정보보호개론

제4장 암호프로토콜 개요

1. 암호프로토콜

프로토콜이란 **어떤 목적을 달성하기 위해 두 명 이상의 참여자(principal)가 정해진 약속에 따라 수행하는 일련의 단계**를 말한다. 이와 달리 혼자 어떤 목적을 달성하기 위해 어떤 정해진 절차를 수행하는 것은 알고리즘이라 한다. 컴퓨팅 분야에서 고려하는 프로토콜은 보통 통신 프로토콜이다. 통신 프로토콜은 어떤 목적을 달성하기 위해 원격에 있는 사용자들이 정해진 약속에 따라 메시지를 교환하는 프로토콜이다. 보통 프로토콜을 기술할 때에는 각 참여자가 교환하는 메시지의 형태뿐만 아니라 메시지를 수신하였을 때 참여자가 해야 하는 내부 행동(메시지가 도착하지 않은 경우, 잘못된 형태의 메시지가 수신된 경우 등)을 포함하여 기술해야 하는 것이 많다. 하지만 프로토콜을 요약하여 기술할 때에는 참여자들이 교환하는 메시지 형태만 나타낸다. 이 교재에서는 주로 메시지 형태만 나열하여 프로토콜을 설명한다.

프로토콜의 특성은 다음과 같다.

- 참여자들이 사전에 알고 있어야 한다.
- 참여자들 간에 동의하고 있어야 한다.
- 각 단계에서 해야 하는 일이 모호하지 않아야 한다.
- 프로토콜 수행을 정상적으로 완료하면 목표하였던 목적을 달성해야 한다.

위 4가지 특성 중 마지막 특성을 **완전성**(completeness) 요구사항이라 한다.

프로토콜은 동시에 병행으로 여러 참여자 간에 수행될 수 있다. 예를 들어 웹 브라우징할 때 브라우저와 웹 서버 간에 수행하는 프로토콜이 HTTP이며, 한 웹 서버는 동시에 여러 브라우저로부터 요청을 수행할 수 있다. 이때 프로토콜의 임의의 단일 실행을 **프로토콜 수행**(protocol run)이라 한다. 어떤 프로토콜 수행에서 교환한 메시지를 해당 수행의 **트랜스크립트**(transcript)라 한다.

암호프로토콜(cryptographic protocol)은 암호 기술을 사용하는 프로토콜을 말한다. 암호프로토콜에서 암호기술을 사용하는 이유는 프로토콜 수행에서 교환하는 각 메시지의 의미를 보장하고, 참여자 또는 제3자가 부정 행위를 하지 못하도록 하기 위한 것이다. 암호프로토콜은 일반 통신 프로토콜과 달리 **공격자(attacker)의 존재**를 가정한다. 일반 프로토콜은 누군가 고의로 일으키지 않은 오류가 발생하였을 때 프로토콜이 정상적으로 수행되도록 하는 메커니즘을 포함하고 있지만, 고의적인 공격으로부터 프로토콜을 보호하기 위한 메커니즘은 설계할 때 고려하지 않는다. 따라서 암호프로토콜은 완전성뿐만 아니라 **안전성**(secureness)을 보장해야 한다. 여기서 안전성이란 공격자가 존재하더라도 프로토콜의 요구사항을 충족하거나 부당한 이득을 얻는 자가 없어야 한다는 것을 말한다.

1.1 암호프로토콜에 대한 공격

암호프로토콜은 고의로 프로토콜을 공격하여 부당한 이득을 얻기 위한 공격자의 존재를 가정한다. 공격자는 영어 문헌에서는 다양한 용어로 표현된다. 이 중 가장 많이 사용하는 것이 attacker, adversary, eavesdropper 등이다. 암호프로토콜에 대한 공격은 크게 수동(passive) 공격과 능동(active) 공격으로 분류할 수 있다. 수동 공격은 프로토콜의 진행을 방해하지 않고 공격하는 것을 말하고, 능동 공격은 프로토콜의 진행에 개입(메시지 삭제(차단), 삽입, 변경,재전송)하여 공격하는 것을 말한다. 수동 공격은 교환된 메시지를 축적하여 암호해독이나 트래픽 분석을 하는 공격을 말하며, 안전한 암호알고리즘을 올바르게 사용하면 큰 위협이 되지 않는다. 반면에 능동 공격은 안전한 암호알고리즘을 올바르게 사용하더라도 프로토콜의 허점 때문에 큰 위협이 될 수 있다. 능동 공격을 방어하기 위한 기본은 참여자가 공격을 알아챌 수 있도록 하는 것이다.

2. 암호프로토콜 참여자

참여자는 실제 사용자를 의미할 수 있고, 사용자가 사용하는 장치 또는 소프트웨어일 수 있다. 참여자는 크게 일반 참여자와 제3의 신뢰 기관(TTP, Trusted Third Party)으로 구분한다. 일반 참여자는 프로토콜을 통해 얻고자 하는 것이 있는 이해 당사자이지만 TTP는 프로토콜의 실행 결과에 대한 어떤 이해관계가 없고, 어떤 참여자와도 특별한 협력 관계가 없지만, 프로토콜의 수행을 원활하기 위해 참여하는 참여자를 말한다. TTP를 다른 말로 중재자(arbitrator)라 한다. 암호프로토콜에서 **신뢰**(trust)한다는 것은 프로토콜에서 정한 규칙대로 프로토콜을 정직하게 수행한다는 것을 말한다. 하지만 어떤 부정 행위도 하지 않는다는 것은 아니다. 참고로 앞서 설명한 것처럼 이와 같은 참여자 외에 공격자가 존재할 수 있으며, 일반 참여자도 공격자가 될 수 있다. 특히 프로토콜의 이해 당사자들은 서로를 신뢰하지 않는다.

2.1 TTP

TTP는 크게 인라인(inline), 온라인(online), 오프라인(offline)으로 분류할 수 있다. 인라인과 온라인은 모든 프로토콜 수행마다 TTP가 참여해야 하며, 오프라인은 필요할 경우에만 또는 프로토콜의 시작 전이나 실행 후에 참여자와 상호작용한다. 프로토콜의 모든 메시지가 항상 TTP를 경유하면 인라인[1]이라 하며 그렇지 않고 프로토콜 수행마다 참여해야 하는 TTP는 온라인이라 한다.

우리가 프로토콜을 설계할 경우에 가장 이상적인 형태는 TTP를 사용하지 않고 이해 당사자 간의 프로토콜을 진행할 수 있도록 만드는 것이다. 이와 같은 형태의 프로토콜을 **자체 강화 프로토콜**(self-enforcing protocol) 이라 한다. 하지만 모든 경우에 자체 강화 방식으로 만들기 어렵다. 따라서 신뢰할 수 있는 TTP를 참여자로 활용하는 프로토콜을 많이 사용한다. 이 경우에도 온라인 형태보다는 오프라인 형태가더 바람직하다.

온라인 형태의 중재자를 사용하는 프로토콜의 경우에는 다음과 같은 문제점이 발생할 수 있다.

