

2026/09/20, 2026 보안 기초 세미나



암호학과 네트워크 보안

-17장 전송층 보안: SSL과 TLS-

민 윤 홍(yunhong@pel.sejong.ac.kr)

세종대학교 프로토콜공학연구실

목 차

- 개요
- SSL 구조
- SSL 프로토콜
- SSL 메시지 형식
- 전송계층 보안(TLS, Transport Layer Security)

목 차

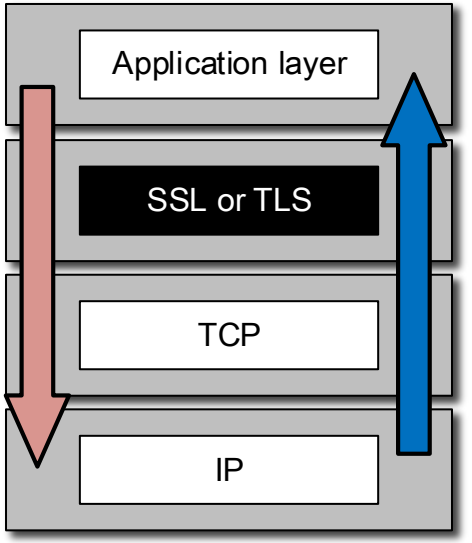
- 개요
- SSL 구조
- SSL 프로토콜
- SSL 메시지 형식
- 전송계층 보안(TLS, Transport Layer Security)

개요

- SSL (Secure Sockets Layer)
  - 정의
    - TCP 위에서 동작하며 전송 계층과 응용 계층 사이에서 상대를 인증하고 데이터를 암호화, 무결성 검증을 제공하는 프로토콜
  - 특징
    - 응용 계층과 전송 계층 사이에 독립적인 프로토콜을 만들어서 동작함
    - Handshake 과정을 인증이 이루어짐
    - SSL 3.0이후 TLS라는 이름으로 재등장함
    - HTTP, SMTP, IMAP등 TCP 기반 프로토콜에 사용 가능함



<그림 1> 특정 홈페이지에 SSL/TLS가 적용되어있는 모습



<그림 2> TCP/IP 모델에서 SSL과 TLS의 위치

개요

- TLS (Transport Layer Security)
  - 정의
    - SSL 3.0을 기반으로 보안 취약점을 개선하기 위해 IETF*가 표준화한 프로토콜
  - 특징
    - SSL에서 사용하던 자체 암호 연산을 표준 방식으로 교체함
    - 메시지에 확장(Extension)을 담당하는 가변 길이 필드를 추가함

*국제 인터넷 표준화 기구(IETF, Internet Engineering Task Force)

[표 1] SSL / TLS 연대기

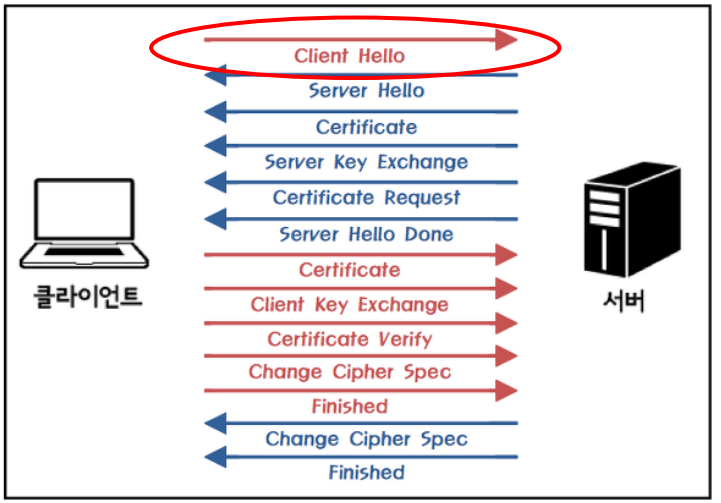
구분	등장 시기	상태	이번 버전에서 개선된 점
SSL 2.0	1995	폐기	-
SSL 3.0	1996	폐기	Finished 메시지를 도입해 변조 탐지
TLS 1.0	1999	폐기	암호 연산 교체, 확장 필드 도입
TLS 1.1	2006	폐기	오류 정보 제공 삭제
TLS 1.2	2008	사용중	SHA-1에서 SHA-256으로 교체
TLS 1.3	2018	사용중	RSA 키 교환 삭제, 인증서 암호화

목 차

- 개요
- SSL 구조
- SSL 프로토콜
- SSL 메시지 형식
- 전송계층 보안(TLS, Transport Layer Security)

SSL 구조

- 개요
  - SSL이 **보안 연결**을 생성하기 위해, 응용 계층과 전송 계층 사이에서 동작하는 구조
- 종류
  - 키 교환 알고리즘
  - 암호화/복호화 알고리즘
  - 해시 알고리즘
  - 암호 그룹
  - 압축 알고리즘
  - 암호학적 매개변수 생성
  - 세션과 연결



ClientHello

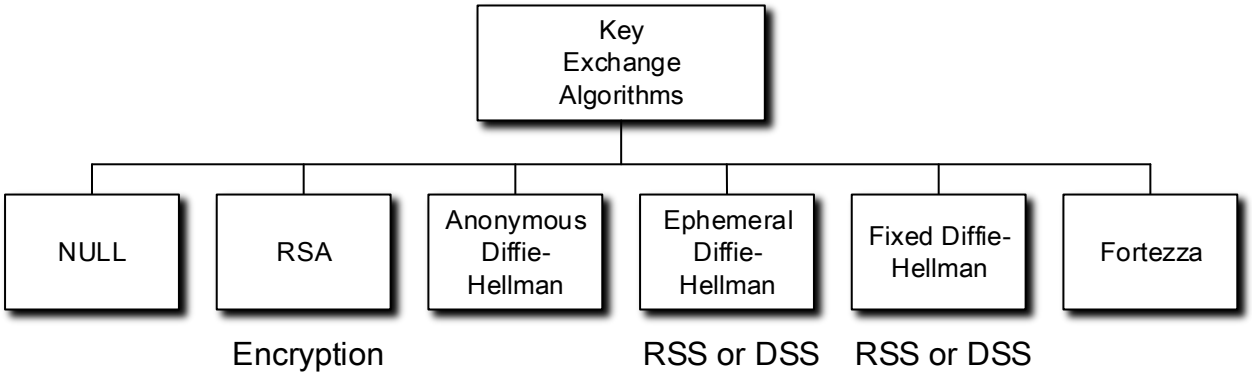
Version
Client random number
~~Session ID~~
Cipher suite
Compression methods

<그림 3> SSL 보안 연결을 위한 설정값의 위치

SSL 구조

- 키 교환 알고리즘
  - 통신하는 클라이언트와 서버 사이의 사전 마스터 비밀*을 설정하기 위해 사용하는 알고리즘
  - 종류
    - NULL
    - RSA
    - 익명 Diffie-Hellman (Anonymous Diffie-Hellman)
    - 임시 Diffie-Hellman (Ephemeral Diffie-Hellman)
    - 고정 Diffie-Hellman(Fixed Diffie-Hellman)
    - Fortezza

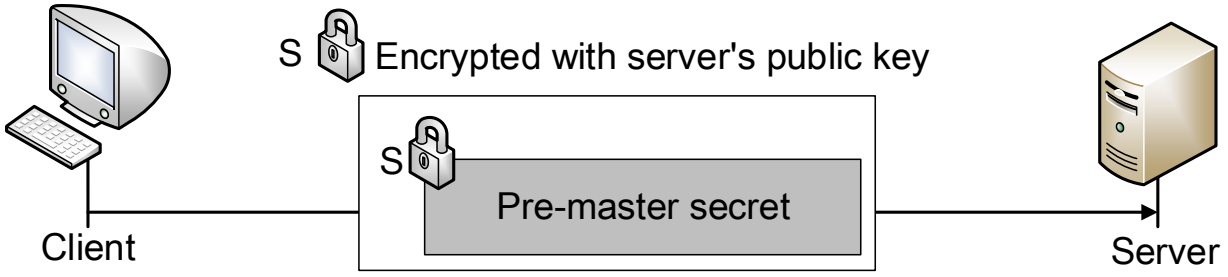
*사전-마스터 비밀(Pre-master secret)
키 교환을 통해 통신을 하고자 하는 양측이 공유하게 되는 값으로,
키를 생성할 때 쓰이는 마스터 비밀(Master secret)을 생성할 때 쓰임



<그림 4> 키 교환 방법

SSL 구조

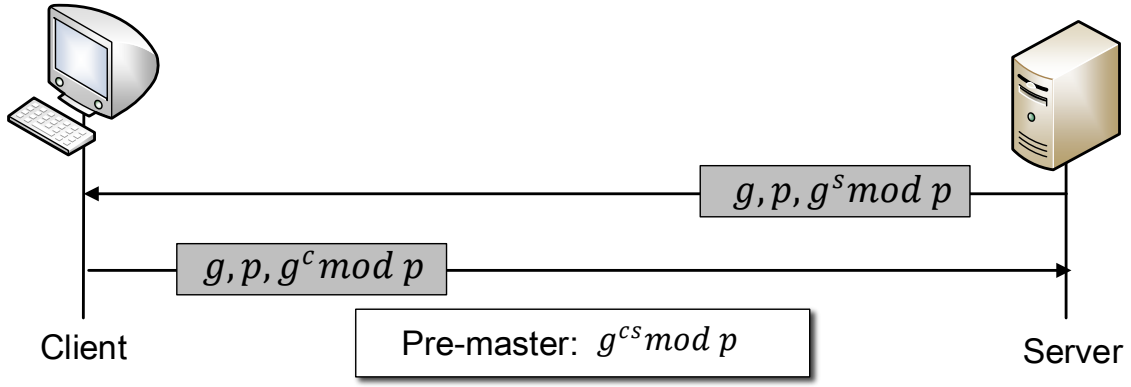
- 키 교환 알고리즘
  - NULL
    - 키 교환 없이 클라이언트와 서버 간의 사전 마스터 비밀이 설정되지 않음
  - RSA
    - 클라이언트가 생성한 48바이트 난수인 사전 마스터 비밀을 서버의 RSA 공개 키로 암호화하여 서버로 보내는 방식



<그림 5> RSA 키 교환 : 서버 공개 키

SSL 구조

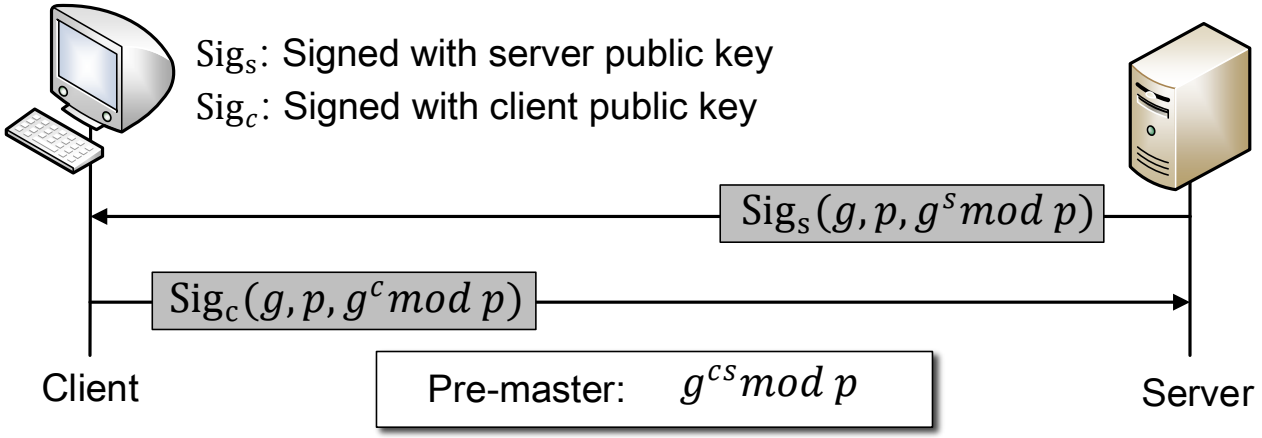
- 키 교환 알고리즘
  - 익명 Diffie-Hellman(Anonymous Diffie-Hellman)
    - Diffie-Hellman 절반-키(half-key)를 평문으로 보내는 방식
    - 특징
      - g, p는 서버가 정한 공개 값, c, s는 각각 클라이언트와 서버가 정한 비밀값, $g^s mod\ p, g^c mod\ q$는 절반-키임
        - $g^s mod\ p, g^c mod\ q$를 합친 $g^{cs} mod\ p$이 키가 되므로 각각은 절반-키로 명시함
      - 인증서와 서명을 보내지 않아 본인이 누군지 알려지지 않음
        - 클라이언트가 g^s의 출처를 알 수 없기 때문에 중간자 공격(Man-in-the-middle attack)에 취약함



<그림 6> 익명 Diffie-Hellman 키 교환

SSL 구조

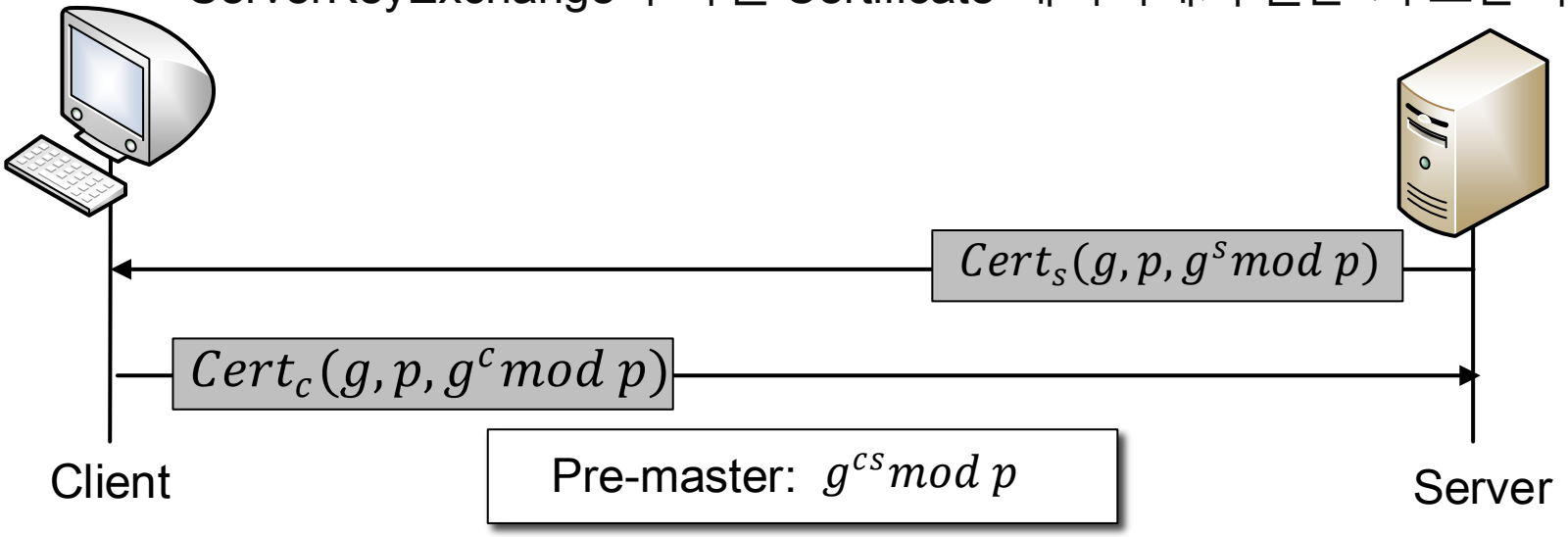
- 키 교환 알고리즘
  - 임시 Diffie-Hellman(Ephemeral Diffie-Hellman)
    - 비밀값인 c, s를 인증 때마다 설정하는 Diffie-Hellman방식
    - 특징
      - 수신자는 송신자의 공개 키를 사용하여 서명을 검증함
      - 검증을 위한 공개 키는 RSA 혹은 DSS (Digital Signature Standard) 디지털 서명 인증서를 사용하여 교환함



<그림 7> 임시 Diffie-Hellman 키 교환

SSL 구조

- 키 교환 알고리즘
  - 고정 Diffie-Hellman(Fixed Diffie-Hellman)
    - 비밀값인 c, s를 고정하여 사용하는 Diffie-Hellman방식
    - 특징
      - 각 개별 절반-키는 인증기관(CA, Certificate Authority)에 의하여 검증된 인증서에 삽입됨
      - 키 교환 메시지가 아닌 인증서 교환만 이루어짐
        - ServerKeyExchange가 아닌 Certificate 메시지에서 절반-키 교환이 이루어짐



<그림 8> Fortezza Crypto Card

SSL 구조

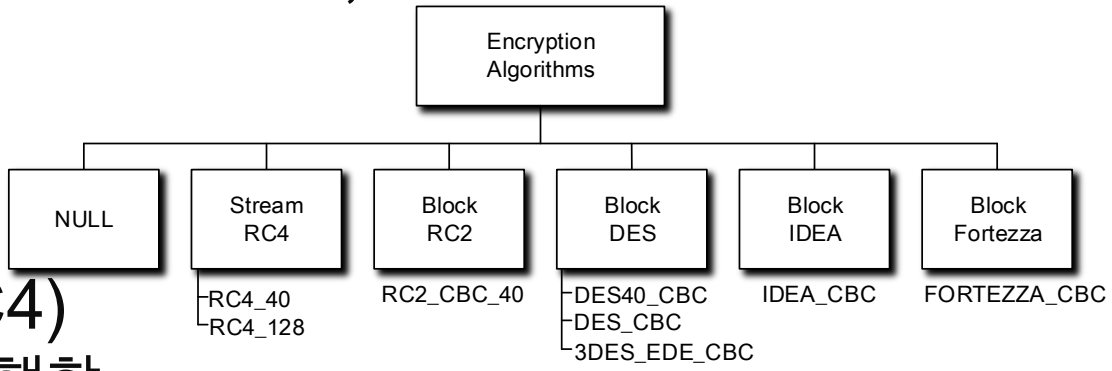
- 키 교환 알고리즘
  - Fortezza
    - 미국 정부, 군사 및 은행용으로 개발된 보안 프로토콜
    - 특징
      - Fortezza Crypto Card를 사용하여 암호화를 진행함
      - 암호화할 때 정부 기관 키를 이용함
      - 알고리즘을 공개하지 않았기에 범용성이 없어 폐지됨



<그림 9> Fortezza Crypto Card

SSL 구조

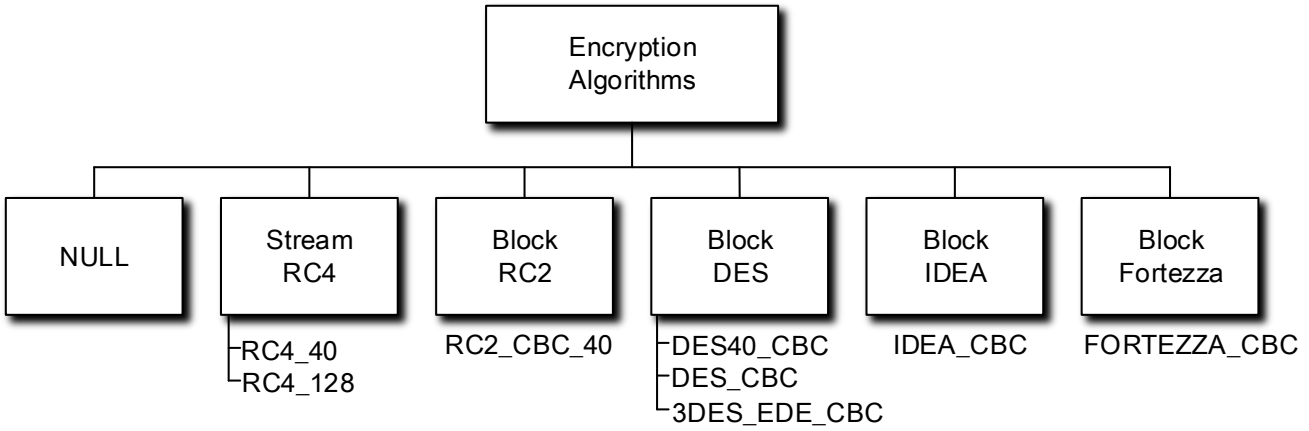
- 암호화/복호화 알고리즘
  - 응용 계층에서 수신된 데이터를 암호화하거나 전송 계층에서 수신된 데이터를 복호화할 때 사용하는 알고리즘
  - 초기 벡터로 Fortezza는 20바이트를, 나머지는 8바이트를 사용함
  - 종류(1/2)
    - NULL
    - 스트림 암호 (Stream RC4)
      - 비트 단위로 암호화를 진행함
    - 블록 RC (Block RC2)
      - 블록 단위로 암호화를 진행함
    - DES (Block DES)
      - 56비트의 키를 이용하는 대칭키 암호화 시스템을 사용함



<그림 10> 암호화, 복호화 알고리즘

SSL 구조

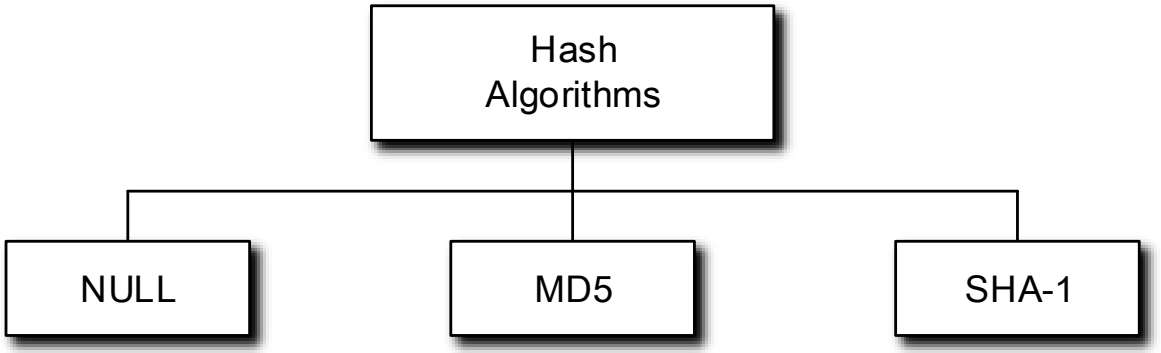
- 암호화/복호화 알고리즘
  - 종류(2/2)
    - IDEA (Block IDEA, International Data Encryption Algorithm)
      - 데이터가 고정된 크기의 블록으로 나누어진 후 각 블록에 동일한 암호화 알고리즘을 반복해서 적용함
    - Fortezza (Block Fortezza)
      - Fortezza Crypto Card를 사용하여 암호화를 진행



