스내의 범위특성을 가진 데이터를 암호화하기 위한 방안으로서 버킷방식과 블룸필터 방식을 이용한 복합적인 방법을 제안하였다. 버킷방식만을 적용한 경우에 비하여 본 논문에서 제안한 버킷방식과 블룸필터방식을 융합하여 적용한 경우에는 버킷의 개수를 늘일 수 있고, 이에 따른 사용자 데이터의 분포 노출을 방지할 수 있으며, 결과적으로 검색속도를 높일 수 있음을 알 수 있다.

ABSTRACT

Recently, there are some social issues that personal sensitive data in database were let out. The best method to protect these personal sensitive data is used by the database encryption method. But the encrypting database makes the query difficult. So, there are a lot of study to protect the database and increase the query efficiency as well. In this paper, we analysed recent research trend to protect the sensitive data and propose the combined method using buckets and the bloom filter for the medical database with range property. Compared to bucket index model, the proposed method can increase bucket index value and protect data distribution exposure. We can estimate that this proposed method can improve searching time and efficiency.

키워드

버킷, 블룸필터, 데이터베이스 암호화, DAS

Keywords

Bucket, Bloom Filter, Database encryption, DAS

* 정회원 : 동의대학교 정보통신학과
** 종신회원 : 목원대학교 전자공학과
*** 정회원 : 세명대학교 정보통신학부
**** 정회원 : 세명대학교 정보통신학부 (교신저자, jhkim@semyung.ac.kr)

접수일자 : 2011. 01. 06
심사완료일자 : 2011. 01. 25

Ⅰ. 서 론

최근 데이터베이스의 규모가 점차 대형화하고, 이를 관리하기 위한 비용이 증가함에 따라, 데이터베이스에 대한 관리를 외부 데이터베이스 전문 업체에 의뢰하는 경우가 증가하고 있다. 그림1은 DAS(Database As a Service) 시스템의 일반적인 구조이다[1]. 일반적인 특징으로는 외부 전문 업체에 데이터베이스의 관리만을 위탁하도록 하며, 데이터베이스에 보관된 고객의 중요 데이터에 대해서는 암호화 방법을 이용하여 데이터베이스 관리자뿐 아니라, 일반 웹 사용자들로 보터 보호할 필요가 있다.

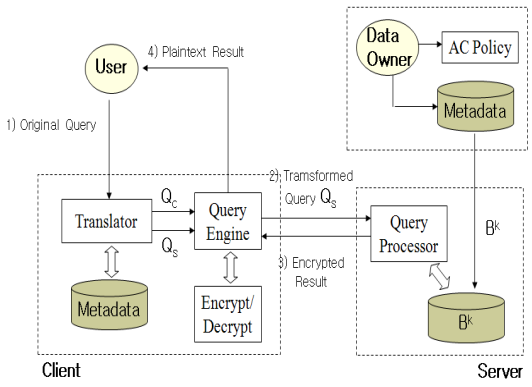

그림 1. 암호/복호 모듈을 사용한 DAS 시스템
Fig. 1 DAS system using the encryption and decryption modules

그림1에서 클라이언트 모듈에 있는 메타시스템이란 사용자의 평문 질의어를 암호화된 DB 서버에서 수행할 수 있도록 질의어를 변환시키는 역할을 한다.

표 1. 서버와 클라이언트간의 암/복호 기능
Table 1. The encrypt and decrypt functions executed in server and client

암/복호 기능	비교
DB 서버	처리속도가 빠른 반면에 보안상의 위험이 따른다.
DB 클라이언트	암호화 데이터에 대한 쿼리를 실행하기 위한 메타데이터가 필요하며, 처리속도가 느린 반면에 안전하다.

그림1과 같이, 암/복호 기능을 클라이언트에 두고, DB 서버에는 암호화된 데이터를 보관함으로써, 외부 전문 업체의 DB 관리자 혹은 데이터 분실 등으로 부터 사용자의 민감한 데이터를 보호할 수 있다. 또한 클라이언트와 서버 간에 암호화된 데이터가 전송될 때, 불법 도청에 의한 위험도 방지할 수 있다. 이와 같이 민감 데이터에 대한 보안을 위해 암호화 방법을 사용하면, 데이터베이스 질의어에 대한 응답 성능이 저하된다. 그러므로 데이터베이스에 대한 보안을 위하여 어떤 알고리즘을 사용하느냐에 따라 질의어에 대한 성능과 민감 데이터에 대한 보안간의 상충관계를 해결하기 위한 많은 노력이 필요하다. 실제로 기존에 제안된 데이터베이스 암호화 방법은 이러한 문제를 근본적으로 해결하지 못하고 있다[2,3].

