2021/03/28, 2021 보안 기초 세미나

# Network Security Essentials

## - Chapter 5 전송-레벨 보안 -

박 재 형(jaehyoung@pel.sejong.ac.kr)

세종대학교 프로토콜공학연구실

# 목 차

- 웹 보안
- 안전 소켓 계층(SSL)
- 전송 계층 보안(TLS)
- HTTPS
- SSH

# 목 차

# 웹 보안

- 개요
  - 웹(WWW, World Wide Web)
    - 정의
      - 인터넷에 연결된 사용자들이 서로의 정보를 공유할 수 있는 공간
    - 특징
      - 인터넷 상에서 텍스트나 그림, 소리, 영상 등과 같은 정보를 하이퍼텍스트 방식으로 연결하여 제공
        - 하이퍼텍스트(Hypertext)
          - 문서 내부에 참조를 넣음으로써, 웹 상에 존재하는 여러 문서끼리 서로 참조할 수 있는 기법
        - 하이퍼링크(Hyperlink)
          - 문서 내부에서 또 다른 문서로 연결되는 참조를 의미

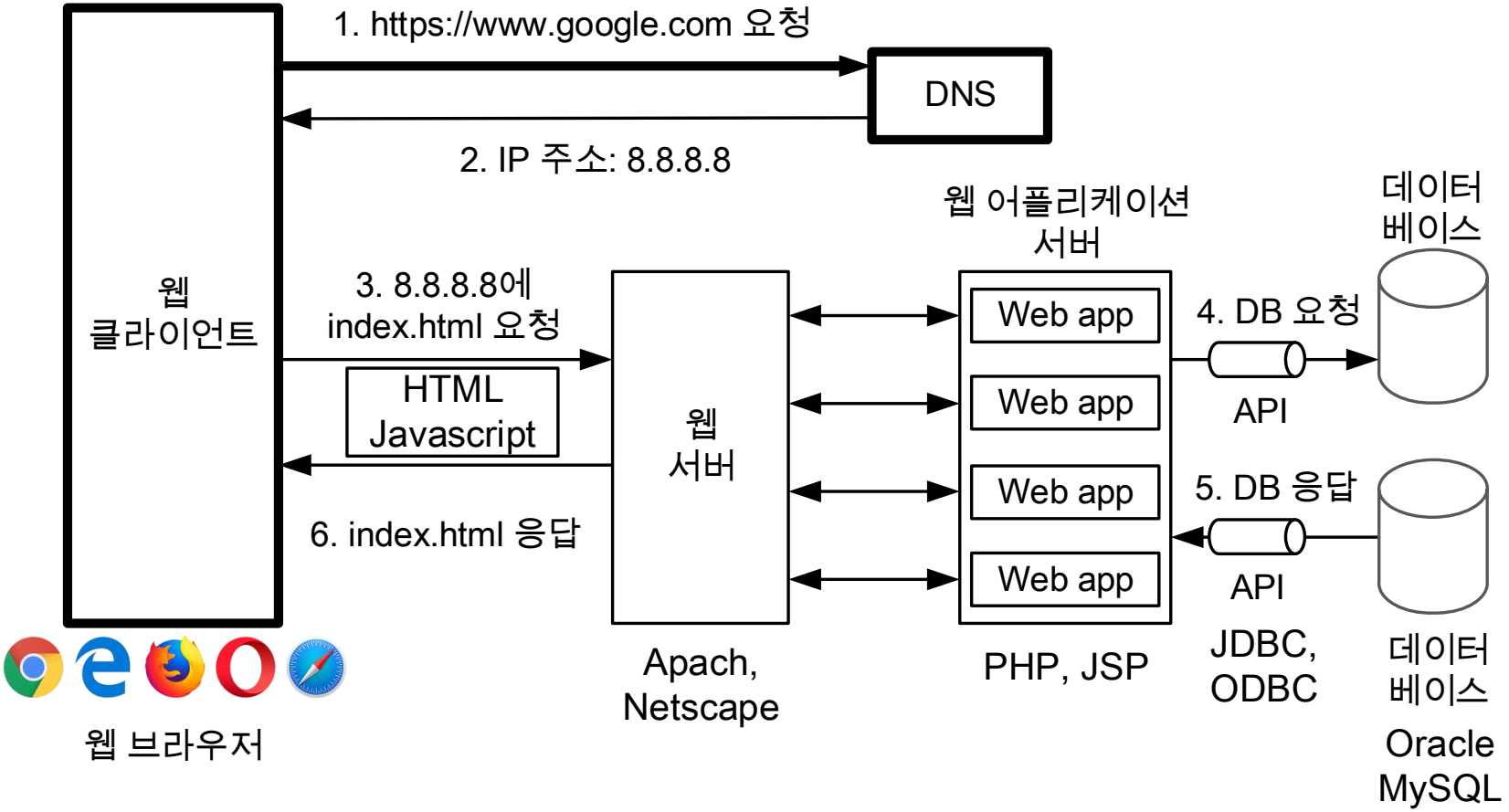
# 웹 보안

- 개요
  - 웹(WWW, World Wide Web)
    - 구성

| 구성 | 설명 |
|---|---|
| 웹 페이지<br>(Web Page) | HTML(HyperText Markup Language)언어를 사용하여 작성된<br>하이퍼텍스트 문서 |
| 웹 사이트<br>(Web Site) | 웹 페이지들 중에서 서로 관련된 내용으로 작성된<br>웹 페이지들의 집합 |
| 웹 서핑<br>(Web Surfing) | 사용자가 웹 페이지에 포함된 하이퍼링크를 따라 다른<br>웹 페이지들로 이동하는 행위 |
| 웹 브라우저<br>(Web Browser) | 사용자가 웹 페이지를 검색하기 위해 사용하는 프로그램 |

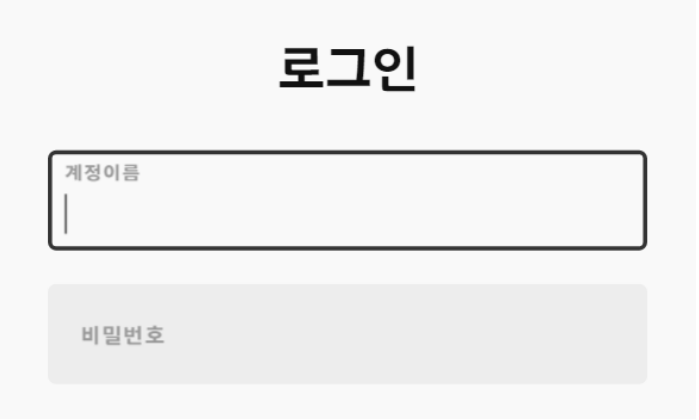
# 웹 보안

- 개요
  - 웹(WWW, World Wide Web)
    - 구조



# 웹 보안

- 웹 취약점
  - 명령 삽입
    - 웹 서버와 연동된 데이터 베이스에 입력되는 값에 대한 검증을 수행하지 않아 발생
    - SQL Injection
      - 데이터베이스와 연동되어 있는 애플리케이션의 입력 값을 조작
        - DBMS(database management system)가 의도하지 않은 결과 반환
      - 로그인 폼에서의 인증 절차
        1. 로그인 창에 아이디, 패스워드 입력
        2. SQL Query 생성
        3. Database에 Query 전송
        4. Database에 Query 실행
        5. 반환되는 Return 값으로 인증 여부 판단



# 웹 보안

- 웹 취약점
  - 명령 삽입
    - SQL Injection
      - 로그인 폼에서의 인증 우회
        - Query문에서의 로그인 인증 우회
          - FALSE OR TRUE = TRUE 조건 사용
          - 더블 대시(--) 사용

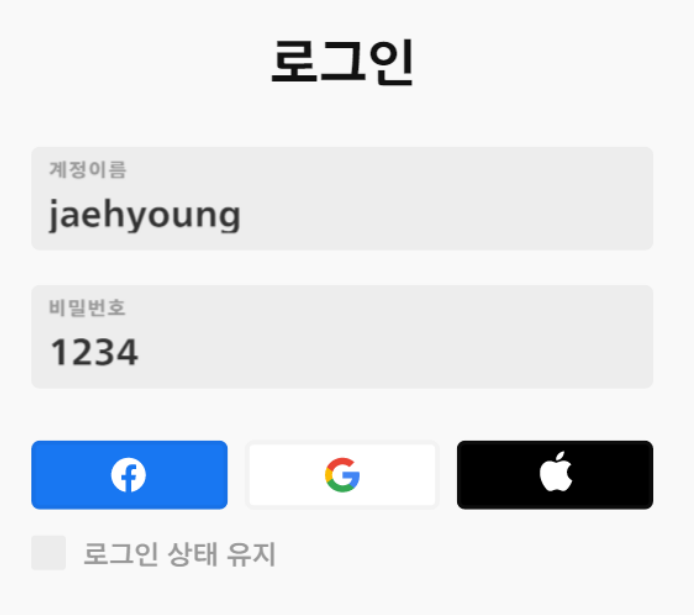
<기존 SQL Query 명령>

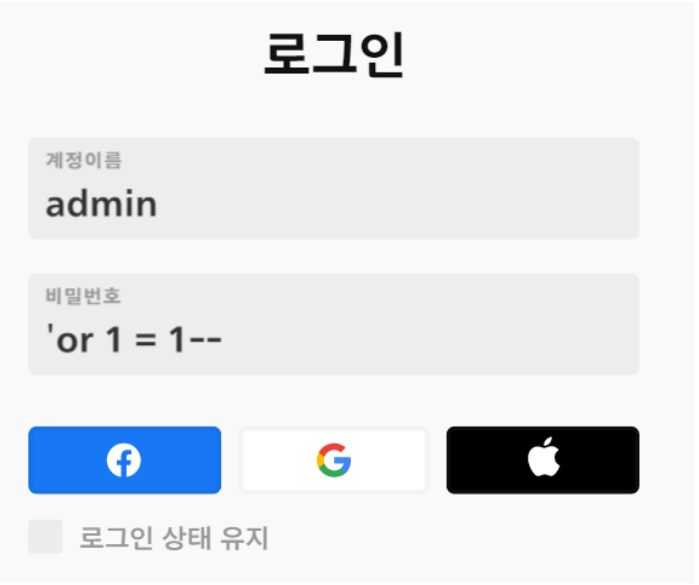
```
SELECT *
FROM users
WHERE ID='jaehyoung' and PW='1234'
```

<SQL Injection 명령>

```
SELECT *
FROM users
WHERE ID='admin' and PW='' or 1=1--'
```

‘‘와 1=1(True) or 연산하게 되어 무조건 True 반환
더블 대시(--)는 SQL 주석 처리이므로 뒷문장은 제거됨



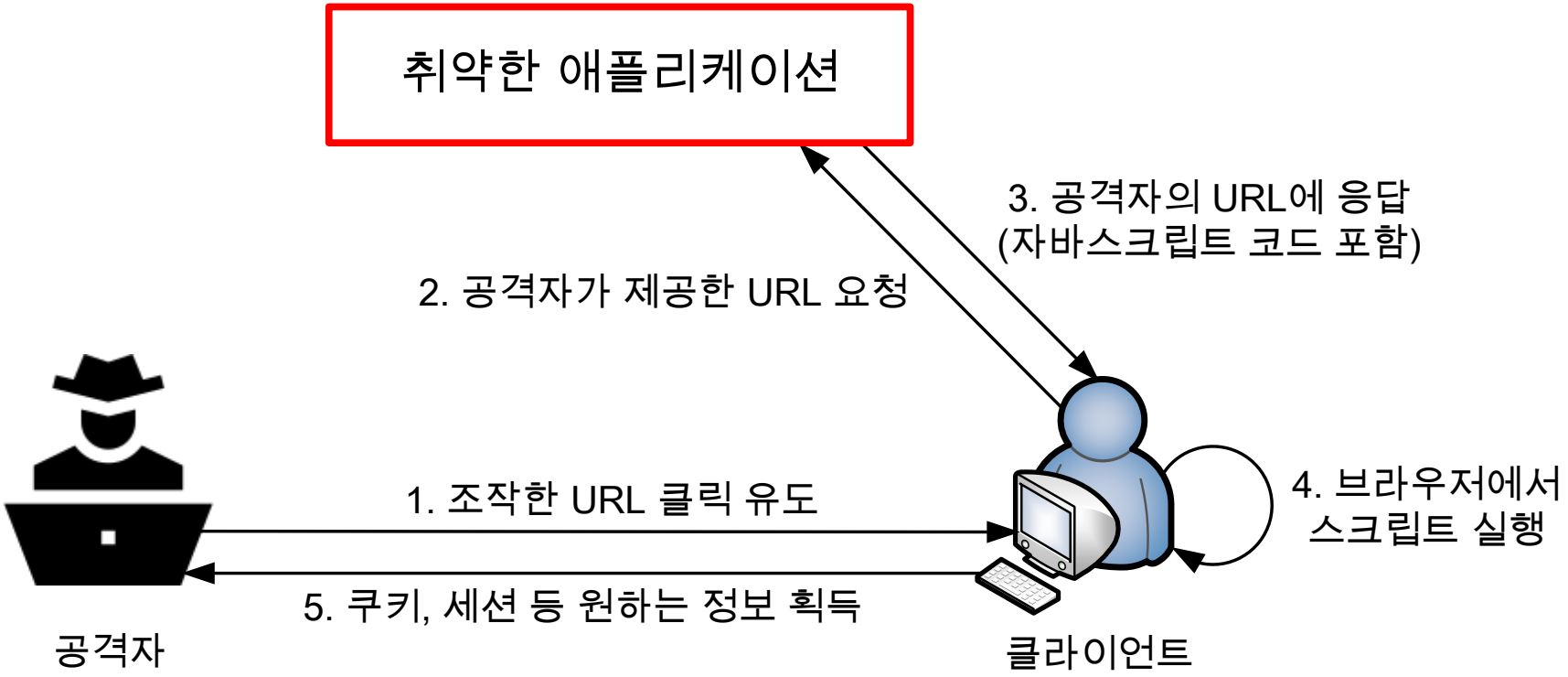



# 웹 보안

- 웹 취약점
  - XXS(Cross Site Scripting) 취약점
    - 다른 사용자의 웹 브라우저 내에서 적절한 검증 없이 실행되어 발생하는 취약점
    - 공격자가 작성한 스크립트를 통해 정상 사용자의 세션을 탈취하거나, 웹 사이트 변조, 악의적인 사이트로 이동 가능
    - 종류
      - Reflected XSS
        - 매개변수를 통해 입력 받은 값을 다시 출력해주는 로직이 있을 때 발생
          - URL로 이용한 요청/응답
      - Stored XSS
        - 게시판의 게시물 등 사용자와 상호작용하여 서버 측에 저장되는 애플리케이션에서 발생

