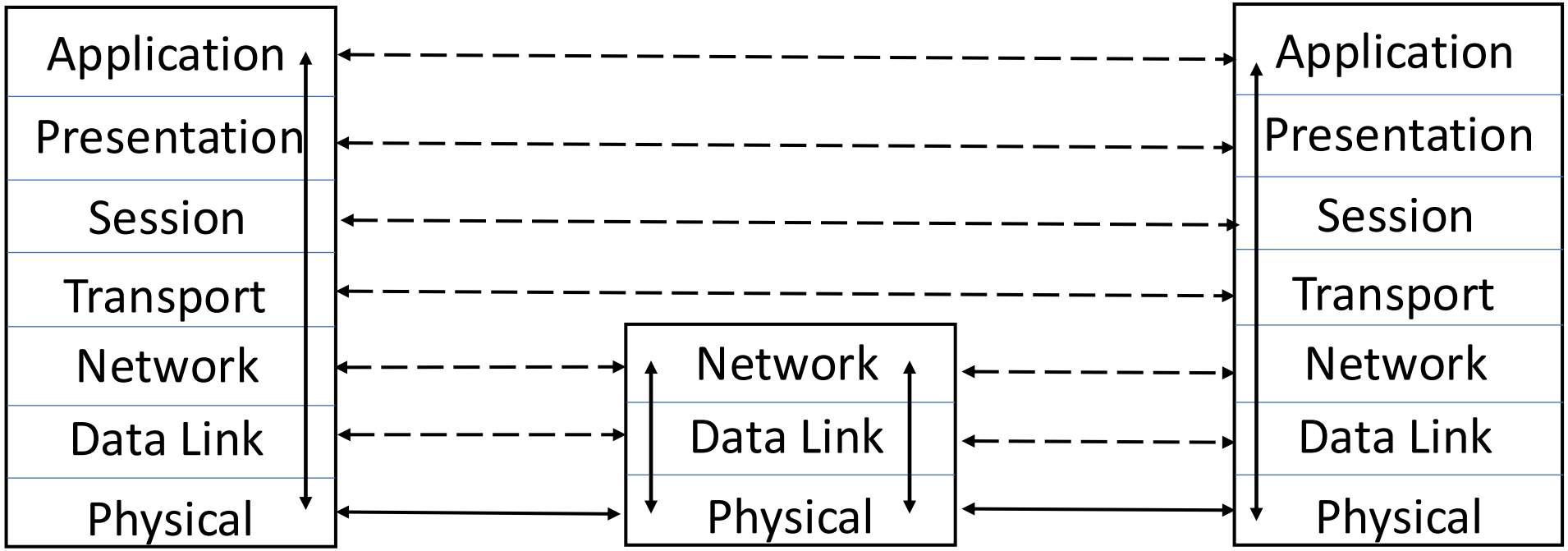
Lecture 15, May 1, 2026

What These Mean

- Use of strong cryptosystems, well-chosen (or random) keys not enough to be secure
- Other factors:
  - Protocols directing use of cryptosystems
  - Ancillary information added by protocols
  - Implementation (not discussed here)
  - Maintenance and operation (not discussed here)

