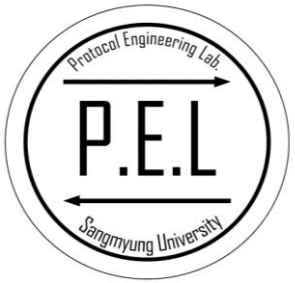


네트워크 보안 에센셜

(5장 전송 레벨 보안)

Sa-Rang Wi (sarang@pel.smuc.ac.kr)
Protocol Engineering Lab., Sangmyung University

Content

- 웹 보안
- SSL(안전소켓계층)/TLS(전송계층보안)
- HTTPS
- SSH

웹 보안

- 웹이란?
  - WWW(World Wide Web)
  - 인터넷과 TCP/IP 인트라넷상에서 운영하는 기본적인 클라이언트/서버 응용 프로그램
  - 웹 환경의 변화

- 웹 보안의 필요성
  - 웹 보안 위협 증가
    - ✓ 웹 기반의 정보서비스, 전자상거래 활성화로 인한 고급 정보 및 직접적인 이득 획득 기회 증가
  - 웹 보안 취약성 증가
    - ✓ HTTP 프로토콜 보안 취약

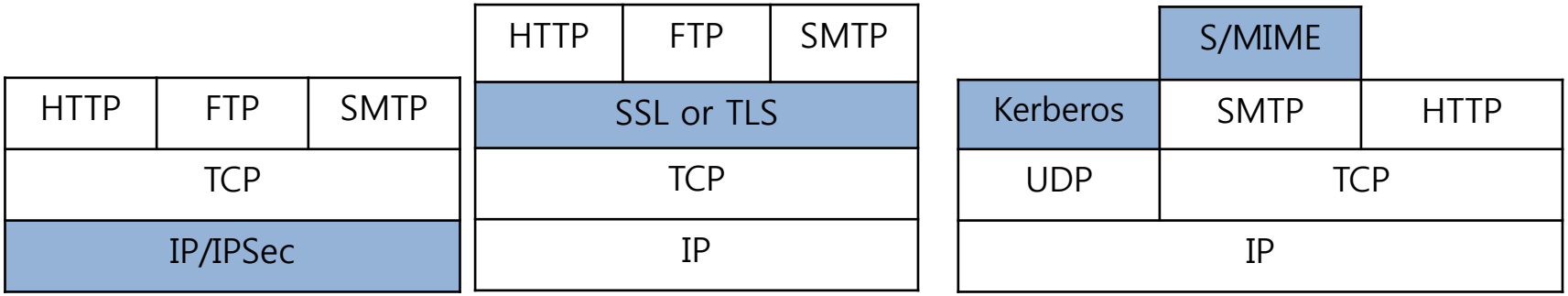
웹 보안

- 웹 사용시 나타날 수 있는 보안 위협의 유형

	위협	피해사항	대응방법
무결성	• 사용자 데이터 변경 • 트로이 목마 브라우저 • 메모리 변경 • 전송중 메시지 트래픽 변경	• 정보의 손실 • 기계에 대한 침해 • 다른 위협에 대한 취약성	암호적 검사합
기밀성	• Net 도청 • 서버에서 정보 훔치기 • 클라이언트에서 데이터 훔치기 • 네트워크 구성에 대한 정보 알아내기 • 서버와 통신중인 클라이언트 알아내기		암호화, 웹 프로시
서비스 거부	• 사용자 스레드 중단시키기 • 메모리나 디스크영역 차지하기	• 사용자의 작업 방해	예방힘듬
인증	• 합법적 사용자로 위조 • 데이터 위조	• 사용자에 대한 식별 오류 • 가짜 정보를 진짜로 오인	암호적 기술

웹 보안

- 웹 보안 방법



(a) 네트워크 레벨 (b) 전송 레벨 (c) 응용 레벨

➢ 네트워크 레벨 보안

✓ IP 보안

➢ 전송 레벨 보안

✓ SSL / TLS

➢ 응용 레벨 보안

✓ 프로그램 별 보안

Content

- 웹 보안
- SSL(안전소켓계층)/TLS(전송계층보안)
- HTTPS
- SSH

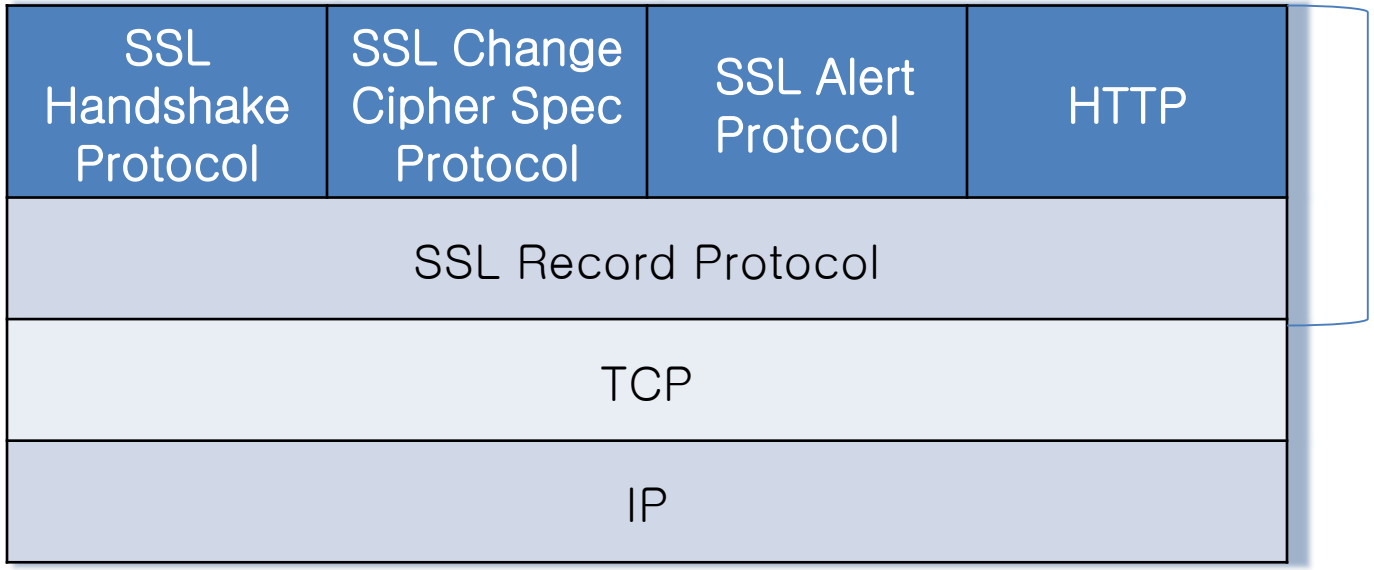
SSL(안전소켓계층)/TLS(전송계층보안)

- SSL(Secure Sockets Layer)
  - 배경 : 1933년 웹 서버와 브라우저간의 안전한 통신을 위해 넷 스케이프 사에서 개발
  - 기존의 TCP/IP에서 고려하지 않은 보안 세션을 추가하여 메시지에 대한 암호화와 상호인증을 수행하는 프로토콜

- TLS(Transport Layer Security)
  - SSL을 기반으로 만든 기술
  - IETF(Internet Engineering Task Force) 표준

SSL(안전소켓계층)/TLS(전송계층보안)

- SSL 구조



- ➢ SSL 핸드쉐이크 프로토콜
- ➢ SSL 레코드 프로토콜
- ➢ SSL 암호명세변경 프로토콜
- ➢ SSL 경고 프로토콜

SSL(안전소켓계층)/TLS(전송계층보안)

- SSL 세션과 연결
  - ➢ SSL 세션
    - ✓ 클라이언트와 서버 사이의 연결
    - ✓ 핸드쉐이크 프로토콜에 의해 생성
    - ✓ 다수의 연결이 공유하는 암호적 보안 매개변수 정의
    - ✓ 각 연결마다 해야하는 새 보안 매개변수 협상을 피하기 위해 사용
  - ➢ SSL 세션의 파라미터
    - ✓ 세션 식별자:서버가 선택하는 임의의 바이트열
    - ✓ 인증서:서버와 클라이언트의 X509.v3인증서
    - ✓ 압축 방법:암호화를 하기 전 압축에 사용하는 알고리즘
    - ✓ 암호명세(Ciper spec):MAC 계산에 사용되는 용량이 큰 데이터에 대한 암호 알고리즘 및 해시 알고리즘
    - ✓ Master secret: 클라이언트와 서버가 공유하는 48bytes 비밀값

SSL(안전소켓계층)/TLS(전송계층보안)

- SSL 세션과 연결
  - SSL 연결
    - ✓ Peer to Peer 관계
    - ✓ 일시적이고 세션과 연관되어있음
  - SSL 연결의 파라미터
    - ✓ 서버/ 클라이언트 **난수**: 연결에 사용되는 서버/클라이언트가 선택하는 임의의 바이트
      - ✓ 난수 : timestamp(4byte) + 랜덤넘버생성기로 만든 값(28byte)
    - ✓ **server write MAC secret**: 서버가 데이터에 MAC 연산 시 사용하는 비밀
    - ✓ **client write MAC secret**: 클라이언트가 데이터에 MAC 연산 시사용되는 비밀
    - ✓ **서버 키(server write key):** 서버가 데이터를 암호화하고 클라이언트가 복호화 할 때 사용하는 암호 키
    - ✓ **클라이언트 키(client write key):**클라이언트가 데이터를 암호화하고 서버가 복호화 할 때 사용하는 암호 키
    - ✓ **초기화 벡터**: Server와 Client가 각각 CBC모드의 블록 암호화에 사용되는 초기값
    - ✓ 순서번호(Sequence numbers): 송수신 되는 각 메시지에 대한 순서번호.

SSL(안전소켓계층)/TLS(전송계층보안)

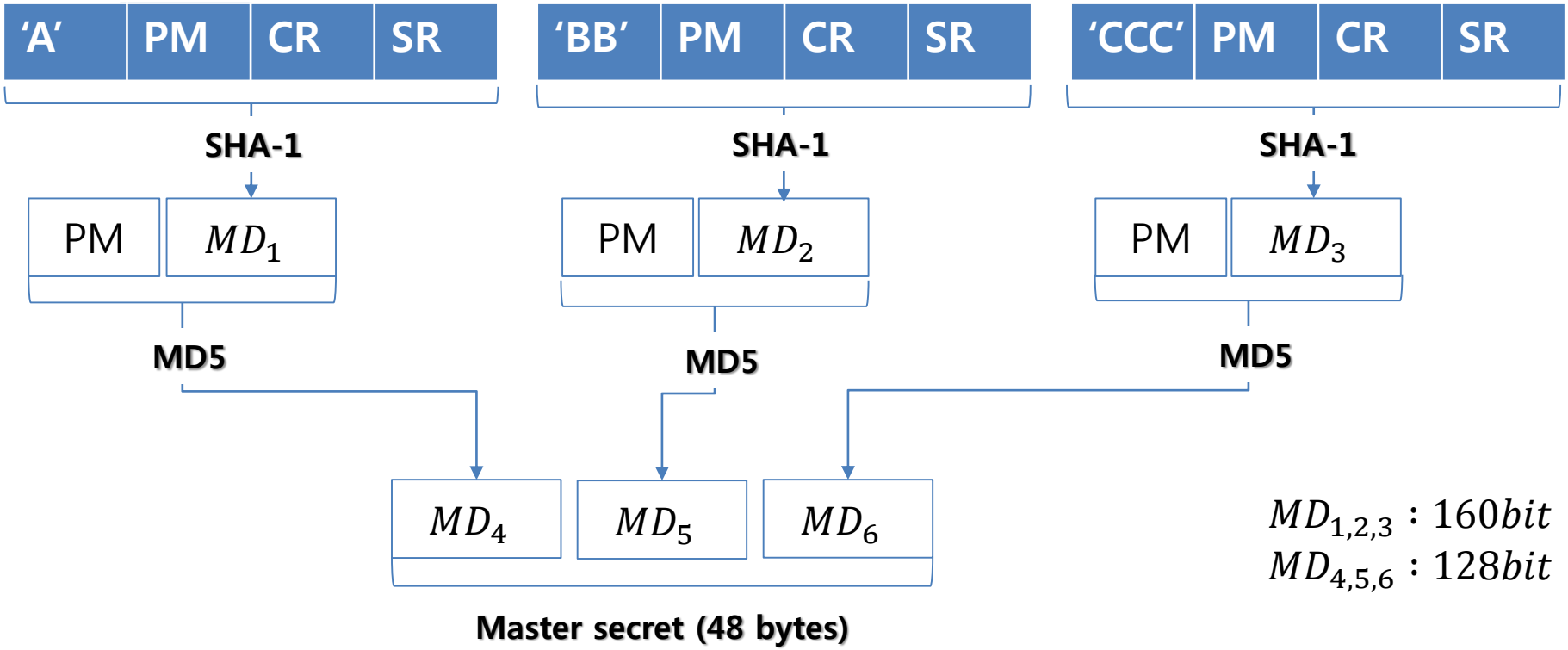
- 키 교환 알고리즘(서버와 클라이언트간의 같은 pre-master secret 필요)
  - 핸드쉐이크프로토콜 내에서 일어나는 키교환 시 사용
  - 알고리즘의 종류
    - ✓ RSA
      - 클라이언트가 생성한 pre-master secret 서버의 공개키로 암호화해서 서버에게 전송
      - Pre-master secret : client_version(2bytes)+랜덤넘버생성기로 만든 값(46byte)
    - ✓ 익명(anonymous) 디피헬만 : 기본 DH 알고리즘
      - 인증서 교환x
      - g, p, g^x를 평문으로 보냄
      - 상대방을 모르기 때문에 중간자 공격에 취약
      - Pre-master secret : $g^{cs} mod\ p$

