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

암호프로토콜 공격 방법

NOTE 07

한국기술교육대학교 컴퓨터공학부 김상진

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

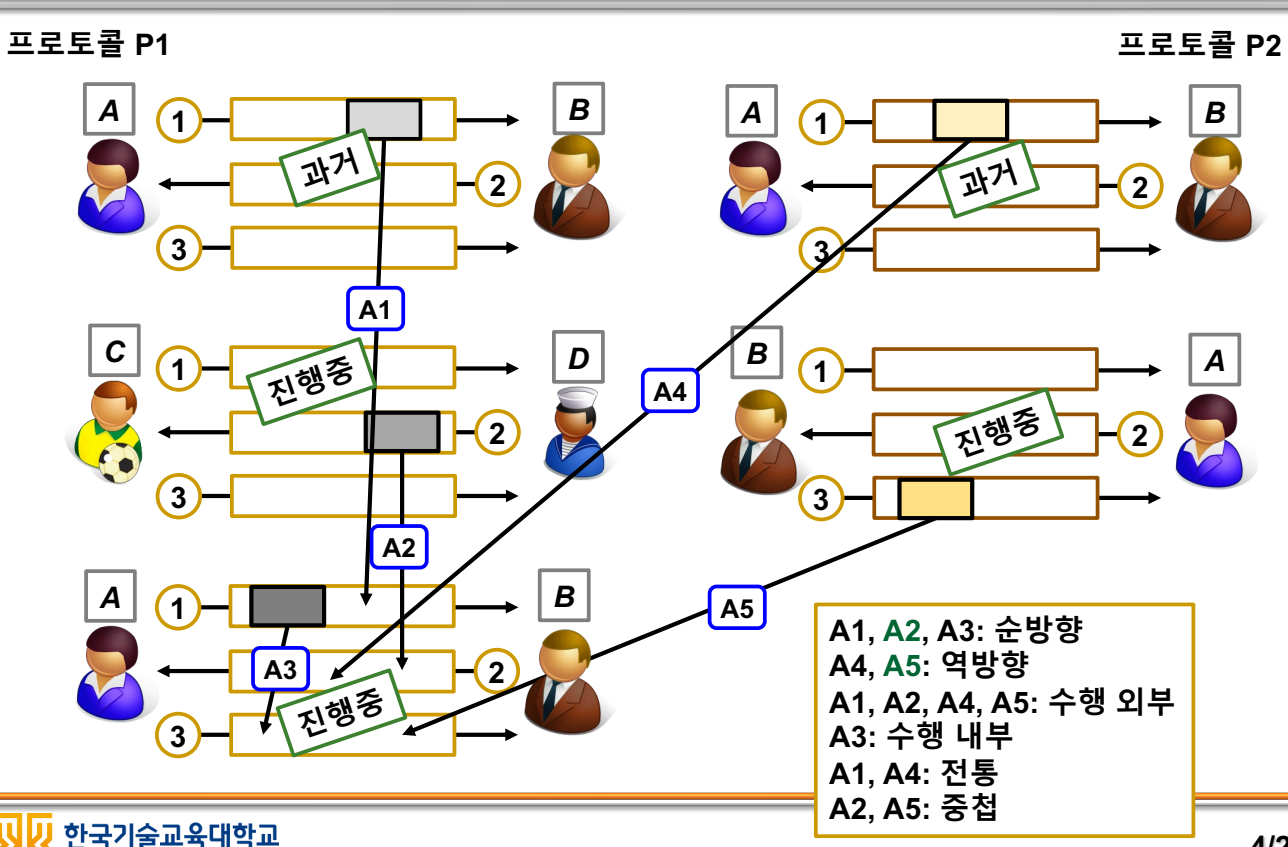
교육목표

- 재전송 공격
- 서비스 거부 공격
- 타입 공격
- 중간자 공격
- 키 노출 관련 공격
- 프로토콜 상호작용

암호프로토콜에 대한 공격

- 암호프로토콜에 대한 능동 공격은 매우 다양함
- 암호프로토콜을 설계할 때에는 이와 같은 공격들을 고려하여야 함
  - 명백한 허점 제거에 용이함
- 프로토콜마다 목적이 다르기 때문에 같은 공격도 프로토콜마다 의미가 다름
  - 특정 공격이 특정 프로토콜에 대해 가능하다고 하여 그것이 문제가 되지 않을 수 있음
  - 프로토콜 목적에 영향을 주어야 유효한 공격이라고 할 수 있음

재전송 공격



재전송 공격의 분류 기준 (1/2)

