# 다자간 키 확립 프로토콜

NOTE 12



한국기술교육대학교 컴퓨터공학부 김상진

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

## 교육목표

- 개요
  - 다자간 키 확립 프로토콜의 요구사항
- 중앙집중형: 단일 중앙서버가 키를 분배하는 방식
  - Statefull: LKH, OFT
  - Stateless: CS, SD
- 탈중앙형: 키를 분배하는 서버를 여러 개 사용하는 방식
  - Iolus
- 분산형
  - TGDH
- 기타: Bresson 등의 기법
- 수신제한시스템(CAS, Conditional Access System)
- 메신저: 그룹 채팅

# 개요

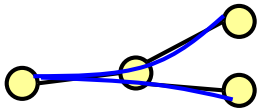
unicast vs. multicast vs. broadcast

- VOD 서비스, 화상회의처럼 그룹에 속한 멤버들에게만 비밀스럽게 데이터를 전송해야 하는 경우가 있음
- 사용자 쌍마다 다른 비밀키를 사용하여 암호화하여 전달하는 것은 네트워크 대역폭 측면과 연산량 측면에서 비효율적임
  - 각 사용자에게 동일 메시지를 보내고 싶음
  - 네트워크 대역폭을 효과적으로 사용하고 싶음

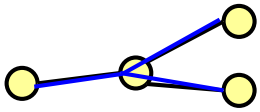
$A \rightarrow B$: $\{M\}.K_{AB}$
$A \rightarrow C$: $\{M\}.K_{AC}$
$A \rightarrow D$: $\{M\}.K_{AD}$

$A \rightarrow B$: $\{M\}.K$
$A \rightarrow C$: $\{M\}.K$
$A \rightarrow D$: $\{M\}.K$

- 해결책
  - 그룹에 속한 모든 멤버들이 같은 비밀키를 공유하도록 한 다음에 이 비밀키로 데이터를 암호화하고, IP 멀티캐스트(multicast) 기법을 사용하여 전달함
  - IP 멀티캐스트 기법은 가장 적은 네트워크 대역폭을 사용하여 동시에 여러 수신자에게 패킷을 전달하여 줌



2개의 unicast



1개의 multicast



# 다자간 키 확립 프로토콜의 요구사항

- 요구사항에 영향을 주는 요소
  - 응용의 종류 (예: 실시간 방송, 다자간 회의, 메신저 단톡 등)
  - 그룹의 크기와 동적성 (그룹 멤버의 변화)
  - 확장성
  - 신뢰 모델
- 그룹키를 변경하기 위해 모든 참여자가 프로토콜에 참여해야 하면 확장성 문제 때문에 현실성이 떨어질 수 있음
- 그룹키 확립도 키 확립 프로토콜의 한 종류이므로 그룹키 확립 프로토콜도 키 확립 프로토콜의 요구사항을 모두 충족해야 하지만, 현실적으로 힘듦
  - 예) 키의 비밀성: 그룹 멤버들만 알 수 있어야 함
  - 예) 키의 확인: 모든 그룹 멤버가 동일한 키를 가졌는지 확인하는 것은 현실성이 없음.
  - 예) 키의 최근성: 난스 기법을 사용하기 힘듦

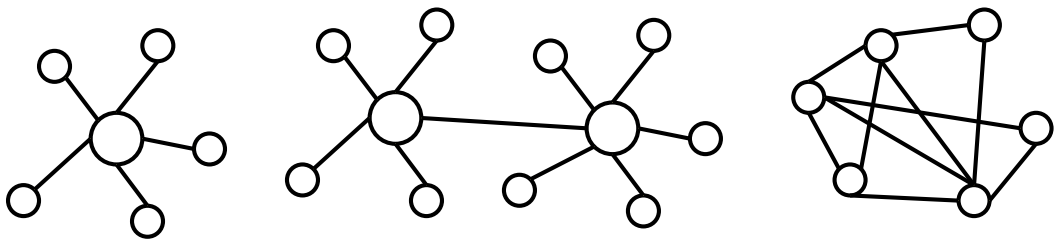
# 다자간 키 확립 프로토콜의 요구사항

- 동적 그룹(그룹 멤버가 빈번하게 변하는 그룹)의 경우 다음을 추가로 요구할 수 있음
  - 전방향 안전성(forward secrecy): 그룹을 탈퇴한 멤버를 포함하여 이전 그룹키를 알고 있는 공격자는 새 그룹키를 알 수 없어야 함
    - 참고. 기존 키 확립 프로토콜에서 정의한 전방향 안전성 내용과 일치하지 않음
  - 후방향 안전성(backward secrecy): 그룹에 새롭게 가입한 멤버를 포함하여 현재 그룹키를 알고 있는 공격자는 이전 그룹키를 알 수 없어야 함
- 키 독립성(key independence): 몇 개의 그룹키를 알고 있는 공격자는 이 키들을 제외한 다른 그룹키들을 알 수 없어야 함
  - 키 독립성은 전방향, 후방향 안전성을 포함한 개념임
- 전방향, 후방향 안전성을 보장하기 위해서는 그룹 멤버가 변할 때 그룹키를 다시 확립해야 함 (join, leave 프로토콜 필요)
- 응용에 따라 전방향 안전성의 보장이 더 중요할 수 있음



# 다자간 키 확립 프로토콜의 종류

- 중앙집중형(centralized): 단일 키 분배 서버를 사용하는 방식
- 탈중앙형(decentralized): 전체 그룹이 여러 개의 소그룹으로 분리되어 있으며, 각 소그룹의 키 분배 서버를 사용함
- 분산형(distributed): 키 분배 서버를 전혀 사용하지 않는 방식임



- 중앙집중, 탈중앙, 분산을 비교할 때 자주 등장하는 그림
- 이 그림이 잘못되었다고 지적되는 경우가 많음
- 가장 왼쪽 네트워크는 중앙집중이 확실함
- 중간 형태는 분산이지만 진정한 탈중앙은 아니고, 가장 오른쪽은 분산이고 탈중앙이라고 주장하기도 함

| | 중앙집중 | 탈중앙 | 분산 |
|---|---|---|---|
| 서버 수 (architectural) | 1 | 독립적으로 동작 가능한 여러 개 서버 | 공통 목표를 위해 여러 개의 동등 노드가 상호작용함<br>사용자는 단일 시스템으로 인식함 |
| 의사결정 (political) | 단일 주체가 단독으로 결정 | 각 노드가 독자적으로 결정 | 처리는 여러 노드에 나누어 진행되지만 의사결정은 중앙집중일 수 있음 |
| 논리적 측면 (logical) | | 분할되어도 계속 동작 가능 | |
| 데이터베이스 | 단일 DB를 단일 물리적 위치에 유지 | 다중 독립 DB를 여러 물리적 위치에 유지 | 논리적으로는 단일 DB: 단일 DB를 물리적으로 여러 곳에 나누어 유지 |
| 공격 저항성 | bad (단일 실패점) | good | |
| 아키텍처 | 클라이언트-서버 | P2P | |
| 성능 문제 | 수직 확장 | 수평 확장 | |

혼란에 핵심은 3개를 비교하기 때문
비교는 중앙집중 vs. 탈중앙, 중앙집중 vs. 분산을 비교해야 올바른 비교
중앙집중 vs. 탈중앙의 핵심은 단일 제어 vs. 다중 제어
중앙집중 vs. 분산의 핵심은 한 곳 vs. 여러 곳
예) 구글 서버: 통제는 중앙집중이지만 서버는 분산되어 있음
예) 비트코인: 노드들이 분산되어 있으며, 한 노드가 통제를 하지 않으므로 탈중앙임

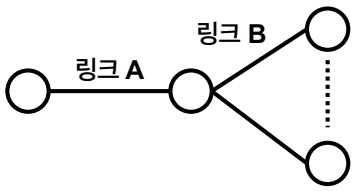


# Stateless VS. Stateful

- 상태 기반(stateful): 모든 키 갱신 세션에 빠짐없이 참여할 경우에만 최신 그룹키를 계산할 수 있는 방식
- 비상태 기반(stateless): 현재 세션에 전달된 키 갱신 메시지와 초기 상태에 대한 정보만 있으면 과거 여러 세션 동안 키 갱신 메시지를 수신하지 못하였어도 최신 그룹키를 계산할 수 있는 방식
  - 실시간 유료 방송 서비스의 경우는 비상태 기반 방식이 필요할 수 있음
    - 참여자의 기기가 항상 on 상태는 아닐 수 있음

# 여기서 잠깐

- 다중 유니캐스트 vs. 단일 멀티캐스트
  - 예) $n$개의 노드에 크기($m$)가 동일한 서로 다른 데이터의 유니캐스트 vs. $nm$ 크기의 단일 멀티캐스트
    - 단점
      - 각 노드는 크기가 $m$인 메지지 대신 $nm$ 크기의 메시지 수신
      - 대여폭(↑)




