인터넷에서 사용되는 여러 기술 : HTTPS 4

조인준
KBS 미디어기술연구소 차장

지난 편에서는 HTTPS(Hypertext Transfer Protocol Secure)를 위한 인증서에 관해 소개드렸습니다. 이번 편에서는 지금까지의 내용을 기반으로 HTTPS의 보안 기술인 SSL/TLS의 동작을 위한 서버 인증서 확인, 암호화 방식 결정, 암호화의 효율성을 높이는 대칭키 생성 등에 관한 핸드셰이킹 과정에 대해 알아보겠습니다.

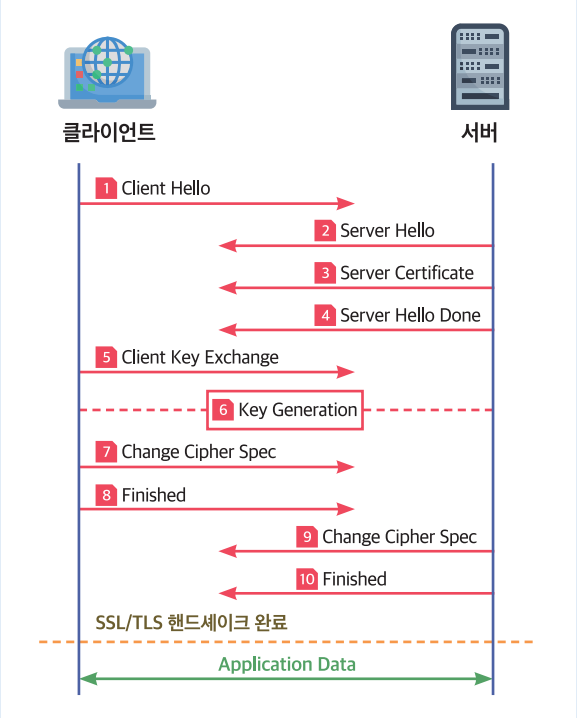

그림 1. 암호화된 데이터 통신을 수립하기 위한 SSL/TLS의 핸드셰이킹 과정

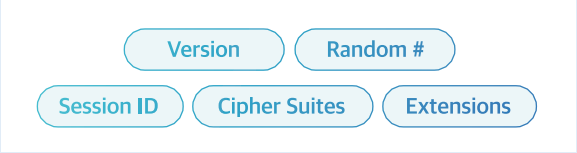

그림 2. Client Hello

[그림 1]은 암호화된 데이터 통신을 수립하기 위한 SSL/TLS의 핸드셰이킹 과정을 ① Client Hello부터 ⑩ Finished까지 단계별로 표시한 것입니다. 각 단계에서 수행되는 내용들은 아래와 같습니다.

1 Client Hello

SSL/TLS 핸드셰이킹 시작 시 클라이언트가 서버와 안전한 연결을 위해 보내는 첫 번째 메시지이며 [그림 2]의 항목들을 포함합니다.

- **Version**
  - 클라이언트가 지원하는 SSL/TSL의 최상위 버전
- **Random Number**
  - 클라이언트가 생성한 32바이트의 랜덤 번호로 처음 4바이트는 초 단위까지의 타임스탬프
- **Session ID**
  - 8바이트로 표시되며 초기 Client Hello에서는 0으로 채워짐
- **Cipher Suites**
  - 클라이언트가 지원하는 암호화 방식들의 리스트
- **Extensions**
  - 클라이언트의 요구사항 및 추가기능 지원 관련 정보 제공

2 Server Hello

SSL/TLS 핸드셰이크 프로세스에서 Client Hello에 대한 응답으로 서버가 보내는 메시지이며 [그림 2]의 Client Hello와 동일한 항목을 포함합니다.

• **Version**
  - 서버가 지원하는 SSL/TSL의 최상위 버전
• **Random Number**
  - 서버가 생성한 32바이트의 랜덤 번호로 처음 4바이트는 초 단위까지의 타임스탬프
• **Session ID**
  - 8바이트로 표시되며 서버가 임의로 생성한 세션의 ID가 전송됨
• **Cipher Suites**
  - 클라이언트가 보낸 암호화 방식 중 하나를 선택하여 클라이언트에 보냄
• **Extensions**
  - 클라이언트의 요구사항에 대한 서버의 추가기능 지원 관련 정보 제공

3 Server Certificate

서버가 클라이언트에게 인증서를 보내는 단계이며 일반적으로 지난 편에서 소개된 서버 인증서와 중간 인증서를 보냅니다. 루트 인증서는 추가로 보낼 수도 있지만, 보통 클라이언트가 이미 신뢰하는 루트 인증기관의 공개키를 알고 있기 때문에 필요에 따라 보내게 됩니다.

4 Server Hello Done

서버가 지금까지 필요한 메시지를 모두 보냈음을 나타내기 위해 클라이언트에게 보내는 확인 메시지이며 내용은 비어 있습니다.

