CSCI 454/554 Computer and Network Security

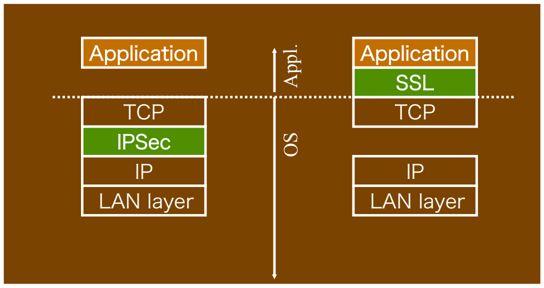
Topic 8.3 SSL/TLS

Outline

1. Overview
2. The SSL Record Protocol
3. The SSL Handshake and Other Protocols

Overview of SSL

Reminder: What Layer?



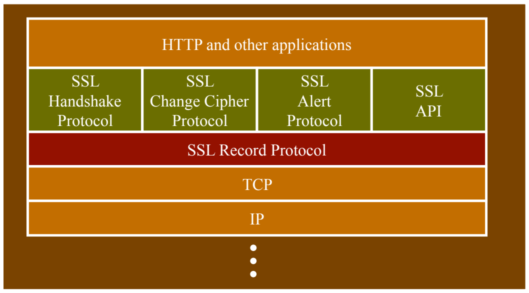
Protocols

- Goal: application independent security
  - Originally for HTTP, but now used for many applications
  - Each application has an assigned TCP port, e.g., https (HTTP over SSL) uses port 443
- Secure Sockets Layer (SSL)
  - the de facto standard for web-based security
  - v3 was developed with public review
- Transport Layer Security (TLS)
  - TLS v1.0 very close to SSL v3.1

Protocol over SSL/TLS

```
Keyword        Decimal        Description
-------        -------        -----------
nsiiops        261/tcp        IIOP Name Service over TLS/SSL
https          443/tcp        http protocol over TLS/SSL
ddm-ssl        448/tcp        DDM-SSL
smtps          465/tcp        smtp protocol over TLS/SSL
nntps          563/tcp        nntp protocol over TLS/SSL
sshell         614/tcp        SSLshell
ldaps          636/tcp        ldap protocol over TLS/SSL
ftps-data      989/tcp        ftp protocol, data, over TLS/SSL
ftps           990/tcp        ftp, control, over TLS/SSL
telnets        992/tcp        telnet protocol over TLS/SSL
imaps          993/tcp        imap4 protocol over TLS/SSL
ircs           994/tcp        irc protocol over TLS/SSL
pop3s          995/tcp        pop3 protocol over TLS/SSL
```

SSL Architecture



HTTP and other applications

SSL Handshake Protocol | SSL Change Cipher Protocol | SSL Alert Protocol | SSL API

SSL Record Protocol

TCP

IP

- Relies on TCP for reliable communication

Architecture (Cont'd)

- Handshake protocol: establishment of a session key
- Change Cipher protocol: start using the previously-negotiated encryption / message authentication
- Alert protocol: notification (warnings or fatal exceptions)
- Record protocol: protected (encrypted, authenticated) communication between client and server

SSL Services

- Peer authentication
- Negotiation of security parameters
- Generation / distribution of session keys
- Data confidentiality
- Data integrity

Connections and Sessions

- SSL *Session*
  - an association between peers
  - created through a handshake, negotiates security parameters, can be long-lasting
- SSL *Connection*
  - a type of service (i.e., an application) between a client and a server
  - transient
- Multiple connections can be part of a single session

Session Parameters

- Session ID
- X.509 public-key certificate of peer
- Compression algorithm to use
- Cipher specification: encryption algorithm, message digest, etc.
- Master (session) secret: 48-byte (384 bits) secret negotiated between peers

Connection Parameters

- Server and client nonces
- Server and client authentication keys
- Server and client encryption keys
- Server and client initialization vectors
- Current message sequence number

Ciphers Supported by SSL

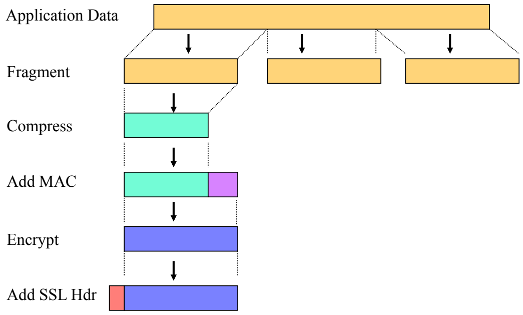
- DES+HMAC/SHA-1
- 3DES+HMAC/SHA-1
- RC4+MD5
- RC2+MD5
- +others