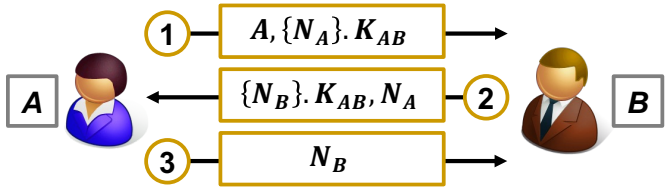
- 기준 1. 억압(suppressed) 여부: 공격에 이용할 메시지를 원 수신자가 받은 적이 있을 수 있고, 공격자가 차단하여 도달되지 못한 경우도 있음
  - 억압된 재전송 공격은 중복 검사로 방어할 수 없음
- 기준 2. 변경 여부: 메시지를 재전송할 때 원래 메시지를 그대로 전송할 수 있고, 일부 내용을 변경하여 전송할 수 있음
  - 메시지 중 암호화되지 않은 부분은 쉽게 변경 가능
- 기준 3. 의도변경: 원 메시지와 동일한 효과를 위해 재전송할 수 있고, 다른 효과를 위해 재전송할 수 있음
- 기준 4. 순방향, 역방향: 메시지를 원 수신자에게 재전송하면 순방향이고, 거꾸로 송신자에게 전송하면 역방향이 됨
- 기준 5. 수행 내부, 수행 외부: 한 프로토콜 수행에 사용된 메시지를 해당 수행에서 사용하면 수행 내부 공격이고, 다른 프로토콜 수행에 사용된 메시지를 이용하면 수행 외부 공격이라 함
- 기준 6. 전통, 중첩(interleaved): 공격에 성공하기 위해 두 개의 프로토콜을 병행으로 수행해야 하면 중첩 공격, 병행 수행이 필요 없으면 전통 공격이라 함

재전송 공격의 분류 기준 (2/2)

- 역방향 공격이 가능하기 위해서는 양방향으로 전달되는 암호문의 형태가 같아야 함
- 수행 외부 공격의 경우 같은 암호키를 여러 프로토콜에서 사용하면 다른 종류의 프로토콜 메시지를 활용하여 공격할 수 있음
- 보통 재전송 공격은 최근성 보장 기법(타임스템프, 카운터, 난스 등)을 이용하여 방어함
- 역방향, 수행 내부, 수행 외부를 방어하기 위한 방법
  - 방법 1. 프로토콜의 각 메시지 내 암호문의 형태와 크기를 다르게 함
    - 크기를 맞추기 위해 암호문의 일부를 이용하여 공격할 수 있음
  - 방법 2. 각 메시지의 암호문 내에 메시지 번호, 메시지 방향, 프로토콜 수행 식별자를 포함함
  - 방법 3. 각 방향(예: A→B, B→A)마다 다른 키를 사용함
  - 방법 4. 같은 장기간 키를 서로 다른 프로토콜에서 사용하지 않음

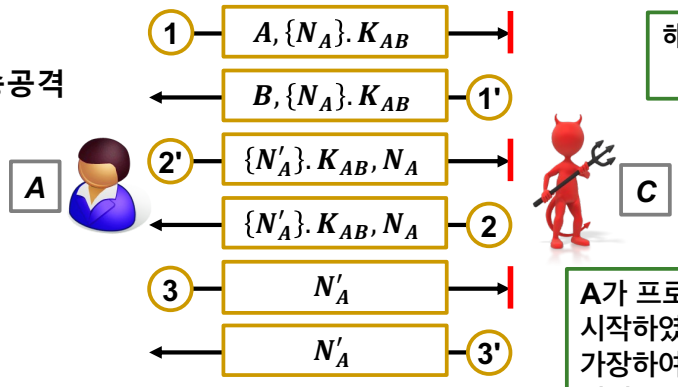
재전송 공격의 예

- 예)



- K_{AB}: Alice와 Bob 간의 공유된 비밀키
- 프로토콜의 목적: 상호 인증(mutual authentication)
- K_{AB}는 오직 둘 만이 알고 있기 때문에 Alice는 N_A를 통해 Bob의 존재를 확인할 수 있고, Bob도 같은 이유로 N_B를 통해 Alice의 존재를 확인할 수 있음

억압 수행외부
중첩 역방향 재전송공격



reflection attack

해결책 1. Naming 기법
$A, \{B||N_A\}.K_{AB}$

해결책 2. 각 방향마다 다른 키를 사용

A가 프로토콜(1~3 메시지)을 시작하였는데, 공격자가 B로 가장하여 A와 프로토콜(1'~3')을 병행으로 수행하여 공격하고 있음

Msg 1. $A \rightarrow B$:	N_A
Msg 2. $B \rightarrow A$:	N_B, Sig. $B(N_A)$
Msg 3. $A \rightarrow B$:	Sig. $A(N_B)$

Msg 1. $A \rightarrow C$:	N_A
Msg 2. $C \rightarrow A$:	N_C, Sig. $C(N_A)$
Msg 3. $A \rightarrow C$:	Sig. $A(N_C)$

Msg 1. $C \rightarrow B$:	N_C
Msg 2. $B \rightarrow C$:	N_B, Sig. $B(N_C)$
Msg 3. $C \rightarrow B$:	Sig. $C(N_B)$

Msg 1. $B \rightarrow A$:	N_B
Msg 2. $A \rightarrow B$:	N_A, Sig. $A(N_B)$
Msg 3. $B \rightarrow A$:	Sig. $B(N_A)$

Msg 1. $A \rightarrow B$:	N_A
Msg 2. $B \rightarrow A$:	N_B, Sig. $B(A\|\|N_A)$
Msg 3. $A \rightarrow B$:	Sig. $A(B\|\|N_B)$