<그림 11> 암호화, 복호화 알고리즘

SSL 구조

- 해시 알고리즘
  - SSL에서 메시지 무결성을 위해 사용하는 알고리즘
  - 종류
    - NULL
    - MD5
      - 128 키 MD5 해시 알고리즘을 사용함
    - SHA-1
      - 160비트 키 SHA-1 해시 알고리즘을 사용함



<그림 12> 해시 알고리즘

SSL 구조

- 암호 그룹(Cipher suite)
  - 키 교환, 해시, 암호화 알고리즘의 조합을 정의한 그룹
  - SSL_키 교환 방식_WITH_암호화 알고리즘_해시 알고리즘으로 표기됨

[표 2-1] SSL 암호 그룹 목록

Cipher suite	Key exchange	Encryption	Hash
SSL_NULL_WITH_NULL_NULL	NULL	NULL	NULL
SSL_RSA_WITH_NULL_MD5	RSA	NULL	MD5
SSL_RSA_WITH_NULL_SHA	RSA	NULL	SHA-1
SSL_RSA_WITH_RC4_128_MD5	RSA	RC4	MD5
SSL_RSA_WITH_RC4_128_SHA	RSA	RC4	SHA-1
SSL_RSA_WITH_IDEA_CBC_SHA	RSA	IDEA	SHA-1

SSL 구조

- 암호 그룹
  - 키 교환, 해시, 암호화 알고리즘의 조합을 정의한 그룹
  - SSL_키 교환 방식_WITH_암호화 알고리즘_해시 알고리즘으로 표기됨

[표 2-2] SSL 암호 그룹 목록

Cipher suite	Key exchange	Encryption	Hash
SSL_RSA_WITH_DES_CBC_SHA	RSA	DES	SHA-1
SSL_RSA_WITH_3DES_EDE_CBC_SHA	RSA	3DES	SHA-1
SSL_DH_anon_WITH_RC4_128_MD5	DH_anon	RC4	MD5
SSL_DH_anon_WITH_DES_CBC_SHA	DH_anon	DES	SHA-1
SSL_DH_anon_WITH_3DES_EDE_CBC_SHA	DH_anon	3DES	SHA-1
SSL_DHE_RSA_WITH_DES_CBC_SHA	DHE_RSA	DES	SHA-1
SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA	DHE_RSA	3DES	SHA-1
SSL_DHE_DSS_WITH_DES_CBC_SHA	DHE_DSS	DES	SHA-1

SSL 구조

- 암호 그룹
  - 키 교환, 해시, 암호화 알고리즘의 조합을 정의한 그룹
  - SSL_키 교환 방식_WITH_암호화 알고리즘_해시 알고리즘으로 표기됨

[표 2-3] SSL 암호 그룹 목록

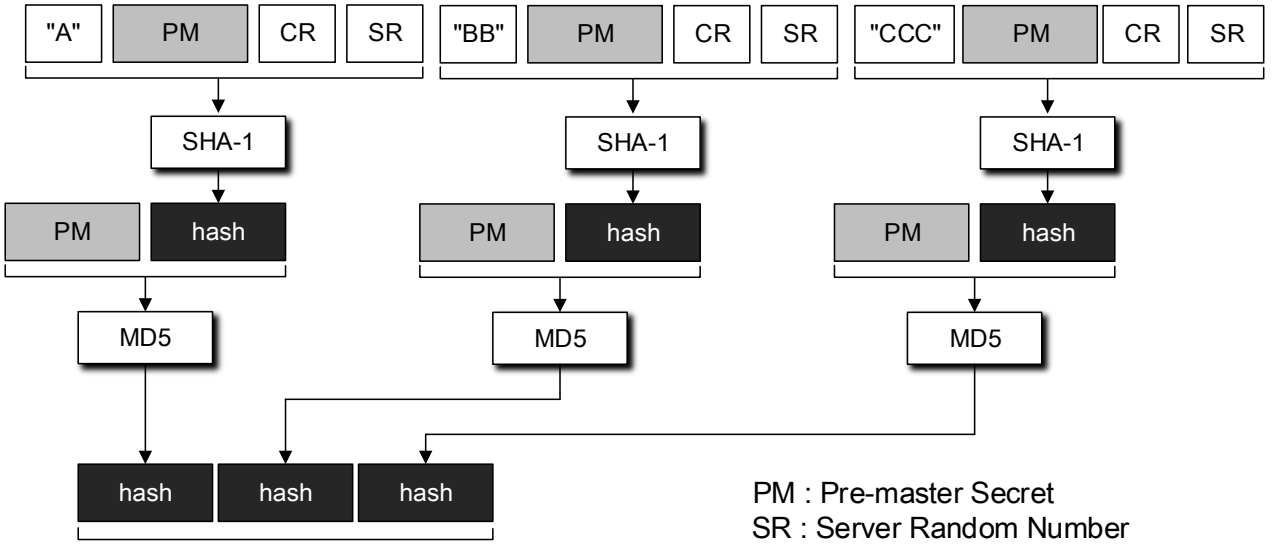
Cipher suite	Key exchange	Encryption	Hash
SSL_DHE_DSS_WITH_3DES_EDE_CBC_SHA	DHE_DSS	3DES	SHA-1
SSL_DH_RSA_WITH_DES_CBC_SHA	DH_RSA	DES	SHA-1
SSL_DH_RSA_WITH_3DES_EDE_CBC_SHA	DH_RSA	3DES	SHA-1
SSL_DH_DSS_WITH_DES_CBC_SHA	DH_DSS	DES	SHA-1
SSL_DH_DSS_WITH_3DES_EDE_CBC_SHA	DH_DSS	3DES	SHA-1
SSL_FORTEZZA_DMS_WITH_NULL_SHA	Fortezza	NULL	SHA-1
SSL_FORTEZZA_DMS_WITH_FORTEZZA_CBC_SHA	Fortezza	Fortezza	SHA-1
SSL_FORTEZZA_DMS_WITH_RC4_128_SHA	Fortezza	RC4	SHA-1

SSL 구조

- 압축 알고리즘
  - 전송량을 줄여 성능을 향상시키기 위한 알고리즘
  - 특징
    - SSL 버전3에서 등장함
    - 특정 알고리즘을 정의하지 않아서 기본 압축 방법은 NULL임
    - 시스템은 원하는 압축 알고리즘을 사용할 수 있음
    - 동일한 문자열이 포함되면 압축률이 올라가는 것을 이용한 CRIME 공격에 취약하여 TLS 1.3에서 제거됨

SSL 구조

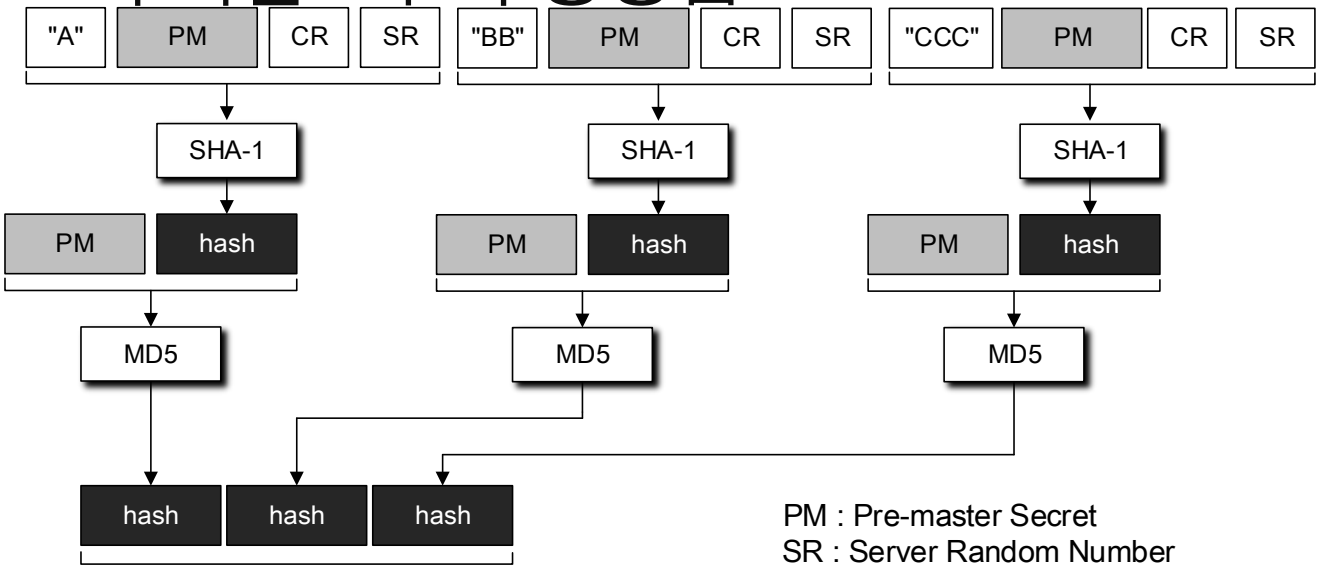
- 암호학적 매개변수
  - 키와 블록 암호화를 위해 필요한 값
  - 특징
    - 메시지의 무결성과 기밀성을 얻기 위해 사용
    - 4개의 암호학적 비밀 키와 2개의 초기 벡터가 필요함
    - 클라이언트와 서버가 각각 메시지 인증 키, 암호화 키, 블록 암호화를 위한 초기 벡터를 필요로 함



<그림 12> 사전-마스터 비밀로부터의 마스터 비밀 계산 방식

SSL 구조

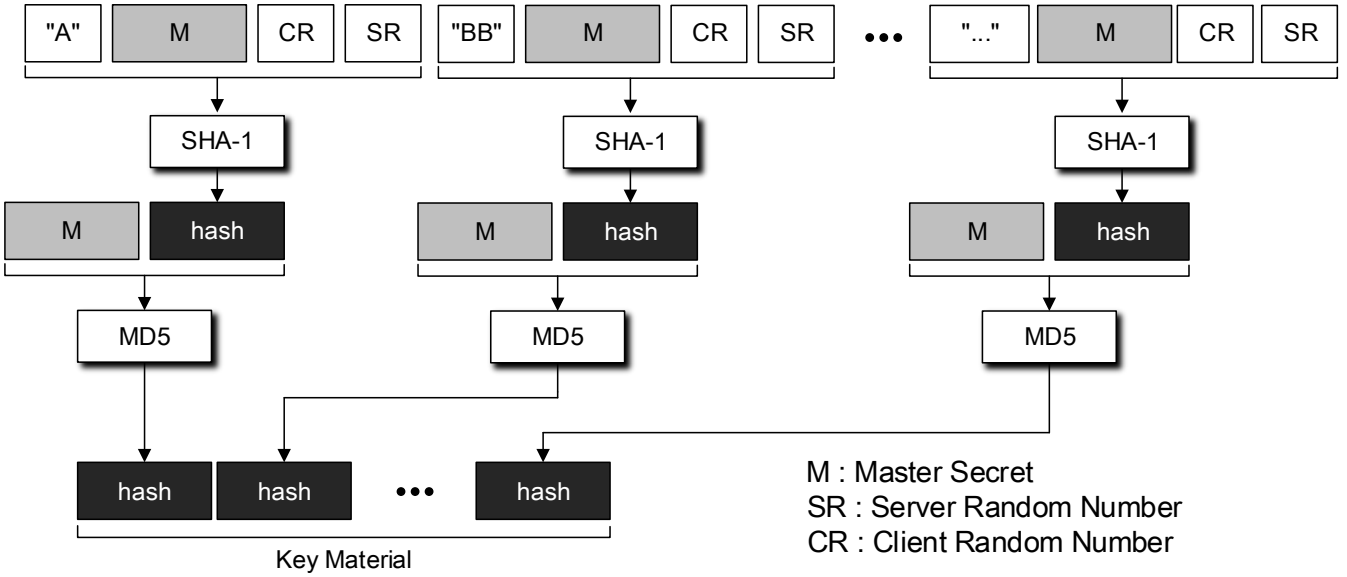
- 암호학적 매개변수
  - 생성 과정 (1/3)
    - 클라이언트와 서버는 각각 하나의 난수를 생성하고 교환함
    - 클라이언트와 서버는 이전에 논의된 키 교환 알고리즘 중 하나를 사용하여 하나의 사전-마스터 비밀을 교환함
    - 48바이트 마스터 비밀은 적용 중인 SHA-1과 MD5 해시 함수에 의해 사전-마스터 비밀로부터 생성됨



<그림 12> 사전-마스터 비밀로부터의 마스터 비밀 계산 방식

SSL 구조

- 암호학적 매개변수
  - 생성 과정 (2/3)
    - 마스터 비밀은 서로 다른 상수를 이용한 사전교섭과 적용 중인 동일 해시 함수의 합으로 가변길이 키 재료를 생성하는데 사용됨
      - 키 재료의 블록 길이는 암호문에서 요구된 키의 크기와 선택된 암호문에 의존함

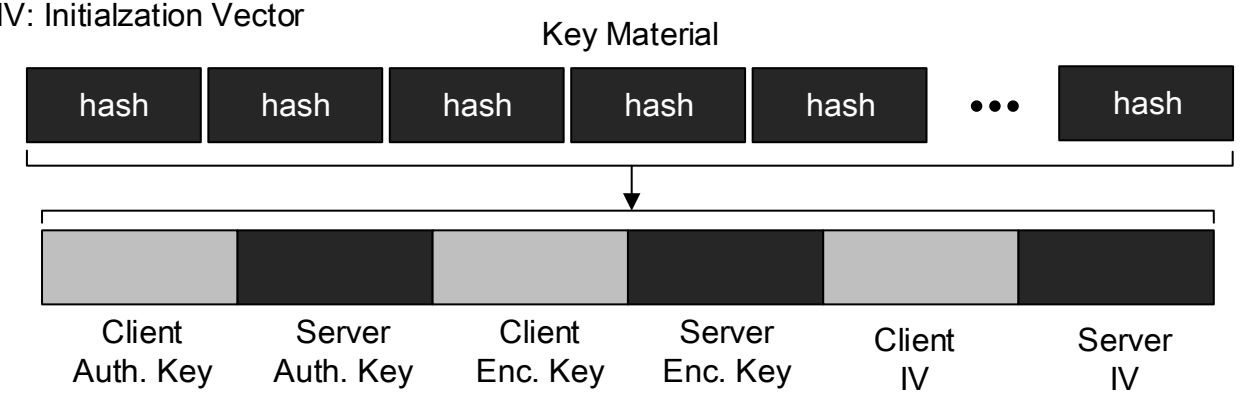


<그림 13> 마스터 비밀로부터 키 재료의 계산 방식

SSL 구조

- 암호학적 매개변수
  - 생성 과정 (3/3)
    - 최종적으로 사용될 6개의 서로 다른 키는 키 재료로부터 추출됨
      - Auth.Key는 MAC를 계산할 때 사용됨
      - Enc.Key는 데이터 암호화를 할 때 사용됨
      - IV는 CBC (Cipher Block Chaining)의 첫 블록에 사용됨
        - CBC는 직전 암호문과 XOR 연산을 하여 암호화하는 방식임

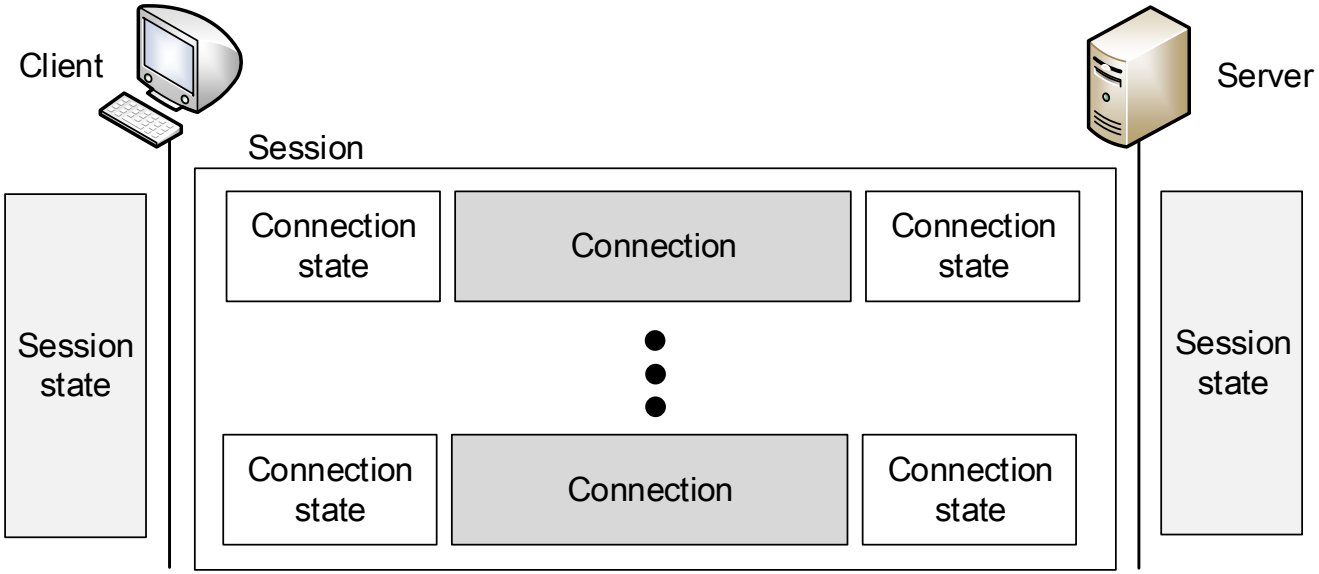
Auth. Key: Authentication Key
Enc. Key: Encryption Key
IV: Initialzation Vector



<그림 14> 키 재료로부터 암호학적 비밀의 추출 과정

SSL 구조

- 세션과 연결
  - 세션 (Session)
    - 서버와 클라이언트가 데이터를 주고 받는 연결 상태
  - 연결 (Connection)
    - 세션 안에서 두 개체가 데이터 교환을 하는 통로



<그림 15> 세션과 연결 개요

SSL 구조

- 세션과 연결
  - 세션
    - 특징
      - 새로운 세션을 설정하기 위해선, 클라이언트와 서버는 협의 절차를 거쳐야 함
      - 세션이 설정된 후, 클라이언트와 서버는 암호 그룹, 메시지 인증 암호화를 위한 마스터 비밀과 같은 공통 정보를 가짐
      - 여러 개의 연결로 구성할 수 있음
      - 세션을 일시 중지하거나 다시 시작할 수 있음

SSL 구조

- 세션과 연결
  - 세션
    - 상태 매개변수
      - 데이터 교환에 필요한 값을 저장하는 매개변수

[표 2] 세션 상태 매개변수

Parameter	Description
Session ID	세션을 식별하는 값으로 서버가 정함
Peer Certificate	상대방의 인증서(X.509 v3)를 저장하는 값
Compression Method	합의된 압축 방식을 저장하는 값
Cipher Suite	합의된 암호 그룹을 저장하는 값
Master Secret	48바이트의 비밀을 저장하는 값
Is resumable	세션 재시작 가능 여부를 나타내는 값

SSL 구조

- 세션과 연결
  - 연결
    - 상태 매개변수
      - 데이터 교환에 필요한 값을 저장하는 매개변수

[표 3] 연결 상태 매개변수

Parameter	Description
Server and client random numbers	서버와 클라이언트가 생성한 난수를 저장하는 값
Server write MAC secret	서버가 보내는 메시지 무결성용 MAC키를 저장하는 값 클라이언트가 검증함
Client write MAC secret	클라이언트가 보내는 메시지 무결성용 MAC키를 저장하는 값 서버가 검증함
Server write secret	서버가 보내는 데이터 암호화 키를 저장하는 값
Client write secret	클라이언트가 보내는 데이터 암호화 키를 저장하는 값
Initialization vectors	CBC 모드에서 블록 암호의 초기값으로 사용되는 값
Sequence numbers	MAC 계산에 사용하는 시퀀스 번호를 저장하는 값 $2^{64} - 1$를 초과하지 않음

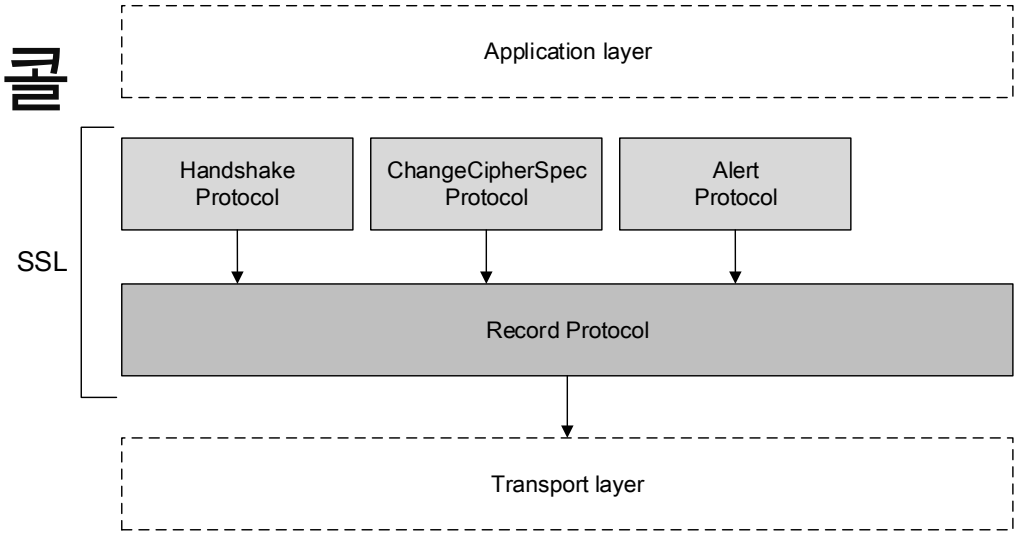
목 차

- 개요
- SSL 구조
- SSL 프로토콜
- SSL 메시지 형식
- 전송계층 보안(TLS, Transport Layer Security)

