Securing Wireless Networks
PGP Secure Socket Layer (SSL) and TLS

Never Stand Still

Professor Sanjay K. Jha

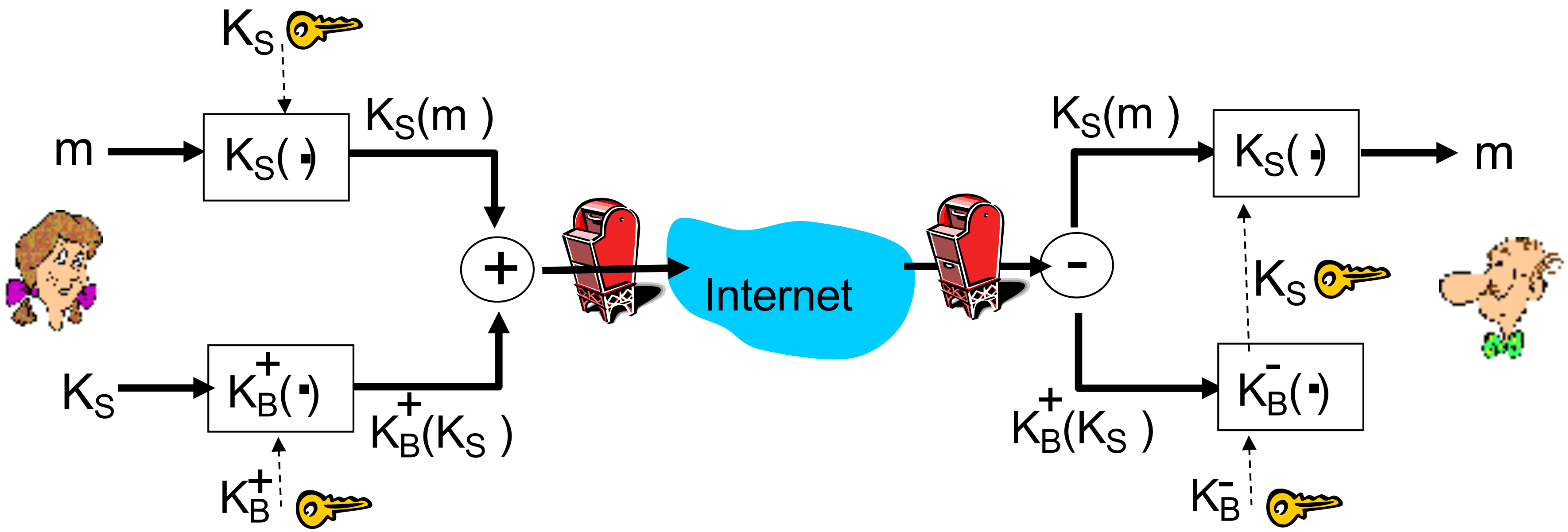
School of Computer Science and Engineering, UNSW

Security at Internet Protocol Layers

- Security can be implemented at various layers
- Application Layer Security
  - *A quick glimpse at secure email (PGP)*
- Secure Socket Layer (SSL) / Transport Layer Security (TLS)
  - this week
- Network Layer Security
  - IPSec next week
- Link Layer Security
  - WLAN covered earlier, more on Enterprise later

Secure e-mail: Alice

- Alice wants to send confidential e-mail, m, to Bob.

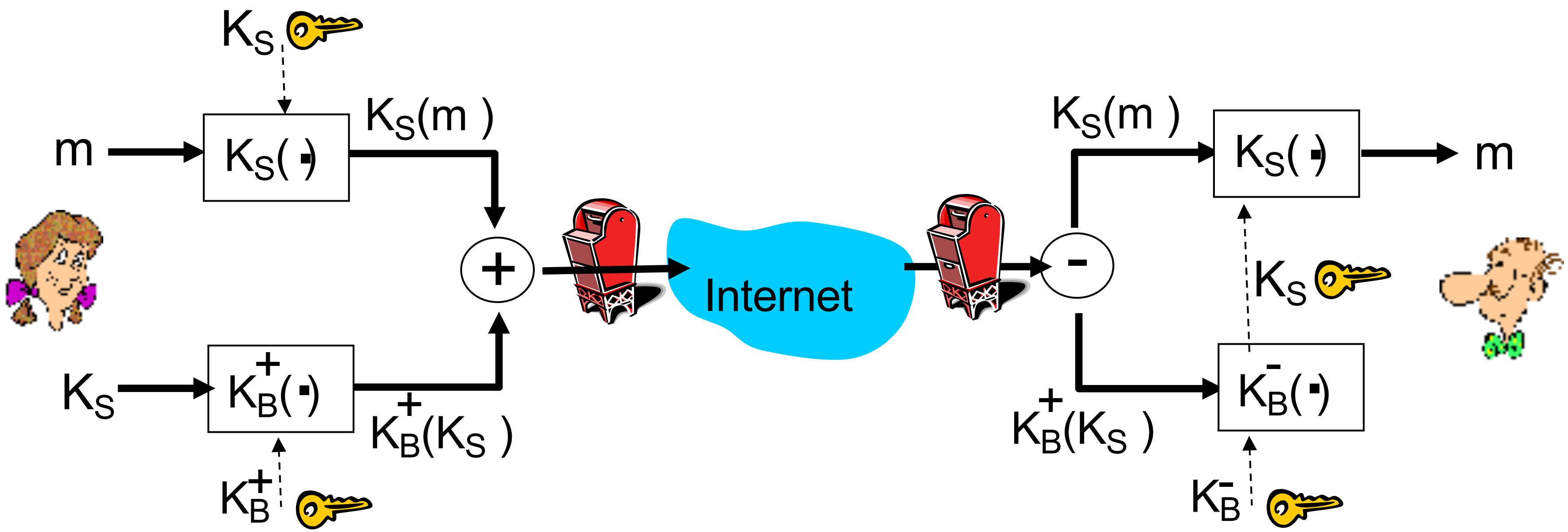


Alice:

- generates random *symmetric* private key, K_S
- encrypts message with K_S (for efficiency)
- also encrypts K_S with Bob's public key
- sends both $K_S(m)$ and $K_B(K_S)$ to Bob

Secure e-mail: Bob

- Alice wants to send confidential e-mail, m, to Bob.