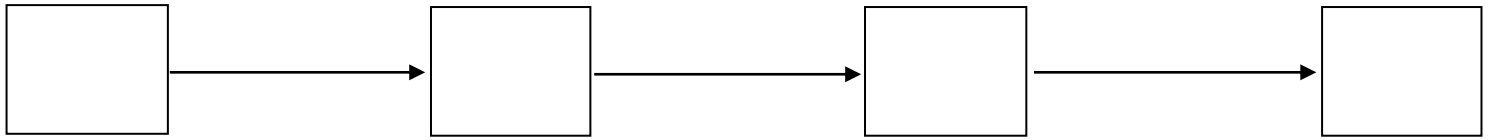
Networks and Cryptography

- ISO/OSI model
- Conceptually, each host communicates with peer at each layer

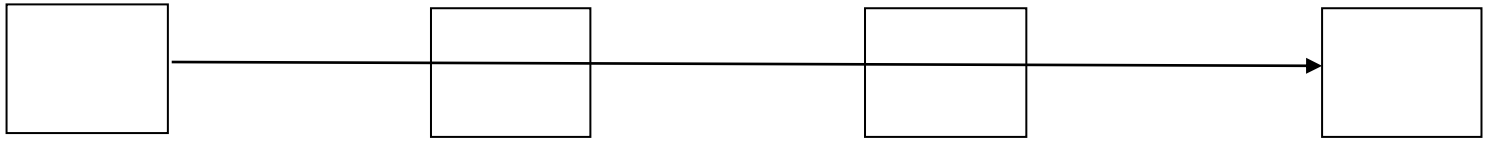


Link and End-to-End Protocols

Link Protocol



End-to-End (or E2E) Protocol



Encryption

- Link encryption
  - Each host enciphers message so host at “next hop” can read it
  - Message can be read at intermediate hosts
- End-to-end encryption
  - Host enciphers message so host at other end of communication can read it
  - Message cannot be read at intermediate hosts

Examples

- SSH protocol
  - Messages between client, server are enciphered, and encipherment, decipherment occur only at these hosts
  - End-to-end protocol
- PPP Encryption Control Protocol
  - Host gets message, deciphers it
    - Figures out where to forward it
    - Enciphers it in appropriate key and forwards it
  - Link protocol

Cryptographic Considerations

- Link encryption
  - Each host shares key with neighbor
  - Can be set on per-host or per-host-pair basis
    - Windsor, stripe, seaview each have own keys
    - One key for (windsor, stripe); one for (stripe, seaview); one for (windsor, seaview)
- End-to-end
  - Each host shares key with destination
  - Can be set on per-host or per-host-pair basis
  - Message cannot be read at intermediate nodes

Traffic Analysis

- Link encryption
  - Can protect headers of packets
  - Possible to hide source and destination
    - Note: may be able to deduce this from traffic flows
- End-to-end encryption
  - Cannot hide packet headers
    - Intermediate nodes need to route packet
  - Attacker can read source, destination

Example Protocols

- Securing Electronic Mail (OpenPGP, PEM)
  - Applications layer protocol
  - Start with PEM as goals, design described in detail; then lool at OpenPGP
- Securing Instant Messaging (Signal)
  - Applications layer protocol
- Secure Socket Layer (TLS)
  - Transport layer protocol
- IP Security (IPSec)
  - Network layer protocol

Transport Layer Security

- Internet protocol: TLS
  - Provides confidentiality, integrity, authentication of endpoints
  - Focus on version 1.2
- Old Internet protocol: SSL
  - Developed by Netscape for WWW browsers and servers
  - Use is deprecated

