정보보호개론

제6장 키 확립 프로토콜 개요

1. n명 간의 비밀통신 문제

n명의 사용자가 대칭 암호알고리즘을 이용하여 각 사용자 간의 메시지를 비밀스럽게 교환하고 싶다. 이것이 가능하기 위해서는 각 사용자 쌍마다 서로 다른 비밀키의 공유가 필요하다. 암호키의 특성상 장기적으로 사용하면 안전성이 약해(우연한 난스 중복 등)지기 때문에 사용 중인 키를 새 키로 바꾸는 키 갱신도 필요하다. 이 문제를 해결하기 위한 여러 해결책을 살펴보자.

1.1 비밀키만 사용하는 방식

이 절에서는 대칭키만을 사용하여 n명 간의 비밀 통신 문제를 해결하는 3가지 방법을 살펴본다. n명 간의 비밀 통신 문제를 해결하기 위한 가장 단순한 방법은 $n(n-1)/2$개의 비밀키를 생성한 후 각 사용자에게 $n-1$개의 키를 오프라인으로 주는 것이다. 이 방법은 각 사용자가 유지하는 키가 사용자 수 n에 비례하기 때문에 확장성이 없다. 그뿐만 아니라 키 갱신이나 새로운 사용자가 추가되었을 때 이를 해결하기 위한 수단도 마땅히 없다.

이와 같은 확장성 문제를 해결하기 위해 신뢰 기관을 사용하는 방식을 생각해 볼 수 있다. 사용자는 신뢰 기관과 장기간 비밀키를 공유하고, 각 사용자 간의 비밀키는 필요할 때 신뢰 기관을 통해 키를 확립하는 방식을 사용하면 각 사용자는 하나의 비밀키만 유지하면 된다. 물론 신뢰 기관은 n개의 키를 유지해야 하지만 이것은 충분히 실제 현장에서 사용 가능한 모델이다. 다만, 각 사용자와 신뢰 기관이 최초에 장기간 키를 확립하는 문제와 필요할 때 또는 주기적으로 이 키를 갱신하는 문제의 해결이 필요하다. 이 확립과 갱신을 오프라인(예: 스마트폰 USIM에 등록된 키로 이동통신사와 장기간 대칭키 공유)으로 할 수 있지만, 현재 대부분의 서비스에서는 이 과정을 오프라인 방식으로 진행하기에는 적절하지 않다.

사용자가 유지해야 하는 키의 개수가 1은 아니지만, 확장성을 가지도록 할 수 있는 방법도 있다. Blom의 방법[1]과 Blundo 등의 방법[2]이 여기에 해당하는데, 전자는 대칭 행렬을 활용하며, 후자는 $f(x,y) = f(y,x)$인 이변량 함수(bivariate function)를 사용한다. 둘 다 1부터 n 사이의 값인 보안 변수 t를 결정하면 각 사용자는 t에 비례한 정보만 유지하면 된다. 각 사용자는 이 정보만 있으면 $n-1$명의 다른 사용자와 사용할 수 있는 독특한 비밀키를 생성할 수 있다. 물론 이 키는 변하지 않는다. 이 문제를 해결하기 위해 매번 해당 키로 세션키를 확립하여 사용할 수 있다. 이 방법에서 각 사용자는 최초에 신뢰 서버로부터 t개의 정보를 안전하게 받아 비밀로 유지해야 하며 t가 클수록 안전성이 높아진다. 또 이 정보는 각 사용자가 개별적으로 갱신할 수 있는 것은 아니다.

1.2 공개키를 사용하는 방식

2자 간 공개키를 이용하여 세션키를 확립하고 싶으면 한 사용자가 세션키를 생성하여 다른 사용자의 공개키로 암호화하여 전달하면 된다. 이때 메시지 수를 줄이기 위해 다음과 같이 비밀로 전달하고 싶은 메시지 M를 해당 세션키

K로 암호화하여 함께 보낼 수 있다.

$$\text{Msg 1. } A \rightarrow B: \ C = \{M\}.K_1, \text{MAC}.K_2(C), \{T_A||K\}.+K_B$$

여기서 K_1과 K_2는 K를 이용하여 계산한 서로 독립적인 키이다. 하지만 B는 이 메시지가 A가 보냈다고 확신할 수 없다. 여기에 이 기능을 추가하는 방법은 Sig.$A(M)$과 같은 값을 함께 전달하면 된다. 참고로 A는 이 메시지를 보내기 전에 B의 인증서를 받아 확인하여야 한다.

