정보보호개론(CSE441)
www.facebook.com/KoreaTech.Crypto

고급 암호기술 1부: 임계 기반 비밀 공유, 데이터 아웃소싱 보호 기술

NOTE 13

한국기술교육대학교 컴퓨터공학부 김상진

sangjin@koreatech.ac.kr
www.facebook.com/sangjin.kim.koreatech

교육목표

- Signcryption: 서명과 암호화를 동시에
- 비밀공유 기법
- 데이터 아웃소싱 (클라우드 컴퓨팅)
  - 서버 신뢰 문제
    - 데이터 보호와 공유
    - 무결성 서비스
  - 데이터 아웃소싱에서 사용되는 암호기술
    - 프록시 재암호화
    - 속성기반 암호화
    - 검색가능 암호시스템

Signcryption

- 목적. 전자서명과 암호화를 각각 별도로 하는 것보다 한 번에 효과적으로 할 수 있도록 해줌
- Alice가 Bob에게 signcryption하여 메시지를 전달하였을 때 요구사항
  - R1. Bob만 메시지를 복호화할 수 있어야 함
  - R2. Bob은 Alice의 서명을 확인할 수 있어야 함
  - R3. 전체 비용은 각각을 별도로 하였을 때보다 연산 비용이 효율적이어야 하며, 결과 값의 길이가 줄어야 함
- 각각 하는 것과 비교하였을 때 차이점
  - 일반 서명과 달리 제3자는 서명 결과를 확인하는 것이 어려움
- 각각 별도로 하는 방식: $\mathrm{Sig.}-K_A(M), E.K(M), E.+K_B(K)$
- Signcryption 방식: $\mathrm{Signcrypt}(-K_A, +K_B, M)$
  - RSA의 예) $\left(M^{d_A} \bmod n_A\right)^{e_B} \bmod n_B$
    - RSA sincryption의 한계점. 작은 메시지만 위처럼 할 수 있음. 보통은 큰 메시지도 signcryption을 할 수 있음 (교재 참고)

$\{\mathrm{Sig.}-K_A(M)\}.+K_B$

비밀 공유 기법 (1/2)

- 특정 권한을 특정 사용자가 홀로 가지고 있으면 해당 권한을 쉽게 남용할 수 있음
  - 이 문제는 권한 분산을 통해 해결함
  - 보통 권한을 n명에게 분산하고 t명 이상이 협조하면 해당 권한을 행사할 수 있도록 함 (t: 보안 변수)
    - 이를 (t, n) 임계(threshold) 기반 방식이라 함
    - 암호기술에서도 가용성 문제 때문에 (n, n) 방식은 사용하지 않음
- 암호기술에서 권한은 암호키의 소유 여부에 의해 결정됨
  - 예) 인증기관의 서명키, 신원기반 시스템 PKG의 마스터키
- 암호기술에서는 주로 암호키를 나누어 권한을 분산함
  - 이때 사용하는 기술을 비밀 공유 기법이라 함
  - 각 사용자가 유지하는 값을 그 사용자의 몫(share, shadow)이라 함

비밀 공유 기법 (2/2)

- 비밀 공유 기법의 분류: 중앙집중 vs. 분산
  - 공유할 값을 나누고 몫을 각 사용자에게 전달하는 서버가 있는 경우
    - 요구사항. 서버로부터 받은 몫이 유효한 몫인지 확인할 수 있어야 함 (verifiable secret sharing)
    - 서버는 해당 비밀을 알고 있어 비밀 공유 기법의 원래 목적과 상충되는 측면이 있음
  - 서버를 사용하지 않고 참여하는 사용자 간의 값을 안전하게 나눌 수 있는 경우 ⟹ 분산 비밀 공유 기법이라 함

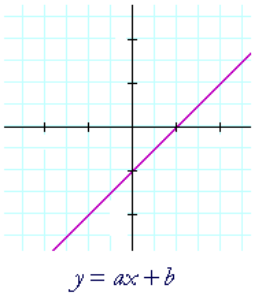
간단한 비밀 공유 기법

- 방법 1. 대칭키 K를 n명이 공유하는 간단한 방법
  - K와 같은 크기의 S_i를 $n-1$개 랜덤하게 생성함
  - $S_n = K \oplus S_1 \oplus \cdots \oplus S_{n-1}$를 계산하고 각 사용자 i에게 S_i값을 전달함
    - S_i가 사용자 i의 몫임
  - n명의 사용자가 모두 모이면 K를 복원할 수 있음
  - 문제점
    - 문제점 1. 사용자의 몫을 생성해 줄 중앙 신뢰 서버가 필요함
    - 문제점 2. 모두 모여야 복원할 수 있음
    - 문제점 3. 하나의 조각이 분실되면 복원할 수 없음
    - 문제점 4. 한 번 복원하면 비밀키가 노출됨
  - 문제점 2와 3은 이 방식이 (n, n) 방식이기 때문에 나타나는 것임
  - 대칭키는 문제점 4를 근본적으로 해결하기 힘듦
    - 공개키 방식에서는 몫을 노출하지 않고 활용 가능함

Shamir의 (t, n) 비밀 공유 기법 (1/2)

- 방법 2. $t-1$차 다항식을 사용한다.
  - 예) $t=2, f(x)=ax+K$
    - 공유하고자 하는 값은 K임
    - 사용자들은 a와 K를 모름
    - 각 사용자 i마다 $f(i)$ 값을 줌
    - 임의의 두 명의 사용자가 모이면 K값을 계산할 수 있음

$$K = f(i) - (f(i) - f(j))/(i-j) \times i$$



- 이 방식을 사용하면 문제점 2와 3이 해결됨
- 문제점 1를 극복하는 방법
  - 각 사용자는 a_i와 K_i를 임의로 선택하여 다항식 $f_i(x) = a_i x + K_i$를 생성함
  - 최종적으로 사용하는 다항식 $f(x) = \sum a_i\, x + \sum K_i$가 됨
  - 각 사용자는 다른 사용자 j에게 $f_i(j)$ 값을 계산하여 전달함. $f(j) = \sum f_i(j)$
  - 두 명이 모이면 위와 동일하게 K를 계산할 수 있음
    - 그런데 어떻게 활용???

