Secure Socket Layer (SSL)

Overview of SSL

Marilyn Allmond & Mary Sweat
allmond@us.ibm.com sweatm@us.ibm.com

IBM S/390 Security
Advanced Technical Support
Washington Systems Center
Gaithersburg, MD

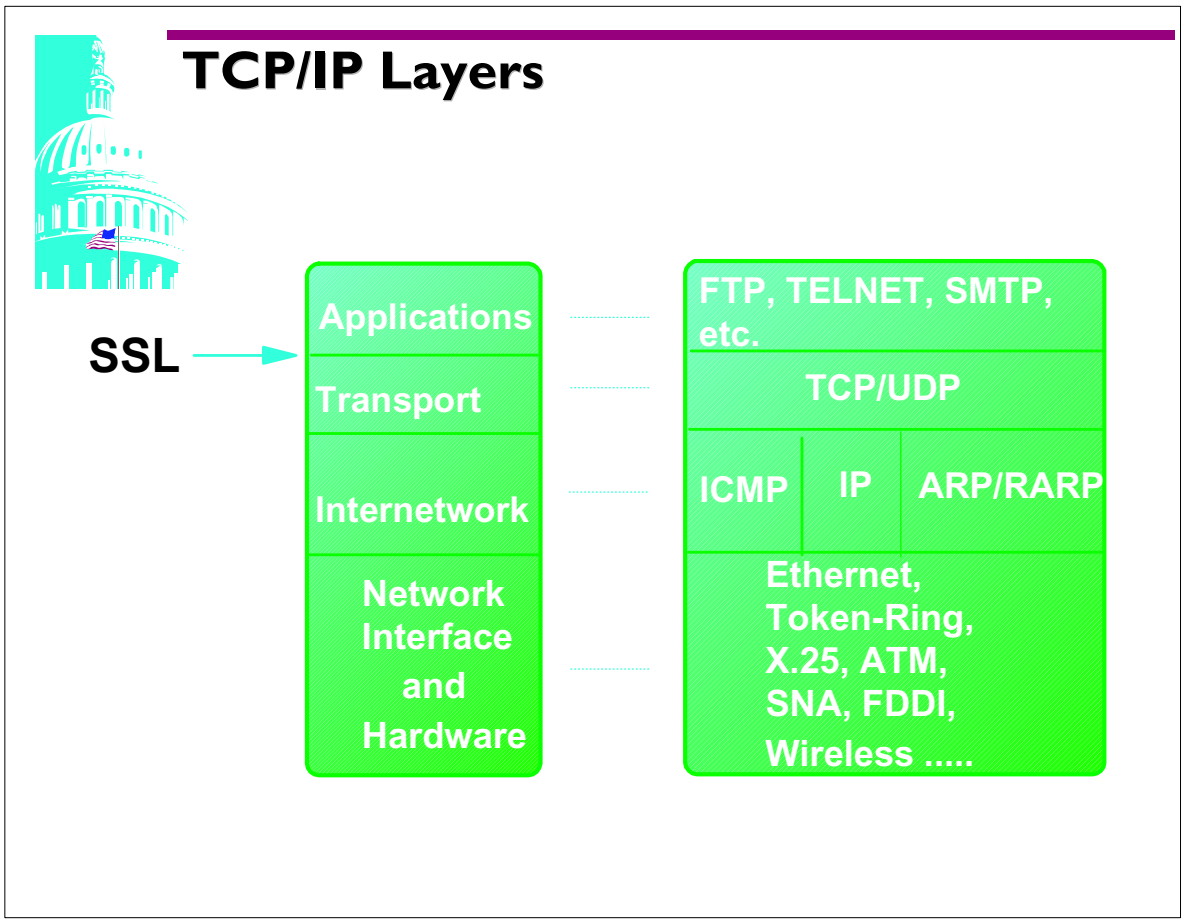
Objectives

- To introduce Secure Socket Layer algorithm and how it provides secure communications for transmitting data

SSL: What is it

- SSL, Secure Sockets Layer, is a protocol developed by Netscape, Inc.(TM). SSL is a "de facto" standard due to its wide use by applications.
- Purpose
  - communication protocol
  - allows a session to be established between two parties, a client and a server
    - with varying degrees of security based on a negotiated agreement between them.

- Applications that use SSL are client/server applications.
- Any application wishing to communicate via a client/server structure can use the Secure Sockets Layer protocol.
- SSL provides the means to have a commonly accepted communication structure, thereby. expanding the "audience" of the application.



- SSL is placed between the TCP (connection-oriented network layer) and the application protocol layer.

SSL: Why Use It

- Any application communicating via a client/server can use the SSL protocol
- Increase the security level of communicated data (beyond what the communication method already provides)
- Ensure who is communicating (client and server can authenticate each other)
- Provide secure/private connections
- Efficiency
  - optional caching scheme that reduces the number of connections established from scratch

- Application Data is protected while in transit.
- A password and user ID could flow in the clear before the handshake is completed. Depending on how the application has been written, a URL may require a userid and password before it has started an HTTPS session.
- A client and server can negotiate with each other regarding what encryption algorithm and cryptographic keys will be used. They do not need to know each other's code and encryption is used after an initial contact.
- Since cryptographic operations can be highly CPU intensive, using SSL is more efficient. SSL uses an optional caching scheme which reduces the number of connections that need to be established from scratch which reduces network activity.

SSL: Functions

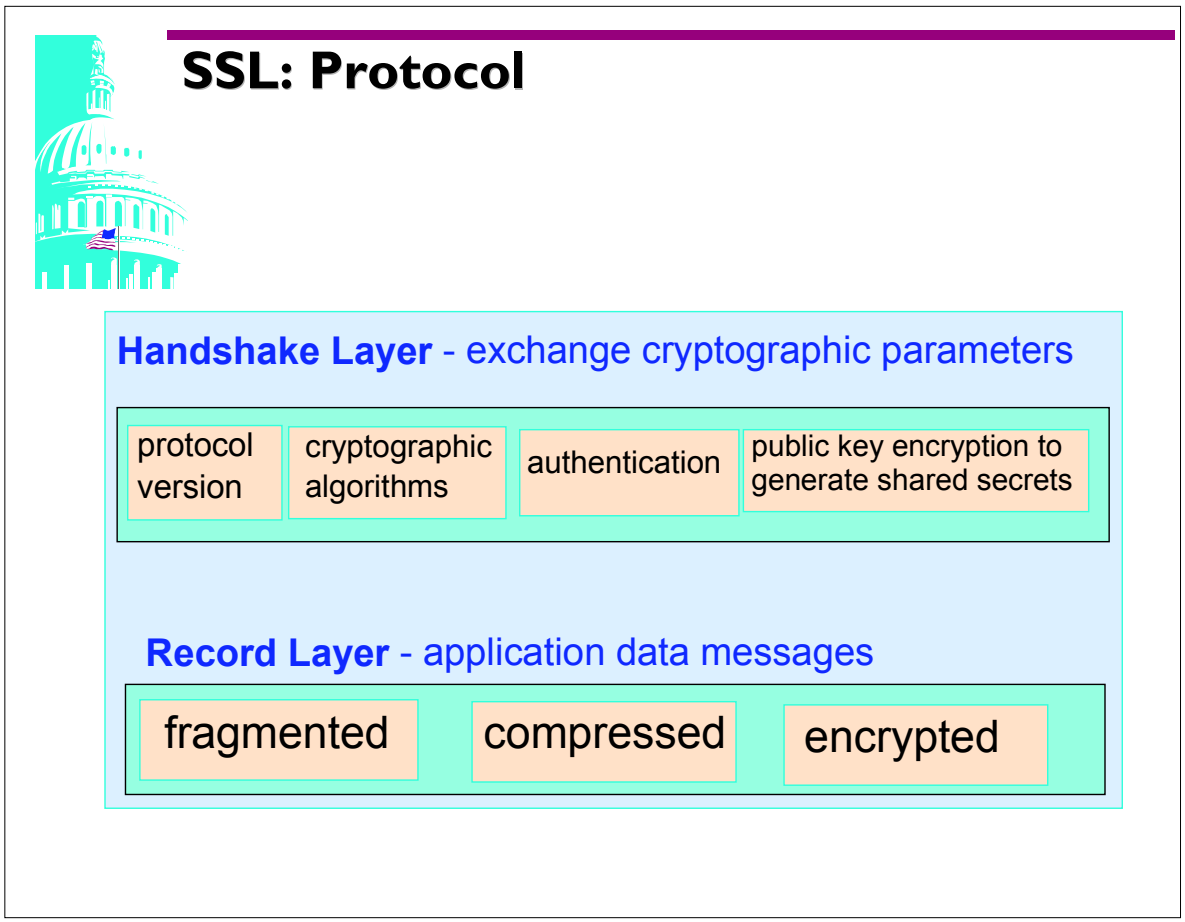
Server

1. provides information and data to the client at the client's request
2. decides what data should be protected
3. is usually an application written to provide data services outbound
4. has the responsibility to protect its identity (will prove its identity via a certificate)

