Secure Sockets Layer

Secure Sockets Layer

SSL/TLS provides endpoint authentication and communications privacy over the Internet using cryptography.

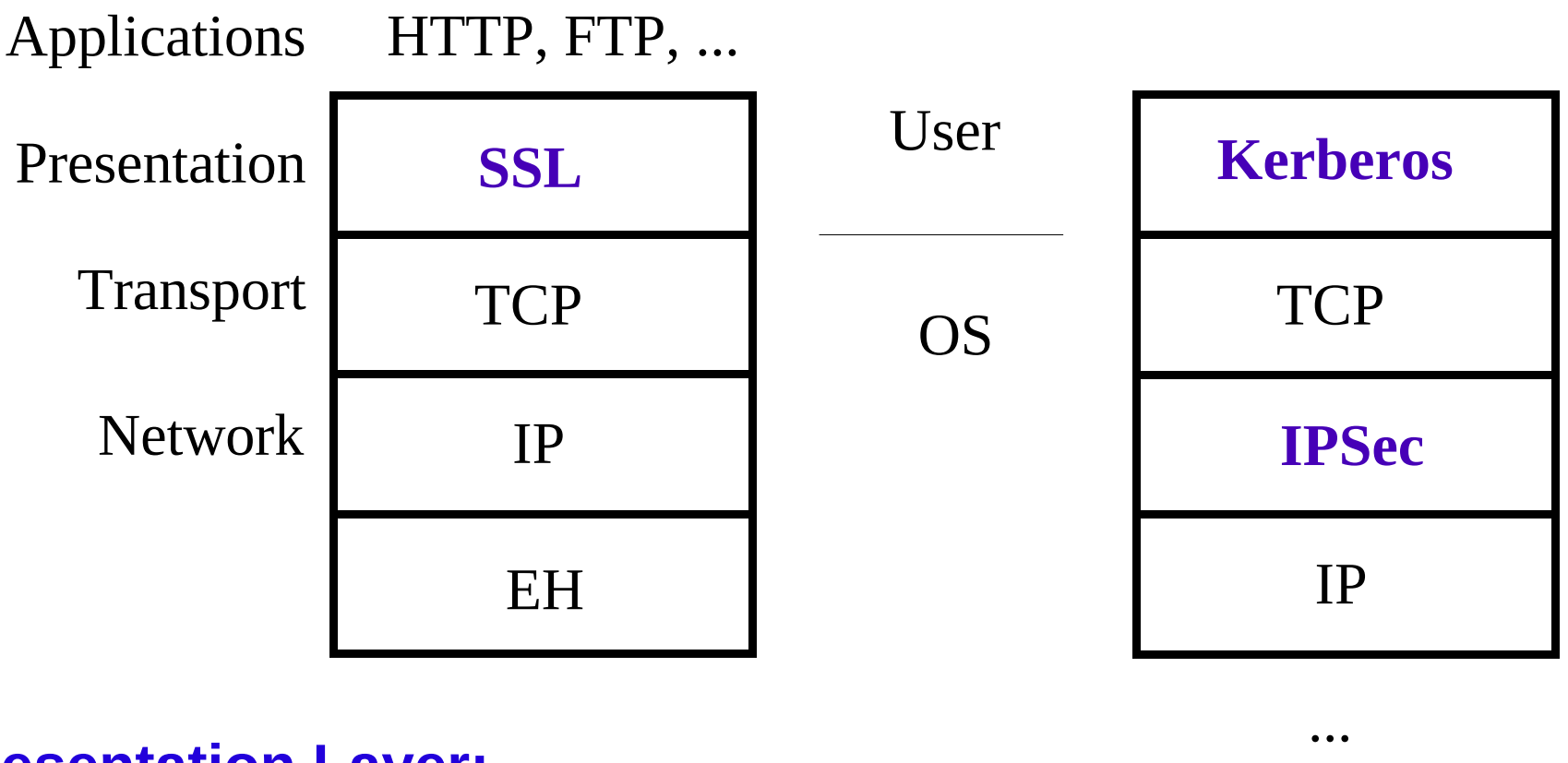
For web browsing, email, faxing, other data transmission.

In typical use, only the server is authenticated while the client remains unauthenticated

Mutual authentication requires PKI deployment to clients.

Protocols allow client/server applications to communicate in a way designed to prevent eavesdropping, tampering, and message forgery.

Secure Sockets Layer



Presentation Layer:

Provides independence from differences in data representation among applications. OS functions translate data formats (app protocol syntax) to a uniform network format (bit stream to be transmitted), and vice versa so as to eliminate network compatibility problems. Takes care of encryption and data compression.

Secure Sockets Layer

Developed by Netscape Communications Corp (1995)

Ensures data privacy: transmission of data via encryption

Supports Server and Client authentication

Supports authentication of service via certificate

Ensures data integrity

Application independent - ftp, http, telnet are layered on top of it. Mainly used in https applications.

Can negotiate encryption keys

Sits on top of TCP/IP, does not require OS changes

Can be used to create a tunnel for VPN

Encryption and compression apply only to application layer

Secure Sockets Layer

Three phases:

1. Peer negotiation for algorithm support (see below)
2. Public key encryption based key exchange and certificate based authentication
3. Symmetric cipher based traffic encryption

Secure Sockets Layer

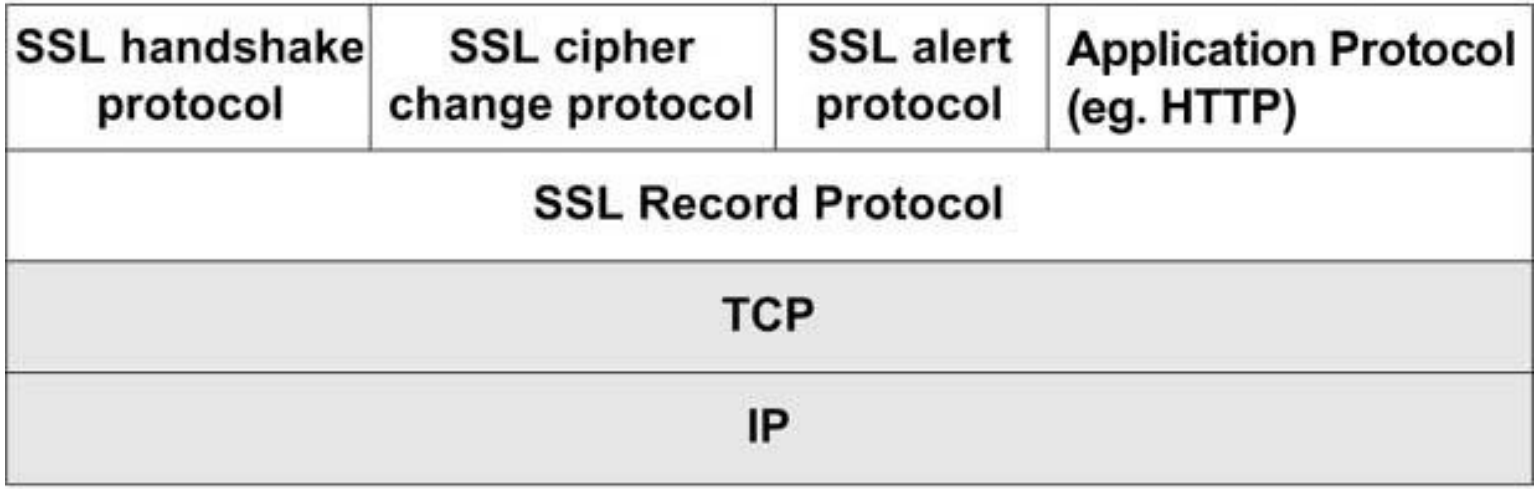
Security features:

- Numbering all the records and using the sequence number in the Msg. Authen. Codes. (to prevent replay attacks)
- Uses HMAC for message integrity check
- Protection against several other known attacks
  - MIM attack involving a downgrade of the protocol to a less secure version via integrity check of initial handshake
  - identity fraud via digital signatures
  - known plaintext attacks via strong cryptographic algorithms
- The message that ends the handshake ("Finished") sends a hash of all the exchanged data in handshake (Integrity check to prevent Man in the Middle and Truncation attacks).

Secure Sockets Layer

Secure Sockets Layer

Uses several protocols organized into layers as follows:



SSL Record Protocol: handles data security and integrity; encapsulates data sent by higher level protocols

Handshake, Cipher change, Alert: establish a connection; session management, crypto management, SSL message transfer

Secure Sockets Layer

Definitions:

Connection: logical 2-node peer-to-peer link – provides a service

Session: association between peers defining crypto algorithms, sequence numbers, etc. Created by handshake protocol. Used to avoid renegotiation of parameters from connection to connection

Session State:

session identifier: generated by the receiver
peer certificate: X.509 spec
compression method: prior to encryption
CipherSpec: encryption, integrity, and hash algorithms
MasterSecret: 48 byte shared secret

Secure Sockets Layer

Definitions:

Connection State:

Random numbers - chosen by server and client to make crypto breaking harder

Server write MAC secret - used on data from server

Client write MAC secret - used on data from client

Server write secret key - server encryption, decryption by client

Client write secret key - client encryption, decryption by server

Initialization vectors - for CBC ciphers

Sequence number - for both transmitted and received messages on both client and server sides

Secure Sockets Layer

SSL Record Protocol:

Fragment the data that needs to be sent: create records
Encapsulate them with appropriate headers
Create an encrypted object that can be sent over TCP

Header of each record: length of record and of data block

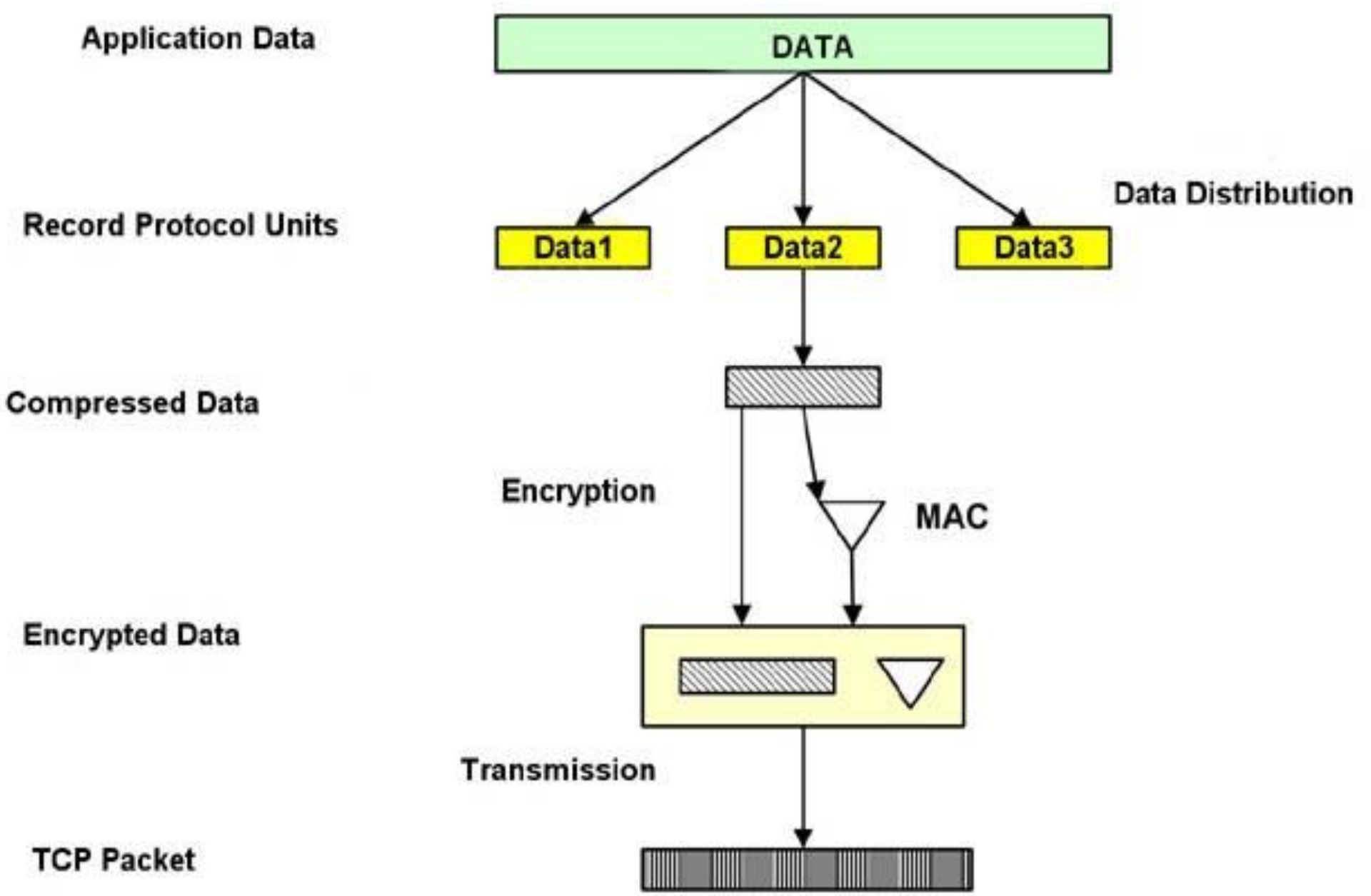
Contents of record after header: data, padding, MAC

MAC = hash {secret key, data+padding, sequence number}
where hash uses specified (negotiated) algorithm like SHA-1

Encrypted Object: encrypt record plus MAC

Header of EO: content-type: which of four protocols to use to handle the data in the EO after decryption.
Protocol Major and Minor version numbers.

