# NETWORK SECURITY ESSENTIALS

**- HTTPS & SSH -**

**Ki woon Moon**

**P**rotocol **E**ngineering **L**ab. **Sangmyung** University

2015. 01. 20

# Content

- **HTTPS**
- **SSH**

# HTTP (HyperText Transfer Protocol) (1/5)

## HTTP (HyperText Transfer Protocol)

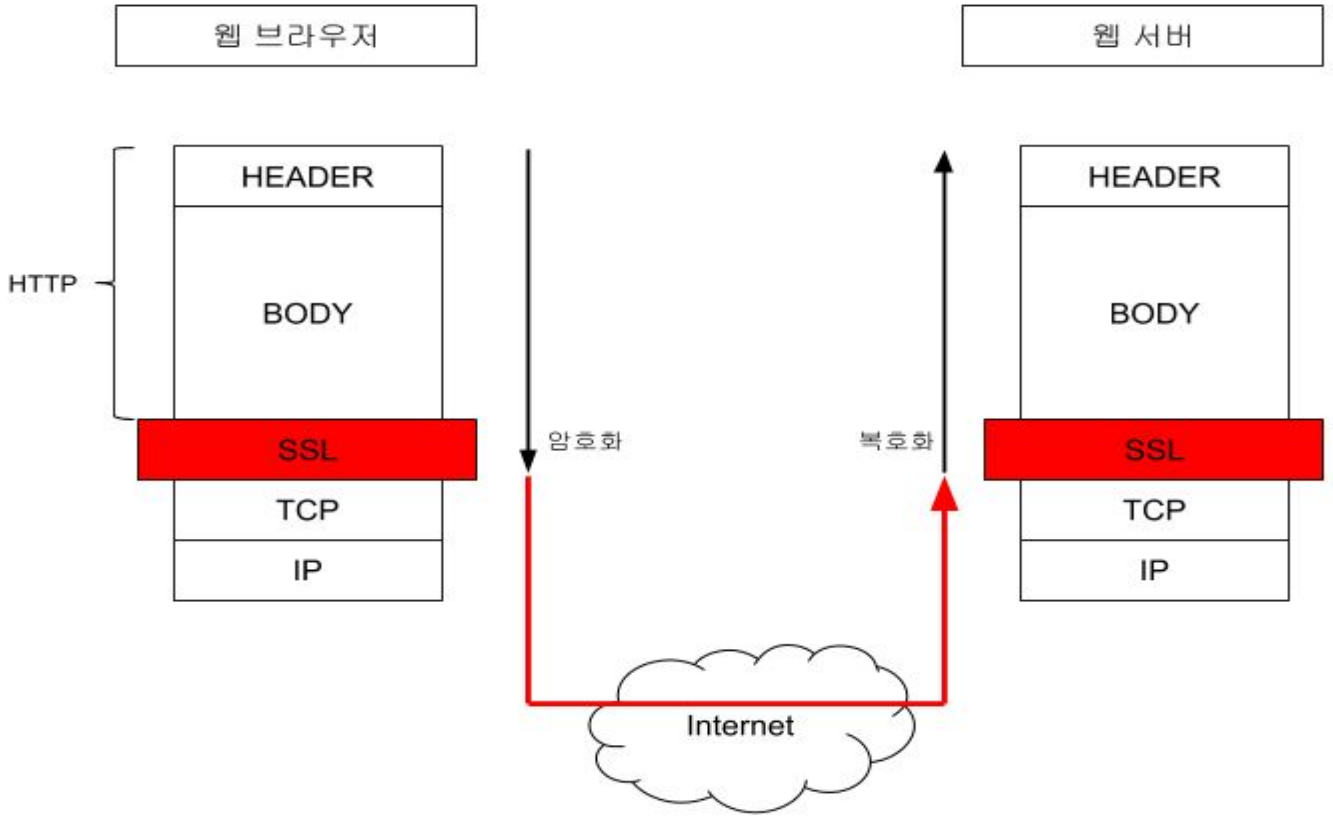
인터넷 상에서 데이터를 주고 받기 위한 서버/클라이언트 모델을 따르는 프로토콜