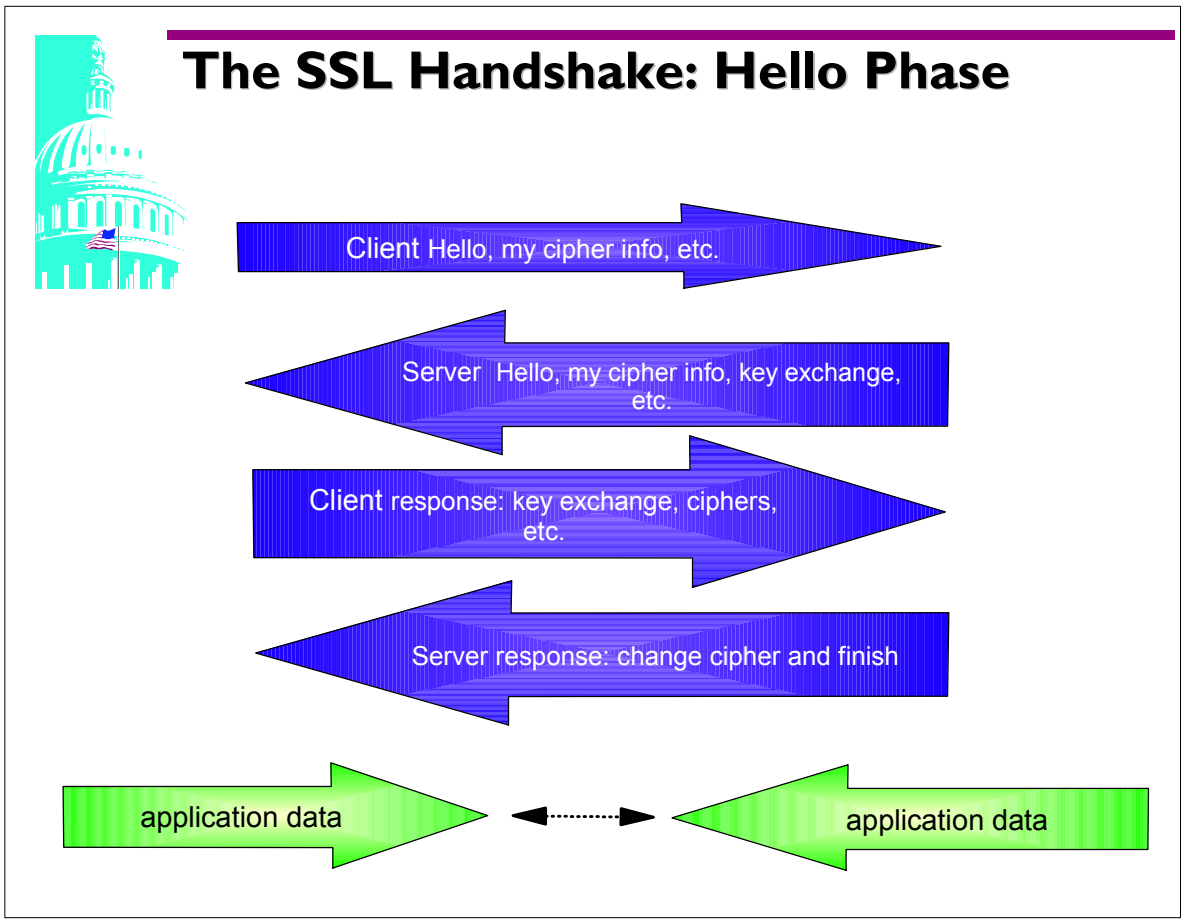
Client

1. initiates the communications
2. generally selects the data to be provided by the Server
3. most are browsers but not necessarily
4. most applications do not care who the clients are; some do
5. can prove its identity by also having a certificate

