HTTP/HTTPS

HTTP

HTTP(Hypertext Transfer Protocol)는 클라이언트와 서버 간 HTML과 같은 하이퍼미디어 문서를 전송하기 위한 통신 규칙 세트 또는 프로토콜이다. 사용자가 웹 사이트를 방문하면 사용자 브라우저가 웹 서버에 HTTP 요청을 전송하고 웹 서버는 HTTP 응답으로 응답한다.

즉, HTTP는 웹에서 이루어지는 모든 데이터 교환의 기초이며, 클라이언트-서버 프로토콜인 것이다. **클라이언트-서버 프로토콜**이란 수신자 측에 의해 요청이 초기화되는 프로토콜을 의미한다.

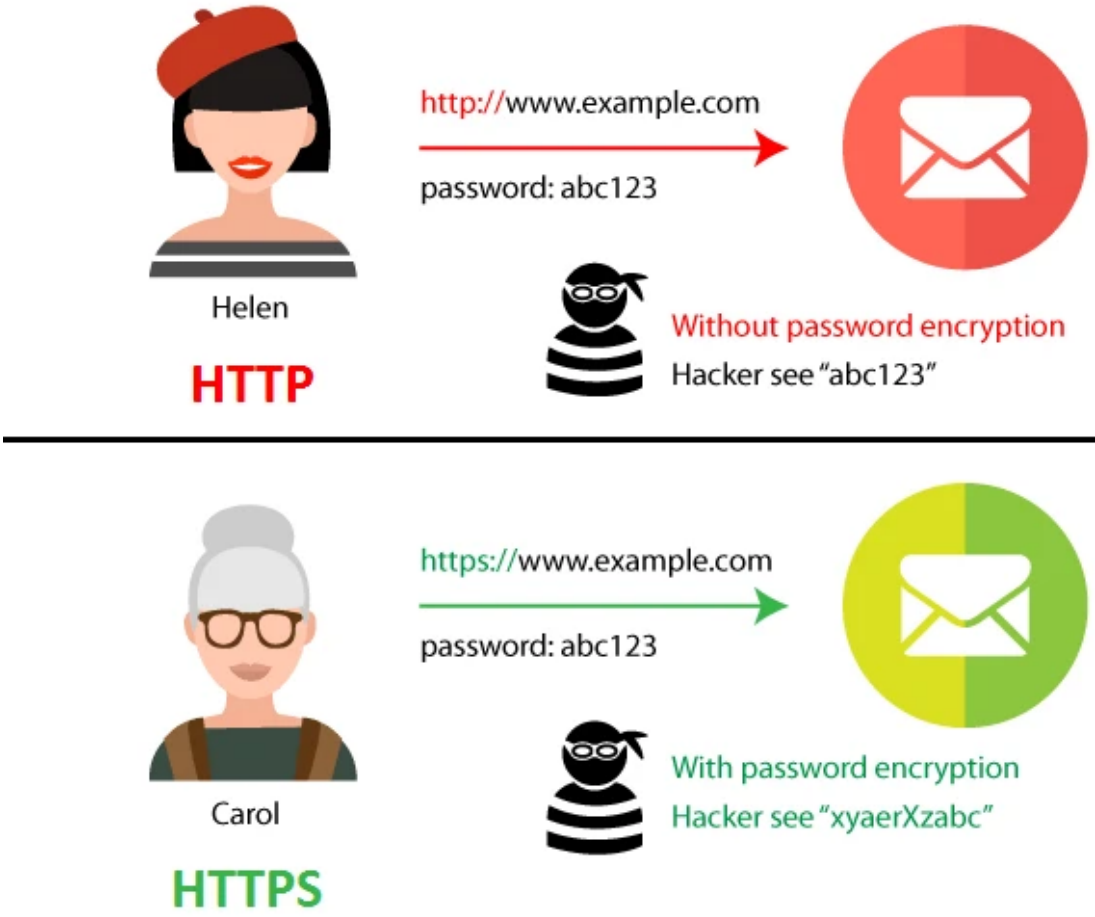
클라이언트와 서버는 개별적인 메시지 교환에 의해 통신한다.

- **요청(Requests)**: 클라이언트에 의해 전송되는 메시지
- **응답(Responses)**: 서버에서 응답으로 전송되는 메시지

HTTP가 진화하면서, 하이퍼텍스트 문서뿐만 아니라 이미지, 비디오, HTML 폼 결과와 같은 내용을 서버로 POST하거나, 필요할 때마다 웹 페이지를 갱신하기 위해 문서의 일부를 가져올 수도 있게 되었다.

