情 報 保 護 學 會 誌
第19卷 第2號, 2009. 4

검색 가능 암호 기술의 연구 동향

김 선 영*, 서 재 우**, 이 필 중**

요 약

검색 가능 암호 시스템은 암호화된 자료를 복호화하지 않은 상태에서도 원하는 자료를 검색할 수 있도록 하는 암호 기반 기술이다. 정보를 암호화하여 저장하는 기법은 데이터베이스 보안을 위한 핵심 기술 가운데 하나로 각광받고 있지만, 암호화를 통한 보안성 확보는 데이터베이스의 검색 속도를 현저히 저하시키는 문제를 야기한다. 검색 가능 암호 시스템은 이러한 문제의 해결 방안 중 하나로 주목 받고 있으며, 학계를 중심으로 많은 연구가 진행되고 있는 상황이다. 본 고에서는 이제까지 발표되었던 중요한 검색 가능 암호 기술에 대해 살펴보기로 한다.

Ⅰ. 서 론

현대 사회에서는 컴퓨터 기술의 발전으로 인해 정보를 디지털화하여 저장하거나 처리하며, 네트워크를 통해 공유하여 사용하고 있다. 또한 인터넷의 보급은 이러한 디지털 정보를 개인의 컴퓨터뿐 만이 아니라 외부 서버에 보관할 수 있는 기능을 제공하였다. 처리하는 자료의 양이 증가하고 다양한 서비스에 대한 요구가 늘어남에 따라 특화된 외부 저장 공간의 활용이 늘어 나고 있는 것이다.

기업이 고객 정보를 효율적으로 관리하기 위해 사용하는 데이터베이스 또한 개인의 정보가 외부 저장 공간에 저장되는 예로 볼 수 있다. 고객 개인 정보 등의 데이터는 기업이나, 정부, 여러 기관 등 대부분의 조직에서 매우 소중한 자산으로 업무상 중요한 의사 결정이 이러한 데이터의 무결성과 신뢰성, 정확성 등에 의해 좌우된다고 해도 과언이 아니다.

그러나 최근 정부 기관의 고객 개인정보 누출, 유명 카드사 고객정보 노출 등의 개인 정보 유출 사고들이 연이어 발생하면서[1,2], 외부 저장 공간에 저장된 데이터에 대한 보안 문제가 이슈가 되고 있다. 현재 많은 기업에서는 데이터 누출 방지를 위한 기술적 대책 마련에 고심하고 있으나, 대부분 운영 체제와 네트워크 계층에서의 침입 탐지 측면에 국한되어 있는 것이 현실이다. 공격자로부터의 침입에 대비하여 방화벽(Firewall), 침입 탐지 시스템(Intrusion Detection System)을 구축하는 등의 접근 제어 방법은 외부 침입자를 막는 데에는 유효한 방법이지만 저장 공간의 소유자가 저장되어 있는 자료를 열람하는 것을 근본적으로 방지하지는 못한다.

이와는 달리, 정보를 안전하게 보호하기 위해 데이터 자체를 암호화하는 기법은 내부자로부터 데이터를 보호할 수 있다는 측면에서 매우 매력적이다. 그러나 데이터의 암호화는 기존 데이터베이스에서 제공하는 고유 기술 가운데 하나인 인덱싱 기술을 사용할 수 없기 때문에 검색 속도가 현저히 떨어지는 문제를 야기한다. 저장 매체에 대한 효율적인 검색 서비스와 암호화를 통한 기밀성 및 privacy의 보장은 동시에 이루기 어려운 목표로 인식될 수 있지만, 암호화에 따른 검색 성능 저하 문제를 해결하고자 하는 시도는 학계를 중심으로 많은 연구가 진행되고 있다. 검색 가능 암호 시스템은 그러한 연구 결과의 하나로, 사용자의 검색 요청을 처리하는 과정에서 서비스 제공자가 저장된 자료의 내용에 대한 정보를 얻는 것을 방지하는 방법을 제공한다. 사용자는 미리 정해진 분류에 따라 각 파일에 키워드를 설정하고, 이를 검색 가능한 암호화 방법으로 암호화 한다. 그 후 암호화된 파일과 암호화된 키워드를 온라인 저장 매체에 저장한다. 나중에 사용자가 검색을 필요로 할 때 자신만이 만들 수 있는 검색 요청 데이터를 만들어 저장

* 포항공과대학교 정보통신대학원 (kimsy@postech.ac.kr)
** 포항공과대학교 전자전기공학과 (jwseo@postech.ac.kr, pjl@postech.ac.kr)

매체를 관리하는 온라인 서버에 보낸다. 온라인 서버는 검색 요청 데이터와 자신이 저장하고 있는 암호화된 키워드를 검색할 수 있으나, 키워드의 내용은 전혀 알 수가 없다. 암호화된 데이터 상에서의 키워드 검색 방법은 2000년에 처음 제안된 이후[3], 다양한 논문이 발표되었다. 본 고에서는 검색 가능 암호 시스템의 구조, 요구조건 등을 살펴본 후 지금까지 제안된 중요한 검색 가능 암호 시스템에 대해서 살펴볼 것이다.

Ⅱ. 배경 지식

검색 가능 암호 시스템에서는 암호화의 대상이 되는 정보를 자료(document)라 부른다. 즉, 자료는 사용자가 숨기고 싶은 정보이다. 또한, 사용자가 자신이 원하는 자료를 검색하기 위한 목적으로 쓰이는 정보를 키워드(keyword)라고 부른다. 일반적으로 자료는 그 자료에 포함된 키워드들의 집합으로 정의된다.

검색 가능 암호 시스템은 키 생성 과정(key generation), 암호화 과정(build index), 트랩도어 생성 과정(trapdoor generation), 검색을 위한 테스트 과정(test for search)의 4가지 단계로 이루어진다. 키 생성 단계는 사용자가 앞으로 사용할 검색 가능 암호 시스템을 준비하는 단계이다. 키 생성 단계에서 각 사용자는 자신의 비밀키를 생성하여 보관하고 암호 시스템의 공개 정보는 서버나 다른 사용자들에게 공개한다. 암호화 단계에서 사용자는 주어진 자료에 대해 자료 전체를 암호화한 암호문과 자료에 포함된 키워드의 정보를 포함한 인덱스(index)를 생성한다. 암호문과 인덱스는 모두 서버에 저장된다. 암호화 단계에서 사용자의 대칭키가 사용되는 검색 가능 암호 시스템을 대칭키 기반 검색 가능 암호 시스템 (symmetric-key based searchable encryption)이라 부르고 공개키 시스템을 이용하여 암호문 및 인덱스의 생성이 가능한 시스템을 공개키 또는 비대칭키 기반 검색 가능 암호 시스템 (public-key based searchable encryption)이라 부른다. 본 고에서는 대칭키 기반 시스템과 공개키 기반 시스템으로 각각 나누어 살펴볼 것이다.