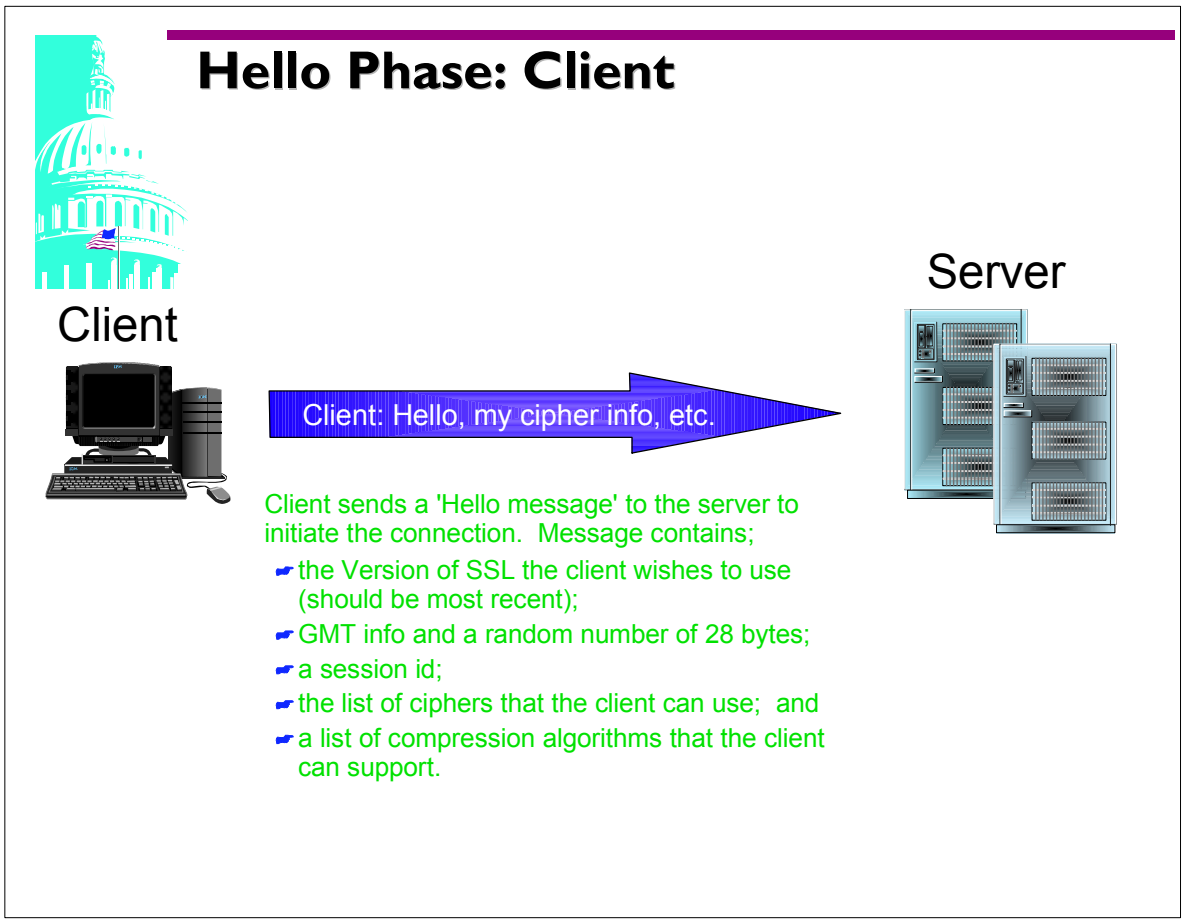
SSL Protocol Basics



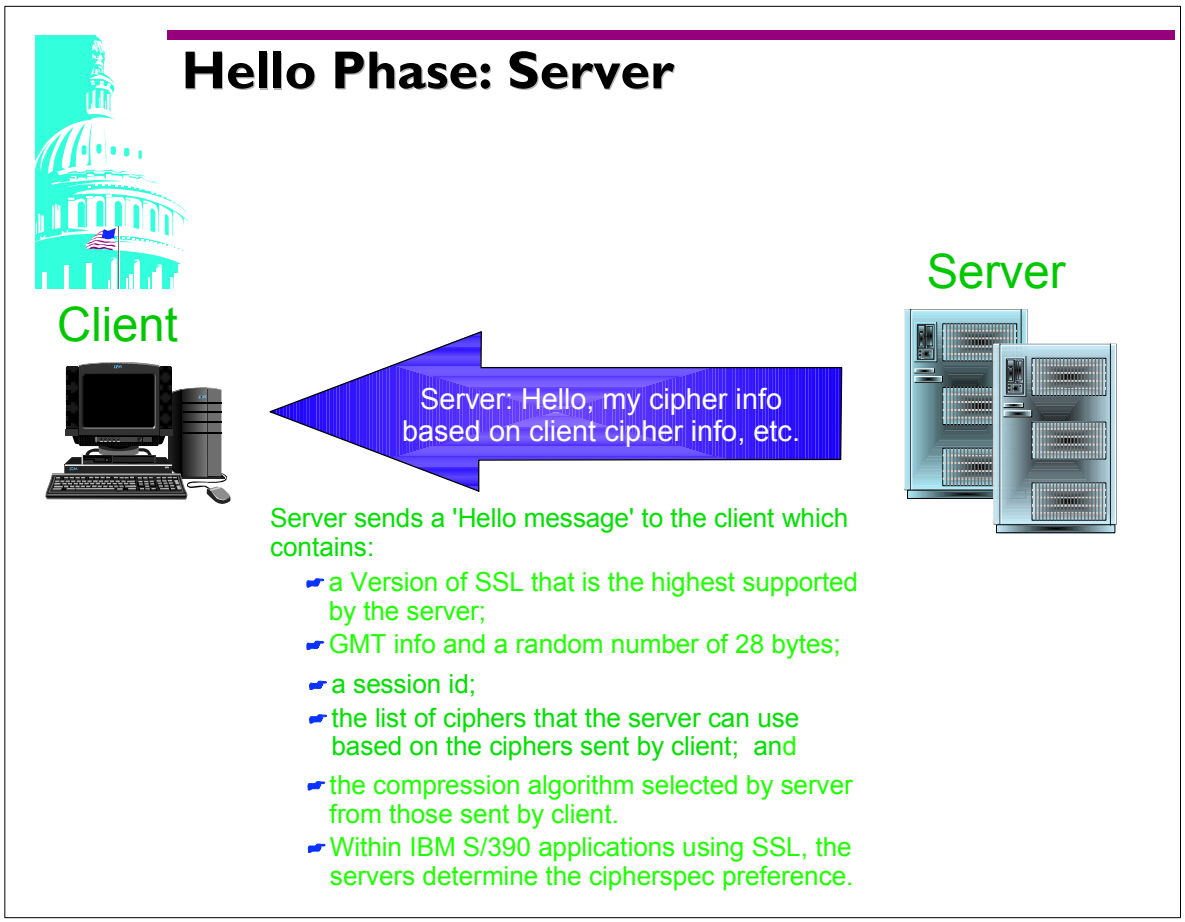
- SSL is a layered protocol. At each layer, messages may include fields for length, description, content.
- There are 2 distinct bounds of communication between the client and server called the handshake layer and the record layer.
- Handshake Layer operates on top of the SSL Record Layer. In the Handshake Layer the client contacts the server and information is exchanged to determine the capabilities of each and to come to an agreement on possible information required later.
- In the Record Layer the client and server exchange data, based on the functions selected in the Handshake Layer.



- From a high level this is the information the SSL client and server exchange with one another. In the following material the actual contents are discussed in detail.



- The client and server hello messages generate random numbers created by a secure random number generator. These numbers must be different and will be used later when creating a master key which is used for encrypting the application data. The client also passes a list of algorithms used to compress data prior to encryption and a CipherSuite list. Each CipherSuite defines both a key exchange algorithm and a CipherSpec (bulk data encryption algorithm and MAC algorithm, hash size, etc.)



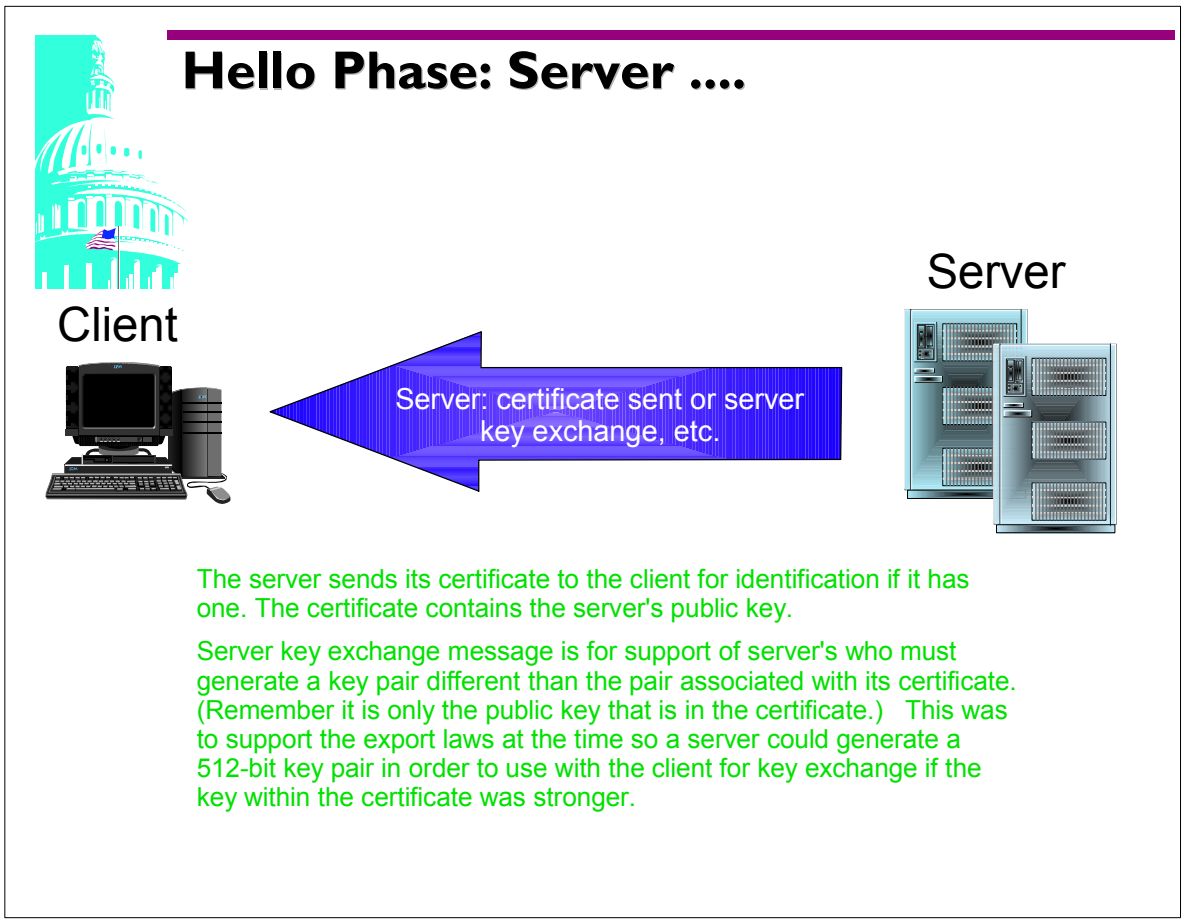
- During the Handshake process, the server processes the client's hello and responds with either a server hello message or a handshake_failure alert message. A failure occurs if no acceptable cryptographic algorithms are available to the server. The server selects a CipherSuite and a Compression algorithm from the list supplied by the client.

