HTTPS vs HTTP & 인증서

HTTPS vs HTTP

HTTP 프로토콜은 정보를 전달하는 것 자체에 목적을 맞춤
그렇기 때문에 정보가 전달되는 과정에 대한 보호가 부족함
이로 인해서 중간에 정보가 빼돌려지거나, 수정되는 경우가 존재할 수 있음

HTTPS는 HTTP와 **암호화(Encryption)**가 합쳐진 것.
HTTPS는 HTTP보다 보안 성능이 높음. S는 'secure'을 의미
두 프로토콜의 유일한 차이점은 HTTP 요청과 응답에 **TLS(SSL)** 사용 유무
HTTPS 프로토콜을 사용하는 웹사이트의 경우, 스킴은 https
HTTP 프로토콜을 사용하는 웹사이트의 경우, 스킴은 http

```
GET /hello.txt HTTP/1.1
User-Agent: curl/7.63.0 libcurl/7.63.0
OpenSSL/1.1.l zlib/1.2.11
Host: www.example.com Accept-
Language: en
```

attacker가 보는 HTTP 요청 메시지

```
t8Fw6T8UV81pQfyhDkhebbz7+oiwldr1j2
gHBB3L3RFTRsQCpaSnSBZ78Vme+DpDV
JPvZdZUZHpzbbcqmSW1+3xXGsERHg9Y
DmpYk0VVDiRvw1H5miNieJeJ/FNUjgH0B
mVRWII6+T4MnDwmCMZUI/orxP3HGwY
CSIvyzS3MpmmSe4iaWKCOHQ==
```

attacker가 보는 HTTPS 요청 메시지

※ attacker : eavesdropper(도청자), hacker

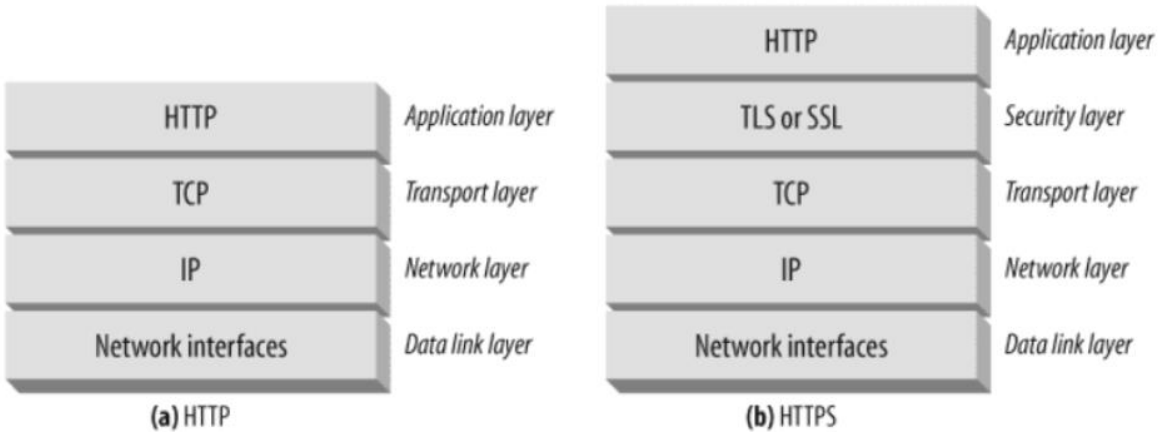
TLS와 SSL

SSL(Secure Sockey Layer), TLS(Transport Layer Security)는 암호 프로토콜로, 전달되는 데이터의 보안성 제공
응용 계층 프로토콜(HTTP, FTP, SMTP, IMAP)을 암호화 하기 위해 TCP 계층 위에 존재

※ 응용 계층 프로토콜
HTTP: HyperText Transfer Protocol
FTP: File Transfer Protocol
SMTP: Simple Mail Transfer Protocol
IMAP: Internet Message Access Protocol, 서버에서 이메일을 읽는 프로토콜

SSL은 넷스케이프에 의해 개발
1995년 SSL 2.0을 통해 처음으로 등장
1996년 SSL 3.0등장
1999년 SSL 3.0기반 TLS 등장