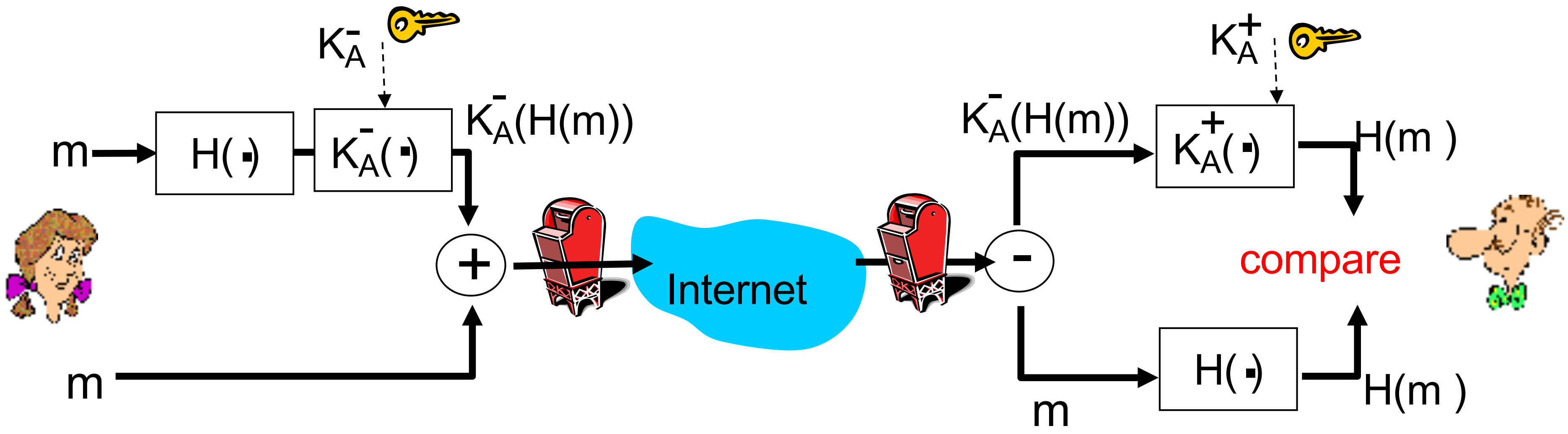


Bob:

- uses his private key to decrypt and recover K_S
- uses K_S to decrypt $K_S(m)$ to recover m

Secure e-mail (continued)

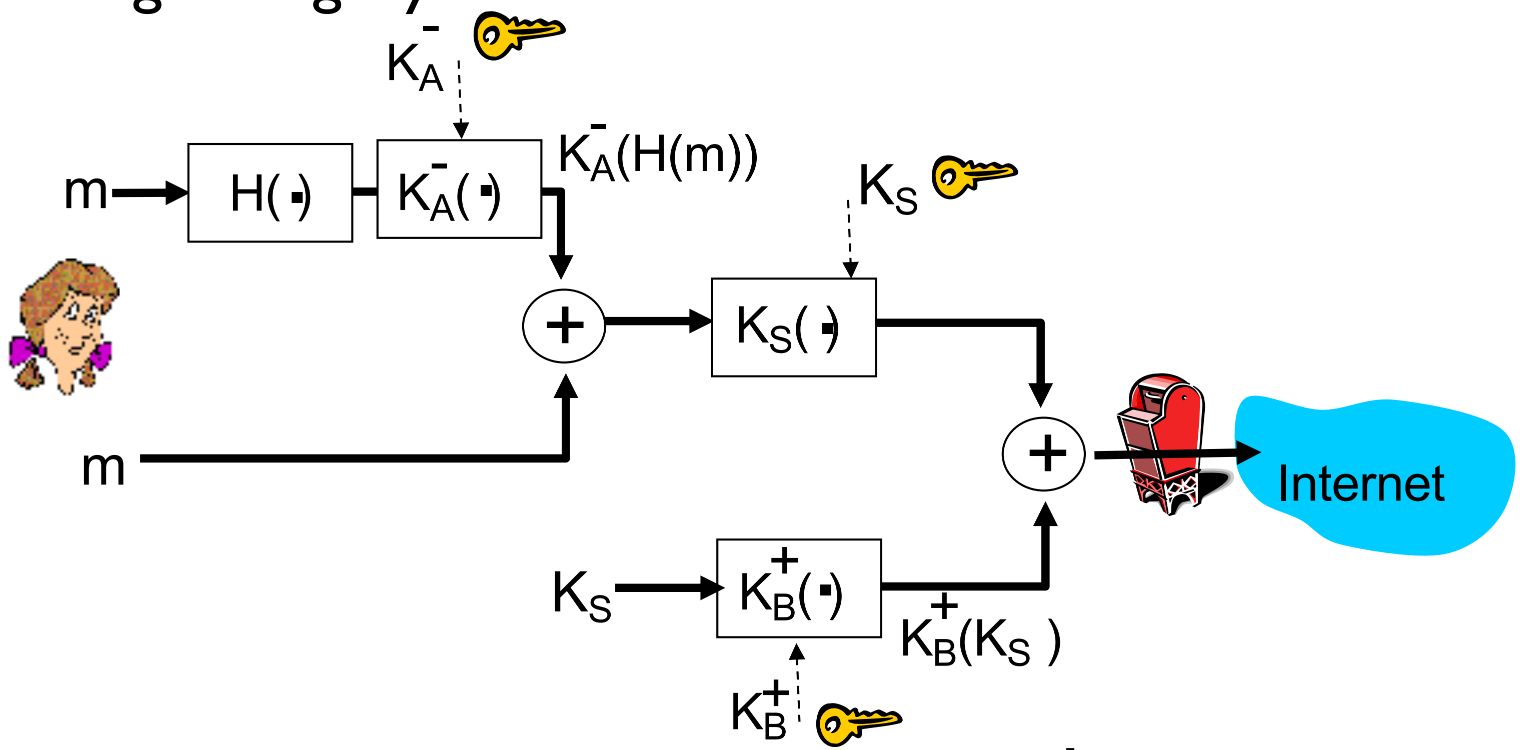
- Alice wants to provide sender authentication message integrity