Shamir의 (t, n) 비밀 공유 기법 (2/2)

- (t, n) 비밀 공유 기법
  - 시스템 설정
    - $t-1$개의 랜덤 계수 a_1부터 a_{t-1}를 생성한 후에 다음 다항식을 사용

$$f(x) = a_{t-1}x^{t-1} + \cdots + a_1 x + K,$$

    - 각 사용자 i에게 $f(i)$값을 비밀스럽게 전달함
  - Lagrange Interpolation Formula
    - t개의 좌표를 지나는 독특한 $t-1$차 다항식을 다음과 같이 만들 수 있음

$$f(x) = \sum_{j=1}^{t} f(j) \prod_{1 \le k \le t, k \ne j} \frac{x-k}{j-k}$$

    - $f(0) = K, \quad K = \sum_{j=1}^{t} f(j) \prod_{1 \le k \le t, k \ne j} \frac{k}{k-j}$
    - $L_j = \prod_{1 \le k \le t, k \ne j} \frac{k}{k-j}, \quad K = \sum_{j=1}^{t} f(j) L_j$

Threshold ElGamal 암호기법 (1/2)

$y = g^x$
$(g^r, C = E.K_1(M), MAC.K_2(C))$
$K_1||K_2 = KDF(y^r)$

- 시스템 설정
  - 군 설정: $\mathbb{Z}_p^*$의 부분군 G_q=⟨g⟩
  - 서버는 법 q에서 $t-1$차 다항식을 생성함
    $$f(x) = a_{t-1}x^{t-1} + \cdots + a_1 x + s, \qquad \forall a_i \in_R \mathbb{Z}_q^*$$
  - 각 참여자 i에게 $s_i = f(i)$를 비밀스럽게 전달함(몫)
  - 시스템의 공개키는 $y = g^s$가 됨
  - Lagrange Interpolation 기법에 의해 다음이 성립함
    $$b_j = \prod_{1 \le k \le t, k \ne j} \frac{k}{k-j}, \qquad s = \sum_{j=1}^{t} s_j b_j$$
  - 서버가 $c_{t-1} = g^{a_{t-1}}, \cdots, c_2 = g^{a_2}, c_1 = g^{a_1}$을 공개하면 각 사용자는 자신의 몫을 다음을 이용하여 확인할 수 있음
    $$g^{s_i} ? = y \prod_{j=1}^{t-1} c_j^{ij}$$

Threshold ElGamal 암호기법 (2/2)

- 시스템의 공개키 y를 이용하면 누구나 ElGamal 혼합 암호기법을 이용하여 다음과 같이 메시지 M을 암호화할 수 있음
  $$(g^r, C = E.K_1(M), MAC.K_2(C)), \qquad K_1||K_2 = KDF(y^r)$$
- t명이 모이면 다음과 같이 이 암호문을 복호화할 수 있음
  - 단계 1. 각 참여자는 $w_i = (g^r)^{s_i}$를 공개함. 추가적으로 이것의 유효성을 증명할 수 있음
  - 단계 2. 공개된 값들을 이용하면 누구나 다음을 이용하여 복호화할 수 있음
    $$\prod w_i^{b_i} = \prod g^{r s_i b_i} = g^{r \sum s_i b_i} = g^{rs} = y^r$$
- 특징. 문제점 4가 극복됨
- 각 참여자가 각자 다항식을 만들어 분산 임계기반 ElGamal 암호기법을 만들 수 있음

데이터 아웃소싱 (1/2)

- 데이터 아웃소싱(data outsourcing)이란 데이터를 내부 서버에 유지하는 것이 아니라 외부 서버에 위탁하여 관리하는 것을 말함
- 데이터 관리 비용(장비 구입 및 유지 관리 비용, 개발 비용, 인건비, 공간 사용료 등)을 줄이기 위해 현재 데이터 아웃소싱이 증가하고 있으며, 클라우드 컴퓨팅의 확산으로 이와 같은 현상이 더욱 가속화되고 있음
- 문제점. 중요한 데이터, 사적인 데이터를 자신이 통제할 수 없는 외부에 유지하는 것이 불안함
  - 정보 노출 문제, 정보 조작, 손실 문제 등 불안 요소가 많음
- 해결방법
  - 정보 노출 문제: 데이터를 암호화하여 유지하는 것이 필요
    - 서비스 제공자도 볼 수 없도록 하고 싶음
  - 정보 조작, 손실 문제: 무결성 검증 서비스 필요
    - 서비스 제공자는 책임을 회피하고 싶을 수 있음
- 해결 방법을 사용하여도 편리성(검색)은 그대로 유지할 수 있나?

데이터 아웃소싱 (2/2)

- 데이터 아웃소싱 요구사항
  - 정보 노출, 조작 문제 등을 해결하면서 편리성은 그대로 유지
    - 비밀성, 프라이버시 보장
    - 접근제어 보장: 공유, 공개 범위 설정 등
    - 효율적인 검색
- 이와 같은 요구사항이 충족되면서 서버에 대한 신뢰 의존 정도는 최소화하고 싶음. 가능할까?
  - 서버에게 데이터를 노출하지 않고 서비스를 받고 싶음

아웃소싱된 데이터의 보호

- 아웃소싱된 데이터의 보호를 강화하기 위해 데이터를 암호화하여 유지함
  - 이 공간은 사용자가 직접 관리할 수 있는 공간이 아님
  - 문제점. 필요한 저장 공간이 증가함 (IV벡터, 채우기, 암호화된 키 등)
- 사용자들은 자신이 원하는 데이터를 다운받아 복호화하고 싶음
- 해결책 1. 저장 서버가 모든 권한을 가지는 경우
  - 서버가 데이터를 암호화하고 암호화키를 관리함
  - 사용자가 요청하면 서버가 접근 권한을 확인하고 암호화된 데이터와 이를 복호화하기 위한 키를 함께 제공함
  - 문제점. 서버를 무조건 신뢰하여야 함
- 해결책 2. 사용자가 모든 권한을 가지는 경우
  - 사용자는 각 데이터를 업로드할 때 이를 랜덤 대칭키로 암호화함
  - 랜덤 대칭키는 자신의 공개키로 암호화하여 함께 저장함
  - 사용자가 요청하면 서버는 접근 권한을 확인한 후에 암호화된 데이터와 공개키로 암호화된 대칭키를 제공함
  - 문제점. 공유는 어떻게? 서버 신뢰 문제는 해결?