The SSL Record Protocol

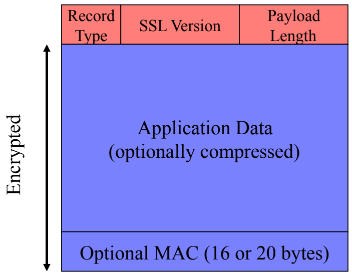
Protocol Steps

1. Fragment data stream into records
   - each with a maximum length of 2^{14} (=16K) bytes
2. Compress each record
3. Create message authentication code for each record
4. Encrypt each record

Steps… (cont'd)

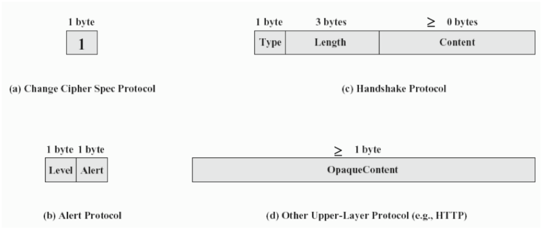


SSL Record Format


Record Type
SSL Version
Payload Length
Application Data (optionally compressed)
Optional MAC (16 or 20 bytes)
Encrypted

- There is, unfortunately, some version number silliness between v2 and v3; see text for (ugly) details

Possible Record "Payloads"

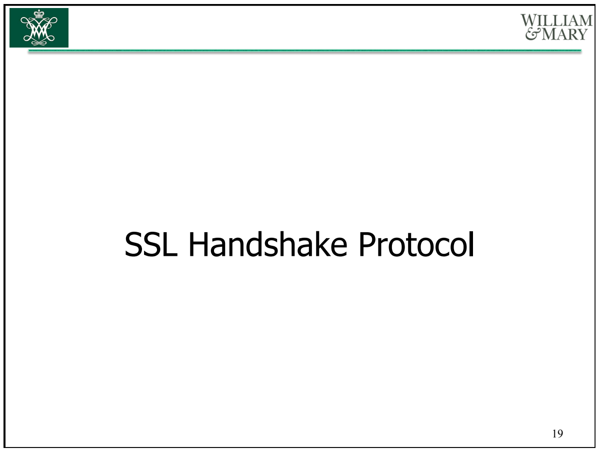


(a) Change Cipher Spec Protocol

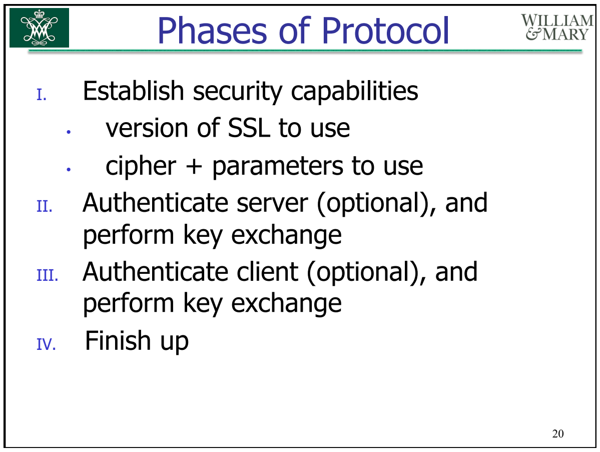
(c) Handshake Protocol

(b) Alert Protocol

(d) Other Upper-Layer Protocol (e.g., HTTP)



