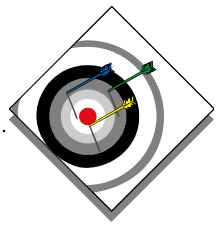


데이터베이스 암호화와 검색 기술 동향

이승민* 이동혁** 남택용*** 장종수**** 염흥렬*****

개인정보의 유출에 따른 개인정보보호 문제가 사회적 이슈가 됨에 따라 최근 데이터베이스 보안에 대한 논의도 활발히 진행되고 있다. 데이터베이스 보안을 위한 핵심 기술 가운데 데이터베이스 암호화 기술은 내부의 데이터를 보호할 수 있다는 측면에서 매우 매력적인 기술로 각광받고 있으나, 암호화를 통한 보안성 확보는 데이터베이스의 검색 속도를 현저히 저하시키는 문제를 야기함으로써 현실적으로 데이터베이스에 암호화 기술을 적용하는 데 많은 제약이 되고 있다. 본 고에서는 데이터베이스 암호화와 검색 알고리즘에 대한 기존의 연구 결과를 바탕으로 알고리즘별로 특징과 장단점을 분석함으로써, 데이터베이스 암호화 시 고려해야 할 점과 향후 연구 방향을 짚어보기로 한다. ▨

목 차

Ⅰ. 서 론

Ⅱ. DB 모델 및 고려사항

Ⅲ. DB 암호 알고리즘

Ⅳ. 비교 분석

Ⅴ. 결 론

* ETRI 개인정보보호연구팀/선임연구원
** ETRI 개인정보보호연구팀 /연구원
*** ETRI 개인정보보호연구팀 /팀장
**** ETRI 보안응용그룹/그룹장
***** IITA 정보보호전문위원실/PM

Ⅰ. 서 론

개인정보 유출사고의 증가와 개인정보보호법의 시행 요구에 따라 데이터베이스 보안에 대한 관심이 날로 증가하고 있다. 데이터베이스 보안을 위한 핵심 기술로서 데이터베이스 암호화가 있다.

데이터베이스 암호화란 데이터베이스 내부의 데이터 자체를 암호화하는 기술이다. 암호화 기술은 특히 내부자로부터 데이터를 보호할 수 있다는 면에서 매우 매력적인 기술이다. 그러나 데이터베이스에 암호화를 적용하면, 기존 데이터베이스에서 제공하는 고유 기술 가운데 하나인 인덱싱 기술을 사용할 수 없기 때문에 검색 속도가 현저히 떨어지는 문제를 야기한다[1].

데이터베이스 암호화에 따른 검색 성능을 해결하고자 하는 시도는 2000 년 이후부터 학계를 중심으

로 많은 연구가 진행되어 오고 있다[2]. 연구 방향은 크게 데이터베이스 커뮤니티를 중심으로 하는 실용성을 강조한 접근 방법과 암호 커뮤니티를 중심으로 하는 알고리즘의 안전성을 강조한 접근 방법이 있다. 그러나 암호 커뮤니티를 중심으로 한 연구는 아직 초보적인 단계이고 실용성이 많이 부족하다고 판단되어, 본 고에서는 데이터베이스 커뮤니티 중심의 기존 연구를 바탕으로 한 DB 암호화 및 검색 기술에 대하여 살펴보기로 한다.

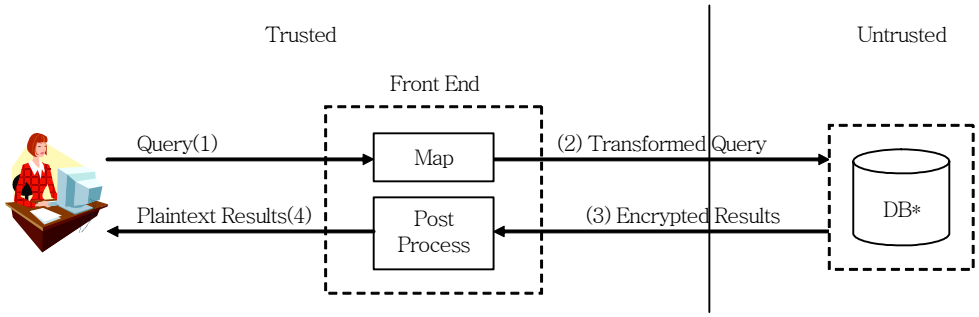
본 고는 Ⅱ장에서 DB 암호화에 사용되는 모델과 암호 알고리즘 선택 시 고려 사항을 알아보고, Ⅲ장에서는 기존 DB 암호화 및 검색 알고리즘의 특징과 장단점을 분석한다. Ⅳ장에서는 Ⅱ장에서 제시한 기준에 따라 Ⅲ장에서 분석한 알고리즘별로 특징을 비교 분석한다. 마지막으로 Ⅴ장에서 결론을 맺는다.