- TLS는 SSL의 차세대 프로토콜
- TLS는 IETF(Internet Engineering Task Force)에 의해 표준화
- TLS는 모든 종류의 인터넷 트래픽을 암호화



암호화

암호화와 복호화를 위해선 '키'가 필요

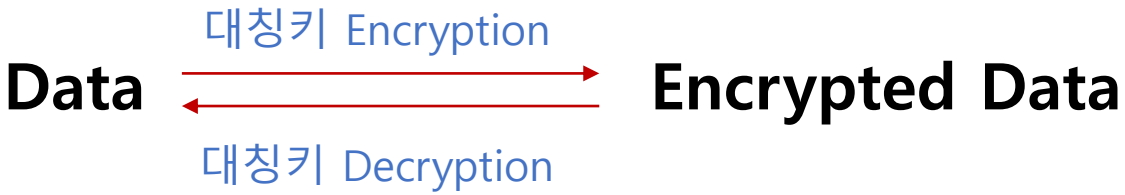
암호화 [데이터] + [키] → [암호화 알고리즘] → [암호화된 데이터]
복호화 [암호화된 데이터] + [키] → [복호화 알고리즘] → [데이터]

암호화 방법으로는,
1. 대칭키 암호화(Symmetric key algorithm)
하나의 키로 암호화와 복호화 사용
키를 해킹당하면 데이터가 노출됨

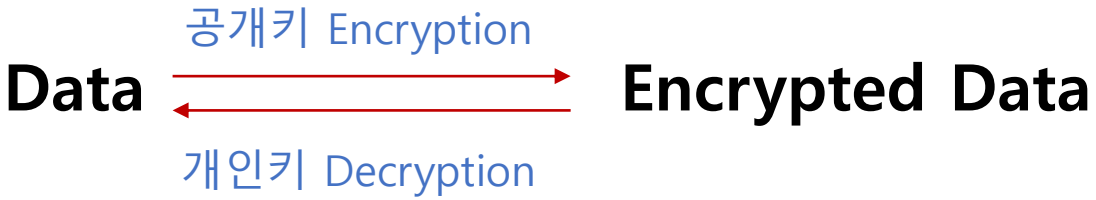
2. 비대칭키 암호화(Asymmetric key algorithm)
두개의 키(공개키와 비밀키)로 암호화와 복호화 사용
A, B 두개의 키가 있다면,
A키로 암호화된 데이터는 B키로만 복호화 가능
B키로 암호화된 데이터는 A키로만 복호화 가능. 이 과정을 **RSA 알고리즘**이라고 함
A, B 두개의 키 중 하나는 공개 키(Public Key) 하나는 비밀 키(Private Key)
키를 해킹당해도 데이터가 노출되지 않음

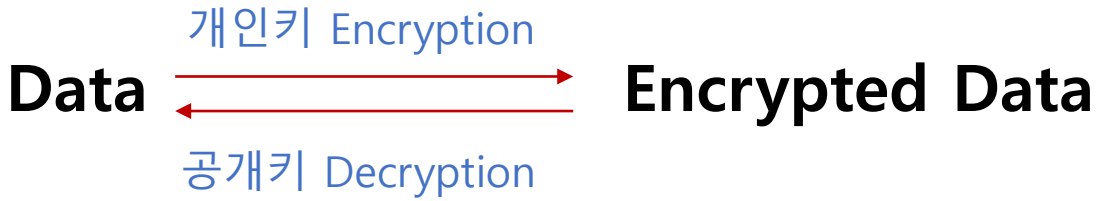
TLS는 대칭키 방식과 비대칭키 방식 모두 사용

암호화



대칭키 암호화





비대칭키 암호화

A의 공개 키: 1111, 비밀 키: cccc
B의 공개 키: 9999, 비밀 키: dddd

A → B : 내 공개키는 1111이야
B → A : 내 공개키는 9999야

A : '수원에서 만나자'를 9999로 암호화
B : 받은 메시지를 dddd로 복호화

B : '그래 좋아'를 1111로 암호화
A : 받은 메시지를 cccc로 복호화

RSA 알고리즘

암호화

TLS는 왜 대칭키와 비대칭키 방식을 같이 사용할까?

