

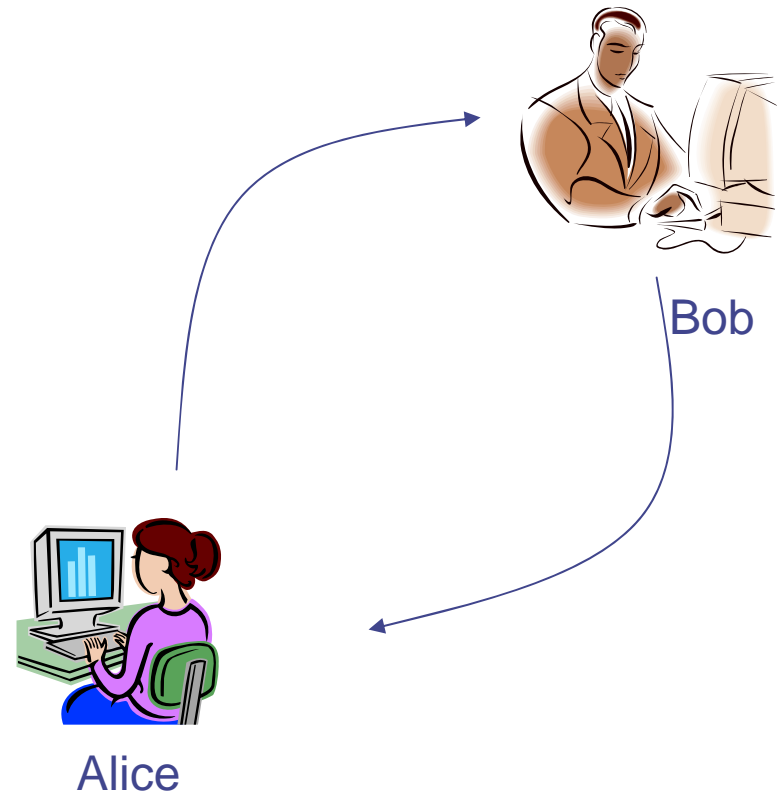
Web Security

Yuan Xue



Case Study

- ◆ Bob sells BatLab on Internet
 - Software
 - License
- ◆ Alice buys BatLab via Web
 - Credit card information
 - Number of licenses



Security Issues

◆ Client → Server

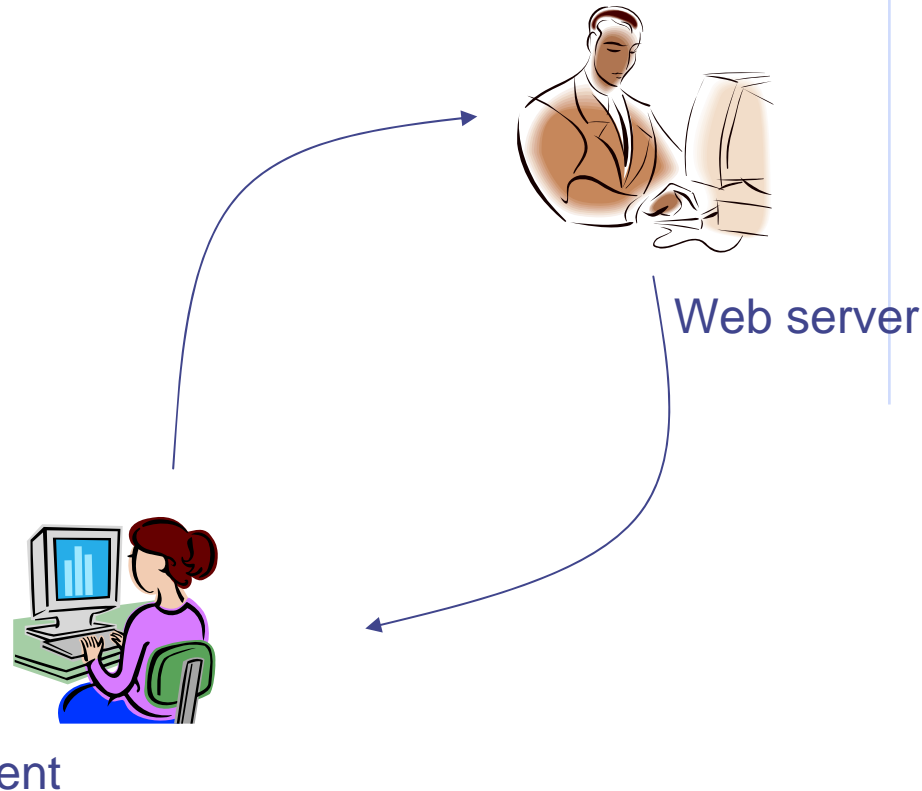
- Authentication of Bob
- Confidentiality and integrity of the order information

