알기 쉬운

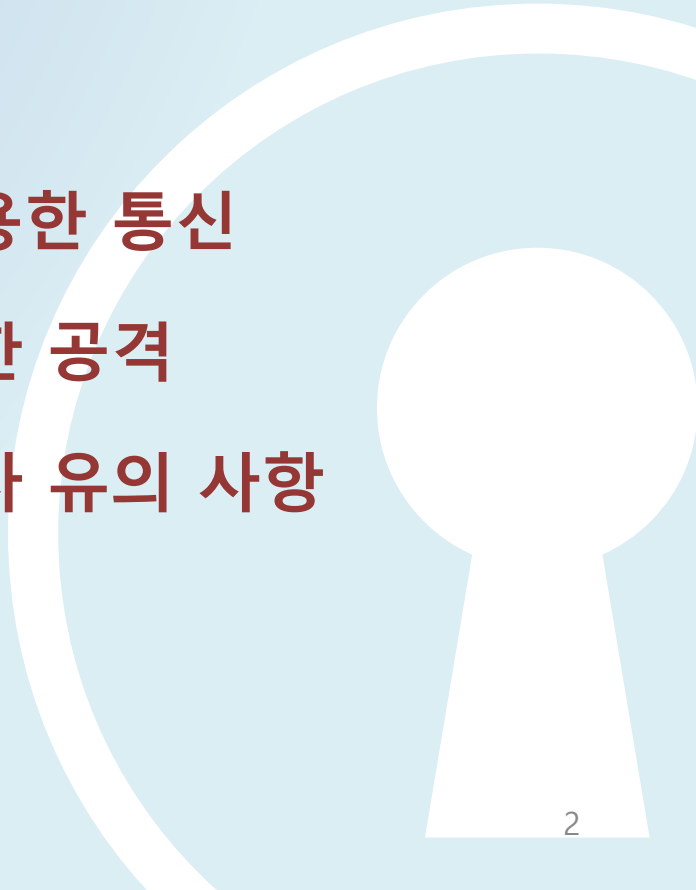
정보보호개론 3판

흥미로운 암호 기술의 세계

INFORMATION SECURITY and CRYPTOGRAPHY

INFINITYBOOKS
인피니티북스

CHAPTER 15 SSL/TLS



Section 01
SSL/TLS란

1.1 앨리스가 밥 서점에서 책을 사다

1.2 클라이언트와 서버

1.3 SSL/TLS상의 HTTP

1.4 SSL/TLS의 역할

1.5 SSL/TLS은 다른 프로토콜도 지킬 수 있다

1.6 암호 스위트

1.7 SSL과 TLS의 차이

1.1 앨리스가 밥 서점에서 책을 사다

- 온라인상으로 전송되는 신용카드 번호는 안전한가?
- https://로 시작되는 페이지에서 입력한 정보는 SSL/TLS를 사용해서 보내지므로 안전

1.2 클라이언트와 서버

- 실제 통신은 앨리스가 사용하고 있는 웹 브라우저와 밥 서점의 웹 서버 사이의 통신
- HTTP 프로토콜 활용
  - HTTP **클라이언트:** 웹 브라우저
  - HTTP **서버:** 웹 서버
  - **요청 (request)**, 요구에 대한 응답을 **응답 (response)** 으로 통신
  - 통신이 평문으로 이루어질 경우 도청의 위험성