Ⅱ. DB 모델 및 고려 사항

본 절에서는 데이터베이스 암호화에 적용되는 모델과 고려 사항에 대해서 살펴본다.

1. DB 모델

데이터베이스 암호화에서 일반적으로 사용되는 모델은 (그림 1)의 DAS(Database As a Service) 모델이다[3]. DAS 는 데이터를 저장하는 영역을 외부에 맡기는 데이터베이스 아웃소싱 개념이다. 즉, 암호화된 데이터는 외부의 DBMS 영역에서 관리하며 이 영역은 신뢰할 수 없다고 가정한다. 그리고 Front-End 부분은 사용자의 검색 요청에 대하여 암호화된 검색어로 변경하고, 암호화된 검색 결과를 복호화하는 일을 수행한다. 사용자와 Front-End 영역은 서로 신뢰한다고 가정한다. 3rd 파티에서 제공하는 DB 암호화 제품도 DAS 모델과 유사하다고 볼 수 있다.



(그림 1) DAS 모델

2. 고려 사항

데이터를 DBMS 에서 안전하게 관리하기 위해서는 암호 알고리즘의 안전성이 고려되어야 한다. 그리고 암호화의 경우에 DB 검색 시 속도 저하를 최소화하고, 검색 기능을 최대한 많이 제공할 수 있는 암호 알고리즘이어야 한다. 알고리즘의 안전성은 암호화 DB 에서 검색의 기능성과 상반되는 점이 있는데, 암호 알고리즘의 안전성을 높이면 기능성이 떨어지고 반대로 기능성을 높이면 안전성이 낮아지는 경우가 일반적이다.

가. 알고리즘의 안전성

암호화 DB 를 효율적으로 검색하기 위해서는 기존에 알려진 표준 암호 알고리즘(DES, AES 등)과는 차별된 알고리즘이 필요하다. 이 알고리즘은 데이터베이스 컬럼을 직접 암호화하거나 인덱스를 암호화하기 위하여 사용될 수 있다. 특히, 데이터베이스를 위한 암호 알고리즘은 안전성이 고려되어야 하는데, 일반적으로 암호 알고리즘의 안전성을 평가하기 위해서 알고리즘 자체의 안전성에 초점을 맞추어 다음과 같은 네 가지 공격 모델을 적용하기로 한다. 즉, 데이터베이스에 암호 알고리즘을 적용했을 경우에는 알고리즘 자체의 안전성과는 별도로 구현상의 안전성도 있지만, 이 부분은 제외하기로 한다.

- Ciphertext Only Attack: 공격자가 암호 알고리즘과 암호문만 획득하여 암호 분석을 시도하는 경우로 암호화된 특정 테이블이나 컬럼을 획득하여 평문을 유추할 수 있는 경우임
- Known Plaintext Attack: 일부 평문(m_i), 암호문(c_i)의 쌍(m_i,c_i)과 복호화해야 할 암호문 c 가 공격자에게 주어졌을 때, 공격자는 c 에 해당되는 평문을 유추하려는 경우와 평문에 대한 통계적 정보(도메인, 빈도수)를 이미 알고 암호문 c 에 대한 평문을 유추하려는 경우임
- Chosen Plaintext Attack, Chosen Ciphertext Attack: 각각 공격자가 평문을 임의로 선택해 해당하는 암호문을 볼 수 있거나 공격자가 암호문을 임의로 선택해 이에 해당하는 평문을 볼 수 있는 경우로서, 공격자는 KPA 에서 할 수 있는 공격 수단을 포함하여 그 이상의 공격수단을 고려할 수 있음

나. 알고리즘의 기능성

DB 암호화 알고리즘을 적용했을 경우 패턴 일치검색과 집계 검색이 어떤 기능에 대하여 얼마나 빠른 속도로 검색 결과를 제공할 수 있는지를 기준하여 평가할 수 있다.