◆ Server → Client

- Confidentiality and integrity of the licenses
- Integrity of the software

◆ Other Issues

- Deny the order
- Replay the order
- DoS
- Attacks to web server or client



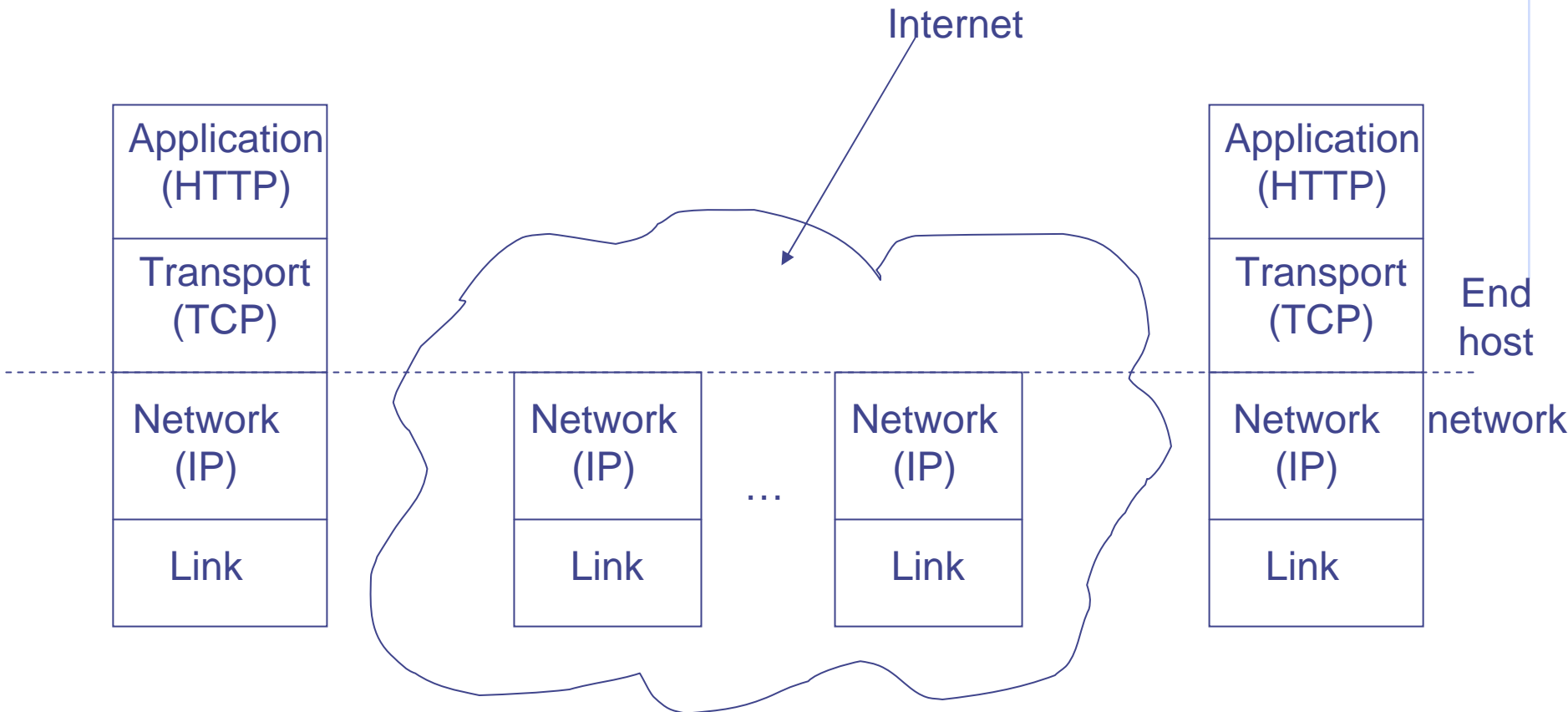
Security Mechanisms

- ◆ What do we need?
 - Authentication
 - ◆ Certificate
 - Encryption
 - ◆ Symmetric ciphers
 - Key distribution
 - ◆ Public key



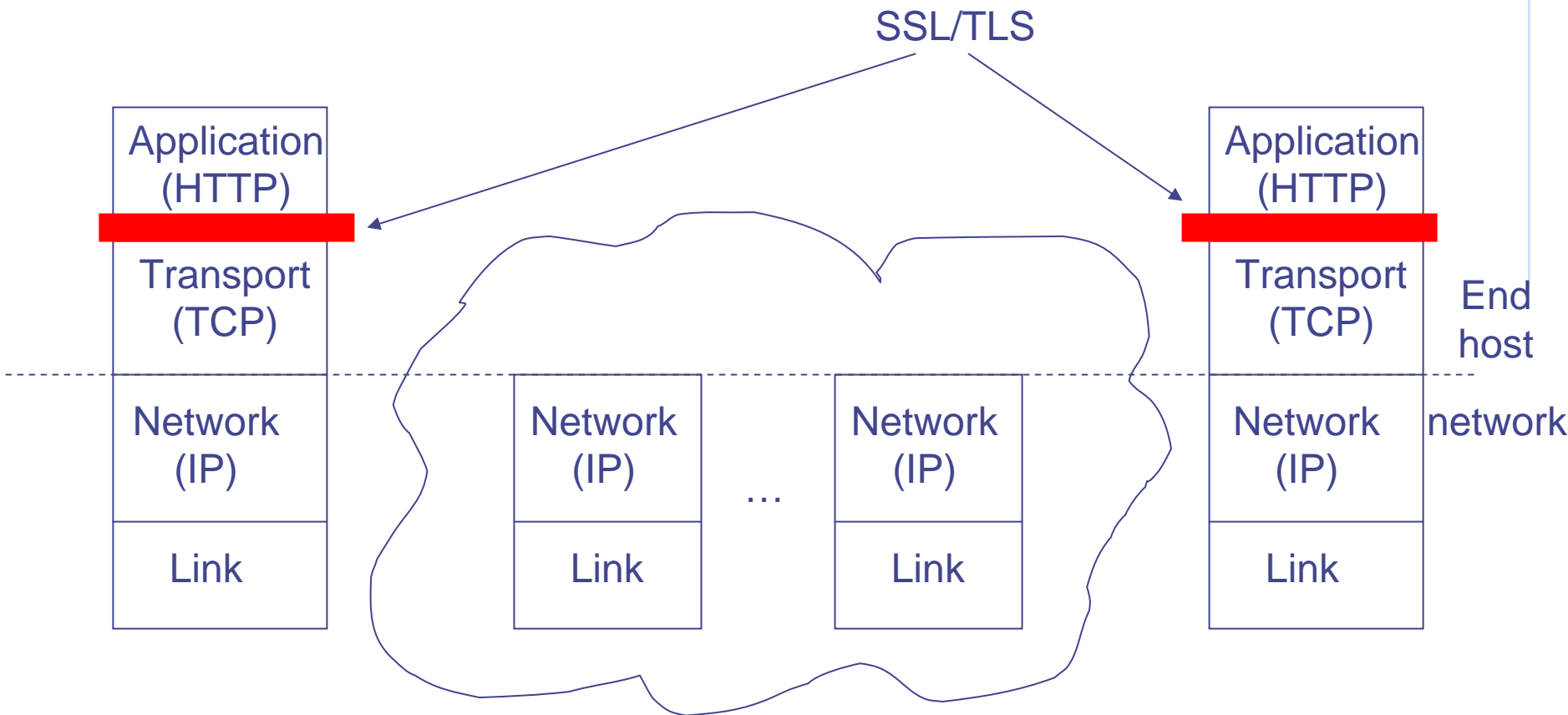
Networking Design

◆ Network Stack/Layer



Security Mechanism Placement

- ◆ SSL (Secure Socket Layer)
- ◆ TLS (Transport Layer Security)



