Unit: Network Security

B.Tech VIII

NETWORK AND SYSTEM SECURITY (CORE ELECTIVE - 5) (CS424)

Book :

Computer Security: Principles and Practice
By William Stallings

Chapter 22: Internet Security Protocols and Standards
Chapter 23 Internet Authentication Applications
Chapter 24 Wireless Network Security

Contents

- **Chapter 22: Internet Security Protocols and Standards**
  - Secure E-Mail and S/MIME
  - DomainKeys Identified Mail
  - Secure Sockets Layer (SSL) and Transport Layer Security (TLS)
  - HTTPS
  - IPv4 and IPv6 Security

Network Security

- Network security provide protection to data against attacks while data is in transit on a network.
- To achieve this goal, many real-time security protocols have been designed.
- There are popular standards for real-time network security protocols such as S/MIME, SSL/TLS, SSH, and IPsec.
- These protocols work at different layers of networking model.

S/MIME and DKIM

Secure E-Mail and S/MIME

- **E-Mail:**
  - Electronic mail (E-mail) is **a computer-based application for the exchange of messages between users**.
  - E-mail is the electronic equivalent of a letter, but with advantages in timeliness and flexibility.
- **SMTP**
  - SMTP is the de facto standard responsible for sending email messages from one server to another over TCP/IP based networks.
  - SMTP is an **application layer protocol.**
  - The client who wants to send the mail opens a TCP connection to the SMTP server and then sends the mail across the connection.

Secure E-Mail and S/MIME

- **Drawback of SMTP**
  - SMTP protocol supports only 7-bit ASCII text communications that were both unauthenticated and unencrypted.
  - With only 94 printable characters in ASCII, the system cannot deal with binary files or characters in non-English languages that use different writing systems, accented letters, etc.
  - It does not accommodate sending video or audio data.
  - The default design of every SMTP server was an open mail relay that lets anyone send emails through it, not just those from or to known users.
  - These limitations made SMTP communications vulnerable to email spoofing, spamming, worms and man-in-the-middle (MitM) attacks.

Secure E-Mail and S/MIME

- **Email spoofing:**
  - Email Spoofing is creating and sending an email with a modified sender's address.
  - The sender's address is forged in such a way that the receivers will trust the email, thinking it has been sent by someone they know or from any trusted official source.
- **Email spamming**:
  - Email spam, also referred to as junk email, spam mail, or simply spam, is **unsolicited messages sent in bulk by email.**
- **Email worms**:
  - Warm emailing is **a targeted, personalized outreach strategy in which each email is handcrafted and written for one person only**

Secure E-Mail and S/MIME

- **Multipurpose Internet Mail Extensions(MIME):**
  - MIME is an extension of the SMTP (based on RFC 822 specification) of an Internet mail format.
  - RFC 822 defines a simple header with To, From, Subject, and other fields that can be used to route an e-mail message through the Internet and that provides basic information about the e-mail content.
  - MIME standard was developed to extend the functionality of SMTP and use characters sets other than ASCII.
  - It is an email application program that extends the email messages format to support more than just textual messages, such as audio, video, pictures, and so on.
  - MIME is not a mail protocol and cannot replace SMTP.

Secure E-Mail and S/MIME

- **SMTP vs. MIME**

SMTP	MIME
Simple Mail Transfer Protocol	Multipurpose Internet Mail Extensions
Part of TCP/IP protocol suit and controls the transmission of email message on the Internet	Email Application Program that extends the email message format
Supports ASCII text format only	Supports non-ASCII data through SMTP
Audio/Video attachment is not possible	Audio/Video attachment is possible
Follows RFC822 to format email message simple header with To, From, Subject, and other fields that can be used to route an e-mail message	Provides a number of new header fields that supports multi-media contents

S/MIME (Secure MIME)

- It is an extension of MIME.
- **S/MIME allows you to sign digitally your emails so that only the intended receiver can receive the emails.**
- S/MIME encrypts and digitally signs emails to ensure that the email is authenticated and its contents have not been altered in any way.
- S/MIME is defined as a set of additional MIME **content types** and provides the ability to sign and/or encrypt e-mail messages.

Content-Type

- Content Type is also known as **MIME (Multipurpose internet Mail Extension)** Type.
- It is an **HTTP header** that provides the description about what are you sending to the browser.
- List of Content Types
  - text/html
  - text/plain
  - application/msword
  - application/vnd.ms-excel
  - application/jar
  - application/pdf
  - audio/mp3
  - video/mp4
  - video/quicktime etc.

MIME Content Type

Type	Subtype	Description
Text	Plain	Unformatted text; may be ASCII or ISO 8859.
	Enriched	Provides greater format flexibility.
Multipart	Mixed	The different parts are independent but are to be transmitted together. They should be presented to the receiver in the order that they appear in the mail message.
	Parallel	Differs from Mixed only in that no order is defined for delivering the parts to the receiver.
	Alternative	The different parts are alternative versions of the same information. They are ordered in increasing faithfulness to the original, and the recipient's mail system should display the "best" version to the user.
	Digest	Similar to Mixed, but the default type/subtype of each part is message/rfc822.
Message	rfc822	The body is itself an encapsulated message that conforms to RFC 822.
	Partial	Used to allow fragmentation of large mail items, in a way that is transparent to the recipient.
	External-body	Contains a pointer to an object that exists elsewhere.
Image	jpeg	The image is in JPEG format, JFIF encoding.
	gif	The image is in GIF format.
Video	mpeg	MPEG format.
Audio	Basic	Single-channel 8-bit ISDN mu-law encoding at a sample rate of 8 kHz.
Application	PostScript	Adobe Postscript
	octet-stream	General binary data consisting of 8-bit bytes.

S/MIME Content Type

Type	Subtype	smime Parameter	Description
Multipart	Signed		A clear-signed message in two parts: one is the message and the other is the signature.
Application	pkcs7-mime	signedData	A signed S/MIME entity.
	pkcs7-mime	envelopedData	An encrypted S/MIME entity.
	pkcs7-mime	degenerate signedData	An entity containing only public-key certificates.
	pkcs7-mime	CompressedData	A compressed S/MIME entity.
	pkcs7-signature	signedData	The content type of the signature subpart of a multipart/signed message.

S/MIME Functions

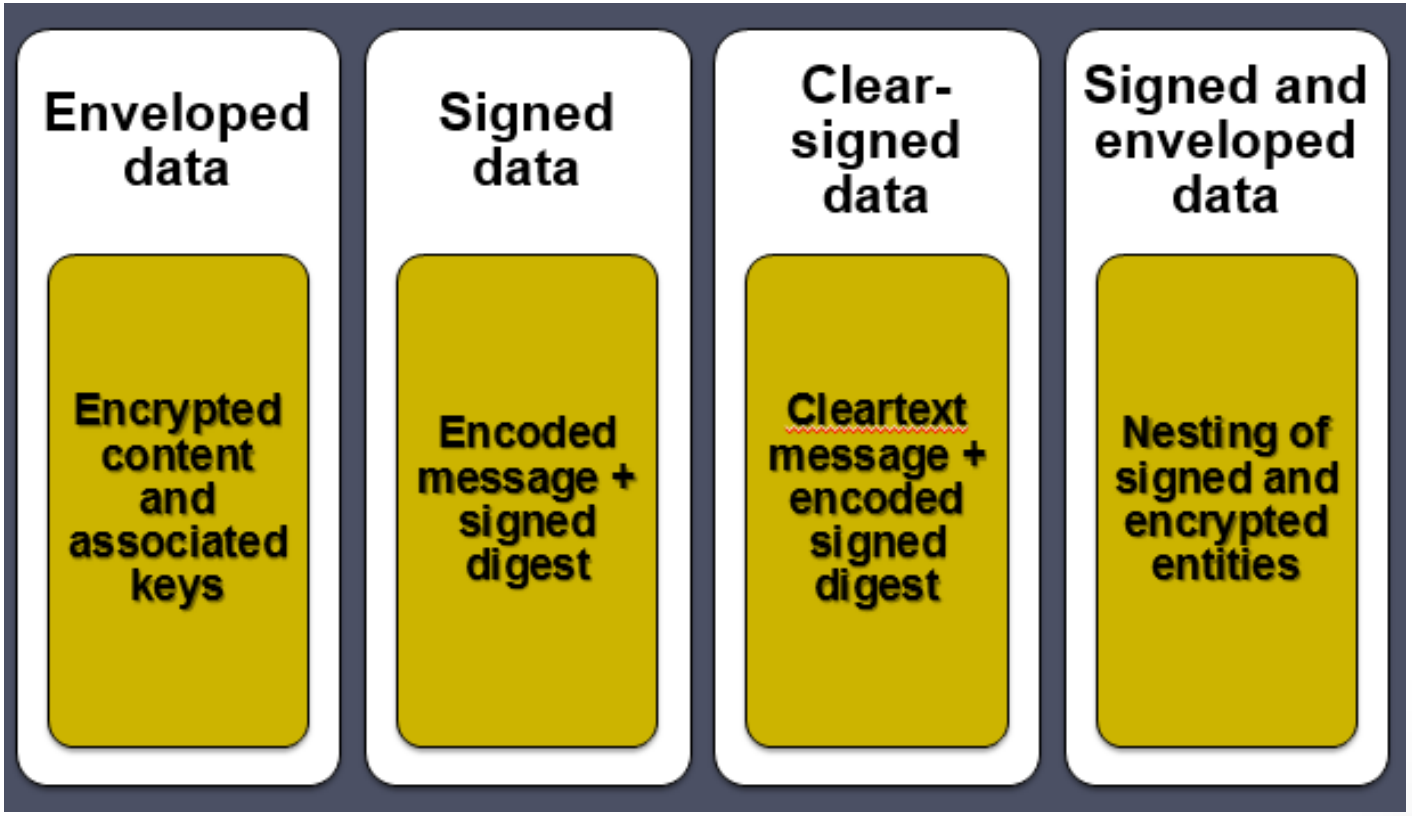
- S/MIME content-types support four new functions:

1. **Enveloped data**
   - This consists of encrypted content of any type and encrypted-content encryption keys for one or more recipients.
2. **Signed data**
   - A digital signature is formed by taking the message digest of the content to be signed and then encrypting that with the private key of the signer.
   - The content plus signature are then encoded using base64 encoding.
   - A signed data message can only be viewed by a recipient with S/MIME capability.

S/MIME Functions

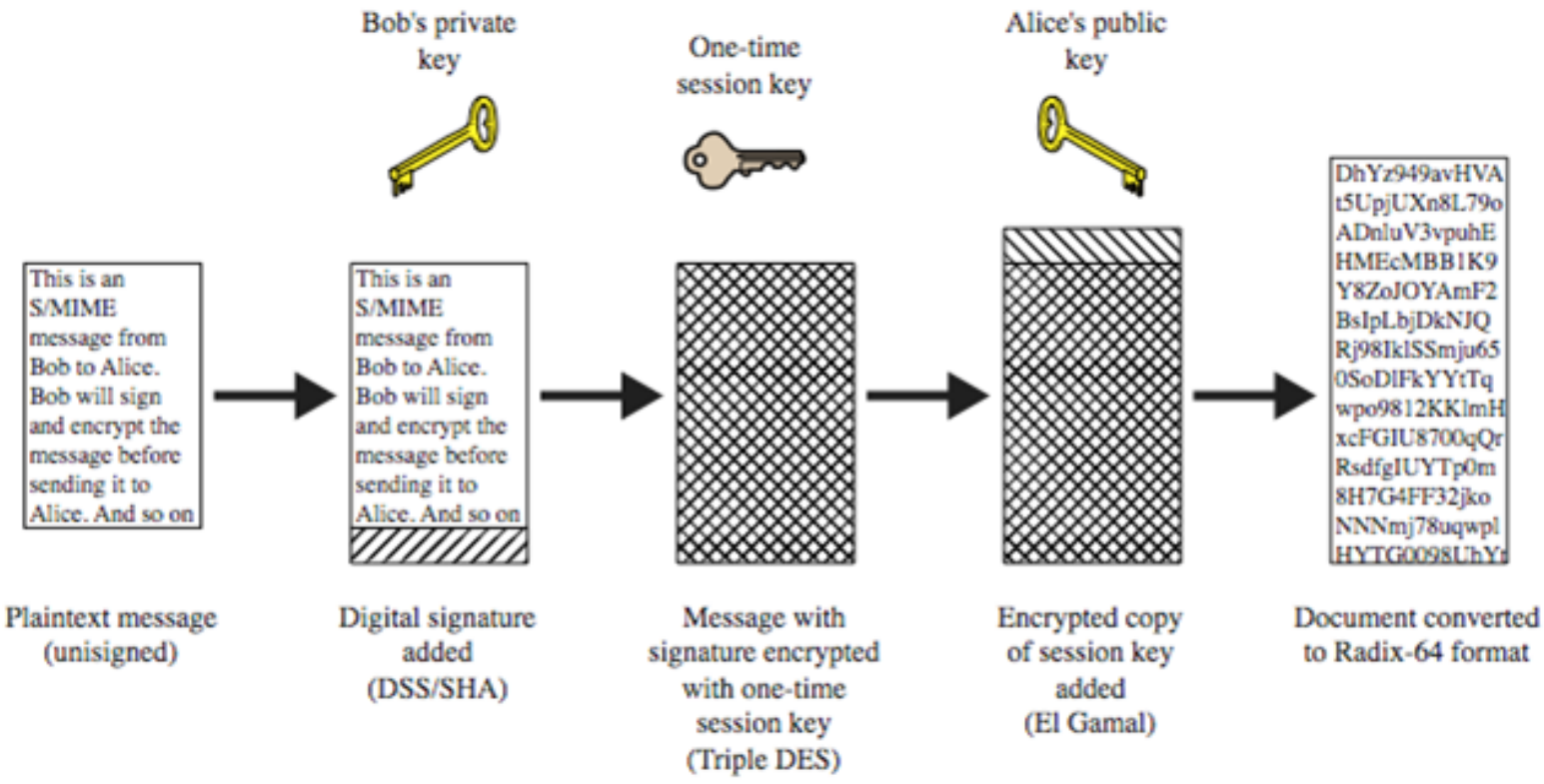
3. **Clear-signed data**
   - A digital signature of the content is formed.
   - Only the digital signature is encoded using base64.
   - As a result, recipients without S/MIME capability can view the message content, although they cannot verify the signature.
4. **Signed & enveloped data**
   - Signed-only and encrypted-only entities may be nested, so that encrypted data may be signed and signed data or clear-signed data may be encrypted.