트랩 도어 생성 단계는 사용자에 의해서 실행되며 주어진 키워드에 해당하는 트랩 도어를 생성한다. 트랩 도어는 오직 사용자의 비밀키로부터 생성이 가능하다. 마지막, 검색단계는 주어진 트랩 도어에 대응하는 자료를 찾는 단계로 서버에 의해서 실행된다. 검색의 결과로 서버는 주어진 트랩 도어와 일치하는 자료의 암호문 또는 자료의 식별자(identifier)를 사용자에게 전달한다.

검색 가능 암호 시스템은 다음과 같은 요구 조건을 만족시켜야 한다. 우선 검색 단계에서는 주어진 트랩 도어와 일치하는 모든 자료들이 검색되어야 하며, 검색에서 발생할 수 있는 오류는 최소화되어야 한다. 여기에서 오류란 주어진 트랩 도어에 대응하는 키워드를 포함하고 있지 않은 자료가 검색 결과에 포함되는 것을 의미한다. 또한 암호학적 안전도 측면에서 볼 때, 검색 과정에서 유출되는 정보의 양은 가능하면 작아야 한다. 즉, 주어진 트랩도어와 관계없는 또는 일부 키워드만을 포함하는 자료에 대한 정보는 유출이 되어서는 안 된다.

Ⅲ. 대칭키 기반 검색 가능 암호 기술

대칭키 기반의 검색 가능 암호 시스템에서는 사용자가 문서들을 서버에 저장한다고 가정한다. 여기서 사용자는 서버로부터도 문서의 기밀성이 유지되기를 원하므로, 문서를 암호화하여 저장한다. 이후에 원하는 문서를 검색할 때는 사용자만이 알고 있는 값을 이용하여 트랩도어를 만들어 서버에게 전송하고, 서버는 트랩도어와 저장 되어 있는 암호문들을 사용하여 검색 기능을 수행한다.

3.1 Single Keyword Search

2000년에 Song 등은 문서들이 암호화되어 있는 저장 매체에서 서버로부터도 자료의 기밀성을 보장하면서 원하는 자료들만 찾을 수 있는 실질적인 방법을 처음으로 제안하였다[3]. 논문에서 제안한 방식은 평문의 정보를 유출하지 않으면서 검색을 할 수 있도록 고안되었으며, 대칭키 방식에 기반하고 있다. 이 방식에서는 기본적인 대칭키 암호 알고리즘을 이용하여 평문을 암호화한 후에, 그것을 다시 스트림 암호를 통해 생성해 낸 난수열과 결합하여 최종 암호문을 만들어낸다.

하나의 문서는 여러 개의 워드들($W_1, W_2, ..., W_l$)로 구성되며 사용자는 각 문서들을 워드 단위로 암호화하게 되는데, 아래의 과정을 통하여 검색 가능한 암호문이 생성된다.

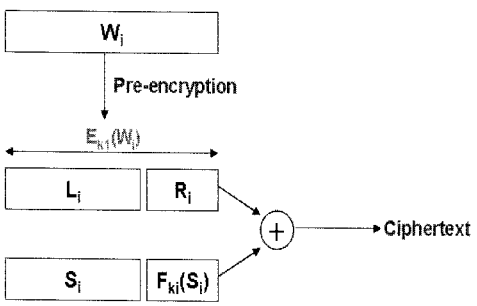
1) 대칭키 암호(pseudorandom permutation)을 사용

하여 각 워드 W_i를 암호화하고, 결과로 나온 암호문을 X_i라고 한다. 즉, $X_i = E_{k_1}(W_i)$이다. 여기서 k_1은 대칭키 암호에 사용되는 키를 의미하며, 각 X_i는 n bit이다.

2) X_i를 L_i와 R_i, 두 부분으로 나누는데 여기서 L_i는 왼쪽에서부터 $n-m$ bit, R_i는 나머지 m bit를 의미한다. 이 중에서 L_i를 이용하여 각 워드에 해당하는 비밀키 k_i를 만드는데, k_i는 pseudorandom function f를 사용하여 $k_i = f_{k_2}(L_i)$로 표현할 수 있다. 여기서 k_2는 pseudorandom function에서 사용하는 키 값이다.

3) 임의의 스트림 암호(pseudorandom generator)를 사용하여 pseudorandom value $S_1, ..., S_i$을 만든다. 여기서 각 S_i는 $n-m$ bit이다. 입력이 $n-m$ bit, 출력이 m bit인 pseudorandom function F에 이 값들을 통과시키면 $V_i = F_{k_i}(S_i)$를 얻을 수 있고, $< S_i, F_{k_i}(S_i) >$를 T_i라 한다.

4) 그러면 T_i와 X_i를 XOR 연산하여 최종 암호문을 얻을 수 있는데, 이 과정을 그림으로 표현하면 [그림 1]과 같다.



[그림 1] Single Keyword Search Scheme의 구조

이 때, 암호문을 검색하기 위해 필요한 트랩 도어는 $E_{k_1}(W_i) = (L_i \| R_i)$와 $k_i = f_{k_2}(L_i)$이다. 검색을 하기 위해서는, 서버에 저장된 암호문 C에 대해서 $C \oplus E_{k_1}(W_i)$를 계산한다. 그 결과가 $S \| F_{k_i}(S)$의 형태를 만족하면 그것이 바로 원하는 검색 결과가 되며, 사용자에게 전달한다. 이렇게 함으로써 암호문 C를 해독하지 않은 상태에서도 평문의 특정 정보를 포함하고 있는 문서들에 대한 검색을 수행할 수 있다.

3.2 Secure Index

2003년에 Goh는 Bloom filter[4,5]를 사용하는, 대칭키 기반의 검색 가능 암호 시스템을 제안하였다[6]. 또한 검색 가능 암호 시스템에 대한 안전성 정의와 모델을 제시하였으며, 그에 기반 하여 제안한 암호 시스템의 안전성을 증명하였다. 이 방식에서는 검색 가능 암호문을 고정된 크기의 데이터 구조인 Bloom filter에 저장하기 때문에, 검색 가능 암호문의 크기가 데이터의 키워드 개수에 비례하여 증가하지 않는다. 따라서 메모리 측면에서 효율적인 장점이 있다.

사용자는 Bloom filter에 사용할 상수 r과 비밀키 $k_1, k_2, ..., k_r$을 임의로 선택한다. 또한 일방향 함수 f를 선택하여 주어진 키워드 w에 대해서 $x_1 = f(k_1, w), ..., x_r = f(k_r, w)$을 계산한다. 다시 각각의 자료에 대해서 자료의 식별자 D_{ID}를 이용하여 $y_1 = f(x_1, D_{ID}), ..., y_r = f(x_r, D_{ID})$를 계산한다. 자료 D_{ID}의 키워드 w에 대해 사용자는 n bits의 Bloom filter를 정의한다. 초기의 Bloom filter는 모두 0의 값으로 채워지고, 후에 D_{ID}와 w가 결정되면 앞에서 나온 계산을 통해 얻어진 y_i번째 bit의 값을 1로 변경한다.