Msg 1. $A \rightarrow C$:	N_A
Msg 2. $C \rightarrow A$:	N_C, Sig. $C(A\|\|N_A)$
Msg 3. $A \rightarrow C$:	Sig. $A(C\|\|N_C)$

Msg 1. $C \rightarrow B$:	N_C
Msg 2. $B \rightarrow C$:	N_B, Sig. $B(C\|\|N_C)$
Msg 3. $C \rightarrow B$:	Sig. $C(B\|\|N_B)$

Msg 1. $B \rightarrow A$:	N_B
Msg 2. $A \rightarrow B$:	N_A, Sig. $A(B\|\|N_B)$
Msg 3. $B \rightarrow A$:	Sig. $B(A\|\|N_A)$

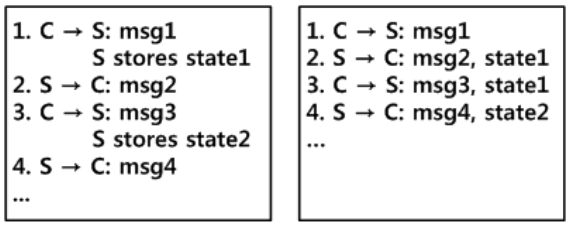
Msg 1. $A \rightarrow B$:	N_A	N_A
Msg 2. $B \rightarrow A$:	N_B, Sig. $B(A\|\|N_A)$	N_B, Sig. $B(2\|\|A\|\|N_A)$	N_B, Sig. $B(A\|\|N_A)$
Msg 3. $A \rightarrow B$:	Sig. $A(B\|\|N_A\|\|N_B)$	Sig. $A(3\|\|B\|\|N_B)$	Sig. $A(N_B\|\|B)$

서비스 거부 공격 (1/4)

- 서비스 거부(DoS, Denial of Service) 공격: 적법한 사용자가 프로토콜을 수행할 수 없도록 만드는 공격
  - 서비스 거부 공격의 분류
    - 분류 1. 자원 소모 공격(resource depletion attack): 서버 또는 클라이언트 시스템의 자원을 소모하기 위한 공격
      - 모바일 단말과 같이 고정된 전원을 사용하지 않는 단말에 대해서는 전원 소모 공격이 큰 문제가 될 수 있음
    - 분류 2. 연결 소모 공격(connection depletion attack): 서버 또는 클라이언트가 허용하는 연결을 고갈하기 위한 공격
      - 자원 소모 공격의 한 종류로 볼 수 있음
- 서버스 거부 공격이 일어나지 않도록 원천적으로 차단하는 것은 가능하지 않음
- 일반적으로 암호프로토콜의 설계를 통해 방어할 수 있는 공격이 아님

서비스 거부 공격 (2/4)

- DoS 공격의 피해를 줄이기 위한 방법 1. 비상태기반(stateless) 프로토콜 사용 (Aura와 Nikander가 제안, 1997)
  - 서버가 접속된 클라이언트의 연결 상태 정보를 유지하기 위한 공간은 본질적으로 한정되어 있으며, 이 공간이 고갈되면 더 이상 클라이언트의 접속을 허용할 수 없음
  - 서버 대신에 클라이언트가 연결 상태 정보를 보관하도록 하면 이 문제를 극복할 수 있음
    - 서버 분산에도 효과적임
    - 통신 비용과 계산 비용은 높아짐
  - 예) 쿠키의 사용
    - 로그인한 사용자 정보



상태기반 프로토콜 비상태기반 프로토콜

서비스 거부 공격 (3/4)

Msg 1. $C \rightarrow S:$ $\mathbf{msg}_1$

Msg 2. $S \rightarrow C:$ $\mathbf{msg}_2, \mathbf{state}_1, \mathbf{MAC}.K_S(\mathbf{state}_1)$

Msg 3. $C \rightarrow S:$ $\mathbf{msg}_3, \mathbf{state}_1, \mathbf{MAC}.K_S(\mathbf{state}_1)$

⋮

(1) 무결성만 보장하는 방법

- 상태 정보를 보호하기 위해 사용하는 대칭키들은 서버만 알고 있음
- 상태 정보의 구성을 통해 재전송 공격을 방어할 수 있음
- 추가적으로 키를 자주 변경하여 재전송 공격에 대해 더 강건하도록 만들 수 있음

Msg 1. $C \rightarrow S:$ $\mathbf{msg}_1$

Msg 2. $S \rightarrow C:$ $\mathbf{msg}_2, \{\mathbf{state}_1\}.K_1, \mathbf{MAC}.K_2(\{\mathbf{state}_1\}.K_1)$

Msg 3. $C \rightarrow S:$ $\mathbf{msg}_3, \{\mathbf{state}_1\}.K_1, \mathbf{MAC}.K_2(\{\mathbf{state}_1\}.K_1)$