Secure Sockets Layer



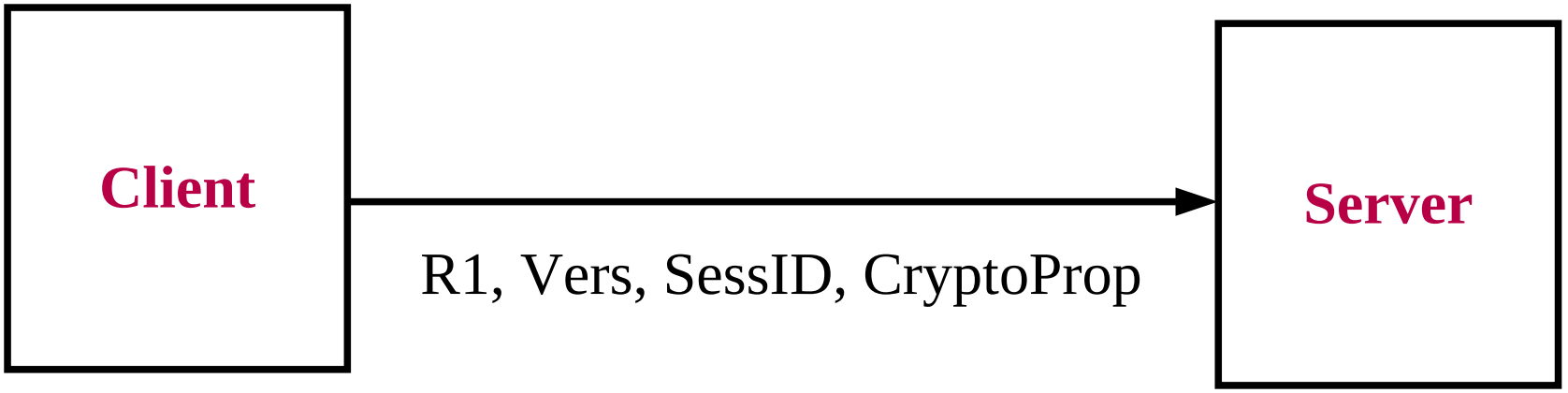
Secure Sockets Layer

Handshake Protocol:

To initiate a session: crypto negotiation

phase 1: initiate (client_hello), identity not revealed!

Establish logical connection, negotiate session parms



Version: Highest SSL version supported by client
R1: random number
SessionID: non-0: resume earlier session, modify parms of this session, spawn an independent connection without handshake
0: initiate a session

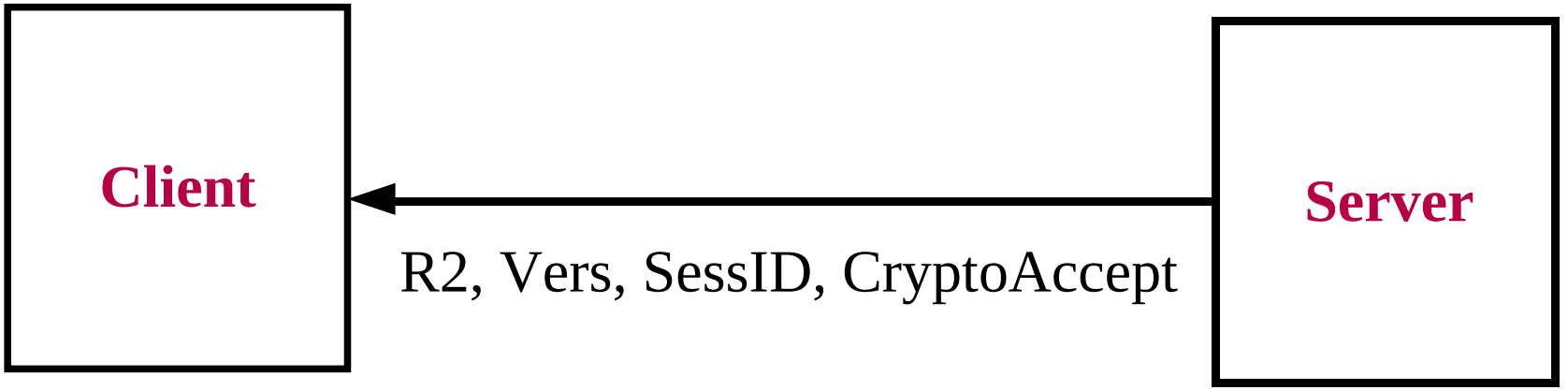
Secure Sockets Layer

Handshake Protocol:

To initiate a session: crypto negotiation

phase 1: initiate (client_hello), identity not revealed!

Establish logical connection, negotiate session parms



Version: Lowest SSL version supported by server
R2: a random number
SessionID: same as client's, if non-0, otherwise an ID decided by server
CryptoAccept: key exchange method, encrypt algorithm, hash function

Secure Sockets Layer

Handshake Protocol:

To initiate a session: crypto negotiation

phase 2: authenticate (server_hello)

Certificate(s): chain of certificates to a trusted CA to authenticate server
Diffie-Hellman value: optional
Request for certificate from client: optional
server_done: completes the message sequence