- 중재자가 동작하지 않거나 중재자와 다른 참여자 간의 통신이 중단되면 프로토콜 자체를 수행할 수 없게 된다. 이와 같은 문제를 **단일 실패점**(single-point-of-failure)이라 한다.
- 중재자를 실제 신뢰하기 힘들 수 있다. 특히, 특정 프로토콜의 목적을 고려하였을 때 해당 프로토콜에서 사용할 수 있는 신뢰 기관을 찾는 것이 어려울 수 있다.
- 중재자를 유지하고 관리하기 위한 추가 비용이 소요되며, 자체 강화 방식보다 통신 지연이 증가할 수밖에 없다.

[1]우리가 널리 사용하고 있는 카카오톡과 같은 메신저 서비스는 인라인 방식으로 사용자 간의 메시지를 교환한다.

- 모든 프로토콜 수행마다 관여하여야 하므로, 병목현상이 발생할 수 있으며, 단일 실패점이므로 서비스 거부 공격의 목표가 될 수 있다.
- 특정 참여자를 공격하면 해당 참여자 행세를 할 수 있지만, 그 피해는 한 참여자로 제한된다. 반면에 중재자를 공격하여 중재자 행세를 할 수 있으면 모든 참여자에게 피해를 줄 수 있다. 따라서 중재자는 좋은 공격 대상이 된다.

신뢰 기관과 신뢰 서버가 같은 것으로 생각할 수 있지만, 신뢰 서버는 컴퓨팅 서버로 신뢰 기관이 운영하는 서버이다. 즉, 일반적으로 신뢰는 소프트웨어, 서비스에 대한 것이 아니라 그것을 운영하는 기관에 대한 것이다.

3. 암호프로토콜의 설계 절차

암호프로토콜마다 달성해야 하는 목적이 다르므로 프로토콜마다 만족해야 하는 요구사항이 다르다. 따라서 암호프로토콜을 설계할 때 가장 먼저 해야 하는 것은 이 프로토콜을 통해 달성하기 위한 목적이 무엇인지 명확하게 정의하는 것이다. 암호프로토콜에 대한 정의는 어떤 환경에서 어떤 참여자들이 어떤 목적을 달성하기 위해 무엇을 이용하여 프로토콜을 수행하는지 등의 내용을 포함해야 한다.

프로토콜의 정의를 명확하게 도출한 후에는 프로토콜의 요구사항을 분석해야 한다. 요구사항 분석을 통해 프로토콜이 제공해야 하는 기능이나 안전성 수준을 더 구체적이고 세부적으로 기술해야 한다. 예를 들어 전자선거 프로토콜에서 투표자는 오직 한 번만 투표할 수 있어야 하는 요구사항이 있다.

요구사항 분석 후에는 프로토콜에서 사용할 가정을 나열하여야 한다. 이 가정에는 참여자에 대한 가정, 프로토콜을 수행할 환경에 대한 가정, 공격자에 대한 가정 등을 포함한다. 프로토콜을 수행할 환경에 대한 가정에는 사용하는 장치, 통신 환경 등이 포함된다.

참여자에 대한 가정에는 참여자 간의 신뢰 관계를 포함한다. 보통 이해 당사자들은 서로 신뢰하지 않는다고 가정한다. 예를 들어 고객과 인터넷 쇼핑몰 간에는 서로 신뢰하지 않는다. 프로토콜에 신뢰기관이 참여하는 경우에는 보통 일반 참여자는 신뢰 기관이 정해진 절차대로 프로토콜을 충실히 수행한다고 믿는다. 하지만 프로토콜에 따라 모든 절차나 기능을 신뢰하는 것이 아니라 일부에 대해서만 신뢰하는 경우도 있다.

공격자에 대한 가정은 기본적으로 다음을 포함한다.

- 가정 1. 공격자는 프로토콜을 통해 교환하는 모든 메시지를 얻을 수 있다.
- 가정 2. 공격자는 프로토콜의 진행을 방해할 수 있다. 특히, 메시지를 변경, 삽입, 차단할 수 있으며, 다른 목적지로 전달할 수 있다.
- 가정 3. 공격자는 프로토콜에 정상적으로 참여할 수 있는 사용자일 수 있고, 제3자일 수 있다.
- 가정 4. 오래된 세션키는 공격자에게 노출될 수 있다.
- 가정 5. 공모 공격을 할 수 있다.

가정 4와 관련 보통 프로토콜을 분석할 때 사용하는 키가 노출되면 어떤 문제가 발생할 수 있는지도 파악하게 되며, 그 파급효과가 최소가 되도록 프로토콜을 설계한다.

설계가 완료되면 주어진 가정 하에 요구사항이 모두 만족하는지 형식 증명이 필요하다. 하지만 프로토콜에 대한 형식 증명은 알고리즘에 대한 증명보다 더 복잡하고 어렵다.

<표 4.1> 암호프로토콜 기술에 사용되는 표기법

표기	의미	표기	의미
A,B,C,S	프로토콜의 참여자	$H(M)$	M에 대한 해쉬값
K	비밀키	$\{M\}.K$	암호키 K를 이용한 M의 암호화
$+K$	공개키	Sig.$A(M)$	M에 대한 사용자 A의 전자서명값
$-K$	개인키	Sig.$K(M)$	서명키 K를 이용한 M에 대한 전자서명값
T	시간을 나타내는 타임스템프	MAC.$K(M)$	비밀키 K를 이용한 M에 대한 MAC 값
N	난스	$M_1 \| \| M_2$	M_1과 M_2의 비트 결합

3.1 표기법

암호프로토콜을 기술할 때 이 교재에서는 주로 표 4.1에 제시된 표기법을 사용한다. 또한 아랫첨자를 사용하여 키의 소유자나 타임스템프나 난스를 생성한 사용자를 나타낸다. 예를 들어 K_{AB}는 사용자 A와 B가 공유한 비밀키를 나타내며, T_S는 신뢰기관 S가 생성한 타임스템프를 나타낸다.

4. 암호프르토콜의 예: 키 확립 프로토콜

암호프로토콜이 어떻게 설계되고 잘못 설계되었을 때 어떤 문제점이 발생할 수 있는지 쉽게 이해하기 위해 가장 기초적인 암호프로토콜인 키 확립 프로토콜을 이용하여 설명한다. 키 확립 프로토콜이란 둘 이상의 참여자가 비밀키를 공유할 수 있도록 해주는 암호프로토콜을 말한다. 키 확립 프로토콜을 통해 얻어지는 비밀키는 보통 단일 세션에 사용하기 위한 세션키이다. 이처럼 확립된 비밀키를 한 세션에만 사용하는 이유는 다음과 같다.

- 이유 1. 같은 키로 암호화된 암호문을 제한할 수 있다.
- 이유 2. 키가 노출되었을 때 누설되는 정보의 양을 제한할 수 있다.
- 이유 3. 키를 휘발성 메모리에 유지할 수 있다.
- 이유 4. 세션 간 또는 응용프로그램 간의 독립성을 제공할 수 있다.