SSL Design

- ◆ What do we want ultimately?
 - Communication between client and server
 - ◆ Confidentiality + data integrity + source authentication
- ◆ How?
 - Authentication → public-key based authentication
 - Confidentiality → Symmetric encryption
 - Integrity → MAC
- ◆ What do we need?
 - Certificate for authentication
 - Shared key 1 for encryption
 - Shared key 2 for MAC
 - Initialization vector for mode of operation



SSL Design

◆ A simple illustration

Application data

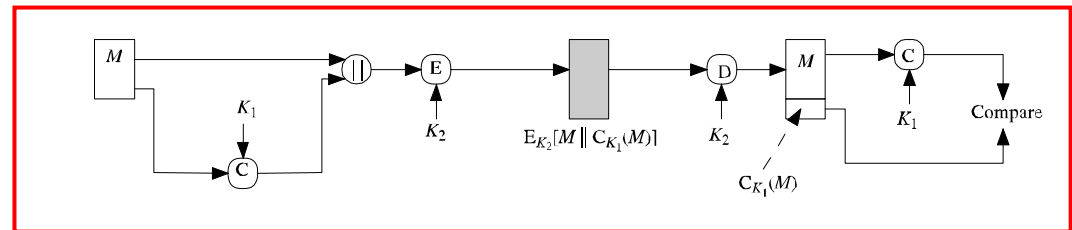
fragment

fragment

fragment

fragment MAC

Encrypted



SSL Design

◆ Improving the performance

- Key hierarchy
 - ◆ Master secret key: between client and server
 - ◆ Session secret key: for each connection
- Compression

◆ Choice of cryptographic algorithms

- Feasibility in symmetric cipher
 - ◆ Block ciphers: DES, 3DES, IDEA, etc
 - ◆ Stream ciphers: RC4 (RC4-40, RC4-128)
- Choice of MAC
 - ◆ HMAC? -- Well... a similar one, replace XOR with concatenation
 - ◆ Either MD5 or SHA-1



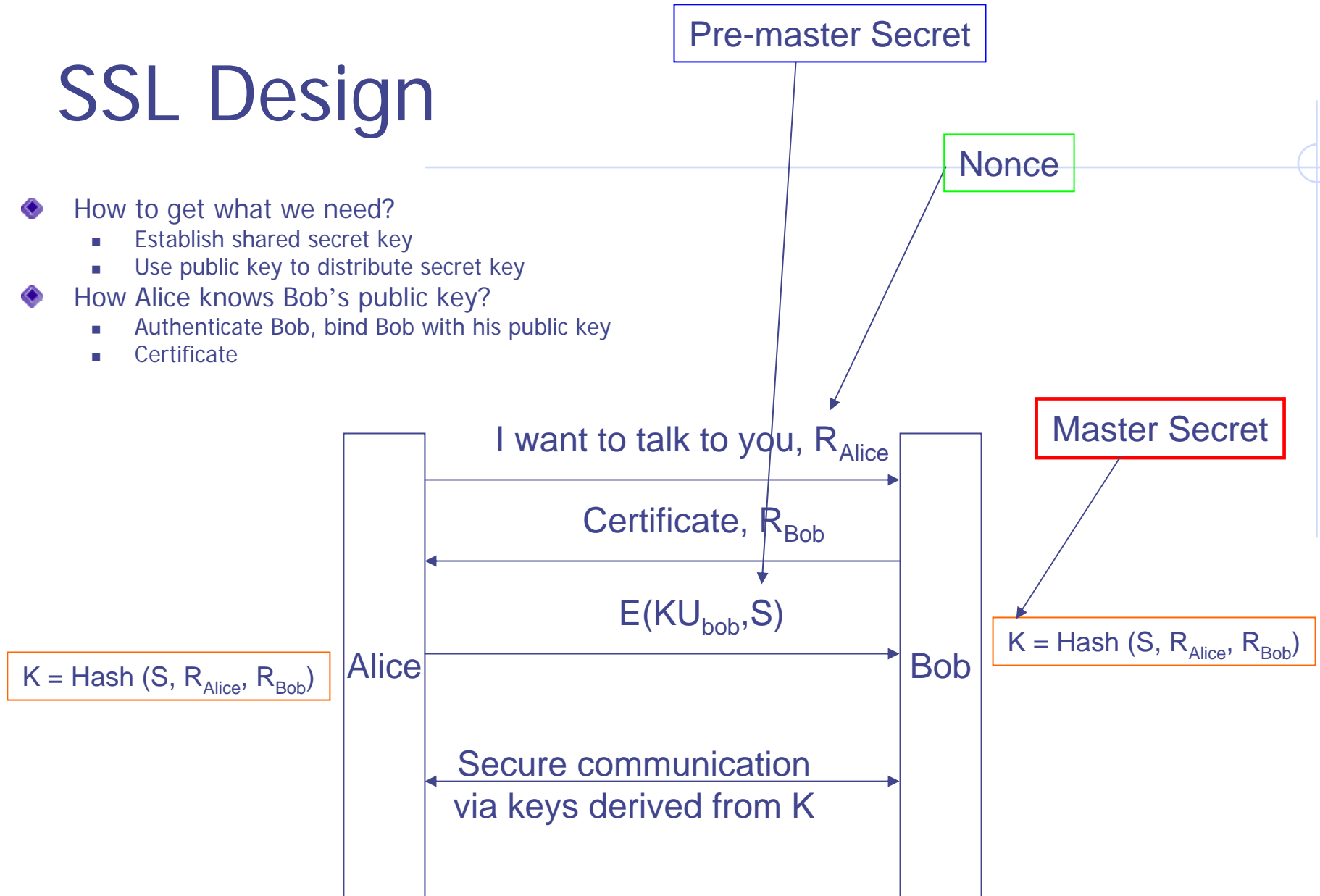
SSL Design

- ◆ How to get what we need?
 - Establish shared secret key
 - Use public key to distribute secret key
- ◆ How could Alice know Bob's public key?
 - Authenticate Bob, bind Bob with his public key
 - Certificate