⋮

(2) 비밀성과 무결성을 모두 보장하는 방법

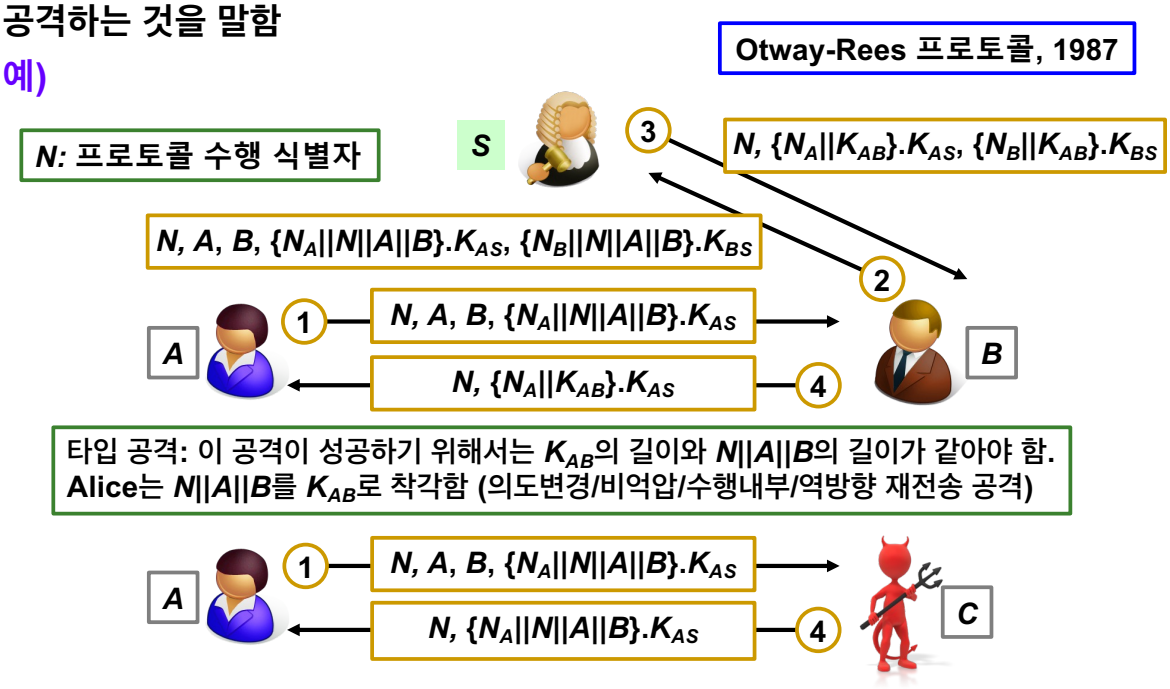
서비스 거부 공격 (4/4)

- **DoS 공격의 피해를 줄이기 위한 방법 2.** 프로토콜의 모든 메시지를 인증함
  - 공격자와 정상 사용자를 구분하는 것이 목적
  - 메시지를 인증하기 위한 비용이 소요되므로, 점진적으로 프로토콜이 진행됨에 따라 확인하기 위한 비용을 높이 것이 더 효과적임
  - 이를 위해 클라이언트 퍼즐이라는 것을 활용하기도 함
    - 하나의 퍼즐을 해결하는 것은 비용이 높지 않지만 이것을 여러 개 해결하고자 하면 비용이 상대적으로 높아지는 문제
      - 예) 해시퍼즐 (강의노트 11)
    - 오늘날 분산 서비스 거부 공격의 형태를 고려하였을 때 퍼즐을 사용하는 효과가 별로 없음

타입 공격

- 타입 공격(typing attack)은 메시지의 요소를 잘못 해석하도록 하여 공격하는 것을 말함
- 예)

Otway-Rees 프로토콜, 1987



타입 공격: 이 공격이 성공하기 위해서는 K_{AB}의 길이와 $N||A||B$의 길이가 같아야 함.
Alice는 $N||A||B$를 K_{AB}로 착각함 (의도변경/비억압/수행내부/역방향 재전송 공격)

크기가 달라도 $\{N_A||N||A||B\}.K_{AS}$를 $\{N_A||K_{AB}\}.K_{AS}$ 크기와 같도록 앞 부분만 활용하여 공격하는 것도 사용한 암호화 모드에 따라 가능할 수 있음

중간자 공격 (1/3)

- 다른 프로토콜에 대해서도 일어날 수 있는 공격이지만 **MITM** 공격은 주로 기본 **Diffie-Hellman** 프로로콜의 취약점을 설명할 때 언급되는 공격임

Diffie-Hellman 프로토콜에 대한 중간자 공격(man-in-the-middle attack)

$S_{AB} = (g^c)^a = g^{ca}$ $S_{AB} = (g^a)^c = g^{ac}$ $S_{BA} = (g^b)^c = g^{bc}$ $S_{BA} = (g^c)^b = g^{cb}$

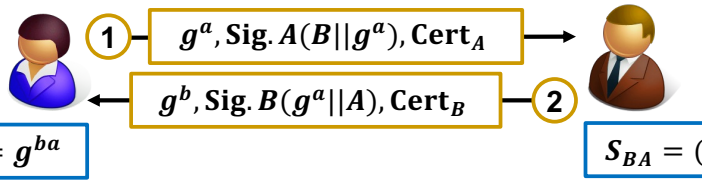
$K = f(S_{AB})$ $K' = f(S_{BA})$

- Alice는 K를 Bob와 공유한 키로 착각하게 되며,
- Bob은 K'를 Alice와 공유한 키로 착각하게 됨

- 공격자가 K와 K'을 계산할 수 있어야 유효한 중간자 공격이 됨
- 중간자는 키 확립 이후 교환되는 메시지를 볼 수 있어야 하며, 이를 참여자들이 인식할 수 없어야 함