위 이유에 대해 보충 설명하면 다음과 같다. 첫 번째 이유는 암호해독하는 것이 어려워지도록 하기 위함이다. 같은 키로 암호화된 암호문이 적으면 적을수록 암호해독하기 어렵다. 두 번째 이유는 첫 번째 이유의 결과이다. 장기간 키의 경우에는 비휘발성 메모리에 유지해야 하며, 이 키를 보호하기 위한 메커니즘이 정보보호 서비스에 매우 중요한 요소가 된다. 따라서 키를 사용하고 즉시 폐기하는 형태가 효율적이며 더 안전하다. 네 번째 이유는 다른 세션에서 사용한 암호문을 활용하여 다른 세션을 공격하는 것을 방지할 수 있다.

4.1 키 확립 프토토콜의 요구사항

키 확립 프로토콜은 원격에 있는 참여자 간에 안전하게 사용할 수 있는 비밀키를 확립하는 것이다. 따라서 키 확립 프로토콜의 요구사항은 다음과 같다.

- 요구사항 1. 참여자는 의도한 다른 참여자와 사용할 수 있는 새로운 키를 얻어야 한다.
- 요구사항 2. 새로운 키는 의도된 참여자 외에 다른 참여자는 얻을 수 없어야 한다.

Msg 1. $A \rightarrow S: \quad A, B$
Msg 2. $S \rightarrow A: \quad K_{AB}$
Msg 3. $A \rightarrow B: \quad A, K_{AB}$

<그림 4.1> 첫 번째 시도 키 확립 프로토콜

제시한 요구사항을 세부적으로 분석하여 보자. 의도한 다른 참여자란 **키의 용도**를 확인할 수 있어야 한다는 것을 말한다. 키의 용도란 누구와 사용하기 위한 키인지 확인하는 것이다. A가 B와 사용하고 싶은 키를 요청하여 받았을 경우 A는 받은 키가 B와 사용하기 위한 키인지 확인할 수 있어야 한다. 이때 생성 주체에 대한 인증이 필요할 수 있다. 키 확립 프로토콜에서 확립하는 키는 세션키이며, 세션키는 다른 세션과 독립성을 위해 이전에 사용하지 않은 새로운 키이어야 한다. 이를 **키 최근성**(freshness) 요구사항이라 한다. 확립한 키를 실제 사용하기 위해서는 참여자들이 같은 키를 가지고 있어야 한다. 이를 위해 보통 **키 확인**(confirmation) 과정이 필요하다. 키 확인 과정에서는 프로토콜에 따라 상대방이 실제 내가 통신하고자 하는 상대방인지 인증해야 할 수 있다. 두 번째 요구사항은 **키의 비밀성**이 요구된다는 것이다. 제3자들은 교환되는 메시지를 통해 확립한 키를 얻을 수 없어야 한다.

키 확립 프로토콜에서 세션키는 참여자 중 한 참여자가 홀로 생성하여 다른 참여자에게 전달할 수 있고, 여러 참여자가 함께 세션키 생성에 관여할 수 있다. 전자의 경우에는 세션키를 수신한 다른 참여자는 이 키가 프로토콜에서 정의한 해당 참여자가 생성한 것인지 확인할 수 있어야 한다.

4.2 키 확립 프로토콜의 설계

지금부터 다섯 번에 걸쳐 대칭 암호알고리즘만 사용하는 키 확립 프로토콜을 설계하고자 한다. 이 프로토콜의 참여자는 크게 일반 참여자와 키를 발급하는 신뢰 기관, 두 종류가 존재한다. 각 참여자는 신뢰 기관과 사전에 안전하게 비밀키를 공유하고 있다고 가정한다. 실제 우리가 사용하는 스마트폰은 통신사에서 발급하여 주는 USIM 카드가 설치되며, 이 카드에는 통신사와 공유하는 비밀키가 저장되어 있으며, 이 키를 통해 상호 인증한다.

4.2.1 첫 번째 시도

그림 4.1에 제시된 키 확립 프로토콜의 진행 흐름은 다음과 같다. A는 신뢰 기관 S에 자신이 A이며 B와 통신하기 위한 키가 필요하다고 요청하면 S는 새로운 키를 생성하여 A에게 회신하게 된다. A는 받은 키를 B에게 전달하여 둘 간의 동일한 키를 공유하게 된다. 이처럼 진행 흐름에 있어서 이 프로토콜은 문제가 없다. 하지만 이 프로토콜은 키 확립 프로토콜의 요구사항을 하나도 만족하지 못하고 있다.

키를 암호화하지 않은 상태로 전달하였기 때문에 키 비밀성을 보장할 수 없으며, 참여자는 수신한 키가 신뢰 기관이 만든 것인지도 확인할 수 없다. 또 A와 B 모두 받은 키가 상대방과 사용하기 위한 키인지 알 수 없으며, 이 키가 최근에 생성한 키인지도 확인할 수 없다. 그뿐만 아니라 서로 동일한 키를 가졌는지 확인하지 않고 있다. 그림 4.1 프로토콜에서 K_{AB}라는 표기에서 아랫첨자는 키의 원래 의도를 나타내기 위한 것이지 A나 B가 실제 수신하는 것은 랜덤한 비트 문자열이며, 해당 비트 문자열로부터 A와 B에 대한 어떤 정보도 얻을 수 없다.

4.2.2 두 번째 시도

그림 4.2에 제시된 키 확립 프로토콜은 첫 번째 시도와 달리 신뢰 기관 S가 A와 B와 각각 공유하고 있는 K_{AS}, K_{BS}로 키를 암호화하여 주고 있다. 이렇게 함으로써 프로토콜의 요구사항 중 키 비밀성은 만족하고 있다. 하지만 키 최근성, 키 용도, 생성 주체에 대한 인증, 키 확인 요구사항은 모두 만족하고 있지 못하다.

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

<그림 4.2> 두 번째 시도 키 확립 프로토콜

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

<그림 4.3> 세 번째 시도 키 확립 프로토콜

이렇기 때문에 키 용도와 관련하여 다음과 같은 공격이 가능하다. 첫째, 공격자 C가 메시지 3을 전달히지 못하도록 차단한 후에 아래와 같이 A를 C로 변경하여 B에게 전달하면 B는 수신한 키가 C와 사용하기 위한 키로 착각하게 된다.

$$\text{Msg } 3'.\ C \rightarrow B:\ C, \{K_{AB}\}.K_{BS}$$

둘째, 공격자가 첫 번째 메시지를 억압하고 B를 C로 바꾸어 신뢰 기관에 전달하면 서버는 다음과 같은 메시지를 A에게 전송하게 된다.

$$\text{Msg } 2'.\ S \rightarrow:\ \{K_{AC}\}.K_{AS}, \{K_{AC}\}.K_{CS}$$

이 메시지를 받은 A는 암호문에 포함된 키가 B와 사용하기 위한 키로 알게 된다. 하지만 해당 키는 C가 알고 있으며, C가 메시지 3을 억압하면 A는 C를 B로 착각하고 비밀 통신을 하게 된다.