SSL Design

- ◆ How to get what we need?
 - Establish shared secret key
 - Use public key to distribute secret key
- ◆ How Alice knows Bob's public key?
 - Authenticate Bob, bind Bob with his public key
 - Certificate



SSL Design

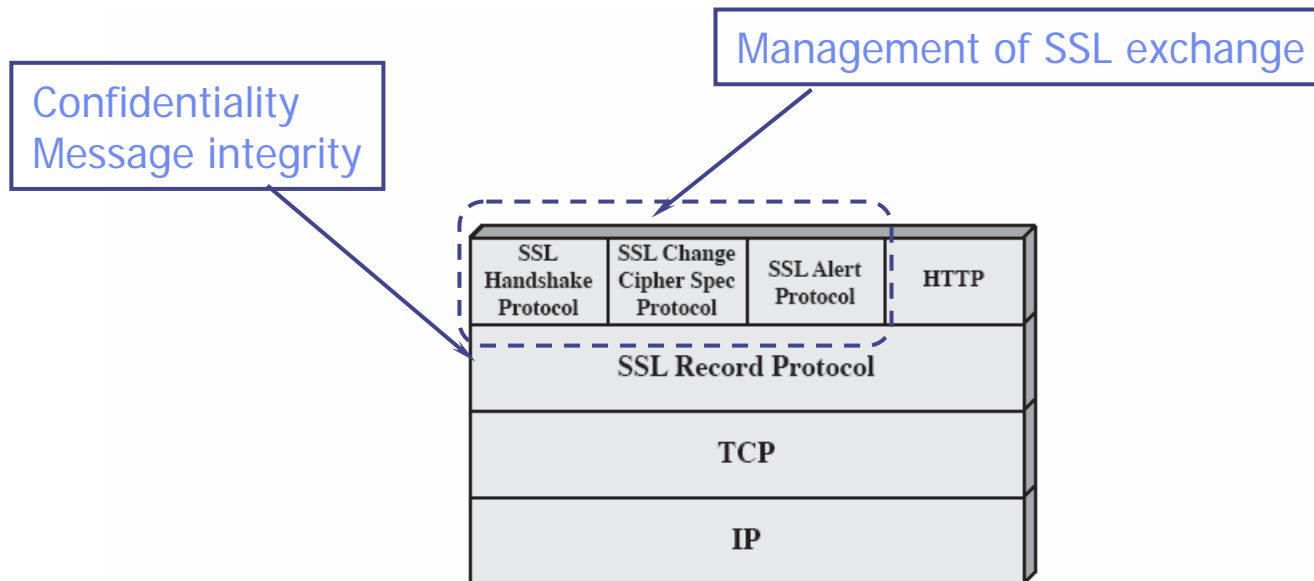
◆ Other considerations

- Authentication of client
- What if RSA can not be used
 - ◆ Diffie-Hellman
- How does Bob know what ciphers Alice wants to use?
- ...



Finally... Full Version of SSL

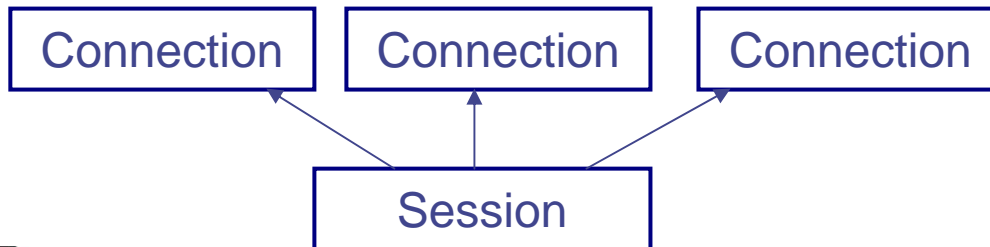
- ◆ SSL consists of two layers of protocols
 - SSL Record Protocol
 - ◆ Basic security services to higher layer protocols, e.g., HTTP
 - SSL Handshake Protocol
 - ◆ Server and client authenticate each other
 - ◆ Negotiate encryption, MAC algorithm, and cryptographic keys
 - SSL Change Cipher Spec Protocol
 - SSL Alert Protocol



Full version of SSL

◆ SSL session vs. SSL connection

- Session state
 - ◆ Session ID
 - ◆ Master secret key
 - ◆ Cipher spec
 - data encryption algorithm (DES, IDEA..)
 - hash function (MD5, SHA-1, ...)
 - cryptographic attribute (hash size)
 - ◆ peer certificate
 - ◆ compression method
 - ◆ Is resumable
 - Whether the session can be used to initiate new connections