앨리스의 웹 브라우저(클라이언트)와 밥 서점의 웹 서버(서버)가 HTTP로 통신

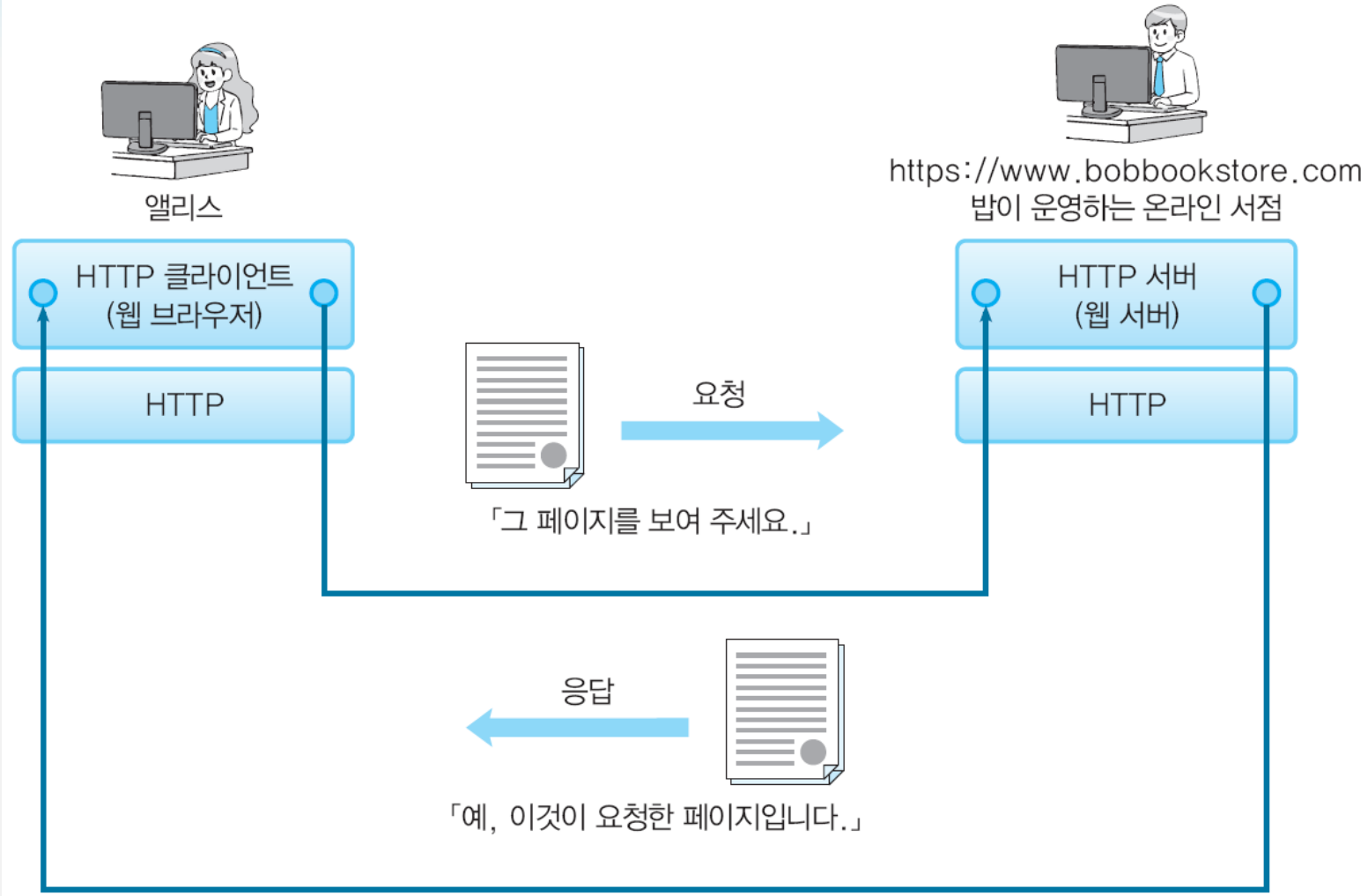


그림 15-1 • 앨리스의 웹 브라우저(클라이언트)와 밥 서점의 웹 서버(서버)가 HTTP로 통신을 수행한다

SSL/TLS를 사용하지 않고 신용카드 번호를 보냈을 경우

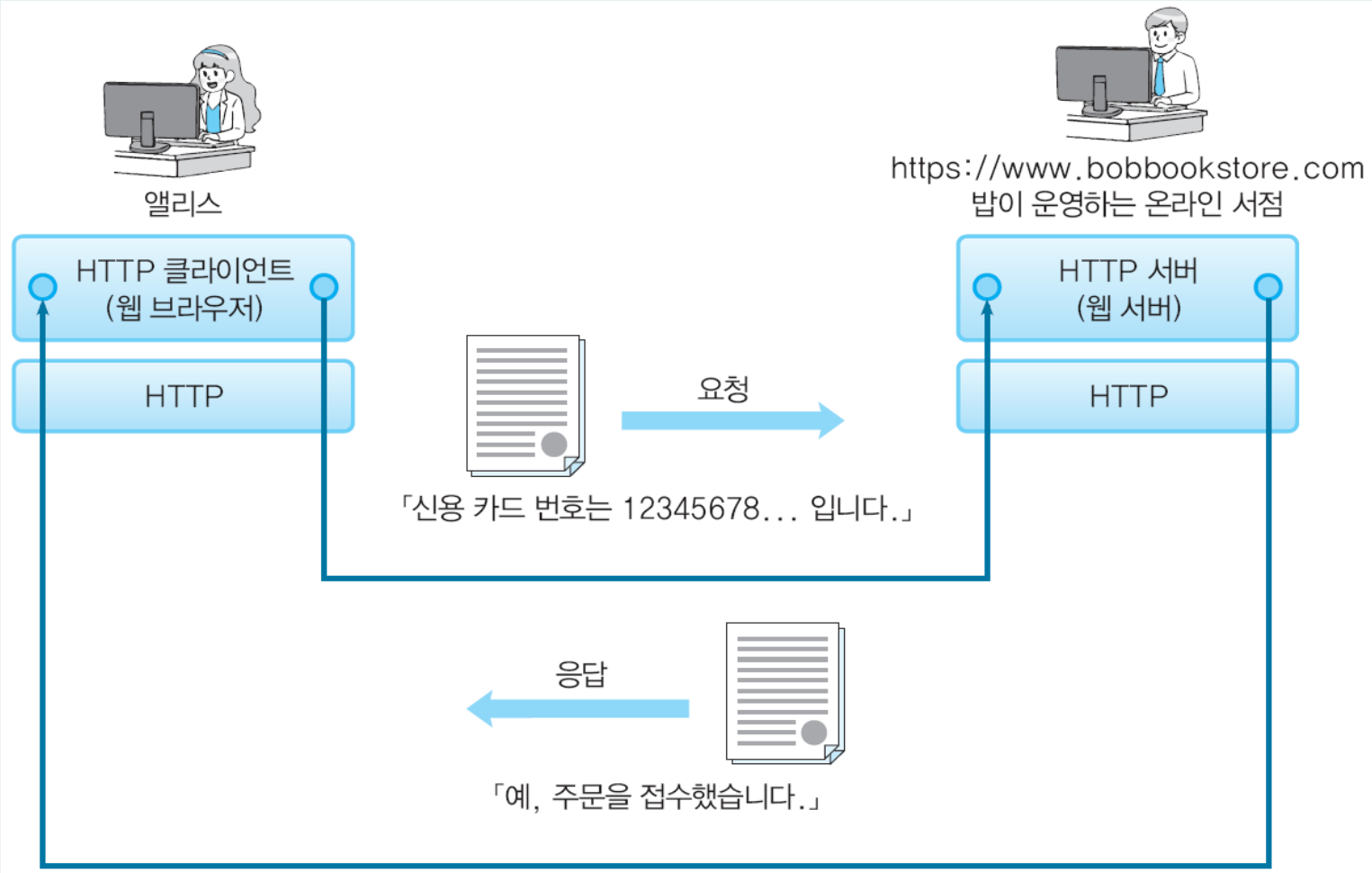


그림 15-2 • SSL/TLS를 사용하지 않고 신용카드 번호를 보냈을 경우

1.3 SSL/TLS상의 HTTP

- 통신 내용을 암호화해 주는 프로토콜로서 **SSL**(Secure Socket Layer) 혹은 **TLS**(Transport Layer Security)를 이용
- 프로토콜의 이중 구조
  - SSL/TLS 상에 HTTP를 올린다
  - HTTP의 통신(요청과 응답)은 암호화되어 도청을 방지