키 최근성이 보장되지 않았기 때문에 다음과 같은 공격도 가능하다. 먼저 공격자가 이전 프로토콜 수행에서 교환된 메시지 2를 보관하고 있으며, 우연히 해당 수행에서 교환된 세션키를 알게 되었다고 가정하자. 이때 A가 새롭게 B와 키 확립을 위해 프로토콜을 시작하면 공격자는 메시지 1을 억압한 후[2]에 보관 중인 메시지 2를 A에게 전달하였다고 하자. A는 해당 메시지의 최근성을 확인할 수 없기 때문에 이전에 사용한 이미 노출된 키를 사용하게 되며, 공격자는 모든 비밀통신을 알게 된다. 이처럼 노출된 키를 사용하여 공격하는 것을 **기지 키 공격**(known-key attack)이라 하며, 과거에 수신한 메시지를 다시 해당 사용자에게 전달하여 공격하는 것을 **재 전송 공격**(replay attack)이라 한다.

4.2.3 세 번째 시도

그림 4.3에 제시된 프로토콜은 메시지 2의 암호문 내에 키의 용도를 나타내는 상대방 식별자를 포함하여 키의 비밀성뿐만 아니라 키의 용도와 서버가 생성한 것이라는 것을 확인할 수 있도록 하였다. 암호문을 복호화한 사용자는 식별자를 통해 자신이 사용한 복호화키가 맞다는 것을 확인할 수 있으며, 이 암호키는 본인과 신뢰 기관만이 알고 있는 키이므로 암호문 내에 있는 키는 신뢰 기관이 생성한 것임을 확신할 수 있다. 이렇게 식별자처럼 비밀성이 요구되지 않지만, 함께 암호화하여 복호화가 올바르게 된 것인지 확인할 수 있도록 해주는 정보를 **여분 정보**(redundant information)라 한다.

키 용도 때문에 시도 2에 대해 가능했던 공격은 더 이상 성공할 수 없다. 예를 들어 공격자가 메시지 3에서 암호문 외부에 있는 A를 C로 바꿀 수 있지만 암호문 내에 있는 식별자는 바꿀 수 없기 때문에 B는 이와 같은 공격을 알아챌 수 있다. 메시지의 1을 조작하는 공격도 메시지 2의 두 암호문에 포함하는 식별자 때문에 성공할 수 없다. 이처럼

[2]메시지 1을 억압하는 대신에 메시지 2를 억압하거나 교체할 수 있음

Msg 1. $A \rightarrow S:$ A, B, N_A
Msg 2. $S \rightarrow A:$ $\{B||N_A||K_{AB}\}.K_{AS}, \{A||K_{AB}\}.K_{BS}$
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 4.4> 네 번째 시도 키 확립 프로토콜

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

<그림 4.5> 다섯번째 시도 키 확립 프로토콜

식별자를 암호문 내에 포함하여 어떤 의미를 추가하는 것을 **명명 기법**(naming technique)이라 한다. 명명 기법의 사용에도 불구하고 이 프로토콜은 여전히 시도 2와 마찬가지로 기지키 재전송 공격에 취약하다.

4.2.4 네 번째 시도

그림 4.4에 제시된 프로토콜에서는 난스를 사용하여 A에게는 키의 최근성을 확인할 수 있도록 하고 있다. 이 프로토콜은 실제 Needham과 Schroeder가 제안한 프로토콜이다[1]. 난스는 한 번만 사용하는 값이기 때문에 두 번째 메시지에 있는 K_{AS}로 암호화되어 있는 암호문는 첫 번째 메시지를 보아야만 만들 수 있다. 이 때문에 최근성이 보장되지 않아 가능했던 공격은 더 이상 가능하지 않다. 예를 들어 이전에 사용한 키를 공격자가 알고 있고, 해당 키를 확립할 때 주고받은 메시지를 보관하고 있다고 하자. A가 다시 B와 세션키를 확립하고자 할 때 신뢰 기관 대신에 이전 메시지 2를 보내면 이전 메시지에 포함된 난스 값과 현재 난스 값이 같을 수 없기 때문에 A는 이 메시지가 재전송 공격이라는 것을 알 수 있다. 하지만 B는 시도 3과 마찬가지로 여전히 기지키 재전송 공격에 취약하다.

두 번째 메시지에서 난스 N_A 때문에 A가 최근성과 관련하여 확신할 수 있는 것은 K_{AS}로 암호화되어 있는 암호문 전체이고, 해당 암호문 내에 포함된 K_{AB}가 최근에 생성된 키임을 확신할 수는 없다. 이 확신은 신뢰를 이용해야 확신할 수 있다. 즉, 해당 암호문은 이번 요청에 대한 결과로 만들어진 암호문이고, 신뢰하는 서버가 만든 것이기 때문에 프로토콜 규칙에 의하면 서버는 이 메시지에 새로운 세션키를 생성해서 포함해야 하므로 A는 이를 믿을 수 있게 되는 것이다.

이 프로토콜은 메시지 1, 2와 나머지가 독립적인 측면이 있다. 따라서 동일 공격자가 메시지 2를 재전송하는 대신에 메시지 3을 B에게 재전송하면 기지키 재전송 공격에 성공할 수 있다.

이 프로토콜에서 메시지 4는 키 확인 과정이다. 하지만 여기서도 한쪽만 키 확인이 된다. A는 B가 자신과 같은 키를 가졌는지 알 수 없지만, B는 메시지 5를 통해 A가 자신과 같은 키를 가졌는지 확인할 수 있다. 메시지 5에서 더하기 연산을 한 이유는 메시지 4와 5를 구분하기 위함이다. 만약 메시지 4와 5가 같으면 K_{AB}가 없더라도 누구나 메시지 4를 재전송함으로써 메시지 5를 만들 수 있다.

4.2.5 다섯 번째 시도

그림 4.5에 제시된 프로토콜이 최종 시도이다. 실제 이 프로토콜은 Bauer 등이 제안한 프로토콜이다[2]. 이 프로토콜은 기존 4개의 시도와 달리 통신하는 흐름이 바뀌었다. 그 이유는 A와 B의 난스를 모두 신뢰기관에 전달해야 하기 때문이다. 메시지 3의 두 개 암호문에 각 사용자가 생성한 난스가 포함되어 있으므로 각 사용자는 수신한 키의 최근성을 확신할 수 있다. 또한 메시지 4의 두 번째 암호문과 메시지 5를 통해 A와 B는 모두 상대방이 자신과 동일한 키를 가졌는지 확인할 수 있다. 이처럼 키 확인 과정을 추가하는 것은 어렵지 않기 때문에 키 확인 과정 부분을 프로토콜 기술에서 생략하는 경우도 있다. 하지만 기술 과정에서 생략하였다고 필요 없는 것은 아니다.

