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

키 확립 프로토콜 개요

NOTE 06

한국기술교육대학교 컴퓨터공학부 김상진

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

교육목표

- n명 간의 비밀 통신 문제
  - 대칭 암호알고리즘만 사용하는 기법
    - Blom의 기법
  - 공개키 암호알고리즘을 사용하는 기법
- 키 확립 프로토콜 요구사항 및 분류
  - 분류. 키 전송, 키 동의
  - Diffie-Hellman 키 동의 프로토콜

개요

- 근본 문제: n명의 사용자가 서로 메시지를 비밀스럽게 교환하고 싶음
  - 사용자 쌍마다 메시지를 암/복호화할 수 있는 서로 다른 대칭키가 필요함
  - 대칭키는 장기적으로 사용하면 안전성이 약해지기 때문에 키를 바꿀 수 있어야 함
- 해결책의 분류: 사용하는 암호기술에 따른 분류
  - 분류 1. 대칭키 방식만을 사용하는 방법
  - 분류 2. 공개키를 활용하는 방법

비밀키만 사용하는 방식

- 방법 1. TTP가 각 사용자에게 $n-1$개의 대칭키를 분배함
  - 확장성의 문제가 있으며, 키가 변하지 않고, 사용자 추가는?
- 방법 2. 중앙 온라인 TTP 사용
  - 각 사용자는 공통된 온라인 TTP와 장기간(long-term) 대칭키를 공유함
    - 각 사용자는 오직 하나의 키만 유지함
  - 각 사용자 간에는 암호프로토콜(TTP 참여)을 통해 필요할 때마다 새 일회용 대칭키(세션키)를 확립하여 사용함
    - 매번 다른 키를 사용함
- 방법 3. 이 방법을 해결하기 위한 특수한 방법을 사용함
  - 예) Blom의 기법, Blundo 등의 기법 등
    - TTP가 각 사용자에게 다른 $n-1$명의 사용자와 독특한 대칭키를 공유하기 위해 필요한 정보를 사전에 분배함
    - 이 정보는 n에 비례하지 않음 (확장성과 안전성 간의 상반관계)
    - 방법 1과 마찬가지로 키가 변하지 않음

대칭키 방식만 사용하는 해결책 비교

	방법1	방법2	방법3
서버 역할	사전 분배	초기키 발급 세션키 분배	사전 분배
사용자쌍키	고정 (고정키로 세션키를 교환할 수는 있음)	매번 새 세션키 확립 통신 오버헤드 발생	고정 (고정키로 세션키를 교환할 수는 있음)
사용자 저장공간 비용	$n-1$개의 대칭키	1개의 장기간키	$n-1$보다 적은 정보 유지가능 저장공간 비용이 적을수록 안전성이 약함

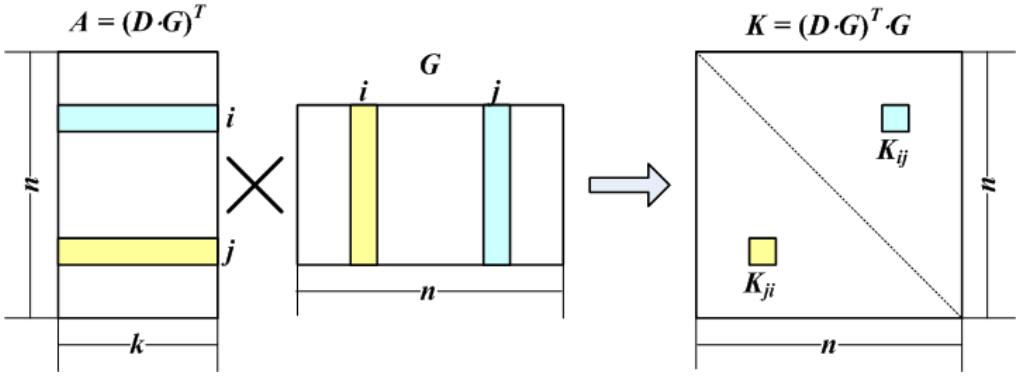
키 확립 프로토콜

장기간 키는 사전에 분배되어 있어야 함

Blom의 키 분배 기법 (1/2)

