CS 556 – Computer Security
Spring 2018

Dr. Indrajit Ray
Email: indrajit.ray@colostate.edu

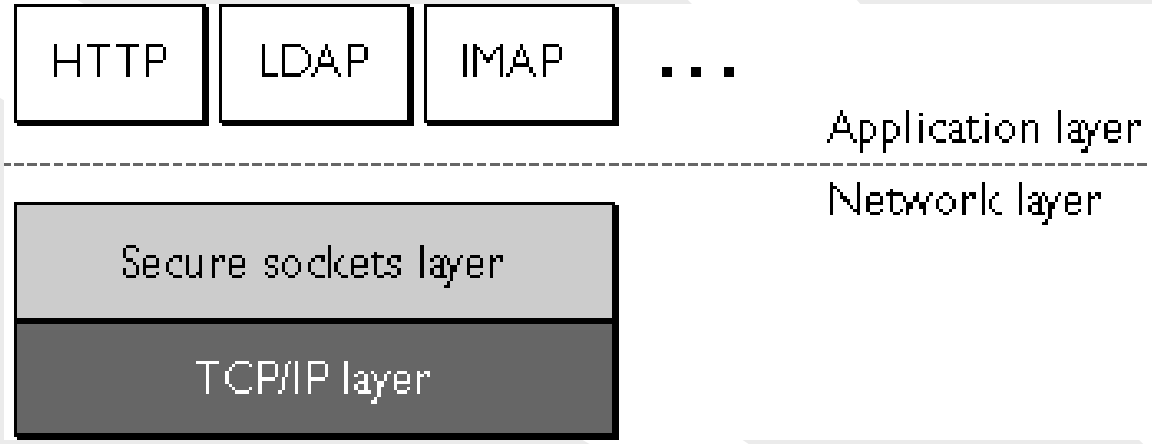
Department of Computer Science
Colorado State University
Fort Collins, CO 80523, USA

Security Protocols

Secure Sockets Layer Protocol

- Designed to establish a secure connection between a client and a server communicating over an insecure channel
- Attacker assumptions
  - Have substantial computational resources
  - Can capture, modify, delete, replay and otherwise tamper with messages
  - Cannot obtain secret information from sources outside the protocol

Placement in the IP Stack



SSL runs above TCP/IP and below high level application programs.

SSL Services

- Security parameter negotiation
- Peer entity authentication
- Data confidentiality
- Data authentication and integrity
- Compression / decompression