HTTP를 SSL/TLS 상에 올려서 요청과 응답을 암호화

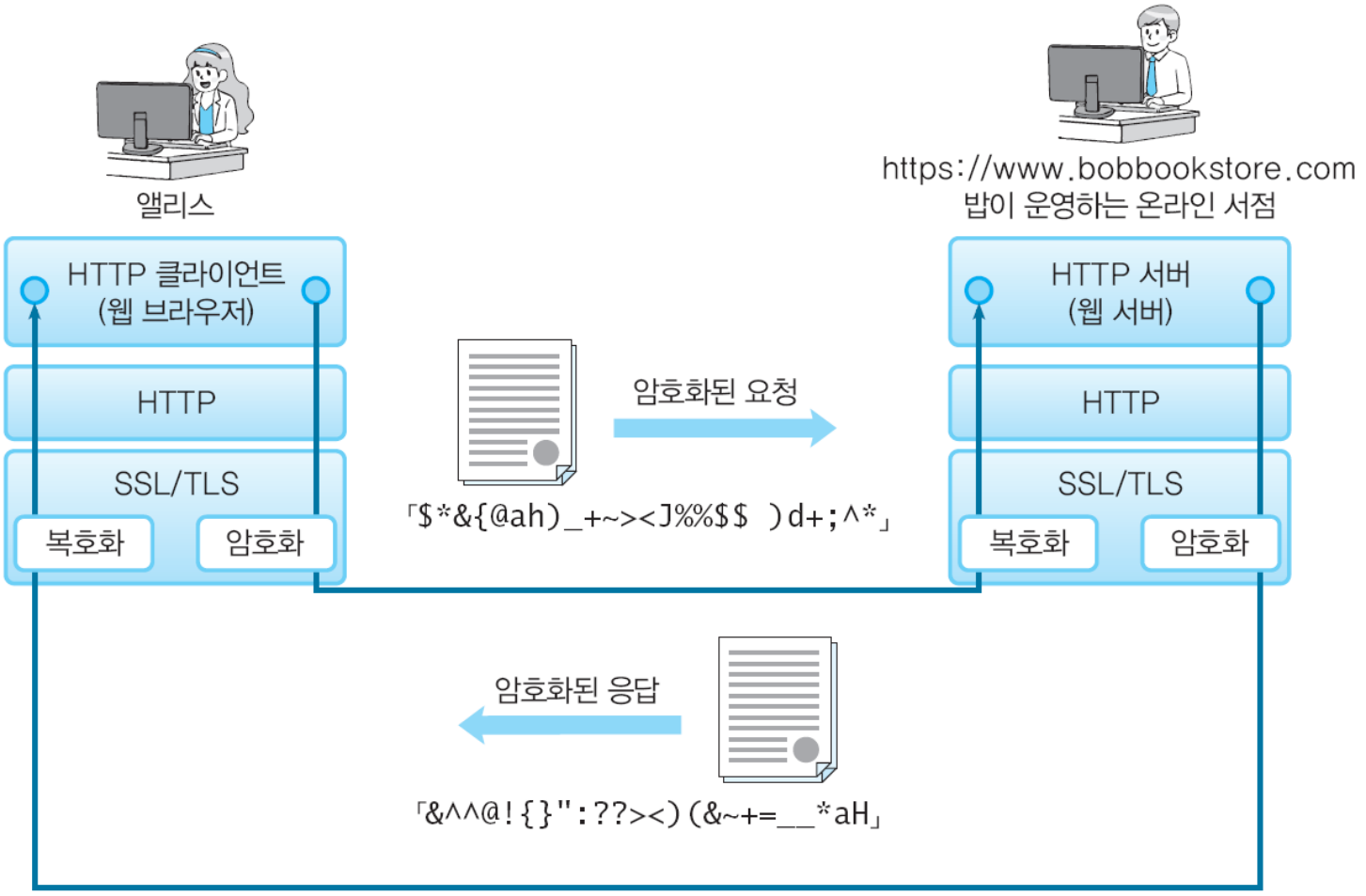


그림 15-3 • HTTP를 SSL/TLS 상에 올려서 요청과 응답을 암호화한다

1.4 SSL/TLS의 역할

- 신용카드 번호 보내기
  1) 기밀성:
     - 앨리스가 송신하는 신용카드 번호와 주소를「도청」당하는 일 없이 밥 서점에 보내고 싶다.
  2) 무결성:
     - 앨리스가 송신하는 신용카드 번호와 주소를「조작」당하는 일 없이 밥 서점에 보내고 싶다.
  3) 인증:
     - 통신 상대의 웹 서버가「진짜 밥 서점」이라는 것을 확인하고 싶다.

해결책

- 기밀성: 대칭 암호
  - 대칭키 생성: 의사난수 생성기
  - 대칭 암호 키 공유: 공개 키 암호, Diffie-Hellman 키 교환
- 데이터 인증: **메시지 인증 코드**
  - 메시지 인증 코드 생성: **일방향 해시 함수**
- 상대 인증: 공개 키에 **디지털 서명**을 붙인 인증서

1.5 SSL/TLS은 타 프로토콜도 지킬 수 있다

- SSL/TLS 상에 올릴 수 있는 다른 프로토콜
  - SMTP(Simple Mail Transfer Protocol): 메일 전송
  - POP3(Post Office Protocol):메일 수신

HTTP, SMTP, POP3을 SSL/TLS 상에 올린다

그림 15-4 • HTTP, SMTP, POP3을 SSL/TLS 상에 올린다

1.6 암호 스위트

- SSL/TLS에서 사용하는 대칭 암호, 공개 키 암호, 디지털 서명, 일방향 해시 함수 등은 부품과 같이 교환가능
- 암호 기술의「추천 세트」가 SSL/TLS로 규정
- 이 추천 세트를 **암호 스위트**(cipher suite)

1.7 SSL과 TLS의 차이

- **SSL**(Secure Socket Layer)
  - 1994년 Netscape사가 제작
  - 웹 브라우저 Netscape Navigator에 내장
  - 많은 웹 브라우저에서 사용되어 사실상의 업계의 표준
  - 1995년에 Version 3.0 로 업그레이드

TLS

- **TLS**(Transport Layer Security)
  - SSL 3.0을 기초로 해서 IETF가 만든 프로토콜
  - 1999년에 RFC2246으로서 발표된 TLS 1.0은 SSL 3.1
  - 2006년에는 TLS 1.1이 RFC 4346으로 발표
  - CBS어택이라 불리는 공격자에 대한 대책이 마련
  - TLS 1.1에는 대칭암호 알고리즘으로서 AES가 추가
  - TLS 1.2(RFC 5426)에는 인증 암호로 GCM (Galois /Counter Mode), CCM(Counter with CBC-MAC), HMAC-SHA 256 추가