- 공모의 안전성 기준이 되는 k를 결정함
  - k명 이상 공모하면 다른 사용자 간에 공유된 키를 계산할 수 있음
    - k는 보안 변수 (조절할 수 있는 값을 말함)
- 시스템 설정
  - TTP는 $k \times n$ 크기의 유한체 F_q의 원소로 구성된 행렬 G를 생성하여 공개함
    - G의 행렬의 임의의 k열은 선형독립이어야 함
  - $k \times k$ 크기의 랜덤 대칭 행렬 D를 생성함. 이 행렬 값은 비밀로 유지해야 함
  - TTP는 각 사용자 i에게 $(DG)^T$의 i번째 행만 분배함
    - 각 사용자가 유지해야 하는 정보는 $O(k)$임

Blom의 키 분배 기법 (2/2)



- 행렬 $K = (DG)^T G$의 각 항이 각 사용자 간에 공유할 대칭키가 됨
  - i번째 사용자와 j번째 사용자는 행렬 K의 행 i, 열 j에 있는 값 $K[i][j]$을 대칭키로 사용함
  - 이 행렬은 대칭 행렬이므로 $K[i][j] = K[j][i]$가 성립함
- 사용자 i는 사용자가 j와 공유할 대칭키를 계산하고 싶으면 G의 j 열과 내적을 구하여 $K[i][j]$를 계산함

$$K = (DG)^T G = G^T D^T G = G^T DG = K^T$$

공개키를 사용하는 방식 (1/2)

- 간단한 방법
  - 각 사용자는 자신의 공개키 쌍만 유지함
  - 다른 사용자와 비밀 통신이 필요할 때 랜덤 대칭키를 생성하여 메시지를 암호화하고 이 키를 상대방의 공개키로 암호화하여 전달함
    - TLS 2.0에서 많이 사용하는 방법

K_i는 K로부터 계산된 서로 독립된 키

  - 요구가 발생할 때마다 새로운 대칭키를 생성하여 사용함
  - 요구되는 연산 비용이 상대적으로 높으며, PKI가 잘 구축(?)되어 있어야 사용할 수 있음
- 비밀키만 사용하는 방식 2에서 계정 개설, 초기 키 분배, 키 갱신할 때에만 공개키를 사용할 수 있음

공개키를 사용하는 방식 (2/2)

- 키 확립 프로토콜에서 공개키 암호알고리즘 사용에 따른 장점
  - 장점 1. 인증 서비스를 상대적으로 쉽게 제공할 수 있음
  - 장점 2. 대칭 암호알고리즘 기반에서 널리 사용하는 제3의 온라인 TTP를 사용하지 않고 프로토콜을 구성할 수 있음
    - 인증기관은 보통 오프라인 TTP로 분류함
- 키 확립 프로토콜에서 공개키 암호알고리즘 사용에 따른 단점
  - 단점 1. 공개키 연산은 대칭키 암호화 연산보다 상대적으로 비용이 많이 소요됨
  - 단점 2. 암호키의 길이가 대칭키보다 상대적으로 길어 키 관리 측면에서 불리함
  - 단점 3. 공개키 기반 구조가 필요함 (오프라인 서버)

키 확립 프로토콜

- 키 확립 프로토콜(key establishment protocol): 둘 이상의 참여자들이 비밀키를 공유할 수 있도록 해주는 암호프로토콜
- 키 확립 프로토콜을 통해 얻어지는 비밀키는 단일 세션에 사용하기 위한 세션키(session key)임
- 키 확립 프로토콜의 구성
  - 키 전송 또는 동의 과정
  - 키 확인(key confirmation) 과정: 상대방이 자신과 같은 키를 가지고 있는지 확인하는 과정
- 키 확립 프로토콜의 분류
  - 분류 1. 키를 생성하는 주체에 따른 분류
  - 분류 2. 사용하는 암호기술에 따른 분류
  - 분류 3. 키를 확립하는 사용자 수에 따른 분류