4.2.6 블록 암호의 특징

세 번째 시도부터는 세션키만 암호화하지 않고 참여자의 식별자나 난스를 함께 암호화하고 있다. 이와 같은 암호문의 의미를 올바르게 이해하기 위해서는 사용하는 블록 암호의 특징을 이해해야 한다. 블록 암호는 정해진 고정된 크기의 입력을 고정된 크기의 출력으로 바꾸어주기 때문에 암호화해야 하는 평문의 크기가 블록 크기보다 작거나 크면 암호화 모드를 사용해야 한다. 이 절에서는 2장에서 살펴본 ECB 모드를 사용한다고 가정하고 이전 절에서 살펴본 프로로콜에 포함된 암호문을 해석해 보자.

세 번째 시도에서 S는 참여자의 식별자와 세션키를 함께 암호화하였다. 이때 평문을 C 프로그래밍 언어로 표현하면 다음과 같은 구조체로 정의할 수 있다.

```
1 struct sessionKeyMsg{
2     char id[16];
3     byte key[16];
4 }
```

우리가 사용하는 대칭 암호알고리즘의 블록 크기가 128bit(16byte)이고, 해당 알고리즘에서 사용하는 비밀키의 길이도 128bit라 가정하면 **sessionKeyMsg**는 두 개의 평문 블록으로 나누어 암호화해야 한다. ECB 모드를 사용한다고 가정하였고, 추가로 채우기를 고려하지 않으면 결과 암호문 $\{B||K_{AB}\}.K_{AS}$의 크기는 평문의 크기와 같다. 이를 $C_1||C_2$라 가정하자.

ECB는 각 평문 블록을 독립적으로 암호화하기 때문에 B를 암호화한 결과가 C_1이고, K_{AB}를 암호화한 결과가 C_2이다. C_i를 조작하거나 C_i를 원래 암호화할 때 사용한 키가 아닌 다른 키로 복호화하면 그 결과를 예측할 수 없다. 이 때문에 C_1을 복호화하여 기대하는 B를 확인하면 복호화한 사용자는 C_1이 어떤 키를 사용하여 암호화한 암호문인지 확신할 수 있다. 그러나 C_2는 랜덤한 값을 암호화한 암호문이므로 C_2가 어떤 키를 사용하여 암호화한 암호문인지 확신할 수 없다. 따라서 공격자가 C_2를 랜덤한 같은 크기의 다른 값을 바꾸더라도 A는 그것을 알아채기 어렵다.

더욱이 공격자가 이전에 사용한 암호문 $C_1'||C_2' = \{C||K_{AC}\}.K_AS$를 가지고 있으면 C_i 중 하나를 C_i'로 바꿀 수 있다. 따라서 이전 절에서 B와 K_{AB}가 함께 암호화되어 있기 때문에 K_{AB}의 용도를 확인할 수 있다는 것은 정확한 설명은 아니다. 실제 평문의 각 요소의 크기, 사용하는 암호알고리즘의 블록 크기, 사용하는 암호 모드에 따라 그것이 보장되지 않을 수 있다. 이 때문에 인증 암호화의 사용이 필요하다. 인증 암호화를 하면 무결성까지 확인할 수 있으며, 평문에서 각 요소의 크기, 사용하는 암호알고리즘과 무관하게 식별자나 난스를 세션키와 함께 암호화하여 그 키에 의미를 부여할 수 있다.

5. 암호기술과 암호프르토콜

암호프로토콜은 공격자가 존재하는 상황에서 프로토콜이 그 목적을 달성하거나 공격자가 부당한 이득을 취하지 못하도록 암호기술을 사용하며, 암호기술 중에 가장 기초적인 것이 암호알고리즘이다.

5.1 암호프로토콜과 암호알고리즘

암호알고리즘은 암호프로토콜의 각종 요구사항을 충족하기 위해 사용하는 가장 기본적이며 핵심 도구이다. 따라서 암호알고리즘의 안전성은 암호프로토콜의 안전성에 큰 영향을 주는 것은 당연하다. 하지만 암호프로토콜을 설계할 때에는 이상적인 암호알고리즘의 존재를 가정한다. 이를 통해 설계의 복잡성을 줄일 수 있다. 이상적인 암호알고리즘이란 각 암호알고리즘이 만족해야 하는 요구조건이 무조건적으로 만족한다는 것을 말하며, 프로토콜 설계 과정에서는 기능을 충족하는 블랙박스처럼 사용한다. 하지만 실제로 사용한 각 블랙박스가 어떤 수준의 안전성을 요구하는지는 구체적으로 기술해야 하며, 암호프로토콜을 구현할 때에는 반드시 고려해야 한다. 예를 들어 4.2 절에서 살펴본 키 확립 프로토콜은 대칭 암호알고리즘을 사용하고 있지만 어떤 대칭 암호알고리즘을 사용하는지 제시하지 않아도 프로토콜을 이해하거나 분석하는데 문제가 없다. 하지만 구체적으로 제시하지는 않았지만 NM 특성을 만족하는 대칭 암호알고리즘을 사용해야 한다. 만약 암호문에 포함되어 있는 식별자를 특정 식별자로 바꿀 수 있으면 키 용도와 관련하여 제시하였던 여러 공격이 여전히 가능하다.

5.2 암호화의 용도

프로토콜에서 메시지나 메시지의 일부를 암호화하는 이유는 비밀성이나 인증을 제공하거나 메시지 요소를 바인딩하기 위함이다. 통신 메시지의 일부 또는 전부를 비밀성을 위해 암호화하였을 경우 오직 의도된 수신자만이 복호화키를 가지고 있어야 기본적인 비밀성이 보장된다. 대칭 암호알고리즘의 경우에는 당연히 송신자도 복호화키를 가지고 있으며, 키를 분배한 신뢰 기관도 가지고 있을 수 있다. 하지만 비대칭 암호알고리즘의 경우에는 수신자만이 복호화키를 가지도록 할 수 있다. 예를 들어 A에게 비밀스럽게 메시지 M을 전달하고 싶으면 $\{M\}.+K_A$와 같이 A의 공개키를 이용하여 암호화하여 전달하면 된다. 4.2 절에서 살펴본 키 확립 프로토콜에서 서버는 세션키의 비밀성을 보장하기 위해 대칭 암호알고리즘을 사용하여 메시지를 암호화하고 있다.

어떤 특정 메시지를 자신이 생성 또는 전송한 것이라고 증명하기 위해 해당 메시지를 특정 키로 암호화할 수 있다. 이때에는 생성 또는 전송한 사용자만이 가지고 있는 암호키로 암호화하여야 한다. 대표적으로 공개키 방식에서 개인키를 이용하여 메시지를 암호화하여 전송하면 생성자나 전송자를 수신자에게 인증할 수 있다. 프로토콜 표기법을 이용하여 표현하면 $\{M\}.-K_A$와 같다. 참고로 모든 공개키 암호알고리즘이 같은 알고리즘을 이용하여 공개키 또는 개인키를 이용하여 암호화할 수 있는 것은 아니다. 예를 들어 RSA 암호알고리즘은 같은 알고리즘을 이용하여 공개키 또는 개인키로 암호화할 수 있지만 이산대수 문제에 기반하는 공개키 방식의 경우에는 같은 알고리즘을 이용하여 암호화할 수 없다. 보통 공개키로만 암호화할 수 있고, 개인키를 이용하여 인증을 제공하고 싶으면 전혀 다른 알고리즘을 사용해야 한다.

