

Network Security

2025.08

자동차융합대학



GENERAL MOTORS
GM TECHNICAL CENTER KOREA



국민대학교
KOOKMIN UNIVERSITY

CONTENTS

01 네트워크 보안 개요

02 TLS Protocol

01

네트워크 보안 개요

■ 개념

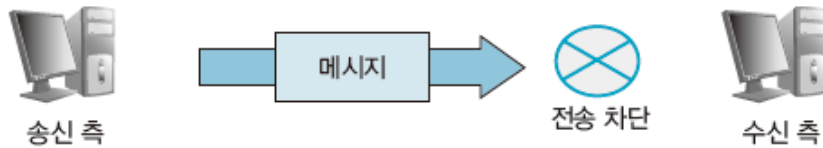
- 전통적인 보안은 크게 컴퓨터 보안과 네트워크 보안으로 구분할 수 있음. 쉽게 말해, 컴퓨터 보안은 컴퓨터 자체의 데이터를 보호하는 것이고, 네트워크 보안은 컴퓨터 간에 데이터를 안전하게 전송하는 것

■ 네트워크 보안 요구사항

- 기밀성 (Confidentiality)
 - 자산이 인가(Authorization)된 당사자에 의해서만 접근하는 것을 보장하는 것
 - 위협 요소: 도청
- 무결성 (Integrity)
 - 자산이 인가된 당사자에 의해서, 인가된 방법으로만 변경 가능 것과 자산의 완전성과 정확성을 보장하는 것
 - 위협요소: 변조
- 가용성 (Availability)
 - 자산이 적절한 시간에 인가된 당사자에게 접근 가능해야 하는 것
 - 위협요소: DoS, DDoS

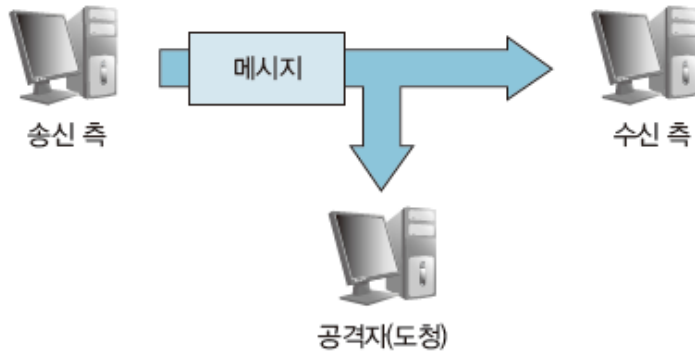
■ 전송 차단

- 송신 측에서 수신 측에 메시지를 전송할 때 제 3자가 수신 측과 연결할 수 없도록 하는 위협
- DoS/DDoS 공격에 활용



■ 가로채기 (Interruption)

- 송신 측과 수신 측이 데이터를 주고받는 사이 제3자(공격자)가 도청하는 위협, 이 경우를 Interruption이라고 하며, 송신 측과 수신 측의 중요한 정보가 유출되는 심각한 문제가 발생하며, Snipping 공격에 활용됨.



■ 변조 (Modification)

- 송신 측에서 수신 측으로 전송할 데이터를 제3자(공격자)가 가로채서 데이터의 일부 또는 전부를 변경하여 잘못된 데이터를 수신 측에 전송하는 위협으로 Spoofing 공격에 활용
- 이 경우를 Modification이라고 하며, 수신 측에서는 송신 측에서 잘못된 데이터를 전송한 것으로 오인할 수 있음.

■ 위조 (Fabrication)

- 제3자(공격자)가 마치 송신 측이 메시지를 전송한 것처럼 위조하여 수신 측에 전송하는 위협으로 Spoofing 공격에 활용
- 이때는 modification과 달리 아예 송신 측에서 만들지 않은 메시지를 수신 측으로 전송하는 문제가 발생함.

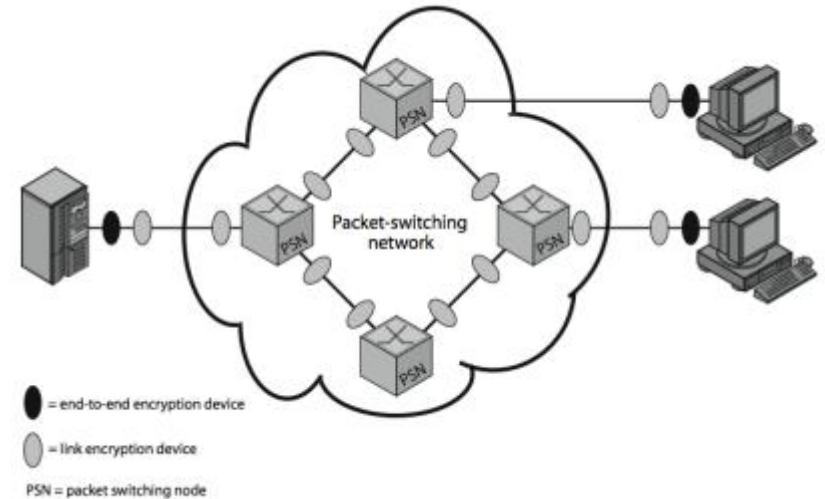


■ Link Encryption

- 데이터 링크 또는 물리적 계층에서 모든 정보를 암호화
- 네트워크 장비 중 라우터에서는 패킷의 목적지를 알아야 하기 때문에 각 홉마다 그 정보들을 복호화
- 다시, 하드웨어 암호화 장치들이 모든 데이터를 암호화

■ End-to-End Encryption

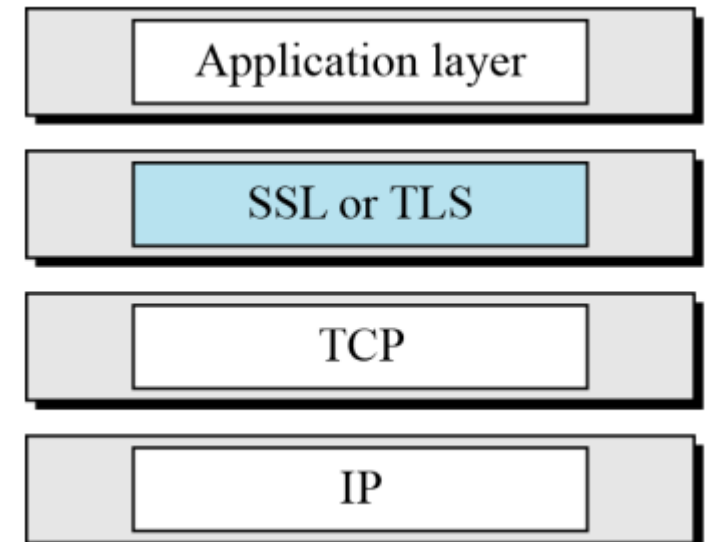
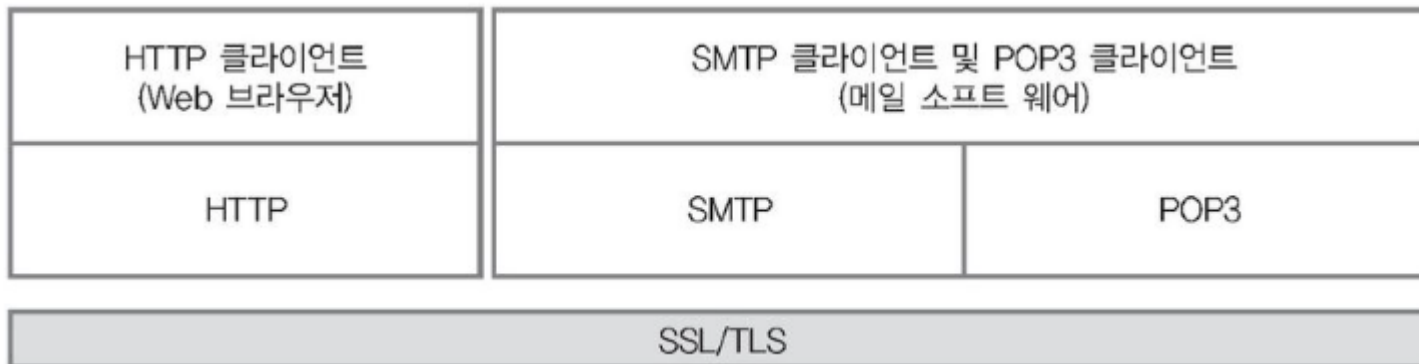
- 헤더와 트레일러 부분이 암호화되지 않아서 암호화/복호화를 다시 적용할 필요 없음
 - 패킷의 헤더 부분에 있는 라우팅 정보가 암호화되지 않으면, 라우터가 패킷의 헤더에 있는 라우팅 정보를 확인하고 다음 네트워크 장비로 데이터를 보낼 수 있기 때문에 링크 암호화처럼 다시 암호화하고 하는 등의 작업이 필요 없음
- Application Layer에서 암호화 수행



| 구분 | Link Encryption | End-to-End Encryption |
|----|---|--|
| 특징 | <ul style="list-style-type: none">모든 데이터 암호화 (트래픽 분석 어려움)ISP, 통신업자가 암호화하기 때문에 end user가 알고리즘을 통제할 수 없음 | <ul style="list-style-type: none">라우팅 정보가 포함된 헤더 부분은 암호화하지 않음사용자가 암호화하기 때문에 end user가 알고리즘을 통제할 수 있음 |
| 장점 | <ul style="list-style-type: none">라우팅 정보까지 암호화해서 트래픽 분석을 어렵게 함User Transparent하게 암호화되어 운영이 간단함즉, user 관점에서 암호화의 과정을 인식하지 않아도 되기 때문에 운영이 간단함 | <ul style="list-style-type: none">Application Layer에서 암호화가 이루어지기 때문에, 사용자 인증 등 높은 수준의 다양한 보안 서비스 제공이 가능중간에 데이터를 복호화 할 필요가 없어서 중간 노드에서도 데이터가 암호문으로 존재함 |
| 단점 | <ul style="list-style-type: none">중간 노드에서 데이터를 복호화하는 과정에서 데이터가 평문으로 노출됨모든 노드가 암호화 장비를 갖추어야 하기 때문에 네트워크 구축 비용 증가 | <ul style="list-style-type: none">라우팅 정보를 암호화하지 않아서 트래픽 분석에 취약 |