S/MIME Functions (Summary)



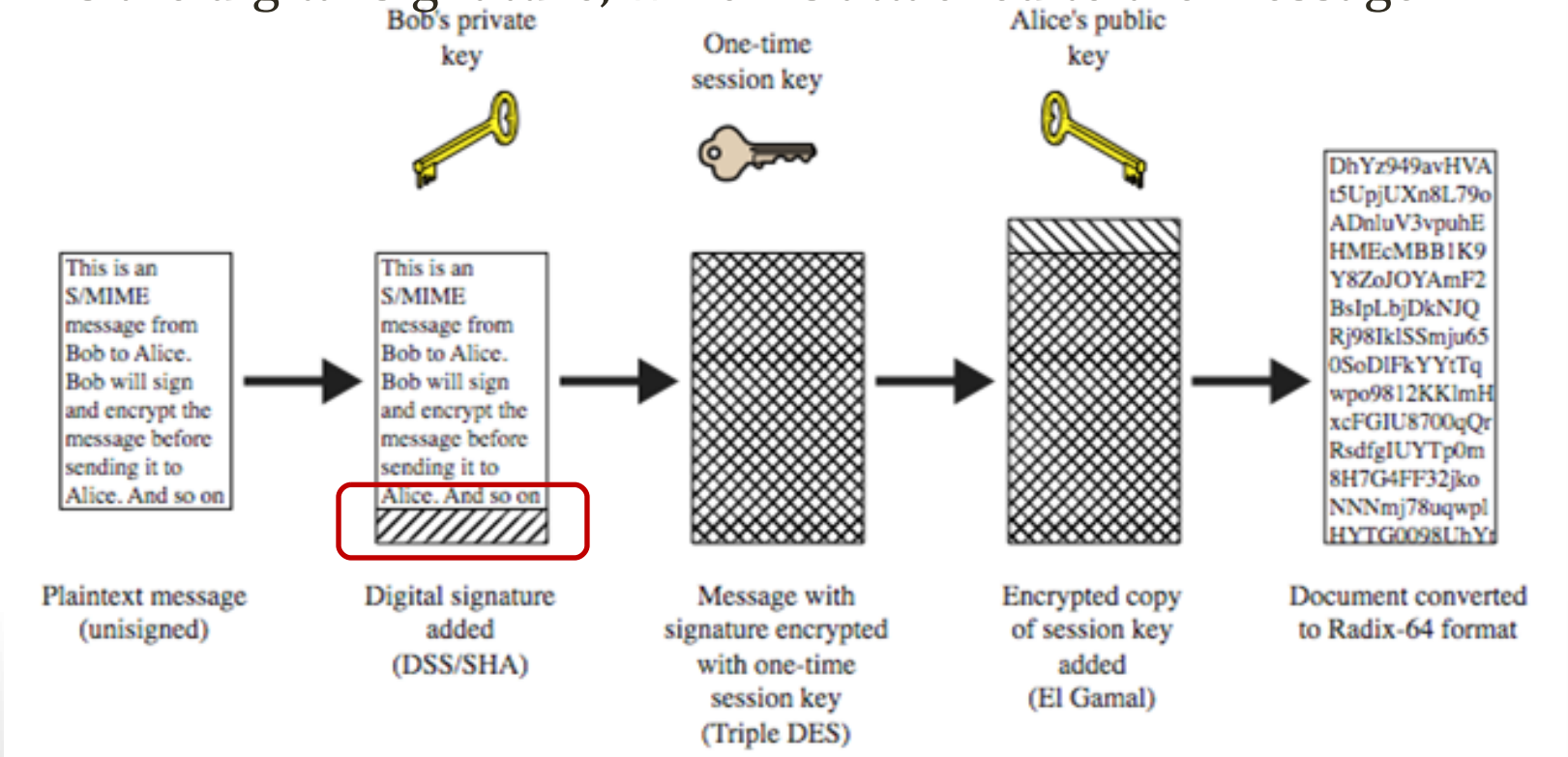
S/MIME Process

- **Steps**
- Assume a sender (Bob) sends email to receiver (Alice).



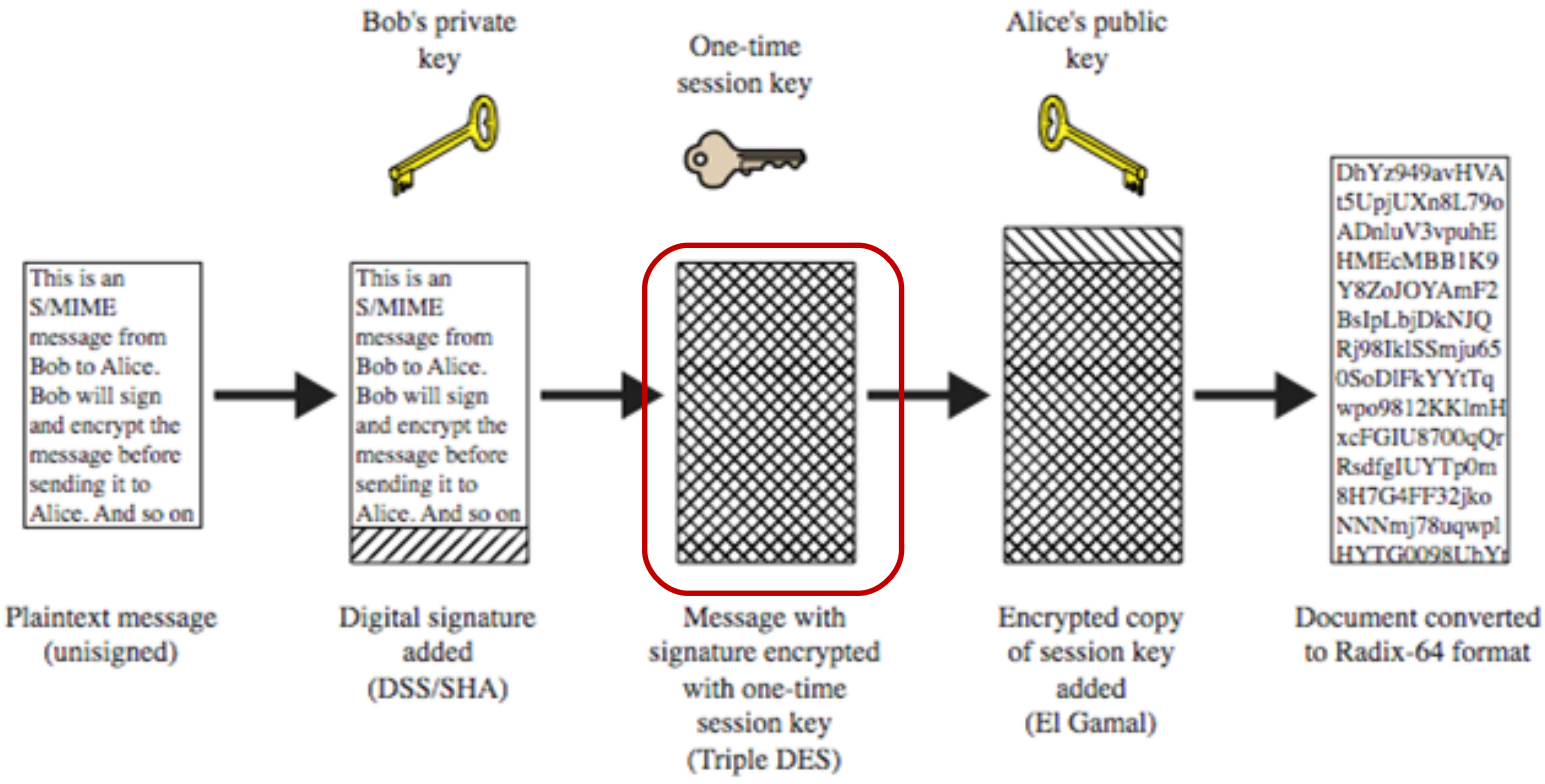
S/MIME Process

- **Steps**
- From an input message, S/MIME generates a 160 bit digest by applying SHA-1 (Secure Hash Algorithm).
- S/MIME encrypts the digest using DSS (the Digital Signature Standard) and the sender's (Bob) private DSS key. The result is the digital signature, which is attached to the message



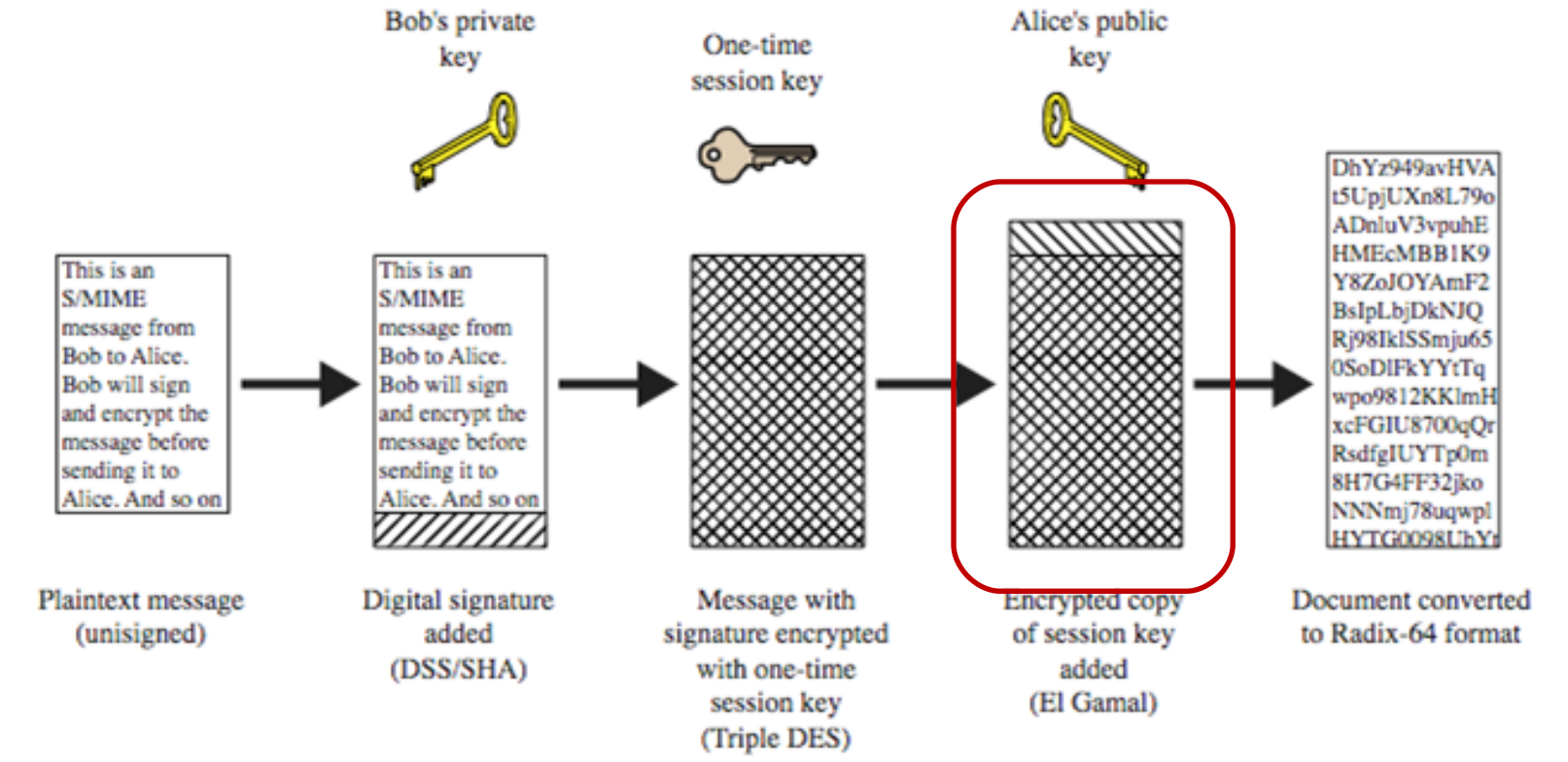
S/MIME Process

- **Steps**
- The Message with the signature is encrypted with one-time session key (secret key randomly selected by sender) by applying symmetric key algorithm (Triple DES(Data encryption standard)).