HTTP의 문제점

1. **보안 취약성**: HTTP는 데이터를 암호화하지 않고 평문으로 전송한다. 이는 중간에 제3자가 네트워크 상에서 데이터를 엿볼 수 있으며, 개인 정보나 민감한 데이터가 노출될 수 있는 보안 취약점을 가지고 있다.
2. **무결성 문제**: HTTP는 데이터의 무결성을 보장하지 않는다. 데이터가 전송 중에 변경되거나 조작될 수 있으며, 이로 인해 정보의 정확성이 보장되지 않는 문제가 있다.
3. **신원 보증의 부재**: HTTP는 서버의 신원을 확인하는 기능이 없다. 따라서 중간에 제3자가 서버를 위장하여 클라이언트와 통신할 수 있으며, 이는 사이버 공격의 가능성을 높이는 요인이 된다.



HTTPS

HTTPS(Hypertext Transfer Protocol Secure)는 하이퍼 텍스트 전송 프로토콜 보안으로 표준 HTTP와 동일한 방식으로 작동한다. 서버와 주고받는 데이터가 암호화되기 때문에 웹사이트에 추가적인 보호를 제공한다. 즉, 개인 데이터를 훔치거나, 해킹하거나 볼 수 없도록 작동한다.

즉, HTTP의 주요한 문제들을 해결하기 위해 HTTPS가 나오게 되었다. 맨 마지막 문자가 Secure인 만큼 보안 강화에 큰 역할을 한다.

1. **보안 취약성 해결**: HTTPS는 SSL/TLS 프로토콜을 사용하여 데이터의 암호화를 제공한다. 클라이언트와 서버 간의 통신은 대칭키(세션 키)를 사용한 암호화 방식으로 이루어지며, 중간에 제3자가 데이터를 엿볼 수 없도록 보안적으로 강화된다.

2. **무결성 보장**: HTTPS는 데이터의 무결성을 보장하기 위해 메시지 다이제스트(Message Digest)를 사용한다. 데이터를 전송할 때 각 데이터 블록에 대한 다이제스트 값을 생성하고, 이를 서버에서 검증하여 데이터의 변조 여부를 확인한다. 이를 통해 데이터가 전송 중에 변경되지 않았는지를 확인하고 데이터의 무결성을 보장한다.
3. **신원 보증**: HTTPS는 인증 기관(Certificate Authority)에 의해 발급된 서버 인증서를 사용하여 서버의 신원을 확인한다. 클라이언트는 서버의 인증서를 검증하고, 서버의 공개키를 획득하여 안전한 통신을 위한 대칭키 교환에 사용한다. 이를 통해 중간에 제3자가 서버를 위장하거나 조작하는 것을 방지하고, 신뢰할 수 있는 서버와의 통신을 보장한다.

- 이때 헤더는 암호화되지 않고, 바디 값만 암호화함

HTTPS의 동작 과정

HTTPS는 대칭키 암호화와 비대칭키 암호화를 모두 사용하여 빠른 연산 속도와 안정성을 모두 얻고 있다.

