Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.24, NO.4, Aug. 2014 ISSN 2288-2715(Online)
http://dx.doi.org/10.13089/JKIISC.2014.24.4.739

스마트폰 데이터베이스 환경에서 암호화된 데이터에 대한 효율적인 키워드검색 기법*

김 종 석,† 최 원 석, 박 진 형, 이 동 훈‡
고려대학교 정보보호대학원

An Efficient-keyword-searching Technique over Encrypted data on Smartphone Database*

Jong-Seok Kim,† Won-Suk Choi, Jin-hyung Park, Dong-hoon Lee‡
Graduate School for Information Security, Korea University

요 약

많은 사람들이 일상생활뿐만 아니라 다양한 업무에서 스마트폰을 활용하고 있다. 이에 따라 스마트폰에는 사용자의 개인정보 및 업무상의 민감한 정보가 저장된다. 그러나 스마트폰은 다양한 데이터 및 개인 정보를 데이터베이스에 평문 형태로 저장하고 있어 악성 앱이나 단말기 분실, 데이터 복원 등을 통하여 데이터베이스에 저장된 데이터 및 개인정보가 외부로 노출되고 2차 공격에 사용될 수 있다. 이러한 사용자의 정보유출 피해를 차단하기 위해서는 데이터베이스 암호화 기술이 필요하지만, 데이터베이스를 암호화하는 경우 성능저하를 야기 시킨다. 대표적인 예로, 암호화된 상태에서 특정 키워드를 통해 데이터를 검색하는 경우 모든 데이터를 복호화하거나 인덱스 정보 없이 순차적인 검색을 해야 하는 오버헤드가 발생한다[1].

본 논문에서는 데이터베이스가 암호화된 상태에서, 데이터를 검색할 때 발생하는 오버헤드를 최소화하기 위한 검색 가능한 암호 기법을 제안한다. 특히, 스마트폰과 같이 자원이 제한된 환경에서 사용되는 로컬 데이터베이스에 대해, 가변길이 블룸 필터를 사용하는 암호화된 데이터상에서의 효율적인 키워드 검색 기법을 제안하고 기존의 대칭키 방식의 검색 가능한 암호 기법들과 비교 분석한다. 그리고 안드로이드 스마트폰에서 제안하는 기법을 구현하여 제안 기법의 적절성과 성능을 검증한다. 구현을 통한 실험 결과, 본 논문에서 제안하는 방법이 암호화된 상태에서의 단순 검색에 비해 약 50% 이상의 검색 속도 향상 및 기존 고정 길이 블룸필터에 비해 동일한 긍정오류율 상에서 약 70% 이상의 공간을 절약할 수 있음을 확인할 수 있었다.

ABSTRACT

We are using our smartphone for our business as well as ours lives. Thus, user's privacy data and a company secret are stored at smartphone. By the way, the saved data on smartphone database can be exposed to a malicous attacker when a malicous app is installed in the smartphone or a user lose his/her smartphone because all data are stored as form of plaintext in the database. To prevent this disclosure of personal information, we need a database encryption method. However, if a database is encrypted, it causes of declining the performance. For example, when we search specific data in condition with encrypted database, we should decrypt all data stored in the database or search sequentially the data we want with accompanying overhead[1].

In this paper, we propose an efficient and searchable encryption method using variable length bloom filter under limited resource circumstances(e.g., a smartphone). We compare with existing searchable symmetric encryption. Also, we implemented the proposed method in android smartphone and evaluated the performance the proposed method. As a result through the implementation, We can confirm that our method has over a 50% improvement in the search speed compared to the simple search method about encrypted database and has over a 70% space saving compared to the method of fixed length bloom filter with the same false positive rate.

Keywords: Searchable Encryption, Bloom filter, Smartphone

I. 서 론

스마트폰은 단순히 통화나 메시지 기능뿐만 아니라 메일, 은행업무, 일정관리, 인터넷 검색 등 다양한 기능을 통해 편의성을 제공하여 현대 사회의 생활필수용품으로 자리 잡게 되었다. 이에 따라 일상생활 및 개인적인 업무에서 발생하는 민감한 데이터와 개인정보가 스마트폰에 저장되어 사용되고 있다. 스마트폰에는 연락처, 문자메시지, 음성녹음, e-mail, 위치 정보, 웹 방문기록 등 다양한 데이터 및 개인 정보가 데이터베이스에 저장된다. 이러한 정보들은 데이터베이스에 평문의 형태로 저장된다. 이로 인해 단말기를 분실하거나 악성앱 또는 악성코드가 스마트폰에 침입하는 경우 데이터베이스에 저장된 데이터 및 개인정보가 외부로 노출될 수 있기 때문에, 데이터의 기밀성을 위한 스마트폰 데이터베이스 암호화의 필요성이 대두되고 있다. 이와 함께 암호화된 데이터베이스의 성능 이슈에 대한 대책으로 암호화된 상태에서의 효율적인 검색 기법 또한 필수적으로 요구되고 있다[2,3,4,5,6].

검색 가능한 암호화 기법은 암호화되어 저장된 데이터를 복호화하지 않은 상태에서 원하는 데이터를 검색할 수 있도록 해주는 것을 말한다. 이를 위해 각 데이터 별로 검색을 위한 키워드를 설정하고, 사용자g용해 키워드를 적절한 형태로 변형하여 저장한다.

본 논문에서는 스마트폰과 같은 자원이 제한된 환경에서 암호화된 데이터베이스의 키워드 검색을 위해 가변 길이 블룸 필터를 이용한 키워드 검색 기법을 제안한다. 제안하는 방법은 형태소 분리기를 이용하여 검색 키워드를 설정하고, 암호화된 상태에서 해당 키워드로 검색할 수 있도록 키워드가 저장된 블룸 필터를 트랩도어로 사용한다. 기존의 블룸 필터를 이용한 암호화된 상태에서의 검색 기법[3,4,7]과는 다르게 본 논문에서는 스마트폰의 제한적인 연산속도와 한정된 저장 공간이라는 제약조건을 고려하여 가변길이의 블룸 필터를 사용한다. 블룸 필터는 길이가 작을수록 저장 공간의 효율성은 높아지지만 긍정오류율(false positive rate)도 높아지기 때문에 가변 길이 블룸 필터는 저장 공간과 긍정오류율 사이의 적정선을 고려하여 블룸 필터의 길이를 선택하는 것이 중요하다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구를 통해 블룸 필터와 검색 가능한 암호 기법에 관한 연구들을 살펴본다. 3장에서는 로컬 데이터베이스에서 가변 길이 블룸 필터를 이용하여 암호화된 데이터에 대해 효율적인 키워드 검색 기법을 제안한다. 4장과 5장에서는 본 논문에서 제안하는 방법을 기존 기법과 비교 분석하고, 실제 스마트폰에 구현하여 성능을 측정한다. 6장에서는 제안하는 기법과 관련된 여러 가지 논의점들을 살펴보고, 7장에서 결론을 맺는다.

