정보보호개론

제7장 암호프로토콜 공격 방법

1. 암호프로토콜에 대한 공격

암호프로토콜에 대한 공격은 매우 다양하기 때문에 지금까지 알려진 공격에 대해 암호프로토콜이 강건함을 보이는 것만으로는 완벽하게 안전하다고 주장할 수 없다. 하지만 이미 알려진 공격 방법에 대해 안전하다는 것이 의미가 없다는 것은 아니다. 알려진 공격 방법을 고려하여 프로토콜을 설계해야 명백한 허점을 만드는 일을 줄일 수 있다.

프로토콜마다 목적이 다르므로 프로토콜에 대한 특정 공격의 의미가 다를 수 있다. 더욱이 어떤 공격에 대해 특정 프로토콜이 취약하더라도 해당 프로토콜 요구사항에 어떤 영향도 주지 못할 수 있다. 이 경우 특정 프로토콜은 그와 같은 공격을 설계에 고려하지 않아도 된다. 이 장에서는 암호프로토콜에 대한 다양한 공격을 살펴본다. 주로 키 확립 프로토콜에 대한 공격을 살펴보지만 이들 공격은 다른 다양한 암호프로토톨에서도 발생 가능한 공격이다.

2. 제전송 공격

암호프로토콜에 대한 공격은 크게 수동 공격과 능동 공격으로 분류되며, 능동 공격 중 재전송 공격이 초기에는 가장 위협이 되는 공격이었다. 재전송 공격이란 현재 프로토콜 수행을 포함하여 사용된 메시지의 일부 또는 전체를 다시 전송하여 공격하는 것을 말한다. 재전송 공격은 크게 다음과 같은 속성을 기준으로 분류하여 분석할 수 있다.

- 억압(suppressed) 여부: 공격에 이용할 메시지를 원래 수신자가 받은 적이 있을 수 있고, 공격자가 차단하여 받은 적이 없을 수 있다.
- 변경 여부: 메시지를 재전송할 때 원래 메시지를 그대로 전송할 수 있고, 일부 내용을 변경하여 전송할 수 있다.
- 의도변경: 원 메시지와 동일한 효과를 위해 재전송할 수 있고, 다른 효과를 위해 재전송할 수 있다.
- 순방향, 역방향: 메시지를 원래 수신자에게 재전송하면 순방향이고, 거꾸로 송신자에게 전송하면 역방향이 된다.
- 수행 내부, 수행 외부: 한 프로토콜 수행에 사용된 메시지를 해당 수행에서 사용하면 수행 내부 공격이고, 다른 프로토콜 수행에 사용된 메시지를 이용하면 수행 외부 공격이다.
- 전통, 중첩: 공격에 성공하기 위해 두 개의 프로토콜을 병행으로 수행하여야 하면 중첩(interleaved) 공격이고, 병행 수행이 필요 없으면 전통 공격에 해당한다.

재전송 공격은 프로토콜의 종류가 같아야만 가능한 것은 아니다. 프로토콜이 다르더라도 같은 암호키를 사용하면 한 프로토콜의 메시지를 전혀 다른 프로토콜을 공격하는데 사용할 수 있다. 재전송과 달리 앞으로 사용할 메시지 또는 메시지의 일부를 사용하여 프로토콜을 공격하는 것을 사전전송(preplay) 공격이라 한다.

$$
\begin{array}{lll}
\text{Msg 1.} & A \rightarrow B: & A, \{N_A\}.K_{AB} \\
\text{Msg 2.} & B \rightarrow A: & \{N_B\}.K_{AB}, N_A \\
\text{Msg 3.} & A \rightarrow B: & N_B
\end{array}
$$

<그림 7.1> 재전송 공격에 취약한 상호 인증 프로토콜

$$
\begin{array}{lll}
\text{Msg 1.} & A \rightarrow B: & A, \{N_A\}.K_{AB} \\
\text{Msg 1'.} & C \rightarrow A: & B, \{N_A\}.K_{AB} \\
\text{Msg 2'.} & A \rightarrow B: & \{N'_A\}.K_{AB}, N_A \\
\text{Msg 2.} & C \rightarrow A: & \{N'_A\}.K_{AB}, N_A \\
\text{Msg 3.} & A \rightarrow B: & N'_A \\
\text{Msg 3'.} & C \rightarrow A: & N'_A
\end{array}
$$

<그림 7.2> 그림 7.1의 프로토콜에 대한 재전송 공격

메시지가 억압되면 단순 중복 검사를 통해 재전송 공격을 방어할 수 없다. 보통 기존 메시지를 보관하는 비용과 검사 비용 때문에 재전송 공격을 방어하기 위해 중복 검사를 사용하지는 않는다. 이보다는 메시지의 최근성을 보장하는 기법을 사용하는 것이 더 효율적이며 효과적이다. 역방향 공격이 가능하기 위해서는 주고 받는 암호문의 형태가 같아야 한다. 같은 암호키를 여러 프로토콜에 사용하면 다른 프로토콜의 메시지를 활용하는 수행 외부 공격이 가능할 수 있다. 이 때문에 역방향, 수행 내부, 수행 외부를 방어하기 위한 간단한 방법은 다음과 같다.

- 방법 1. 프로토콜의 메시지와 각 메시지 내 암호문의 형태를 다르게 한다.
- 방법 2. 각 메시지 암호문 내에 메시지 번호, 메시지 방향, 프로토콜 수행 식별자를 포함한다.
- 방법 3. 각 방향마다 다른 세션키를 사용한다.
- 방법 4. 같은 키를 여러 프로토콜에서 사용하지 않는다.