Section 02
SSL/TLS를 사용한 통신

2.1 계층화된 프로토콜

2.2 TLS 레코드 프로토콜

2.3 핸드쉐이크 프로토콜

2.4 암호 사양 변경 프로토콜

2.5 경고 프로토콜

2.6 애플리케이션 데이터 프로토콜

2.7 마스터 비밀

2.8 TLS에서 사용되고 있는 암호 기술의 정리

2.1 계층화된 프로토콜

- **TLS 프로토콜:** 2개의 프로토콜로 구성
  - TLS 레코드 프로토콜
    - 암호화 처리
  - TLS 핸드쉐이크 프로토콜
    - 암호화 이외의 다양한 처리
    - 구성
      - 핸드쉐이크 프로토콜
      - 암호사양변경 프로토콜
      - 경고 프로토콜
      - 애플리케이션 데이터 프로토콜

TLS 프로토콜 계층

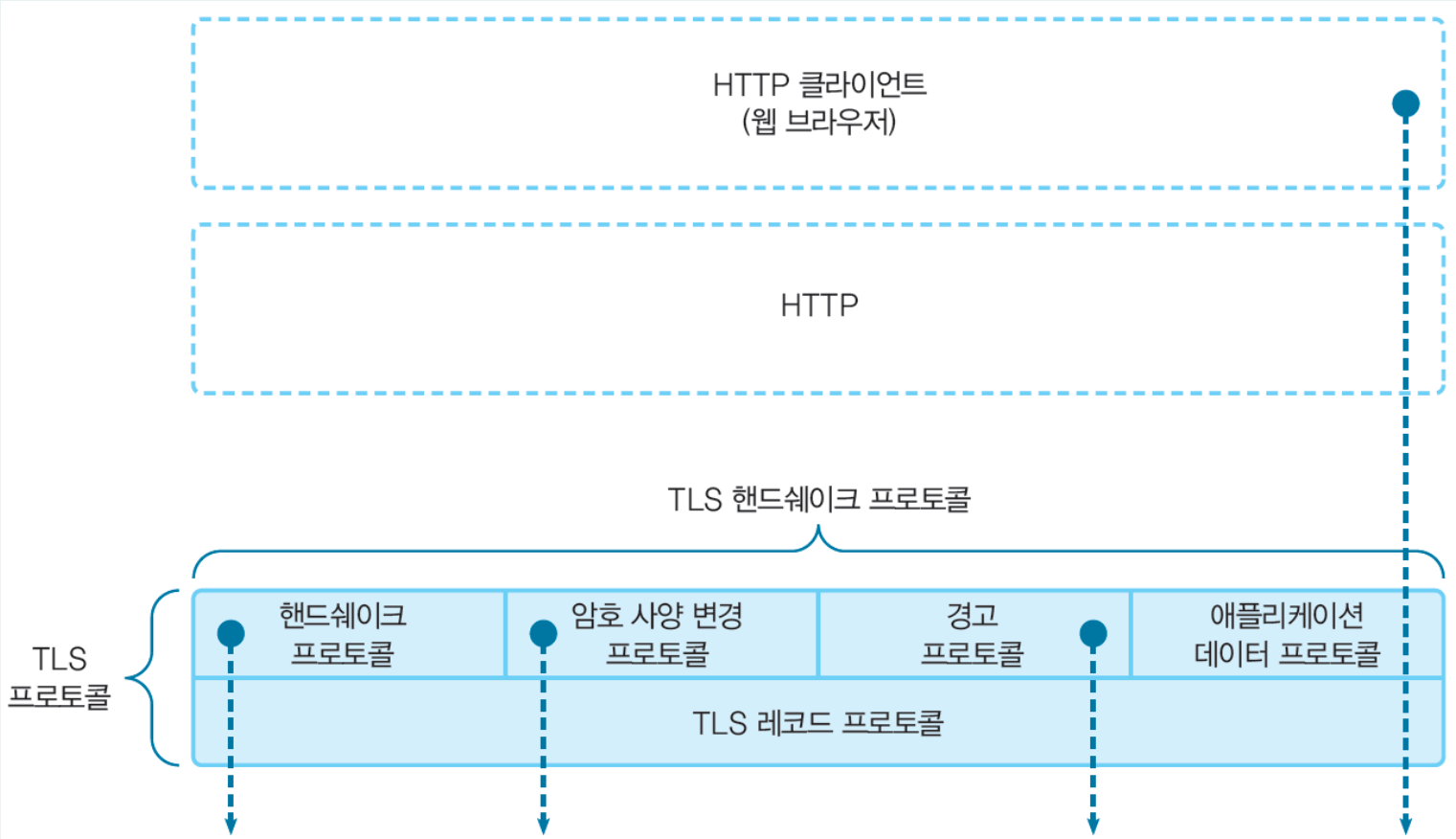


그림 15-5 • TLS 프로토콜 계층

TLS 레코드 프로토콜

- TLS 핸드쉐이크 프로토콜 밑에 위치
- 대칭 암호를 사용해서 메시지를 암호화하고 통신하는 부분
- 대칭 암호와 메시지 인증 코드를 이용
- 알고리즘과 공유 키는 핸드쉐이크 프로토콜을 사용해서 서버와 클라이언트가 상담하여 결정

TLS 핸드쉐이크 프로토콜

- 다음 4개의 서브 프로토콜로 구성
  - 핸드쉐이크 프로토콜
  - 암호 사양 변경 프로토콜
  - 경고 프로토콜
  - 애플리케이션 데이터 프로토콜

핸드쉐이크 프로토콜

- TLS 핸드쉐이크 프로토콜의 일부로 클라이언트와 서버가 암호 통신에 사용할 알고리즘과 공유 키를 결정
- 인증서를 이용한 인증
- 4개의 서브 프로토콜 중 가장 복잡

암호 사양 변경 프로토콜

- 암호 방법을 변경하는 신호를 통신 상대에게 전달

경고 프로토콜

- 뭔가 에러가 발생했다는 것을 통신 상대에게 전달

애플리케이션 데이터 프로토콜

- TLS 상에 올라 있는 애플리케이션의 데이터를 통신 상대에게 전달

TLS 프로토콜에 대한 설명

- TLS 레코드 프로토콜
- TLS 핸드쉐이크 프로토콜
  - 핸드쉐이크 프로토콜
  - 암호사양변경 프로토콜
  - 경고 프로토콜
  - 애플리케이션 데이터 프로토콜

2.2 TLS 레코드 프로토콜

- 단 한 개의 TLS 레코드 프로토콜로 구성
- 메시지의 압축, 암호화, 그리고 데이터의 인증
- 메시지를 여러 개의 짧은 단편(fragment)으로 분할
  - 단편 단위로 압축
    - 압축에 사용하는 알고리즘은 상대와 합의한 것을 이용
- 압축한 단편에 메시지 인증 코드를 부가
  - 재전송 공격을 막기 위해, 단편에 붙여진 번호를 포함해서 메시지 인증 코드를 계산
    - 일방향 해시 함수 알고리즘과 메시지 인증 코드에서 사용하는 키는 상대와 합의한 것을 이용