키워드 w에 대한 트랩 도어는 $x_1 = f(k_1, w), ..., x_r = f(k_r, w)$로 비밀키 $k_1, k_2, ..., k_r$를 모두 알고 있는 사용자만이 구성이 가능하다. 또한 주어진 트랩 도어로부터 다른 키워드에 대한 트랩 도어를 생성하는 것은 일방향 함수의 안전성으로부터 보장된다. 서버는 주어진 트랩 도어와 각 자료의 식별자인 D_{ID}를 이용하여 $y_i = f(x_i, D_{ID})$를 계산한다. 각 자료의 Bloom filter를 검사하여 y_i 번째 bit에 1의 값을 가지는 자료가 나오면 그것이 원하는 검색 결과가 되므로 사용자에게 통보한다.

3.3 Conjuctive Keyword Search

앞에서 언급했던 두 논문들은 모두 암호화된 문서들을 검색할 수 있는 방법들을 제안하였다. 하지만 이 방법들은 모두 서버가 하나의 키워드에 대해서만 검색할 수 있으며, 사용자가 여러 키워드를 포함하는 문서를 찾고 싶을 경우 여러 개의 트랩도어를 생성해야 한다. 사용자가 찾고자 하는 각 키워드에 대해서 트랩도어를 만들어 서버에게 주면, 서버는 이를 이용하여 검색하고 검색한 결과들의 교집합 연산을 하여 사용자가 원하는 문

서를 얻을 수 있다. 하지만 이 방법은 서버에게 추가적인 정보를 주게 되고, 서버는 검색 결과로부터 각 키워드가 포함되어 있는 문서들의 집합을 구분할 수 있다.

다른 방법으로는 가능한 키워드들의 모든 조합에 대해서 meta-keyword를 만드는 것이다. 예를 들어, 한 문서의 키워드가 "Bob", "urgent", "finance" 3가지라고 하면 이 키워드들을 이용하여 가능한 조합은 23(=8)가지이고, 서버는 8개의 meta-keyword를 저장하고 있어야 한다. 하지만 이 방법을 사용하면 어느 문서가 m개의 키워드를 포함하고 있을 경우 2m개의 meta-keyword가 추가로 필요하게 되고, 서버가 저장하고 있어야 하는 데이터의 양이 많아진다는 단점이 있다.

따라서 위에서 언급한 두 방법 모두 여러 키워드를 포함하는 문서를 검색하는데 적합하지 않다. 2004년에 Golle 등은 이러한 문제를 해결하기 위하여 conjunctive search 방법을 제안하였다[7]. 이 논문에서는 keyword field라는 개념을 사용하는데, 예를 들어 서버에 저장되는 문서가 메일이라면 여기서 가능한 keyword field는 "From", "To", "Date", "Subject"로 구분할 수 있다. 즉, 검색하려는 항목별로 field를 분류하고 이 field에 맞게 키워드 값을 할당한다. 논문에서는 각 문서당 m개의 Keyword field가 존재한다고 가정한다. Keyword field를 사용하기 위해 논문에서 추가적으로 2가지 가정을 하는데 첫 번째는 동일한 키워드가 서로 다른 keyword field에 나타나지 않는다는 것이며, 이 조건은 다음과 같이 키워드를 설정하면 만족시킬 수 있다. "From"이라는 field와 "To"라는 field가 있을 때, 각 키워드를 "From : Bob", "To : Bob"처럼 field의 이름을 포함하여 만든다. 두 번째 가정은 모든 keyword field가 모든 문서에서 정의된다는 것이다. 예를 들어 제목이 없는 메일의 경우, "Subject" field에 "Subject : NULL"값을 할당한다.

각 문서에 m개의 키워드가 존재하면 이를 문서마다 벡터로 표현할 수 있는데, i번째 문서에 해당하는 벡터를 D_i, D_i에서 j번째 keyword field에 해당하는 키워드를 W_j라고 하면 $D_i = (W_{i,1}, W_{i,2}, ..., W_{i,m})$으로 나타낼 수 있다. 또한 서버에 저장하려는 문서 자체는 일반적인 대칭키 암호화 알고리즘을 사용하여 암호화하고, D_i와 함께 서버에 저장한다. 문서를 서버에 암호화하기 위해서 먼저 0에서 $q-1$까지 정수 중에서 a값을 랜덤하게 뽑는다. Keyword field 벡터에서 각 키워드 값 $W_i (1 \le i \le m)$를 hash function에 통과시킨 값을 V_i라고 하면 암호 알고리즘의 출력은 $(g^a, g^{aV_1}, g^{aV_2}, ..., g^{aV_m})$이다.

트랩도어는 오프라인으로 계산할 수 있는 부분과 그렇지 않은 부분으로 나눌 수 있는데, 오프라인으로 계산할 수 있는 부분을 미리 연산을 수행하게 되면 온라인에서의 계산량을 줄일 수 있다. 트랩도어를 만드는 과정은 아래와 같다.

0에서 $q-1$까지 정수 중에서 s값을 랜덤하게 뽑고 $H(g^s)$값을 계산하는데, 이 값은 오프라인으로 수행할 수 있다. 여기서 H는 hash function을 의미한다.

$j_1, j_2, ..., j_t$는 m개의 keyword field에서 키워드의 위치를 나타내는 인덱스이고, $(W_{j1}, W_{j2}, ..., W_{jt})$는 그 인덱스에 해당하는 키워드 값이다. 이 때, $C = S + f_k(W_{j_1}) + f_k(W_{j_2}) + ... + f_k(W_{j_t})$을 계산하고 $(H(g^s), C, j_1, j_2, ..., j_t)$를 최종 트랩도어로 출력한다.

테스트를 하기 위해서는 사용자로부터 받은 트랩도어를 사용하여 g^{aC}, $g^{-a(V_{j_1}+V_{j_2}+...+V_{j_t})}$값을 계산하고 이 값을 R이라 한다. 그리고 $H(R)$과 $H(g^s)$값이 일치하는지 확인한다. 일치하면 그 문서는 사용자가 찾는 키워드들을 포함하는 문서이고 서버는 문서를 사용자에게 전달한다.

3.4 Efficient Keyword Search

2005년 Chang 등이 제안한 방식은 매우 작은 트랩도어를 사용하는 등 검색 가능 암호 시스템에 대한 현실적인 요구를 반영하여 고안되었다[8]. 기존의 검색 가능 암호 시스템들이 각각의 키워드에 대한 인덱스를 별도로 저장했던 것에 비해서 이 시스템은 각각의 키워드마다 단지 1bit의 정보만을 사용한다. 즉, 자료가 주어진 키워드를 포함하는지의 여부만을 저장하는 것이다. 또한 해시 테이블을 이용한 방식으로 저장량을 최소화할 수도 있다. 단, 이 경우 검색 과정 동안 사용자와 서버간의 통신 횟수가 증가한다. 시스템은 다음과 같은 방식으로 작동한다.