6장에서 설명하였듯이 최근에는 방향마다 다른 키를 사용하며, 심지어 메시지마다 다른 키를 사용한다. 여기서 방향이란 A와 B 간의 프로토콜을 진행하면 A가 B에게 메시지를 보낼 때와 B가 A에게 보낼 때 다른 키를 사용하는 것을 말한다. 이와 같이 방향마다 메시지마다 다른 키를 사용하면 재전송 공격 방어에 매우 효과적이다. 더욱이 이들 키는 보통 장기간 키가 아니므로 여러 개의 키를 사용한다고 키 관리 비용이 증가하는 것이 아니다.

재전송 공격의 예로 그림 7.1에 제시된 프로토콜을 살펴보자. 이 프로토콜은 난스를 이용하여 상호 인증하는 프로토콜이다. 하지만 메시지 1의 암호문과 메시지 2의 암호문의 형태가 동일하다. 또한 공격자가 메시지를 무조건적으로 복호화 해주는 형태이므로 오라클로 활용할 가능성도 있다.

그림 7.2에 제시된 재전송 공격에서 공격자는 A로부터 B로 가는 모든 메시지를 가로챌 뿐만 아니라 차단한다. 또한 B 행세를 하여 동일 프로코콜을 A와 중첩으로 수행한다. 따라서 이 공격은 억압, 역방향, 수행 외부, 중첩, 재전송 공격이다. 공격자는 메시지 1을 그대로 다시 A에게 전달하여 두 번째 프로토콜을 병행으로 시작한다. 이와 같은 공격이 가능한 이유는 A가 프로토콜을 시작하거나 B가 시작하거나 메시지 1의 형태가 같기 때문이다. 이 문제를 해결하는 가장 단순한 방법은 메시지 1의 암호문에 암호문 생성자나 수신자의 식별자를 포함하면 된다. 예를 들어 $\{B||N_A\}.K_{AB}$처럼 수신자의 식별자를 포함하면 공격자는 메시지 1′과 같이 역방향 재전송 공격에 성공할 수 없다. 또 다른 해결책은 방향마다 다른 키를 사용하면 이 공격을 방어할 수 있다.

$$
\begin{array}{lll}
\text{Msg 1.} & C \rightarrow S: & \text{msg}_1 \\
\text{Msg 2.} & S \rightarrow C: & \text{msg}_2, \text{state}_1, \text{MAC}.K_S(\text{state}_1) \\
\text{Msg 3.} & C \rightarrow S: & \text{msg}_3, \text{state}_1, \text{MAC}.K_S(\text{state}_1) \\
\text{Msg 4.} & S \rightarrow C: & \text{msg}_4, \text{state}_2, \text{MAC}.K_S(\text{state}_2) \\
 & \vdots &
\end{array}
$$

(1) 무결성만 보장하는 방법

$$
\begin{array}{lll}
\text{Msg 1.} & C \rightarrow S: & \text{msg}_1 \\
\text{Msg 2.} & S \rightarrow C: & \text{msg}_2, C_1 = \{\text{state}_1\}.K_1, T_1 = \text{MAC}.K_2(C_1) \\
\text{Msg 3.} & C \rightarrow S: & \text{msg}_3, C_1, T_1 \\
\text{Msg 4.} & S \rightarrow C: & \text{msg}_4, C_2 = \{\text{state}_1\}.K_1, T_2 = \text{MAC}.K_2(C_2) \\
 & \vdots &
\end{array}
$$

(2) 비밀성과 무결성을 모두 보장하는 방법

<그림 7.3> 비상태 기반 프로토콜에서 상태 정보의 보호

3. 서비스 거부 공격

서비스 거부 공격은 가용성에 대한 공격이며, 종종 매체 기사를 통해 접하는 공격이다. 예를 들어 2014년에 소니사가 북한 김정은 암살 영화인 “디 인터뷰” 상영하고자 하자 소니사 운영 사이트가 서비스 거부 공격을 받아 다운된 적이 있으며, 2009년 3월 4일에는 청와대 등 40곳이 디도스 공격을 받은 사례도 있다.

서비스 거부 공격은 암호프로토콜을 통해 방어할 수 있는 공격은 아니다. 또한 다른 메커니즘을 이용하더라도 원천적으로 방어하기 힘들다. 서비스 거부 공격은 크게 자원 소모 공격(resource depletion attack)과 연결 소모 공격(connection depletion attack)으로 분류할 수 있다. 자원 소모 공격이란 서버 또는 클라이언트 시스템의 자원을 소모하여 서비스를 제공할 수 없도록 하는 공격이다. 특히, 모바일 단말과 같이 고정된 전원을 사용하지 않는 단말들에 대해서는 전원 소모 공격이 큰 문제가 될 수 있다. 물론 서비스 거부 공격은 클라이언트가 아니라 보통 서버에 대한 공격이다. 연결 소모 공격이란 서버 또는 클라이언트가 허용하는 연결을 고갈하기 위한 공격이다. 연결도 또 다른 자원이라고 보면 연결 소모도 자원 소모 공격의 한 종류로 볼 수 있다. 참고로 연결 시도는 유효한 시도인지 판단하기 위한 계산이 필요하며, 이 계산 비용이 높으면 공격을 식별하기 위한 비용이 높아져 서비스 거부 공격이 더욱 쉬워질 수 있다.

서비스 거부 공격은 앞서 언급한 바와 같이 암호프로토콜의 설계만으로는 방어하기 어렵다. 하지만 암호프로토콜을 설계할 때 서비스 거부 공격에 대해 고려할 수 있는 부분도 있다. 암호프로토콜에서 사용하는 몇 가지 기법을 소개하면 다음과 같다.