다중 유니캐스트
링크 A: $nm$
링크 B: $m$

단일 멀티캐스트
링크 A: $nm$
링크 B: $nm$

  - 예) 동일 메시지를 $n$개 노드의 전달: 당연히 멀티캐스트가 윈

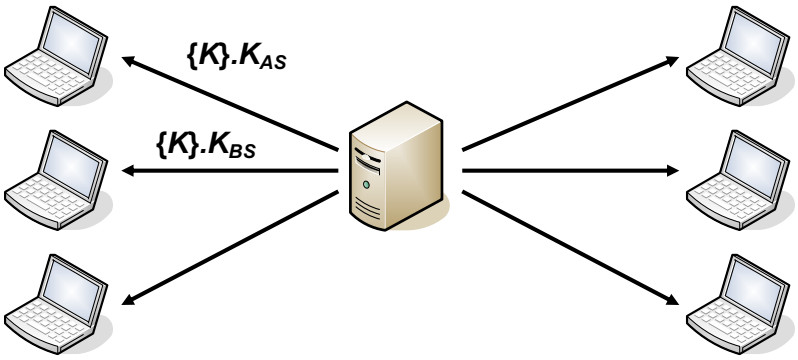
다중 유니캐스트
링크 A: $nm$
링크 B: $m$

단일 멀티캐스트
링크 A: $m$
링크 B: $m$



# 중앙집중형

- 단순 접근 방법 1.
  - 중앙 서버는 각 멤버와 장기간 비밀키를 공유하고 있음
  - 설정 프로토콜
    - 중앙 서버는 각 멤버에게 그룹키를 그 멤버와 공유한 비밀키로 암호화하여 전달함
  - 가입 프로토콜과 탈퇴 프로토콜
    - 설정 프로토콜을 다시 수행함

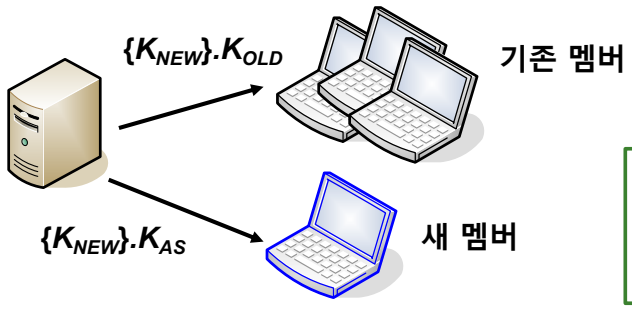



설정/가입/탈퇴 비용.
통신비용: $O(n)$ 유니캐스트
서버의 계산비용: $O(n)$
$n$은 멤버 수

# 중앙집중형

개별 사용자의 비용이 아니라 서버 비용을
확장성 있게 만드는 것이 목표
물론 사용자 비용이 확장성이 없으면 문제가 됨

- 단순 접근 방법 2.
  - 중앙 서버는 각 멤버와 장기간 비밀키를 공유하고 있음
  - 설정 프로토콜: 단순 접근 방법 1과 동일
  - 가입 프로토콜
    - 기존 그룹키로 새 키를 암호화하여 기존 멤버들에게 제공함
    - 새롭게 가입한 사용자에게는 그 멤버와 공유한 비밀키로 암호화하여 전달함
  - 탈퇴 프로토콜: 설정 프로토콜과 동일



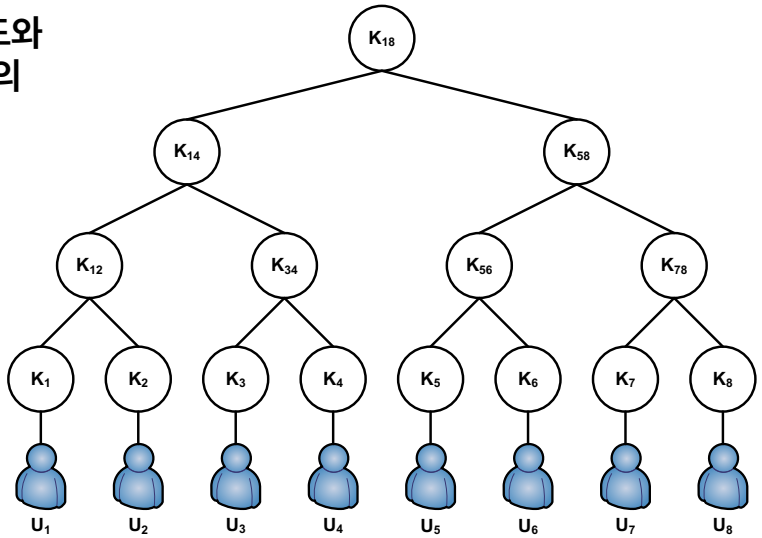

설정/탈퇴 비용.
통신비용: $O(n)$ 유니캐스트
서버의 계산비용: $O(n)$

가입 비용: $O(1)$ (이 이상 좋아지기 힘듦)
통신비용:
$O(1)$ 유니캐스트 + $O(1)$ 멀티캐스트
서버의 계산비용: $O(1)$



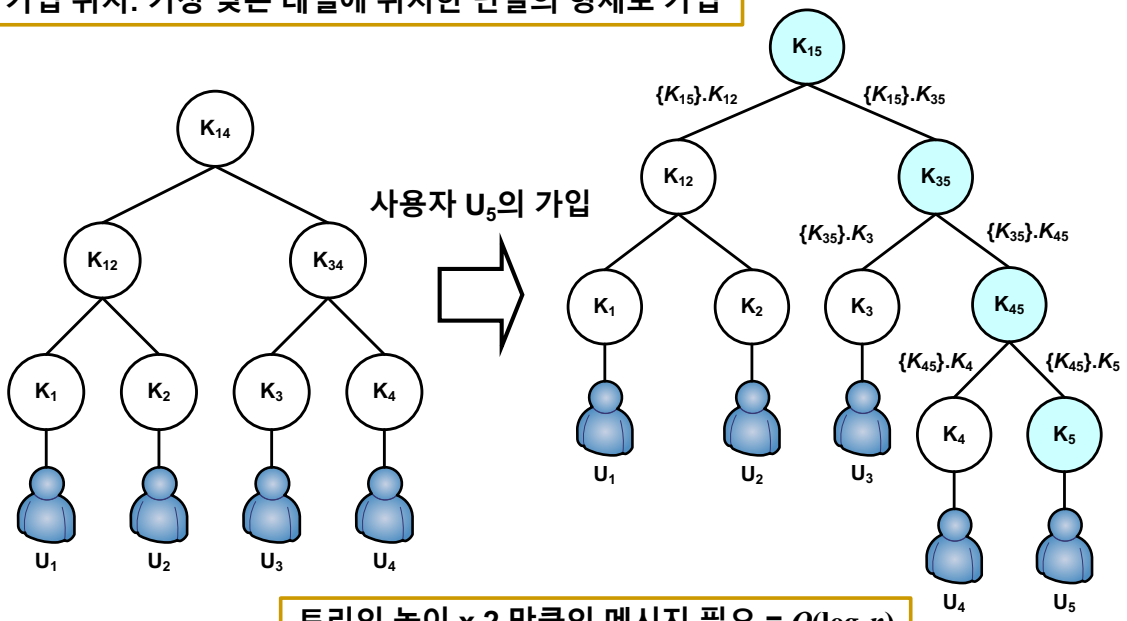
# LKH(Logical Key Hierachry), 1998

- Wong 등, Wallner 등이 제안한 상태 기반 프로토콜
- 중앙 서버는 다음과 같은 이진 트리를 내부적으로 유지함
- 노드마다 하나의 키가 할당됨
- 트리의 루트 노드에 할당된 키가 그룹키가 됨
- 각 사용자는 하나의 단말 노드와 연관되며, 그 노드와 그 노드의 조상 노드에 있는 모든 키를 유지해야 함
- 사용자 1은 $K_1$, $K_{12}$, $K_{14}$, $K_{18}$을 유지해야 함
- 사용자가 유지하는 키의 개수는 $O(\log n)$임
- 중간 노드 키를 부분 그룹키로 활용 가능



