정보보호개론

제10장 키 확립 프로토콜 대표 사례

1. 키 확립 프로토콜 수행 비용 절감 방안

일반적으로 키 확립 프로토콜을 통해 확립된 비밀키는 한 통신 세션에서 사용하고 버리는 일회용 키이다. 하지만 필요할 때마다 매번 키 확립 프로토콜을 수행하는 것이 효율성 측면에서 환경에 따라 부담이 될 수 있다. 더욱이 자체 강화 방식이 아니라 서버를 활용하는 경우에는 가용성 측면에서도 문제가 될 수 있다. 이를 극복하는 방안으로 한 번 확립한 비밀키를 여러 번 사용하는 방식과 한 번에 하나의 비밀키를 확립하는 것이 아니라 여러 개를 확립하는 방식이 제 안되었다. 전자를 **티켓(ticket) 방식**이라 하고, 후자를 **다중 벡터 방식**이라 하는데, 두 방식 모두 확립된 비밀키를 장기간 유지해야 하는 단점이 있다. 즉, 추가적인 키 관리 및 저장 공간이 요구된다. 또한 티켓 방식의 경우 같은 비밀키를 여러 번 사용하게 되므로 안전성도 근본적으로 약해진다. 참고로 꼭 대칭키 암호알고리즘을 사용하는 방식에서만 사용할 수 있는 기법은 아니지만 두 기법 모두 보통 대칭키 암호알고리즘만 사용하는 환경을 위해 제안된 기법이다.

1.1 티켓 기반 프로토콜

세션마다 비밀키를 새로 확립하는데 소요되는 비용을 줄이기 위해 확립된 비밀키를 일정한 기간 사용할 수 있도록 하는 방식을 티켓 방식이라고 한다. 이 방식에서는 비밀키, 비밀키의 용도, 사용 기간과 같은 정보를 암호화하여 만든 티켓을 사용한다. 여기서 비밀키의 용도는 이 티켓을 사용할 수 있는 사용자를 제한하기 위한 것이며, 티켓의 용도는 티켓을 생성할 때 사용하는 암호키에 의해 결정된다. 티켓 방식 프로토콜에는 티켓을 발급하는 자, 티켓을 발급받는 자, 티켓을 받고 서비스를 제공하는 자, 3종류의 참여자로 구성된다.

보통 티켓 방식에서 티켓은 사용자가 발급받아 유지하며, 서비스를 받고자 할 때 서비스 제공자에게 제시하고 해당 티켓을 사용할 수 있는 사용자임을 증명한다. 이 증명은 티켓 내에 포함된 비밀키를 알고 있다는 것을 통해 이루어진다. 하지만 사용자는 티켓을 보통 복호화할 수 없으므로 티켓을 발급 받을 때 티켓 뿐만 아니라 티켓에 포함된 비밀키를 받아야 하며, 이 키를 티켓과 함께 안전하게 유지해야 한다. 서비스 제공자는 티켓을 수신하면 티켓의 유효성과 사용자가 제시한 증명을 확인하기 위해 티켓을 복호화할 수 있어야 한다. 따라서 티켓은 보통 티켓 발급자와 서비스 제공자 사이에 공유된 비밀키를 이용하여 암호화한다.

티켓의 전형적인 모습은 다음과 같다. 이 티켓은 사용자 A가 B의 서비스를 사용하기 위해 S로부터 발급받은 티켓이다.

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

여기서 T_S는 티켓의 시작 시각이고, L은 티켓의 수명이며, 이 두 정보를 통해 티켓의 유효기간이 정의된다. 또 티켓에 포함된 식별자는 이 티켓을 사용할 수 있는 사용자의 식별자이다. 티켓은 타임스탬프를 사용하고 있기 때문에 시스템 간 시간 동기화가 필요하다. 이와 같은 형태의 티켓을 사용하기 위해서는 티켓을 발급하는 서버는 응용 서버와 공유된 대칭키를 가지고 있어야 한다.

Msg 1. $A \rightarrow S:$ A, G, N_A
Msg 2. $S \rightarrow A:$ $\{K_{AG}||G||N_A\}.K_{AS}, \{T_S||L||K_{AG}||A\}.K_{GS}$
Msg 3. $A \rightarrow G:$ $\{T_S||L||K_{AG}||A\}.K_{GS}, \{A||T_A\}.K_{AG}, B, N'_A$
Msg 4. $G \rightarrow A:$ $\{K_{AB}||B||N'_A\}.K_{AG}, \{T_G||L||K_{AB}||A\}.K_{BG}$
Msg 5. $A \rightarrow B:$ $\{T_G||L||K_{AB}||A\}.K_{BG}, \{A||T'_A\}.K_{AB}$
Msg 6. $B \rightarrow A:$ $\{T'_A\}.K_{AB}$

<그림 10.1> Kerberos V5 프로토콜

1.1.1 Kerberos

커버로스(Kerberos)는 MIT에서 개발한 네트워크 인증시스템이며 ITEF에서 인터넷 표준 RFC4120으로 채택하고 있다[1]. 원격에서 유닉스 서버가 제공하는 telnet, ftp, pop과 같은 네트워크 서비스에 접근할 때 널리 사용한 적도 있지만 현재는 ssh와 같은 다른 방법을 더 많이 사용한다. 이와 같은 네트워크 서비스는 계정명과 패스워드를 통해 사용자를 인증하지만, 이들은 기본적으로 인증 과정에 대한 어떤 보안도 하지 않는다. 따라서 개방된 네트워크로 전달되는 패킷 분석을 통해 쉽게 다른 사용자의 패스워드를 알 수 있다. 이 문제를 해결하기 위해 개발된 것이 커버로스이다.