SSL(안전소켓계층)/TLS(전송계층보안)

- 키 교환 알고리즘(서버와 클라이언트간의 같은 pre-master secret 필요)
  - ✓ 임시(ephemeral) 디피헬만
    - 중간자 공격을 막기 위해 설계
    - 양측은 개인키로 서명된 g, p, g^x를 보냄 ; 수신자는 송신자의 공개키를 사용해 서명 검증
    - Pre-master secret :$g^{cs} mod\ p$
  - ✓ 고정(fixed) 디피헬만
    - 중간자 공격을 막기 위해 설계
    - 각 개체는 고정 DH 매개변수(g, p), g^x를 생성할 수 있는 메커니즘을 가짐
      ->인증서의 내용에 g^x 포함(g^x를 인증기관의 개인키로 서명하여 인증)
    - RSA 혹은 DSS 인증서 활용
    - Pre-master secret :$g^{cs} mod\ p$

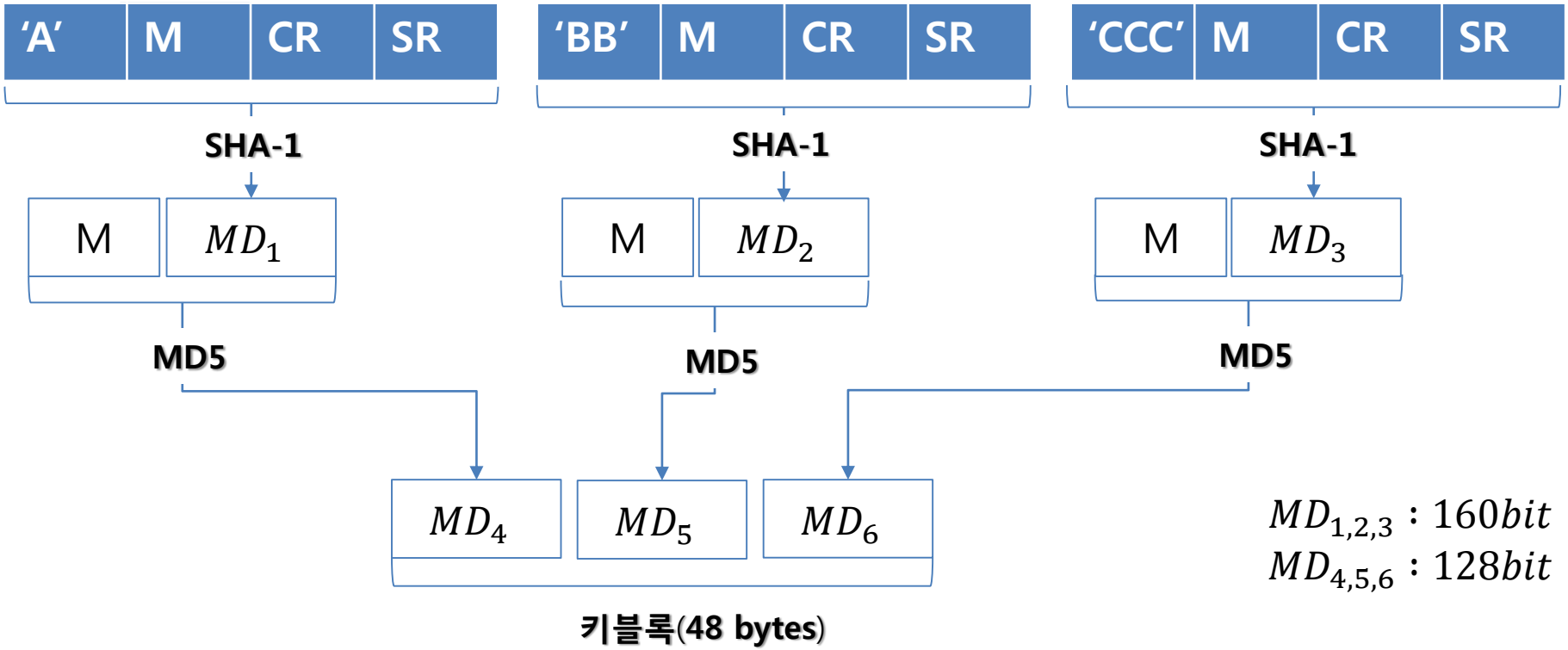
SSL(안전소켓계층)/TLS(전송계층보안)

- Premaster secret으로 master secret 계산
  - PM : pre-master secret
  - CR/SR : client/server 난수



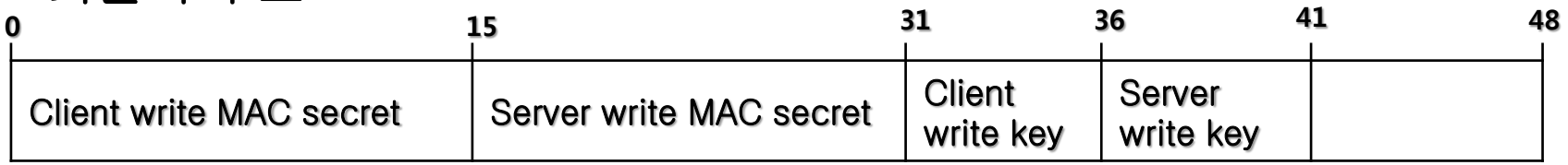
SSL(안전소켓계층)/TLS(전송계층보안)

- master secret 이용해 키 블록생성
  - M : master secret
  - CR/SR : client/server 난수

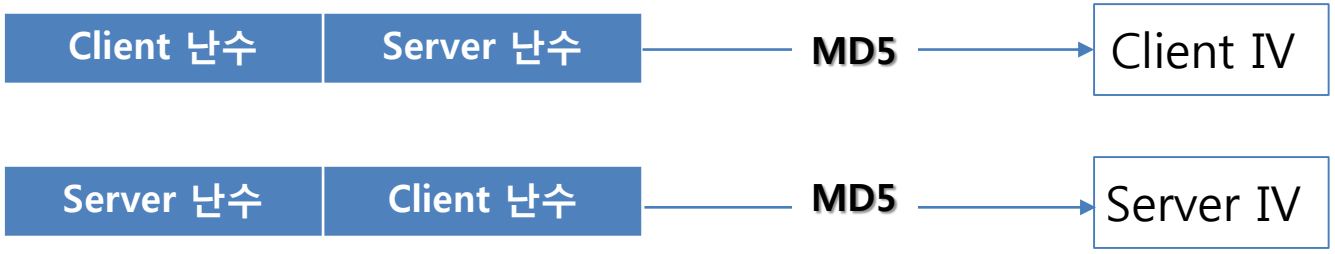


SSL(안전소켓계층)/TLS(전송계층보안)

- 키블록 구조

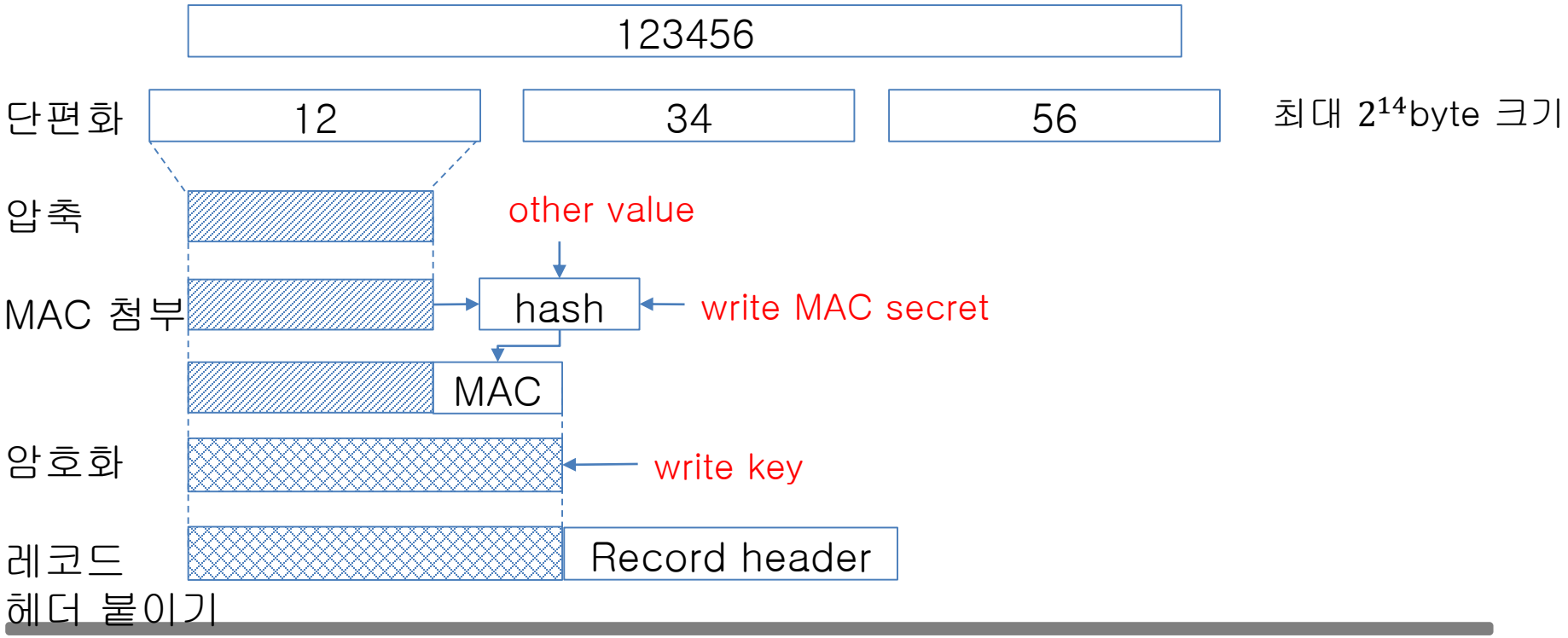


- Client/Server 초기화 벡터 IV



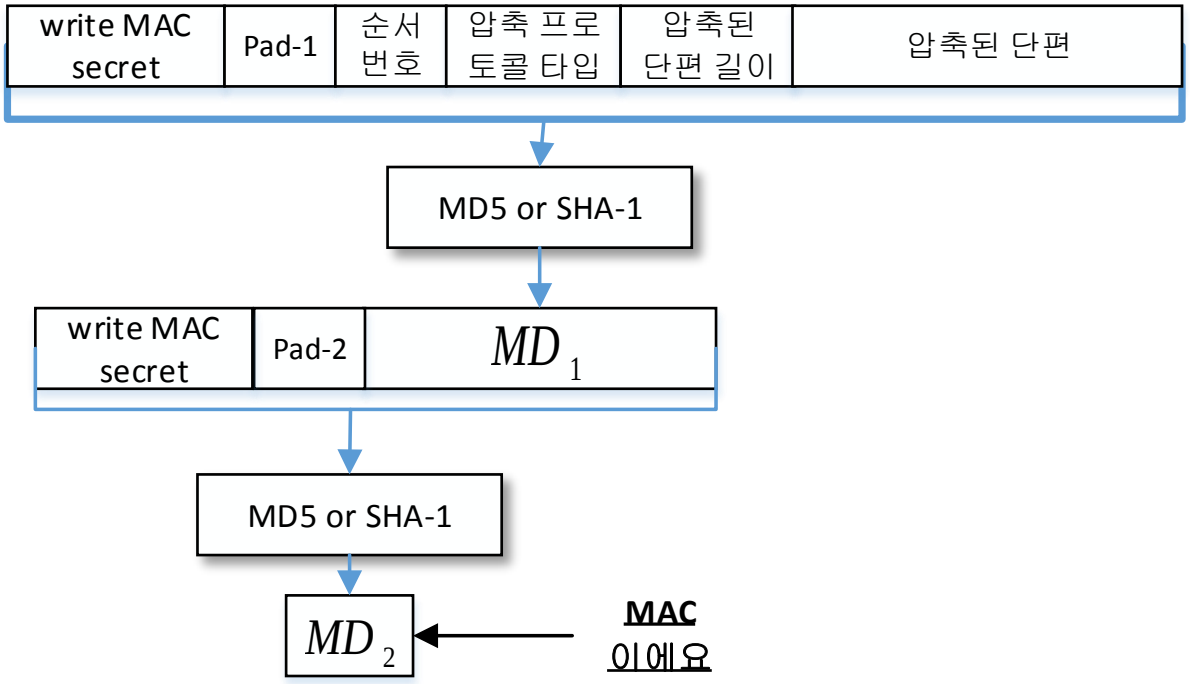
SSL(안전소켓계층)/TLS(전송계층보안)