SSL 프로토콜

- 개요
  - SSL이 응용 계층과 전송 계층 사이에서 암호화된 데이터를 전송하기 위해 사용하는 4개의 통신 규약

- 종류
  - Handshake 프로토콜
  - ChangeCipherSpec 프로토콜
  - Alert 프로토콜
  - Record 프로토콜



<그림 16> 4개의 SSL 프로토콜 구조

SSL 프로토콜

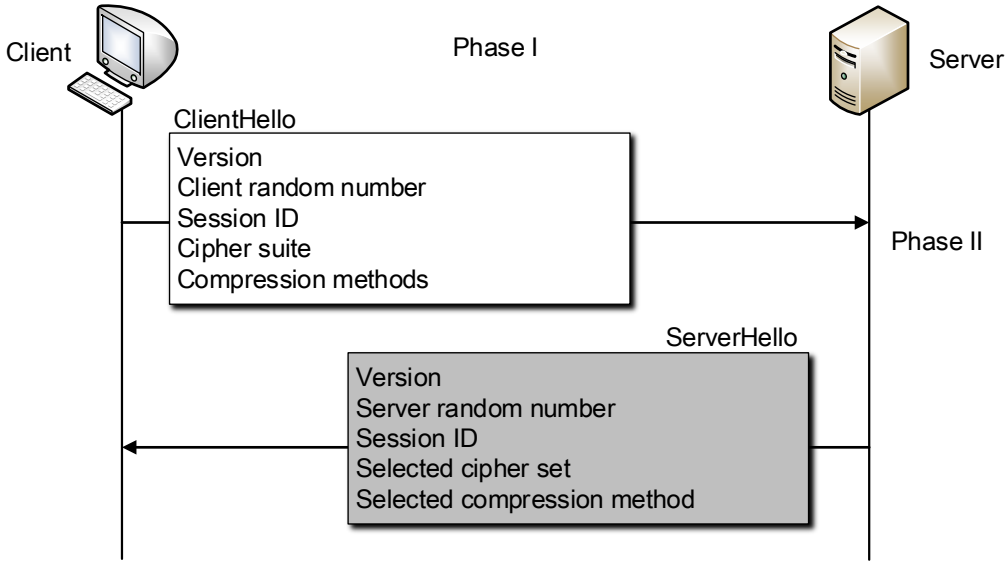
- Handshake 프로토콜
  - 암호 알고리즘 결정, 키 분배, 서버 및 클라이언트 인증을 수행하기 위해 사용되는 프로토콜
  - 동작 과정
    - 보안 기능 설정
    - 서버 키 교환과 인증
    - 클라이언트 키 교환과 인증
    - 종결과 종료



<그림 17> Handshake 프로토콜 과정

SSL 프로토콜

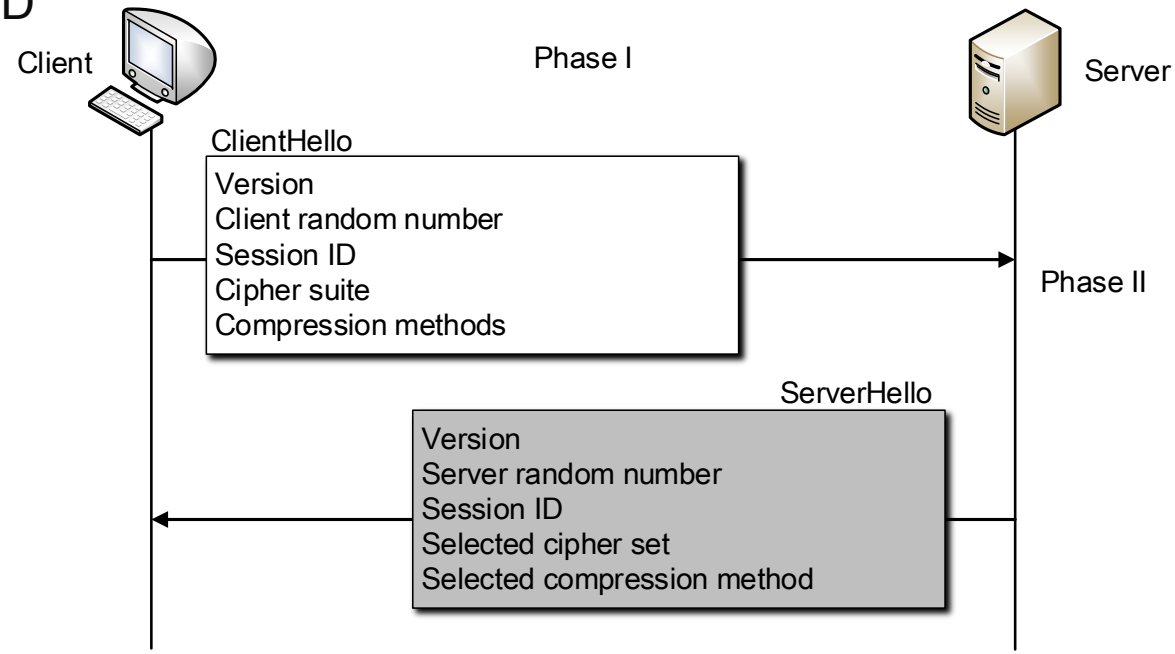
- Handshake 프로토콜
  - 보안 기능 설정
    - 클라이언트와 서버가 암호화 방식과 압축 방법, 사용 SSL 버전, 키 생성을 위한 2개의 난수를 결정하는 과정
    - 메시지 교환 과정 (1/2)
      - 클라이언트는 서버에게 ClientHello 메시지를 보냄
        - 클라이언트를 지원할 수 있는 가장 높은 SSL 버전 번호
        - 클라이언트가 마스터 비밀을 생성하는데 사용될 수 있는 32바이트 난수
        - 세션을 나타내는 세션 ID
        - 클라이언트가 지원할 수 있는 알고리즘 목록인 암호 목록
        - 클라이언트가 지원할 수 있는 압축 방법 목록



<그림 18> Handshake 프로토콜의 단계 1

SSL 프로토콜

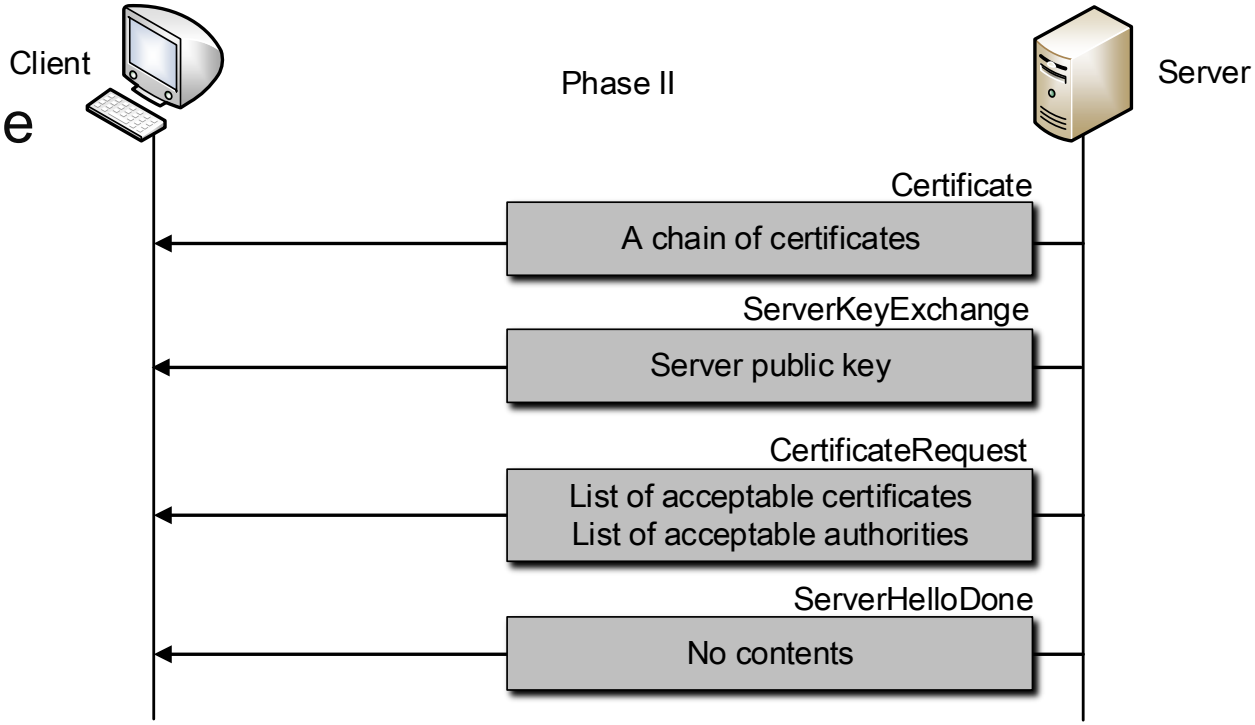
- Handshake 프로토콜
  - 보안 기능 설정
    - 메시지 교환 과정 (2/2)
      - 서버는 클라이언트에게 ServerHello 메시지로 응답을 보냄
        - 클라이언트를 지원할 수 있는 가장 높은 SSL 버전 번호
        - 서버가 마스터 비밀을 생성하는데 사용될 수 있는 32바이트 난수
        - 세션을 나타내는 세션 ID
        - 클라이언트 목록에서 선택된 암호 집합
        - 클라이언트 목록에서 선택된 압축 방법



<그림 18> Handshake 프로토콜의 단계 1

SSL 프로토콜

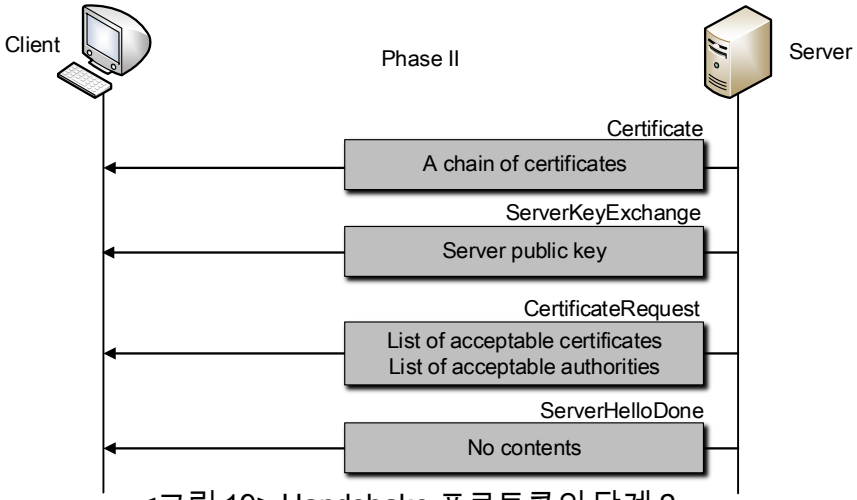
- Handshake 프로토콜
  - 서버 키 교환과 인증
    - 서버가 자신의 인증서와 공개 키를 클라이언트에 보내는 과정
    - 완료 시, 서버는 클라이언트에 대해 인증됨
    - 메시지 교환 과정
      - Certificate
      - ServerKeyExchange
      - CertificateRequest
      - ServerHelloDone
    - 키 교환 알고리즘
      - RSA
      - Anonymous DH
      - Ephemeral DH
      - Fixed DH



<그림 19> Handshake 프로토콜의 단계 2

SSL 프로토콜

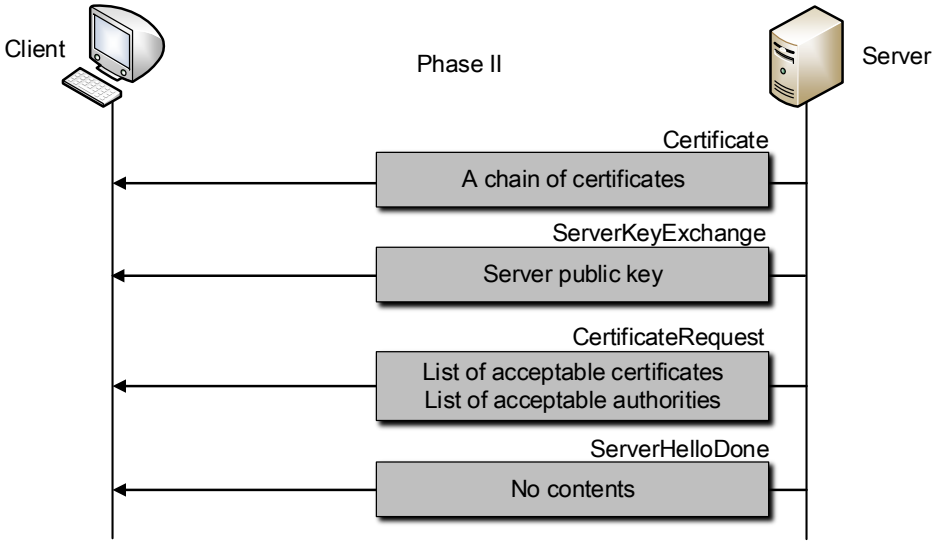
- Handshake 프로토콜
  - 서버 키 교환과 인증
    - 메시지 교환 과정 (1/2)
      - 서버가 자신을 인증하기 위해 Certificate 메시지를 보냄
        - 클라이언트가 서버에게 인증서를 요구했을 때 보내는 메시지임
        - X.509 형태의 인증서 목록 포함
        - 키 교환 알고리즘이 익명 Diffie-Helman이면 인증서가 필요하지 않음
      - 서버의 공개키가 SSL 인증서 내부 없는 경우 ServerKeyExchange 메시지를 보냄
        - 사전-마스터 비밀에 정보가 들어있는 메시지임



<그림 19> Handshake 프로토콜의 단계 2

SSL 프로토콜

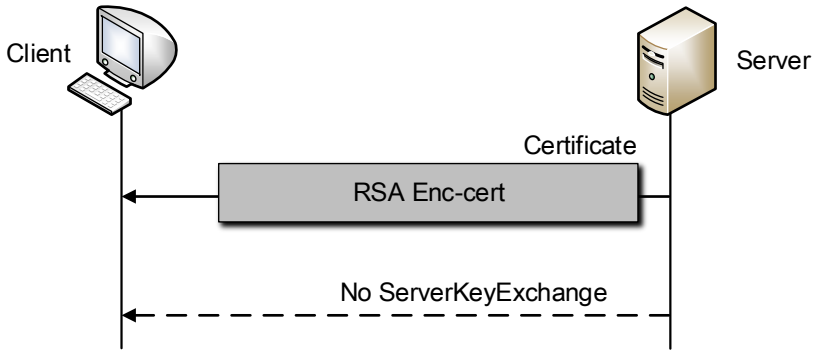
- Handshake 프로토콜
  - 서버 키 교환과 인증
    - 메시지 교환 과정 (2/2)
      - 서버가 클라이언트의 인증을 요구하는 경우 클라이언트에게 CertificateRequest 메시지를 보냄
        - 익명 Diffie-Hellman이 사용되면 클라이언트로부터 인증서를 요구할 수 없음
      - 서버 키 교환과 인증이 끝난 후 키 교환과 인증을 시작할 때 ServerHelloDone 메시지를 보냄



<그림 19> Handshake 프로토콜의 단계 2

SSL 프로토콜

- Handshake 프로토콜
  - 서버 키 교환과 인증
    - 키 교환 알고리즘 (1/3)
      - 사전-마스터 비밀로 사용하는 방식의 종류에 따라 Certificate, ServerKeyExchange로 보내는 메시지가 달라짐
      - RSA
        - 서버가 Certificate 메시지에 RSA 암호화/복호화 공개 키 인증서를 클라이언트에게 보냄
        - 사전 마스터 비밀이 생성되었기 때문에 ServerKeyExchange 메시지는 비어있음
        - 서버가 사전 마스터 비밀을 수신하면 개인 키로 복호화함

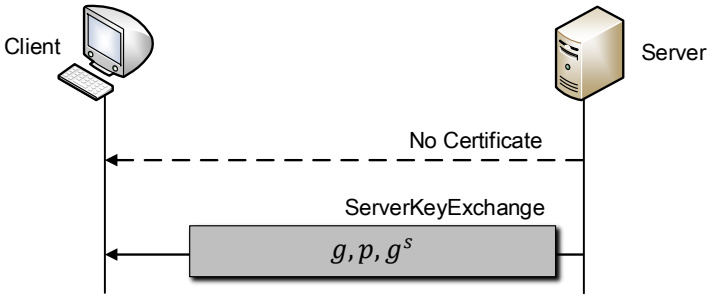


a . RSA

<그림 20-1> 단계 2에서의 키 교환 알고리즘 - RSA

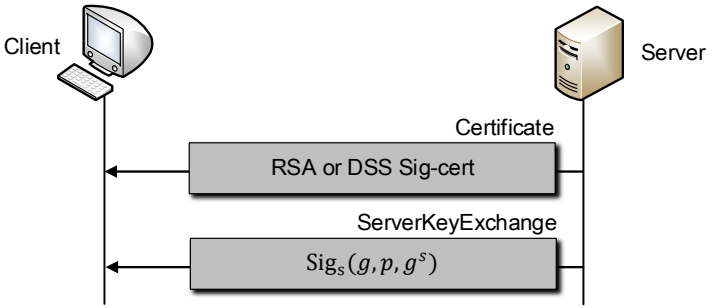
SSL 프로토콜

- Handshake 프로토콜
  - 서버 키 교환과 인증
    - 키 교환 알고리즘 (2/3)
      - 익명 Diffie-Hellman(Anonymous Diffie-Hellman)
        - 익명의 개체는 인증서를 가지지 않기 때문에 Certificate 메시지는 없음
        - ServerKeyExchange 메시지에서 서버는 Diffie-Hellman 매개변수와 절반 –키를 송신함
        - 서버는 익명 Diffie-Hellman으로 인증될 수 없음
      - 임시 Diffie-Hellman(Ephmeral Diffie-Hellman)
        - 서버가 Certificate 메시지로 RSA 혹은 DSS 디지털 서명 인증서를 보냄
        - 서버가 ServerKeyExchange 메시지로 Diffie-Hellman 매개변수와 개인 키에 의해 서명된 절반-키를 보냄



b. Anonymous DH

<그림 20-2> 단계 2에서의 키 교환 알고리즘 – 익명 DH

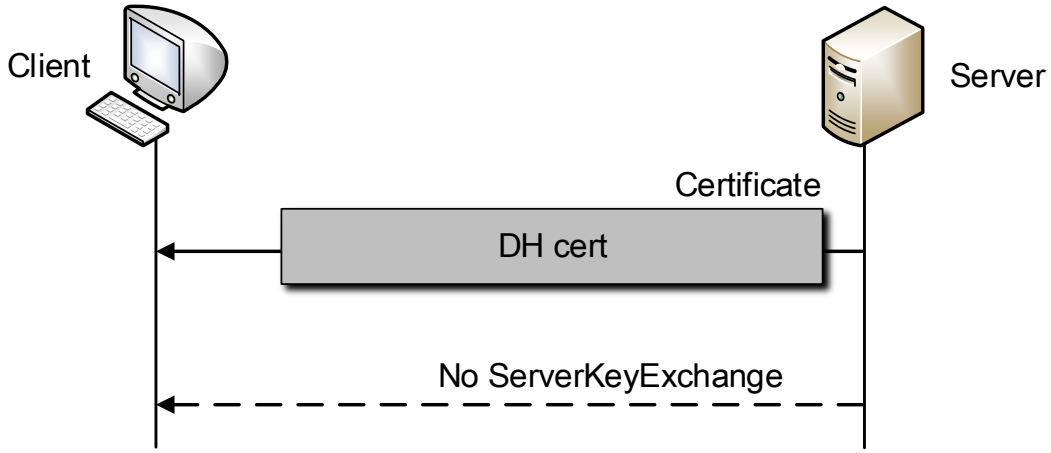


c . Ephmeral DH

<그림 20-3> 단계 2에서의 키 교환 알고리즘 – 임시 DH

SSL 프로토콜

- Handshake 프로토콜
  - 서버 키 교환과 인증
    - 키 교환 알고리즘 (3/3)
      - Fixed DH
        - 서버는 Certificate 메시지로 등록된 DH 절반-키를 포함한 RSA 혹은 DSS 디지털 서명 인증서를 보냄
        - 인증서는 CA의 개인 키로서 서명되고 CA의 공개키를 사용하여 클라이언트에 의해 검증됨
        - ServerKeyExchange 메시지는 비어 있음