커버로스는 Needham-Schroeder 프로토콜에 기반하고 있으며, 버전 5까지 개발되어 있다. 커버로스는 전형적인 티켓 방식 프로토콜로 참여하는 시스템 간 클럭 동기화가 필요하다. 커버로스에는 인증서버(Authentication Server), 티켓승인서버(TGS, Ticket Granting Sever), 응용서버 총 3종류의 서버가 참여한다. 인증서버는 사용자를 인증해주는 서버로서, 각 사용자와 장기간 비밀키를 공유하고 있다. 이 서버는 사용자에게 티켓승인서버와 사용할 수 있는 티켓을 발급하여 주며, 티켓승인서버는 응용서버와 사용할 수 있는 티켓을 발급하여 준다. 따라서 티켓승인서버는 각종 응용서버와 장기간 비밀키를 공유하고 있어야 한다. 응용서버는 각종 서비스를 제공하는 실제 사용자가 접속하고자 하는 서버이며, 사용자가 유효한 티켓을 제시하면 접속을 허용한다.

실제 티켓승인서버를 사용하지 않고 인증서버가 응용서버와 사용하기 위한 티켓을 발급할 수 있다. 하지만 커버로스에서는 티켓승인서버를 추가적으로 사용하고 있다. 그 이유는 사용자의 장기간 키의 사용을 최소화하기 위한 것이다. 사용자는 인증서버에 접속하여 자신을 인증한 이후에는 티켓요청 티켓의 유효기간 동안에는 외부적으로는 티켓과 티켓에 포함된 비밀키만 사용한다. 여러 응용서버에 접속이 필요하더라도 해당 티켓의 유효기간 동안에는 장기간 키의 사용이 필요 없으며, 인증서버와 상호작용할 필요도 없다. 이처럼 한 번 인증한 이후 일정 기간 동안 인증 없이 계속 여러 서비스를 사용할 수 있게 해주는 것을 SSO(Single-Sign On) 서비스라 한다.

커버로스 V5 프로토콜은 그림 10.1과 같다. 여기서 S는 인증서버이고, G는 티켓승인서버이다. 이 프로토콜 서술에서 알 수 있듯이 사용자는 티켓의 내부 내용을 볼 수 없다. 따라서 사용자는 티켓뿐만 아니라 티켓과 함께 받은 암호문을 함께 유지해야 한다. 이 암호문에 티켓에 포함된 비밀키가 들어 있다. 티켓을 제시할 때에는 티켓을 사용할 수 있는 사용자임을 증명하기 위해 티켓에 포함된 비밀키를 이용하여 자신의 식별자와 현재 타임스탬프를 암호화하여 준다.

1.1.2 Kehne 등의 프로토콜

그림 10.2에 제시된 Kehne 등[2]의 프로토콜은 티켓 기반 방식임에도 불구하고 시스템 간의 시간 동기화가 필요하지 않다. 이 프로토콜에서는 인증서버나 티켓승인서버 등이 티켓을 발급하지 않고 응용서버가 직접 발급한다. 응용서버가 직접 발급하기 때문에 티켓의 유효기간을 설정하고 확인하는 주체가 동일하다. 따라서 시스템 간의 시간 동기화가 필요 없으며, 만료 시간만 티켓에 포함하여도 유효기간을 확인하는 데 문제가 없다. 그뿐만 아니라 티켓을 직접 발급하기 때문에 티켓을 암호화하는 키를 다른 주체와 공유할 필요가 없다. 이 때문에 K_{BB}로 표현한

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

Msg 1. $A \rightarrow B:$ $N_A, \{T_B||A||K_{AB}\}.K_{BB}$
Msg 2. $B \rightarrow A:$ $N_B, \{N_A\}.K_{AB}$
Msg 3. $B \rightarrow A:$ $\{N_B||N_A\}.K_{AB}$

<그림 10.2> Kehne 등의 프로토콜

것이다. 또 프로토콜의 흐름을 보면 사용자가 바로 응용서버에 접속한다. 이 때문에 커버로스와 달리 사용자가 인증 서버나 티켓승인서버에 별도 접속할 필요가 없어진다. 따라서 사용자에게 더욱 투명하게 서비스(기존과 차이 없이) 제공이 가능하다. 물론 사용자는 인증 서버와 장기간 키를 사전에 공유하고 있어야 한다.

1.2 다중 벡터 방식

티켓 방식은 한 번 확립된 비밀키를 여러 번 사용하며, 사용자가 티켓을 유지하고, 응용 서버는 매번 티켓을 받아 검증하여야 한다. 반면에 다중 벡터 방식은 키 확립을 할 때 하나의 비밀키를 확립하는 것이 아니라 여러 개의 비밀키를 확립하여 사용하며, 각 비밀키는 한 세션에만 사용하고 버린다. 이 때문에 사용자가 여러 개의 비밀키를 유지해야 하는데, 소형 단말과 같이 메모리가 제한된 환경에서는 이것이 부담될 수 있다. 이 문제 때문에 티켓 방식과 정반대로 응용 서버가 유지하는 형태를 사용할 수 있다.