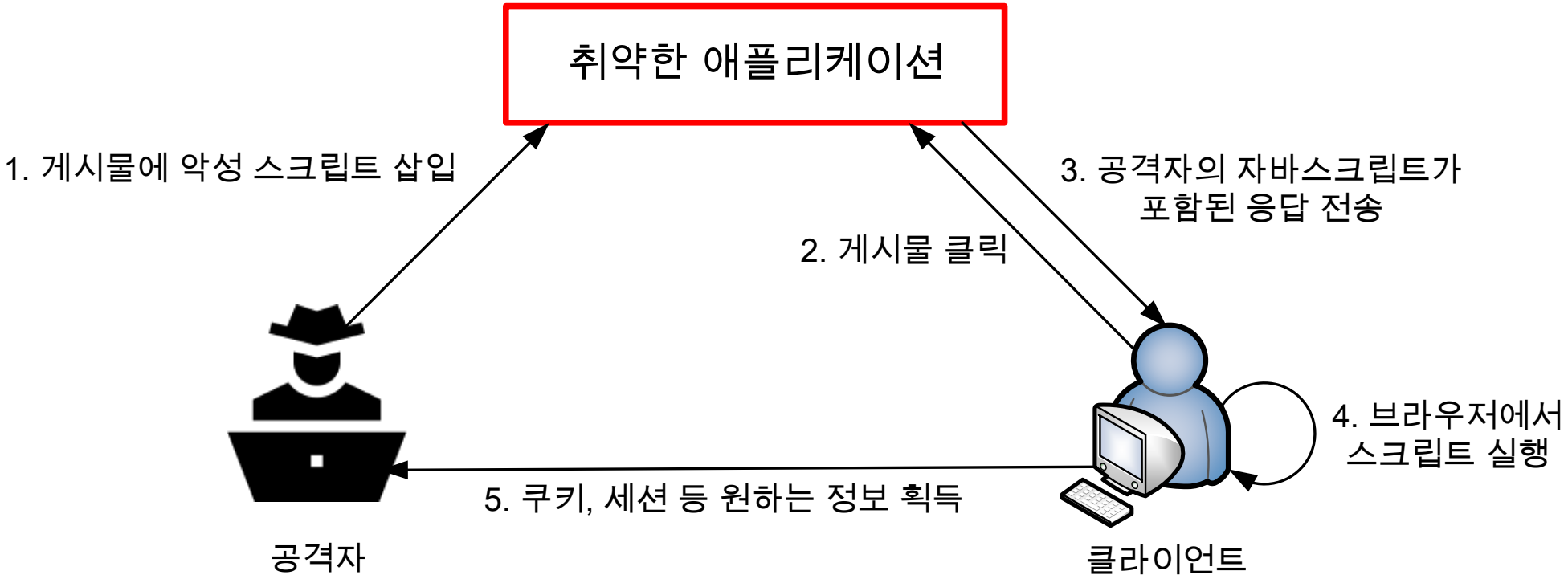
# 웹 보안

- 웹 취약점
  - Reflected XSS
    - 동작 과정



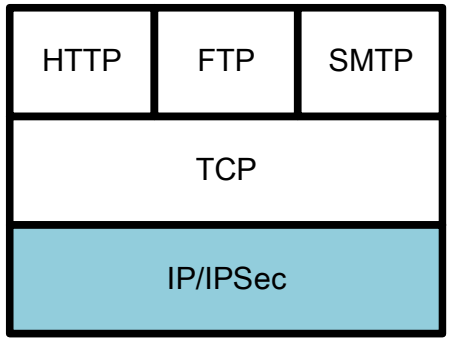
# 웹 보안

- 웹 취약점
  - Stored XSS
    - 동작 과정

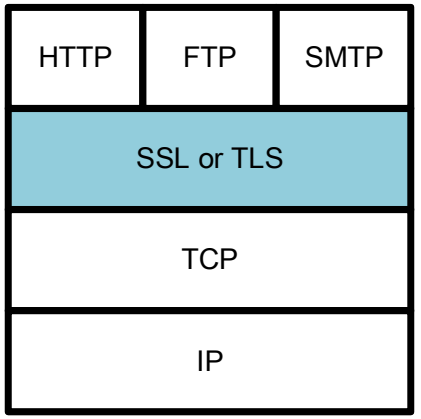


# 웹 보안

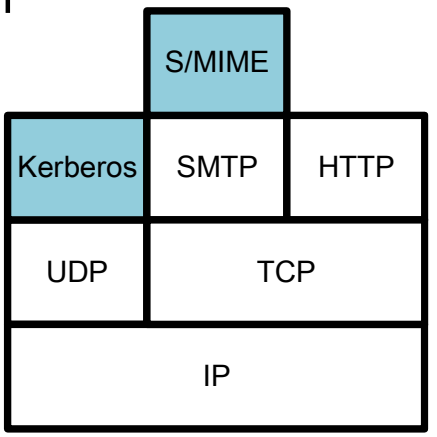
- 웹 트래픽
  - 정의
    - 웹 사이트에 방문한 사람들이 데이터를 주고 받는 양
      - 방문자 수와 방문 페이지 수에 따라 결정됨
  - 웹 트래픽 보안 방법
    - 웹 응용 범위나 TCP/IP 프로토콜마다 다양한 보안 방법이 존재
      - TCP/IP 프로토콜 스택에서 보안 기능의 위치



네트워크 레벨



전송 레벨



응용 레벨

# 웹 보안

- 웹 트래픽 보안 방법
  - 네트워크 레벨
    - IP 보안(IPSec) 사용
      - IP 패킷을 안전하게 보호하기 위해 암호화, 인증 수행
      - IPSec은 필터링 기능을 가지고 있기 때문에 오직 선별된 트래픽에 만 IP 보안 처리를 담당함
  - 전송 레벨
    - 안전 소켓 계층(SSL), 전송 계층 보안(TLS) 사용
      - SSL 또는 TLS를 기존 프로토콜 집합의 일부로 사용하여 응용프로그램에 투명성 제공
  - 응용 레벨
    - 해당 응용프로그램에 보안 서비스를 내장 시킴
    - 응용프로그램의 특정 요구사항에 맞게 서비스 변경 가능

# 목 차

- 웹 보안
- 안전 소켓 계층(SSL)
- 전송 계층 보안(TLS)
- HTTPS
- SSH

# 안전 소켓 계층(SSL)

- Secure Socket Layer
  - 개요
    - 미국의 인터넷, 소프트웨어, 통신 산업 회사인 넷스케이프가 처음 만듦
      - 신뢰할 수 있는 End-to-End의 안전한 서비스를 제공하기 위해 TCP를 사용하여 만들어짐
  - 정의
    - 웹 사이트와 브라우저 사이에 전송된 데이터를 암호화하여 인터넷 연결의 보안을 유지하는 표준 기법

# 안전 소켓 계층(SSL)

- SSL 주요 개념
  - 연결(Connection)
    - OSI 계층 모델에 해당하는 서비스를 제공하기 위해 대등 관계에서의 데이터 전송을 의미
    - 모든 연결은 한 개의 세션과 연관됨
  - 세션(Session)
    - 클라이언트와 서버 사이의 핸드셰이크 프로토콜을 사용한 연결을 의미
    - 각 연결마다 해당하는 새로운 보안 매개변수 협상을 방지하기 위해 사용

# 안전 소켓 계층(SSL)

- SSL 매개변수
  - 연결 상태

| 매개변수 | 설명 |
|---|---|
| 세션 식별자<br>(Session identifier) | 활동 상태나 재시작 할 수 있는 세션 상태를 나타내기 위해 서버가 선택하는 임의의 값 |
| 대등 인증서<br>(Peer Certificate) | 대등의 X509.v3 인증서를 의미 |
| 압축 방법<br>(Compression Method) | 암호화를 하기 전 압축에 사용하는 알고리즘 |
| 암호명세<br>(Cipher Spec) | MAC 계산에 사용되는 용량이 큰 데이터에 대한 암호 알고리즘과 해시알고리즘을 나타냄 |
| 마스터 비밀<br>(Master Secret) | 클라이언트와 서버가 공유하는 48 바이트 비밀키 |
| 재시작 여부<br>(Is Resumable) | 새 연결을 시작하기 위해서 세션을 사용할 수 있는 있는지 아닌지를 나타내는 플래그 |

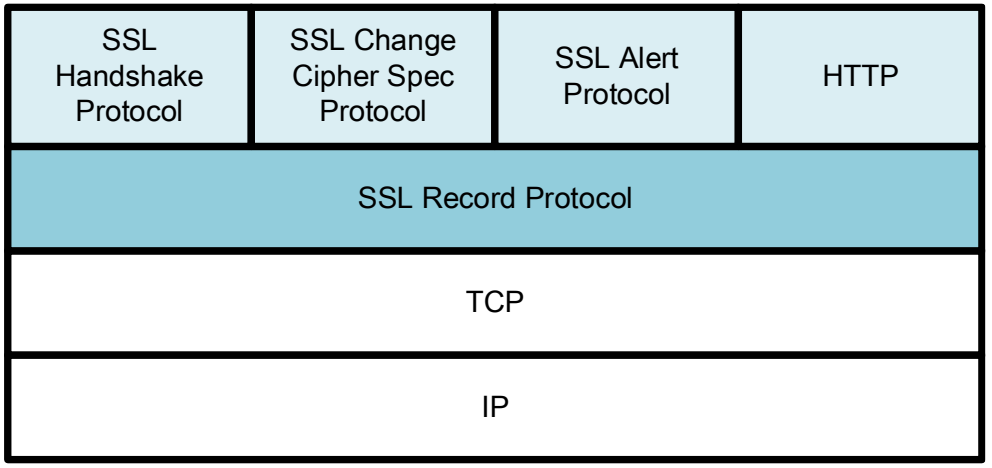
# 안전 소켓 계층(SSL)

- SSL 매개변수
  - 세션 상태

| 매개변수 | 설명 |
|---|---|
| 서버와 클라이언트 랜덤<br>(Server and Client Random) | 각 연결에 사용하려고 서버와 클라이언트가 선택하는 바이트 값 |
| 서버 기록 MAC 비밀<br>(Server Write MAC Secret) | 서버가 보낸 데이터로 MAC를 계산할 때 사용하는 비밀키 |
| 클라이언트 기록 MAC 비밀<br>(Client Write MAC Secret) | 클라이언트가 보낸 데이터로 MAC을 계산할 때 사용하는 비밀키 |
| 서버 기록 키<br>(Server Write Key) | 서버가 데이터를 암호화하고 클라이언트가 복호화 할 때 사용하는 대칭 암호키 |
| 클라이언트 기록<br>(Client Write Key) | 클라이언트가 데이터를 암호화하고 서버가 복호화 할 때 사용하는 대칭 암호키 |
| 초기화 벡터<br>(Initialization Vectors) | SSL 핸드셰이크 프로토콜에 의해 초기화 되는 값으로 각 레코드의 마지막 암호문은 다음 레코드와 함께 사용하기 위해 보존 |
| 순서 번호<br>(Sequence Numbers) | 송·수신되는 각 메시지에 대한 순서 번호 값 |

# 안전 소켓 계층(SSL)

- SSL 구조
  - SSL Handshake Protocol
    - 통신 이전에 필요한 보안 설정을 협상
  - SSL Change Cipher Spec Protocol
    - 보안 설정 협상의 적용 상태를 알림
  - SSL Alert Protocol
    - 세션의 종료 또는 오류 발생 시, 알림
  - SSL Record Protocol
    - 데이터 압축 및 암호화




SSL 프로토콜 스택

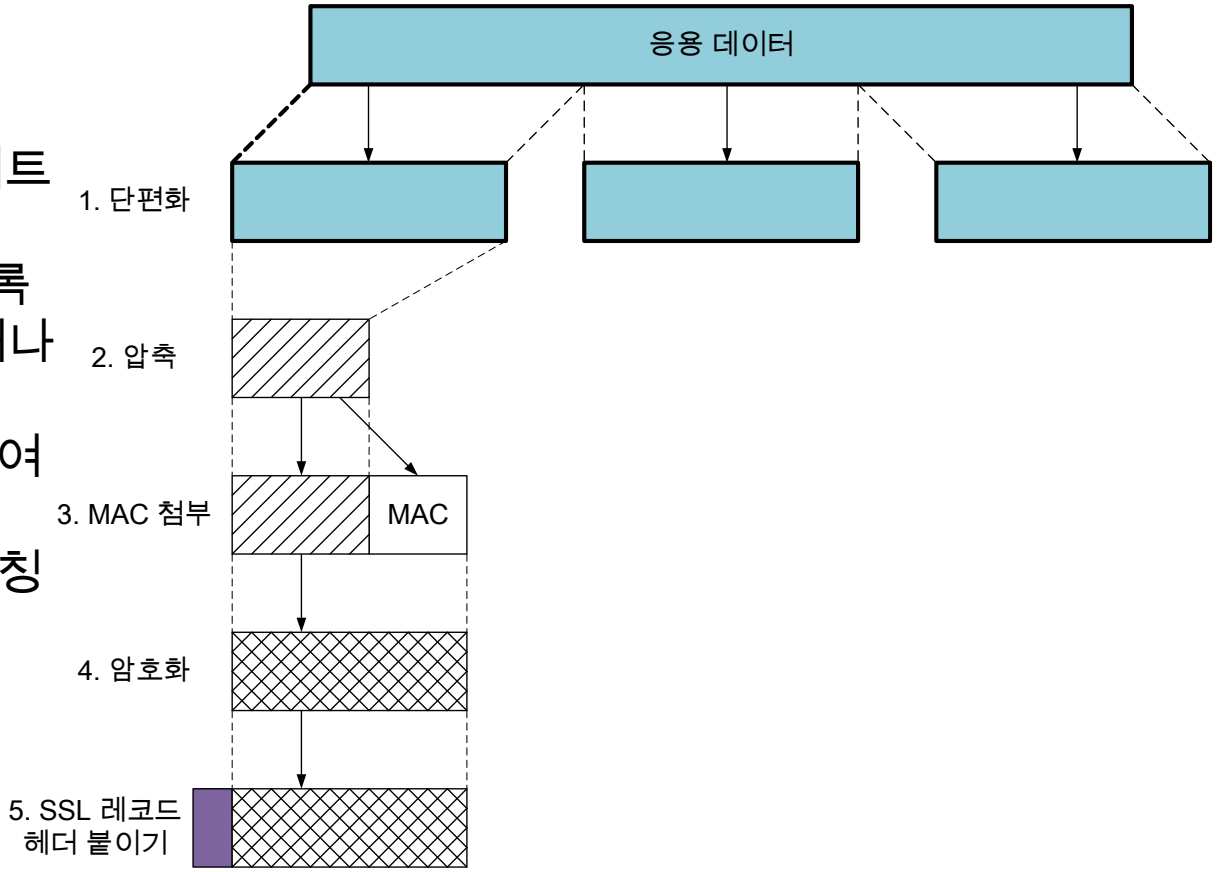
# 안전 소켓 계층(SSL)

- SSL 레코드 프로토콜(SSL Record Protocol) (1/3)
  - 정의
    - 데이터의 압축을 수행하여, 안전한 TCP 패킷으로 변환하는 프로토콜
  - 기능
    - 전송할 응용 메시지를 다룰 수 있는 크기의 블록으로 단편화
    - 데이터 압축, MAC을 적용한 암호화
    - 데이터 암호화 및 인증으로 기밀성 및 무결성 보장