대칭키만을 사용하는 방식과 비교하면 각 사용자는 인증서와 개인키만 유지하면 된다. 따라서 확장성에 문제가 없다. 물론 대칭키와 비교하였을 때 공개키의 길이는 상대적으로 길다. 사용자는 편리성을 위해 다른 사용자의 인증서를 보관할 수 있지만, 이것이 확장성에 문제가 되지는 않는다. 반면에 공개키를 사용하면 인증서를 확인하는 비용을 포함하여 공개키 연산 비용이 대칭키에 비해 상대적으로 높다는 단점이 있다. 이외에도 공개키 기반구조가 구축되어 있어야 하는 단점이 있다. 하지만 인증기관은 오프라인 서버인 반면에 대칭키 방식에서 키를 발급하는 신뢰 기관은 온라인 서버이다.

1.1절에서 소개한 두 번째 방법에서 최초 장기간 키를 확립하는 문제와 필요할 때 장기간 키를 갱신하는 문제의 해결이 필요하다고 하였는데, 이 문제를 공개키 기반 프로토콜을 사용하여 해결할 수 있다. 이와 같은 방법을 사용하면 최초 장기간 키를 확립할 때와 갱신이 필요할 때만 공개키 기반 프로토콜을 사용하고, 평소에는 대칭키만 사용하는 프로토콜을 통해 효율적으로 서비스를 받을 수 있다.

2. 키 확립 프로토콜

키 확립 프로토콜은 4장에서 이미 살펴본 바와 같이 둘 이상의 참여자들이 비밀키를 공유할 수 있도록 해주는 암호프로토콜이다. 키 확립 프로토콜을 통해 얻어지는 비밀키는 보통 단일 세션에 사용하기 위한 세션키이다. 키 확립 프로토콜은 크게 키 전송 또는 동의 과정과 키 확인 과정으로 구성된다. 키 확인 과정은 키 확립 프로토콜을 통해 키를 공유한 사용자들이 서로 같은 키를 가졌는지 확인하는 과정을 말한다. 4장에서 언급한 바와 같이 키 확인 과정은 비교적 쉽게 추가할 수 있다.

키 확립 프로토콜은 다음과 같이 다양하게 분류할 수 있다.

- 분류 1. 키를 생성하는 주체에 따른 분류
- 분류 2. 사용하는 암호기술에 따른 분류
- 분류 3. 키를 확립하는 사용자 수에 따른 분류

2.1 키 생성 주체에 따른 분류

참여자 중 어느 한 참여자가 홀로 키를 생성하여 다른 참여자에게 주는 방식을 **키 전송 프로토콜**(key transport protocol)이라 하고, 특정 참여자가 홀로 키를 생성하지 않는 방식을 **키 동의 프로토콜**(key agreement protocol)이라 한다. 위 두 가지 요소를 다 가지고 있으면 혼합 프로토콜(hybrid protocol)이라 한다. 하지만 혼합 프로토콜은 실제 사용하는 경우는 거의 없다. 보통 키 동의 프로토콜에서 키를 확립하는 모든 참여자는 동등하게 키 생성에 기여하며, 가장 널리 사용하고 있는 키 동의 프로토콜은 신뢰 서버를 사용하지 않고 공개키 기술을 이용하는 자체 강화 방식의 프로토콜이다.

대칭키만을 사용할 때는 주로 키 전송 프로토콜을 많이 사용한다. 이와 같은 프로토콜에서 키를 홀로 생성하는 주체는 보통 신뢰 기관이다. 이 신뢰 기관을 키 분배 센터(KDC, Key Distribution Center)라 한다. 신뢰 기관이 키를

생성하지 않고 사용자들 사이에 안전하게 중계해주는 역할만 하면 키 중계 센터(KTC, Key Translation Center)라 한다. 하지만 신뢰 모델이라는 측면에서는 보통 키 분배 센터 방식을 더 선호한다.