첫째, 상태 기반 프로토콜을 비상태(stateless) 기반으로 바꾸는 것이다[1]. 보통 서버는 프로토콜을 수행하면서 현재 어디까지 수행하고 있는지 등을 알기 위해 프로토콜 수행과 관련된 상태 정보를 유지한다. 그런데 서버가 접속된 클라이언트의 연결 상태 정보를 유지하기 위한 공간은 본질적으로 한정되어 있으며, 이 공간이 고갈되면 더는 클라이언트의 접속을 허용할 수 없게 된다. 연결 소모 공격은 이것을 목표로 하는 공격이다. 따라서 서버 대신에 클라이언트가 연결 상태 정보를 보관하도록 하면 이 문제를 극복할 수 있다. 현재 웹 브라우징할 때 클라이언트가 보관하는 쿠키 정보가 이와 유사한 역할을 한다. 프로토콜이 비상태 기반이 되면 서버를 분산(여러 개 사용)하는 측면에서도 유리하다. 하지만 비상태 기반에서 상태는 클라이언트가 유지하여야 하며, 현재 상태를 매번 서버에 전달해야 하므로 통신 비용과 계산 비용이 커지는 단점이 있다.

상태 정보는 그림 7.3처럼 암호기술을 사용하여 보호할 수 있다. 그림 7.3.(1)은 MAC만 사용하여 무결성을 제공하는 기법이고, 그림 7.3.(2)는 인증 암호화를 이용하여 비밀성과 무결성을 모두 제공하는 기법이다. 두 경우 모두 상태 정보와 MAC은 서버만 확인하고 서버만 내용을 알면 되기 때문에 여기에서 사용하는 비밀키는 클라이언트를 포함하여 누구와도 공유할 필요가 없다.

Msg 1. $A \rightarrow B:\quad N, A, B, \{N_A||N||A||B\}.K_{AS}$
Msg 2. $B \rightarrow S:\quad N, A, B, \{N_A||N||A||B\}.K_{AS}, \{N_B||N||A||B\}.K_{BS}$
Msg 3. $S \rightarrow B:\quad N, \{N_A||K_{AB}\}.K_{AS}, \{N_B||K_{AB}\}.K_{BS}$
Msg 4. $B \rightarrow A:\quad N, \{N_A||K_{AB}\}.K_{AS}$

<그림 7.4> Otway와 Rees 프로토콜

상태에 타임스템프를 포함하여 재전송 공격에 강건하도록 만들 수 있다. 세션별로 새로운 비밀키를 생성하여 한 세션에 사용한 상태 정보를 다른 세션에 사용할 수 없게 만들 수 있다. 하지만 각 접속마다 다른 키를 사용한다는 것이기 때문에 이 키는 세션 정보에 해당하므로 비상태 기반 프로토콜의 목적에 맞지 않는 방법이다. 오히려 상태 정보를 보호하기 위해 사용하는 비밀키를 자주 갱신하는 것이 더 좋은 방법이다.

둘째, 무의미한 서비스 요청을 할 수 없도록 요청을 인증하는 방법을 사용할 수 있다. 하지만 인증하는 비용이 많이 들면 이것은 오히려 서버스 거부 공격을 유리하게 만들 수 있다. 따라서 프로토콜이 진행됨에 따라 점진적으로 인증 비용을 높여가는 방식도 사용한다. 이를 위해 클라이언트 퍼즐을 사용하기도 한다. 클라이언트 퍼즐은 간단한 문제이지만 그것을 여러 개 해결하기 위해서는 비용이 상대적으로 높아지는 문제를 말한다. 하지만 오늘날 분산 서비스 거부 공격은 하나의 컴퓨터를 이용하여 다른 컴퓨터를 공격하는 것이 아니기 때문에 퍼즐을 사용하는 효과가 크지 않다.

4. 타입 공격

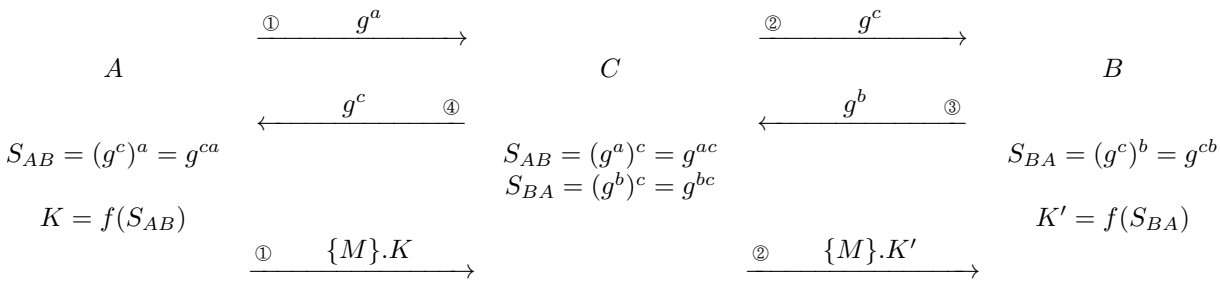
그림 7.4는 Otway와 Rees가 제안한 키 전송 프로토콜이다[2]. 여기서 N은 프로토콜 수행 식별자이다. 메시지 3의 두 암호문 내에는 키 용도를 나타내는 식별자가 포함되어 있지 않다. 하지만 메시지 2의 암호문을 통해 각 난스가 식별자와 바인딩 되어 있으므로 메시지 3의 암호문에서는 생략이 가능하다. 이미 어떤 정보(A, B)와 바인딩된 요소(N_A)만 포함하고, 해당 정보(A, B)를 포함하지 않더라도 그 정보(A, B)까지 바인딩할 수 있다.

