



# 키 확립 프로토콜 대표 사례

NOTE 10

DATA

한국기술교육대학교 컴퓨터공학부 김상진

[sangjin@koreatech.ac.kr](mailto:sangjin@koreatech.ac.kr)  
[www.facebook.com/sangjin.kim.koreatech](http://www.facebook.com/sangjin.kim.koreatech)

## 교육목표

- 티켓 방식 vs. 다중 벡터 방식
- TLS: 웹 보안
- WEP, WPA, WPA2: 무선랜 보안
- Signal: 메신저 보안



# 꼭 일회용 세션키를 사용해야 하나?

- 일반적으로 키 확립 프로토콜은 한 세션에서만 사용하는 세션키를 확립하기 위한 목적으로 활용함
- 하지만 필요할 때마다 매번 키 확립 프로토콜을 수행하는 것이 효율성 측면에서는 환경에 따라 부담이 될 수 있으며, 자체 강화 방식이 아니라 서버를 활용하는 경우에는 가용성 측면에서도 문제가 될 수 있음
- 이를 극복하는 방안으로 **한 번 확립한 세션키를 여러 번 사용**하는 방식이 제안되었으며, **한 번에** 하나의 세션키를 확립하는 것이 아니라 **여러 개를 확립**하는 방식도 제안되었음
- 전자를 **티켓 방식**이라 하고, 후자를 **다중 벡터 방식**이라 함
- 두 방식은 모두 세션키를 장기간 유지해야 하는 단점이 있음
  - 두 방식은 모두 추가적인 키 관리 및 저장공간이 요구됨
  - 티켓방식의 경우에는 하나의 비밀키를 여러 세션에 사용하므로 안전성에 나쁜 영향을 줌
- 두 방식 모두 보통 대칭키 암호알고리즘만 사용하는 환경을 위해 제안된 방식임

## 티켓 기반 프로토콜 (1/2)

- 티켓 방식: 매 세션마다 세션키를 새로 확립하는 비용을 줄이기 위해 **확립된 세션키를 일정한 기간 사용할 수 있도록 함**
- **티켓**: 세션키, 세션키의 용도, 유효기간 등의 정보를 암호화한 암호문
  - 세션키의 용도: 보통 티켓을 사용할 수 있는 사용자 정보
  - 티켓의 용도는 티켓을 암호화할 때 사용한 키에 의해 결정됨
- 티켓 방식의 참여자 분류
  - 티켓을 발급하는 참여자: 보통 제3의 신뢰 서버
  - 티켓을 발급받는 사용자
  - 제시된 티켓을 검증하여 서비스를 제공하는 참여자



## 티켓 기반 프로토콜 (2/2)

- 티켓의 일반적인 형태: Alice가 Bob으로부터 서비스를 받기 위해 사용할 수 있는 티켓

$$\{T_S || L || K_{AB} || A\}.K_{BS}$$

- S: 티켓을 발급해주는 신뢰 서버
- $T_S$ : 시작 시각, L: 수명  $\leftarrow$  유효기간을 나타내기 위한 정보
  - 수명 대신에 끝 시각 사용 가능
- 기존 대칭키 기반 키 전송 프로토콜에서 키 분배 서버가 세션키를 분배하기 위한 암호문( $\{N_B || K_{AB} || A\}.K_{BS}$ )의 최근성 확인을 위한 부분이 유효기간 정보로 바뀌는 형태임
- 티켓은 클라이언트가 유지하며, 서비스를 받고자 할 때 매번 제시해야 함
  - 서버는 티켓을 복호화할 수 있어야 함
  - 클라이언트는 티켓에 포함되어 있는 세션키를 알고 있다는 것을 증명해야 함
- 클라이언트는 티켓을 발급받을 때 티켓에 포함된 세션키를 별도 받아야 하며, 이를 티켓과 함께 안전하게 유지해야 함
  - **참고.** 클라이언트는 티켓을 복호화할 수 없음