# 안전 소켓 계층(SSL)

- SSL 레코드 프로토콜(SSL Record Protocol) (2/3)
  - 동작 과정

1. 단편화: 상위 계층 메시지를 $2^{14}$바이트 이하 블록으로 단편화
2. 압축: 데이터 손실이 발생하지 않도록 하며, 길이가 1024 바이트 이상 늘어나지 않도록 압축함
3. MAC 첨부: 공유된 비밀키를 이용하여 메시지 인증 코드 계산
4. 암호화: 압축된 메시지와 MAC을 대칭 암호로 암호화
5. SSL 레코드 헤더 붙이기



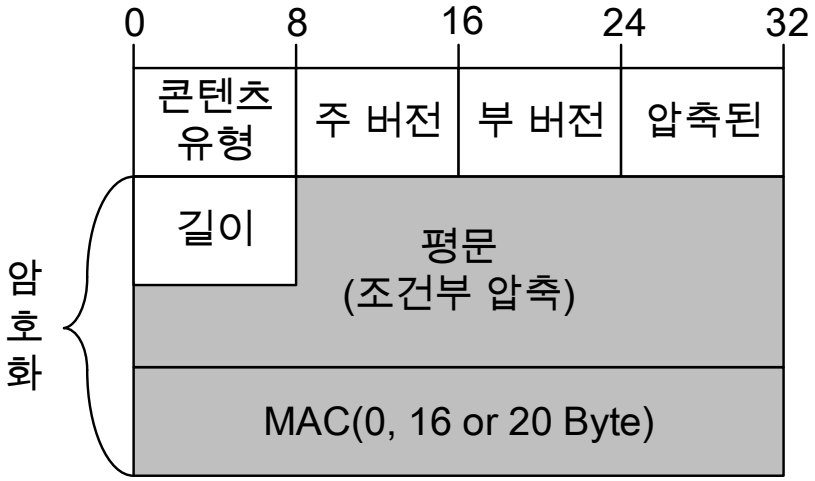
# 안전 소켓 계층(SSL)

- SSL 레코드 프로토콜(SSL Record Protocol) (3/3)
  - MAC 계산

$$hash(MAC\_Write\_Secret \,||\, pad\_2 \,||\, hash(MAC\_Write\_Secret \,||\, pad\_1 \,||\, Seq\_num \,||\, SSLCompressed.type \,||\, SSLCompressed.length \,||\, SSLCompressed.fragment))$$

| 용어 | 설명 |
|---|---|
| $\|\|$ | 메시지 이어 붙여쓰기 |
| $MAC\_Write\_Secret$ | 공유 비밀키 |
| $hash$ | 해시 알고리즘 |
| $pad\_1, pad\_2$ | 공유 비밀키와 함께 연산 되는 패드 값 |
| $Seq\_num$ | 메시지의 순서번호 |
| $SSLCompressed.type$ | 단편 처리에 사용되는 상위계층 프로토콜 |
| $SSLCompressed.length$ | 압축된 단편의 길이 |
| $SSLCompressed.fragment$ | 압축된 단편 |

  - 포맷



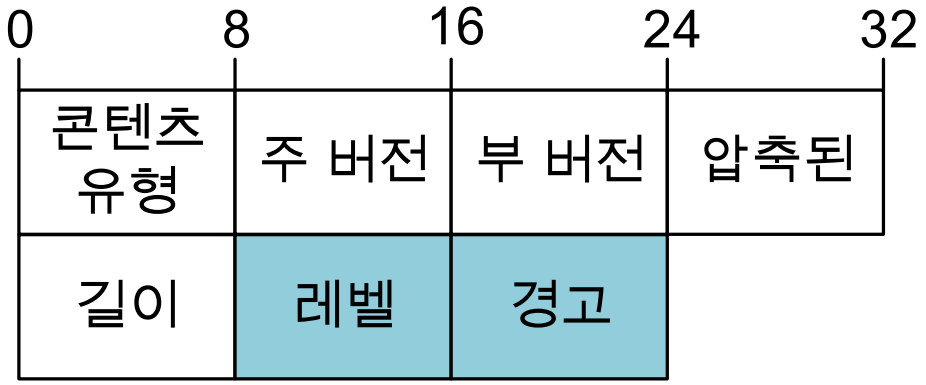

| 필드 | 설명 |
|---|---|
| 콘텐츠 유형 | 단편을 처리할 때 사용하는 상위 계층 프로토콜 식별 |
| 주 버전 | 사용 중인 SSL 주 버전 (버전 3일 경우 3) |
| 부 버전 | 사용 중인 서브 버전 (버전 3일 경우 0) |
| 압축된 길이 | 평문 단편의 바이트 단위 길이 (최대값: $2^{14} + 2048$) |

# 안전 소켓 계층(SSL)

- 암호명세 변경 프로토콜(Change Cipher Spec Protocol)
  - 정의
    - 핸드셰이크 프로토콜에서 암호화 ,키 교환, MAC, 해시 알고리즘이 합의 된 사실을 알리는 프로토콜
  - 포맷

# 안전 소켓 계층(SSL)

- 경고 프로토콜(Alert Protocol) (1/2)
  - 정의
    - 클라이언트와 웹서버 중 에러, 세션 종료, 비정상적인 동작 발생 시, 사용되는 프로토콜
  - 포맷




| 필드 | 설명 |
|---|---|
| 레벨 | 위험(Warning), 심각(Fatal) 2가지 값을 가짐 |
| 경고 | 특정 경고를 나타내는 코드 삽입 |

# 안전 소켓 계층(SSL)

- 경고 프로토콜(Alert Protocol) (2/2)
  - 항상 심각(Always Fatal) 경고

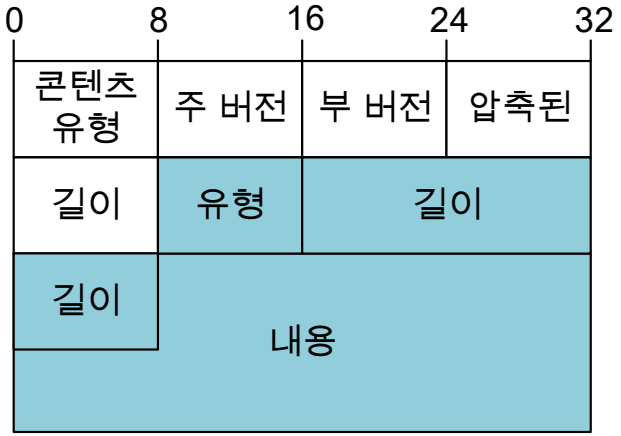
| 경고 | 설명 |
|---|---|
| Unexpected_Message | 적합하지 않은 메지시의 수신 |
| Bad_Record_MAC | 부정확한 MAC 수신 |
| Decompressed_Failure | 압축 해제 함수에 적합하지 않은 입력<br>(압축 풀기 불가 또는 최대 허용길이 보다 큰 경우 등) |
| Handshake_Failure | 사용할 수 있는 옵션이 주어졌지만 송신자와 협상 불가 |
| Illegal_Parameter | 핸드셰이크 메시지 안의 한 필드가 범위 밖이거나 다른 필드와 맞지 않음 |

  - 일반적인 경고

| 경고 | 설명 |
|---|---|
| Close_Notify | 송신자가 이 연결에서 더 이상 메시지를 보내지 않을 것을 수신자에게 알림 |
| No_Certificate | 인증서 요청에 대한 응답으로 자신의 인증서가 없는 것을 알림 |
| Bad_Certificate | 수신된 인증서가 오류가 존재하는 것을 알림 |
| Unsupported_Certificate | 수신된 인증서의 유형을 지원하지 않는 것을 알림 |

# 안전 소켓 계층(SSL)

- 핸드셰이크 프로토콜(Handshake Protocol)
  - 정의
    - 안전한 세션을 구축하기 위한 설정들을 합의하는 프로토콜
  - 기능
    - 한 세션동안 이용되는 암호 매개변수 생성
    - 서버와 클라이언트 간의 인증 제공
    - 암호화 통신에 사용할 암호 알고리즘, 키 교환 알고리즘, MAC 암호화, 해시 알고리즘들을 협상
  - 포맷

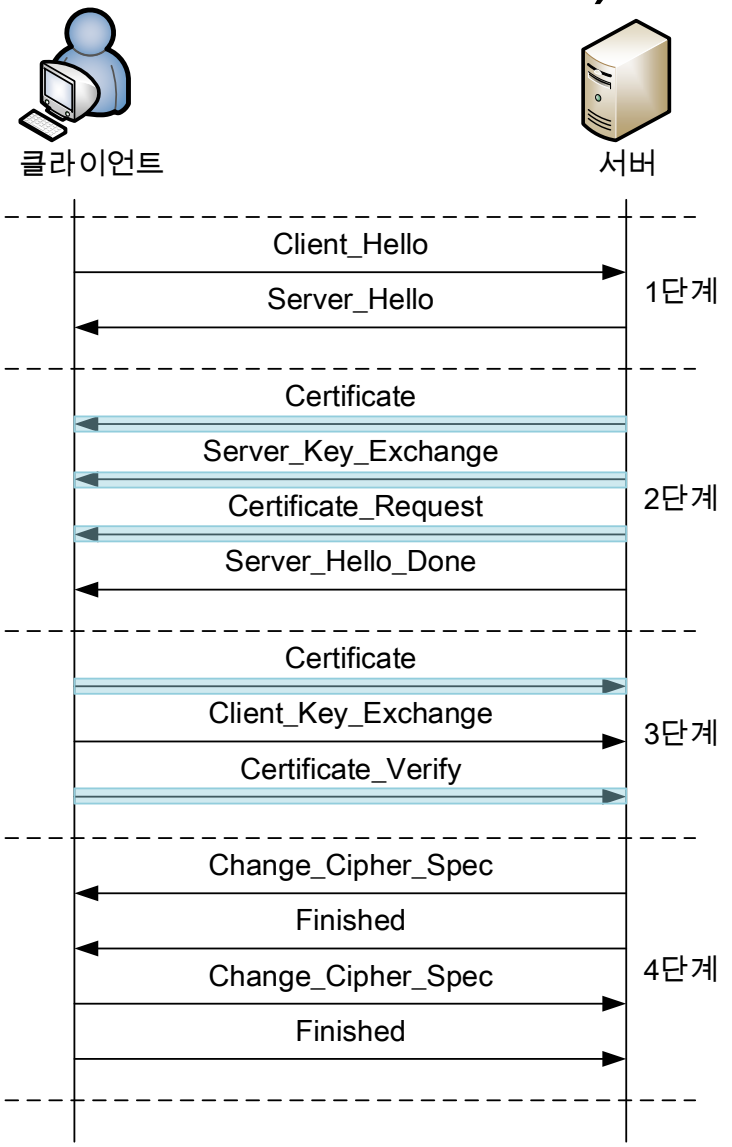



| 필드 | 설명 |
|---|---|
| 유형 | 핸드셰이크프로토콜 메시지 식별 |
| 길이 | 메시지 길이 |
| 내용 | 유형에 해당하는 내용 |

# 안전 소켓 계층(SSL)

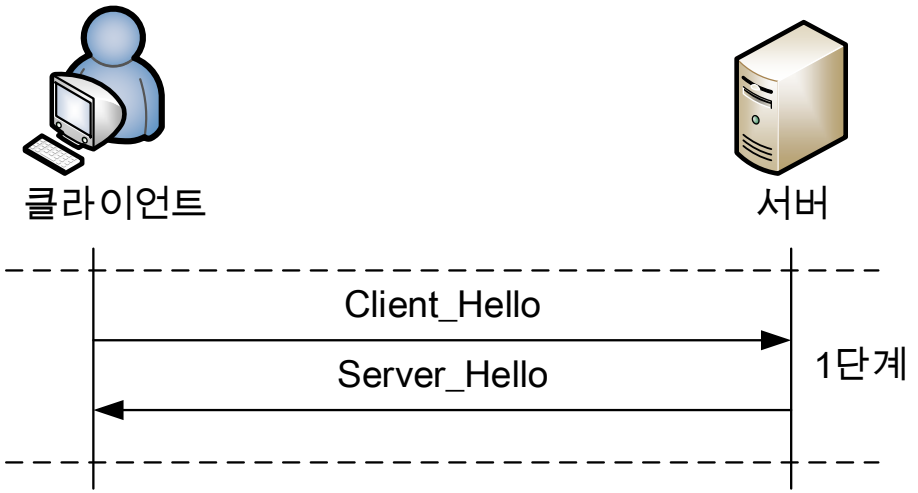
- 핸드셰이크 프로토콜(Handshake Protocol)
  - 동작과정
    - 파란부분은 옵션을 의미

| 메시지 유형 | 매개변수 |
|---|---|
| Hello_Request | 없음(Null) |
| Client_Hello | 버전(Version), 랜덤(Random), 세션 ID(Session ID), 암호도구 (Cipher Suite), 압축 방법 (Compression Method) |
| Server_Hello | |
| Certificate | 연속된 X.509v3 인증서 (Chain of X.509v3 Certificate) |
| Server_Key_Exchange | 매개변수(Parameters), 서명(Signature) |
| Certificate_Request | 유형(Type), 기관(Authorities) |
| Server_Done | 없음(Null) |
| Certificate_Verify | 서명(Signature) |
| Client_Key_Exchange | 매개변수(Parameters), 서명(Signature) |
| Finished | 해시 값(Hash Value) |



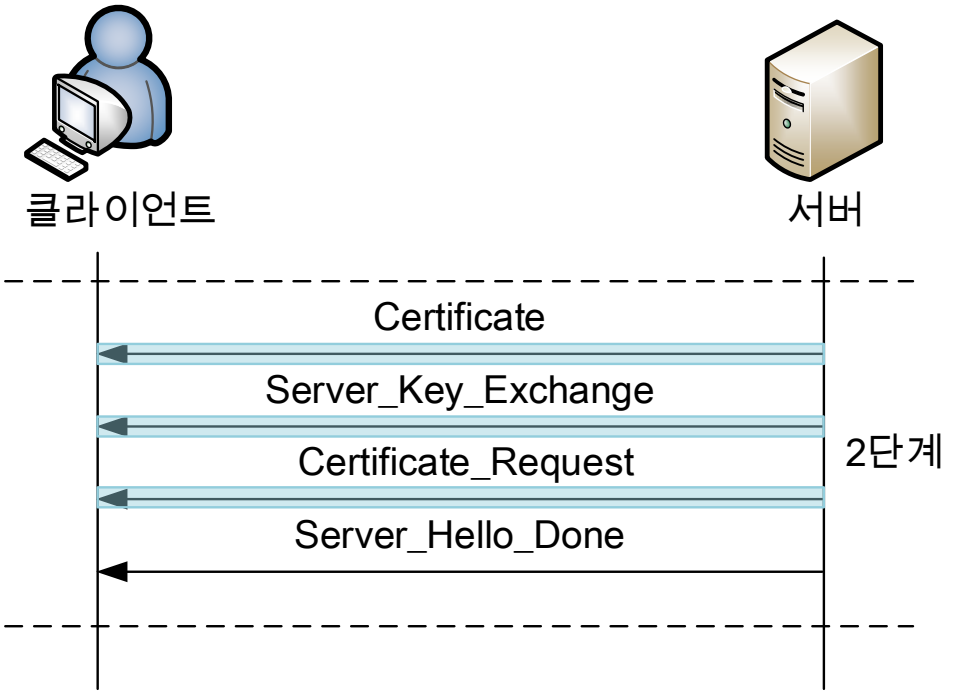
# 안전 소켓 계층(SSL)