2.2 키를 확립하는 사용자 수에 따른 분류

키를 확립하는 사용자 수에 따라 분류하면 크게 2자 간, 3자 간, 다자 간으로 나뉘어진다. 가장 많이 사용하는 것은 2자 간이다. 2000년도에 Joux[3]가 3자 간 자체 강화 방식의 키 동의 프로토콜을 처음으로 제안한 이후 2자 간, 다자 간으로 분류하던 것을 2자 간, 3자 간, 다자 간으로 분류하고 있다. 다자 간은 다른 말로 그룹키(group key) 프로토콜, 회의키(conference key) 프로토콜이라고 한다. 회의키라는 이름에서 알 수 있듯이 원격에서 다자 간 영상, 문자 회의를 할 때 다자간 키 확립 프로토콜이 필요하다.

2.3 키 확립 프로토콜의 요구사항

3장에서 이미 키 확립 프로토콜의 요구사항을 살펴본 바 있으며, 크게 다음과 같이 요약할 수 있다.

- 키의 비밀성: 키를 확립하는 참여자 외에는 확립하는 비밀키를 얻을 수 없어야 한다. 단, 신뢰 기관이 참여하는 프로토콜에서는 보통 신뢰 기관이 생성하여 주기 때문에 이 기관도 비밀키를 알고 있다.
- 키의 최근성: 키를 확립하는 참여자는 수신한 키가 이전에 사용한 적이 없는 이번에 생성한 키임을 확신할 수 있어야 한다.
- 키의 용도: 키를 확립하는 참여자는 수신한 키가 본인이 생각하고 있는 상대방과 사용하기 위한 키인지 확인할 수 있어야 한다.
- 생성 주체 확인, 참여자 확인: 키 전송 프로토콜의 경우 키를 수신한 참여자는 이 키를 프로토콜에서 정의한 주체가 생성한 것인지 확인할 수 있어야 한다. 프로토콜에 따라 현재 참여하고 있는 상대방이 자신이 의도한 상대방인지 명백하게 확인이 필요할 수 있다.
- 키 확인: 키를 확립하는 참여자는 상대방이 같은 키를 가졌는지 확인할 수 있어야 한다. 키 확인을 프로토콜에 추가하는 것은 어렵지 않기 때문에 프로토콜 서술에서 생략하는 경우도 있다.

참여자 확인의 경우 프로토콜의 구성에 따라 상대방에 대한 직접적인 확인을 하지 않는 경우도 많다. 예를 들어 4장에서 살펴본 키 확립 프로토콜에서 A와 B는 직접적으로 상호 인증을 하지 않는다. 서버로부터 받은 K_{AB}가 상대방과 사용하기 위한 키임을 확신하게 되면 이 키를 가지고 있는 사용자를 상대방으로 확신할 수 있다. 하지만 자체 강화 방식의 키 동의 프로토콜에서는 상대방에 대한 명백한 인증이 꼭 필요하다.

위 다섯 가지 요구사항 외에 키 무결성이라는 개념도 있다. 키 무결성이 보장된다는 것은 키가 분배될 때 키값이 원래 생성된 그대로 수정 없이 전달되었다는 것을 말한다. 따라서 키 분배 센터를 이용한 2자 간 키 전송 프로토콜에서 키 무결성이 보장되면 키 분배 센터에 대한 신뢰 때문에 두 참여자는 동일한 키를 얻게 된다. 하지만 키 무결성보다 키 확인이 더 중요하며, 키 무결성을 보장하지 않더라도 키 비밀성과 키 확인 요구사항이 충족되면 해당 키는 안전하게 사용할 수 있는 키가 된다. 예를 들어 키값이 0110 4bit라고 가정하자. 그런데 두 사용자는 0110 대신에 둘 다 0100을 수신하였다고 하자. 그러면 키 무결성은 만족하지 못하지만, 키 확인은 가능할 것이며, 해당 값의 비밀성이 보장되었다면 세션키로 사용하는 데 문제가 없다.