- 패턴 일치(pattern matching) 기능으로 크게 일치검색, 범위검색, 전방일치 등이 있음
- 집계 검색(aggregation query)으로 Min, Max, Count, Sum, Avg 등이 있음

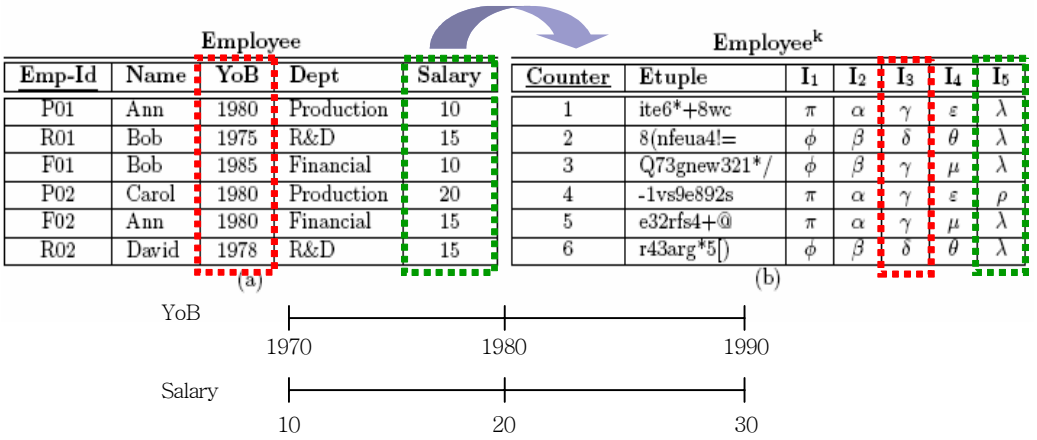
III. DB 암호 알고리즘

본 절에서는 기존에 학계, 산업체, 연구소를 중심으로 연구되어 온 데이터베이스 암호화 및 검색 알고리즘에 대하여 특징을 분석하였다.

1. Bucket-based index[3]

가. 개요

- 암호화 DB 검색 알고리즘에 관한 최초의 시도로서 2002 년 Hacigümüs 가 제안함
- 인덱스를 생성하기 위하여 별도의 인덱스 컬럼을 생성하는 방법임



(그림 2) Bucket-based index 예

나. 알고리즘

- 테이블에 있는 원래의 컬럼값들을 Etuple 이라는 컬럼에서 일괄적으로 표준 암호 알고리즘으로 암호화하고, 버킷 기반의 인덱스 방법으로 인덱스에 사용될 컬럼을 추가로 생성함
- 매핑함수를 정의하여 (그림 2)와 같이 데이터의 범위를 지정된 크기들로 나누는 버킷팅을 한 다음, 동일한 버킷 내에 있으면 동일한 버킷 ID 를 할당함

다. 장단점 분석

(1) 장점

- 매핑함수를 잘 정의하면 숫자와 문자에 대하여 모두 적용 가능함

(2) 단점

- 일치검색의 경우, 정확하게 일치하는 평문값을 알기 위해서 추가 필터링이 필요함
- 범위검색의 경우, 범위 내의 값이 포함된 버킷을 검색하고, 이어서 버킷 내의 값을 모두 복호화하는 재필터링 과정이 필요하므로 실질적으로 범위검색이 된다고 볼 수 없음
- 버킷정보가 known plaintext attack 이상의 공격에 이용될 수 있기 때문에 안전성을 보장할 수 없음. 그리고 버킷의 크기를 어떻게 정하느냐가 검색의 효율성과 안전성 측면에서 서로 배타적으로 작용할 수 있음

2. Hash-based index[4],[5]

가. 개요

(1) bucket-based index 방법과 유사점

- 인덱스에 사용될 컬럼을 해시함수로 암호화하고, 해시값이 버킷 ID 역할을 함
- 일치검색과 범위검색을 할 때 추가 필터링이 필요할 수 있음
- 숫자와 문자에 상관없이 적용 가능

(2) bucket-based index 방법과 차이점

- 일방향 해시함수를 이용함
- collision 이 없는 해시함수를 사용하면, 정확한 일치검색이 가능하나 유추 공격에 취약함
- collision 이 있는 해시함수를 사용하면, 검색 결과에 대해 추가 필터링이 필요함. 즉, 검색 결과가 bucket-based index 결과와 유사하나 랜덤하게 동일한 값을 부여하므로 유추 공격에 대하여 안전성을 높일 수 있음