타입 공격이란 메시지의 요소를 잘못 해석하도록 하여 공격하는 것을 말한다. 그림 7.4에 제시된 프로토콜에 대해 타입 공격이 가능하기 위해서는 $N||A||B$의 길이와 K_{AB}의 길이가 같아야 한다. 이들의 길이가 같으면 암호문 $\{N_A||N||A||B\}.K_{AS}$와 $\{N_A||K_{AB}\}.K_{AS}$의 길이가 같아진다. 따라서 공격자가 메시지 4를 차단하고 대신에 $\{N_A||N||A||B\}.K_{AS}$를 전송하면 A는 $N||A||B$를 세션키로 착각하게 되며, 공격자는 이 값들을 알고 있으므로 이후 교환되는 비밀 메시지가 노출될 수 있다.

이와 같은 문제는 암호문의 길이를 다르게 하거나, 암호문마다 메시지 번호를 추가하여 방어할 수 있다. 또 방향마다 다른 키를 사용하여도 방어할 수 있다. 그런데 암호문의 길이가 다른 것만으로는 충분한 방어가 안 될 수 있다. ECB 모드를 사용하지 않겠지만 이 프로토콜에서 ECB 모드를 사용하여 암호화하면 블록 크기, $N||A||B$의 길이에 따라 여전히 공격이 가능하다. 이것이 어떻게 가능한 것인지는 연습문제를 통해 고민하여 보자.

5. 중간자 공격

중간자(man-in-the-middle) 공격은 프로토콜을 수행하는 참여자들 중간에 공격자가 자리 잡고 능동 공격을 하고 있지만, 참여자들은 본인들 사이에 어떤 공격자가 능동 공격을 하고 있다는 것을 인식하지 못하는 공격을 말한다. 기본 Diffie-Hellman 공격은 그림 7.5에 제시된 것처럼 중간자 공격에 취약하다. 공격자는 주고받는 메시지를 중간에서 차단하고, 두 사용자에게 모두 g^c를 전달하면 A와 B는 모두 키를 g^c를 이용하여 계산하게 된다. 공격자는 c를 알고 있기 때문에 각 참여자가 계산한 키를 계산할 수 있다. 공격자는 이를 이용하여 A와 B가 교환하는 모든 메시지를



<그림 7.5> Diffie-Hellman에 대한 중간자 공격

Msg 1. $A \rightarrow B:$ A, g^a, N_A
Msg 2. $B \rightarrow A:$ $B, g^b, N_B, \text{Sig.}B(g^b||N_A||A)$
Msg 3. $A \rightarrow B:$ $\text{Sig.}A(B||N_B||g^a)$

$$S_{AB} = (g^b)^a = g^{ba}, \quad S_{BA} = (g^a)^b = g^{ab}$$

<그림 7.6> MITM에 강건한 Diffie-Hellman 키 동의 프로토콜

가로채어 모든 메시지를 볼 수 있으며, 이를 다시 암호화하여 보내 중간에 자신이 있다는 것을 인식하지 못하게 할 수 있다. 이처럼 중간자 공격이 유효한 공격이 되기 위해서는 두 참여자가 계산하게 되는 세션키를 중간자도 계산할 수 있어야 한다.

Diffie-Hellman에 대한 중간자 공격은 교환되는 메시지를 인증하지 않았기 때문에 가능한 공격이다. 이를 방어하는 방법은 크게 두 가지다. 하나는 그림 7.6에 제시된 것처럼 교환되는 메시지를 인증하는 것이고, 다른 하나는 그림 7.7에 제시된 것처럼 세션키를 계산하는 방법을 바꾸어 중간자 공격이 있더라도 중간자가 키를 계산할 수 없도록 하는 것이다[4].

보통 키 동의 프로토콜에서 각 참여자가 키 계산에 기여하는 값을 충분히 랜덤하게 선택하였다면 각 사용자는 자신이 생성한 키 값의 최근성을 믿을 수 있다. 이 때문에 키 전송 프로토콜과 달리 키 최근성을 보장하는 별도 메커니즘을 사용할 필요가 없다고 생각할 수 있다. 하지만 아래와 같이 프로토콜을 구성하게 되면 이 허점을 활용한 공격이 가능하다.

Msg 1. $A \rightarrow B:$ $A, g^a, \text{Sig.}A(B||g^a)$
Msg 2. $B \rightarrow A:$ $B, g^b, \text{Sig.}B(g^b||A)$

공격자가 g^a에서 a를 우연히 알게 되면 공격자는 메시지 1을 재전송하여 이 프로토콜을 수행할 수 있으며, 이 공격자는 확립된 세션키를 이용하여 B에게 A 행세를 할 수 있다.

그림 7.6에 제시된 프로토콜은 공격자가 a를 우연히 알게 되더라도 B가 제시한 난스가 포함된 메시지 3의 서명을 만들 수 없어 이 공격에 대해 강건하다. 따라서 전자서명을 통해 상호인증하는 것만으로 부족하고, 서명의 최근성이 보장되어야 한다. 국제 표준은 별도 난스를 사용하지 않고 그림 7.8과 같이 g^a와 g^b를 난스로 활용하고 있다 [3].