# HTTPS (HTTP over SSL) (2/5)

- **HTTPS**
  - ❖ HTTP의 경우 평문 데이터 전송을 원칙으로 하며 보안적인 문제점이 있음
  - ❖ HTTP와 SSL/TLS의 결합
  - ❖ URL 주소가 http:// 가 아닌 https://로 시작
  - ❖ HTTP의 기본 포트 번호는 80, HTTPS의 기본 포트 번호는 443
  - ❖ 암호화 되는 통신 요소
    - 요청된 문서의 URL
    - 문서의 내용
    - HTTP 헤더의 내용

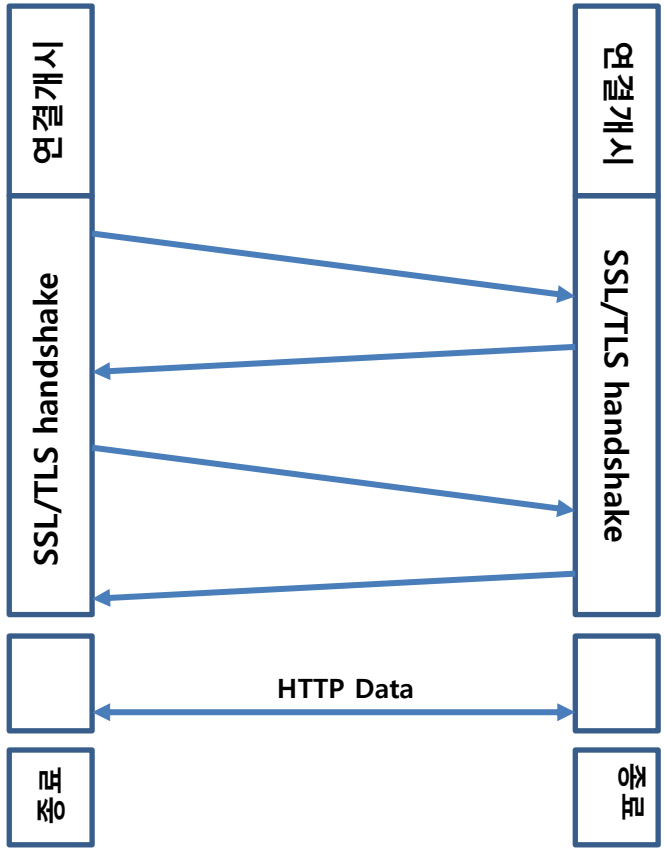
# HTTPS (HTTP over SSL) (3/5)

- **HTTPS 기본 동작**



# HTTPS (HTTP over SSL) (4/5)

- **연결 개시**
  - 적절한 포트를 통해 서버에 연결 개시
  - SSL/TSL 핸드쉐이크 시작 위해 TLS ClientHello 전송
  - SSL/TLS 핸드쉐이크 종료 시 첫 HTTP 요청 시작
  - 모든 HTTP 데이터는 SSL/TLS 응용데이터로서 전송
  - 이후 일반적 HTTP 동작



# HTTPS (HTTP over SSL) (5/5)

- **연결 종료**

❖ HTTP 클라이언트나 서버는 HTTP 레코드에 Connection: close라는 라인을 포함

❖ HTTPS 연결을 종료하기 위해서는, TLS가 원격에 있는 상대방의 TLS 개체와의 연결을 종료해야 함

- ➢ 하위 TCP 연결 종료를 포함
- ➢ 양 측에서 close_notify 경보(alert)를 보내는 TLS 경보 프로토콜 사용이 적절

❖ 연결을 종료하기 전에 종료 경보(closure alert)를 교환해야 함

❖ 비정상 종료 - 보안 경고(warning)를 발행해야 함

- ➢ HTTP 클라이언트는 하위 TCP 연결이 사전에 close_notify 경보와 Connection: close 지시자를 보내지 않고 종료되는 상황
- ➢ 서버의 프로그램 오류나 TCP 연결 중지를 야기시키는 통신 오류 발생

# SSH (Secure Shell) (1/14)

- **Telnet**
  - ❖ 원격지에 위치한 컴퓨터 시스템을 온라인으로 연결하여 사용하는 서비스
  - ❖ RFC15를 시작으로 1969년에 개발되었으며 최초의 인터넷 표준들 가운데 하나로서 IETF STD 8로 표준화
  - ❖ 가장 오래된 원격 접속 방법
  - ❖ 텔넷의 포트 번호는 23
  - ❖ 전송되는 내용들을 암호화 하지 않으므로 보안에 취약