Msg 1. $A \rightarrow S:$ A, B, N_A
Msg 2. $S \rightarrow A:$ $\{B||N_A||K_{AB}||\{A||K_{AB}\}.K_{BS}\}.K_{AS}$
Msg 3. $A \rightarrow B:$ $A, \{A||K_{AB}\}.K_{BS}$
Msg 4. $B \rightarrow A:$ $\{N_B\}.K_{AB}$
Msg 5. $A \rightarrow B:$ $\{N_B - 1\}.K_{AB}$

<그림 6.1> Needham-Schroeder 프로토콜

Msg 1. $A \rightarrow S:$ A, B
Msg 2. $S \rightarrow A:$ $\{B||T_S||K_{AB}||\{A||T_S||K_{AB}\}.K_{BS}\}.K_{AS}$
Msg 3. $A \rightarrow B:$ $A, \{A||T_S||K_{AB}\}.K_{BS}$

<그림 6.2> Denning-Sacco 프로토콜

2.4 키 전송 프로토콜의 예

그림 6.1은 Needham과 Schroeder가 제안한 프로토콜로서[4], 3장에 제시된 4번째 시도 프로토콜과 유사하다. A만 키의 최근성을 확인할 수 있기 때문에 B는 기지키 재전송 공격에 취약하다. 또 메시지 2에서 불필요하게 이중 암호화를 하고 있으며, B만 키 확인을 하고 있다.

그림 6.2에 제시된 키 전송 프로토콜은 그림 6.1에 제시된 프로토콜의 문제점을 타임스탬프를 통해 해결하고 있다[5]. 하지만 여전히 불필요한 이중암호화를 하고 있다. 이 두 프로토콜을 통해 70년대 후반부터 80년대 초까지는 키 확립 프로토콜조차 제대로 설계하지 못하였다는 것을 알 수 있다.

2.5 키 동의 프로토콜의 예

2.5.1 이산대수 문제

공개키 암호알고리즘에서 가장 널리 사용하고 있는 문제 중 하나가 **이산대수**(discrete logarithm) 문제이다. 이산대수를 이해하기 위해서는 군(group)이라는 수학 개념을 이해할 필요가 있다. 쉽게 생각하면 군은 이항(binary) 연산에 의해 닫혀 있는 집합이다. 닫혀 있다는 것은 집합에 있는 임의의 두 원소를 선택하여 군의 이항 연산을 적용하면 그 결괏값이 집합의 원소가 된다는 것을 말한다.

예를 들어 집합 $\mathbb{G} = \{1, 2, 3, 4, 5, 6\}$과 $\circ = \times \bmod 7$ 연산을 생각하여 보자. 집합 $\mathbb{G}$는 이항 연산 $\circ$에 대해 닫혀있다. 또한 1이 항등원(identity element)이며, 모든 원소는 역원을 가지고 있다. 항등원이란 모든 원소 a에 대해 $a \circ e = a$가 되는 e를 말하며, 어떤 원소 a의 역원은 $a \circ b = e$가 되는 b를 말한다.

앞으로 편리상 $\circ$ 연산을 곱셈이라 하자. $\mathbb{G}$에서 항등원은 1이며, $3 \circ 5 = 1$이기 때문에 3의 역원은 5이다. 이를 통해 $3 \circ x = 2$와 같은 방정식의 해를 구할 수 있다. 이 식 양변에 5를 곱하면 5는 3의 역원이므로 $x = 2 \circ 5 = 3$임을 알 수 있다.

3을 계속 거듭제곱하면 다음과 같이 집합에 있는 모든 수를 생성할 수 있다.

$$3^1 = 3, 3^2 = 2, 3^3 = 6, 3^4 = 4, 3^5 = 5, 3^6 = 1$$

이처럼 어떤 특정 원소를 계속 거듭제곱하여 군에 있는 모든 요소를 생성할 수 있으면 이 군을 순환군(cyclic group)

$$\text{Msg 1.} \quad A \to B: \quad g^a$$
$$\text{Msg 2.} \quad B \to A: \quad g^b$$

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

<그림 6.3> Diffie-Hellman 프로토콜

이라 하고, 해당 원소를 군의 생성자(generator)라 한다. 또 기저 3에 대한 2의 이산대수는 2가 된다. 즉, $y = g^x$일 때 기저 g에 대한 y의 이산대수는 x가 된다.

임의의 소수 p를 선택하면 이항연산 $\circ = \times \bmod p$에 대해 집합 $\{1, 2, \ldots, p-1\}$은 순환군이 된다. 이 군을 $\mathbb{Z}_p^*$로 보통 표기한다. 군의 크기를 군의 위수(order)라 하며, $\mathbb{Z}_p^*$의 위수는 $p-1$이다. 이산대수 문제란 순환군의 정보, 생성자 g, 군의 임의의 원소 y가 주어졌을 때 기저 g에 대한 y의 이산대수를 구하는 문제를 말한다. 기저를 이용하여 계속 거듭제곱하면 궁극에는 답을 찾을 수 있다. 하지만 순환군의 위수가 일정 크기 이상이면 이 문제를 해결하는 것은 계산적으로 어렵다.