- Alice digitally signs message
- sends both message (in the clear) and digital signature

Secure e-mail (continued)

- Alice wants to provide secrecy, sender authentication, message integrity.



Alice uses three keys: her private key, Bob's public key, newly created symmetric key (Exercise: see how Bob's side works)

DNS Attacks

- Domain Name Service: translation between a host name and the corresponding IP address, maintained by a hierarchy of name servers.
- DoS Attack: A number of Denial of Service (DOS) attacks in recent years
- MiTM: attacker can impersonate a DNS server
  - return a bogus address , divert traffic to a malicious server, allows into collect user passwords or other credentials.
- Cache poisoning attack aka DNS spoofing:
  - attackers to plant bogus addresses, thus diverting a user request to malicious servers.

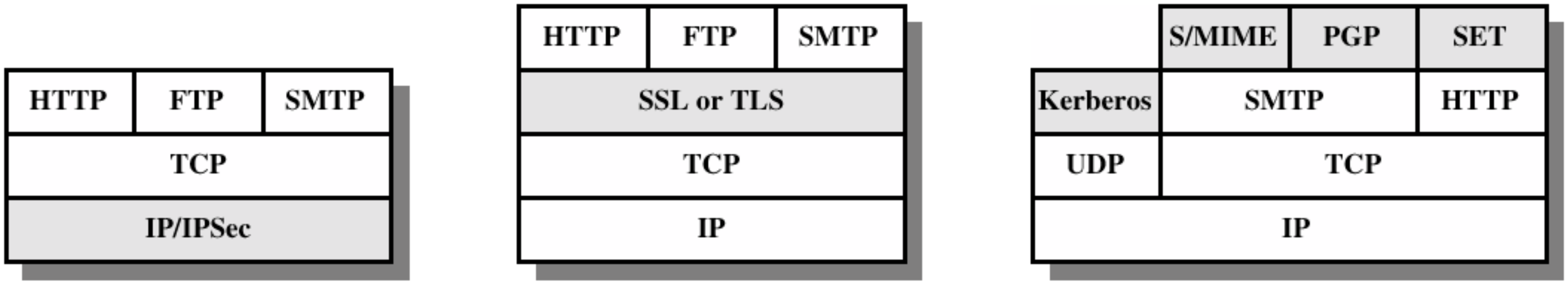
DNS Security

- IETF DNS Security Extensions (DNSSEC)
- Approach similar to PGP
  - response signed by the private key of a DNS server.
  - a requester can verify using the corresponding public key.
  - a digital signature also provides the integrity of the response data, Confidentiality not an issue for DNS records
- Recent study : only 1% of domains use the DNSSEC
- Very few registrars support DNSSEC .
- Mechanism for communicating DNSSEC information has several security vulnerabilities.
- DDoS defence in not part of DNSSEC
  - Intrusion Detection/Prevention System for DOS (later week)

SSL: Secure Sockets Layer

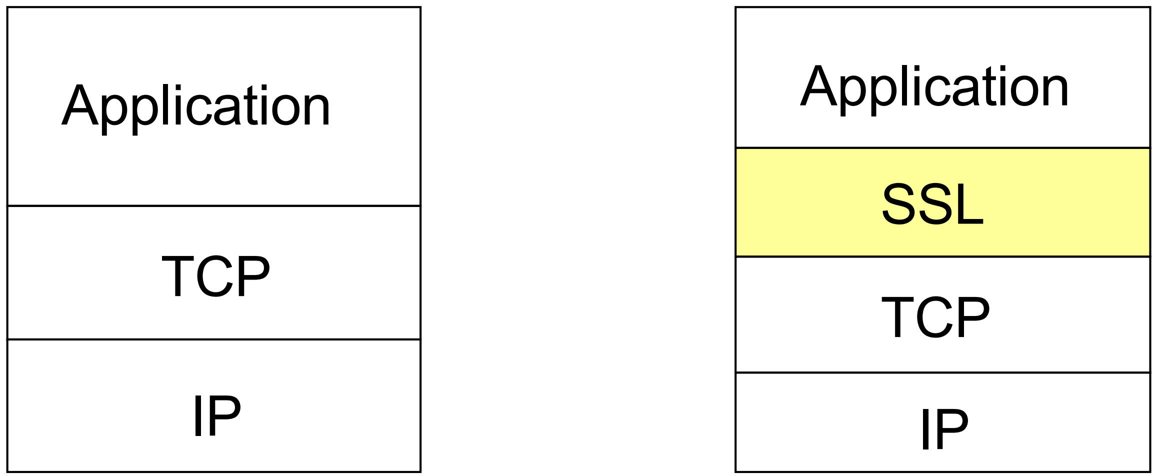
- widely deployed security protocol
  - supported by almost all browsers, web servers
  - https
  - billions $/year over SSL
- mechanisms: [Woo 1994], implementation: Netscape
- variation -TLS: transport layer security, RFC 2246
- provides
  - *Confidentiality, integrity, authentication*
- original goals:
  - Web e-commerce transactions
  - encryption (especially credit-card numbers)
  - Web-server authentication
  - optional client authentication
  - minimum hassle in doing business with new merchant
- available to all TCP applications
  - secure socket interface