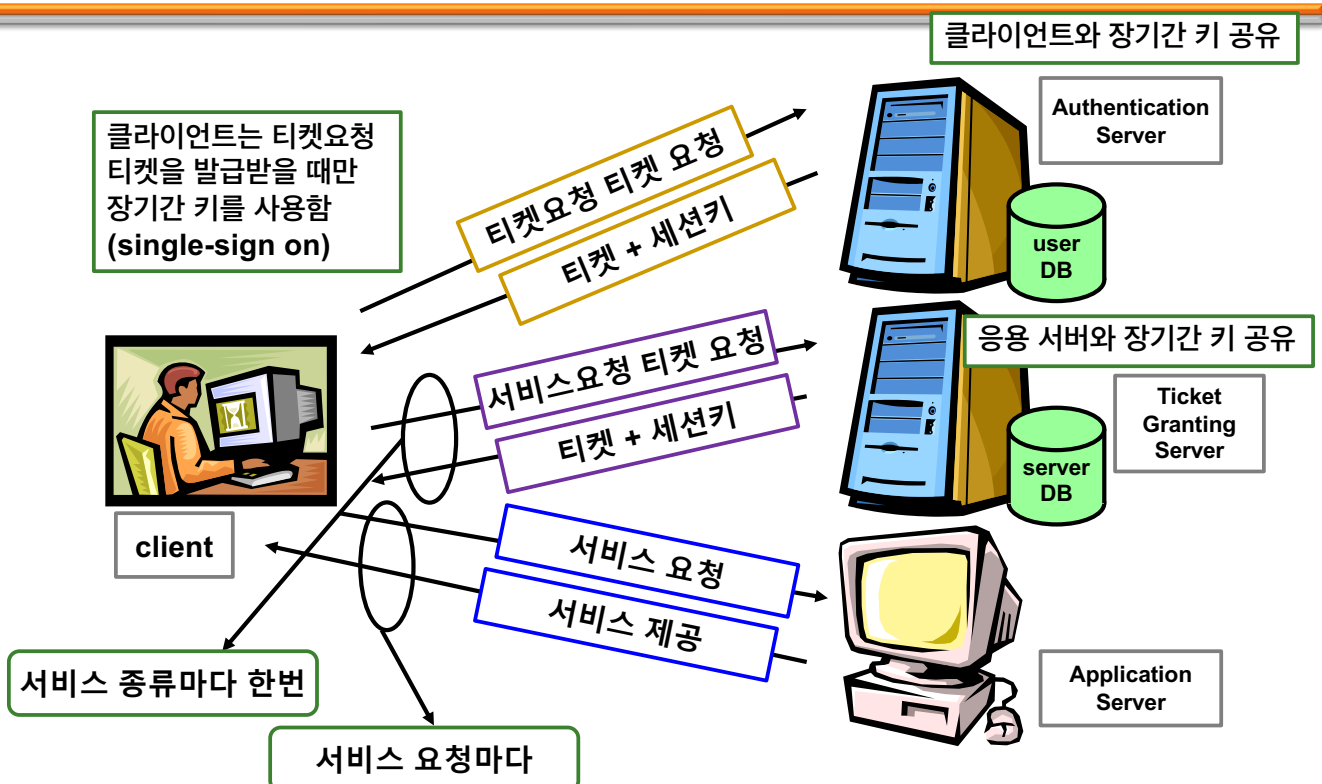
## 커버로스(Kerberos) (1/4)

- MIT에서 개발한 네트워크 인증시스템
- IETF 표준: RFC4120
- telnet, ftp, pop과 같은 서비스를 사용하기 위해 서버에 사용자들이 접속을 함
  - 이 서비스들은 계정명과 패스워드를 통해 사용자를 인증함
  - **문제점.** 패스워드가 평문 형태로 전달하기 때문에 패킷 분석을 통해 쉽게 다른 사용자의 패스워드를 알 수 있음
  - 지금은 커버로스보다는 TLS나 SSH 등을 더 많이 사용함
- Needham-Schroeder 프로토콜에 기반하고 있으며, 현재 버전 5까지 개발되어 있음
- 타임스탬프를 이용하므로 시스템 간의 시간 동기화가 필요함