키 생성 주체에 따른 분류

- 키 전송 프로토콜(key transport protocol): 참여자 중 한 명이 키를 생성함
  - 대칭키만을 사용할 경우에는 보통 TTP를 사용하는 중재 방식의 프로토콜이 대부분임
  - 이때 TTP는 크게 키 중계 센터(KTC, Key Translation Center)와 키 분배 센터(KDC, Key Distribution Center)로 구분함
    - KTC는 키를 직접 생성하지 않고 안전하게 중계하는 역할만 하는 서버이고, KDC는 직접 세션키를 생성함. 보통 후자 방식을 주로 사용함
- 키 동의 프로토콜(key agreement protocol): 특정 사용자가 홀로 키를 생성하지 않는 방식
  - 보통 키를 확립하는 모든 참여자들이 동등(보통 랜덤값을 서로 교환한 후에 이를 이용하여 각자 생성)하게 키 생성에 기여함
  - 대부분 공개키 기반 프로토콜임
- 혼합 방식의 프로토콜(hybrid protocol): 참여자 중 일부만 키 동의 방식을 사용하는 프로토콜

$C=\{M\}.K_1,\ MAC.K_2(C),\ \{T_A||K\}.+K_B$

슬라이드 8의 프로토콜은 자체 강화 방식의 키 전송 프로토콜

키를 확립하는 사용자 수에 따른 분류

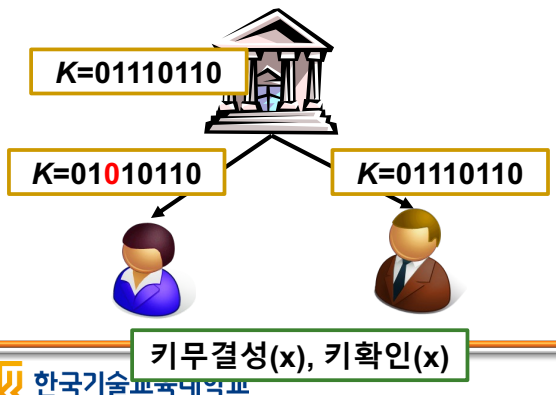
- 대부분의 키 확립 프로토콜은 2자 간의 세션키를 공유하기 위한 프로토콜임
- 보통 키 확립 프로토콜은 2자간과 다자간 프로토콜로 분류되었지만 현재는 2자 간, 3자 간, 다자 간으로 분류됨
  - 2000년도에 Joux가 겹선형 사상이라는 수학적 개념을 이용하여 3자 간 자체 강화 방식의 키 동의 프로토콜을 제안한 이후 이와 같이 분류하고 있음
  - 다자간 프로토콜을 다른 말로 회의 키(conference key) 프로토콜 또는 그룹 키(group key) 프로토콜이라 함

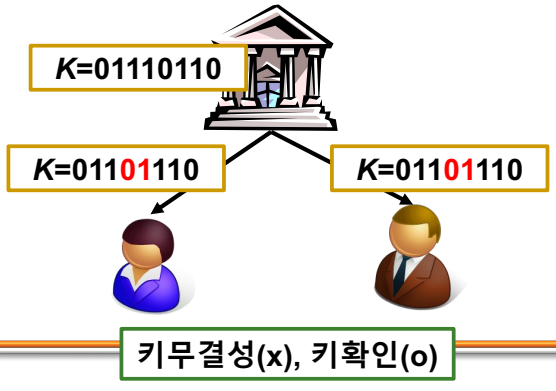
키 확립 프로토콜의 요구사항

- 키의 비밀성: 키를 확립하는 참여자 외에는 키를 얻을 수 없어야 함
- 키의 최근성: 키를 확립하는 참여자는 수신한 키가 이전에 사용한 적이 없는 이번에 생성한 키임을 확신할 수 있어야 함
- 키의 용도: 키를 확립하는 참여자는 수신한 키가 본인이 생각하고 있는 상대방과 사용하기 위한 키인지 확인할 수 있어야 함
- 생성 주체 확인, 참여자 확인
  - 키 전송 프로토콜의 경우 키를 수신한 참여자는 이 키를 프로토콜에서 정의한 주체가 생성한 것인지 확인할 수 있어야 함
  - 프로토콜에 따라 현재 참여하고 있는 상대방이 자신이 의도한 상대방인지 명백하게 확인할 필요가 있을 수 있음
    - 자체 강화 방식의 키 동의 프로토콜에서 상대방에 대한 인증 필요
    - 키 전송 방식에서는 서버에 대한 신뢰를 통해 간접적으로 상대방을 인증함
- 키 확인: 키를 확립하는 참여자들은 서로 같은 키를 가지고 있는지 확인할 수 있어야 함