SSL in TCP/IP protocol stack



(a) Network Level (b) Transport Level (c) Application Level

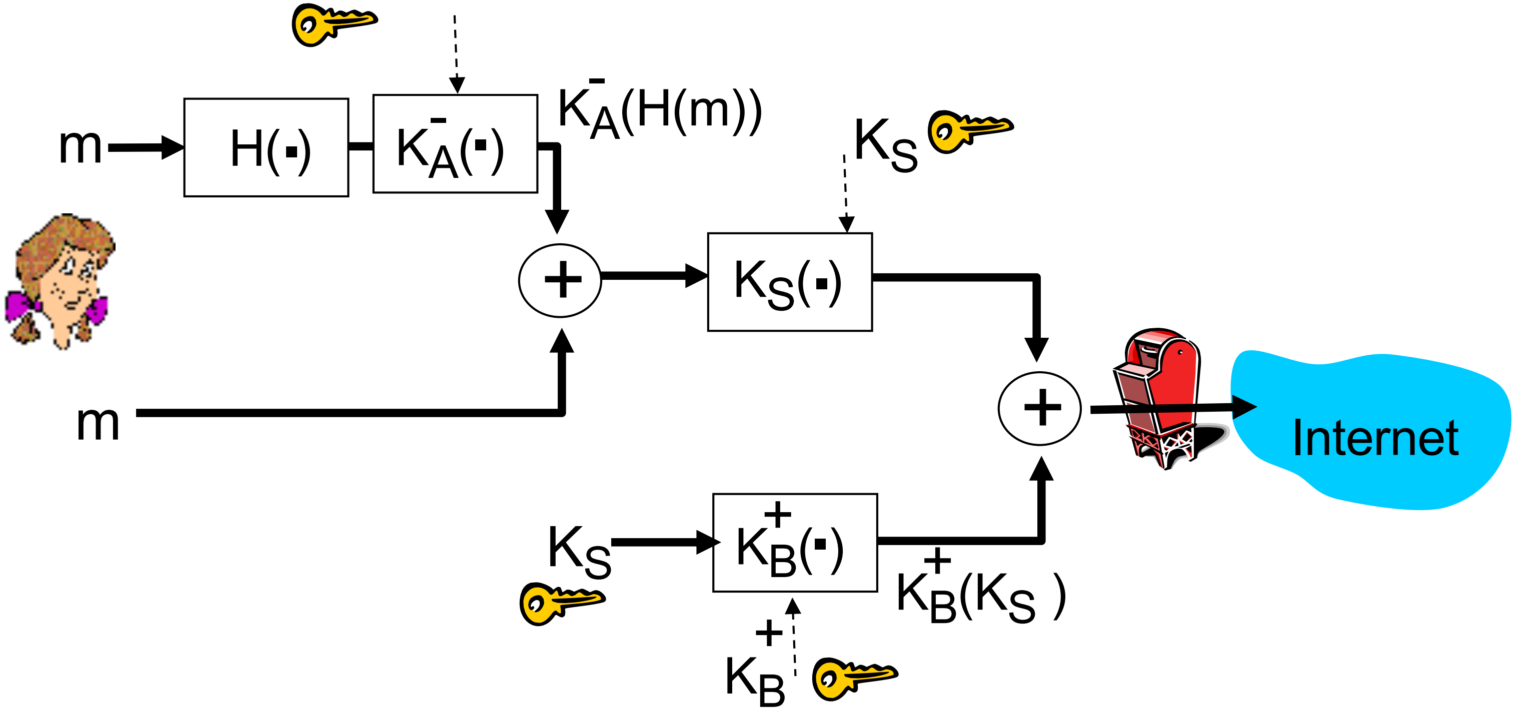
SSL and TCP/IP



normal application　　*application with SSL*

- SSL provides application programming interface (API) to applications
- C and Java SSL libraries/classes readily available