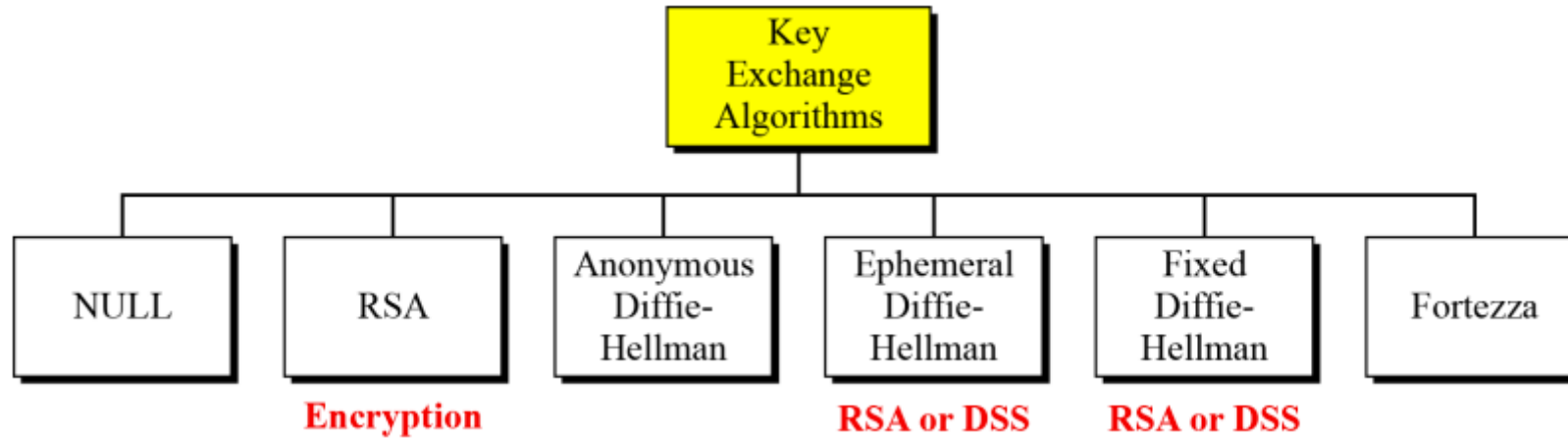
■ TLS(SSL)

- Transportation Layer Security (Secure Socket Layer)는 컴퓨터 네트워크에 통신 보안을 제공하기 위해 설계된 암호 규약으로 도청, 간섭, 위조를 방지하기 위해 설계되었다.
- SSL 3.0을 기초로 해서 IETF가 만든 프로토콜로 SSL 3.0이 TLS 1.0이다. (현재는 TLS 1.3)
- TLS의 3단계 기본 절차
 - [Handshake Protocol] 지원 가능한 알고리즘 서로 교환
 - [Handshake Protocol] 키 교환, 인증
 - [Record Protocol] 대칭키 암호로 암호화하고 메시지 인증



02

TLS Protocol

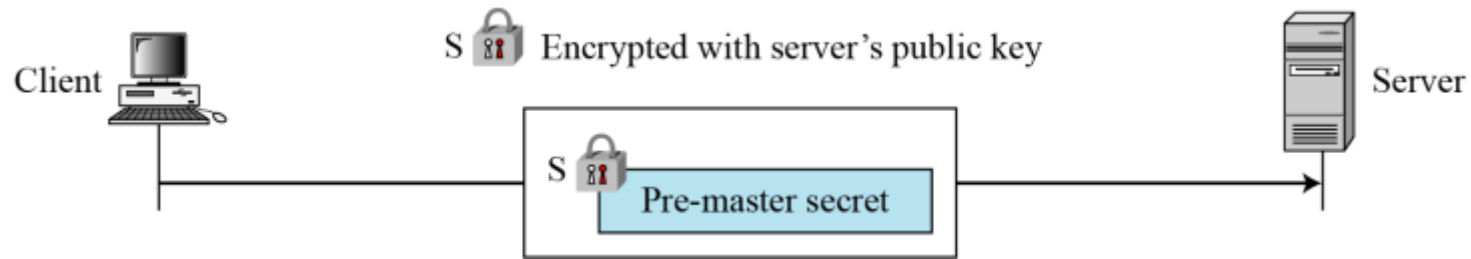


■ NULL

- There is no key exchange in this method. No pre-master secret is established between client and server.
- To generate a session key, need the pre-master secret

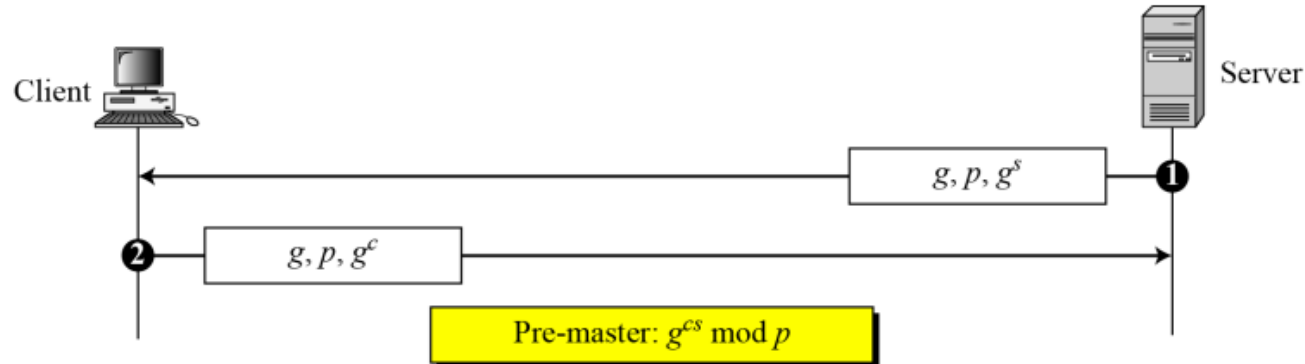
■ RSA key exchange; server public key

- Pre-master secret: 48-byte random number created by the client



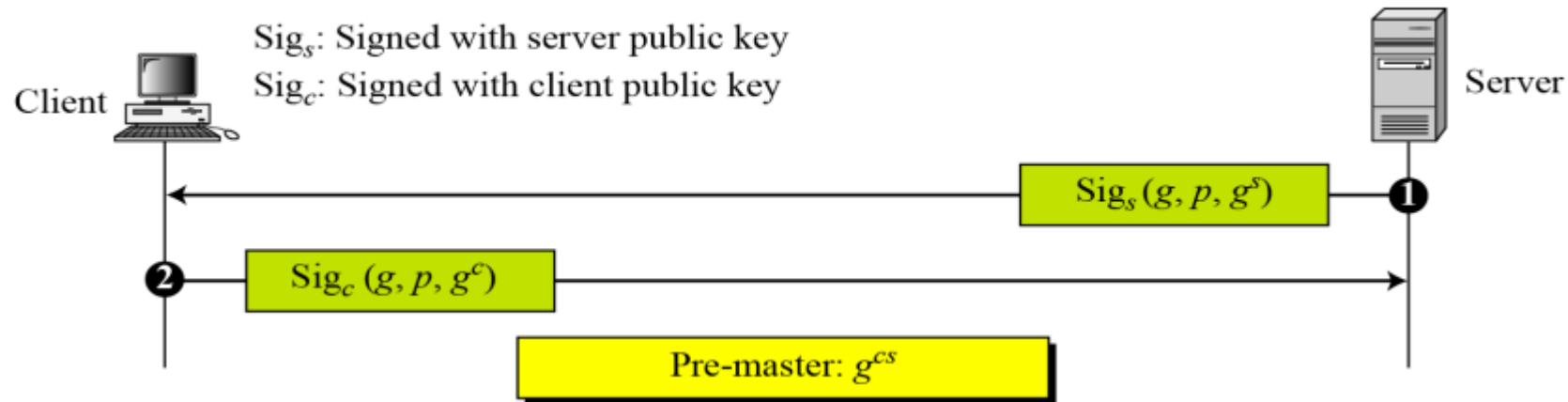
■ Anonymous Diffie-Hellman key exchange

✗ Insecure one



■ Ephemeral Diffie-Hellman key exchange

✗ Need certificate



- The combination of key exchange, hash, and encryption algorithms defines a cipher suite for each SSL session. See Table 17.1