- 핸드셰이크 프로토콜(Handshake Protocol)
  - 동작과정
    - 1단계: 보안 기능 설정
      1. Client_Hello: 지원 가능한 암호 방식, 키 교환 방식,서명 방식, 압축 방식을 서버에게 알림
      2. Server_Hello: 수용 가능한 암호 방식, 키 교환 방식,서명 방식, 압축 방식을 응답. 또한, 새로운 세션 ID 할당



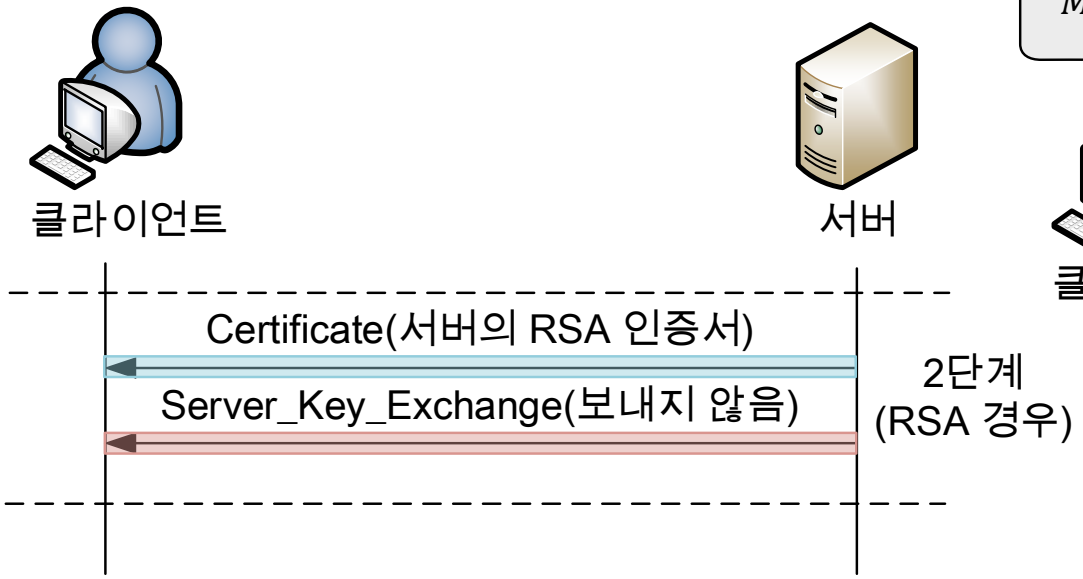
# 안전 소켓 계층(SSL)

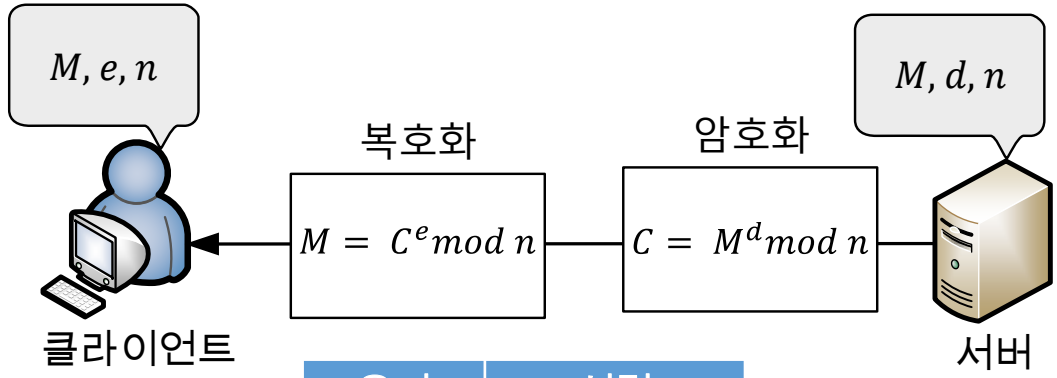
- 핸드셰이크 프로토콜(Handshake Protocol)
  - 동작과정
    - 2단계: 서버 인증과 키 교환
      1. 서버의 인증서 전달
      2. 키 교환 알고리즘에 따라, Server_Key_Exchange 송신
         - RSA, 익명 Diffie_Hellman, 임시 Diffie_Hellman, 고정 Diffie_Hellman 등
      3. 클라이언트의 인증서 요청
      4. Server_Done 메시지 전송



# 안전 소켓 계층(SSL)

- 핸드셰이크 프로토콜(Handshake Protocol)
  - 동작과정
    - 2단계: 서버 인증과 키 교환(RSA)
      1. 서버의 인증서 전송
      2. Server_Key_Exchange는 전송하지 않음

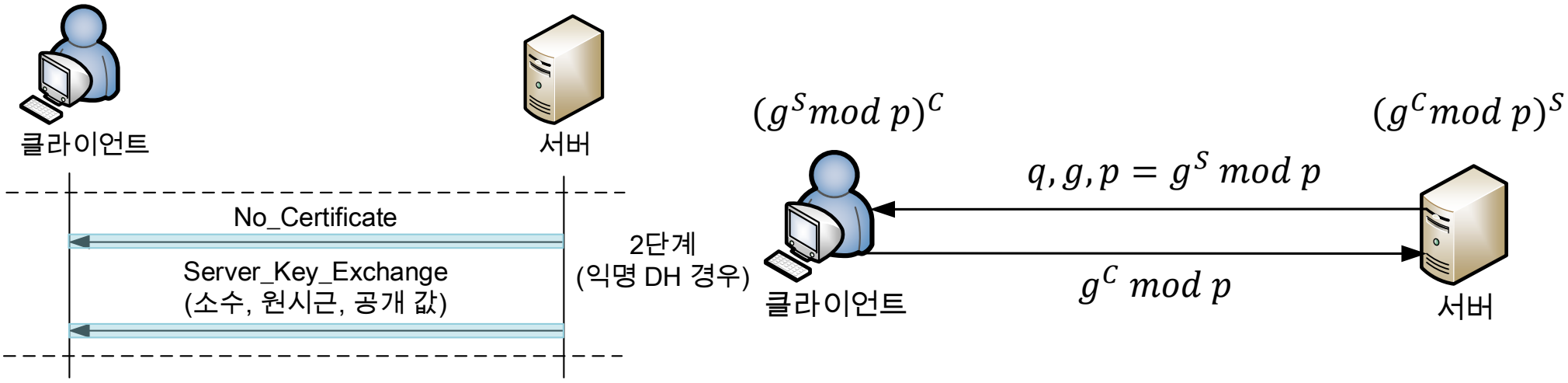



| 용어 | 설명 |
|---|---|
| $C$ | 암호문 |
| $M$ | 평문 |
| $d$ | 개인키 |
| $e$ | 공개키 |
| $n$ | 소수의 곱 |

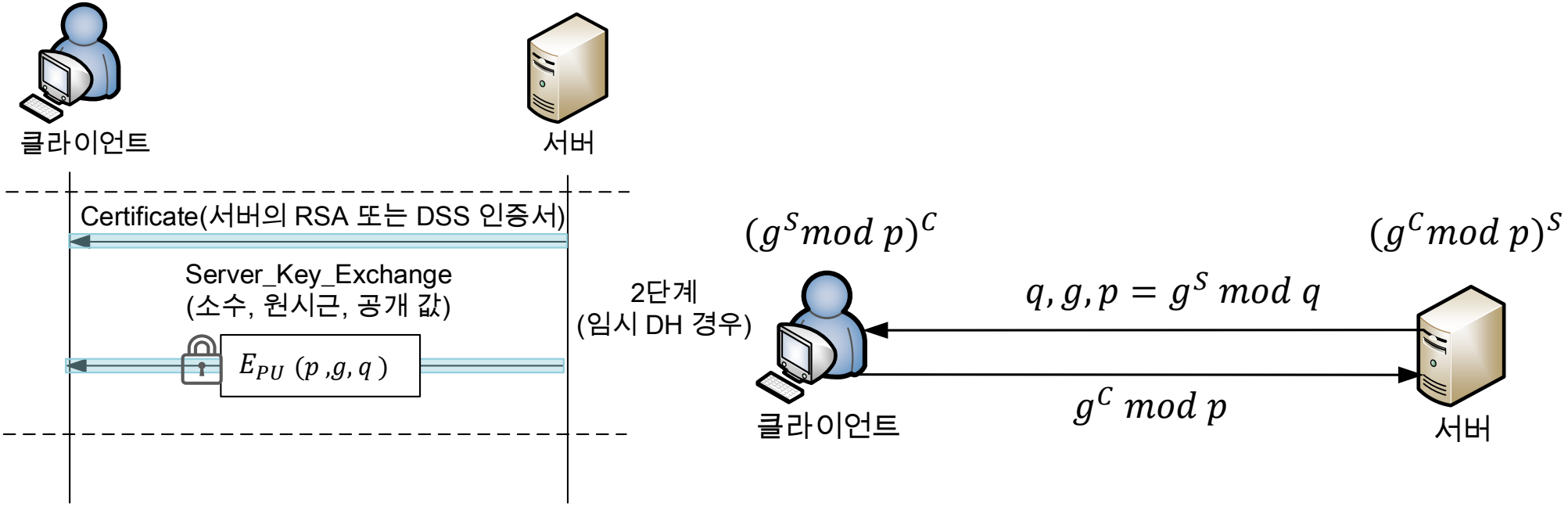
# 안전 소켓 계층(SSL)

- 핸드셰이크 프로토콜(Handshake Protocol)
  - 동작과정
    - 2단계: 서버 인증과 키 교환(익명 Diffie_Hellman)
      1. No_Certifiacet 메시지 전송
      2. Server_Key_Exchange 메시지로 Diffie_Hellman에 사용되는 소수($q$), 원시근($g$), 공개 값($p$)를 전송
      - Pre_Master_Secret 계산: $g^{CS} \ mod \ p$



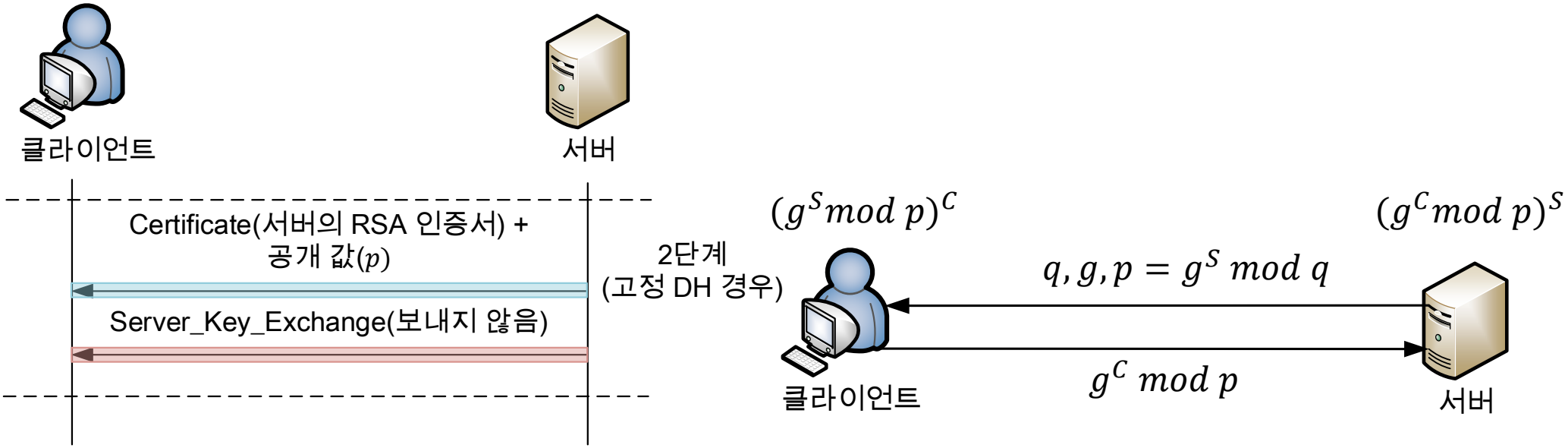
# 안전 소켓 계층(SSL)

- 핸드셰이크 프로토콜(Handshake Protocol)
  - 동작과정
    - 2단계: 서버 인증과 키 교환(임시 Diffie_Hellman)
      1. RSA 또는 DSS(Digital Signature Standard) 인증서 전송
      2. Server_Key_Exchange 메시지로 Diffie_Hellman에 사용되는 소수($q$), 원시근($g$), 공개 값($p$)를 공개키로 암호화($E_{PU}$)하여 전송
    - Pre_Master_Secret 계산: $g^{CS}$



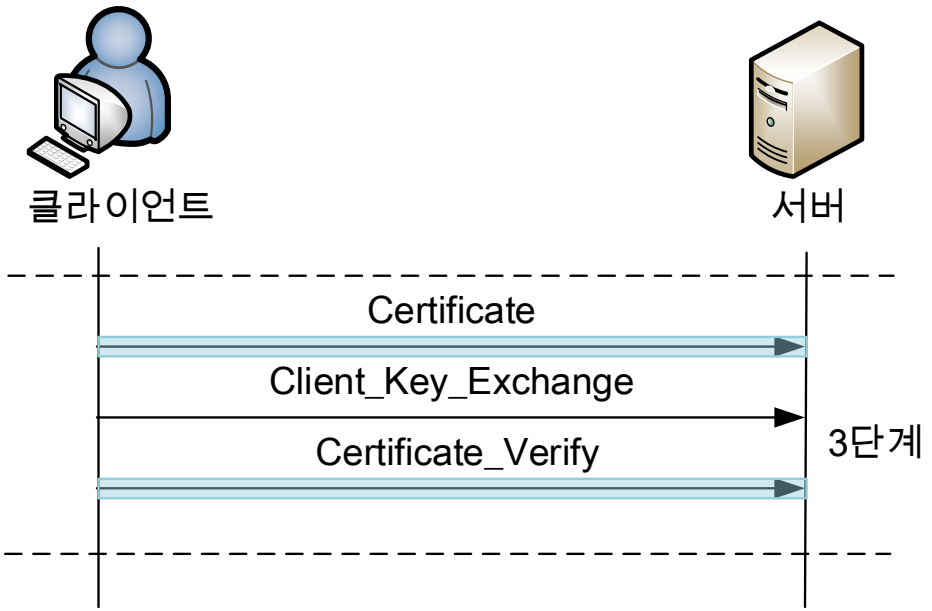
# 안전 소켓 계층(SSL)

- 핸드셰이크 프로토콜(Handshake Protocol)
  - 동작과정
    - 2단계: 서버 인증과 키 교환(고정 Diffie_Hellman)
      1. RSA 또는 DSS 인증서와 함께 공개 값($p$) 전송
      2. Server_Key_Exchange 메시지는 전송하지 않음
      - Pre_Master_Secret 계산: $g^{cs}\ mod\ p$