SSL Architecture

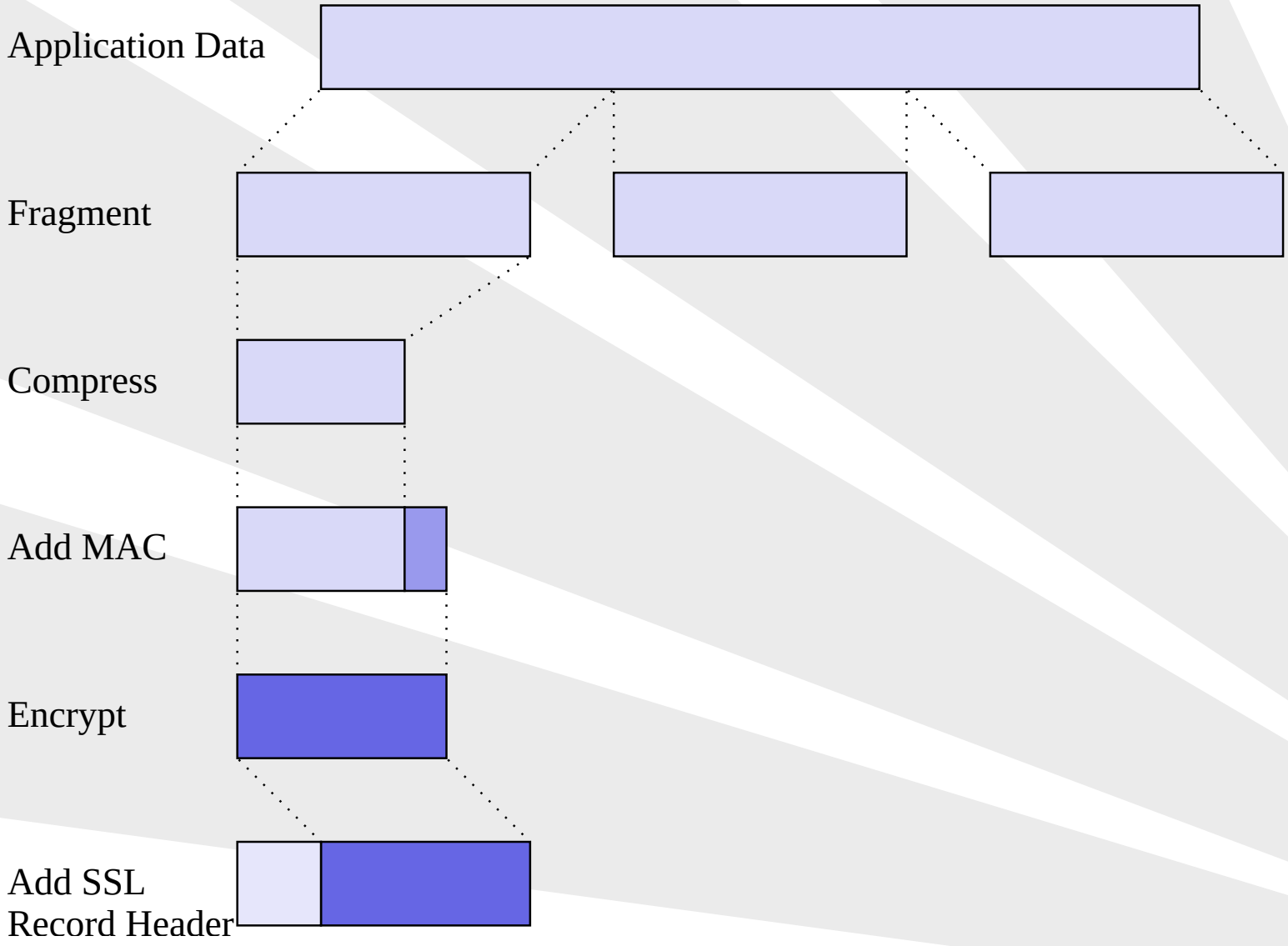
<table>
<tr><td>SSL Handshake Protocol</td><td>SSL Change Cipher Spec Protocol</td><td>SSL Alert Protocol</td><td>HTTP</td><td>Other Application Protocols</td></tr>
<tr><td rowspan="3"></td><td colspan="4">SSL Record Protocol</td></tr>
<tr><td colspan="4">TCP</td></tr>
<tr><td colspan="5">IP</td></tr>
</table>

SSL Record Protocol

Record Protocol Services

- Provides the secure tunnel for communication between client and server
- Five steps by sender to transmit packet
  - Fragmentation of message into packets
  - Compression of packets into smaller packets
  - Generating and appending Message Authentication Code to each packet
  - Encryption of packet (with MAC)
  - Add SSL record header and transmit
- Receiver perform the corresponding steps in reverse order

Record Protocol Overview



SSL Session and Connection

SSL Session

- An association between a client and a server
- Generated by the SSL Handshake Protocol
- Define a set of cryptographic security parameter that can be shared among multiple connections
  - Used to avoid expensive negotiation of new security parameters for each connection

Session State Elements

session identifier An arbitrary byte sequence chosen by the server to identify an active or resumable session

peer certificate X.509v3 certificate of the peer. This element of the state may be null

compression method The algorithm to compress the data prior to encryption

Session State Elements

cipher spec Specifies the bulk encryption algorithm (such as null, DES etc.) and a MAC algorithm (such as MD5 or SHA). It also defines cryptographic attributes such as the `hash_size`

master secret 48-byte secret shared between the client and the server

is resumable A flag indicating whether the session can be used to initiate new connections

SSL Connection

- Connections are peer-to-peer relationships
- Are transient
- Every connection associated with one session
- Sessions can be reused across multiple secure connections
- Handshake protocol
  - establishes new session and connection together
  - may use existing session for new connection

Connection State Elements

server and client random Byte sequences that are chosen by the server and client for each connection

server write MAC secret The secret used in MAC operations on data written by the server

client write MAC secret The secret used in MAC operations on data written by client

server write key The bulk cipher key for data encrypted by the server and decrypted by the client

Connection State Elements

client write key The bulk cipher key for data encrypted by the client and decrypted by the server

initialization vectors When a block cipher in CBC mode is used, an initialization vector is maintained for each key. This field is first initialized by the SSL handshake protocol. Thereafter the final ciphertext block from each record is preserved for use with the following block

sequence numbers Each party maintain s separate sequence numbers for transmitted and received messages for each connection. When a party sends or receives a `change cipher spec` message, the appropriate sequence number is set to zero