- The random number generated by the server must be different from the client.

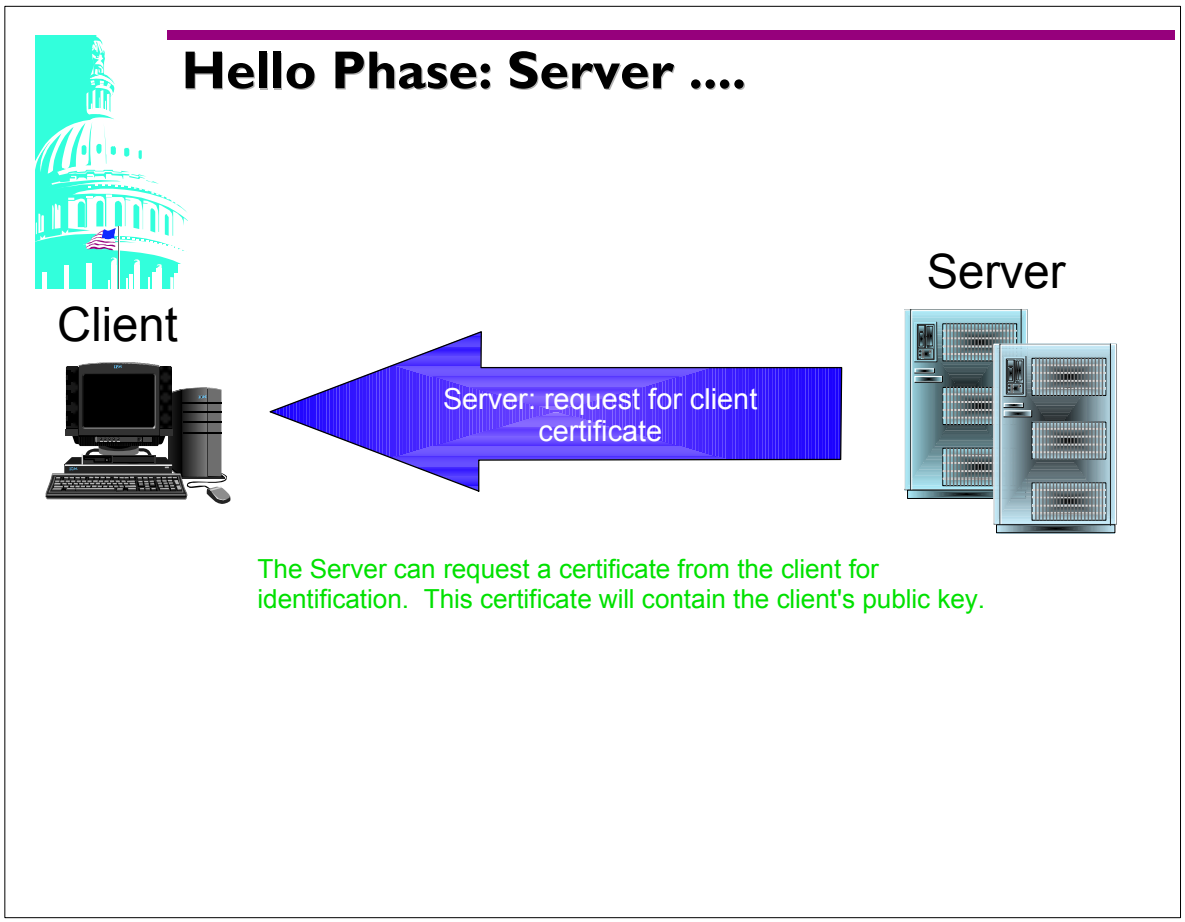
- If the client hello contains a non-empty session id, the server looks in its session cache for a match. If one is found and the server wants to connect with the same value, the server will use the same cipher_spec, same client random #, and compression algorithm.

- Whether to cache information or not is made by the server based on server configuration values or it may be hard coded in the server code.

- S/390 selection of cipherspec preference is based on the order in which the server configuration cipher values are listed in the configuration setup.



- The server sends the client its certificate for authentication if it has one. This occurs immediately following the server hello message.
- Certificates are in a sequence chain, ordered by the senders certificate first and the root certificate authority last.
- The certificate contains the certificate format info, the server's distinguished name, its public key, algorithms used to generate the key, validity dates, name of the CA, CA signature and algorithms used to sign the certificate (usually RSA).
- The server's private keys are stored on a keyring file or database depending on the utility used.
- If the server has no certificate or the certificate is only used for signing (meaning the certificate can not be used for key distribution) the server will send it's server key exchange message.
- The key exchange message contains a key exchange algorithm (RSA, Diffie-Hellman or Fortezza), key exchange parameters (public encryption key), the hash value of these parameters and the Signature algorithm used.
- On S/390 there are a variety of methods to generate an RSA key pair for the server. A certificate request must be generated and sent to a certification authority to be certified. The utilities available on S/390 are;
  1. GSKKYMAN utility (provided with base S/390)
  2. the RACF component
  3. use a certificate from the IBM Vault Registry (AIX details located at www-4.ibm.com/software/security/registry)



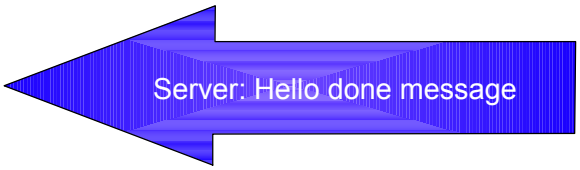
- As a client on OS/390 you can generate a RSA key pair for yourself. Once you have a key pair you must request a certificate request which you will send to a certificate authority of your choice to be certified.

- The utilities available on S/390 are;
  1. GSKKYMAN utility (provided with base S/390
  2. the RACF component 3

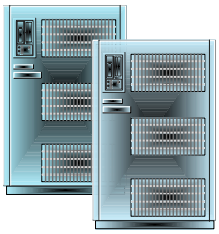
- The client private key is stored on a secure keyring file or database depending on the utility used.

Hello Phase: Server ....