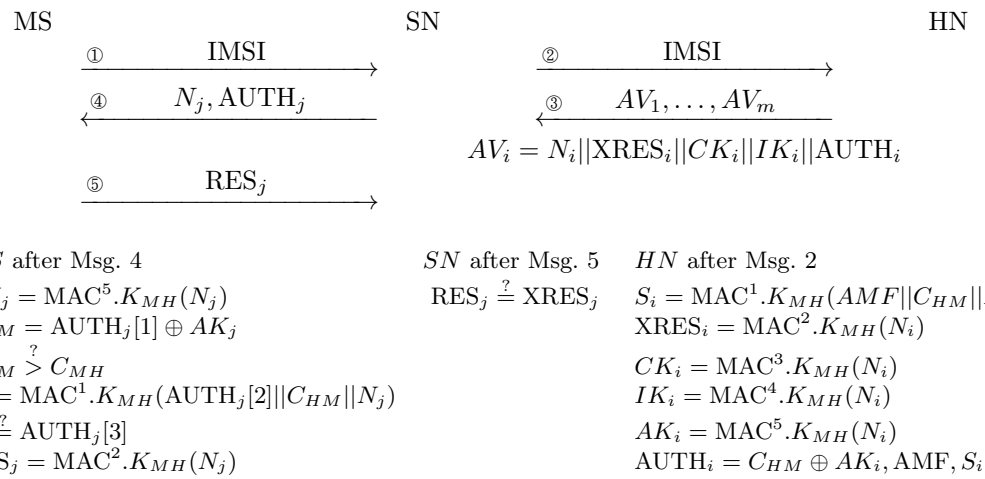
최근 키 확립 프로토콜 추세를 보면 하나의 키를 확립한 후 이 키로부터 여러 개의 독립적인 키를 생성하여 여러 용도로 사용한다. 실제 이때 생성할 수 있는 키 개수는 제한이 없다. 따라서 다중 벡터 방식을 사용하는 대신에 기존 키 확립 프로토콜을 통해 확립된 단일 세션키를 이용하여 여러 개의 키를 만들어 사용할 수 있다. 참여자는 확립한 단일 세션키를 유지하여 필요할 때마다 정해진 약속에 따라 세션키를 생성하여 사용하거나 여러 개를 미리 생성하여 유지한 후에 정해진 순서에 따라 사용할 수 있다. 하지만 다중 벡터 방식은 이와 같은 방식을 사용하는 것이 아니라 티켓 방식에서 티켓과 같은 것을 여러 개 만들어 사용한다. 다른 점은 이 티켓은 여러 번 사용하는 것이 아니라 한 번만 사용한다.

1.2.1 UMTS AKA

UMTS(Universal Mobile Telecommunication System)는 유럽에서 사용하였던 3G 이동통신 표준이며, UMTS는 단말 인증을 위해 AKA(Authentication and Key Agreement) 프로토콜을 사용한다[3].

UMTS에서 각 이동 단말(MS, Mobile Station)은 IMSI(International Mobile Subscriber Identifier)라는 독특한 식별자를 가진다. 프라이버시를 위해 IMSI를 한 번 사용한 이후에는 일정 기간 TMSI(Temporary MSI)를 이용할 수 있다. 하지만 해당 기간에는 TMSI를 변경하지 않고 계속 사용하기 때문에 불연결성을 제공하지 못한다. MS는 특정 홈네트워크(HN, Home Network)에 소속되어 있으며, 해당 네트워크의 HLR(Home Location Register)에 사용자와 기기 정보가 저장되어 있고, AuC(Authentication Center)와는 장기간 대칭키와 카운터를 공유하고 있다. 유럽에서는 국가 간 이동이 자유롭기 때문에 이동 단말이 홈네트워크가 관장하는 지역을 벗어날 수 있다. MS 입장에서 홈네트워크가 아닌 지역을 외부 네트워크(foreign network) 또는 SN(Serving Network)이라 한다.

어떤 MN이 외부 네트워크에 진입하면 해당 네트워크의 사용자와 기기 정보를 유지하는 VLR(Visiting Location Register)은 MN을 인증할 수 있는 정보가 없으므로 MN의 IMSI를 이용하여 MN의 홈네트워크 HLR에게



<그림 10.3> UMTS AKA 프로토콜

<표 10.1> 티켓 방식과 다중 벡터 방식의 비교

	티켓 방식	다중 벡터 방식
공통점	키 확립 후 일정기간 동안(일정 회수만큼)은 인증서버와 통신하지 않고 서비스 이용 가능	
발급시점	하나의 티켓만 발급	여러 개의 벡터 발급
통신비용		발급시점에 대여폭을 많이 소모
저장공간	티켓 하나	여러 개의 벡터
사용방법	유효기간 동안 동일 티켓 사용	한 번에 하나의 벡터만 사용
안전성	하나의 비밀키를 여러 세션 동안 사용	하나의 비밀키는 오직 한 세션에서만 사용
시간 동기화	보통 필요	벡터 사용을 1회로 제한하기 위해 카운터 사용

인증 정보를 요청한다. SN에서 HN에 접속하는 비용을 줄이기 위해 HN의 AuC는 한 번에 m개의 인증 벡터(AV, Authentication Vector)를 SN에게 전달한다. SN은 MS를 인증해야 할 때마다 받은 벡터를 하나씩 정해진 차례대로 사용한다. MS는 공유한 카운터를 이용하여 SN이 전달한 인증 벡터의 최근성을 확인한다. MS가 인증되면 SN은 MS와 CK_i, IK_i를 이용하여 인증 암호화를 통해 메시지를 교환한다.

UMTS AKA 프로토콜은 그림 10.3과 같다. 여기서 C_{HM}은 홈네트워크 AuC에 유지되고 있는 카운터 값이고, C_{MH}는 단말에 유지되고 있는 대응되는 카운터 값이다. VLR은 단말과 AuC 간에 공유하고 있는 K_{MH}를 모르며, 이것을 모르는 상태에서 단말을 인증해야 하기 때문에 AuC가 단말의 응답을 확인할 때 사용하는 값인 XRES_i까지 AuC가 만들어 준다. 이 방식에서 벡터 전달을 위해 네트워크 대여폭이 많이 소모되며, SN에 비교적 많은 저장 공간이 요구된다. 하지만 SN는 모바일 단일이 아니라 서버이므로 이 공간 요구가 부담되는 것은 아니다.