RSA 알고리즘 암호화 방식은 복잡한 수학적 원리로 이루어져 있어, CPU 리소스를 크게 소모한다는 단점
비대칭키 방식은 CPU 리소스를 크게 소모한다는 단점. 대칭키 방식에 비해서 해킹에 강함
대칭키 방식은 해킹에 취약하다는 단점. 비대칭키 방식에 비해 CPU 리소스 적게 소모

TLS는 처음에 대칭키 전달은 비대칭키 방식으로 진행
이후에 데이터 전달은 대칭키 방식으로 진행

A의 **공개 키: 1111** / **비밀 키: cccc**

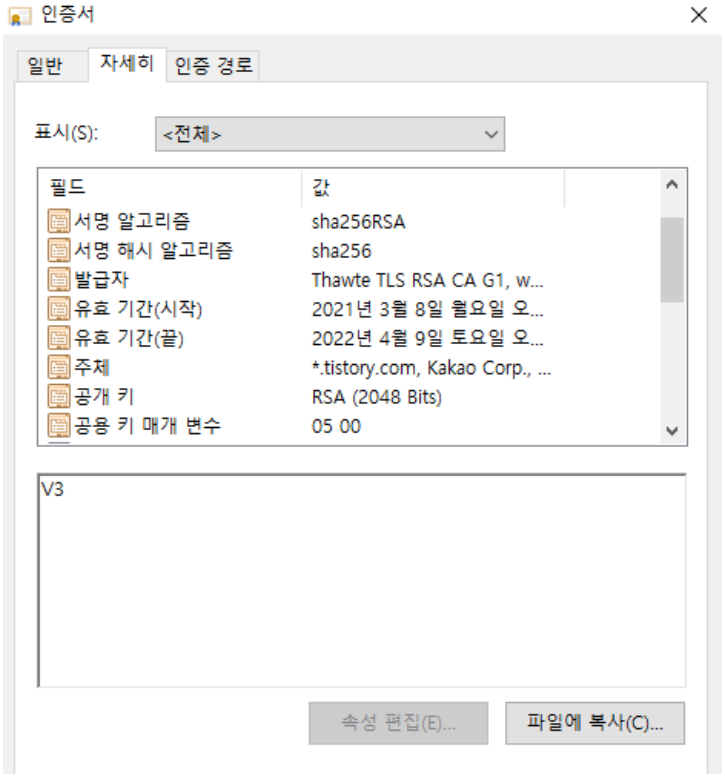
A → B : 내 공개 키는 1111이야
B → A : 대칭키로 사용할 1234를 1111로 암호화 및 전달
A : 받은 메세지를 cccc로 복호화 및 대칭키가 1111임을 알아냄

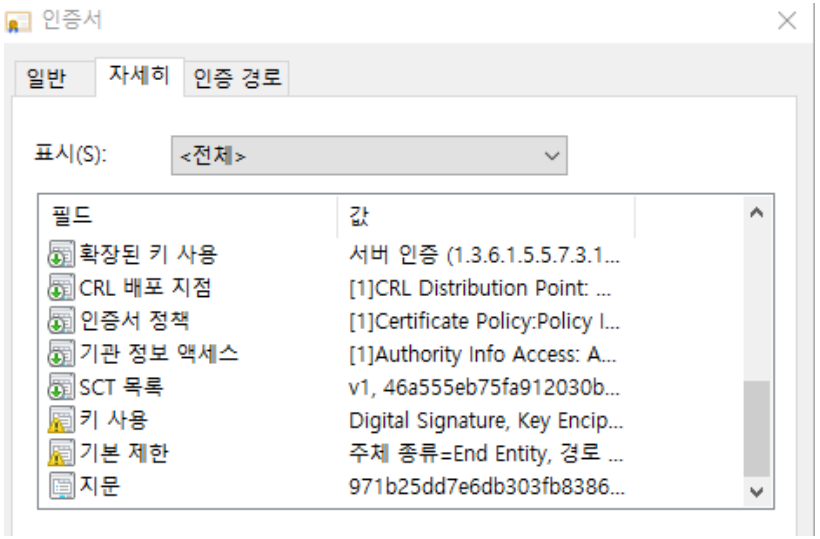
A → B : '수원에서 만나자'를 1234로 암호화
B → A : 받은 메세지를 1234로 복호화