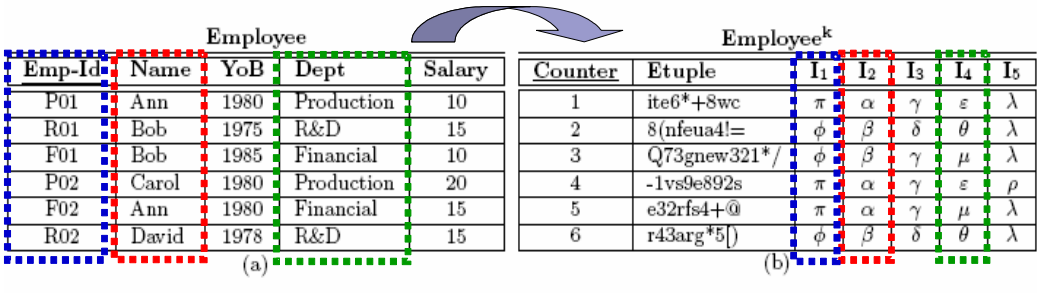
나. 알고리즘

- collision 이 없는 해시함수를 사용하면 정확한 일치검색이 가능하나, 고의로 collision 이 있는 해시함수를 사용하여 안전성을 높이려는 시도임

다. 장단점 분석

(1) 장점

- bucket-based index 에 비하여, 안전성이 높은 일치검색을 지원함
- 숫자와 문자 모두 적용 가능함



Employee

Emp-Id	Name	YoB	Dept	Salary
P01	Ann	1980	Production	10
R01	Bob	1975	R&D	15
F01	Bob	1985	Financial	10
P02	Carol	1980	Production	20
F02	Ann	1980	Financial	15
R02	David	1978	R&D	15

(a)

$Employee^k$

Counter	Etuple	I_1	I_2	I_3	I_4	I_5
1	ite6*+8wc	π	α	γ	ε	λ
2	8(nfeua4!=	ϕ	β	δ	θ	λ
3	Q73gnew321*/	ϕ	β	γ	μ	λ
4	-1vs9e892s	π	α	γ	ε	ρ
5	e32rfs4+@	π	α	γ	μ	λ
6	r43arg*5[)	ϕ	β	δ	θ	λ

(b)

(그림 3) Hash-based index 예

(2) 단점

- 일치검색에 대해서만 안전성을 높인 인덱싱 방법이며, 범위검색은 지원하지 않음. 일치검색에 대해서도 추가 필터링이 필요할 수 있음
  * 즉, 제안 방법은 x=y 이면 h(x)=h(y) 가 될 뿐만 아니라, collision 이 있는 경우 x≠y 이라도 h(x)=h(y) 인 경우가 발생하므로, 결과적으로 암호화된 중복값의 평문이 같은지 다른지 확신할 수 없으므로 안전성을 높이자는 것임
  * 반면, 표준 암호 알고리즘을 사용하면, x=y 이면 E(x)=E(y)이므로, 결과적으로 암호화된 값들의 평문이 같으므로 빈도수를 이용한 유추 공격에 취약함

3. B+tree index[4]

가. 개요

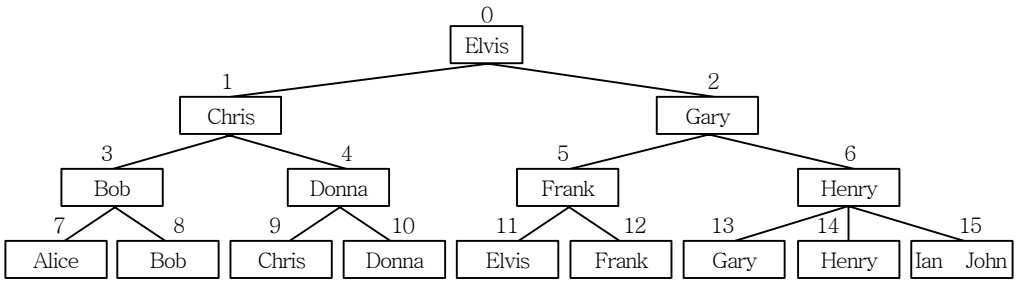
- 평문의 순서대로 B+ tree 를 별도로 구성하여 인덱스로 사용하고자 하는 경우, B+ tree 상에 평문의 정보가 그대로 노출되어 데이터 순서 기반 추론 공격이 용이하게 됨. 이러한 문제를 해결하기 위한 방법으로, B+ tree 를 구성하는 작업을 신뢰된 Front-End 에 맡기는 방법이 있음
- Damiani 가 제안한 B+ tree 기법은 이와 같이 B+ tree 의 구성과 traversal 을 신뢰된 Front-End 만 가능하도록 하여 데이터를 안전하게 보호하자는 방법임

나. 알고리즘

(1) 암호화 단계

① (그림 4 (a))와 같이 신뢰된 Front-End 가 평문 데이터에 대한 B+ tree 인덱스를 구성

② 각 B+ tree 에 ID 를 부여하여 혹은 인덱스에 생성된 ROWID 를 가져와서 (그림 4 (b))의



(a)