키 무결성과 키 확인

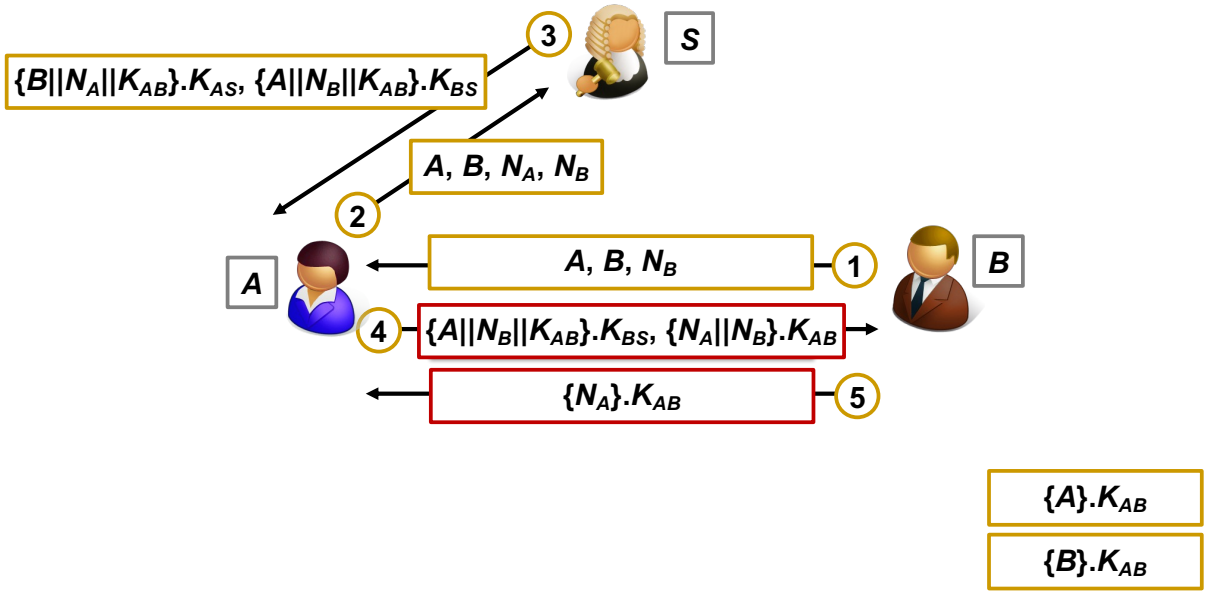
- 키 무결성(key integrity): 프로토콜 수행 과정에서 키나 키 생성에 사용한 값이 전송 과정에서 조작되지 않은 경우
  - 키 전송 프로토콜: 참여자가 받은 키가 원래 생성한 키와 동일
  - 키 동의 프로토콜: 참여자가 받은 키 생성에 사용한 값이 송신자가 보낸 값과 동일
- 키 무결성과 키 확인은 다른 의미임. 키 무결성보다는 키 확인이 중요함
  - 키 비밀성이 보장되면 키 무결성이 보장되지 않더라도 키 확인이 되면 사용할 수 있지만 키 확인이 되지 않으면 사용할 수 없음





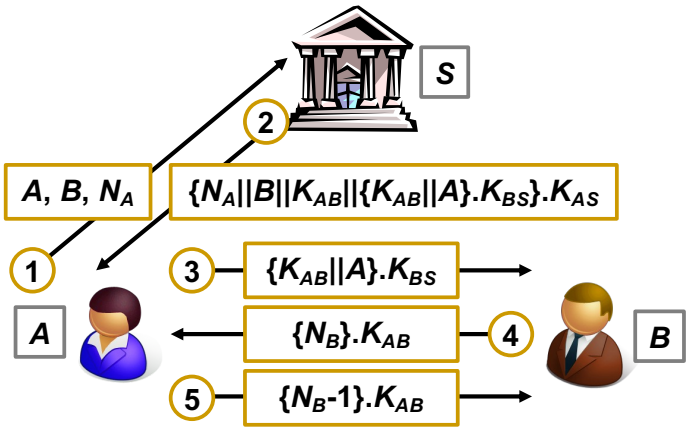
키 확인

- 키 확인은 항상 비교적 쉽게 추가할 수 있음
  - 키 확인 과정의 추가는 보통 라운드의 증가가 필요함



키 전송 프로콜의 예 (1/2)