인증서

그렇다면 가짜 서버에 데이터를 보내는 상황은 어떻게 해야할까?
서버가 신뢰할 수 있는 서버라는 것을 확인하는 작업이 필요. 이때 사용하는 것이 **'인증서'**

인증서는 다음 정보를 포함
(1) 서비스 정보(인증서를 발급한 CA, 서비스의 도메인 등)
(2) 서버 측 공개키(공개키, 공개키 암호화 방법)
(3) 지문, 디지털 서명 등





인증서

인증서는 어떻게 생성할까?
CA(Certificate Authority)는 인증서를 발급해주는 기관으로, **Root Certificate**라고도 부름
TLS 통신을 하기 위해서는 이 CA를 통해서 인증서를 발급 받아야 함
그리고 **CA는 자체적으로 공개키와 비밀키를 가지고 있고**, 비밀키는 절대 누설되서는 안됨

인증서 요청

도메인, 공개키 등 전달

A회사 검토
A회사 공개키를 SHA-256등으로 **해시하여 인증서에 '지문'으로 등록**
(Finger Print 과정)

지문을 CA의 **비밀키로 암호화하여 인증서에 '서명'으로 등록**
(Digital Signing 과정)

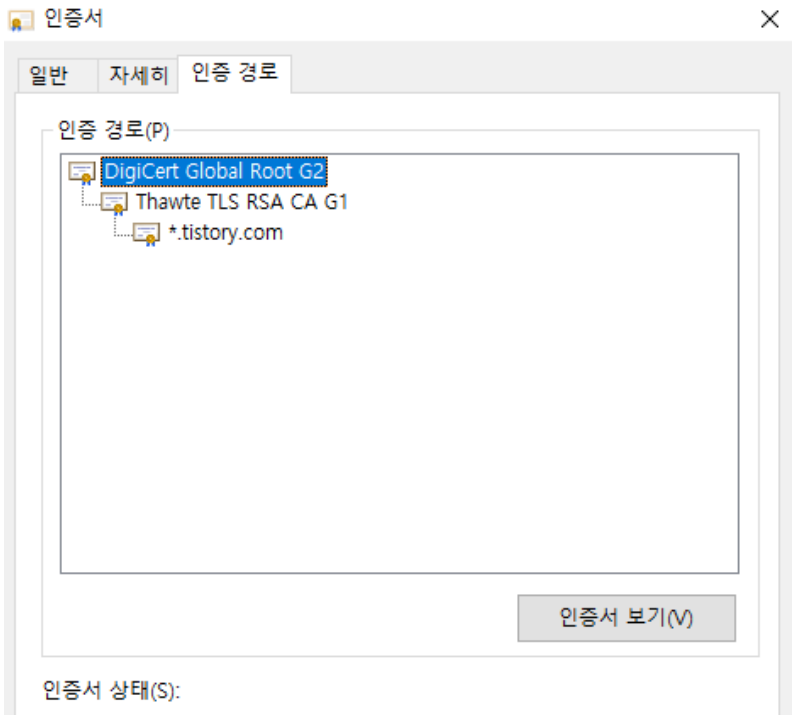
인증서 발급

인증서

하위 CA가 상위 CA로부터 인증서를 발급받고, 더 하위 CA가 상위 CA로부터 인증서를 발급받는 **'인증서 체인'** 존재

tistory의 인증서는 상위 인증서인 Thawte TLS RSA CA G1의 인증기관의 비밀키로 암호화된 것이며,
Thawte TLS RSA CA G1 인증서는 DigiCert Global Root G2의 인증기관의 비밀키로 암호화된 것
DigiCert Global Root G2는 상위 인증기관이 없는 **Root CA**이므로, **Self-Signed** 돼있음

※ Self-Signed
자신의 공개키를 해시한 후, 자신의 비밀키로 지문을 암호화 하여 서명 등록



인증서

CA 인증 없이 인증서를 생성할 수 있을까?