SSL Handshake Protocol



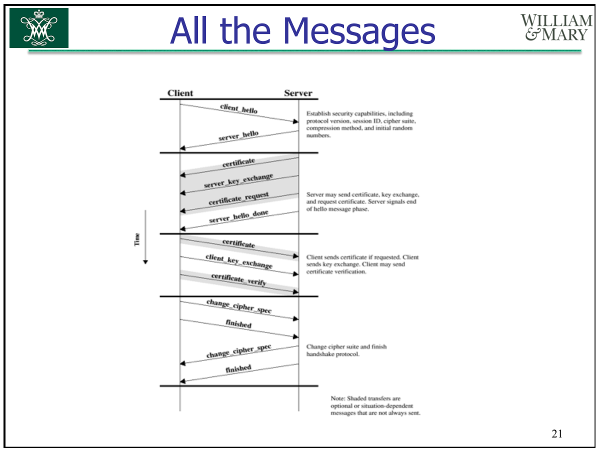
Phases of Protocol

I. Establish security capabilities
- version of SSL to use
- cipher + parameters to use

II. Authenticate server (optional), and perform key exchange

III. Authenticate client (optional), and perform key exchange

IV. Finish up



All the Messages

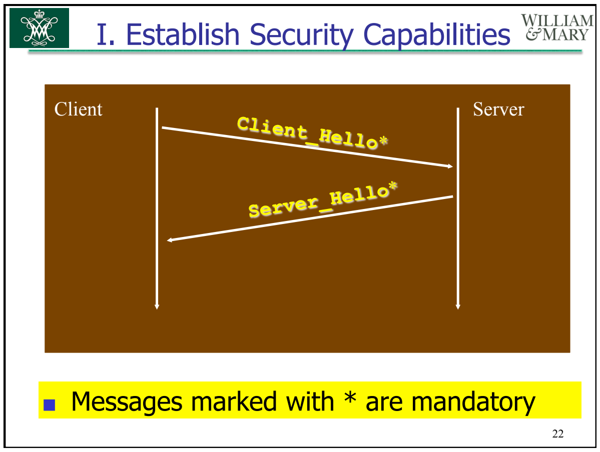
Establish security capabilities, including protocol version, session ID, cipher suite, compression method, and initial random numbers.

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

Client sends certificate if requested. Client sends key exchange. Client may send certificate verification.

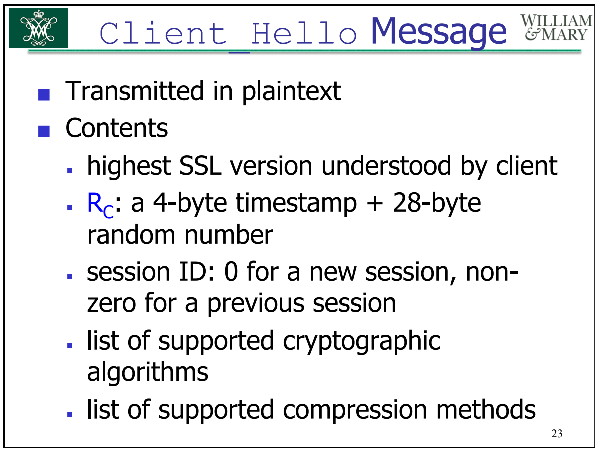
Change cipher suite and finish handshake protocol.

Note: Shaded transfers are optional or situation-dependent messages that are not always sent.



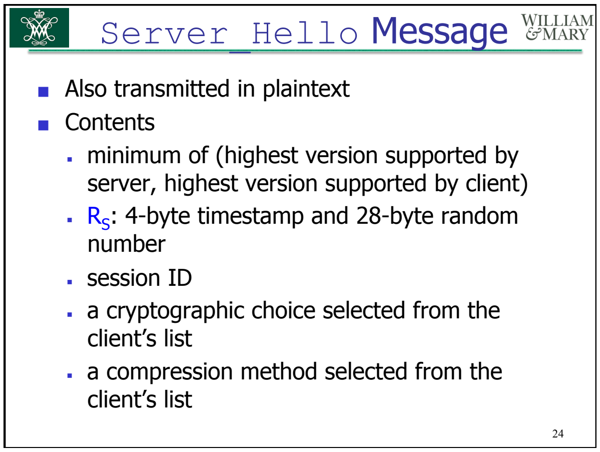
I. Establish Security Capabilities

- Messages marked with * are mandatory



Client_Hello Message

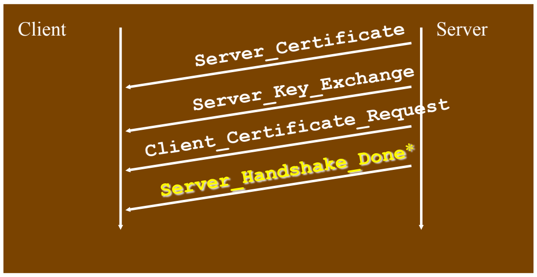
- Transmitted in plaintext
- Contents
  - highest SSL version understood by client
  - R_C: a 4-byte timestamp + 28-byte random number
  - session ID: 0 for a new session, non-zero for a previous session
  - list of supported cryptographic algorithms
  - list of supported compression methods