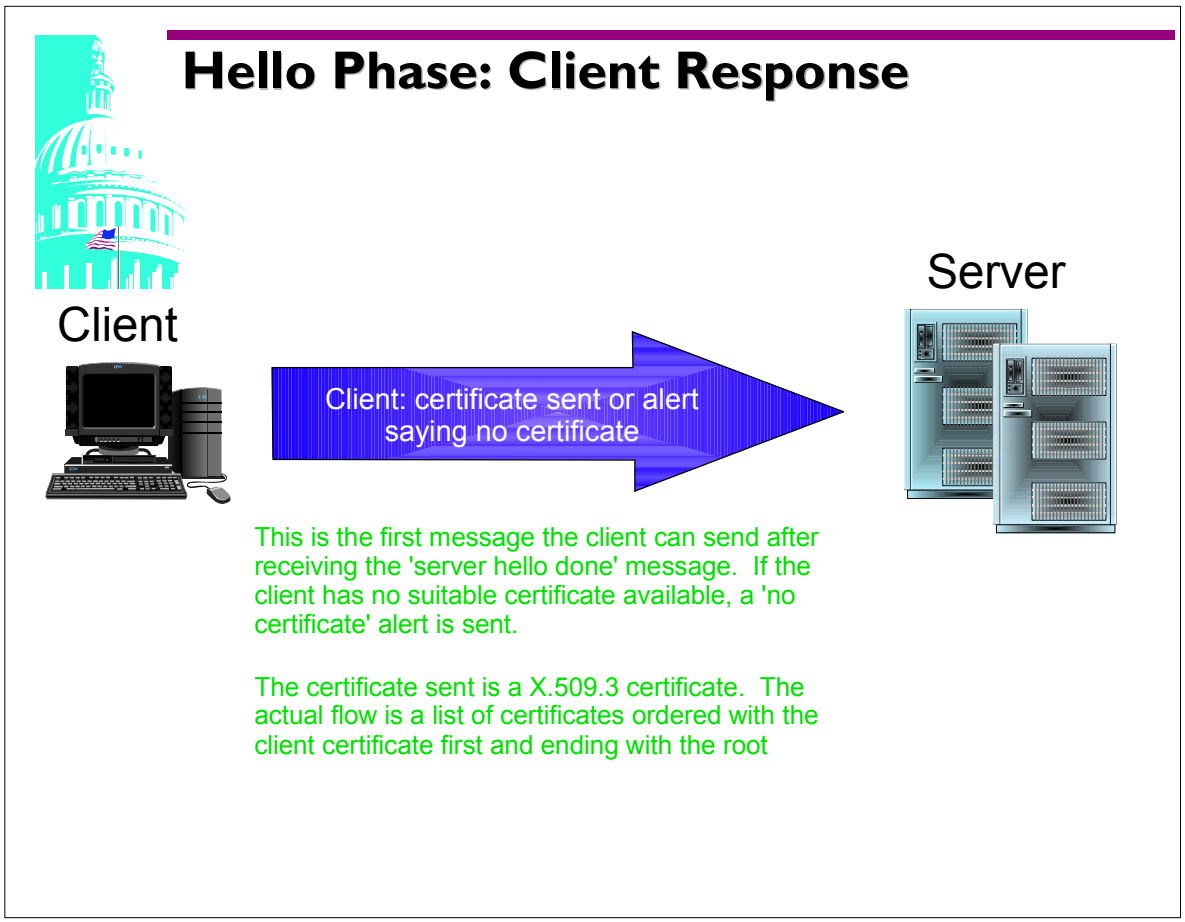
Client



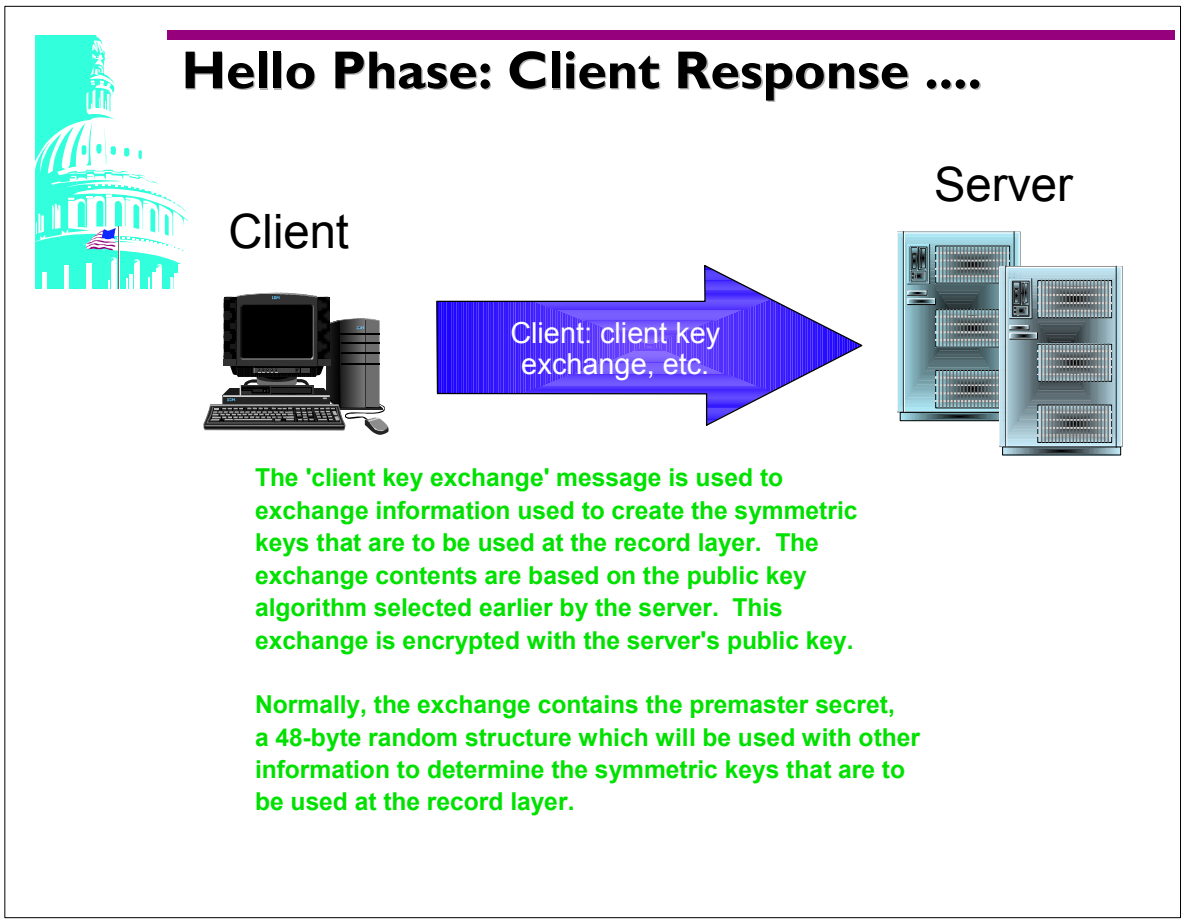
Server



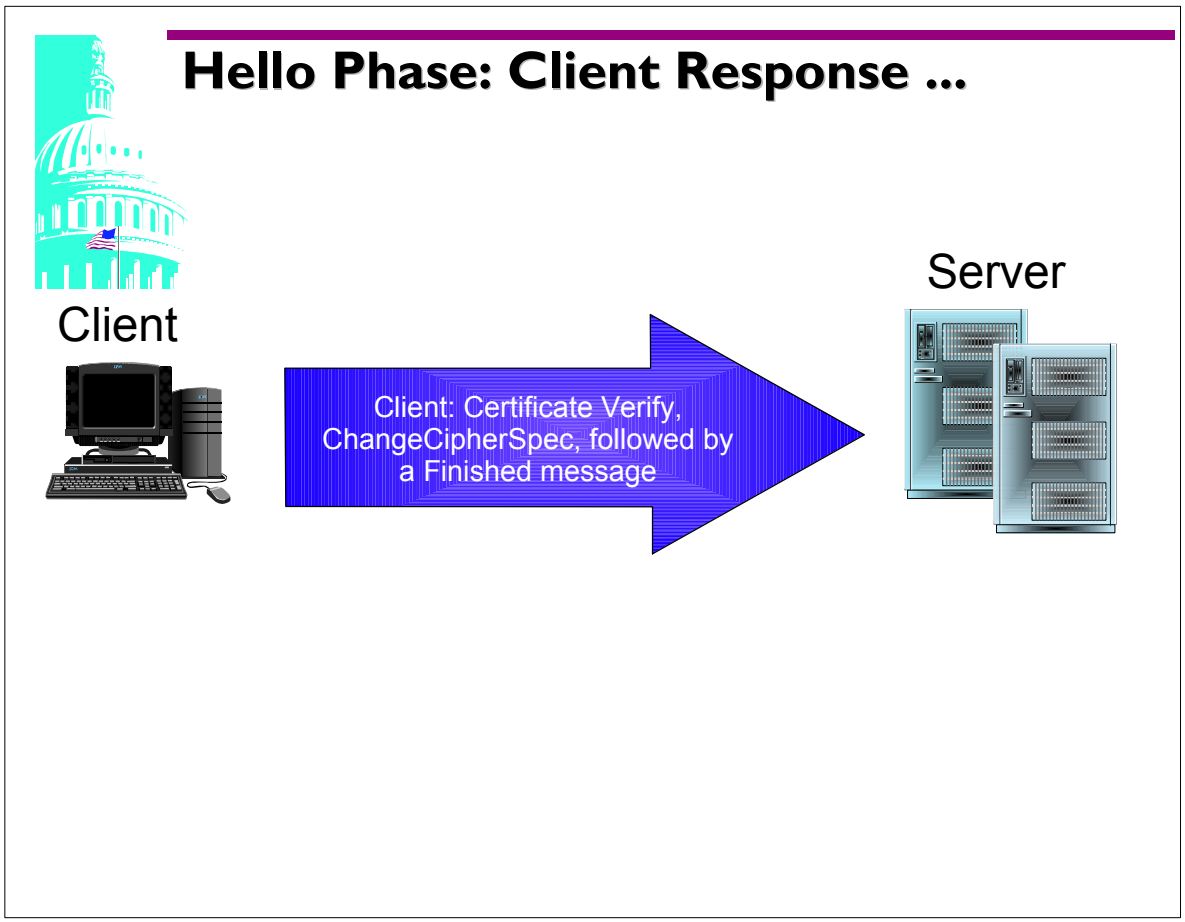
The Server indicates that its hello and associated messages are complete. The Server now waits for a response from the client.