1.3 티켓 방식과 다중 벡터 방식의 비교

티켓 방식과 다중 벡터 방식은 표 10.1에 제시된 것처럼 장단점이 있다. 또 실제 현재는 하나의 비밀키를 확립한 후에 KDF를 이용하여 이 키로부터 여러 개의 독립적인 키를 생성하여 사용 가능하기 때문에 이와 같은 방법을 사용하지 않고도 비슷한 효과를 얻을 수 있는 방법이 있다,

2. 키 확립 프로토콜 사례

2.1 TLS 프로토콜

TLS(Transport Layer Security) 프로토콜[4]은 현재 인터넷에서 가장 널리 사용하고 있는 프로토콜이다. 이 프로토콜은 웹 브라우저로 한때 유명했던 Netscape 사에서 개발한 SSL(Secure Socket Layer) 프로토콜을 인터넷 표준으로 만든 프로토콜로서, 웹 브라우저와 웹 서버 간의 통신을 보호해주기 위해 개발된 프로토콜이다. 하지만 웹 통신을 보호하는 것에 제한되지 않고, TCP를 사용하는 대부분의 프로토콜의 메시지를 보호하기 위해 사용할 수 있다. 이 프로토콜은 인증서 기반 공개키를 활용하는 프로토콜이며, 인증서를 통해 상호 인증이 가능하지만 보통 브라우저가 웹 서버만 인증하는 형태로 많이 사용한다. 이것은 인터넷의 특성으로 웹 서버는 브라우저를 인증할 필요가 없지만, 브라우저는 특정 주소를 서비스하는 웹 서버가 맞는지 인증할 필요가 있다.

TLS는 악수 프로토콜(handshake protocol)과 레코드 프로토콜(record protocol)로 구성된다. 악수 프로토콜을 통해 클라이언트는 서버를 인증하고 서버와 키 확립을 하게 되며, 레코드 프로토콜은 확립된 세션키를 이용하여 데이터를 서로 암호화하여 교환할 수 있도록 해준다.

TLS는 고정된 암호알고리즘을 사용하지 않고, 클라이언트와 서버가 사용할 알고리즘을 협상할 수 있도록 해준다. TLS 1.2까지 가장 많이 사용한 것은 RSA 기반 TLS이다[5]. 이 방식에서 클라이언트는 랜덤하게 PMS(Pre-Master Secret)를 선택하여 서버의 공개키로 암호화하여 전달하면 이것을 근거로 둘 다 동일한 MS(Master Secret)를 생성한다. 이 방식에서 인증은 같은 MS를 생성할 수 있는지 확인함으로써 이루어진다. 이 과정에서 전자서명을 활용할 수 있다.

악수 프로토콜을 통해 총 4개의 암호키가 확립된다. 브라우저와 서버가 각각 2개씩 사용하며, 하나는 암호화용이고, 다른 하나는 MAC 용이다. 브라우저와 서버가 다른 키를 사용함으로써 역방향 재전송 공격은 원천적으로 가능하지 않다. TLS 1.2에서는 mac-then-encrypt 방식의 인증 암호화는 사용하였지만 가장 최신 버전인 TLS 1.3[4]부터는 더 안전한 encrypt-then-mac 형태의 인증 암호화의 사용을 의무화하였다.

TLS 1.3 버전에서는 기존에 가장 많이 사용하던 RSA 기반 TLS의 사용을 중단하고, DH(Diffie-Hellman) 기반의 3가지 버전만 사용한다. 물론 이 과정에서 RSA 기반 전자서명을 사용할 수 있지만 PMS는 항상 DH 키 동의 방식을 통해 확립한다.

6장에서 살펴본 바와 같이 기본 DH 키 동의 프로토콜은 중간자 공격에 취약하다. TLS는 보통 상호 인증하지 않고 서버만 클라이언트에 인증하는 형태를 많이 사용하기 때문에 양방향 서명을 통해 중간자 공격을 방어할 수 없다. 이에 6장에서 소개한 SIGMA 프로토콜을 약간 변형하여 다음과 같이 사용한다.

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

여기서 K는 g^{ab}를 통해 계산된 키이다.

공격자가 중간자 공격을 시도하더라도 공격자는 서버의 서명을 위조할 수 없으면 g^b을 다른 것으로 교체할 수 없다. 즉, 중간자는 중간에서 클라이언트가 서버로 전달하는 것은 임의로 수정할 수 있지만, 서버가 전달하는 것은 임의로 수정하여 클라이언트에게 중계할 수 없다. 이 때문에 중간자는 클라이언트가 인식 못 하도록 중간자 공격을 할 수 없다.

TLS는 매번 복합한 프로토콜을 온전히 실행하지 않고 클라이언트가 이전에 서버에 접속한 적이 있으면 메시지 라운드 수가 줄어든 간결한 버전의 프로토콜을 수행할 수 있도록 해준다. 이것을 세션 재개(session resumption)라 한다. 기본적으로 세션 재개는 이전에 확립된 세션키를 활용하게 된다. TLS 1.2에서 세션 재개는 세션 식별자 또는

세션 티켓을 사용하였지만 TLS 1.3에서는 PSK(Pre-Shared Key)를 사용하는 방식으로 바뀌었다.

세션 식별자 방식에서 클라이언트와 서버는 모두 이전 세션 정보를 유지해야 한다. 이것은 수 많은 클라이언트를 처리하는 서버에게는 부담이 될 수 있다. 세션 티켓을 사용할 경우 서버는 세션을 종료하기 전에 세션 재개에 필요한 티켓을 만들어 클라이언트에 전달하면 클라이언트는 이것을 유지하다 재개하고 싶을 때 서버에 전달한다. 세션 티켓은 세션 ID 방식을 비상태 기반으로 바꾼 방식이다. PSK는 세션 티켓을 간소화한 방식으로 생각할 수 있다. PSK는 온전한 TLS 악수 프로토콜을 수행하는 과정에서 티켓 형태로 서버로부터 받게 된다.