# LKH – 가입 프로토콜

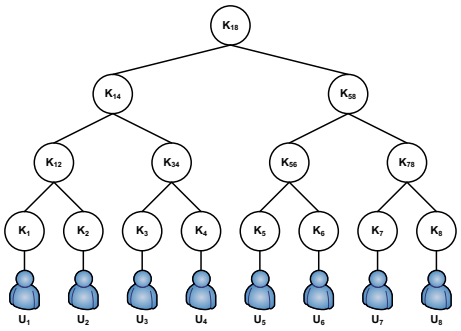
노드 가입 위치: 가장 낮은 레벨에 위치한 단말의 형제로 가입




트리의 높이 x 2 만큼의 메시지 필요 = $O(\log_2 n)$

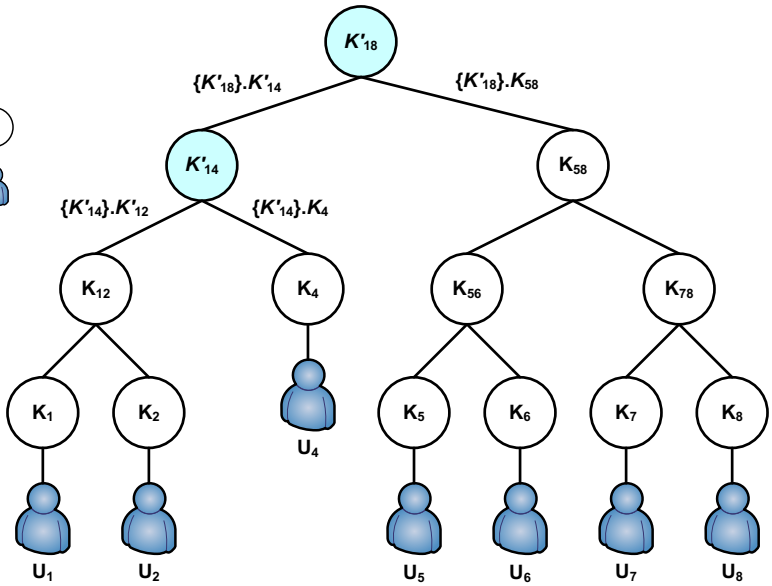


# LKH – 탈퇴 프로토콜



사용자 $U_3$의 탈퇴

트리의 높이 x 2 만큼의 메시지 필요 = $O(\log_2 n)$

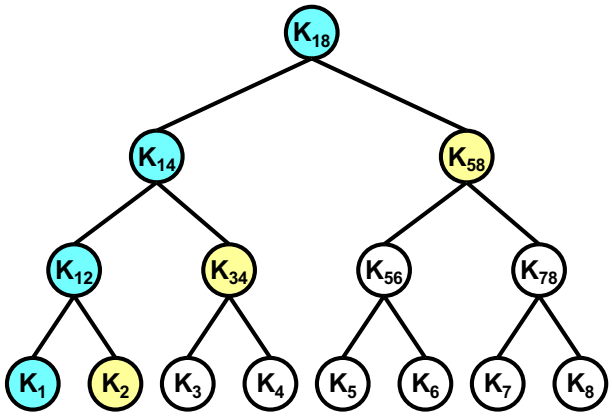


# OFT(One-way Function Tree), 1998

- McGrew과 Sherman이 제안
  - Merkle 해시 트리
- $g(K_i)$를 $K_i$의 은닉키(blind key)라 함
- 각 사용자는 자신의 형제 집합에 대한 은닉키를 유지함
- 예) 사용자 $U_1$: $K_1$, $g(K_2)$, $g(K_{34})$, $g(K_{58})$을 유지함

$$K_i = f\left(g\left(K_{left(i)}\right)||g\left(K_{right(i)}\right)\right)$$

예) $K_{14} = f(g(K_{12})||g(K_{34}))$

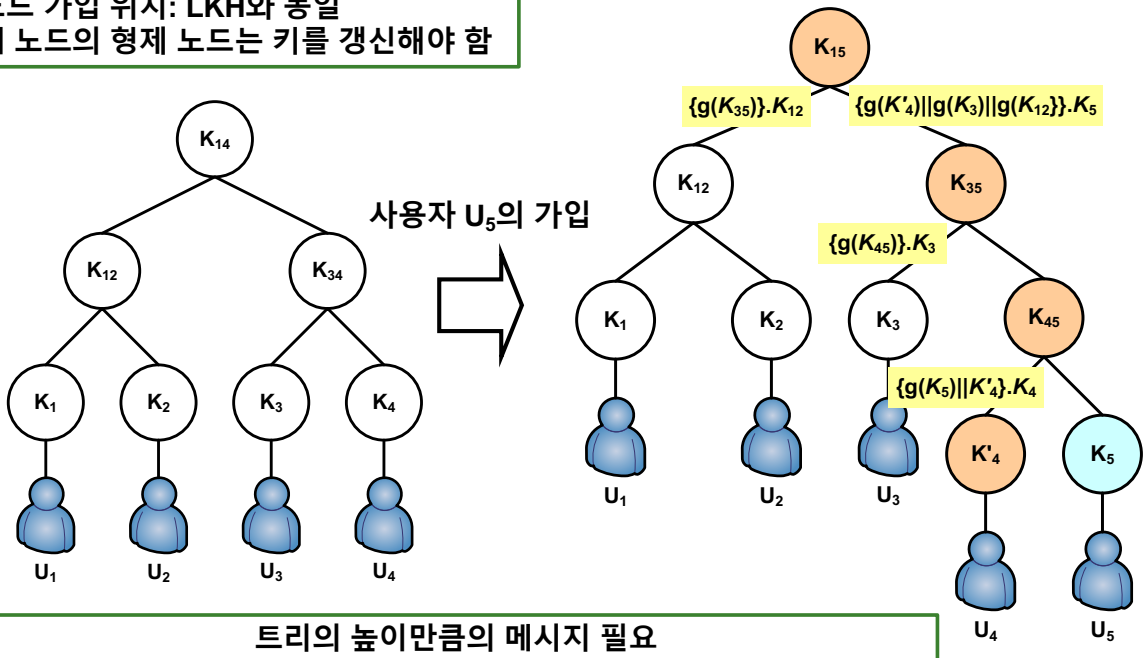



key path: 단말 노드를 기준으로 해당 노드를 포함한 조상 노드의 집합
co path: 단말 노드를 기준으로 루트 노드를 제외한 key path에 있는 노드의 형제 노드



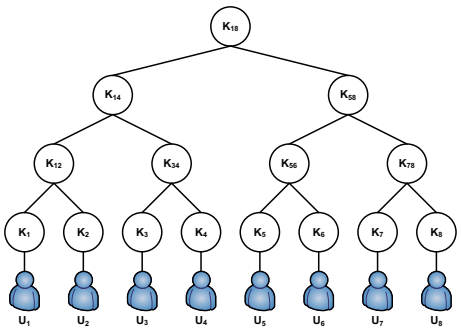
# OFT – 가입 프로토콜

노드 가입 위치: LKH와 동일
새 노드의 형제 노드는 키를 갱신해야 함

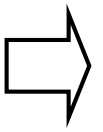



트리의 높이만큼의 메시지 필요
기존 사용자는 중간노드 키를 계산하기 위한 절반 정보를 이미 알고 있음

# OFT – 탈퇴 프로토콜

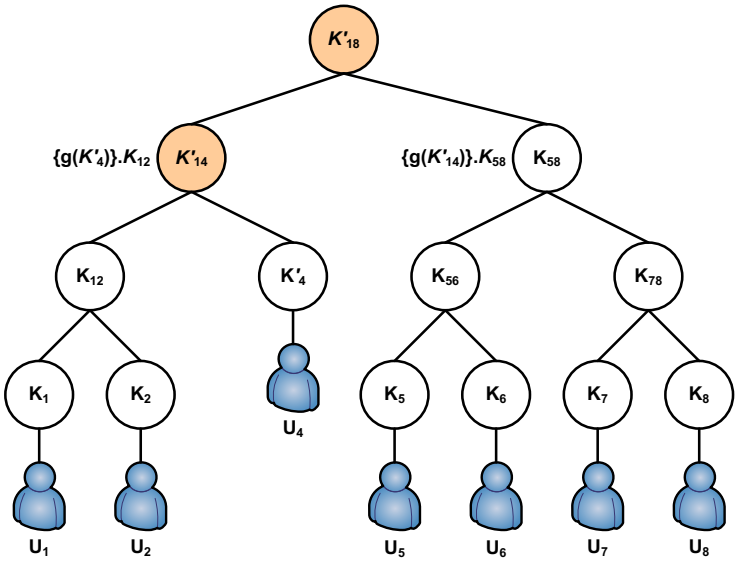


사용자 $U_3$의 탈퇴



탈퇴 노드의 형제 노드는 키를 갱신해야 함

트리의 높이만큼의 메시지 필요

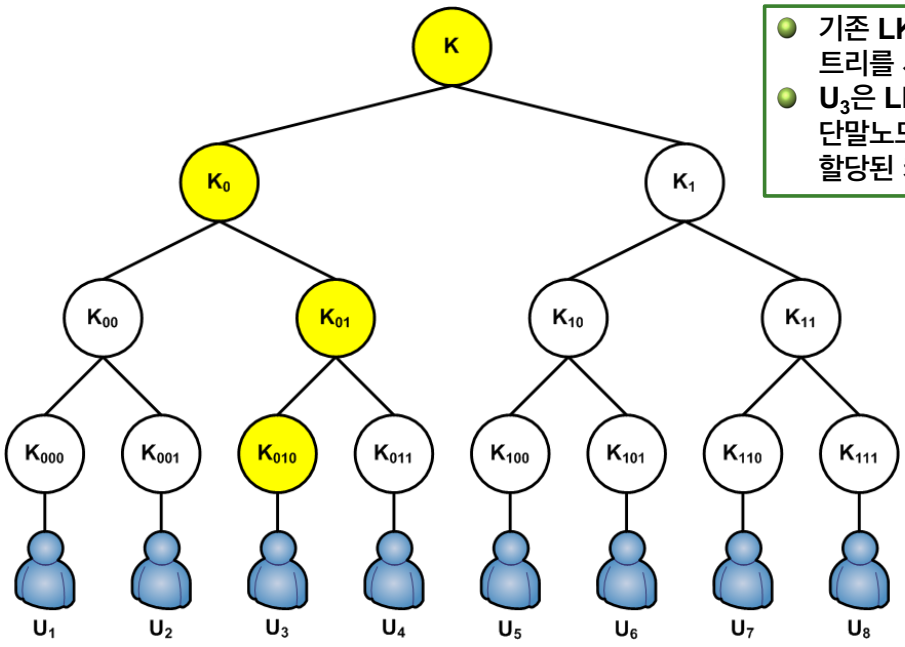





# Naor 등의 broadcast encryption 기법

- LKH, OFT 기법과 동일하게 논리적 키 계층 구조를 사용하지만 루트 노드에 할당된 값이 그룹키가 아님
  - 주목적. 전체 멤버 중 일부를 제외하고 비밀 그룹 통신을 하고 싶음
    - 변경된 그룹키를 효과적으로 전달하기 위한 구조
    - 논리 키 트리에 할당된 키는 갱신되지 않고 계속 사용하는 구조
  - 비상태 기반
    - 새 그룹키를 암호화하여 전송하는 키는 바뀌지 않음
- 서버 역할: 키 분배 뿐만 아니라 데이터를 멀티캐스팅하는 역할까지 수행
  - 비밀 그룹 통신 절차. ① 사용자 ⟹ 서버, ② 서버 ⟹ 대상 부분그룹
  - LKH, OFT에서 서버는 그룹키 분배 역할만 수행
- 두 종류의 기법 제시
  - 완전 부분 트리(CS, Complete Subtree) 기법
  - 부분 트리 차이(SD, Subtree Difference) 기법