중간자 공격 (2/3)

강한 개체 인증

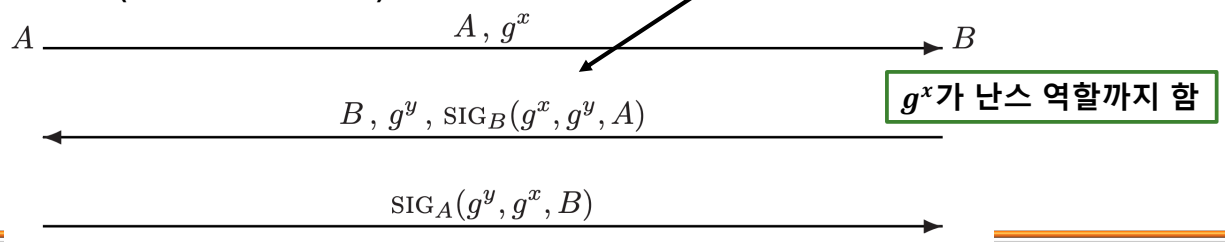


$S_{AB} = (g^b)^a = g^{ba}$

$S_{BA} = (g^a)^b = g^{ab}$

- 중간자 공격은 교환하는 값을 인증하지 않아 가능한 공격임
- 교환하는 값을 인증하면 이 공격을 방어할 수 있음
- 각 사용자는 기존과 달리 전자서명을 생성하는 비용과 확인하는 비용이 추가되며, 메시지의 크기도 증가함
- 공격자가 g^a에서 a를 알게 되면 재전송 공격을 할 수 있음. B는 메시지 1의 최근성을 확인할 수 없음

ISO 표준 (ISO/IEC IS 9798-3)

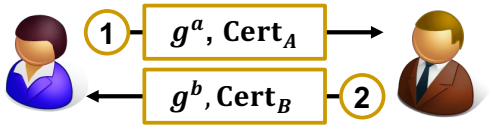


g^x가 난스 역할까지 함

중간자 공격 (3/4)

Matsumoto et al. 프로토콜(MTI 프로토콜), 1986

$S_{AB} = (g^{x_B})^a (g^b)^{x_A}$
$= g^{x_B a} g^{b x_A} = g^{a x_B + b x_A}$



$S_{BA} = (g^a)^{x_B} (g^{x_A})^b$
$= g^{a x_B} g^{x_A b} = g^{a x_B + b x_A}$

$y_A = g^{x_A}$

$y_B = g^{x_B}$

y_A: Alice의 공개키
x_A: Alice의 개인키
Cert_A: Alice와 y_A를 바인딩해주는 인증서

- 세션키를 계산하는 방법을 바꾸어 공격자가 중간자 공격을 하더라도 세션키를 계산할 수 없도록 함
- 이를 위해 장기간 개인키를 키 계산에 사용함

$S_{AB} = (g^{x_B})^a (g^c)^{x_A}$
$= g^{a x_B + c x_A}$

$S'_{AB} = (g^{x_B})^? (g^{x_A})^c$

$S_{BA} = (g^{x_B})^c (g^{x_A})^?$

$S_{BA} = (g^c)^{x_B} (g^{x_A})^b$
$= g^{c x_B + b x_A}$

중간자 공격 (4/4)

- 중간자 공격을 방어하는 두 가지 대표적인 방법
  - 방법 1. 교환하는 값을 인증(보통 전자서명)할 수 있도록 함
    - 교환되는 값을 누가 생성하였는지 확인할 수 있도록 하여 방어함
  - 방법 2. MTI 프로토콜처럼 세션키를 계산하는 방법을 제한함
    - 중간자는 계산할 수 없지만 각 참여자들은 계산할 수 있도록 하기 위해 보통 참여자의 장기간 개인키를 세션키 계산에 포함함
  - 두 방법 모두 양 참여자가 장기간 공개키 쌍을 가지고 있어야 함. 특히, MTI는 같은 군에서 장기간 공개키 쌍이 필요함
- 방법 2가 방법 1보다 통신 비용, 계산 비용이 효율적임

	방법1	방법2
교환 메시지	g^a, 서명값, 인증서*	g^a, 인증서*
계산비용	인증서 확인 비용 서명, 서명확인	인증서 확인 비용 세션키 계산 비용 증가

  - 안전성은 방법 1이 더 우수함 (전방향 안전성)

SIGMA (1/2)

- SIGMA(SIGn-and-MAc) 프로토콜

$A \xrightarrow{\quad g^x \quad} B$

$A \xleftarrow{\quad g^y\,,\ B\,,\ \mathrm{SIG}_B(g^x, g^y)\,,\ \mathrm{MAC}_{K_m}(B) \quad}$

