14.6 디지털 인증서

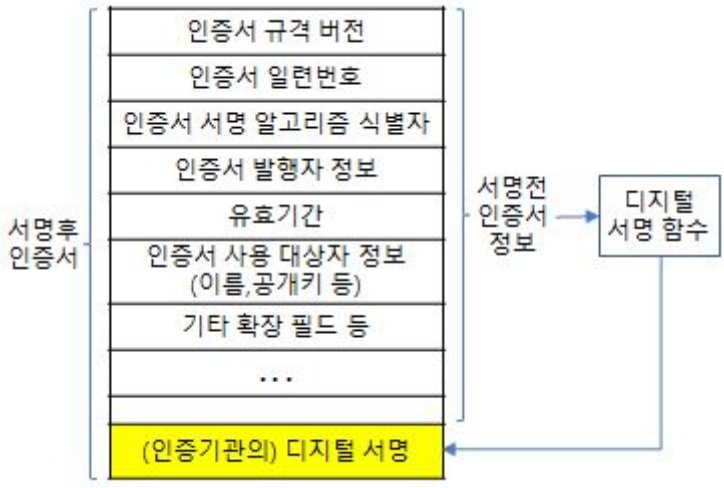
디지털 인증서: 신뢰할 수 있는 기관으로 부터 보증받은 사용자나 회사에 대한 정보를 담고 있음.

14.6.1 인증서의 내부

'인증 기관'에 의해 디지털 서명된 정보의 집합을 가지고 있음.

EX)

- 대상 이름(사람, 서버, 조직)
- 유효 기간
- 인증서 발급자
- 인증서 발급자의 디지털 서명



14.6.2 X.509 v3 인증서

디지털 인증서에 세계적 단일 표준은 없음.

대신 요즘 대부분 인증서는 정보를 X.509라는 표준화 서식에 저장함.

<표 8> X.509 v3 인증서

영역이름	설명
Version	인증서의 버전
Serial number	인증서의 일련번호
Signature algorithm id	인증서를 서명하는데 사용하는 알고리즘 식별자
Issuer name	인증서 발행자의 이름
Validity period	인증서의 유효기간(유효개시일과 종료일을 표시)
Subject name	인증서 소유자의 이름
Subject publickey info,	인증되는 공개키에 대한정보(공개키가 사용될 알고리즘의 식별자와 공개키를 포함)
Issuer unique identifier (선택)	발행자를 정확하게 표기하기 위해 사용되는 비트스트링
Subject unique identifier (선택)	소유자를 정확하게 표기하기 위해 사용되는 비트스트링
Extensions (선택)	표준단체나 사용자 조직에서 필요에 따라 정의할 수 있는 부분(확장영역의 이름과 Critical 여부와 실질적인 값을 포함)
Signature	위 영역에 대한 발행자의 서명값

X.509 기반 인증서 예시

- **웹 서버 인증서**
- 클라이언트 이메일 인증서
- 소프트웨어 코드사인 인증서

14.6.3 서버 인증을 위해 인증서 사용하기

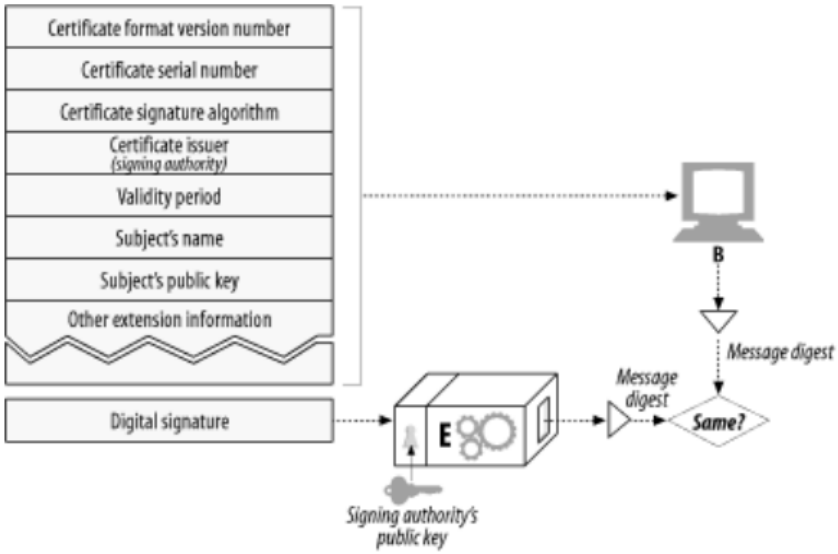
유저가 HTTPS를 통해 웹 트랜잭션 시작 시, 브라우저가 자동으로 접속한 서버의 디지털 인증서를 가져옴. 서버에 인증서가 없으면 보안 커넥션은 실패함.