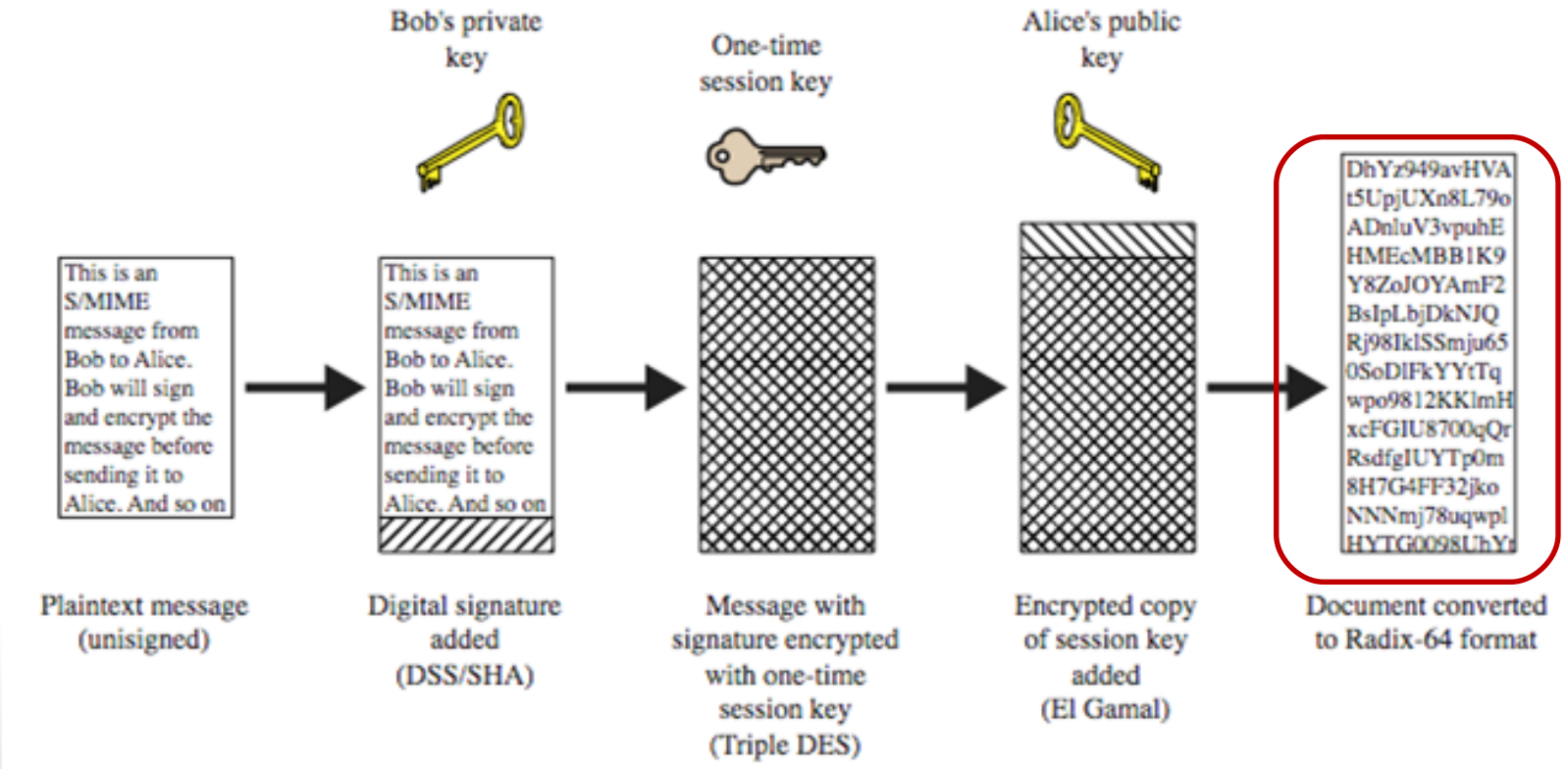
S/MIME Process

- **Steps**
- Further the session key is encrypted using receiver's (Alice) public key by applying public key algorithm (RSA/Elgamal).
- This encrypted session key is attached with the encrypted message.



S/MIME Process

- **Steps**
- To protect integrity of data, either the signature alone or the signature plus the message are mapped into printable ASCII characters using a scheme known as radix-64 or base64 mapping.



S/MIME Process

- On the receiving end, S/MIME uses the receiver's private ElGamal key to recover the secret key and then uses the secret key and 3DES to recover the plaintext message.
- If encryption is used alone, radix-64 is used to convert the ciphertext to ASCII format.

S/MIME Cryptographic Algorithms

- Digital signatures: DSS & RSA
- Hash functions: SHA-1 & MD5
- Session key encryption: ElGamal & RSA
- Message encryption: AES, 3DES, etc
- MAC: HMAC with SHA-1
- Must map binary values to printable ASCII
  - use radix-64 or base64 mapping

S/MIME Public Key Certificates

- S/MIME has effective encryption and signature services
- But also need to manage public-keys
- S/MIME uses X.509 v3 certificates
- Each client has a list of trusted CA's certs
- And own public/private key pairs & certs
- Certificates must be signed by trusted CA's

DomainKeys Identified Mail (DKIM)

- **DomainKeys Identified Mail (DKIM)**
  - A specification for cryptographically signing e-mail messages
  - permits a signing domain to claim responsibility for a message
  - Message recipients (or agents acting in their behalf) can verify the signature.
  - Verification is done by signer’s public key.
  - Provides a digital signature at email level that is only used by, and visible to, the recipient's email service provider.
  - This DKIM digital signature is not visible or usable by the actual recipient.

DomainKeys Identified Mail (DKIM)

- The objective of DKIM is to **protect against forged emails**.
- The receiver uses DKIM signatures in message headers to differentiate legitimate email from fraudulent email, such as business email compromise and phishing scams.
- DKIM is a proposed Internet Standard (RFC 4871)
- DKIM has been widely adopted by a range of e-mail providers, including **corporations, government agencies, gmail, yahoo, and many Internet service providers (ISPs).**

DomainKeys Identified Mail (DKIM)

- **DKIM vs. S/MIME** (Theoretically)
- **S/MIME**
  - Enables end users to authenticate messages.
  - Digital signatures are incorporated into the message body.
  - While authenticate the message body, S/MIME signature does not mention anything about the system (domain) sending the message.
- **DKIM**
  - DKIM signatures are incorporated into the message headers of authenticated emails, not generated or authenticated by end users.
  - DKIM signatures are generated by the sending mail server and authenticated by the receiving mail server.

DomainKeys Identified Mail (DKIM)

- DKIM works along with the following protocols
  - **SPF (Sender Policy Framework)**
  - **DMARC (Domain-based Message Authentication, Reporting and Conformance)**
- DKIM (with SPF and DMARC) helps email providers and enterprises in validating their emails.
- It prevents domain high jacking which is used for phishing scams, email spoofing and other malicious email-based attacks.

SPF, DMARC and DKIM

- **SPF(Sender Policy Framework)**
  - It authenticates the sending email server to send email on behalf of the organization's domain.
  - Records information about authorized email servers in the sender's **domain name system (DNS) records**, which are accessible to any internet-connected system.
  - Used by all major email service providers such as **Google Gmail, Yahoo email services and others -- that send and receive email for many different organizations.**

SPF, DMARC and DKIM

- **DKIM(DomainKeys Identified Mail)**
  - Enables an email-sending organization to digitally sign each individual message sent by an authorized email server.
  - Relies on SPF to identify whether a message was sent by an authorized email server and relies on DMARC to determine the appropriate policy when an email message fails authentication.

SPF, DMARC and DKIM

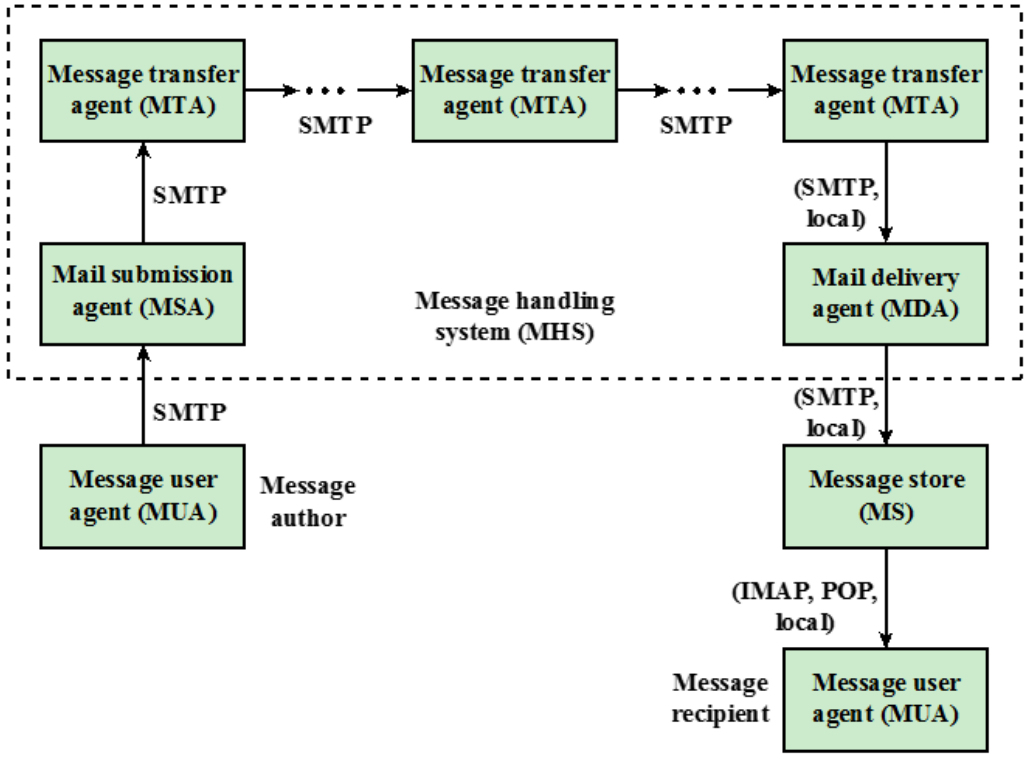
- **DMARC(Domain-based Message Authentication, Reporting and Conformance)**
  - Enables an email-sending organization to specify the procedures which the receivers should take when they receive email that has not been authenticated.
  - When an email server not listed in the SPF record attempts to send a message or when DKIM authentication fails, DMARC policies provide guidance to the receiver as to whether the message should be delivered, quarantined -- i.e., sent to spam folder -- or rejected entirely.

Internet Mail Architecture

- Internet mail architecture consists of
  - **Message User Agents (MUA)**
  - **Message Handling System/Service (MHS)**
    - composed of Message Transfer Agents (MTA)
    - accepts a message from one user and delivers it to one or more other users, creating a virtual MUA-to-MUA exchange environment.
  - Supports three types of interoperability:
    - **MUA-to-MUA (Directly between users)**
    - **MUA-to-MHS(MTA) (Between user and MHS)**
    - **MTA-to-MTA (Between MTA and MTA under MHS)**

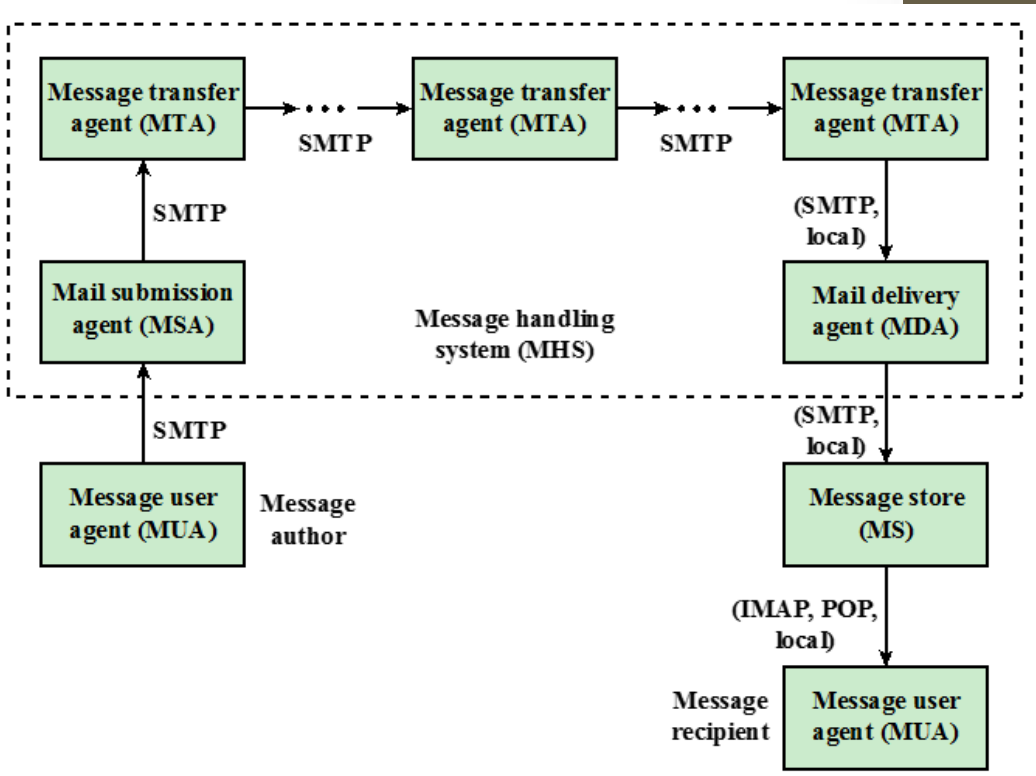
Internet E-Mail Architecture

- **Message User Agent (MUA)**:
  - Works on behalf of user actors and user applications within the e-mail service.
  - Formats message and performs initial submission into MHS via MSA.
  - The recipient MUA processes the received mail for storage and/or display to the recipient user.



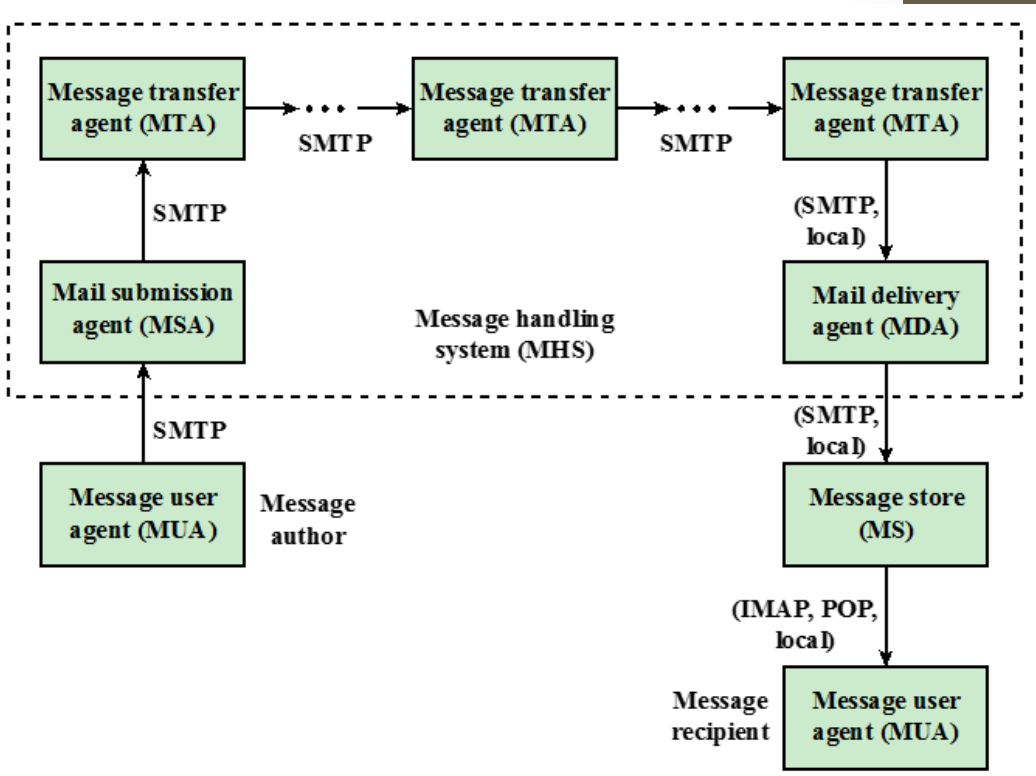
Internet E-Mail Architecture

- **Mail Submission Agent (MSA)**:
  - Accepts message from MUA
  - Enforces policies of the hosting domain and the requirements of Internet standards.
  - It may be along with MUA or separate functionality (using SMTP).