$A \xrightarrow{\quad A\,,\ \mathrm{SIG}_A(g^y, g^x)\,,\ \mathrm{MAC}_{K_m}(A) \quad}$

- 강한 개체 인증을 제공하지 않음
- 대신 MAC을 통해 중간자가 이름을 조작할 수 없도록 함

$A \xrightarrow{\quad g^x \quad} B$

$A \xleftarrow{\quad g^y\,,\ \{B\,,\ \mathrm{SIG}_B(g^x, g^y)\,,\ \mathrm{MAC}_{K_m}(B)\}_{K_e} \quad}$

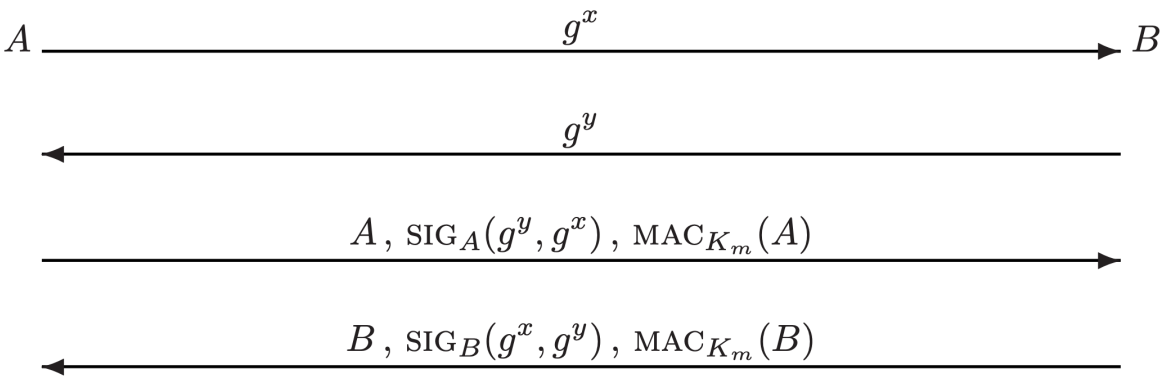
$A \xrightarrow{\quad \{A\,,\ \mathrm{SIG}_A(g^y, g^x)\,,\ \mathrm{MAC}_{K_m}(A)\}_{K_e} \quad}$

K_c, K_m은 g^{xy}로부터 계산된 키

- 식별자의 노출을 보호 (identity protection)
- 수동 공격에 대해서는 *A*, *B*를 모두 보호
- 능동 공격에 대해서는 개시자만 보호

SIGMA (2/2)

- TLS1.3에서 이와 유사하게 키 확립을 하고 있음

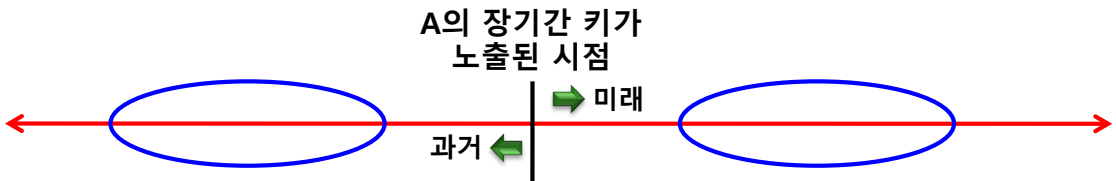


- 참여자 중 한 쪽만 장기간 공개키 쌍을 가지고 있으면?
  한 참여자만 전자서명하면 중간자 공격을 방어할 수 있는가?
- 실제 TLS에서는 웹서버만 인증서를 보유하고 있음
  (Note 10)

장기간 키의 노출 (1/3)

장기간 키는 해당 사용자를 인증할 때 사용하는 암호키임

- 공격자가 하고 싶은 것



전방향 안전성
목표: A가 사용한 과거 세션키

키 노출 위장 공격
목표: A에게 다른 사용자로 접근

공격자는 A 행세를 할 수 있음
⇒ 이것을 막을 수는 없음
⇒ 키 갱신을 빨리하는 것이 유일한 방어 수단

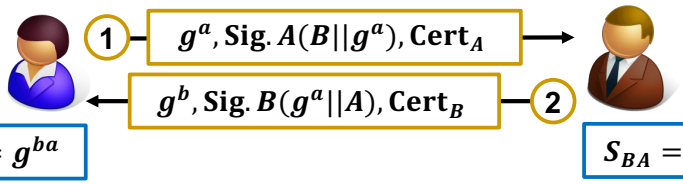
A가 앞으로 사용할 세션키는?
(후방향 안전성 개념)

- 키가 노출되면 해당 키로 암호화하여 교환한 정보의 노출될 수 있음
- 키가 노출되더라도 그 피해(노출되는 정보, 파급효과)를 최소화하는 것이 설계 목표

장기간 키의 노출 (2/3)