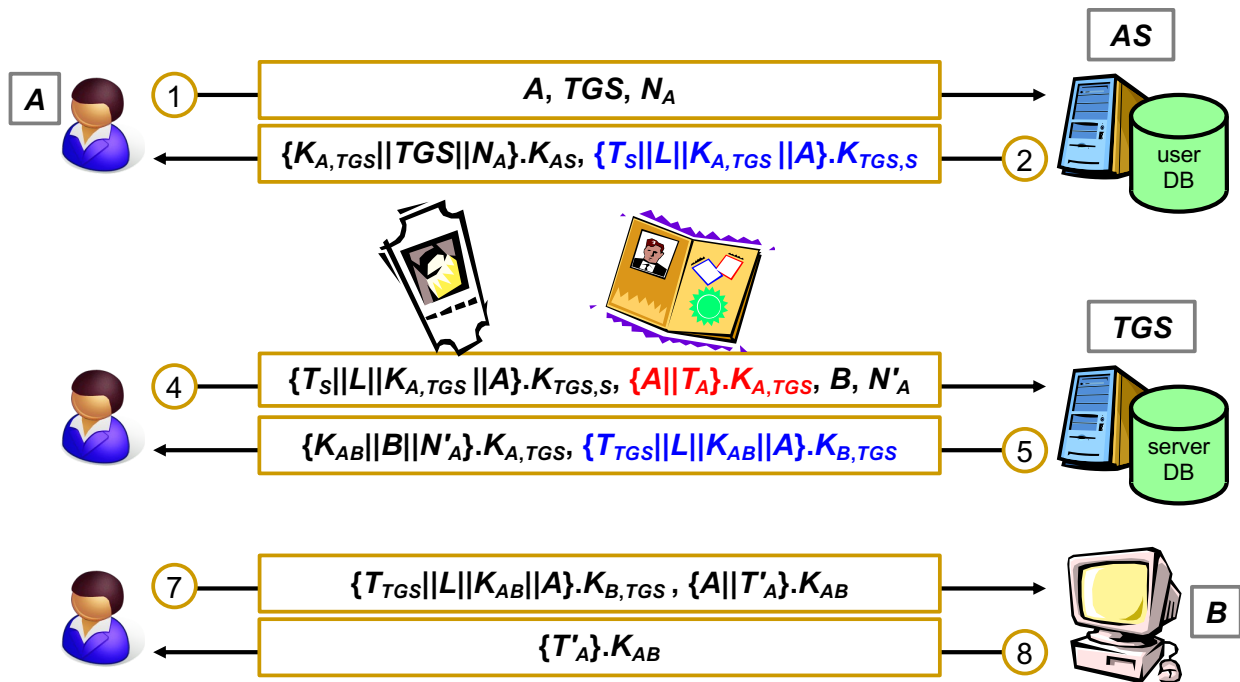
## 커버로스 (2/4)

- 목적은 사용자가 응용서버에 접속하는 것임
- 구성요소
  - 인증서버(Authentication Server): 사용자의 인증을 담당하는 서버
    - 티켓승인서버와 사용할 수 있는 티켓을 발급하여 줌
    - 사용자와 장기간 대칭키를 공유함
  - 티켓승인서버(TGS, Ticket Granting Server): 서비스를 제공하는 응용서버와 사용할 수 있는 티켓을 발급하는 서버
    - 응용서버와 장기간 대칭키를 공유함
  - 응용서버: 서비스를 제공하는 실제 사용자가 접속하고자 하는 서버
- 티켓승인서버를 사용하는 이유: 장기간 키의 사용을 최소화하기 위함

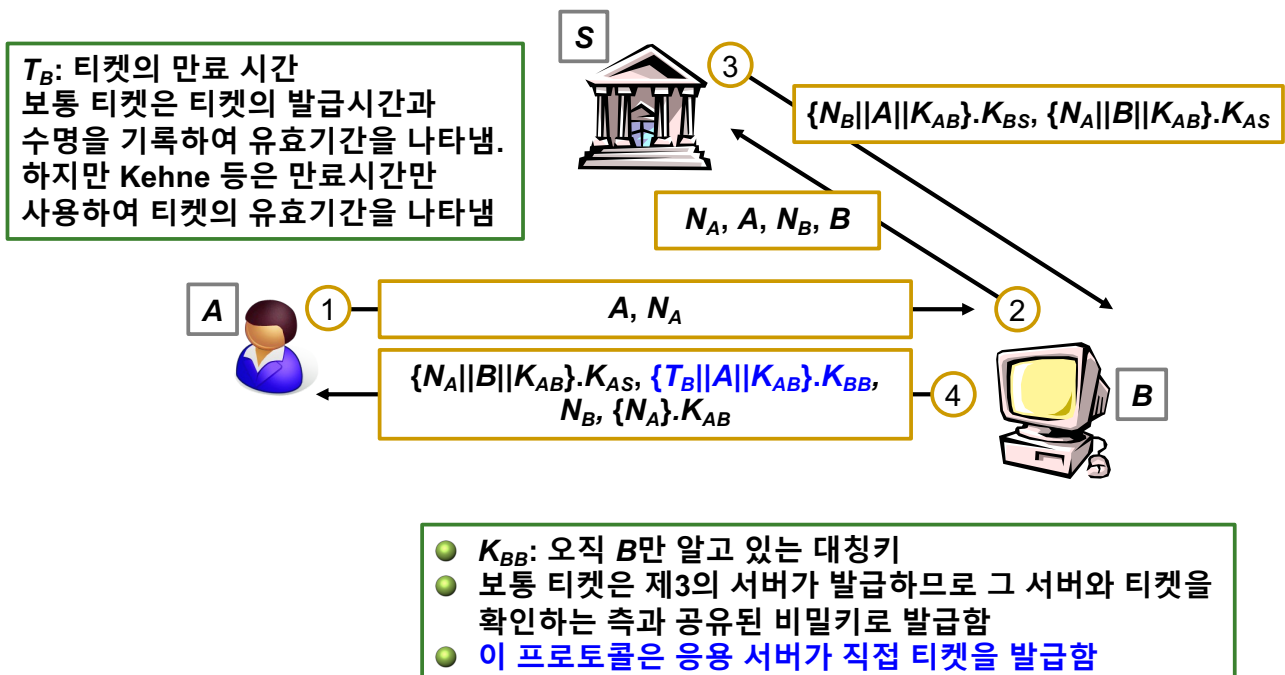
## 커버로스 (3/4)