B+ -tree Table

ID	Node
0	(1, Elvis, 2, _, _)
1	(3, Chris, 4, _, _)
2	(5, Gary, 6, _, _)
3	(7, Bob, 8, _, _)
4	(9, Donna, 10, _, _)
5	(11, Frank, 12, _, _)
6	(13, Henry, 14, _, _)
7	(Alice, _, 8)
8	(Bob, _, 9)
9	(Chris, _, 10)
10	(Donna, _, 11)
11	(Elvis, _, 12)
12	(Frank, _, 13)
13	(Gary, _, 14)
14	(Henry, _, 15)
15	(Ian, John, _)

Encrypted B+ -tree Table

ID	C
0	SeCS0U/7ZIY.A
1	/WKu5y8laqK82
2	jzKzVi0Dlas8E
3	AXYaqohgyVObU
4	IUf7R.PK5h5fU
5	rzaslxohWS212
6	EXITGCUIYTVBc
7	uOtdm/HDXNSqU
8	GLDWRnBGIvYBA
9	a9yl36PA3LeLk
10	H6GwdJpXiU8MY
11	uOtdm/HDXNSqU
12	zj33kVaNvLFVk
13	V9rMw904cix3w
14	xTFcWtd6.IE.A
15	ji.gtDER6Hjis

(b)

(그림 4) B+ tree index 예

좌측과 같이 구성

③ ①에서 구성한 B+ tree 의 각 속성값을 암호화함. 단, 이 경우 좌우측 노드와 데이터가 모두 암호화되어 하나의 암호값으로 구성됨

④ 각각의 암호값을 ID 와 함께 (그림 4 (b))의 우측과 같이 구성함

(2) 복호화 단계

① 사용자로부터 검색이 요청되면, 신뢰된 Front-End 는 일련의 노드들을 액세스하기 위한 여러 개의 쿼리를 생성함

② 생성한 쿼리를 통하여 데이터베이스에 질의함으로써, 요청된 정보를 가져올 수 있음

다. 장단점

(1) 장점

- B+ tree 노드의 내용이 신뢰되지 않은 DBMS 에 보이지 않음
- 빈도수 기반 추론 공격에 안전함

(2) 단점

- B+ tree traversal 은 신뢰된 Front-End 에 의해서만 수행될 수 있음
- 데이터의 삽입, 수정, 삭제가 빈번할 경우, B+ tree 의 재생성이 필요함
- R 개의 데이터를 검색하기 위하여 R+N 만큼의 SQL 질의가 요구되어, 다량의 데이터에 대하여 적용하는데 한계가 있음(R: 범위 내의 데이터 수, N: 트리 깊이만큼의 노드 수)
- 노드상의 데이터는 직접적으로 필요한 정보가 아님에도 불구하고 Leaf 까지 도달하기 위하여 반드시 복호화하여야 함

4. Random number based encryption[6]

가. 개요

- 기존 암호화 알고리즘을 사용하여 데이터베이스를 암호화할 경우, 평문의 순서를 유지할 수 없으므로 인덱싱이 불가능하며 범위검색 등 DBMS 고유기능을 사용하는 데 한계가 있음
- 제안 방법은 암호화 결과가 평문의 순서와 동일한 순서유지 암호화 방법임

나. 알고리즘

- 암호화 및 복호화 알고리즘

Encryption:

$E_K(1) := 1 + z_1$

$E_K(n+1) := E_K(n) + z_n + 1$

Decryption:

Input: Y

Output: $D_K(Y)$

Begin

i:= 0; W := Y;

while W > i do

begin i := i+ 1; W := W − z_i

endwhile
if W = i then return $D_K(Y)$:= i else output "Failure";
End.

- 암호화 알고리즘은 평문과 그에 해당하는 수만큼의 의사난수에 대한 전체 합으로 암호화된 데이터를 생성하는 방식임
- 복호화 알고리즘은 의사난수를 지속적으로 발생시켜 암호 데이터에 그 난수를 빼는 것으로 원래의 평문을 알 수 있음

다. 장단점

(1) 장점

- 암호화 데이터에 대한 업데이트 문제를 고려할 필요가 없음
- 암복호화 절차가 개념적으로 간단하여 구현이 용이함

(2) 단점

- 분포 추정을 통한 추론 공격이 가능함
- 각 의사난수의 합이 누적됨으로 매우 큰 오버헤드가 발생함
- 순서가 그대로 노출되어 다른 컬럼과 함께 사용되었을 때 평문 정보가 노출될 수 있음
- 정수만 취급 가능하여 실수에 대하여 암호화할 수 없음