CA 인증과 상관 없이 발행하는 인증서를 **'사설 인증서'**라고 함
이 사설 인증서는 Root CA처럼 Self-Signed 되어 있음.

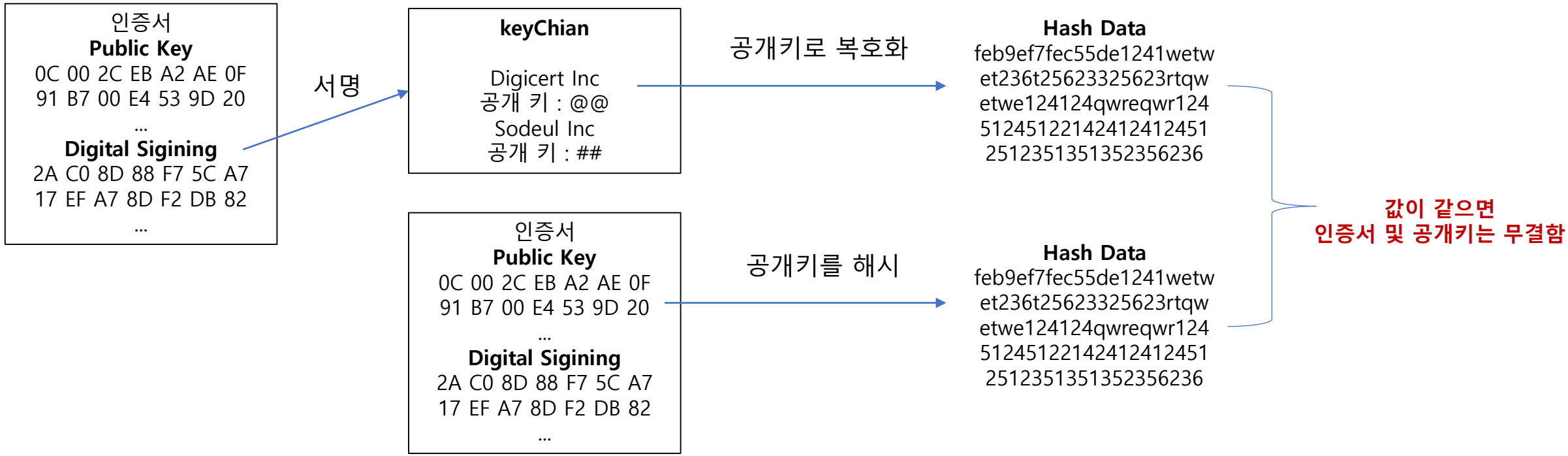
누구도 보증해주지 않는 인증서로, 사설 인증서를 받은 홈페이지에 접속 시 '신뢰할 수 없는 사이트입니다' 표시

인증서

인증서로 서버를 인증하는 방법은?
클라이언트는 인증서가 CA에서 발급받은 것인지, 중간에 누가 조작했는지 어떻게 확인할까?
무결성을 어떻게 증명?

클라이언트들은 이미 CA리스트를 갖고 있음
OS 설치시 PC에 포함(Mac은 keyChain에)되거나 브라우저가 포함(소스 코드에)하고 있음

디지털 서명을 CA 기관의 공개키로 복호화하여 나온 해시 값과
공개키를 해시한 값이 일치 한다면, 인증서가 위조되지 않았음을 인증



인증서

(Cont'd) 이후부터는 이전에 설명한 것과 동일하게 진행

클라이언트는 자신의 대칭키를 TLS 암호화 방식으로 서버에 전달
클라이언트는 서버의 공개키로 대칭키를 암호화하여 서버로 전달
서버는 자신의 비밀키로 복호화하여 클라이언트의 대칭키를 알아냄

HTTP와 HTTPS 차이

https://rsec.kr/?p=426https://smartits.tistory.com/209
https://www.itworld.co.kr/news/113007
https://www.youtube.com/watch?v=8R0FUF_t_zk
https://babbab2.tistory.com/4
https://rsec.kr/?p=426