2.2 무선랜 보안

무선 통신은 공기를 매개로 데이터를 전송하기 때문에 유선보다 도청하는 것이 상대적으로 쉽다. 이에 접속 권한이 있는 장치만 사용하도록 하고, 이들 장치와 무선 AP 간에 교환하는 데이터를 보호하기 위해 사용하는 프로토콜이 WEP(Wired Equivalent Privacy)과 WPA(WiFi Protected Access) 프로토콜이다. WEP은 1997년에 표준으로 제정된 프로토콜이지만 심각한 결함이 발견되어 2003년도에 WPA가 임시 해결책으로 도입되었으며, 2004년도에 WPA2[6]로 전면 개편되었다. 2018년도에 WPA2를 개선한 WPA3이 발표되었다.

무선 AP와 무선 장치 간의 인증은 크게 두 종류로 구분한다. 하나는 무선 AP가 자체적으로 무선 장치와 상호 인증하는 방법이고, 다른 하나는 네트워크에 별도 인증 서버를 두어 인증 서버가 무선 장치를 인증하는 방법이다. 전자는 주로 집이나 카페와 같은 곳에서 사용하며, 모든 장치가 패스워드에 기반한 동일한 키를 사용한다. 이 때문에 이 방식과 후자를 구분하기 위해 PSK(Pre-Shared Key)가 WPA 뒤에 붙는다. 후자는 대학, 기업, 공공 WIFI 등에서 많이 사용하며, 이 방식을 사용하면 각 기기는 다른 키를 사용한다. 후자에서 사용하는 인증 방법이 EAP(Extensible Authentication Protocol)이다. 이 절에서 전자에 대해서만 설명한다.

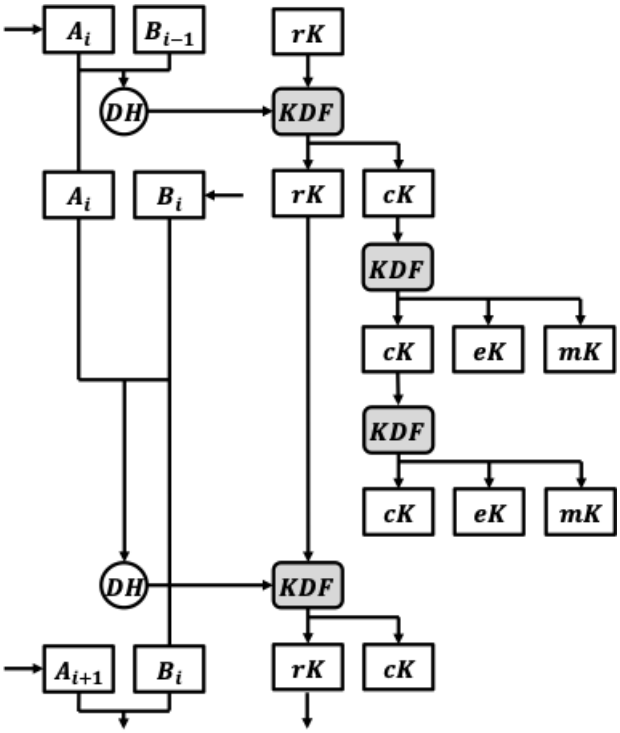
무선 AP와 무선 장치 간 보안은 다양한 장치를 지원하여야 하며, 속도와 성능 측면에서 우수하여야 한다. 이에 참여자 간 알고리즘 협상은 무겁기 때문에 알고리즘을 고정하고 성능 때문에 대칭 암호알고리즘을 사용한다. 기본적으로 무선 AP에 설정한 패스워드로부터 무선 AP와 모든 접속 장치가 같은 대칭키를 생성하여 세션키를 확립하여 메시지를 암호화하여 교환한다. WEP는 RC4 스트림 암호방식을 사용하며, 매번 다른 키를 사용하기 위해 24비트 카운터 IV를 40bit 키와 결합하여 사용한다. 또 간단한 CRC checksum을 이용하여 무결성을 제공한다. WEP는 여러 가지 문제점이 있지만 가장 심각한 문제점은 IV를 사용하는 방식이다. 스트림 방식에서는 절대 같은 키 스트림을 두 번 이상 사용하면 안 된다. 하지만 IV가 비교적 짧기 때문에 충분히 순환될 수 있으며, 심지어 AP를 초기화하면 다시 0이 되었다.

WPA는 임시 해결책으로 도입된 것이기 때문에 여전히 RC4를 사용하였지만 CRC checksum 대신 MAC을 사용하는 등 몇 가지 개선 사항을 도입하여 WEP의 많은 문제점을 해소하였다. WPA2는 RC4 대신 AES를 사용하며, 인증 암호화까지 지원한다. WPA2 프로토콜을 요약하여 기술하면 다음과 같다.

$$
\begin{array}{lll}
\text{Msg 1.} & C \to AP: & N_C \\
\text{Msg 2.} & AP \to C: & N_A, \text{MAC}.K_2(N_A) \\
\text{Msg 3.} & C \to AP: & \text{MAC}.K_2(\text{“ready to start”})
\end{array}
$$

AP는 메시지 1을 수신하면 자신이 전달할 난스 N_A를 생성한 후에 저장된 패스워드 pwd와 두 개의 난스를 이용하여 4개의 키를 생성한다. 그 중 하나가 위에 사용한 K_2이고 K_3과 K_4는 인증 이후 메시지를 교환할 때 인증 암호화에 사용하는 암호화 키와 MAC 키이다. 첫 번째 키는 그룹키로 특수한 용도로 활용한다.