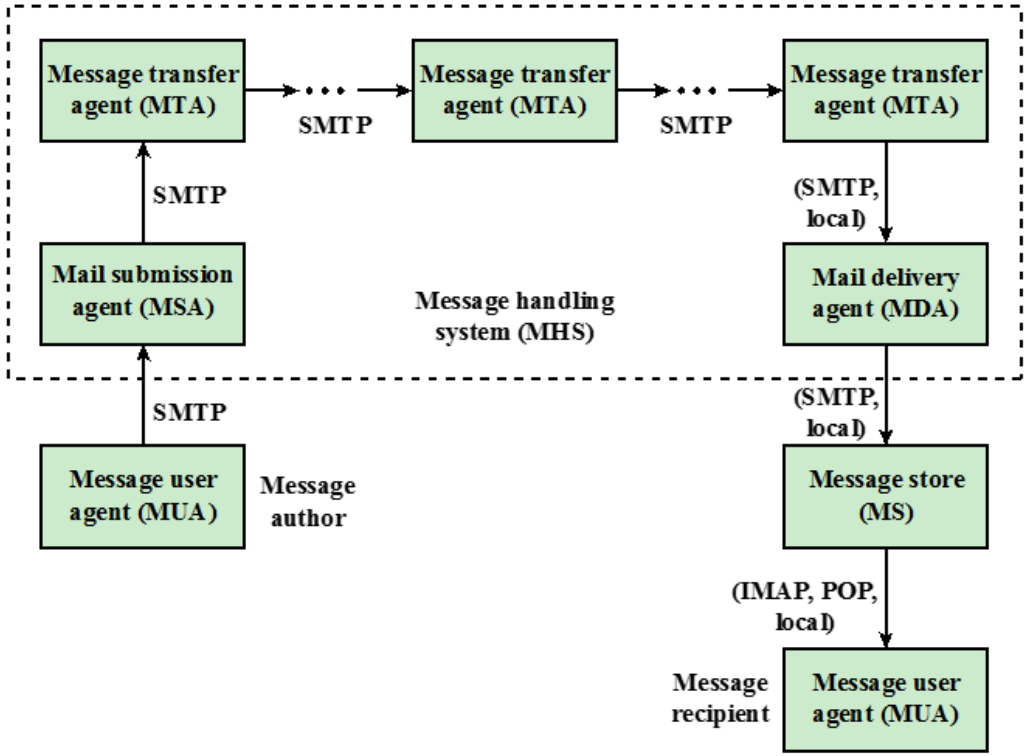
Internet E-Mail Architecture

- **Message Transfer Agent (MTA)**:
  - Like a packet switch or IP router to make mail routing assessments.
  - Relaying is performed by a sequence of MTAs until the message reaches a destination MDA.
  - Adds trace information to the message header.
  - SMTP is used between MTAs and between MTA and MSA.



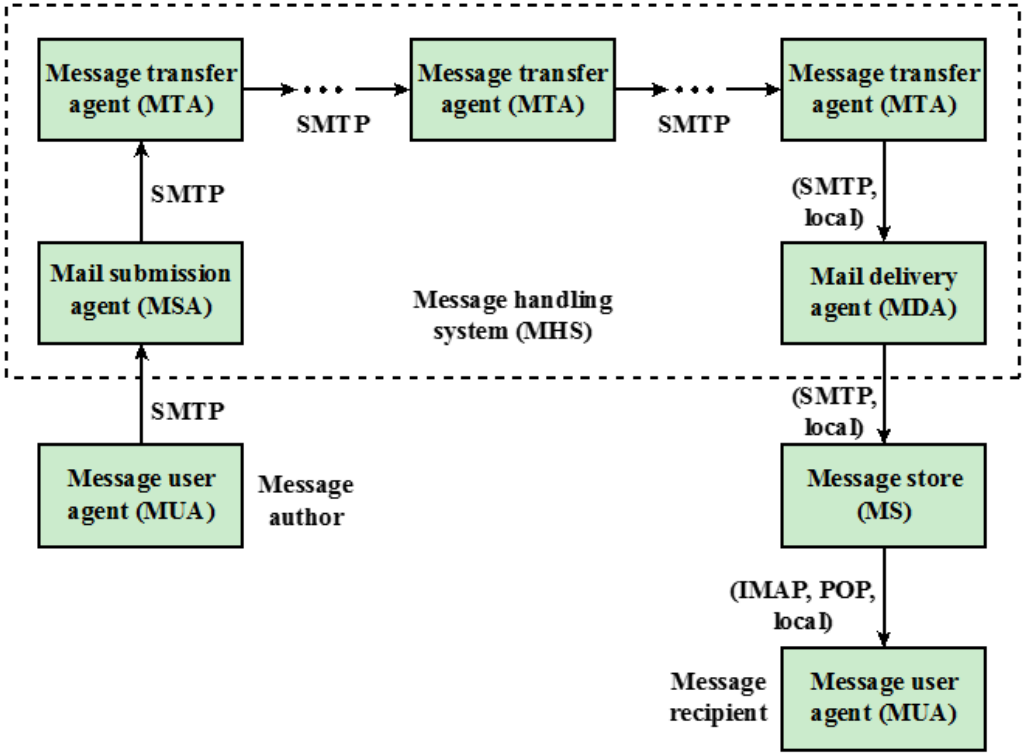
Internet E-Mail Architecture

- **Mail Delivery Agent (MDA)**:
  - Responsible for transferring the message from the MHS to the MS.



Internet E-Mail Architecture

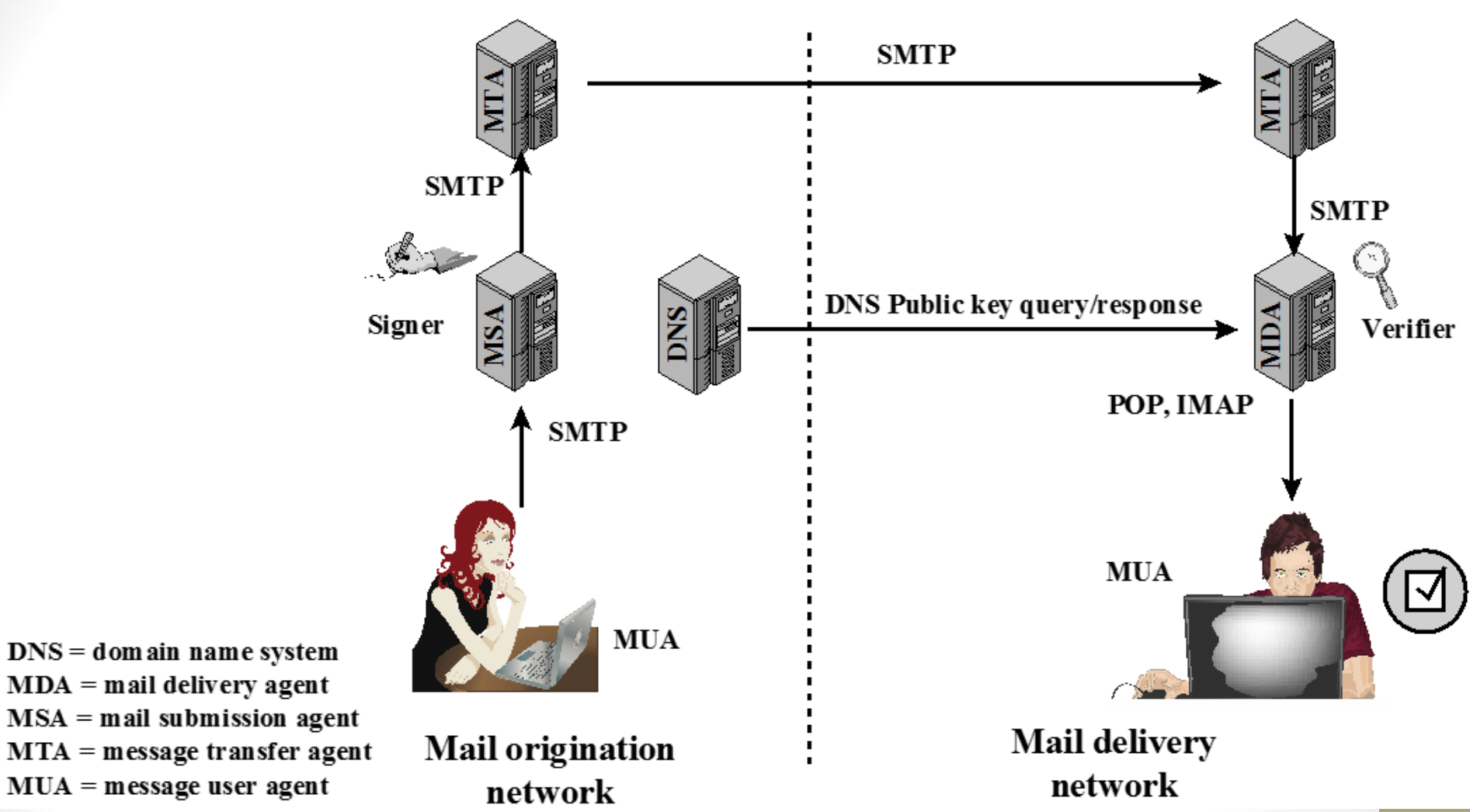
- **Message store (MS)**:
  - An MUA can employ a long-term MS.
  - MS can be located on a remote server or on the same machine as the MUA.
  - Between MUA and MS, messag
    Communication is done using
    POP (Post Office Protocol) or
    IMAP (Internet Message
    Access Protocol).



DKIM Strategy

- DKIM provides an e-mail authentication technique that is transparent to the end user.
- A user's e-mail message is signed by a private key of the administrative domain from which the e-mail originates.
- The signature covers all of the content of the message and some of the RFC 5322 message headers(Internet Message Format)
- At receiving end, MDA can access the corresponding public key via a DNS and verify the signature, thus authenticating that the message comes from the claimed administrative domain.

DKIM Deployment



DKIM vs. S/MIME

- S/MIME depends on both sender and receiver users using S/MIME.
- S/MIME only signs the contents; email header may be compromised.
- DKIM is not implemented in client programs (MUAs) – transparent to the users.
- DKIM applies to all emails.
- DKIM allows good senders to prove they sent a particular message.

SSL and TLS

Internet Security Protocols and Standards

- For TCP/IP protocol based network, physical and data link layers are typically implemented in the user terminal and network card hardware.
- TCP and IP layers are implemented in the operating system.
- Anything above TCP/IP is implemented as user process.

- **SSL/TLS are protocols above Transport Layer and offer security to data before they have been forwarded to Transport Layer.**

Why Transport Layer Security?

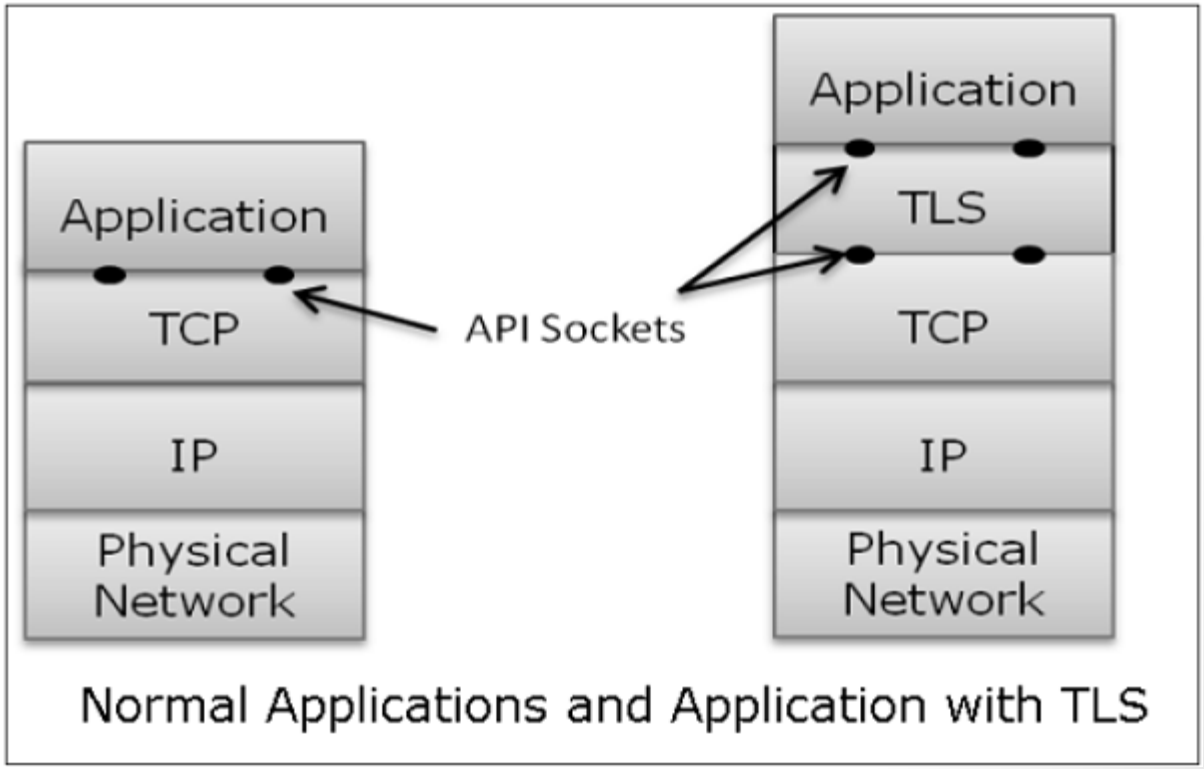
- **Assume a user visit to some e-commerce website and purchase some goods. He has already done the payment via credit card. Then he simply waits for the delivery of the purchased items.**
- Now the problem may be
  - If transactions did not use **confidentiality (encryption)**, an attacker could obtain user's payment card information. The attacker can then make purchases at user's expense.
  - If no data **integrity** measure is used, an attacker could modify the user's order in terms of type or quantity of goods.
  - if no server **authentication** is used, a server could display the authenticated e-commerce website logo. After receiving the user's order, he could take user's money and flee. Or he could carry out an identity theft by collecting user's name and credit card details.
- So **Security (confidentiality, integrity, authentication) is necessary for critical data communication.**

Why Transport Layer Security?