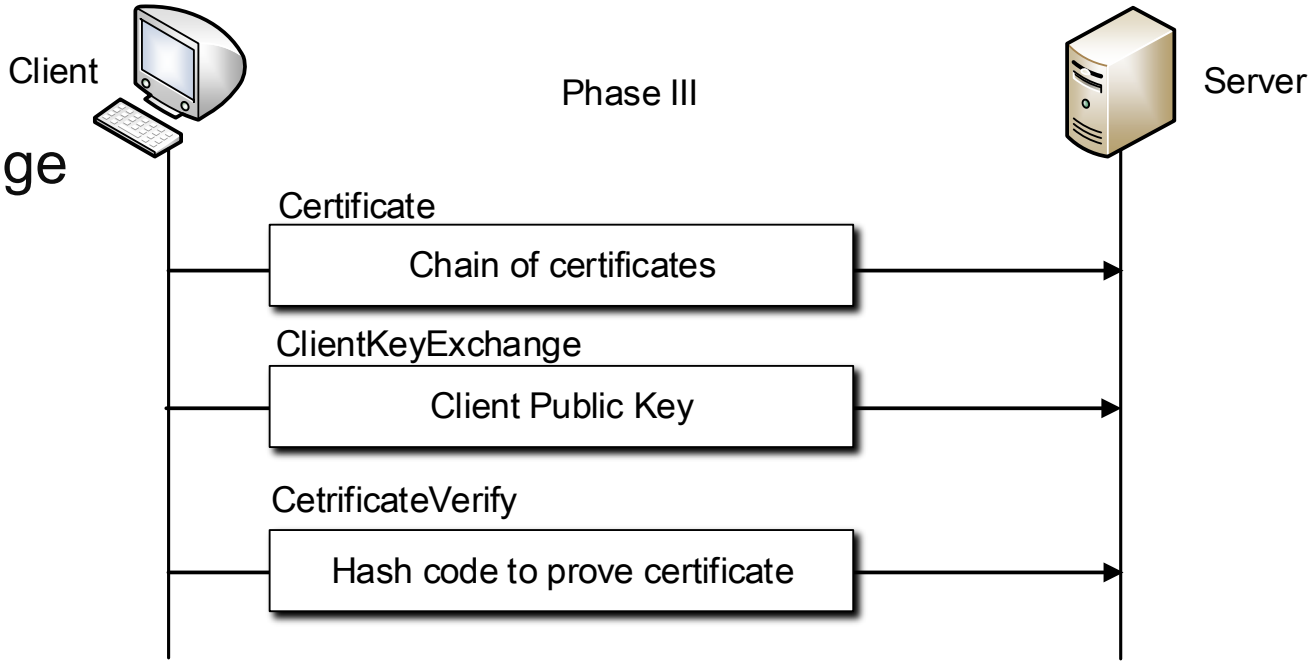


d . Fixed DH

<그림 20-4> 단계 2에서의 키 교환 알고리즘 – 고정 DH

SSL 프로토콜

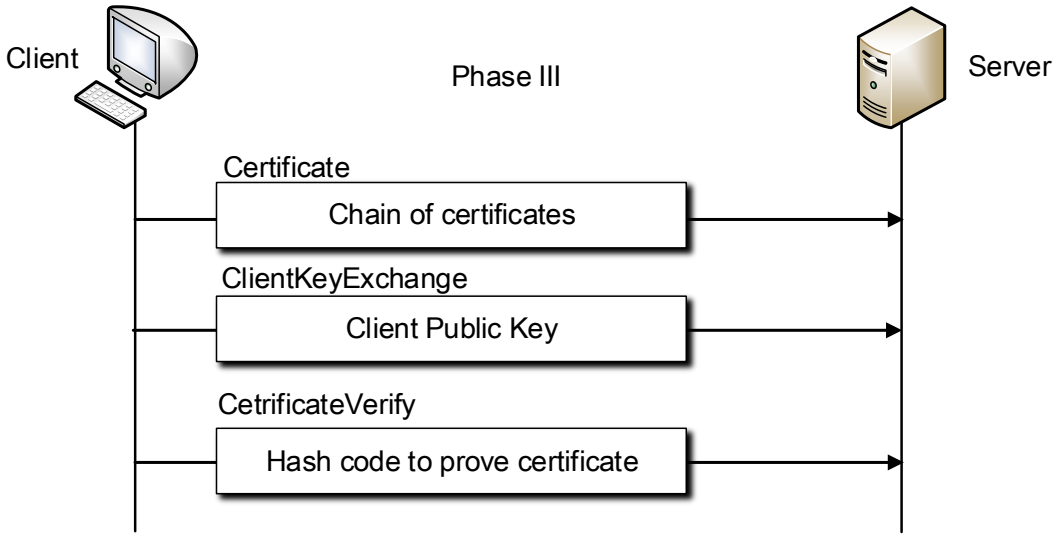
- Handshake 프로토콜
  - 클라이언트 키 교환과 인증
    - 클라이언트가 서버에 대해 본인을 인증 받는 과정
    - 클라이언트는 서버에 대해 인증되고 클라이언트와 서버는 사전 마스터 비밀을 알게 됨
    - 메시지 교환 과정
      - Certificate
      - ClientKeyExchange
      - CertificateVerify
    - 키 교환 알고리즘
      - RSA
      - Anonymous DH
      - Ephemeral DH
      - Fixed DH



<그림 21> Handshake 프로토콜의 단계 3

SSL 프로토콜

- Handshake 프로토콜
  - 클라이언트 키 교환과 인증
    - 메시지 교환 과정 (1/3)
      - Certificate
        - 서버 키 교환 인증 절차에서 서버가 인증서를 요청할 때만 보내짐
        - 클라이언트가 서버에게 자신을 인증하기 위해 보내는 메시지
        - 클라이언트를 인증하는 인증서 체인을 포함하고 있음
        - 클라이언트가 보낼 인증서가 없다면, 인증서가 없다는 경고와 함께 Alert 메시지를 전송함



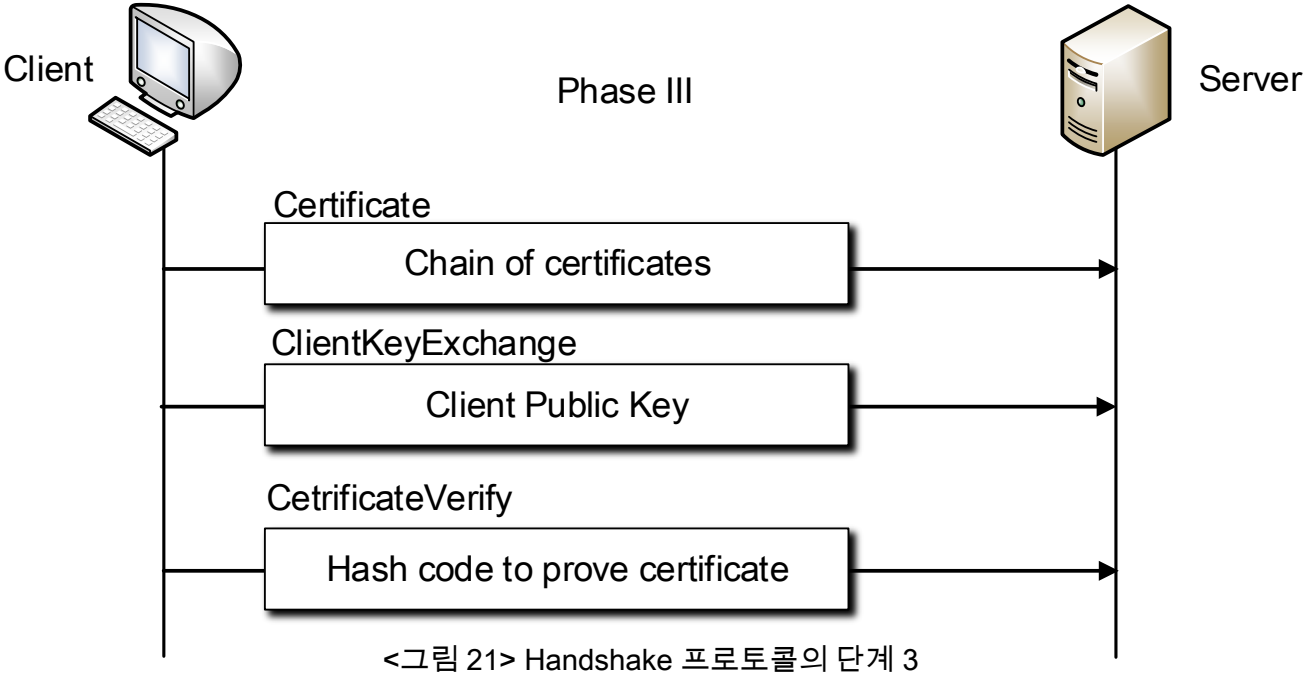
<그림 21> Handshake 프로토콜의 단계 3

SSL 프로토콜

- Handshake 프로토콜
  - 클라이언트 키 교환과 인증
    - 메시지 교환 과정 (2/3)
      - ClientKeyExchange
        - 클라이언트는 사전 마스터 비밀에 대한 역할을 포함한 메시지를 키 교환 알고리즘을 기반으로 보냄
        - 키 교환 알고리즘이 RSA인 경우 클라이언트는 모든 사전 마스터 비밀을 생성하고 서버의 RSA 공개 키로 암호화함
        - 키 교환 알고리즘이 익명 혹은 임시 Diffie-Hellman인 경우 클라이언트는 Diffie-Hellman 절반-키를 보냄
        - 키 교환 알고리즘이 Fortezza인 경우 클라이언트는 Fortezza 매개변수를 보냄

SSL 프로토콜

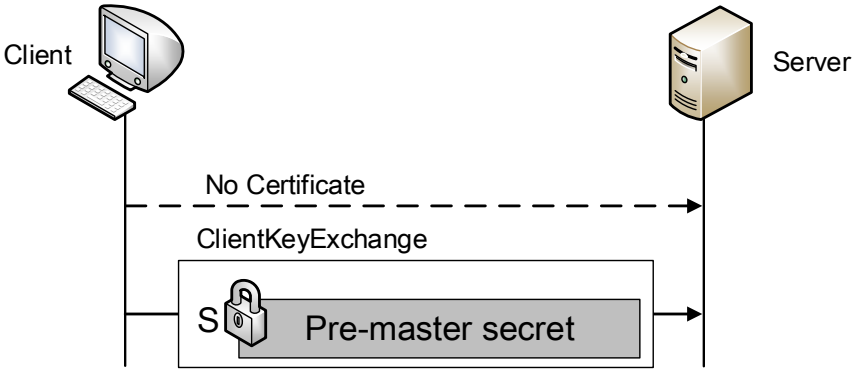
- Handshake 프로토콜
  - 클라이언트 키 교환과 인증
    - 메시지 교환 과정 (3/3)
      - CertificateVerify
        - 서버가 보낸 공개 키에 대응하는 개인 키를 알고 있다는 것을 증명하는 메시지
        - 메시지를 생성하고 개인 키로 서명함으로서 개인 키 소유를 증명함



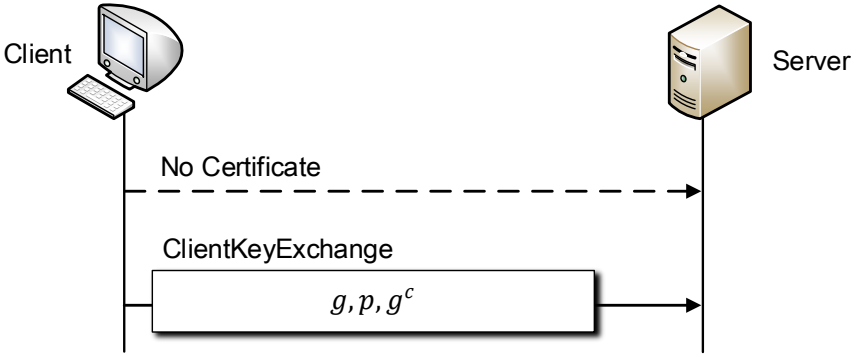
<그림 21> Handshake 프로토콜의 단계 3

SSL 프로토콜

- Handshake 프로토콜
  - 클라이언트 키 교환과 인증
    - 키 교환 알고리즘 (1/2)
      - RSA
        - 서버가 요구하지 않는 한, Certificate 메시지는 없음
        - RSA 공개 키로 암호화한 사전 마스터 키를 ClientKeyExchange 메시지로 전송함
      - 익명 Diffie-Hellman
        - 클라이언트와 서버가 익명이기 때문에 인증서를 요구할 권리가 없으므로, Certificate 메시지가 없음
        - 클라이언트는 DH 매개변수와 절반-키를 서명해서 ClientKeyExchange로 보냄



a . RSA



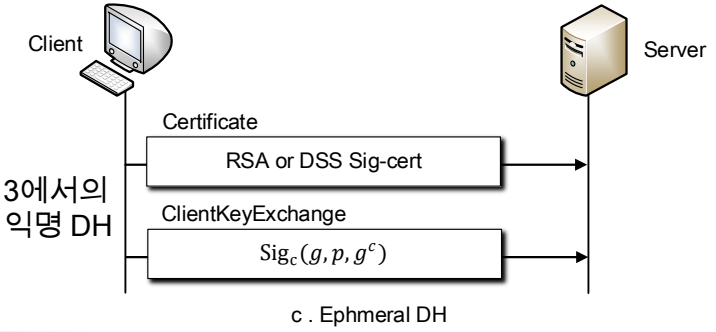
b. Anonymous DH

<그림 22-1> 단계 3에서의 키 교환 알고리즘 – RSA

<그림 22-2> 단계 3에서의 키 교환 알고리즘 – 익명 DH

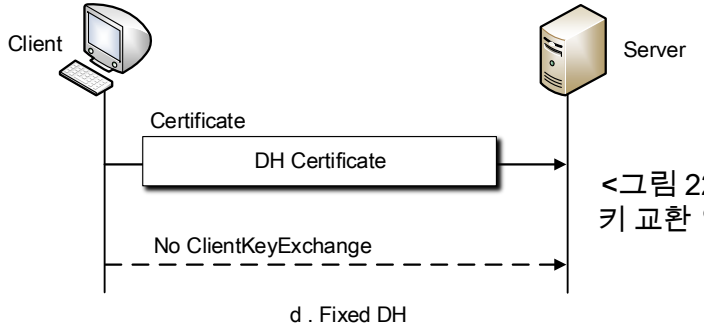
SSL 프로토콜

- Handshake 프로토콜
  - 클라이언트 키 교환과 인증
    - 키 교환 알고리즘 (2/2)
      - 임시 Diffie-Hellman
        - 클라이언트는 합의된 암호 방식에 따른 RSA 혹은 DSS 인증서를 Certificate 메시지로 보내야 함
        - 클라이언트는 DH 매개변수와 절반-키를 서명하여 ClientKeyExchange 메시지로 서버에게 보냄
        - 클라이언트가 인증서를 가지지 않았다면, Alert 메시지를 송신하도록 서버에게 요청함
      - 고정 Diffie-Hellman
        - 클라이언트는 항상 DH 인증서를 Certificate 메시지로 보냄
        - ClientKeyExchange는 전송하지 않음



<그림 22-3> 단계 3에서의 키 교환 알고리즘- 익명 DH

c . Ephmeral DH

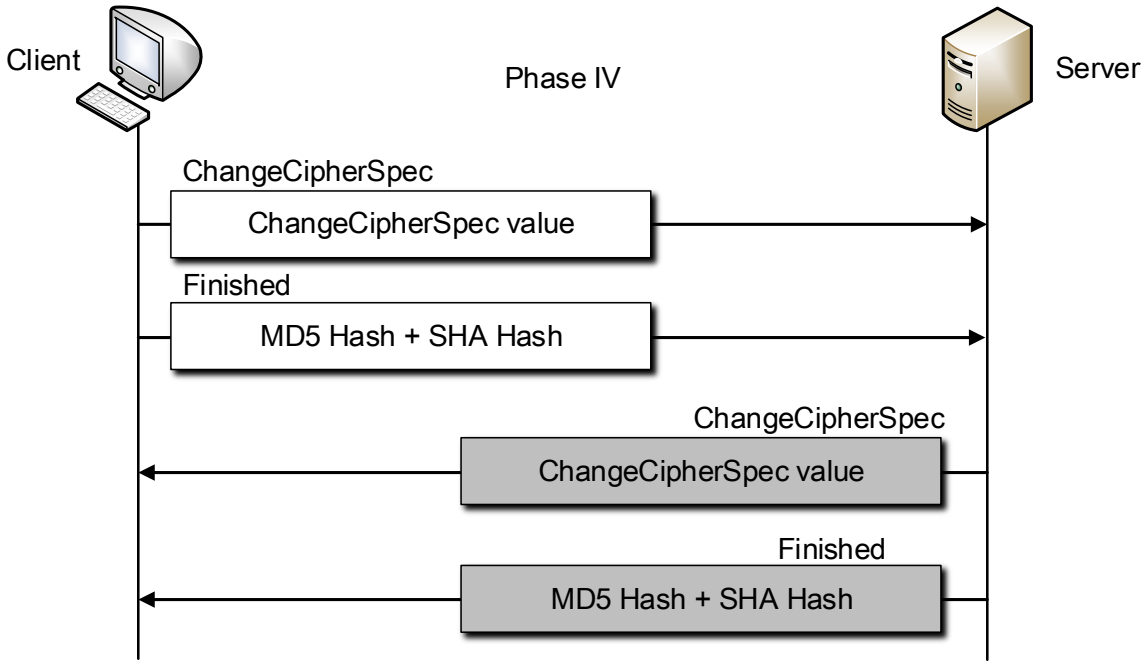


<그림 22-4> 단계 3에서의 키 교환 알고리즘- 고정 DH

d . Fixed DH

SSL 프로토콜

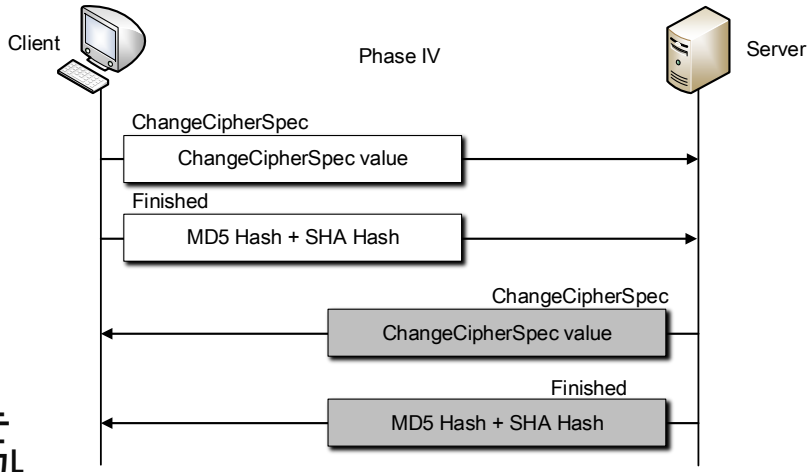
- Handshake 프로토콜
  - 종결과 종료
    - 클라이언트와 서버가 합의된 암호 규격으로 암호를 변경하고 데이터를 교환할 준비를 하는 과정
    - 메시지 교환 과정 (1/2)
      - ChangeCipherSpec
      - Finished
      - ChangeCipherSpec
      - Finished



<그림 23> Handshake 프로토콜의 단계 4

SSL 프로토콜

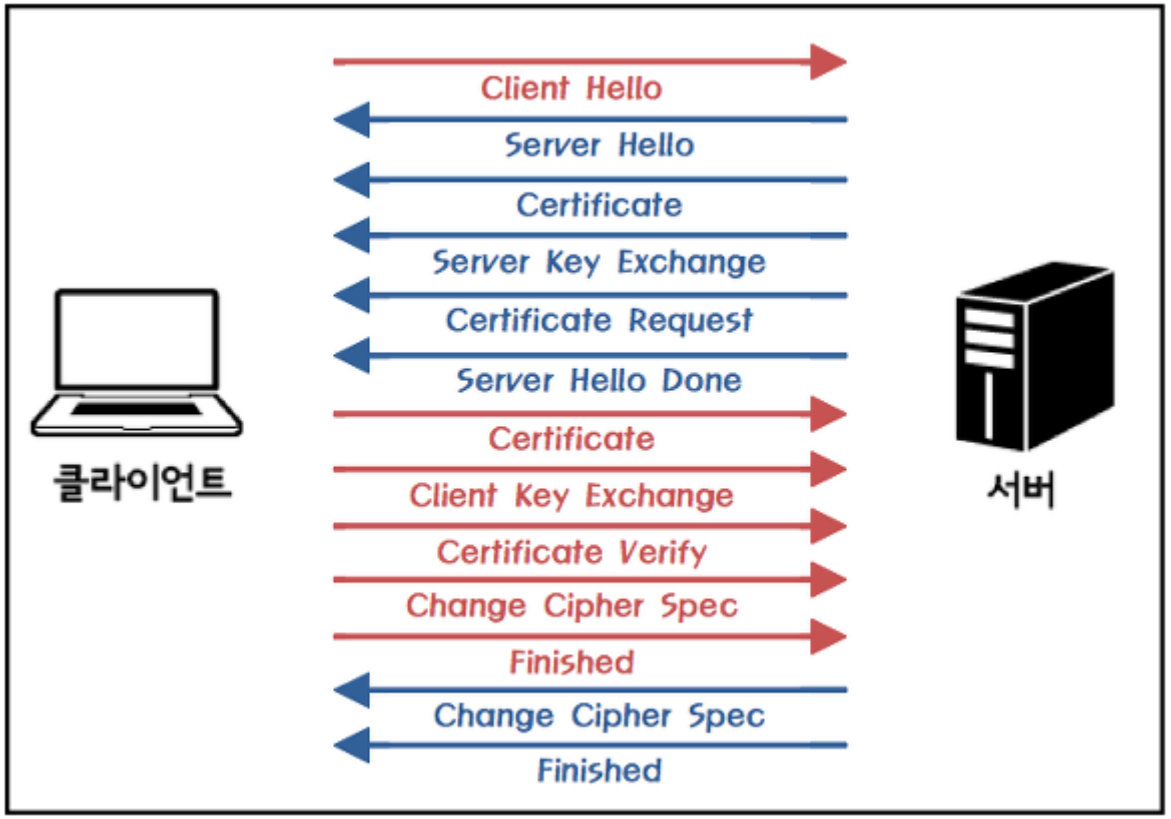
- Handshake 프로토콜
  - 종결과 종료
    - 메시지 교환 과정 (2/2)
      - ChangeCipherSpec
        - 클라이언트가 서버에게 이후에 전송되는 모든 메시지는 서버와 협상된 알고리즘과 키를 이용해 암호화한다는 것을 알리는 메시지
      - Finished
        - 클라이언트가 서버에게 프로토콜의 종료를 알리는 메시지
      - ChangeCipherSpec
        - 서버가 클라이언트에게 이후에 전송되는 모든 메시지는 서버와 협상된 알고리즘과 키를 이용해 암호화한다는 것을 알리는 메시지
      - Finished
        - 서버가 Handshake 프로토콜이 완전히 종료되었다는 것을 클라이언트에게 알리는 메시지