# 커버로스 (4/4)



# Kehne 등의 프로토콜, 1992 (1/2)



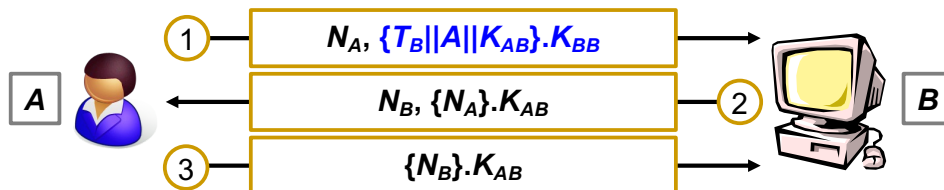
# Kehne 등의 프로토콜 (2/2)

- Kehne 등의 프로토콜의 장점

- 티켓 방식임에도 불구하고 시스템 간의 시간 동기화가 필요 없음
  - 티켓을 서비스 제공 응용 서버가 직접 타임스탬프를 기록하여 발급하고, 이 타임스탬프를 나중에 해당 서버가 확인함
- 사용자에게 투명하게 서비스를 제공할 수 있음
  - 사용자는 인증서버, 티켓승인서버와 같은 추가적인 서버에 접속할 필요가 없음

티켓이 있는 경우에 사용하는 프로토콜

타임스탬프를 사용하지 않고 상호인증하기 위해서는 3개의 메시지가 필요



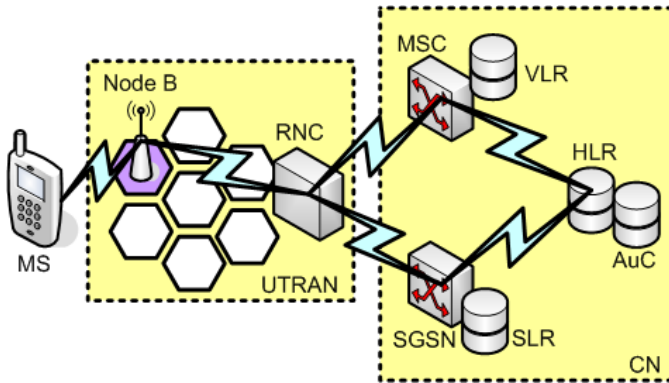
## 다중 벡터 방식

- 티켓 방식은 한 번 확립된 비밀키를 여러 번 사용함
  - 이때 사용자가 티켓을 유지하며, 서버는 매번 티켓을 받아 검증함
- 다중 벡터 방식은 키를 확립할 때 하나의 비밀키를 확립하는 것이 아니라 여러 개의 비밀키(일회용 티켓)를 확립하여 사용함
  - 클라이언트가 여러 개의 비밀키를 유지하는 것은 환경에 따라 클라이언트에게 부담이 될 수 있음
  - 이 문제 때문에 티켓 방식과 정반대로 응용 서버(서비스를 제공하는 서버)가 유지하는 형태를 사용할 수 있음
- 하나의 키를 확립한 후에 KDF를 이용하여 서로 독립적인 여러 개의 키를 만들어 사용할 수 있음
  - 다중 벡터는 이와 같은 형태는 아님

# UMTS AKA (1/3)

- UMTS(Universal Mobile Telecommunication System)
  - 유럽에서 사용한 3G 이동통신 표준
  - AKA(Authentication and Key Agreement) 프로토콜을 사용하여 단말을 인증함
- UMTS의 구조