비밀키를 이용하는 대칭 암호알고리즘이나 MAC은 여러 사용자가 같은 암호키를 가지고 있기 때문에 인증 용도로는 메시지를 암호화할 수 없다고 생각할 수 있다. 하지만 A와 B만 알고 있는 비밀키 K_{AB}가 있을 때 $\{M\}.K_{AB}$나 $M, \mathrm{MAC}.K_{AB}(M)$을 A가 B에게 전달하였다고 하자. B가 해당 암호문이나 MAC 값을 본인이 생성한 적이 없다는 것을 확신할 수 있으면 A가 그것들을 만들었다고 확신할 수 있다. 4.2 절에 제시된 프로토콜에서는 암호문을 복호화한 후에 난스나 상대방 식별자를 확인함으로써 해당 암호문을 서버가 생성한 암호문임을 확신할 수 있다. 서버가 세션키를 생성하여 두 사용자에게 전달한 경우 해당 키는 실제 2명이 아니라 3명이 공유하는 키가 된다. 하지만 이 경우에도 서버를 신뢰할 수 있으면 상대방을 인증하는데 사용할 수 있다.

비밀성 보장이 필요하지 않은 것을 필요한 것과 함께 암호화하는 경우가 많다. 이것은 비밀성이 필요한 것에

어떤 의미를 부여하기 위한 것이다. 4.2 절에 제시된 세 번째 이후 프로토콜에서 사용한 암호문 $\{B||N_A||K_{AB}\}.K_{AS}$에서 비밀이 요구되는 정보는 K_{AB} 뿐이다. 나머지 B와 N_A는 K_{AB}의 용도와 최근성을 확인할 수 있도록 하기 위해 함께 암호화한 것이다. 이 때문에 K_{AB}와 같이 암호화된 식별자 B와 난스 N_A는 K_{AB}에 바인딩된 정보라 한다. 참고로 $\{B||N_A\}.K_{AS}$, $\{K_{AB}\}.K_{AS}$와 같이 전달하면 독립적으로 암호화된 정보는 서로 영향을 줄 수 없어 B와 N_A가 K_{AB} 해석에 어떤 영향도 줄 수 없다.

바인딩이 성립하기 위해서는 반드시 함께 암호화된 것임을 확신할 수 있어야 한다. 하지만 블록 암호 방식의 경우 정해진 블록 단위로 암호화할 수밖에 없으므로 4.2.6 절에서 설명한 것처럼 사용하는 암호화 모드에 따라 함께 암호화된 것인지 확신을 못할 수 있다. 이 측면에서 다음과 같이 해시함수나 MAC을 통해 바인딩을 제공했다면 사용한 암호화 모드와 무관하게 필요한 바인딩이 잘 되었다고 확신할 수 있다.

$$\{K_{AB}\}.K_{AS}, \mathrm{MAC}.K_{AS}(B||N_A||K_{AB})$$

A는 MAC 값을 통해 $\{K_{AB}\}.K_{BS}$에 포함된 키가 B와 사용하기 위해 신뢰 기관이 생성하여 준 최신 키임을 확신할 수 있다. 또 이와 같이 사용하면 K_{AB}의 무결성까지 확인할 수 있다.

하지만 이와 같은 방법이 키 추측 공격을 어렵게 하지는 않으며, MAC은 비밀성을 보장하지 않는다. 이 때문에 이 방법보다는 encrypt-then-mac 방법의 인증 암호화를 사용하는 것이 더 바람직하다.

$$C = \{B||N_A||K_{AB}\}.K_{AS}, \mathrm{MAC}.K'_{AS}(C)$$

위 예에서 제시한 것처럼 하나의 키를 여러 용도로 사용하지 않는 것이 바람직하다. 따라서 암호화할 때 사용하는 키와 MAC 키는 독립적인 서로 다른 키를 사용해야 한다. encrypt-then-mac 방법의 또 다른 이점은 무결성이 확인되지 않으면 복호화 자체를 하지 않는다. encrypt-then-mac 방법은 먼저 태그(MAC값)를 검증한 후에 태그가 유효하면 암호문을 복호화한다. 이 예에서 태그가 확인되고 본인이 생성한 것이 아니라는 것을 확신할 수 있으면 이 태그는 서버가 생성한 것이라고 확신할 수 있다. 또 태그가 확인되면 C는 전송 과정에서 조작되지 않았다는 것을 확신할 수 있다. 따라서 메시지의 최근성[3]만 추가로 확인되면 프로토콜의 약속에 따라 만들어진 정상적인 메시지임을 확신할 수 있다.

보통 비밀이 요구되는 정보에 바인딩되는 정보는 복호화는 측이 알고 있거나 기대하는 값이기 때문에 암호문을 복호화한 후에 이 값을 확인하여 이 암호문이 어떤 키로 암호화된 것인지 확인할 수 있다. $\{K_{AB}\}.K_{AS}$와 같이 랜덤한 값만 암호화된 암호문은 같은 크기의 랜덤한 값과 구분할 수 없다. $\{B||K_{AB}\}.K_{AS}$와 같은 암호문의 경우에는 K_{AS}로 복호화하여 B가 확인되면 대응되는 암호 블록은 반드시 K_{AS}로 암호화된 암호문임을 확신할 수 있다. 하지만 평문을 어떻게 구성하였고, 어떤 암호화 모드를 사용했는지에 따라 차이가 있지만 B가 포함되어 있지 않은 평문 블록에 대응되는 암호 블록은 그것이 K_{AS}로 암호화된 암호문임을 확신할 수 없다. 이 때문에 인증 암호화가 필요한 것이다. 인증 암호화하였다면 전송 과정에서 조작되지 않았다는 것을 확신할 수 있고, 송신자 또는 메시지 생성자를 신뢰할 수 있으면 복호화하여 B를 확인하지 않아도 해당 암호문은 해당 암호키를 암호화된 암호문임을 확신할 수 있다. 하지만 재전송된 메시지일 수 있으므로 인증 암호화를 한다고 키 용도나 최근성 확인을 위해 함께 암호화한 식별자나 난스를 생략할 수는 없다.

5.3 대칭과 비대칭 암호알고리즘의 사용

A와 B 간의 공유한 비밀키 K_{AB}로 메시지 M을 A가 암호화하여 B에게 전달하였다고 하자. 그러면 이 암호문은 메시지 M에 대한 비밀성을 제공할 수 있다. 사용된 암호알고리즘이 안전하면 통신으로 전달되는 $\{M\}.K_{AB}$를 누가 가로채더라도 M을 얻을 수 없다고 A와 B는 둘다 확신할 수 있다. A 입장에서는 오직 B만 M을 얻을 수 있다고 확신할 수 있다.

[3]이 프로토콜 수행을 위해 만들어진 메시지이고 프로토콜에서 이 순서를 위해 만들어진 메시지라는 것을 확인해야 한다.