<그림 23> Handshake 프로토콜의 단계 4

SSL 프로토콜

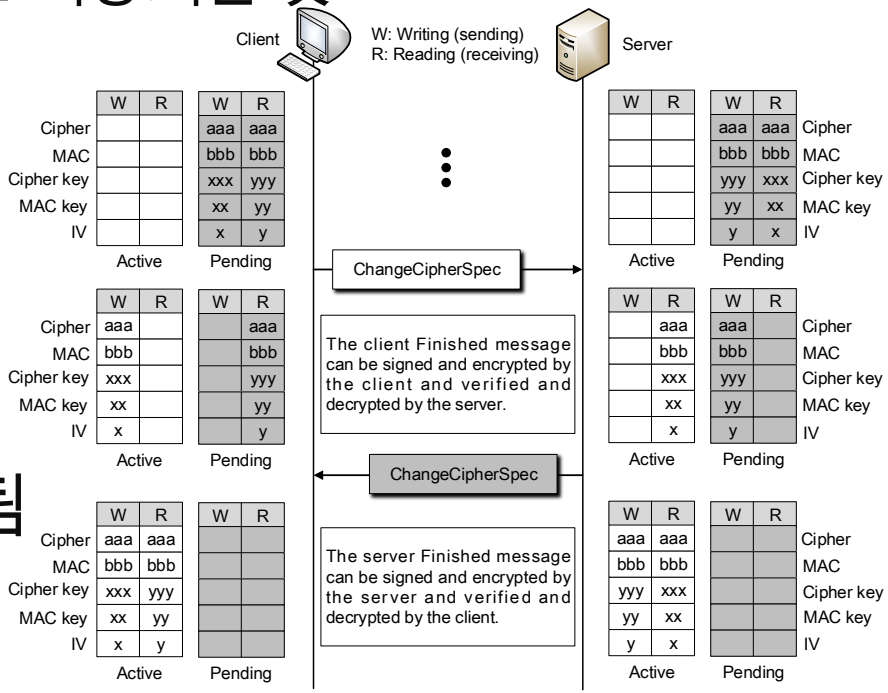
- Handshake 프로토콜
  - 전체 흐름도



<그림 24> Handshake 프로토콜 흐름도

SSL 프로토콜

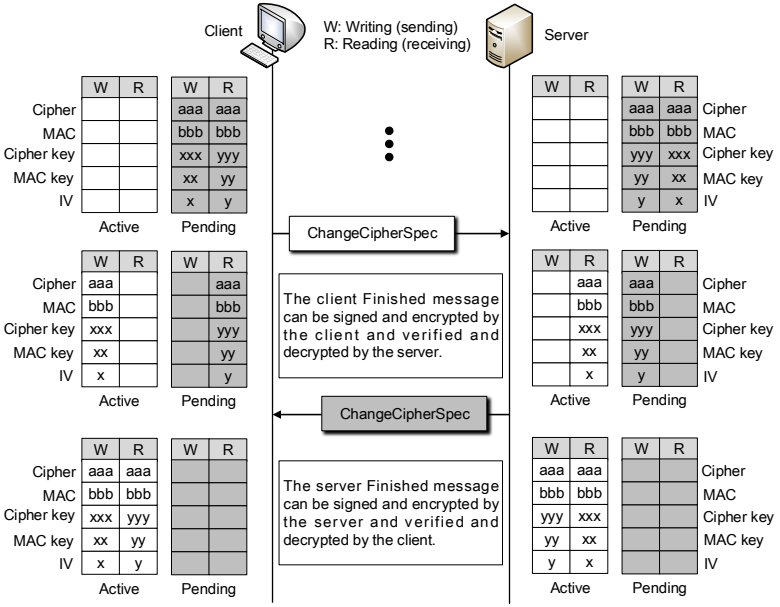
- ChangeCipherSpec 프로토콜
  - 대기와 활성 상태를 이용해 협의한 암호화 알고리즘과 교환된 키로 암호화를 결정하는 프로토콜
    - 대기(Pending)
      - Handshake 과정 중에 협의 중인 암호 알고리즘, 마스터 비밀, 암호화 키 등 매개변수들이 임시로 저장되는 곳
    - 활성(Active)
      - 암,복호화에 사용되는 매개변수들이 위치하는 곳
  - 특징
    - ChangeCipherSpec 메시지를 보내거나 받을 때 매개변수와 비밀을 사용할 수 없을 때 사용됨



<그림 25> 대기상태에서 활성 상태로 매개변수의 이동

SSL 프로토콜

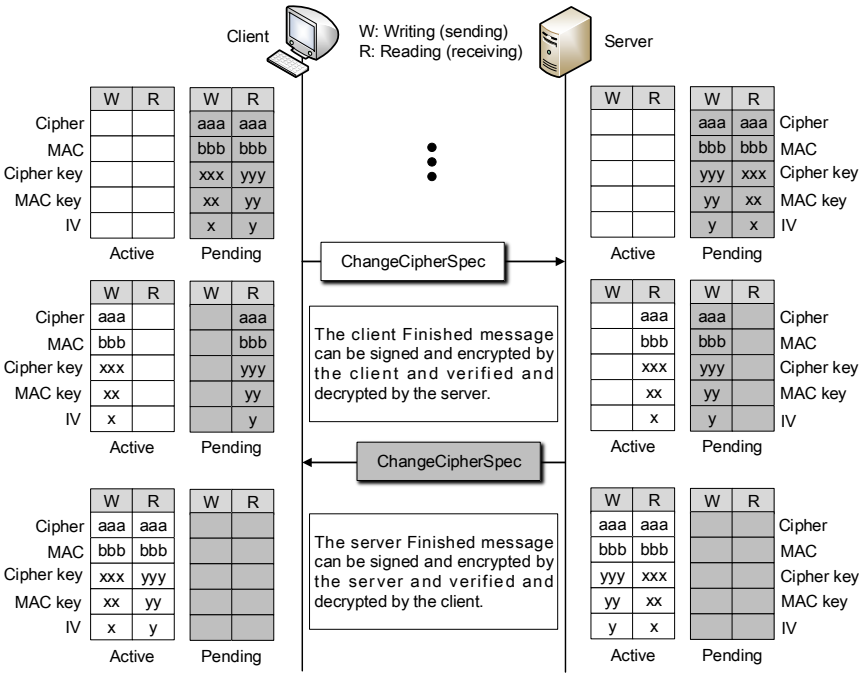
- ChangeCipherSpec 프로토콜
  - 동작 과정(1/2)
    - 클라이언트가 ChangeCipherSpec 메시지를 서버에게 보냄
      - 클라이언트의 write 매개변수가 대기에서 활성으로 변함
        - write 매개변수는 보내는 메시지를 암호화하고 MAC 을 붙일 때 사용됨
    - 클라이언트가 Finished 메시지를 암호화하여 서버에게 보냄
    - 서버는 ChangeCipherSpec메시지를 수신함
      - 서버의 reading 매개변수가 대기에서 활성으로 변함
        - read 매개변수는 받은 메시지를 복호화하고 MAC을 검증할 때 사용됨



<그림 25> 대기상태에서 활성 상태로 매개변수의 이동

SSL 프로토콜

- ChangeCipherSpec 프로토콜
  - 동작 과정(2/2)
    - 서버는 수신한 Finished 메시지를 복호화한 후 검증함
    - 서버가 클라이언트에게 ChangeCipherSpec 메시지를 보냄
      - 서버의 write 매개변수가 대기 상태에서 활성 상태로 변함
    - 서버가 클라이언트에게 Finished 메시지를 보냄
      - 클라이언트의 read 매개변수가 대기 상태에서 활성 상태로 변함



<그림 25> 대기상태에서 활성 상태로 매개변수의 이동

SSL 프로토콜

- Alert 프로토콜
  - HandShake 과정에서 발생한 오류와 비정상 상태를 알리기 위한 프로토콜
  - 문제점과 경고를 설명하는 메시지만을 보냄
  - 메시지 종류 (1/3)
    - CloseNotify
      - 상대방에게 더 이상 보낼 메시지가 없음을 알리며 연결을 종료하겠다고 알림
    - UnexpectedMessage
      - 상대방에게 메시지는 적당하지 않은 메시지임을 알림
    - BadRecordMAC
      - 상대방에게 메시지의 내용과 MAC가 일치하지 않는다는 것을 알림
    - DecompressionFailure
      - 상대방에게 메시지의 데이터의 압축을 풀지 못한다는 것을 알림

SSL 프로토콜

- Alert 프로토콜
  - 메시지 종류 (2/3)
    - HandshakeFailure
      - 상대방에게 handshake를 마무리하지 못했다는 것을 알림
    - NoCertificate
      - 상대방에게 자신은 인증서가 없다는 것을 알림
    - BadCertificate
      - 상대방에게 받은 인증서의 형식이 맞지 않거나, 인증을 할 수 없는 인증서임을 알림
    - UnsupportedCertificate
      - 상대방에게 받은 인증서는 지원할 수 없는 인증서 형식임을 알림
    - CertificateRevoked
      - 상대방에게 받은 인증서가 발급자로부터 폐기되었다는 것을 알림

SSL 프로토콜

- Alert 프로토콜
  - 메시지 종류 (3/3)
    - CertificateExpired
      - 상대방에게 받은 인증서가 사용 기간이 지났음을 알림
    - CertificateUnknown
      - 상대방에게 받은 인증서에 알 수 없는 문제가 있다는 것을 알림
    - IllegalParameter
      - 상대방에게 받은 메시지가 형식에 맞지 않다는 것을 알림

SSL 프로토콜

- Alert 프로토콜
  - 메시지 종류

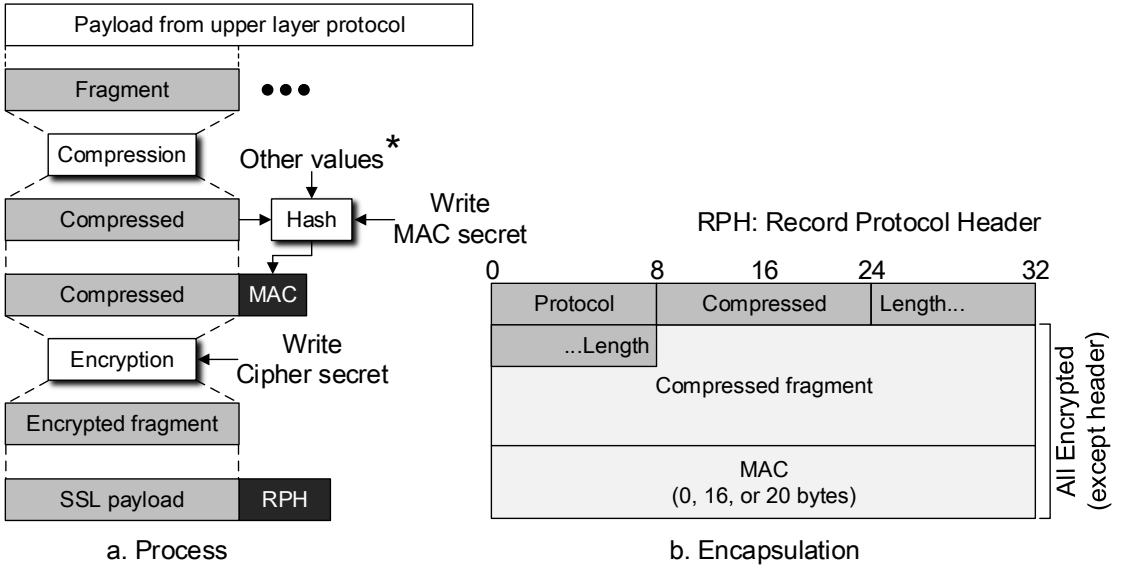
[표 4] SSL에서 규정한 경고

value	Description	Meaning
0	CloseNotify	Sender will not send any more messages.
10	UnexpectedMessage	An inappropriate message received.
20	BadRecordMAC	An incorrect MAC received.
30	DecompressionFailure	Unable to decompress appropriately.
40	HandshakeFailure	Sender unable to finalize the handshake.
41	NoCertificate	Client has no certificate to send.
42	BadCertificate	Received certificate corrupted.
43	UnsupportedCertificate	Type of received certificate is not supported.
44	CertificateRevoked	Signer has revoked the certificate.
45	CertificateExpired	Certificate expired.
46	CertificateUnknown	Certificate unknown.
47	IllegalParameter	An out-of-range or inconsistent field.

SSL 프로토콜

- Record 프로토콜
  - Handshake, ChangeCipherSpec, Alert 프로토콜 혹은 응용 계층으로부터 오는 메시지를 전송 계층으로 보내는 프로토콜
  - 동작 과정
    - 단편화
    - 압축 및 MAC 추가
    - 서명 및 검증
    - 암호화 및 복호화
    - 프레임 구성 및 해제

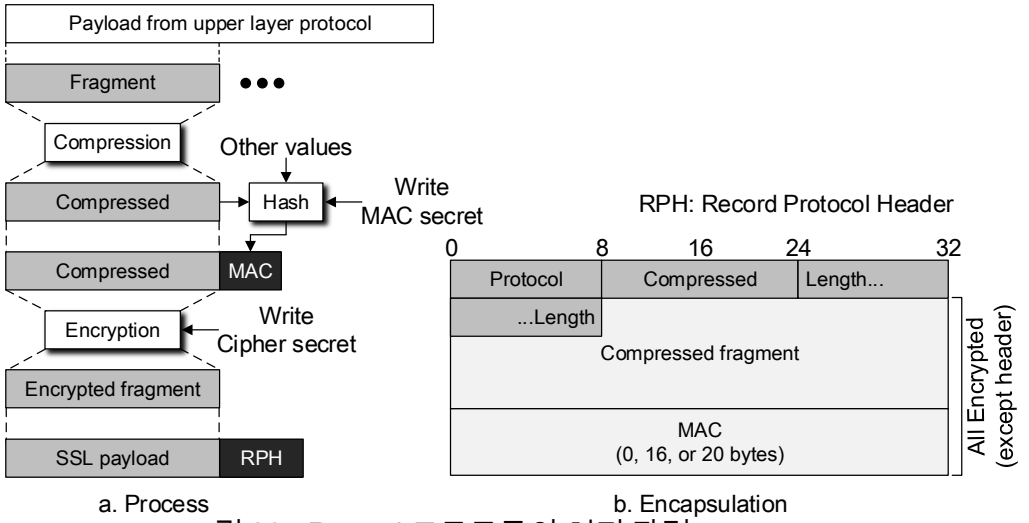
*Other values
MAC 계산에 들어가는 입력값들임 (Sequence number, type, version, length)



<그림 26> Record 프로토콜의 처리 과정

SSL 프로토콜

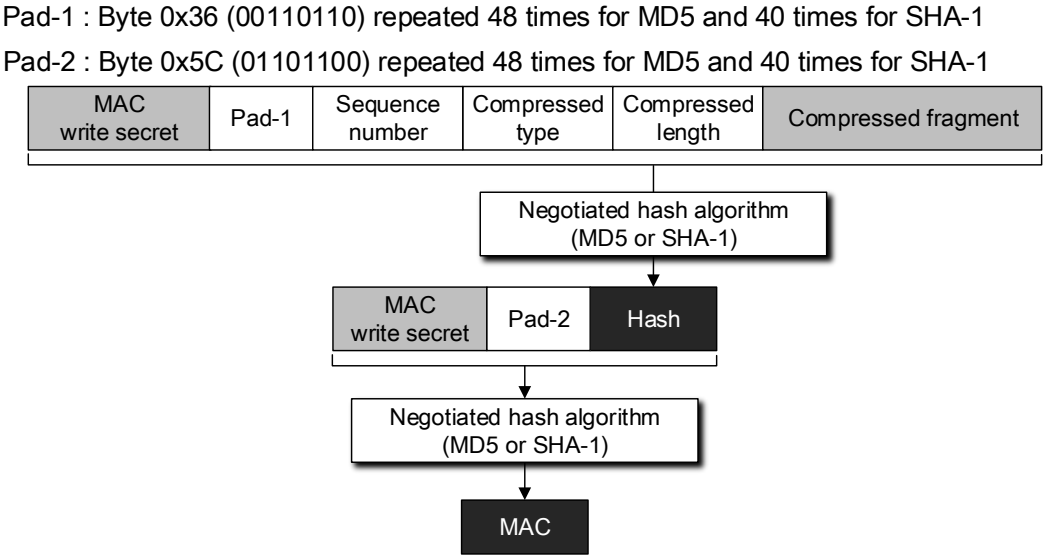
- Record 프로토콜
  - 동작 과정 (1/3)
    - 단편화
      - 상위 프로토콜에서 온 메시지를 214바이트의 블록으로 단편화됨
      - 수신자 측에서, 단편화는 원래의 메시지로 만들기 위해 결합됨
    - 압축
      - Handshake에서 합의한 방법으로 압축을 진행함
      - 압축률이 데이터 내용에 따라 달라진다는 점을 이용한 CRIME 공격 이후 제거됨



<그림 26> Record 프로토콜의 처리 과정

SSL 프로토콜

- Record 프로토콜
  - 동작 과정 (2/3)
    - MAC 추가
      - 무결성 보장과 재전송 방지를 위해, Handshake 과정에서 협의된 인증 방법으로 MAC을 계산한 후 압축된 메시지 뒤에 붙임
      - write 매개변수, 패딩 상수, 시퀀스 번호, 압축 타입, 압축 길이, 압축된 단편에 해시를 씌운 후, 다시 write 매개변수와 패딩 상수를 붙인 후 다시 해시화를 하여 최종 MAC를 생성함



<그림 27> MAC 계산 과정

SSL 프로토콜

- Record 프로토콜
  - 동작 과정 (3/3)
    - 암호화
      - 기밀성을 보장하기 위해 송신자가 압축된 단편과 MAC를 write 매개변수를 사용하여 암호화 하고, 수신자는 read 매개변수를 사용하여 복호화함
      - 암호화될 메시지 크기를 블록 크기의 배수로 맞추기 위해 패딩 상수를 추가함
    - 프레임 구성 및 해제
      - 송신자는 Record 프로토콜 헤더를 암호화된 메시지 앞에 붙임
      - 수신자는 메시지를 복호화하기 전에 헤더를 제거함

목 차

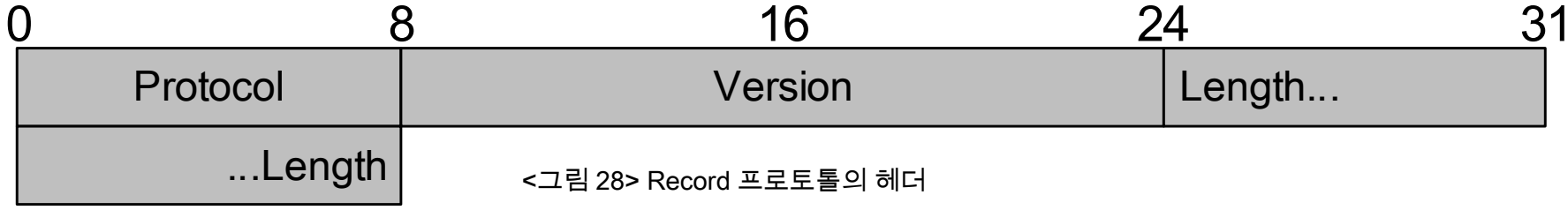
- 개요
- SSL 구조
- SSL 프로토콜
- SSL 메시지 형식
- 전송계층 보안(TLS, Transport Layer Security)

SSL 메시지 형식

- 개요
  - 프로토콜에 의해 캡슐화되는 메시지의 형식
- 종류
  - Record 프로토콜
  - ChangeCipherSpec 프로토콜
  - Alert 프로토콜
  - Handshake 프로토콜

SSL 메시지 형식

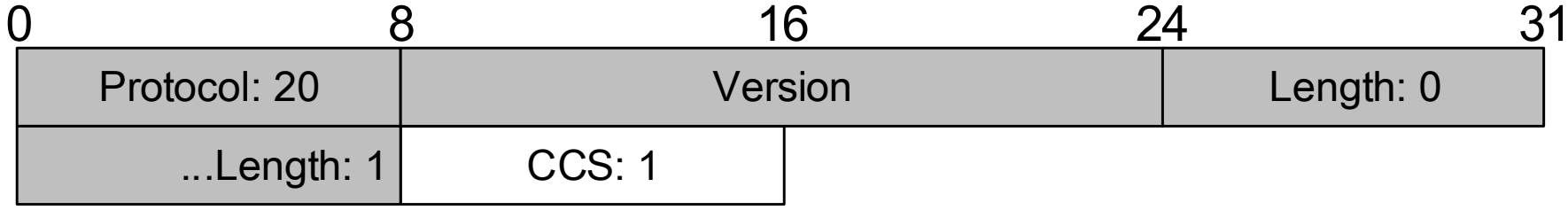
- Record 프로토콜
  - 프로토콜(Protocol)
    - 메시지의 발신지 혹은 목적지를 표시함
    - 20(ChangeCipherSpec), 21(Alert), 22(Handshake), 23(응용 계층)을 나타냄
  - Version
    - SSL의 버전을 나타냄
  - Length
    - 헤더없는 메시지의 크기를 바이트 단위로 나타냄



<그림 28> Record 프로토톨의 헤더

SSL 메시지 형식

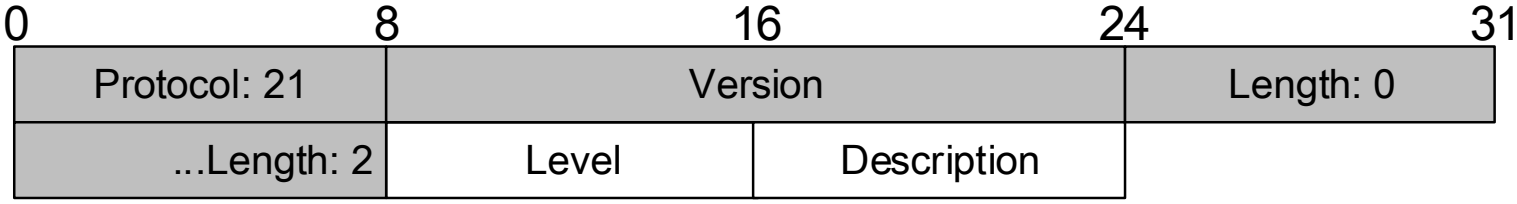
- ChangeCipherSpec 프로토콜
  - 프로토콜 값이 20인 Record 프로토콜 메시지에 캡슐화됨
  - CCS
    - ChangeCipherSpec 프로토콜이 가지는 1바이트의 메시지임