II. 관련 연구

2.1 블룸 필터

블룸 필터는 원소가 집합에 속하는지 여부를 검사하는데 사용되는 확률적 구조이다[8]. 블룸 필터는 m 비트의 비트열을 '0'으로 초기화 한 후 k개의 해시함수 h_a: $\{0,1\}^* \rightarrow [1,m]$ $(1 \le a \le k)$ 를 이용하여 생성된다. 집합 $S = \{e_1, e_2, ..., e_n\}$ 에 속한 모든 원소 $e_i (1 \le i \le n)$ 에 대해 $h_1(e_i), h_2(e_i), ..., h_{k-1}(e_i), h_k(e_i)$ 를 계산하여 결과 값을 인덱스로 하는 블룸 필터의 비트를 '1'로 치환하여 색인을 설정한다. 이후에 어떤 원소 x 가 집합 S에 속하는지 확인하기 위해 $h_1(x), h_2(x), ..., h_{k-1}(x), h_k(x)$ 를 계산하고, Fig.1.와 같이 결과 값을 인덱스로 하P는 블룸 필터의 모든 비트가 '1'인지 체크한다. 모두 '1' 이라면 S의 원소로 판별하고, 그렇지 않으면 S의 원소가 아니라고 판별한다.

블룸 필터는 긍정오류율이 존재할 수 있다. 어떤 원

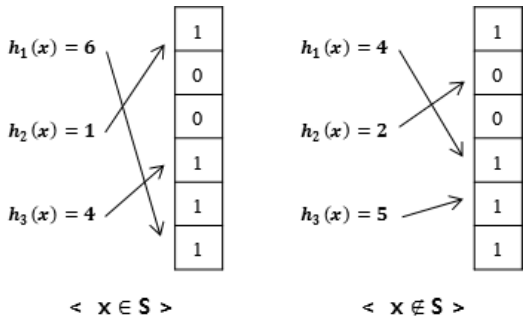


Fig.1. Membership check through the bloom filters (k = 3)

접수일(2014년 6월 13일), 수정일(2014년 8월 7일), 게재확정일(2014년 8월 7일)
* 이 논문은 2014년도 정부(미래창조과학부)의 재원으로 한국연구재단-차세대정보·컴퓨팅기술개발사업의 지원을 받아 수행된 연구임(No. 2010-0020726)
† 주저자, kjs890114@naver.com
‡ 교신저자, donghlee@korea.ac.kr(Corresponding author)

소 x가 집합 S에 속하는지 판단하기 위해 생성하는 해시 값 $h_1(x), h_2(x), \ldots, h_{k-1}(x), h_k(x)$의 인덱스에 해당하는 블룸 필터의 비트가 블룸 필터 생성과정에서 두 개 이상의 원소에 의해 '1'로 바뀌었을 수도 있기 때문이다. 블룸 필터의 긍정오류율에 영향을 미치는 요소는 크게 블룸 필터의 길이 m, 해시 함수의 개수 k, 집합의 원소의 개수 n 이 있다. 같은 조건에서 블룸 필터의 길이가 길수록, 원소의 개수가 적을수록 긍정오류율은 낮아지게 된다.

2.2 검색 가능한 암호화 기법

데이터베이스 암호화는 암호화를 통해 데이터를 보호할 수 있지만 기존의 데이터베이스 시스템에서 제공하는 기능 중 하나인 색인(indexing)을 사용할 수 없기 때문에 키워드 검색을 하는 경우 검색 속도가 현저하게 저하된다[5]. 이러한 검색에 대한 이슈를 해결하기 위해 암호화된 상태에서 검색이 가능하도록 하는 다양한 검색 가능 암호 기법에 관한 연구가 진행되었는데, 크게 대칭키 방식을 기반으로 하는 대칭키 기반 검색 가능 암호 기술과 공개키 방식을 기반으로 하는 공개키 기반 검색 가능 암호 기술로 나눌 수 있다. 기존 검색 가능 암호 기법에서는 일반적으로 클라우드 환경에서 사용자가 정해진 분류에 따라 각 데이터에 키워드를 설정하고 이를 암호화 한다. 암호화된 데이터와 암호화된 키워드를 제 3의 매체에 저장하고 나중에 사용자가 검색을 필요로 할 때 자신만이 만들 수 있는 검색 요청 데이터를 만들어 서버에서 검색한다[1]. 이러한 방법을 사용하면 검색 키워드를 노출시키지 않으면서 원하는 데이터를 얻을 수 있다.

대칭키 기반의 검색 가능 암호 기법은 상대적으로 빠른 연산속도와 효율적인 공간 활용이 가능하다는 장점을 가지고 있고, 공개키 기반의 검색 가능 암호 기법은 다양한 부가기능과 더욱 완벽한 안전성을 제공할 수 있다는 장점을 가지고 있다.

대칭키 방식의 검색 가능한 암호 기법은 Song 등[2]에 의해 최초로 제안되었다. Song 등의 기법은 클라우드 환경에서 사용자가 문서에 대한 기밀성을 보장받으면서 특정 키워드에 대한 검색을 가능하게 한다. 해당 기법은 Fig.2.와 같이 각 문서를 고정된 크기의 워드로 분리하여 1차적으로 블록 암호를 통해 암호화를 하고, 키워드 검색을 가능하게하기 위해 2차적으로 의사난수순열과 의사난수함수를 이용하여 재암호화를

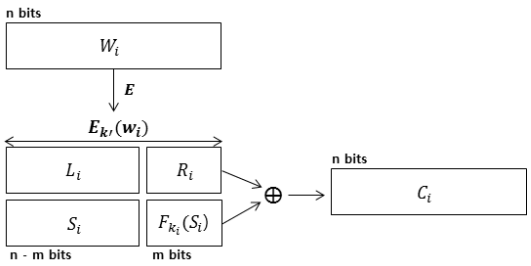


Fig. 2. Document encryption process of the Song et al.'s method

한 후 서버에 저장한다. 검색단계에서 사용자는 자신의 비밀키를 이용하여 생성한 트랩도어를 서버에 보내면, 서버는 순차적인 검색을 통해 문서를 찾아내어 사용자에게 보내준다. 해당 기법은 사용자의 문서 기밀성을 보장하면서 검색을 위한 추가적인 색인 테이블을 저장할 필요가 없다는 장점을 갖지만, 사용자측에서 연산량이 많고 통계적인 방법을 통해 평문에 대한 정보를 얻을 수 있다. 또한, 고정된 길이의 키워드 사용은 실제 환경에 적용하는데 무리가 있다.

Goh의 기법[3]에 의해 Song 등의 기법의 문제점이 어느 정도 해결되었다. 해당 기법은 블룸 필터를 통해 색인을 설정하여 색인 테이블과 암호화 된 문서가 독립성을 가짐으로써 통계적 공격에 안전하다. 또한, 해당 기법은 추가적으로 의사난수함수를 이용하여 IND-CKA(Indistinguishability against chosen keyword attack)와 IND-CKA2를 만족하는 Z-IDX를 제안함으로써 최초로 안전성이 증명된 대칭키 방식의 검색 가능 암호 기법으로 평가 받고 있다. 해당 기법은 해시함수를 통해 색인을 하여 블룸 필터를 생성하는 것이 아니라 의사난수 함수를 통해 비밀키를 생성하고, 생성한 비밀키와 의사난수함수를 이용하여 나온 결과 값을 가지고 블룸 필터에 색인을 한다. 문서 검색 단계에서는 사용자가 검색하고자 하는 키워드와 비밀키를 이용해 색인생성과정과 동일한 방법으로 블룸 필터를 생성해 서버에 전송하면, 서버는 저장 되어있는 색인 테이블과 비교하여 검색된 문서를 사용자에게 보내준다. 해당 기법은 블룸 필터를 사용하여 검색 속도가 빠르지만, 각 문서에 대한 블룸 필터를 저장하기 위해 추가적인 공간이 필요하며 긍정오류가 발생하는 문제점이 있다.