WPA3는 9장에서 설명하는 패스워드 기반 프로토콜을 사용하여 안전하지 못한 패스워드를 설정하여 사용하더라도 키가 노출되지 않도록 하였다. 위에 제시된 WPA2 프로토콜에서 제3자는 메시지 1과 메시지 2를 확보하면 패스워드를 추측한 다음에 추측한 결과가 맞는지 메시지 2에 포함된 MAC 값을 이용하여 확인할 수 있다. WPA3는 SAE(Simultaneous Authentication of Equals)[7] 프로토콜을 사용하여 이 문제를 해결하였다. 하지만 이에 대한



<그림 10.4> 시그널 프로토콜의 이중 톱니바퀴

설명은 11장으로 미룬다.

2.3 시그널 프로토콜

시그널 프로토콜은 위스퍼시스템이 메신저 보안을 위해 개발한 프로토콜로서 현재는 시그널이라는 메신저 외에 왓츠앱, 페이스북 메신저, 구글의 allo 등 여러 메신저에서 사용하고 있다. 메신저 보안은 다른 응용과 달리 상대방이 현재 온라인 상태가 아닐 수 있으며, 한 번 구축된 대화가 불연속적으로 기간의 제한 없이 이어질 수 있다. 따라서 2자간 온라인 상태라는 가정하에 수행하는 기존 키 확립 프로토콜을 그대로 메신저 환경에 적용할 수 없으며, 불연속적으로 지속하는 대화를 유지하기 위해 키 관리가 단순할수록 바람직하다. 더욱이 사용자는 국기 기관의 감청 등으로부터도 사생활을 보호받고 싶으며, 자신의 대화 내용이 서버를 포함하여 노출되지 않기를 원한다. 또 하나 차이점은 사용자는 하나의 장치만 이용하는 것이 아니라 여러 장치를 이용하여 서비스를 사용할 수 있다. 보통 다중 장치 문제는 각 장치를 사용하는 사용자를 다른 사용자인 것처럼 처리하는 형태로 해결한다.

이 교재에서 시그널 프로토콜의 자세한 사항을 모두 설명하는 것은 교재 범위를 벗어나기 때문에 핵심적인 부분만 설명하고자 한다. 첫째, 상대방이 오프라인일 수 있기 때문에 이를 극복하기 위해 각 사용자는 DH 프로토콜 수행에 필요한 값들을 사전에 서버에 여러 개 등록하도록 하고 있다. 따라서 A가 오프라인인 B와 키를 확립하고 싶으면 서버로부터 등록된 g^b를 받는 방식을 사용하고 있다. 따라서 각 사용자는 서버에 등록한 각 g^a에 대해 (g^a, a) 쌍을 여러 개 유지하고 있어야 한다.

둘째, 한 번 키를 확립한 다음에는 메시지를 주고받을 때마다 DH 키 확립을 위한 다음 값을 상대방으로부터 받는 방식을 사용하고 있다. 예를 들어 g^{a_0}가 A가 선택한 값이고, g^b가 서버로부터 받은 값이면 $g^{a_0 b}$가 최초 공유 비밀이 되며, B가 A의 메시지에 대한 응답을 하게 되면 해당 메시지에 g^{b_0}가 포함되어 두 번째 공유 비밀은 $g^{a_0 b_0}$가 되며, 그다음은 $g^{a_1 b_0}$가 된다. 이처럼 톱니바퀴가 돌아가듯이 DH 키 확립이 이루어지기 때문에 이 과정을 “ratchet”이라 한다.

셋째, 메시지마다 다른 비밀키를 사용하기 위해 KDF를 이용하여 현재 세션키로부터 다음 세션키를 계산한다. 예를 들어 A가 B에게 메시지 2개를 연속으로 보내고 그것에 대한 회신을 받았다고 하자. 그러면 A가 보낸 2개의 메시지는 모두 다른 키로 암호화하여 전달한다. 이 과정은 그림 10.4에 기술되어 있으며, 이를 간단하게 설명하면 KDF은 두 개의 키를 출력하게 되는데 하나는 루트키(rK)이고 다른 하나는 체인키(nk)이다. 이 nK는 또 다른 KDF에 입력되어 다음 nk와 메시지 인증 암호화에 필요한 cK, iK를 생성하게 된다. 이처럼 메시지 암호화키도 매번 바뀌기 때문에 두 개의 톱니바퀴가 돌아간다고 하여 **이중 톱니바퀴**(double ratchet)라 한다[8].

넷째, 키 관리 측면에서 살펴보면 각 사용자는 상대방의 현재 DH 값과 현재 루트 키와 체인키를 유지하고 있어야 하며, 비동기화 방식으로 메시지들이 교환되기 때문에 생성된 복호화키 중 아직 사용하지 않은 키들은 유지해야 한다. 그 외에 키들은 사용한 즉시 삭제할 수 있다. 그림 10.4에서 $cK_{s,2}$를 사용할 차례라고 하자. 그러면 A는 a_i, rK_s, $nK_{s,2}$만 유지하고 있다. 실제는 이 값 외에 장기간 키와 서버에 등록해 놓은 DH 값들을 유지해야 한다.

공격자 입장에서는 특정 메시지의 복호화키를 확보하더라도 다음 키나 이전 키를 생성할 수 없으며, nK를 확보하면 다음 키를 얻을 수 있지만, 사용자가 해당 키를 사용할지 여부는 알 수 없으며 상대방이 메시지를 보낸 순간에 다음 DH 값이 계산되므로 쓸모없는 값이 된다. a_i와 rK_s를 둘 다 얻으면 다음 키들을 얻을 수 있지만, 이 역시 A가 새 메시지를 보내는 순간 쓸모없는 값이 된다.

3 접촉 추적

코로나19 팬데믹 초기에는 코로나19에 걸린 사람을 빨리 찾아 격리 치료하고, 그 사람과 접촉이 있던 사람들을 자가격리하여 확산을 방지하는 것이 매우 중요하였다. 우리나라는 감염자와 접촉이 있던 사람을 찾기 위해 정보기술을 이용한 자동 추적 방법을 사용하지 않고 역학조사자에 의한 확진자 인터뷰를 주로 사용하였다. 이 과정에서 인터뷰의 진실을 파악하기 위해 신용카드 사용 내역, 휴대전화 위치 정보 등을 활용하였다.