<그림 29> ChangeCipherSpec 메시지

SSL 메시지 형식

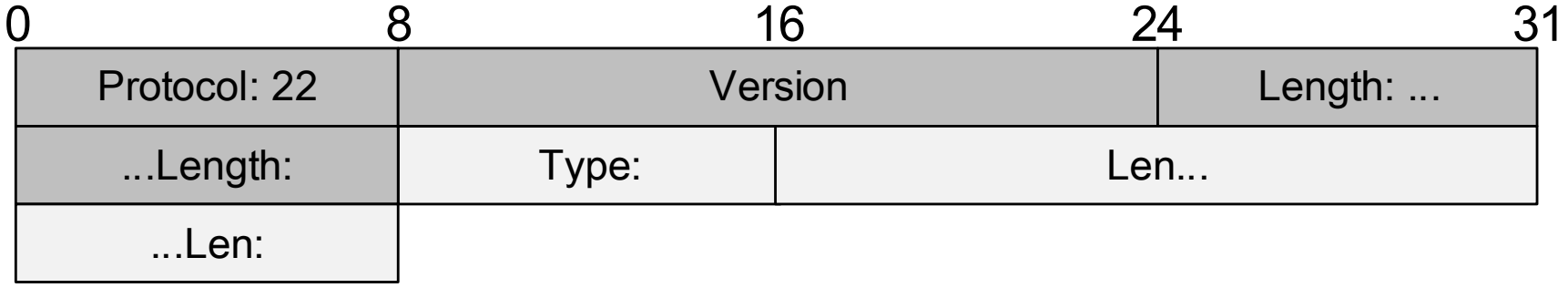
- Alert 프로토콜
  - 프로토콜 값이 21인 Record 프로토콜 메시지에 캡슐화됨
  - Level
    - 오류의 레벨을 나타냄
    - Warning
      - 1의 값을 가지며, 연결을 종료할 필요 없이 주의해야하는 것만 알림
    - Fatal
      - 2의 값을 가지며, 연결을 즉시 종료해야하는 문제가 생겼다는 것을 알림
  - Description
    - 오류의 유형을 나타냄



<그림 30> Alert 메시지

SSL 메시지 형식

- Handshake 프로토콜
  - 프로토콜 값이 20인 Record 프로토콜 메시지에 캡슐화됨
  - Len, Length
    - 유형 길이와 필드 길이를 포함하는 메시지의 길이를 나타냄
  - Type
    - 메시지의 유형을 나타냄



<그림 31> Handshake 프로토콜의 헤더

SSL 메시지 형식

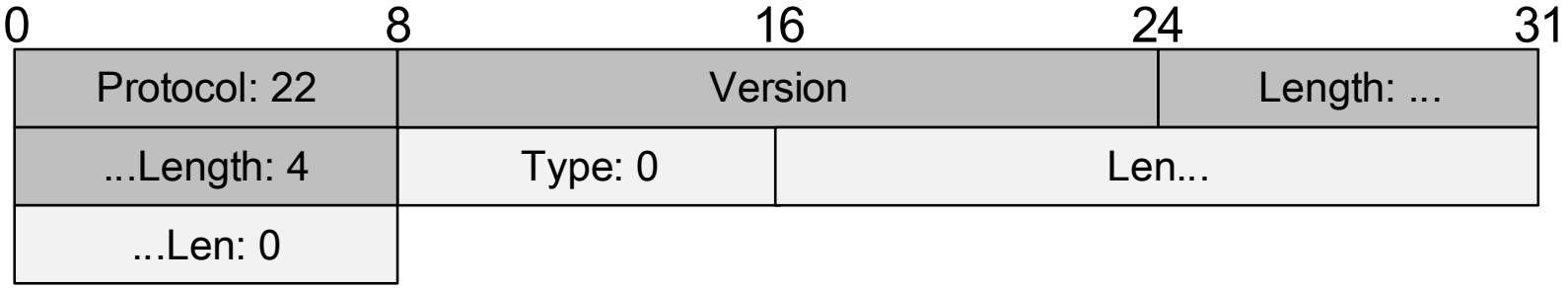
- Handshake 프로토콜
  - Type (1/16)
    - 메시지의 유형을 나타냄
    - 종류
      - HelloRequest
      - ClientHello
      - ServerHello
      - Certificate
      - ServerKeyExchange
      - CertificateRequest
      - ServerHelloDone
      - CertificateVerify
      - ClientKeyExchange
      - Finished

[표 5] Handshake 메시지의 유형

Type	Message
0	HelloRequest
1	ClientHello
2	ServerHello
11	Certificate
12	ServerKeyExchange
13	CertificateRequest
14	ServerHelloDone
15	CertificateVerify
16	ClientKeyExchange
20	Finished

SSL 메시지 형식

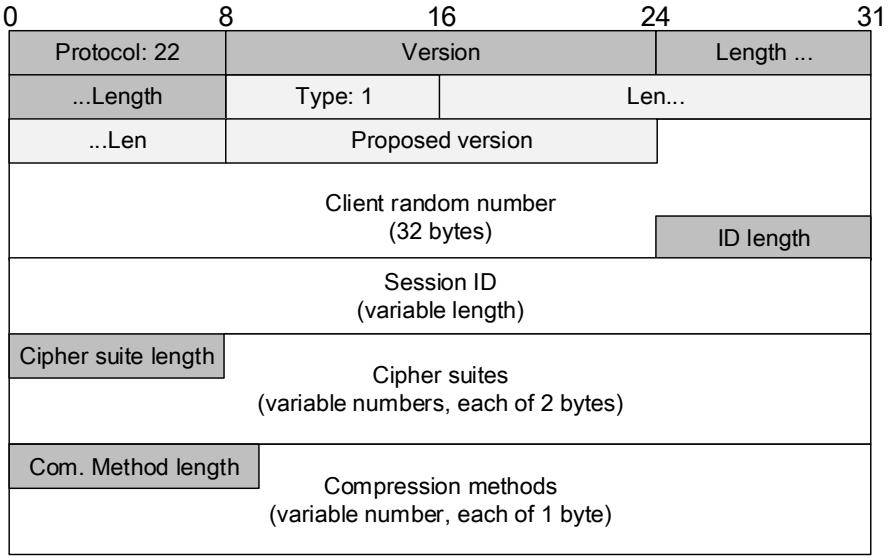
- Handshake 프로토콜
  - Type (2/16)
    - HelloRequest
      - 서버에서 클라이언트로 요청되는, 세션의 시작을 알리는 메시지임
      - 클라이언트는 ClientHello 메시지와 협의한 보안 매개변수를 송신해야함
      - 유형 필드값은 0임



<그림 32> HelloRequest 메시지 구성

SSL 메시지 형식

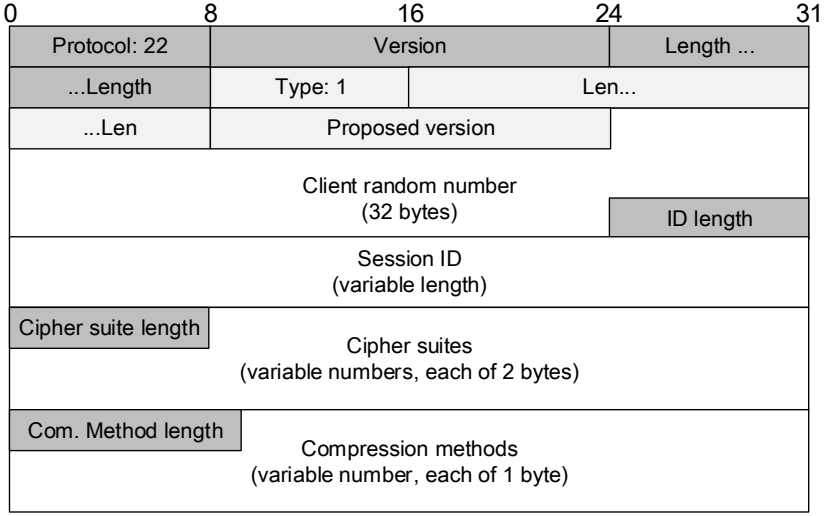
- Handshake 프로토콜
  - Type (3/16)
    - ClientHello (1/3)
      - Handshake 중에 첫 번째로 교환되는 메시지임
      - 유형 필드값은 1임
      - 버전(Version)
        - SSL에서 사용된 버전을 보여주는 2바이트 필드임
      - 클라이언트 랜덤 난수(Client Random Number)
        - 보안 매개변수를 생성할 때 필요한 클라이언트 난수를 송신하는 32바이트 필드임
      - 세션 ID 길이(Session ID Length)
        - 세션 ID 길이를 정의하는 1바이트 필드임



<그림 33> ClientHello 메시지 구성

SSL 메시지 형식

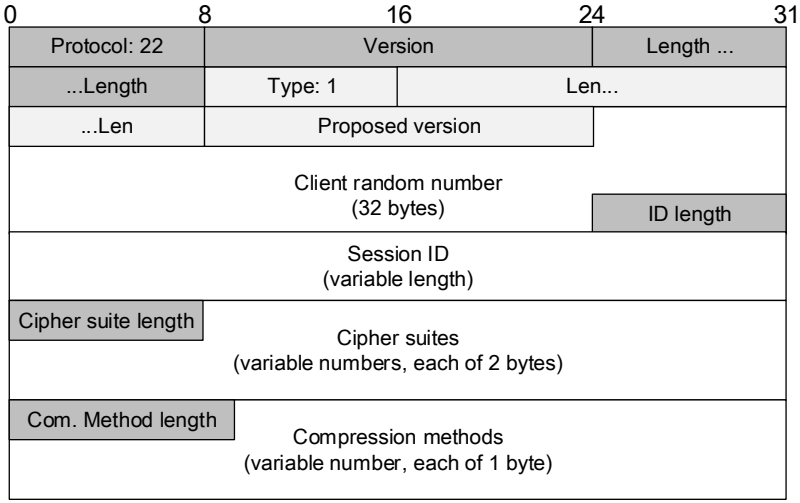
- Handshake 프로토콜
  - Type (4/16)
    - ClientHello (2/3)
      - 세션 ID(Session ID)
        - 서버가 생성하는 세션 ID를 저장하는 32 바이트 필드임
        - 클라이언트는 세션 ID를 저장하여 현재 세션이 끝난 후에도 재사용하여 세션을 다시 시작하는 게 가능함
      - 암호 그룹 길이(Cipher Suite Length)
        - 클라이언트가 제안한 암호 그룹 목록을 나타내는 2바이트 필드임
      - 암호 그룹 목록(Cipher Suite List)
        - 클라이언트가 지원하는 암호 그룹의 목록을 제공하는 가변 길이 필드임
        - 가장 선호하는 것에서 덜 선호하는 것으로 내림차순으로 제공함
        - 각 암호 그룹은 2바이트 수로 부호화됨



<그림 33> ClientHello 메시지 구성

SSL 메시지 형식

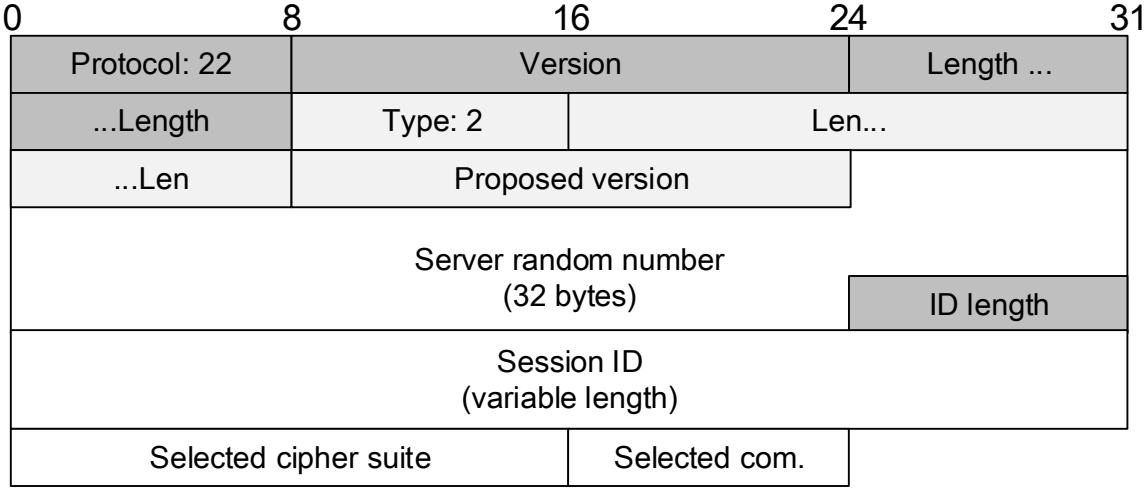
- Handshake 프로토콜
  - Type (5/16)
    - ClientHello (3/3)
      - 압축 방법 길이(Compression Methods Length)
        - 클라이언트가 제안한 압축 방법의 길이를 나타내는 1바이트 필드임
      - 압축 방법 목록(Compression Methods List)
        - 클라이언트가 지원하는 압축 방법의 목록을 제공하는 가변 길이 필드임
        - 가장 선호하는 것에서 덜 선호하는 것으로 내림차순으로 제공함
        - 각 암호 그룹은 1바이트 수로 부호화됨



<그림 33> ClientHello 메시지 구성

SSL 메시지 형식

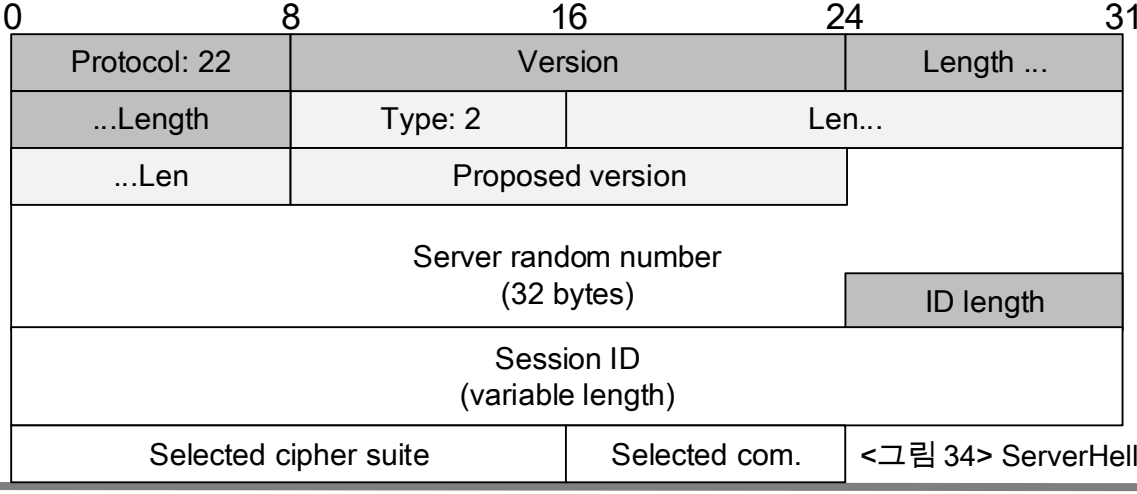
- Handshake 프로토콜
  - Type (6/16)
    - ServerHello(1/2)
      - ClientHello 메시지에 대한 서버의 응답 메시지이며 유형 값은 2임
      - 버전(Version)
        - ClientHello와 동일함
      - 서버 난수 필드(Server random number)
        - 서버에 의해 선택된 값으로 정의함
      - 세션 ID 길이(Session ID Length)
        - ClientHello와 동일함



<그림 34> ServerHello 메시지 구성

SSL 메시지 형식

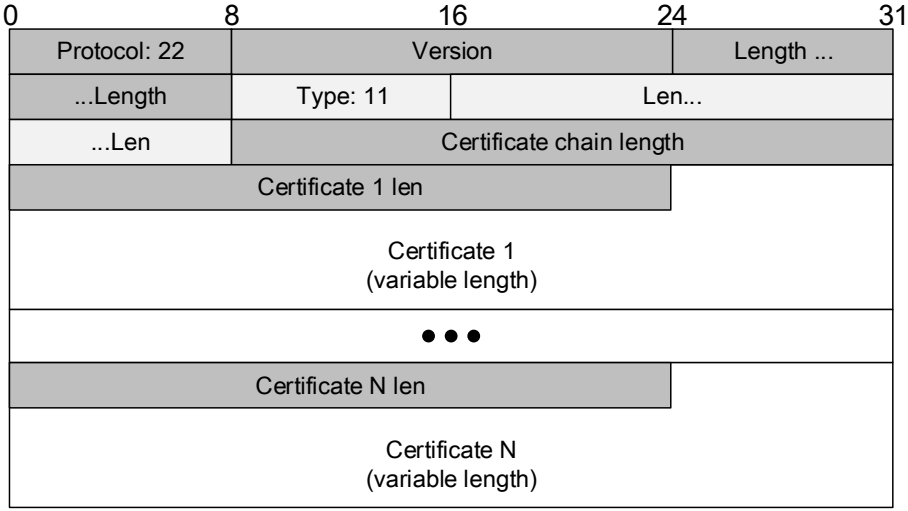
- Handshake 프로토콜
  - Type (7/16)
    - ServerHello(2/2)
      - 세션 ID(Session ID)
        - ClientHello와 동일하나 오래된 세션을 다시 시작하지 않는 한 공란이며 길이도 0으로 설정되어 있음
      - 선택된 암호 그룹(Selected cipher suite)
        - 서버에 의해 선택된 단일 암호 그룹을 나타내는 16바이트 필드임
      - 압축 방법 필드(Selected Com.)
        - 서버에 의해 선택된 압축 방법을 나타내는 8바이트 필드임



<그림 34> ServerHello 메시지 구성

SSL 메시지 형식

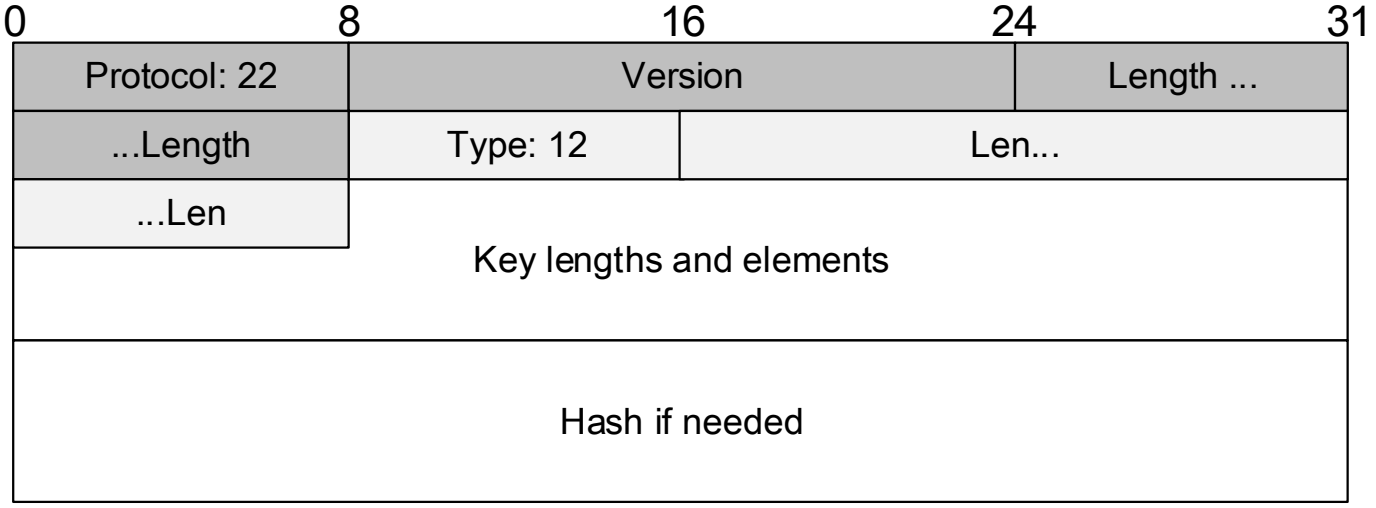
- Handshake 프로토콜
  - Type (8/16)
    - Certificate
      - 공개 키 인증서의 체인 목록을 보내는 메시지임
      - 유형 필드값은 11임
      - 인증서 체인 길이(Certificate chain length)
        - 인증서 체인의 길이를 보여주는 3바이트 필드임
      - 인증서 체인(Certificate chain)
        - 클라이언트 또는 서버가 전달하는 공개 키 인증서의 체인을 나열하는 가변 길이 필드임
        - 각 인증서에는 3바이트 길이 필드와 가변 길이 인증서 필드를 포함됨



<그림 35> Certificate 메시지 구성

SSL 메시지 형식

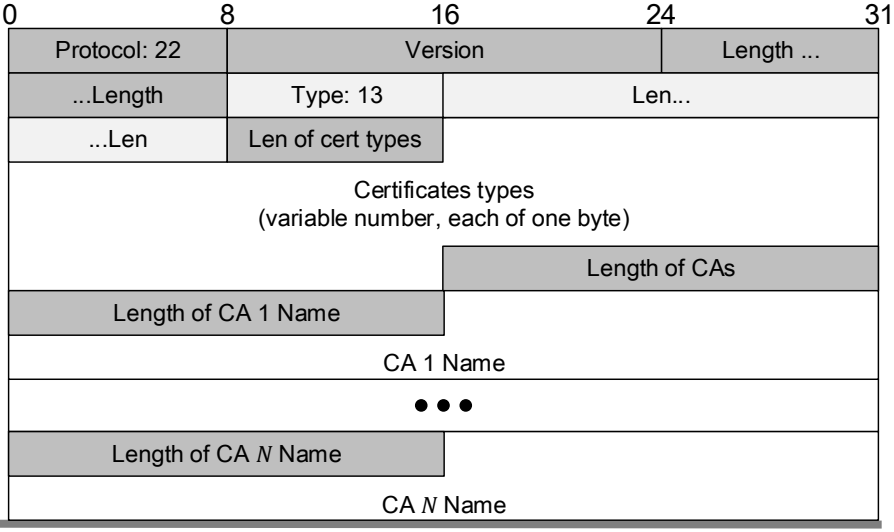
- Handshake 프로토콜
  - Type (9/16)
    - ServerKeyExchange
      - 서버가 클라이언트에게 서버의 공개 키를 보내주는 메시지임
      - 공개 키는 클라이언트와 서버가 정한 알고리즘에 따라 달라짐
      - 공개 키는 키 교환에 사용되고, 클라이언트는 받은 공개 키를 사용해 세션키를 암호화해서 서버에게 보냄
      - 유형 값은 12임