TLS 레코드 프로토콜

- 압축한 단편과 메시지 인증 코드를 합치고 그것을 대칭 암호로 암호화
  - 암호화에는 CBC 모드를 이용
    - CBC 모드의 초기화 벡터(IV)는 마스터 비밀로부터 생성
    - 대칭 암호 알고리즘과 공통 키는 상대와 합의한 것을 이용
- 데이터 타입, 버전 번호, 압축한 길이로 이루어진 헤더를 부가

데이터 타입이란

- TLS 레코드 프로토콜 상에 올라 있는 4개의 서브 프로토콜 중 어느 것인지를 나타낸 정보
  - 핸드쉐이크 프로토콜
  - 암호 사양 변경 프로토콜
  - 경고 프로토콜
  - 애플리케이션 데이터 프로토콜

TLS 레코드 프로토콜의 처리

그림 15-6 • TLS 레코드 프로토콜의 처리

2.3 TLS 핸드쉐이크 프로토콜

- 다음에 대해서 차례로 설명하기로 한다
  - 핸드쉐이크 프로토콜
  - 암호사양변경 프로토콜
  - 경고 프로토콜
  - 애플리케이션 데이터 프로토콜

핸드쉐이크 프로토콜

- 공유 키를 생성하고 인증서를 교환
  - 공유 키를 생성하는 것은 암호 통신을 행하기 위한 것
  - 인증서를 교환하는 것은 서로 상대를 인증하기 위한 것
- 최초의 핸드쉐이크 프로토콜 단계의 주고받기는 암호화되어 있지 않은 상태로 수행
- 공개 키 암호 또는 Diffie-Hellman 키 교환 활용

TLS 핸드쉐이크 프로토콜

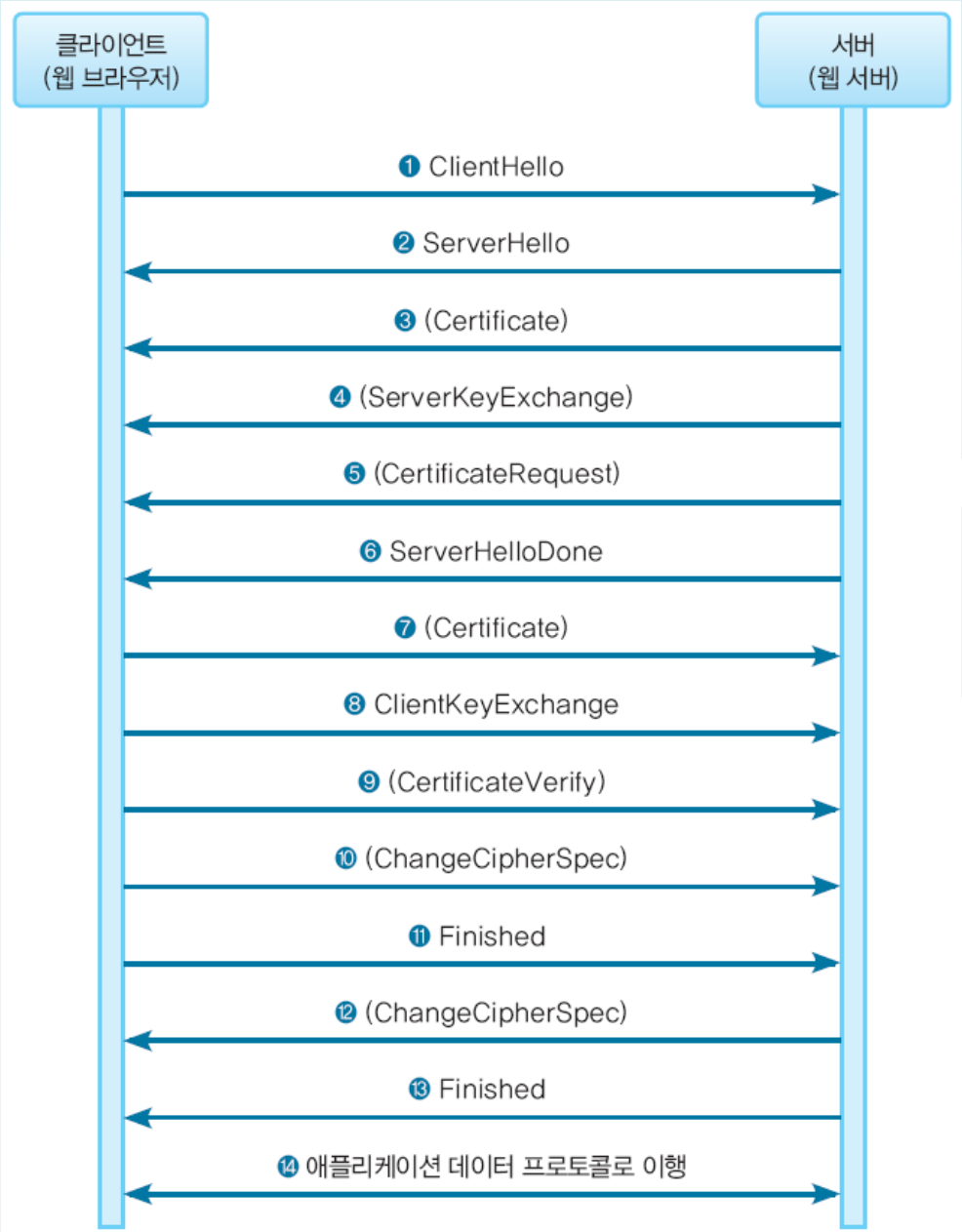


그림 15-7 • TLS 핸드쉐이크 프로토콜

(1) ClientHello(클라이언트→서버)

- 추가 정보
  - 사용할 수 있는 버전 번호
  - 현재 시각
  - 클라이언트 랜덤
  - 세션 ID
  - 사용할 수 있는 암호 스위트 목록
  - 사용할 수 있는 압축 방법 목록

(2) ServerHello(클라이언트←서버)

- 추가 정보
  - 사용하는 버전 번호
  - 현재 시각
  - 서버 랜덤
  - 세션 ID
  - 사용하는 암호 스위트 목록
  - 사용하는 압축 방법 목록