Server_Hello Message

- Also transmitted in plaintext
- Contents
  - minimum of (highest version supported by server, highest version supported by client)
  - R_S: 4-byte timestamp and 28-byte random number
  - session ID
  - a cryptographic choice selected from the client’s list
  - a compression method selected from the client’s list

II. Server Auth. / Key Exchange

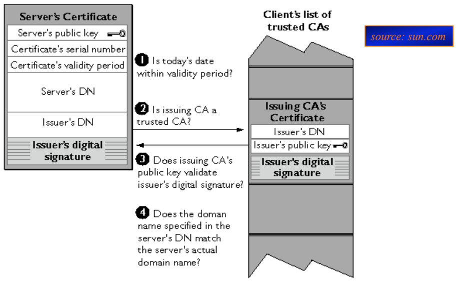


- The Server_Certificate message is optional, but almost always used in practice

Server_Certificate Message

- Contains a certificate with server's public key, in X.509 format
  - or, a chain of certificates if required
- The server certificate is necessary for any key exchange method except for anonymous Diffie-Hellman

Authenticating the Server


Server's Certificate
Server's public key
Certificate's serial number
Certificate's validity period
Server's DN
Issuer's DN
Issuer's digital signature
Client's list of trusted CAs
Issuing CA's Certificate
Issuer's DN
Issuer's public key
Issuer's digital signature
1 Is today's date within validity period?
2 Is issuing CA a trusted CA?
3 Does issuing CA's public key validate issuer's digital signature?
4 Does the doman name specified in the server's DN match the server's actual domain name?

source: sun.com

- Step #4: Domain name in certificate must match domain name of server (not part of SSL protocol, but clients should check this)

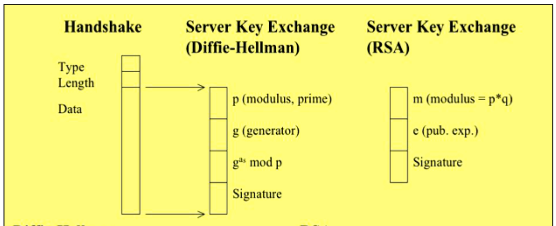
Key Exchange Methods Supported

- RSA (server must have a certificate)
- Ephemeral Public Key
  - public keys are exchanged, signed using long-term RSA keys
- Fixed Diffie-Hellman
  - server provides the D-H public parameters in a certificate
  - client responds with D-H public key either in a certificate, or in a key exchange message
- Anonymous Diffie-Hellman
  - Diffie-Hellman without authentication
  - Susceptible to Man-in-the-middle attack

Server_Key_Exchange Message

- Needed for…
  - anonymous D-H
  - ephemeral public key

Server Key Exchange

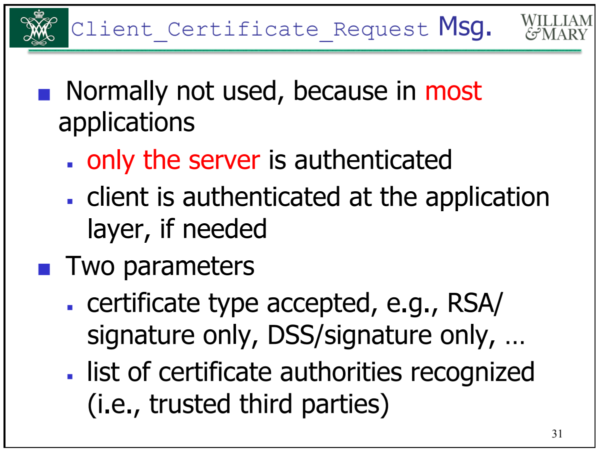


Diffie-Hellman

Client Computes: PreMasterSecret $= (g^{as})^{ac} \bmod p$
Client Sends : g^{ac} to server
Server Computes: PreMasterSecret $= (g^{ac})^{as} \bmod p$