서버인증서는 많은 필드를 가지는데 아래의 것들이 대표적이다

- 웹 사이트의 이름과 호스트명
- 웹 사이트의 공개키
- 서명 기관의 이름
- 서명 기관의 서명

브라우저는 인증서를 받으면 서명 기관을 검사한다. 서명 기관이 신뢰할만한 곳이라면 브라우저는 해당 기관의 공개키를 가지고 있을 것이다(**브라우저는 여러 서명 기관의 인증서가 설치되어있음**).

Figure 14-12. Verifying that a signature is real



브라우저가 해당 서명 기관을 모르는 경우, 사용자에게 신뢰하는지 물어봄.

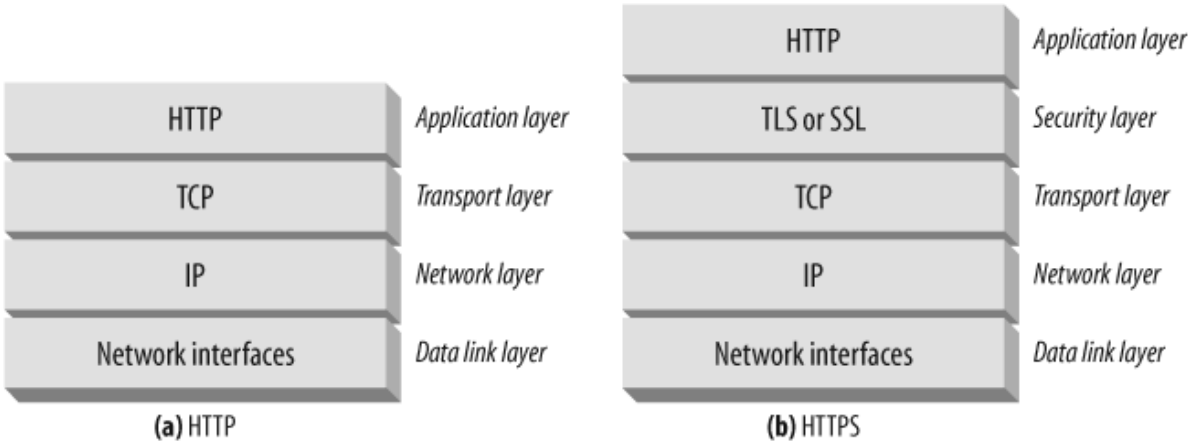
14.7 HTTPS의 세부사항

HTTPS: HTTP 프로토콜에 대칭, 비대칭 인증서 기반 암호 기법의 강력한 집합을 결합한 것.

14.7.1 HTTPS 개요

HTTPS는 그냥 **보안 전송 계층**을 통해 전송되는 HTTP. 즉, HTTP를 TCP로 보내기 전에 암호화하는 보안 계층으로 보냄.

오늘날의 보안 계층은 **SSL/TLS**로 구현되어있음.