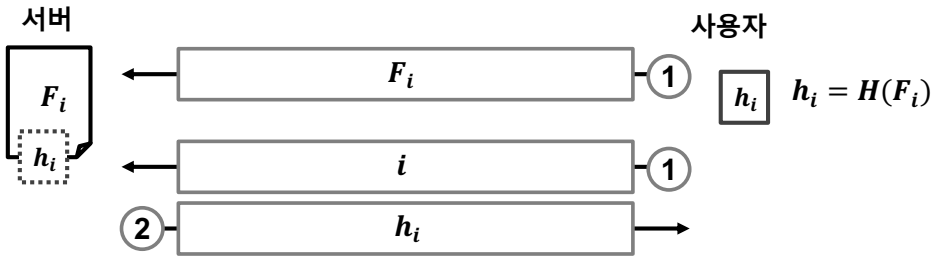
아웃소싱된 데이터의 무결성 검증

- 목표. 아웃소싱된 데이터를 자신이 필요할 때 변하지 않은 상태로 접근할 수 있기를 원함
  - 무결성을 확인해야 하는 데이터의 크기는 다양할 수 있고, 매우 클 수 있음
    - 사용자는 파일 단위, 폴더 단위로 확인하고 싶을 수 있음
  - 무결성 검증은 해당 데이터를 다운받지 않은 상태에서 확인할 수 있어야 함
    - 사용자는 해당 데이터를 로컬에 가지고 있지 않다고 가정함
    - 하지만 무결성 확인하기 위한 무언가는 유지하고 있어야 함
  - 서버도 무결성 증명을 위해 데이터 전체에 대한 접근 없이 제공할 수 있어야 효과적인 서비스가 될 수 있음
  - 무결성 서비스에 대해 저장 서버를 신뢰할 수 없음
    - 데이터 변조에 대한 책임 회피 목적

POR(Proof Of Retrievability) (1/6)

- 해시함수 기반 간단 해결책 1.
  - 사용자는 파일의 해시값을 유지함. 사용자의 공간 복잡도: $O(1)$
  - 사용자 요청에 따라 파일의 해시값을 계산한 다음 서명하여 전달함
  - 문제점 1. 파일 크기에 따라 해시값을 계산하는 것이 고비용임
  - 문제점 2. 서버는 항상 해시값을 보관한 다음에 매번 새롭게 계산하지 않고 이전 해시값을 전달할 수 있음

파일 단위 공간 복잡도



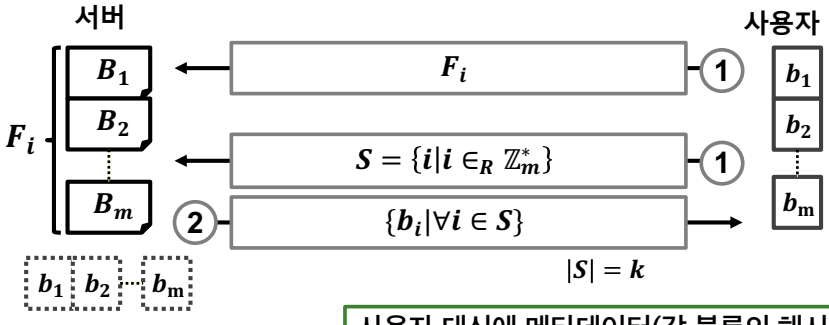
서버는 매번 새롭게 계산하도록 해야 함

사용하는 저장 공간이 증가하지 않음

POR (2/6)

- 해시함수 기반 간단 해결책 2.
  - 전체 파일을 정해진 블록 크기로 나누어 관리함
  - 사용자는 각 블록의 해시값을 유지함. 사용자의 공간 복잡도: $O(m)$
  - 일정한 수의 랜덤 블록에 대한 해시값을 요청함
  - 문제점 1. 사용자는 블록 개수만큼의 해시값을 유지해야 함
    - 차라리 파일을 유지하는 것이….
  - 문제점 2. 서버도 각 블록의 해시값을 유지하여 속일 수 있음
    - 하지만 서버의 이득은 별로…

m: 파일의 블록 수

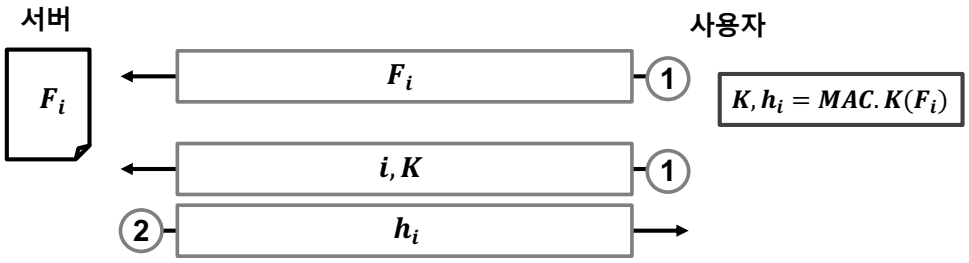


사용자 대신에 메타데이터(각 블록의 해시값)를 유지하고 검증하여 주는 서버를 사용하는 모델도 있음

POR (3/6)

- MAC 기반 간단 해결책 1.
  - 사용자는 랜덤한 키 *K*를 준비하여 MAC.*K*(*F*)를 계산함
  - 사용자는 MAC 값과 키를 유지함
  - 사용자는 서버에 *K*를 전달하여 MAC 값을 요청함
  - 키를 여러 개 준비하면 여러 번 수행할 수 있음. 사용자의 공간 복잡도: $O(n)$
    - 사용자는 준비한 *K*마다 MAC 값을 유지해야 함
  - 해시 기반 해결책 1의 문제점 2만 해결됨

n: 사전 계산한 MAC 값의 수

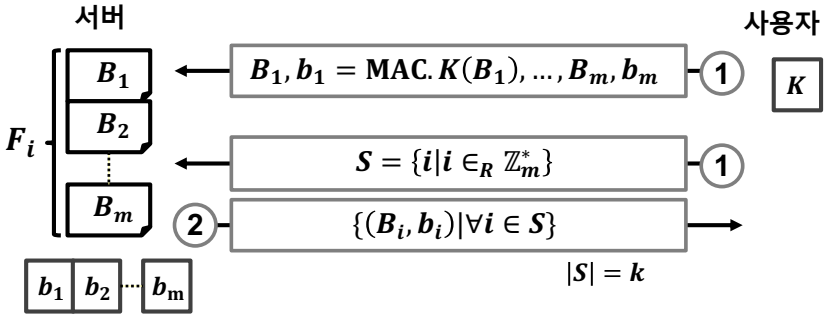


POR (4/6)