5 Client Key Exchange

이 단계에서 클라이언트는 아래와 같은 Pre-master Secret이라는 것을 생성한 후 서버로부터 받은 인증서에 포함된 공용키로 암호화하여 보냄으로써 이후의 통신에서 사용될 대칭키를 생성하는 데 이용함과 동시에 서버가 인증서의 실제 소유주가 맞는지 확인할 수 있습니다. 서버가 인증서에 포함된 공용키에 대응되는 사설키를 소유하고 있다면 클라이언트가 암호화하여 보낸 Pre-master Secret을 정상적으로 복원하여 이후의 절차에 이상 없이 대응할 수 있기 때문입니다.

• **Pre-master Secret**
  - 모두 48바이트로 구성
  - 앞의 2바이트는 이전 단계에서 교섭된 SSL/TLS 버전
  - 나머지 46바이트는 무작위로 생성된 숫자

※ 현재의 예에서 설명한 Key Exchange 방식은 통상적인 설명에 이용되는 RSA 방식이며 Diffie-Hellman과 같은 다른 방식도 존재합니다. 이에 대해서는 추후 관련 내용을 소개하겠습니다.

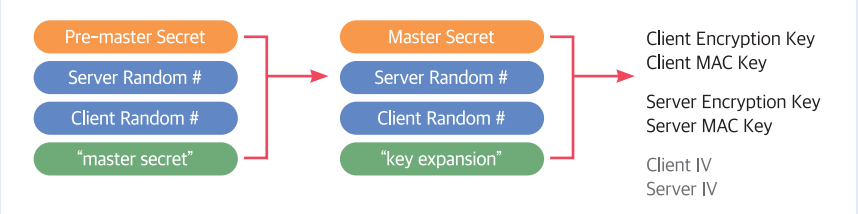

그림 3. Key Generation 프로세스

6 Key Generation

Pre-master Secret은 Client Key Exchange에서 사용한 알고리즘에 따라 다른 길이를 가지므로 [그림 3]과 같이 클라이언트와 서버 모두가 이전의 과

정을 통해 공유하고 있는 Pre-master Secret, Server Random Number, Client Random Number 및 'master secret'이라는 문자열을 이용하여 48비트의 Master Secret을 만들고 이를 다시 Server Random Number, Client Random Number 및 'key expansion'이라는 문자열과 함께 이용하여 이후의 통신에서 사용될 대칭키를 만듭니다. 대칭키는 클라이언트가 암호화를 위해 사용하는 Client Encryption Key와 클라이언트가 보낸 메시지의 손상 여부를 확인하기 위한 Client MAC Key 한 쌍, 서버가 암호화를 위해 사용하는 Server Encryption Key와 서버가 보낸 메시지의 손상 여부를 확인하기 위한 Server MAC Key 한 쌍, 총 2쌍이 만들어지고 서버와 클라이언트가 동일한 2쌍의 대칭키들을 각자 생성하여 소유하게 됩니다. 이로써 클라이언트의 암호화된 메시지는 서버가 소유한 클라이언트 대칭키 쌍을 통해 복원하고, 서버가 보낸 암호화된 메시지는 클라이언트가 소유한 서버 대칭키 쌍을 통해 복원합니다. 클라이언트와 서버가 각자가 보내는 메시지의 암호화에 다른 대칭키 쌍을 이용하는 이유는 서로가 같은 메시지를 보내더라도 다른 대칭키로 암호화되었기 때문에 겉보기에는 다른 메시지로 보이며, 행여나 누군가가 운 좋게 서버나 클라이언트의 암호화 키를 알아내더라도 다른 한쪽이 사용하는 암호화 키를 알 수 없으므로 보안이 100% 뚫리는 것을 방지할 수 있습니다.

7 Change Cypher Spec

클라이언트가 성공적으로 대칭키를 생성했으며 이후의 메시지는 지금까지의 과정을 통해서 정해진 암호화 방식으로 진행할 것임을 알립니다.

8 Finished

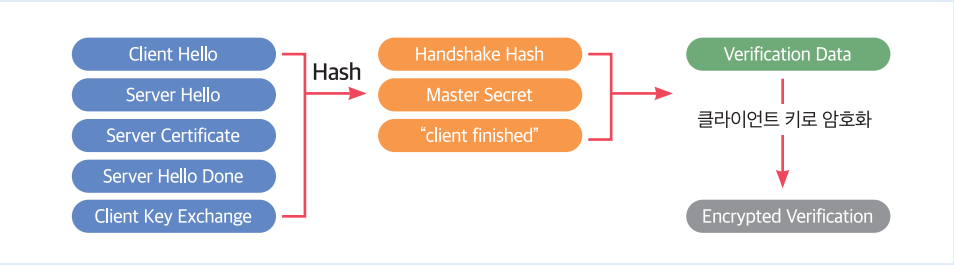

그림 4. Client의 Encrypted Verification 데이터 생성

지금까지의 과정을 통해서 암호화를 통한 통신에 관한 모든 준비가 완료된 것으로 보이지만 한 가지 확인되지 않은 것이 있습니다. 바로 클라이언트와 서버 각각이 생성한 대칭키 쌍들이 동일한지입니다.