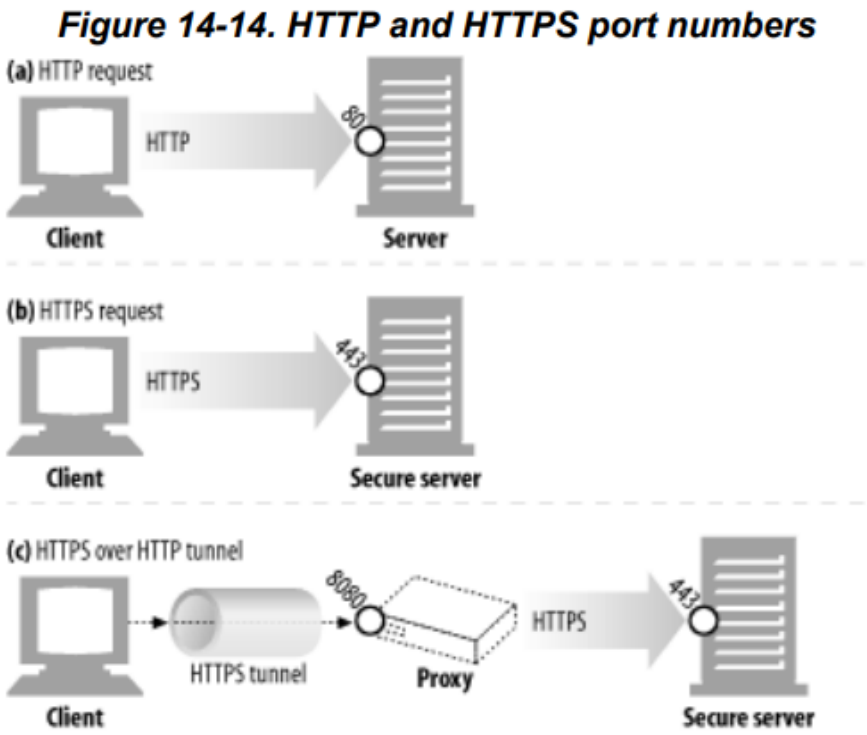


14.7.2 HTTPS 스킴

웹서버 요청 시, HTTPS로 통신한다고 알려줘야할 필요가 있는데, URL 스킴으로 이뤄짐

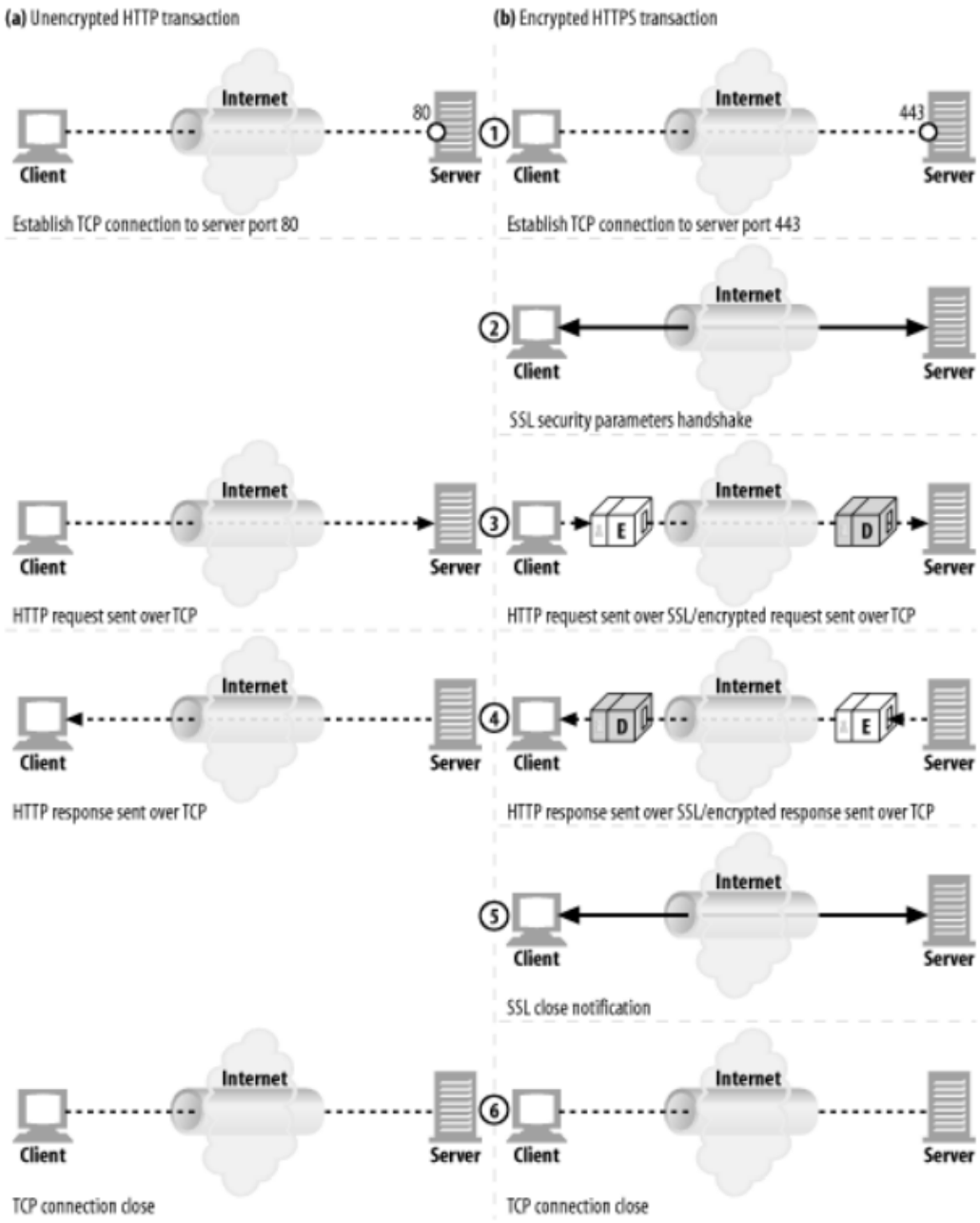
EX)