데이터베이스 암호화 방식으로는 레코드 단위, 튜플 단위, 속성 단위의 암호화 방식으로 구분할 수 있다. 일반적으로 속성단위의 암호화방식은 민감한 데이터만을 암호화할 수 있는 반면에 암호화를 하기 위해 너무 많은 패딩이 필요하다는 단점을 가지고 있다. 반면에 블록단위의 암호화 방식의 경우에는 블록 내에 실제 필요한 데이터는 일부에 해당되기 때문에 DB 서버와 클라이언트 간의 통신트래픽이 증가하고, 클라이언트에서 복호 처리할 데이터양이 크게 증가한다. 튜플 방식의 암호화 방식은 해당 튜플 전체를 복호하고, 다시 필요한 부분만을 선택하여야 한다는 단점이 있기 때문에 버킷 등을 이용한 보완방법이 많이 사용되고 있다. 실제로 튜플 단위의 암호화 방법과 속성 단위의 암호화 방식이 많이 사용되고 있으며, 각 방식의 단점을 보완하기 위하여 질의어를 보다 효율적으로 처리하기 위한 논문들이 다수 있다[4,5,6].

본 논문은 다음과 같이 구성된다. 2장에서는 데이터 암호화를 위한 최근 연구동향을 살펴보고, 3장에서는 보다 효율적인 구조의 의료정보 데이터 보안을 위한 방법을 제시한다. 4장에서는 제안한 방법에 대한 이론적인 분석을 한다. 제안한 방법인 버킷(Bucket) 방식과 블룸필터 방식을 혼합한 적용 방식의 성능이 우수함을 증명한다. 마지막으로 5장에서 결론으로 마무리한다.

Ⅱ. 최근 연구 동향

암호화된 DB 서버를 구성하는 방법은 다양한 방법이 있으나, 현재까지 가장 많이 알려진 방법을 살펴본다. 먼저 버킷기반 (bucket based Index) 방식은 hacigumus et al.[2]이 제안한 방법으로서, 튜플 전체를 암호화하고, 각 필드의 속성 도메인을 버킷 단위로 분류하여 처리하는 기법이다. 즉, 임의의 구간을 버킷이라는 중복되지 않은 연속적인 값을 가진 부분집합으로 나누는 과정을 버킷화 과정(bucketization process)이라 부르며, 모두 동일한 크기의 버킷들로 생성한다. 버킷기반의 방식에서 범위검색 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 검색성능은 저하되고, 반면에 버킷의 범위를 좁게 하면, 검색성능은 향상되지만, 세부화된 버킷 정보로 인하여 데이터 분포가 노출된다는 단점이 있다.

Damiani et al.[3]이 제안한 암호문에 대한 B+ 트리인덱스 방식은 평문 데이터베이스에 적용되고 있는 B+ 트리인덱스 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+ 트리인덱스 방식은 최종 검색하고자 하는 요소를 찾기 위해서는 DB 서버와 DB 클라이언트간의 B+ 인덱스트리의 깊이(depth) 만큼의 통신을 하여야 한다. 즉, 서버와 클라이언트간의 통신량이 많아진다는 단점과 요소 단위의 암호화 방식이므로 DB 내의 메모리를 많이 소모한다는 단점을 가지고 있다.

Bala Iyer에 의해 제안된 PPC(Partition Plaintext and Ciphertext) 방식[4]은 한 개의 페이지를 두 개의 서브페이지로 구분하여 민감하지 않은 정보는 평문으로 저장하고, 민감한 정보에 대해서는 서브페이지 단위로 암호화하여 저장하는 방법을 제안하였다. 이러한 방식은 민감한 정보와 민감하지 않은 정보를 함께 암호화하는 튜플 단위의 암호화 방식에 비하여 일부 측면에서는 성능이 우수한 편이나, 범위검색이나, 집계연산 질의어에 대해서는 페이지 단위의 복호화를 수행하여야 하므로 성능이 떨어지는 편이다.

스탠포드 대학의 G. Aggarwal에 의해 제안된 Fragmentation Based 방식[5]은 아웃소싱 데이터베이스에 대한 보안방법으로 튜플에 저장된 속성들을 특성에 의해 서로 분리된 서버들에 저장하는 방식으로, 만일 한 서버에서의 데이터가 누출되더라도, 실질적으로 데이터 기밀성을 보장할 수 있는 방법을 제안하였다. 두 개의 서버 간에는 일종의 장벽이 설치되어 서로 통신할 수 없도록 구성된다. 논문에서는 수평 단편화와 수직 단편화 방법을 적용하여 두 개의 DB 서버로 분리 저장하는 방법을 제안하였다.

① 수평단편화 (horizontal fragmentation) : 관계 데이터베이스 R을 두 개의 서버 S_1에 R_1, S_2에 R_2를 저장하며, 이들 관계는 $R = R_1 \cup R_2$로 구성된다.

② 수직단편화 (vertical fragmentation) : 관계 R의 속성을 S_1과 S_2에 나누어서 저장된다. 그러나 키 속성은 두 사이트에 동시에 저장되며, 다른 속성들은 기본적으로 각 서버에 분리되어 저장되지만, 질의어 응답성능을 높이기 위해 동시에 저장될 수 있다. S_1, S_2 서버에 분리 저장될 관계 R_1, R_2는 $R = R_1 \otimes R_2$로 표시되며, 여기서 $\otimes$는 자연조인함수를 의미한다.