다른 나라로 여행가면 해당 국가 이동통신사는 방문인의 모바일폰을 인증할 수 있는 정보를 가지고 있지 않음

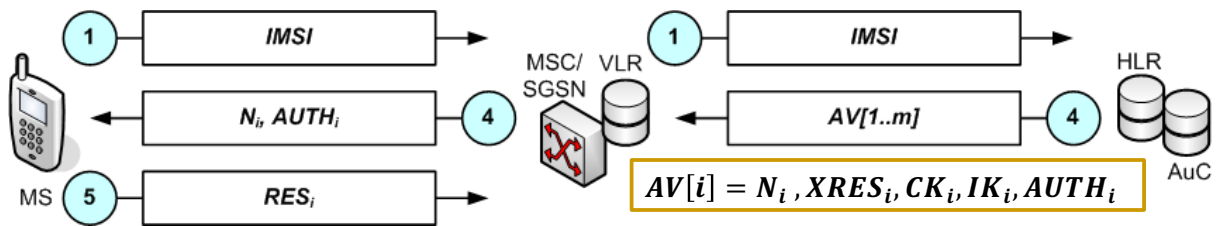


- MS: Mobile Station
- HLR: Home Location Register
- AuC: Authentication Center
- UTRAN: UMTS Terrestrial Radio Access Network
- RNC: Radio Network Controller
- CN: Core Network
- MSC: Mobile Switching Center
- VLR: Visitor Location Register
- SN: Serving Network
- HN: Home Network

# UMTS AKA (2/3)

- MS는 IMSI(International Mobile Subscriber Identifier)로 독특하게 식별할 수 있음
  - 프라이버시를 위해 가능하면 IMSI 대신에 TMSI(Temporary MSI)를 이용하여 식별함 (최초 인증 이후에는 TMSI 사용)
- MS는 특정 HN에 등록되어 있으며, HN과 장기간 대칭키와 카운터를 공유함
  - SN은 MS를 인증할 수 있는 정보를 가지고 있지 않음
  - 따라서 SN은 MS의 HN에 접속하여 MS를 인증할 수 있는 정보를 받아 MS를 인증해야 함
  - HN에 접속하는 비용을 줄이기 위해 SN은 한번에  $m$ 개의 인증 벡터(AV, Authentication Vector)를 받으며, MS를 인증해야 할 때마다 하나의 벡터를 사용함
  - MS는 공유한 카운터를 이용하여 SN이 전달한 인증 벡터의 최근성을 확인함

# UMTS AKA (3/3)



$AK_j = MAC^5 \cdot K_{MH}(N_j)$   
 $C_{HM} = AUTH_i[0] \oplus AK_j$   
**compare**  $C_{HM} > C_{MH}$   
 $S_j = MAC^1 \cdot K_{MH}(AUTH_j[1] || C_{HM} || N_j)$   
**verify**  $S_j = AUTH_j[2]$   
 $RES_j = MAC^2 \cdot K_{MH}(N_j)$

$S_i = MAC^1 \cdot K_{MH}(AMF || C_{HM} || N_i)$   
 $XRES_i = MAC^2 \cdot K_{MH}(N_i)$   
 $CK_i = MAC^3 \cdot K_{MH}(N_i)$   
 $IK_i = MAC^4 \cdot K_{MH}(N_i)$   
 $AK_i = MAC^5 \cdot K_{MH}(N_i)$   
 $AUTH_i = C_{HM} \oplus AK_i, AMF, S_i$

- 이 방식의 문제점
  - HN과 SN 사이에 많은 대역폭을 사용함
  - SN의 많은 저장 공간이 필요함
- 위 문제는 티켓 방식을 통해 해결 가능하지만 다음과 같은 차이점이 있음
  - AV는 벡터 간에 독립적인 반면에 티켓의 경우에는 티켓에 포함된 세션키가 노출되면 공격자가 티켓을 활용할 수 있게 됨

## 티켓 vs. 다중벡터

	티켓	다중벡터
공통점	키 확립 후 일정기간 동안(일정 회수만큼)은 인증서버와 통신하지 않고 서비스 이용 가능	
발급시점	하나의 티켓만 발급	여러 개의 벡터 발급
통신비용	적음	상대적으로 많음
저장공간	적음	상대적으로 많이 요구됨
사용방법	유효기간 동안 계속 동일 티켓 사용	각 키는 오직 한번만 사용함
사용방법에 따른 효과	<ul style="list-style-type: none"> <li>● 동일 비밀키를 여러 번 사용</li> <li>● 유효기간의 범위 한정</li> </ul>	<ul style="list-style-type: none"> <li>● 비밀키는 한번만 사용</li> <li>● 키를 한번만 사용하도록 제한하는 방법 필요(예: 카운터)</li> </ul>