Could do something like PGP

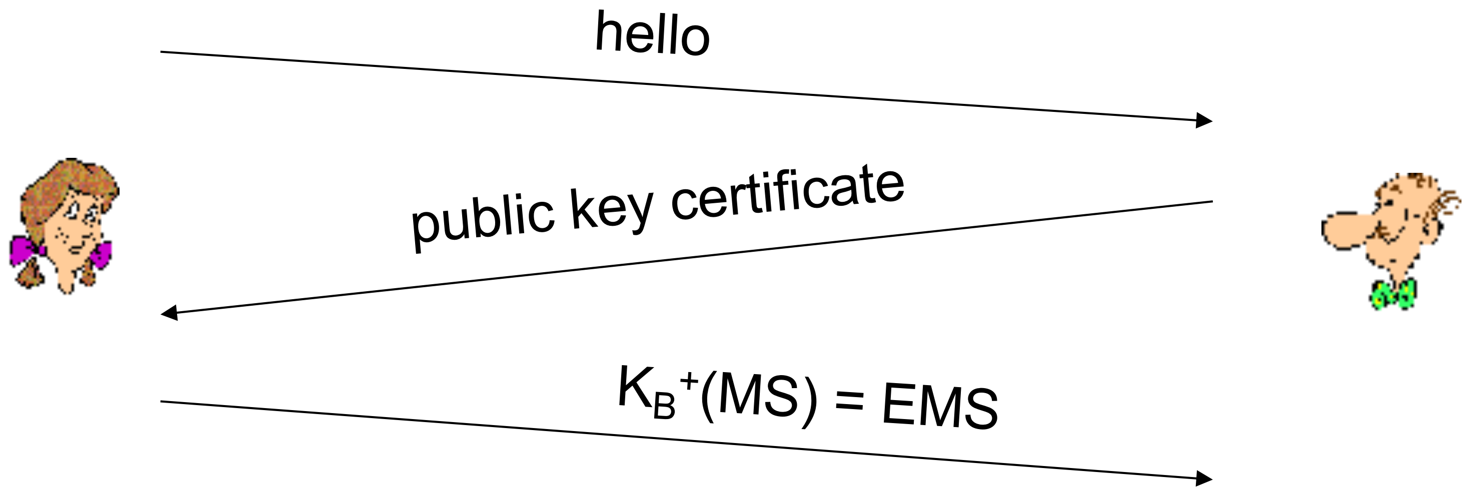


- but want to send byte streams & interactive data
- want set of secret keys for entire connection
- want certificate exchange as part of protocol: handshake phase

Toy SSL: a simple secure channel

- *handshake:* Alice and Bob use their certificates, private keys to authenticate each other and exchange shared secret
- *key derivation:* Alice and Bob use shared secret to derive set of keys
- *data transfer:* data to be transferred is broken up into series of records
- *connection closure:* special messages to securely close connection

Toy: a simple handshake



MS: master secret

EMS: encrypted master secret

Toy: key derivation

- considered bad to use same key for more than one cryptographic operation
  – use different keys for message authentication code (MAC) and encryption
- four keys:
  – K_c = encryption key for data sent from client to server
  – M_c = MAC key for data sent from client to server
  – K_s = encryption key for data sent from server to client
  – M_s = MAC key for data sent from server to client
- keys derived from key derivation function (KDF)
  – takes master secret and (possibly) some additional random data and creates the keys

Toy: data records

- why not encrypt data in constant stream as we write it to TCP?
  - where would we put the MAC? If at end of TCP connection, no message integrity until all data processed.
  - e.g., with instant messaging, how can we do integrity check over all bytes sent before displaying?
- instead, break stream in series of records
  - each record carries a MAC
  - receiver can act on each record as it arrives
- issue: in record, receiver needs to distinguish MAC from data
  - want to use variable-length records

Toy: sequence numbers

- *problem:* attacker can capture and replay record or re-order records
- *solution:* put sequence number into MAC:
  - MAC = MAC(M_x, sequence||data)
  - *note: no sequence number field*

- *problem:* attacker could replay all records in future
- *solution:* use nonce

Toy: control information

- *problem:* truncation attack:
  - attacker forges TCP connection close segment
  - one or both sides thinks there is less data than there actually is.
- *solution:* record types, with one type for closure
  - type 0 for data; type 1 for closure
- MAC = MAC(M_x, sequence||type||data)