5. Polynomial function based encryption[6]

가. 개요

- Random Number 기반의 open-form 알고리즘은 숫자의 크기만큼 의사난수를 생성하여 합하게 됨. 따라서, 수치가 증가함에 따라 암호화 값 또한 기하급수적으로 증가하게 되므로 매우 큰 오버헤드가 발생하게 되며 정수만 취급함
- 따라서, 이러한 문제를 해결하고자 실수에도 적용 가능한 단조 증가 함수를 사용한 순서유지 암호화 방법임

나. 알고리즘

(1) Single Encryption Function

- Single Encryption Function 은 다음과 같은 다항식 F(x)를 암호화 함수로 사용함

$$F(x) = C_n x^n + C_{n-1}x^{n-1} + \ldots + C_1 x + C_0$$

- 여기서, 보안 등급(security degree)는 F(x)에 사용되는 constants(계수값)에 의존함

(2) Multiple Encryption Function

- Single n 차원 다항식의 역함수를 찾는 것보다, 각각의 역함수를 갖는 n 개의 함수를 반복적으로 적용하는 것이 효율적일 수 있음
- Multiple Encryption Function 에 사용되는 각 함수들에 대한 역함수는 아래와 같음

Function	Inverse	Range
$f_1(x) = C_0x + C_1$ $f_2(x) = C_2x^2 + C_3$ $f_3(x) = C_6x^3 + C_7$	$f_{1-1}(y) = 1/C_0\, y - C_1/C_0$ $f_{2-1}(y) = + (y/C_2 - C_3/C_2)^{1/2}$ $f_{2-1}(y) = - (y/C_2 - C_3/C_2)^{1/2}$ $f_{3-1}(y) = (y/C_6 - C_7/C_6)^{1/3}$	For all x For $x \geq 0$ For $x \leq 0$ For all x

- 만약, $f_1(x)=C_0x+C_1$ 이고, $f_2(x)=C_2x^2+C_3$ 이며, $f_3(x)=C_6x^3+C_7$ 일 경우 Encryption Function 을 다음과 같이 구성할 수 있음

$$E(x)=f_3(f_2(f_1(x)))=C_6(C_2(C_0x+C_1)^2+C_3)^3+C_7$$

- Multiple Encryption 에서도, 각각의 f_i는 monotone 이 됨. 즉, $x_1>x_2$ 이면 $f(x_1)>f(x_2)$임

다. 장단점

(1) 장점

- 암복호화시 다항식의 계수 정보만 필요로 하므로 절차가 매우 간편함
- 실수체계에도 적용 가능하여 Random Number 방식의 한계점을 극복할 수 있음
- 암호화된 데이터에 대한 업데이트 문제를 고려할 필요가 없음

(2) 단점

- known plaintext attack 에 매우 취약함. 즉, n 차식의 다항식의 경우 (n+ 1)개의 평문과 암호문의 쌍만 있으면 다항식을 구성할 수 있음
- ciphertext only attack 에서도 평문의 분포에 따른 암호문의 분포가 다항식에 의해 결정되므로 추론 공격이 가능함
- 순서가 그대로 노출되어 다른 컬럼과 함께 사용되었을 때 평문 정보가 노출될 수 있음

6. OPES[7]

가. 개요

- OPES(Order Preserving Encryption Scheme)는 순서유지 암호 분야의 대표적인 알고리즘으로서 수치 데이터에 대하여 암호화 값의 순서를 유지함
- 수치 데이터의 경우 공격자는 암호문에 대하여 정확한 평문을 모르더라도 평문에 매우 근접한 값을 얻을 수 있다면 이것은 암호화의 큰 취약점이 될 수 있기 때문에 OPES 의 경우에는 암호문에 의한 공격에 대하여 평문을 유추하기 매우 어렵게 하는 것이 특징임