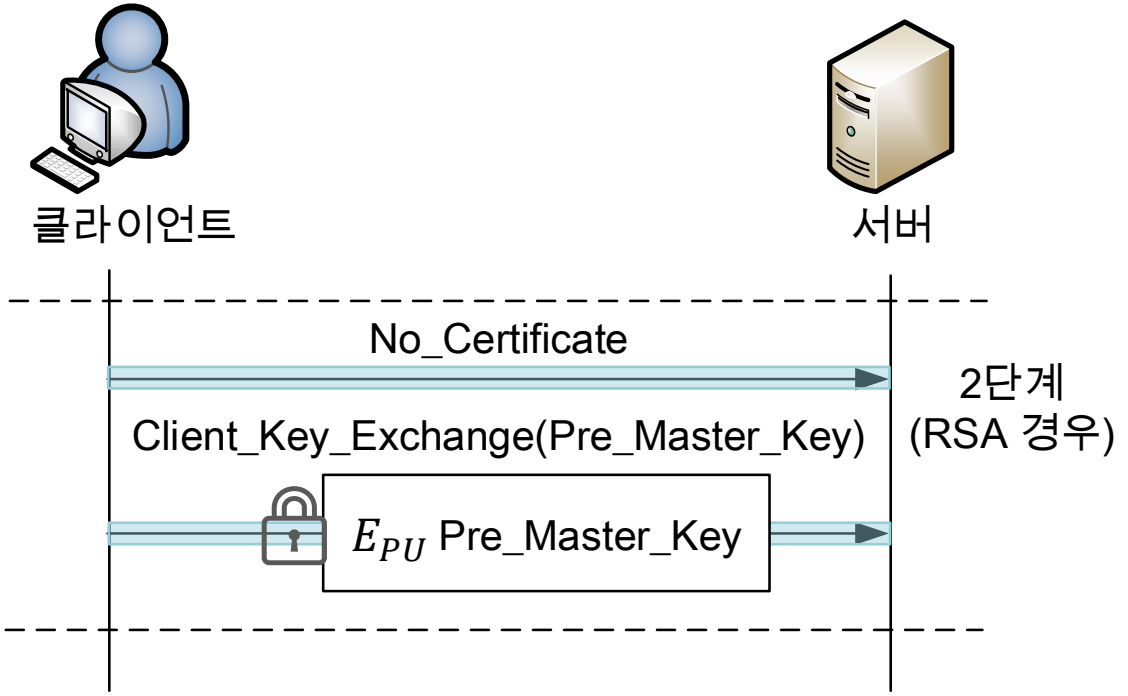
# 안전 소켓 계층(SSL)

- 핸드셰이크 프로토콜(Handshake Protocol)
  - 동작과정
    - 3단계: 클라이언트 인증과 키 교환
      1. 클라이언트의 인증서 전달
      2. 키 교환 알고리즘에 따른 응답 값과 Pre_Master_Key 전송
         - RSA, 익명 Diffie_Hellman, 임시 Diffie_Hellman, 고정 Diffie_Hellman 등
      3. 클라이언트인증서의 확인을 위한 Certificate_Verify 메시지를 보냄



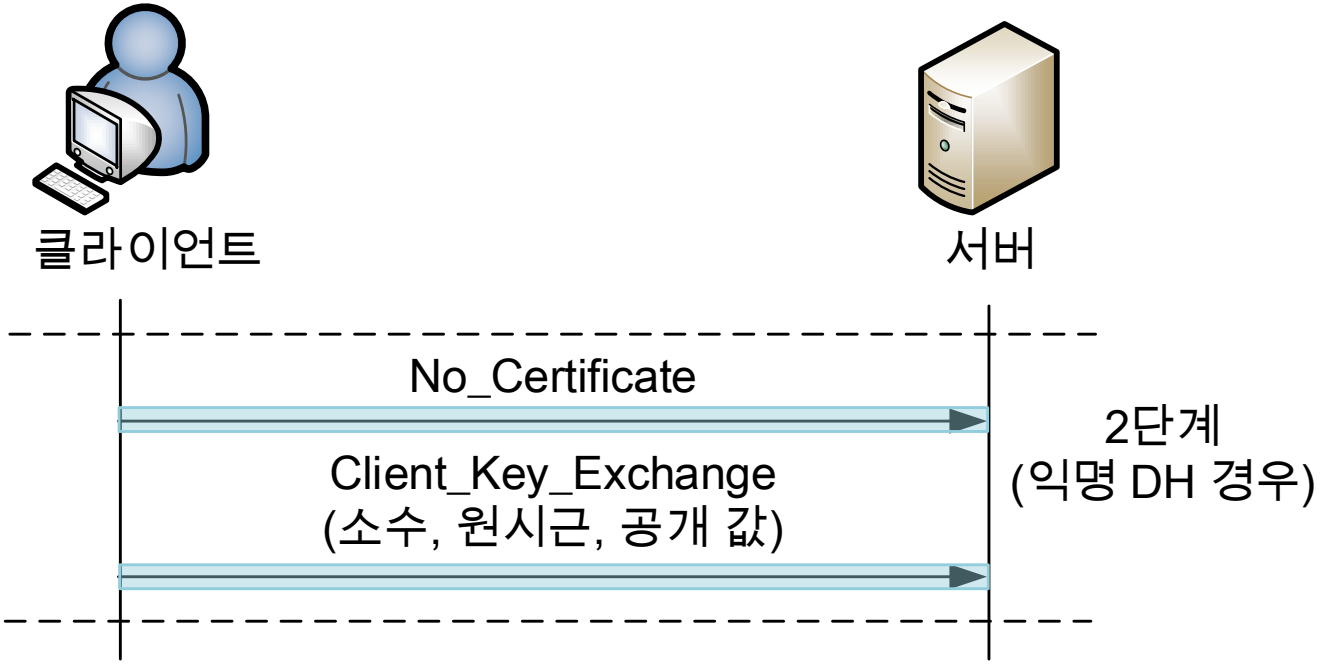
# 안전 소켓 계층(SSL)

- 핸드셰이크 프로토콜(Handshake Protocol)
  - 동작과정
    - 3단계: 클라이언트 인증과 키 교환(RSA)
      1. No_Certificate 전송
      2. Pre_Master_Secret 계산: Client_Version + Nonce
      3. 서버의 공개키($PU$)로 Pre_Master_Secret를 암호화($E$)하여 전송



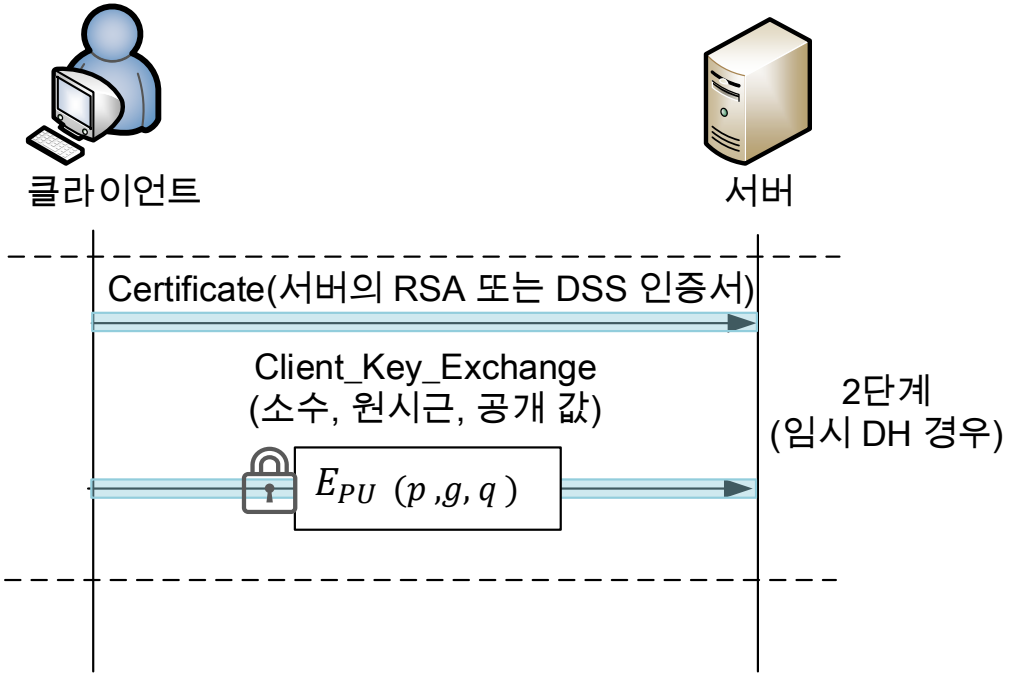
# 안전 소켓 계층(SSL)

- 핸드셰이크 프로토콜(Handshake Protocol)
  - 동작과정
    - 3단계: 클라이언트 인증과 키 교환(익명 Diffie_Hellman)
      1. No_Certificate 전송
      2. Client_Key_Exchange 메시지로 Diffie_Hellman에 사용되는 소수($q$), 원시근($g$), 공개 값($p$)를 전송



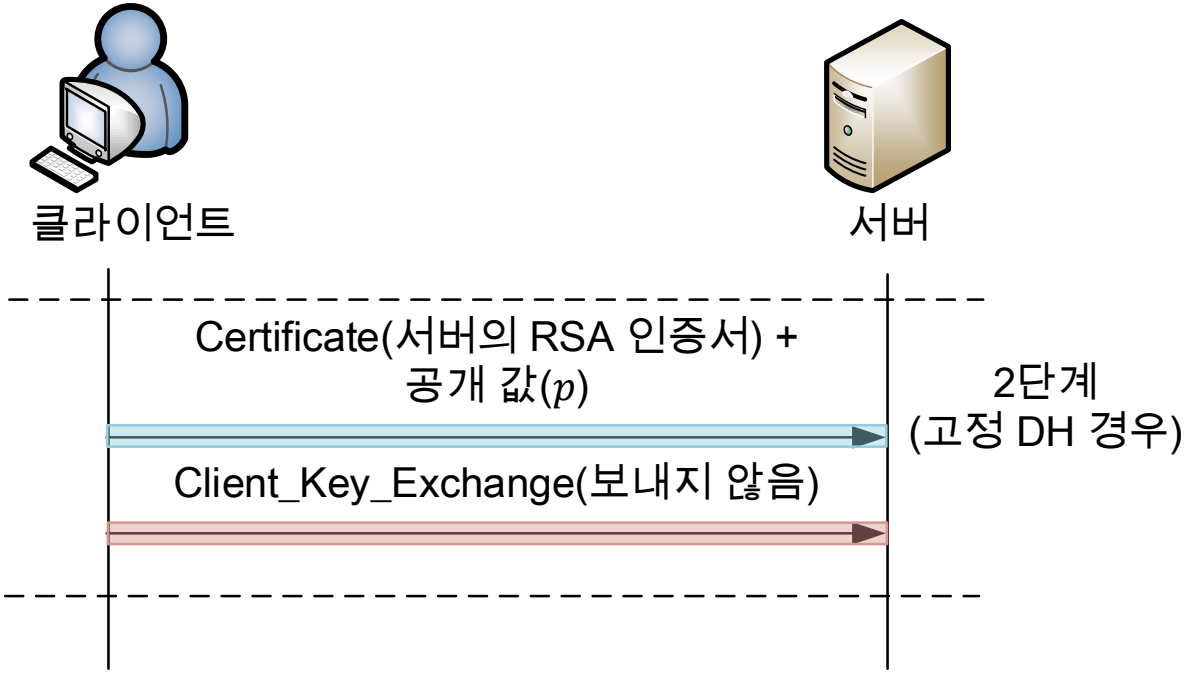
# 안전 소켓 계층(SSL)

- 핸드셰이크 프로토콜(Handshake Protocol)
  - 동작과정
    - 3단계: 클라이언트 인증과 키 교환(임시 Diffie_Hellman)
      1. RSA 또는 DSS 인증서 전송
      2. Client_Key_Exchange 메시지로 Diffie_Hellman에 사용되는 소수($q$), 원시근($g$), 공개 값($p$)를 공개키로 암호화($E_{PU}$)하여 전송



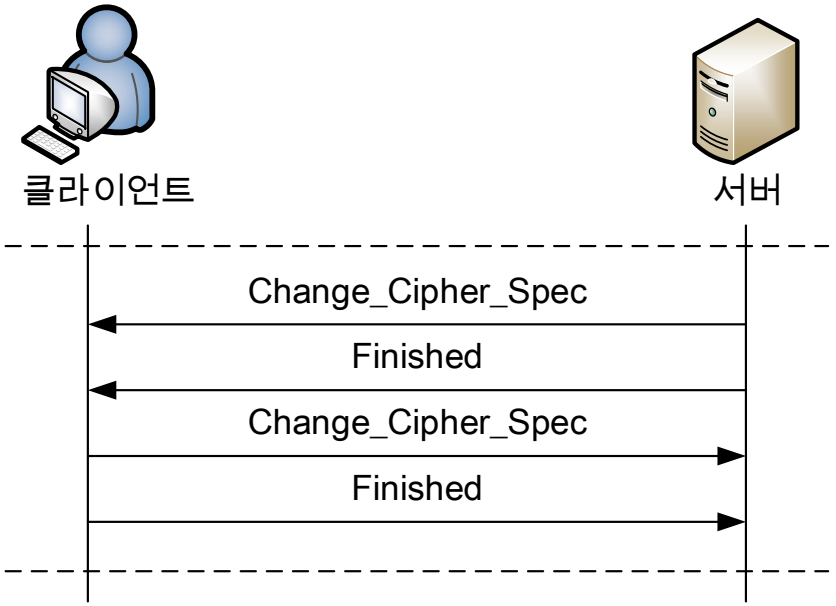
# 안전 소켓 계층(SSL)

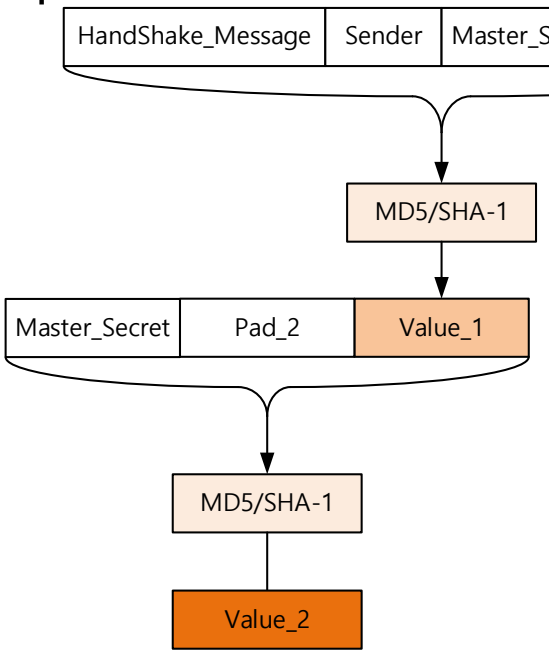
- 핸드셰이크 프로토콜(Handshake Protocol)
  - 동작과정
    - 3단계: 클라이언트 인증과 키 교환(고정 Diffie_Hellman)
      1. RSA 또는 DSS 인증서와 함께 공유 변수($p$) 전송
      2. Client_Key_Exchange 메시지는 전송하지 않음



# 안전 소켓 계층(SSL)

- 핸드셰이크 프로토콜(Handshake Protocol)
  - 동작과정
    - 4단계: 종료
      1. 변경되거나 합의된 암호 방식, 키 교환,서명, 압축 방식을 알림
      2. 핸드셰이크 종료를 알리고 암호화 하여 Finished 메시지 송신
      3. Finished를 수신한 개체는 이를 복호화하여 협상한 암호화 및 압축방식에 대해 검사