(3) Certificate(클라이언트←서버)

- 추가 정보
  - 인증서 목록

(4) ServerKeyExchange(클라이언트←서버)

- 키 교환을 위한 정보
- Certificate 메시지만으로는 정보가 부족할 때, 클라이언트에게 필요한 정보를 전달
- 보내는 정보는 암호 스위트에 있는 키 교환 알고리즘에 따라 달라짐
  - RSA
  - Diffie-Hellman
- 정보가 불필요한 경우 ServerKeyExchange 메시지는 보내지지 않음

(5) CertificateRequest (클라이언트←서버)

- 서버가 클라이언트에게 인증서를 요구
- 추가 정보
  - 서버가 이해할 수 있는 인증서 타입 목록
  - 서버가 이해할 수 있는 인증기관 이름 목록
- 클라이언트 인증을 하지 않을 경우 CertificateRequest 메시지는 보내지 않음

(6) ServerHelloDone (클라이언트←서버)

- 서버가 보낸 메시지의 끝을 나타냄

(7) Certificate(클라이언트→서버)

- 서버의 CertificateRequest 메시지에 대한 응답
- 클라이언트가 서버에게 자신의 인증서를 전송

(8) ClientKeyExchange(클라이언트→서버)

- (4)의 ServerKeyExchange 메시지에 대응하여 적합한 키 교환 알고리즘을 선정하여 필요한 정보를 전송
- 추가 정보
  - 클라이언트가 선택한 암호 스위트의 키 교환 알고리즘에 따라 달라지는 추가 정보
    - RSA일 경우: **암호화한 사전 마스터 비밀**
    - Diffie-Hellman일 경우: **Diffie-Hellman의 공개 값**

사전 마스터 비밀

- 클라이언트가 만든 난수
- 마스터 비밀(Master Secret)을 만들기 위한 종자
- 서버의 공개 키로 암호화된 뒤에 서버에게 전송
- 사전 마스터 비밀→마스터 비밀→ 생성하는 것

  - 대칭 암호 키
  - 메시지 인증 코드 키
  - 대칭 암호의 CBC 모드에서 이용하는 초기화 벡터(IV)

(9) CertificateVerify(클라이언트→서버)

- 서버로부터 CertificateRequest를 받은 뒤
- 클라이언트는 자신의 인증서 속 공개키와 쌍이 되는 정당한 개인 키를 가지고 있다는 것을 서버에게 주장하는 것
- 클라이언트는 「마스터 비밀」및「핸드쉐이크 프로토콜로 주고받은 메시지」의 해시값을 취해, 그것에 자신의 개인 키로 디지털 서명을 한 다음에 서버에게 전송

(10) ChangeCipherSpec(클라이언트→서버)

- 이 메시지를 이용해서 암호를 변경할 수 있다
- 이 단계를 거치면서 TLS 레코드 프로토콜은 「합의된 암호를 사용한 통신」상태가 된다

(11) Finished(클라이언트→서버)

- 핸드쉐이크 프로토콜 종료를 요청한다
- Finished 메시지는 합의하여 변경된 암호 스위트를 사용해서 전송
- 실제 암호화는 TLS 레코드 프로토콜임
- 서버는 암호문을 복호화하고 Finished 메시지가 잘 복호화 되었는지 점검

(12) ChangeCipherSpec(클라이언트←서버)

- 서버가 클라이언트에게 암호를 교환하자고 메시지를 전송

(13) Finished(클라이언트←서버)

- 서버도 클라이언트에게 Finished 메시지를 전송
- 합의하에 변경된 암호 스위트를 사용해서 전송
- 실제 암호화는 TLS 레코드 프로토콜임

(14) 애플리케이션 데이터 프로토콜로 이행

- 이어서 애플리케이션 데이터 프로토콜과 TLS 레코드 프로토콜을 이용해서 암호 통신을 수행

핸드쉐이크 프로토콜의 목적

- 클라이언트는 서버의 바른 공개 키를 확보하고 서버를 인증
- 서버는 클라이언트의 바른 공개 키를 확보하고 클라이언트를 인증
- 클라이언트와 서버는 암호 통신용의 공유 키를 확보
- 클라이언트와 서버는 메시지 인증 코드용 공유 키를 확보

암호 사양 변경 프로토콜

- **암호 사양 변경 프로토콜**(change cipher spec protocol)
  - TLS 핸드쉐이크 프로토콜의 일부로 암호를 변경하는 신호를 보내기 위한 것
  - 원할 때는 언제나 암호 스위트를 변경할 수 있다
  - 최초에는 암호합의가 없으므로 평문통신을 한다

경고 프로토콜

- TLS의 **경고 프로토콜**(alert protocol)
  - TLS 핸드쉐이크 프로토콜의 일부로 뭔가 에러가 발생했다는 것을 통신 상대에게 전하기 위한 것
  - 핸드쉐이크 도중 다음과 같은 이상이 발생하면 활용
    - 메시지 인증 코드 이상
    - 압축 데이터 확장 이상

애플리케이션 데이터 프로토콜

- **애플리케이션 데이터 프로토콜**
  - TLS 핸드쉐이크 프로토콜의 일부로 애플리케이션의 데이터를 통신 상대와 주고받는 프로토콜
  - TLS 상에 HTTP가 올라 있을 경우에는, HTTP의 요청과 응답은 TLS의 애플리케이션 데이터 프로토콜과 TLS 레코드 프로토콜을 통해서 상호 교환

2.4 마스터 비밀

- 마스터 비밀(Master Secret)
  - TLS의 클라이언트와 서버가 합의한 비밀 값으로 매우 중요
    - 이유: TLS에 의한 암호 통신의 기밀성과 데이터 인증은 모두 이 값에 의존

마스터 비밀 계산

- 다음 정보를 기초로 해서 클라이언트와 서버가 각각 계산
  - 사전 마스터 비밀
    - RSA: 암호화된 사전 마스터 비밀
    - Diffie-Hellman: 공개 값
  - 클라이언트 랜덤
    - 솔트
  - 서버 랜덤
    - 솔트

사전마스터비밀→마스터 비밀

- 사전 마스터 비밀로부터 마스터 비밀을 계산할 때
  - 2개의 일방향 해시 함수(MD5와 SHA-1)를 조합해서 만든 의사난수 생성기를 이용
    - 이유: 2개의 일방향 해시 함수를 조합하는 것은 안전성을 높이기 위해