SSL Handshake Protocol

No Session Identifier, No Client Authentication

Message Type	Direction	Data transferred
Client-hello	C → S	version, challenge-data, cipher-specs
Server-hello	S → C	connection-id, server-certificate, version, modified-cipher-specs
Client-master-key	C → S	cipher-kind, clear-master-key, {secret-master-key}server-public-key
Client-finish	C → S	{connection-id}client-write-key
Server-verify	S → C	{challenge-data}server-write-key
Server-finish	S → C	{session-id}server-write-key

Client-hello Message Contents

- Version - highest SSL version that client can support
- Challenge data
  - 4 byte timestamp, 28 byte random value
  - It is used to authenticate the server at a later stage
- A list of ciphers and compression techniques that the client can support

Server-hello Message Contents

- Connection-id
- Server certificate
  - Used by the client to obtain the server's public key and verify the identity of the server using some certification authority
- Version - version that server agrees to support
- Modified list of ciphers and compression that both the server and the client can support

Client-master-key Message Contents

- Final choice of the cipher (cipher-kind)
- Master-key in two parts
  - clear-master-key: if chosen cipher is export grade, then 88 bits of master key is sent in clear
  - secret-master-key: remaining 40 bits of key is sent encrypted with server's public key
  - if full strength cipher key is chosen, then entire key is encrypted with server's public key and clear-master-key field is empty

Read-write Keys from Master-Key

- The master-key is not used as it is for bulk encryption at the SSL record protocol level
  - master-key is used to calculate two sets of keys
    - client-write-key = server-read-key: client uses this key to encrypt data for the server
    - server-write-key = client-read-key: server uses this key to encrypt data for the client
  - some ciphers may require more than one actual key to encrypt or decrypt
    - e.g. Triple DES needs three keys each way

Client-finish Message Contents

- Connection-id originally sent by the server
  - Acts as a nonce value to prevent replay attacks
  - This is encrypted with the client-write-key

Server-verify Message Contents

- Challenge-data originally sent by the client
  - Encrypted using the server-write-key
  - Only the real server will have the private key necessary to decrypt the secret-master-key in the client-master-key message
    - Thus the receipt and decryption of this message authenticates the server to the client

Server-finish Message Contents

- Session-id
  - Used in subsequent handshakes between the same client and same server to avoid having to go through all the cipher and master key negotiations again
  - Cached by each party after a connection is closed and only reused for subsequent connections provided they do not occur too far apart
    - Recommended that session-id have a 100s life in cache

Session Identifier Used, No Client Authentication

Message Type	Direction	Data transferred
Client-hello	C → S	challenge-data, session-id, cipher-specs
Server-hello	S → C	connection-id, session-id-hit
Client-finish	C → S	{connection-id}client-write-key
Server-verify	S → C	{challenge-data}server-write-key
Server-finish	S → C	{session-id}server-write-key

Session Identifier Used, No Client Authentication

- Challenge-data in client-hello message has the same use as in the initial case
- The server uses the session-id sent in client-hello message to look up the cipher kind and the master-key
- The cipher-specs is resend in client-hello in case the server has timed out the session-id by the time the client-hello message reaches it
  - In this case the server can continue as if its a new session

Session Identifier Used, No Client Authentication

- If session-id was found in server's cache
  - Server returns a connection-id and session-id-hit flag
  - Rest of the message exchange is the same as in the previous case except that no client-master-key message is exchanged
- If session-id was not found in server's cache
  - This is a new session id – exchange messages as if no session identifier used

No Session Identifier Used, Client Authentication

Message Type	Direction	Data transferred
Client-hello	C → S	version challenge-data, cipher-specs
Server-hello	S → C	connection-id, server-certificate, version, modified-cipher-specs
Client-master-key	C → S	cipher-kind, clear-master-key, {secret-master-key}server-public-key
Client-finish	C → S	{connection-id}client-write-key
Server-verify	S → C	{challenge-data}server-write-key

No Session Identifier Used, Client Authentication

Message Type	Direction	Data transferred
Request- certificate	S → C	{auth-type, cert-chal-data}server-write-key
Client- certificate	C → S	{cert-type, client-cert, resp-data}client-write-key
Server-finish	S → C	{session-id}server-write-key

Request-certificate Message Contents

- Authentication type
  - Specifies a message digest function and a form of public key encryption
- Certificate challenge data
  - Created by the server and is the data which the client uses to create a digital signature and return as response