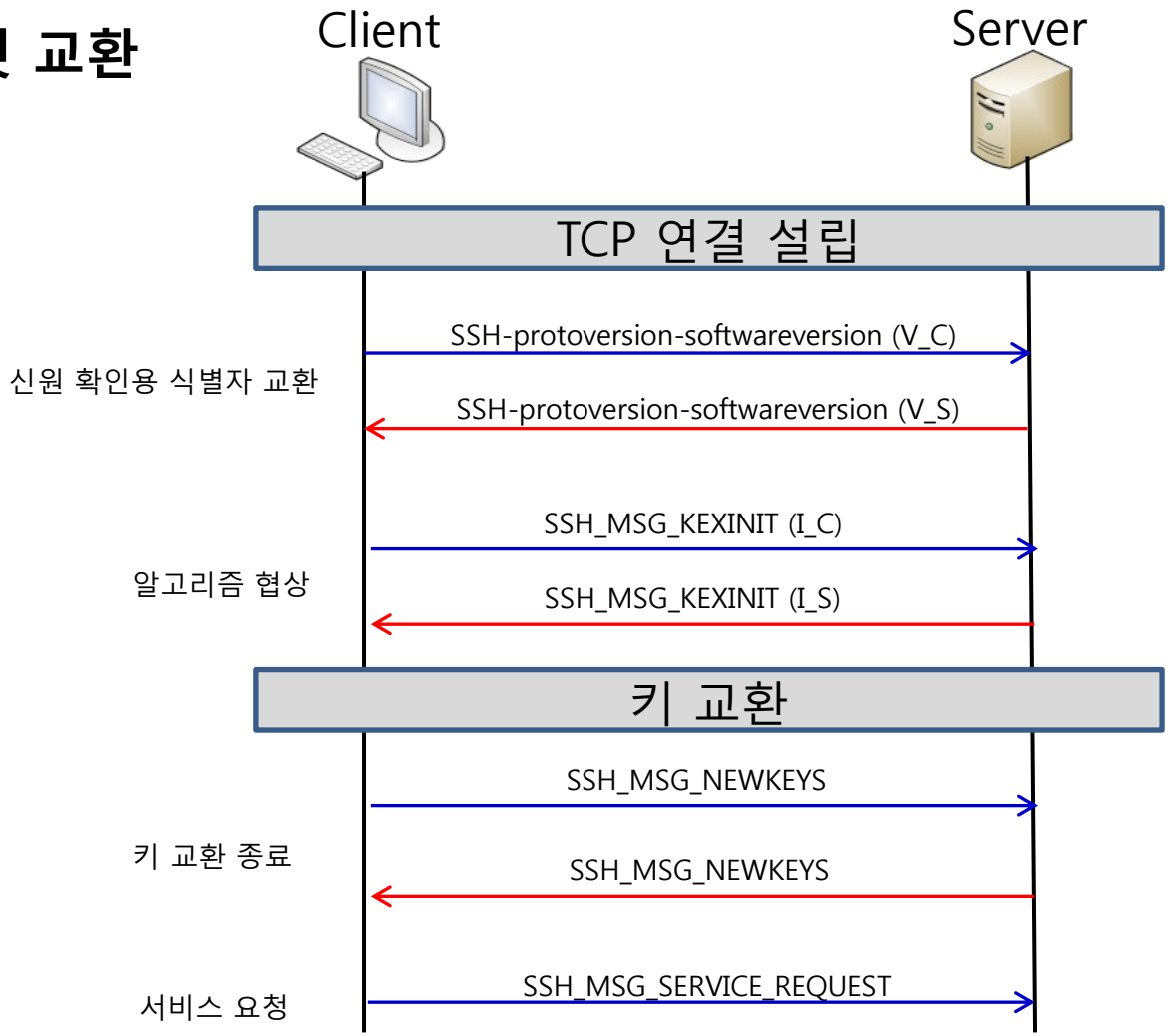
# SSH (Secure Shell) (2/14)

