



# HTTP/HTTPS



목차

🤔 HTTP란?

[구조](#)

[버전](#)

[HTTP/0.9 \(One-Line Protocol\)](#)

[HTTP/1.0](#)

[HTTP/1.1 \(표준 프로토콜의 등장\)](#)

[HTTP/2.0](#)

[HTTP/3.0 : QUIC](#)

🦊 HTTPS란?

[암호화 방식](#)

[비대칭키 암호화](#)

[암호화 선택](#)

[HTTPS 동작 과정](#)

[HTTPS 연결 흐름](#)

🔑 [HTTP vs HTTPS](#)

[🔗 참고](#)

## 🤔 HTTP란?

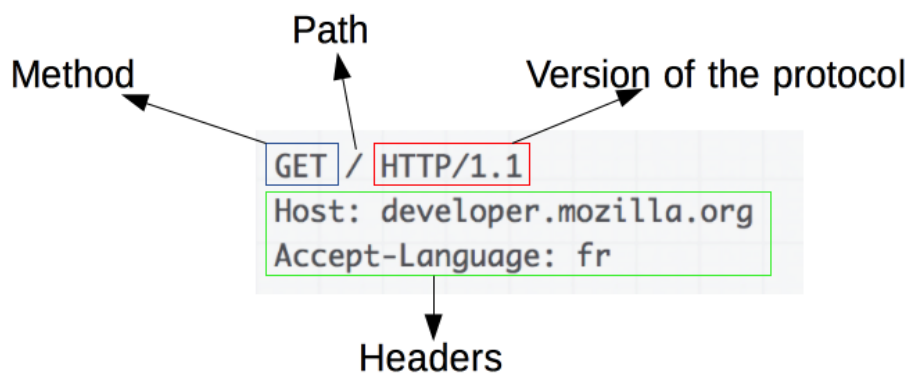
- HTTP는 Hyper Text Transfer Protocol의 약자로 서버-클라이언트 모델을 따라 데이터를 주고 받기 위한 프로토콜
- 즉, HTTP는 인터넷에서 하이퍼텍스트를 교환하기 위한 통신 규약으로 80번 포트를 사용하고 있음
  - HTTP 서버는 80번 포트에서 요청을 기다리고 있으며, 클라이언트는 80번 포트로 요청을 보내게 됨



## 하이퍼텍스트?

- 다른 문서와 쉽게 연결이 되도록 하는 '링크'의 모음으로 구성된 문서
- Web 기술은 인터넷 서버에 올린 문서를 네트워크를 통해 다른 사람도 열람할 수 있도록 하는데, 이 문서 형식에 "하이퍼텍스트"가 사용된다
- 하이퍼텍스트 내에는 "하이퍼링크"라고 하는 다른 문서에 대한 참조 정보가 들어있다

## 구조



- HTTP는 Application Level의 프로토콜
- TCP/IP 위에서 작동한다
- Stateless 프로토콜이다
- Method, Path, Version, Headers, Body 등으로 구성된다
  - 그러나 HTTP는 암호화 되지 않은 평문 데이터를 전송하는 프로토콜
  - 따라서, HTTP로 비밀번호나 주민등록번호 등을 주고받으면 제3자가 정보를 조회할 수 있다
  - 이런 문제를 해결하기 위해 HTTPS가 등장

## 버전

### HTTP/0.9 (One-Line Protocol)

- 요청은 단일 라인으로 구성, method는 GET만 존재
- 상태 혹은 오류 코드가 없었다

## HTTP/1.0

- HTTP 헤더 개념을 통해 request와 response 모두 meta data 전송을 허용해 프로토콜을 유연하고 확장 가능하게 함
- 버전 정보와 method가 함께 전송되기 시작
- response 시작 부분에 상태 코드 라인이 추가되어 브라우저 요청의 성공, 실패를 파악 가능
- HTML 이외의 문서 전송 기능이 가능해짐
  - 그렇지만 커넥션 하나 당 요청 하나와 응답 하나만 처리가 가능하다
  - 비효율적이며 서버 부하 문제가 발생