Bocsh 등의 기법[4]은 Goh 의 기법과 유사하게 블룸 필터와 의사난수순열을 사용한다. Goh 의 기법은 서버에 저장되어 있는 블룸 필터로부터 문서가 포

Single wildcard scheme
: { flower, *flower, flower*, *lower, ... , flowe*, *ower, ... , flow*, *wer, f*er, fl*r, flo*, *er, f*r, fl*, *r, f* }

Dual wildcard scheme
: { flower, *flower*, *flowe*, ... , *f*ower , f*ower*, ... , fl*wer*, ... , flo*er*, ..., flow*r*, ... , w*r*, *er*}

Fig.3. Keyword addition for wildcard search (Keyword = "flower")

함하고 있는 키워드가 무엇인지 알 수 없지만 같은 키워드를 가지고 있는 문서가 있다는 것을 유추할 수 있다. 이러한 문제점을 해결하기 위해 색인 생성 단계에서 의사난수 함수를 이용하여 블룸 필터에 추가적인 마스킹을 수행한다. 또한, 색인을 설정하기 전에 Fig.3.과 같은 추가적인 키워드 삽입을 통해 와일드카드 검색을 가능하도록 한다. 문서 검색 단계는 Song 등의 기법이나 Goh 의 기법과 달리 2라운드의 프로토콜 과정을 거친다. 사용자가 문서 검색을 위한 트랩도어를 서버에 보내면 서버는 키워드를 포함하고 있는 문서가 아니라 트랩도어 조건을 만족하는 마스킹 색인 테이블을 사용자에게 보낸다. 사용자는 마스킹을 없앤 후 얻은 정보를 이용하여 다시 한 번 서버에 원하는 문서만을 요청하고, 서버는 요청에 해당하는 문서를 보내준다. Bosch 등의 기법은 Goh 의 기법과 마찬가지로 블룸 필터를 사용하기 때문에 블룸 필터를 저장하기 위한 추가적인 공간이 필요하고 긍정오류가 발생할 수 있다. 하지만, 와일드카드 검색이 가능하며 2라운드의 프로토콜을 설계함으로써 사용자의 유연한 문서 선택을 가능하게 하여 불필요한 트래픽을 줄일 수 있다는 장점이 있다.

III. 제안하는 방법

본 장에서는 암호화된 데이터베이스 상에서 가변길이 블룸 필터를 이용한 효율적인 키워드검색 기법에 대해 설명한다. 제안하는 방법은 데이터를 암호화하여 저장하고 키워드를 통해 암호화된 데이터베이스 상에서 검색을 할 수 있도록 시스템을 설정하는 i)데이터 저장 단계, 블룸 필터를 이용해 검색하고자 하는 키워드를 포함하는 데이터를 찾는 ii) 데이터 검색 단계로 구성 된다.

데이터베이스를 암호화 하는 방법에는 데이터베이스 파일 자체를 암호화 하는 방법과 데이터베이스에 저장되어있는 각각의 레코드를 암호화 하는 방법이 있다. 본 장에서 설명하는 방법은 파일 단위가 아닌 각각의 레코드 단위로 암호화를 수행하는 경우에 적용될 수 있다. 키워드검색을 위한 블룸 필터의 사용은 빠른 연산 속도를 보장 받을 수 있지만 블룸 필터를 저장하기 위한 추가적인 공간이 필요하다는 문제점을 가지고 있다. 기존의 블룸 필터를 이용한 연구들[3,4,7]은 클라우드 환경에서 서버 측에 데이터와 생성한 색인들을 저장하기 때문에 사용자 측면에서 블룸 필터의 사용으로 인한 저장 공간 크기는 문제가 되지 않았다. 그러나 소량의 한정된 저장 공간을 갖는 스마트폰과 같은 환경의 데이터베이스에서는 블룸 필터 사용에 따른 추가적인 공간 사용의 문제점을 고려해야만 한다.

Table 1. Notations

Mark	Explanation
m	Length of bloom filter
F	False positive rate
n	Number of keyword
k	Number of encoding for each keyword
B	Bloom filter
X	Keyword
S	Bloom filter storage space in database
P	Number of data(message)
x_p ($1 \le p \le j$)	Keyword extracted by sea-rch keyword
l_q ($1 \le q \le i$)	Representative lengths
$B_q(X)(1 \le q \le i)$	Bloom filter of keyword X about each representative length
r_q ($1 \le q \le i$)	Message rate about each bloom filter's length

3.1 데이터 저장 단계

데이터 저장 단계에서는 데이터를 데이터베이스에 삽입(insert)할 때 키워드검색이 가능하도록 블룸 필터를 생성하고, 생성된 블룸 필터와 암호화된 데이터를 함께 저장한다. 기존의 데이터베이스 테이블 구조

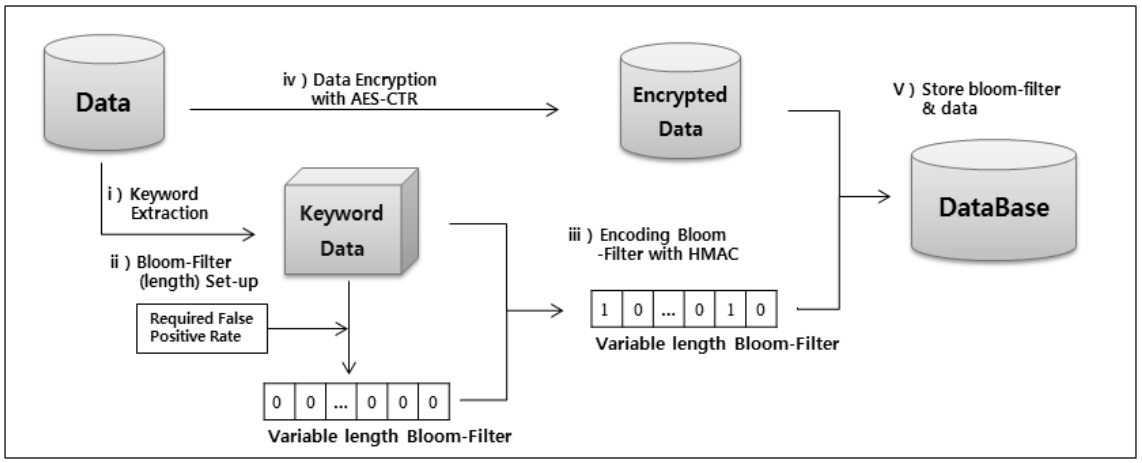


Fig.4. The overview of data storing step

에서, 키워드 검색을 지원하기 위한 열(column)은 블룸 필터를 저장하기 위한 하나의 열이 추가적으로 필요하다. 데이터 저장 단계는 Fig.4.과 같이 i) 키워드 추출 단계, ii) 블룸 필터 길이 결정 단계, iii) 블룸 필터 색인 생성 단계, iv) 데이터 암호화 단계, 그리고 v) 데이터 저장 단계로 총 5단계의 과정으로 구성된다.

• Step 1 : 키워드 추출 단계
데이터베이스에 삽입하려는 데이터로부터 검색을 위한 의미 있는 키워드를 추출한다.

• Step 2 : 블룸 필터 길이 결정 단계
추출한 키워드의 개수를 기반으로 특정 긍정오류율을 만족하는 블룸 필터의 길이를 결정한다.

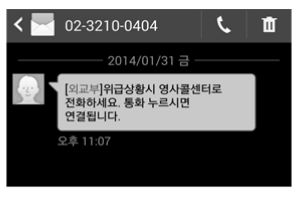
• Step 3 : 블룸 필터 색인 설정 단계
추출한 키워드에 대해 HMAC 함수를 이용하여 블룸 필터에 색인을 설정한다.