- **Transport layer security schemes**
  - can address these problems
  - enhances TCP/IP based network communication with confidentiality, data integrity, server authentication, and client authentication.
  - used to secure HTTP based web transactions on a network.
  - can be employed by any application running over TCP.

Transport Layer Security (TLS) Design

- **Transport Layer Security protocols operate above the TCP layer.**
  - **SSL(Secure Socket Layer)/TLS(Transport Layer Security)**
  - Design of these protocols use popular Application Program Interfaces (API) to TCP, called "**sockets**" for interfacing with TCP layer.



Why TLS popular?

- A major reason behind security at transport layer is **simplicity** in design.
- No need to do any changes in TCP/IP protocols that are implemented in an operating system.
- Only user processes and applications needs to be designed/modified which is less complex.

Secure Socket Layer(SSL)

- **History**
  - **Netscape** developed **SSLv2** and used in Netscape Navigator 1.1 in 1995.
  - The SSL version1 was never published and used.
  - Further up gradation concerning security flaws in SSLv2 is performed by Netscape and developed SSLv3.
  - The Internet Engineering Task Force (IETF) subsequently, introduced a similar TLS (Transport Layer Security) protocol as an open standard.
  - TLS protocol is non-interoperable with SSLv3.

Salient Features of SSL

- SSL provides network connection security through –
  - **Confidentiality** – Information is exchanged in an encrypted form.
  - **Authentication** – Communication entities identify each other through the use of digital certificates. Web-server authentication is mandatory whereas client authentication is kept optional.
  - **Reliability** – Maintains message integrity checks.
- SSL is available for all TCP applications.
- Supported by almost all web browsers.
- Provides ease in doing business with new online entities.
- Developed primarily for Web e-commerce.

Transport Layer Security(TLS)

- IETF released **Transport Layer Security (TLS)** protocol in January 1999 as Internet Standard in RFC 5246.
- **Features**
  - TLS protocol has same objectives as SSL.
  - It enables client/server applications to communicate in a secure manner by authenticating, preventing eavesdropping and resisting message modification.
  - TLS protocol sits above the reliable connection-oriented transport TCP layer in the networking layers stack.
  - Though SSLv3 and TLS protocol have similar architecture, several changes were made in architecture and functioning particularly for the handshake protocol.

SSL vs. TLS

SSL(Secure Socket Layer)	TLS (Transport Layer Security)
More complex than TLS	Simple
Less secured as compared to TLS	Provides high security
Less reliable and slower	Highly reliable and upgraded
Depreciated	Widely used
Uses port to set up explicit connection	Uses protocol to set up implicit connection
SSL handshake involves several round trips as authentication and key exchange take place. So increases latency.	TLS handshake involves a single round trip. Less latency than SSL and so faster than SSL.

TLS Architecture

- TLS/SSL uses TCP to provide a reliable end-to-end secure service.
- TLS/SSL is not a single protocol but rather two layers of protocols.
  - **Lowe sub-layer** includes SSL Record Protocol that provides integrity and confidentiality services.
  - **Upper sub-layer** comprises of three SSL-related protocol components and an application protocol.
  - Application component provides the information transfer service between client/server interactions.
  - Three SSL related protocol components are – **SSL Handshake Protocol, Change Cipher Spec Protocol, Alert Protocol**.

TLS Architecture

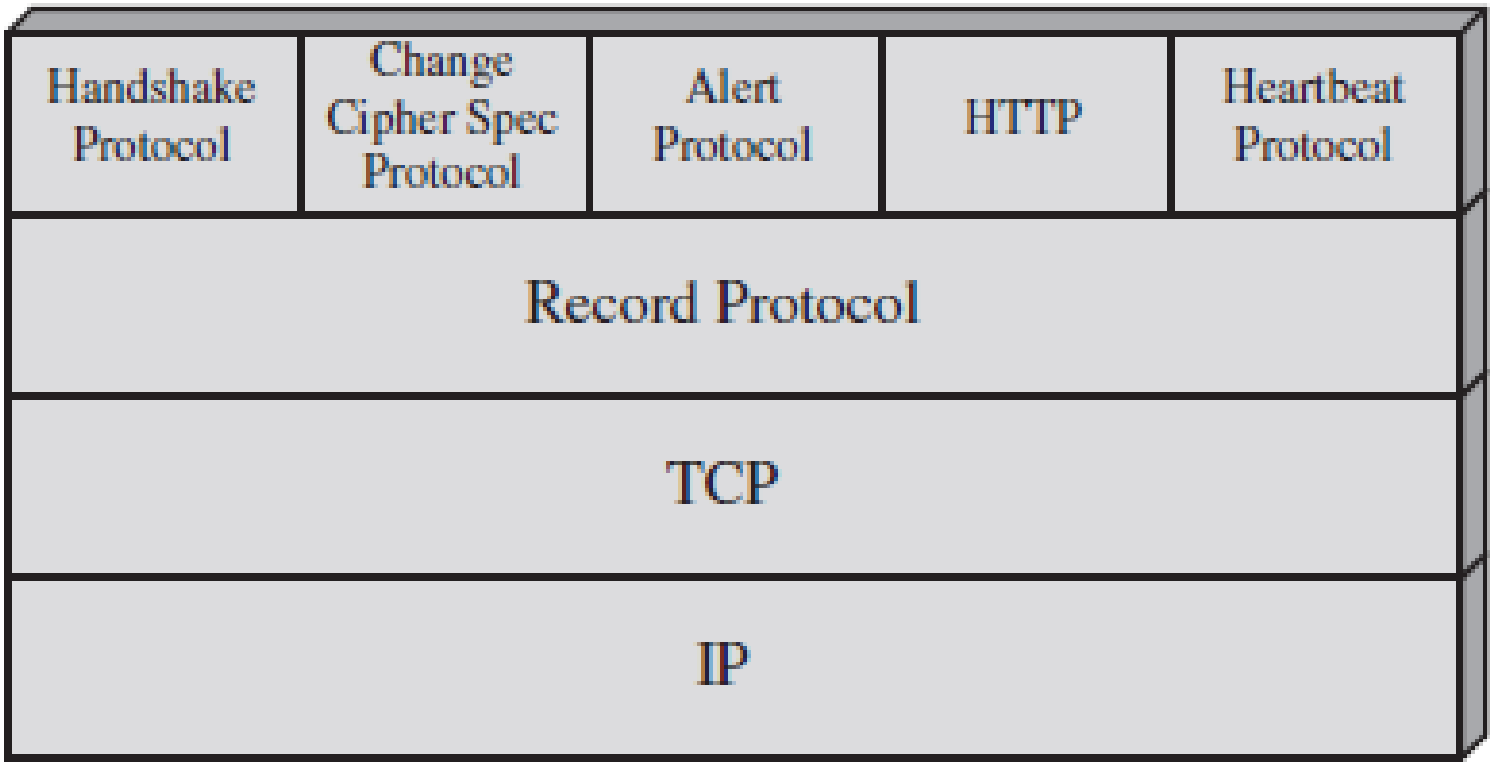


Figure 22.4 **SSL/TLS Protocol Stack**

TLS Architecture

- **TLS Record Protocol**
  - Provides basic security (integrity and confidentiality) services to various higher-layer protocols.
  - In particular, the Hypertext Transfer Protocol (HTTP), which provides the transfer service for Web client/server interaction, can operate on top of TLS.
- **Three higher-layer protocols**
  - **Handshake Protocol, Change Cipher Spec Protocol, and Alert Protocol.**
  - These TLS-specific protocols are used in the management of TLS exchanges.

TLS Session and TLS Connection

- **TLS Session**
  - An association between a client and a server.
  - Created by the **Handshake Protocol**.
  - Defines a set of cryptographic security parameters, which can be shared among multiple connections.
  - Used to avoid the expensive negotiation of new security parameters for each connection.
- **TLS Connection**
  - A transport (in the OSI layering model definition) that provides a suitable type of service.
  - For TLS, such connections are peer-to-peer relationships.
  - They are transient. Every connection is associated with one session.
- **Between any pair of parties (applications such as HTTP on client and server), there may be multiple secure connections.**

TLS Protocols

- **Record Protocol** : The SSL Record Protocol provides two services for SSL connections:
  - **Confidentiality**: The **Handshake Protocol** defines a shared secret key that is used for symmetric encryption of SSL payloads.
  - **Message integrity**: The **Handshake Protocol** also defines a shared secret key that is used to form a message authentication code (MAC).

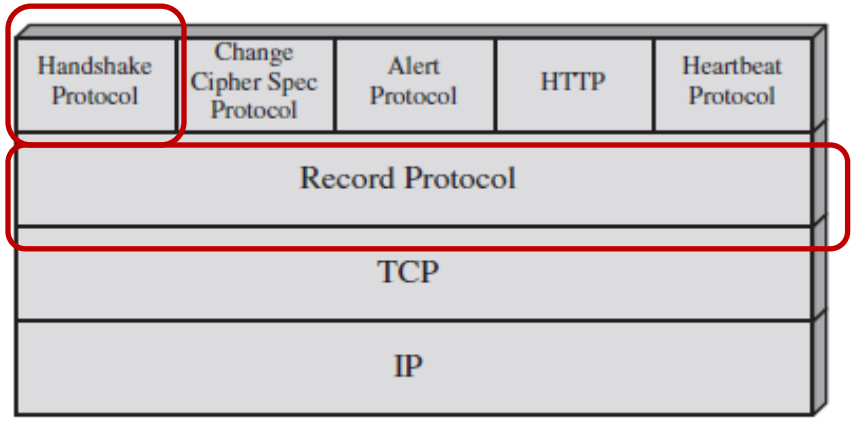


Figure 22.4 **SSL/TLS Protocol Stack**

TLS Protocols

- **Record Protocol** :
  - Overall Operation

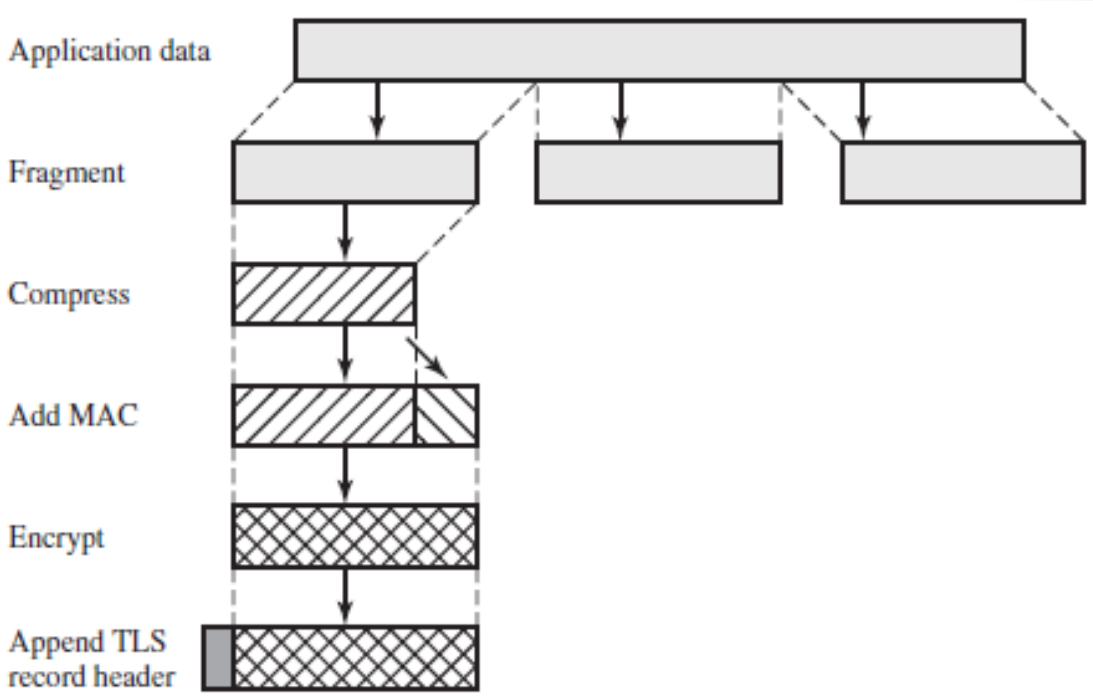


Figure 22.5 **TLS Record Protocol Operation**

- Once TLS unit is ready, the Record Protocol transmits it to TCP layer.
- Received data are decrypted, verified, decompressed, and reassembled, and then delivered to higher-level users.

TLS Protocols

- **Change Cipher Spec Protocol**:
  - Uses the TLS Record Protocol, and it is the simplest.
  - It comprises of a **single message(1 byte)** exchanged between two communicating entities, the client and the server.
  - As each entity sends the ChangeCipherSpec message, it changes its side of the connection into the **secure state** as agreed upon.
  - The cipher parameters pending state is copied into the current state.
  - Exchange of this Message indicates all future data exchanges are encrypted and integrity is protected.