2.5.2 Diffie-Hellman 키 동의 프로토콜

그림 6.3에 제시된 Diffie와 Hellman이 제안한 키 동의 프로토콜은 암호기술에 있어 역사적이며, 매우 중요한 프로토콜이다[6]. 공개키 개념이 소개된 최초 학술적 논문에 등장한 프로토콜이며, 지금까지도 가장 널리 사용하고 있는 프로토콜이다.

이 프로토콜을 수행하기 위해서는 먼저 사용할 군을 결정해야 한다. 프로토콜이 사용하는 군이 $\mathbb{G}$이고, 그것의 생성자가 g이며, 이 군의 위수가 n이라고 가정하고 프로토콜을 설명하면 다음과 같다. A는 $a \in \{1, \ldots, n\}$를 임의로 선택하여 그만큼 g를 거듭제곱한 값을 B에게 전달하면 B도 $b \in \{1, \ldots, n\}$를 임의로 선택하여 그만큼 g를 거듭제곱한 값을 전달한다.

두 사용자는 받은 값을 이용하여 g^{ab}를 계산하며, 이 값을 이용하여 사용할 비밀키를 계산하여 사용한다. 이때 사용하는 함수를 키 유도 함수(KDF, Key Derivation Function)라 한다. 군이 충분히 크다는 가정하에 두 사용자가 계산한 g^{ab}는 두 사용자 외에는 이산대수 문제 때문에 계산하기가 매우 어렵다. 이 때문에 이 값은 두 사용자만이 알게 되는 비밀값이 되며, 그림에서 S_{AB}, S_{BA}로 표현하고 있다.

이 프로토콜의 안전성은 이산대수 문제는 물론 Diffie-Hellman 계산 문제와 결정 문제가 어려워야 한다. Diffie-Hellman 계산 문제란 군 정보, 군 생성자 g, g^a, g^b가 주어졌을 때 g^{ab}를 계산하는 것이고, 결정 문제란 계산 문제 때 주어지는 것을 포함하여 군의 또 다른 원소 z가 주어졌을 때 z가 g^{ab}와 같은지 여부를 답하는 문제이다. 이 프로토콜은 수동 공격에 대해서는 안전하지만 서로 교환하는 값을 인증하고 있지 않으므로 능동 공격에 취약하다. 이 문제는 7장에서 자세히 살펴본다.

지금까지 살펴본 키 확립 프로토콜은 모두 키 전송 프로토콜이었다. 하지만 그림 6.3에 제시된 프로토콜은 키 동의 프로토콜이다. 키 동의 프로토콜은 어떤 특정 참여자가 홀로 키를 결정하지 않는다. 보통 키를 확립하는 모든 참여자가 키를 결정하는데 동등하게 참여한다. Diffe-Hellman 키 동의 프로토콜에서는 g^{ab}를 이용하여 세션키를 계산하게 되는데, A와 B는 모두 이 값을 결정하는 데 있어 대칭적 기여를 하였다. 이처럼 키 동의 프로토콜에서는 각 참여자가 같은 수준의 계산 및 통신 비용이 소요되도록 설계하며, 각자 선택한 랜덤값이 세션키 계산에 포함된다. 따라서 각 참여자 입장에서 자신이 선택한 랜덤요소가 충분히 랜덤하면 프로토콜을 통해 생성한 세션키도 역시 충분히 랜덤하다는 것을 믿을 수 있다. 이 때문에 키 전송 프로토콜들과 달리 키 동의 프로토콜에서는 키의 최근성을 입증하기 위한 별도 메커니즘의 사용이 필요 없다. 하지만 반대로 키 제어 요구사항처럼 키 전송 프로토콜에는 없는 요구사항도 있다.