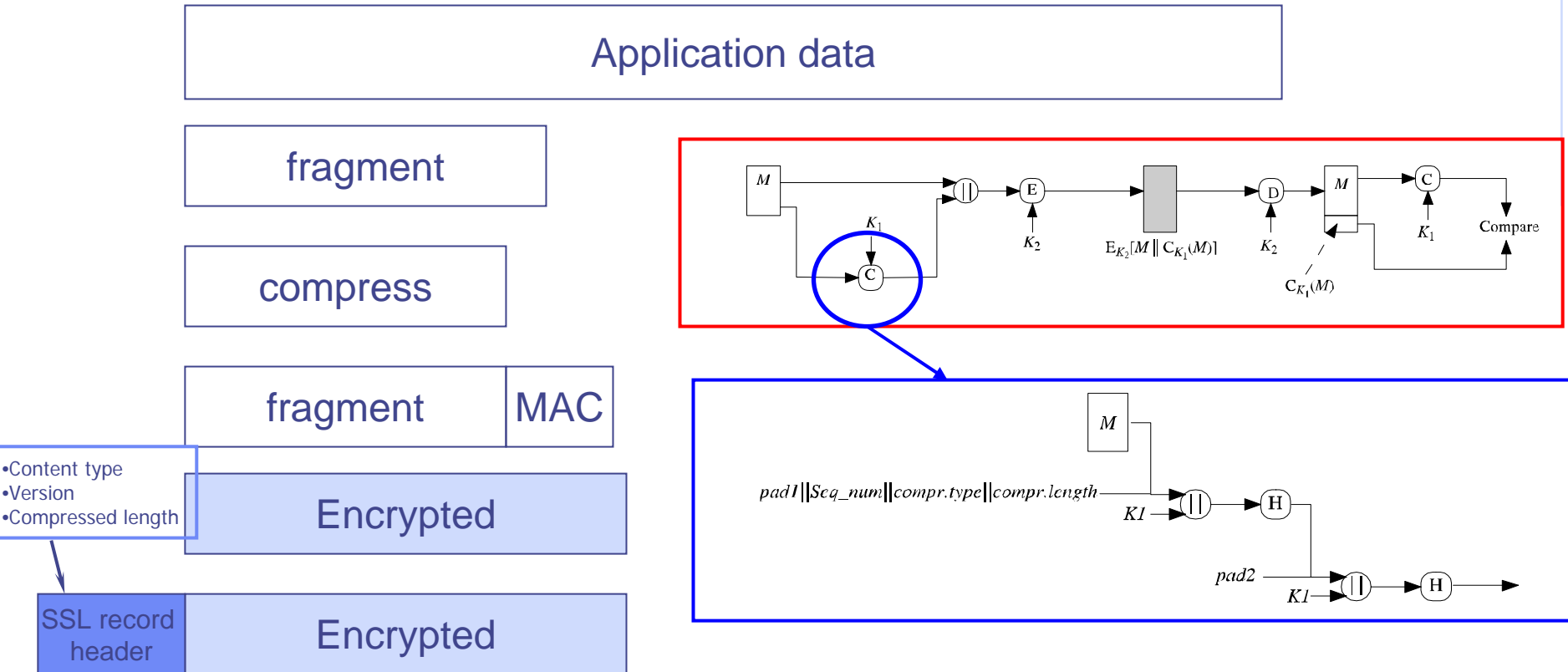
- Connection state
 - ◆ Server and client random
 - ◆ Server write MAC secret
 - The secret key used in MAC send by the server
 - ◆ Client write MAC secret
 - ◆ Server write key
 - Encryption key for data encrypted by the server and decrypted by the client
 - ◆ Client write key
 - ◆ Initialization vectors
 - ◆ Seq number



SSL Record Protocol

Services

- Confidentiality – symmetric encryption
- Message Integrity – MAC



Handshake Protocol

◆ Function

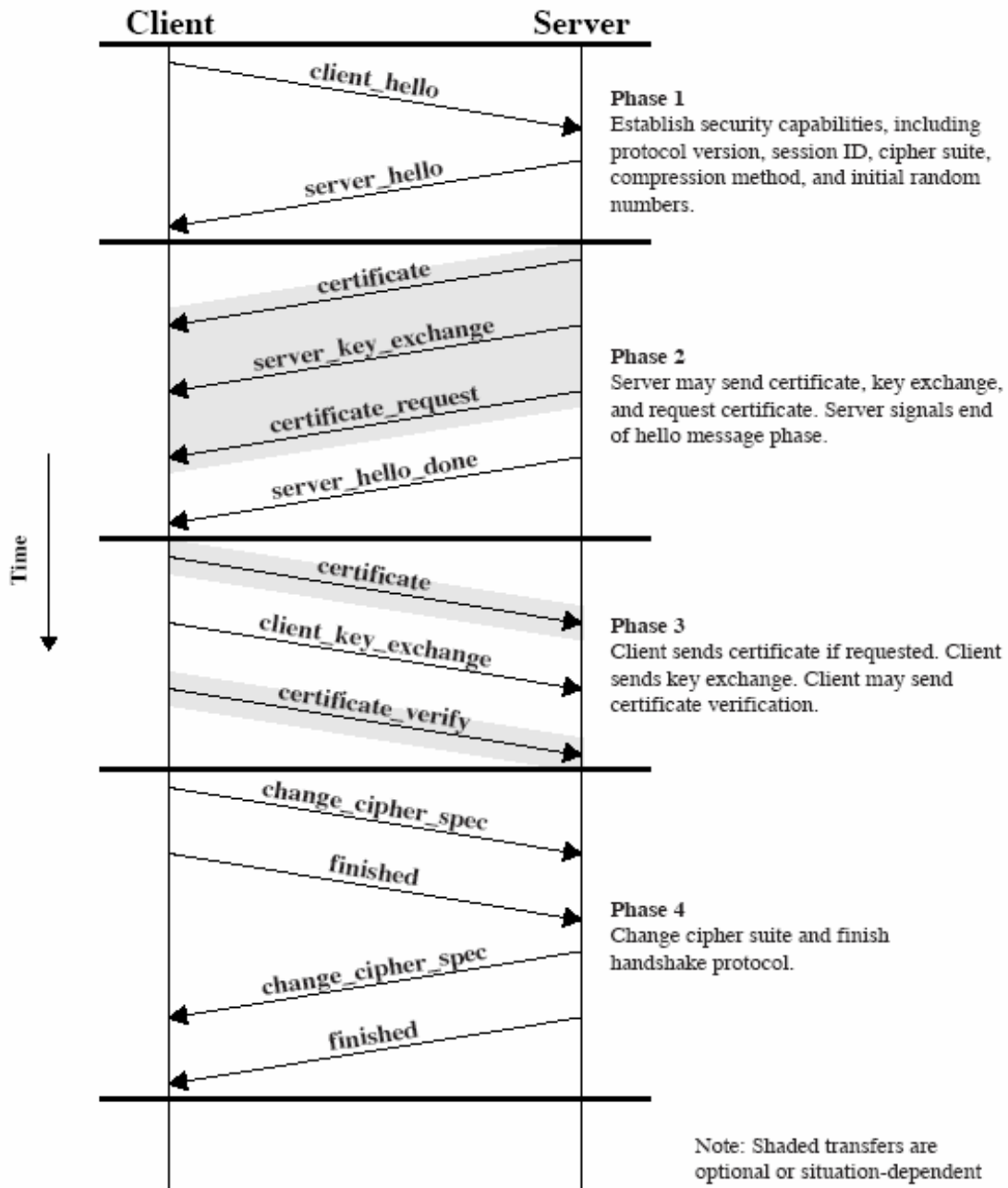
- Server and client authenticate each other
- Negotiate encryption, MAC algorithm, and cryptographic keys