우선 클라이언트와 서버가 동일한 클라이언트 키를 생성했는지 확인하기 위해 클라이언트 측에서 핸드셰이크 과정을 통해 양쪽이 주고받은 Client Hello, Server Hello, Server Certificate, Server Hello Done, Client Key Exchange 메시지를 모두 해시하여 Handshake Hash를 생성하고 이를 Master Secret과 'client finished'라는 문자열과 함께 이용하여 Verification Data를 생성합니다. 이 Verification Data를 클라이언트 키로 암호화한 Encrypted Verification을 서버로 보내면 서버는 자신이 소유한 클라이언트 키로 Verification Data를 복원합니다. 서버도 클라이언트와 동일한 Client Hello, Server Hello, Server Certificate, Server Hello Done, Client Key Exchange 메시지를 가지고 있으므로 자신이 가지고 있는 데이터를 이용하여 클라이언트와 동일한 방식으로 Verification Data를 생성할 수 있고 이를 수신 후 복원한 Verification Data와 비교할 수 있습니다.

만약 자신이 생성한 Verification Data와 수신한 Verification Data가 같다면 서버가 생성한 클라이언트 키는 클라이언트가 생성한 키와 동일하다는 확인이 됩니다. 동시에 Client Hello, Server Hello, Server Certificate, Server Hello Done, Client Key Exchange로 이어지는 핸드셰이크 과정에서 중

간에 누군가 데이터를 변조하지 않았다는 증명도 됩니다. 만약 누군가 중간에서 데이터를 변조했다면 클라이언트와 서버가 동일한 핸드셰이크 관련 메시지를 소유하고 있을 수 없고, 이로써 서로 다른 Verification Data 계산 결과를 갖기 때문입니다.

9 Change Cypher Spec

서버가 성공적으로 대칭키를 생성했으며 이후의 메시지는 지금까지의 과정을 통해서 정해진 암호화 방식으로 진행할 것임을 알립니다.

10 Finished

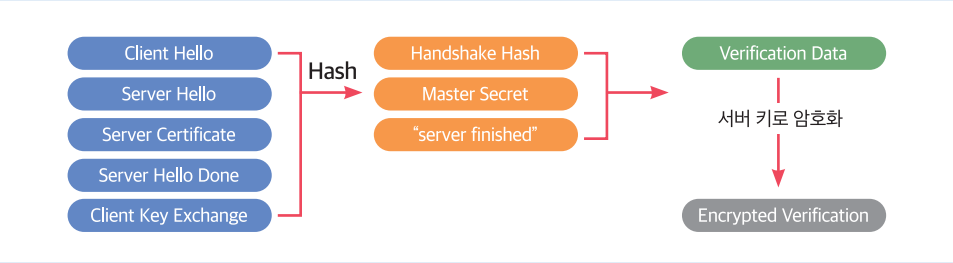

그림 5. Server의 Encrypted Verification 데이터 생성

서버도 클라이언트와 마찬가지로 각자가 생성한 대칭키 쌍들이 동일한지 확인합니다. 앞서 클라이언트 측에서 Encrypt Verification을 생성하여 보낸 것과 같이 서버도 같은 방법으로 Client Hello, Server Hello, Server Certificate, Server Hello Done, Client Key Exchange를 해시하여 Handshake Hash를 생성하고 이를 Master Secret과 'server finished'라는 문자열과 함께 이용하여 Verification Data를 생성합니다. 이 Verification Data를 서버 키로 암호화한 Encrypted Verification을 클라이언트로 보내면 클라이언트는 자신이 소유한 서버 키로 Verification Data를 복원합니다. 클라이언트도 서버와 동일한 Client Hello, Server Hello, Server Certificate, Server Hello Done, Client Key Exchange 메시지를 가지고 있으므로 자신이 가지고 있는 데이터를 이용하여 서버와 동일한 방식으로 Verification Data를 생성할 수 있고 이를 수신 후 복원한 Verification Data와 비교할 수 있습니다.

만약 자신이 생성한 Verification Data와 수신한 Verification Data가 같다면 클라이언트가 생성한 서버 키는 서버가 생성한 키와 동일하다는 확인이 됩니다. 동시에 앞선 내용과 같이 Client Hello, Server Hello, Server Certificate, Server Hello Done, Client Key Exchange로 이어지는 핸드셰이크 과정에서 중간에 누군가 데이터를 변조하지 않았다는 증명도 됩니다.

위의 과정을 거쳐 SSL/TLS 핸드셰이크가 완료되면 서버와 클라이언트 간 암호화된 통신이 시작됩니다. 다음 편에서는 Client Key Exchange에서 RSA 이외의 방식에 관한 내용을 소개하겠습니다.

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.