- The client verifies that the server provided a valid certificate, if one was required, and checks that the parameters passed by the server are acceptable. If a client certificate was requested and the client has one, it is sent immediately after the server's hello done message is received.



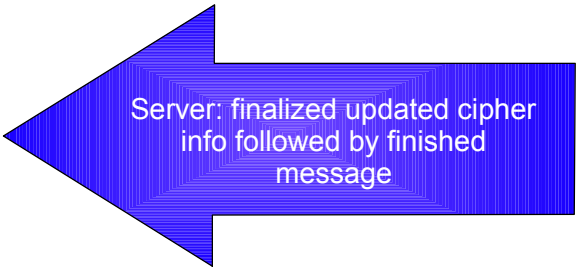
- The client key exchange message uses the public key exchange algorithm specified in the server's certificate or key exchange message. There are three public key algorithms that can be used in SSL: RSA, Fortezza DMS and Diffie-Hellman.
- If using RSA or Fortezza the client generates a pre_master_secret key and encrypts it with the server's public key. This public key was supplied in the server certificate or during the server key exchange.
- For Diffie-Hellman the pre_master_key is created from a Diffie_Hellman computation.
- The SSL protocol specifically outlines how the pre_master_key is to be manipulated. This encrypted pre_master_key is sent to the server and the server decrypts. Both the client and the server convert the pre_master_key into the master_secret key using the random generate numbers created by the client and server earlier in the handshake.
- This master_secret key is the one use by the client and server to encrypt application data.



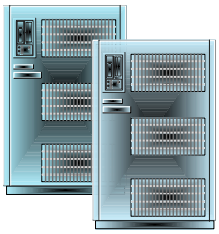
- A change cipher spec is sent from client to the server to indicate that the new key and algorithms should be used for future communication. This is followed by a Finished message.
- This Finished message is the first message encrypted under the new algorithm so both the client and the server must decrypt this message and verify that the contents are correct.
- The "finished" message is always sent after a "change cipher specs' message. The finished messages are the end of the handshake.

Hello Phase: Server Response

Client

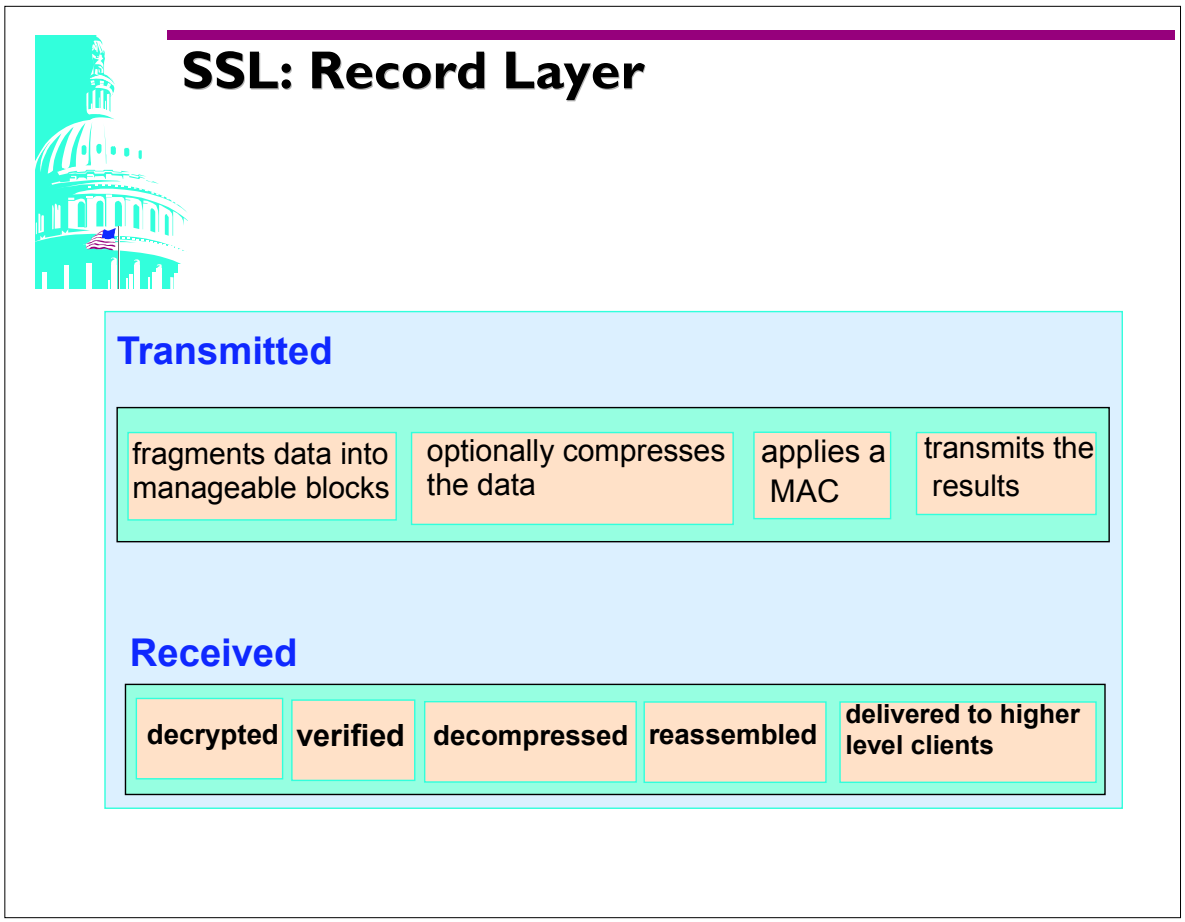


Server



The Server will send his own 'change cipher spec' message if needed. The Server must also send a 'finished' message and change its negotiated info from pending to current status. This ends the handshake layer.