- **SSH 개요**

  - 1995년 핀란드의 Tatu Ylönen이 개발해 공개, SSH Communications Security사를 설립하고, 상용화
  - 암호화 기법을 사용하기 때문에, 통신이 노출된다 하더라도 암호화된 문자로 보임
  - SSH의 포트 번호는 22
  - 초기 버전 SSH1은 원격 로그온 기능을 제공하는데 중점을 두고 개발
  - SSH2는 RFC 4250~4256 문서의 표준으로 제안됨

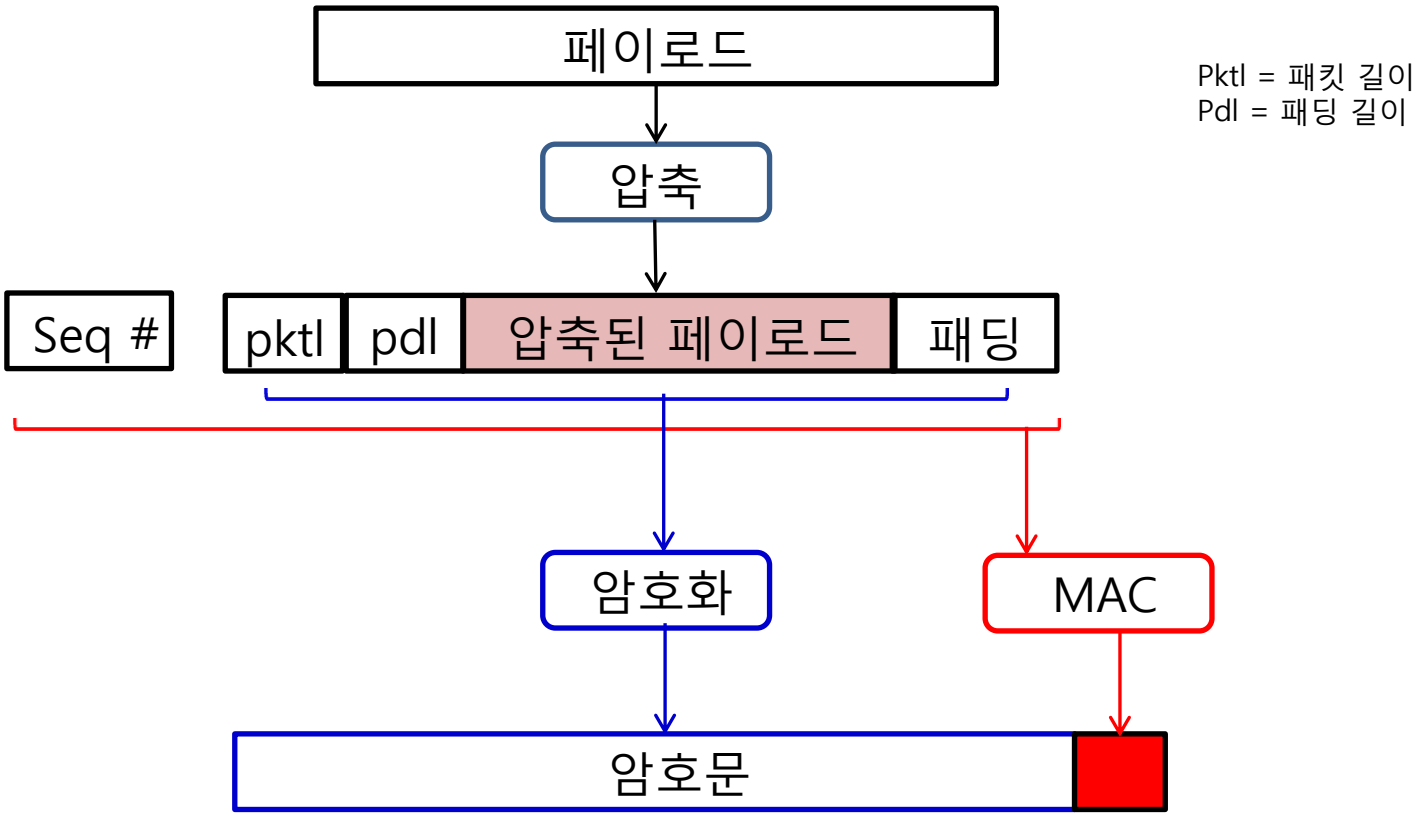
# SSH (Secure Shell) (3/14)