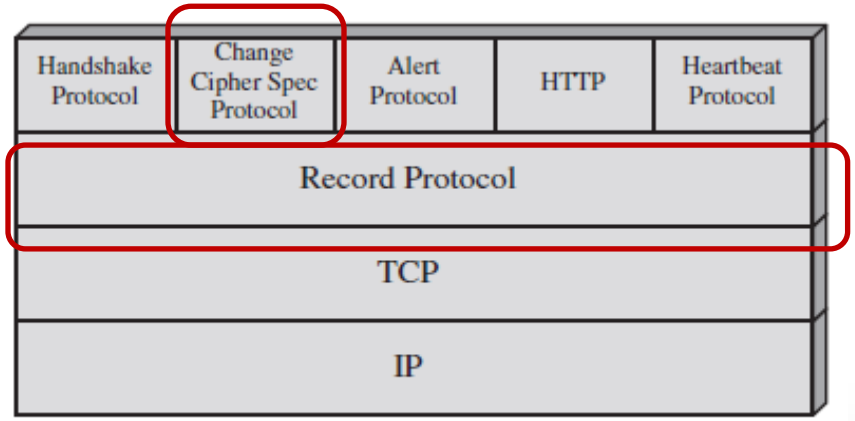


Figure 22.4 **SSL/TLS Protocol Stack**

TLS Protocols

- **Alert Protocol**:
  - This protocol is used to report errors – such as **unexpected message, bad record MAC, security parameters negotiation failed**, etc.
  - It is also used for other purposes – such as **notify closure of the TCP connection, notify receipt of bad or unknown certificate**, etc.
  - Each message in this protocol consists of **two bytes**.
  - **First byte : Values - warning(1) or fatal(2)** to convey the severity of the message.
  - If the level is fatal, TLS immediately terminates the connection. Other connections on the same session may continue, but no new connections on this session may be established.
  - **Second byte :** contains a code that indicates the specific alert.
  - An example of a fatal alert is an incorrect MAC.

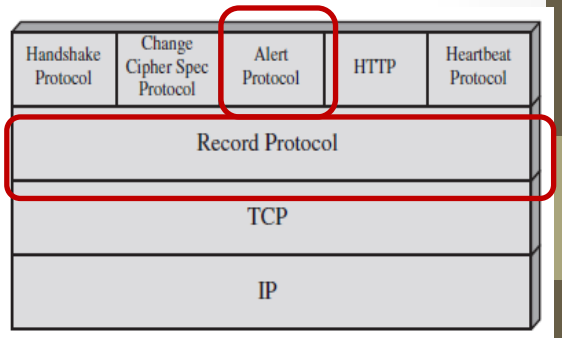


Figure 22.4 SSL/TLS Protocol Stack

TLS Protocols

- **Handshake Protocol**:
  - It is the most complex part of TLS/SSL.
  - It is invoked before any application data is transmitted.
  - It creates SSL sessions between the client and the server.
  - Establishment of session involves **Server authentication, Key and algorithm negotiation, Establishing keys and Client authentication (optional).**
  - A session is identified by unique set of cryptographic security parameters.
  - Multiple secure TCP connections between a client and a server can share the same session.

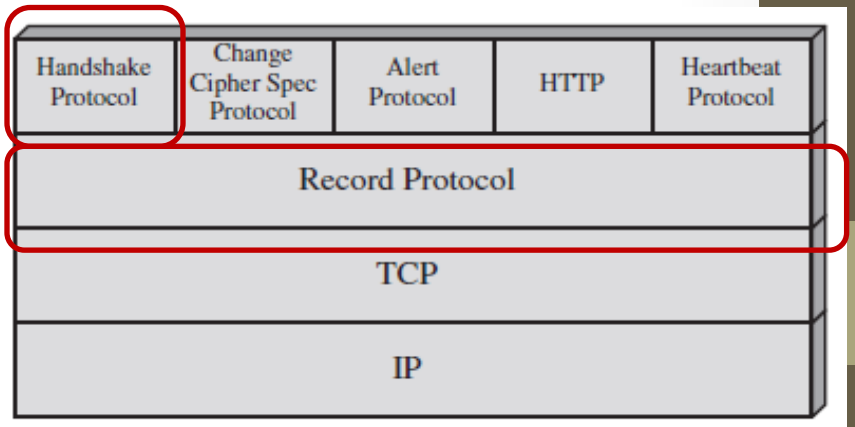


Figure 22.4 **SSL/TLS Protocol Stack**

TLS Protocols

- **Handshake Protocol**:
  - The Handshake Protocol consists of a series of messages exchanged by client and server.
  - The initial exchange needed to establish a logical connection between client and server.
  - The exchange can be viewed as having **four** phases.

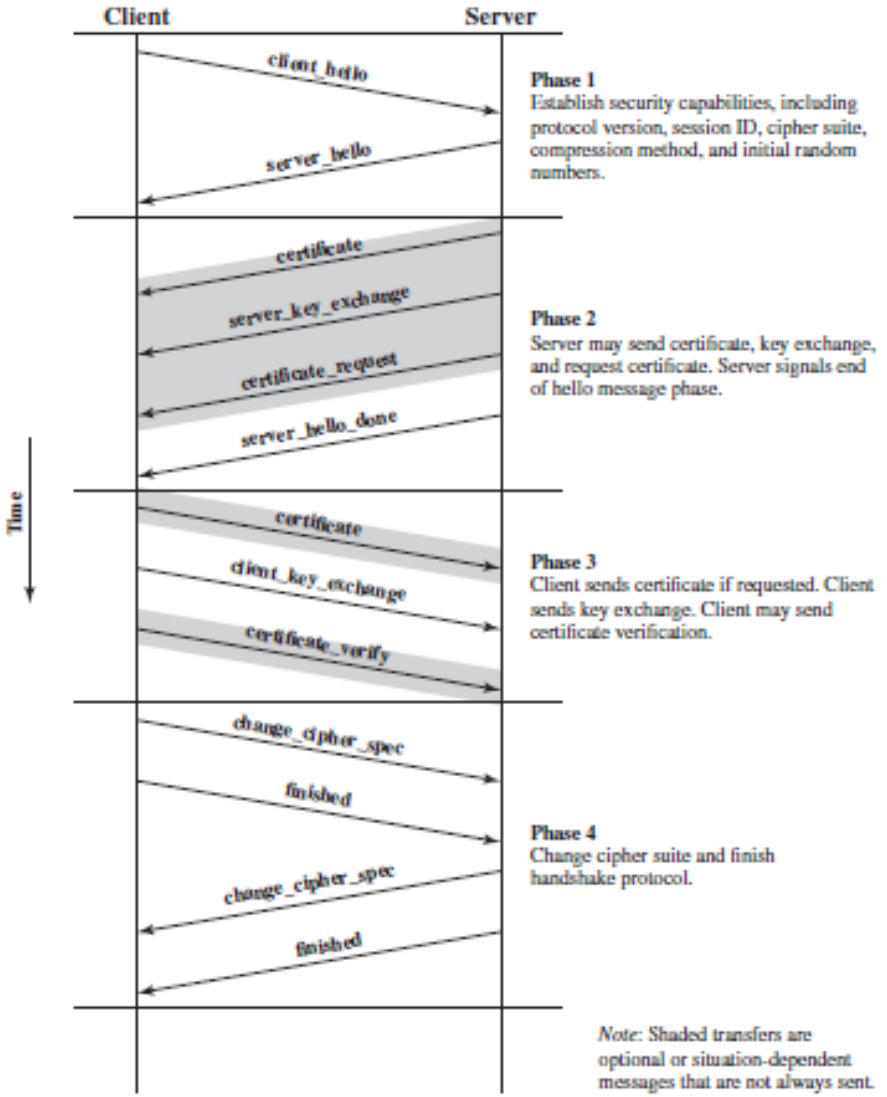
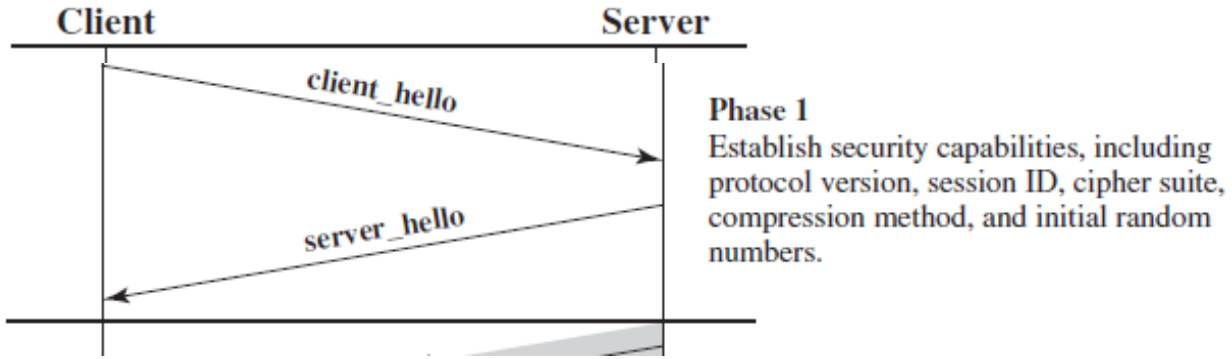


Figure 22.6 **Handshake Protocol Action**

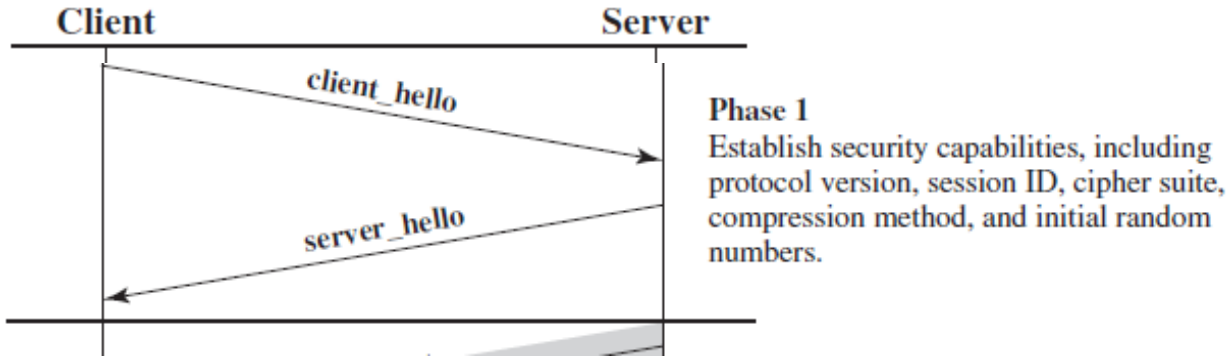
TLS Protocols

- **Handshake Protocol**:
- **Phase 1: Establishing security capabilities**
  - Comprises of exchange of two messages – **Client_hello and Server_hello**
  - **Client_hello** - contains of list of cryptographic algorithms supported by the client, in decreasing order of preference.
  - **Server_hello** - contains the selected Cipher Specification (**CipherSpec**) and a new session_id.



TLS Protocols

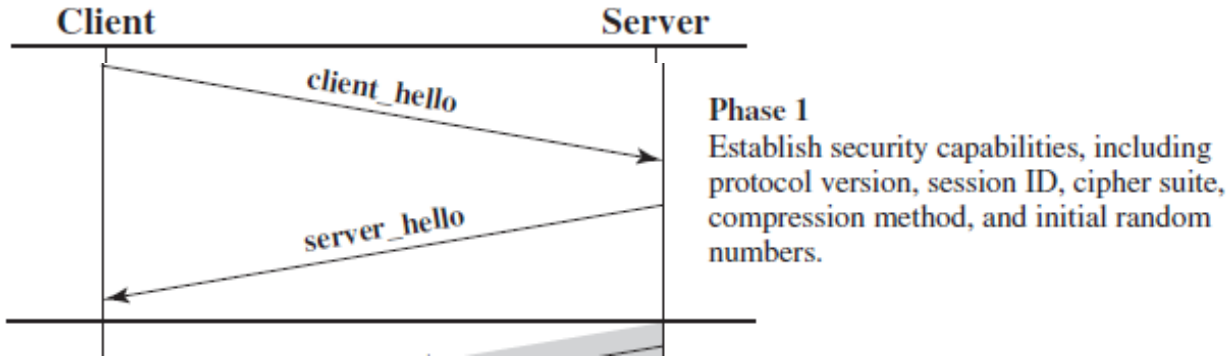
- **Handshake Protocol**:
- **Phase 1: Establishing security capabilities**
  - Comprises of exchange of two messages – **Client_hello and Server_hello**
  - **Client_hello** - contains of list of cryptographic algorithms supported by the client, in decreasing order of preference.
  - **Server_hello** - contains the selected Cipher Specification (**CipherSpec**) and a new session_id.



  - **Version**: The highest TLS version understood by the client.

TLS Protocols

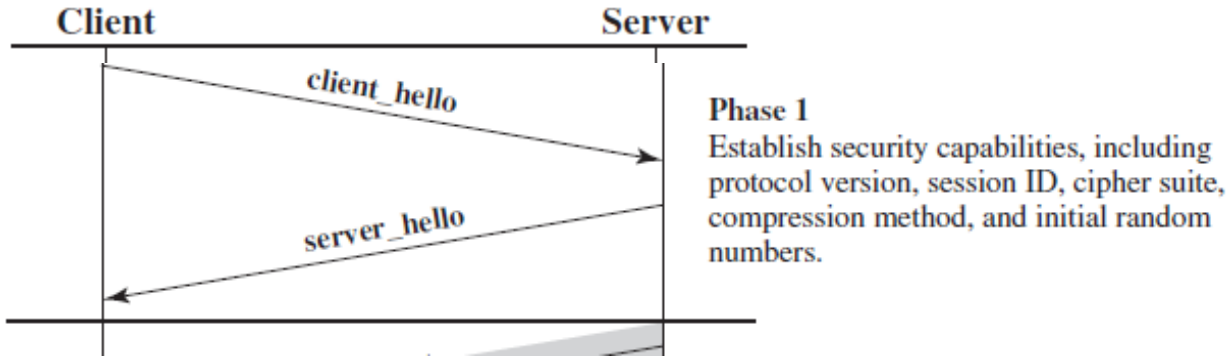
- **Handshake Protocol**:
- **Phase 1: Establishing security capabilities**
  - Comprises of exchange of two messages – **Client_hello and Server_hello**
  - **Client_hello** - contains of list of cryptographic algorithms supported by the client, in decreasing order of preference.
  - **Server_hello** - contains the selected Cipher Specification (**CipherSpec**) and a new session_id.