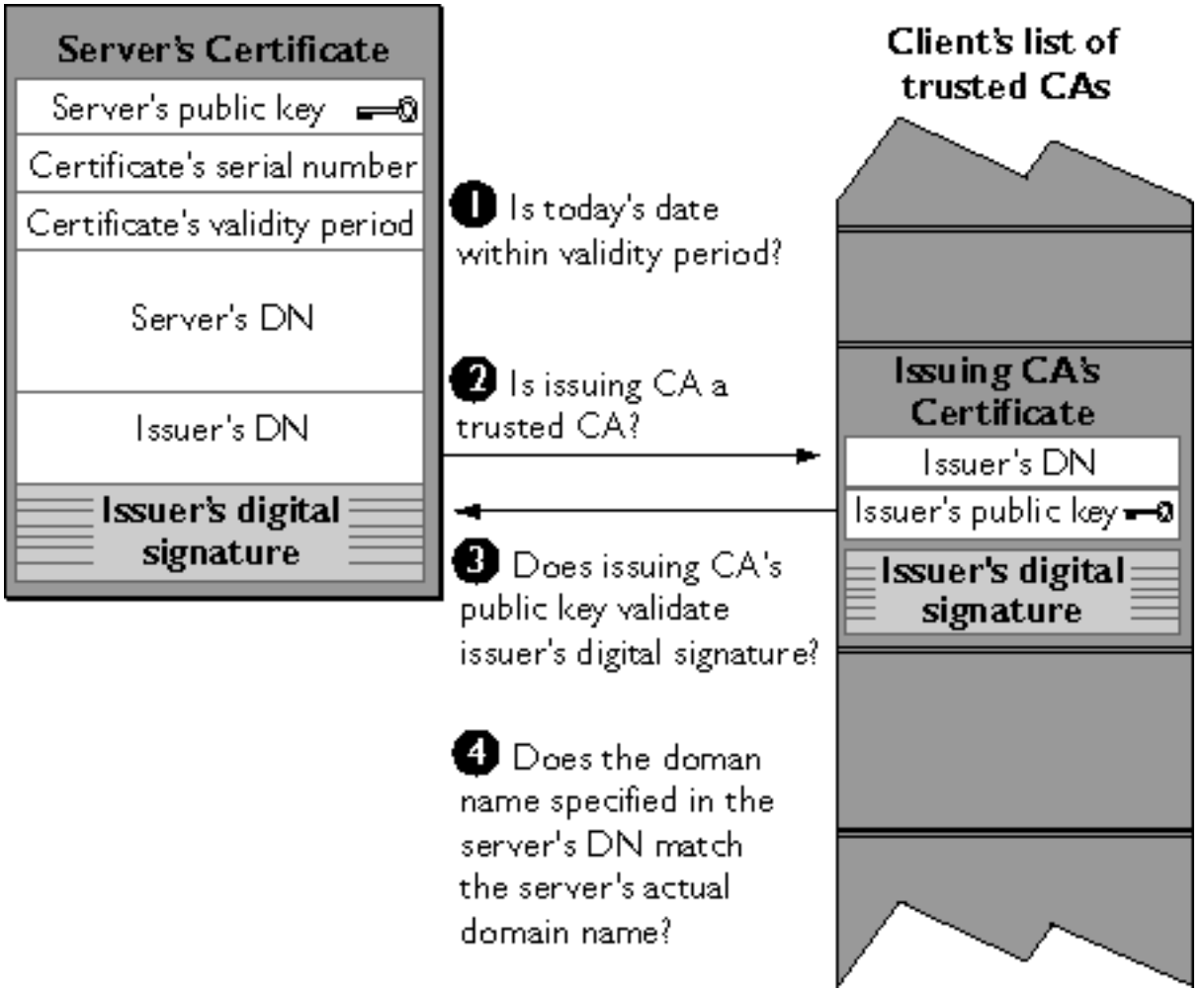
Secure Sockets Layer

Handshake Protocol:

To initiate a session: crypto negotiation

phase 2: authenticate (server_hello)

Client verifies certificate, checks date and invalidation lists
Client checks that the certifying authority is trusted
Client checks the CA's public key against that of the certificate
Client checks that the domain name in the certificate matches that of server



How the client authenticates the server

Secure Sockets Layer

Handshake Protocol:

To initiate a session: too many secrets

Client chooses random number S (pre-master secret) and encrypts it with server's public key (server{S})

Client computes master key as *K*=*f*(S,R1,R2), computes hash(*K*+*msgs*)

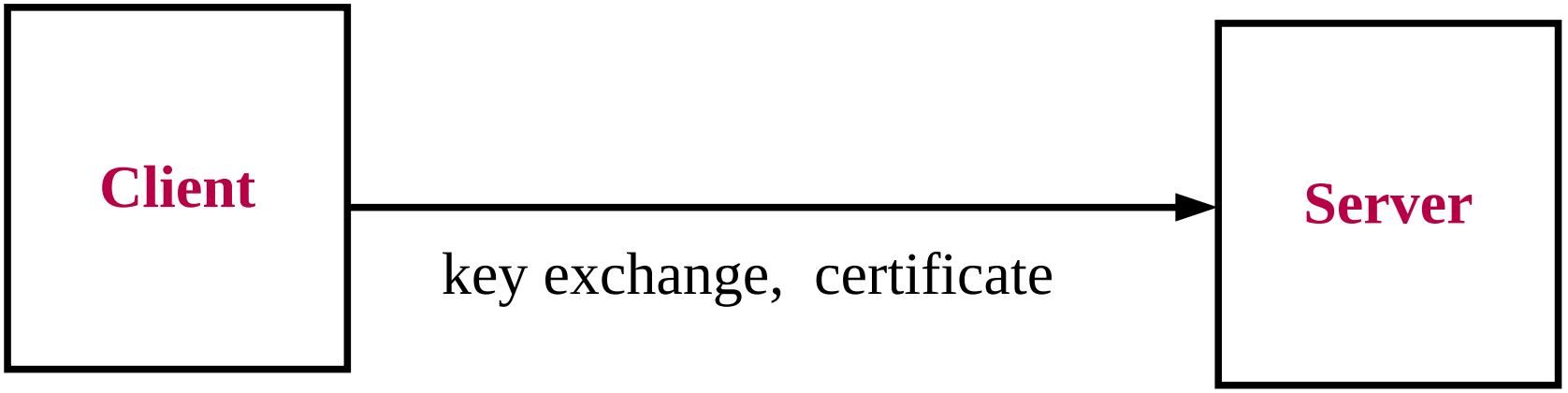
Six secret keys - 3 from client to server and 3 from server to client integrity, encryption, initialization vector (derived from S)

Secure Sockets Layer

Handshake Protocol:

To initiate a session: crypto negotiation

phase 3: start key exchange



S sent encrypted by server's public key, server computes master secret

Key exchange msg: delivers the keys, depends on the agreed key exchange method

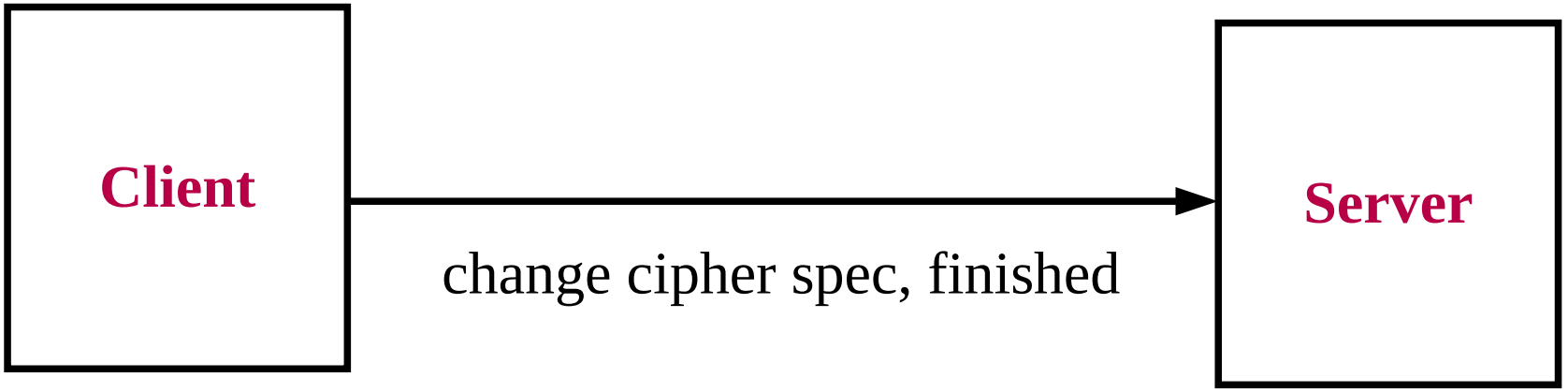
Send certificate of client: if requested by server

Secure Sockets Layer

Handshake Protocol:

To initiate a session: crypto negotiation

phase 4: confirmation and setup



Change cipher spec msg: crypto spec now considered agreed
Setup of algorithms: make cryptosystems ready to go
Finished msg: encrypted with agreed upon crypto algorithms and keys
so server can verify that communication is possible.
Is the hash of all the messages in the handshake.

Secure Sockets Layer

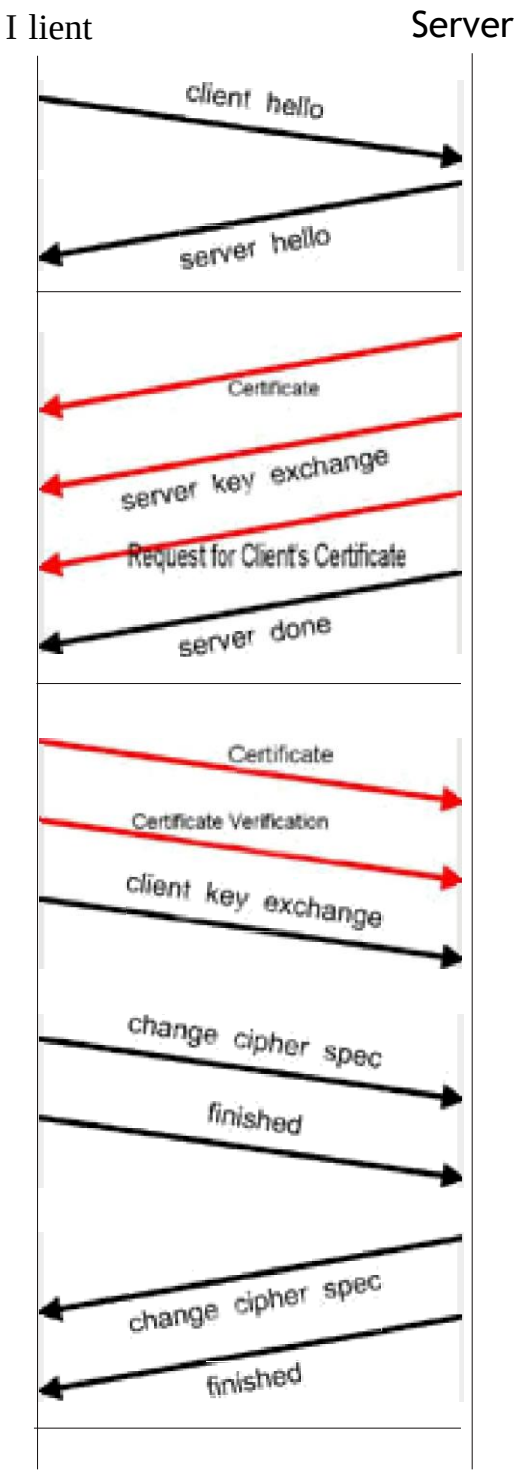
Handshake Protocol:

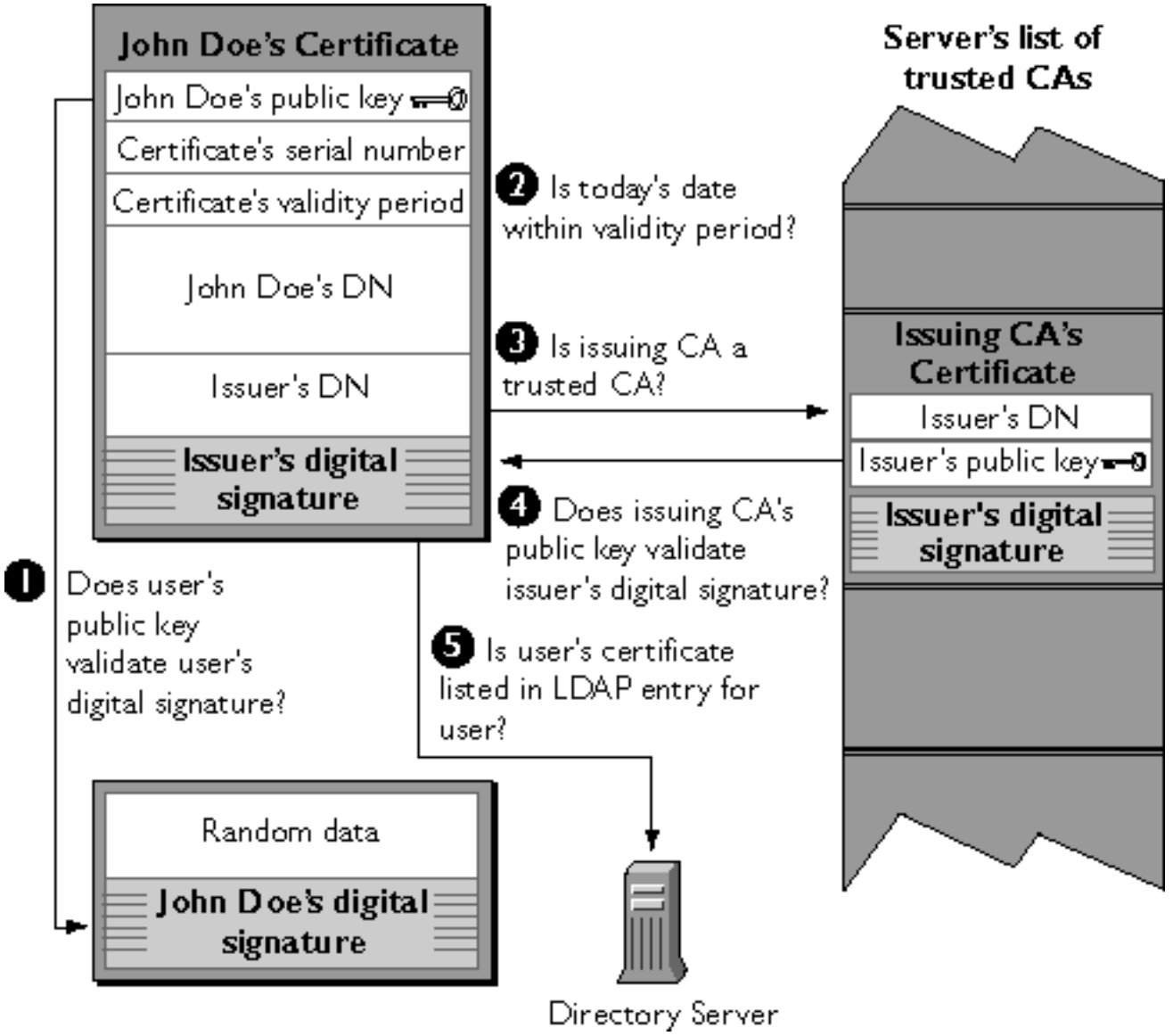
To initiate a session: crypto negotiation