• Step 4 : 데이터 암호화 단계
AES-CTR 를 이용하여 데이터를 암호화 한다.

• Step 5 : 데이터 저장 단계
암호화 된 데이터와 Step 1~3.에서 생성한 블룸 필터와 함께 데이터베이스에 저장한다.

Fig.5.를 예로 들어 각 단계를 설명하면, Step 1은 형태소 분리기를 통해 암호화 대상 데이터를 키워드 단위로 구분 한다[9]. 여기서 키워드는 독립적인 의미를 갖도록 하기 위하여 체언단위로 구분한다. 예를 들면, Fig.5.처럼 문자메시지에서 외교부, 위급, 상황, 영사, 콜센터, 전화, 통화, 연결을 키워드로 추출한다.

Step 2는 키워드를 추출한 후 저장 공간의 효율적 사용을 고려하여 저장하고자하는 데이터의 키워드 개



Morophological Seperation

	단어	품사		단어	품사
1	[	괄호표	2	외교부	일반명사
3	]	괄호표	4	위급	일반명사
5	상황	일반명사	6	시	명사파생접미사
7	영사	일반명사	8	콜센터	고유명사
9	로	부사격조사	10	전화	일반명사
11	하	동사파생접미사	12	시	선어말어미
13	어요	연결어미	14	.	마침표
15	통화	일반명사	16	누르	동사
17	시	선어말어미	18	면	연결어미
19	연결	일반명사	20	되	동사파생접미사
21	ㅂ니다	연결어미	22	.	마침표

Fig.5. Keyword extraction through morophological separation

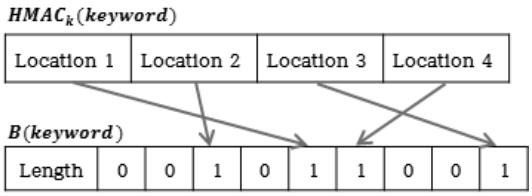


Fig.6. Encoding with HMAC (k = 4)

수에 따라 블룸 필터의 길이를 설정한다. 블룸 필터의 길이가 작을수록 효율적인 저장 공간 활용이 가능하지만 블룸 필터의 길이가 작아질수록 긍정오류율이 커지는 문제가 발생한다. 여기서 긍정오류율은 해당 데이터에 포함되어 있는 키워드가 아니지만 검색 결과에서는 해당 데이터에 속한 키워드로 판명되는 비율을 말한다. 따라서 키워드의 개수와 긍정오류율 두 가지 모두를 고려하여 블룸필터의 길이를 정해야 한다.

긍정오류율 F 는 다음과 같다[10,11].

$$F = \left(1-\left(1-\frac{1}{m}\right)^{nk}\right)^{k} \cong (1-e^{-\frac{nk}{m}})^{k} \quad (2)$$

여기서, m은 블룸 필터의 비트 길이이며, n은 키워드의 개수, k는 각 키워드에 대해 블룸 필터에 설정되는 색인의 개수이다. (1)식을 우리가 찾고자 하는 블룸 필터의 길이 m에 대해 정리하면 다음을 얻을 수 있다.

$$m = -\frac{nk}{\log(1-(F)^{\frac{1}{k}})} \quad (3)$$

키워드 검색 단계에서 문서 검색 및 구현의 효율성을 높이기 위해 (2)식에서 구해진 블룸 필터 길이 m을 그대로 사용하지 않고 필터 길이의 일정 범위마다 범위 안에서 대푯값을 설정하여 블룸 필터 길이로 사용한다. 대푯값을 사용하지 않으면 수신한 데이터마다 모두 다른 길이의 블룸 필터를 사용하게 되므로 검색 단계에서, 저장 되어 있는 모든 블룸 필터의 길이와 동일한 블룸 필터를 모두 생성해야 한다는 문제점이 발생한다.

Step 3은 추출한 키워드마다 HMAC을 통해 블룸 필터에 색인을 설정한다. 기존의 블룸 필터의 경우 k번의 해시를 통해 색인을 했지만, 본 논문에서는 한 번의 HMAC을 통해 k개의 색인을 설정한다. m 비트 길이의 블룸 필터의 경우 한 번의 색인을 위해서는 실제적으로 $\log_2 m$ 비트가 필요하고, 따라서 k번의 색인을 하기 위해서 $k\log_2 m$ 비트가 필요하다. 스마트폰 환경에서 문자메시지나 어플리케이션의 데이터베이스의 경우 추출 되는 키워드의 개수가 많지 않기 때문에 Step 2에서 생성한 블룸 필터의 길이 또한 길지 않으므로 Fig.6.처럼 한 번의 HMAC을 통해 색인 설정이 가능하다. 예를 들어 k=4, F=0.1, 128비트를 출력하는 HMAC을 사용하는 경우 (3)식을 이용하여 계산하면 긍정오류율을 0.1 이하를 유지하면서 최대 약 3,460,000개의 키워드의 색인을 포함하는 블룸 필터를 생성할 수 있다. 실제적으로 스마트폰 상에서 추출되는 키워드의 개수는 훨씬 작기 때문에 한 번의 HMAC의 결과만으로 색인 설정이 가능하다.

마지막으로, 데이터를 AES-CTR로 암호화 하고 데이터 및 블룸 필터를 함께 데이터베이스에 저장한다.

3.2 데이터 검색 단계

데이터 검색 단계는 데이터베이스에서 검색 키워드를 포함하는 데이터를 찾기 위해 블룸 필터를 생성하여 블룸 필터를 저장하고 있는 열과 순차적인 비교를 통해 데이터를 검색한다. 데이터 검색 단계는 다음과 같이 i) 검색어 추출 단계, ii) 블룸 필터 생성 단계, iii) 블룸 필터 검색 단계로 총 3단계의 과정으로 구성된다.

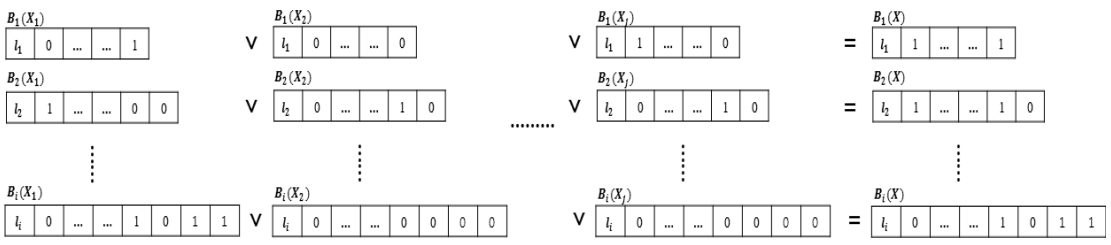


Fig.7. Generate bloom filter with primarily seperated keyword

• Step 1 : 검색어 추출 단계

사용자가 입력한 검색 키워드 X를 형태소 분리기를 통해 $x_1, x_2, \cdots, x_j$ 로 분리한다[9].

• Step 2 : 블룸 필터 생성 단계

모든 대푯값 길이에 대해 검색을 위한 블룸 필터를 생성한다.

• Step 3 : 블룸 필터 검색 단계

검색을 위한 블룸 필터를 저장하고 있는 열의 블룸 필터들과 순차적인 비교를 통해 키워드를 포함한 데이터를 검색한다.