IBM에 근무하는 Hakan Hacigumus와 Bala Iyer와 캘리포니아 대학에 근무하는 Sharad Mehrotra에 의해 제안된 논문[6]에서는 암호문에 대한 집계연산(aggregation query)이 가능한 프라이버시 호모모피즘(Privacy Homomorphism)에 대한 정의가 소개되었다. 준동형성(Homomorphism)이란 암호문 상에서 가산, 곱셈을 적용하고 이를 복호한 결과가, 평문 상에서 가산, 곱셈을 한 결과와 동일한 함수를 말한다. 특히 가산특성을 만족하는 호모모피즘을 가산 호모모피즘(additive homomorphism)이라하며, 곱셈특성을 만족하는 호모모피즘을 곱셈 호모모피즘(multiplicable homomorphism)이라 부른다. 이와 더불어, 암호문에 대해서 기본적인 연산(+,-,X)을 만족하는 암호화 함수를 프라이버시 호모모피즘(Privacy Homomorphism)이라 부른다. 프라이버시 호모모피즘 함수는 숫자 데이터의 합계 및 평균을 계산하기 위한 집계연산에 적용될 수 있으나, 범위검색, 최대 최소값 검색 등에는 보완되어야 한다.

Aggarwal이 제안한 OPES(Order Preserving Encryption Schema) 방식[7]은 숫자 데이터에 대한 크기 특성을 암호문에서도 유지하기 위한 암호화 방식으로서, 평문의 분포를 균등하게 함으로써 평문의 분포를 알 수 없게 하는 암호화하는 방식이다. 이러한 방식은 암호문에 대한 동일값 검색(equality search), 범위 검색(range search) 기능을 만족시키는 방법이지만, 집계연산 특성은 만족시킬 수 없으며, 암호문의 크기를 통해 평문의 크기를 유추할 수 있다는 단점을 가지고 있다. 지금까지 데이터베이스의 암호화 방안으로 제안된 방식을 비교하면 다음 표2와 같다.

표 2. 기존 방식들 간의 질의어 처리비교
Table 2. Supporting queries for Indexing methods

Index	암호 단위	Query		
		Equality	Range	Aggregation
Bucket-Based	튜플, 요소	O	Δ	-
B+Tree	요소	O	O	O
PPC	페이지	O	O	O
Fragmentation	요소	O	O	O
Homomorphism	요소	O	-	O
OPES	요소	O	O	Δ

(O : 지원, Δ : 부분지원, - : 지원 않음)

마지막으로 데이터베이스의 암호화와 함께, 이를 보완하기위한 방법으로 키워드 검색을 위한 블룸필터(bloom filter) 방식을 사용할 수 있다. 블룸필터 방식은 m 비트로 구성된 필터를 사용하며, n개의 키워드 요소에 대하여 k 개의 해쉬 함수 결과에 따라 해당 비트를 "1"로 설정하는 방식이다. 일반적으로 해쉬 함수의 결과는 균등분포 확률로 가정하고, 각각의 키워드에 대한 해쉬 함수 결과 값이 중복되는 경우에도 이전에 "1"로 설정되어 있으면, 그대로 "1"로 유지하는 방식이다. 이와 같이 블룸필터 방식은 검색하고자 하는 키워드에 대한 해쉬값이 설정된 지 여부로 키워드가 존재하는 지를 판단하기 때문에, 여러 개의 키워드 요소에 의해 해쉬값이 동일하게 설정될 수 있으므로 "false positive"확률을 가진다.

이러한 false positive 확률은 $f=(1-e^{-kn/m})^k$로 표시되며, 이러한 확률이 최소가 되는 해쉬 함수의 갯수는 $k=\frac{m}{n}\ln 2$가 되며, 결국 false positive 확률은 다음과 같다[8].

$$f=(\frac{1}{2})^k \approx (\ln 2)^{\frac{m}{n}} \quad (1)$$

이와 같이 블룸필터 방식은 비록 false positive 확률에 의해 1차 검색 후, 2차 복호화 과정을 거쳐서 올바른 결과 값을 찾을 수 있는 방식이다.

Ⅲ. 의료정보 DB 보안 방법

3.1. 의료정보의 속성분석

데이터베이스에 저장되는 데이터는 각각 서로 다른 속성을 가진다. 문자형 데이터의 경우에는 크게 범위형 데이터와 문장형 데이터로 구분되며, 범위형 데이터(예, 혈액형, 주거도시 등)의 경우에는 일정한 구간단위로 분포되는 문자형 데이터이다. 예를 들면 혈액형과 같은 형태의 데이터로 분류할 수 있다. 또한 문장형 데이터(예, 의사의 소견서, 주소 등)는 키워드로 검색이 가능한 문자열 형식의 데이터에 적용할 수 있다. 반면에 숫자형 데이터의 경우에는 범위형 데이터와 집계형 데이터로 구분되며, 범위형 데이터는 숫자 데이터를 구간별로 구분하여 인덱스를 구성할 수 있다. 집계형 데이터인 경우에는 합과 평균을 계산할 필요가 있는 데이터에 대한 보안 방법으로 준동형 암호화방식이 사용되지만, 본 논문에서는 범위형 숫자 데이터만을 고려한다.