나. 알고리즘

- 타깃 분포를 미리 임의로 정의하고 입력분포(평문의 분포)를 타깃 분포로 변환하는 개념

(1) Model 단계

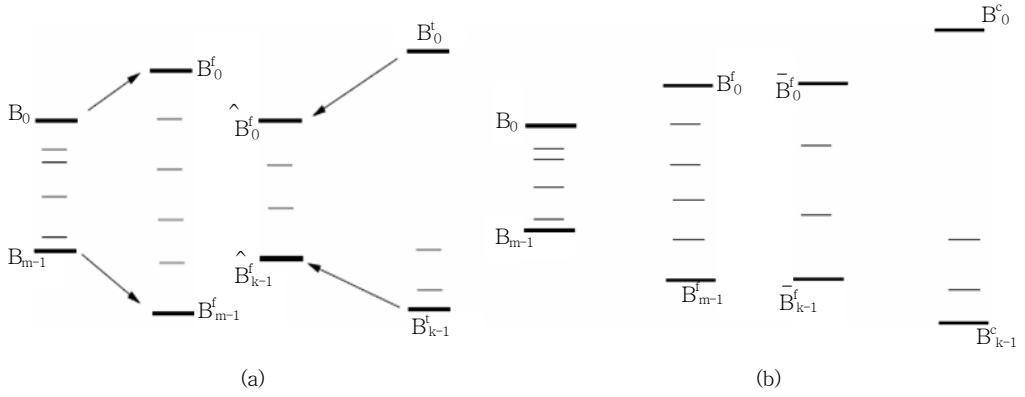
- 입력분포와 타깃 분포의 데이터를 버킷팅하여 각 버킷의 분포를 piecewise linear splines 으로 모델화 함

(2) Flatten 단계

- 각각 모델링한 버킷 내의 값을 uniform 분포가 되도록 변환((그림 5)의 (a))
- 버킷이 m 개 일 때, m+ 1 개의 버킷 boundaries 와 flatten 에 필요한 m 개의 계수 및 m 개의 스케일값이 암호화 키가 됨

(3) Transform 단계

- 입력분포와 타깃 분포로부터 flatten 한 결과의 스케일을 같게 하여 변환함으로써, 입력분



(그림 5) OPES 동작 흐름

포의 데이터가 타깃 분포를 따르는 데이터로 변환됨((그림 5)의 (b))

다. 장단점 분석

(1) 장점

- 순서유지 암호화이므로 암호화된 상태에서 일치검색, 범위검색, Min, Max, Count, GROUP BY, ORDER BY 등의 검색이 가능함
- 평문과 암호문의 분포를 다르게 함으로써, ciphertext only attack 에 대해 안전성이 높음

(2) 단점

- 숫자 데이터에 대해서만 적용됨
- known plaintext attack 이상의 공격에 대해서는 취약함
- 순서자체가 중요한 정보가 될 수 있으므로, 동일한 테이블에서 암호화되지 않은 컬럼과 함께 사용되거나 동일한 테이블에서 서로 다른 컬럼이 OPES 에 의해 암호화되었을 때 순서 통계량에 의해 평문의 정보가 노출될 수 있음

IV. 비교 분석

본 절에서는 Ⅱ 장의 고려사항에서 제시한 안전성과 기능성을 기준으로 Ⅲ 장에서 살펴본 암호화 DB 검색 알고리즘의 특징을 비교 분석한다. <표 1>은 알고리즘의 안전성을 <표 2>는 기능성을 비교한 것이다.

알고리즘은 크게 인덱스 암호화(Bucket, Hash, B+ tree 등)와 순서유지 암호화(Random

<표 1> DB 암호 알고리즘의 안전성 비교

algorithms	ciphertext only attack	known plaintext attack (CPA, CCA)
Bucket	동일한 버킷 ID 를 가지는 값들은 비슷한 범위에 있다는 정보로부터 평문의 정보를 유추할 수 있음	암호화된 동일 버킷 ID 정보로부터 평문의 범위를 보다 쉽게 유추할 수 있음
Hash	평문의 정확한 분포를 숨길 수 있음	일부 (m_i,c_i)쌍을 알더라도 동일한 평문이 서로 다른 해시값이 가지므로 안전성이 높음
B+ tree	평문의 정확한 분포를 숨길 수 있음	행전체를 암호화하므로 안전성이 높음
Random number	암호문의 분포와 평문의 분포 형태가 동일하므로 평문의 정보를 유추할 수 있음	일부 (m_i,c_i)쌍으로부터 키를 모르더라도 원하는 암호문 c 의 평문(또는 범위)을 유추할 수 있음
Polynomial	다항식에 따라 평문에 대한 암호문의 분포가 정해지므로 평문의 정보를 유추할 수 있음	일부 (m_i,c_i)쌍으로부터 다항식의 계수 즉, 키값을 쉽게 알 수 있으므로 안전성이 매우 낮음
OPES	암호문의 분포로부터 평문의 분포를 숨길 수 있으므로 안전성이 높음	일부 (m_i,c_i)쌍으로부터 키를 모르더라도 원하는 암호문 c 의 평문(또는 범위)을 유추할 수 있음

number, Polynomial, OPES 등)로 나눌 수 있다. <표 1>의 안전성 비교는 알고리즘 자체에 대한 것이며, 실제 DB에 적용할 때 발생하는 안전성은 제외하였다. 적용시 발생하는 예로 암호화된 컬럼이 암호화되지 않은 컬럼과 함께 사용됨으로써 노출되는 추가 정보(예; 순서유지암호의 순서정보, 버킷 기반 인덱스의 버킷정보 등)로 인한 취약성이 있을 수 있다. 따라서, DB 암호화시의 알고리즘의 안전성은 실제 적용하는 DB 환경에 따라 다를 수 있으므로 사용 환경에 따른 안전성 기준을 명확히 제시할 필요가 있다.