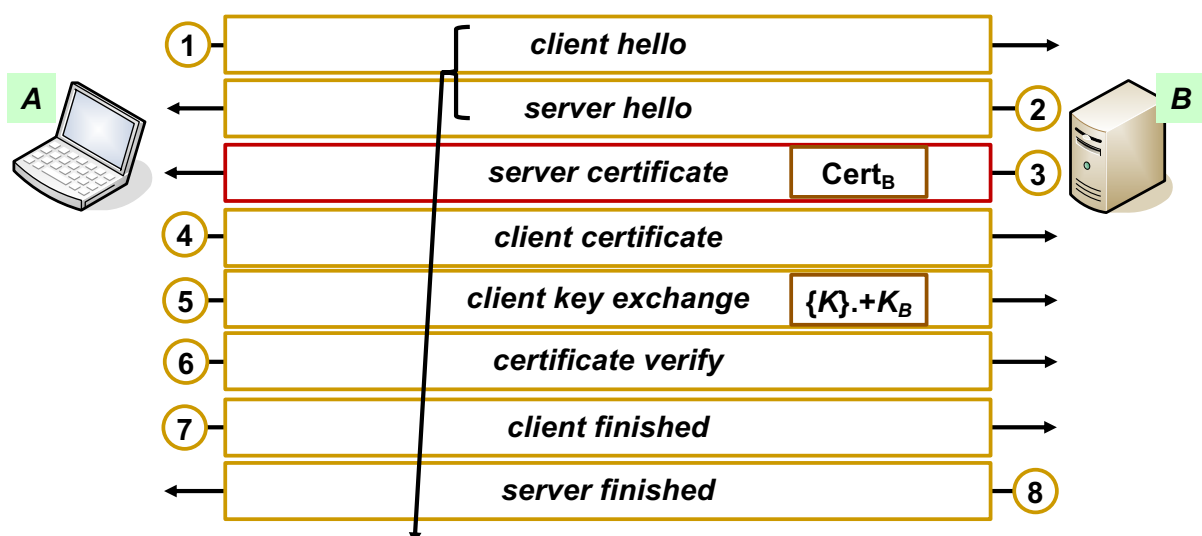
# TLS 프로토콜 (1/5)

- TLS(Transport Layer Security) 프로토콜
  - SSL(Secure Socket Layer) 프로토콜의 표준화된 버전
  - 웹에서 현재 가장 많이 사용하고 있는 암호프로토콜
- 실제 현장에서 사용하는 프로토콜의 경우에는 특정 종류의 알고리즘만 사용하지 않고, 다양한 알고리즘을 사용할 수 있도록 함
  - 클라이언트와 서버가 협상을 통해 사용할 암호알고리즘을 결정한 후에 실제 프로토콜을 진행함
  - 협상에서 결정한 암호알고리즘 집합을 cipher suite라 함
- 구성
  - handshake protocol: 키 확립하는 부분
    - RSA 버전, Diffie-Hellman 버전 등 여러 버전이 있음
  - record protocol: 확립된 키로 데이터를 교환하는 부분