일반적으로 의료정보는 개인 신상정보와 의료정보로 구분될 수 있다. 개인 신상정보란 개인의 주소, 이름, 생년월일, 주민등록번호 등과 같이 개인의 민감한 정보에 해당된다. 의료정보는 신장, 체중, 혈액형, 혈압 등과 같은 신체에 대한 기본 정보와 병력 및 진료기록에 대한 의료정보로 구분될 수 있다.

위의 데이터베이스내의 정보특성에 따라, 본 연구에서는 문자형 데이터인 경우에는 범위형과 문장형을 다루며, 숫자형 데이터인 경우에는 범위형 숫자데이터에

국한하여 의료정보 데이터베이스를 보호하는 방안을 제시한다. 다음의 그 예시를 나타내고 있다.

- 주소 : 범위형(시, 도), 문장형 문자데이터
- 혈액형 : 범위형 문자데이터
- 주민등록번호 : 범위형(출생년월) 및 문장형 문자데이터
- 의료 소견서 : 문장형 문자데이터
- 신장, 체중 : 범위형 숫자데이터

3.2. 의료정보 보안방안 제안

의료정보 데이터베이스의 경우에 저장되는 데이터를 크게 범위형 데이터(문자데이터와 숫자데이터)와 문장형 데이터로 구분한다. 문장형 데이터인 경우에는 문장 내에 포함된 키워드를 블룸필터에 의해 바로 검색한다. 반면에 범위형 데이터의 경우에는 일정한 크기로 분리되는 버킷을 설정한다. 범위형 숫자데이터의 경우에는 먼저 구간별로 버킷을 설정한다. 문자형 범위 데이터인 경우에는 범위를 나타내는 문자데이터를 버킷으로 사용한다.

예를 들어 혈액형의 경우에는 A, B, AB, O형으로 분류되므로, 이에 대한 버킷값을 설정하고, 해당 버킷값을 다시 블룸필터에 적용한다. 혈액형에 대해 버킷값만을 적용하면, 혈액형의 분포를 바로 알 수 있기 때문에 블룸필터를 통하여 보안상의 목적으로 false positive 확률을 적절히 유지하기 위함이다. 결국 블룸필터의 false positive 확률을 이용하여 적절하게 보안을 유지할 수 있다. 범위형 숫자 데이터의 경우에도 마찬가지 방법으로 동일한 크기의 버킷으로 분할하여, 버킷 값을 생성하고, 이를 블룸필터에 적용하는 방식이다. 그림2는 각 영역에서의 값에 대한 버킷 값의 생성의 예이다. 그림3은 제안하는 방식에 대한 버킷 인덱스의 생성 및 블룸필터 적용의 예이다.

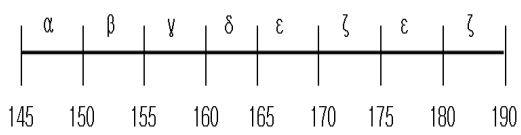

그림 2. 신장에 대한 버킷구분
Fig. 2 The bucket classification for height

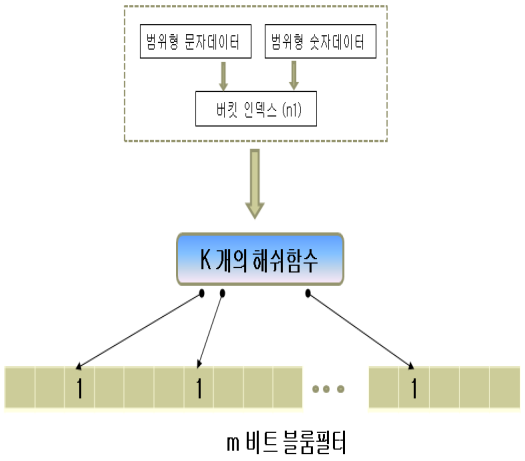

그림 3. 제안 방식
Fig. 3 The proposed algorithm

즉, 암호화된 데이터 검색을 위하여 평문 범위형 데이터에 대해서는 다음과 같은 단계를 거친다.

1) 버킷 인덱스 생성
2) 블룸필터 적용

3.3. 제안방식을 적용한 샘플

본 제안방식을 설명하기 위하여 다음과 같이 범위형 데이터로 구성된 간단한 데이터베이스를 예로 사용하였다. 먼저 범위형 숫자데이터에 해당되는 신장과 체중, 범위형 문자데이터에 해당되는 혈액형은 각각 구간으로 표현되는 버킷으로 분류하고, 이를 블룸필터에 적용한다.

표 3. 평문 데이터베이스
Table 3. The plaintext database

이름	혈액형	키	체중	도시
Ann	A	165	57	서울
Bob	O	52	52	대전
Carol	O	171	76	대구
David	B	165	63	광주

서버에는 튜플 단위로 암호화한 암호화 값과 각 필드별로는 분류된 해당 버킷을 블룸필터에 적용한 결과는 표4와 같다.