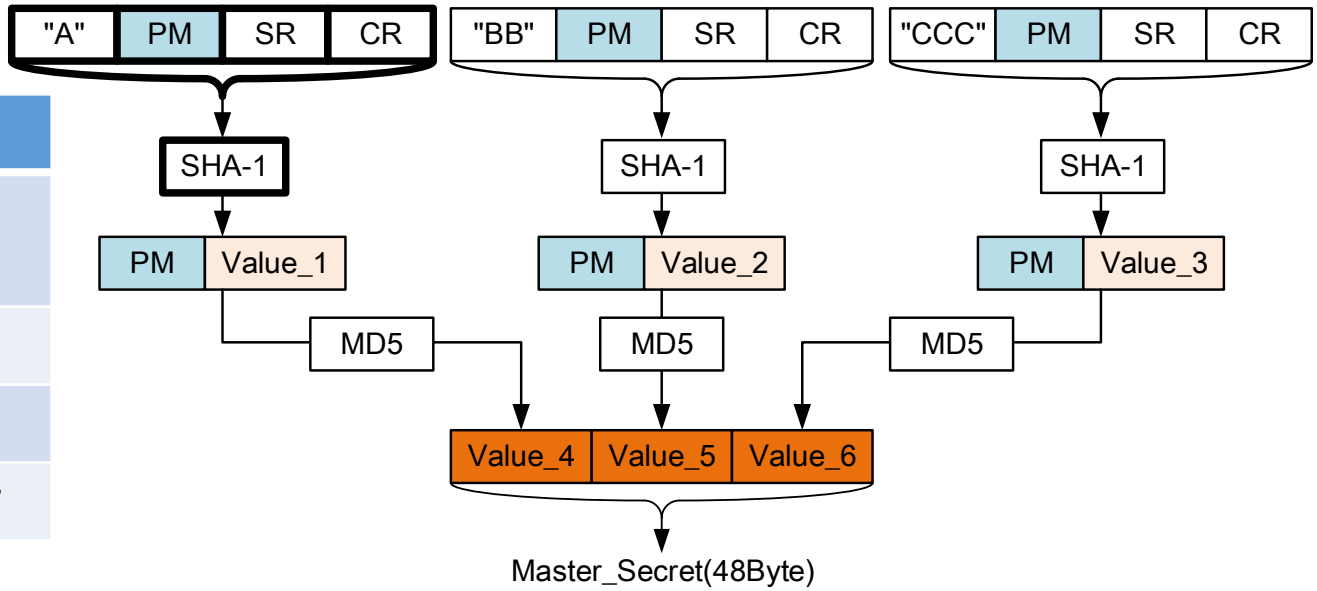




| 메시지 | 설명 |
|---|---|
| Handshake _Message | 지금까지의 모든 핸드셰이크 메시지 데이터를 나타냄 |
| Sender | 보내는 주체를 나타냄 |
| Master_ Secret | Pre_Master_Key 값으로 생성된 메시지 |

# 안전 소켓 계층(SSL)

- 마스터 비밀키($master\_secret$)
  - 서버와 클라이언트 간에 세션을 위해 키 교환 방법으로 생성한 48 바이트 값
  - 생성 과정
    1. 핸드셰이크 과정에서 $pre\_master\_secret$ 교환
    2. $master_secret$을 서버와 클라이언트에서 각각 계산
       - $pre\_master\_secret$, Client_Hello.Random, Server_Hello.Random 값 사용

| 용어 | 설명 |
|---|---|
| “A”, “BB”, “CCC” | 블록 식별을 위한 문자열 입력 값 |
| PM | Pre_Master_Key |
| SR | 서버의 랜덤 값 |
| CR | 클라이언트의 랜덤 값 |

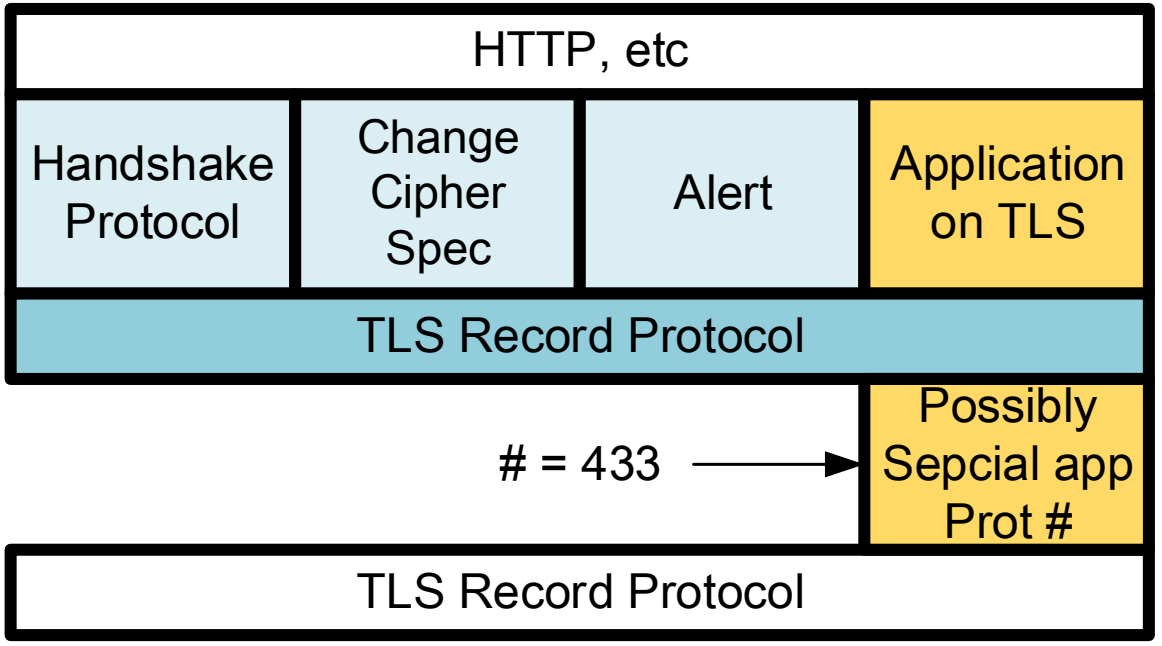


# 목 차

- 웹 보안
- 안전 소켓 계층(SSL)
- 전송 계층 보안(TLS)
- HTTPS
- SSH

# 전송 계층 보안(TLS)

- Transport Layer Security
  - 개요
    - IETF(Internet Engineering Task Force)에서 기존 SSL을 기반으로 개발한 표준
  - 구조



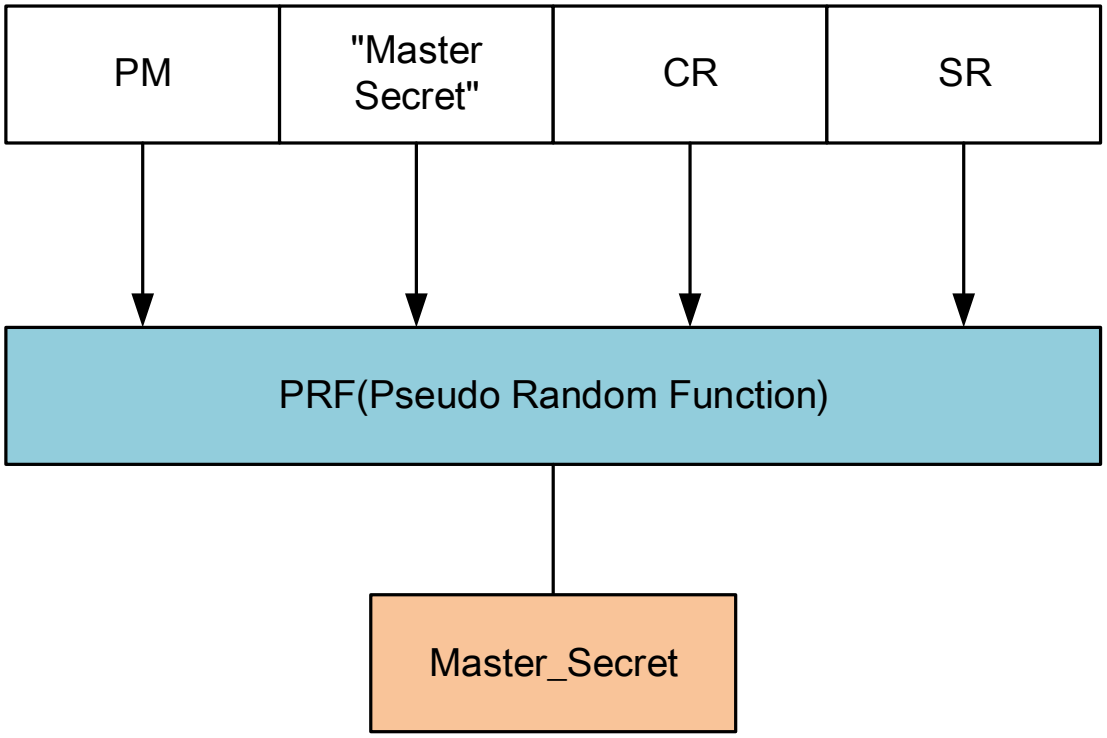
# 전송 계층 보안(TLS)

- Transport Layer Security
  - 차이점
    - 레코드 형식
      - SSL의 레코드 형식과 동일
        - 주 버전: 3 표기
        - 부 버전: 1 표기
    - 암호도구
      - 키 교환 알고리즘: fortezza는 지원하지 않음
      - 대칭 암호 알고리즘: fortezza는 지원하지 않음
        - Fortezza: 미국 정부에서 개발된 암호화를 위한 PC 카드
          - 해시 함수, 개인키, 카드 내부적으로 생성한 메시지를 기반으로 암호화

# 전송 계층 보안(TLS)

- Transport Layer Security
  - 차이점
    - Master Secret 암호 계산
      - 의사 랜덤 함수(PRF, Pseudo Random Function) 적용
      - 구조

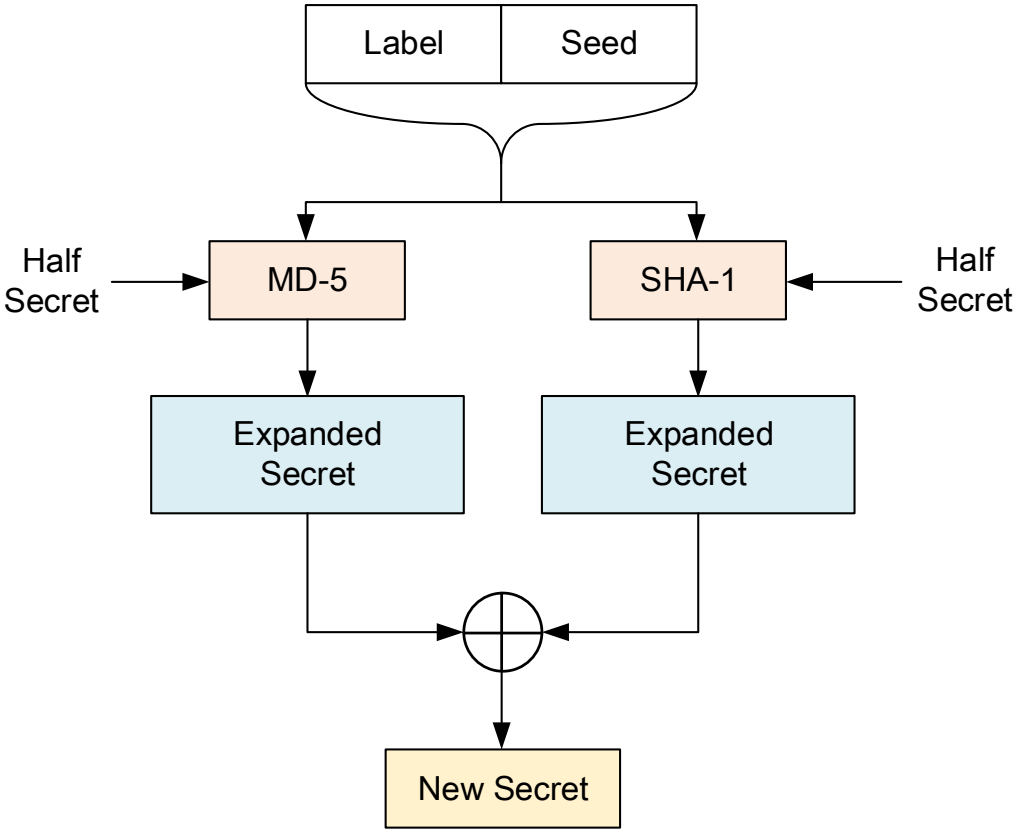
| 매개 변수 | 설명 |
|---|---|
| PM | Pre_Master_Key |
| “Master Secret” | 문자열 입력 값 |
| CR | 클라이언트의 랜덤 값 |
| SR | 서버의 랜덤 값 |



# 전송 계층 보안(TLS)

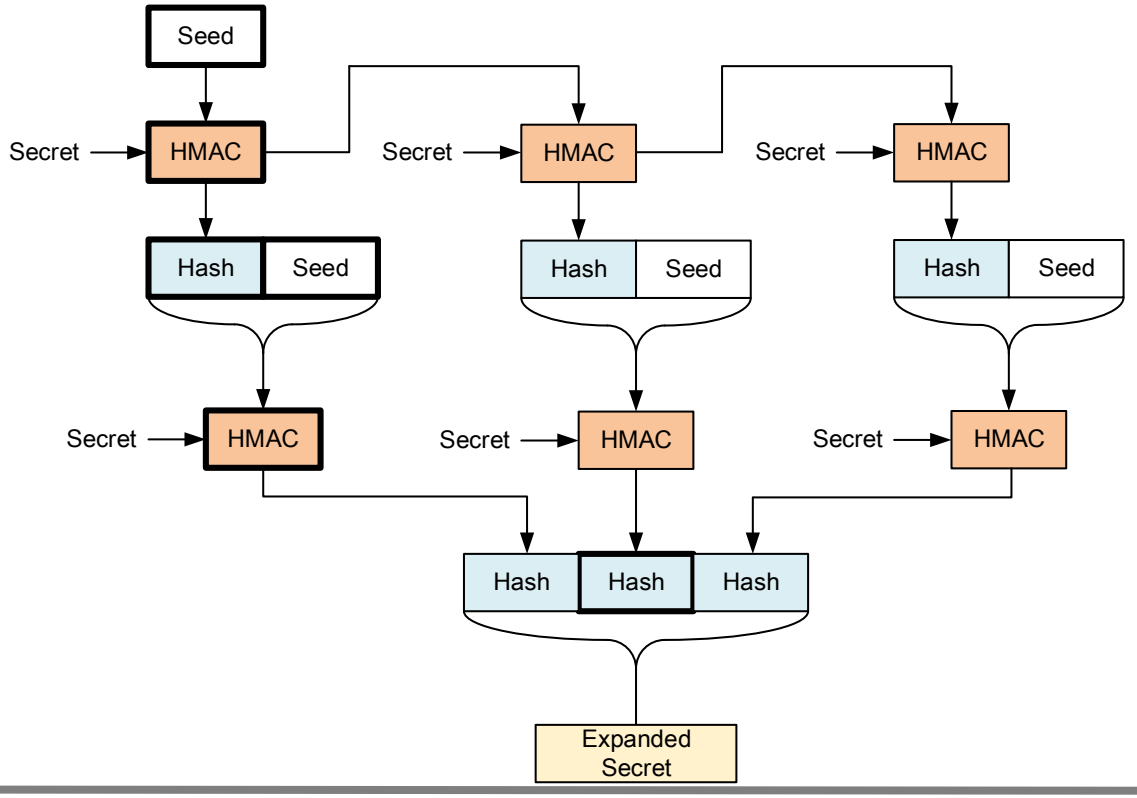
- Transport Layer Security
  - 차이점
    - PRF(Pseudo Random Function) 사용
      - RPF와 MD-5/SHA-1과 결합되어 새로운 비밀 값 생성
      - 구조

| 매개변수 | 설명 |
|---|---|
| Label | 매개 변수 식별을 위한 입력 값 (e.g., “Master Secret”) |
| Seed | 종자 값 (e.g., CR, SR) |
| Half Secret | 기존 비밀 값의 반 |
| Expanded Secret | 확장된 비밀 값 |