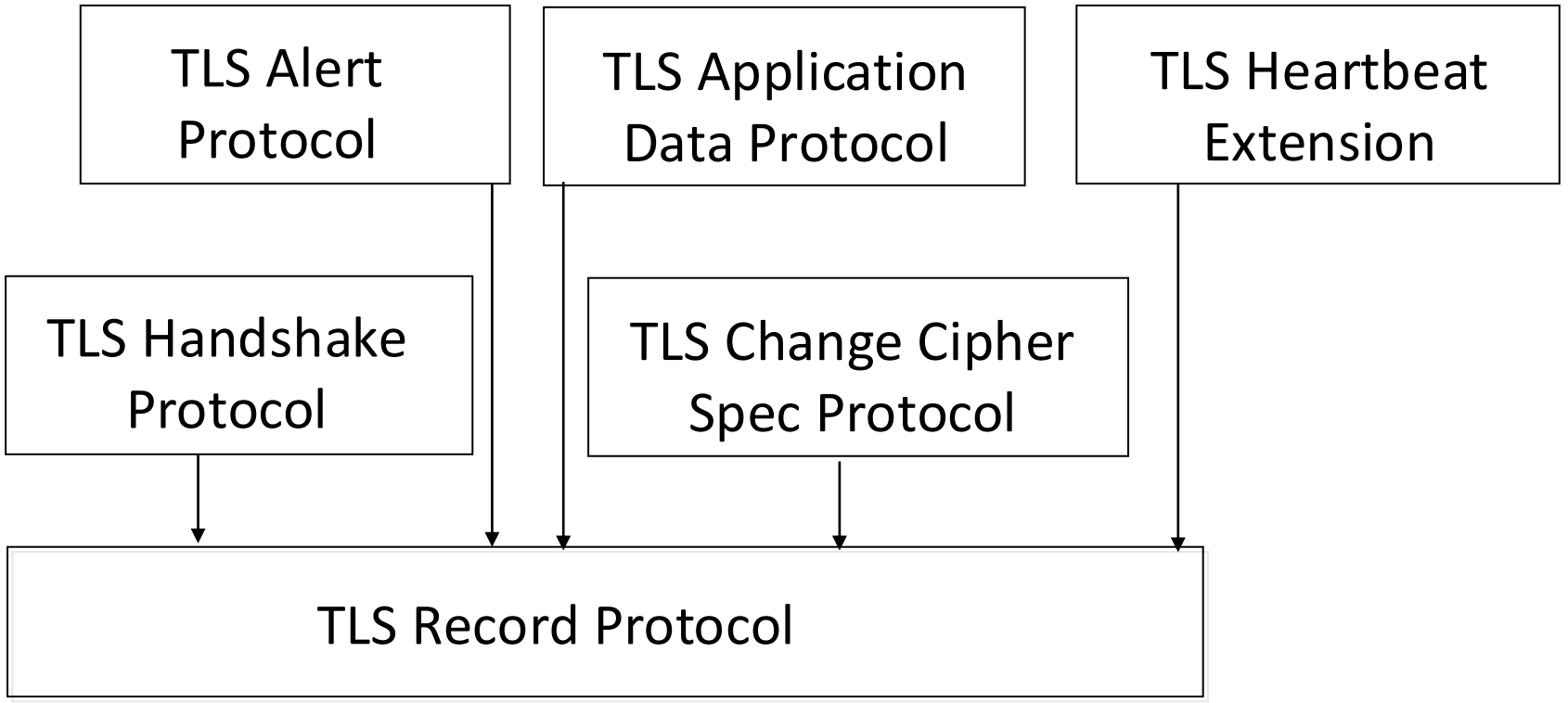
TLS Session

- Association between two peers
  - May have many associated connections
  - Information related to session for each peer:
    - Unique session identifier
    - Peer's X.509v3 certificate, if needed
    - Compression method
    - Cipher spec for cipher and MAC
    - "Master secret" of 48 bits shared with peer
    - Flag indicating whether this session can be used to start new connection

TLS Connection

- Describes how data exchanged with peer
- Information for each connection
  - Whether a server or client
  - Random data for server and client
  - Write keys (used to encipher data)
  - Write MAC key (used to compute MAC)
  - Initialization vectors for ciphers, if needed
  - Sequence numbers for server, client

Structure of TLS



Supporting Cryptography

- All parts of TLS use them
- Initial phase: public key system exchanges keys
  - Messages enciphered using classical ciphers, checksummed using cryptographic checksums
  - Only certain combinations allowed
    - Depends on algorithm for interchange cipher
  - Interchange algorithms: RSA, El Gamal, Diffie-Hellman

Diffie-Hellman: Types

- Diffie-Hellman: certificate contains D-H parameters, signed by a CA
  - DSS or RSA algorithms used to sign
- Ephemeral Diffie-Hellman: DSS or RSA certificate used to sign D-H parameters
  - Parameters not reused, so not in certificate
- Anonymous Diffie-Hellman: D-H with neither party authenticated
  - Use is “strongly discouraged” as it is vulnerable to attacks
- Elliptic curve Diffie-Hellman supports Diffie-Hellman and ephemeral Diffie-Hellman
  - But not anonymous Diffie-Hellman

Derivation of Master Secret

- *master_secret* = *PRF*(*premaster*, "master secret", r_1 || r_2)
  - *premaster* set by client (RSA encrypted), sent to server during setup (TLSv1.2)
  - *premaster* generated using Diffie-Hellman key exchange (TLSv1.2, TLSv1.3)
  - r_1, r_2 random numbers from client, server respectively
- *PRF*(*secret*, *label*, *seed*) = P_*hash*(*secret*, *label* || *seed*)
- P_*hash*(*secret*, *seed*) = HMAC_*hash*(*secret* || *A*(1) || *seed*) ||
  HMAC_*hash*(*secret* || *A*(2) || *seed*) ||
  HMAC_*hash*(*secret* || *A*(3) || *seed*) || …
  - Use first 48 bits of output to set *PRF*
- *A*(0) = *seed*, *A*(*i*) = HMAC_*hash*(*secret*, *A*(*i*-1)) for *i* > 0