phase 4: confirmation and setup

Same finished message is sent back to client

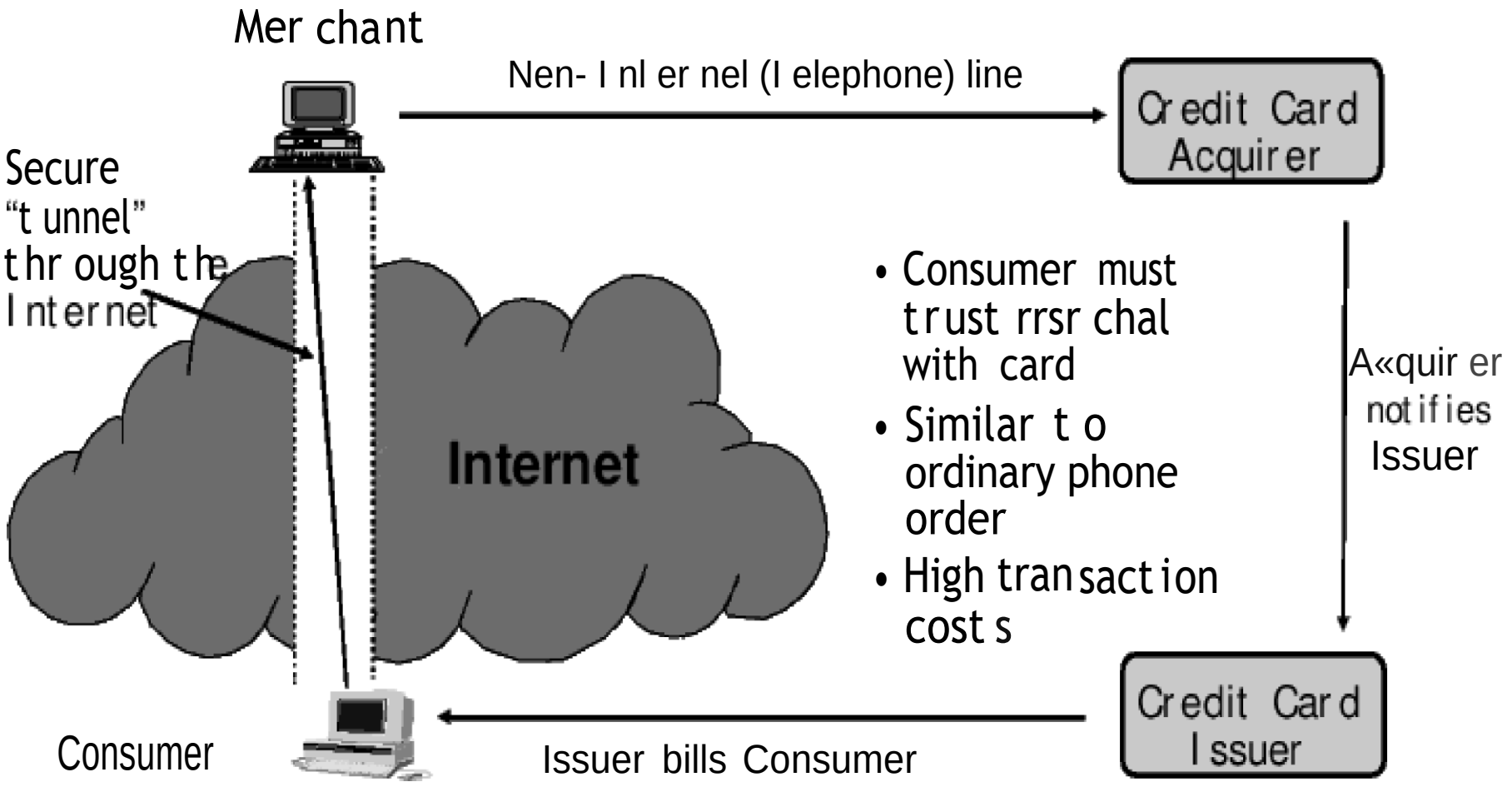
Session is terminated and the TCP connection is closed but the "state" of the session is saved to be reopened later with same parameters.