코르나19 팬데믹에 대처하기 애플과 구글은 특정 사람과 접촉이 있는 사람을 효과적으로 찾아주는 블루투스를 이용한 접촉 추적 방식을 개발하였다. 팬데믹에 대처하기 위한 기술이기 때문에 사회 안녕에 더 중요한 기능을 할 수 있다면 기존 다른 서비스와 달리 개인 프라이버시 침해를 일정 수준은 양해할 수 있다.

애플과 구글이 개발한 자동 접촉 추적 기술에서 각 사용자는 휴대한 모바일 기기에서 주기적(예: 5분마다)으로 랜덤값을 블루투스를 이용하여 주변에 방송한다. 각 사용자의 모바일 기기는 다른 사용자가 전송한 랜덤값을 기록한다. 보통 반경 2m 이내에 있는 다른 사용자의 값을 자신의 기기에 기록하는 형태가 된다. 기록된 랜덤값은 14일 동안 기기에 보관한다. 감염된 사용자는 자신이 보냈던 랜덤값을 공개하게 되며, 다른 사용자들은 자신에 기기에 보관된 값을 이용하여 자신이 해당 감염자와 밀접 접촉하였는지 확인을 할 수 있다. 실제 밀접 접촉자를 판별하는 방법은 중앙집중 방식과 탈중앙 방식에 따라 차이가 있다.

각 사용자는 랜덤값을 주기적으로 방송하기 위해 다음과 같이 2개의 키를 사용한다.

- 추적키(256bit) K_A: 기기에 유지하며, 절대 밖으로 노출하지 않는다.
- 일별키 $K_{20220510} = F(K_A, 20220510)$: 추적키를 이용하여 매일 일별키를 생성한다.

실제 사용자는 일별키와 방송한 시각 정보를 이용하여 랜덤값 $R = G(K_20220510, 09:10)$을 계산하여 방송하게 된다.

중앙집중 방식은 중앙 서버가 감염자와 밀접 접촉한 사용자를 판별하고 자가 격리 대상에게 통보하는 방식이다. 반면에 탈중앙 방식은 각 사용자가 직접 자신이 감염자와 밀접 접촉하였는지 판별하게 된다. 중앙집중 방식에서 각 사용자는 일별키를 중앙 서버에 항상 알려주어야 한다. 감염자를 파악하게 되면 감염자의 모바일 기기로부터

감염자가 수신한 랜덤값과 각 사용자의 일별키를 이용하여 밀접 접촉자를 찾게 된다. 탈중앙 방식에서는 중앙 서버는 감염자의 랜덤값만 공개한다. 랜덤값을 직접 공개하면 공개해야 하는 값이 너무 많으므로 실제 공개하는 것은 감염자의 일별키이다. 밀접 접촉한 것으로 의심되는 사용자는 공개된 감염자의 일별키와 자신의 모바일 기기에 보관중인 수신한 랜덤값을 이용하여 자신이 밀접 접촉자인지 스스로 확인한다.

두 방식을 비교하면 탈중앙 방식이 사용자 프라이버시 보호 측면에서 더 좋은 방법이다. 반면에 중앙집중 방식은 밀접 접촉자를 더 정확하게 많이 파악할 수 있다. 일별키를 더 세분화하여 시간별키를 사용하면 사용자의 프라이버시를 더 효과적으로 보호할 수 있다. 예를 들어 프라이버시에 민감한 시간대에 사용한 시간별키는 공개하지 않아도 되도록 서비스를 운영할 수 있다.

이와 같은 자동 접촉 추적 서비스는 자동으로 밀접 접촉자를 찾아낼 수 있지만 몇 가지 다음과 같은 단점도 있다.

- 모든 사용자가 자발적으로 이 서비스를 사용해야 하며, 탈중앙 방식에서는 각 사용자가 능동적으로 서비스에 참여해야 하는데, 이것의 보장이 어려울 수 있다.
- 허위양성이 많을 수 있다. 랜덤값을 수신하였다고 무조건 밀접 접촉자가 되는 것은 아니다.
- 사용자 모바일 기기의 배터리가 많이 소모될 수 있다.
- 사용자 프라이버시를 침해하기 위한 공격 수단으로 활용할 수 있다. 예를 들어 공격자는 돌아나면서 랜덤값과 랜덤값을 보낸 자와 현재 위치를 연결할 수 있는 정보를 수집하여 사용자들의 위치 정보를 수집할 수 있다.

참고문헌

[1] C. Neuman, S. Hartman, K. Raeburn, “The Kerberos Network Authentication Service (V5),” IETF RFC 4120, 2005.

[2] A. Kehne, J. Schonwalder, H. Langendorfer, “A Nonce-Based Protocol for Multiple Authentications,” ACM Operating Systems Review, Vol. 26, No. 4, pp. 84–89, Oct. 1992.

[3] Universal Mobile Telecommunication Systems (UTMS), 3G security, Security Architecture, 3GPP TS 33.102 version 7.0.0, 2005.

[4] E. Rescorla, “The Transport Layer Security (TLS) Protocol: Version 1.3,” IETF RFC 5246, Aug. 2018.

[5] T. Dierks, E. Rescorla, “The Transport Layer Security (TLS) Protocol: Version 1.2,” IETF RFC 5246, Aug. 2008.

[6] IEEE 802.11i-2004: Ammendment 6: Medium Access Control (MAC) Security Enhancements, IEEE Standards, 2004.

[7] Dan Harkins, “Simultaneous Authentication of Equals: A Secure, Password-Based Key Exchange for Mesh Networks,” Proc. of the 2nd Int’l Conf. on Sensor Technologies and Applications, pp. 839–844. Sept. 2008.