The SSL Handshake: Complete

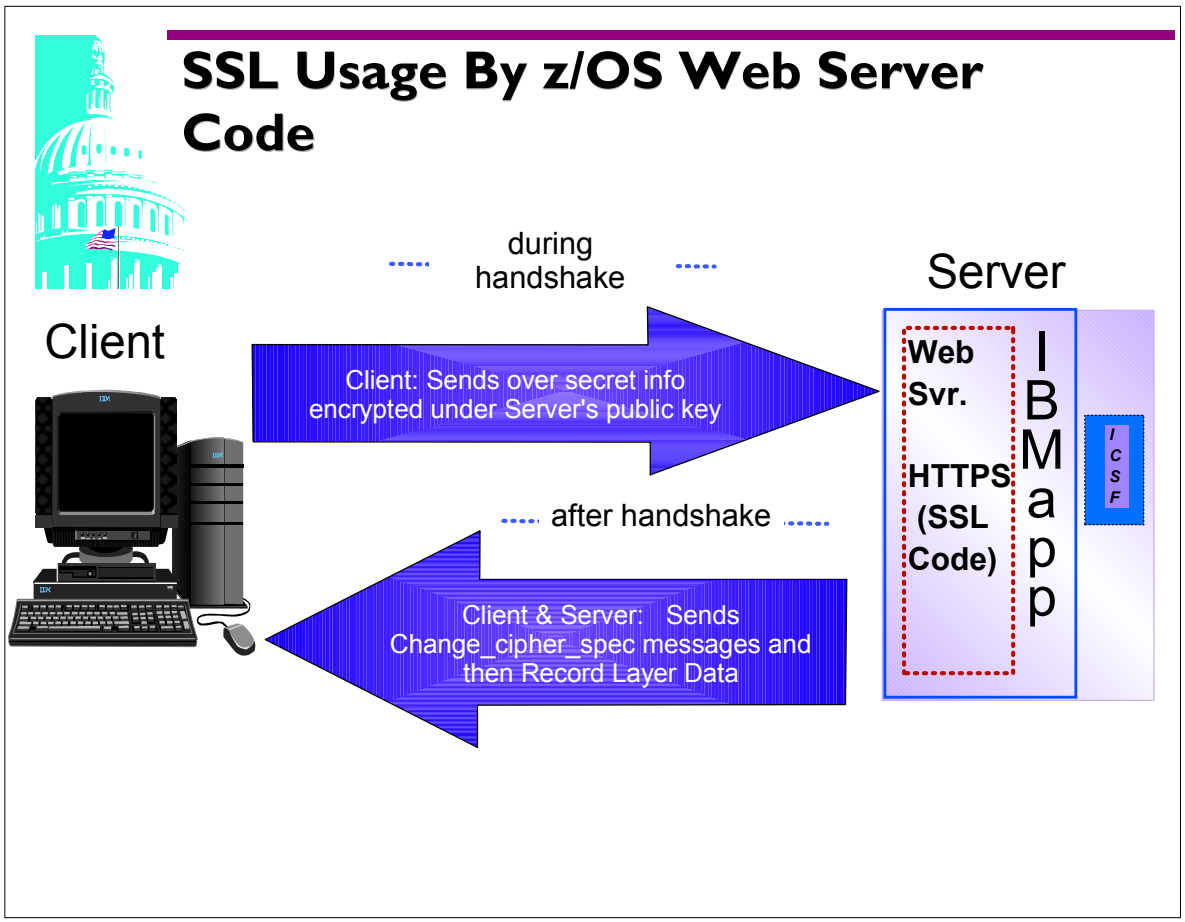


- A hash algorithm is processed against the data to be transmitted and a numeric value is generated. This value along with packaging information (algorithm used etc.) and the data is sent to the receiver. The receiver will run the data through the same algorithm used by the sender. If the numeric value is the same as the value sent, the receiver knows that the data was not changed or modified during transit. Generating the same numeric value for different data is not feasible.

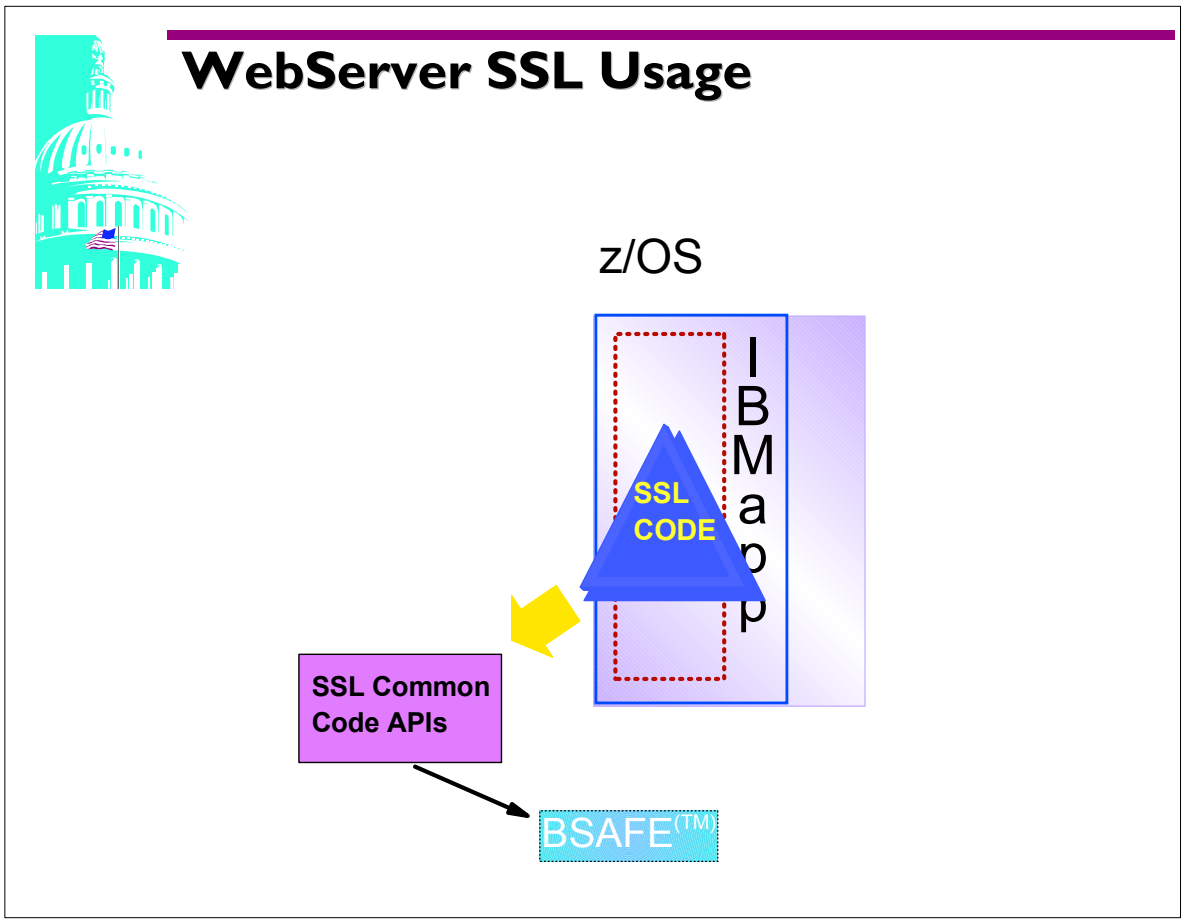
- Message Authentication Codes are hash functions with a private key. To create or verify the MAC, one must have the key. This is useful for verifying that hashes have not been tampered with during transmission.

- A digital Signature is the result of hash data that is encrypted by the senders private key. This provides non-repudiation, the proof that only the sender could have sent the data because only the sender has access to their private key.

SSL Usage and S/390 Crypto Hardware



- The server must decrypt the secret information sent by the client during the handshake phase.
- The client and server both encrypt their own messages before sending and each must decrypt messages received.