# 완전 부분 트리 기법 (1/2)

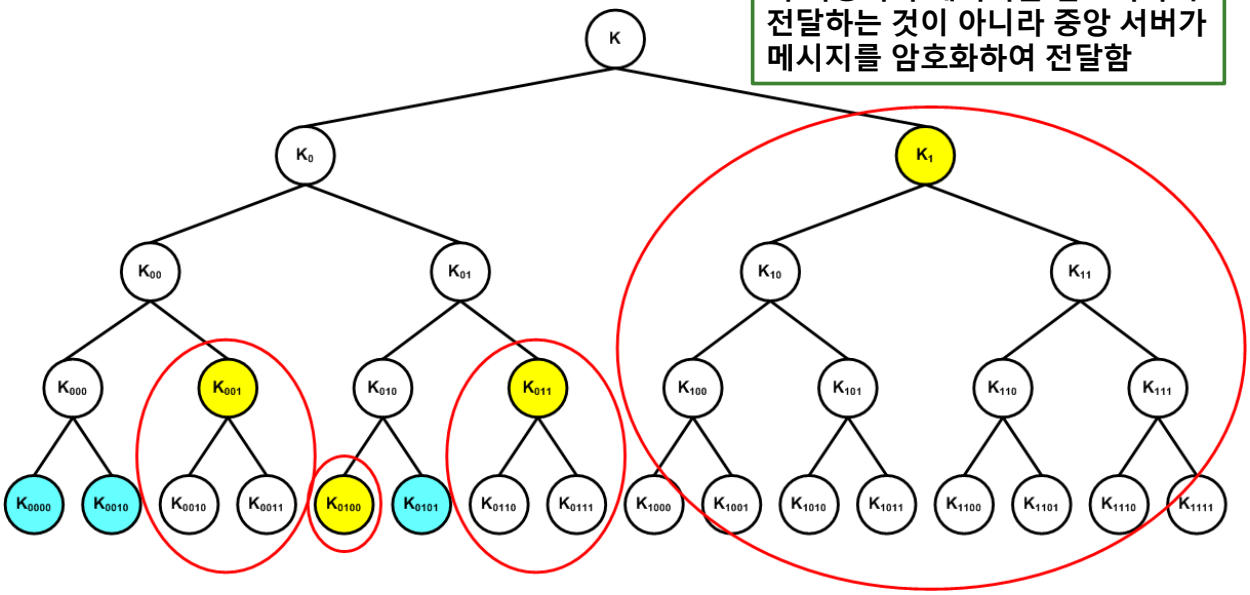



- 기존 LKH와 동일한 트리를 사용함
- $U_3$은 LKH와 동일하게 단말노드부터 루트노드까지의 할당된 키 값을 유지



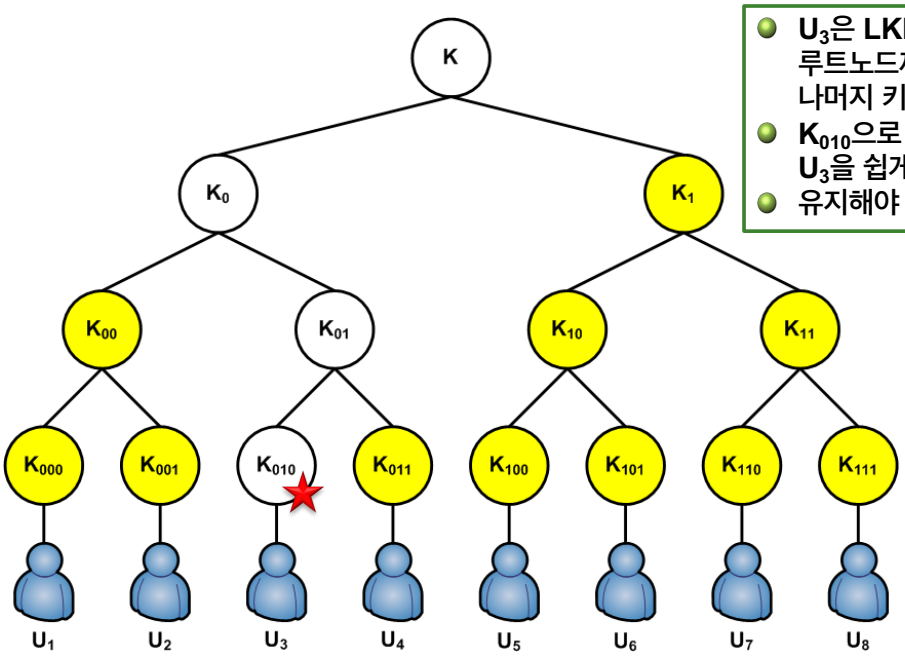
# 완전 부분 트리 기법 (2/2)

각 사용자가 메시지를 암호화하여 전달하는 것이 아니라 중앙 서버가 메시지를 암호화하여 전달함




$U_1$, $U_2$, $U_6$가 수신하지 못해야 하는 경우 $K_{001}$, $K_{0100}$, $K_{011}$, $K_1$으로 새 그룹키를 암호화하여 전달 → 즉, 가장 완전한 부분 트리의 루트 키만을 사용하여 새 그룹키를 암호화하여 전달

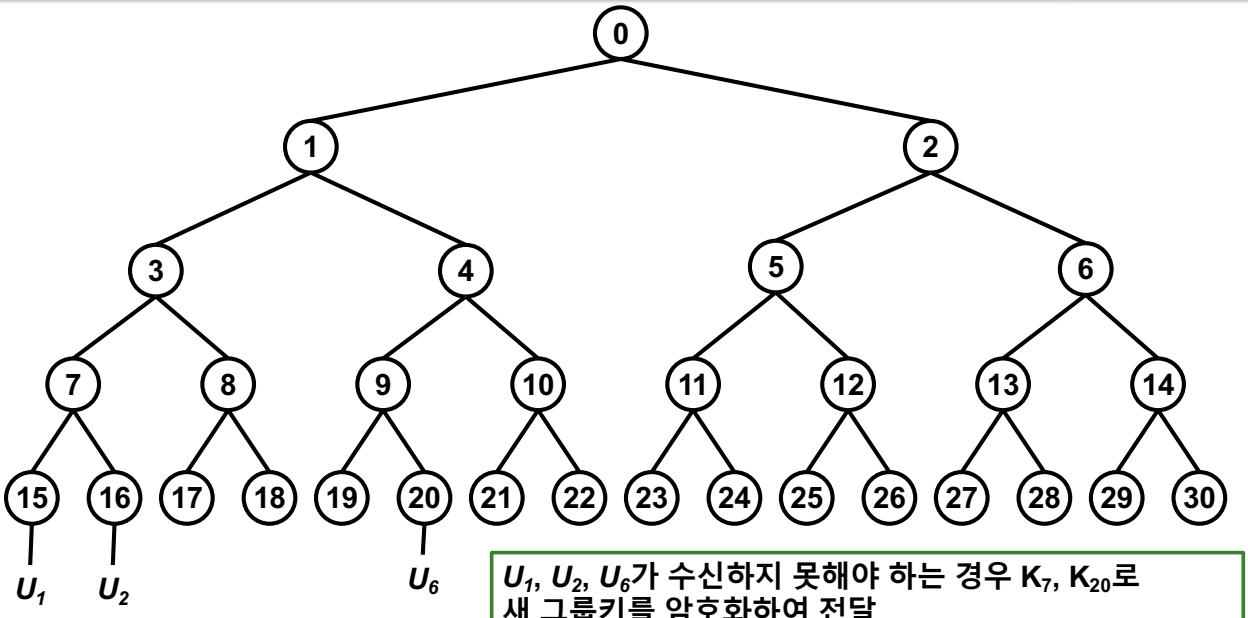
# 부분 트리 차이 기법 (1/4)




- $U_3$은 LKH와 정반대로 단말노드부터 루트노드까지 할당된 키를 제외한 나머지 키를 유지
- $K_{010}$으로 그룹키를 암호화하면 $U_3$을 쉽게 배제할 수 있음
- 유지해야 하는 키가 너무 많음