length	type	data	MAC

SSL Architecture

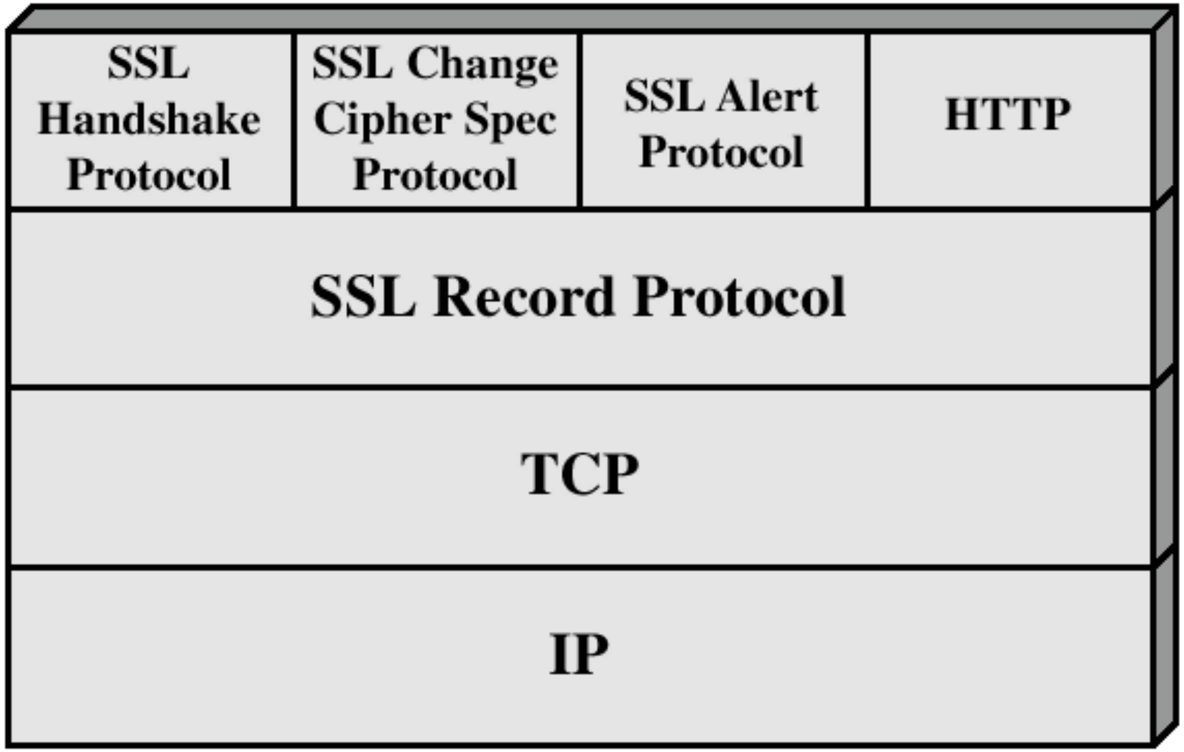


Figure 7.2 SSL Protocol Stack

Real SSL: handshake (1)

Purpose

1. server authentication
2. negotiation: agree on crypto algorithms
3. establish keys
4. client authentication (optional)

Real SSL: handshake (2)

1. client sends list of algorithms it supports, along with client nonce
2. server chooses algorithms from list; sends back: choice + certificate + server nonce
3. client verifies certificate, extracts server's public key, generates pre_master_secret, encrypts with server's public key, sends to server
4. client and server independently compute encryption and MAC keys from pre_master_secret and nonces
5. client sends a MAC of all the handshake messages
6. server sends a MAC of all the handshake messages

Real SSL: handshaking (3)

Why last two messages with MAC exchanged?

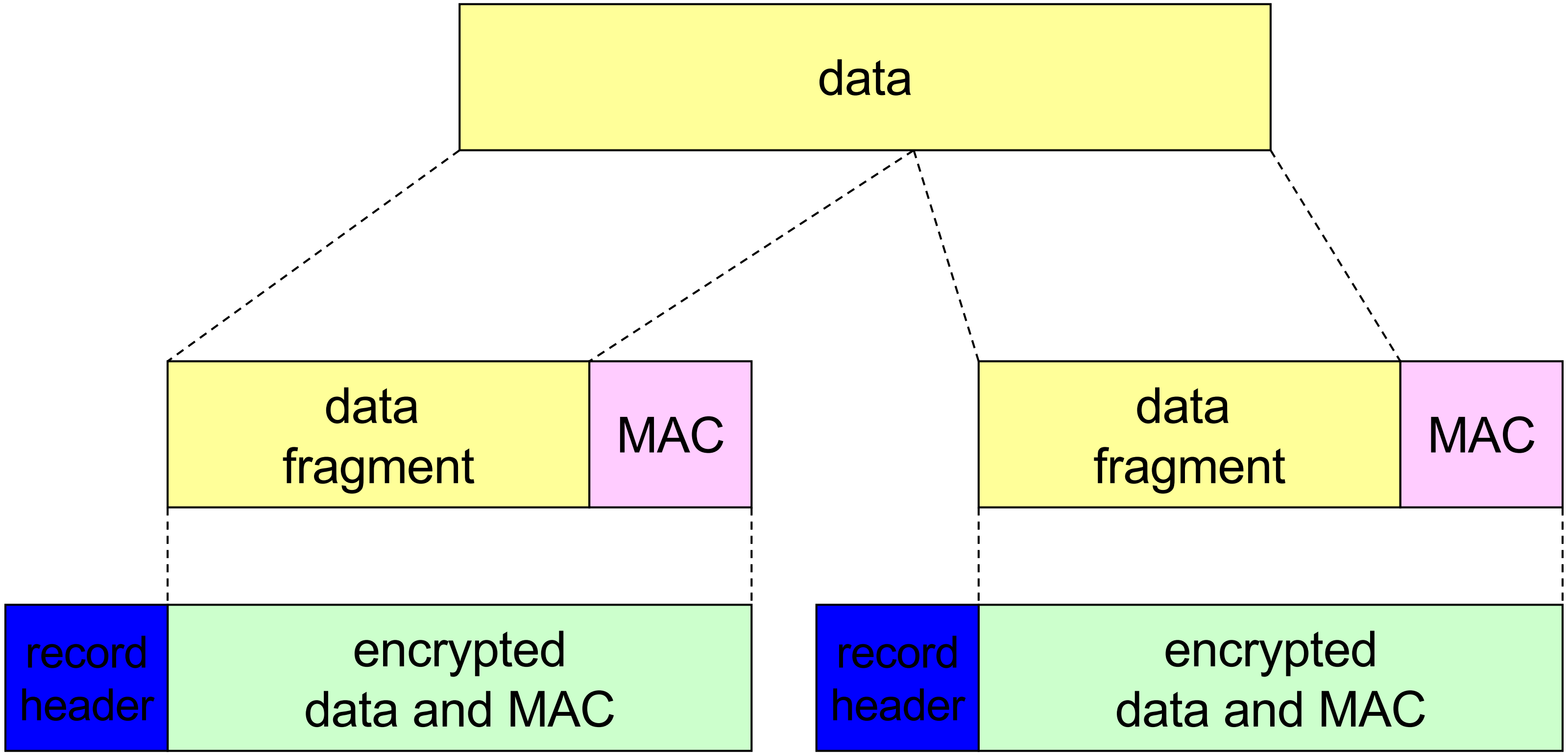
- client typically offers range of algorithms, some strong, some weak
- man-in-the middle could delete stronger algorithms from list
- last 2 steps prevent this
  - last two messages are encrypted

Real SSL: handshaking (4)

Why two random nonces?