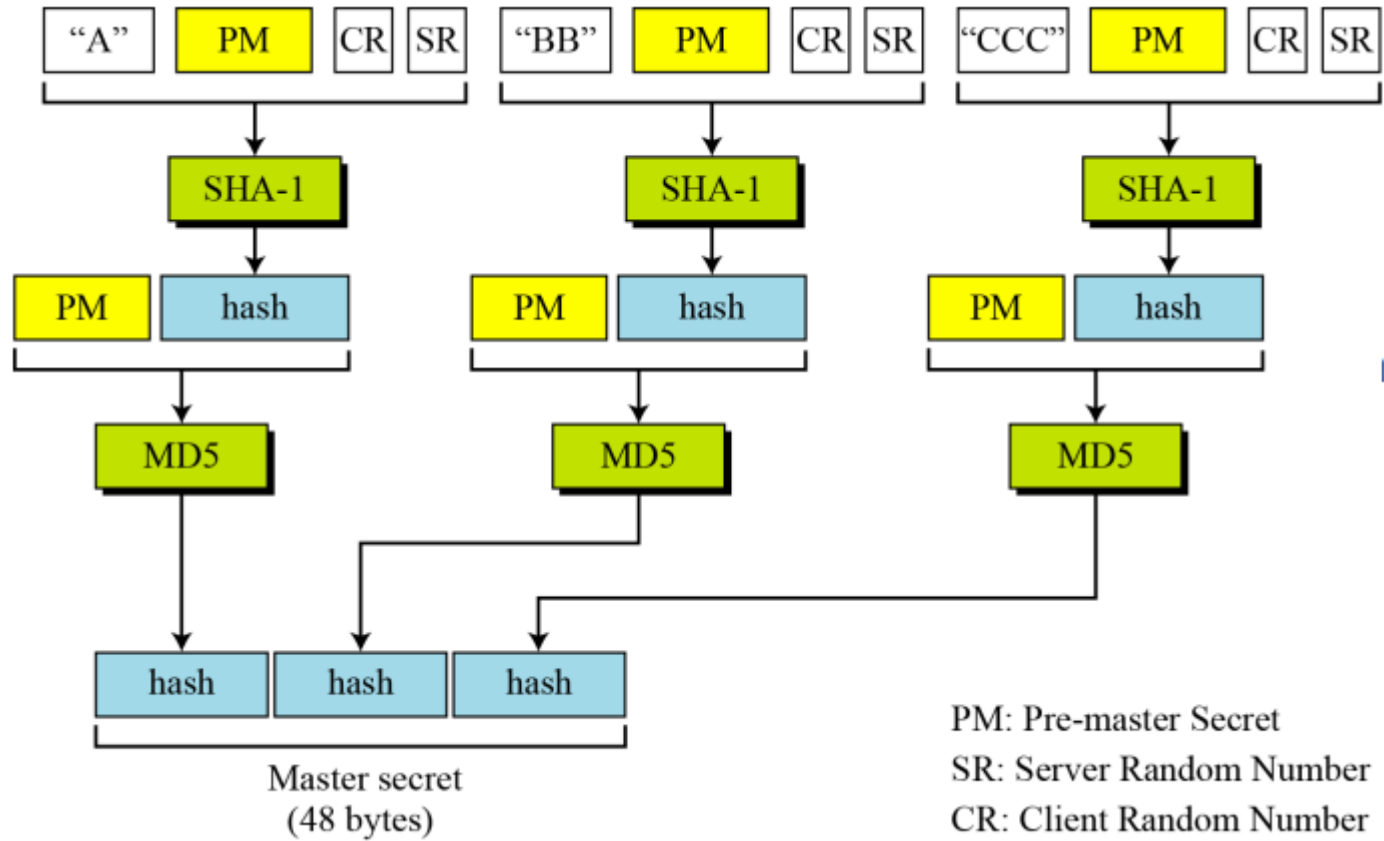
SSL_DHE_RSA_WITH_DES_CBC_SHA

✕ Ephemeral DH with RSA sig. cert., DES_CBC, & SHA

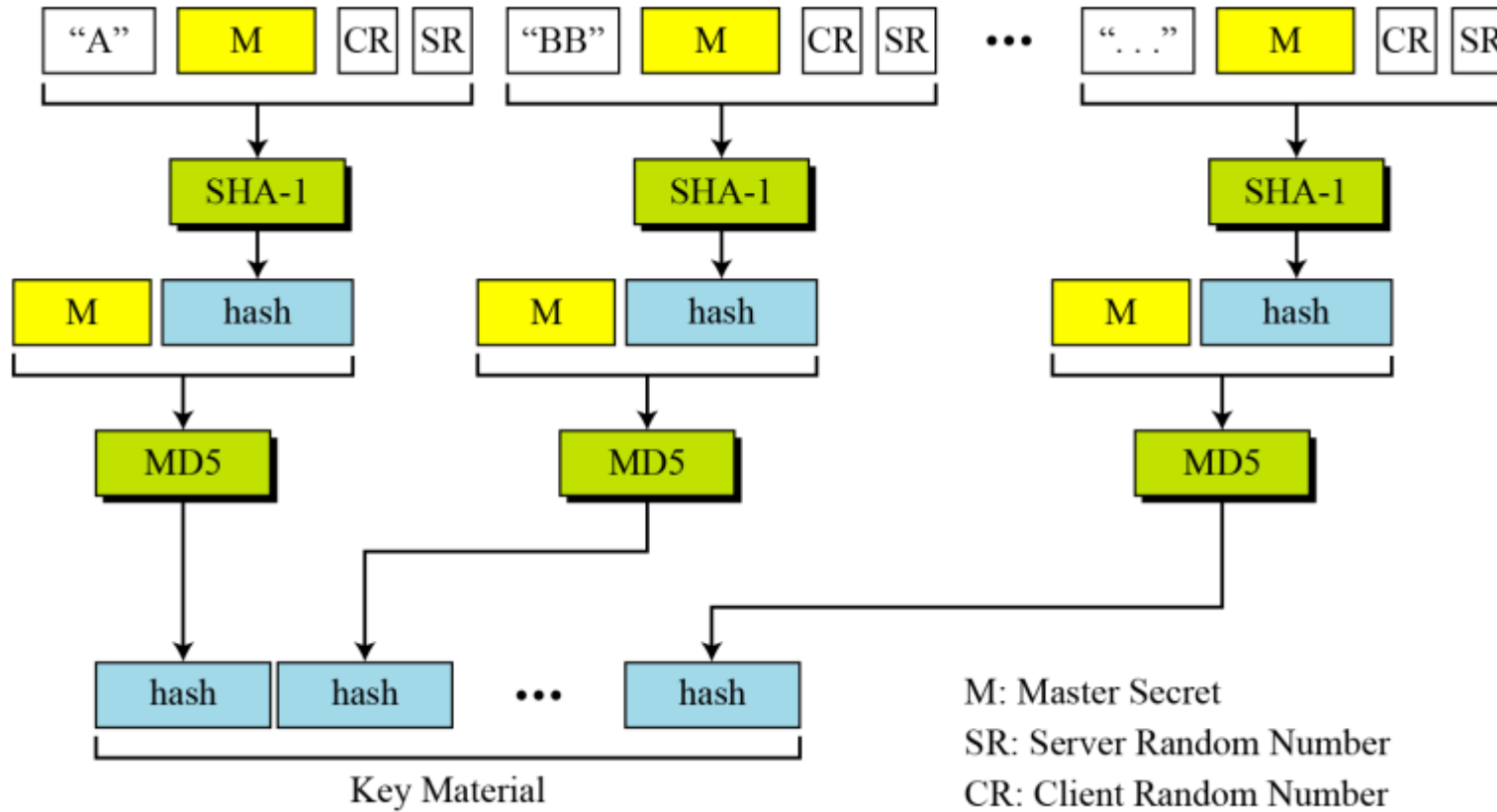
■ Compression Algorithms

- Compression is optional in SSLv3. No specific compression algorithm is defined for SSLv3. Therefore, the default compression method is NULL.