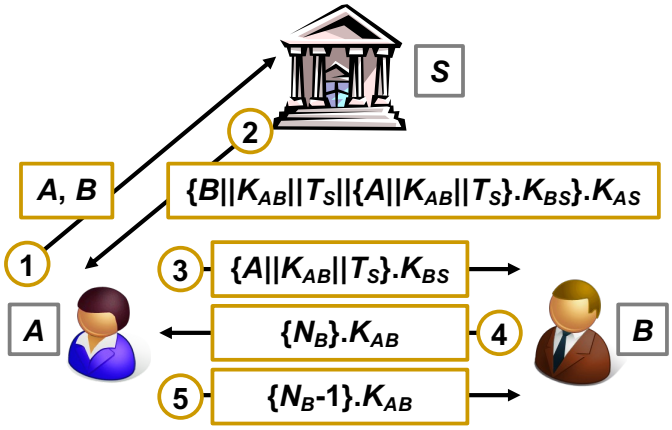
- Needham-Schroeder 프로토콜, 1978



  - 문제점 1. Bob은 K_{AB}의 최근성을 확인할 수 없음
  - 문제점 2. Alice는 Bob과 달리 Bob이 자신과 동일한 키 K_{AB}를 가지고 있다는 것을 확인할 수 없음

키 전송 프로토콜의 예 (2/2)

- Denning과 Sacco 프로토콜, 1981

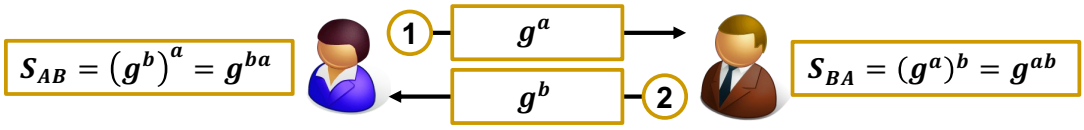


- 타임스템프 기반 메시지 최근성 보장 기법을 이용하여 노트 2에서 살펴본 시도 4 프로토콜의 문제점을 해결하였음

이산대수 문제

- 군(group)은 이항 연산에 의해 닫혀 있는 대수구조를 말함
- 예) $\circ = \times \bmod 7$
  - 결과집합: {1,2,3,4,5,6}, 항등원: 1, 모든 원소는 역원이 있음
    - $4 \circ 3 \equiv 5 \pmod 7, 4 \circ 2 \equiv 1 \pmod 7$, 2와 4는 서로의 역원임
  - 역원을 곱하여 나눗셈 효과를 얻을 수 있음
    - 예) $4 \circ x \equiv 5,\ 2 \circ 4 \circ x \equiv 2 \circ 5,\ x \equiv 3$
  - 5을 계속 거듭제곱하면 군에 있는 모든 원소를 만들 수 있음
    - $5^1 = 5, 5^2 = 4, 5^3 = 6, 5^4 = 2, 5^5 = 3, 5^6 = 1$
  - 이와 같은 원소를 생성자(generator)라 하며, 생성자가 있는 군을 순환군(cyclic group)이라 함
  - $5^2 = 4$에서 2는 기저 5에서 4에 대한 이산대수임. $\log_5 4 = 2$
- 이산대수 문제: 순환군의 정보, 생성자 g, 임의의 군 원소 y가 주어졌을 때 기저 g에 대한 y의 이산대수를 찾는 문제
- 순환군의 크기가 일정 크기 이상이면 이 문제는 계산적으로 어려운 문제임

키 동의 프로토콜 (1/2)