RSA

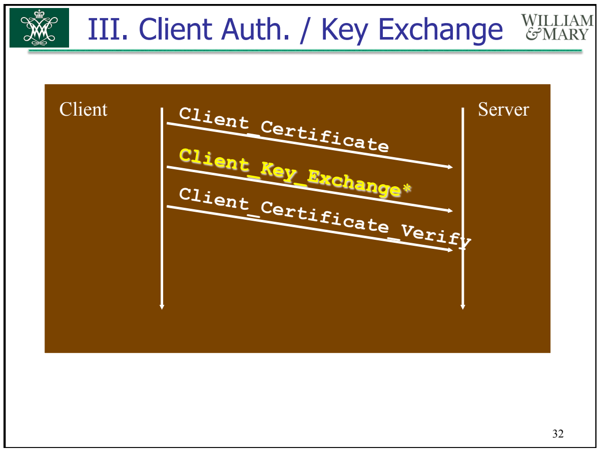
Client Computes: $y =$ PreMasterSecret$^{e} \bmod p$
Client sends : y to server
Server Computes: PreMasterSecret $= y^{d} \bmod p$



Client_Certificate_Request Msg.

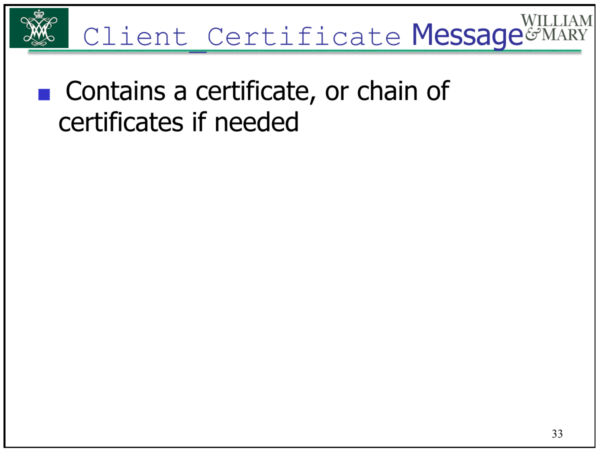
WILLIAM & MARY

- Normally not used, because in most applications
  - only the server is authenticated
  - client is authenticated at the application layer, if needed
- Two parameters
  - certificate type accepted, e.g., RSA/ signature only, DSS/signature only, …
  - list of certificate authorities recognized (i.e., trusted third parties)



III. Client Auth. / Key Exchange

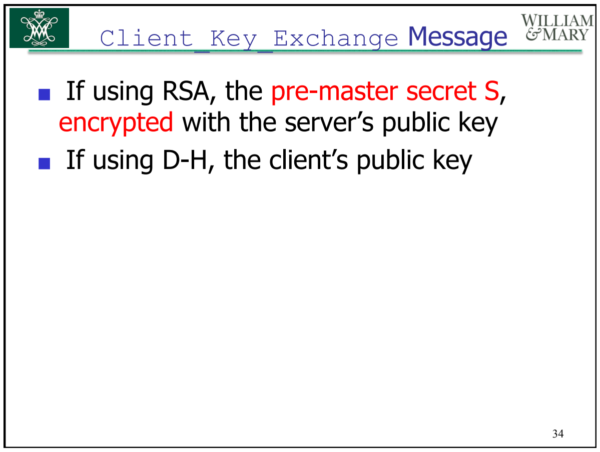
WILLIAM & MARY



Client_Certificate Message

WILLIAM & MARY

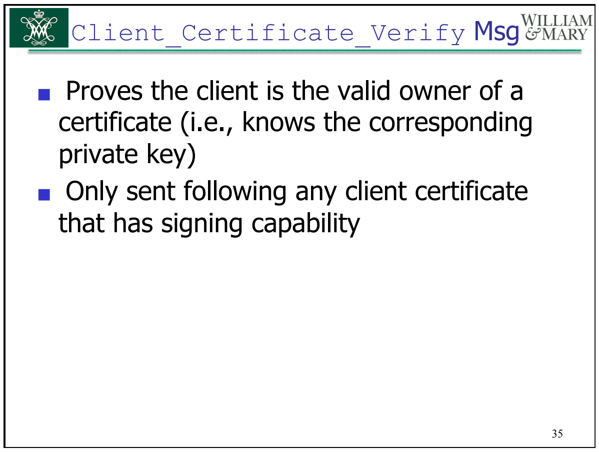
- Contains a certificate, or chain of certificates if needed



