인터넷에서 사용되는 여러 기술 : HTTPS 6

조인준
KBS 미디어기술연구소 차장

지난 편에서는 암호화 통신 시 사용할 대칭키 생성을 위한 키 교환(Key Exchange) 단계에서 사용하는 방식 중 가장 널리 쓰이는 방식의 하나인 RSA 방식에 대해 설명했습니다. 이번 편에서는 DH(Diffie-Hellman) 키 교환 방식에 대해 설명하겠습니다. DH는 이 방식의 고안자들의 이름인 Diffie와 Hellman의 첫 글자를 따서 만든 약자입니다.

DH 키 교환 방식은 [그림 1]과 같이 누군가 정보를 훔쳐볼 수 있는 네트워크에서 A와 B가 암호화 통신을 하기 위해 필요한 비밀키를 생성하고자 할 때 이용하는 방식입니다.

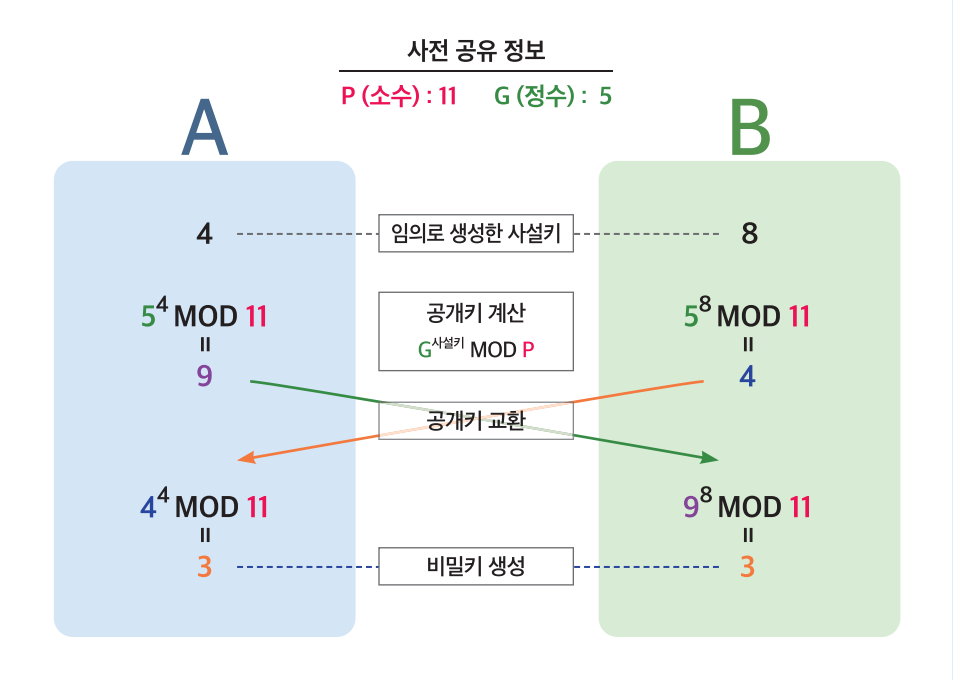

그림 1. DH(Diffie-Hellman) 키 교환 방식 프로세스

암호화 통신을 시작하기 위해 A는 B에게 소수 P([그림 1]의 11)와 1 ~ P-1 사이의 적당한 정수 G([그림 1]의 5)를 공개합니다. “암호화를 한다면서 암호화에 필요한 숫자를 보안이 적용되지 않은 상태로 네트워크에 공개한다고?”라는 질문이 제일 먼저 떠오르겠지만, P와 G는 공개되어도 이후의 암호화된 정보를 누군가 해독하려 할 때 약점이 되지 않습니다. A와 B가 P와 G를 공유하면 A와 B는 임의로 사설키에 해당하는 번호를 각각 생성합니다. A가 생성한 번호(A의 사설키)는 B가 알 수 없으며, B가 생성한 번호(B의 사설키)도 A가 알 수 없습니다. 사설키가 생성된 이후에 A는 $G^{사설키}\ MOD\ P$ 식(G와 P는 [그림 1]에 표시된 문자)을 이용하여 자신의 공개키 $9(5^4\ MOD\ 11 = 9)$를 생성하고 B도 같은 방식으로 자신의 공개키 $4(5^8\ MOD\ 11 = 4)$를 생성합니다. 이렇게 각자의 공개키를 생성한 이후에 서로의 공개키를 교환합니다. 비밀키 생성은 $상대의공개키^{자신의사설키}\ MOD\ P$ 식을 이용하여 계산됩니다. A의 경우 $4^4\ MOD\ 11 = 3$을 통해 비밀키 3이 생성되었고, B의 경우도 $9^8\ MOD\ 11 = 3$을 통해 비밀키 3이 생성되었습니다. A와 B 이외의 제삼자는 A와 B의 사설키 4와 8을 알 수 없습니다. 그렇기에 이 사설키를 기반으로 생성된 비밀키 또한 알 수 없습니다. 여기서 소수 P가 충분히 클 경우, 외부에서 비밀키를 알아내기 위해 정보를 훔쳐보는 이가 있다 해도 [식 1]과 같은 형태의 $G^{사설키}\ MOD\ P$ 식을 이용해 A와 B의 사설키 값을 알아내는 것이 매우 어렵다고 알려져 있습니다. 매우 큰 P 값을 썼을 때 제삼자가 알아내기 힘들고 A와 B만 아는 비밀키를 생성할 수 있으므로 이를 대칭키로 이용한 암호화를 통해 이후 통신을 안전하게 암호화할 수 있습니다.