이 암호문은 비밀성뿐만 아니라 인증도 제공할 수 있다. 수신한 B가 본인이 생성한 암호문이 아니라는 것을 확인할 수 있다면 B는 이 암호문을 A가 생성하였다고 확신할 수 있다. 물론 비밀키 방식에서는 통신의 참여자인 A와 B 외에 신뢰 기관도 K_{AB}을 알고 있을 수 있다. 그렇다 하더라도 신뢰 개념 때문에 확신에 대한 믿음은 여전히 성립한다. 여기서 다시 한번 강조해야 하는 것은 원격에 있는 두 사용자가 대칭 암호알고리즘을 이용하기 위해서는 먼저 안전하게 비밀키를 공유하여야 한다. 하지만 이것은 간단한 문제가 아니다.

대칭 암호알고리즘과 달리 공개키 암호알고리즘은 알고리즘에 따라 메시지를 공개키로도 암호화할 수 있고 개인키로도 암호화할 수 있다. A가 본인의 개인키 $-K_A$로 메시지 M을 암호화하여 전달하면 중간에서 누구든지 가로채어 A의 공개키를 이용하여 복호화함으로써 M을 얻을 수 있다. 따라서 개인키를 이용한 암호화로는 메시지의 비밀성을 보장할 수 없다. 하지만 누구나 동일한 과정을 통해 A가 생성한 메시지임을 확신할 수 있다. 반대로 A가 B의 공개키로 메시지 M을 암호화하여 전달하면 공격자가 이 메시지를 가로채더라도 M을 얻을 수 없으므로 비밀성이 보장된다. 공개키 방식에서는 개인키나 공개키를 사용하기 위해서는 먼저 공개키가 누구의 키인지 확인할 수 있는 메커니즘이 필요하다. 참고로 대칭이나 비대칭 암호알고리즘은 기본적으로 무결성을 제공해주지는 않는다.

6. 암호프로토콜의 안전성

암호프로토콜은 그것의 보안 요구사항을 모두 충족하였을 때 안전한 프로토콜이라 한다. 하지만 암호프로토콜은 안전한 것만으로는 부족하고, 사용하는 응용에서 요구하는 효율성도 충족해야 한다. 보통 안전성과 효율성은 상반관계이다. 즉, 안전성을 높이면 효율성이 떨어지고, 효율성을 높이면 안전성이 줄어든다. 그러나 응용에 따라서는 안전성을 희생하는 것이 절대 용인되지 않을 수 있다.

암호알고리즘은 충족해야 하는 요구사항이 알고리즘의 종류에 따라 고정되어 있다. 하지만 암호프로토콜은 프로토콜마다 충족해야 하는 요구사항이 매우 다양하며, 하나의 프로토콜이 만족해야 하는 요구사항이 매우 많을 수 있다. 따라서 암호프로토콜의 안전성을 증명하는 것은 암호알고리즘과 비교하였을 때 상대적으로 어렵다. 이 때문에 보통 지금까지 알려진 공격에 대해서만 안전하거나 여러 가정 하에서만 증명이 가능한 경우도 있으며, 특정 요구사항 충족에 대해서만 증명이 가능한 경우도 있다.

암호프로토콜이 안전하다고 하여 전체 시스템이 안전한 것은 아니다. 시스템의 모든 요소가 같은 수준의 안전성을 제공하여야 시스템이 안전하다고 할 수 있다. 또한 서비스 거부 공격 등 프로토콜의 설계를 통해서 방지할 수 없는 보안 위협도 많다.

6.1 안전성 증명

암호프로토콜의 안전성 증명은 형식에 있어서는 암호알고리즘과 유사하다. 먼저 해당 암호프로토콜이 충족해야 하는 요구사항을 나열한다. 즉, 설계할 때 분석한 내용을 정리한 다음 이를 바탕으로 증명하기 위한 보안 모델을 수립한다. 보안 모델이란 프로토콜의 안전성을 논하기 위해 세우는 가정의 집합을 말하며, 이와 같은 가정 중에 가장 중요한 것은 공격자의 능력이다. 따라서 보안 모델을 다른 말로 공격자 모델(adversary model, threat model)이라 한다. 이 모델에서 정의한 공격자가 존재하더라도 각종 요구사항이 충족됨을 증명하여야 한다. 여기서 공격자는 원래 프로토콜에 참여할 수 있는 참여자일 수 있고, 참여할 수 없는 제3자일 수 있다. 전자를 내부 공격자라 한다.

가정하는 공격자의 능력에 따라 특정 공격의 성공 가능성이 다르다. 또한 특정 공격을 논할 때 항상 가정 하는 공격자의 능력도 있다. 당연하지만 공격자의 능력이 강력할수록 공격이 성공할 가능성은 커지며, 이와 같은 상태에서도 프로토콜이 안전하다고 증명할 수 있으면 프로토콜의 안전성 수준이 매우 높다는 것을 의미한다. 또 참여자들이 공모하게 되면 공격이 성공할 가능성이 높아진다. 따라서 안전성 증명에서는 공모 공격에 대해서도 반드시 고려하여야 한다.

7. 암호프로토콜의 효율성

암호프로토콜에서는 크게 계산 효율성(computational efficiency), 통신 효율성(communications efficiency), 두 가지 측면에서 효율성을 고려한다. 계산 효율성은 프로토콜의 참여자가 프로토콜을 완료하기 위해 계산해야 하는 양에 의해 측정된다. 이 양은 사용하는 암호기술에 의해 결정되는데, 상대적으로 차이가 나는 연산을 사용할 경우에는 가장 비용이 많이 소요되는 연산을 사용하여 측정 또는 비교한다. 예를 들어 공개키와 비밀키를 같이 사용하는 프로토콜이라면 공개키 연산을 몇 번 하는지 계산하여 효율성을 논할 수 있다. 따라서 공개키 연산을 사용하는 프로토콜은 공개키 연산을 최소화할 필요가 있다. 또 사용하는 공개키 암호알고리즘 특성에 따라 효율성을 측정하는 방법이 달라질 수 있다. 예를 들어 RSA 공개키 암호알고리즘의 경우에는 개인키를 사용하는 연산보다 공개키를 사용하는 연산이 더 효율적이기 때문에 어떤 키를 이용하는지를 고려하여 비교 분석하는 것이 필요할 수 있다. 프로토콜의 효율성은 기술의 발달에 따라 변할 수 있기 때문에 이에 대한 고려도 필요하다.

통신 효율성은 메시지의 수와 각 메시지의 크기에 의해 측정한다. 통신 효율성에서는 필요한 라운드(round)의 수를 줄이는 것이 가장 효과적이다. 즉, 메시지 크기를 늘리는 대신 라운드 수를 줄일 수 있다면 효율성을 높일 수 있다는 것이다. 한 라운드에는 한 시점에서 병렬로 전달할 수 있는 모든 메시지를 포함한다. 만약 메시지가 서로 독립적이지 않으면 서로 다른 라운드에 포함해야 한다. 예를 들어 4.2 절에서 시도 4의 프로토콜에서 메시지 2는 1의 난스 값이 있어야 만들 수 있으므로 다른 라운드에 포함된다.