[8] Trevoe Perrin, Moxie Marlinspike, “The Double Ratchet Algorithm,” Open Whisper Systems Speiocifications, http://whispersystems.org/docs/specifications/doubleratchet/, 2016.

퀴즈

1. 메신저를 사용할 경우 대화의 프라이버시를 보장하기 위해 종단간 암호화를 할 수 있다. 메신저는 다른 일반적인 통신 프로토콜과 성격이 다른 점이 있다. 이와 관련된 다음 설명 중 틀린 것은?

   ① 대화하고자 하는 상대방이 오프라인 상태일 수 있다.

   ② 사용자는 보통 하나의 기기만 사용하여 서비스를 이용한다.

③ 통신은 항상 메신저 서버를 경유하는 인라인 방식이다.
④ 대화가 불연속적으로 기간의 제한없이 계속 진행될 수 있다.

2. TLS는 웹 브라우저와 웹 서버 간에 교환하는 메시지를 암호화하여 통신하기 위해 사용하는 HTTPS가 내부적으로 사용하는 키 확립 프로토콜이다. TLS와 관련된 다음 설명 중 틀린 것은?

① 웹 서버의 공개키 인증서를 통해 웹 브라우저는 자신이 기대한 사이트에 올바르게 접속한 것인지 확인하게 된다.
② 실제적 프로토콜을 수행하기 전에 사용할 암호알고리즘을 협상할 수 있도록 해준다.
③ 최신 버전인 1.3에서는 encrypt-then-mac 방식의 인증 암호화 사용을 의무화하였고, 전방향 안전성이 보장되도록 RSA 버전의 키 확립 프로토콜은 사용하지 못하도록 하였다.
④ 보통 웹 브라우저와 웹 서버가 상호 인증을 한다.

3. UMTS AKA는 다중 벡터 방식을 사용하는 프로토콜이다. 다중 벡터에서는 서비스 제공 서버 또는 클라이언트가 여러 개 벡터를 발급받은 후 한번에 하나의 벡터만 사용한다. UMTS AKA와 관련된 다음 설명 중 틀린 것은?

① 각 벡터를 한번만 사용하도록 하기 위해 타임스템프를 활용하고 있다.
② UMTS AKA에서는 클라이언트인 모바일 단말 대신에 서비스를 제공하는 외부 네트워크 서버가 벡터를 유지한다.
③ 외부 네트워크는 최초 모바일 단일이 보내는 IMSI를 통해 그것의 홈 네트워크를 식별한다.
④ 모바일 단말과 그것의 홈네트워크는 대칭키를 공유하고 있으며, 이 대칭키를 통해 상호 인증한다.

4. 매번 키 확립 프로토콜을 수행하는 것이 부담될 경우 티켓 방식, 다중 벡터 방식을 이용할 수 있다. 두 기법에 대한 다음 비교 설명 중 틀린 것은?

① 티켓 방식은 하나의 티켓을 일정 기간 동안 계속 사용하는 반면에 다중 벡터 방식은 발급받은 각 벡터를 한 번만 사용한다.
② 티켓은 하나의 티켓만 유지하면 되지만 다중 벡터 방식은 여러 개의 벡터를 유지하여야 한다.
③ 티켓 방식과 다중 벡터 방식은 모두 하나의 키를 여러 번 사용하게 된다.
④ 티켓 방식과 다중 벡터 방식은 키 확립 프로토콜을 진행한 이후 일정 기간 동안 또는 일정 회수 동안 인증 서버와 통신하지 않고 서비스를 사용할 수 있다.

연습문제

1. 티켓 기반 프로토콜의 경우에는 티켓을 클라이언트가 유지하지만 반대로 다중벡터 방식에서는 서비스를 제공하는 응용 서버가 다중벡터들을 유지한다. 다음 각각에 대해 설명하시오.

① 다중벡터 방식에서는 다중벡터들을 서비스를 제공하는 응용 서버가 유지하는 이유를 설명하시오.
② 티켓 방식에서 티켓은 클라이언트에 유지하지만 티켓은 서비스를 제공하는 서버와 티켓발급서버 간에 공유된 키로 암호화되어 있다. 그러면 다중벡터 방식에서는 다중 벡터들은 어떤 키로 암호화되어 있어야 하는지 논하시오.
③ 티켓 방식에서는 시간 개념을 이용하여 티켓 사용을 제한함. 이 경우 티켓을 발급하는 서버와 서비스를 제공하는 응용서버 간에 클럭 동기화가 필요하다. 하지만 유효기간을 사용하는 티켓 방식의 경우에도 참여자 간에 클럭 동기화가 필요하지 않도록 구성할 수 있음. 이 구성에 대해 간단히 설명하시오.
④ 다중 벡터 방식에서는 시간 개념을 사용하지 않음. 그 이유는 한 세션키를 기존과 같이 한번만 사용하기 때문이다. 그러면 다중 벡터 방식에서는 어떻게 한 세션키를 한번만 사용하도록 제한하는지 간단히 설명하시오.

2. TLS에서 Diffie-Hellman을 사용할 경우 서버의 값만 인증하고, 클라이언트의 값은 인증하지 않는다. 하지만 서명 외에 추가로 확립된 세션키를 이용하여 계산되는 MAC 값을 서로 교환한다. 아래와 같이 한 쪽만 인증하는 형태로 프로토콜을 수행할 경우 중간자 공격 관련 어떤 문제가 있는지 논하시오.

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

힌트. 중간자가 g^a를 g^c로 바꾸면 어떻게 진행되는지, 또 g^b를 g^c로 바꾸면 어떻게 진행되는지 생각해 보세요.

3. 메신저 서비스가 다른 서비스와 구별되는 키 확립 프로토콜에서 고려되어야 하는 특징을 설명하시오.