# TLS 프로토콜 (2/5)

## ● TLS 1.2 handshake protocol

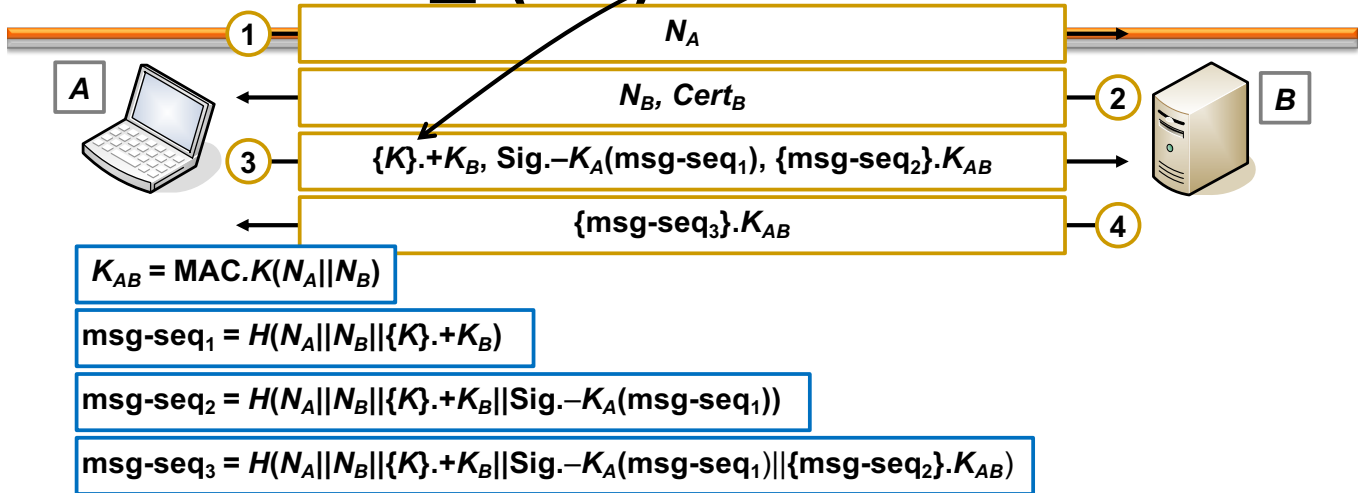


알고리즘 협상 (1.3의 경우)

- 키 확립 프로토콜: ECDH (타원곡선: P-256, X22519 등)
- 전자서명 알고리즘: RSA, ECDSA, EdDSA
- HMAC, HKDF: SHA-256
- 인증 암호화: AES-GCM, ChaCha20-Poly1305

# TLS 프로토콜 (3/5)

RSA version: PMS



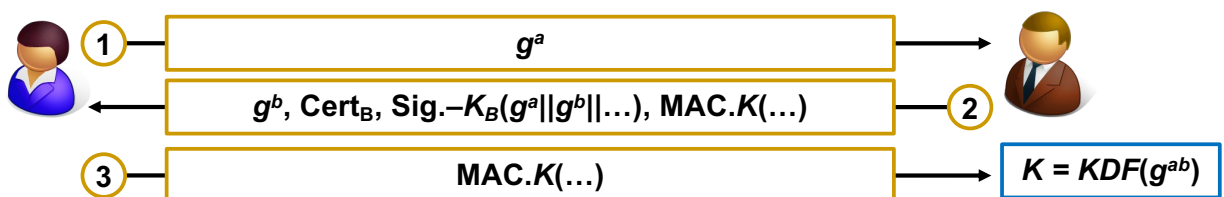
## 주요 특징

- 클라이언트가 PMS(pre-master secret)를 생성함
- 인증은 인증서를 이용하며, 상호인증이 가능하지만 보통 서버만 클라이언트가 인증함 (인터넷의 특성)
- 총 4개의 키를 확립: 클라이언트, 서버 각각 2(암호화용, MAC용)개
- TLS 1.2까지는 2-pass MAC-then-encrypt 방식 사용

# TLS 프로토콜 (4/5)

## TLS 1.3

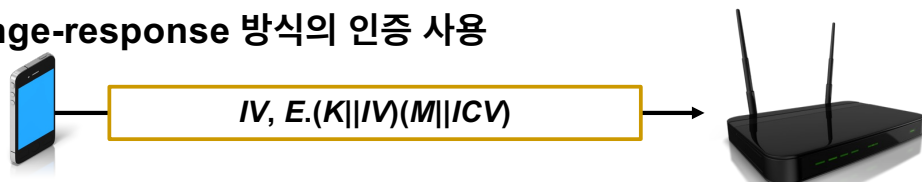
- 2018년에 제정 (RFC 8446)
- 전방향 안전성 의무화: RSA 키 교환 버전은 더 이상 사용하지 않음
  - DHE-RSA, ECDHE-DSS, ECDHE-ECDSA만 사용
- Handshake 프로토콜 개선: 메시지 수 축소
- encrypt-then-mac 형태의 인증 암호화 사용 의무
- TLS에서 DH는 어떻게?
  - DH는 중간자 공격에 취약하지만 TLS는 보통 상호 인증을 하지 않음
  - SIGMA와 유사함 (Alice의 서명만 없음)



# TLS 프로토콜 (5/5)

- 세션 재개(session resumption)
  - 세션 ID 방법. 클라이언트와 서버가 모두 세션 정보 유지(세션키 포함)
    - **문제점.** 서버가 유지해야 하는 정보가 많음
  - 세션 티켓 방법. 클라이언트만 세션 정보 유지
  - TLS 1.3 PSK(Pre-Shared Key) 방법
    - PSK를 세션 티켓 형태로 서버가 발급하여 클라이언트가 유지함
    - 원래 의미는 사전 등록된 키를 이용하는 방법을 말함
- 현장에서 사용하는 프로토콜은 과거 버전에 문제가 있어 새 버전이 출시되어도 과거 버전을 사용하지 못하도록 강제화하는 것이 어렵다는 문제점이 있음
  - 버전 협상을 하위 버전으로 만들어 하위 버전의 허점을 일부러 활용하는 공격이 가능함