- Diffie-Hellman 키 동의 프로토콜 (1976년)
- 여기서 g는 유한순환군의 생성자이며, 이 군에서 이산대수 문제와 Diffie-Hellman 계산 및 결정 문제는 계산적으로 어려움
  - DH 계산 문제: 군 정보, g^a, g^b가 주어졌을 때 g^{ab}를 계산하는 문제
  - DH 결정 문제: 군 정보, g^a, g^b, g^c가 주어졌을 때 g^c와 g^{ab}가 같은지 결정하는 문제
- Alice와 Bob만 g^{ab}를 계산할 수 있음
  - g^{ab}를 직접 세션키로 사용하지 않고, 키 유도 함수(KDF, Key Derivation Function)에 g^{ab}를 입력하여 세션키를 계산함
- 문제점. 서로 g^a와 g^b가 상대방이 보낸 값인지 알 수 없음

키 동의 프로토콜 (2/2)

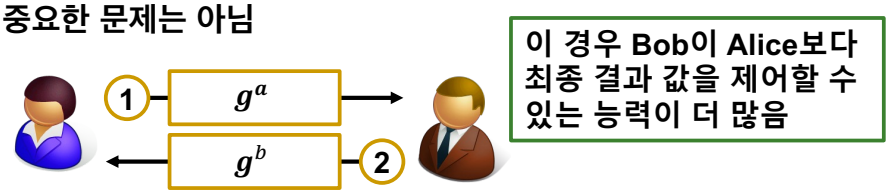
- 키 동의 프로토콜은 어떤 특정 참여자가 홀로 키를 결정하지 않음
- 보통 키 확립하는 모든 참여자가 키를 결정하는데 동등하게 참여함
  - 보통 각 참여자들이 동일한 수준의 랜덤 값을 교환하고, 이 값들을 이용하여 키를 결정함
- 각 참여자 입장에서 자신이 선택한 랜덤 요소가 충분히 랜덤하면 프로토콜을 통해 생성된 세션키도 역시 충분히 랜덤하다는 것을 믿을 수 있음
- 각 참여자 입장에서 자신이 선택한 랜덤 요소가 최근에 생성된 것이면 프로토콜을 통해 생성된 세션키도 최근에 생성되었다는 것을 믿을 수 있음
  - 키 전송 프로토콜들과 달리 키 동의 프로토콜에서는 키의 최근성을 제공하기 위한 별도 메커니즘의 사용이 필요 없음
- 하지만 반대로 키 제어 요구사항처럼 키 전송 프로토콜에는 없는 요구사항도 있음

$$
\begin{aligned}
&Msg\ 1. A \rightarrow S: A, B, \{B||N_A\}.K_{AS}\\
&Msg\ 2. S \rightarrow B: \{A||N_A\}.K_{BS}\\
&Msg\ 3. B \rightarrow S: \{A||N_B||N_A\}.K_{BS}\\
&Msg\ 4. S \rightarrow A: \{N_A||N_B||B\}.K_{AS}\\
&Msg\ 5. A \rightarrow B: \{N'_A||A\}.K_{AB}\\
&Msg\ 6. B \rightarrow A: \{N'_A\}.K_{AB}
\end{aligned}
$$

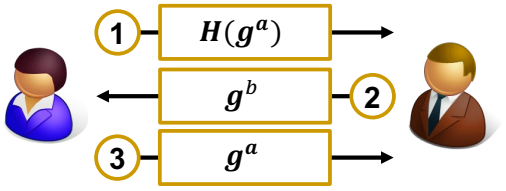
$K_{AB} = H(N_A||N_B)$

키 제어 요구사항

- 키 제어 요구사항. 참여자 중 어느 누구도 공유되는 세션키가 사전에 미리 계산되거나 선택되어진 값이 되도록 만드는 것은 계산적으로 어려워야 함
- Mitchell 등은 참여자 중 자신의 랜덤요소를 가장 늦게 제공하는 참여자는 다른 참여자보다 키를 제어할 수 있는 확률이 높다는 것을 보였음
  - 이 문제점은 랜덤 요소를 해시한 값을 미리 제공한 후에 나중에 실제 랜덤 요소를 전달하여 해결할 수 있음
  - 이 해결책의 단점은 프로토콜의 라운드 수가 하나 증가함
  - 중요한 문제는 아님