# 부분 트리 차이 기법 (2/4)

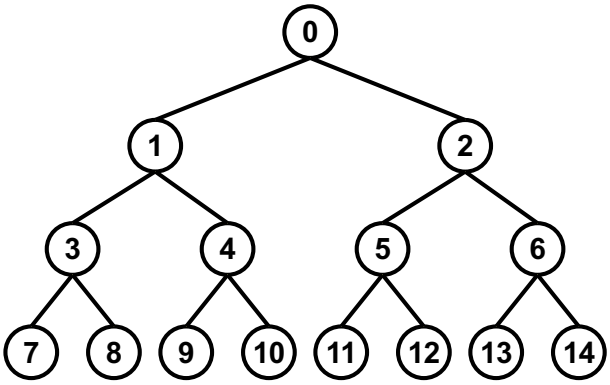



$U_1$, $U_2$, $U_6$가 수신하지 못해야 하는 경우 $K_7$, $K_{20}$로 새 그룹키를 암호화하여 전달

문제점. 각 노드가 유지해야 하는 값이 많음
문제점. $U_1$과 $U_2$는 $K_{20}$를 알고 있고, $U_2$는 $K_7$를 알고 있음

# 부분 트리 차이 기법 (3/4)

- 부분집합 개념 이용: $S_{i,j} = G_i - G_j, G_i \supset G_j$
- 각 사용자는 자신이 소속된 모든 부분집합의 키를 가지고 있어야 함
- 예) $U_1$
  - (0,2), (0,5), …, (0,14) ⟹ (0,2)
  - (0,4), …, (0, 10) ⟹ (0,4)
  - (0,8)
  - (1,4), …, (1,10) ⟹ (1,4)
  - (1,8)
  - (3,8)

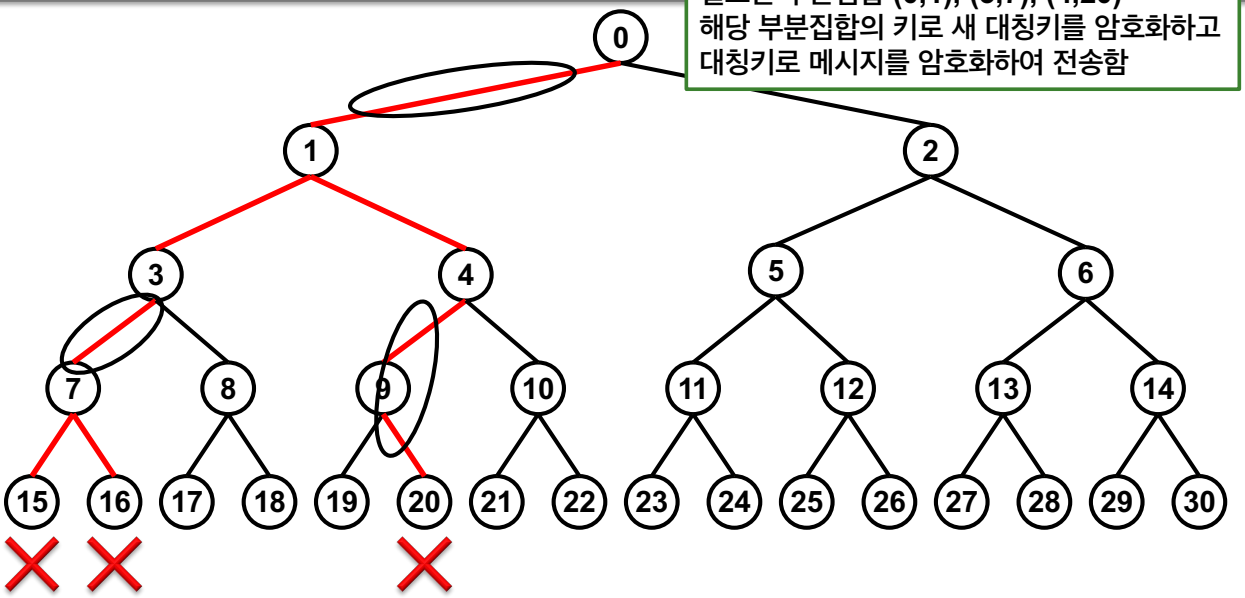



- 각 노드에 랜덤 $S_i$가 할당됨
- $S_{i,j}$ 키는 $S_i$에 할당된 키로부터 모두 계산할 수 있음
  - $S_{0,1} = h(S_0), S_{0,2} = g(S_0), S_{0,3} = h(S_{0,1}), S_{0,4} = g(S_{0,1}), \ldots$
- 각 사용자는 약 $(\log n)^2$개의 키만 유지하고, 필요할 때 나머지는 계산함



# 부분 트리 차이 기법 (4/4)

필요한 부분집합 (0,1), (3,7), (4,20)
해당 부분집합의 키로 새 대칭키를 암호화하고
대칭키로 메시지를 암호화하여 전송함



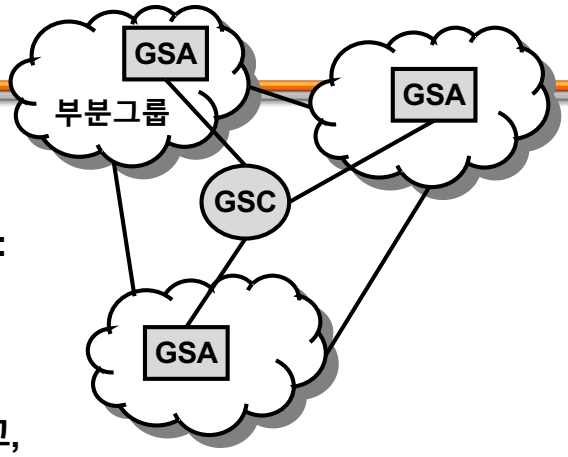

빨간색에 해당하는 트리를 구함 (루트부터 제외할 노드만 포함하는 트리, **Steiner tree**)
트리에서 **maximal chain**을 구함
**chain**: 트리에 있는 경로로 경로에 있는 마지막 노드를 제외하고 노드의 자식이 하나만 있는 경우
$U_1$, $U_2$, $U_{20}$은 (0,1), (3,7), (4,20)을 모두 자신들이 가지고 있는 키를 이용하여 계산할 수 없음
$U_1$: (0,2), (0,4), (0,8), (0, 16), (1,4), (1, 8), (1, 16), (3, 8), (3, 16)
$U_{20}$: (0,2), (0,3), (0,10), (0,19), (1, 3), (1, 10), (1, 19), (4, 10), (4, 19)

# 탈중앙형

- 전체 그룹 멤버가 여러 개의 부분 그룹으로 나누어지며, 부분 그룹마다 그 그룹을 담당하는 서버를 두는 방식임
- 추가적인 요구사항
  - 지역키 독립성: 부분그룹의 멤버 변화는 전체에 영향을 주지 않아야 함
  - 그룹 간 독립성: 한 멤버는 여러 부분 그룹에 동시에 가입할 수 없어야 함
  - 키와 데이터 간 관계: 키 관리 경로와 데이터 전달 경로가 독립적이어야 함
    - 부분 그룹의 멤버 변화가 데이터 전달을 방해하거나 지연시키지 않아야 함



# Iolus, 1997




- Mittra가 제안
- 구성요소
  - GSC(Group Security Controller): 전체 그룹을 관리하는 서버
  - GSA(Group Security Agent): 각 부분 그룹을 관리하는 서버
- GSA들은 GSC를 통해 그룹키를 형성하고, 각 사용자는 자신이 속한 부분 그룹의 GSA를 통해 부분 그룹키를 형성함
- 한 사용자가 전체 그룹 멤버들에게 메시지를 전송하는 과정
  - 부분 그룹키로 멀티캐스트함
  - 사용자가 속한 그룹의 GSA는 다른 GSA에게 멀티캐스트함
  - 수신한 GSA는 부분 그룹키를 이용하여 그룹 멤버들에게 전송함

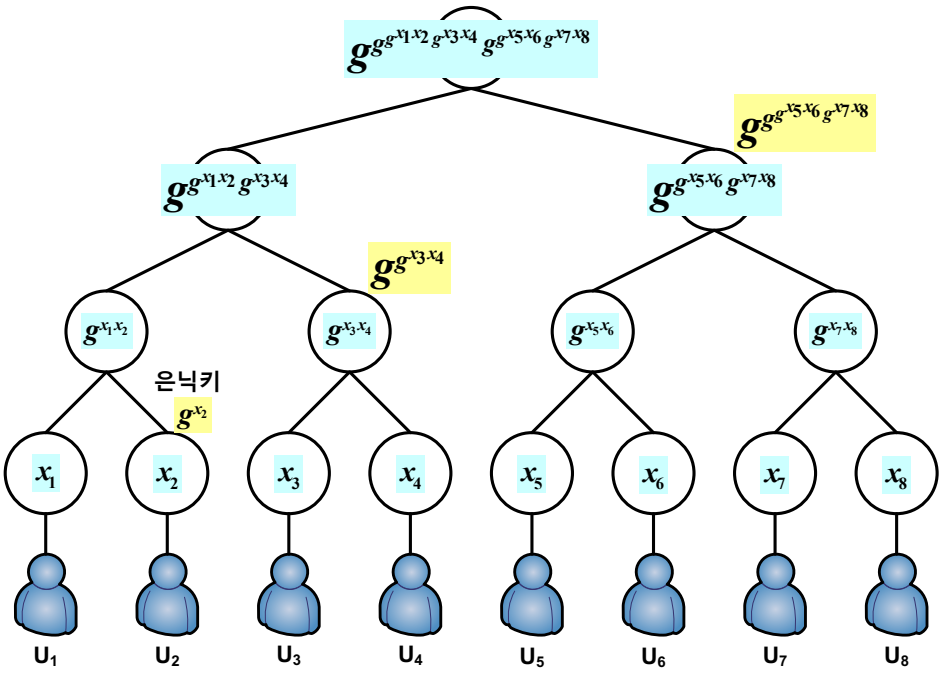
# Iolus – 가입/탈퇴 프로토콜

- 가입 프로토콜
  - 각 사용자는 가입하고 싶은 부분 그룹 관리자 GSA에게 가입을 요청함
  - 가입이 승인된 사용자는 GSA와 비밀키를 공유함
  - 해당 그룹에서 사용하는 그룹키 프로토콜을 통해 그룹키를 갱신함
- 탈퇴 프로토콜
  - 해당 그룹에서 사용하는 그룹키 프로토콜을 통해 그룹키를 갱신함