Msg 1. $A \rightarrow B:$ A, g^a
Msg 2. $B \rightarrow A:$ B, g^b

$$S_{AB} = (g^b)^{x_A}(y_B)^a = g^{bx_A + x_B a}, \quad S_{BA} = (g^a)^{x_B}(y_A)^b = g^{ax_B + x_A b}$$

<그림 7.7> Matsumoto 등의 Diffie-Hellman 키 동의 프로토콜

Msg 1. $A \rightarrow B:$ A, g^a
Msg 2. $B \rightarrow A:$ $B, g^b, \text{Sig.}B(g^b||g^a||A)$
Msg 3. $A \rightarrow B:$ $\text{Sig.}A(g^a||g^b||B)$

<그림 7.8> DH 키 동의 프로토콜 ISO 표준

Msg 1. $A \rightarrow B:$ g^a
Msg 2. $B \rightarrow A:$ $B, g^b, \text{Sig.}B(g^a||g^b), \text{MAC.}K(B)$
Msg 3. $A \rightarrow B:$ $A, \text{Sig.}A(g^b||g^a), \text{MAC.}K(A)$

<그림 7.9> 기본 SIGMA 프로토콜

그림 7.7에서 A의 개인키는 x_A이고 공개키는 $y_A = g^{x_A}$이다. 이 프로토콜에서는 세션키를 계산할 때 장기간 키와 이번 세션에서 만든 랜덤 정보를 함께 사용한다. 공격자는 각 사용자의 장기간 키를 모르기 때문에 중간자 공격을 하더라도 두 사용자가 계산하는 세션키를 계산할 수 없다. 이 방식은 최근성과 관련된 어떤 조처도 하고 있지 않다. 하지만 앞서 살펴본 것처럼 공격자가 사용자들이 기존에 사용한 g^a의 이산대수를 알게 되더라도 장기간 개인키까지 확보하지 않는 이상 재전송 공격을 하여 성공할 수 없다. 하지만 안전성이 장기간 키에 의존하기 때문에 두 사용자의 장기간 키가 모두 노출되면 세션키를 계산을 할 수 있다.

두 방식을 비교하면 인증 방식은 통신 비용이 증가한 반면에 Matsumoto 등의 방식과 달리 계산 비용은 변하지 않은 것으로 생각할 수 있다. 하지만 인증 방식에서는 서명을 생성하고 확인하는 비용이 추가되었기 때문에 오히려 전체적인 계산 비용은 Matsumoto 등의 방식보다 더 많이 요구된다. 하지만 안전성 측면에서는 인증 방식은 다음 절에서 설명하는 완벽한 전방향 안전성을 제공하기 때문에 더 안전한 방법이다.

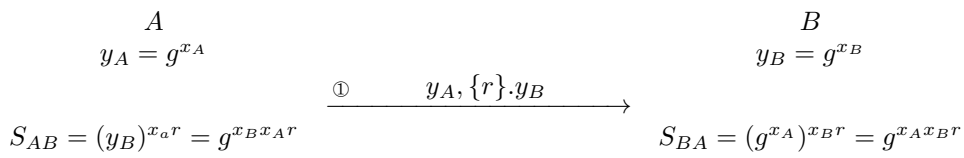
두 방법은 모두 참여자가 장기간 공개키 쌍을 가지고 있어야 한다. 특히, Matsumoto 등의 방식에서 참여자는 Diffie-Hellman 키 동의 프포토콜에서 사용하는 군에서 생성한 장기간 공개키 쌍이 필요하다. 실제 서비스 환경에서는 양 참여자가 모두 장기간 공개키 쌍이 없는 경우가 많으며, 장기간 공개키 쌍이 있더라도 전혀 다른 방식의 장기간 공개키 쌍을 가지고 있을 수 있다. 예를 들어 웹 브라우저와 웹 서버의 경우, 브라우저는 장기간 공개키 쌍을 가지고 있지 않을 수 있다.

중간자 공격을 방어하기 위해 제안된 그림 7.9에 제시된 SIGMA(SIGn-and-MAc)라는 프로토콜이 있다[5]. 이 프로토콜은 ISO 표준과 비교하면 서명에 상대방 식별자를 포함하지 않고 대신에 g^{ab}를 이용하여 계산한 K를 사용하여 식별자에 대한 MAC 값을 계산하여 전달하고 있다. 이 프로토콜에서 A는 메시지 3에서 자신의 신원을 밝힘으로 B는 메시지 2의 서명에서 강한 인증을 제공할 수 없다. 즉, 응용에 따라 상대방의 신원을 알 수 없는 경우도 있고, 신원을 밝히는 순서도 제한적인 경우도 있다. 이에 따라 프로토콜 구성이 달라질 수밖에 없다.

ISO 표준이나 그림 7.9에 제시된 프로토콜은 참여자의 식별자가 노출된다. 이 노출을 방지하고 싶으면 다음과 같이 메시지 2와 메시지 3을 암호화하여 교환할 수 있다.

$$\text{Msg } 2. B \rightarrow A: g^b, \{B||\text{Sig.}B(g^a||g^b)||\text{MAC.}K_m(B)\}.K_e$$

여기서 K_e와 K_m은 g^{ab}로부터 계산된 서로 독립된 키이다. 그림 7.9에 제시된 프로토콜과 달리 4개의 대칭적 메시지로 구성된 버전도 있으며, 이 프로토콜은 현재 현장에서 사용하고 있는 TLS 등에서 활용하고 있다.



<그림 7.10> Agnew 등의 프로토콜

6. 키 노출 관련 공격

사용자 부주의 등의 이유로 사용하는 암호키가 노출될 수 있다. 암호프로토콜은 이처럼 암호키가 노출되더라도 그것의 파급효과가 최소화되도록 설계해야 한다.