$\{B||N_A||K_{AB}\}.K_{AS}$, $\{A||N_B||K_{AB}\}.K_{BS}$

- 장기간 키의 노출 관련 요구사항
  - 전방향 안전성(forward secrecy): 참여하는 일부 참여자의 장기간 키가 노출되어도 과거의 세션키가 노출되지 않는 경우
    - 키 전송 프로토콜의 경우에는 보통 장기간 키로 세션키를 암호화하여 교환하므로 기본적으로 전방향 안전성을 제공할 수 없음
  - 완벽한 전방향 안전성(perfect forward secrecy): 참여하는 모든 참여자의 장기간 키가 노출되어도 과거의 세션키가 노출되지 않는 경우
    - 예)



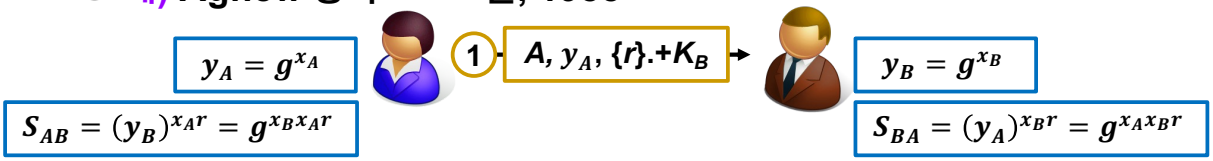
    - 여기서 장기간 키는 서명에 사용하는 개인키임
    - 두 사용자의 개인키가 모두 노출되더라도 S_{AB}를 계산할 수 없으므로 완벽한 전방향 안전성이 보장됨

MTI: $g^{ax_B+bx_A}$

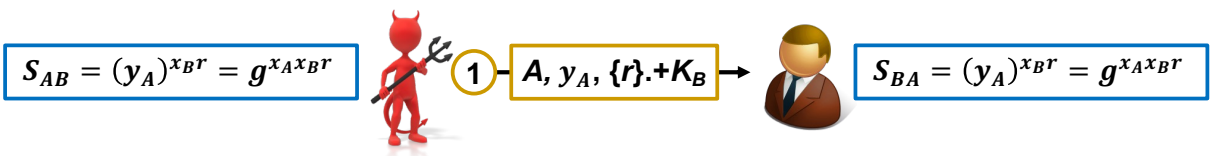
장기간 키의 노출 (3/3)

- 장기간 키의 노출 관련 요구사항 (계속)
  - 키 노출 위장 공격(key compromise impersonation attack). 노출된 사용자의 장기간 키를 이용하여 노출된 사용자에게 다른 사용자로 위장하여 접근하는 공격
    - 주로 키 동의 프로토콜에서 나타나는 공격
    - 예) Agnew 등의 프로토콜, 1988

장기간 키 + 랜덤값



    - Bob의 장기간 키를 알고 있는 사용자는 임의의 사용자로 위장하여 Bob에게 접근을 할 수 있음



Bob의 장기간 키를 확보하면 Bob이 할 수 있는 것의 대부분을 공격자도 할 수 있음

세션키의 노출 (1/2)

- 세션키의 노출
  - 기지 키 공격(known-key attack): 노출된 과거의 세션키를 이용하여 공격하는 방법
    - 노출된 과거 세션키가 교환되었던 프로토콜 트랜스크립트를 이용하여 공격함
    - 목적
      - 노출된 과거 세션키를 새 세션키로 사용하도록 함
      - 노출된 과거 세션키를 이용하여 다른 세션키를 계산함
        - 세션키 값 간에 독립성이 보장되면 이와 같은 공격은 가능하지 않음
- 독립성이 보장되지 않는 경우
  - 예) K가 현재 세션키일 때 다음 세션키는 $H(K)$를 사용함
    - 특정 세션키가 노출되더라도 과거 세션키는 노출되지 않음
    - 매번 새로운 독립적으로 세션키를 확립하면 특정 세션키를 얻더라도 과거나 미래 세션키를 계산할 수 없음

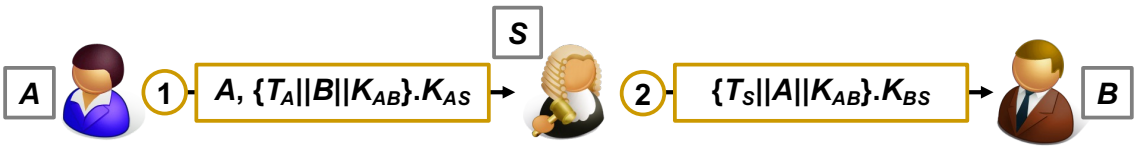
세션키의 노출 (2/2)

- 세션키가 독립적이지 않는 특수 환경, 응용의 등장으로 완벽 전방향 안전성(PFS)이 최근에는 다른 의미로 사용됨 (예: 메신저 보안)
- 완벽한 전방향 안전성
  - 세션키가 노출되더라도 해당 세션키보다 과거에 확립한 세션키는 노출되지 않음 (이전 개념은 장기간 키의 노출)
    - 이전 슬라이드의 해시함수를 이용한 세션키 갱신은 완벽한 전방향 안전성을 제공함
- 미래 안전성(future secrecy)
  - 세션키가 노출되더라도 미래 세션키는 노출되지 않음

프로토콜 상호작용 (1/3)