Client_Key_Exchange Message

WILLIAM & MARY

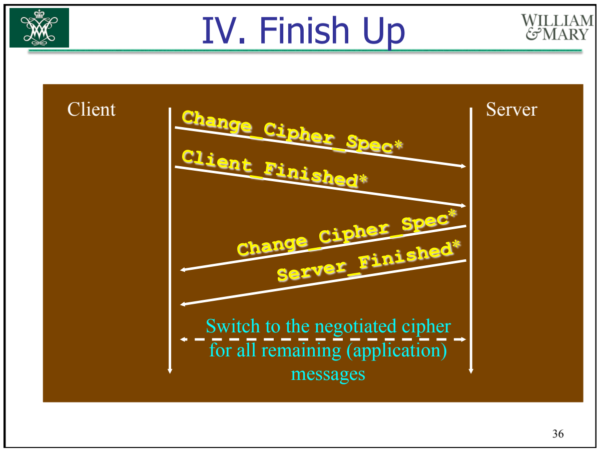
- If using RSA, the pre-master secret S, encrypted with the server's public key
- If using D-H, the client's public key



Client_Certificate_Verify Msg

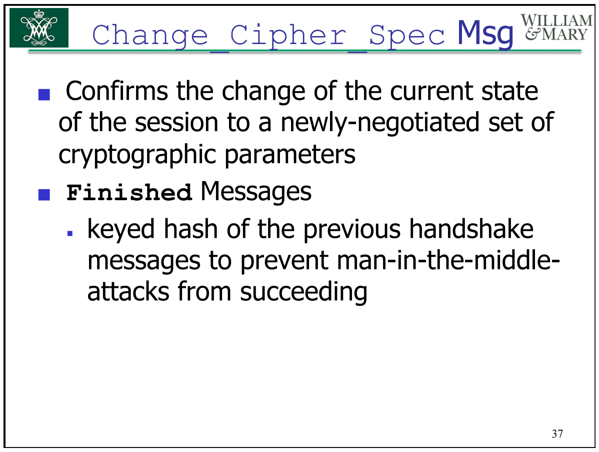
WILLIAM & MARY

- Proves the client is the valid owner of a certificate (i.e., knows the corresponding private key)
- Only sent following any client certificate that has signing capability



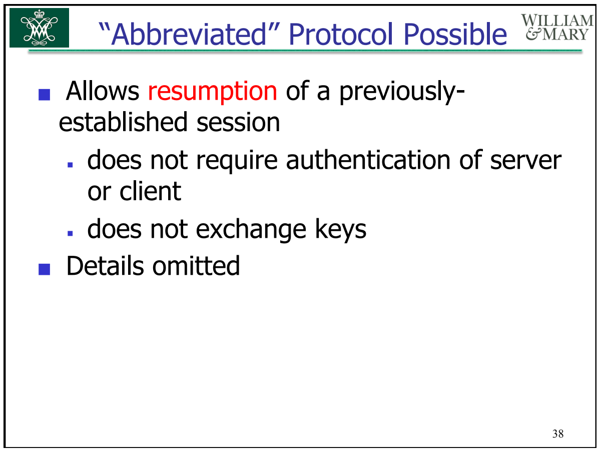
IV. Finish Up

WILLIAM & MARY



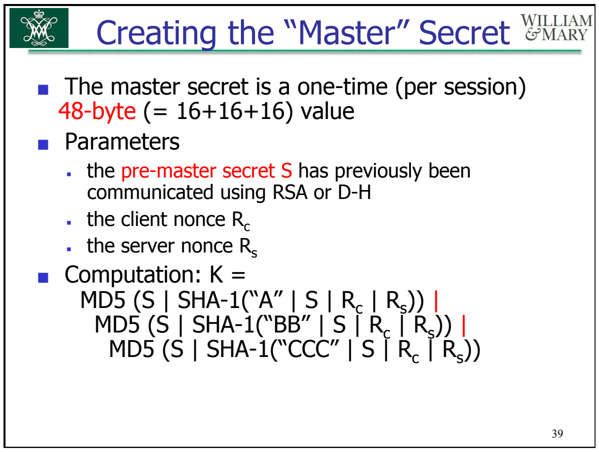
`Change_Cipher_Spec` Msg

- Confirms the change of the current state of the session to a newly-negotiated set of cryptographic parameters
- **`Finished`** Messages
  - keyed hash of the previous handshake messages to prevent man-in-the-middle-attacks from succeeding