표 4. 암호문 데이터베이스
Table 4. The encrypted database

이름	암호화 값	블룸필터 값
Ann	E (A, 165, 57, 서울)	0x7155cd42
Bob	E (O, 155, 52, 대전)	0x5ab49321
Carol	E (O, 171, 76, 대구)	0x4e6ff3125
David	E (B, 165, 63, 광주)	0x39ebc291

참고로 기존에 제안된 버킷 값을 이용한 튜플 암호화 방식에서 신장의 경우만 적용한 경우는 표5와 같으며, 신장에 대한 튜플 암호화 값으로 저장되어 있으며, 이러한 정보를 이용하여 신장에 대한 정보를 추정할 수 있다는 단점이 있다.

표 5. 암호문 데이터베이스
Table 5. The encrypted database

이름	암호화 값	신장
Ann	E (A, 165, 57, 서울)	δ
Bob	E (O, 155, 52, 대전)	β
Carol	E (O, 171, 76, 대구)	ζ
David	E (B, 165, 63, 광주)	δ

본 논문에서 제안하는 방식을 설명하기 위하여 신장 데이터에 대한 다음과 같은 예를 이용한다.

(예제) A 병원의 환자 의료정보에서, 신장 데이터에 대한 분포를 8 개의 등급으로 나누었을 경우와 4개의 등급으로 나누었을 경우에 다음과 같다.

표 6. 8 등급 데이터
Table 6. The eight Class data.

신장 (하한, 상한)	인원
145-150	5
150-155	15
155-160	110
160-165	200
165-170	250
170-175	120
175-180	45
180-185	15

표 7. 4 등급 데이터
Table 7. The four class data

신장 (하한, 상한)	인원
145-155	20
155-165	310
165-175	370
175-185	60

버킷 값을 이용한 튜플 암호화 방식을 적용하는 경우에는 사용자의 신장 정보에 대한 통계적인 노출을 방지하기 위하여 8등급 데이터(표6)를 사용하는 것보다 대부분 4등급 데이터(표7)를 사용한다. 그러나 버킷 방식에 블룸필터를 적용한 본 논문에서의 제안방식을 사용하는 경우에는 8등급 이상으로 세분화된 버킷 방식을 사용할 수 있다. 왜냐하면 버킷 값을 블룸필터에 적용함으로서, 사용자의 정보가 노출되지 않으며, 필요한 범위의 false positive를 보장할 수 있기 때문이다.

이와 같이 본 논문에서 제안한 블룸필터에 의해 저장된 암호화된 의료정보에 대한 복호화 과정은 다음과 같다.

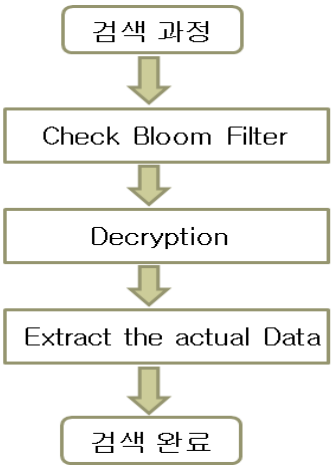

그림 4. DB 검색과정
Fig. 4 The DB search process

(1) 블룸필터 체크과정

먼저 클라이언트는 메타데이터에 수록된 신장데이터에 대한 해당 버킷 값을 읽고, 이에 대한 해쉬함수

를 계산하고, 블룸필터에 해당 비트가 설정된 데이터를 서버에 요구한다. 서버는 해당 비트가 설정된 암호화 튜플들을 클라이언트로 보낸다.

(2) 복호화 과정
클라이언트는 서버로부터 받은 암호화 데이터를 복호 한다.

(3) 최종값 추출
복호된 평문데이터로부터, 질의어에 해당되는 적합한 결과들을 추출하여 사용자에게 응답한다.

이러한 과정을 자세히 살펴보면, 실제 복호화 과정에서 시간이 가장 많이 소요된다. 즉, 서버로부터 과도한 암호화된 데이터를 수신하면, 이를 복호하는데, 많은 시간이 소요된다. 결과적으로 본 논문에서 주장하는 바와 같이 버킷에 대한 정보 노출을 방지하기 위하여 블룸필터에 적용하고, 이는 곧 버킷의 개수를 증가시켜서 (2)번 복호화 과정의 속도를 높일 수 있게 된다. 결국 버킷의 수를 l배 증가시키면, 복호화시간은 $1/l$배로 단축되며, 복호화 성능은 l배로 향상된다.

Ⅳ. 제안 방법에 대한 분석

4.1. 버킷개수와 성능과의 관계

일반적으로 버킷 수를 증가시키면, 검색 성능은 향상되지만, 데이터의 분포를 알 수 있다는 단점을 가진다. 즉, 버킷의 수를 증가시킨다는 것은 보다 적은 구간단위로 요소들을 묶을 수 있기 때문이다. 따라서 본 논문에서는 이러한 범위형 데이터를 다시 블룸필터에 적용하여, 블룸필터값 만을 저장함으로서, 특정인의 신체 정보가 노출되지 않도록 한다.