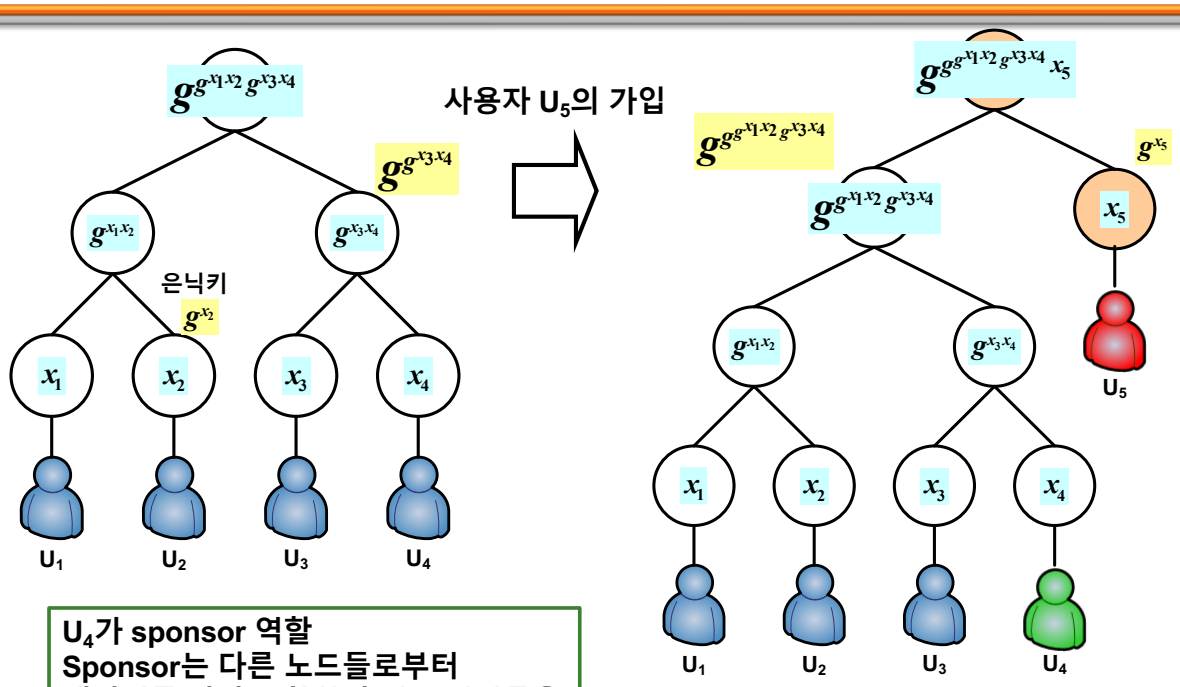


# Perrig의 기법, 1997

- 분산형 LKH



# Kim 등의 확장, 2000 (1/3)




U4가 sponsor 역할
Sponsor는 다른 노드들로부터 데이터를 받아 종합하여 새 은닉키들을 멤버들에게 멀티캐스트함

OFT와 달리 루트에 가입할 경우 가입 노드의 형제 노드 키 값을 변경할 필요가 없음

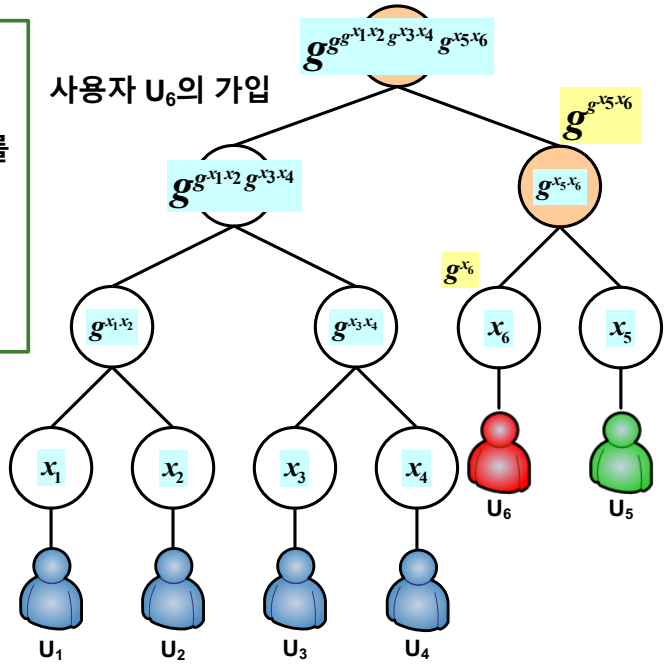


# Kim 등의 확장 (2/3)

가입기준.
가장 오른쪽 노드 중 트리의 높이를 증가시키지 않는 노드에 삽입
현재 트리가 완벽 이진 트리이면 루트를 기준으로 가입이 이루어짐

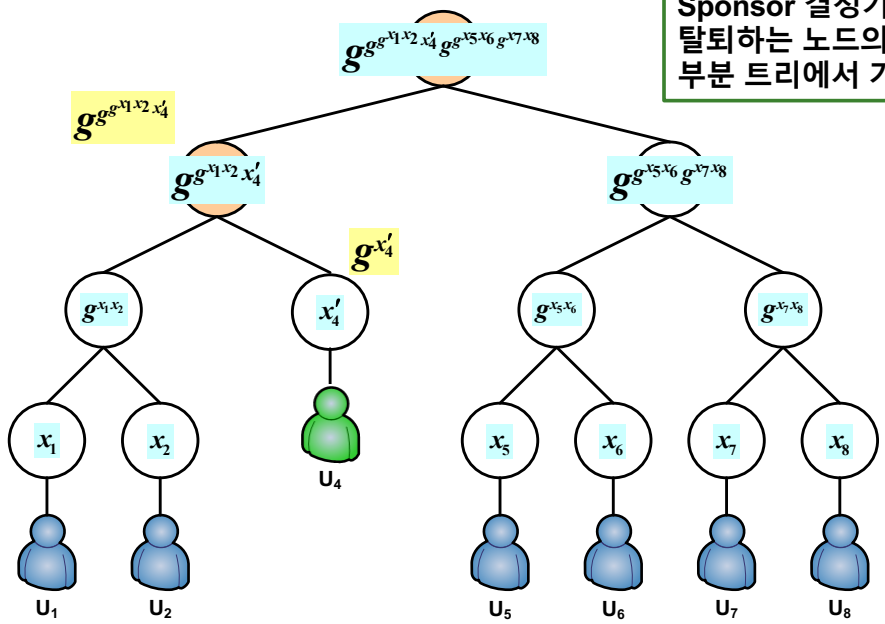
Sponsor 결정기준.
삽입되는 노드가 포함된 부분 트리의 가장 오른쪽 노드

1. 새 가입자는 Sponsor에게 자신의 은닉키를 전달함
2. Sponsor는 노드들을 갱신한 다음에 새 은닉키들을 나머지 사용자들에게 멀티캐스트함



# Kim 등의 확장 (3/3)

Sponsor 결정기준.
탈퇴하는 노드의 부모를 기준으로 하는 부분 트리에서 가장 오른쪽 노드



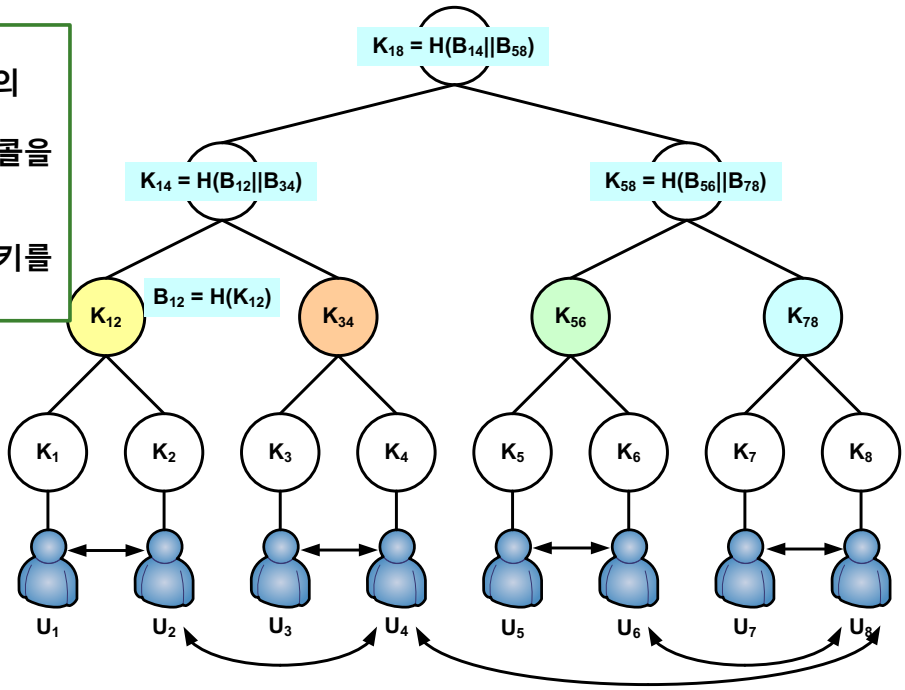



# EGAKA, 2004

- Ren 등이 제안(ICU)