- MAC 기반 간단 해결책 2.
  - 사용자는 각 블록의 MAC 값을 계산하여 전달함. 사용자는 키만 유지함
  - 서버는 파일과 각 블록의 MAC 값을 유지함
    - 서버는 MAC 값을 생성할 수 없음
  - 일정한 수의 랜덤 블록에 대한 블록과 MAC 값 쌍을 요청함
  - 서버 비용 증가, 통신 비용 증가

로컬에는 유지하지 않지만 서버가 유지함. 비용 증가

전체 파일을 다운받지 않아도 파일의 일부를 다운받아 확인하는 형태



POR (5/6)

- RSA 기반: Filho와 Barreto 기법
  - 사용자: $N = pq, k = F \bmod \phi(N)$를 유지함
  - 증명 과정:
    - 사용자 → 서버: $g \in_R \mathbb{Z}_N^*, N$
    - 서버 → 사용자: $s = g^F \bmod N$
      - 사용자: $s? = g^k \bmod N$
  - 핵심 생각
    - 사용자는 매번 새로운 g를 전달하기 때문에 매번 새롭게 계산해야 함
    - 서버는 k를 계산할 수 없기 때문에 직접 F를 이용해 s를 계산할 수밖에 없음

- File $= F_1 \| F_2 \| \ldots \| F_n$
- $F = \sum F_i$
- $s_i = g^{F_i} \bmod N$
- $s = \prod s_i$

- MAC 기반 간단한 해결책 1의 문제점을 해결하는 기법
- 서버는 매번 새롭게 MAC 값을 계산하지만 사용자는 여러 개 MAC 값을 유지할 필요가 없는 기법
- 깔끔한 해결책이지만 서버 계산 비용이 비교적 높은 문제점은 있음

POR (6/6)

동형 암호: 암호화된 데이터를 복호화하지 않고 연산을 적용할 수 있는 암호
예) $E(M_1)E(M_2) = E(M_1 + M_2)$

- MAC 기반 간단 해결책 2의 개선
  - 기존 해결책의 문제점. 통신 비용, 서버는 매번 새롭게 계산하는 것이 아님
    - 서버는 c개의 블록과 그 블록의 MAC 값을 전달해야 함
  - 해결방법: 동형 암호(homomorphic encryption)
    - 서버는 c개의 파일 블록을 결합하여 하나의 블록 값을 만들고 c개 MAC을 결합하여 하나의 MAC 값을 만들어 전달함
    - 사용자는 이를 확인하여 c개 블록을 검증함

d, v: 사용자의 개인키
m_i: i번째 블록
T_{i,m_i}: MAC값
g_s: 사용자가 전달한 도전값

$$T_{i,m_i} = (h(W_i)g^{m_i})^d \bmod n, W_i = v \| i$$

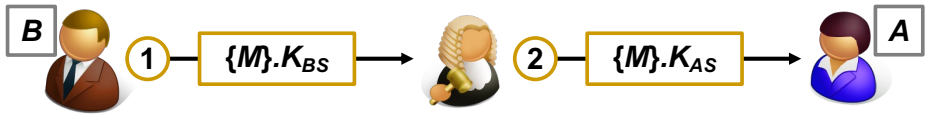
$$T = T_{i_1,m_{i_1}} T_{i_2,m_{i_2}} \cdots T_{i_c,m_{i_c}}$$
$$= \left(h(W_{i_1}) \cdots h(W_{i_c}) g^{m_{i_1}+\cdots+m_{i_c}}\right)^d \bmod n$$
$$\rho = H\left(g_s^{m_{i_1}+\cdots+m_{i_c}} \bmod n\right), g_s = g^s$$

$$\tau = h(W_{i_1}) \cdots h(W_{i_c}) \bmod n$$
$$H\left(\left(T^e \tau^{-1}\right)^s \bmod n\right) ? = \rho$$

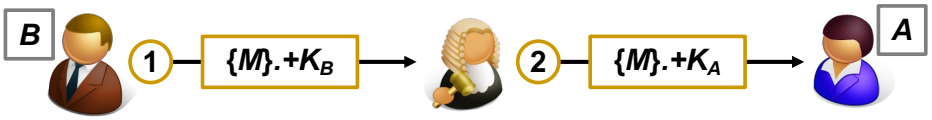
Ateniese 등의 CCS'07 E-PDP

프록시 재암호화

- 프록시 재암호화(proxy re-encryption): 한 사용자의 개인키로 복호화할 수 있는 암호화된 데이터를 프록시가 다른 사용자의 개인키로 복호화할 수 있도록 변환하여 주는 것을 말함
  - 이때 프록시는 데이터를 볼 수 없어야 함



이렇게 하는 것은 프록시 재암호화가 아님



RK_{BA}

B의 공개키로 암호화된 암호문을 A의 공개키로 암호화된 암호문으로 바꿀 수 있는 re-encryption 키
→ B가 생성함
B가 권한을 주는 형태임

왜 B가 직접 $+K_A$로 암호화하지 않고 이렇게 할까?
M을 주어야 하는 사용자가 많으면?

프록시 재암호화를 이용한 그룹 통신

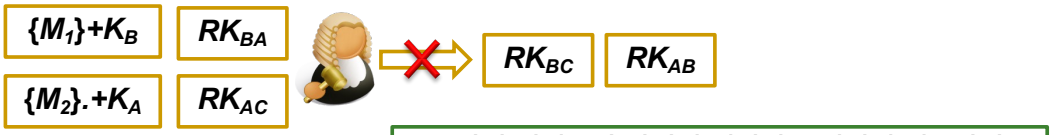
- 동일 데이터를 여러 사용자(n명)에게 비밀로 전송하고 싶을 경우 사용할 수 있는 방법은?
- 방법 1. 그룹키 프로토콜을 사용함
  - 비교적 초기설정 비용이 고가임. 하지만 오랫동안 전방향/후방향 안전성을 제공하고 싶으면 가장 효과적인 방법일 수 있음
- 방법 2. 프록시 재암호화 기법을 사용함
  - 전송자는 한 번만 자신의 공개키로 암호화함
  - 프록시는 n번의 재암호화를 해야함
  - n개의 재암호화키가 생성되어야 하며, 프록시에게 전달되어야 하고, 프록시는 이들을 유지해야 함
    - 하지만 이 키들은 계속 사용 가능함
- 방법 3. 속성기반 암호기법 사용 (슬라이드 25)
- 3가지 방법의 공통점. TTP의 참여 필요 (키 분배 서버, 프록시, 키 발급 서버)