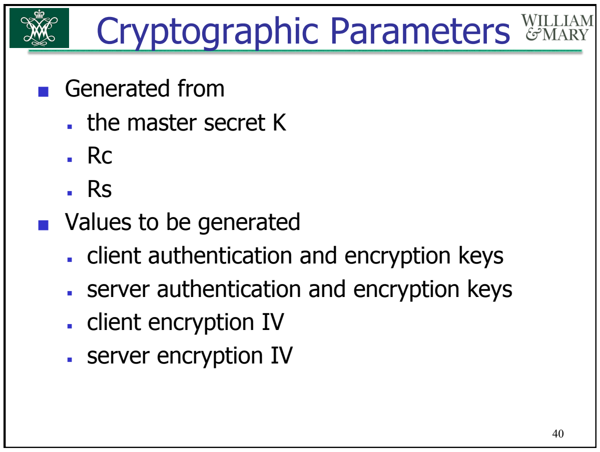
"Abbreviated" Protocol Possible

- Allows resumption of a previously-established session
  - does not require authentication of server or client
  - does not exchange keys
- Details omitted



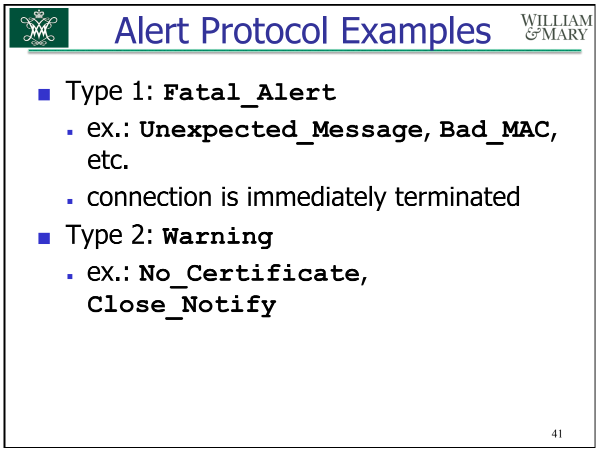
Creating the "Master" Secret

- The master secret is a one-time (per session) 48-byte (= 16+16+16) value
- Parameters
  - the pre-master secret S has previously been communicated using RSA or D-H
  - the client nonce R_c
  - the server nonce R_s
- Computation: K =
  MD5 (S | SHA-1("A" | S | R_c | R_s)) |
  MD5 (S | SHA-1("BB" | S | R_c | R_s)) |
  MD5 (S | SHA-1("CCC" | S | R_c | R_s))



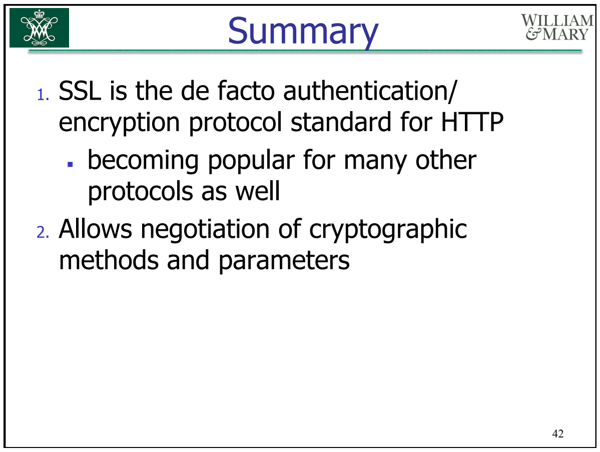
Cryptographic Parameters

- Generated from
  - the master secret K
  - Rc
  - Rs
- Values to be generated
  - client authentication and encryption keys
  - server authentication and encryption keys
  - client encryption IV
  - server encryption IV



Alert Protocol Examples

- Type 1: **`Fatal_Alert`**
  - ex.: **`Unexpected_Message`**, **`Bad_MAC`**, etc.
  - connection is immediately terminated
- Type 2: **`Warning`**
  - ex.: **`No_Certificate`**, **`Close_Notify`**



Summary

1. SSL is the de facto authentication/ encryption protocol standard for HTTP
   - becoming popular for many other protocols as well
2. Allows negotiation of cryptographic methods and parameters