■ Calculation of 48-byte master secret from pre-master secret



■ Calculation of key material from master secret

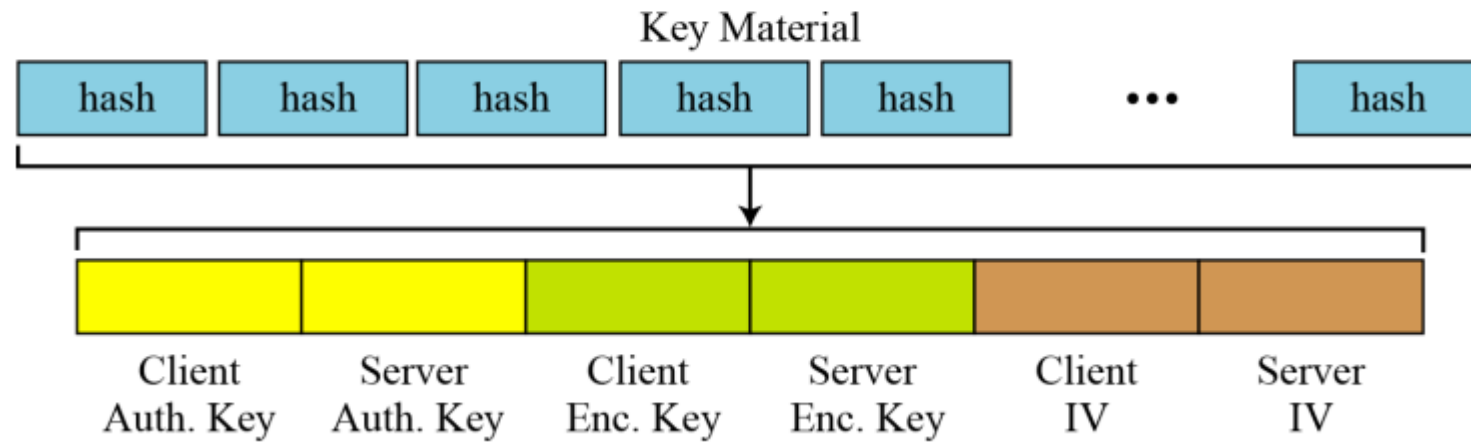


■ Extractions of cryptographic secrets from key material

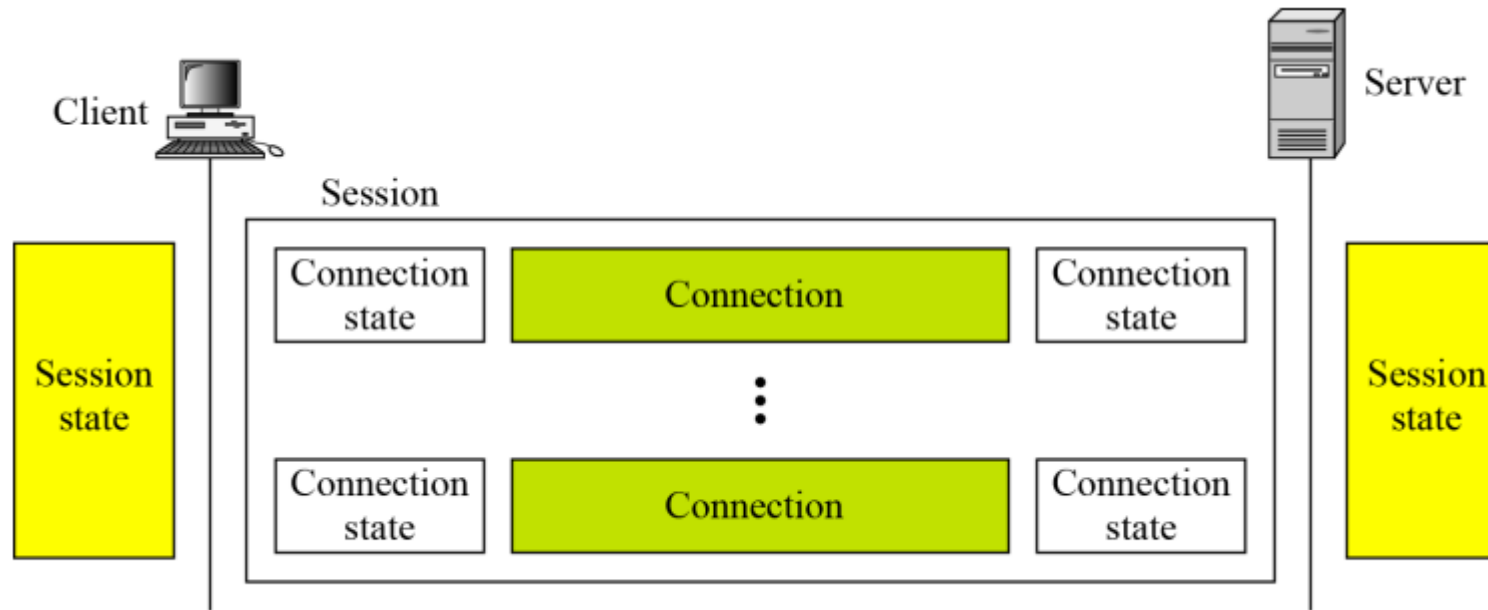
Auth. Key: Authentication Key

Enc. Key: Encryption Key

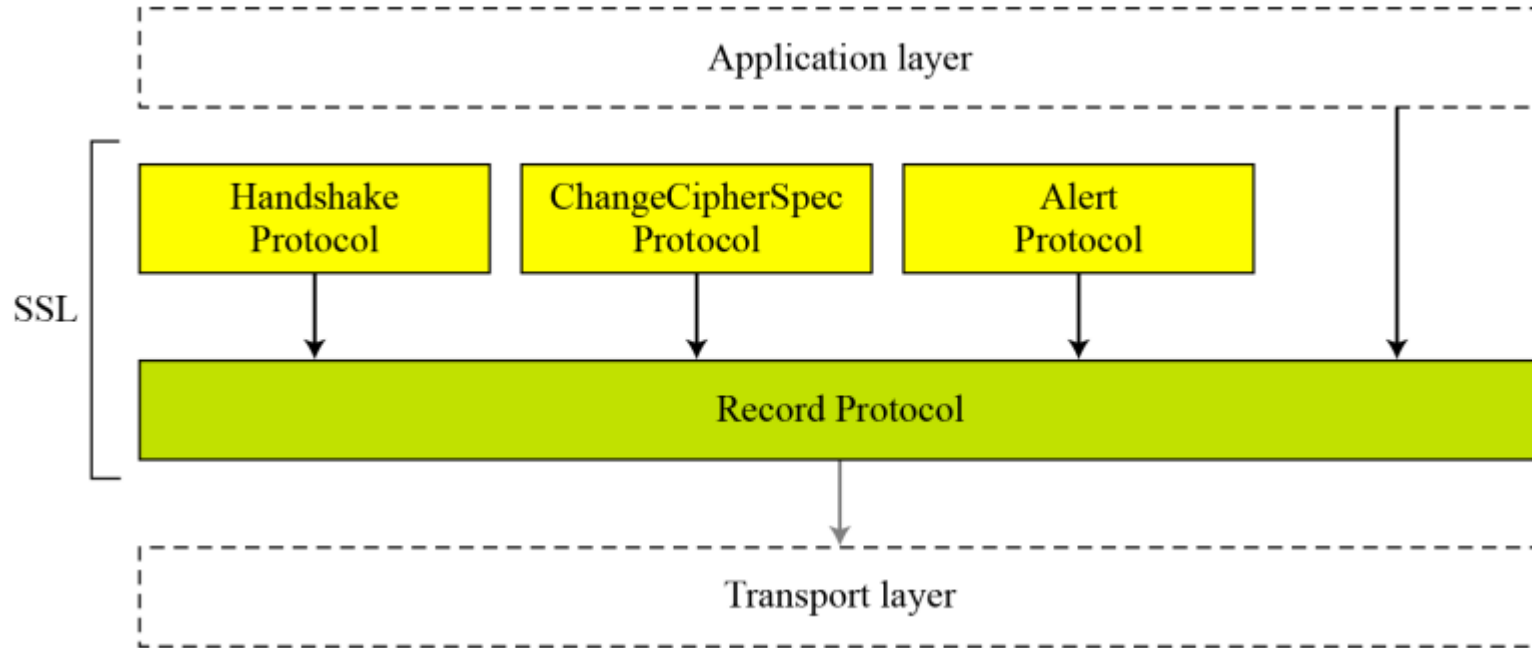
IV: Initialization Vector



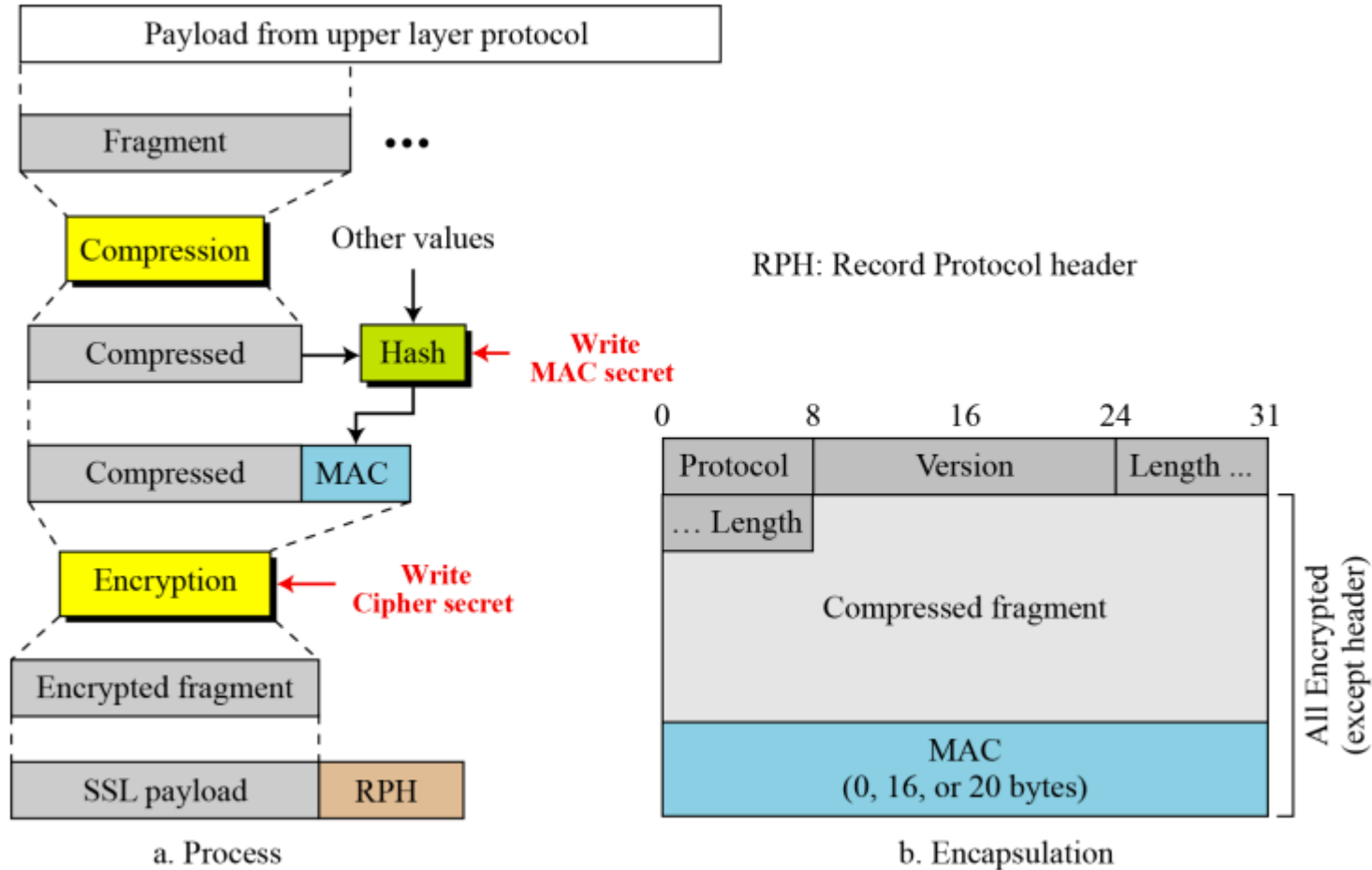
- In a session, one party has the role of a client and the other the role of a server; in a connection, both parties have equal roles, they are peers