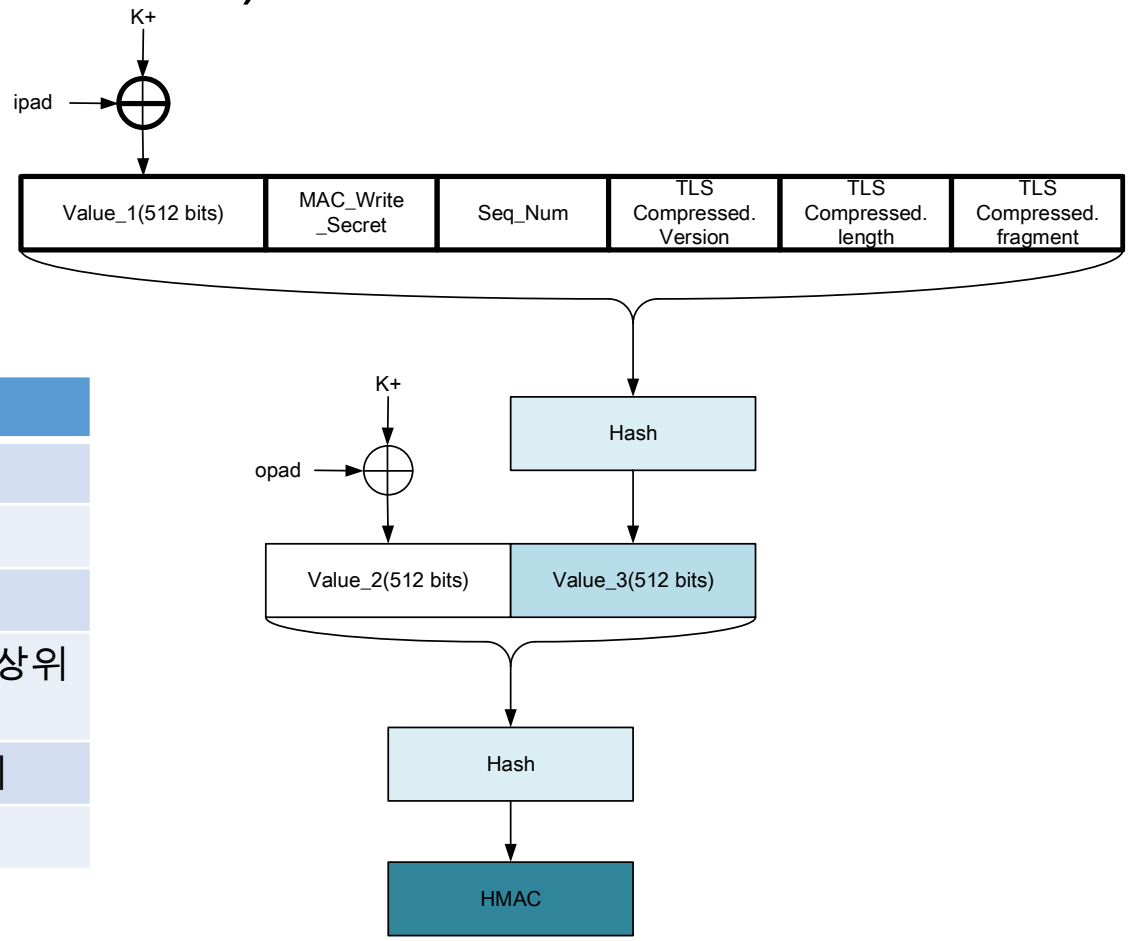
# 전송 계층 보안(TLS)

- Transport Layer Security
  - 차이점
    - PRF(Pseudo Random Function) 사용
      - 데이터 확장 함수
        - 비밀 값을 길게 확장하기 위한 함수
          - HMAC 사용
        - 구조



# 전송 계층 보안(TLS)

- Transport Layer Security
  - 차이점
    - PRF(Pseudo Random Function) 사용
      - 데이터 확장 함수
        - HMAC 구조




| 매개 변수 | 설명 |
|---|---|
| MAC_Write_Secret | 공유 비밀키 |
| Hash | 해시 알고리즘 |
| Seq_Num | 메시지의 순서번호 |
| TLSCompressed.Version | 단편 처리에 사용되는 상위 계층 프로토콜 |
| TLSCompressed.length | 압축된 단편의 길이 |
| TLSCompressed.Fragment | 압축된 단편 |

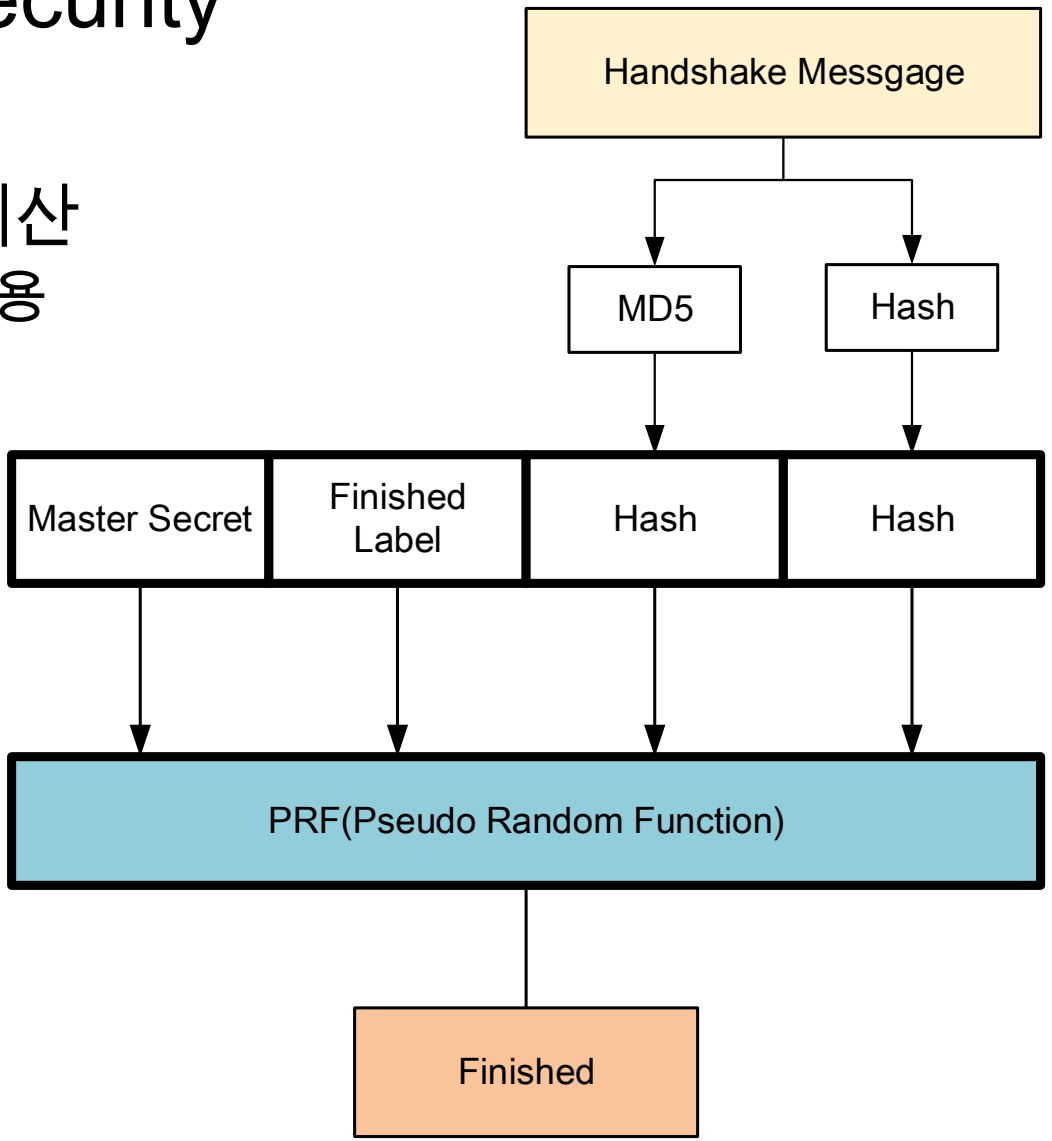
# 전송 계층 보안(TLS)

- Transport Layer Security
  - 차이점
    - 경고 코드
      - No_Certificate를 제외한 나머지 경고 코드 동일
      - 항상 심각 경고 추가

| 경고 | 설명 |
|---|---|
| Record_overflow | $2^{14}$+2048Byte가 초과하는 페이로드를 수신하거나 복호화 했을 경우 |
| Unknown_CA | CA 인증서를 찾을 수 없거나 알려진 신뢰 CA가 아닌 경우 |
| Access_denied | 정확한 인증서가 수신되었지만 접근 통제를 하면 송신자가 협상을 수행하지 않기로 결정 했을 경우 |
| Decord_Error | 한 필드가 지정 범위를 벗어났거나 메시지 길이가 부정확한 경우 |
| Insuffcient_security | 더 안전한 암호를 요구하는 경우 |
| User_canceled | 프로토콜의 실패와는 무관하게 다른 이유로 핸드셰이크를 취소하는 경우 |
| No_renegotiation | 송신자가 재협상을 할 수 없다는 것을 나타내는 경고 |
| Internal_error | 내부 오류로 인해 연결을 계속할 수 없다는 것을 나타내는 경고 |

# 전송 계층 보안(TLS)

- Transport Layer Security
  - 차이점
    - Finished 메시지 계산
      - 의사 랜덤 함수 적용

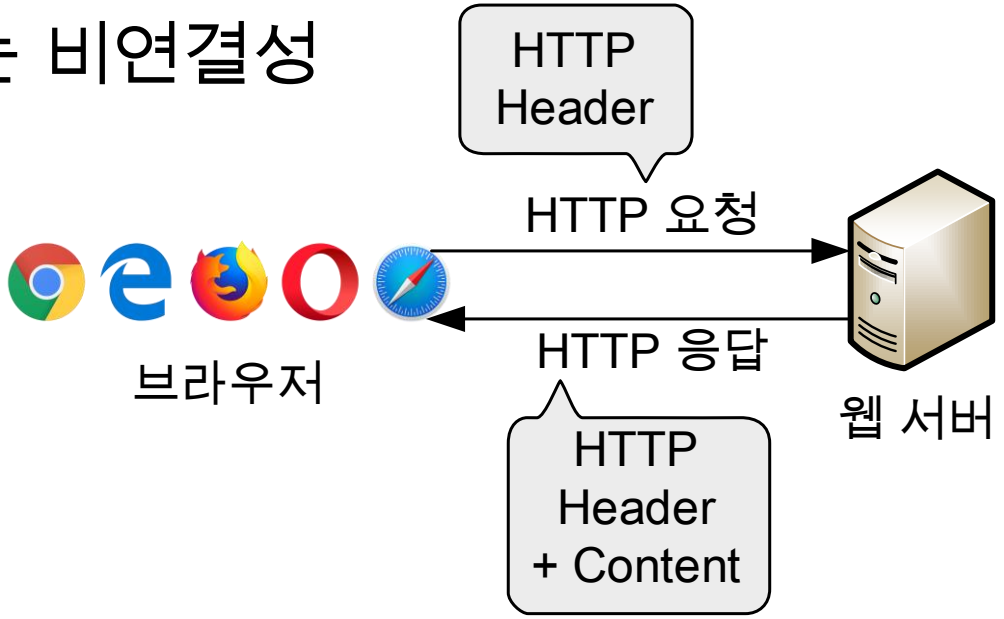


# 목 차

- 웹 보안
- 안전 소켓 계층(SSL)
- 전송 계층 보안(TLS)
- HTTPS
- SSH

# HTTPS

- HTTP(Hypertext Transfer Protocol)
  - 정의
    - 웹 브라우저와 웹 서버 간의 통신하기 위한 프로토콜
  - 특징
    - TCP/IP를 이용하는 응용 프로토콜
    - 연결 상태를 유지하지 않는 비연결성
    - 요청/응답 방식으로 동작
    - 포트 번호 80을 사용
    - URL이 http://로 시작



# HTTPS

- HTTPS(Hypertext Transfer Protocol over SSL/TLS)
  - 개요
    - 등장배경
      - 기존 HTTP는 요청 문서 URL, 브라우저 양식 등이 노출됨
      - 웹 브라우저와 웹 서버 간의 안전한 통신을 구현하기 위해 등장
  - 특징
    - HTTPS는 HTTP와 SSL/TLS의 결합
    - 현재 대부분의 웹 브라우저에 내장됨
    - SSL/TLS를 호출하기 위한 포트번호 443 사용
    - URL이 https://로 시작

# HTTPS

- Hypertext Transfer Protocol over SSL
  - 암호화 요소
    - 요청 문서의 URL
    - 문서의 내용
    - 브라우저 양식 내용
    - 브라우저가 송신한 쿠키, 서버가 송신한 쿠키
    - HTTPS 헤더 내용


# HTTPS

- Hypertext Transfer Protocol over SSL
  - 동작 과정
    1. 연결 개시
       - 클라이언트는 웹 서버에 SSL/TLS 핸드셰이크 시작
       - 이후, HTTP 요청을 보냄
         - 모든 HTTP 데이터는 TLS 응용 데이터로써, 전달됨
    2. 연결 종료
       - 연결을 종료하기 위해 Close_Notify와 HTTP에 Connection: Closed 지시자를 전송
         - 해당과정 생략 시, 비정상 종료
       - 비정상 종료
         - 경고(Alert)를 없이 종료한 경우
         - 하위 TCP 연결이 TLS 수준 연결 종료 보다 먼저 발생한 경우
         - 서버 프로그램 오류나 제 3자의 공격이 있을 경우