Step 1은 데이터 저장 단계에서 설정된 키워드뿐만 아니라 설정된 키워드를 포함하는 모든 검색키워드에 대해서도 검색을 가능하게하기 위해 형태소 분리과정을 거친다. 예를 들면, 데이터 저장 단계에서는 체언만을 키워드로 구분하므로 '외교부를', '외교부에서' 등의 키워드에 대해서는 검색이 불가능하다. 하지만, 검색키워드에 대해 형태소 분리를 통해서 일차적으로 '외교부' 라는 키워드를 포함한 모든 데이터를 검색하고, 검색 결과를 복호화 하여 처음 검색한 키워드를 포함한 데이터를 찾아낸다.

Step 2는 Fig.7.과 같이 각 대푯값 길이 l_q ($1 \le q \le i$) 에 대해 키워드 x_p ($1 \le p \le j$) 의 블룸 필터를 생성하고 키워드 x_p ($1 \le p \le j$) 들을 한 번의 비교로 모두 검색하기 위해 같은 길이를 갖는 블룸 필터들을 OR 연산하여 키워드 x_p ($1 \le p \le j$) 의 색인 비트들을 모두 포함하는 키워드 X에 대한 i 개의 블룸 필터를 생성한다. Step 3에서는 순차적으로 데이터베이스에 저장되어 있는 블룸 필터들에 대해 Step 2에서 생성한 i개의 블룸 필터 $B_q(X)$ $(1 \le q \le i)$ 중 길이가 같은 $B_s(X)$를 선택하고 데이터베이스에 저장되어 있는 블룸 필터에 키워드 X가 포함되었는지를 확인하기 위해 $B_s(X)$의 색인 비트와 동일한 위치에 색인이 모두 되어있는지 확인한다. 색인의 동일 위치 확인 은 하나하나씩 비교하는 것이 아니라 AND 연산의 성질을 이용하여 비교한다. AND 연산은 피연산자 모두가 1인 경우에만 1을 결과값으로 갖는다. 즉 $B_s(X)$의 색인 비트에 대해 비교 대상은 적어도 $B_s(X)$의 색인 비트를 포함하고 있는 경우에만 AND 연산의 결과값으로 $B_s(X)$를 반환하게 된다. 그러므로 색인의 여부를 확인하기 위해 AND 연산을 하여 결과 값이 $B_s(X)$ 가 되는 것을 체크하고, AND 연산의 결과가 $B_s(X)$ 가 되는 경우 해당 블룸 필터의 문서를 복호화 하여 키워드 X가 있는지 한 번 더 확인하는 과정을 거친다.

Table 2. Computational performance of different search schemes, where n is the number of message(documents) in the database, q is the number of words per message, |△| is the number of keywords per message, |D(w)| is the number of message(documents) containing the keyword w.

Scheme	Step of data store	Step of data search
Song[2]	O(nq)	O(nq) + O(|D(w)|q)
Goh[3]	O(n|△|)	O(n)
Bosch[4]	O(n|△|)	O(n)
Ours	O(n|△|)	O(n)

IV. 비교 · 분석

본 장에서는 제안하는 방법과 기존 검색 가능 암호 기법을 계산복잡도, 키워드에 따른 블룸필터 필요 길이, 저장 공간 사용량 측면에서 비교하여 설명한다.

4.1 계산복잡도(computational complexity)

Table 2.는 본 논문에서 제안하는 기법과 기존의 검색 가능한 암호화 기법[2,3,4]을 계산복잡도 관점에서 비교하여 나타낸 것이다. 비교는 데이터 저장 단계와 데이터 검색단계로 나누어 수행하였다.

본 논문에서 제안하는 방법은 데이터 저장단계에서 문자메시지 마다 추출된 키워드의 개수만큼 블룸 필터에 색인을 하기 때문에 O(n·|△|)의 계산복잡도를 가진다. 유사하게 색인을 위해 블룸 필터를 사용하는 Goh의 기법과 Bosch 등의 기법도 같은 결과를 갖는다. 반면, Song의 기법은 고정된 길이로 데이터를 나누어 모두 저장하기 때문에 O(nq)의 계산복잡도를 가진다. 제안하는 방법은 Goh와 Bosch의 기법과는 동일하지만, |△| ≤ q 이므로 Song의 기법에 비해 더 적은 계산 과정을 거친다.

데이터 검색 단계는 크게 대푯값에 따르는 블룸 필터 생성, 데이터베이스에 저장되어있는 블룸 필터와 비교, 긍정오류를 제거하기 위한 추가적인 체크로 나뉜다. 블룸 필터 생성 과정에서는 대푯값의 개수가 l

개이고 처음에 한 번만 생성하므로 $O(l)$ 만큼의 계산 복잡도를 가지고, 블룸 필터 비교 과정에서는 검색 키워드에 해당하는 블룸 필터를 생성하여 데이터베이스에 저장 되어 있는 모든 문자메시지의 블룸 필터와 AND 연산을 하여 비교하므로 Goh와 Bosch등의 기법과 마찬가지로 $O(n)$ 만큼의 계산복잡도를 가진다. 마지막으로, 검색 결과에서 발생한 긍정오류를 제거하기 위해 비교 결과에 따라 한 번 더 체크하는 과정에서는 $O(1)$ 만큼의 계산복잡도를 가지므로 제안하는 방법의 데이터 검색 단계는 $O(l+n+1)$ 의 계산복잡도를 가진다. 하지만, n값에 비해 l은 고정 된 작은 상수 값이므로 데이터 검색 단계는 결국 $O(n)$ 만큼의 계산복잡도를 가진다.

4.2 블룸 필터의 길이 및 대푯값 설정

제안하는 방법은 데이터 저장 단계에서 수식 (3)에 따라 키워드의 개수에 따른 블룸 필터의 길이를 설정하였다. 키워드의 개수에 따라 각각의 다른 길이의 블룸 필터를 사용할 경우 데이터 검색 단계에서 키워드에 대한 블룸 필터를 생성할 때 모든 가능한 길이에 대한 블룸 필터를 생성해야 하는 문제가 발생한다. 따라서 이를 보완하기 위해 블룸 필터 길이의 일정 범위마다 특정한 블룸 필터의 길이를 대푯값으로 설정하였다.

Table 3.은 k = 4, 긍정오류율 = 0.1 일 때 키워드 개수에 따라 계산된 블룸 필터 길이와 해당 블룸 필터에 색인을 하기 위해 필요한 비트 수를 나타난 표이다. 키워드의 개수가 5와 6 일 때를 비교해보면 키워드의 개수가 다르기 때문에 그에 따라 블룸 필터의 길이도 다르지만 색인을 하기 위해 필요한 비트수 $\log_2 m$을 정수 단위로 올림 해야 하므로 색인을 위해 필요한 비트수는 같게 되는 결과를 가져온다. 이때, 5비트로 나타낼 수 있는 인덱스의 범위는 0~31 사이가 되는데 키워드의 개수가 5인 경우 블룸 필터의 길이가 25 비트이므로 인덱스가 블룸 필터의 범위를 벗어난 위치를 가리키는 문제가 발생할 수 있다. 이를 위해 인덱스 범위를 조정하는 추가 연산을 수행할 수도 있지만, 연산을 추가하지 않고 설정한 대푯값에 따라 블룸 필터 길이를 지정함으로써 문제를 해결하였다.

즉, 키워드의 개수가 6개 이하인 경우는 32 비트, 7~13개 이하인 경우는 64 비트, 14~26개 이하인 경우는 128 비트, 27~52개 이하인 경우는 256 비트, 53~105개 이하인 경우는 512 비트, 106~211개 이하인 경우는 1024 비트, 그 이상은 2048 비트로 블룸 필터의 길이를 2의 지수승의 되도록 구간별 대푯값을 설정하였다.