$$G^X\ MOD\ P = N$$

식 1

다시 강조하지만, DH 키교환 방식의 보안성은 [식 1]에서 P가 매우 클 경우 G, P, N을 알고 있다고 해도 X를 찾아내는 것이 현실적으로 매우 힘들다는 것에 기반하고 있습니다. A와 B의 사설키를 알아내야 A와 B가 생성한 비밀키도 알아낼 수 있는데, 이 경우 유일한 해법은 경우의 수를 모두 대입해 보는 것뿐이라고 합니다. 하지만 이전의 RSA 방식에서와 같이 충분히 큰 소수 P를 사용한다면 G, P와 A와 B의 공개키를 알고 있다고 해도 엄청난 시간이 걸리는 계산 후에야 A와 B의 사설키 및 비밀키를 알아낼 수 있으니 이미 A와 B 사이에 통신이 모두 끝나고도 아주 긴 시간이 지난 후인 거겠죠?

DH(Diffie-Hellman) 키 교환(Key Exchange)은 암호화에 필요한 키를 교환하는 데 사용되는 프로토콜이지만, 직접적으로 메시지를 암호화하거나 복호화하는 방식은 아닙니다. DH 방식은 대칭키로 사용되는 비밀키를 생성할 뿐이고 이 키를 사용하여 메시지를 암호화하거나 복호화하는 방식은 별도로 존재합니다. 따라서 DH 키 교환 방식을 사용할 경우 이 키를 이용하여 메시지를 암호화하고 복호화하는 방식 또한 필연적으로 따라붙게 되어있습니다. DH 키 교환을 사용하여 양측이 공유한 비밀키로 메시지를 암호화하고 복호화하는 매우 간단한 예를 하나 들어보겠습니다. C군이 독자 여러분의 흥미를 위해 제공하는 본 예는 실제로 사용할 경우 누구나 해독이 가능한 매우 보안에 취약한 방식이므로 그냥 재미로만 보시길 부탁드립니다. 실제로 사용하는 암호화/복호화 알고리즘은 훨씬 복잡합니다. C군이 독자 여러분의 재미를 위해 고안한 방식은 DH 방식으로 생성된 키와 XOR 연산을 사용한 암호화/복호화 연산입니다.

XOR 연산은 ⊕ 기호로도 표시되며 [표 1]과 같은 연산을 갖습니다. [식 2]는 XOR의 연산 성질 중 하나이며 붉은색으로 표시한 특성을 이용하여 메시지의 각 문자([식 2] 붉은색 B)를 키값([식 2] 붉은색 A)과 XOR 하여 암호화하고, 다시 같은 키값과 XOR 하여 복호화할 수 있습니다.

A	B	XOR(⊕)
0	0	0
0	1	1
1	0	1
1	1	0

표 1. XOR 연산

$$A \oplus A = 0$$
$$A \oplus A \oplus B = B \oplus A \oplus A = A \oplus B \oplus A = B$$

식 2. XOR 연산 특성

문자	10진수	이진수
H	72	01001000
E	69	01000101
L	76	01001100
L	76	01001100
O	79	01001111

표 2. HELLO의 ASCII 코드

그럼 'HELLO'의 ASCII 값을 DH로 생성한 키값 '3'과 XOR 연산을 통해 암호화 및 복호화해보겠습니다. 우선 비밀키 '3'을 8비트 이진수로 변환하면 '00000011'입니다. 'HELLO'는 ASCII 값으로 변환하면 [표 2]와 같습니다.

메시지 'HELLO'의 각 문자를 키값과 XOR(⊕) 연산하여 암호화하면 다음과 같습니다. 이렇게 암호화된 메시지를 상대방에게 보내면 중간에서 누군가 메시지를 가로채도 바로 알아보기는 힘듭니다. 아무리 간단한 XOR 암호화라고 해도 조금 생각할 시간은 필요하겠죠?

H : 01001000 ⊕ 00000011 = 01001011
E : 01000101 ⊕ 00000011 = 01000110
L : 01001100 ⊕ 00000011 = 01001111
L : 01001100 ⊕ 00000011 = 01001111
O : 01001111 ⊕ 00000011 = 01001100

비밀키와 XOR(⊕) 연산을 통해 매우 취약하게(?) 암호화된 메시지를 받으면 이를 다시 키값과 XOR(⊕) 연산하여 다음과 같이 복호화할 수 있습니다.