데이터 아웃소싱과 프록시 재암호화

- 공유 문제 해결
  - 사용자가 데이터를 아웃소싱할 때 랜덤 대칭키로 암호화하고 대칭키는 자신의 공개키로 암호화하여 함께 유지한다고 가정함
    - $\{F_1\}.K_1$, $\{K_1\}.+K_A$, …, $\{F_n\}.K_n$, $\{K_n\}.+K_A$
  - 사용자는 특정 데이터에 대한 재암호화키를 발급하여 다른 사용자와 해당 데이터를 공유할 수 있음
    - F_1를 *B*와 *C*에게 공유하고 싶으면 RK_{AB}, RK_{AC}를 프록시에 제공
- 문제점. 지금까지 제안된 프록시 재암호화 기술은 세밀한 수준의 접근제어를 제공하지 못함
  - 사용자 *A*에서 사용자 *B*로 재암호화키가 있는 경우 사용자 *A*의 공개키로 암호화된 모든 문서를 재암호화할 수 있음
    - 위 예에서 프록시는 F_1부터 F_n까지 모두 재암호화해줄 수 있음

프록시 재암호화 요구사항

- 가장 기본적인 특성
  - 사용자는 재암호화에 대해서만 프록시를 신뢰함 (honest-but-curious)
    - 프록시가 협조를 하지 않으면 이 기능은 아무 의미가 없음
- 기본 요구사항
  - R1. 프록시는 재암호화 과정에서 평문을 얻을 수 없어야 함
  - R2. 프록시 권한 제한: 프록시는 권한이 부여된 재암호화만 할 수 있어야 함
    - 사용자가 재암호화키를 만들어 주지 않으면 재암호화가 가능하지 않아야 함
  - R3. 공모 안전성(collusion-safety): 사용자 간 공모 또는 사용자와 프록시 간의 공모를 통해 불법적인 복호화가 가능하지 않아야 함



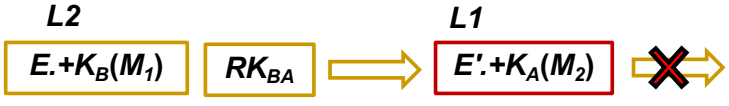
공모하기 위해 *C*가 자신의 개인키를 서버에 제공하여도 다른 사용자에게 나쁜 영향을 미칠 수 없어야 함

프록시 재암호화 추가 요구사항

- 단방향(unidirectional): Alice에서 Bob으로 재암호화 권한이 있어도 자동으로 Bob에서 Alice로의 재암호화 권한이 위임되지 않음
  - $RK_{AB} \not\Rightarrow RK_{BA}$
- 비추이성(non-transitive): 프록시는 스스로 재암호화 권한을 위임 받을 수 없어야 함
  - Alice에서 Bob으로 재암호화키, Bob에서 Carol로 재암호화키를 프록시가 보유하고 있어도 Alice에서 Carol로 재암호화키를 만들 수 없어야 함
  - $RK_{AB}, RK_{BC} \not\Rightarrow RK_{AC}$
- 비전이성(non-transferable): 프록시는 사용자들과 공모하여도 위임 받지 않은 권한을 위임 받을 수 없어야 함
  - Alice에서 Bob으로 재암호화키, Bob의 개인키, Carol의 공개키가 있더라도 Alice에서 Carol로 재암호화키를 생성할 수 없어야 함
    - $RK_{AB}, -K_B, +K_C \not\Rightarrow RK_{AC}$
- 비상호작용(non-interactive): Alice가 Bob을 위한 재암호화키를 생성하고자 할 때 제3의 신뢰 서버와 어떤 형태의 상호작용도 필요 없어야 함

두 단계 암호화 수준

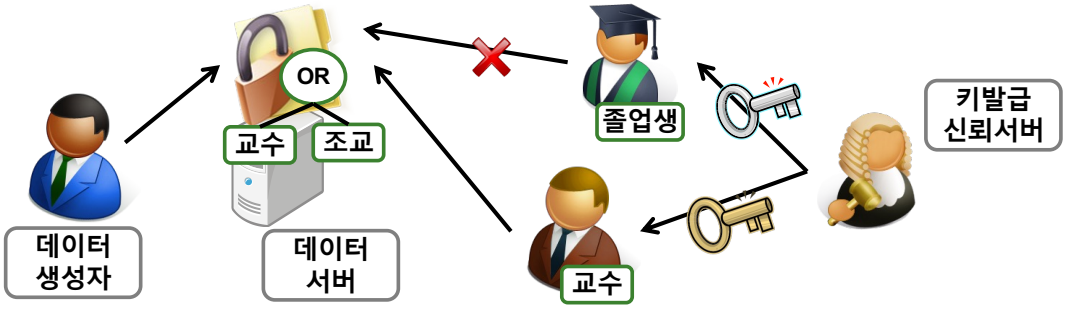
- 공모 안전성, 비추이성, 비전이성 요구사항을 만족시키기 위해 보통 두 단계 암호화 수준 제공
  - 재암호화가 가능하지 않은 암호알고리즘 L1과 재암호화가 가능한 암호알고리즘 L2를 함께 사용함 (L1, L2는 암호문 형태/구조가 다름)
    - L2 암호화를 재암호화하면 L1 암호화가 되도록 설계함
    - 두 단계 암호화가 제공되더라도 재암호화 키가 있으면 L2 형태로 암호화된 모든 암호문은 재암호화할 수 있음
      - 아주 세밀한 수준의 접근제어를 제공할 수 없음



- 시간제한(temporary): 일정한 기간동안만 재암호화가 가능해야 함

속성기반 암호화 (1/2)

- 속성기반 암호화(attributed-based encryption): 복호화키와 암호화된 데이터에 할당된 속성(예: 교수) 또는 접근 정책(예: 교수 or 조교)을 바탕으로 암호화된 데이터에 대한 접근 제어 메커니즘을 제공하여 주는 기술



- 다자간 통신, 클라우드 컴퓨팅의 공유 문제에 대한 해결책으로 사용 가능
- 신원기반 시스템을 확장한 개념 (신원 대신에 속성, 공개키 기반)
  - 키 발급 서버가 막강한 권한 (임계 기반 기법의 사용 필요)
  - 하나의 공개키에 여러 개의 개인키를 연결하여 사용하는 개념