- http://example.com:80/
- https://example.com:443/



HTTP와 달리 HTTPS는 서버와 **'바이너리 포맷의 SSL 보안 매개변수'**를 교환하는 **'핸드셰이크'**를 하고 암호화된 HTTP가 뒤를 잇는다.

14.7.3 보안 전송 셋업



HTTP

TCP 연결 -> 연결된 TCP로 HTTP 요청 -> 연결된 TCP로 HTTP응답 -> TCP 연결 끝

HTTPS

TCP 연결 -> SSL 핸드셰이크 -> SSL을 통해 암호화된 HTTP를 연결된 TCP로 요청 -> SSL로 암호화된 HTTP를 연결된 TCP로 응답 -> SSL 닫김 통지 -> TCP 연결 끝

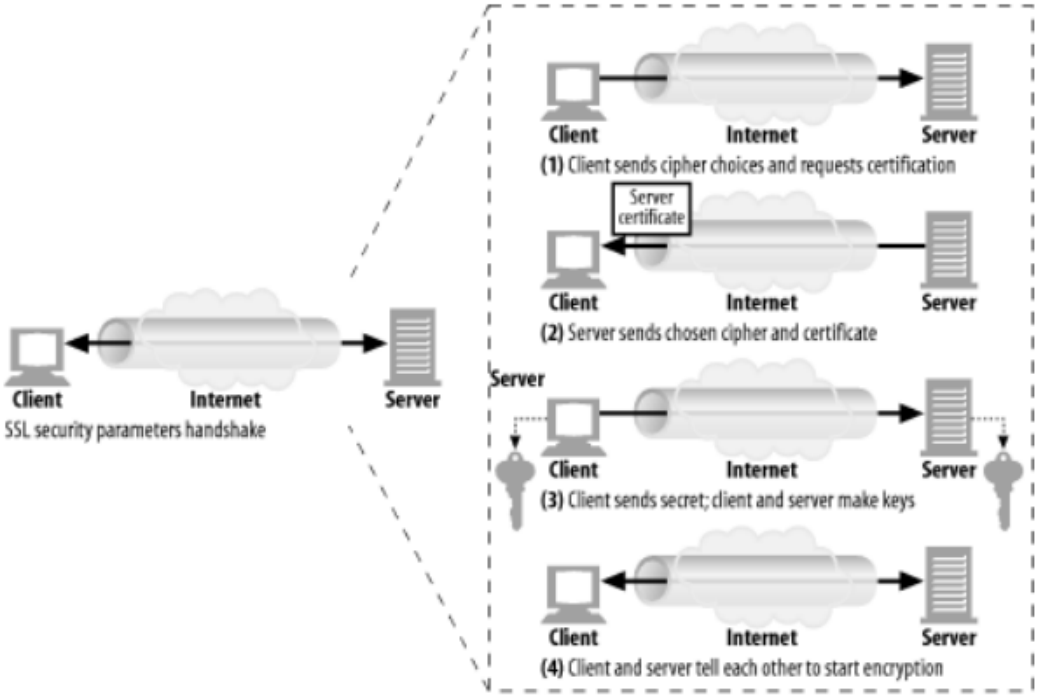
14.7.4 SSL 핸드셰이크

핸드셰이크 시 다음과 같은 일이 발생

- 프로토콜 저번 번호 교환
- 양쪽이 아는 암호 선택
- 양쪽 신원 인증
- 채널 암호화를 위한 임시 세션 키 생성

위를 통해 오가는 데이터 양은 상당함.