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

<그림 6.4> 키 제어 요구사항이 충족되는 Diffie-Hellman 프로토콜

$$
\begin{array}{lll}
\text{Msg 1.} & A \rightarrow B: & A, B, N_A \\
\text{Msg 2.} & B \rightarrow S: & A, B, N_A, N_B \\
\text{Msg 3.} & S \rightarrow B: & C_{AS} = \{B||N_A||K_{AB}\}.eK_{SA}, T_{AS} = \text{MAC}.mK_{SA}(C_{AS}) \\
 & & C_{BS} = \{A||N_B||K_{AB}\}.eK_{SB}, T_{BS} = \text{MAC}.mK_{SB}(C_{BS}) \\
\text{Msg 4.} & B \rightarrow A: & C_{AS}, T_{AS}, \{N_A||N_B\}.K_1 \\
\text{Msg 5.} & A \rightarrow B: & \{N_B\}.K_1 \\
\text{Msg 6.} & B \rightarrow A: & C_1 = \{M_1\}.K_4, \text{MAC}.K_5(C_1) \\
\text{Msg 7.} & A \rightarrow B: & C_2 = \{M_2\}.K_2, \text{MAC}.K_3(C_2) \\
\text{Msg 8.} & B \rightarrow A: & C_3 = \{M_3\}.K_4', \text{MAC}.K_5'(C_3) \\
 & \vdots &
\end{array}
$$

<그림 6.5> 최근 추세를 반영한 키 중재 서버 기반 키 확립 프로토콜

키 제어 요구사항은 키 동의 프로토콜에서만 필요한 요구사항으로 참여자 중 누구도 공유할 세션키를 사전에 미리 계산할 수 있거나 선택한 값이 되도록 만드는 것이 계산적으로 어려워야 한다는 것을 말한다. Diffe-Hellman 프로토콜에서 두 사용자는 랜덤하게 a와 b를 선택하기 때문에 최종값 g^{ab}가 특정 값이 되도록 만드는 것은 매우 어렵다. 하지만 Mitchell 등[7]은 참여자 중 랜덤요소를 가장 늦게 제공하는 참여자는 다른 참여자보다 키를 제어할 수 있는 확률이 높다는 것을 보였다. 이 문제점은 그림 6.4처럼 랜덤요소를 해시한 값을 미리 제공한 후에 실제 랜덤요소를 나중에 전달하여 해결할 수 있다. B는 해시함수의 일방향성 때문에 메시지 3을 받아야 A가 선택한 값을 알 수 있으며, A는 해시함수의 충돌회피성 때문에 자신이 선택한 값을 나중에 바꿀 수 없다. 이 해결책의 단점은 프로토콜의 라운드 수가 하나 증가한다는 것이다.

2.6 키 확립 프로토콜의 최근 추세

이제는 기본 암호화 방식으로 암호화 후 MAC 값을 계산하는 인증 암호화 방식을 사용하기 때문에 하나의 세션키가 필요한 것이 아니라 두 개의 키가 필요하다. 두 개의 키 중 하나는 암호화에 사용하고 다른 하나는 MAC 값을 계산하기 위해 사용한다. 확립해야 하는 키의 개수가 많아진다고 키 확립 프로토콜을 여러 번 수행하는 것은 아니며, 키 전송 방식의 경우 필요한 개수의 키를 생성하여 분배하지 않는다. 기존처럼 하나의 비밀키 또는 비밀 정보를 확립한 후, 이 값으로부터 필요한 개수의 키를 생성하여 사용한다.

그림 6.5의 프로토콜은 4장에서 제시한 프로토콜을 최근 추세를 반영하여 수정한 프로토콜이다. 여기서 각 사용자는 서버 S와 기존처럼 하나의 키를 공유하고 있는데, 이 키로부터 4개의 키 eK_{AS}, mK_{AS}, eK_{SA}, mK_{SA}를 생성하여 사용한다. 여기서 eK_{AS}는 암호화키이고, mK_{AS}는 MAC키이며, 아랫첨자는 사용하는 키의 방향을 나타낸다. 이 프로토콜은 여전히 기존과 마찬가지로 똑 같은 길이의 하나의 대칭키 K_{AB}만 교환한다. 하지만 이 키로부터 양 사용자는 필요한 만큼의 키를 계산하여 사용한다. 하나의 키에서 실제 사용할 여러 개의 키를 생성할 때 사용하는 것이 키 유도 함수이다. 이 함수는 주어진 입력을 이용하여 필요한 크기의 랜덤 비트 문자열 출력하여 준다. 출력된 비트 문자열은 서로 독립적이기 때문에 256bit 출력을 2개의 128bit로 나누었을 때, 이 중 하나가 노출되더라도 다른 하나를 추측하는 데 도움이 되지 않는다. 키 유도 함수에 대해서는 11장에서 자세히 살펴본다.