속성기반 암호화 (2/2)

- 공개키 기반이므로 누구나 문서를 암호화할 수 있지만 권한이 있는 사용자만 복호화할 수 있음
- 이 기술을 사용하면 데이터를 서비스하여 주는 서버가 접근 제어를 판단하거나 검사하지 않고 데이터를 암호화하는 주체가 접근 제어를 결정할 수 있음
  - 저장 서버에 대한 신뢰 문제를 완화할 수 있음
  - 하지만 본질적으로는 속성기반 암호기술을 사용하도록 준비하여 주는 신뢰 서버가 필요하며, 이 서버는 막강한 권한과 능력을 가지고 있게 됨
- 공모 문제를 해결해야 함
  - 두 사용자가 각각은 접근할 수 없는 것이면 공모를 하더라도 접근을 할 수 없어야 함
- 철회 문제. 사용자로부터 또는 암호화된 문서로부터 속성을 철회할 수 있나?
  - 암호문의 속성 변경은 새로 만든 것이 더 효과적임
  - 사용자의 속성 철회는 키 변경이 필요함

속성기반 요구사항

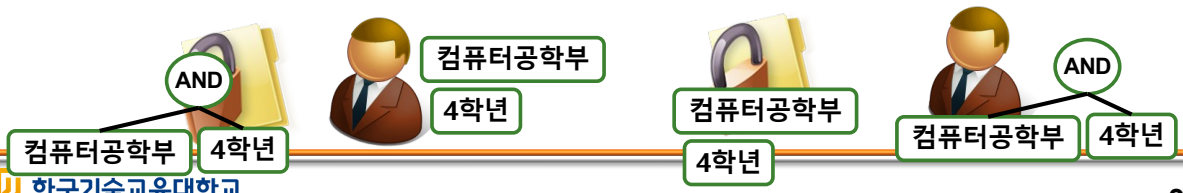
- 데이터 기밀성(data confidentiality): 권한이 없는 사용자(충분한 속성을 가지고 있지 않은 사용자)는 데이터를 얻을 수 없어야 함
- 공모에 대한 저항성(collusion resistance): 여러 사용자가 공모하더라도 각 개인이 얻을 수 있는 데이터밖에 얻을 수 없어야 함
  - Alice와 Bob이 각각 A, B 속성과 B, C 속성을 가지고 있더라도 A, C 속성이 필요한 데이터를 얻을 수 없어야 함
- 전방향 안전성(forward secrecy): 특정 속성이 철회된 사용자는 속성이 철회된 이후에는 해당 속성이 필요로 하는 데이터를 얻을 수 없어야 함
- 후방향 안전성(backward secrecy): 새 속성을 얻은 사용자가 현재 권한으로 접근할 수 있었던 이전 데이터를 얻을 수 없어야 함
  - 꼭 요구되는 사항은 아님
- 정책이나 속성에 대한 비밀 유지는 고려하지 않음

속성기반 암호기술의 표현성

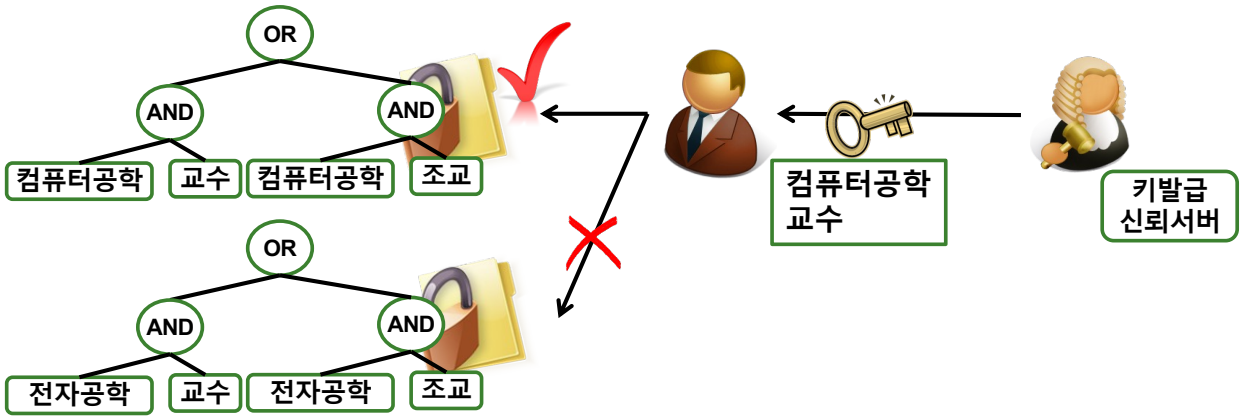
- 최초 ABE는 임계기반 접근 정책만 제공함
  - 일정 수 이상의 필요한 속성을 가지고 있으면 접근 가능
- 트리 기반 접근 구조 및 논리곱, 논리합, 임계기반 허용하도록 개선됨
- 이후 논리부정까지 사용할 수 있도록 개선됨
  - 효과적이지는 않지만 A라는 속성이 있을 때 (NOT A)라는 속성을 만들면 논리부정을 제공할 수 있음

속성 기반 암호화 종류

- 암호문 정책 속성기반 암호화(CP-ABE, Ciphertext-Policy ABE): 속성기반 암호의 접근 구조를 문서를 암호화할 때 결정
  - 각 암호문은 접근 구조(정책)와 연관됨
  - 각 복호화키는 속성들과 연관됨
    - 각 사용자는 속성들을 가지고 있으며, 이 속성들에 해당하는 키(또는 키들)를 받음
- 키 정책 속성기반 암호화(KP-ABE, Key-Policy ABE): 속성기반 암호의 접근 구조를 사용자 키를 생성할 때 결정
  - 각 암호문은 속성들과 연관됨
  - 각 복호화키는 접근 구조(정책)와 연관됨
- 속성 기반 암호화에서도 문서는 일반 대칭키로 암호화하고 대칭키를 속성 기반 키로 암호화하는 형태임