- **SSH 전송 계층 패킷 교환**



# SSH (Secure Shell) (4/14)

- **SSH 전송 계층 패킷 구성**



# SSH (Secure Shell) (5/14)

- **SSH 전송 계층 패킷 구성**

❖ Packet length: 패킷의 길이(MAC의 길이는 미포함)

❖ Padding length: 랜덤 패팅의 길이

❖ Payload: 패킷의 대한 정보가 담겨있음. 어떤 암호화 알고리즘을 사용할지 결정 후 압축

❖ Random padding : 암호화 알고리즘 결정 후, 랜덤 패팅 필드를 추가,
임의의 바이트 길이이며 총 패킷 길이가 암호문 블록크기의 배수가
되게 만듬, 스트림 암호의 경우 8바이트

❖ MAC : MAC 필드를 제외한 전체 패킷과 Sequence Number에 대해 MAC 값을 계산

# SSH (Secure Shell) (6/14)

- **SSH에 쓰이는 알고리즘 유형별 목록**
  - **지원할 수 있는 알고리즘을 선호도 순으로 정렬한 목록**
  - **SSH_MSG_KEXINIT 포함시켜 알고리즘 협상에 쓰임**

| 암호 알고리즘 | |
|---|---|
| 3des-cbc | CBC 모드 3중 DES |
| blowfish-cbc | CBC 모드 Blowfish |
| twofish256-cbc | 256-비트키 CBC모드 twofish |
| 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 |
| arctour | 128-비트키 RC4 |
| cast128-cbc | CBC모드 CAST-128 |

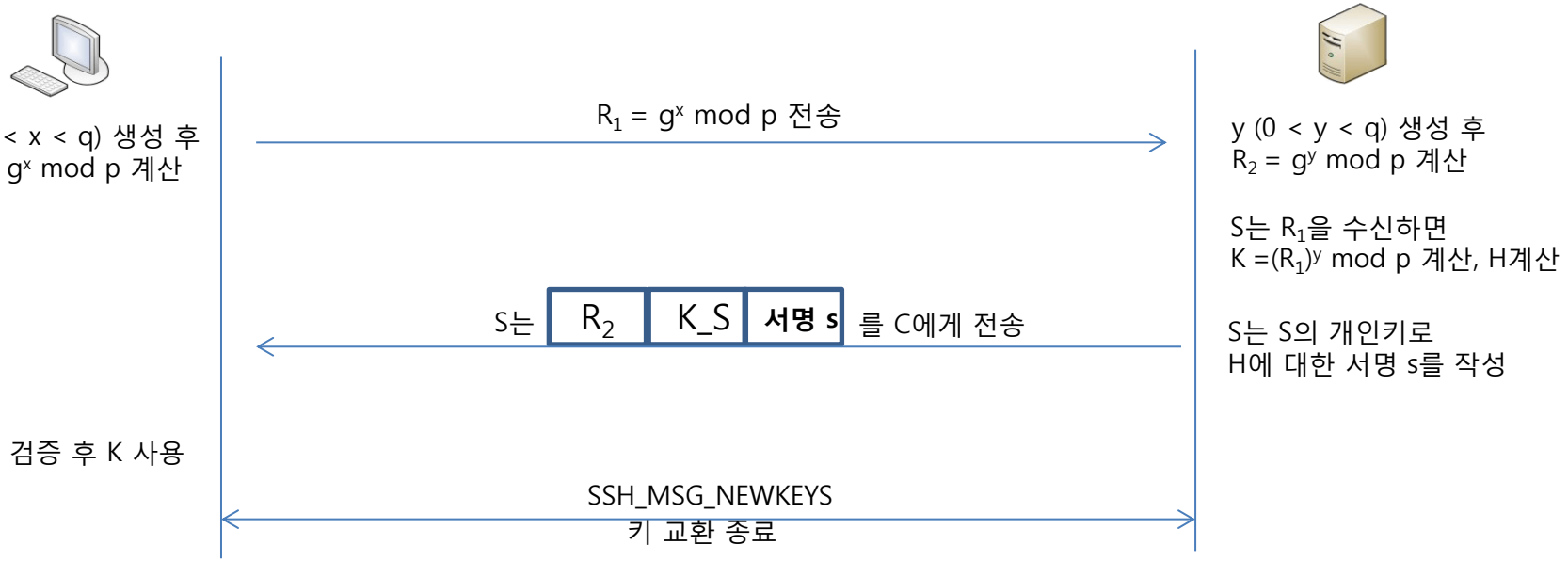
| MAC 알고리즘 | |
|---|---|
| hmac-sha1 | HMAC-SHA1; 해시길이=키 길이=20 |
| hmac-sha1-96 | HMAC-SHA1의 첫 비트 96비트; 해시 길이 12 |
| hmac-md5 | HMAC-SHA1; 해시 길이=키 길이=16 |
| hmac-md5-96 | HMAC-SHA1의 첫 비트 96비트; 해시 길이 =12, 키 길이 = 16 |