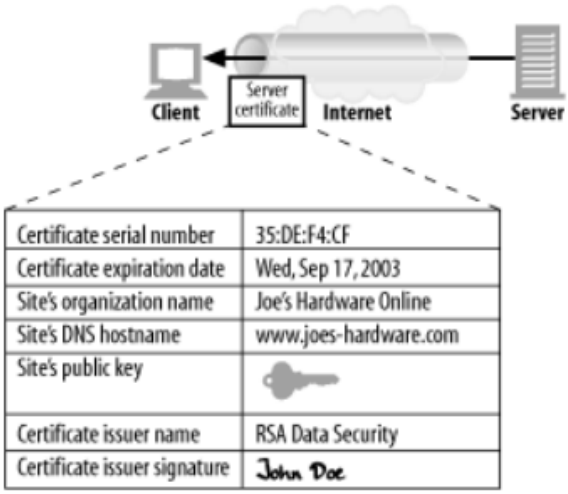
Figure 14-16. SSL handshake (simplified)



14.7.5 서버 인증서

HTTPS 트랜잭션은 항상 서버 인증서를 요구함: 클라이언트가 웹 서버에 중요한 정보를 보낼 때 해당 서버 (기관 혹은 조직)이 **신뢰할 수 있을 만한 곳인지를 검증하기 위함.**

Figure 14-17. HTTPS certificates are X.509 certificates with site information



14.7.6 사이트 인증서 검사

SSL자체는 서버 인증서 검사를 요구하지 않지만, 브라우져에서 기본적인 검사를 진행해줌.

이때 인증서 검사를 위한 알고리즘 수행 단계는 다음과 같음

1. 날짜 검사: 인증서의 유효기간을 혹인
2. 서명자 신뢰도 검사: 서버를 보증하는 인증기관 (Certificate authority, CA)에 의해 서명되었는지 확인. 웹 브라우저는 신뢰할만한 서명기관의 목록을 가지고 있음. 확인이 안되는 서명기관인 경우 경고를 띄움
3. 서명 검사: 서명기관의 공개키를 서명에 적용 후 체크섬과 비교하여 인증서의 무결성을 검사함
4. 사이트 신원 검사: 인증서 복사 및 트래픽 갈취를 방지하기 위해, 인증서의 도메인 이름과 서버 도메인을 비교하여 검사함. 호스트명이 인증서의 신원과 맞이 않으면 인증서 에러와와 함께 커넥션을 끊어야함

14.7.7 가상 호스팅과 인증서

가상 호스트(하나의 서버에 여러 호스트 명)로 운영되는 사이트의 경우, 인증서의 이름과 호스트 명이 다른 경우 경고를 표시할 수 있다.

이를 피하기 위해서는, 보안 트랜잭션을 시작하는 모든 사용자를 **인증서에 등록된 호스트로 리다이렉트** 하면 된다.

14.8 진짜 HTTPS 클라이언트

SSL은 복잡한 바이너리 프로토콜 -> 이를 위한 오픈 소스 라이브러리들이 존재

14.8.1 OpenSSL

OpenSSL: SSL과 TLS의 가장 인기 있는 오픈 소스.

14.8.2 간단한 HTTPS 클라이언트

1. TCP 네트워킹 및 SSL 지원을 위한 객체 생성
2. 호스트명을 ip 주소로 변환
3. 소켓을 생성하여 서버의 443 포트로 tcp 커넥션을 생성
4. SSL 레이어를 tcp 커넥션에 붙히고, SSL 핸드셰이크를 통해 암호 선택 및 인증 교환을 완료
5. 인증서 검증 (코드 상에서는 그냥 내용 출력)
6. HTTP 요청 및 응답 주고받음.