이전에는 A와 B 간의 키를 확립하면 A도 B도 같은 키를 사용하여 메시지를 암호화하여 상대방에게 전달하였다.

또 이 키 자체를 키 확인 과정에서 사용하였다. 최근에는 이와 달리 각 용도마다 다른 키를 사용한다. 이를 키 분리(key separation) 원리라 한다. 그림 6.5에 제시된 프로토콜은 K_{AB}로부터 K_1부터 K_5까지 다섯 개의 키를 생성하여 사용한다. 이 중 K_1은 키 확인할 때 사용하며, K_2, K_3은 A가 B에게 메시지를 전달할 때 사용하고, K_4, K_5는 B가 A에게 메시지를 전달할 때 사용한다.

키 확인을 위한 두 개의 암호문은 인증 암호화를 하고 있지 않다. 이 암호문의 평문 크기가 한 블록으로 충분히 표현할 수 있어 생략하고 있다. 만약 이 두 암호문도 인증 암호화가 필요하면 최초 5개가 아니라 6개를 만들어 사용할 수 있다.

키 분리가 극단화되어 한 번 확립된 키를 계속 사용하는 것이 아니라 메시지마다 다른 키를 사용하도록 그림 6.5의 메시지 8처럼 프로토콜을 구성할 수 있다. 여기서 $K_4' = H(K_4)$이고 $K_5' = H(K_5)$이다. 메시지 10에서는 다시 메시지 8에서 사용한 키를 해시하여 새 키를 만들어 사용할 수 있다. 이렇게 하면 특정 메시지에 사용한 키가 노출되어도 그 이전 메시지에 사용한 키는 알 수 없다. 물론 그 메시지 이후 사용된 키는 계산할 수 있다.

참고문헌

[1] Rolf Blom, "An Optimal Class of Symmetric Key Generation Systems," Advances in Cryptology, EUROCRYPT 84, LNCS 209, pp. 335–338, Springer, 1985.

[2] C. Blundo, A. D. Santis, A. Herzberg, S. Kutten, U. Vaccaro, M. Yung, "Perfectly Secure Key Distribution for Dynamic Conferences," Advances in Cryptology, CRYPTO 92, LNCS 740, pp. 471–486, 1993.

[3] Antoine Joux, "A One Round Protocol for Tripartite Diffie–Hellman," 4th Int'l Symp. on Algorithmic Number Theory, LNCS 1838, pp. 385–393, Springer, 2000.

[4] Roger Needham, Michael Schroeder, "Using Encryption for Authentication in Large Networks of Computers," Communications of the ACM, Vol. 21, No. 12, pp. 993–999, 1978.

[5] Dorothy E. Denning, Giovanni Maria Sacco, "Timestamps in Key Distributed Protocols," Communication of the ACM, Vol. 24, No. 8, pp. 533—535, 1981.

[6] W. Diffie, M. Hellman, "New Directions in Cryptography," IEEE Trans. on Information Theory, Vol. 22, No. 6, pp. 644–654, 1976.

[7] John. C. Mitchell, Mike Ward, Piers Wilson, "On Key Control in Key Agreement Protocols," Electronic Letters, Vol. 34, pp. 980–981, 1998.

퀴즈

1. $\{1, 2, \ldots, 10\}$은 곱하기 한 후 11로 나머지를 취하는 이항 연산에 의해 닫혀 있는 곱셈군이다. 이 군에서 5의 역원은?

   ① 2

   ② 8

   ③ 9

   ④ 3

2. 최근 키 확립 프로토콜은 키 분리 원리를 엄격하게 적용하여 용도마다, 방향마다 다른 키를 사용한다. 심지어 메시지마다 다른 키를 사용하기도 한다. 이와 관련된 설명 중 틀린 것은?

   ① 각 참여자가 유지해야 하는 키가 많아지지만 이들은 장기간 키가 아니라 일회용 키이므로 키 관리 비용이 증가하는 것은 아니다.