Table 3. Length of variable length bloom filter and representative value of bloom filter by number of keyword
(k = 4 , false positive rate = 0.1)

Number of keyword	Length of variable length bloom Filter (bits)	Number of bits which is needed to index	Representative value of bloom Filter (bits)
5	25	5	32
6	30	5	32
7	34	6	64
13	63	6	64
20	97	7	128
26	126	7	128
50	243	8	256
52	252	8	256
100	485	9	512
105	509	9	512
150	727	10	1024
200	969	10	1024
211	1022	10	1024
423	2048	11	2048

4.3 저장 공간 사용량

본 절에서는 실험을 위해 수집한 문자메시지 데이터에 대해 Goh, Bosch 등의 기법에서 사용한 블룸 필터와 본 논문에서 제안하는 가변 길이 블룸 필터의 저장 공간 사용량에 대해 비교해본다.

4.3.1 실험 데이터 수집

일반적인 블룸 필터와 가변 길이 블룸 필터 간의 저장 공간 사용량 비교와 5장의 구현 및 성능 평가를 위한 실험 데이터로 사용하기 위해 스마트폰의 문자메시지 표본을 수집하였다. 실험 데이터 수집을 위해 10~50대 연령별 남녀 각 3명씩, 총 30명을 대상으로 문자메시지를 수집하였다. 개인당 스마트폰에 저장된 약 500개의 문자메시지를 수집하여 총 14794개의 문자 메시지를 수집하였으며, 이를 제안하는 방법의 구

Table 4. Comparison of variable length bloom filter and standard bloom filter (k = 4 , false positive rate = 0.1)

Range of the number of keyword	# message (ratio of total message)	Length of variable length bloom filter (bits)	Comparison rate of filter length	Saving space (bytes)	# total message
					14794
					# min keyword
0 ~ 6	9978(67.4%)	32	0.015	2,512,456	0
7 ~ 13	4070(27.5%)	64	0.031	1,009,360	# max keyword
14 ~ 26	472(3.1%)	128	0.062	113,280	
27 ~ 52	162(1.0%)	256	0.125	15,552	145
53 ~ 105	102(0.6%)	512	0.250	6,528	# ave keyword
106 ~ 211	10(0.06%)	1024	0.500	1,280	
212 ~	0(0.00%)	2048	1	0	6.0778

현 및 결과 비교에 활용하였다.

4.3.2 블룸 필터별 저장 공간 사용량 비교

블룸 필터를 사용하는 검색 가능한 암호화 기법은 블룸 필터를 저장하기 위해 추가적인 저장 공간을 필요로 한다. Goh 와 Bosch 등의 기법에서 사용하는 고정된 길이 블룸필터는 (블룸 필터 길이) x (문자메시지 개수) 만큼의 추가적인 저장 공간 소요가 발생한다. 이러한 저장 공간 사용량을 본 논문에서 제안하는 가변 길이 블룸 필터와 비교하였다.

2048 비트의 고정 길이 블룸 필터의 경우 키워드의 개수가 0~6개인 9978(67.4%)개의 문자메시지에 대해서는 2,554,368 바이트만큼, 키워드의 개수가 7~13개인 4070(27.5%)개의 문자메시지에 대해서 1,041,920 바이트만큼의 저장 공간을 사용하게 된다. 그러나 키워드의 개수가 0~6 개인 경우 32 비트만으로 긍정오류율 0.1(10%) 수준을 달성할 수 있고, 키워드의 개수가 7~13개인 경우에는 64 비트만으로 긍정오류율 0.1 수준을 달성할 수 있기 때문에 고정 길이의 블룸 필터를 사용하는 경우에는 불필요한 공간을 많이 사용하고 있음을 알 수 있다.

가변 길이 블룸 필터를 사용하는 경우, 키워드의 개수가 0~6개인 9978(67.4%)개의 문자메시지에 대해 39,912 바이트만큼, 키워드의 개수가 7~13개인 4070(27.5%)개의 문자메시지에 대해 32,560 바이트만큼의 저장 공간을 사용하게 된다. 따라서 고정 길이 2048 비트를 사용하는 블룸 필터에 비해 키워드 개수가 0~6개인 9978개(67.4%)의 문자메시지에 대해 2,512,456 바이트만큼의 저장 공간을 절약할 수 있으며, 마찬가지로 키워드의 개수가 7~13개인 4070(27.5%)개의 문자메시지에 대해 1,009,360 바이트만큼의 저장 공간을 절약할 수 있다. 스마트폰의 문자메시지 특성상 단문 구성이 많기 때문에 가변길이의 블룸 필터는 저장 공간을 효율적으로 사용할 수 있게 한다.

Fig.8.는 Table 4.에서 나눈 키워드 개수 구간에 대해 문자 메시지의 분포가 실제 수집한 실험 데이터의 분포를 갖는 경우와 구간별 메시지 분포가 모두 동일하다고 가정한 경우에 대해 블룸 필터별 저장 공간 사용량의 크기를 비교한 것이다.

문자메시지 개수에 따르는 블룸 필터의 저장 공간의 크기 S 를 구하는 식은 다음과 같다.

$$S = \sum_{i=1}^{q} l_i \cdot r_i \cdot P \tag{4}$$

여기서, l_i는 가능한 블룸 필터의 길이이며, r_i는 l_i의 블룸 필터 길이를 갖는 문자메시지 비율, P는 전체 문자메시지의 개수이다. 고정 길이 블룸 필터는 q = 1 이고 l_1 = 256, r_1 = 1 이므로

$$\begin{aligned} S_1 &= \sum_{i=1}^{1} l_i \cdot r_i \cdot P \\ &= 256 \cdot 1 \cdot P \\ &= 256P \end{aligned} \tag{5}$$

의 결과를 얻을 수 있다. 이는 문자 메시지의 분포가 실제 수집한 실험 데이터의 분포를 갖는 경우와 구간별 메시지 분포가 모두 동일하다고 가정한 경우 모두 동일하게 나타난다. 반면 제안하는 기법은 수집한 문자메시지의 분포를 갖는 경우에 Table 4.의 결과를 참조하여 계산하면

$$S_2 = \sum_{i=1}^{7} l_i \cdot r_i \cdot P$$
$$= 4 \cdot 0.674 \cdot P + 16 \cdot 0.275 \cdot P + \ldots$$
$$+ 128 \cdot 0.0006 \cdot P + 256 \cdot 0 \cdot P$$
$$= 6.17P \quad (6)$$

의 결과를 얻을 수 있다. 블룸 필터 길이별 문자메시지의 분포가 동일한 경우에는 S_2 계산 과정에서 l_i 는 동일하며 r_i = 1/q 로 계산하면 된다. 따라서

$$S_3 = \sum_{i=1}^{7} l_i \cdot r_i \cdot P$$
$$= 4 \cdot (1/7) \cdot P + 16 \cdot (1/7) \cdot P + \ldots$$
$$+ 128 \cdot (1/7) \cdot P + 256 \cdot (1/7) \cdot P$$
$$= 72P \quad (7)$$

의 결과를 얻을 수 있다.