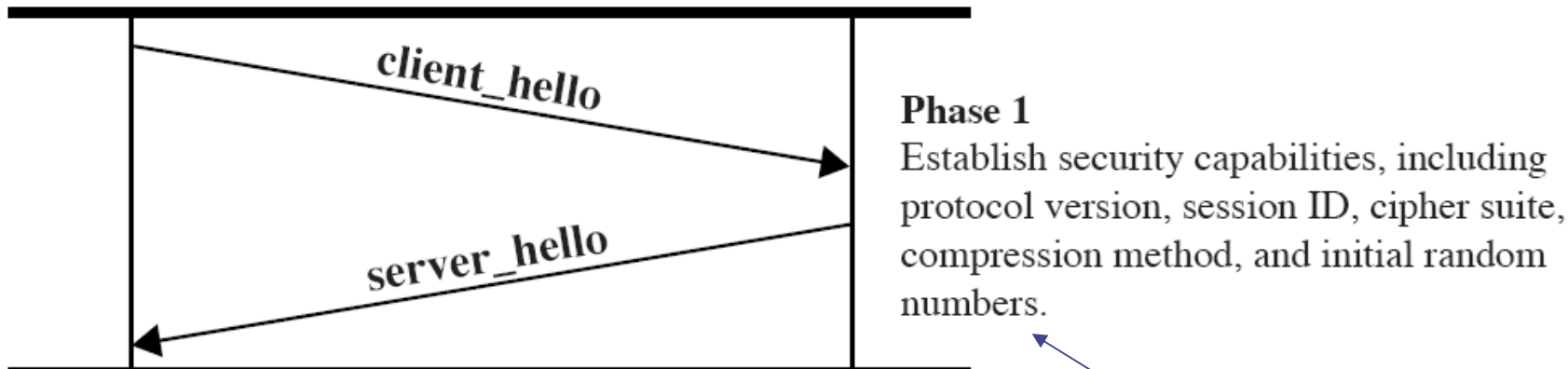
◆ Message format

- Type: one of the 10 messages
 - ◆ Hell_request; client_hello; server_hello;etc..
- Length
- Content: parameters

1 byte	3 bytes	0 bytes
Type	Length	Content







Nonce: Timestamp(32 bit)
+ random number(28 bit)
→ Prevent replay attack

CipherSuite

- Key exchange method

- RSA

- Fixed Diffie-Hellman: based on public parameter in server's CA; fixed secret key

- Ephemeral Diffie-Hellman: one time secret key; most secure D-H options

- Anonymous Diffie-Hellman: no authentication, vulnerable to man-in-the-middle attack

- CipherSpec

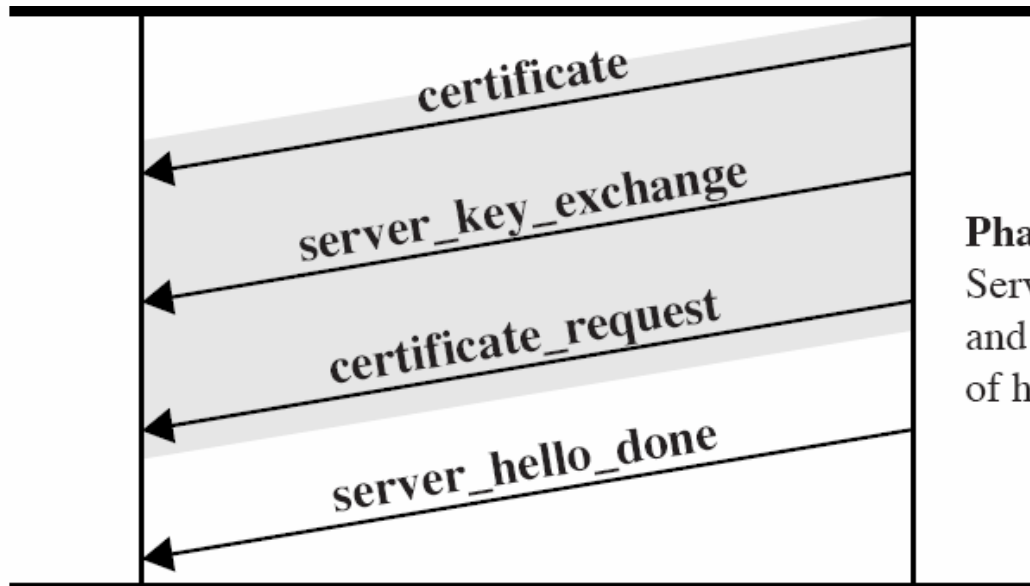
- Cipher Algorithm: RC4; RC2; DES, 3DES, ...

- MAC Algorithm: MD5 or SHA-1

- CipherType: MD5 or SHA-1

- HashSize: IV Size (for CBC mode)...