   ② 방향마다 다른 키를 사용할 경우 이들은 서로 독립적인 키이기 때문에 이들 중 하나가 노출되어도 다른 키의 노출로 이어지지 않는다.

③ 메시지마다 다른 키를 사용하기 위해 해시 함수에 기존 키를 입력하여 새 키를 계산하여 사용할 수 있다. 이 경우 키가 노출되면 과거 키는 계산할 수 없지만 노출된 이후 메시지에 사용된 키는 계산할 수 있다.

④ 최초 확립한 비밀키 또는 비밀 정보가 노출되어도 실제 사용하는 키에는 영향이 없다.

3. 공개키를 이용할 수 있으면 다음과 같이 비교적 간단하게 2자간의 세션키를 확립하여 교환하는 메시지의 비밀성을 보장할 수 있다. 이 방식과 관련된 설명 중 틀린 것은?

$$\text{Msg 1. } A \rightarrow B : C = \{M\}.K_1, \text{MAC}.K_2(C), \{T_A||K\}. + K_B$$

여기서 K_1과 K_2는 K로부터 계산한 대칭키이다.

① 자체 강화 방식이다.

② 키 전송 방식이다.

③ 세션키의 최근성을 보장할 수 없다.

④ 수신자는 송신자를 인증할 수 있다.

4. Diffie-Hellman 키 동의 프로토콜은 이산대수 문제, DH 계산 문제, DH 결정 문제가 어려운 유한순환군을 이용한다. 이와 같은 특성을 만족하는 유한순환군 G의 생성자 g가 있을 때 A와 B은 1에서 $|G|-1$ 사이의 수 a와 b를 임의로 선택한 후 g^a, g^b를 서로 교환하여 키를 확립한다. 공격자가 1에서 $|G|-1$ 사이의 수 c를 선택하여 g^c를 계산한 후에 중간에서 교환되는 값을 차단하고, 그 값 대신에 A와 B에게 g^c를 주었다고 하자. 그러면 A와 B이 각각 계산하는 비밀값은?

① A: g^{ab}, B: g^{ab}

② A: g^{bc}, B: g^{bc}

③ A: g^{ac}, B: g^{ac}

④ A: g^{ac}, B: g^{bc}

5. $\{1, 2, \ldots, 10\}$은 곱하기 한 후 11로 나머지를 취하는 이항 연산에 의해 닫혀 있는 군이다. 이 군의 원소 2는 이 군의 생성자이다. 이 군에서 기저 2에 대한 7의 이산대수는?

① 7

② 6

③ 8

④ 4

연습문제

1. 1.2 절에 다음과 같은 하나의 메시지를 이용하는 키 확립 프로토콜이 제시되어 있다. 이 프로토콜을 사용하기 위한 전제 조건은 무엇이며, 이 메시지를 받은 B이 해야 할 일은 무엇인지 구체적으로 서술하시오.

$$\text{Msg 1. } A \rightarrow B : \ C = \{M\}.K_1, \text{MAC}.K_2(C), \{T_A||K\}.+K_B$$

2. 키 확인(key confirmation)과 키 무결성(key integrity) 개념을 비교하시오. 이때 키 무결성보다는 키 확인이 중요한 이유와 그것이 보장되기 위한 조건이 무엇인지 설명하시오.

3. 법 7에서 곱셈을 이항연산으로 사용하는 곱셈군 $G = \{1, 2, 3, 4, 5, 6\}$이 있을 때, 이 군에서 3^{80}을 구하시오.

4. 법 11에서 곱셈을 이항연산으로 사용하는 곱셈군 $G = \{1, 2, \ldots, 10\}$에서 2의 역원을 구하시오.

5. 그림 6.5에서 주어진 프로토콜에서는 키 분리 원칙에 따라 확립된 비밀 정보 K_{AB}로부터 K_1부터 K_5까지 생성하여 모두 다른 용도로 사용한다. 심지어 A가 K_2와 K_3를 이용하여 인증 암호화하여 메시지를 교환할 때, 메시지마다 $K_i' = H(K_i)$를 이용하여 이전 키를 이용하여 새 키를 생성하여 사용한다. 이 경우 K_{AB}가 노출된 경우, K_i가 노출된 경우 추가로 노출되는 키를 나열하고, 그 이유를 설명하시오.