Derivation of Keys

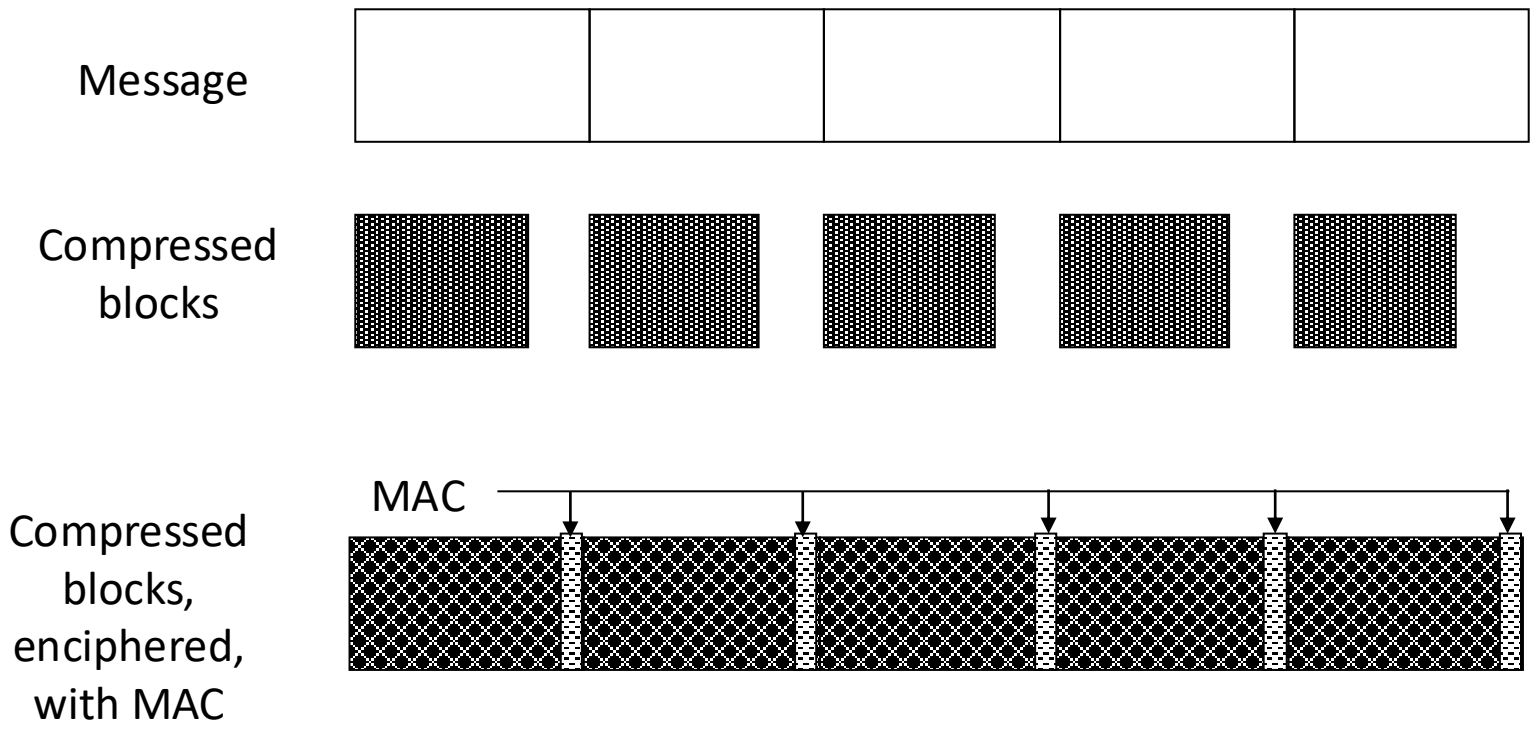
- *key_block* = PRF(*master*, "key expansion", r_1 || r_2)
  - r_1, r_2 as before
- Break it into blocks of 48 bits
  - First two are client, server keys for computing MACs
  - Next two are client, server keys used to encipher messages
  - Next two are client, server initialization vectors
    - Omitted if cipher does not use initialization vector

MAC for Block

hash(*MAC_ws*, *seq* || *TLS_comp* || *TLS_vers* || *TLS_len* || *block*)

- *MAC_ws*: MAC write key
- *seq*: sequence number of *block*
- *TLS_comp*: message type
- *TLS_vers*: TLS version
- *TLS_len*: length of *block*
- *block*: block being sent