마스터 비밀의 목적

- 다음 6개 정보 생성
  - 대칭 암호 키(클라이언트→서버)
  - 대칭 암호 키(클라이언트←서버)
  - 메시지 인증 코드 키(클라이언트→서버)
  - 메시지 인증 코드 키(클라이언트←서버)
  - 대칭 암호 CBC 모드에서 이용하는 초기화 벡터(클라이언트→서버)
  - 대칭 암호 CBC 모드에서 이용하는 초기화 벡터(클라이언트←서버)

키 소재의 의존 관계

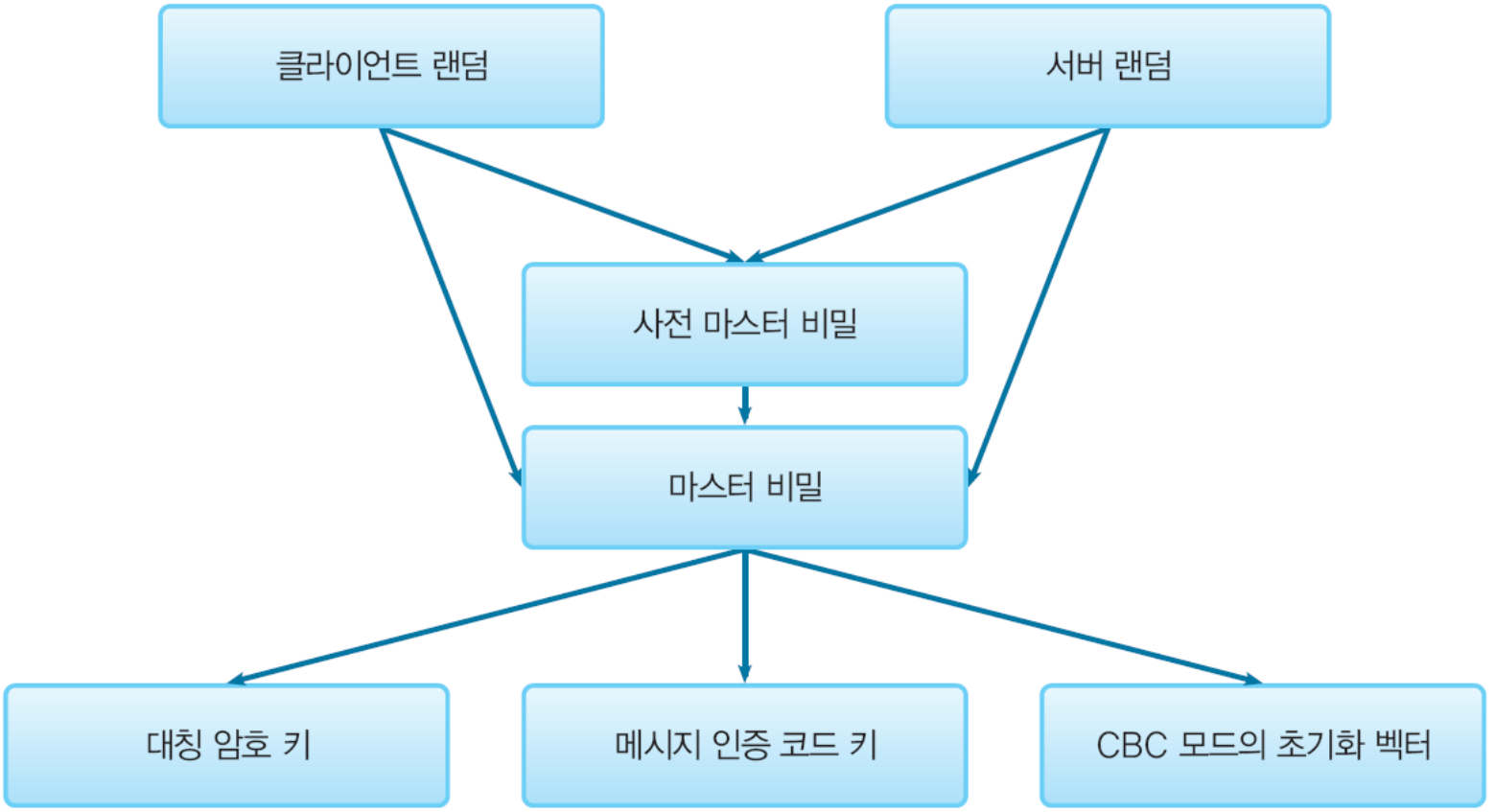


그림 15-8 • 마스터 비밀로 생성된 정보

2.5 TLS에서 사용되고 있는 암호 기술 정리

- TLS에서 사용되고 있는 암호 기술을 프로토콜 별로 정리해서 표로 작성해보자

TLS 핸드쉐이크 프로토콜에서 사용하는 암호 기술

표 15-1 TLS 핸드쉐이크 프로토콜에서 사용한 암호 기술

암호 기술	역할
공개 키 암호	사전 마스터 비밀을 암호화한다.
일방향 해시 함수	의사난수 생성기를 구성한다.
디지털 서명	서버나 클라이언트의 인증서를 검증한다.
의사난수 생성기	사전 마스터 비밀을 만든다. 마스터 비밀로부터 키(암호 매개변수)를 만든다. 초기화 벡터를 만든다.

TLS 레코드 프로토콜에서 사용하는 암호 기술

표 15-2 TLS 레코드 프로토콜에서 사용한 암호 기술

암호 기술	역할
대칭 암호(CBC 모드)	단편의 기밀성을 유지한다.
메시지 인증 코드	단편의 무결성을 유지하고, 인증을 행한다.
인증 암호(AEAD)	단편의 무결성과 기밀성을 유지하고, 인증을 행한다.