사용자는 비밀키 s와 r을 랜덤하게 선택한다. 인덱스를 구성하기 위해서 사용자는 각각의 자료에 대해서 2^d bit의 배열을 구성하는데 초기에는 모든 배열은 0의 값을 가진다. 만약 자료가 i번째 키워드 w_i를 포함한다면, 유사 난수 생성 함수 $P_s(i)$의 값을 계산하여 배열의

$P_s(i)$번째 bit를 1로 치환한다.

만약 이러한 배열로 이루어진 인덱스를 서버에 그대로 저장한다면 서버는 각각의 배열 값을 보고 주어진 자료가 어떤 키워드를 포함하는지의 정보를 얻을 수 있다. 따라서 배열을 암호화할 필요가 있는데, 이 때, $M_j[i] = I_j[i] \oplus G_{r_i}(j)$와 같이 각 bit에 대해 마스크를 만들어 적용하게 된다. 여기에서 $r_i = F_r(i)$로 F_k와 G_k는 각각 k에 의해서 결정되는 유사 난수 발생 함수 패밀리이다. 키워드 w에 대한 트랩 도어는 다음과 같다.

$$T =< p = P_s(i), f = F_r(p) >$$

사용자는 각 키워드에 대한 키워드 인덱스 i에 대한 사전을 별도로 저장하고 있어야 한다. 이러한 사전을 해시 함수 등을 사용하여 생성하는 것도 생각해 볼 수는 있지만, 해시 함수의 충돌쌍 문제에 의해서 오류가 발생할 수 있으므로 사전은 별도로 생성하여 저장하여야 한다. 이 사전을 서버에 저장하는 경우, 사용자의 저장량을 최소화시킬 수 있지만, 트랩 도어를 생성하기 위해서 서버에 키워드에 해당하는 키워드 인덱스를 질의하여야 하므로 통신 복잡도가 증가하게 된다.

주어진 트랩 도어에 대해서 서버는 다음과 같이 검색을 수행한다. 우선, 모든 j에 대해서 $I_j[p] = M_j[p] \oplus G_f(j)$를 계산하여 만약 $I_j[p] = 1$이면, 서버는 자료의 식별자를 사용자에게 전송한다.

3.5 Searchable Symmetric Encryption Providing a Constant Search Time

대다수의 검색 가능 암호 시스템에서는 서버의 문서 검색 시간은 저장하고 있는 문서의 양에 비례하여 증가한다. Curtmola 등은 2006년에 대칭키 암호를 사용한 새로운 안전성 모델을 정의하고, 이러한 모델 하에서 효율적인 검색 가능 암호 기법인 SSE(Searchable Symmetric Encryption)을 제안하였다[9]. 또한 이전까지의 시스템이 한 사용자들에 대해서만 다루었던 반면에, 본 논문은 특정 사용자들의 그룹이 서버에 검색을 요청할 수 있는 다수 사용자들에 대한 안전성 모델과 검색 가능 암호 기법을 제안하였다.

이 논문에서 제안하는 기법들은 배열, 링크드 리스트, 해쉬 테이블 등의 데이터 구조를 사용한다. 문서의 키워드를 서버에 저장하는 단계에서 이러한 데이터 구조를 사용하여 일정한 문서 검색 시간을 제공할 수 있다. 기존에 제안된 검색 가능 암호 시스템이 서버에 저장된 모든 자료의 인덱스에 대해 검색을 수행하던 것과 달리, SSE는 주어진 키워드에 대응하는 자료의 인덱스만을 검사하는 것이다. 따라서 검색 속도 면에서 기존의 시스템들과 차별성을 지닌다. 하지만, 하나의 자료가 여러 키워드와 관련성을 지닐 때, 중복 저장이 발생하여 저장 공간 면에서는 상당히 비효율적이라 할 수 있다.

키 생성 과정에서 사용자는 유사 난수 생성 함수 P와 비밀키 s, y, z를 선택한다. 주어진 키워드 w_i에 대해서 사용자의 모든 자료를 조사하여 키워드 w_i를 포함하는 자료들로 집합인 D_i를 구성한다. 이 D_i를 링크드 리스트의 형태로 서버에 저장한다. 링크드 리스트는 일종의 배열로 다음과 같은 구조를 지닌다. 사용자는 초기에 적당한 크기의 배열을 선언한다. 배열의 각 원소 또는 노드는 유일한 주소 값이 정해져 있다. 또한 각각의 노드에는 자신의 노드 다음에 연결되어 따라올 노드의 주소를 저장하기 위한 공간이 할당되어 있는데 이러한 노드의 주소를 링크(link)라고 한다. 그리고 노드들이 링크에 의해서 연결되어 있기 때문에 링크드 리스트라고 불린다. SSE에서 사용하는 노드는 다음과 같이 구성되어 있다.

$$N_{i,j} =< ID(D_{i,j}) \| k_{i,j} \| P_s(ctr+1) >$$

노드의 첫번째 값은 이 노드에 대응하는 자료의 식별자이다. 그 이후의 $k_{i,j} \| P_s(ctr+1)$은 다음에 따라 올 노드를 위한 링크인데, 이 중 $P_s(ctr+1)$은 다음 노드의 주소를 나타내고, $k_{i,j}$는 다음 노드를 암호화하기 위한 비밀키를 의미한다. 유사 난수 생성 함수인 $P_s(ctr+1)$을 이용하여 각 자료의 위치를 배열 내에서 임의로 선택하고, $k_{i,j}$를 이용하여 각 노드를 암호화하여 서버가 저장된 노드로부터 자료에 대한 정보를 얻을 수 없도록 한다.

사용자가 D_i를 링크드 리스트에 저장하고자 할 때는 우선 ctr 값을 0으로 초기화하고 링크드 리스트가 시작될 위치를 $P_y(w_i)$로 계산한다. D_i의 첫번째 자료를 $P_y(w_i)$를 주소값으로 갖는 노드에 할당하고 자료의 식별자를 저장한다. 다음 $P_s(ctr+1)$을 계산하여 다음 노드를 선택하고 다음 자료를 저장한다. 이 과정을 D_i에 속한 모든 자료를 저장할 때까지 반복하며 마지막 노드

의 링크는 NULL로 비워둔다. 마지막 단계로 D_i를 저장하는 데에 사용된 모든 노드를 각각 암호화한다. 사용자는 리스트의 첫번째 노드의 위치와 암호화 키를 $T[P_z(w_i)] =< P_y(w_i), k_{i,0} >$의 형태로 해시 테이블 T에 저장한다. 자료가 저장되어 있는 배열과 마지막에 구성한 해시 테이블은 서버에 저장된다. 위의 과정에서 만약 하나의 자료가 n개의 키워드를 포함하고 있다면 이러한 자료는 모두 n개의 서로 다른 링크드 리스트에 저장이 되어야 하므로 n번 중복이 발생한다. 이러한 이유로 실제 자료의 수에 비해 훨씬 큰 저장 공간이 필요하게 된다.