이 경우 Bob이 Alice보다 최종 결과 값을 제어할 수 있는 능력이 더 많음

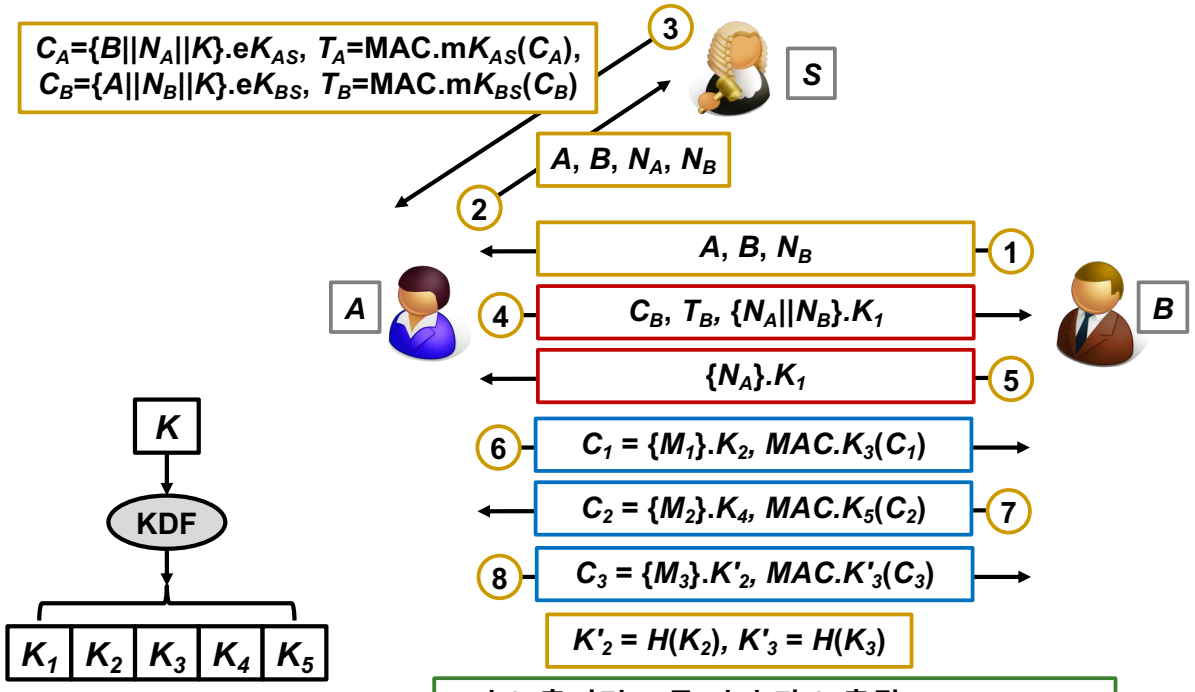


이 경우 Alice와 Bob이 최종 결과 값을 제어할 수 있는 능력은 동일함

최근 추세 (1/2)

- 기본 암호화 방식으로 인증 암호화 중 encrypt-then-mac 방법을 사용함
  - 1개의 대칭키가 아니라 2개의 대칭키 필요
    - encryption key, mac key
  - 2개의 대칭키를 확립하는 것이 아니라 확립된 하나의 비밀 정보로부터 필요한 개수의 키를 계산하여 사용함
    - 이때 사용하는 것이 키 유도 함수임
    - 키 유도 함수에서 생성되는 값은 서로 독립적임
      - 256비트 출력을 반으로 나누었을 때 한 쪽이 노출되어도 다른 쪽 비밀에 아무런 영향을 주지 못함
- 키 확인 과정에서 사용하는 키와 데이터 암호화할 때 사용하는 키도 다른 키를 사용함 (키 분리 원리)
- 2자 간에 각 방향마다 다른 키를 사용함
- 총 5개의 키(키 확인, 양방향 암호화키, 양방향 MAC키)가 필요함
  - 심지어 메신저 보안 프로토콜에서는 메시지마다 다른 키를 사용함

최근 추세 (2/2)



K가 노출되면 모든 키가 다 노출됨
K_i중 하나가 노출되어도 다른 것에는 영향이 없음
키 분리 원리(key separation principle)