■ Four TLS protocols



■ Carries messages from the upper layer



```
■ struct {  
    ContentType type;  
    ProtocolVersion version;  
    uint16 length;  
    opaque fragment[TLSPplaintext.length];  
} TLSPplaintext;
```

type

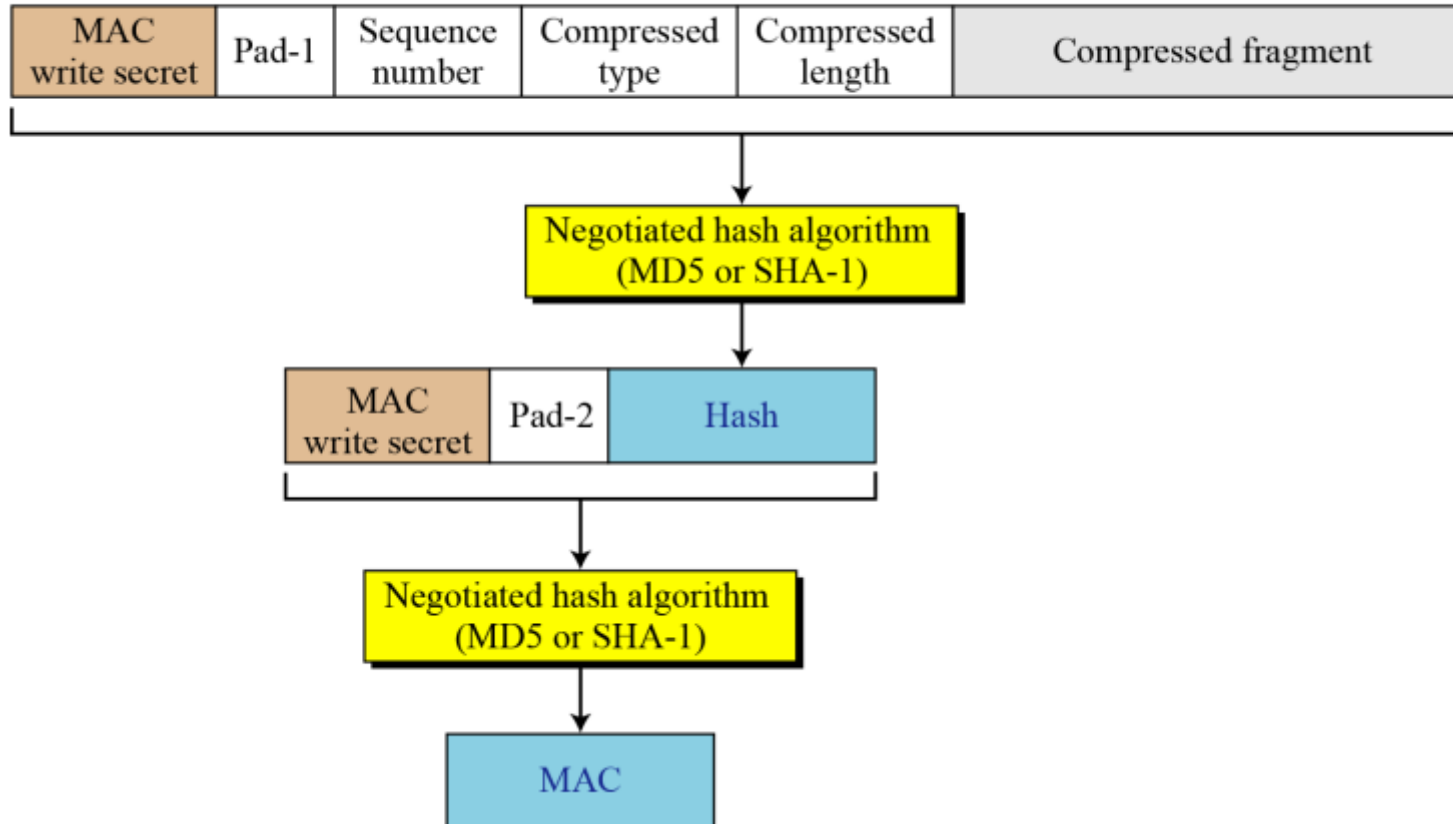
The higher level protocol used to process the enclosed fragment.

- 20 : ChangeCipherSpec
- 21 : Alert protocol
- 22 : Handshake protocol
- 23 : Application protocol data

■ Calculation of MAC

Pad-1: Byte 0x36 (00110110) repeated 48 times for MD5 and 40 times for SHA-1

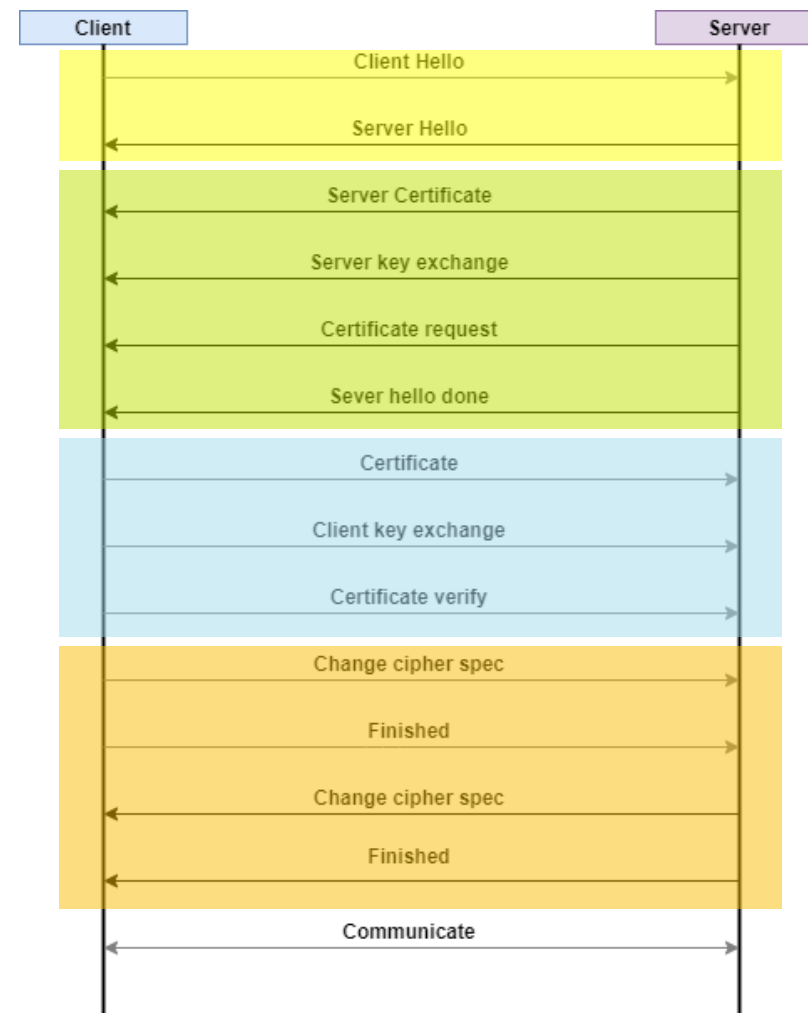
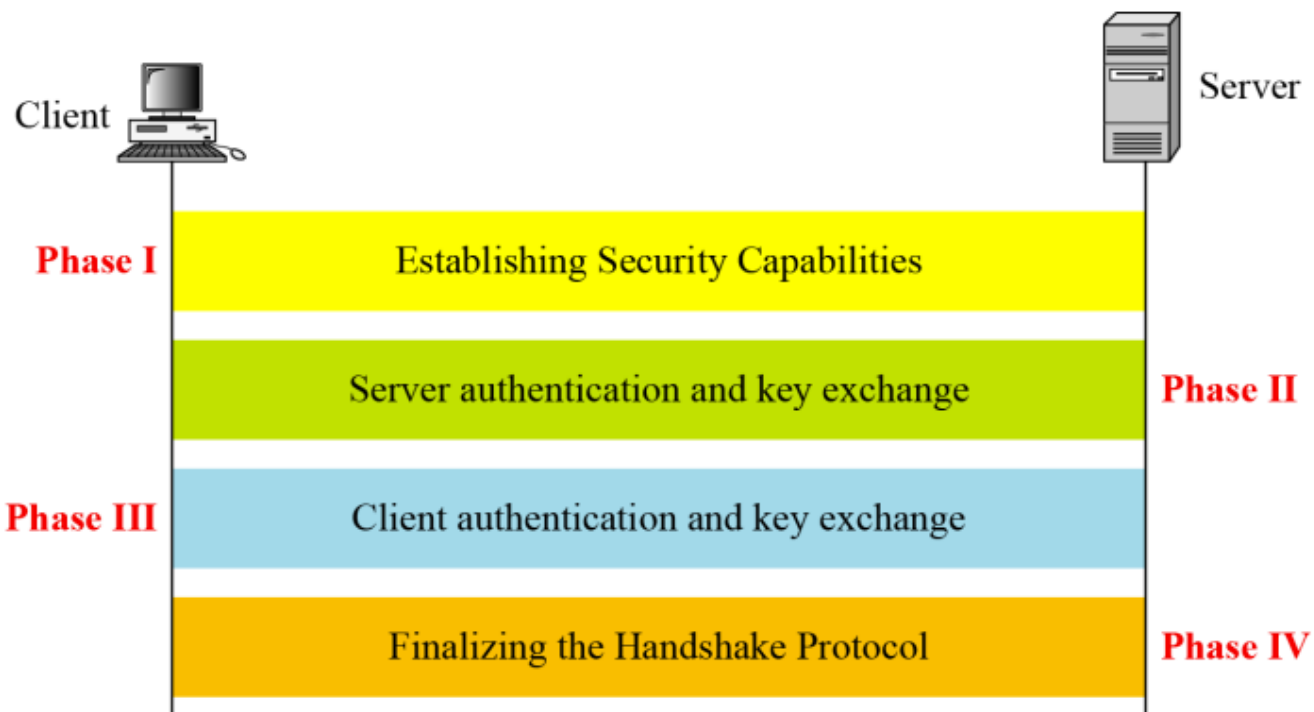
Pad-2: Byte 0x5C (01011100) repeated 48 times for MD5 and 40 times for SHA-1