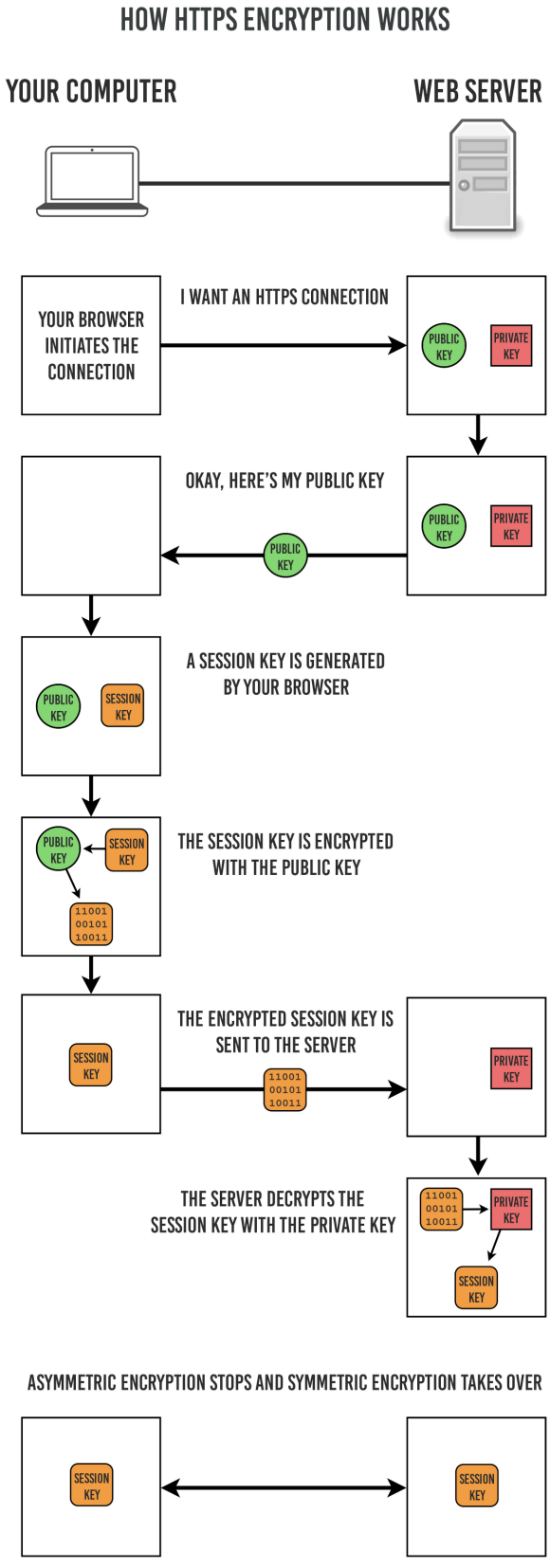
HTTPS 연결 과정(Hand-Shaking)에서는 먼저 서버와 클라이언트 간에 세션키를 교환한다. 여기서 세션키는 주고 받는 데이터를 암호화하기 위해 사용되는 대칭키이며, 데이터 간의 교환에는 빠른 연산 속도가 필요하므로 세션키는 대칭키로 만들어진다.
문제는 이 세션키를 클라이언트와 서버가 어떻게 교환할 것이냐 인데, 이 과정에서 비대칭키가 사용된다.

즉, 처음 연결을 성립하여 안전하게 세션키를 공유하는 과정에서 비대칭키가 사용되는 것이고, 이후에 데이터를 교환하는 과정에서 빠른 연산 속도를 위해 대칭키가 사용되는 것이다.

실제 HTTPS 연결 과정이 성립되는 흐름을 살펴보면 다음과 같다.

1. 클라이언트(브라우저)가 서버로 최초 연결 시도를 함
2. 서버는 공개키(엄밀히는 인증서)를 브라우저에게 넘겨줌
3. 브라우저는 인증서의 유효성을 검사하고 세션키를 발급함
4. 브라우저는 세션키를 보관하며 추가로 서버의 공개키로 세션키를 암호화하여 서버로 전송함
5. 서버는 개인키로 암호화된 세션키를 복호화하여 세션키를 얻음