주어진 키워드 w에 대한 트랩 도어는 $(P_z(w), P_y(w))$로 주어진다. 서버는 우선 해시 테이블에서 $T[P_z(w)]$를 찾아내어 링크드 리스트의 첫 노드의 주소와 이에 해당하는 암호화 키를 찾아낸다. 서버는 링크를 이용하여 배열 내에서 링크드 리스트를 구성하는 모든 노드를 찾아내어 복호화 한다. 이 복호화를 통해서 서버는 링크드 리스트에 저장되어 있는 자료의 식별자를 얻을 수 있고, 링크가 NULL 값을 가지는 노드를 발견하면 검색을 중단하고 지금까지 발견한 모든 자료의 식별자를 사용자에게 전달한다.

Ⅳ. 공개키 기반 검색 가능 암호 기술

공개키 기반 검색 가능 암호 시스템에서는 시스템 또는 사용자의 공개키 정보만을 이용하여 누구나 암호문을 생성할 수 있다. 따라서 서버 등의 외부 저장 공간에 저장되는 자료의 제공자와 검색을 하기 위해 트랩도어를 생성하는 사용자가 동일하던 대칭키 기반 기술과는 다른 응용 환경에서 사용될 수 있다. 예를 들어 이메일 서버 등의 환경에서 적용될 수 있는데, Bob이 Alice에게 메일을 보낸다고 가정해 보자. Bob은 Alice의 공개키를 사용하여 메일을 암호화하고 Alice에게 전송한다. Alice가 자신에게 수신된 메일들 중에서 "urgent"라는 키워드를 포함하는 메일을 찾고 싶다면 Alice는 자신의 비밀키를 사용하여 메일 서버에게 보낼 트랩도어를 만들고, 서버는 이 트랩도어를 사용하여 검색 기능을 수행하여 "urgent"라는 키워드가 포함된 메일만을 검색하여 Alice에게 제공한다.

4.1 Public-key Encryption with Keyword Search

2004년에 Boneh 등은 처음으로 공개키 기반의 검색 가능 암호 시스템의 정의를 제시하였으며, 이를 만족하는 최초의 공개키 기반 검색 가능 암호 시스템을 제안하였다[10]. 이들은 두 개의 알고리즘을 소개하였는데 이 중에서 하나는 겹선형 사상이라는 수학적인 기술 요소를 사용하였고, 다른 하나는 trapdoor permutation을 사용하여 시스템을 구성하였다. 또한 저자들은 검색 가능 암호 시스템의 안전성 모델을 제시하고 그것을 바탕으로 해서 제안한 시스템이 안전하다는 것을 증명하였다.

먼저 논문에서 제안하는 알고리즘의 이해를 돕기 위하여 겹선형 사상(bilinear map)에 대해서 간단하게 소개하고자 한다.

Prime order p를 갖는 두 개의 group G_1, G_2가 있다고 가정하면 이들 두 group 사이에 형성되는 겹선형 사상은 $e : G_1 \times G_1 \rightarrow G_2$이고, 겹선형 사상은 아래의 3가지 성질을 만족한다.

1) Computable : G_1의 두 원소 g, h가 주어졌을 때, $e(g,h) \in G_2$를 계산할 수 있는 polynomial time 알고리즘이 존재한다.
2) Bilinear : 1과 p 사이에 존재하는 임의의 두 정수 x, y에 대해서, $e(g^x, h^y) = e(g,h)^{xy}$이다.
3) Non-degenerate : g가 G_1의 generator이면, $e(g,g)$는 G_2의 generator이다.

또한 논문에서는 임의의 길이를 갖는 입력값을 고정된 길이의 출력값으로 만드는 hash function이 사용되는데, 이들을 H_1, H_2로 표시한다. H_1은 G_1의 원소, H_2는 $\log p$ 길이의 데이터가 출력값으로 나온다.

$$H_1 : \{0,1\}^* \rightarrow G_1,\ H_2 : \{0,1\}^* \rightarrow \{0,1\}^{\log p}$$

공개키와 비밀키를 생성하는 과정은 다음과 같다.

사용자는 겹선형 사상이 잘 정의된 두 개의 군을 선택하고 비밀키 a와 공개키 $h = g^a$를 선택한다. 자료 D가 키워드 w를 포함하고 있다면 사용자는 임의의 r을 선택하여 $PEKS(w) =< A, B >=< g^r, H_2(e(H_1(w), h^r)) >$를 계산하여 인덱스를 생성한다.

키워드 w에 대한 트랩도어는 $T = [H_1(w)]^a$이다. 서버

는 주어진 트랩도어 T와 인덱스로 저장된 모든 $PEKS(w)$에 대해서 $H_2(e(T, A^a)) = B$를 만족하는 모든 자료를 검색의 결과로 사용자에게 전송한다.

Boneh 등이 제안한 본 시스템은 이후 설계된 많은 공개키 기반 검색 가능 암호 시스템의 바탕이 되었으며, 안전성 정의와 증명 방법 또한 여전히 사용되고 있다.

4.2 Conjunctive Keyword Search

Golle 등이 대칭키 암호화 방식에서 conjunctive keyword search 방법을 제안한 데 이어, Park 등은 2004년에 공개키 기반에서 conjunctive search가 가능한 방법을 처음으로 제안하였다[11].

이 방식에서도 PEKS와 마찬가지로 bilinear map을 사용하여 알고리즘을 구성한다. prime order를 갖는 G_1 G_2가 있고 이 두 그룹의 order를 q라고 할 때, 이 두 그룹 사이에 형성되는 bilinear map은 $e : G_1 \times G_1 \rightarrow G_2$ 이다. 또한 이 논문에서 사용되는 hash function H는 임의의 크기를 갖는 입력값을 받아 G_1의 원소를 출력하는데, 이를 $H : \{0,1\}^* \rightarrow G_1$으로 나타낼 수 있다.

메일 시스템을 예를 들어 제안하는 방법을 설명하면, Bob이Alice에게 메일을 보낸다고 가정한다. Bob은 보내려는 메일을 일반적인 공개키 암호화 방식을 사용하여 암호화하고, 서버가 사용자가 찾는 키워드들에 대해서 검색할 수 있도록 PECK(Public-key Encryption with Conjunctive field Keyword search)라는 형태의 데이터를 덧붙인다.

키를 생성하기 위해서 사용자는 0에서 $p-1$까지의 정수 중에서 s_1, s_2값을 랜덤하게 뽑는다. G_1의 generator를 g라고 하면, 사용자의 공개키는 $< g, y_1 = g^{s_1},\ y_2 = g^{s_2} >$이고, 비밀키는 $< s_1, s_2 >$이다.

사용자에게 암호화된 문서를 보내기 위해서는 0에서 $p-1$까지의 정수 중에서 r값을 랜덤하게 뽑는다. 본 방식에서도 keyword field 벡터를 사용하며, i번째 문서에 해당하는 벡터는 $D_i = (W_{i,1}, W_{i,2}, ..., W_{i,m})$이다. 이 때, i번째 문서에 대한 PECK는 $[e(H(W_1)^r, y_1), e(H(W_2)^r, y_1), ..., e(H(W_m)^r, y_1, y_2^r, g^r])$가 된다.