  - **Random**: A client-generated random structure, consisting of a 32-bit timestamp and 28 bytes generated by a secure random number generator. These values are used during key exchange to prevent replay attacks

TLS Protocols

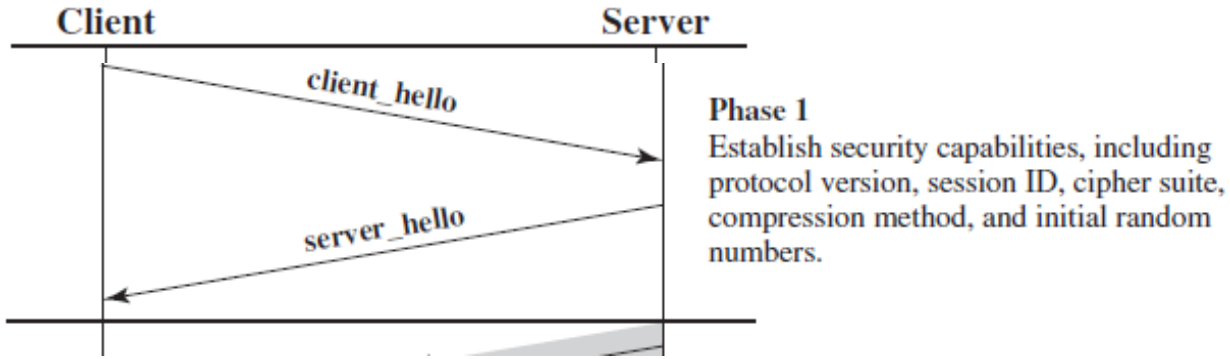
- **Handshake Protocol**:
- **Phase 1: Establishing security capabilities**
  - Comprises of exchange of two messages – **Client_hello and Server_hello**
  - **Client_hello** - contains of list of cryptographic algorithms supported by the client, in decreasing order of preference.
  - **Server_hello** - contains the selected Cipher Specification (**CipherSpec**) and a new session_id.



  - **Session ID:** A variable-length session identifier. A **nonzero** value indicates that the client wishes to update the parameters of an existing connection or create a new connection on this session. A **zero** value indicates that the client wishes to establish a new connection on a new session.

TLS Protocols

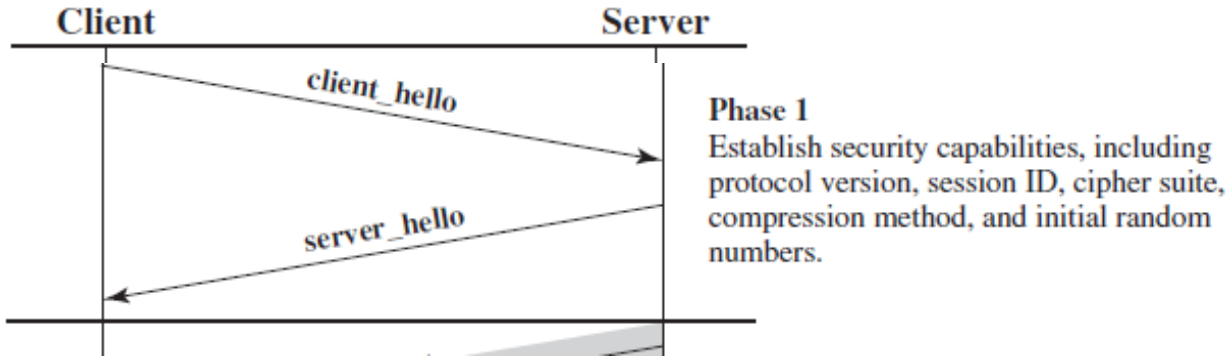
- **Handshake Protocol**:
- **Phase 1: Establishing security capabilities**
  - Comprises of exchange of two messages – **Client_hello and Server_hello**
  - **Client_hello** - contains of list of cryptographic algorithms supported by the client, in decreasing order of preference.
  - **Server_hello** - contains the selected Cipher Specification (**CipherSpec**) and a new session_id.



  - **CipherSuite:** This is a list that contains the combinations of cryptographic algorithms supported by the client, in decreasing order of preference. Each element of the list (each cipher suite) defines both a key exchange algorithm and a CipherSpec.

TLS Protocols

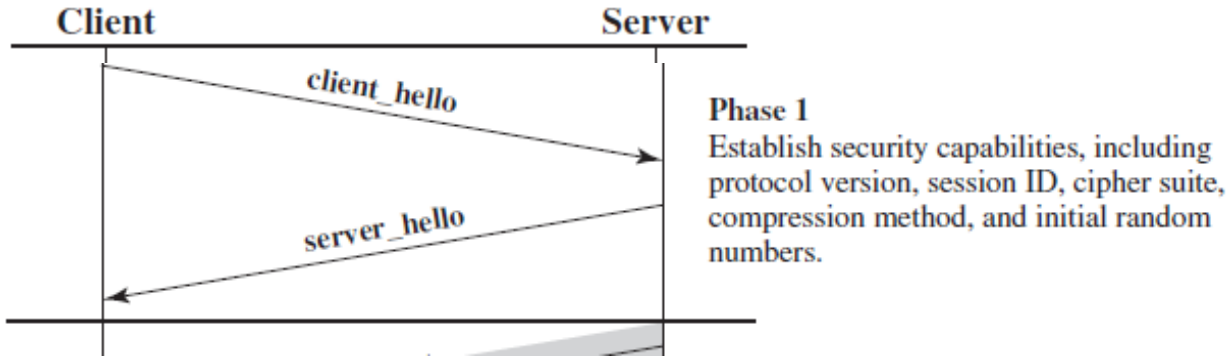
- **Handshake Protocol**:
- **Phase 1: Establishing security capabilities**
  - Comprises of exchange of two messages – **Client_hello and Server_hello**
  - **Client_hello** - contains of list of cryptographic algorithms supported by the client, in decreasing order of preference.
  - **Server_hello** - contains the selected Cipher Specification (**CipherSpec**) and a new session_id.



  - **Compression method:** This is a list of the compression methods the client supports.

TLS Protocols

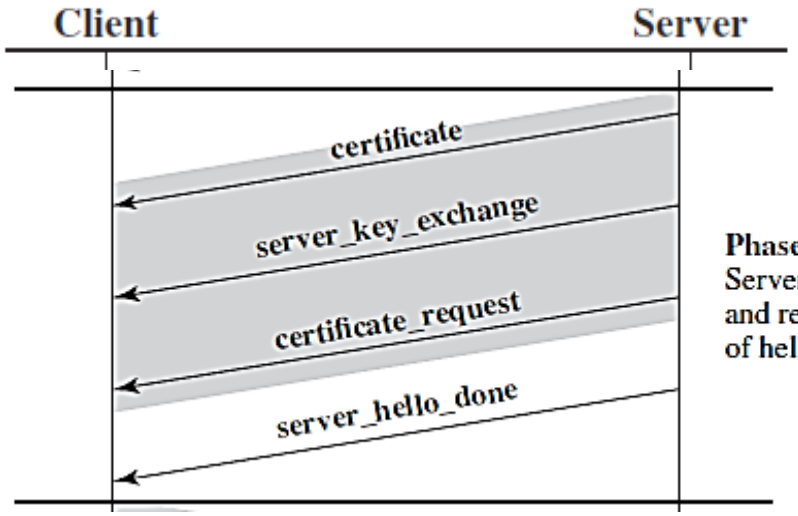
- **Handshake Protocol**:
- **Phase 1: Establishing security capabilities**
  - Comprises of exchange of two messages – **Client_hello and Server_hello**
  - **Client_hello** - contains of list of cryptographic algorithms supported by the client, in decreasing order of preference.
  - **Server_hello** - contains the selected Cipher Specification (**CipherSpec**) and a new session_id.



  - After sending the client_hello message, the client waits for the server_hello message, which contains the same parameters as the client_hello message.

TLS Protocols

- **Handshake Protocol**:
- **Phase 2: Server authentication and key exchange**
  - Depend on the underlying public-key encryption scheme that is used.
  - In some cases, the server passes a certificate to the client, possibly additional key information, and a request for a certificate from the client.

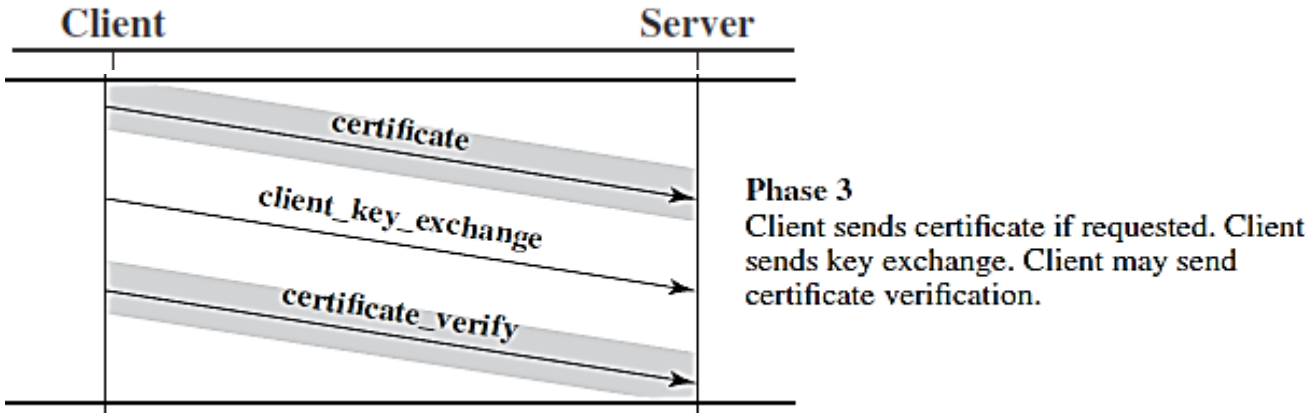


Phase 2
Server may send certificate, key exchange, and request certificate. Server signals end of hello message phase.

  - The last message is the **server_done message,** which is sent by the server to indicate the end of the server hello and associated messages.
  - After sending this message, the server will wait for a client response.

TLS Protocols

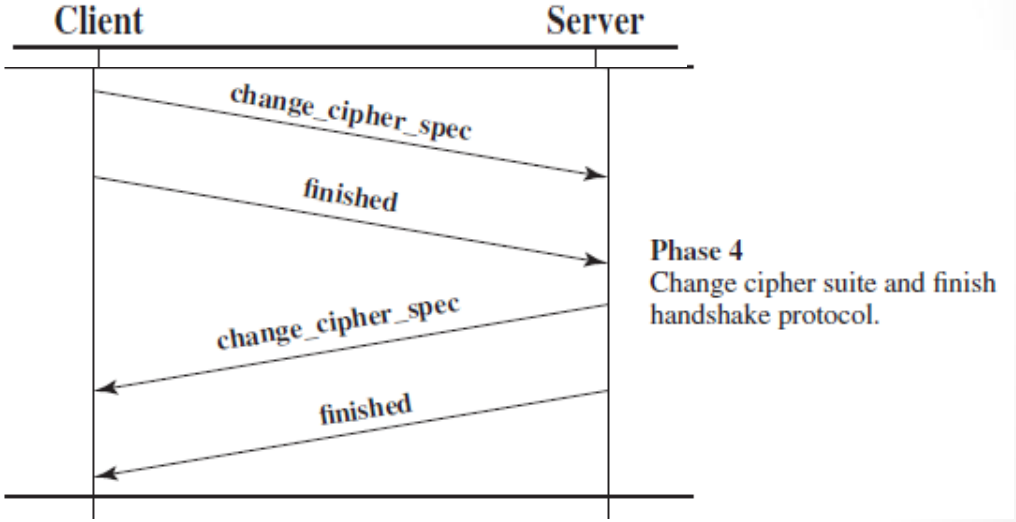
- **Handshake Protocol**:
- **Phase 3 – Client authentication and key exchange.**
  - Upon receipt of the **server_done message**, the client should verify that the server provided a valid certificate if required and check that the server_hello parameters are acceptable.



  - If all is satisfactory, the client sends one or more messages back to the server, depending on the underlying public-key scheme.

TLS Protocols

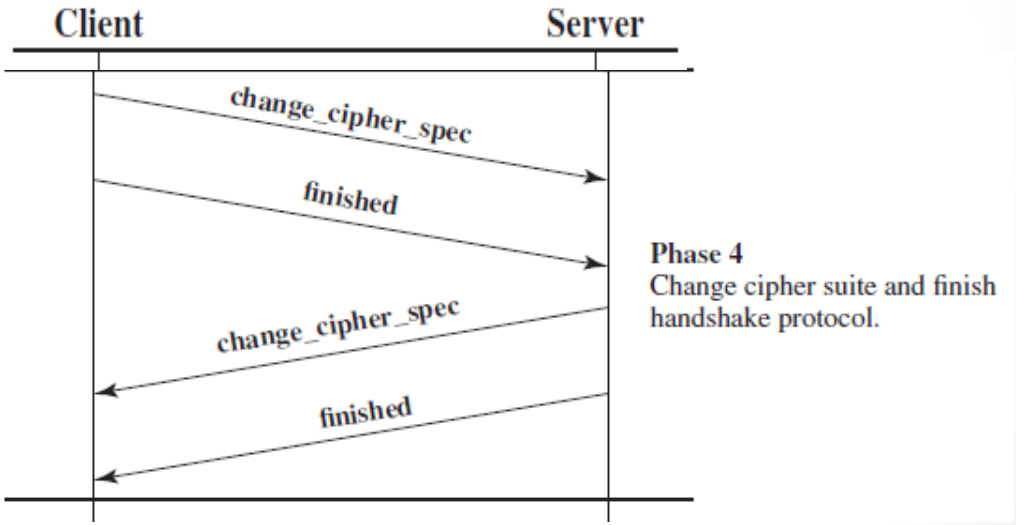
- **Handshake Protocol**:
- **Phase 4 – Finish**
  - Completes the setting up of a secure connection.
  - Client sends a **change_cipher_spec message** and copies the pending CipherSpec into the current CipherSpec.



  - The client then immediately sends the **finished** message under the new algorithms, keys, and secrets.
  - The **finished** message verifies that the key exchange and authentication processes were successful.

TLS Protocols

- **Handshake Protocol**:
- **Phase 4 – Finish**
  - Completes the setting up of a secure connection.
  - Client sends a **change_cipher_spec message** and copies the pending CipherSpec into the current CipherSpec.



  - In response to these two messages, the server sends its own **change_cipher_spec** message, transfers the pending to the current CipherSpec, and sends its **finished** message.
  - At this point, the **handshake is complete** and the client and server may begin to exchange application layer data.