- suppose Trudy sniffs all messages between Alice & Bob
- next day, Trudy sets up TCP connection with Bob, sends exact same sequence of records
  - Bob (Amazon) thinks Alice made two separate orders for the same thing
  - solution: Bob sends different random nonce for each connection. This causes encryption keys to be different on the two days
  - Trudy's messages will fail Bob's integrity check

SSL record protocol

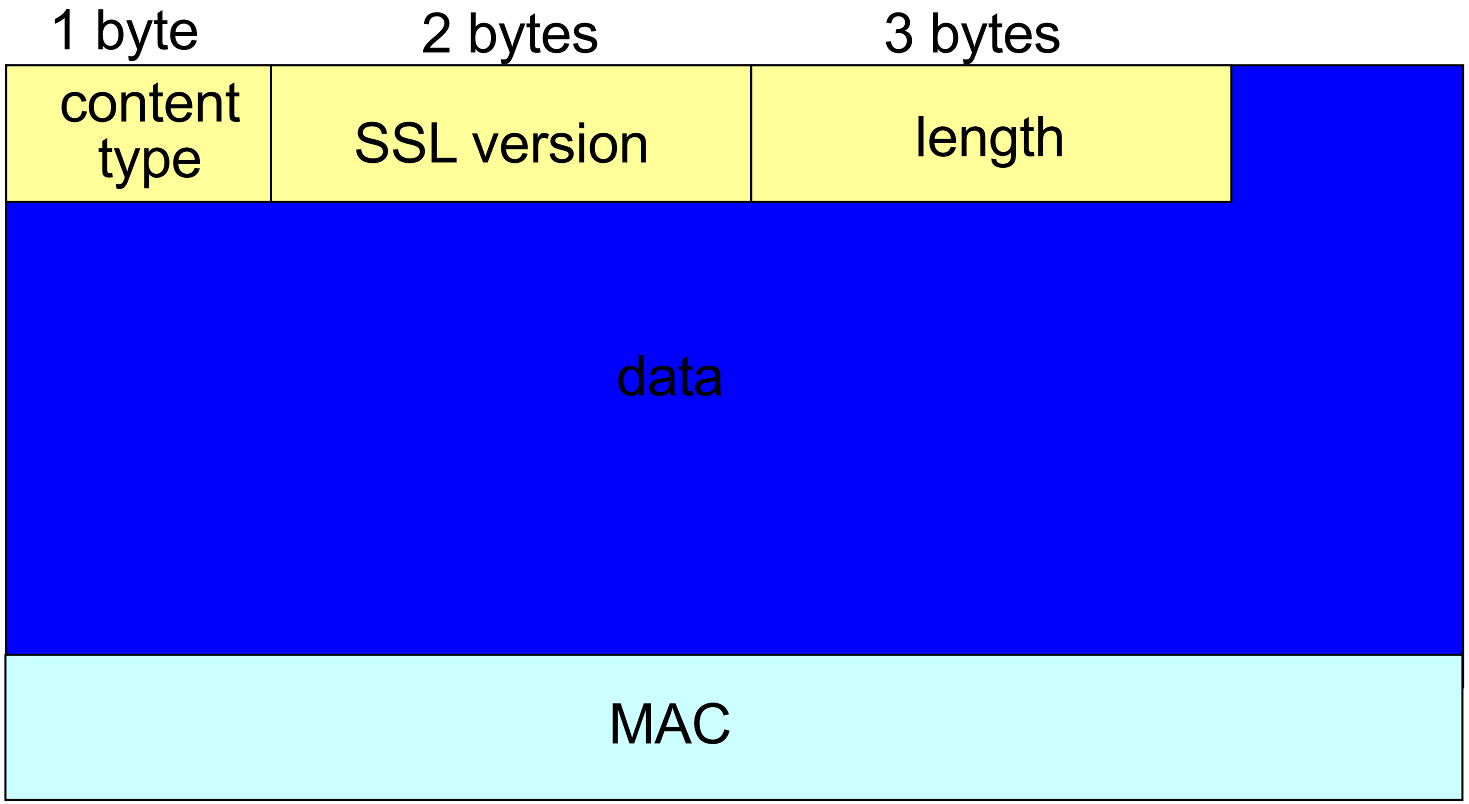


record header: content type; version; length

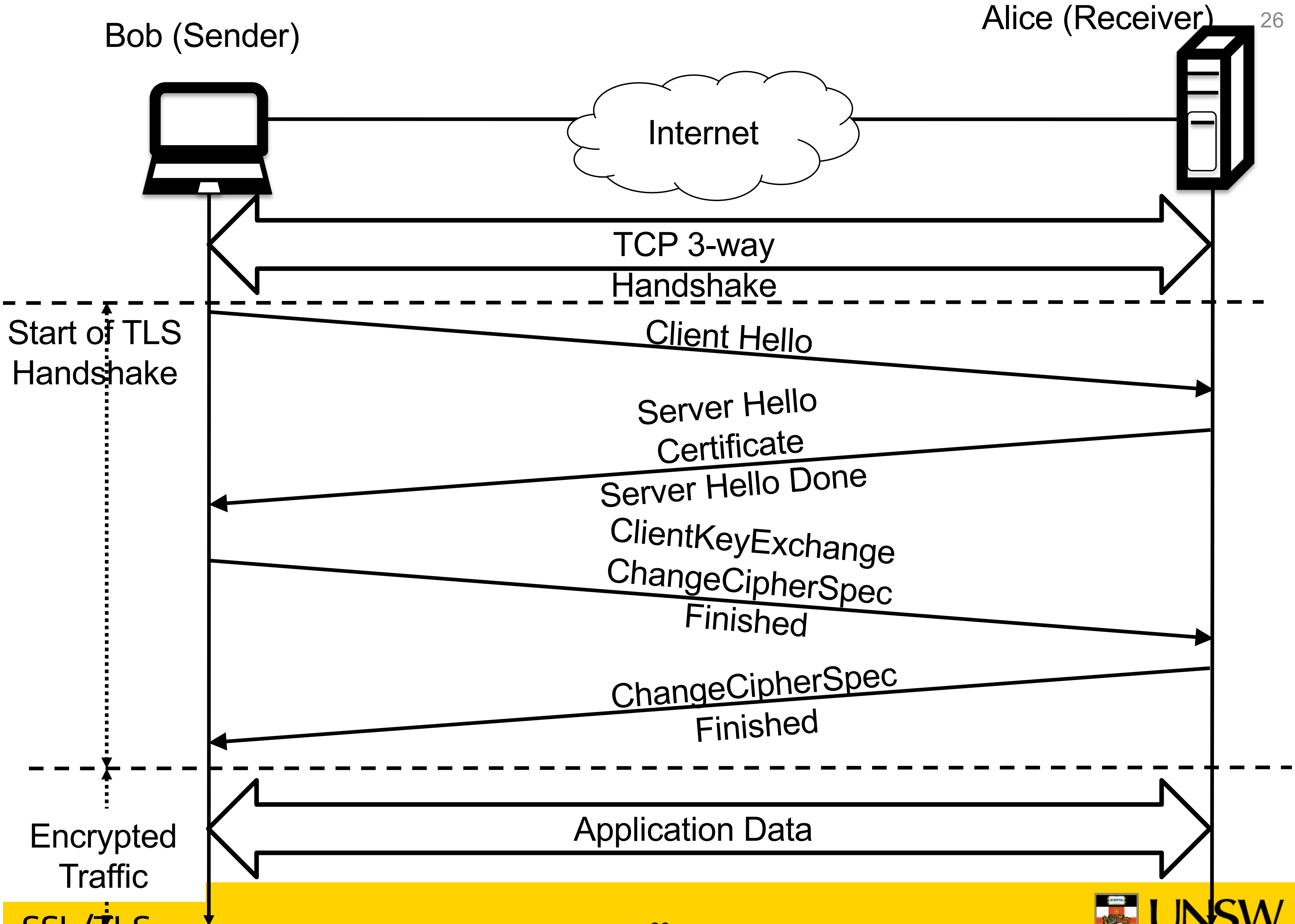
MAC: includes sequence number, MAC key M_x

fragment: each SSL fragment 2^{14} bytes (~16 Kbytes)

SSL record format



data and MAC encrypted (symmetric algorithm)


Bob (Sender)
Alice (Receiver)
Internet
TCP 3-way Handshake
Start of TLS Handshake
Client Hello
Server Hello
Certificate
Server Hello Done
ClientKeyExchange
ChangeCipherSpec
Finished
ChangeCipherSpec
Finished
Application Data
Encrypted Traffic

Handshake

- First Bob and Alice exchange the three-way TCP SYN, SYNACK and ACK messages.
- Bob then sends a ClientHello message to Alice along with the cipher suites (ciphers and thehash functions it supports) and a nonce, a large, random number, chosen specifically for this run of the protocol.
- Alice responds with a ServerHello message along with her choice from the cipher suites (e.g.,AES for confidentiality, RSA for the public key, SHA2 for the Message Authentication Code (MAC)), a certificate containing her public key and a nonce. Additionally, she could also request the client's certificate and parameters for other TLS extensions.

Handshake (Contd)

- Bob checks validity of the certificate and is assured that it belongs to Alice. He initiates theClientKeyExchange message. This can use a range of key exchange methods, e.g., RSA orthe Diffie-Hellman (and variants) to establish a symmetric key for the ensuing session.