- SSL 레코드 프로토콜(Record Protocol)
  - 상위 계층(핸드쉐이크등)으로부터 오는 메시지 TCP에 전달
  - 기밀성, 무결성 제공
  - 전반적인 동작 과정



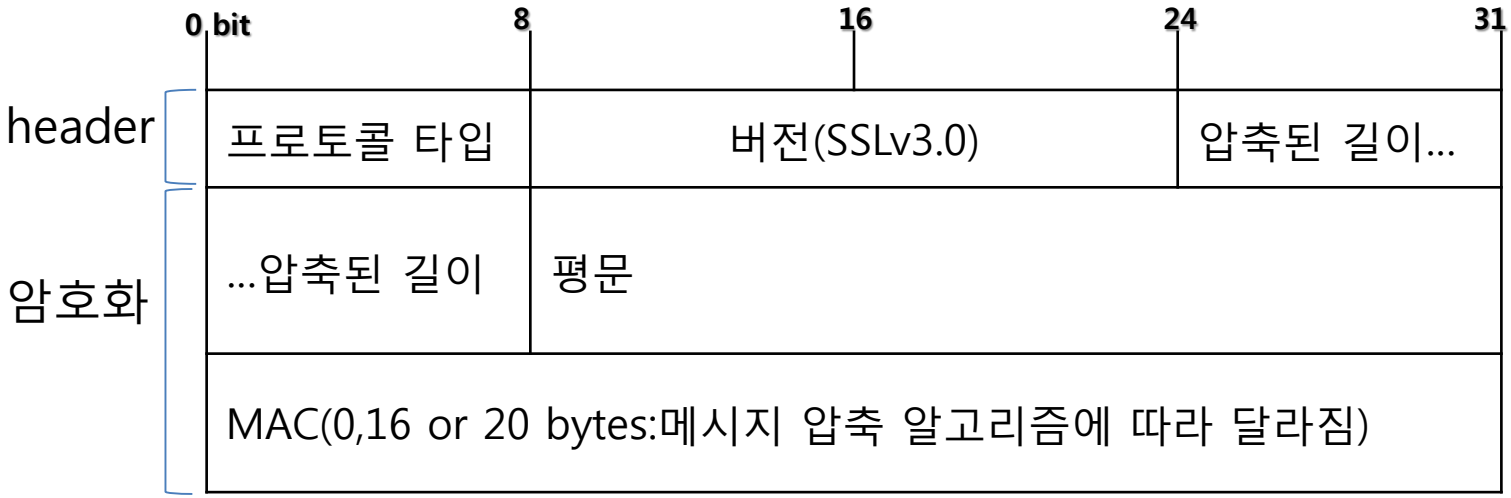
SSL(안전소켓계층)/TLS(전송계층보안)

- SSL 레코드 프로토콜(Record Protocol)
  - ➢ MAC 계산방법
    - ✓ Pad-1: Byte 0x36(00110100) MD5,48반복;SHA-1,40반복
    - ✓ Pad-2: Byte 0x5C(01011100) MD5,48반복;SHA-1,40반복



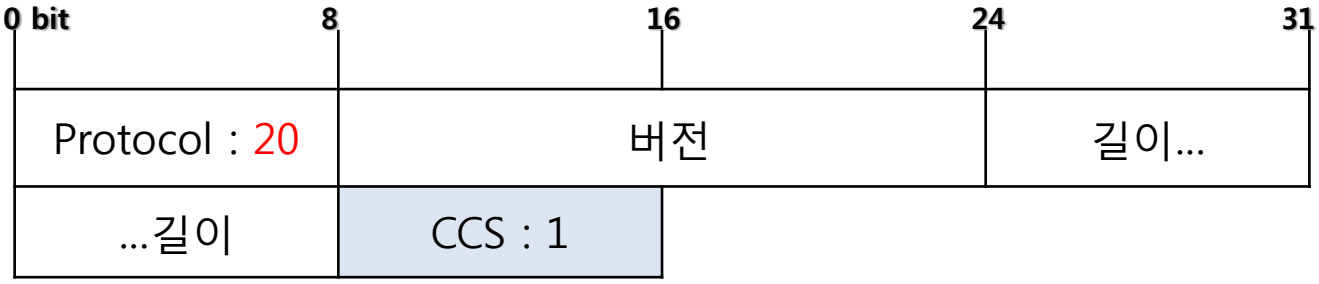
SSL(안전소켓계층)/TLS(전송계층보안)

- SSL 레코드 필드
  - ➢ 프로토콜 타입 : 캡슐화된 메시지의 발신지 또는 목적지 나타냄
    - ✓ 암호명세변경 프로토콜 : 20
    - ✓ 경고 프로토콜 : 21
    - ✓ 핸드쉐이크 프로토콜 : 22
    - ✓ 응용 층: 23



SSL(안전소켓계층)/TLS(전송계층보안)

- SSL 암호명세변경 프로토콜(Change Cipher Spec Protocol)
  - SSL-지정 프로토콜 중 하나
  - 1byte이고 값 1을 가지는 한개의 메시지로 구성됨
  - 핸드쉐이크 프로토콜을 통해 알게된 압축, MAC, 매개변수들을 사용가능함을 알리는 역할

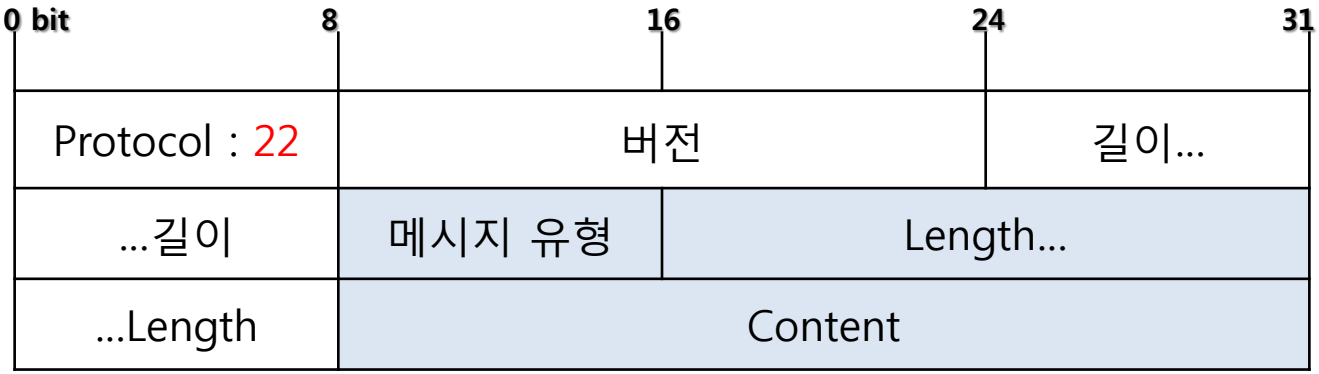


SSL(안전소켓계층)/TLS(전송계층보안)

- SSL 경고 프로토콜(Alert Protocol)
  - SSL-관련 경고를 할때 사용
  - 각 메시지는 2byte로 구성
    - ✓ 첫번째 바이트(LEVER): 경고(1),심각(2)
    - ✓ 두번째 바이트(Alert): 세부적인 에러코드
      - 핸드쉐이크, 암호명세변경, 레코드 프로토콜 수행 시 발생하는 오류메시지

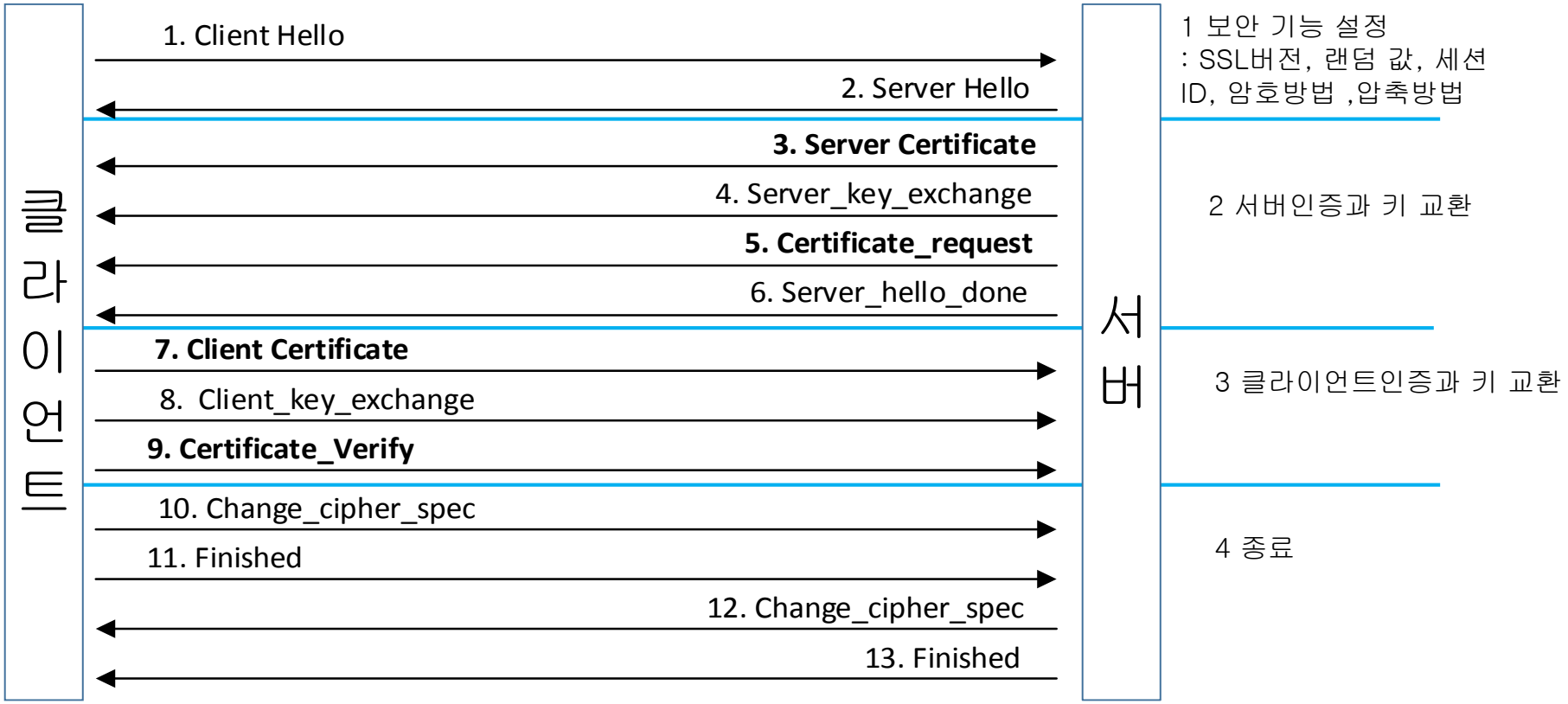
SSL(안전소켓계층)/TLS(전송계층보안)

- SSL 핸드쉐이크 프로토콜(Handshake Protocol)
  - ➢ 데이터를 전송하기 이전에 실행됨
    - ✓ 사용할 파라미터 협상
  - ➢ 필요 시 클라이언트가 서버에 대해 서버가 클라이언트에 대해 인증
    - ✓ 인증서
  - ➢ master secret 확립을 위한 정보를 교환하기 위해 사용



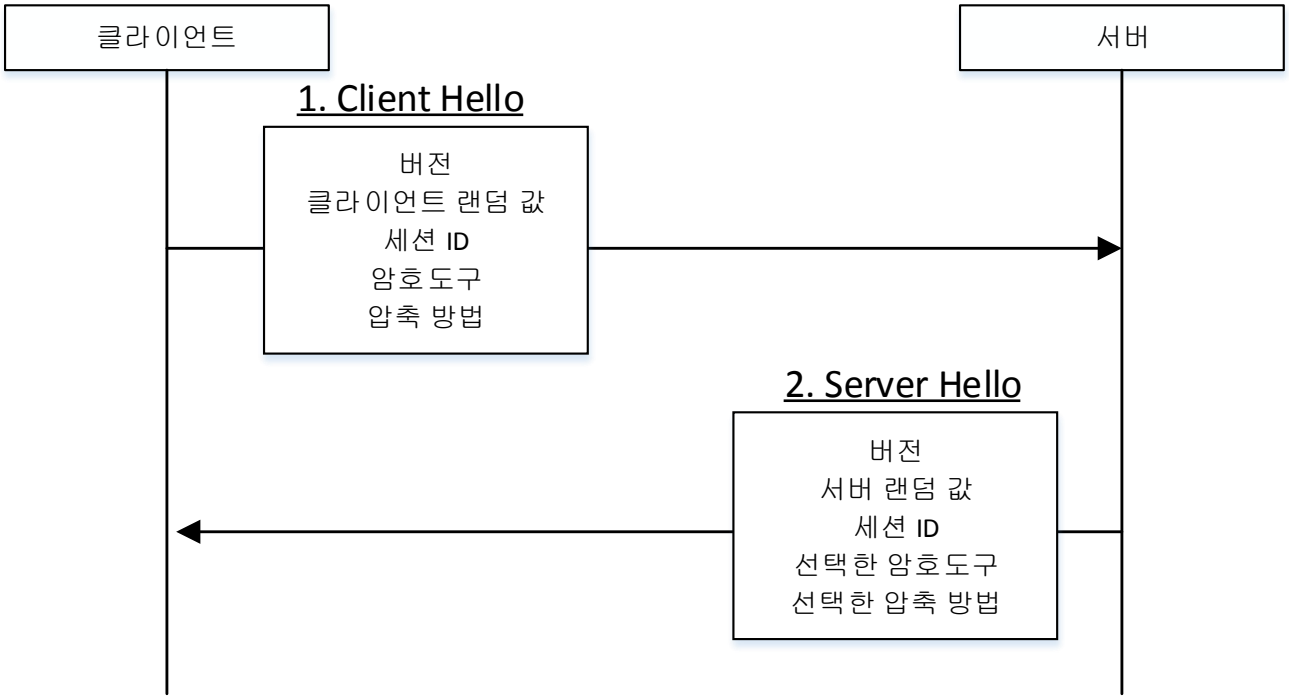
SSL(안전소켓계층)/TLS(전송계층보안)