검색을 하기 위한 트랩 도어는 $T = [T_1, T_2, j_1, j_2, ..., j_t]$이다. 여기서 $j_1, j_2, ..., j_t$ 값은 m개의 keyword field 중 찾고자 하는 키워드들의 위치를 나타내는 인덱스이고 u는 0에서 $p-1$까지의 정수 중에서 랜덤하게 뽑은 값이다. 또한 T_1는 $(s_1/(s_1+u) \bmod p \cdot (H(W_{j_1}) + H(W_{j_1}) + ... + H(W_{j_t}))$이고 T_2는 u이다.

서버는 사용자로부터 받은 트랩도어와 암호화된 문서를 이용하여 사용자가 원하는 키워드들을 포함하는 문서가 있는지 검색한다. 위에서 계산된 암호 알고리즘의 출력 결과를 편의상 $[A_1, A_2, ..., A_m, B, C]$라고 하면, 서버는 $A_{j_1} \cdot A_{j_2} \cdot ... \cdot A_{j_t}$값과 $e(T_1, B + T_2 C)$값이 일치하는지 확인한다. 두 값이 일치하면 그 문서는 사용자가 찾는 키워드들이 포함된 문서이므로, 서버는 문서를 사용자에게 전달한다.

4.3 Multiuser Settings

이전에 제안된 방법들은 모두 암호문을 생성하는 사람과 검색을 위한 트랩도어를 생성하는 사람이 모두 한 명인 상황을 가정하고 있다. 따라서 사용자 Alice가 다수의 사람에게 암호화된 e-mail을 보내고자 할 때, Alice는 수신자들의 각 공개키를 이용하여 해당 e-mail에 대한 검색 가능 암호문들을 만든다. 이 때, n명의 사람에게 보낸다고 하면, Alice는 n번의 동일한 연산을 반복적으로 수행해야 하므로 비효율적이다. 하지만 검색 가능 암호 시스템이 적용 가능한 실제 서비스들은 다중 사용자 환경이 훨씬 일반적이므로, 다중 사용자 환경에서도 효율적인 검색 가능 암호 시스템을 설계하는 것이 필요하다.

2007년에 Hwang 등은 최초로 공개키 기반의 다중 사용자 환경을 위한 검색 가능 암호 시스템을 제안하였다[12]. 이 방식을 사용하면, 기존에 제안된 방법을 다중 사용자 환경에 적용하였을 때보다 검색 가능 암호문을 생성하는데 필요한 연산량 및 서버가 저장하고 있어야 하는 암호문의 크기를 줄일 수 있다. 이 방식에서는 다중 사용자 환경을 위한 PECK를 mPECK (multi-user Public Key Encryption with Conjunctive Keyword search)라고 부른다. mPECK는 앞에서 다루었던 PECK와 마찬가지로 키 생성 알고리즘, 암호화 알고리즘, 트랩도어 생성 알고리즘, 검색을 위한 테스트 알고리즘으로 구성되어 있으며, 사용자의 수가 n명이라고 가정한다.

키 생성 알고리즘은 Security parameter 1^k를 입력으로 받아, 다음과 같이 각 사용자에 해당하는 n개의 공개키/비공개키 쌍을 생성한다. 그리고, 앞에서 PECK가 공개키를 이용하여 키워드들의 집합 W에 대한 검색 가

능 암호문을 생성하였다면, mPECK에서는 n개의 공개키를 이용하여 W에 대한 공개키 검색 가능 암호문을 생성한다. 트랩도어 생성 과정은 특정 사용자의 비공개키와 키워드를 입력으로 받아 주어진 키워드 query에 대한 트랩도어를 생성하는 것으로 이루어진다. 검색 기능을 수행하는 사람은 트랩도어와 해당 사용자의 공개키 , 검색 가능암호문을 이용하여, 그 검색 가능 암호문이 키워드 query를 포함하는지 확인한다. 실제 암호문 생성과정과 검색 과정은 다음과 같다.

$x_1, ..., x_n$이 랜덤하게 선택된 Z_p^*의 원소일 때, $y_1 = g^{x_i}$라고 한다. 그러면 이 알고리즘으로부터 생성된 n개의 공개키/비공개키 쌍들은 $(pk_i, sk_i) = (y_1, x_1)$이다.

특정 문서의 키워드들의 집합이 $W = \{w_1, ..., w_l\}$라고 할 때, 이 알고리즘을 수행하는 사람은 Z_p^*에서 두 개의 랜덤한 값 s와 r을 선택하고, 다음의 값들을 계산한다. $A = g^r,\ B_j = y_j^s\ (1 \le j \le n),\ C_i = h_i^r f_i^s (1 \le i \le l)$

위의 식에서 $h_i = H_1(w_i),\ f_i = H_2(w_i)$ 이다. 그러면 이 알고리즘으로부터 생성되는 검색 가능 암호문은 $< A, B_1, ..., B_n, C_1, ..., C_l >$이다. 트랩도어를 생성하기 위해서는 Z_p*의 원소 t를 랜덤하게 선택하고 $T_{j,Q_1} = g^t$, $T_{j,Q_2} = (h_{I_1}, ..., h_{I_m})^t, T_{j,Q_3} = (f_{I_1}, ..., f_{I_m})^{t/x_j}$를 계산한다. 위에서 $Q = \{I_1, ..., I_m, w_{I_1}, ..., w_{I_m}\}$는 키워드 query를 의미한다. 이 때, 이 알고리즘으로부터 생성되는 트랩도어는 $(T_{j,Q_1}, T_{j,Q_2}, T_{j,Q_3}, I_1, ..., I_m)$이다.

검색을 수행하는 사람은 상대방으로부터 받은 트랩도어와 상대방의 공개키, 검색 가능 암호문을 이용하여 $e(T_{j,Q_1}, \Pi_{i=1}^m C_{I_i}) = e(A, T_{j,Q_2}) e(B_j, T_{j,Q_3})$인지 확인한다. 만약 관계식이 성립하면, 해당 검색 암호문이 찾는 키워드를 포함하고 있다고 판단하고 검색자에게 전달한다.

4.4 Conjunctive, Subset, Range Search

이전까지 제안되었던 검색 가능 암호 기법들은 모두 단순 비교(equality test)만을 제공하였다. 즉, 어떤 키워드를 검색 가능 암호를 통해서 암호화 하면, 나중에 사용자의 query하는 키워드가 encrypted keyword에 사용된 키워드와 같은지만을 확인할 수 있는 것이다. 2007년에 Boneh와 Waters는 이 같은 equality test 외에도 다양한 부가 검색 기능을 지니는 공개키 기반 검색 가능 암호 시스템을 제안하였다[13]. 이들은 일반적인 키워드 인덱스에 대해 효율적으로 conjunctive 검색을 수행할 수 있도록 시스템을 고안하였고, 키워드 인덱스의 결합을 통해서 subset과 범위 검색을 효율적으로 처리하였다.