전체 구간 t를 b개의 버킷으로 분류한 경우, 각 버킷 내의 요소의 개수는 t/b가 된다.

동일 값을 검색하는 경우에는 해당 값을 포함하는 1개의 버킷을 읽어 들이고, 이를 복호하여 최종 값을 얻을 수 있다. 이와 같이 버킷 내에서 동일 값을 찾을 확률은 $p = b/t$ 이며, 버킷 내에서 동일 값을 찾을 수 없는 false positive 확률은 $p = 1 - b/t$이다.

범위검색의 경우에는 특정 범위 $[r_{lo}, r_{hi}]$를 포함하는 버킷들을 읽어 들이고, 이를 복호하여 최종 값을 얻을 수 있다. 범위의 하한구간이 포함되는 버킷 R_{lo}로부터 상한구간이 포함되는 버킷 R_{hi}를 이용한다. $R_{hi} - R_{lo} + 1$개의 버킷을 읽어 들이고, 이를 복호하여 최종 값을 얻을 확률은 식2와 같다.

$$p_b = (r_{hi} - r_{lo} + 1)/(R_{hi} - R_{lo} + 1) \quad (2)$$

이러한 확률 p_b는 특정구간내의 false positive가 존재하지 않는 n 개의 구간과 false positive가 존재하는 하한과 상한, 2개의 구간으로 표시하면 식3이 된다.

$$p_b = (n + r)/(n + 2) \quad (3)$$

여기서 r 값은 버킷경계에 포함되지 않는 상, 하한구간의 값으로서 $0 < r < 2$의 값을 갖는다. 그러므로 범위검색에 대한 false positive 확률은 식4와 같다.

$$p = 1 - (n + r)/(n + 2) \quad (4)$$

버킷의 개수 b가 증가되면, 특정구간내의 버킷의 개수인 n값이 증가되며, 범위검색에 대한 false positive 확률은 0에 근접함을 알 수 있다.

버킷의 개수가 증가되면, false positive 확률이 낮아질 뿐 아니라, 데이터베이스 서버와 클라이언트간의 통신양이 줄어들고, 클라이언트에서 복호하여야 할 데이터 개수가 급격히 줄어들기 때문에 검색성능이 향상됨을 알 수 있다. 일반적으로 암호화된 데이터에 대한 검색을 위하여 버킷인덱스만을 사용하는 경우에는 버킷인덱스 값으로 암호화된 정보를 개략적으로 짐작할 수 있다는 단점이 있다. 따라서 본 논문에서는 버킷의 개수를 증가시켜 검색성능을 높이고, 데이터 분포노출을 방지하기 위하여 다시 블룸필터를 적용하는 방법을 사용한다.

4.2. 버킷의 개수와 보안과의 관계

버킷화 과정에 대한 버킷의 개수와 성능과의 관계를

분석한 논문은 다수 있지만, 버킷의 개수와 보안과의 관계를 정밀하게 분석한 논문은 Hore가 작성한 논문이 유일하다[9]. 논문에서는 버킷 값을 통한 데이터의 누출을 방지하기 위하여, 범위질의어에 대한 최적 버킷화 알고리즘(Query Optimal Bucketization Algorithm)을 제안하였다.

버킷의 개수를 일정하게 주어진 상태에서 단위 버킷 내의 요소의 개수 N과 각 요소별 빈도 F를 이용하여 계산되는 false positive 값인 $\sum_{B_i} N_{B_i} \times F_{B_i}$ 를 최소화 하는 알고리즘이다. 결국 버킷별 크기를 가변적으로 구성하여, false positive의 개수를 최소화하는 방법을 제안하였다. 이와 더불어 최적 버킷화 과정을 거친 버킷 구조에 대해 표준편차와 엔트로피를 최대화함으로서, 공격에 강한 알고리즘을 제안하였다. 그러나 본 논문에서는 버킷화 과정을 거친 결과에 대해 블룸필터에 다시 적용함으로서, 버킷의 분포를 공격자가 알 수 없도록 제안한다.

4.3. 블룸필터에서의 성능

블룸필터는 최소의 메모리 공간을 이용하여 데이터의 존재유무만을 확인할 수 있도록 하는 방법이다. 그러나 블룸필터의 특성상, false positive의 확률이 존재한다. false positive 란 실제로 존재하지 않지만, 존재하는 것으로 표시되는 오류를 말한다. 블룸필터는 m개의 비트를 가진 비트벡터(bit vector) 이며, n 개의 요소를 가진 유한 집합 $S = \{x_1, x_2, ..., x_n\}$에 멤버십 테스트를 수행할 수 있도록 해준다. 또한 블룸필터는 서로 독립적인 k 개의 균등한(uniform) 해시함수를 사용한다.