라운드 1.
- ↔ 만큼의 2자간 키 동의 프로토콜을 수행함
- 특정한 키 동의 프로토콜을 사용할 필요가 없음
- 라운드 2.
- 확립된 키로 블라인드 키를 서로 교환

공헌.
꼭 사용자 간의 동일한 프로토콜을 사용할 필요는 없음
계속 루트까지 DH 키 동의 프로토콜을 수행할 필요는 없음



# 기타 프로토콜

- Secure Lock 프로토콜
- Bresson 등의 프로토콜



# Secure Lock 프로토콜, 1987 (1/2)

- Chiou와 Chen이 제안한 중앙집중방식
- 중국인 나머지 정리를 이용하여 하나의 값을 멀티캐스트하여 그룹키를 설정함

정리. (중국인의 나머지 정리) $n_1, \ldots, n_k$은 양의 정수라 하자. 단, $1 \le i \ne j \le k$에 대해 $\gcd(n_i, n_j) = 1$이다. 이 때 연립합동식

$$x \equiv a_1 \pmod{n_1}$$
$$x \equiv a_2 \pmod{n_2}$$
$$\ldots$$
$$x \equiv a_k \pmod{n_k}$$

은 $M = n_1 \ldots n_k$을 법으로 유일한 해를 갖는다.

$$\begin{aligned} x &\equiv 1 \pmod 5 \\ x &\equiv 2 \pmod 6 \\ x &\equiv 3 \pmod 7 \end{aligned} \qquad 206 \pmod{210}$$

# Secure Lock (2/2)

- 각 사용자 $i$는 GKM과 장기간 비밀키 $K_i$를 공유하고 있으며, 사용자가 가입하게 되면 각 사용자에게 정수 $n_i$가 할당됨
  - 각 사용자에게 할당된 정수 $n_i$는 모두 서로소이어야 함
- 설정
  - 단계 1. GKM은 그룹키 $K$를 임의로 선택함
  - 단계 2. 중국인 나머지 정리를 이용하여 아래의 식의 해를 구함.

$$M \equiv \{K\}.K_1 \pmod{n_1}$$
$$M \equiv \{K\}.K_2 \pmod{n_2}$$
$$\dots$$
$$M \equiv \{K\}.K_m \pmod{n_m}$$

그 다음에 $M$을 모든 사용자에게 멀티캐스트함

  - 단계 3. 각 사용자는 $\{K\}.K_1 = M \bmod n_1$을 구한 다음 복호화하여 그룹키 $K$를 확보함

비가입자들도 $n_i$와 $M$을 알 수 있으므로 $\{K\}.K_i$를 계산할 수 있다. 하지만 $K_i$를 모르기 때문에 $K$를 얻을 수 없음

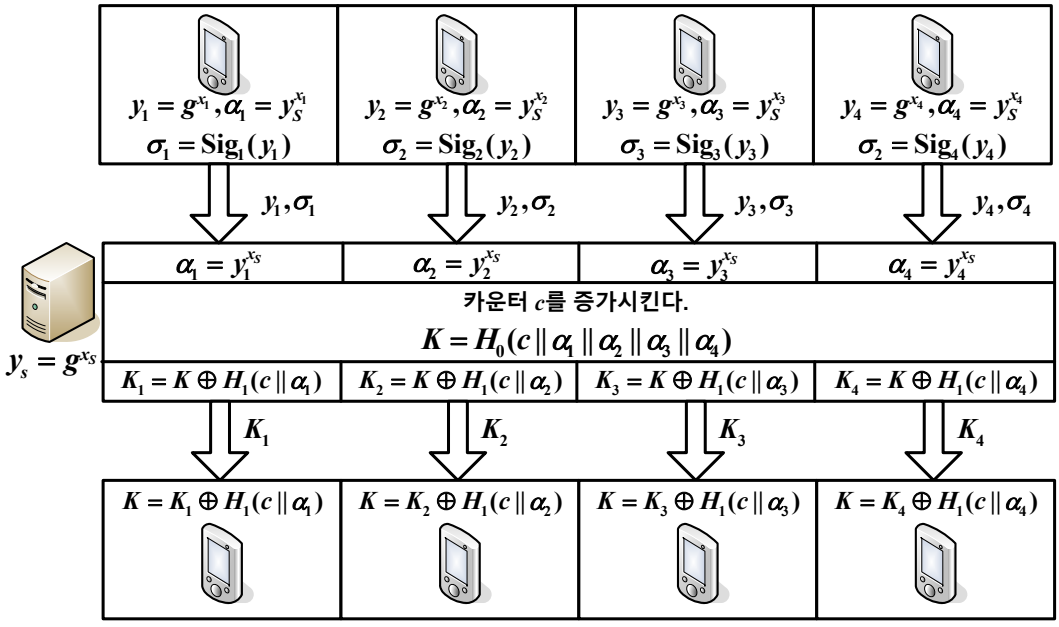
중국인 나머지 정리는 안전성을 높이기 위한 수단이 아니고, 하나의 값을 전달하여 여러 사용자가 그 값으로부터 같은 그룹키를 얻을 수 있게 하는 방법으로 활용된 것임



# Bresson 등의 기법, 2003

- 저전력 환경 고려

초기 설정 비용은 높아도 됨
가정은 온라인 상태에서 수행하는 것이 아님



# CAS(Conditional Access System) (1/6)

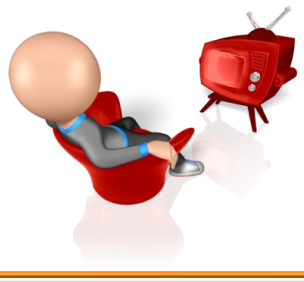
- 유료 방송 환경에서는 가입자만 해당 방송을 볼 수 있어야 함
- CAS는 이 요구사항을 충족하기 위해 사용하는 정보보호 기술임
  - CAS는 기존 방송 통신 환경이 일방향 통신 채널(방송국 ⟹ 가입자)만을 사용할 수 있다는 제한점을 고려하여 개발된 기술임
  - CAS는 통신 채널에 대한 보호만 제공함
    - 디스크램블된 이후 데이터에 대한 보호는 제공하지 못함

- 기본 생각: 방송 콘텐츠를 스크램블(암호화)하여 전달하고, 오직 합법적인 가입자만 이것을 디스크램블(복호화)할 수 있도록 함
  - Scrambling/Descrambling: 대칭키를 이용한 암/복호화로 생각하면 됨
  - 가입자가 다수이므로 그룹키 개념이 필요함
  - 가입자에게 적절한 키가 포함된 스마트카드를 발급하며, 사용자는 스마트카드를 자신의 셋톱박스에 설치함
    - 지금은 설치된 소프트웨어에 내장되어 사용하고 있음



# CAS (2/6)

- 유료방송과 그룹키
  - 가입/탈퇴 시 그룹키 갱신이 필요함
    - 매번 그룹키를 갱신하는 것은 현실적으로 어려움
    - 후방향을 고려하지 않는다고 하여도 전방향은 반드시 제공해야 하므로 탈퇴시 갱신은 필수적임
      - 이 경우에도 탈퇴가 발생한 시점에 즉시 갱신하지 않는 것은 충분히 양보할 수 있음
  - 가입자는 항상 자신의 기기를 켜 놓고 있지 않으므로 갱신 메시지를 항상 수신할 수 있다고 보장할 수 없음
    - 상태기반보다는 비상태기반이 필요함
  - 모든 채널의 콘텐츠가 항상 전달되고 있는 방식임
- 각 업체는 CAS의 자세한 내부 메커니즘을 공개하지 않음



# CAS (3/6)

- CAS는 세 종류의 키를 사용함 (3계층 키)
  - 사용자별 마스터 개인키(MPK, Master Private Key): 공개키 방식
    - 사용자의 스마트카드에 저장되어 있거나 소프트웨어에 안전하게 내장되어 있음 (why 공개키?)
  - AK(Authorization Key): 보통 채널별 대칭키이며, 갱신 주기는 몇 주임
  - CW(Control Word): 콘텐츠를 스크램블/디스크램블할 때 사용하는 대칭키이며, 갱신 주기는 5-20초임 (48비트, 60비트)
    - 이렇게 자주 갱신하는 이유?
      - Maybe 노출되었을 때 파급효과 축소
      - Maybe 어차피 계속 전달하여야 함 (가입자가 언제 요청할지 모름)
    - 자주 갱신하기 때문에 굳이 긴 길이의 키를 사용할 이유가 없음