즉, 제안하는 기법을 사용하면 수집한 문자메시지의 분포를 따르는 경우 일반적인 블룸 필터에 비해 전체적으로 97.6% 저장 공간을 절약할 수 있고, 블룸 필터 길이별 문자메시지의 분포가 동일한 경우에도 약 72%의 저장 공간을 절약할 수 있다. 두 결과 모두 저장 공간의 효율성 측면에서 Goh 와 Bosch 등 고정된 길이 블룸 필터를 사용하는 어떠한 방법보다 본 논문에서 제안하는 방법이 저장 공간 사용 효율면에서 매우 뛰어남을 알 수 있다.

4.4 안전성

제안하는 방법은 데이터베이스에 저장되는 데이터를 AES-CTR을 통해 암호화하여 저장하고, 데이터로부터 생성되는 블룸 필터 또한 HMAC을 통하여 색인을 하기 때문에 데이터베이스에 저장되어 있는 블룸 필터에 어떤 키워드가 저장되어있는지 알기위해서는 HMAC에 사용되는 비밀키를 알아야하므로 저장된 블룸 필터로부터 키워드를 유추할 수 없다. 또한, 기존의 Goh의 기법에서 사용하는 의사난수함수와 동일한 keyed-hash 함수를 사용하기 때문에 키 값을 알지 못하면 색인을 위한 트랩도어 생성이 불가능하므로 기존의 기법과 동일한 안전성 모델을 갖는다.

V. 구현 및 성능 평가

본 장에서는 제안하는 기법을 안드로이드 스마트폰에 사용되는 SQLite 데이터베이스에 직접 구현하여, DB 암호화를 적용하고 데이터 삽입 및 검색에 대한

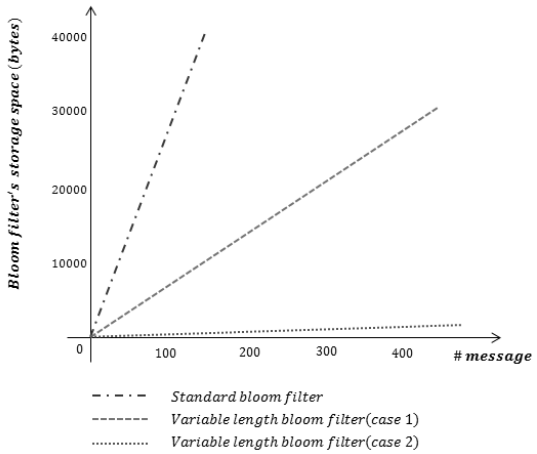


Fig.8. Comparison of standard bloom filter's storage space and variable length bloom filter's storage space. (case 1) is a scenario that message distribution per length of bloom filer is uniform. (case 2) is a scenario that message distribution per length of bloom filer is a distribution of collected message.

성능을 측정하였다. DB 암호화 적용을 위해 기존 평문 데이터가 저장되어 있던 컬럼은 대칭키 암호 알고리즘(Ex. AES)을 이용하여 암호화 시킨 후 암호화된 데이터를 저장하였으며, 본 논문에서 제안하는 키워드 검색 기법을 적용하기 위해 컬럼을 추가로 생성하여 효율적인 키워드 검색을 위한 가변길이 블룸필터 값을 저장하였다.

5.1 실험 환경

제안 기법의 구현 및 성능 측정을 위해 리눅스 환경(우분투 11.10)에서 안드로이드 소스코드(Android 4.0.4)와 한글 형태소 분리기(KOMORAN ver 1.0)을 다운로드받아 해당 코드를 기반으로 구현하였다[9, 12].

안드로이드 소스코드의 Framework 레벨에서 안드로이드 데이터베이스인 SQLite와 관련된 패키지의 코드를 수정하여, DB 암호화 및 키워드 검색이 가능하도록 적용하였다. 수정된 안드로이드 소스코드를 빌드하여 안드로이드 OS 이미지를 생성한 후, 생성한 이미지를 실제 디바이스에 올려 실험을 진행하였다. 실험에는 Android OS 4.0.3 버전이 적용된 Samsung Nexus S를 사용하였다.

5.2 실험 및 성능 분석

실험은 4.3.2에서 수집한 14794개의 문자메시지를 실험 표본으로 하였으며, 문자메시지가 암호화되어 저장된 데이터베이스를 대상으로 본 논문에서 제안하는 키워드 검색 기법의 적용 유무에 따른 데이터 저장 단계와 데이터 검색 단계의 성능을 측정하여 비교하였다.

Table 5.는 데이터베이스에 데이터를 삽입하는데 소요되는 시간과 특정 키워드에 대해 검색(select)하는데 소요되는 시간의 평균 시간을 측정하여 나타낸 것이다. 제안하는 기법은 데이터 저장 단계에서 검색을 위한 추가적인 블룸 필터를 생성하므로, 단순히 DB에 데이터를 암호화하여 저장하는 방식에 비해 약 60ms 정도의 시간이 추가적으로 소요되었다. 그러나 데이터 검색 단계에서 특정한 3가지 검색 키워드에 대해 키워드 검색을 반복 수행해 본 결과, 암호화된 상태에서의 단순한 검색 방식에 비해 제안하는 기법은 절반 정도의 시간만이 소요되었다.

각각의 검색 결과를 살펴보면, 기존 암호화된 상태에서의 단순 검색 방식의 경우 키워드에 따른 검색 결과의 개수가 증가하더라도 검색에 소요되는 시간이 일정했다. 이와 달리, 제안하는 기법은 키워드의 따른 검색 결과의 개수가 증가함에 따라 소요되는 시간도 증가함을 확인할 수 있었다. 이는 일차적으로 가변 길이 블룸 필터를 통해 검색한 결과에 대해 긍정오류로 인한 오검색을 제거하고자 검색된 문자메시지를 복호화하여 한 번 더 확인하는 과정에서 추가적인 시간이 소요되기 때문이다. 그러나 이러한 추가적인 작업 시간을 포함하더라도 기존 암호화 검색 방식에 비해 빠른 검색 속도를 나타낸다.

Table 5. Speed comparison of ours and android SQLite which cryptographic API is added to. (k = 4, false positive rate = 0.1)

	Ours (ms)	SQLite (ms)	# Search result
Step of data store	159.182	98.716	-
Step of data search 〔X = 고려대〕	5746.7	12596.4	51
Step of data search 〔X = 예정〕	6421.7	12876.5	145
Step of data search 〔X = 카드〕	6920.5	12647.1	375

VI. 논 의

기존의 검색 가능한 암호 기법에서는 데이터를 저장할 때 설정한 키워드집합에 대해서만 검색이 가능하다. 이로 인해 사용자가 검색 시 키워드와 유사한 검색어에 대해서는 검색이 불가능하다. 하지만 제안하는 기법은 데이터 저장단계 뿐만 아니라 데이터 검색단계에서도 형태소 분리기를 사용함으로써 설정하지 않은 키워드에 대해서도 검색을 가능하게 한다.

클라우드 환경에서 암호화된 데이터를 검색하기 위해 블룸 필터를 사용하면 블룸 필터의 긍정오류가 발생하여 서버에서 사용자에게 100% 정확한 검색 결과를 보내지 못한다. 이는 서버가 데이터를 검색할 때, 사용자의 비밀 값을 모르는 상태에서 사용자에게 받은 트랩도어만을 가지고 검색을 하기 때문이다. 이로 인해 사용자는 서버에서 받은 검색 결과에 대해 추가적인 작업을 해야 한다. 하지만 사용자와 서버의 역할이 동일한 스마트폰과 같은 환경에서는 데이터 검색 단계 안에 복호화 과정을 넣음으로써 추가적인 작업을 생략할 수 있다.