$$H(x) = \{h_i(x) | 1 \le h_i(x) \le n \text{ for } 1 \le i \le k\}$$

블룸필터의 false positive 확률은 $f = (1 - e^{-kn/m})^k$로 표시되며, 이러한 확률이 최소가 되는 해쉬함수의 개수 $k = \frac{m}{n} \ln 2$가 되며, 이때 false positive 확률은 다음과 같다[9,10].

$$f = (\frac{1}{2})^k \approx (0.6185)^{\frac{m}{n}} \quad (5)$$

본 논문에서는 버킷 방식에서의 버킷을 검색어로 사용하기 때문에 블룸필터의 false positive 확률을 적용시키기 위하여, 요소의 개수 n 대신에 버킷의 개수 b를 대입하고, 전개하면 다음 식과 같다.

$$f = (1 - e^{-kb/m})^k \quad (6)$$

그러므로 본 논문에서 적용된 버킷 인덱스 확률을 블룸필터에 적용시키면 그림5와 같다.

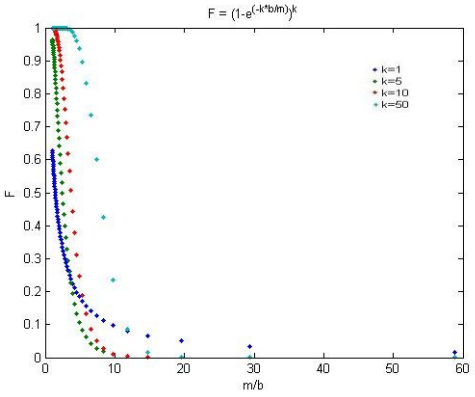

그림 5. m/b 비율에 따른 false positive
Fig. 5 False positive as a function of the ratio of m/b

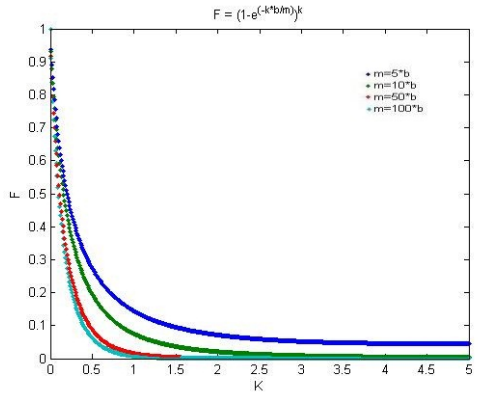

그림 6. 해쉬함수 개수에 따른 false positive
Fig. 6 False positive as a function of the number of hash functions

이와 같은 결과를 통하여, 버킷방식만을 적용한 경우에 비하여 버킷방식과 블룸필터방식을 적용한 경우에는 버킷의 개수를 늘일 수 있고, 이에 따른 사용자 데이터의 분포 노출을 방지할 수 있으며, 결과적으로 검색속도를 높일 수 있음을 알 수 있다.

4.4. 비교 분석

4.1절에서 범위형 데이터에 대한 튜플 암호화와 함께, 버킷인덱스만을 사용한 경우와 4.2절에서 버킷인덱스와 블룸필터를 적용한 경우를 살펴보았다. 버킷인덱스만 사용하였을 경우에는 버킷의 개수를 증가시키면 성능이 향상되는 반면, 데이터 분포에 대한 노출현상이 증가된다. 즉, 버킷인덱스만을 사용하는 경우에는 암호화된 데이터와 버킷 값으로 데이터의 분포와 함께, 데이터를 추정할 수 있는 위험이 발생된다. 그러나 버킷인덱스와 함께, 블룸필터를 적용한 경우에는 버킷인덱스에 대한 메타데이터는 사용자 클라이언트 모듈에서 보관하고, 외부 전문 업체에서 보관하는 데이터베이스에는 암호화 데이터와 블룸필터만을 보관함으로서 버킷에 대한 정보를 알 수 없도록 할 수 있다.

또한 버킷에 대한 정보가 서버 상에 블룸필터의 값으로 보관되기 때문에, 기존의 버킷방식의 암호화 방식에 비해 블룸필터를 혼합하여 적용함으로서, 버킷의 개수를 증가하여 검색속도를 높일 수 있다는 장점이 있다.

동일 값 검색확률의 경우에는 버킷구간을 세분화함으로서, 세분화된 구간 내에서 동일 값을 찾기 때문에 보다 빠른 속도로 처리할 수 있다. 마찬가지로 범위검색인 경우에도 보다 세분화된 구간 내에서 위치를 찾기 때문에 빠른 속도로 처리할 수 있다는 장점을 가진다. MIN/MAX 검색의 경우에도 마찬가지로 세분화된 구간 내에서 최대, 최소값을 찾는데 적용된다.

Ⅴ. 결 론

본 논문에서는 버킷방식과 블룸필터 방식을 혼용하여 범위 데이터에 대한 검색방법을 제안하고 검토하였다.

블룸필터 방식에서 false positive 확률이 존재하는 것과 마찬가지로 버킷 방식에서도 해당 버킷 내에서의 정확한 결과를 얻기 위해 false positive 확률이 존재한다.