01001011 ⊕ 00000011 = 01001000 (H)
01000110 ⊕ 00000011 = 01000101 (E)
01001111 ⊕ 00000011 = 01001100 (L)
01001111 ⊕ 00000011 = 01001100 (L)
01001100 ⊕ 00000011 = 01001111 (O)

위의 예를 통해 메시지 'HELLO'를 DH로 생성한 비밀키 '3'을 사용하여 XOR(⊕) 연산으로 간단히 암호화하고 복호화할 수 있음을 보았습니다.

구분	RSA	DH	DSA
암호화	☑	-	-
서명	☑	-	☑
키교환	☑	☑	-

표 3. RSA, DH, DSA 비교

[표 3]에 정리된 것과 같이 지난 편에서 소개드린 RSA 방식은 암호화(Encryption), 서명(Signature) 및 키교환(Key Exchange)까지 모두 가능한 방식이며, 현재 소개드린 DH 방식은 키교환을 위한 방식이고 암호화 방식은 SSL/TLS 핸드셰이크 단계에서 미리 설정되며, 설정된 암호화 방식을 위해 DH 방식으로 생성된 비밀키를 사용합니다. 그렇다면 RSA를 사용하지 않을 때 서명은 어떤 방식을 사용할까요? 이에 대한 해답이 DSA(Digital Signature Algorithm)입니다. SSL/TLS에서 DSA는 주로 서버 인증서에 대한 디지털 서명에 사용됩니다. 이 디지털 서명은 클라이언트가 서버의 신원을 확인할 수 있도록 합니다.

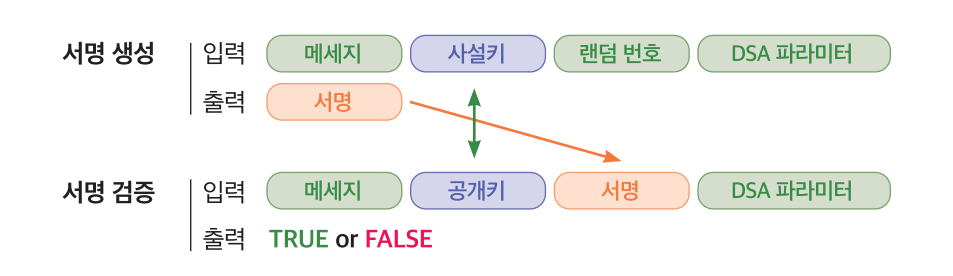

그림 2. DSA 수행 작업

DSA는 [그림 2]와 같은 '서명 생성' 및 '서명 인증'의 두 가지 작업을 수행합니다. 서명 생성을 위해서는 서명을 할 대상인 메시지, 이 메시지를 암호화하는데 사용할 사설키와 랜덤 번호 및 DSA 연산을 수행하기 위해 필요한 값들인 DSA 파라미터가 필요합니다. 서명이 생성된 후 이 서명의 검증을 위해서는 검증할 대상인 메시지, 그리고 서명을 만들 때 사용한 사설키에 대응되는 공개키, 그리고 서명 및 DSA 파라미터가 필요하며, 검증 연산의 결과로 해당 서명이 메시지를 보낸 이에 의해 된 것이 맞는지 여부(TRUE or FALSE)를 알 수 있습니다.

DSA의 서명 생성에서 특히 중요한 것이 '랜덤 번호'인데 이 '랜덤 번호'는 서명될 메시지마다 다른 값을 가져야만 합니다. 이유는 서로 다른 메시지에 같은 '랜덤 번호'가 사용되면 이를 기반으로 서명 생성에 사용된 '사설키'를 유추할 수 있는 단서를 제공하게 됩니다. 누군가 사설키를 알아낸다면 임의로 메시지를 조작하고 서명을 생성해서 마치 조작된 메시지가 진짜인 것처럼 위장할 수 있기 때문입니다. 이렇게 메시지마다 서로 다른 '랜덤 번호'를 생성하기 위한 방법으로는 매우 큰 '랜덤 번호'를 사용해서 중복된 번호 사용이 일어나기 거의 불가능하게 하거나 메시지를 기반으로 한 연산을 통해 번호를 생성하여 메시지가 다르면 다른 번호가 생성되도록 하는 방식 등을 사용할 수 있습니다.

지금까지 SSL/TLS에서 사용되는 키 교환 방식 및 서명 방식에 관한 기본적 내용을 알아봤습니다. 다음 편에서도 SSL/TLS에 대한 더 상세한 내용들을 다루어 보겠습니다.

P.S.

C군이 여러분께 전하는 내용 중 전문적 성격이 짙은 것은 엄밀한 언어를 사용하여 설명하기에는 한계가 있습니다.
본 내용은 설명하는 대상에 대한 전체적 맥락의 이해에만 이용하시고, 그 이상은 권위 있는 전문자료를 참고하시기 바랍니다.