- SSL 핸드쉐이크 프로토콜
  - 상호간의 세션 설정
  - 한 세션동안 이용되는 암호 매개변수를 생성



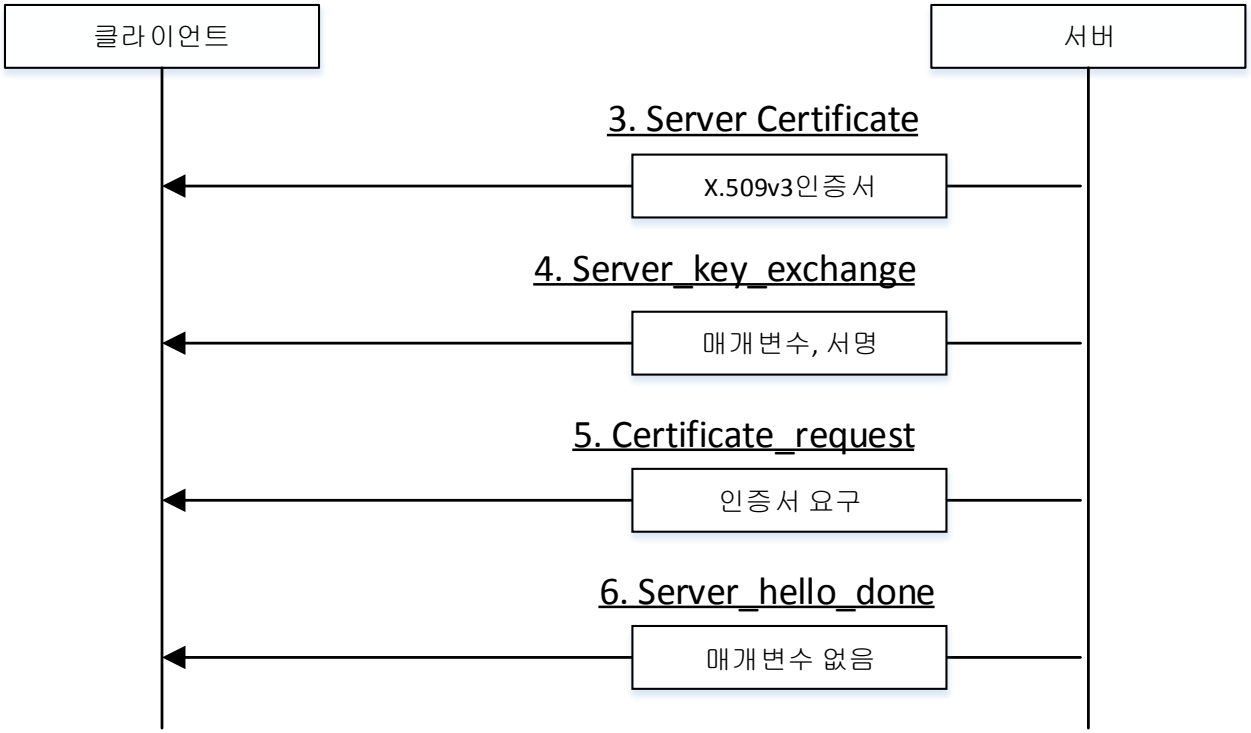
SSL(안전소켓계층)/TLS(전송계층보안)

- SSL 핸드쉐이크 프로토콜
  - 단계 1 보안 기능 설정[그림에서 1,2]



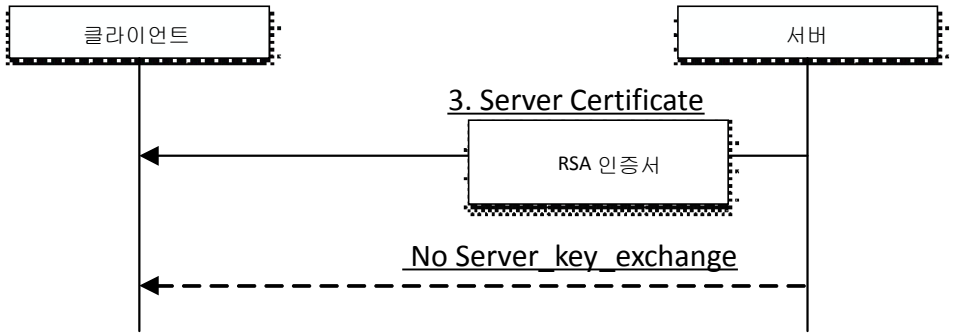
SSL(안전소켓계층)/TLS(전송계층보안)

- SSL 핸드쉐이크 프로토콜
  - 단계 2 서버인증과 키 교환[그림에서 3,4,5,6]

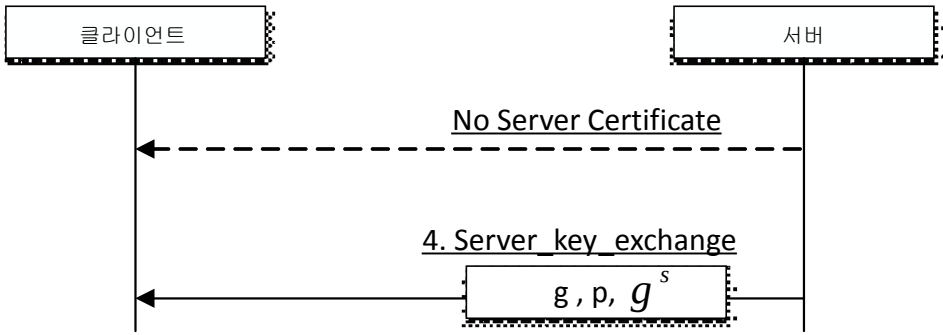


SSL(안전소켓계층)/TLS(전송계층보안)

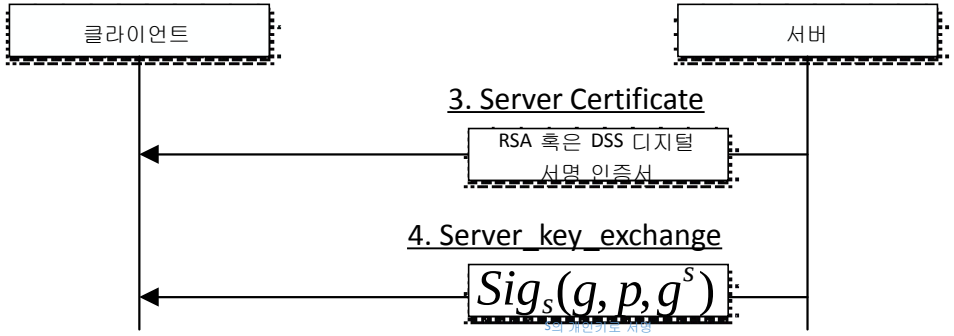
- SSL 핸드쉐이크 프로토콜
  - 단계 2 서버인증과 키 교환에서의 4가지 유형



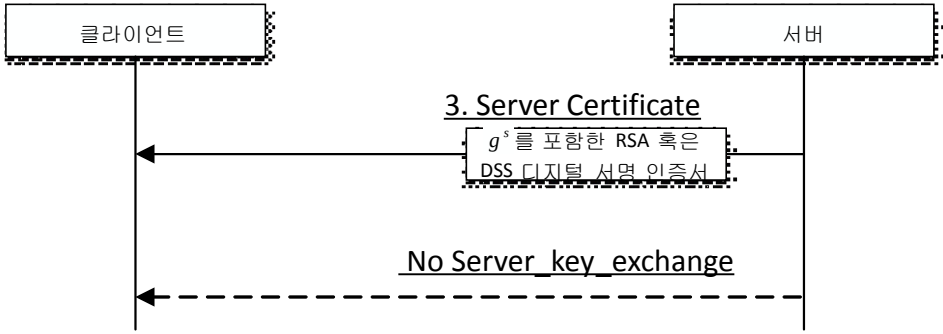
(a) RSA



(b) 익명 디피헬만



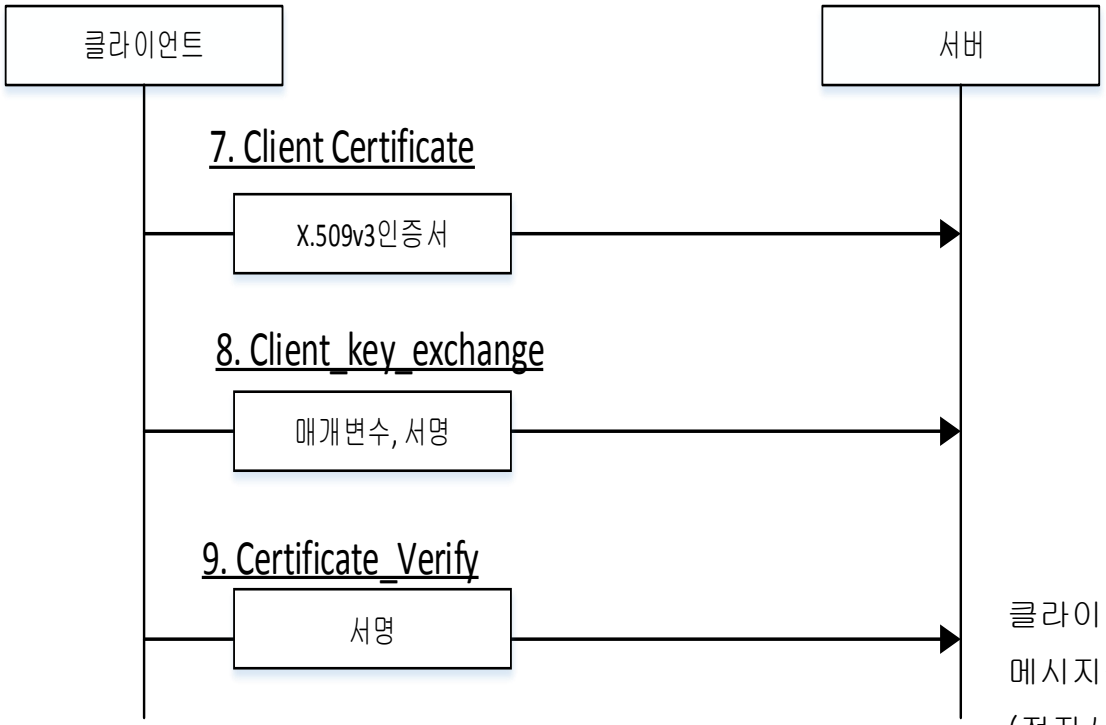
(c) 임시 디피헬만



(d) 고정 디피헬만

SSL(안전소켓계층)/TLS(전송계층보안)

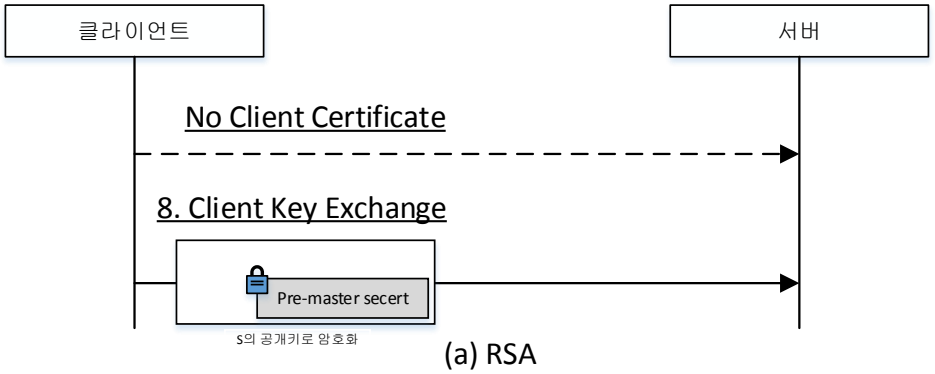
- SSL 핸드쉐이크 프로토콜
  - ➢ 단계 3 클라이언트 인증과 키교환