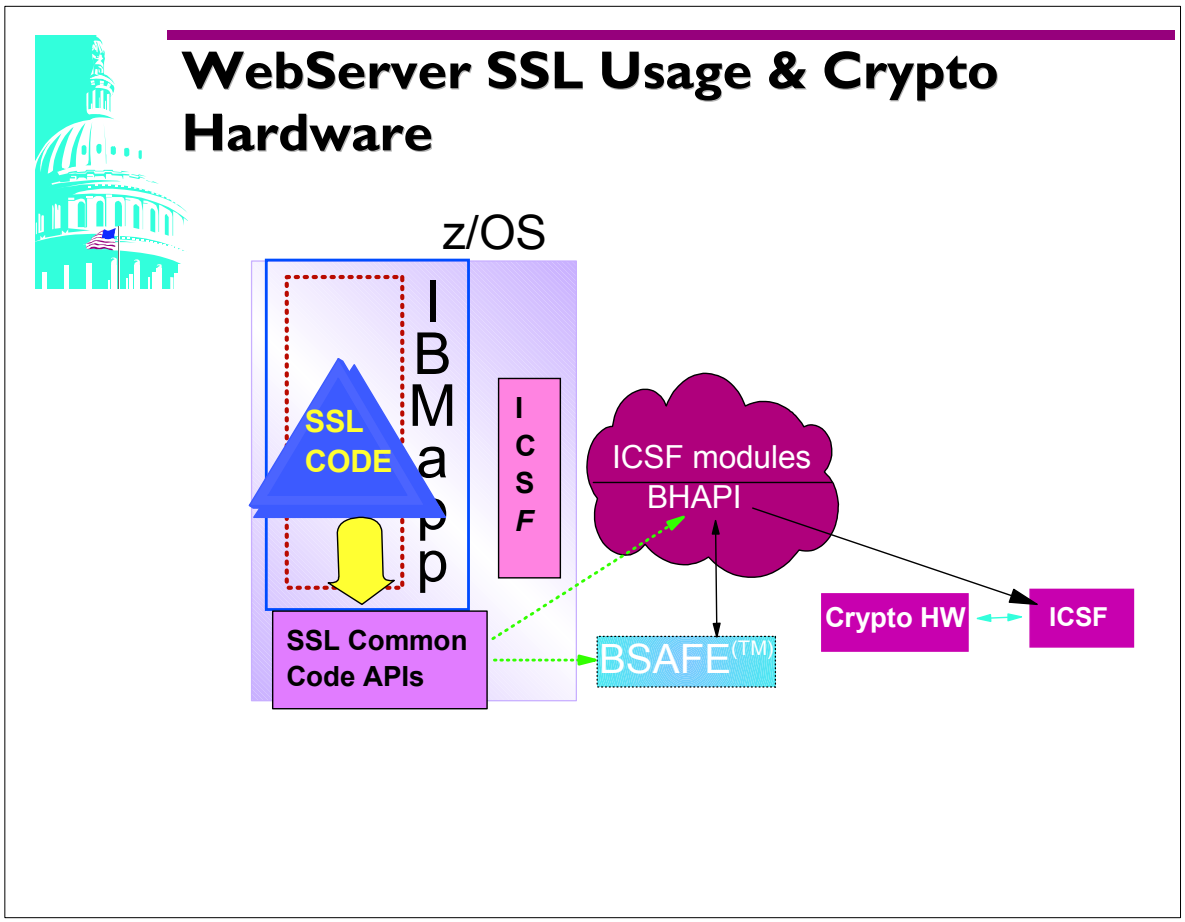


- Secure Sockets Layer, SSL, is the protocol used to securely transmit data on a channel. This type of transfer is recognized by HTTPS (Secure Hypertext Transfer Protocol). SSL is a application protocol independent sublayer under the "higher level" application protocol HTTP application layer.

- The SSL code performs those tasks that are dictated by the protocol.

- There are 2 basic sets of common code libraries used within IBM products to perform SSL: an Internal toolkit (GSkit) and System SSL. As of 2.10 the S/39 Web Server uses System SSL, earlier versions used GSkitt

- Those libraries use the RSA BSAFE(TM) toolkit to provide a software cryptographic engine in case hardware crypto and ICSF are not available.



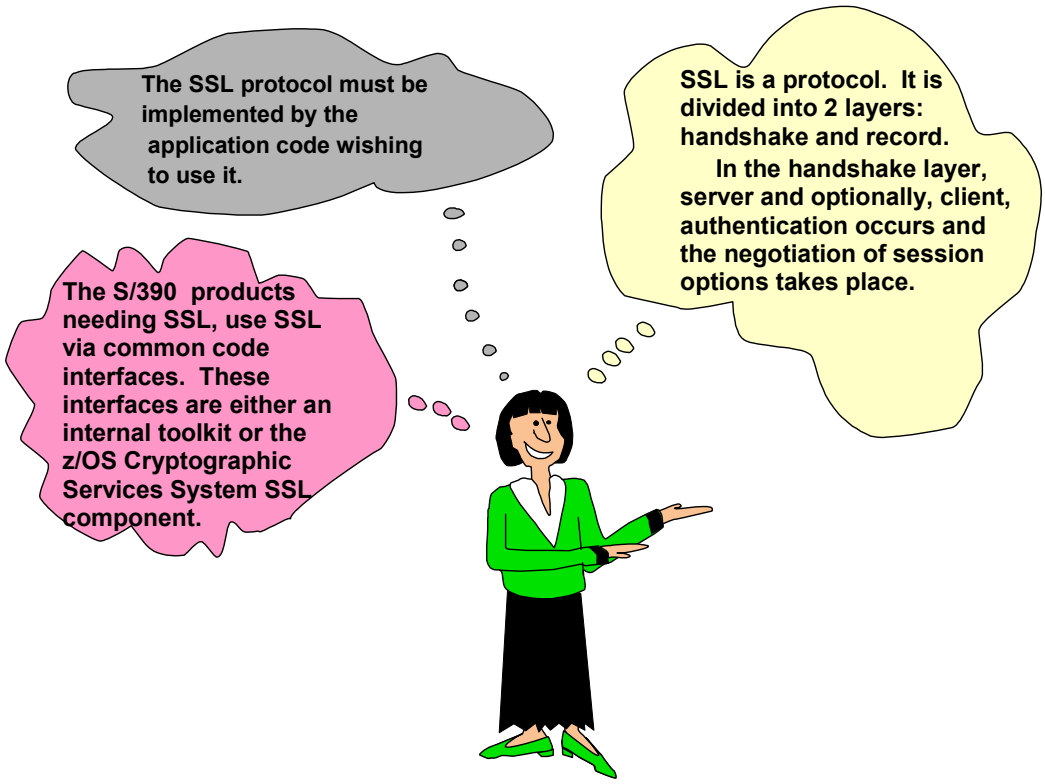
- ICSF provides a library (BHAPI) to interface with BSAFE and provide a limited number of ICSF APIs in the BSAFE programming structure.

- One of the APIs is used to determine whether the S/390 Crypto hardware and ICSF are available.

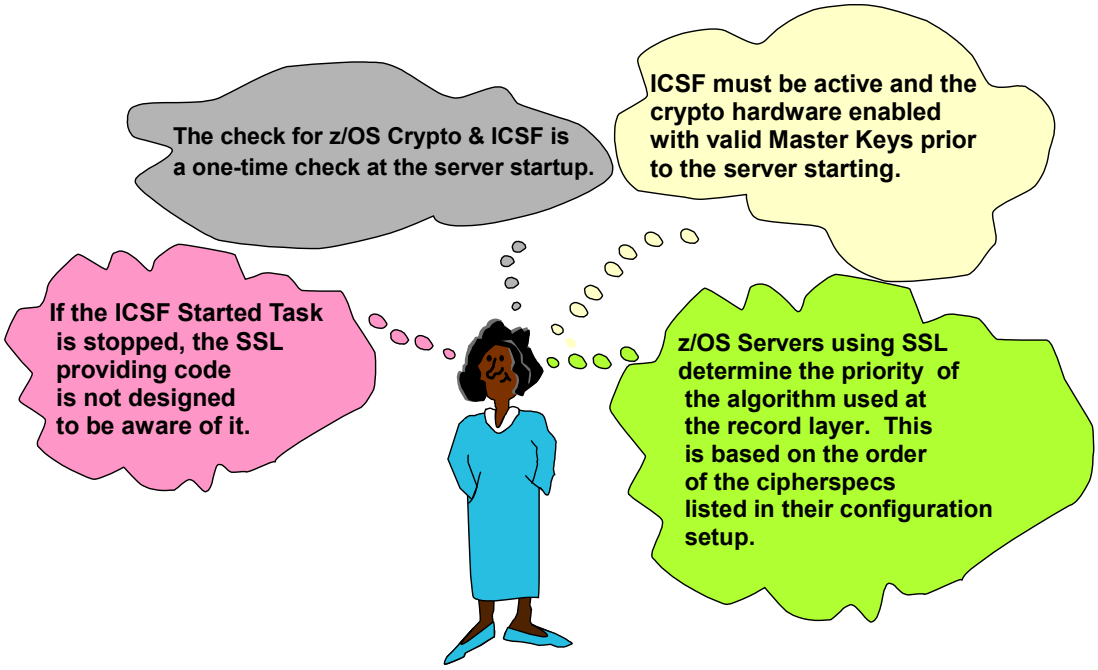
- If they are, the SSL code can chose whether a particular function is to be performed in hardware or software.

- If the Crypto Hardware is valid and ICSF is active, certain requests are sent to the IBM CCA APIs ICSF for processing on the Crypto hardware;
  - decrypt data from under the server's public key
  - negotiated cipherspec is DES or TDES the hardware can encrypt/decrypt using these keys
  - if cipherspec is NOT DES or TDES the requests are sent to BSAFE for processing on the software engine

Important Points to Remember ...



Important Points to Remember ...



Important Points to Remember ...

Reference

- HTTP://HOME.NETSCAPE.COM/ENG/SSL3/DRAFT302.TXT
  - SSL Protocol