## HTTP/1.1 (표준 프로토콜의 등장)

- HTTP의 첫 번째 표준 버전
- Persistent Connection
  - 지정한 timeout 동안에 connection 재사용이 가능
  - 기존 연결에 대해 handshaking 생략
- Pipelining 추가
  - 앞 요청의 응답을 기다리지 않고 순차적으로 여러 요청을 연속적으로 보냄
  - 그 순서에 맞춰 응답을 받는 방식

## HTTP/2.0

- 기존 HTTP1.x 버전의 성능 향상에 초점을 맞춘 프로토콜
- HTTP1.1의 대체가 아닌 확장의 개념
- 기존엔 일반 텍스트 형식을 이용했으나, 추가적인 Binary Encoding을 이용해 파싱 속도와 전송 속도가 빠르고 오류 발생 가능성이 낮아짐

## HTTP/3.0 : QUIC

- QUIC : Quick UDP Internet Connections
  - TCP를 대체하는 범용 목적의 전송 계층 통신 프로토콜

- UDP와 동일하게 Transport Layer에서 구동되지만, UDP 위에 새로운 계층을 추가해 신뢰성을 제공



## HTTPS란?

- HTTP over Secure Socket Layer
- HTTP에 데이터 암호화가 추가된 프로토콜
- HTTP와 다르게 443번 포트를 사용하며, 네트워크 상에서 중간에 제3자가 정보를 볼 수 없도록 암호화를 지원한다

### 암호화 방식

- HTTP는 대칭키 암호화 방식과 비대칭키 암호화 방식을 모두 사용

#### 1. 대칭키 암호화

- 클라이언트와 서버가 동일한 키를 이용해 암호화/복호화를 진행
- 키가 노출되면 매우 위험하지만 연산 속도가 빠르다

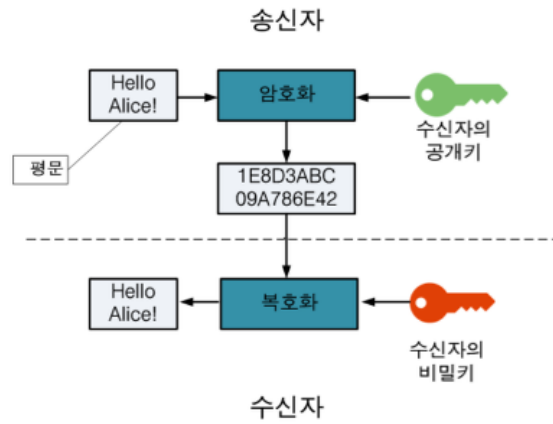
#### 2. 비대칭키 암호화

- 1개의 쌍으로 구성된 공개키와 개인키를 암호화/복호화 하는데 사용함
- 키가 노출되어도 비교적 안전하지만 연산 속도가 느리다

### 비대칭키 암호화

- 비대칭키 암호화는 공개키/개인키 암호화 방식을 이용해 데이터를 암호화
  - 공개키와 개인키는 서로를 위한 한 쌍의 키이다
  - 공개키 : 모두에게 공개 가능한 키
  - 개인키 : 나만 알고 있어야 하는 키