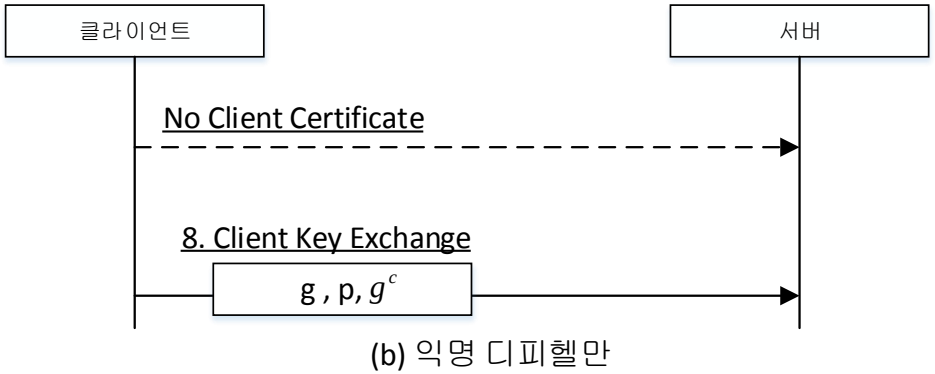
클라이언트가 자신의 인증서 유효성을 스스로 검증
메시지를 자신의 개인키로 서명해서 서버에게 보냄
(전자서명 서명기능이 있는 클라이언트 인증서 다음으로만)

SSL(안전소켓계층)/TLS(전송계층보안)

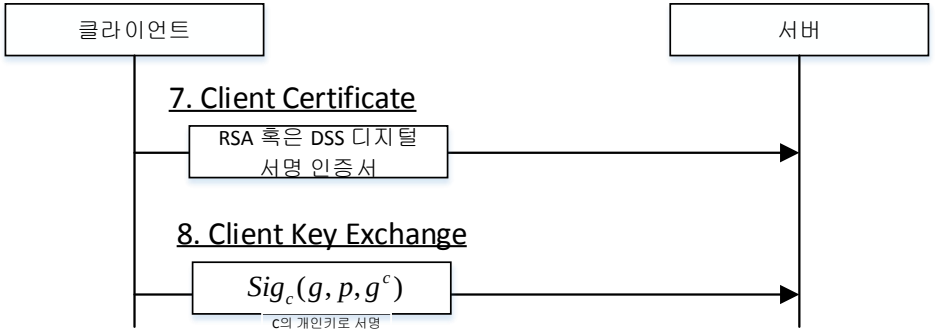
- SSL 핸드쉐이크 프로토콜
  - ➢ 단계 3 클라이언트 인증과 키 교환에서의 4가지 유형



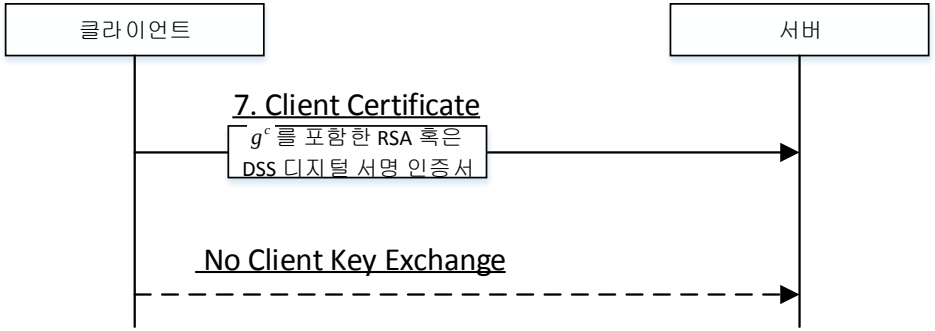
(a) RSA



(b) 익명 디피헬만



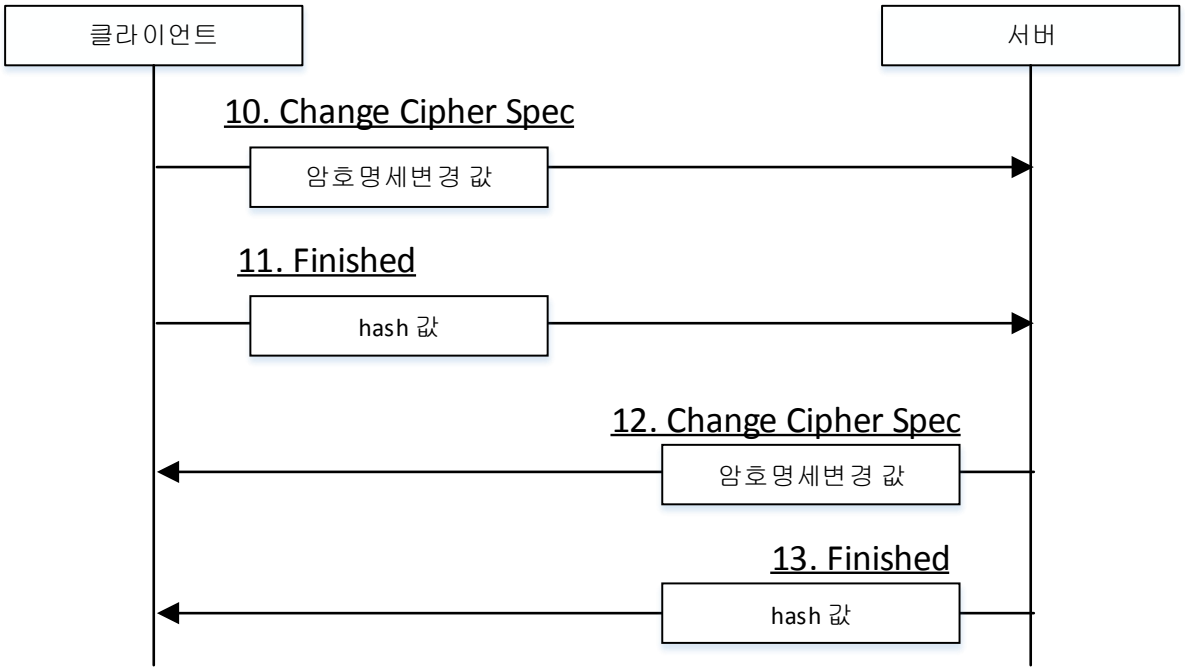
(c) 임시 디피헬만



(d) 고정 디피헬만

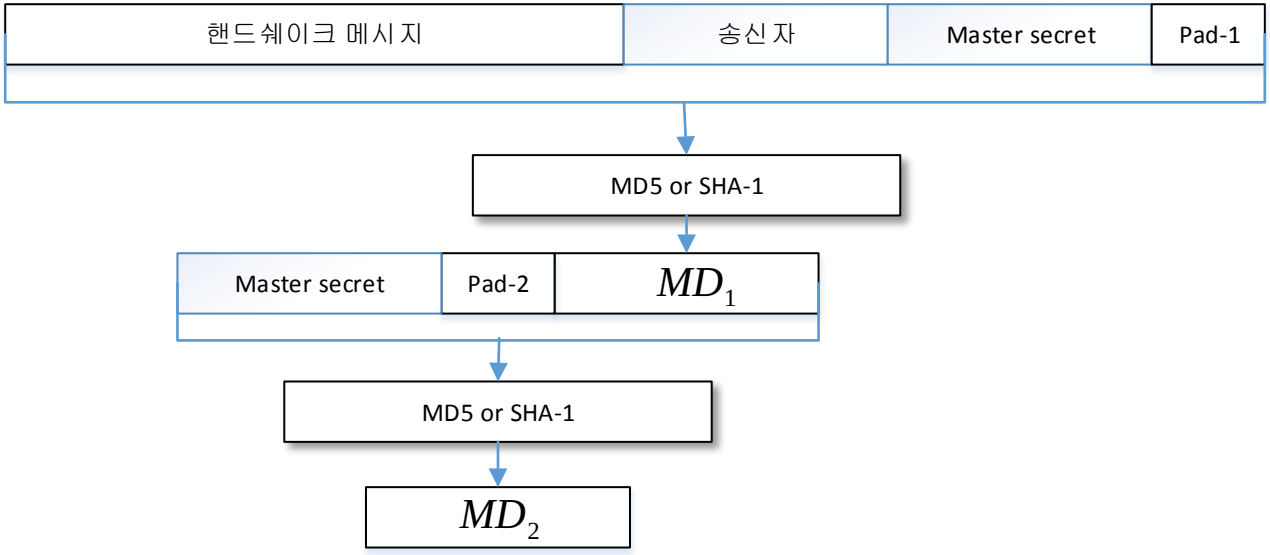
SSL(안전소켓계층)/TLS(전송계층보안)

- SSL 핸드쉐이크 프로토콜
  - ➢ 단계 4 종료
    - ✓ 안전한 연결 설정 종료
    - ✓ Finished 메시지는 키 교환과 인증 과정이 성공적으로 이루어짐 확인(해시값)
      - sender, master secret,핸드쉐이킹 중에 교환된 메시지 등



SSL(안전소켓계층)/TLS(전송계층보안)

- SSL 핸드쉐이크 프로토콜
  - ➢ 단계 4 종료
    - ✓ Finished 메시지에 대한 해시 계산
      - ✓ Pad-1: Byte 0x36(00110100) MD5,48반복;SHA-1,40반복
      - ✓ Pad-2: Byte 0x5C(01011100) MD5,48반복;SHA-1,40반복
      - ✓ 송신자 : 클라이언트 일때 0x434C4E54
        서버 일때 0x53525652

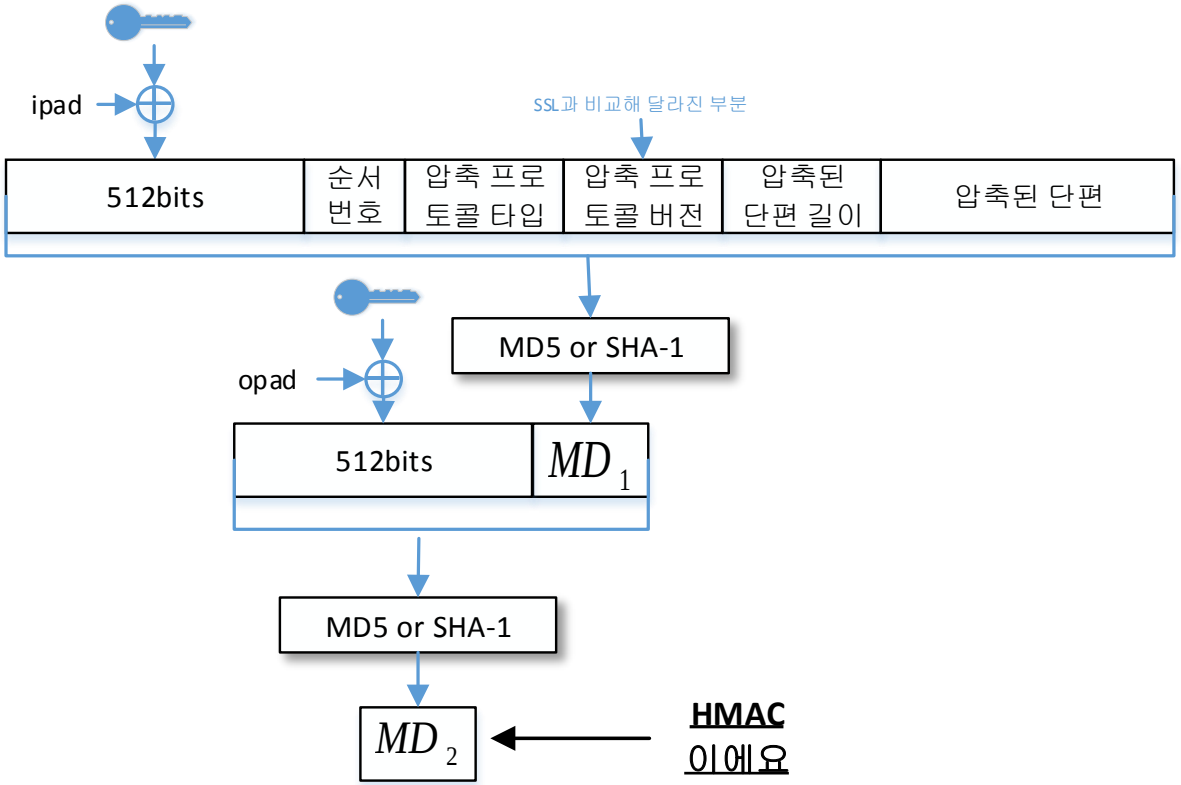


SSL(안전소켓계층)/TLS(전송계층보안)

- TLS(전송 계층 보안)
  - ➢ SSL의 IETF 표준버전
  - ➢ SSL과의 다른 점
    - ✓ 버전 번호
      - SSL버전번호 3.0 TLS 버전번호 1.0
    - ✓ master secret 계산식이 다름
      - PRF(의사난수함수, pseudorandom function)이용
      - 입력값 : secret(premaster secret), Lable(ASCII “Master secret”),Seed(Client 난수|Server 난수)
    - ✓ 핸드쉐이크 프로토콜 일부분
      - Finished 메시지에서 해시 계산을 위해 PRF 사용
      - 입력 값 : secret(Master secret), Lable(ASCII “Finished lable”), seed(핸드쉐이크 메시지를 각각 MD5,SHA-1한 해시값)
    - ✓ 메시지 인증 코드 : HMAC

SSL(안전소켓계층)/TLS(전송계층보안)

- TLS(전송 계층 보안)
  - ➢ 메시지 인증 코드 HMAC이란?
    - ✓ SHA-1같은 암호알고리즘을 이용해 출력한 해시값으로부터 MAC을 만드는 기술
  - ➢ TLS에서 HMAC계산 방법



MAC secret
left-padded to 512 bits
ipad : 0x36값(00110110)을 64번 반복
opad : 0x5C값(01011100)을 64번 반복

SSL(안전소켓계층)/TLS(전송계층보안)

- TLS의 단점
  - PKI와 같은 기능을 지원하기 위해 서버 및 클라이언트 필요.하지만 PKI 지원하지 않는 솔루션 및 클라이언트 존재
  - TCP만은 사용해야 하기 때문에 상당한 메모리 소비
  - 단지 hop-by-hop보안 임.