I lient
Server
client hello
server hello
Certificate
server key exchange
Request for Client's Certificate
server done
Certificate
Certificate Verification
client key exchange
change cipher spec
finished
change cipher spec
finished



How the server authenticates the client

The Mai Usage of SSL



Secure Sockets Layer

Session Resumption:

Per-session master key is established using expensive public key cryptography

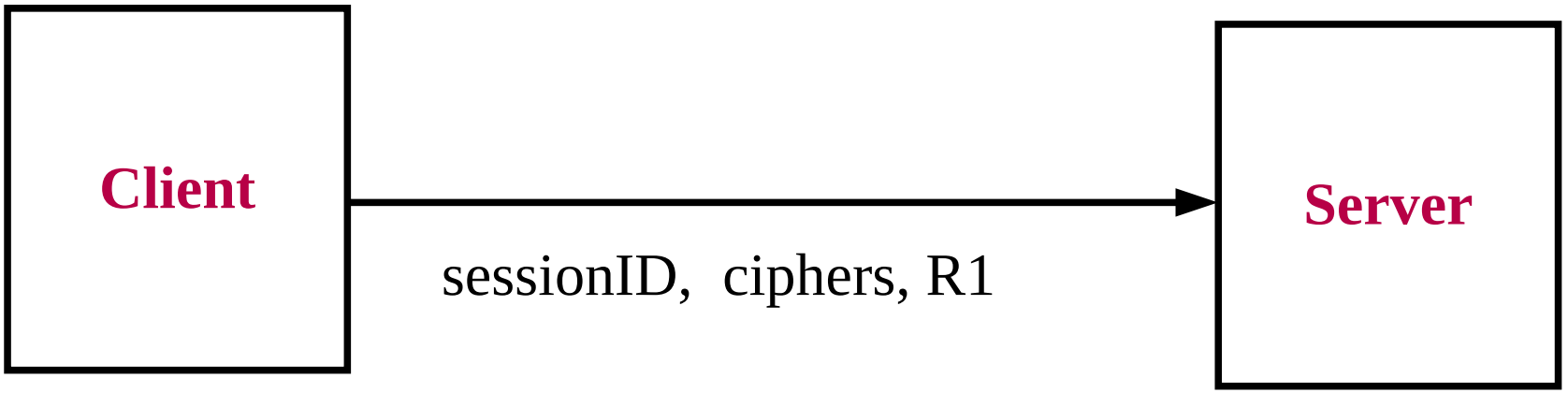
Connections cheaply derived from master key with handshake involving nonces, not public keys

SessionID and master key for the session is stored by server to support resumption

If server loses the state, it can be reestablished by the client sending S encrypted with server's public key