  - For example, when using RSA, Bob could generate a 48-bit Pre-master secret (PMS) and encryptit with Alice's public key obtained using the steps as described above and send it to Alice.
- Bob sends a ClientCiptherSpec and a Finished Message suggesting that the key generation and authentication are complete.
- Alice also has the shared key at this point. She responds with a ChangeCipherSpec and a Finished Message back to Bob.
- Bob decrypts the message with the negotiated symmetric key and performs a message integrity check

Key derivation

- client nonce, server nonce, and pre-master secret input into pseudo random-number generator.
  - produces master secret
- master secret and new nonces input into another random-number generator: "key block"

- key block sliced and diced:
  - client MAC key
  - server MAC key
  - client encryption key
  - server encryption key
  - client initialization vector (IV)
  - server initialization vector (IV)

Transport Layer Security

- The same record format as the SSL record format.
- Defined in RFC 2246.
- Similar to SSLv3.
- Differences in the:
  - version number
  - message authentication code
  - pseudorandom function
  - alert codes
  - cipher suites
  - client certificate types
  - certificate_verify and finished message
  - cryptographic computations
  - padding

Acknowledgements

- Computer Networking A top-Down Approach: Jim Kurose and Keith Ross, Chapter 8, (foils provided by Authors)
  - Reference section 8.5
- Network Security Essentials: Stallings, 6, Foils provided by Henric Johnson, Blekinge Institute of Technology, Sweden
  - Reference Section 6.2 for SSL
  - Optional read Section 6.3 for TLS
- See Cybok NETWORK SECURITYKNOWLEDGE AREA for references and research papers.