Content

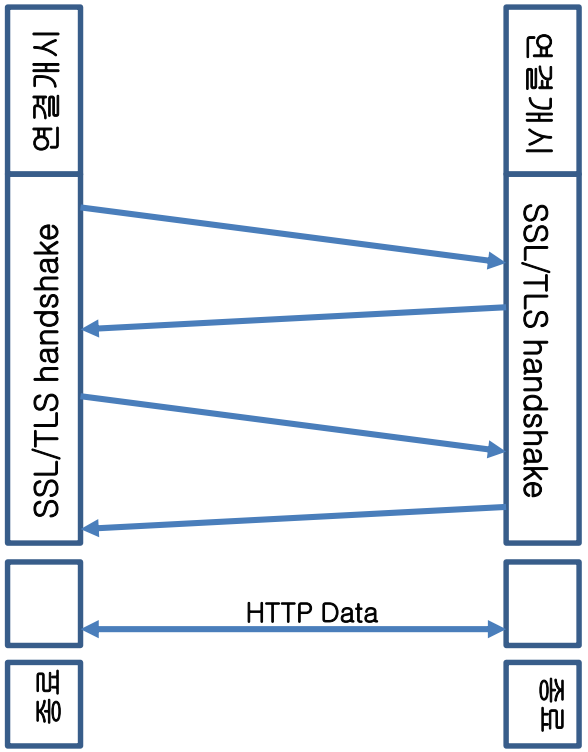
- 웹 보안
- SSL(안전소켓계층)/TLS(전송계층보안)
- HTTPS
- SSH

HTTPS

- HTTPS란?
  - 웹 서버와 웹 브라우저 간의 메시지 교환 프로토콜
  - 웹브라우저와 웹 서버간의 안전 통신을 구현하기 위한 HTTP와 SSL의 결합
    - ✓ Http는 단순 텍스트로 메시지가 교환됨(보안에 취약)
  - URL 주소가 http:// 가 아닌 https://로 시작
  - HTTP의 기본 포트 번호는 80, HTTPS의 기본 포트 번호는 443
  - 암호화 되는 통신 요소
    - ✓ URL
    - ✓ 문서내용
    - ✓ http 헤더 내용

HTTPS

- 연결 개시
  - ➢ HTTP클라이언트는 하위계층(TCP OR TLS/SSL)으로 연결요청 메시지를 보내어 HTTP서버에게 연결 요청함.
  - ➢ 연결 개시 후 동작
    - ✓ 적절한 포트를 통해 서버에 연결 시작
    - ✓ TLS 핸드쉐이크 시작 위해서 TLS ClientHello 전송
    - ✓ 핸드쉐이크 끝나면 첫 HTTP 요청시작
      - 모든 HTTP데이터는 TLS응용데이터
    - ✓ 이후 일반적 HTTP동작



HTTPS

- **연결 종료**
  - HTTP 연결 종료 전에 TLS 연결 종료를 먼저 해야 함
    - ✓ 종료하기 전에 상대방에게 경보를 보내 종료를 알림
    - ✓ 양쪽에서 close_notify경보를 보내는 TLS 경보 프로토콜 사용
  - HTTP 레코드에 Connection:close 집어넣으면 연결 종료
  - 비정상 종료상황
    - ✓ 하위 TCP연결이 사전 close_notify 경보와 Connection:close 지시자 없이 종료되는 상황
    - ✓ 원인 : 서버프로그램 오류, TCP연결 중단을 야기시키는 통신 오류
    - ✓ 대처 : 보안경고 발령

Content

- 웹 보안
- SSL(안전소켓계층)/TLS(전송계층보안)
- HTTPS
- SSH

SSH(Secure Shell)

- SSH 란?
  - 네트워크 상에서 다른 컴퓨터에 로그인하거나 원격 시스템에서 명령을 실행하고 다른 시스템으로 파일을 복사할 수 있도록 해주는 안전한 네트워크 통신용 프로토콜
  - 암호화되지 않은 기존의 Telnet등을 대체하기 위해 설계
  - 클라이언트/서버 구조
  - TCP에서의 보안 채널
  - 포트번호 : 22

SSH(Secure Shell)

- SSH 프로토콜

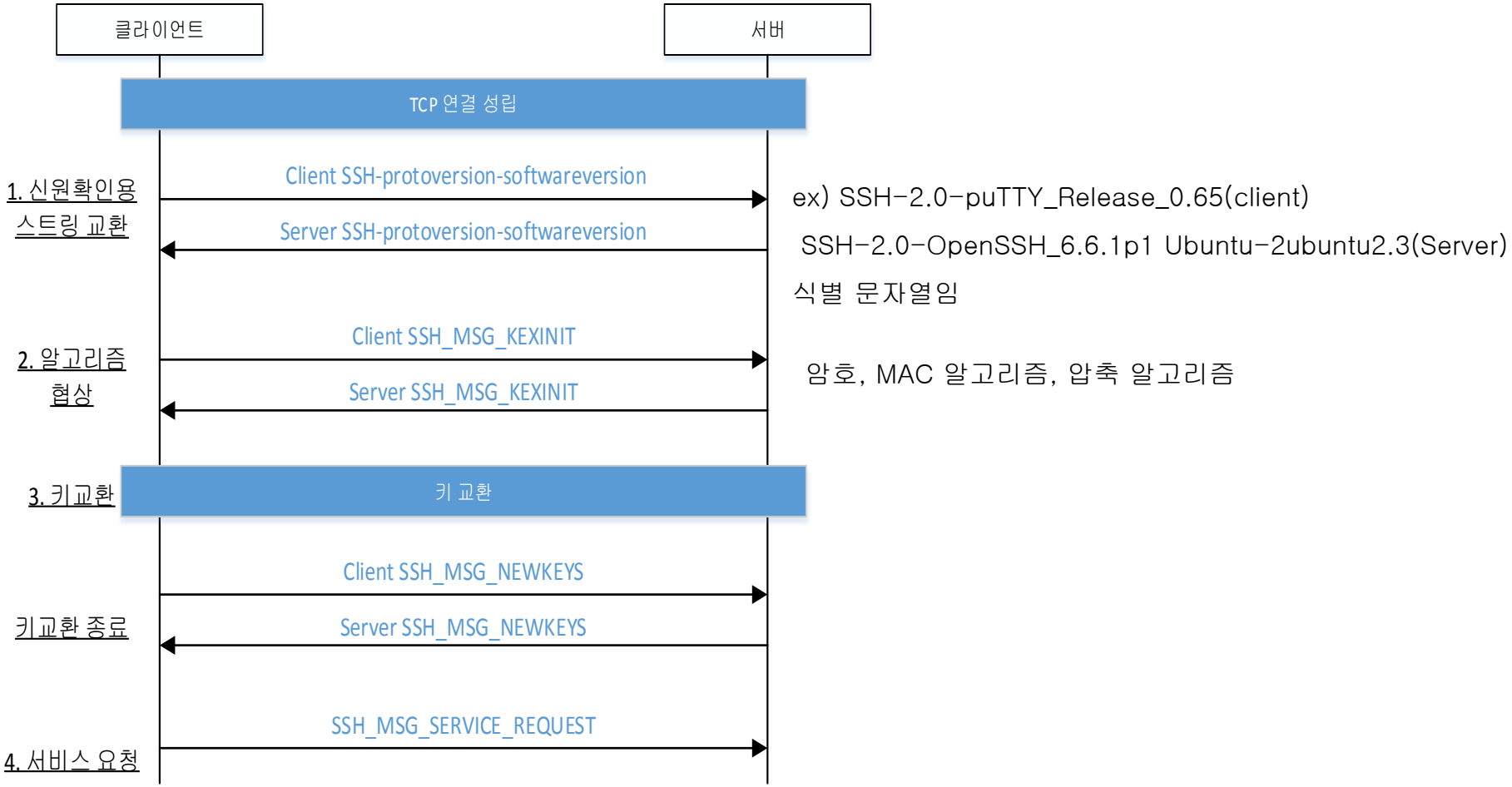
  - ➢ 전송 계층 프로토콜(Transport Layer Protocol)
    - ✓ 데이터 기밀성 및 무결성 제공
  - ➢ 사용자 인증 프로토콜(User Authentication Protocol)
  - ➢ 연결 프로토콜(Connection Protocol)
    - ✓ 1개의 암호화된 터널을 통해 다수의 논리채널 다중화
  - ➢ 응용 프로토콜 : Telnet, SMTP 등

SSH(Secure Shell)

- 전송 계층 프로토콜(Transport Layer Protocol)
  - 서버인증과 암호화를 통해 기밀성, 무결성 보장
  - 암호화에 사용할 알고리즘 협상, 키 교환, 암/복호화 담당
  - 서버 인증?
    - ✓ 클라이언트가 접속한 서버가 자신이 알고 있는 동일서버인지를 확인하는 방법
    - ✓ 최초 접속이 이루어 질 때 서버의 인증 여부를 확인
    - ✓ 이후 접속 시에 저장된 서버의 공개키를 통해 서버를 인증하는 것이 가능
    - ✓ 2가지 신뢰모델
      - 클라이언트가 각 호스트이름과 대응하는 공개키를 짝지어주는 데이터베이스 가짐
      - 호스트이름-키 쌍이 신뢰된 인증기관(CA)에 의해 인증
        - > 클라이언트는 오직 CA 루트 키만 알면, 신뢰한 CA들이 인증한 모든 호스트 키들을 검증
        - 각 호스트 키는 인가(authorization) 이전에 미리 중앙 기관에 의해 적절하게 인증될 수 있어야 함

SSH(Secure Shell)

- 전송 계층 프로토콜(Transport Layer Protocol)



SSH(Secure Shell)

- 전송 계층 프로토콜(Transport Layer Protocol)
  - 클라이언트는 서버와 TCP 연결을 설정함
    - ✓ 연결은 전송 계층 프로토콜이 아닌 TCP 연결을 통해 이루어짐
  - 1. 신원 확인용 스트링 교환
    - ✓ 클라이언트/서버 : 자신을 식별할 수 있는 문자열 보냄

SSH(Secure Shell)

- 전송 계층 프로토콜(Transport Layer Protocol)
  - ➢ 2. 알고리즘 협상
    - ✓ 지원가능 알고리즘을 선호도순으로 정렬한 목록을 SSH_MSG_KEXINIT에 포함해 상대방에게 전달
    - ✓ 목록으로 존재하는 알고리즘 유형 : 키 교환,암호, MAC, 압축 알고리즘

암호	
3des-cbc	CBC 모드 3중 DES
blowfish-cbc	CBC 모드 Blowfish
twofish256-cbc	256-비트키 CBC모드 Twofish
twofish-cbc	
twofish192-cbc	192-비트키 CBC모드 Twofish
twofish128-cbc	128-비트키 CBC모드 Twofish
aes256-cbc	256-비트키 CBC모드 AES
aes192-cbc	192-비트키 CBC모드 AES
aes128-cbc	128-비트키 CBC모드 AES
Serpent256-cbc	256-비트키 CBC모드 Serpent
Serpent192-cbc	192-비트키 CBC모드 Serpent
Serpent128-cbc	128-비트키 CBC모드 Serpent
arcfour	128-비트키 RC4
idea-cbc	CBC모드 idea
cast128-cbc	CBC 모드 CAST-128
des-cbc	CBC모드 des