3계층: $\{AK\}.+K_1, \ldots, \{AK\}.+K_n, \{CW\}.AK$

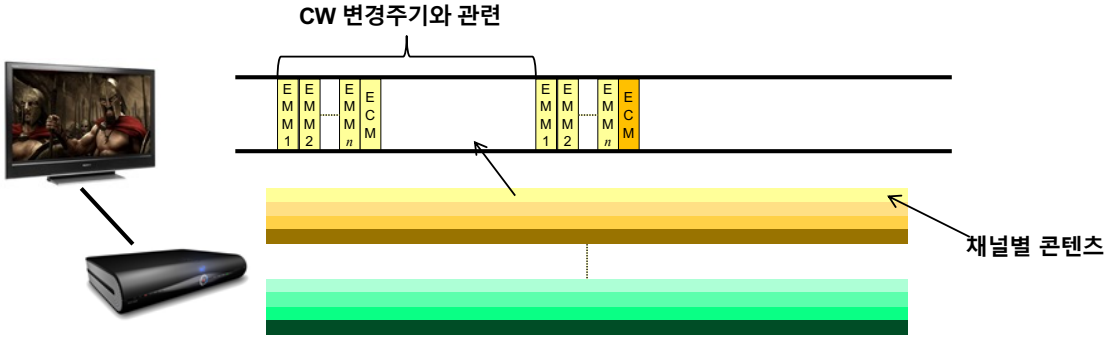
2계층: $\{CW\}.+K_1, \ldots, \{CW\}.+K_n$

- 3계층은 짧은 주기마다 1개 암호문만 갱신
- 일부 제거되어야 하는 암호문은 있음 (굳이)
- 2계층은 짧은 주기마다 $n$개 암호문의 갱신 필요



# CAS (4/6)

- 각 콘텐츠마다 AK를 각 사용자별 마스터키로 암호화한 EMM(Entitlement Management Message)이 포함됨
  - 사용자 수에 비례함 (확장성이 없음)
  - 변하는 빈도가 빈번하지 않음
- 각 콘텐츠마다 CW를 AK로 암호화환 ECM(Entitlement Control Message)이 하나 포함됨
  - 매우 빈번하게 바뀜



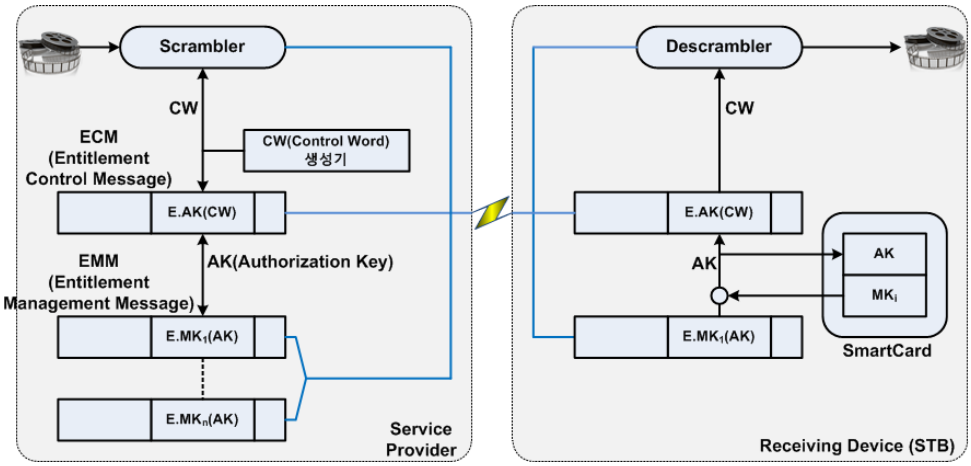
# CAS (5/6)

- 최소 2계층 키가 필요함
  - 사용자별 접근제어를 위해 사용자별 키 필요
  - 각 콘텐츠를 암호화하기 위한 키 필요
    - 2계층 키만 사용하면 채널(콘텐츠마다)당 사용자 수만큼의 콘텐츠 키를 암호화한 암호문 필요
    - 콘텐츠 키를 변경(자주 변경)할 때마다 사용자 수만큼의 새 암호문이 필요
- 1계층을 추가하면 얻을 수 있는 효과
  - 사용자에게 중간키를 사용자별 키로 암호화하여 이미 전달한 경우에는 콘텐츠 키를 변경할 때마다 하나의 새 암호문만 필요
    - 2계층: $n$개 암호문 $\Longrightarrow$ 3계층: $n+1$개 암호문
    - 새 주기마다 필요한 새 암호문: 2계층은 $n$개, 3계층은 1개 필요
  - 서비스 제공자 측면에서 노력이 감소함
  - $n+1$개 암호문을 계속 전달하기 때문에 정보량 측면에서 감소 효과는 없음



# CAS (6/6)

- 방송국에서 각 가입자에게 방송 콘텐츠를 전달할 때 아래와 같은 형태로 전달함



# CAS의 변형

- 하드웨어 고정 CAS를 사용하는 대신에 소프트웨어 모듈로 CAS를 구현함
  - 성능에 영향을 줄 수 있지만 유연성을 높일 수 있음
  - 필요에 따라 CAS를 다운받아 설치하여 사용할 수 있음
    - DCAS(Downloadable CAS)
    - 다운받은 소프트웨어를 인증하는 방법이 필요함
  - 기존 방식에서 키는 별도 스마트 카드에 유지하였지만 DCAS는 소프트웨어 자체에 키를 유지함
    - 난독화 기술 사용하여 보호
- XCAS(eXchangeable CAS), ICAS(Interchangble CAS)도 유사한 개념

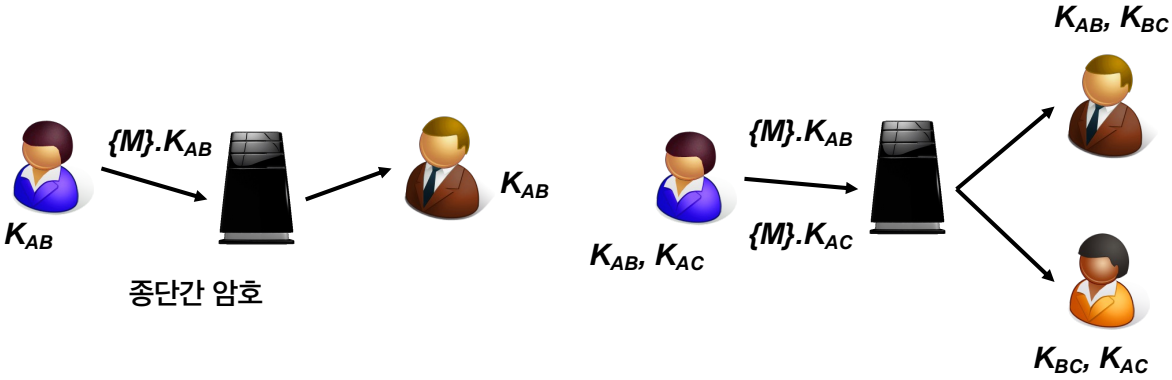


# CAS의 문제점

- CAS는 양방향 채널을 사용할 수 없는 환경을 위해 개발된 기법
  - 현재 IPTV 및 미래 방송 환경을 고려할 때 양방향 채널 활용이 충분히 가능함
    - 양방향 채널이면 기기를 부팅할 때 필요한 키를 수신하여 사용 가능
      - 모든 콘텐츠에 EMM을 포함할 필요 없음
- 참고로 VOD(멀티캐스트가 불필요)는 CAS 대신 DRM 기술이 적용됨

# 메신저 그룹 채팅 보안 (1/3)

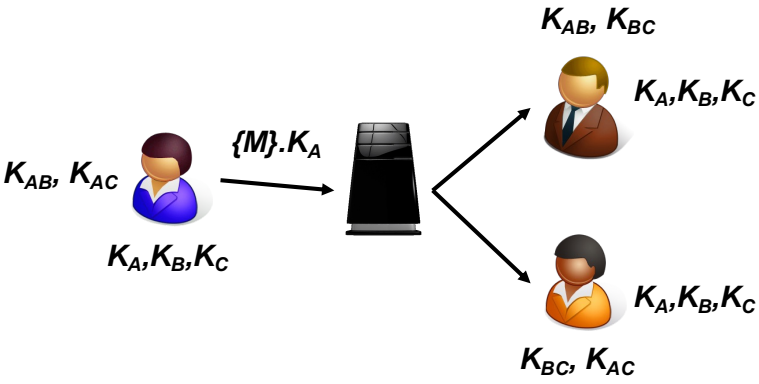
- 메신저와 SMS의 차이: 그룹 채팅 지원 여부
- 메신저와 일반 채팅 프로그램의 차이: 모든 멤버가 온라인 상태가 아닐 수 있음
- 시그널
  - 2자 간을 기계적으로 다자 간으로 확장
    - 송신자는 그룹 수에 비례한 암호문을 생성하여야 함
      - 유지하는 키의 개수도 그룹 크기에 비례함






# 메신저 그룹 채팅 보안 (2/3)

- Whatsapp
  - 모든 사용자는 그룹에서 사용할 그룹키를 각각 생성하고 나머지 그룹 멤버들에게 이 키를 2자간 시그널 프로토콜을 이용하여 전달함
    - 각 사용자는 $n$개의 그룹키 유지
  - 송신자는 하나의 암호문만을 서버를 통해  전달하여 그룹 통신을 할 수 있음
    - 일반 메시지와 차이가 없음



# 메신저 그룹 채팅 보안 (3/3)

- LKH나 OFT를 활용할 수는 없나?
  - 서버가 그룹키 관리하는 것은 목적에 어긋남
  - 그룹을 만든 멤버가 관리
    - 나머지 멤버에게 키 분배를 해야 함
    - 멤버의 추가와 탈퇴가 이루어지면 갱신되어야 함
    - 그룹 관리자가 탈퇴하면?
  - 참고. 각 멤버가 그룹키 분배 서버 역할을 하면 각 멤버마다 논리 트리를 유지해야 하기 때문에 Whatsapp 방식보다 더 비효율적임