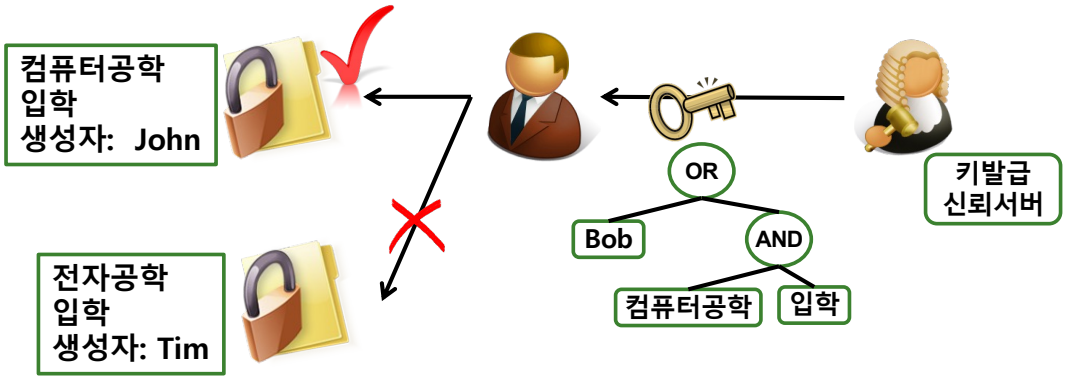


암호문 정책 속성기반 암호화



- 암호문
  - 암호문의 크기: 단말 수에 비례함
- 복호화할 수 있는 권한을 문서를 암호화할 때 결정할 수 있음
  - 특히, 논리 부정까지 지원되면 대단히 세밀하게 접근 제어를 제공할 수 있음

키 정책 속성 기반 암호화



- 암호문: 속성 집합으로 표현
  - 암호문의 크기가 보통 속성에 비례함
- 개인키: 속성에 대한 접근 트리로 표현
- 단점. 암호화된 속성 간 관계를 표현할 수 없음

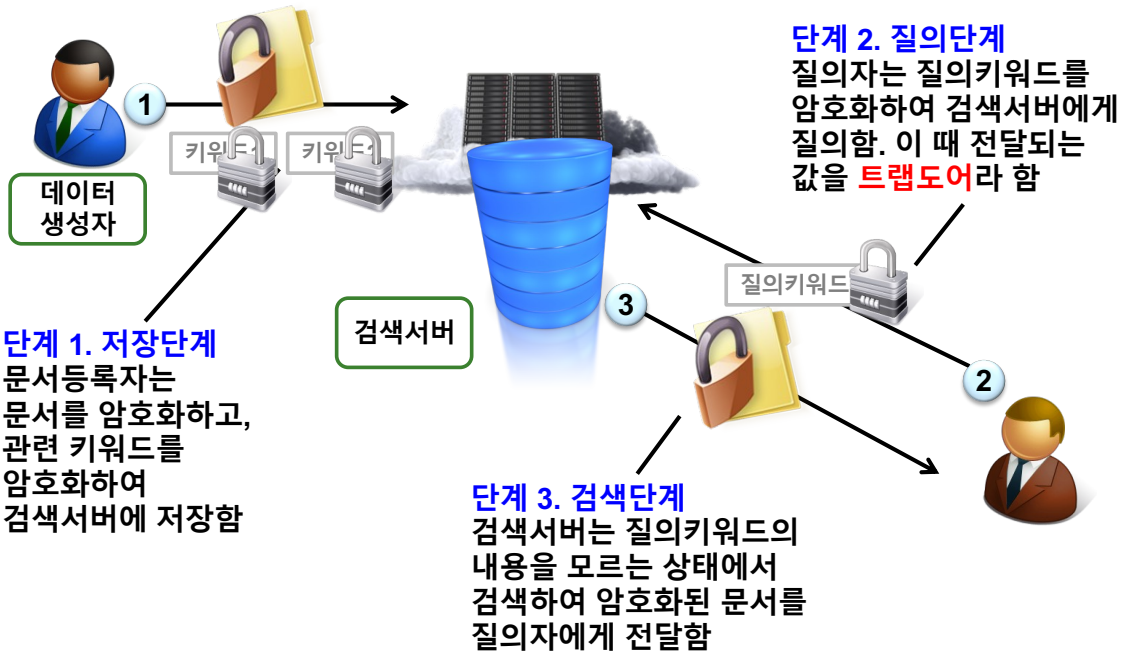
속성 기반 암호화에서 철회 문제

- 속성기반 암호화에는 두 종류의 철회가 있음
  - 암호문의 복호화 권한 변경
    - CP-ABE: 암호화된 문서의 접근 정책의 변경
    - KP-ABE: 암호화된 문서에 연관된 특정 속성을 철회하는 것을 말함
    - 연산을 적용하여 변경하는 것보다 다시 암호화하는 것이 효과적임
  - 사용자의 권한 변경
    - CP-ABE: 속성 추가 및 철회
    - KP-ABE: 접근 정책의 변경
    - 사용자의 모든 권한을 제거해야 할 수 있음
    - 기존에 가지고 있는 키를 사용할 수 없도록 해야 함
      - 암호화된 문서를 모두 갱신해야 함

검색가능 암호화

- 외부 서버에 기밀성/프라이버시 때문에 데이터를 암호화하여 유지할 수 있음
  - 이때 접근제어를 세밀하고 풍부하게 제공하는 문제의 해결도 필요하지만 데이터의 양이 많으면 효과적으로 데이터를 찾는 문제도 해결되어야 함
- 브라우징(폴더기반 정리)과 검색을 구분하여야 함
  - 다중 사용자 환경에서는 브라우징은 한계가 있음
- 데이터가 암호화되어 있으면 다양한 검색 기술을 직접 적용할 수 없음
  - 물론 색인 작업을 하는 검색 서버가 복호화키를 알고 있으면 기존 검색 기술을 그대로 적용할 수 있지만 이 경우 검색 서버를 신뢰하여야 함
- 검색가능 암호화(searchable encryption)는 암호화된 데이터의 기밀성과 프라이버시를 보호한 상태(검색 서버를 포함하여)로 효과적으로 검색할 수 있도록 해주는 기술임
- 검색 서버: 데이터 내용에 대해서는 궁금해하지만 다른 모든 서비스에 대해서는 신뢰 있게 해동한다고 가정함 (curious-but-honest model)