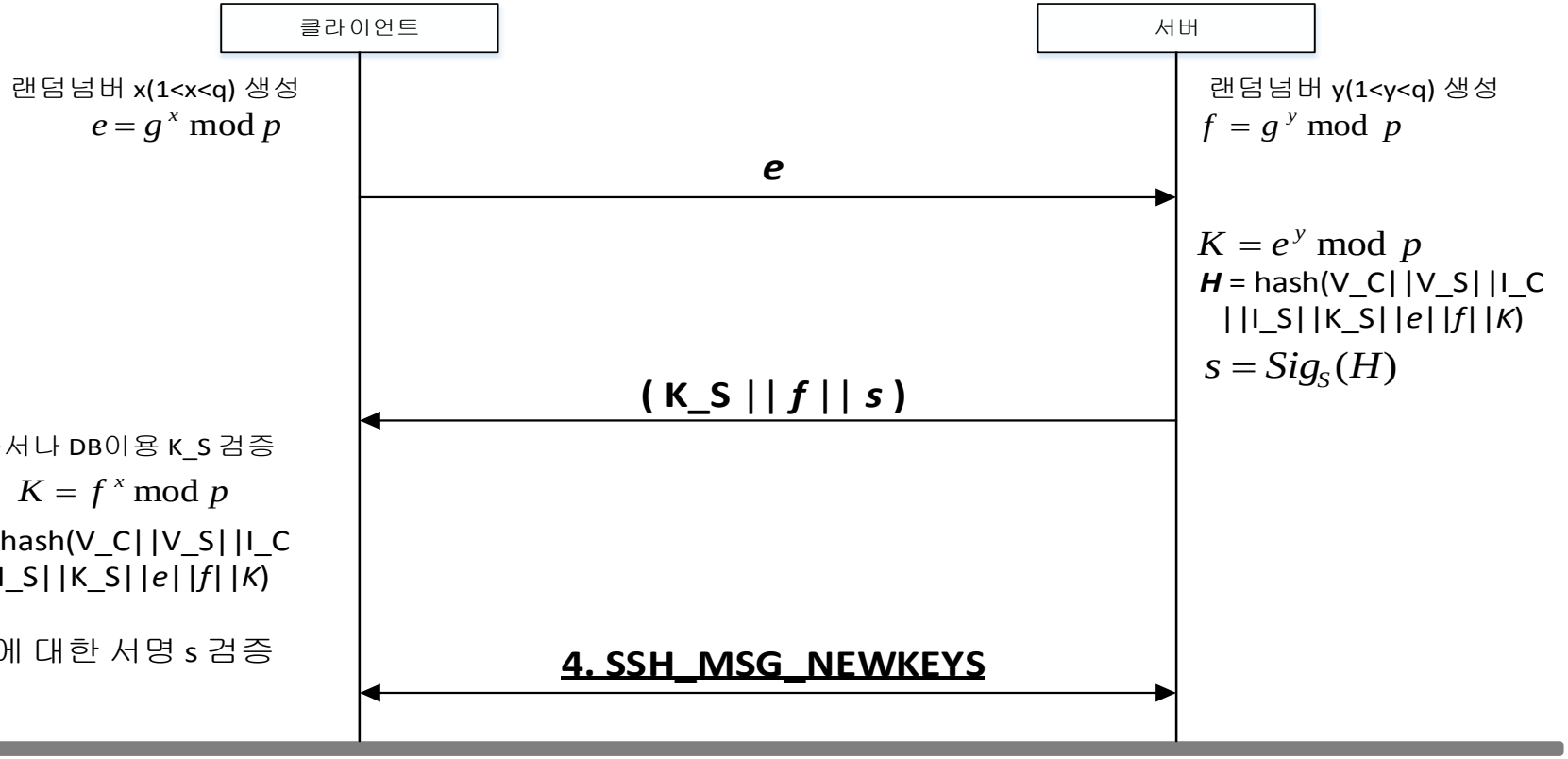
MAC 알고리즘	
hmac-sha1	HMAC-SHA1; 해시길이=키길이=20
hmac-sha1-96	HMAC-SHA1의 첫 96 비트; 해시길이=12
hmac-md5	HMAC-SHA1; 해시길이= 키길이=16
hmac-md5-96	HMAC-SHA1의 첫 96비트; 해시길이=12; 키길이=16

압축 알고리즘	
none	압축없음
zlib	RFC 1950과 RFC 1951에서의 정의

SSH(Secure Shell)

- 전송 계층 프로토콜
  - ➢ 3. 키 교환(Diffie-Hellman)
    - ✓ 교환 후 클/서는 K 가짐

키 교환 주요 인자		
인자	설명	비고(인자를 C/S가 얻게 된 시간)
p	안전한 큰 소수	알고리즘 협상 시
g	서브그룹의 생성자	알고리즘 협상 시
q	서브그룹 위수	알고리즘 협상 시
V_S	Server의 식별 문자열	초반에 신원 확인 할 때
V_C	Client의 식별 문자열	초반에 신원 확인 할 때
I_C	Client의 SSH_MSG_KEYINIT	알고리즘 협상 시
I_S	Server의 SSH_MSG_KEYINIT	알고리즘 협상 시
K_S	Server의 공개키	



SSH(Secure Shell)

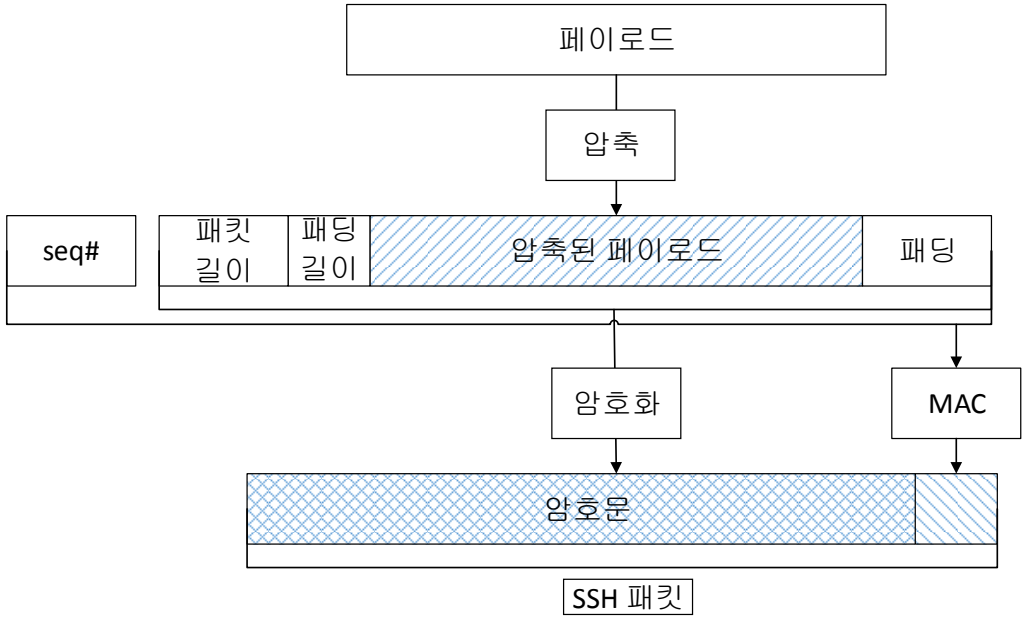
- 전송 계층 프로토콜(Transport Layer Protocol)
  - ➢ 키 교환 후 키 생성
    - ✓ 교환 후 공유된 **마스터 키 K**, 키 교환 시 계산된 **해시값 H**, **세션 식별자**를 이용해 암호화와 MAC에 사용하는 키를 생성(초기 키교환 이후 키교환 없으면 세션식별자=H)

    - ✓ 키(암호화, MAC에 사용되는 키(필요한 모든 IV 포함)
      - 클라이언트가 서버에게 보내는 초기 IV : HASH(*K* || *H* || "A" || session_id)
      - 서버가 클라이언트에게 보내는 초기 IV : HASH(*K* || *H* || "B" || session_id)
      - 클라이언트가 서버에게 보내는 암호화 키 : HASH(*K* || *H* || "C" || session_id)
      - 서버가 클라이언트에게 보내는 암호화 키 : HASH(*K* || *H* || "D" || session_id)
      - 클라이언트가 서버에게 보내는 MAC 키 : HASH(*K* || *H* || "E" || session_id)
      - 서버가 클라이언트에게 보내는 MAC 키 : HASH(*K* || *H* || "F" || session_id)

SSH(Secure Shell)

- 전송 계층 프로토콜(Transport Layer Protocol)
  - ➢ 4. 서비스 요청
    - ✓ 클라이언트는 사용자 인증이나 연결 프로토콜을 요청하기 위해 SSH_MSG_REQUEST 패킷을 보냄
    - ✓ 이 후 모든 패킷은 **SSH 전송 계층 패킷**의 페이로드에 실어 교환(암호화, MAC으로 보안)
    - ✓ SSH 패킷의 구성

- 패킷 길이(Packet Length): MAC 필드는 포함하지 않은 패킷의 바이트 길이
- 패딩 길이(Padding Length):랜덤 패딩의 길이
- 페이로드(Payload): 패킷의 유용한 정보. 어떤 알고리즘을 사용할지 결정 후 압축
- 랜덤 패딩(Random Padding): 암호화 알고리즘 협상이 이 완료되면 랜덤 패딩 필드 추가(임의의 바이트 길이)
- 메시지 인증 코드(MAC): MAC 필드를 제외한 전체 패킷과 순서번호에 대해 MAC값을 계산



SSH(Secure Shell)

- 사용자 인증 프로토콜(User Authentication Protocol)
  - 서버가 클라이언트를 인증 하는 것
  - 사용자 인증 방법
    - ✓ 비밀번호 (Password Authentication)
    - ✓ 공개키 (Public Key Authentication)

SSH(Secure Shell)

- 사용자 인증 프로토콜(User Authentication Protocol)
  - ➢ 비밀번호 인증
    - ✓ 가장 일반적인 방법
    - ✓ 비밀번호가 일치하지 않으면 접속을 허용하지 않음

구분	데이터 타입	메시지 구분
전체 메시지	바이트	SSH_MSG_USERAUTH_REQUEST
	문자열	사용자 이름
	문자열	서비스 명
	문자열	방법 이름 : password
	문자열	비밀번호

SSH(Secure Shell)

- 사용자 인증 프로토콜(User Authentication Protocol)
  - ➢ 비밀번호 인증(실습)
    - ✓ ssh 서버이름@203.237.183.114 입력

```
sarang@ubuntu:~/.ssh$ ssh admin@203.237.183.114
admin@203.237.183.114's password:
Welcome to Ubuntu 14.04.3 LTS (GNU/Linux 3.16.0-53-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

Last login: Tue Jan 12 13:13:44 2016 from 203.237.183.127
admin@pel-lab:~$
```

SSH(Secure Shell)

- 사용자 인증 프로토콜(User Authentication Protocol)
  - ➢ 공개키 인증
    - ✓ 클라이언트가 키쌍을 생성한 후 공개키를 서버에게 전달
    - ✓ 접속 요청이 들어오면 서버는 클라이언트의 공개키를 암호화하여 만들어진 난수를 전달
    - ✓ 클라이언트는 개인키로 난수 해독하여 MD5 해시값을 만들어 서버에게 전달
    - ✓ 서버는 자신이 생성했던 난수를 다시 MD5해시값으로 만든 후 클라이언트 쪽에서 넘겨 받은 것과 비교 일치하는 경우 접속을 허용

구분	데이터 타입	메시지 구분
전체 메시지	바이트	SSH_MSG_USERAUTH_REQUEST
	문자열	사용자 이름
	문자열	서비스 명
	문자열	방법 이름 : publickey
	문자열	알고리즘 이름
	문자열	공개키 데이터
	문자열	서명 값(첫 인증에는 포함x)

SSH(Secure Shell)

- 사용자 인증 프로토콜(User Authentication Protocol)
  - ➢ 공개키 인증(실습)
    1. ssh-keygen 이용해 **클라이언트**의 공개키와 개인키를 생성

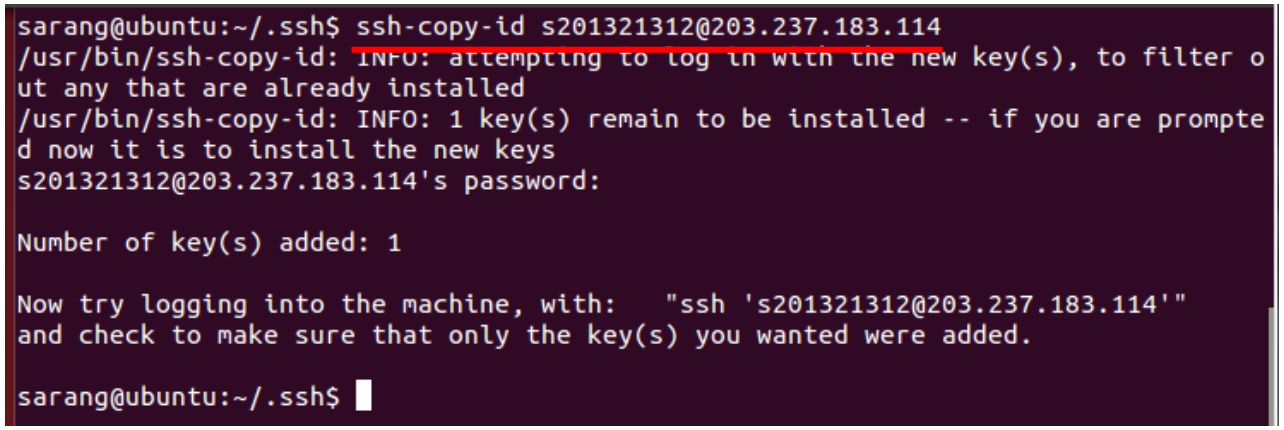
```
sarang@ubuntu:~/.ssh$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/sarang/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/sarang/.ssh/id_rsa.
Your public key has been saved in /home/sarang/.ssh/id_rsa.pub.
The key fingerprint is:
8d:62:43:cb:a3:25:53:9e:d3:bb:84:8c:ed:8d:c2:6b sarang@ubuntu
The key's randomart image is:
+--[ RSA 2048]----+
|                 |
|                 |
|      o          |
|     = + o       |
|    o % S .      |
|     X * .       |
|   .o + o        |
|    E. + .       |
|   ..oo o        |
+-----------------+
sarang@ubuntu:~/.ssh$
```

```
sarang@ubuntu:~/.ssh$ ls
id_rsa  id_rsa.pub  known_hosts
```