4.1절에서 살펴본 바와 같이 버킷방식의 경우에는 버킷의 개수를 증가시키면 false positive 확률이 낮아질 뿐 아니라, 데이터베이스 서버와 클라이언트간의 통신양이 줄어들고, 클라이언트에서 복호하여야 할 데이터 개수가 급격히 줄어들기 때문에 검색성능이 향상됨을 알 수 있다. 일반적으로 암호화된 데이터에 대한 검색을 위하여 버킷인덱스만을 사용하는 경우에는 버킷 인덱스 값으로 암호화된 정보를 개략적으로 짐작할 수 있다는 단점이 있다. 따라서 본 논문에서는 버킷의 개수를 증가시켜 검색성능을 높이고, 데이터 분포노출을 방지하기 위하여 다시 블룸필터를 적용하는 방법을 사용하였다.

본 논문에서 살펴본 바와 같이 블룸필터의 크기 m, 해쉬함수의 개수 k, 버킷의 개수 b를 이용하여 false positive 확률을 조절할 수 있다. 버킷의 개수를 2배로 증가시키면, 복호화시간은 2배로 단축시키며, 복호화 성능은 2배로 향상된다. 이를 블룸필터에 적용하여 false positive 확률을 적절히 선택함으로서, 성능과 보안간의 절충점을 찾을 수 있다. 이와 같은 결론은 보안을 중요시 하는 환경에서는 버킷의 개수를 증가시킴과 동시에, 비교적 높은 false positive 확률을 선택하고, 성능을 우선시하는 환경에서는 버킷의 개수를 증가시킴과 동시에 상대적으로 낮은 false positive 확률을 선택할 수 있다.

따라서 본 논문에서는 비교적 간단한 방법으로 버킷방식을 블룸필터 방식에 이론적으로 적용하였으나, 이러한 이론을 기반으로 범위형 데이터 보안을 위한 방안으로서 적용가능하며, 향상된 데이터베이스 보안에 적용될 수 있을 것으로 판단된다. 향후에는 문장형 데이터를 포함한 블룸 필터를 의료 데이터베이스에 적용하기 위한 연구를 진행할 예정이다. 이러한 이론을 기반으로 하여 데이터베이스의 보안기능을 유지하며, 검색성능을 향상시키기 위한 기초연구로서 활용될 수 있을 것으로 기대된다.

참고문헌

[1] Hakan Hacigümüs, Bala Iyer, and Shrad Mehrotra, "Providing database as a service", In Proc, of the 18th International Conference on Data Engineering, San Jose, California, USA, IEEE Computer Society, pp.21-28., 2002.

[2] 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, pp.45-49, 2002.

[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, pp.12-21, 2003.

[4] 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, pp.123-132, 2004.

[5] 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.

[6] 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, pp,89-96, 2002.

[7] Rakesh Agrawal, Jerry Kiernan, Ramakrishnan Srikant, Yirong Xu, "Order preserving Encryption for numerical Data", in Weikum, G., Konig, A., and Debloch,s., Eds., Proc. of the ACM SIGMOD 2004, Paris, France. ACM, pp.78-84, 2004.

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

[9] Bijit Hore, Sharad Mehrotra, Gene Tsudik, "A Privacy-Preserving Index for Range Queries", Proceeding of the 30th VLDB Conference, Toronto, Canada, pp,103-109, 2004.

[10] Rafael P. Laufer, Pedro B. Velloso, and Otto Carlos M. B. Duarte, "Generalized Bloom Filters, Technical Report", GTA-05-43, COPPE/UFRJ, September, pp.143-150, 2005.

김창규 (Chang-kyu Kim)

1989년 한양대학교 대학원 전자통신공학과 박사
2005년8월~2007년7월: 동의대학교 공과대학 부학장

1988년3월~현재 동의대학교 정보통신공학과 교수
2006년11월~현재 동의대학교 공학교육혁신센터장
※관심분야 : 정보보호, 이동통신

김정태 (Jung-tae Kim)

2001년 연세대학교 대학원 전자공학과 공학박사
1991년 8월~1996년2월: 한국전자통신연구원 선임연구원

2002년10월 ~ 현재 목원대학교 전자공학과 교수
※관심분야 : 정보보안, RFID&USN Security, Hardware Security, 의료정보보호

유천영 (Choun-young Yu)

2005년 세명대학교 컴퓨터과학과 졸업
2007년 세명대학교 전자계산 교육대학원 석사

2008년3월～현재 : 세명대학교 대학원 박사과정
※관심분야 : 정보보호, 데이터베이스 보안, 네트워크 보안

김지홍 (Ji-hong Kim)

1982년 한양대학교 전자공학과 학사
1984년 한양대학교 대학원 전자통신공학과 석사

1996년 한양대학교 대학원 전자통신공학과 박사
1991년3월～현재 세명대학교 정보통신학부 교수
※관심분야 : 네트워크 및 정보보호, 의료정보 데이터베이스 보안