■ Report errors and abnormal conditions

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

■ The negotiation of the cipher suite and the generation of cryptographic secrets



■ Client Hello (client → server)

- 클라이언트가 서버에게 Client Hello라는 메시지를 보낸다.
- 보내는 정보
 - 사용할 수 있는 버전 번호
 - 현재 시각
 - client random
 - Session ID (처음에는 빈 값, 이미 생성된 session이 있다면 해당 session ID)
 - 사용할 수 있는 Cipher Suit 목록
 - 사용할 수 있는 압축 방법 목록

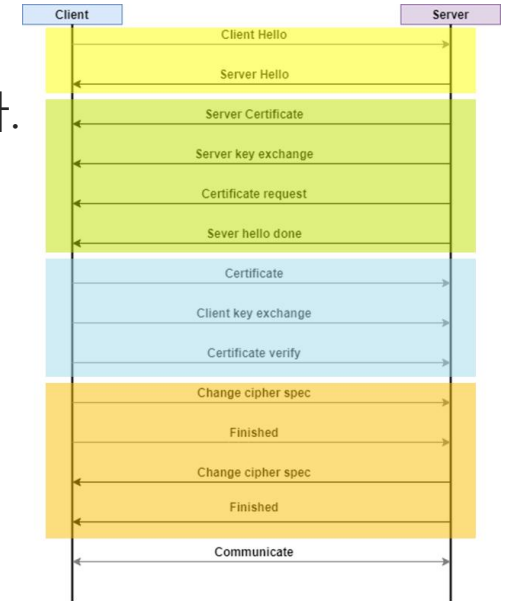
Cipher Suites (16 suites)

Cipher Suite: Reserved (GREASE) (0x0a0a)
Cipher Suite: TLS_AES_128_GCM_SHA256 (0x1301)
Cipher Suite: TLS_AES_256_GCM_SHA384 (0x1302)
Cipher Suite: TLS_CHACHA20_POLY1305_SHA256 (0x1303)
Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 (0xc02b)
Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f)
Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 (0xc02c)
Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030)
Cipher Suite: TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256 (0xcca9)
Cipher Suite: TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 (0xcca8)
Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0xc013)
Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014)
Cipher Suite: TLS_RSA_WITH_AES_128_GCM_SHA256 (0x009c)
Cipher Suite: TLS_RSA_WITH_AES_256_GCM_SHA384 (0x009d)
Cipher Suite: TLS_RSA_WITH_AES_128_CBC_SHA (0x002f)
Cipher Suite: TLS_RSA_WITH_AES_256_CBC_SHA (0x0035)



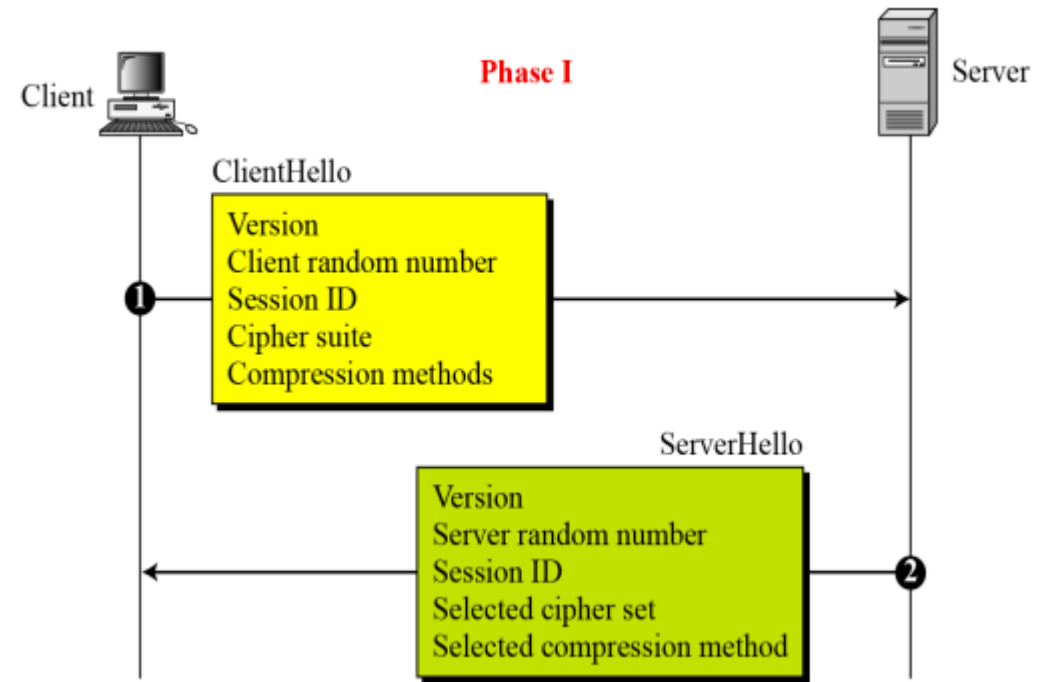
■ Server Hello (server → client)

- 클라이언트로부터 받은 Client Hello에 대해, 서버는 Server Hello라고 하는 메시지를 보낸다.
- 보내는 정보
 - 사용하는 버전 번호
 - 현재 시각
 - Server random
 - Session ID
 - 사용하는 Cipher Suit (결정됨)
 - 사용하는 압축 방법 (결정됨)



■ After Phase I, the client and server know the following:

- The version of SSL
- The algorithms for key exchange, message authentication, and encryption
- The compression method
- The two random numbers for key generation



■ Server Certificate (server → client)

- 서버가 클라이언트에 서버의 인증서 목록을 보내고, 클라이언트는 인증서를 보고 서버가 믿을 만한 혹은 신뢰할 만한 서버인지 확인한다.

■ Server Key Exchange (server → client)

- 키 교환에 필요한 정보를 제공한다. 이때 부터는 앞서 결정된 Cipher Suite에 따라 정보가 다르게 전송될 수 있다.

■ Certificate Request (server → client)

- 서버 역시 클라이언트를 인증해야 할 때 인증서를 요구할 수도 있고, 요청하지 않을 수도 있다. 인증서를 요구할 때에는 서버가 이해할 수 있는 인증서 타입 목록과 인증기관 이름 목록을 보낸다.