| 압축 알고리즘 | |
|---|---|
| none | 압축 없음 |
| zlib | HMAC-SHA1의 첫 비트 96비트; 해시 길이 12 |

# SSH (Secure Shell)

- **SSH Diffie-Hellman 키 교환**

| 키 교환 주요 인자 | | |
|---|---|---|
| 인자 | 설명 | 비고 |
| C | 클라이언트 | |
| S | 서버 | |
| V_S | 서버의 식별 문자열 | 패킷 교환 첫 단계에서 교환 |
| V_C | 클라이언트의 식별 문자열 | 패킷 교환 첫 단계에서 교환 |
| K_S | 서버의 공개키 | |
| I_S | 서버의 SSH_MSG_KEXINIT | 알고리즘 협상 단계에서 교환 |
| I_C | 클라이언트 SSH_MSG_KEXINIT | 알고리즘 협상 단계에서 교환 |




p = 안전한 큰 소수 (300자리 넘는 10진수)
g = 위수가 p-1인 생성자
q = 위수
H = hash(V_C || V_S || I_C || I_S || K_S || $R_1$ || $R_2$ || K )
$K = g^{xy} \bmod p$

# SSH (Secure Shell) (8/14)

- **키 생성**

암호화와 MAC에 사용되는 키들(필요한 IV 포함)은
공유된 비밀키 *K*와 키 교환에서 계산된 해쉬 값 *H*로부터 생성

- 클라이언트가 서버에게 보내는 초기 IV: hash(K || H || "A" || session_id)
- 서버가 클라이언트에게 보내는 초기 IV: hash(K || H || "A" || session_id)
- 클라이언트가 서버에게 보내는 암호화 키 : hash(K || H || "A" || session_id)
- 서버가 클라이언트에게 보내는 암호화 키 : hash(K || H || "A" || session_id)
- 클라이언트가 서버에게 보내는 MAC 키 : hash(K || H || "A" || session_id)
- 서버가 클라이언트에게 보내는 MAC 키 : hash(K || H || "A" || session_id)

# SSH (Secure Shell) (9/14)

- **서비스 요청**

  SSH_MSG_SERVICE_REQUEST 패킷 전송

  이후로 모든 데이터는 SSH 전송 계층 패킷의 페이로드로서 교환

  암호화와 MAC으로 보호

# SSH (Secure Shell) (10/14)

- **사용자 인증 프로토콜**

❖ 클라이언트가 인증을 요청

| | | |
|---|---|---|
| 바이트(byte) | SSH_MSG_USERAUTH_REQUEST (50) | |
| 문자열(string) | 사용자 이름 | 클라이언트가 청구한 인가용 신원 |
| 문자열(string) | 서비스 이름 | 보통 SSH |
| 문자열(string) | 방법 이름 | 사용할 인증 방법 |
| … | 방법에 따른 필드들 | |

❖ 인증을 거절하거나 다른 인증 방법을 추가적으로 요구

| | | |
|---|---|---|
| 바이트(byte) | SSH_MSG_USERAUTH_FAILURE (51) | |
| 이름-목록(string) | 계속되는 인증들 | 대화를 계속할 수 있는 방법의 목록 |
| 논리값(string) | 부분적인 성공 | |