사용자는 겹선형 사상이 잘 정의된 두 개의 군 G_1, G_2를 선택한다. 단, G_1, G_2는 일반적인 겹선형 사상 군이 소수 위수를 갖는 것에 비해 $n = pq$인 합성수 위수를 갖도록 선택한다. G_p, G_q는 각각 p, q를 위수로 갖는 G_1의 부분 군이다. 사용자는 자연수 l을 선택하고, 비밀키 $(u_1, h_1, w_1), ..., (u_l, h_l, w_l) \in G_p^3$을 임의로 선택한다. 또한 $g, v \in G_p,\ g_q \in G_q,\ a \in Z_p$를 비밀키로 선택한다.

시스템의 공개키로 $g_q, V = vR_v, A = e(g, v)^a$와 각각의 $i \in [1, l]$에 대해서 $U_i = u_i R_{u,i}, H_i = h_i R_{h,i},\ W_i = w_i R_{w,i}$를 계산하여 공개한다.

키워드는 l차원의 벡터로 표현되는데, $K = (k_1, k_1, ..., k_l)$에 대응하는 인덱스는 다음과 같이 생성된다. 사용자는 $s \in Z_n$과 $Z, (Z_{11}, Z_{12}), ..., (Z_{l1}, Z_{l2}) \in G_q$를 임의로 선택하여 $C = MA^s, C_0 = V^s Z$ 그리고 각각의 $i \in [1, l]$에 대해서, $C_{i,1} = (U_i^{k_i} H_i)^s Z_{i,1}, C_{i,2} = W_i^s Z_{i,2}$를 생성한다.

이 시스템에서는 키워드로 미리 정의된 '*'를 사용할 수 있는데, '*'는 어떠한 키워드와도 일치한다는 의미를 지닌다. 사용자가 검색하고자 하는 키워드를 $K = = (k_1, k_2, ..., k_l)$라고 하자. 각각의 k는 '*'의 값을 가질 수 있다. 사용자는 각각의 $i \in [1, l]$에 대해서 임의의 $r_{i,1}, r_{i,2}$를 선택한 후, 트랩 도어로 $T = (K, K_0 = g^a \Pi_{i \in [1,l]} (u_i^{k_i} h_i)^{r_{i,1}} w_i^{r_{i,2}}, K_{i,1} = v^{r_{i,1}}, K_{i,2} = v^{r_{i,2}})$를 생성한다. 서버는 주어진 트랩 도어 T에 대해서 $C'/(e(C_0, K_0)/\Pi_{i \in [1,l]} e(C_{i,1}, K_{i,1}) e(C_{i,2}, K_{i,2}))$를 계산하여 자료가 키워드와 일치하는지의 여부를 확인한다.

이 시스템은 인덱스를 생성할 때 키워드를 변형하여 다양한 부가 검색기능이 가능하도록 설계되었다. 예로 Conjunctive 비교 검색을 위해서는 다음과 같이 키워드를 선택한다. 사용자가 임의의 정수들 $a_1, a_2, ..., a_l \in [1, l]$에 대한 conjunctive 검색을 하고자 할 때, 키워드를 총 $l \times n$ bit로 선택하고 이를 l개의 n bit 블록으로 생각한다. 그 후 i 번째 블록의 a번째 bit만을 1로 표현하고 나머지 bit는 모두 0을 입력하여 인덱스를 생성한다. 트랩 도어 또한 동일한 키워드에 대해서 생성하면 검색결과는 모든 키워드 bit가 일치하는 자료만을 출력하므로 원하는 검색결과를 얻을 수 있다.

4.5 Efficiently Searchable Encryption Using Deterministic Algorithm

이전까지 암호학계에서 제안돼 온 검색 가능 암호 시스템에는 모두 공통된 문제가 있는데, 그것은 실제로 서버가 검색을 수행할 때 모든 레코드에 대해서 순차적으로 비교를 해봐야 한다는 것이다. 반면, 효율적인 database architecture를 연구하는 분야에서 제안되어 온 연구결과에서는 효율적인 검색이 가능하면서 부가적으로 암호화를 제공하는 방식들이 주를 이루고 있다. 때문에 안전성이 명확하게 증명되어 있지 않고 암호화와 관련된 기술들이 다소 모호하게 기술되어 있다.

2007년에 Bellare 등은 이러한 두 가지 부류의 연구결과들의 차이점을 메우기 위한 방법을 제안하였다[14]. 여기에서 차이점을 메운다는 뜻은, 효율적이면서도 안전도가 기존의 암호학계에서 요구되는 가장 높은 안전도를 제공한다는 뜻은 아니며, 효율성을 위해서 안전도를 어느 정도 희생해야 얻을 수 있는 것이다. 즉 randomized encryption을 사용하는 대신에 deterministic encryption을 사용함으로써 binary search와 같은 효율적인 검색 방법을 사용할 수 있게 하는 대신에, 동일한 키워드를 암호화하면 항상 같은 결과가 나오게 된다. 그러므로 일반적으로 암호학에서 암호 메커니즘의 가장 높은 안전도를 나타내는 indistinguishability는 제공할 수가 없다.

대신에 Bellare 등은 새로이 제안한 ESE(Efficiently Searchable Encryption)을 위해서 privacy라는 새로운 안전성의 척도를 제시하고 있다. 이것은 공격자가 암호문, 즉 encrypted keyword만 가지고는 사용된 평문 keyword를 추측할 수가 없다는 것이다. 물론, 평문-암호문 쌍을 얻을 수 있다면 이전에 얻었던 평문 keyword에 대한 encrypted keyword는 항상 같게 되므로 동일한 키워드에 대한 encrypted keyword를 구분해 낼 수 있고 어떤 키워드가 사용되고 있는지도 알 수 있게 된다. 따라서 privacy는 indistinguishability보다는 낮은 안전성 모델이라고 볼 수 있다.

ESE에서는 사용자가 생성한 암호문(encrypted keyword)를 얻은 누구나가 해당 암호문으로부터 태그라는 정보를 얻을 수 있도록 하고 있다. 이 태그는 나중에 검색 과정에서 단순 비교(equality test)를 통해서 원하는 키워드에 대한 encrypted keyword인지를 판단하는 데 사용된다. 이 태그들은 encrypted keyword를 얻는 즉시 추출해서 정렬하여 저장해 둘 수 있다. 검색을 원하는 사용자가 전송하는 트랩도어는 실제로 이 태그와 동일한 형식의 것이어서 서버는 단순히 저장돼 있는 태그들 중에서 트랩도어와 같은 것을 찾아내기만 하면 된다. 따라서 각각의 레코드마다 매번 어떤 계산을 수행하고 비교하는 linear search를 사용하는 것이 아니라, binary search와 같은 보다 효율적인 검색 방법을 사용할 수 있다.