14.9 프락시를 통한 보안 트래픽 터널링

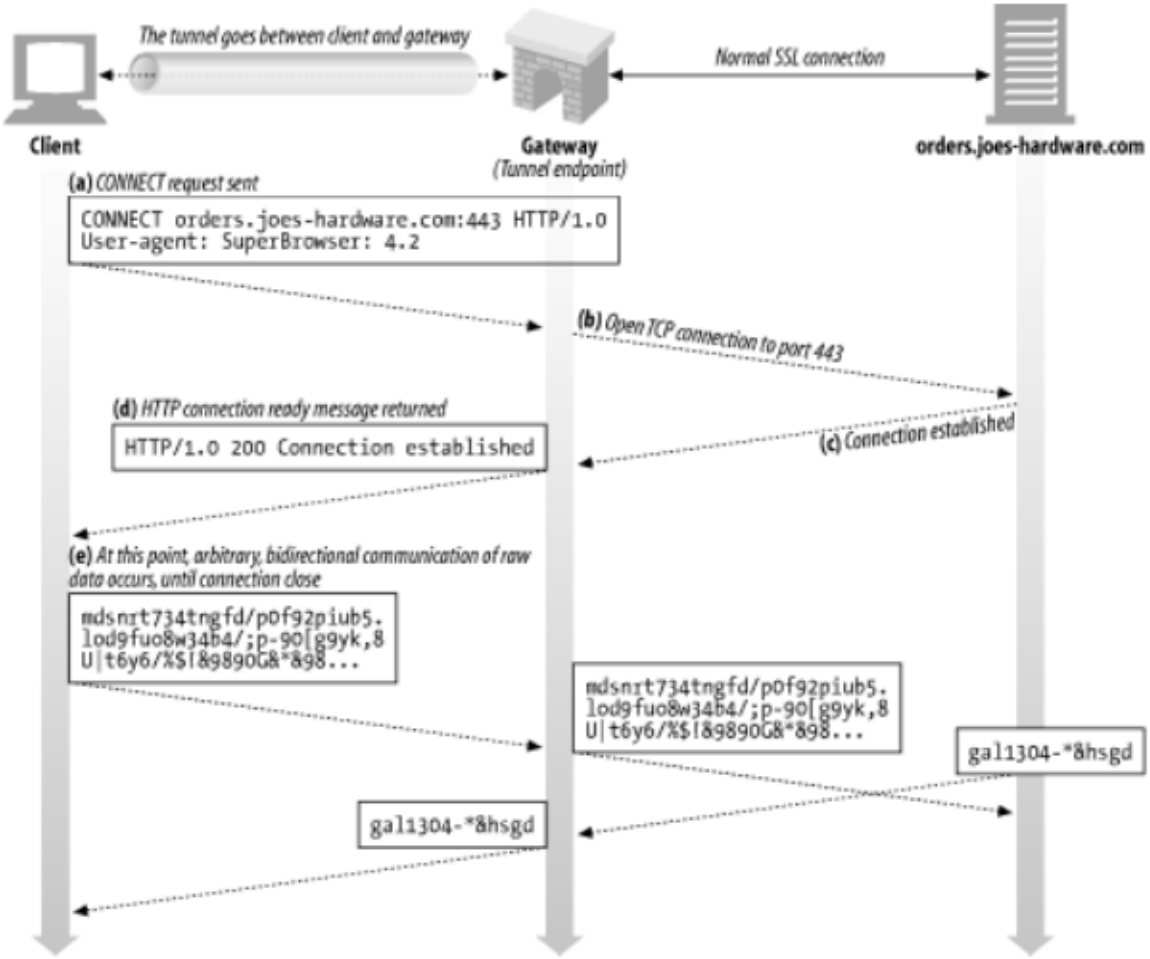
웹 프락시 서버가 존재하는 경우, 클라이언트가 데이터를 서버의 공개키로 암호화 하게되면, 프락시는 http header을 읽을 수 없게되고, 요청을 어디로 해야하는지 모르게 된다.

Figure 14-20. Proxy can't proxy an encrypted request

이를 해결하기 위해서 **클라이언트 프락시에게 어디로 접속하는지 말해주는 방법을 수정해야함.**

대표적인 방법으로는 **HTTP SSL 터널링 프로토콜**이 있다.

Figure 8-10. Using CONNECT to establish an SSL tunnel



1. CONNECT 요청을 통해 **호스트 전달** 및 터널 생성
2. 호스트로 연결 완료 시 성공 했음을 알림 (200 Connection Established)
3. 암호화된 값 주고 받음