<그림 36> ServerKeyExchange 메시지 구성

SSL 메시지 형식

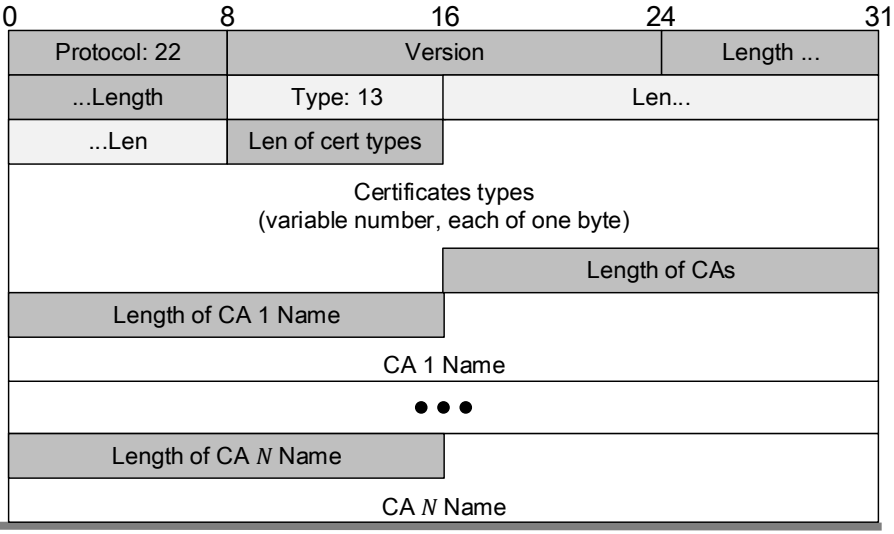
- Handshake 프로토콜
  - Type (10/16)
    - CertificateRequest(1/2)
      - 서버에서 클라이언트에게 자신의 인증을 요청하는 메시지임
      - 유형 필드값은 13임
      - 인증서 유형의 길이(Len of Cert Types)
        - 인증서 유형의 길이를 나타내는 1바이트 필드임
      - 인증서 유형(Certificates Types)
        - 서버에서 수용한 공개 키 인증서 유형의 목록을 나타내는 가변 길이 필드임
        - 각 인증서 유형은 1바이트임



<그림 37> CertificateRequest 메시지 구성

SSL 메시지 형식

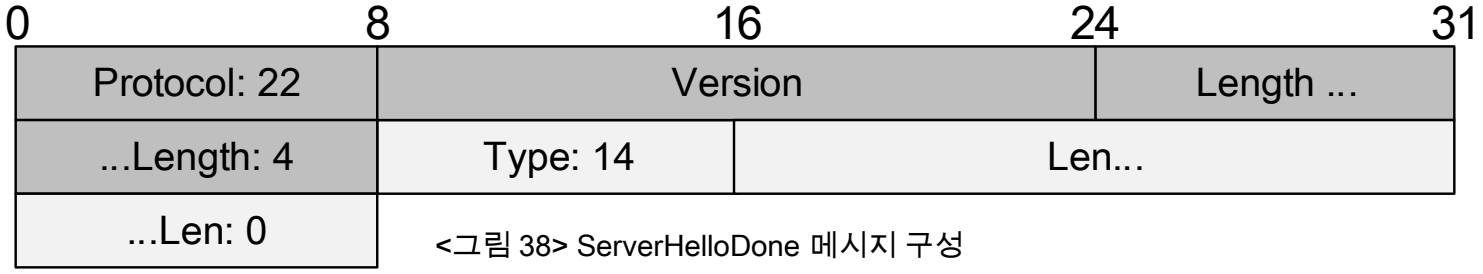
- Handshake 프로토콜
  - Type (11/16)
    - CertificateRequest(2/2)
      - 인증기관의 길이(Length of CAs)
        - 인증기관의 길이를 나타내는 2바이트 필드임
      - x 이름의 인증기관의 길이(Length of CA x Name)
        - x번째 인증기관 이름의 길이를 정의하는 2바이트 필드임
        - x의 값은 1에서 N 사이임
      - 인증기관x 이름(CA x Name)
        - x번째 인증기관의 이름을 나타내는 가변 길이 필드임
        - x의 값은 1에서 N 사이임



<그림 37> CertificateRequest 메시지 구성

SSL 메시지 형식

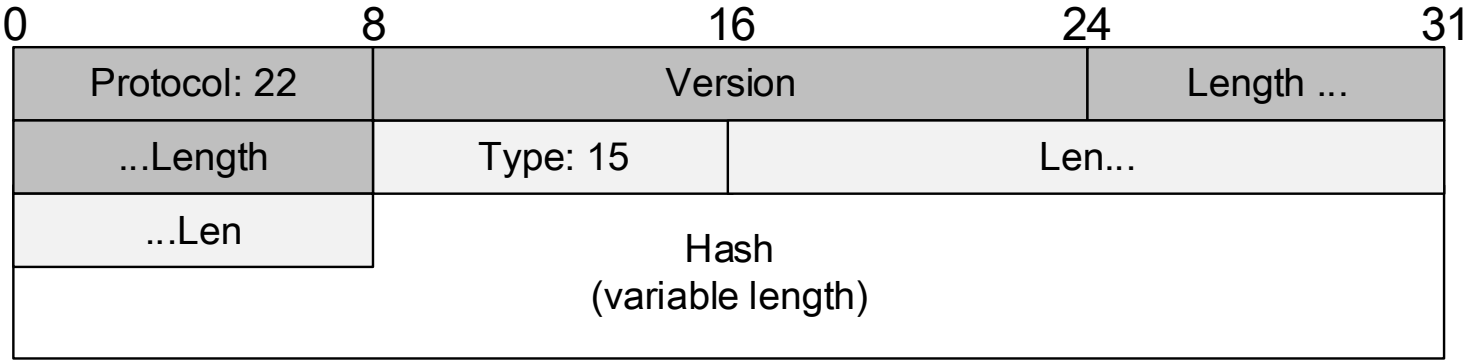
- Handshake 프로토콜
  - Type (12/16)
    - ServerHelloDone
      - 서버가 클라이언트에게 협상 단계에서 서버가 정보를 다 보냈다는 것을 알리는 메시지임
      - 필드를 가지고 있지 않기 때문에 메시지의 길이는 0임
      - 유형 필드값은 14임



<그림 38> ServerHelloDone 메시지 구성

SSL 메시지 형식

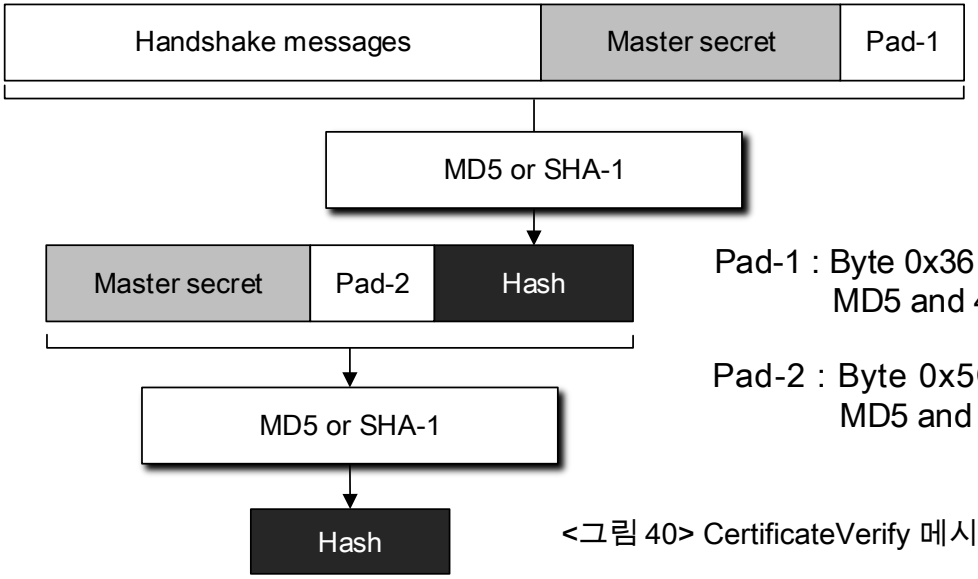
- Handshake 프로토콜
  - Type (13/16)
    - CertificateVerify(1/2)
      - 클라이언트가 공개 키 인증서에 연관된 개인 키가 자신의 것임을 증명하는 메시지임
      - CetrificateVerifty 이전에 송신한 모든 Handshake 메시지의 해시 값을 생성함
      - 클라이언트의 인증서 유형에 근거한 MD5 혹은 SHA-1 알고리즘을 이용하여 서명함



<그림 39> CertificateVerify 메시지 구성

SSL 메시지 형식

- Handshake 프로토콜
  - Type (14/16)
    - CertificateVerify(2/2)
      - 클라이언트의 개인 키가 DSS 인증서와 연관되어 있으면 해시는 SHA-1 알고리즘에 근거하며 필드의 길이는 20바이트임
      - 클라이언트의 개인 키가 RSA 인증서와 연관되어 있으면 해시는 MD5와 SHA-1 알고리즘에 근거한 2개의 해시이며 필드의 길이는 36바이트임



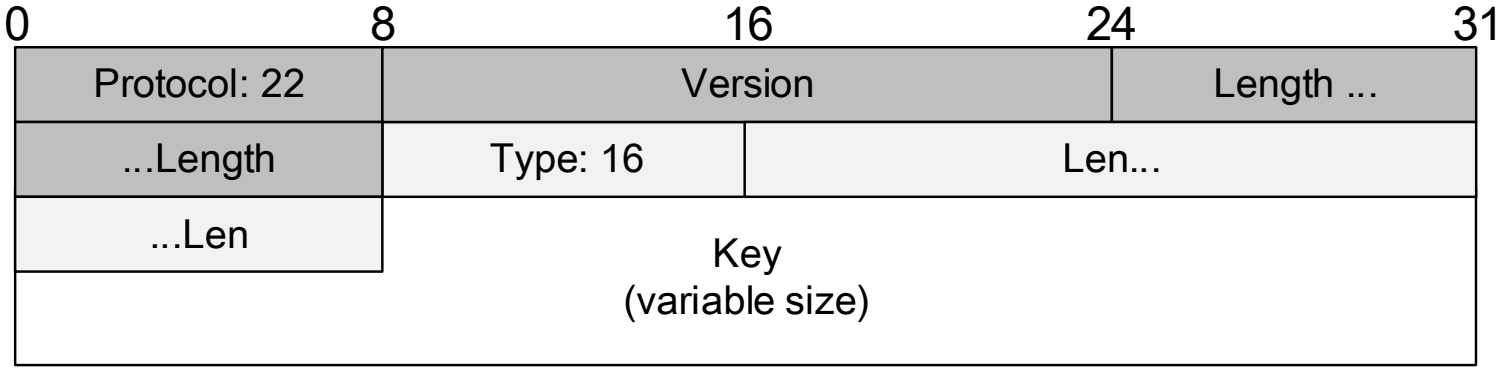
Pad-1 : Byte 0x36 (repeated 48 times for MD5 and 40 times for SHA-1)

Pad-2 : Byte 0x5C (repeated 48 times for MD5 and 40 times for SHA-1)

<그림 40> CertificateVerify 메시지에 대한 해시 계산

SSL 메시지 형식

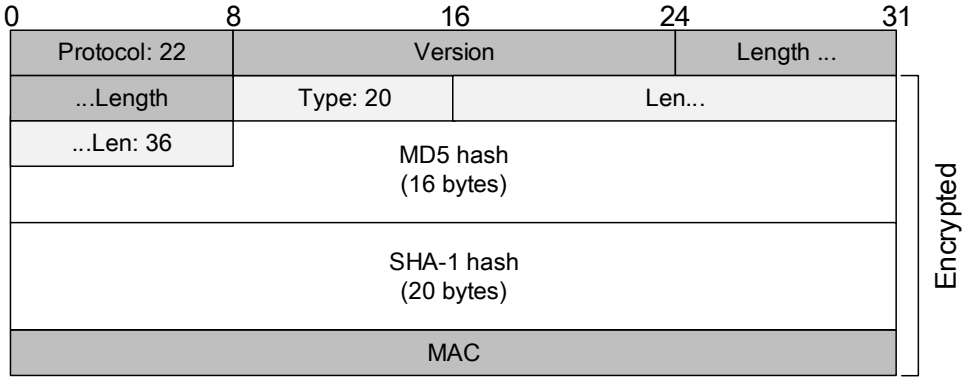
- Handshake 프로토콜
  - Type (15/16)
    - ClientKeyExchange
      - 클라이언트가 공개 키로 암호화된 세션 키를 서버에게 보내주는 메시지임
      - ServerKeyExchange 메시지로부터 얻은 공개 키를 사용함



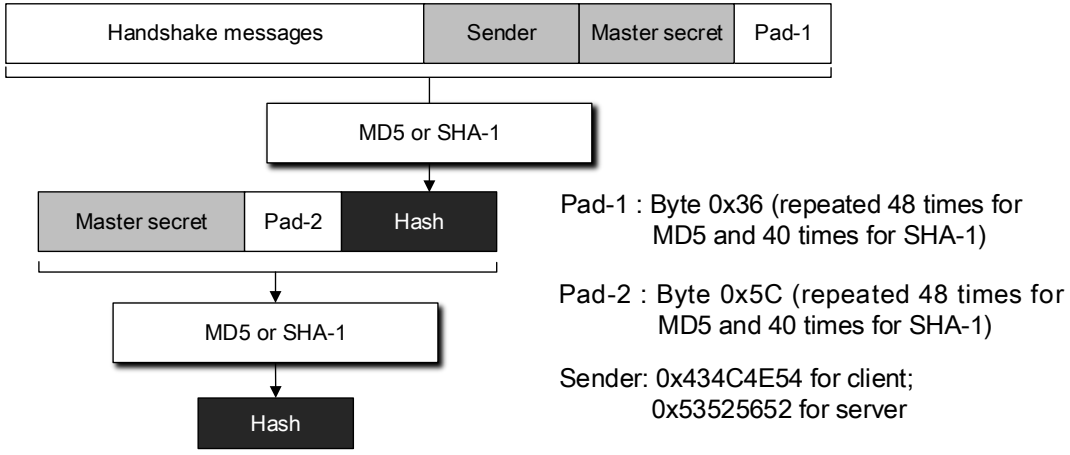
<그림 41> ClientKeyExchange 메시지 구성

SSL 메시지 형식

- Handshake 프로토콜
  - Type (16/16)
    - Finished
      - 클라이언트가 암호화 통신을 위한 협상이 끝났다는 것을 서버에게 알리는 메시지임
      - 세션 키로 협상 과정에서 주고 받은 메시지들을 암호화하여 전달되며 인증하는 역할을 함
        - MD5 알고리즘과 SHA-1 알고리즘을 이용해 암호화함



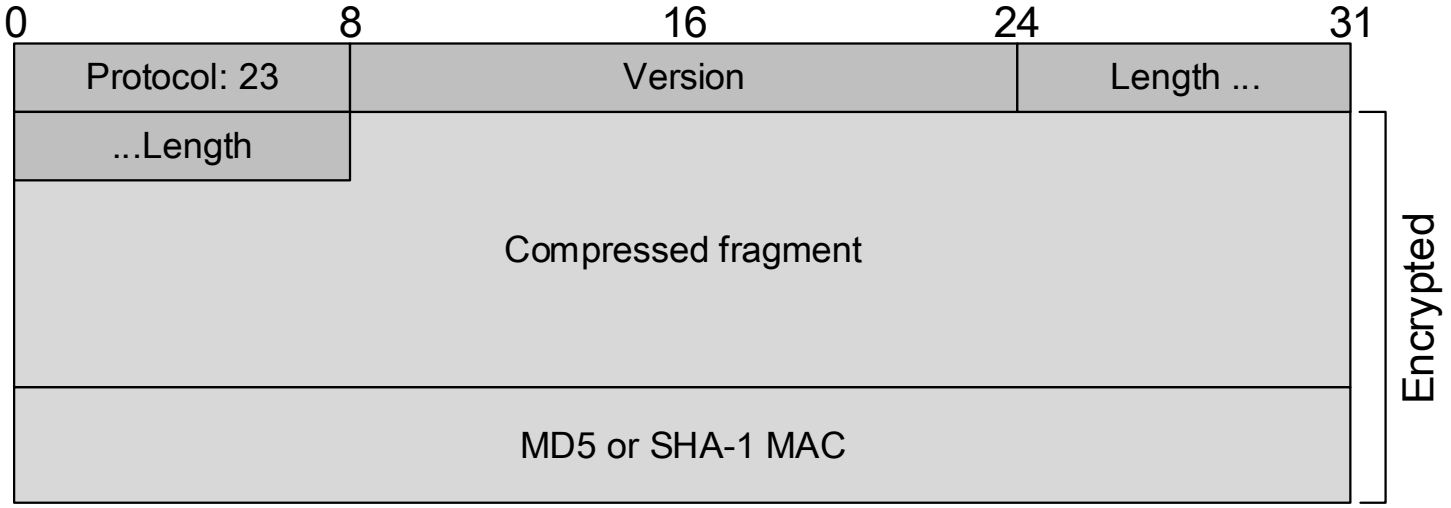
<그림 42> Finished 메시지 구성



<그림 43> Finished 메시지에 대한 해시 계산

SSL 메시지 형식

- 응용 데이터
  - Record 프로토콜은 응용 계층으로부터 오는 단편의 끝에 MAC 서명을 추가함
  - 프로토콜 값 23을 갖는 헤더를 추가한 후, Record 메시지가 보내짐



<그림 44> 응용 데이터에 대한 Record 프로토콜 메시지

목 차

- SSL 구조
- SSL 프로토콜
- SSL 메시지 형식
- 전송계층 보안(TLS, Transport Layer Security)

전송층 보안

- 개요
  - 암호화 및 인증을 지원하는 보안 통신 프로토콜
  - SSL 3.0의 후속으로 TLS 1.0이 등장하였음
  - SSL은 사용되지 않지만, SSL/TLS라는 용어를 많이 사용함
- SSL과의 차이점
  - 암호 그룹
  - 암호학적 비밀 생성
  - Alert 프로토콜
  - Handshake 프로토콜
  - Record 프로토콜

전송층 보안

- 암호 그룹(Cipher suite)
  - TLS는 Foretezza를 지원하지 않음
  - AES와 ECDHE (Elliptic Curve Diffie-Hellman Ephemeral, 타원곡선암호)가 추가됨

[표 6-1] TLS에서 사용되는 암호 그룹-1

Cipher suite	Key exchange	Encryption	Hash
TLS_NULL_WITH_NULL_NULL	NULL	NULL	NULL
TLS_RSA_WITH_NULL_MD5	RSA	NULL	MD5
TLS_RSA_WITH_NULL_SHA	RSA	NULL	SHA-1
TLS_RSA_WITH_RC4_128_MD5	RSA	RC4	MD5
TLS_RSA_WITH_RC4_128_SHA	RSA	RC4	SHA-1
TLS_RSA_WITH_IDEA_CBC_SHA	RSA	IDEA	SHA-1
TLS_RSA_WITH_DES_CBC_SHA	RSA	DES	SHA-1
TLS_RSA_WITH_3DES_EDE_CBC_SHA	RSA	3DES	SHA-1

전송층 보안

- 암호 그룹

[표 6-2] TLS에서 사용되는 암호 그룹-2

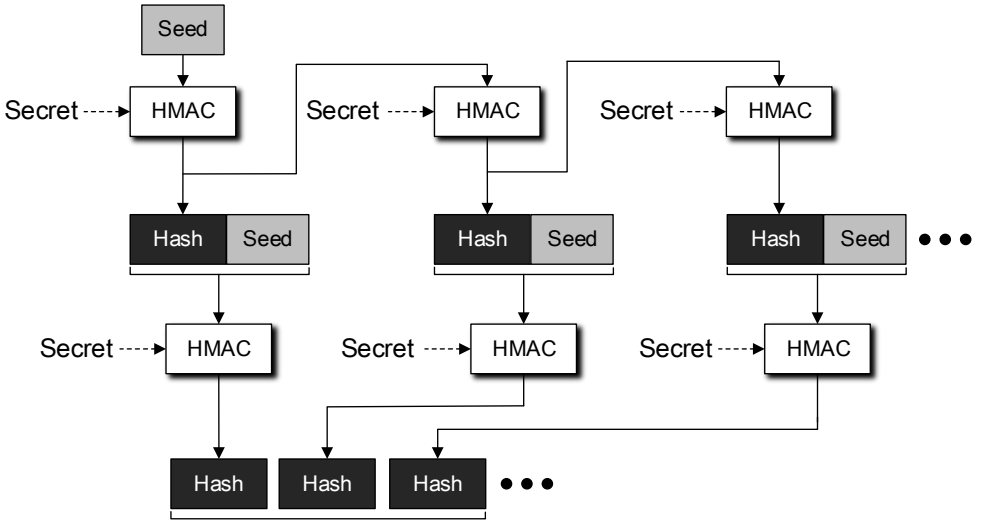
Cipher suite	Key exchange	Encryption	Hash
TLS_DH_anon_WITH_DES_CBC_SHA	DH_anon	DES	SHA-1
TLS_DH_anon_WITH_3DES_EDE_CBC_SHA	DH_anon	3DES	SHA-1
TLS_DHE_RSA_WITH_DES_CBC_SHA	DHE_RSA	DES	SHA-1
TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA	DHE_RSA	3DES	SHA-1
TLS_DHE_DSS_WITH_DES_CBC_SHA	DHE_DSS	DES	SHA-1
TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA	DHE_DSS	3DES	SHA-1
TLS_DH_RSA_WITH_DES_CBC_SHA	DH_RSA	DES	SHA-1
TLS_DH_RSA_WITH_3DES_EDE_CBC_SHA	DH_RSA	3DES	SHA-1
TLS_DH_DSS_WITH_DES_CBC_SHA	DH_DSS	DES	SHA-1
TLS_DH_DSS_WITH_3DES_EDE_CBC_SHA	DH_DSS	3DES	SHA-1

전송층 보안

- 암호학적 비밀 생성(Generation of Cryptographic Secrets)
  - 기존 SSL에서는 MD5와 SHA-1을 중첩하여 비밀을 계산함
  - TLS에서는 MD5와 SHA-1 말고도 다른 해시함수를 사용할 수 있도록 설정함
  - 종류
    - 데이터 확장 함수(Data-expansion function)
    - 의사난수 함수(PRF, PseudoRandom Function)