6.1 장기간 키의 노출

장기간 키가 노출되면 어떤 문제가 발생하는지 살펴보자. 장기간 키가 노출되면 해당 장기간 키를 사용한 과거 프로토콜 수행에서 추가로 노출되는 것이 있을 수밖에 없다. 이를 최소화하는 것이 중요 목표가 된다. 이를 위해 장기간 키가 노출되더라도 해당 장기간 키를 이용하여 확립된 세션키를 계산할 수 없으면 노출의 피해를 최소화할 수 있다. 하지만 대칭키만을 이용한 키 전송 프로토콜에서는 장기간 키로 세션키를 암호화하여 교환하기 때문에 장기간 키가 노출되고, 해당 트랜스크립트를 가지고 있으면 공격자는 세션키를 쉽게 계산할 수 있다. 이와 달리 Diffe-Hellman과 같은 키 동의 프로토콜을 사용하면 장기간 키가 노출되더라도 세션키를 계산할 수 없도록 만들 수 있다. 이 특성을 **전방향 안전성**(forward secrecy)이라 한다. 키 동의 프로토콜에 참여하는 모든 참여자의 장기간 키가 공격자에게 노출되더라도 공격자가 세션키를 계산할 수 없으면 완벽한(perfect) 전방향 안전성을 가지고 있다고 한다[6]. 그림 7.6에 제시된 프로토콜은 완벽한 전방향 안전성을 보장하는 프로토콜이다. 하지만 그림 7.7에 제시된 프로토콜은 전방향 안전성은 제공하지만 완벽한 전방향 안전성을 보장하지 못한다.

당연하지만 장기간 키가 노출되면 공격자는 이를 이용하여 해당 사용자 행세를 할 수 있다. 이에 대한 방어는 노출된 사실을 인지하였을 때 빠르게 키를 갱신하는 것이다. 과거에 대한 공격과 마찬가지로 키 동의 방식을 사용한다면 장기간 키의 노출과 상관없이 노출 이후에 확립된 세션키를 공격자가 계산할 수 없도록 만들 수 있다. 이 특성을 **후방향 안전성**(backward secrecy)이라 한다. 용어가 조금 혼란스러운 측면이 있다.

장기간 키가 노출되었을 때 공격자가 할 수 있는 또 다른 공격은 키 노출 위장(key compromise impersonation) 공격이다. 예를 들어 A의 장기간 키가 노출되었을 때 공격자 C가 A의 키를 이용하여 A에게 접근하여 다른 사용자 행세를 하는 공격을 말한다. 실제 공격자가 다른 사용자의 장기간 키를 얻게 되면 해당 사용자가 할 수 있는 것을 대부분할 수 있게 되며, 키 노출 위장 공격은 이것을 이용한 공격이다.

그림 7.10에 제시된 Agnew 등의 프로토콜[7]은 키 노출 위장 공격에 취약하다. 이 프로토콜에서 A의 개인키는 x_A이고, 공개키는 $y_A = g^{x_A}$이다. 이 프로토콜은 사용자의 장기간 키와 한 사용자만 선택한 랜덤값 r를 이용하여 세션키를 계산한다. 이때 공격자가 B의 장기간 키를 알고 있다고 하자. 그러면 공격자는 B에 접근하여 아무 사용자의 행세를 할 수 있다. 예를 들어 공격자 C가 D 행세를 하고 싶으면 D의 공개키 y_D를 확보한 다음 $y_D, \{r\}.y_B$를 전달하면 된다. C는 x_B를 알고 있으므로 B와 동일한 방식으로 세션키를 계산할 수 있다.

6.2 단기간 키의 노출

장기간 키와 달리 세션키가 노출되면 보통 노출되는 정보는 해당 세션으로 제한된다. 따라서 각 세션키를 독립적으로 생성하면 세션키의 노출에 대한 피해를 최소화할 수 있다. 반대로 독립적이지 않으면 하나의 세션키의 노출이 다른 세션키의 노출로 이어질 수 있다. 또 메시지의 최근성이 보장되지 않으면 노출된 세션키가 확립되었을 때 교환된 메시지를 이용하여 공격할 수 있다. 이 공격은 2장에서 이미 살펴본 바 있으며, 이와 같은 공격을 기지키 공격이라 한다.

최근에는 완벽한 전방향 안전성이 다른 의미로 사용하고 있다. 새 개념은 장기간의 키의 노출이 아니라 세션키의 노출에 초점을 두고 있다. 오늘날 PFS는 세션키가 노출되더라도 그 세션키 이전에 확립된 세션키들이 노출되지 않는다는 것을 의미할 수 있다. 앞서 언급한 바와 같이 세션마다 독립적인 세션키를 사용하면 PFS는 보장된다. 하지만 비용을 절약하기 위해 처음에는 DH 프로토콜을 통해 새로운 세션키 K를 확립하여 사용하고 그다음부터는 $H(K)$를 사용한다고 가정하자. 이 경우에도 해시함수의 일방향성 때문에 특정 세션의 세션키가 노출되더라도 그 이전 세션키들은 노출되지 않는다. 하지만 미래 세션키들은 모두 노출된다. 이와 관련하여 **미래 안전성**(future secrecy)라는 개념도 등장하였다. 특정 세션의 세션키가 노출되더라도 미래 세션키가 노출되지 않으면 미래 안전성을 가지고 있다고 한다.

이와 같은 개념들이 새롭게 등장한 이유는 메신저 보안처럼 특수한 환경을 위한 키 확립 프로토콜은 효율성을 위해 매번 DH 프로토콜을 수행하지 않고 키 확립하는 것이 필요하였기 때문이다. 이에 대해서는 10장에서 자세히 설명한다.