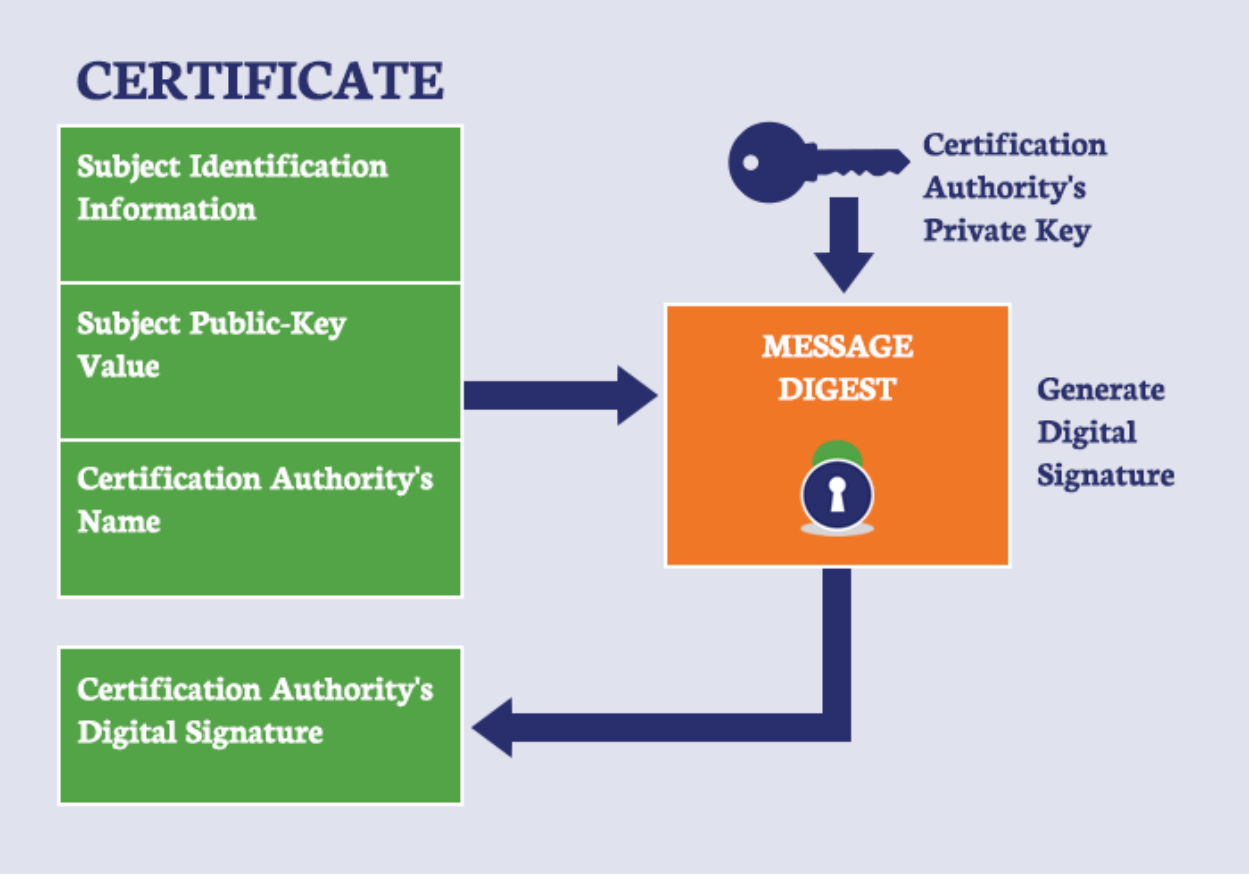
6. 클라이언트와 서버는 동일한 세션키를 공유하므로 데이터를 전달할 때 세션키로 암호화/복호화를 진행함


HOW HTTPS ENCRYPTION WORKS
YOUR COMPUTER
WEB SERVER
YOUR BROWSER INITIATES THE CONNECTION
I WANT AN HTTPS CONNECTION
PUBLIC KEY
PRIVATE KEY
OKAY, HERE'S MY PUBLIC KEY
PUBLIC KEY
PRIVATE KEY
PUBLIC KEY
A SESSION KEY IS GENERATED BY YOUR BROWSER
SESSION KEY
PUBLIC KEY
SESSION KEY
THE SESSION KEY IS ENCRYPTED WITH THE PUBLIC KEY
11001 00101 10011
SESSION KEY
THE ENCRYPTED SESSION KEY IS SENT TO THE SERVER
11001 00101 10011
PRIVATE KEY
THE SERVER DECRYPTS THE SESSION KEY WITH THE PRIVATE KEY
11001 00101 10011
PRIVATE KEY
SESSION KEY
ASYMMETRIC ENCRYPTION STOPS AND SYMMETRIC ENCRYPTION TAKES OVER
SESSION KEY
SESSION KEY

HTTPS의 발급 과정

서버가 비대칭키를 발급받는 과정이다. 서버는 클라이언트와 세션키를 공유하기 위한 공개키를 생성해야 하는데, 일반적으로는 인증된 기관(Certificate Authority)에 공개키를 전송하여 인증서를 발급받는다.

1. A기업은 HTTP 기반의 애플리케이션에 HTTPS를 적용하기 위해 공개키/개인키를 발급함
2. CA 기업에게 돈을 지불하고, 공개키를 저장하는 인증서의 발급을 요청함
3. CA 기업은 CA기업의 이름, 서버의 공개키, 서버의 정보 등을 기반으로 인증서를 생성하고, CA 기업의 개인키로 암호화하여 A기업에게 이를 제공함
4. A기업은 클라이언트에게 암호화된 인증서를 제공함
5. 브라우저는 CA기업의 공개키를 미리 다운받아 갖고 있어, 암호화된 인증서를 복호화함
6. 암호화된 인증서를 복호화하여 얻은 A기업의 공개키로 세션키를 공유함


CERTIFICATE
Subject Identification Information
Subject Public-Key Value
Certification Authority's Name
Certification Authority's Digital Signature
Certification Authority's Private Key
MESSAGE DIGEST
Generate Digital Signature