검색가능 암호시스템 사용시나리오



단계 1. 저장단계
문서등록자는 문서를 암호화하고, 관련 키워드를 암호화하여 검색서버에 저장함

단계 2. 질의단계
질의자는 질의키워드를 암호화하여 검색서버에게 질의함. 이 때 전달되는 값을 트랩도어라 함

단계 3. 검색단계
검색서버는 질의키워드의 내용을 모르는 상태에서 검색하여 암호화된 문서를 질의자에게 전달함

검색 방식

- 초기에 제안된 많은 기법은 단일 키워드를 이용한 질의만 가능함
- 키워드 유무(boolean search)에 의해 질의 만족 여부를 판별함
  - 이와 같은 만족 여부의 판별은 보통 순차 검색을 통해 이루어짐
  - 역색인을 사용하면 문서 기준 대신 키워드 암호문 기준으로 검색할 수 있음
  - 키워드의 정확성 필요
    - 트랩도어를 생성할 때 사용된 키워드와 문서에 연결된 키워드가 정확하게 일치하여야 함
- 여러 개의 트랩도어를 전달하여 다중 키워드 검색을 할 수 있지만 각 트랩도어를 이용하여 검색한 후에 각 결과를 종합하는 수준 밖에는 제공할 수 없음
- 키워드 기반 검색의 한계를 극복하기 위한 여러 기법이 새롭게 제안되고 있지만 여기서는 키워드 기반 검색 위주로 설명함

검색가능 암호시스템의 특징

- 데이터 암호키와 검색가능 암호시스템에서 사용하는 키는 별개임
- 대칭키 기반 기법도 있고, 공개키 기술을 이용하는 기법도 있음
  - 대칭키 기반은 키워드 암호문을 생성할 때 사용하는 비밀키와 트랩도어를 생성할 때 사용하는 비밀키가 같음
    - 생성자와 질의자의 키 공유가 필요함
  - 공개키 기반은 공개키로 키워드 암호문을 생성하고, 개인키를 이용하여 트랩도어를 생성함
- 지금까지 제안된 많은 기법에서 키워드 암호문은 확률적 암호화를 하지만 트랩도어는 결정적 암호화를 함
  - 동일 키워드에 대한 트랩도어가 변하지 않음
- 검색 서버는 트랩도어를 별도 인증하지는 않음
  - 유효하지 않은 트랩도어를 이용한 검색은 없는 키워드를 이용한 검색과 차이가 없음

검색가능 암호화 응용

- 전자우편
  - 전자우편서버로부터 전자우편 내용 보안
  - 검색가능 암호화를 통해 전자우편서버에 저장된 메일을 검색할 수 있음
- 데이터 아웃소싱: 예) 클라우드 스토리지
  - 클라우드 서버에 대한 신뢰를 줄이기 위해 데이터를 암호화하여 서버에 유지할 수 있음

검색가능 암호시스템 요구사항

- R1. 검색 서버는 암호화된 문서의 내용을 알 수 없어야 함
- R2. 검색 서버는 암호화된 키워드의 내용을 알 수 없어야 함
- R3. 검색 서버는 사용자가 전달한 질의 트랩도어의 내용을 알 수 없어야 함
- R4. 유효한 질의 트랩도어는 오직 권한을 가지고 있는 사용자만 생성할 수 있어야 함
  - 트랩도어를 생성할 때 사용하는 키를 통해 R4를 제공함

검색가능 암호시스템의 분류 (1/2)

- 저장하는 주체와 검색하는 주체에 따른 분류
  - SWSR(Single-Writer-Single-Reader) 모델. 특정 사용자만 홀로 데이터를 저장할 수 있고, 검색할 수 있는 모델
    - 필요성이 상대적으로 적음
  - SWMR(Single-Writer-Multi-Reader) 모델. 특정 사용자만 홀로 데이터를 저장할 수 있지만, 검색은 다수가 할 수 있는 모델
  - MWSR(Multi-Writer-Single-Reader) 모델. 여러 사용자가 데이터를 저장할 수 있지만, 오직 한 사용자만 검색할 수 있는 모델
    - 예) 전자우편 시스템
    - 공개키 방식은 기본적으로 이 모델을 제공할 수 있음
  - MWMR(Multi-Writer-Multi-Reader) 모델. 여러 사용자가 데이터를 저장할 수 있고 여러 사용자가 검색할 수 있는 모델

검색가능 암호시스템의 분류 (2/2)

- 다중 사용자가 참여하는 시스템의 경우에는 사용자의 가입과 탈퇴의 처리가 필요할 수 있음. 특히, SWMR과 MWMR 모델에서 필요함
  - SWMR 모델에서는 SW가 그룹 관리를 할 수 있으며,
    MWMR 모델에서는 별도의 그룹 서버를 사용할 수 있고,
    이 그룹 서버는 검색 서버와 달리 내부 서버를 활용할 수 있음
    - 기업이 데이터를 아웃소싱하지만, 그룹 관리는 내부적으로 함
  - 가입/탈퇴 때문에 키 변경이 필요하면 더욱 복잡해짐
- 이와 같은 분류 외에 검색기능에 따른 분류, 사용하는 암호기술에 따른 분류도 가능함

프라이버시는 어디까지 보장되나?

- 암호화된 문서의 내용을 알 수 없음
- 색인 정보의 노출
  - 저장된 키워드 암호문에 대한 키워드 추측 공격이 가능하지 않을 경우, 특정 문서와 연관된 키워드를 알 수 없음
  - 키워드 암호문은 보통 확률적 암호화 기법을 사용하기 때문에 키워드 암호문의 비교를 통해서는 문서 간의 연관 관계를 알 수 없지만, 이 부분은 질의를 통해 어차피 노출됨
- 질의 패턴을 포함한 검색 결과에 의한 정보 노출
  - 트랩도어에 대한 키워드 추측 공격이 가능하지 않을 경우, 트랩도어에 포함된 키워드를 알 수 없음
  - 특정 트랩도어와 연관된 문서의 수는 본질적으로 노출됨
  - 특정 키워드에 대한 트랩도어가 변하지 않으면 사용자의 질의 패턴이 노출됨

검색가능 암호화 미해결 문제점

- 트랩도어의 재사용성, 정보 노출
  - 지금까지 제안된 많은 기법의 경우 동일 키워드에 대한 트랩도어가 변하지 않음
  - 질의의 불연결성(같은 질의를 다시 한 것인지 여부)을 제공하지 못함
    - 이를 위해 안전한 채널로 전달하는 방법이 제안되었지만, 이 경우 여전히 검색 서버에 노출되는 문제점이 있음
- 암호화된 키워드와 암호화된 문서 간에 바인딩이 되어 있지 않음
  - 연관된 키워드 목록을 삭제/교체하여 검색을 방해할 수 있음
  - 트랩도어를 구성할 때 문서를 사용할 수 없기 때문에 실제 바인딩하기 힘듦