# 목 차

- 웹 보안
- 안전 소켓 계층(SSL)
- 전송 계층 보안(TLS)
- HTTPS
- SSH

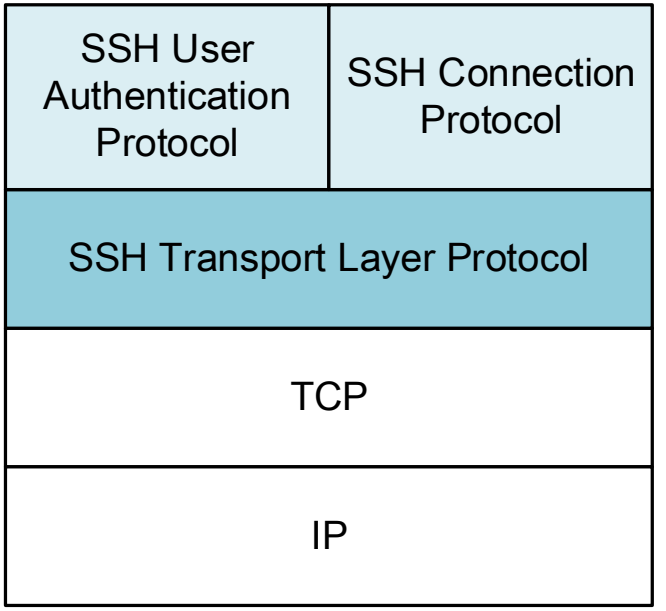
# SSH

- Secure Shell
  - 개요
    - 정의
      - 안전한 네트워크에서 다른 컴퓨터에 로그온, 원격 명령 등을 수행할 수 있도록 안전한 데이터 전송을 보장하는 네트워크 통신 프로토콜
    - 특징
      - 암호화되지 않은 기존의 Telnet 등을 대체하기 위해 설계
        - Telnet: 로컬 네트워크 연결에 쓰이는 네트워크 프로토콜
      - 클라이언트/서버 구조의 TCP 보안 채널 제공
      - 포트번호 22 사용

# SSH

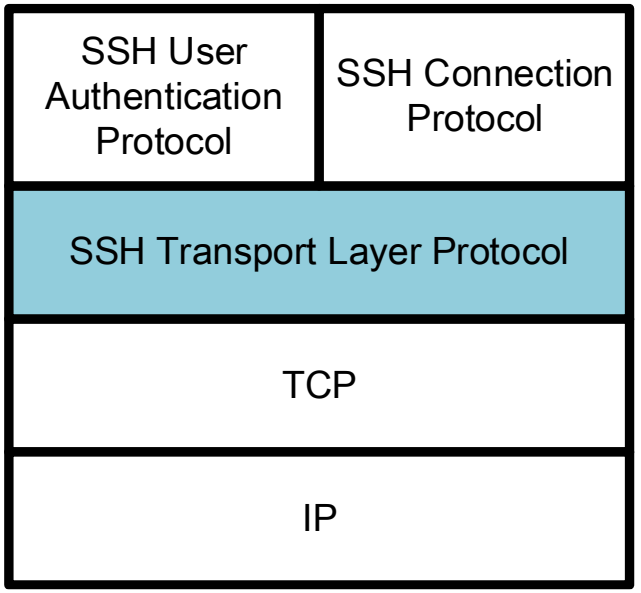
- Secure Shell
  - 개요
    - 구성

| 구성 | 설명 |
| --- | --- |
| 전송 계층 프로토콜 (Transport Layer Protocol) | • 서버의 인증, 데이터 기밀성/무결성 제공<br>• 압축 기능 제공 |
| 사용자 인증 프로토콜(User Authentication Protocol) | • 서버에게 사용자를 인증 |
| 연결 프로토콜 (Connection Protocol) | • 논리 채널 다중화<br>  • SSH 연결을 통해 1개의 암호화된 터널 상에서 여러 개의 논리적 통신 채널을 다중화 |



# SSH

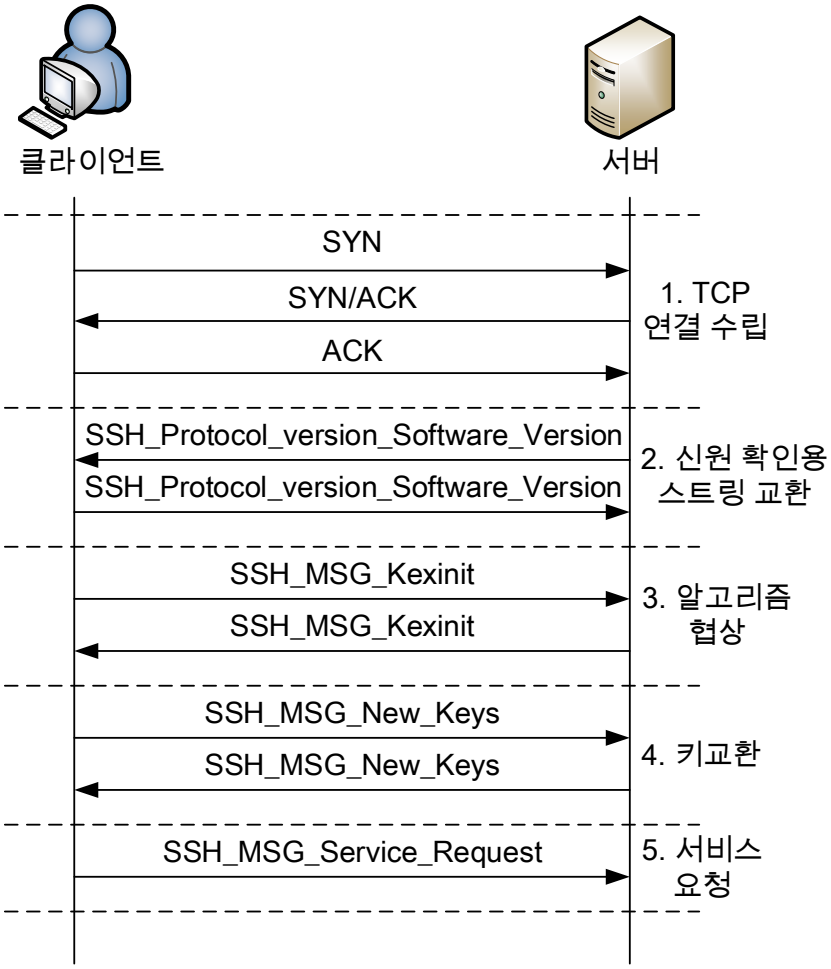
- Secure Shell
  - 전송 계층 프로토콜(Transport Layer Protocol)
    - 기능
      - 서버의 인증, 암호화를 통한 데이터 기밀성/무결성 제공
      - 암호화에 사용할 알고리즘 협상, 키 교환, 암/복호화 담당
      - 압축 기능 제공



# SSH

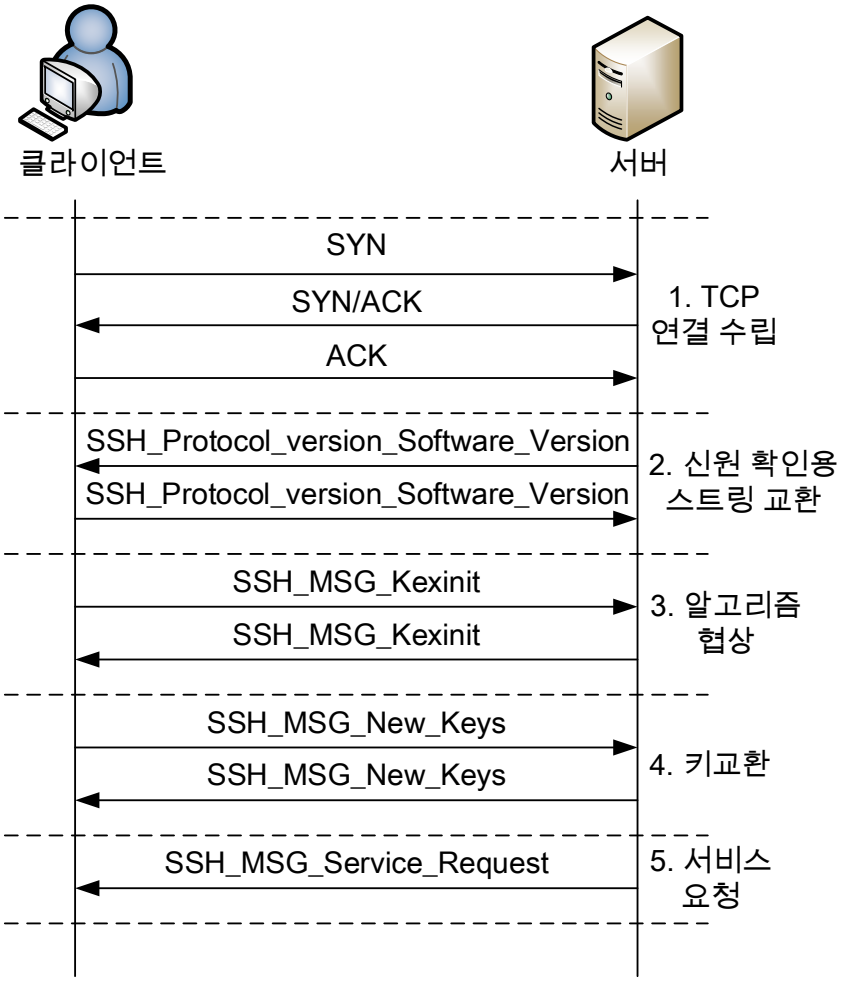
- Secure Shell
  - 전송 계층 프로토콜(Transport Layer Protocol)
    - 동작 과정
      1. TCP 연결 설립
         - TCP를 통한 연결 수립
      2. 신원 확인용 스트링 교환

| 매개 변수 | 설명 |
|---|---|
| SYN | 동기화 요청 메시지 |
| ACK | 승인 메시지 |
| SSH_MSG_Kexinit | 알고리즘을 선호도 순으로 정렬한 메시지 |
| SSH_MSG_New_Keys | 키 교환을 통해 얻은 최종 키 |



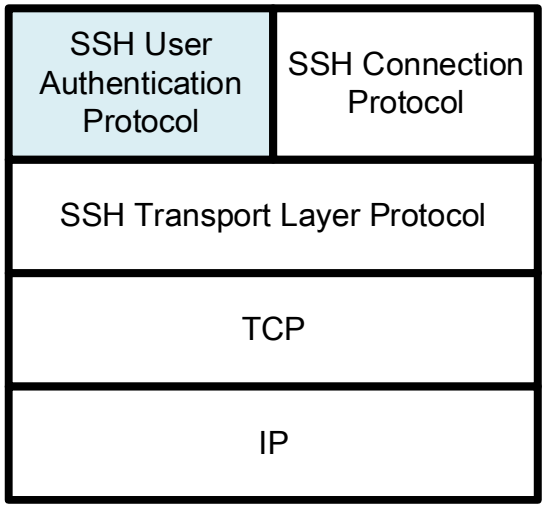
# SSH

- Secure Shell
  - 전송 계층 프로토콜(Transport Layer Protocol)
    - 동작 과정
      3. 알고리즘 협상
         - 지원 가능한 알고리즘을 선호도 순으로 정렬한 목록 전송
           - 암호 알고리즘
             - DES(3DES), AES 등
           - MAC 알고리즘
             - HMAC 사용을 위한 SHA-1, MD5 등
           - 압축 알고리즘
             - 압축 알고리즘 zlib 등
      4. 키 교환
         - e.g., RSA, Diffie-Hellman 등
      5. 서비스 요청



# SSH

- Secure Shell
  - 사용자 인증 프로토콜(User Authentication  Protocol)
    - 서버가 클라이언트에게 인증을 요청하기 위한 프로토콜
    - 메시지 유형
      - 인증 요청
        - SSH_MSG_USERAUTH_REQUEST(10 진수 50)
        - 사용자 이름, 서비스이름, 방법 이름 등
      - 인증 요청 수락
        - SSH_MSG_USERAUTH_SUCCESS(10 진수 52)
      - 인증 요청 거절
        - SSH_MSG_USERAUTH_FALLURE(10 진수 51)

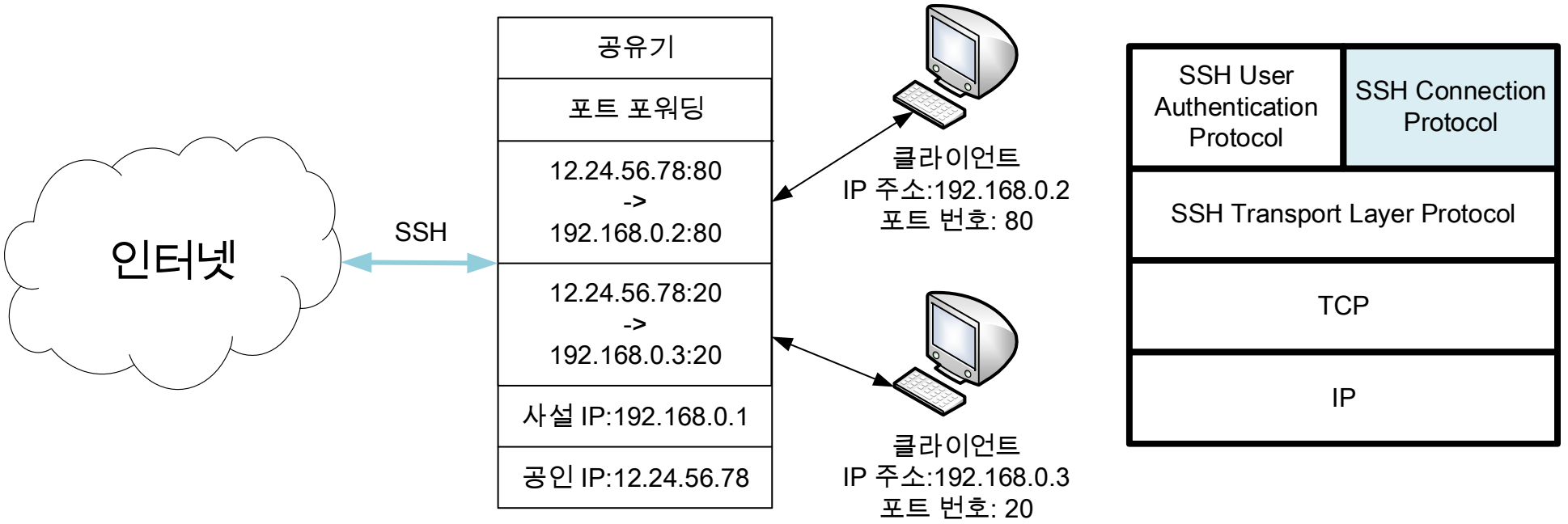


# SSH

- Secure Shell
  - 사용자 인증 프로토콜(User Authentication  Protocol)
    - 서버가 클라이언트에게 인증을 요청하기 위한 프로토콜
  - 사용자 인증 방법
    - 비밀번호 인증
      - 사용자 ID, 패스워드 인증
    - 공개키 인증
      - 공개키 인증서를 이용한 인증
    - 호스트 기반 인증
      - 호스트가 인증(공개키) 되어있고 호스트에서 클라이언트를 인증할 경우, 서버는 호스트를 믿고 클라이언트에게 서비스 제공

# SSH

- Secure Shell
  - 연결 프로토콜(Connection Protocol)
    - 안전한 인증 연결이 수립된 터널들을 각각의 다수 논리 채널들로 다중화 시키는 프로토콜
      - 포트 포워딩



# Thanks!

박 재 형 (jaehyoung@pel.sejong.ac.kr)