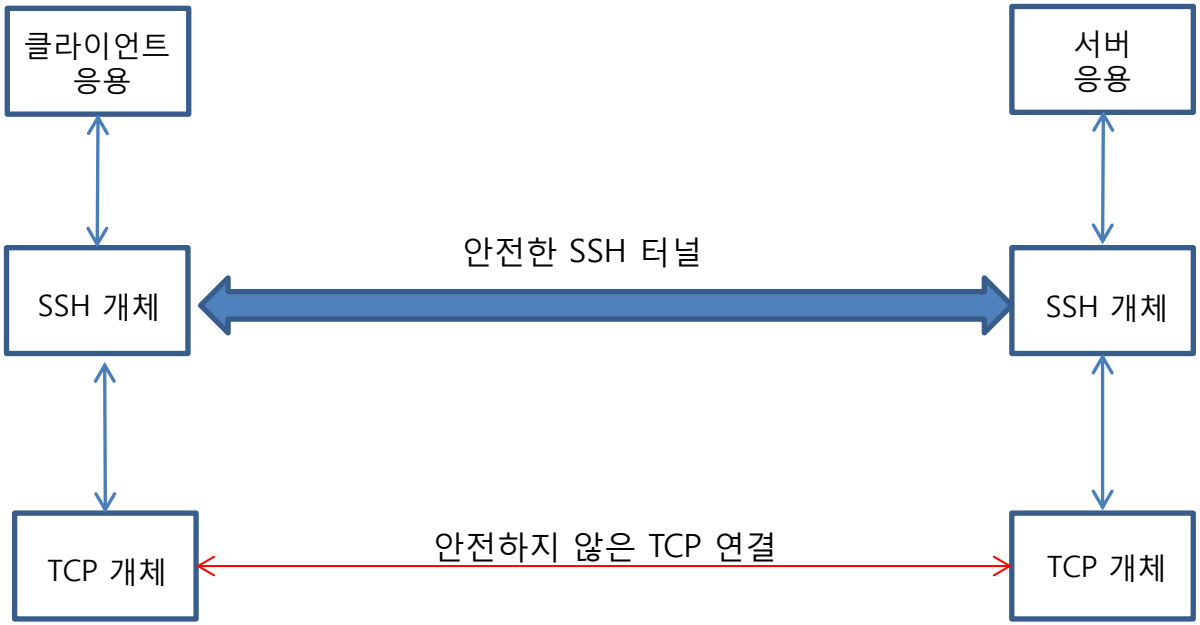
❖ 인증을 거절하거나 다른 인증 방법을 추가적으로 요구

서버가 인증을 수용하면 1 바이트의 메시지인
SSH_MSG_USERAUTH_SUCCESS ( 52 )를 전송

# SSH (Secure Shell) (11/14)

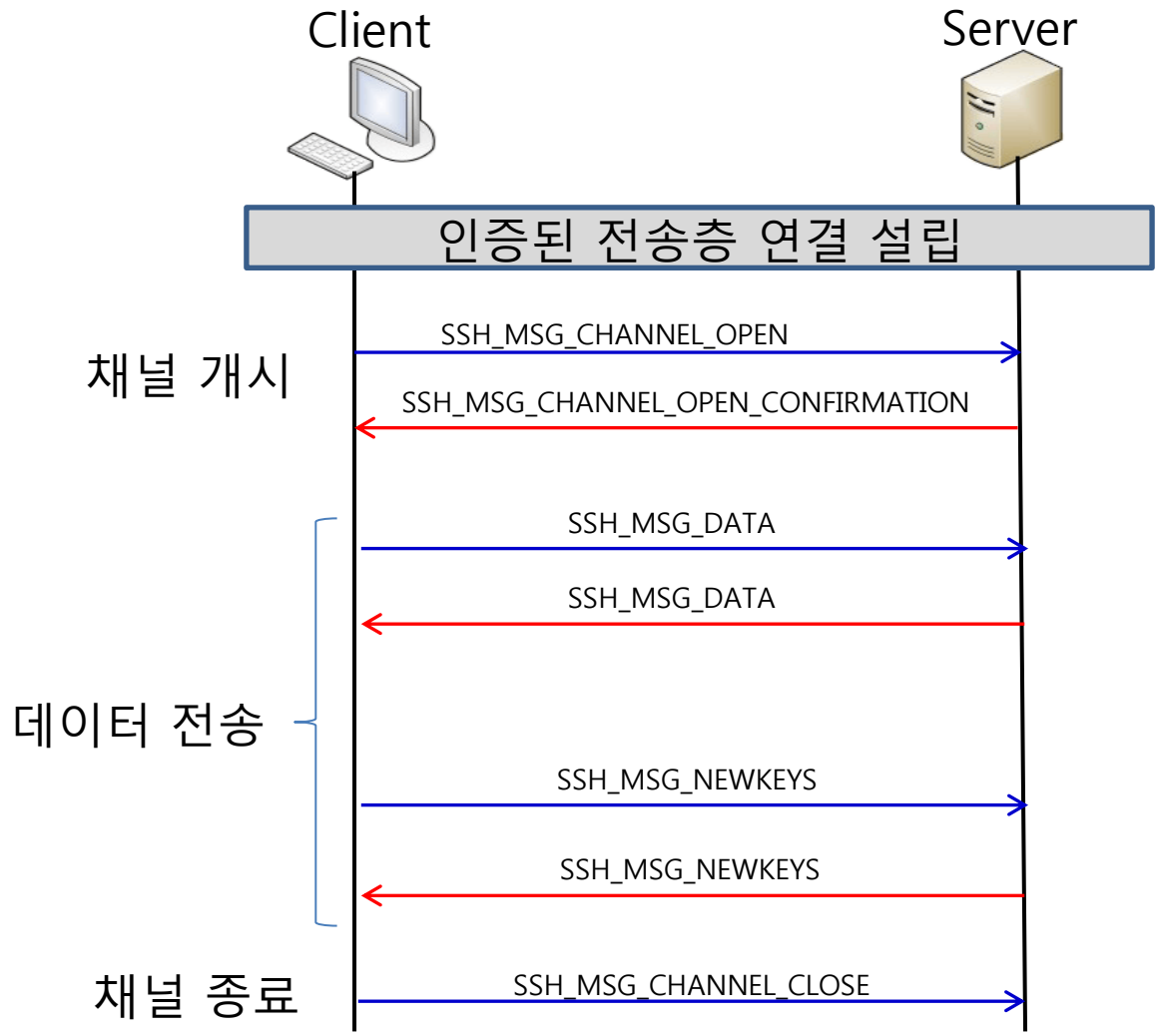
- **연결 프로토콜**

SSH 연결 프로토콜은 SSH 전송 계층 프로토콜 상에서 수행
안전한 인증 연결을 가정, 별도의 채널을 사용하여 통신
안전한 인증 연결은 터널(tunnel)이라고 불림



# SSH (Secure Shell) (12/14)

- **채널 메커니즘**



- **채널 메커니즘**
  - 채널 개시

❖ SSH 연결 프로토콜은 SSH 전송 계층 프로토콜 상에서 수행

| | | |
|---|---|---|
| 바이트(byte) | SSH_MSG_CHANNEL_OPEN | |
| 문자열(string) | 채널 유형 | 이 채널을 위한 애플리케이션을 지정 |
| 무부호정수32(uint32) | 송신자 채널 | 로컬 채널 번호 |
| 무부호정수32(uint32) | 초기 윈도우 크기 | 전송 가능한 최대 채널 데이터 크기 |
| 무부호정수32(uint32) | 최대 패킷 크기 | 전송 가능한 패킷의 최대 크기 |
| … | 채널 유형에 따른 데이터 | |

❖ 상대방이 채널을 개시할 수 있으면 SSH_MSG_CHANNEL_OPEN_CONFIRMATION 메시지를 반환

❖ 채널을 개시할 수 없으면 SSH_MSG_CHANNEL_OPEN_FAILURE 메시지를 반환

# SSH (Secure Shell) (14/14)

- **채널 메커니즘**
  - 데이터 전송
    - ❖ 채널 개시되면 SSH_MSG_CHANNEL_DATA 메시지를 이용하여 데이터 전송
    - ❖ 메시지에는 수신자의 채널 번호와 데이터 블록 포함
  - 채널 종료
    - ❖ 어느 한 쪽에서 채널 종료를 원하면 SSH_MSG_CHANNEL_CLOSE 메시지를 전송
    - ❖ 메시지에는 수신자의 채널 번호 포함