- 장기간 키가 여러 프로토콜에서 사용할 경우에는 한 쪽 프로토콜을 이용하여 다른 프로토콜을 공격할 수 있음
- 장기간 키도 사용하는 응용을 제한해야 한다는 것을 의미함
- 예)

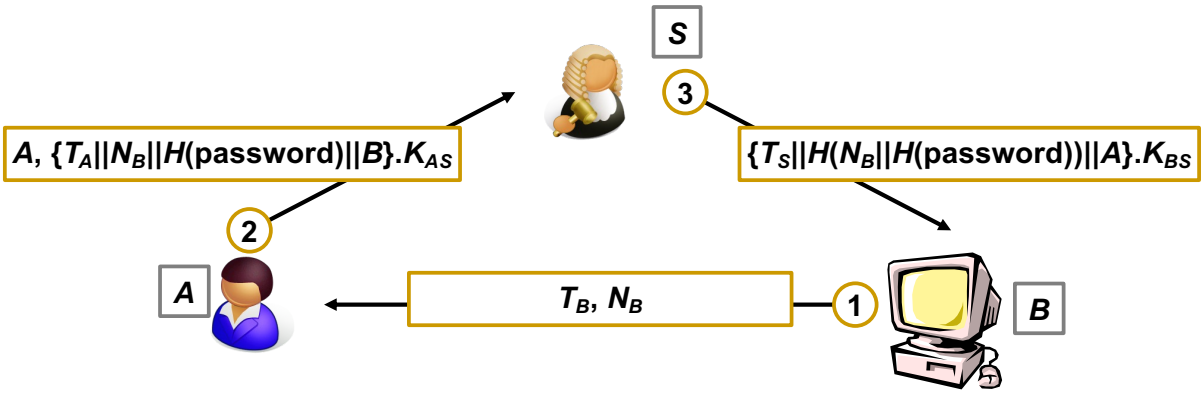
Wide-Mouthed-Frog 프로토콜, 1990, type T1



  - 문제점. Bob은 세션키 생성에 대해 Alice를 신뢰해야 함

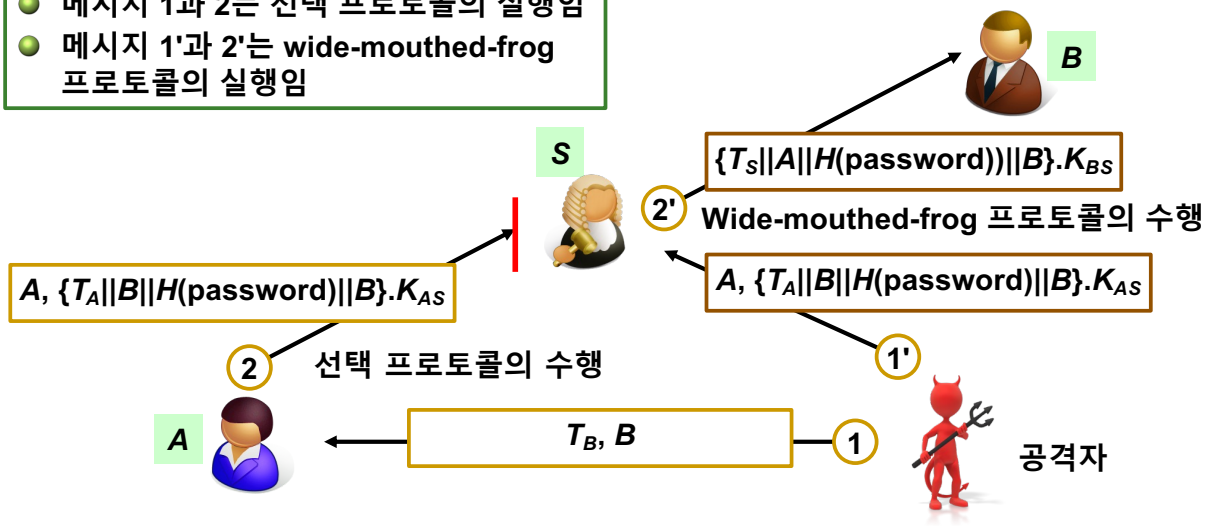
프로토콜 상호작용 (2/3)

- Kesley 등이 제안한 wide-mouthed-frog 프로토콜에 대한 선택 프로토콜 (chosen-protocol) 공격, 1998
  - Wide-mouthed-frog 프로토콜을 공격하기 위한 다음과 같은 선택 프로토콜을 만들어 두 프로토콜을 동시에 사용하도록 함
    - 목적. Alice는 *B* 시스템에 안전하게 로그인하고 싶음



프로토콜 상호작용 (3/3)

- 메시지 1과 2는 선택 프로토콜의 실행임
- 메시지 1'과 2'는 wide-mouthed-frog 프로토콜의 실행임



- Bob은 $H(password)||B$를 K_{AB}로 착각하게 됨 (타입 공격)
- 공격자가 이 정보를 알고 있으면 Alice로 위장하여 Bob과 세션키 확립에 성공하게 됨. 즉, 두 개의 프로토콜에서 동일한 암호키를 사용하면 다양한 공격이 가능함