7. 프로토콜 상호작용 공격

Kesley 등[8]은 프로토콜 P_1의 안전성을 증명하였더라도 이를 공격하기 위한 프로토콜 P_2을 만들어 사용하도록 할 수 있다면 P_1을 공격할 수 있다는 것을 보였다. 이때 공격하기 위해 만든 프로토콜을 선택 프로토콜(chosen protocol)이라 한다. 선택 프로토콜을 통한 공격이 가능하기 위해서는 공격 대상 프로토콜에서 사용하는 장기간 키를 선택 프로토콜에서도 사용하도록 만들어야 한다. 따라서 프로토콜 상호작용 공격의 교훈은 암호키는 한 가지 용도로만 사용해야 한다는 것이지만 그렇게 할 경우 각 사용자가 많은 수의 키를 사용해야 하는 문제점이 발생한다. 그러므로 동일 암호키를 여러 프로토콜에서 사용하면 하나의 프로토콜만 독립적으로 분석하는 것으로 충분하지 않고, 해당 암호키를 사용하는 모든 프로토콜을 종합적으로 분석해야 한다.

참고문헌

[1] Tuomas Aura, Pekka Nikander, “Stateless Connections,” Int’l Conf. on Informations and Communications Security, ICICS’97, LNCS 1334, pp. 87–97. 1997.

[2] Dave Otway , Owen Rees, “Efficient and Timely Mutual Authentication,” ACM SIGOPS Operating Systems Review, Vol. 21, No. 1, pp. 8–10, Jan. 1987.

[3] ISO/IEC IS 9798-3, “Entity authentication mechanisms — Part 3: Entity authentication using asymmetric techniques,” 1993.

[4] T. Matsumoto, Y. Takashima, H. Imai, “On Seeking Smart Public-key Distribution Systems,” Trans. of the IECE, E69, pp. 99–106, 1986.

[5] Hugo Krawczyk, “SIGMA: The ‘SIGn-and-MAc’ Approach to Authenticated Diffie-Hellman and Its Use in the IKE Protocols,” Advances in Cryptology, CRYPTO 2003, LNCS 2729, pp. 400–425, 2003.

[6] Whitfield Diffie, Paul C. Oorschot, Michael J. Wiener, “Authentication and Authenticated Key Exchanges.” Designs, Codes and Cryptography, Vol. 2, No. 2, pp. 107–125, 1992.

[7] G. Agnew, R. Mullin, S. Vanstone, "An Interactive Data Exchange Protocol Based on Discrete Exponentiation," Advances in Cryptology, Eurocrypt 88, LNCS 330, pp. 159–166. 1988.

[8] J. Kelsey, B. Schneier, D. Wagner "Protocol Interactions and the Chosen Protocol Attack," 5th Int'l Workshop on Security Protocols, LNCS 1361, pp. 91–104, 1998.

퀴즈

1. 서비스 거부 공격을 어렵게 만들기 위해 비상태 기반으로 프로토콜을 구성할 수 있다. 비상태 기반에서 서버는 진행 중인 세션에 대해 상태 정보를 유지하지 않고 클라이언트에 전달한다. 이때 상태 정보를 조작할 수 없도록 상태 정보에 대한 MAC값을 계산하여 상태 정보와 함께 클라이언트에 줄 수 있다. 이와 같은 보호 조치와 관련된 다음 설명 중 틀린 것은?

   ① 상태 정보에 세션ID, 메시지번호, 타임스템프 들을 포함하여 서버는 수신한 상태 정보가 유효한 것인지 확인할 수 있다.

   ② 각 세션마다 다른 키를 사용하면 재전송 공격을 방어하는데 효과적이다.

   ③ 상태 정보는 서버가 생성하고 서버가 확인하면 되는 것이기 때문에 서버는 MAC키를 누구와 공유할 필요가 없다.

   ④ 세션별 다른 키를 사용하는 것보다 하나의 키를 사용하되 이 키를 자주 변경하는 것이 비상태 기반 프로토콜 특성에 맞는 방법이다.

2. 전방향 안전성은 장기간 키가 노출되었을 때도 고려하고, 세션키가 노출되었을 때도 고려한다. 장기간 키 또는 세션키가 노출되었을 때 과거 세션키가 노출되지 않으면 전방향 안전성이 제공된다고 한다. 다음 중 전방향 안전성과 관련된 내용 중 틀린 것은?

   ① 서버가 사용자의 장기간 키로 세션키로 암호화하여 분배하는 키 전송 방식의 키 확립 프로토콜은 전방향 안전성을 제공할 수 없다.

   ② 현재 세션키가 K일 때, $H(K)$를 이용하여 다음 세션키를 계산하면 전방향 안전성을 보장할 수 없다.

   ③ $S_{AB} = (y_B)^a(g^b)^{x_A}$와 같이 계산하는 MTI 프로토콜은 전방향 안전성을 보장한다. 여기서 S_{AB}는 A가 계산하는 B와 공유하게 되는 비밀 정보이고, x_A는 A의 장기간 개인키이며, a는 A가 선택한 랜덤 값이다. y_B는 상대방 B의 공개키이고, g^b는 B가 전달한 Diffie-Hellman 값이다.

   ④ 기본 Diffie-Hellman 키 동의 프로토콜은 완벽한 전방향 안전성을 제공한다.

3. 기본 Diffie-Hellman 키 동의 프로토콜은 중간자 공격에 취약하다. 이 공격을 방어하기 위한 방법에는 크게 2 종류가 있다. 하나는 교환되는 값을 인증하여 중간자가 값을 바꿀 수 없도록 하는 것이고, 다른 하나는 중간자가 교환되는 값을 바꾸더라도 중간자가 유효한 키를 계산할 수 없도록 하는 것이다. 두 방법을 비교한 다음 설명 중 틀린 것은?