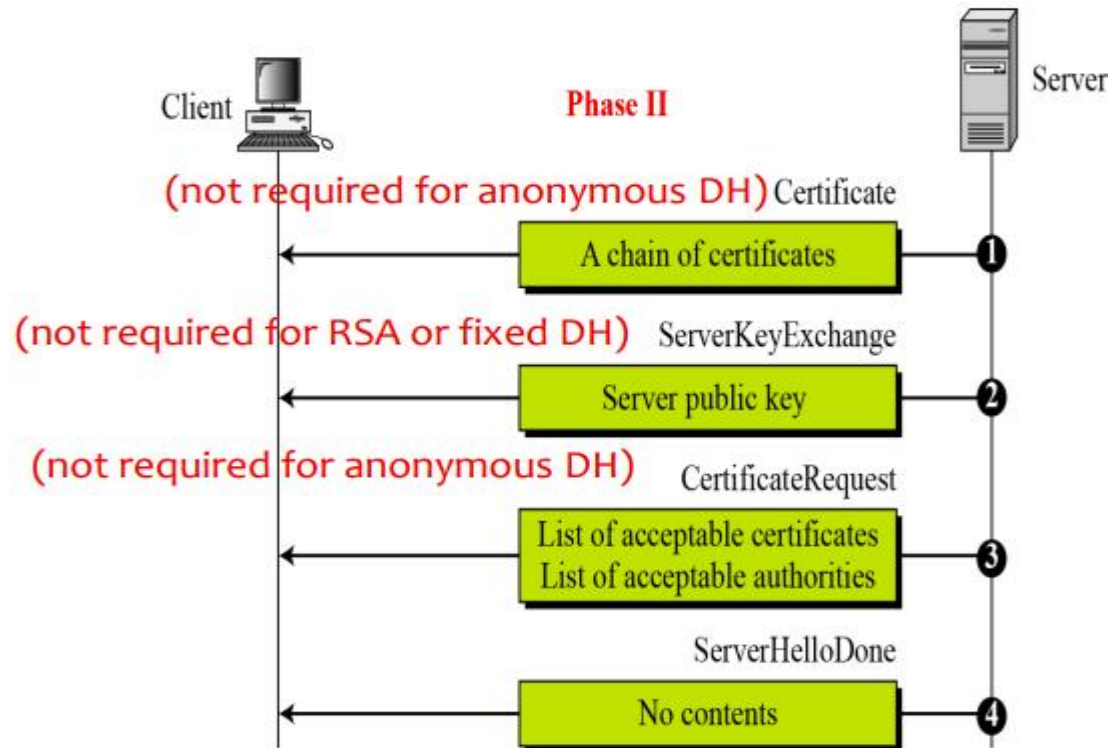
■ Server Hello Done (server → client)

- Server Hello가 끝났음을 클라이언트에게 알린다.

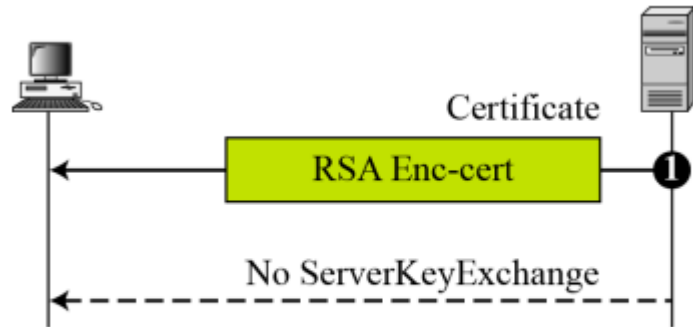


■ After Phase II,

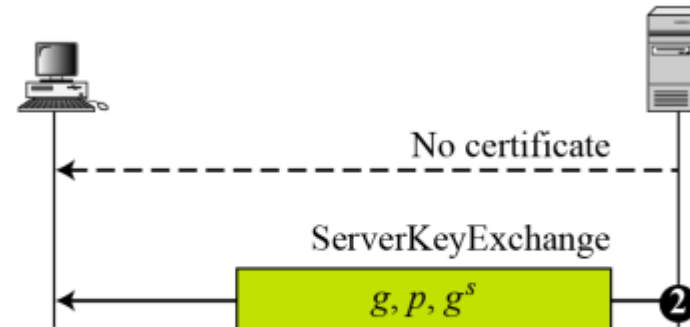
- The server is authenticated to the client.
- The client knows the public key of the server if required.



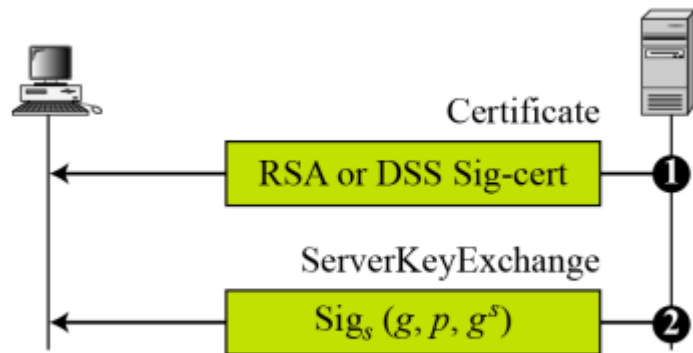
■ Four cases in Phase II



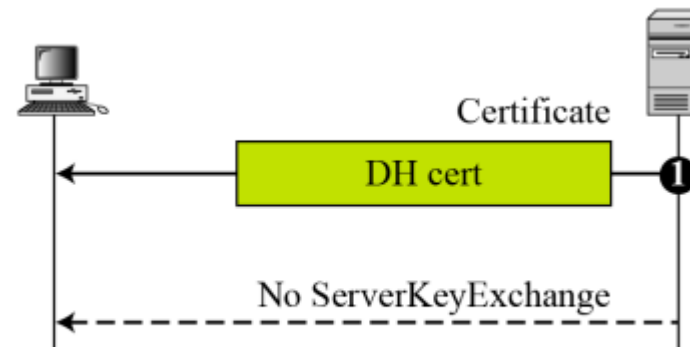
a. RSA



b. Anonymous DH



c. Ephemeral DH



d. Fixed DH

■ Certificate (client → server)

- 방금 전 서버가 요청했던 인증서를 보낼 수 있다. 요청하지 않았다면, 필요 없는 과정이다.

■ Client Key Exchange (client → server)


- 키교환에 필요한 정보를 서버에 제공한다. 이 정보를 Pre-master secret이라고 한다.
- Pre-master secret이 대칭키에 사용되는 것으로 절대 노출이 되면 안된다. 따라서 서버에서 받은 random과 pre-master secret을 조합하고, 서버로 부터 받은 인증서의 공개키를 이용하여 암호화하여 서버에 전송한다.
- 서버는 암호화된 pre-master secret을 서버의 개인키로 복호화 하고, 일련의 과정을 거쳐서 server와 client는 각각 동일한 master key를 생성한다. (Key 생성 알고리즘에 따라...)

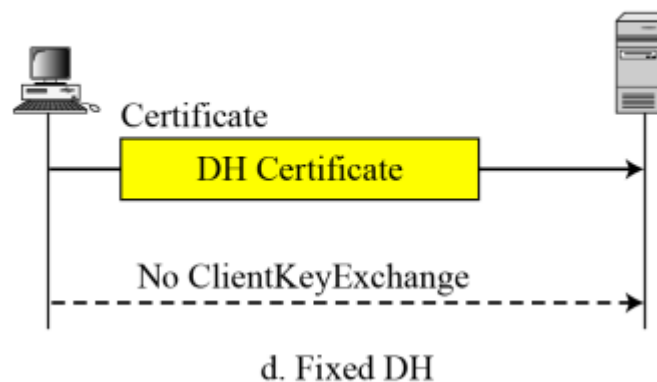
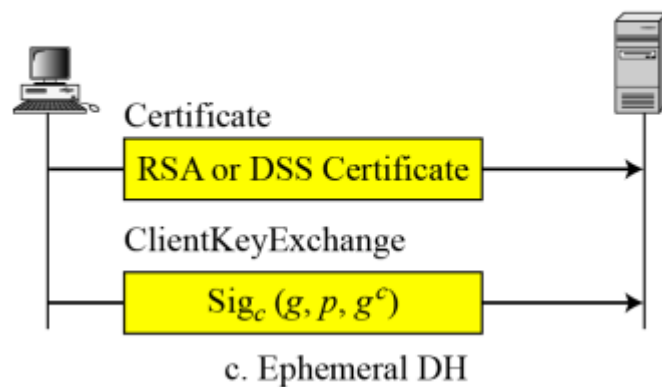
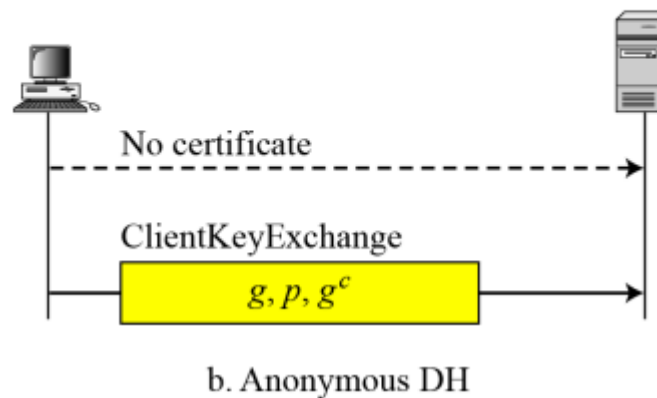
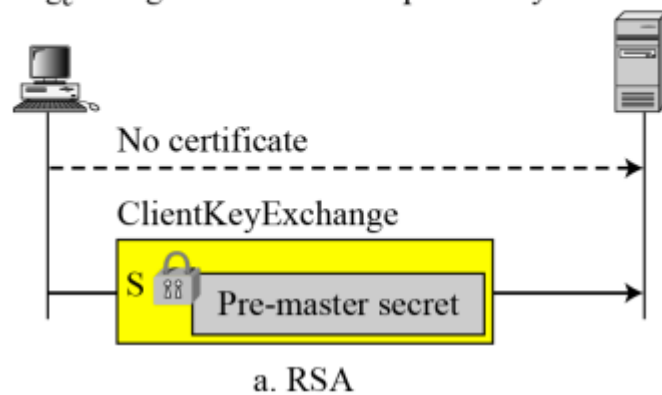
■ Certificate Verify (client → server)

- 클라이언트에 대한 Certificate Request를 받았다면, 보낸 인증서에 대한 개인키를 가지고 있다는 것을 증명한다. 즉, 앞서 생성한 master secret 등을 이용해서 서명 값을 전달한다.