ESE는 다양한 방식으로 구현할 수 있으나 여기에서는 deterministic public key encryption을 사용하는 방식을 설명한다. PubEnc {KeyGen, Enc, Dec}는 세 개의 알고리즘(키 생성, 암호화, 복호화)으로 구성된 공개키 암호 메커니즘이라고 하자. 이 PubEnc에는 일반적으로 널리 알려진 RSA encryption mechanism을 사용할 수도 있다. 단, deterministic encryption이어야 하기 때문에 random padding은 사용하지 않아야 한다.

키 생성(key generation)은 공개키 암호 자체의 공개키와 비밀키 생성과 동일하며 암호화 과정은 공개키를 이용하여 키워드를 암호화하는 것이 된다. 또한 트랩도어는 검색하려는 키워드를 공개키로 암호화해서 생성하며, 검색을 위한 테스트는 서버에 저장된 암호문 중에서 트랩도어와 같은 것이 있는지를 확인하는 과정이 된다.

이 경우는 암호문의 태그가 그 암호문 자체가 된다. 따라서 별도의 추출과정이 필요하지 않다. 주목할 점은 설사 서버가 사용자의 공개키를 가지고 있지 않아도 태그가 무엇인지 알 수 있다는 것이다. 또한 검색을 원하는 사용자도 트랩도어를 만드는 과정에서 비밀키를 사용하지 않는다.

V. 결 론

지금까지 검색 가능 암호 기술의 개요와 학계에서의 연구 동향에 대해 살펴보았다. 검색 가능 암호 기술은 기존의 암호 기술과 같이 암호화된 정보에 대한 안전성을 보장하면서 동시에 특정 키워드를 포함하는 정보를 검색할 수 있도록 고안되었으며 정보가 외부 저장 공간에 저장되면서 발생하는 문제점에 대한 해결 방안 중 하나로 주목받고 있다.

지금까지 진행된 연구에서 기본적인 키워드 검색 기능 외에도 범위 검색, conjunctive 검색 등의 검색 기능과 다중 사용자 환경 지원 등 다양한 기능을 제공하는 시스템이 제안되었다.

검색 가능 암호 기술을 비롯해서, 암호화에 따른 검색 성능 저하 문제를 해결하고자 하는 시도는 학계를 심으로 현재도 많은 연구가 진행되어 가고 있는 상황이며, 앞으로 더 효율적이고 실용적인 검색 가능 암호 시스템이 제안될 것으로 기대된다.

참고문헌

[1] 전자신문, "대만서 사상 최대 정보유출 사건", 2008. 8. 28

[2] 전자신문, "美 사상 최악 신용카드 고객정보 도용", 2009. 1. 22

[3] D. Song, D. Wagner, and A. Perrig, "Practical Techniques for Searching on Encrypted Data," Proceedings of IEEE Symposium on Security and Privacy, pp. 44-55, 2000.

[4] B. H. Bloom, "Space/Time Trade-offs in Hash Coding with Allowable Errors," Communications of the ACM, Vol. 13, No. 7, 1970.

[5] A. Broder, M. Mitzenmacher, "Network Appli cations of Bloom Filters: A Survey," Internet Mathematics, Vol. 1, No. 4, 2003.

[6] E. J. Goh, "Secure Indexes," Technical Report, 2003/216, IACR ePrint Crpytography Archive, 2003.

[7] P. Golle, J. Staddon, and B. Waters, "Secure Conjunctive Keyword Search over Encrypted Data," Applied Cryptography and Network Security Conference-ACNS, LNCS 3089, pp. 31-45, 2004.

[8] Y. C. Chang and M. Mitzenmacher, "Privacy Preserving Keyword Searches on Remote Encrypted Data," Applied Cryptography and Network Security Conference-ACNS, LNCS 3531, pp. 442-255, 2005.

[9] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable Symmetric Encryption: Improved Definitions and Efficient Constructions," Proceedings of the 13th ACM conference on computer and communication s security-ACM-CCS, pp. 79-88, 2006.

[10] D. Boneh, G. Crescenzo, R. Ostrovsky, and G. Persiano, "Public Key Encryption with Keyword Search," Advances in Cryptology-EUROCRYPT, LNCS 3027, pp. 506-522 ,2004.

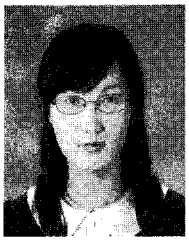
[11] D. J. Park, K. H. Kim and P. J. Lee, "Public Key encryption with conjunctive field keyword search," Workshop on Information Security Applications-WISA, LNCS 3325 pp. 73-86, 2004.

[12] Y. H. Hwang and P. J. Lee, "Public Key Encryption with Conjunctive Keyword Search and its Extension to a Multi-User System," International Conference on Pairing-Based Cryptography-Pairing, LNCS 4575, pp. 2-22, 2007.

[13] D. Boneh and B. Waters, "Conjunctive, Subset and Range Queries on Encrypted Data," Theory of Cryptography Conference-TCC., LNCS 4392, pp. 535-554, 2007.

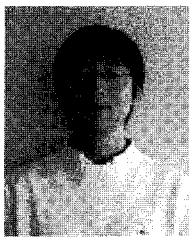
[14] M. Bellare, A. Boldyreva, and A. O'Neill, "Deterministic and efficiently searchable encryption," Advances in Cryptology-CRYPTO, LNCS 4622, pp. 535-552, 2007.

〈著者紹介〉



김 선 영 (Sun Young Kim)
학생회원

2008년 2월: 한동대학교 전산전자공학부 졸업
2008년 3월~현재: 포항공과대학교 정보통신대학원 석사과정
<관심분야> 암호학, 정보보호



서 재 우 (Jae Woo Seo)
학생회원

2005년 2월: 경북대학교 전자전기공학부 졸업
2005년 3월~현재: 포항공과대학교 전자전기공학과 박사과정
<관심분야> 암호이론, 암호 프로토콜, 정보보호

이 필 중 (Pil Joong Lee)
종신회원

1974년 2월: 서울대학교 전자공학과 졸업
1977년 3월: 한국대학교 전자공학과 석사
1982년 6월: U.C.L.A System Science. Engineer
1985년 6월: U.C.L.A Electrical Engineering. Ph.D
1980년 3월~1985년 8월: Jet Propulsion Laboratory. Senior Engineer
1985년 8월~1990년 2월: Bell Communication Research. M.T.S
1990년 2월~현재: 포항공과대학교 전자전기공학과 교수
1996년 2월~1997년 2월: NEC Research Institute 방문 연구원
2000년 9월~2003년 8월: 포항공과대학교 정보통신 연구소장 (정보통신 대학원장 겸임)
2004년 1월~2004년 12월: 한국정보보호학회 회장
2004년 1월~2004년 12월: KT 정보보호 자문위원
2008년 7월~2008년 12월: POSDATA 정보보호 자문위원
2007년 1월~현재: 한국공학한림원 정회원
<관심분야> 정보보호 전반