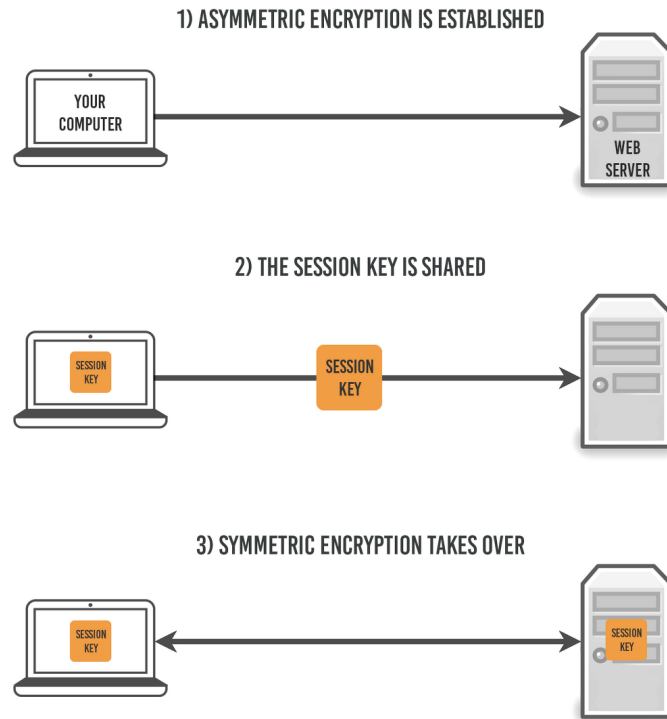
### 암호화 선택



- 암호화를 공개키로 하느냐, 개인키로 하느냐에 따라서 얻는 효과가 다름
  1. 공개키로 암호화를 했을 때
    - 개인키로만 복호화를 할 수 있다
    - 개인키는 나만 가지고 있으므로, 나만 볼 수 있다
  2. 개인키로 암호화를 했을 때
    - 공개키로 복호화를 할 수 있다
    - 공개키는 모두에게 공개되어 있으므로, 내가 인증한 정보임을 정보의 신뢰성을 보장할 수 있다