Section 03
SSL/TLS에 대한 공격

3.1 개개의 암호 기술에 대한 공격

3.2 OpenSSL의 HeartBleed 취약성

3.3 SSL 3.0의 취약성과 POODLE 공격

3.4 FREAK 공격과 암호 수출 규제

3.5 의사난수 생성기에 대한 공격

3.6 인증서의 틈을 노리는 공격

3.1 개개의 암호 기술에 대한 공격

- SSL/TLS는 특정 암호 기술에 의존하고 있지 않다
- 어느 대칭 암호가 약하다는 것이 판명되면, 앞으로는 그 대칭 암호를 사용하지 않는 암호 스위트를 선택하면 된다
- 마치 기계의 부품이 고장나면 고장 난 부품만을 교환하는 것과 같은 이치

3.2 OpenSSL의 HeartBleed 취약성

- **하트블리드(HeartBleed) 취약성**
  - 암호 통신 라이브러리 OpenSSL의 버그
    - 2014년 Google의 Neel Mehta가 발견
  - 사양상의 취약성이 아니라 OpenSSL이라고 하는 소프트웨어의 취약성
    - TLS의 하트 비트 확장이라고 하는 기능에 요구 데이터 사이즈의 체크가 결여
    - 메모리상의 관계없는 정보가 상대에게 전달
    - 공격자는 이 취약성을 내포한 OpenSSL을 사용하고 있는 서버에 접속하여 서버의 정보를 일정 범위까지 갈취 가능

하트블리드 취약성 대책

- 2014년 당시 SSL/TLS 서버의 17%가 이 취약성
- 대책
  - 하트블리드 취약성 대책이 실행된 버전에 OpenSSL을 갱신
  - 하트 비트 확장을 사용하지 않는 옵션을 부착해 재 컴파일

3.3 SSL3.0 취약성과 POODLE 공격

- **SSL 3.0의 취약성을 언급한 POODLE 공격**
  - 2014 년 Google 의 Bodo Möller, Thai Duong, Krzysztof Kotowicz 가 "This POODLE Bite: Exploiting The SSL 3.0 Fallback "이라는 논문(Security Advisory)으로 공개
  - CBS 모드로 암호화를 행할 때 패딩에 자유도존재
  - 메시지 인증 코드에 의한 무결성 점검을 안 함
  - POODLE 공격은 이 취약성을 이용하여 보안 정보를 빼앗을 수 있음
  - CBC 모드 설명에서 기술한 패딩 오라클 공격임

POODLE 공격에 대한 대책

- 특정 조건하의 통신에서는, 공격자가 TLS를 SSL 3.0으로 다운 그레이드한 통신을 강제할 수 있음
  - 즉, TLS를 사용하고 있다고 해도 SSL 3.0으로 다운 그레이드 당하여 POODLE 공격을 받을 취약성이 있다는 것
- POODLE 공격에 대한 효과적인 대책
  - SSL 3.0을 사용하지 않는 것

3.4 FREAK 공격과 암호 수출 규제

- 2015년 한 연구원이 SSL/TLS의 취약성을 언급한 **FREAK 공격**의 존재를 제시
- FREAK(Factoring RSA Export Keys)(수출용 RSA 키의 소인수분해)의 약자
  - SSL/TLS 서버에 대한 RSA Export Suites라고 불리는 약한 암호 스위트(Suite)를 사용하게 하는 공격
  - 공격에 성공하려면 취약성을 지닌 SSL/TLS 서버와 RSA Export Suites를 받아들이는 웹 브라우저(HTTP 클라이언트) 양쪽이 필요

암호 수출 규제

- 미국은 암호를 무기라고 여겨 1990년대 후반에 규제를 완화할 때까지「암호 소프트웨어」의 수출을 금지
- RSA Export Suites
  - 약하게 만든 암호 집합으로 수출 규제를 받지 않도록 제작
  - 512비트의 RSA나 40비트의 DES를 사용

FREAK 공격이 가능한 이유

- 현실 세계의 많은 웹 서버는 RSA Export Suites를 사용할 수 있게 설정되어 있음
- OpenSSL 등을 선두로 하는 SSL/TLS 라이브러리도 RSA Export Suites를 받아들이고 있음

3.5 의사난수 생성기에 대한 공격

- David Wagner 와 Ian Goldberg의 버그
  - 1995년 캘리포니아대학 버클리 분교의 대학원생
  - Netscape Navigator의 버그 발견
    - 의사난수 생성기 부분
    - 종자가 시각이나 프로세스 번호 등의 예측 가능한 범위에 들어 있었기 때문
    - 키 공간 축소가 문제
- 의사난수 생성기를 보강하여 문제해결

3.6 인증서의 틈을 노리는 공격

- 인증서 검증을 위한 최신판의 CRL(인증서 취소 목록)이 필요
- 웹 브라우저가 최신 CRL을 사용하지 않으면 SSL/TLS를 사용해도 통신이 안전하지 않음

04: SSL/TLS 사용자 유의 사항

4.1 인증서의 의미를 착각하지 않도록

4.2 암호 통신 전의 데이터는 지켜지지 않는다

4.3 암호 통신 후의 데이터는 지켜지지 않는다

4.1 인증서의 의미를 착각하지 않도록

- 인증서의 의미
- 통신 상대는 인증기관이 확인한 서버임에 틀림없음
- 온라인 쇼핑의 통신 상대로서 신뢰할 만한지 아닌지는 판단할 수 없음
- 인증서가 바르더라도 신용카드 번호를 보낼 만큼 반드시 신뢰할 수 있는 것은 아님

4.2 암호 통신 전의 데이터는 지켜지지 않는다

- SSL/TLS가 지키는 것은 통신 중의 데이터
- 통신 전의 데이터는 지켜지고 있지 않다
  - 예: 개인 정보 입력을 어깨 너머로 보기

4.3 암호 통신 후의 데이터는 지켜지지 않는다

- SSL/TLS는 통신 후의 데이터도 지켜 주지 않음
- 송신 도중의 신용카드 번호 같은 민감한 개인정보는 SSL/TLS로 지켜짐
- 도착지에서 민감한 개인정보를 어떻게 다루는 지는 또 다른 문제

Quiz 2 SSL/TLS

다음 문장 중 바른 것에는 O, 잘못된 것에는 X를 하시오.

(1) SSL/TLS를 사용하면 통신의 기밀성을 확보할 수 있다.

(2) SSL/TLS에서는 통신 상대를 인증하기 위해 디지털 서명 기술을 이용하고 있다.

(3) SSL/TLS에서는 공개키 암호 혹은 키 교환 기술을 이용하고 있기 때문에 의사난수 생성기의 품질은 낮아도 된다.

(4) SSL/TLS에서는 공개키가 서버로부터 오기 때문에 클라이언트가 공개키를 하나도 가지고 있지 않아도 서버를 인증할 수 있다.

(5) SSL/TLS를 사용하고 있는 업자는 신뢰할 수 있기 때문에 신용카드 번호를 보내도 된다.