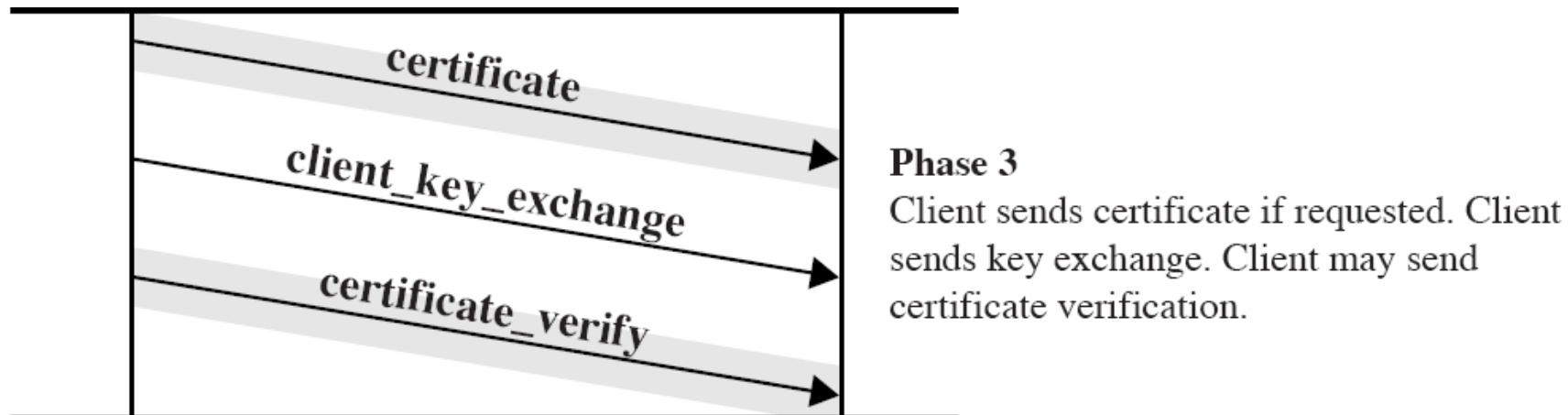


Phase 2

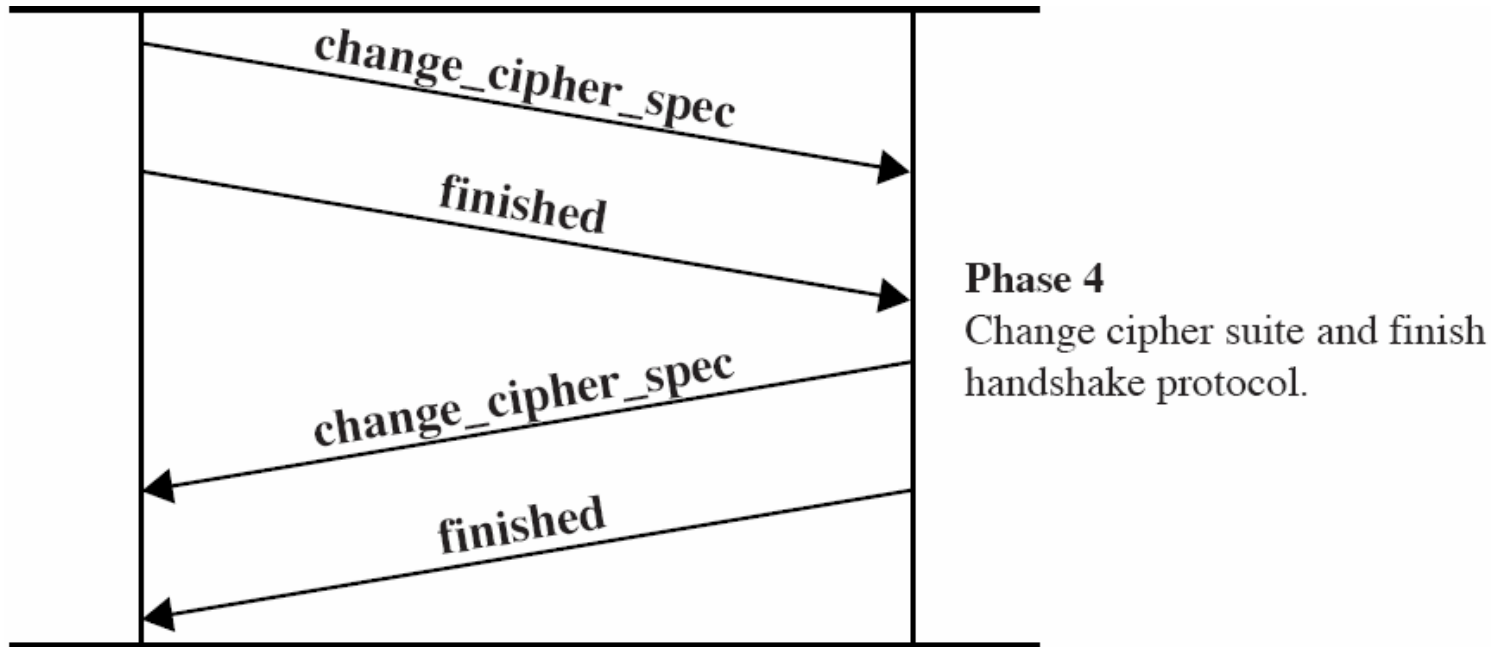
Server may send certificate, key exchange, and request certificate. Server signals end of hello message phase.

◆ Server authentication and Key exchange

- Certificate
 - ◆ Required for all authenticated key change, except anonymous D-H
 - ◆ For Fixed D-H, it contains servers public D-H parameters
- Server_key_exchange_message
 - ◆ Not used when (1) fixed D-H, certificate has parameter; (2) RSA key exchange
 - ◆ Needed: (1) Anonymous D-H; (2) Ephemeral D-H; (3) RSA key exchange, but server only has a signature-only RSA key.
 - ◆ Plus a signature: $\text{hash}(\text{client.random} || \text{server.random} || \text{ServerParameters})$
- Certificate_request
 - ◆ If a non-anonymous server wants to authenticate client



- ◆ Client Authentication and Key exchange
 - Client verifies CA from server
 - Check server_hello parameters
- ◆ Certificate
 - If server requested it
- ◆ Client_key_exchange – depend on the key exchange type
 - RSA: pre-master secret: $S \rightarrow E(KU_{\text{bob}}, S)$
 - Ephemeral or anonymous D-H: client's public D-H parameters
 - Fixed D-H: null, parameters are in certificate
- ◆ Certificate_verify
 - Explicit verification of a client certificate; only sent following any client certificate that has signing capability



- ◆ Change_cipher_spec
- ◆ Finished – verifies key exchange and authentication are successful
 - The content of the finished message is the concatenation of two hash values
 - ◆ MD5(master_secret||pad2||MD5(handshake_msg||sender||master_secret||pad1))
 - ◆ SHA1(master_secret||pad2||SHA1(handshake_msg||sender||master_secret||pad1))
- ◆ Master Secret Creation
 - Master_secret =

$$\text{MD5}(\text{pre_master_secret} \parallel \text{SHA}(\text{'A'} \parallel \text{pre_master_secret} \parallel \text{client.random} \parallel \text{server.random})) \parallel$$

$$\text{MD5}(\text{pre_master_secret} \parallel \text{SHA}(\text{'BB'} \parallel \text{pre_master_secret} \parallel \text{client.random} \parallel \text{server.random})) \parallel \text{MD5}(\text{pre_master_secret} \parallel \text{SHA}(\text{'CCC'} \parallel \text{pre_master_secret} \parallel \text{client.random} \parallel \text{server.random}))$$
- ◆ Generation of session keys (e.g., client write MAC secret ...)