## HTTPS 동작 과정

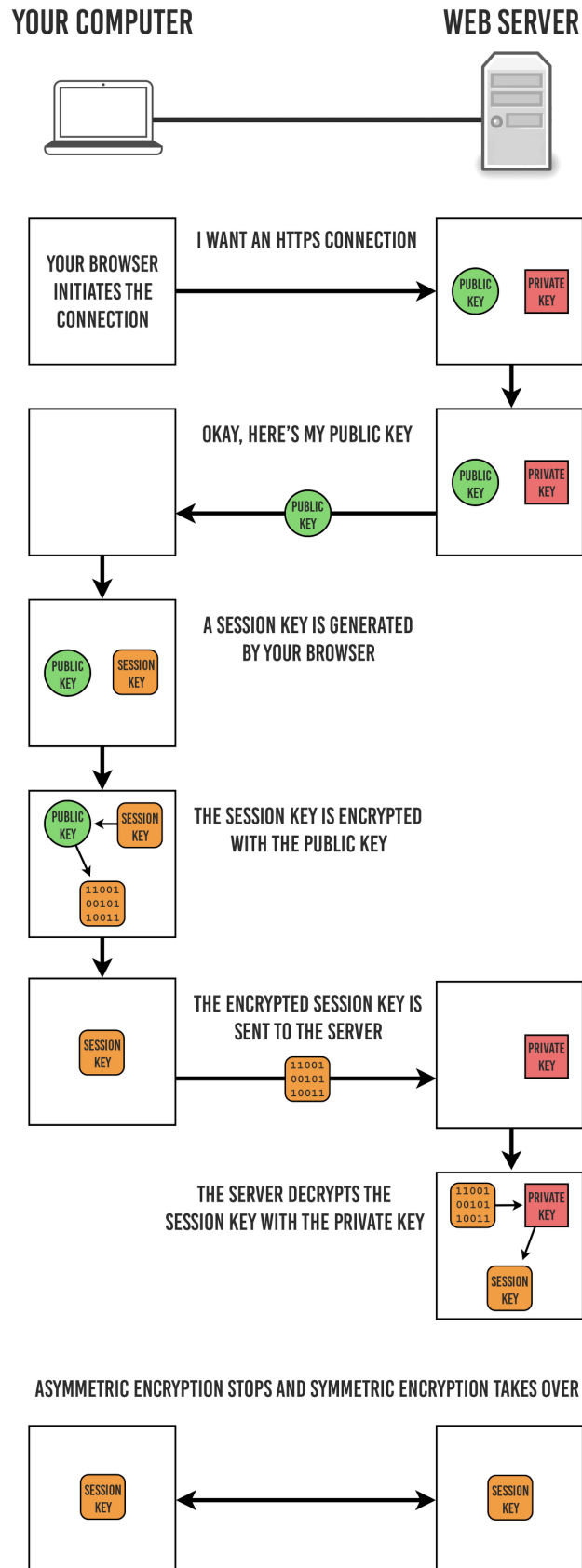
## HOW PKI WORKS



- HTTPS는 대칭키 암호화와 비대칭키 암호화를 모두 사용하여 빠른 연산 속도와 안정성을 얻고 있다
1. HTTPS 연결 과정(hand-shaking)에서는 먼저 서버와 클라이언트 간에 세션키를 교환한다
    - 여기서 **세션키**는 주고 받는 데이터를 암호화하기 위해 사용되는 **대칭키**이다
    - 데이터의 교환에는 빠른 연산 속도가 필요하기 때문
  2. 이 세션키를 교환하는 과정에서 비대칭키가 사용된다
    - 처음 연결을 성립하여 안전하게 세션키를 공유하는 과정에서 비대칭키가 사용
    - 이후에 데이터를 교환하는 과정에선 빠른 연산 속도를 위해 대칭키가 사용

## HTTPS 연결 흐름

## HOW HTTPS ENCRYPTION WORKS



1. 브라우저가 서버로 최초 연결 시도
2. 서버는 공개키(인증서)를 브라우저에게 넘김
3. 브라우저는 인증서의 유효성을 검사하고 세션키(주고 받는 데이터를 암호화하기 위해 사용되는 대칭키)를 발급
4. 브라우저는 세션키를 보관하고, 서버의 공개키로 세션키를 암호화한 후 서버로 전송
5. 서버는 가지고 있는 개인키로 암호화된 세션키를 복호화해서 세션키를 얻음
  - 서버의 공개키로 암호화 된 정보는 서버가 가지고 있는 개인키로만 복호화가 가능하기 때문
6. 클라이언트와 서버는 동일한 세션키를 공유하므로 데이터를 전달할 때 세션키로 암호화/복호화를 진행

## HTTP vs HTTPS


	HTTP	HTTPS
암호화	X	O
속도	비교적 빠름	비교적 느림
추가 비용	추가 비용 발생 X	인증서를 발급하고 유지하기 위한 비용 발생

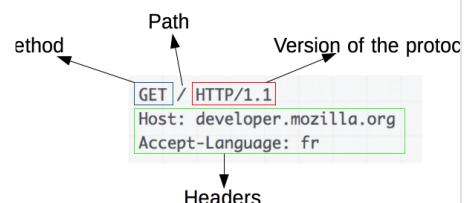
- 개인 정보와 같은 민감한 데이터를 주고 받아야 한다면 HTTPS를 이용
- 노출이 되어도 괜찮은 단순 정보 조회만 처리한다면 HTTP를 이용

## 참고

### ▼ 링크

#### [Web] HTTP와 HTTPS의 개념 및 차이점

1. HTTP란? [ HTTP(Hyper Text Transfer Protocol)란? ]  
HTTP(Hyper Text Transfer Protocol)란 서버/클라이언트 모델을 따라 데이터를 주고 받기 위한 프로토콜이다. 즉, HTTP는  
 <https://mangkyu.tistory.com/98>

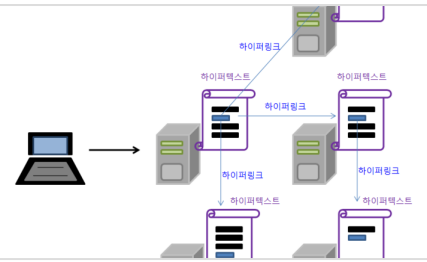




## 하이퍼텍스트(Hypertext)와 URL, 뜻과 이해


하이퍼텍스트(Hypertext) 조직이나 개인이 인터넷에 개설한 홈페이지는 WWW (world wide web), 간단히 ...

 <https://m.blog.naver.com/hai0416/221623211571>



## [Network] HTTP 버전 별 특징: HTTP v0.9 v1.0 v1.1 v2 v3

💡 본 문서는 'HTTP 버전 별 특징: HTTP v0.9 v1.0 v1.1 v2 v3'에 대해 정리해놓은 글입니다. HTTP는 여러 과정을 거쳐 현재의 웹의 표준으로 자리잡게 되었는데, 하단에서는 HTTP의 역사와 함께 변천과정에

 <https://csj000714.tistory.com/733>