Client-certificate Message Contents

- Certificate type
- Client certificate
  - Contains the client's public key
- Response data
  - Certificate challenge data is hashed and then signed using the client's private key
  - If response data properly decrypted with client's public key in certificate then client is verified

Session Identifier Used, Client Authentication

Message Type	Direction	Data transferred
Client-hello	C → S	challenge-data, session-id, cipher-specs
Server-hello	S → C	connection-id, session-id-hit
Client-finish	C → S	{connection-id}client-write-key
Server-verify	S → C	{challenge-data}server-write-key

Session Identifier Used, Client Authentication

Message Type	Direction	Data transferred
Request- certificate	S → C	{auth-type, cert-chal-data}server-write-key
Client- certificate	C → S	{cert-type, client-cert, resp-data}client-write-key
Server-finish	S → C	{session-id}server-write-key

Authentication and Key Exchange

- Three different authetication modes
  - Authentication of both parties
  - Server authenticated with an unauthenticated client
  - Total anonymity

Authentication and Key Exchange

- Anonymous servers cannot authenticate clients
- For any entity that is being authenticated, its certificate message must provide a valid certificate chain leading to an acceptable certificate authority
- Each party is responsible for verifying that the other's certificate is valid and has not expired or been revoked

Anonymous Sessions

- Provides protection only against passive eavesdropping
- Completely anonymous sessions are vulnerable to man-in-the-middle attacks
  - Independent, tamper-proof channel is needed to verify that the finished messages were not replaced by an attacker

SSL Key Exchange Algorithms

- RSA
- Fixed Diffie-Hellman
- Ephemeral Diffie-Hellman
- Anonymous Diffie-Hellman
- Fortezza

SSL Management and Administration

SSL Alert Protocol

- Alert messages convey a description of the alert and the severity of the message – namely `warning`, `fatal`, `other`
- Content type supported by the SSl Record layer
- Alert messages with a level of `fatal` result in the immediate termination of the connection. In this case, other connections corresponding to the session may continue, but the session identifier must be invalidated, preventing the failed session from being used to establish new connections

Closure Alerts

close_notify This message notifies the recepient that the sender will not send any more messages on this connection. The session becomes unresumable if any connection is terminated without the proper `close_notify` messages with level equal to `warning`

Error Alerts

unexpected_message An inappropriate message was received. This alert is always `fatal` and should never be observed in communication between proper implementations

bad_record_mac This alert is returned if a record is received with an incorrect MAC. This message is always `fatal`

decompression_failure The decompression fucntion received improper input (e.g. data that would expand to excessive length). This message is always `fatal`

Error Alerts

handshake_failure Reception of this message indicates that the sender was unable to negotiate an acceptable set of security parameters given the options available. This is a `fatal` error

np_certificate This alert message may be sent in response to a `certification request`, if no appropriate certificate is available

bad_certificate A certificate was corrupt, contained signatures that did not verify correctly etc.

unsupported_certificate A certificate was of an unsupported type

Error Alerts

certificate_revoked A certificate was revoked by its signer

certficate_expired A certificate has expired or is not currently valid

certificate_unknown Some other (unspecified) issue arose in processing of certificate rendering it unacceptable

illegal_parameter A field in the handshake was out of range or inconsistent with other fields. This is always fatal

Applications and SSL

- Use dedicated port numbers for every application that uses SSL
- Negotiate use of SSL during normal TCP/IP connection establishment
- Use normal application port and negotiate security options as part of application protocol

Secure Shell

SSH

- Still evolving
- Originally developed for secure remote login from client to server (host)
- Has potential to evolve into a general purpose transport layer security protocol
  - Alternate to SSL

SSH Protocol Features

- Secure terminal sessions utilizing secure encryption
- Full, secure replacement for FTP and Telnet as well as the UNIX r-series of commands – rlogin, rexec, rcp
- Multiple ciphers for encryption, including 3DES, Blowfish, and AES.
- Transparent and automatic tunneling of X11 connections and arbitrary TCP/IP-based applications, such as e-mail
- Automatic and secure authentication of both ends of connection
- Multiple channels that allow you to have multiple terminal windows and file transfers going through one secure and authenticated connection.