프로토콜에서 사용하는 단말의 특성에 따라 효율성이 매우 강조되는 경우도 있다. 예를 들어 무선 통신 기반 이동 휴대 단말은 전원이 매우 중요한 자원이다. 전원 소모를 줄이면 수명을 연장할 수 있으므로 전원 소모를 최소화할 수 있도록 프로토콜을 설계한다. 참고로 이동 단말에서 전원 소모에 가장 많은 영향을 주는 것은 무선 메시지 전송이다.

전원 소모를 줄이기 위해 내부 연산을 줄이는 방법 중 사전 계산(pre-computation)을 이용하는 방법이 있다. 사전 계산이란 단말이 온라인 상태에서 수행해야 하는 연산을 고정 전원을 사용할 때 미리 계산해 놓은 후 사용하는 것을 말한다. 사전 계산할 수 있는 것은 제한적일 수 있으며, 저장 공간 요구가 높아지는 단점도 있다. 또 다른 방법으로는 이동 단말에서 해야 하는 연산을 서버로 옮기는 방법도 있다.

참고문헌

[1] 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.

[2] R.K. Bauer, T.A. Berson, R.J. Feiertag, “A Key Distribution Protocol using Event Markers,” ACM Transactions on Computer Systems, Vol. 1, No. 3, pp. 249—255, 1983.

퀴즈

1. ECB 모드를 이용한 블록 방식 대칭 암호의 특징이 아닌 것은? (인증 암호화를 하지 않은 경우를 물어보는 것임)
   ① 랜덤한 값만 암호화한 암호문 블록이 있을 경우, 그것을 복호화하며 얻은 평문을 통해 복호화하는 측이 알 수 있는 것은 없다.
   ② 두 개의 암호문 블록을 수신하였을 경우, 한 블록을 통해 얻은 확신을 다른 블록을 해석하는데 사용할 수 있다.
   ③ 잘못된 키로 복호화하였을 때 우연히 식별 가능한 정보를 얻을 확률은 무시할 정도로 작다.
   ④ 특정 블록을 복호화하였더니 기대한 값을 얻었다. 그러면 이 암호문은 복호화한 키로 암호화된 암호문이다.
2. 키 확립 프로토콜을 통해 확립하는 대칭키는 보통 단일 세션에서만 사용하기 위한 세션키이다. 이처럼 확립한 키의 사용을 단일 세션을 제한하는 이유가 아닌 것은?

① 같은 키로 암호화된 암호문을 제한할 수 잇다. 이것은 암호 해독을 어렵게 한다.

② 키가 노출되었을 때 누설되는 정보의 양을 제한할 수 있다.

③ 키를 확립하는 비용이 저렴하기 때문이다.

④ 키를 휘발성 메모리에 유지할 수 있어, 키를 보호하기 위한 메커니즘이 단순해진다.

3. 일반 암호화 대신에 encrypt-then-mac 방식의 인증 암호화를 하였다. 이를 통해 얻어지는 장점이 아닌 것은?

① 전송 과정에 조작되면 이를 발견할 수 있다.

② 메시지의 최근성을 확인할 수 있다.

③ 일반 암호화는 사용하는 암호화 모드와 평문의 구성에 따라 바인딩이 제공되지 않을 수 있지만 인증 암호화는 상대방을 신뢰할 수 있으면 여러 암호 블록이 결합된 것이 아니라는 것을 믿을 수 있기 때문에 바인딩이 보다 더 확실히 제공된다.

④ 암호문을 복호화하여 평문의 내용을 확인하지 않아도 상대방을 신뢰할 수 있으면 암호문이 특정 암호키로 암호화된 암호문임을 믿을 수 있다. 단, 상대방이 인증 암호화할 때 사용하는 MAC키와 암호화키는 사용하는 프로토콜에 의해 지정되어 있다.

4. 제3의 중재자를 인라인 또는 온라인 방식으로 사용하는 암호프로토콜의 문제점이 아닌 것은?

① 중재자가 단일 실패점이 된다.

② 참여자들이 신뢰할 수 있는 컴퓨팅 서버를 구축하는 것이 어렵다.

③ 모든 프로토콜에 관여해야 하므로 병목현상이 발생할 수 있다.

④ 특정 참여자를 공격하여 성공하면 그것의 피해는 한 참여자로 제한되지만 중재자를 공격하여 중재자 행세를 할 수 있으면 모든 참여자에게 피해를 줄 수 있기 때문에 좋은 공격 대상이 된다.

연습문제

1. 암호프로토콜과 일반 통신프로토콜과의 차이점을 설명하시오.

2. 그림 4.4에서 A가 두 번째 메시지를 수신하였을 때 N_A를 통해 수신한 K_{AB}의 최근성은 실제 확인할 수 없다. 그러면 A가 N_A를 통해 무엇의 최근성을 확인할 수 있는지 설명하시오.

3. 다음은 4.2 절에서 살펴본 신뢰하는 키 분배 서버를 이용하는 키 확립 프로토콜이다.

$$\begin{array}{lll} \text{Msg 1.} & A \rightarrow S: & A, B \\ \text{Msg 2.} & S \rightarrow A: & \{B||T_S||K_{AB}\}.K_{AS}, \{A||T_S||K_{AB}\}.K_{BS} \\ \text{Msg 3.} & A \rightarrow B: & \{A||T_S||K_{AB}\}.K_{BS}, \{A||T_A\}.K_{AB} \\ \text{Msg 4.} & B \rightarrow A: & \{T_B||B\}.K_{AB} \end{array}$$

여기서 T_X는 참여자 X가 포함한 현재 시각을 나타내는 타임스템프이다. 이와 관련하여 다음 각각에 대해 답변하시오.

① A와 B는 수신한 키 K_{AB}가 이번 요청을 위해 새롭게 생성한 키임을 어떤 조건 하에 확신을 가질 수 있는지 설명하시오.

② A와 B는 상대방이 자신과 같은 키를 가지고 있는지 확신할 수 있는지 설명하시오.

③ 그림 4.5의 프로토콜과 비교하여 차이점을 설명하시오.

4. 다음과 같은 간단한 프로토콜을 생각하여 보자.

$$\begin{array}{lll} \text{Msg 1.} & A \rightarrow B: & \{N_A\}.+K_B \\ \text{Msg 2.} & B \rightarrow A: & N_A \end{array}$$

이와 관련하여 다음 각각에 대해 답변하시오.

① A는 메시지 2에서 N_A를 수신하면 무엇을 확신할 수 있는지 설명하시오.

② 2번 메시지를 $\text{Sig}_B(A||N_A)$로 변경하면 A는 이를 통해 무엇을 확신할 수 있는지 설명하시오.

5. 전자선거 시스템을 만들고자 한다. 전자선거 시스템이 갖추어야 하는 보안 요구사항을 나열하시오.