■ Four cases in Phase III

S  encrypted with server's public key
 Sig_c : Signed with client's public key



■ Change Cipher Spec. (client → server)

- 만약, 협상된 security parameter를 적용하거나 변경될 때 서버에게 알린다.

■ Finished (client → server)

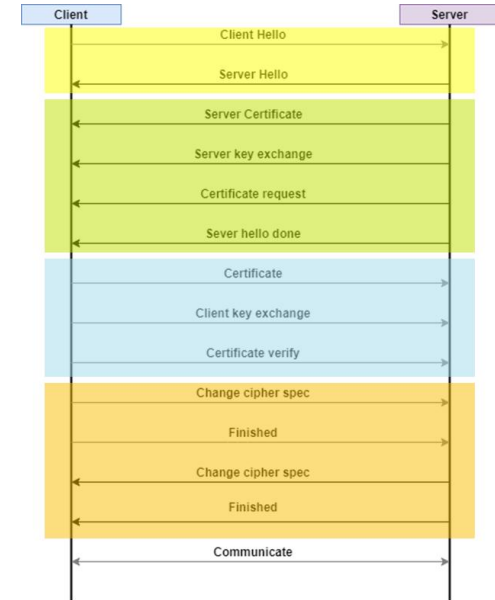
- 클라이언트가 Handshake Protocol에서 종료되었음을 서버에 알린다.

■ Change Cipher Spec. (server → client)

- 이번에는 서버가 클라이언트에게 security parameter가 변경되었음을 알린다.

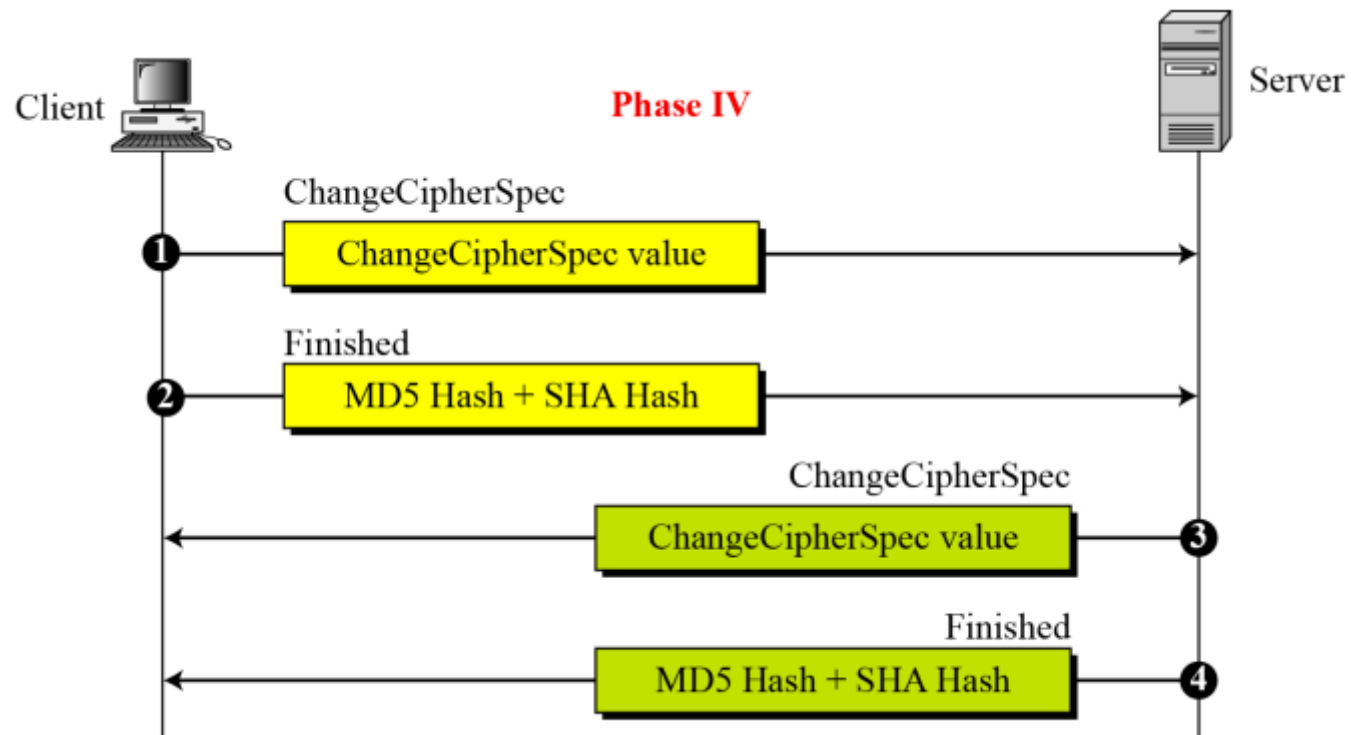
■ Finished (server → client)

- 서버가 Handshake Protocol에서 종료되었음을 클라이언트에 알린다.



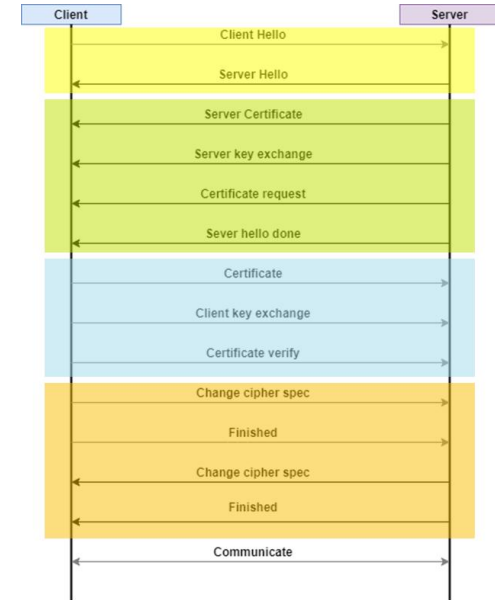
■ After Phase IV,

- The client and server are ready to exchange data.



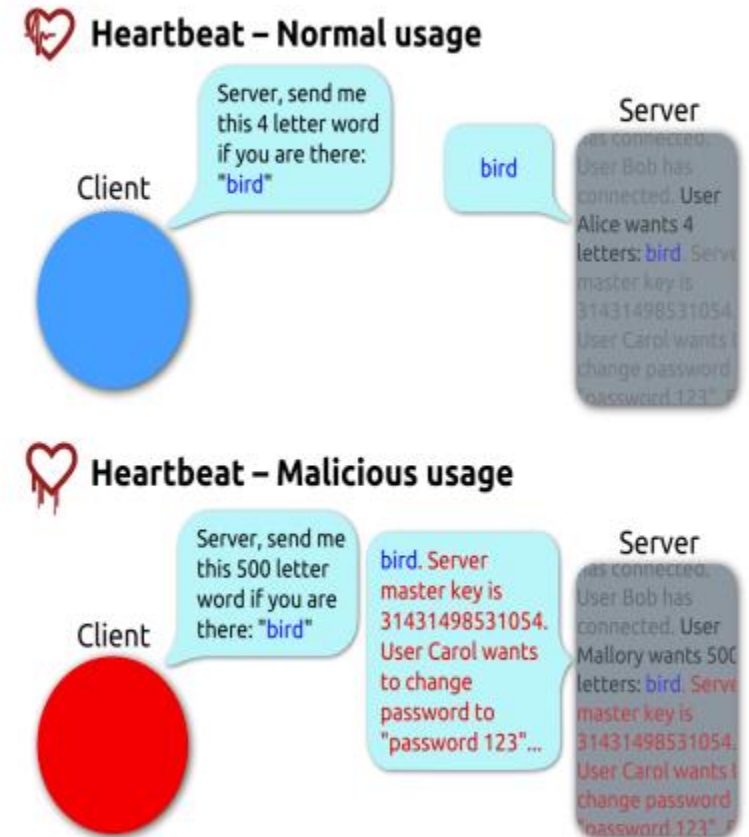
■ Handshake Protocol의 목적

- 클라이언트는 서버의 공개키를 확인할 수 있고, 서버를 인증할 수 있다.
- 서버는 클라이언트의 공개키를 확인 수 있고, 클라이언트를 인증할 수 있다.
- 클라이언트와 서버는 암호 통신용의 공유 키를 얻을 수 있다. (from master secret)
- 클라이언트와 서버는 메시지 인증 코드용 공유 키를 얻을 수 있다. (from master secret)



■ Heartbleed

- Heartbeat
 - Allowing a computer at one end to send a Heartbeat Request message, consisting of a payload, along with the payload's length as a 16-bit integer. The receiving computer then must send exactly the same payload back to the sender
- Should provide bounding check!



Thank You



- Lab: <https://mose.kookmin.ac.kr>
- Email: sh.jeon@kookmin.ac.kr