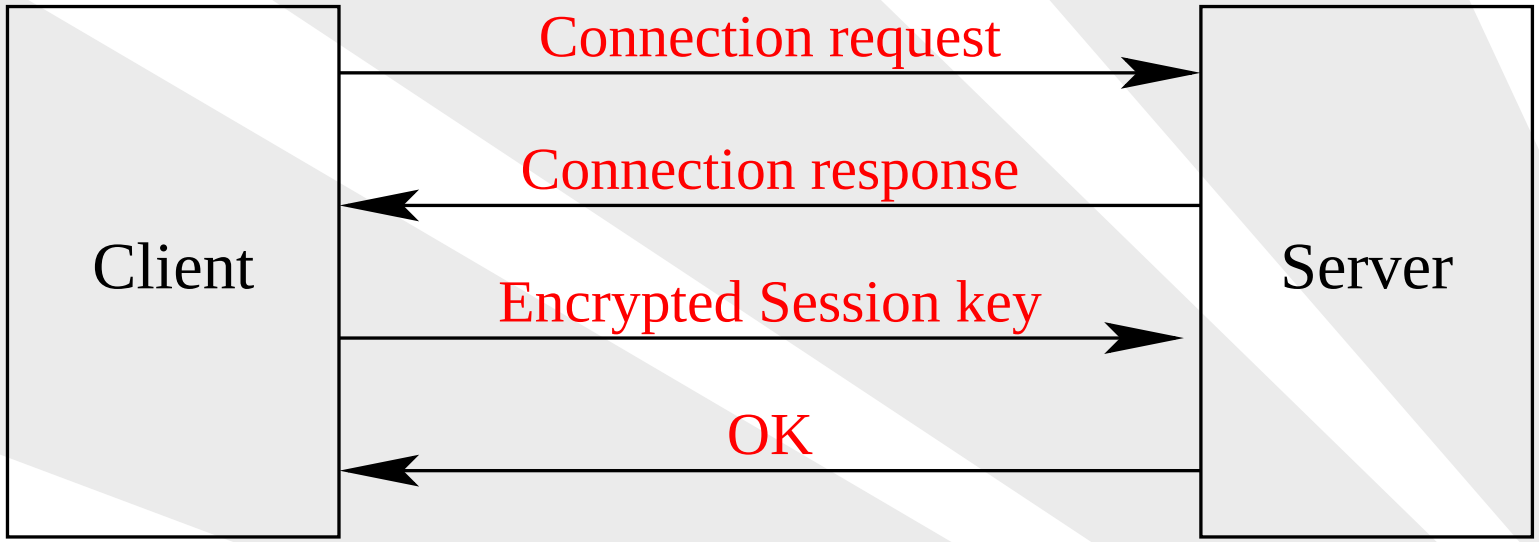
SSH Architecture

- SSH-TRANS
  - Transport layer protocol over TCP connection on port 22
  - Provides server authentication, confidentiality and integrity with optional compression
- SSH-USERAUTH
  - Authenticates client side user to server
- SSH-CONNECT
  - Multiplexes encrypted connection into multiple logical channels

SSH Transport

- Each host must have a public key
  - May have more than one
  - Multiple hosts may share one public key
- Public keys of hosts are stored in a client database
  - Manually configured
  - Accept public keys from servers online
  - Use public key certificates

SSH Transport



SSH Transport

- Connection request
  - Protocol version number
  - Cryptographic algorithms that client can handle
- Connection response
  - Selects crypto algorithms (can be independent in both directions)
  - Public host key (long term)
  - Public server key (short term)
- Encrypted session key
  - 256 bit doubly encrypted with above public keys