Comparison

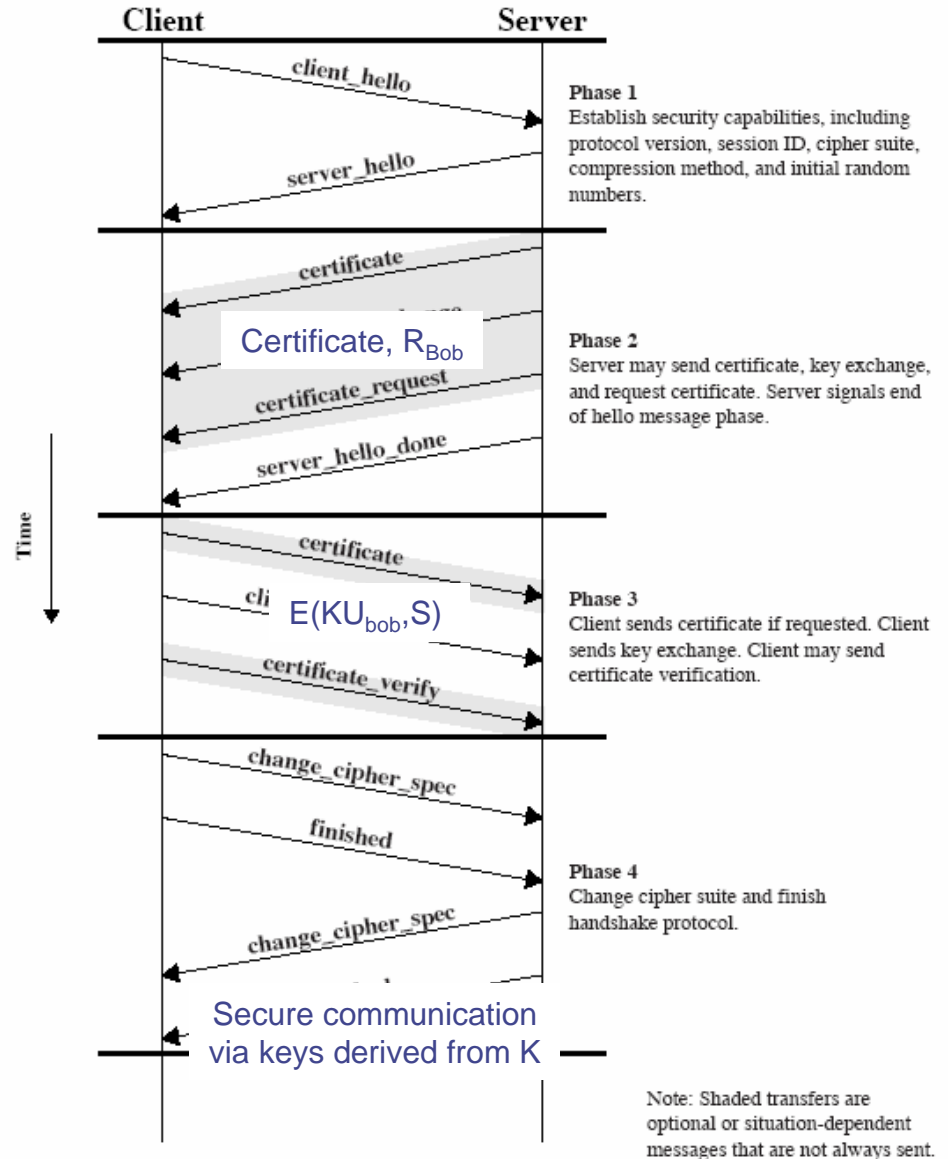
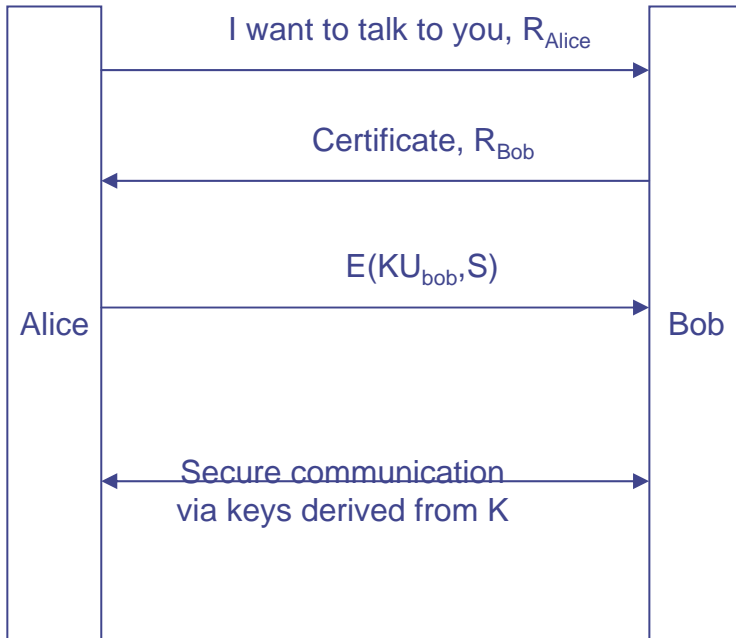


Figure 17.6 Handshake Protocol Action

Other two protocols

◆ Change Cipher Spec Protocol

- Use SSL record protocol
- Update the cipher suite to be used on this connection

◆ Alert Protocol

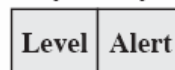
- Control and management protocol

1 byte



(a) Change Cipher Spec Protocol

1 byte 1 byte



(b) Alert Protocol



SSL vs. TLS

◆ A story

- Netscape originated SSL v2 in Navigator 1.1 in 1995
- SSL v3 was published as an Internet draft
- IETF formed a TLS working group
- First published version of TLS is essentially an SSL v3.1, and is backward compatible with SSL v3
- SSL v3 is most commonly deployed
- TLS mandated the use of DSS instead of RSA



Other Topics

◆ HTTPS

- Combination of a normal [HTTP](#) interaction over an [Secure Sockets Layer](#) (SSL) or [Transport Layer Security](#) (TLS) mechanism.
- <http://en.wikipedia.org/wiki/Https>

◆ Cookies

- http://en.wikipedia.org/wiki/HTTP_cookie

◆ SET (Secure Electronic Transaction)

- [WS] Chapter 17