   ① 두 방법 모두 장기간 키가 추가로 필요하다.

   ② 키 계산 방법을 바꾼 MTI 기법의 경우 장기간 키(개인키)가 모두 노출되어도 공격자는 세션키를 계산할 수 없다.

   ③ 두 방법 모두 상대방의 인증서를 이용하여 상대방의 공개키를 인증해야 한다.

   ④ 두 방법 모두 계산 비용이 기본 DH에 비해 증가한다.

4. 암호프로토콜에 대한 공격 중 타입 공격이라는 것이 있다. 타입 공격은 메시지의 구성 요소를 원래와 다른 것으로 해석하도록 하여 공격하는 것으로 재전송 공격의 한 종류이다. 타입 공격을 방어하기 위한 수단으로 적절하지 않은 것은?

   ① 암호화하는 평문의 형태를 타입 공격을 할 수 없도록 적절하게 구성한다.

   ② 암호문 내에 메시지 번호를 포함한다.

   ③ 암호문마다 다른 키를 사용하여 암호화한다.

   ④ 인증 암호화를 한다.

연습문제

1. 앞으로 사용할 메시지 또는 메시지의 일부를 사용하여 프로토콜을 공격하는 것을 사전전송 공격이라 한다. 4장에 제시된 다섯 번째 시도 프로토콜에 대해 공격자는 사용자 A가 사용할 난스 값을 예측할 수 있다고 가정하자. 공격자는 앞으로 사용할 예측한 난스 값을 이용하여 메시지 2를 만들어 서버에 전송할 수 있다. 서버는 이 요청에 대해 응답을 할 것이고, 공격자는 이 응답을 보관할 수 있다. 공격자가 이를 통해 추가로 어떤 공격이 가능한지 제시하시오.

2. 그림 7.1에 제시된 프로토콜은 방향과 상관없이 K_{AB}를 이용한다. 각 방향마다 다른 키를 사용하면 그림 7.2에 제시된 공격이 가능한지 논하시오.

3. 그림 7.4에 제시된 프로토콜은 메시지 1과 2에 포함된 $\{N_A||N||A||B\}.K_{AS}$와 메시지 3과 4에 포함된 $\{N_A||K_{AB}\}.K_{AS}$에서 K_{AB}와 $N||A||B$의 길이가 같을 경우 타입 공격이 가능하다. 이 프로토콜도 각 방향마다 다른 키(사용자가 서버로 메시지를 보낼 때 사용하는 키와 서버가 사용자로 보낼 때 사용하는 키가 다른 경우)를 사용하면 타입 공격이 가능한지 논하시오.

4. 그림 7.4에 제시된 프로토콜은 메시지 1과 2에 포함된 $\{N_A||N||A||B\}.K_{AS}$와 메시지 3과 4에 포함된 $\{N_A||K_{AB}\}.K_{AS}$에서 K_{AB}와 $N||A||B$의 길이가 같을 경우 타입 공격이 가능하다. 이 프로토콜에서 N_A가 64비트, K_{AB}가 128비트, N은 64비트, A와 B는 각 20바이트라 하고, 128비트의 블록 암호를 ECB 모드를 이용하여 메시지를 암호화한다고 하였을 때 타입 공격이 여전히 가능한지 논하시오.

5. 기본 Diffie-Hellman 키 동의 프로토콜은 중간자 공격에 취약하다. 그뿐만 아니라 키 확인 과정이 없다. 물론 키 동의 과정은 쉽게 추가할 수 있다. 이와 관련하여 다음 각각에 대해 답하시오.

   ① 그림 7.6과 그림 7.7에 제시된 중간자 공격에 강건한 두 프로토콜에 키 확인 과정을 추가하시오.

   ② 그림 7.6의 프로토콜은 키 확인 과정과 무관하게 메시지 1과 메시지 2에서 수신한 서명에 문제가 있으면 프로토콜의 수행을 중단하다. 하지만 그림 7.7에 제시된 프로토콜은 인증을 하지 않고 있기 때문에 중간에 중단하지는 않는다. 실제 그림 7.7의 프로토콜에 중간자 공격이 일어났을 때, 키 확인 과정이 있는 경우와 없는 경우 어떤 차이가 있는지 설명하시오.

6. 중간자 공격에 취약한 기본 Diffie-Hellman 키 동의 프로토콜을 개선하기 위해 다음과 같이 프로토콜을 구성하였다.

$$\begin{array}{lll} \text{Msg 1.} & A \rightarrow B: & g^a \\ \text{Msg 2.} & B \rightarrow A: & g^b, B, \text{Sig}.B(g^a||g^b) \\ \text{Msg 3.} & A \rightarrow B: & A, \text{Sig}.A(g^b||g^a) \end{array}$$

   이 프로토콜의 문제점을 찾으시오.

7. 그림 7.6, 그림 7.7, 그림 7.8에 제시된 3개 프로코콜에서 a가 노출되었을 때, 재전송 공격이 가능한지, 가능하지 않으면 왜 가능하지 않는지 설명하시오.

8. 키 노출과 관련하여 단기간 키의 노출과 장기간 키의 노출이 발생하였을 때 우리가 보호하고 싶은 것은 무엇인지 설명하시오. 둘 다 노출 시점을 기준으로 과거와 미래에 대해 나누어 설명하시오.

9. 프로토콜 상호작용 공격이 주는 교훈을 설명하시오.