본 논문에서 제안하는 기법을 구현할 때, 생성된 암호문과 검색을 위한 가변길이 블룸필터 값이 저장된 데이터베이스는 스토리지에 그대로 저장하여도 무방하다. 암호문을 복호화하여 평문을 얻거나 가변길이 블룸필터 값을 이용하여 검색을 하기 위해서는 비밀값인 키가 필요하기 때문에, 이러한 키를 안전하게 보관하는 경우에는 암호문 및 블룸필터 값으로부터 평문을 획득하거나 포함된 키워드를 유추하는 것이 매우 어렵다. 비밀값인 키는 유심(usim)과 같은 안전한 저장소(secure storage)에 안전하게 보관하거나 사용자의 패스워드로부터 키유도함수를 사용하여 구성할 수 있다. 모든 테이블에 동일한 키를 사용할 경우, 키가 노출되면 데이터베이스에 저장된 모든 데이터가 유출될 수 있다. 따라서 각 테이블마다 모두 다른 키를 사용해야한다. 테이블 키만을 사용할 경우 하나의 테이블의 동일한 평문 데이터는 동일한 암호문을 생성하기 때문에 이를 피하기 위한 적절한 조치를 해야한다. 예를 들어 블록 암호의 카운터 모드(counter mode) 암호화 방법을 이용하여 매 컬럼 데이터마다 다른 초기화 벡터(initial vector) 값을 사용하여 암호화하도록 구성할 수 있다.

VII. 결 론

데이터베이스 암호화는 데이터베이스에 저장되는 사용자의 민감한 데이터에 대한 기밀성을 보장할 수 있다. 그러나 데이터베이스 암호화를 적용할 경우에는 인덱스 정보가 없어 모든 데이터를 복호화하여 순차적으로 검색해야만 한다. 이는 검색 성능의 저하를 야기하기 때문에 암호화된 상태에서 검색 가능한 기술은 데이터베이스 암호화의 중요한 이슈이다.

본 논문에서는 스마트폰의 제한적인 연산속도와 한정된 공간이라는 제한된 환경을 고려하여, 가변 길이 블룸 필터를 이용한 효율적인 키워드 검색 기법을 제안하였다. 기존의 블룸 필터를 사용한 검색 가능한 암호 기법과 달리 가변 길이 블룸 필터를 사용함으로써 저장 공간을 약 70~90% 이상 절약하였으며, 기존 연구의 미리 정의된 키워드 집합을 사용한 검색 기법과 달리 키워드집합을 미리 설정하지 않고 형태소 분리기를 이용하여 자체적으로 키워드를 설정함으로써 유연한 키워드 설정과 기존에 존재하지 않는 키워드에 대해서도 검색이 가능하게 된다. 또한 이로 인해 미리 정의된 키워드 집합을 위한 공간 사용도 절약할 수 있다.

제안하는 방법을 실제 안드로이드 스마트폰 상에서 구현한 후 수집한 문자메시지를 이용한 실험을 진행함으로써, 기존 암호화된 상태에서의 검색에 비해 데이터 삽입시 발생하는 약간의 오버헤드만으로 매우 빠른 검색 속도를 나타냄을 확인할 수 있었다.

제안하는 방법은 저장 공간 사용량 측면에서 기존 기법에 비해 큰 효율성을 보였지만, 계산복잡도 측면에서는 Goh 등의 기법과 Bosch 등의 기법과 동일한 수준을 나타내었다. 향후에는 저장 공간 및 연산량 측면에서 모두 효율성을 제공할 수 있는 기법에 대한 연구를 진행할 예정이다.

References

〔1〕 Sun-Young Kim, Jae-Woo Seo and Pil-Joong Lee, "The Study on Trends of Searchable Encryption," Journal of The Korea Institute of Information Security & Cryptology(JKIISC), 19(2), pp.73-90, April. 2010.

〔2〕 D. Song, D. Wagner, and A. Perrig, "Practical techniques for searching on encrypted data," IEEE Symposium on Security and Privacy, pp.44 - 55, May. 2000.

〔3〕 Eu-Jin Goh, "Secure Indexes," IACR ePrint 2003-216, May. 2004.

〔4〕 C. B¨osch, R. Brinkman, P. Hartel, and W. Jonker, "Conjunctive wildcard search over encrypted data," the 8th VLDB Workshop on Secure Data Management, pp.114 - 127, February. 2011.

〔5〕 R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu, "Order preserving encryption for numeric data," the 2004 ACM SIGMOD international conference on Management of data, pp.563 - 574, June. 2004.

〔6〕 D. Boneh, G. di Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," EUROCRYPT 04, LNCS 3027, pp. 506-522, May.2004.

〔7〕 S. Bellovin and W. Cheswick, "Privacy-enhanced searches using encrypted Bloom filters," IACR ePrint 2004-022, September. 2004.

〔8〕 B.H. Bloom, "Space/Time Trade-Offs in Hash Coding with Allowable Errors," Communication of the ACM, vol.13, no.7, pp.422-426, July. 1970.

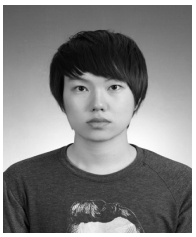
〔9〕 KOMORAN v 1.0, 〔Online〕. Available, http://shineware.tistory.com/m/post/view/id/37

〔10〕 A. Broder and M. Mitzenmacher, "Network Applications of Bloom Filters: A Survey," Internet Mathematics, vol.1, no.4, pp.485-509, 2004.

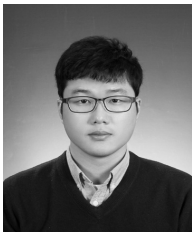
〔11〕 Yan Qiao, Tao Li and Shigang Chen, "Fast Bloom Filters and Their Generalization," IEEE Transactions on Parallel and Distributed System, vol.25, no.1, pp.93-102, January. 2014.

〔12〕 Android Open Source Project, 〔Online〕. Available, http://source.and roid.com/source/index.html

〈저자소개〉



김 종 석 (Jong-Seok Kim) 학생회원
2013년 2월: 서울시립대학교 수학과 학사
2013년 3월~현재: 고려대학교 정보보호대학원 석사과정
〈관심분야〉 암호프로토콜, 스마트폰 보안, DB보안



최 원 석 (Won-suk Choi) 학생회원
2008년 2월: 서울시립대학교 수학과 학사
2013년 2월: 고려대학교 정보보호대학원 석사
2013년 3월~현재: 고려대학교 정보보호대학원 박사과정
〈관심분야〉 정보보호, 의료기기 보안

박 진 형 (Jin-Hyung Park) 학생회원
2010년 2월: 건국대학교 컴퓨터공학과 학사
2012년 2월: 고려대학교 정보보호대학원 석사
2012년 3월~현재: 고려대학교 정보보호대학원 박사과정
〈관심분야〉 암호프로토콜, 스마트폰 보안, 암호 알고리즘, 데이터마이닝

이 동 훈 (Dong Hoon Lee) 종신회원
1983년 8월: 고려대학교 경제학사 졸업
1987년 12월: Oklahoma University 전산학과 석사 졸업
1992년 5월 : Oklahoma University 전산학과 박사 졸업
1993년 3월 ~ 1997년 2월 : 고려대학교 전산학과 조교수
1997년 3월 ~ 2001년 2월 : 고려대학교 전산학과 부교수
2001년 3월 ~ 현재 : 교려대학교 정보보호대학원 교수
〈관심분야〉 암호프로토콜, 암호이론, USN이론, 키 교환, 익명성 연구, PET기술