TLS Record Layer



Record Protocol Overview

- Lowest layer, taking messages from higher
  - Max block size 2^{14} = 16,384 bytes
  - Bigger messages split into multiple blocks
- Construction
  - Block *b* compressed; call it b_c
  - MAC computed for b_c
    - If MAC key not selected, no MAC computed
  - b_c, MAC enciphered
    - If enciphering key not selected, no enciphering done
  - TLS record header prepended

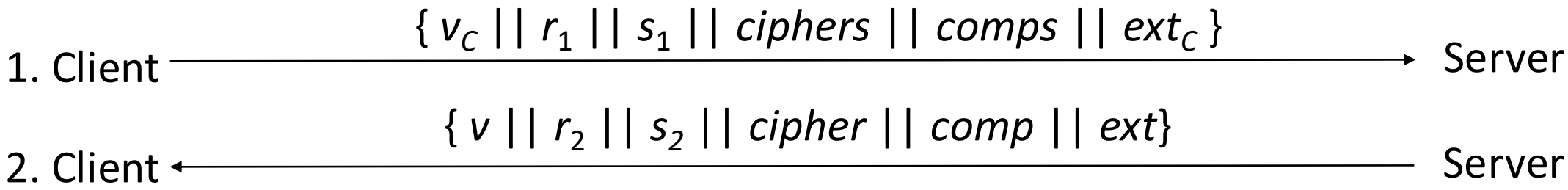
TLS Handshake Protocol

- Used to initiate connection
  - Sets up parameters for record protocol
  - 4 rounds
- Upper layer protocol
  - Invokes Record Protocol
- Note: what follows assumes client, server using RSA as interchange cryptosystem

Overview of Rounds

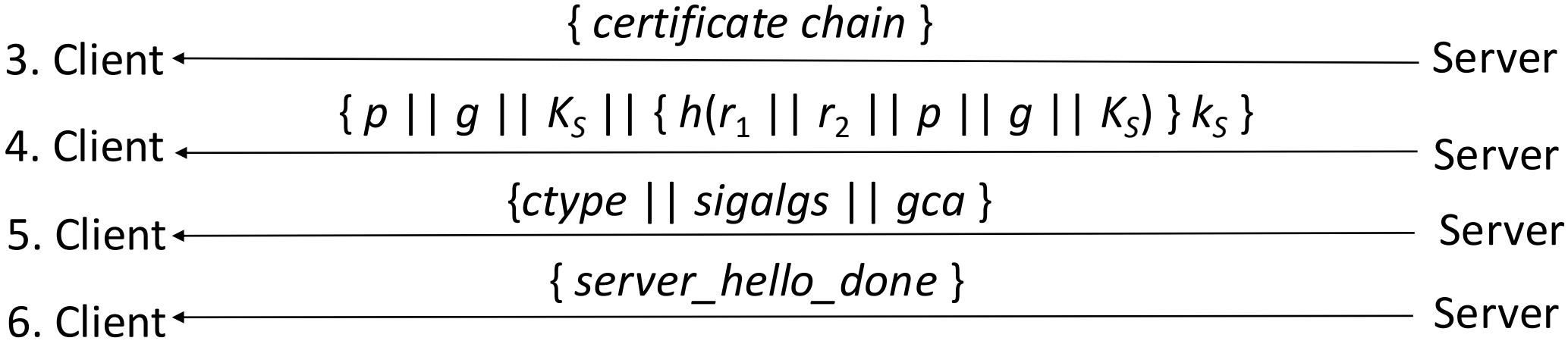
1. Create TLS connection between client, server
2. Server authenticates itself
3. Client validates server, begins key exchange
4. Acknowledgments all around

Handshake Round 1



v_C	Client's version of TLS
v	Highest version of TLS that client, server both understand
r_1, r_2	nonces (timestamp and 28 random bytes)
s_1	Current session id (empty if new session)
s_2	Current session id (if s_1 empty, new session id)
ciphers	Ciphers that client understands
comps	Compression algorithms that client understand
cipher	Cipher to be used
comp	Compression algorithm to be used
ext_C	List of extensions client supports
ext	List of extensions server supports (subset of ext_C)

Handshake Round 2



If server not going to authenticate itself, only last message sent

Second step is for Diffie-Hellman with RSA certificate

Third step omitted if server does not need client certificate

K_S, k_S	Server's Diffie-Hellman public, private keys
ctype	Certificate type accepted (by cryptosystem)
sigalgs	List of hash, signature algorithm pairs server can use
gca	Acceptable certification authorities