SSH Authentication

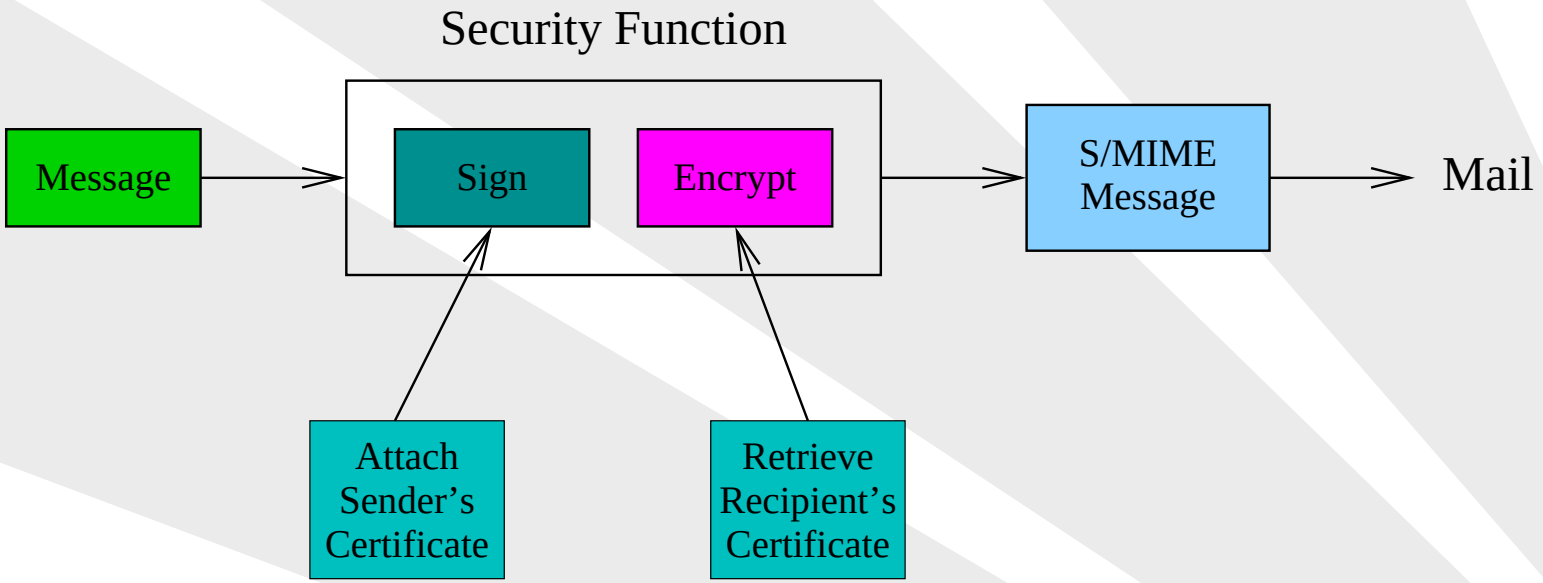
- Client to server authentication alternatives
  - Password based
  - Skey based
  - Public-key based
  - Kerberos
  - Secure-id based

Secure/Multipurpose Internet Mail Extensions (S/MIME)

S/MIME

- Standard for secure electronic messaging – endorsed by Lotus, Microsoft, Netscape and Qualcomm
- Based on IETF's Cryptographic Message Syntax which is almost identical to PKCS #7
- Can be used for
  - Secure email
  - Secure EDI
  - Electronic commerce and online ordering services
  - Secure messaging for legal application and healthcare application

S/MIME – Sending a Message



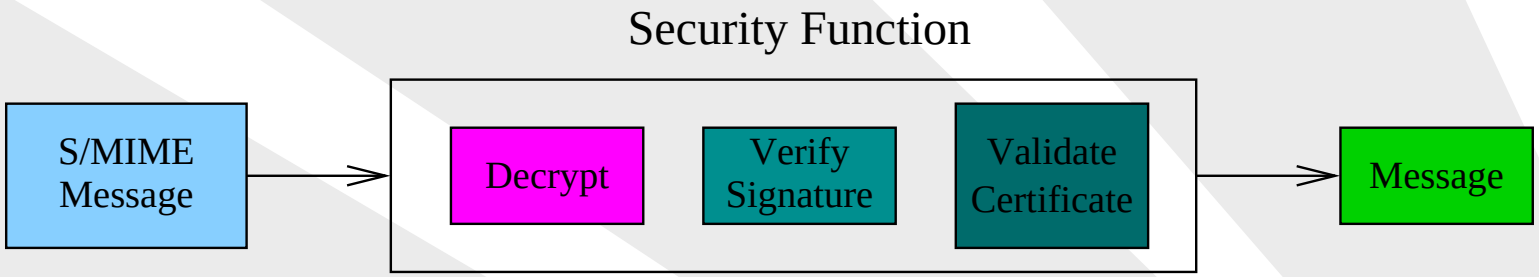
S/MIME – Sending a Message

- To encrypt an outgoing message
  - Obtain a valid certificate from each and every recipient in the address list (cannot be discussion group or mailing list) if not already there
  - Update recipient's certificate if necessary

S/MIME – Sending a Message

- To sign outgoing message
  - Must obtain a valid certificate for the sender if none existing
  - Update sender's certificate if necessary

S/MIME – Receiving a Message



S/MIME – How It Works

- To validate certificate
  - If new self-signed certificate, establish trust manually
  - If new CA-signed certificate, compare to stored CA-signed certificate
  - If old certificate, check against revocation lists