전송층 보안

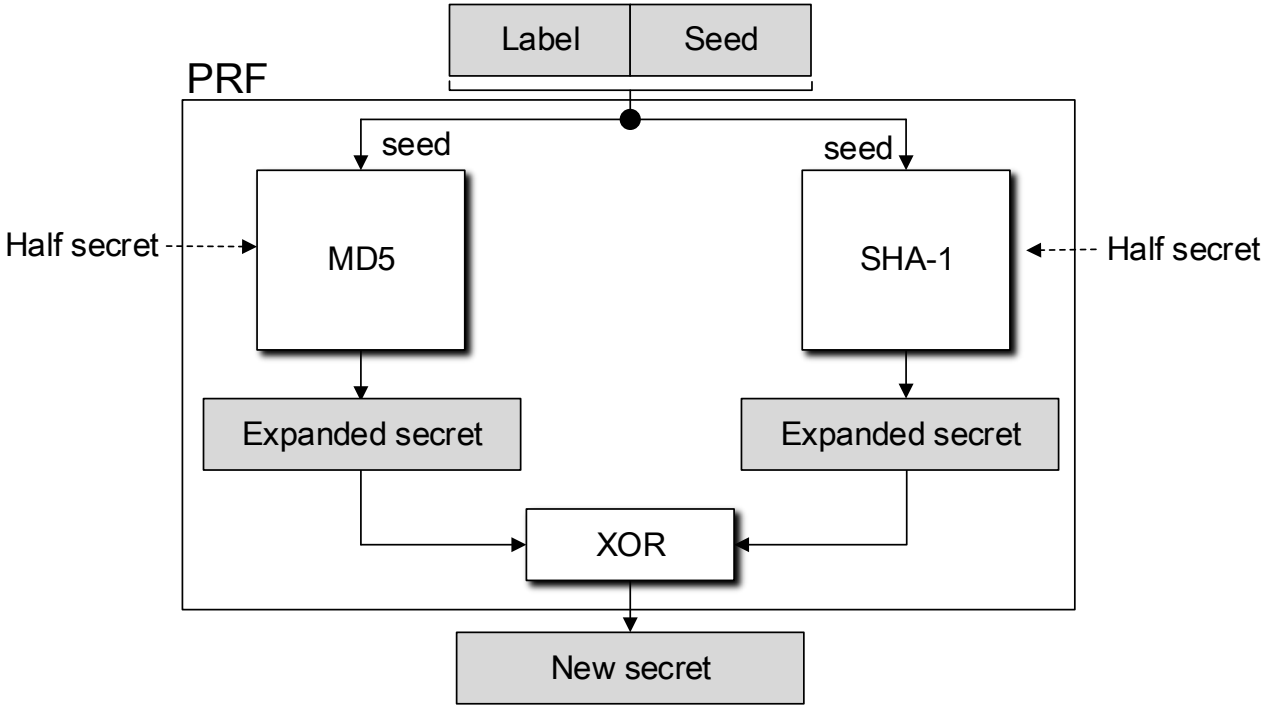
- 암호학적 비밀 생성
  - 데이터 확장 함수(Data-expansion function)
    - 짧은 비밀을 원하는 길이의 키 재료로 확장하기 위해 HMAC을 반복 적용하는 함수
      - HMAC은 해시 기반 MAC 방식임
    - Seed에 HMAC을 적용한 값에 다시 HMAC을 적용하여 두 번째 HMAC 출력값을 얻음
      - A(0)=seed 라고 하면, A(1)=HMAC(secret, A(0))
      - A(2)=HMAC(secret, A(1))



<그림 45> 데이터 확장 함수

전송층 보안

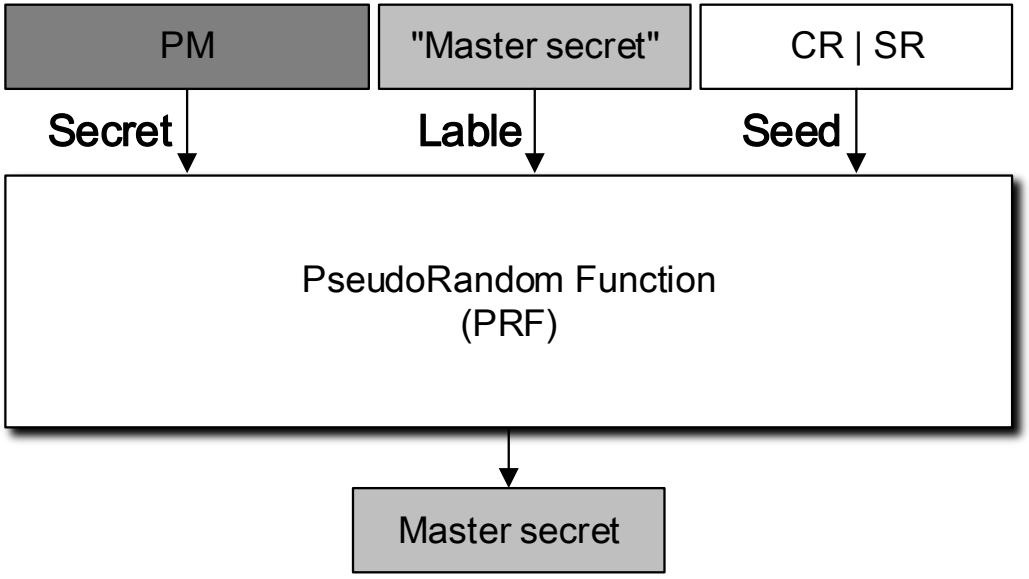
- 암호학적 비밀 생성
  - 의사난수 함수(PRF, PseudoRandom Function)
    - MD5와 SHA-1을 사용하는 2개의 데이터 확장 함수의 출력에 XOR로 결합하여 만든 함수
    - TLS 1.2부터는 SHA256만 사용하도록 변경



<그림 46> 의사난수 함수(PRF)

전송층 보안

- 암호학적 비밀 생성
  - 마스터 비밀(Master Secret)
    - 사전 마스터 비밀로부터 마스터 비밀을 생성하기 위해 의사난수 함수를 사용함

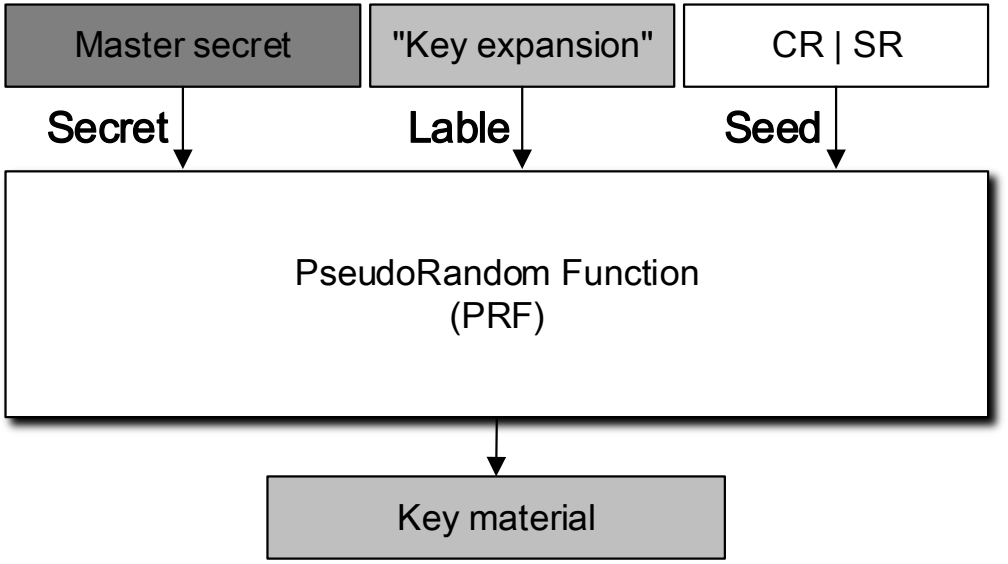


PM: Pre-Master secret
CR: Client Random number
SR: Server Random number
|: Concatenation

<그림 47> 마스터 비밀 생성과정

전송층 보안

- 암호학적 비밀 생성
  - 키 재료(Key Material)
    - 마스터 비밀로부터 키 재료를 생성하기 위해 의사난수 함수를 사용함



CR: Client Random number
SR: Server Random number
|: Concatenation

<그림 48> 키 재료 생성 과정

전송층 보안

- Alert 프로토콜
  - SSL에 비해서 경고 메시지가 다양해짐
    - 클라이언트가 인증서가 없는 건 정상 상황일 수도 있어서 NoCertificate경고를 삭제함
    - UnknownCA
      - 인증서 체인 검증이 실패함을 알림
    - AccessDenied
      - 인증서는 유효하지만 정책상 거부함을 알림
        - e.g., 인증서의 주체가 접근 권한이 없을 때
    - DecodeError
      - 필드의 구조가 형식에 맞지 않음을 알림
        - e.g., 메시지 길이 필드가 실제와 다름, 필드 값이 허용 범위를 벗어남 등
    - DecryptError
      - 암호 연산에 실패했음을 알림

전송층 보안

- Alert 프로토콜
  - SSL에 비해서 경고 메시지가 다양해짐
    - ExportRestriction
      - 미국이 강한 암호를 무기로 분류해서, 미국 외에 수출을 금지함
    - ProtocolVersion
      - TLS 버전 협상에 실패했음을 알림
      - 공격자가 연결을 다운그레이드하는 것을 방지함
    - InsufficientSecurity
      - 서버가 클라이언트의 cipher suite List를 보고 보안 기준 미달임을 알림
    - InternalError
      - 상대방 문제가 아니고 본인의 문제임을 알림
    - UserCanceled
      - 사람이 Handshake를 중단했음을 알림
    - NoRenegotiation
      - 이미 연결된 상태에서 다시 handshake를 하지 않겠다는 것을 알림

전송층 보안

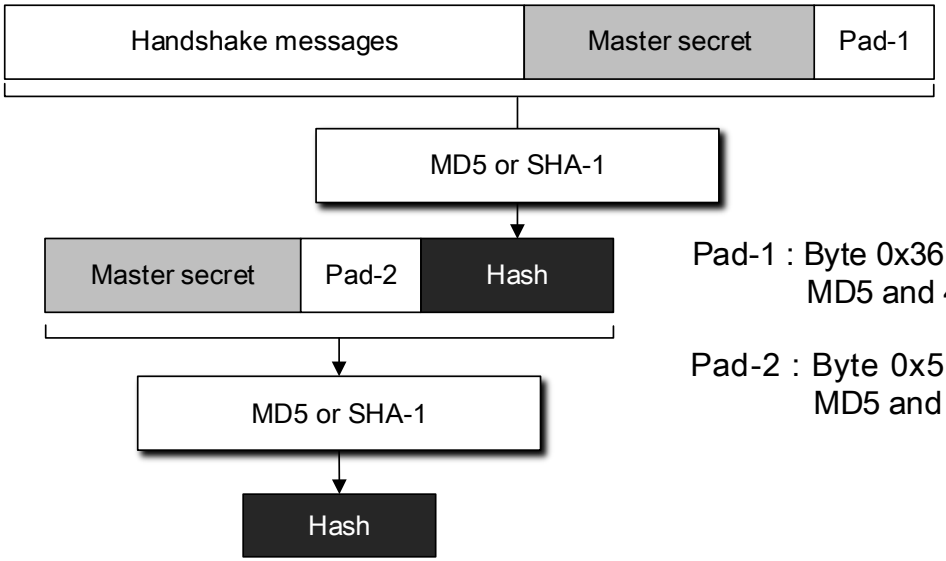
- Alert 프로토콜
  - SSL에 비해서 경고 메시지가 다양해짐

[표 7] TLS에서 정의된 경고

value	Description	Meaning
48	UnknownCA	CA could not be identified.
49	AccessDenied	No desire to continue with negotiation.
50	DecodeError	Received message could not be decoded.
51	DecryptError	Decrypted ciphertext is invalid.
60	ExportRestriction	Problem with U.S. restriction compliance.
70	ProtocolVersion	The protocol version is not supported.
71	InsufficientSecurity	More secure cipher suite needed.
80	InternalError	Local error.
90	UserCanceled	The party wishes to cancel the negotiation.
100	NoRenegotiation	The server cannot renegotiate the handshake.

전송층 보안

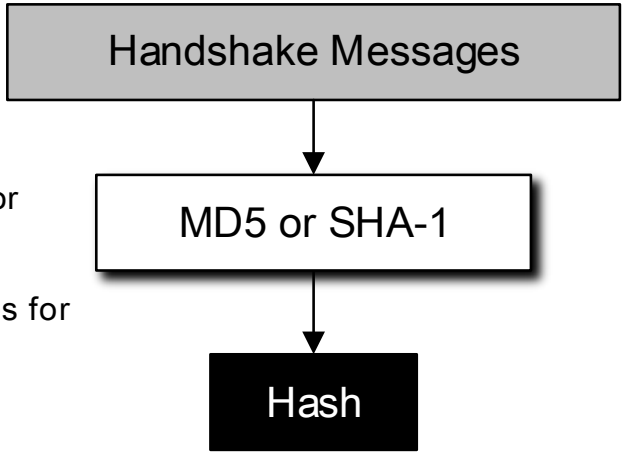
- Handshake 프로토콜
  - CertificateVerify
    - SSL에서는 Handshake 메시지와 마스터 비밀을 이용한 이중 해시값이였음
    - TLS에서는 Handshake 메시지만을 이용해 처리과정을 단축함



Pad-1 : Byte 0x36 (repeated 48 times for MD5 and 40 times for SHA-1)

Pad-2 : Byte 0x5C (repeated 48 times for MD5 and 40 times for SHA-1)

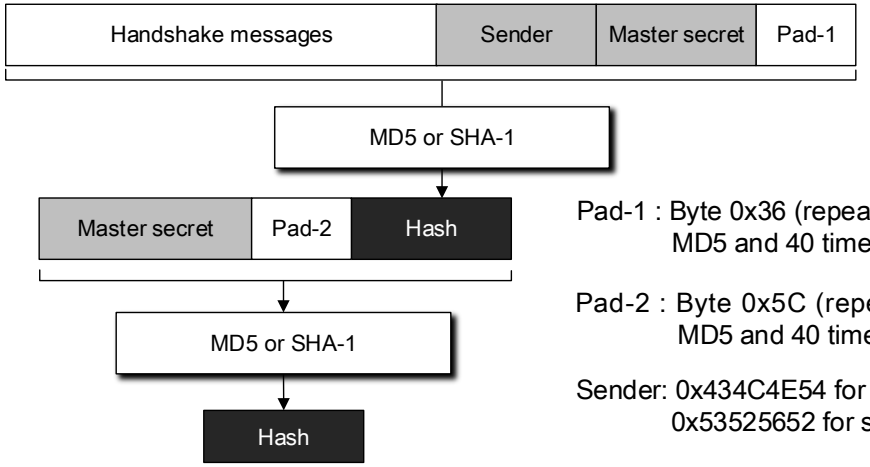
<그림 40> SSL 에서의 CertificateVerify 메시지에 대한 해시 계산



<그림 49> TLS에서 인증서 검증 메시지에 대한 해시

전송층 보안

- Handshake 프로토콜
  - Finished
    - SSL에서는 CertificateVerify와 Finished가 거의 같은 해시 구성을 공유함
    - TLS에서는 handshake 메시지에 MD5, SHA-1 해시를 적용하고 마스터 비밀과 라벨에 이어 붙인 후 의사난수 함수를 이용해 계산함

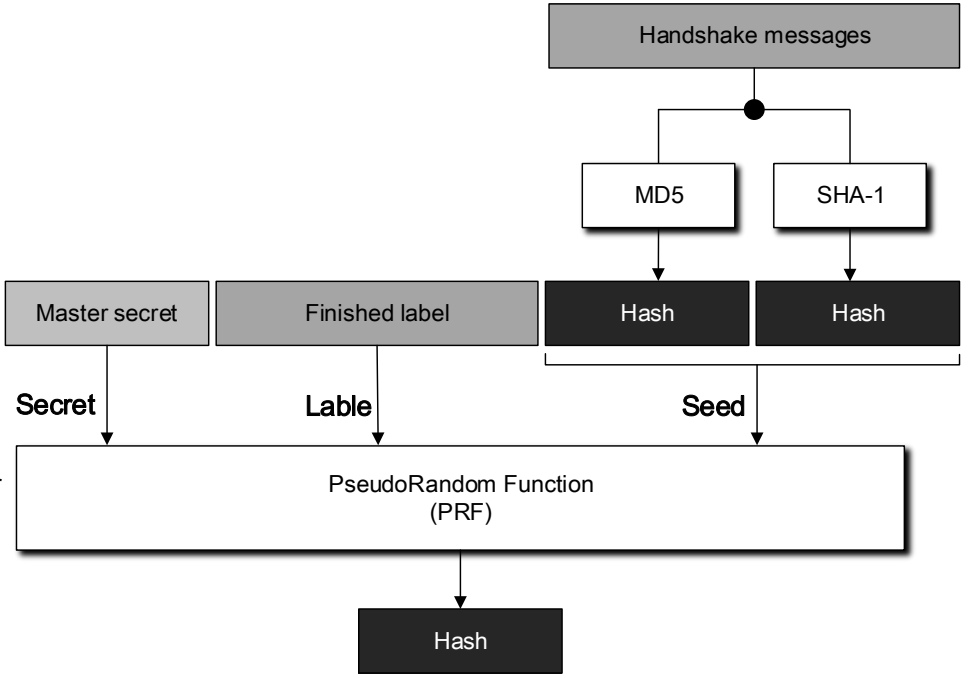


Pad-1 : Byte 0x36 (repeated 48 times for MD5 and 40 times for SHA-1)

Pad-2 : Byte 0x5C (repeated 48 times for MD5 and 40 times for SHA-1)

Sender: 0x434C4E54 for client; 0x53525652 for server

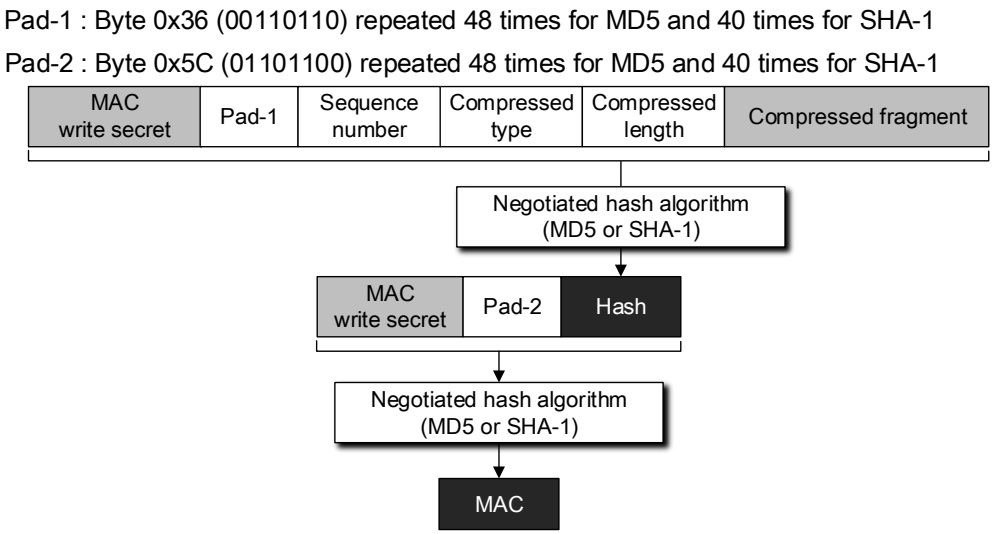
<그림 43> SSL에서 Finished 메시지에 대한 해시 계산



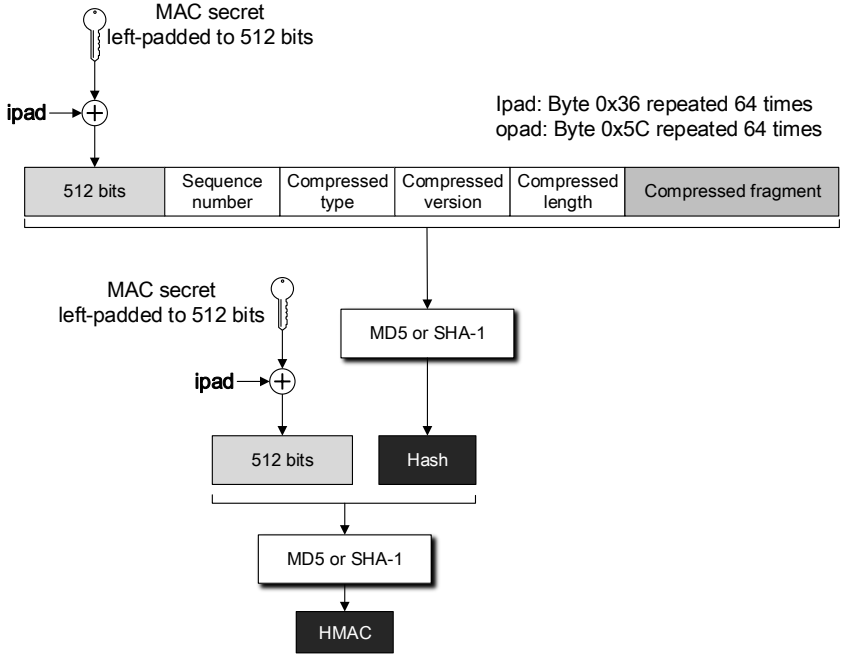
<그림 50> TLS에서 Finished 메시지에 대한 해시

전송층 보안

- Record 프로토콜
  - SSL
    - 키 뒤에 패딩을 이어 붙인 값으로 해시를 두 번 적용하는 구조를 가짐
  - TLS
    - 키와 패딩을 XOR한 값으로 해시를 두 번 적용하는 HMAC 구조를 취함



<그림 27> SSL에서의 MAC 계산 과정



<그림 51> TLS에 대한 HMAC 계산 과정

Thanks!

민 윤 홍(yunhong@pel.sejong.ac.kr)