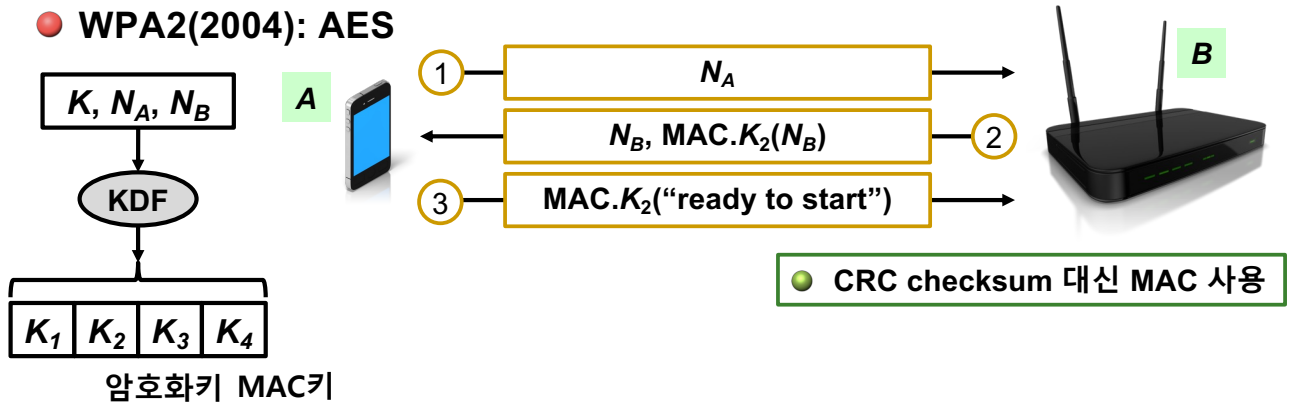
## 무선랜 보안 (1/2)

- 무선AP와 무선장치 간 보안
    - 다양한 장치 지원
    - 암호알고리즘 협상은 무겁기 때문에 적절하지 않음
    - 속도와 성능 측면에서 대칭 암호알고리즘 사용이 바람직함
  - WEP(Wired Equivalent Privacy) 표준. 1997년
    - RC4 스트림 암호 방식 사용: 40bit 키, IV: 24bit
      - 이 키는 우리가 입력한 무선AP 암호로부터 생성함 (패스워드 기반 키)
    - 간단한 CRC checksum 사용 ICV(Integrity Check Value): 32bit
    - challenge-response 방식의 인증 사용
- 
- 문제점
    - 스트림 방식이므로 동일 키 스트림을 두 번 사용하면 안됨
      - IV가 비교적 짧으며, AP를 초기화하면 IV가 0이 됨

# 무선랜 보안 (2/2)

- WPA(WiFi Protected Access): RC4

- WPA2(2004): AES



- WPA3(2018): AES

- 타원곡선을 이용하는 패스워드 기반 프로토콜을 사용함 (Note 11)

- SAE(Simultaneous Authentication of Equals) 기반

- 참고. 우리가 보통 가정에서 사용하는 것은 WPA2-PSK(Pre-Shared Key)임

# 시그널 프로토콜 (1/2)

- 메신저 보안 프로토콜

- 가장 널리 사용하고 있는 종단간 키 확립 프로토콜: WhatsApp, Facebook Messenger, Signal, Goggle의 Allo 등
- 프로토콜의 상세 내용을 공개하고 있으며, 소프트웨어도 오픈 소프트웨어임

- 메신저의 특수한 상황

- 상대방이 오프라인일 수 있음
- 서버에게도 대화 내용이 노출되는 것을 원하지 않음 → 종단간 암호
- 대화가 불연속적으로 기간의 제한없이 계속될 수 있음
- 한 사용자가 여러 기기를 사용할 수 있음

- 오프라인 문제

- DH를 하기 위한 값( $g^a$ )을 미리 서버에 저장
- 각 사용자도 이들( $(g^a, a)$  쌍 여러 개)을 단말에 관리하고 있어야 함



# 시그널 프로토콜 (2/2)

- 이중 톱니바퀴 기술을 이용하여 메시지마다 다른 키를 사용하여 암호화하는 방법을 통해 전방향 안전성, 완전한 전방향 안전성, 미래 안전성 등을 모두 제공함
- DH를 이용하여 공유 비밀 확립
- 확립된 비밀을 이용하여 대칭키 생성
- 메시지를 보낼 때마다 새로운 대칭키 생성
- 다음 대칭키는 이전 대칭키를 이용하여 생성
- 상대방으로부터 메시지를 수신하면 DH를 자동으로 다시 수행함