개인키 공개키

SSH(Secure Shell)

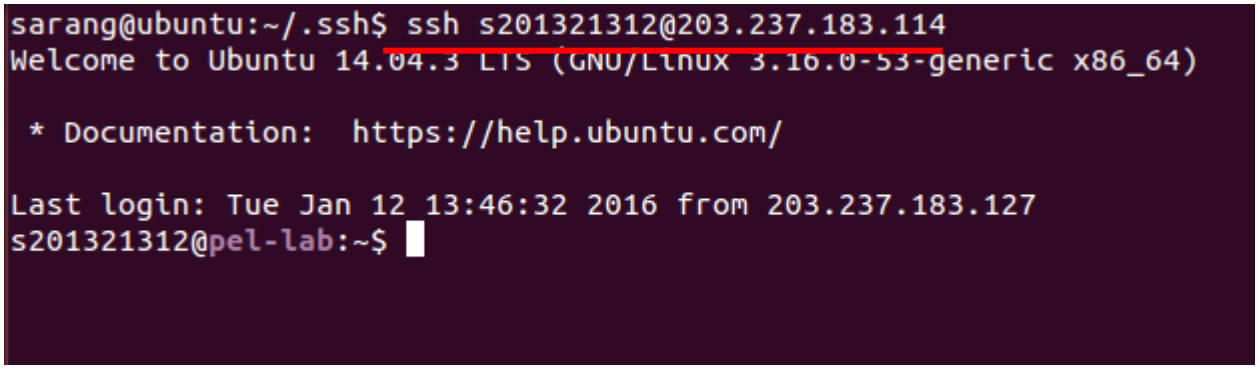
- 사용자 인증 프로토콜(User Authentication Protocol)
  - ➢ 공개키 인증(실습)
    2. 클라이언트의 공개키를 서버의 .ssh/authorized_keys 에 넣어줌(두가지 방법)
       - ① scp로 복사해서 서버의 authorized_keys에 직접 넣어주기
         - ✓ scp $HOME/.ssh/id_rsa.pub 서버@203.237.183.114:id_rsa_pub
         - ✓ cat $HOME/id_rsa.pub >> $HOME/.ssh/authorized_keys
       - ② ssh-copy-id 이용하기
         - ✓ ssh-copy-id 서버@203.237.183.114



SSH(Secure Shell)

- 사용자 인증 프로토콜(User Authentication Protocol)
  - ➢ 공개키 인증(실습)

3. 공개키 인증으로 서버에 접속

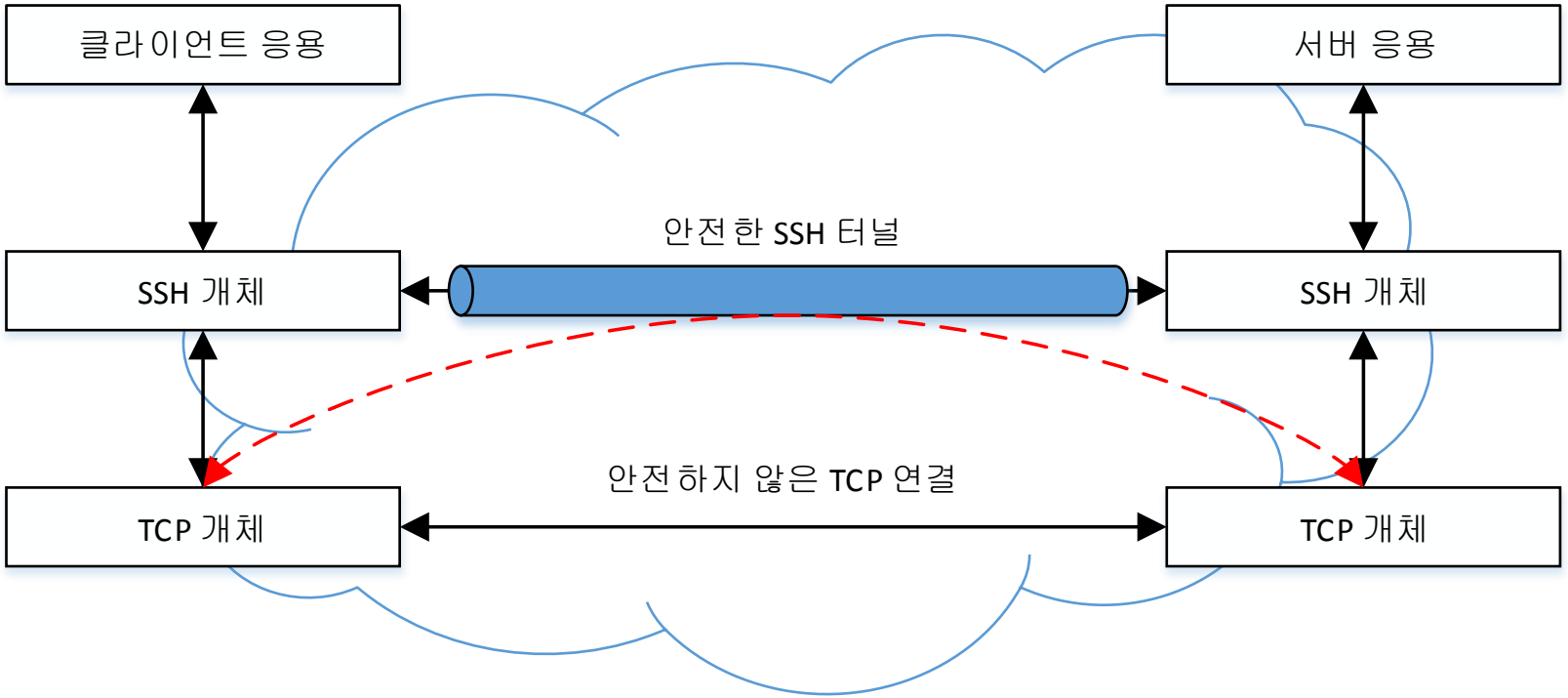


SSH(Secure Shell)

- 연결 프로토콜(Connection Protocol)
  - SSH 전송 계층 프로토콜 상에서 수행
  - 안전한 인증 연결 = 터널
  - 암호화된 터널을 다수의 논리적 채널로 다중화
  - 포트 포워딩(빨간 색)
    - ✓ 어떤 포트를 통해 흘러가는 데이터를 다른 포트로 전달시키는 기능

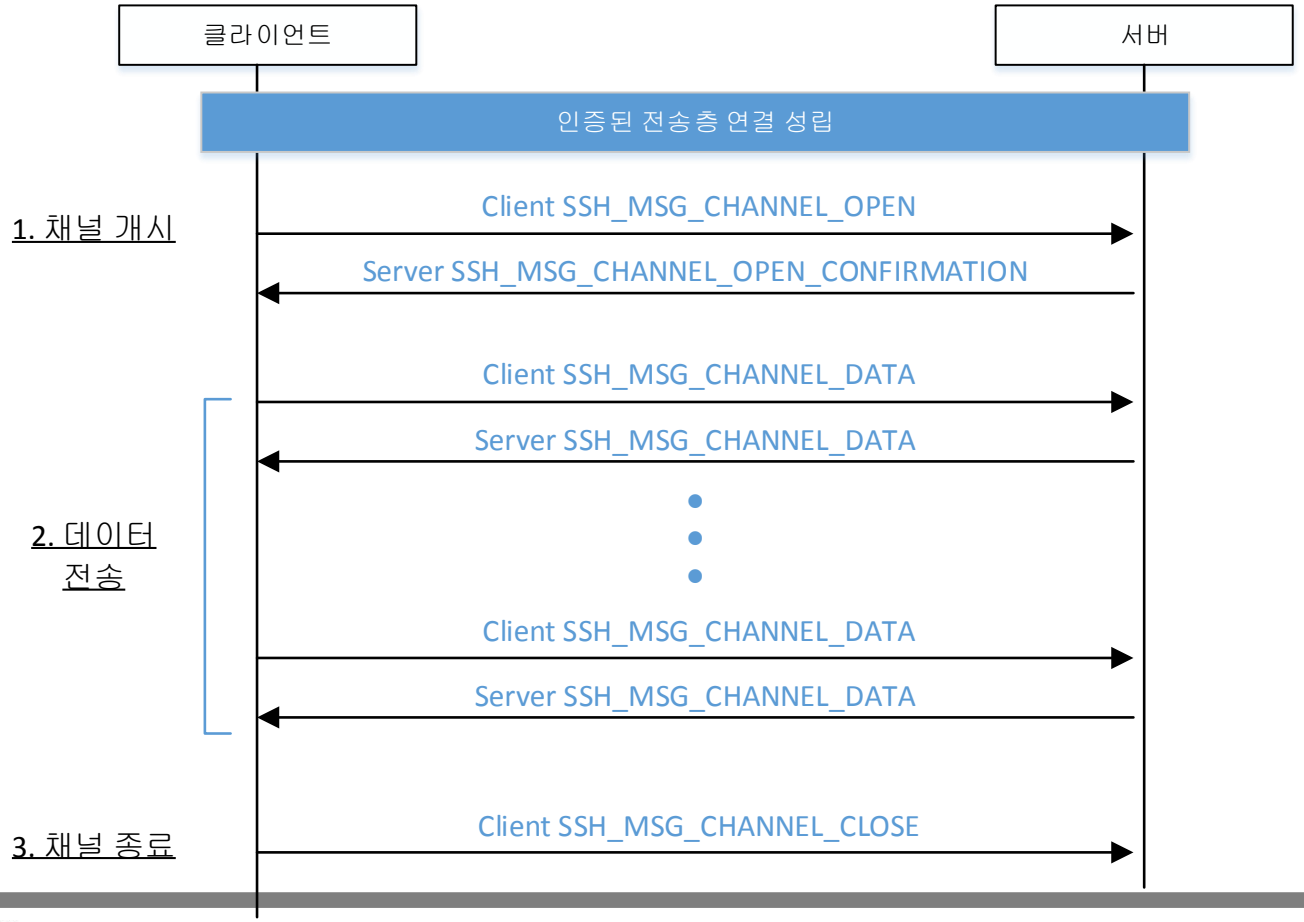
SSH(Secure Shell)

- 연결 프로토콜(Connection Protocol)
  - ➢ SSH 터널을 통한 연결



SSH(Secure Shell)

- 연결 프로토콜(Connection Protocol)
  - ➢ 채널 메커니즘



SSH(Secure Shell)

- 연결 프로토콜(Connection Protocol)
  - ➢ 채널 메커니즘

1. 채널 개시

- 로컬번호를 할당한 후 메시지 전송
- 전송되는 메시지의 형식

바이트(byte)	SSH_MSG_CHANNEL_OPEN	
문자열(string)	채널 유형	*이 채널을 위한 에플리케이션*
무부호정수32(uint32)	송신자 채널	*로컬 채널 번호*
Uint32	초기 윈도우 쿠기	*전송가능한 최대 채널 데이터 크기*
uint32	최대 패킷 크기	*전송 가능한 패킷의 최대 크기*
….	채널 유형에 따른 데이터	

- 원격지에서 채널 개시가능하면 SSH_MSG_CHANNEL_OPEN_CONFIRMATION 메시지 반환

  -> 메시지에 트래픽 송신채널번호, 수신자 채널번호, 윈도우, 패킷크기가 들어있음.

- 채널 개시 실패하면 실패 원인 코드와 SSH_MSG_CHANNEL_OPEN_FAILURE 메시지 반환

SSH(Secure Shell)

- 연결 프로토콜(Connection Protocol)
  - ➢ 채널 메커니즘

2. 데이터 전송

- SSH_MSG_CHENNEL_DATA 메시지를 이용해 데이터 전송
  ->메시지에 수신자 채널 번호, 데이터 블록이 들어있음

채널이 열려있는 동안에는 위의 메시지(SSH_MSG_CHENNEL_DATA)가 양방향으로 전송

3. 채널 종료

- 어느 한쪽에서 채널 종료를 원하면 '수신자 채널 번호가 포함된 SSH_MSG_CHENNEL_CLOSE'

감사합니다~