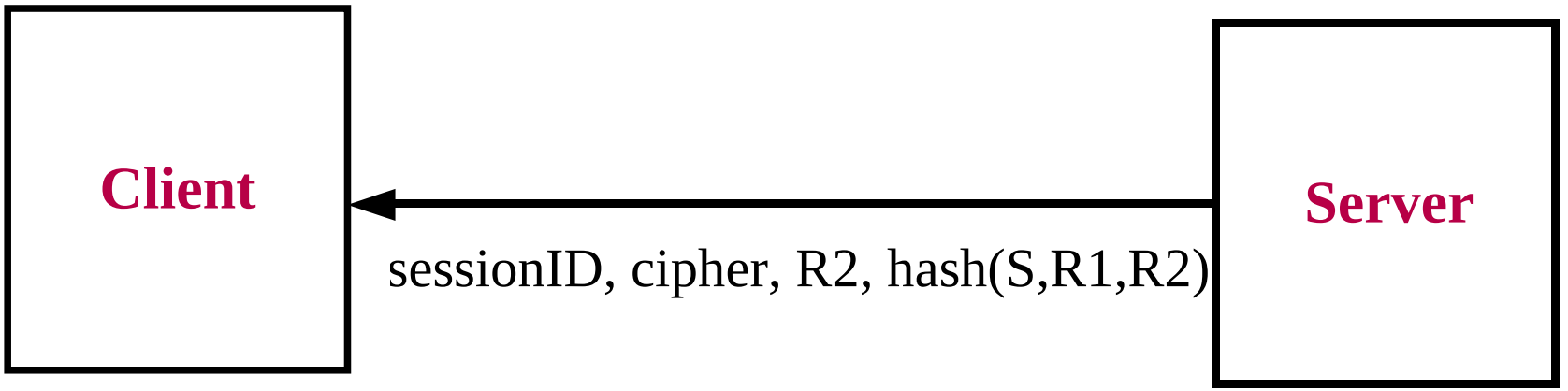
Secure Sockets Layer

Handshake Protocol (Resumption): Using Session-ID



Secure Sockets Layer

Handshake Protocol (Resumption): Using Session-ID

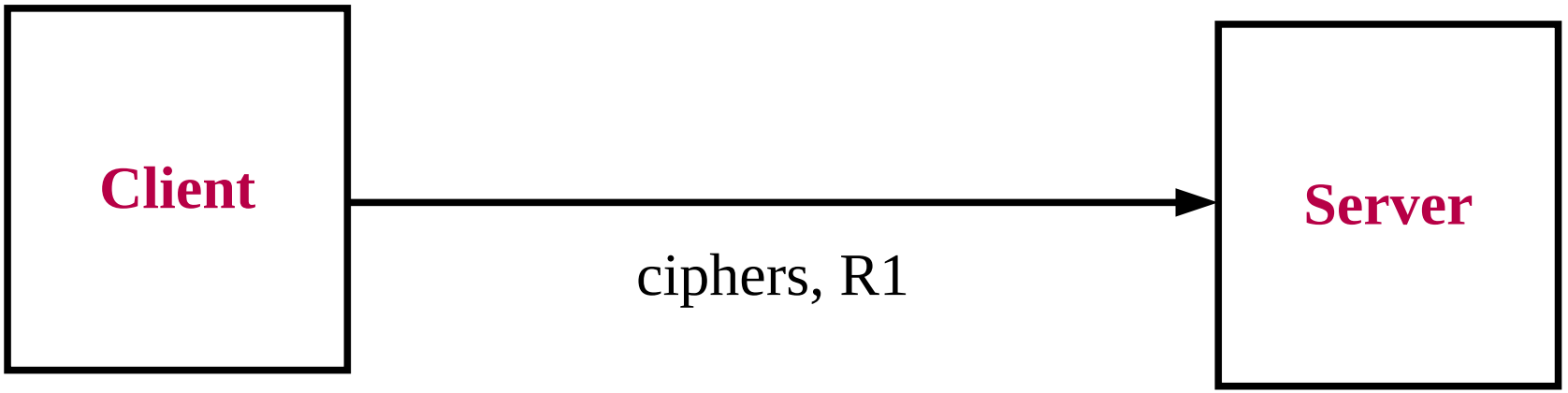


Secure Sockets Layer

Handshake Protocol (Resumption): Using Session-ID

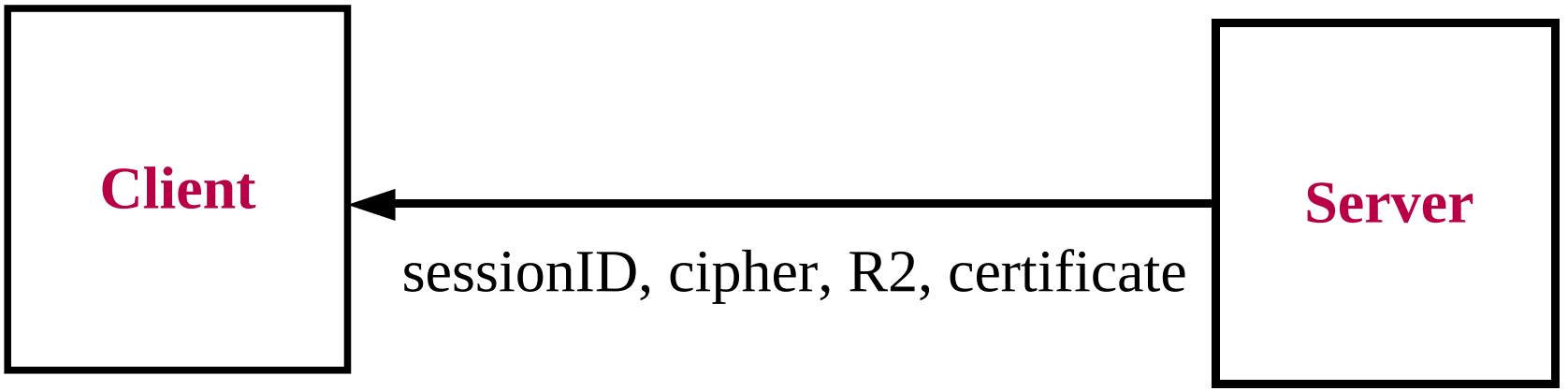
Secure Sockets Layer

Handshake Protocol (Resumption): Forgot Session-ID



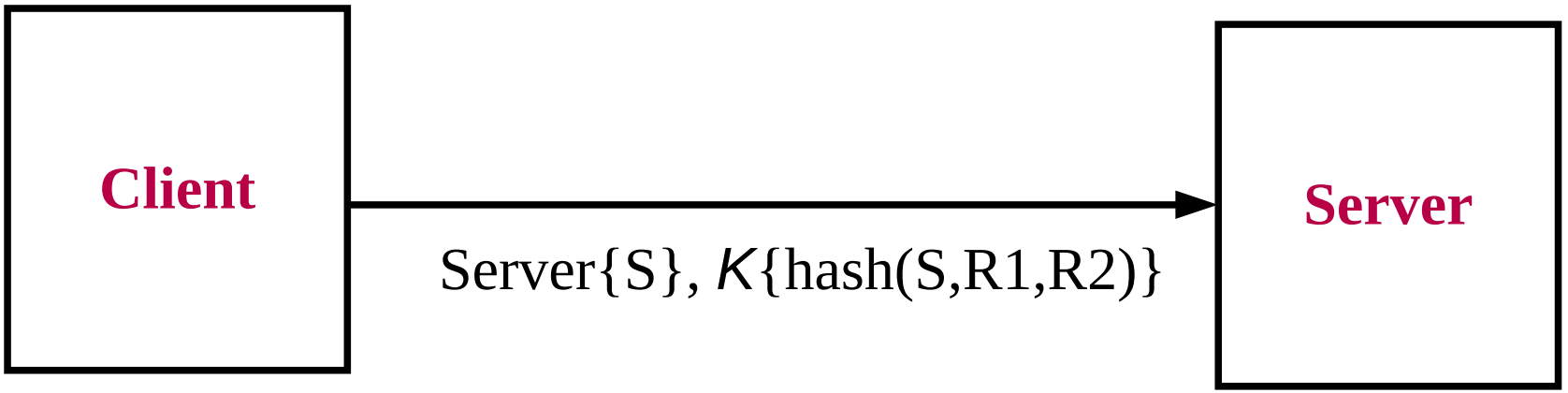
Secure Sockets Layer

Handshake Protocol (Resumption): Forgot Session-ID



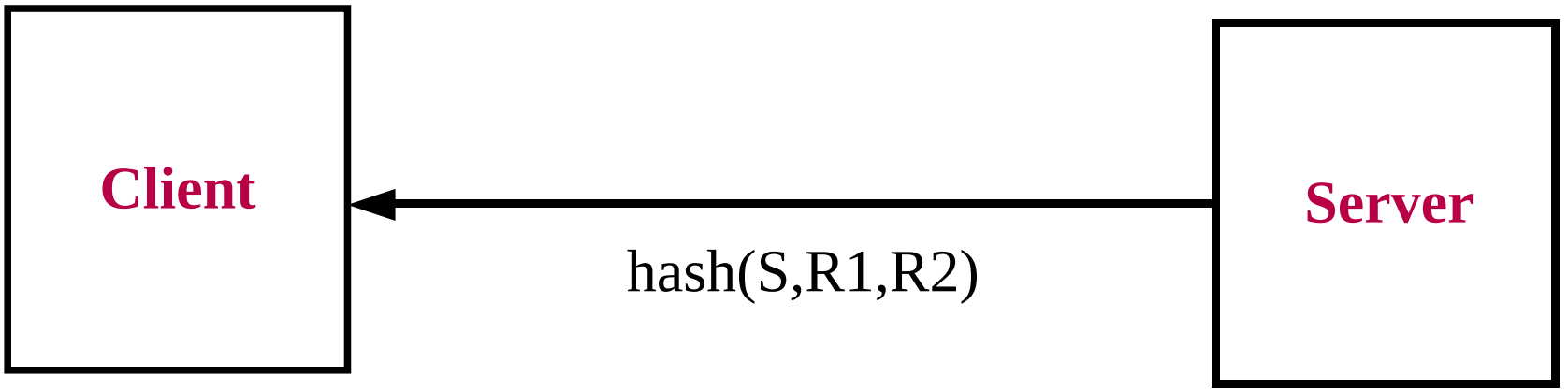
Secure Sockets Layer

Handshake Protocol (Resumption): Forgot Session-ID



Secure Sockets Layer

Handshake Protocol (Resumption): Forgot Session-ID



Secure Sockets Layer

Computing Keys:

If Diffie-Hellman is used to compute pre-master Secret S, fixed DH is allowed! Same session key repeated!! Ephemeral DH, ephemeral elliptic curve DH are also allowed for perfect forward secrecy (e.g. gmail)

If kept around in memory, a fixed DH key could compromise future communications

So it is hashed with nonces to get the master secret (Stealing master secret only affects this communication)