그리고 <표 2>의 암호 알고리즘 기능성을 비교할 때, 순서유지 암호화 방안이 인덱스 암호화 방안에 비하여 범위검색을 포함하여 다양한 검색 기능을 제공함을 알 수 있다. 이것은 데이터의 순서유지를 통하여 데이터베이스에서 제공하는 인덱싱 기능을 사용하기 때문이다. 그 외에도 데이터베이스 검색시 집계검색(aggregation query)으로 SUM, AVG 등에서 암호화된 데이터에 대하여 직접 연산이 가능한 PH(Privacy Homomorphism) 등에 대한 연구도 필요함을 알 수 있다. <표 1>과 <표 2>의 결과를 종합하여 보면, 순서유지 암호와 인덱스 암호를 비교했을 때 안전성과 기능성이 다소 상대적인 면과 상호 보완적인 면이 있다고 볼 수 있다.

<표 2> DB 암호 알고리즘의 기능성 비교

algorithms	data type			range query	aggregation query				
	int	real	char		min	max	count	sum	avg
Bucket	○	○	○	×	×	×	×	×	×
Hash	○	○	○	×	×	×	×	×	×
B+ tree	○	○	○	○	○	○	○	×	×
Random number	○	×	×	○	○	○	○	×	×
Polynomial	○	○	×	○	○	○	○	×	×
OPES	○	○	×	○	○	○	○	×	×

Ⅴ. 결 론

지금까지 DB 암호화시 고려사항과 암호화에 따른 검색 알고리즘의 장단점을 살펴보았다. 기존의 암호 알고리즘을 DB 에 적용할 때 발생하는 검색 속도 저하 문제를 해결하기 위하여 그동안 많은 연구가 시도되었다. 이러한 연구 결과를 현실적으로 적용하기 위해서는 검색 기능과 함께 사용 환경에 따른 안전성에 대한 대응 방안이 함께 고려되어야 할 것이다. 즉, 향후 DB 암호 알고리즘은 사용 환경에 적합한 안전성을 만족하면서 검색의 성능을 최대로 제공하는 방향으로의 연구가 필요하다.

DB 암호화 및 검색 알고리즘에 대한 연구는 궁극적으로 개인정보를 보호하는 데 핵심적인 역할을 수행함으로써 사회·경제적으로 막대한 파급 효과를 가져올 것으로 기대된다.

<참 고 문 헌>

[1] Noel Yuhanna, "The Forrester Wave: Database Encryption Solutions, Q3 2005," Forrester, August 8, 2005.

[2] Sabrina De Capitani di Vimercati, Sara Foresti, Stefano Paraboschi, Pierangela Samarati, "Privacy of Outsourced Data," 2006.

[3] H.Hacigümüs, B.R.Iyer, C.Li, and S.Mehrotra. "Executing SQL over Encrypted Data in the Database-service-provider Model," In Proc. of the ACM SIGMOD Conf. on Management of Data, Madison, Wisconsin, June 2002.

[4] E.Damiani, S.D.C.di Vimercati, S.Jajodia, S.Paraboschi and P.Samarati, "Balancing Confidentiality and Efficiency in Untrusted Relational DBMSS," In Proc. of the 10th ACM Conf. on Computer and Communications Security(CCS), October 2003.

[5] A.Ceselli, E.Damiani, S.D.C.D. Vimercati, S.Jajoda, S.Paraboschi and P.Samarati, "Modeling and Assessing Inference Exposure in Encrypted Databases," ACM Transactions on Information and System Security, Vol.8, No.1, Feb. 2005.

[6] S.C.Gulteckin Ozsoyoglu, David Singer, "Anti-tamper Database: Querying Encrypted Databases," In Proc. of the 17th Annual IFIP WG 11.3 Working Conference on Database and Applications Security, Estes Park, Colorado, August 2003.

[7] R.Agrawal, J.Kiernan, R. Srikant, Y.Xu, "Order Preserving Encryption for Numeric Data," In Proc. of the ACM SIGMOD Conf. on Management of Data, Paris, France 2004.

* 본 내용은 필자의 주관적인 의견이며 IITA 의 공식적인 입장이 아님을 밝힙니다.