TLS Protocols

- **Heartbeat Protocol**
  - A heartbeat is a periodic signal generated by hardware or software to indicate normal operation or to synchronize other parts of a system.
  - A Heartbeat Protocol is typically used to monitor the availability of a protocol entity.
  - Consists of two message types: **heartbeat_request and heartbeat_response.**
  - The use of the Heartbeat Protocol is established during Phase 1 of the Handshake Protocol.

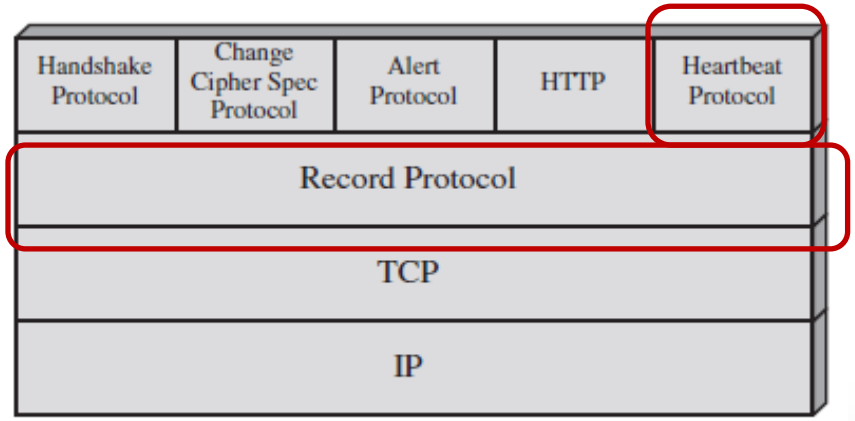


Figure 22.4 **SSL/TLS Protocol Stack**

TLS Protocols

- **Heartbeat Protocol**
  - Whenever a request message is received, it should be answered promptly with a corresponding **heartbeat_response message**.
  - The **heartbeat_request** message includes **payload length, payload, and padding fields.**
  - The **payload** is a random content between 16 bytes and 64 Kbytes in length.
  - The corresponding **heartbeat_response message** must include an exact copy of the received payload.

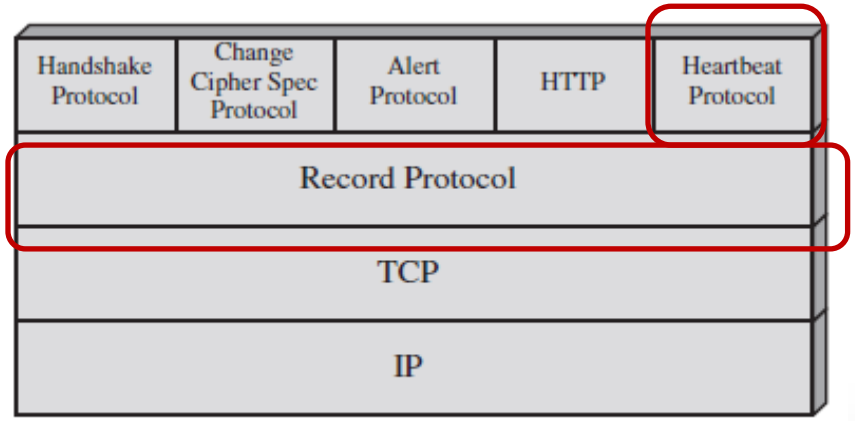


Figure 22.4 **SSL/TLS Protocol Stack**

TLS Protocols

- **Heartbeat Protocol**
  - Whenever a request message is received, it should be answered promptly with a corresponding **heartbeat_response message**.
  - The **heartbeat_request** message includes **payload length, payload, and padding fields.**
  - The **padding** is also a random content and used to perform a path maximum transfer unit (MTU) discovery operation.

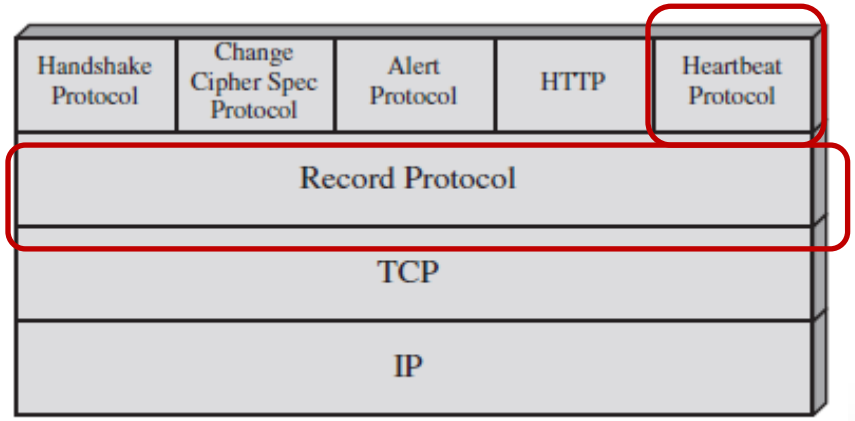


Figure 22.4 **SSL/TLS Protocol Stack**

TLS Protocols

- **Heartbeat Protocol**
  - The heartbeat **serves two purposes**.
  1. It assures the sender that the recipient is still alive, even though there may not have been any activity over the underlying TCP connection for a while.
  2. The heartbeat generates activity across the connection during idle periods, which avoids closure by a firewall that does not tolerate idle connections.

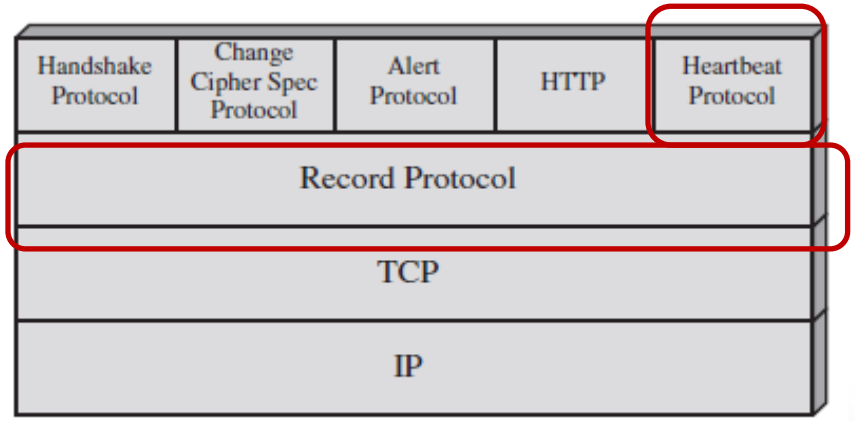


Figure 22.4 **SSL/TLS Protocol Stack**

Attacks on SSL/TLS

Attacks on SSL/TLS

- TLSv1.3 is considered to be the most secure compared to SSL.
- SSL & TLS ver 1.1 are deprecated
  - due to vulnerabilities and different possible attacks.
  - These attacks have exploited the way key exchanges happen between client and server during the negotiation phase.
  - TLSv1.2 mitigates these attacks.
- TLSv1.2 is vulnerable to downgrade attacks such as POODLE.
- These vulnerabilities have been mitigated in TLSv1.3 which protects the handshake during client-server negotiation.

Attacks on SSL/TLS

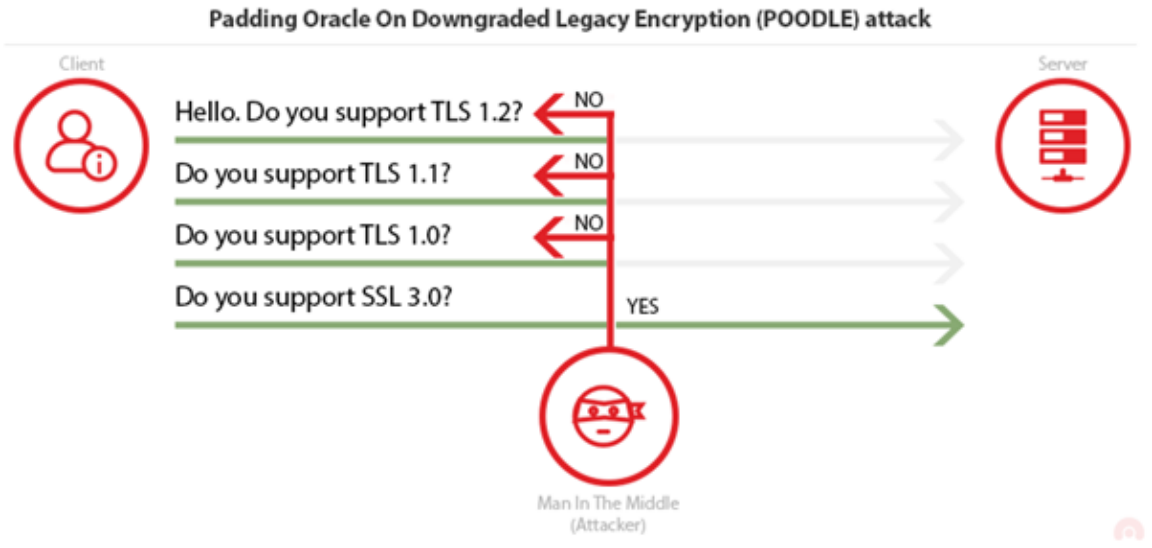
- **SSL Renegotiation Attack**
  - Due to the vulnerability in SSL renegotiation procedure, which allows an attacker to inject plaintext into the victim's requests.
  - **Renegotiation : Instead of canceling a connection and starting a new one, the session renegotiation provides authentication data to the existing connection beyond the initial negotiation that was set up.**
  - **Example**: a customer is browsing an online shop without logging in. Afterwards, he decides to make a purchase, so he enters his authentication credentials and accesses his profile on the eCommerce platform. Here, a typical SSL use renegotiation procedure instead of starting a connection from scratch.
  - Attackers who can hijack an HTTPS connection can add their own requests to the conversation between the client and server.
  - TLS supports **renegotiation indication extension**, that requires the client and server to include and verify information about previous handshakes in any renegotiation handshakes.

Attacks on SSL/TLS

- **SSL/TLS Downgrade Attacks**
  - The attacker tricks a web server into negotiating connections with the older version of TLS/SSL which are insecure.
  - The attacker then tries to intercept and/or alter the information by exploiting flaws in the older protocol versions or cryptographic algorithms.
  - Some of such attacks are
    - PODDLE attack
    - Freak attack
    - Logjam attack

Attacks on SSL/TLS

- **PODDLE : SSL/TLS Downgrade Attacks**
  - **Padding Oracle On Downgraded Legacy Encryption (POODLE)**, was published in 2014.
  - The client initiates the handshake and sends a list of supported SSL/TLS versions.
  - An attacker intercepts the traffic, performing a man-in-the-middle (MITM) attack, and impersonates the server until the client agrees to downgrade the connection to SSL 3.0.



Attacks on SSL/TLS

- **Freak : SSL/TLS Downgrade Attacks**
  - **FREAK (Factoring RSA Export Keys)** attack works by exploiting the deliberately weak **export cipher suites**.
  - **Export Cipher:** Export ciphers' are low-grade cryptographic ciphers that were authorized to be used outside US during the 1990's.
  - FREAK tricks the server into using an export cipher suite that uses RSA moduli of 512 bits or less.
  - Such keys can be easily cracked by today's computing power.
  - To fix this, one must disable support for any export-grade cipher suites in software using SSL/TLS.

Currently, the standard sizes for RSA keys are as follows:

Key size	Key strength
512 bits	Low-strength key
1024 bits	Medium-strength key
2048 bits	High-strength key
4096 bits	Very high-strength key

Attacks on SSL/TLS

- **Logjam : SSL/TLS Downgrade Attacks**
  - Discovered in May 2015, allows an attacker to intercept an HTTPS connection by downgrading the connection to 512-bit, export-grade Diffie-Hellman groups.
  - This is similar to the FREAK attack, except that Logjam attacks the Diffie-Hellman key exchange instead of the RSA key exchange, as is the case in Freak attack.
  - To overcome this, one must disable support for all export-grade Diffie-Hellman cipher suites on your servers.

Attacks on SSL/TLS

- **DROWN Attack**
  - **Decrypting RSA using Obsolete and Weakened Encryption(DROWN)**
  - DROWN is the cross-protocol security bug that attacks servers supporting modern SSLv3/TLS protocol suites by using their support for the obsolete, insecure, SSLv2 protocol.
  - Such attack is possible due to server configurations not being updated.
  - It is also possible because most people will not buy multiple certificates, a server will use the same RSA private key for both TLS and SSLv2 protocols meaning that any bugs from SSLv2 could easily affect the TLS.

Attacks on SSL/TLS

- **Truncation attack**
  - A TLS truncation attack blocks a victim's account logout requests so that the user unknowingly remains logged into a web service.
  - When the sign out request is sent by the client, the attacker intercept it and sends a TCP FIN message to client to close the connection.
  - **The server does not receive the logout request, and is unaware of the abnormal termination.**
  - To prevent this, SSLv3 onward has a closing handshake, so the recipient knows the message has not ended until this has been performed.

Attacks on SSL/TLS

- **MITM (Man in The Middle Attack)**
- Occur when a attacker is able to get unauthorized access and intercept the secure communication between the sender and the receiver.
- Attacker can perform MITM attacks in many ways
  - By getting access to SSL/TLS private keys that bind the certificate authenticity and unsecured end points.
  - Poorly secured Intermediate Certificate Authority's private keys can get compromised, leading to a much bigger impact on all the certificates issued by them.
  - If the end point system is vulnerable and an attacker is able to add a fake trusted Root CA certificate in the trusted root authority list.

Attacks on SSL/TLS

- **SSL Stripping attacks**
  - An attacker establishes their self as a router and establishes HTTPS connections with Internet servers.
  - Usually, the end-user connects with the attacker over the unsecured HTTPS connection believing it's an authenticated router.
  - The attacker is then able to read the communication, forward the request to the server, and pass the response back to the user.
  - The intent of such attacks is to read data such as usernames, passwords, and any payment related data that the attacker can later exploit.

Attacks on SSL/TLS

- **SSL Hijacking attacks**
  - Also known as **cookie hijacking**, is the exploitation of a valid session by gaining unauthorized access to the session key/ID information.
  - Normally, when a user logs in to the website, the server sets a temporary remote cookie in the client's browser to authenticate the session.
  - This enables the remote server to remember the client's login status.
  - In order to execute session hijacking, a hacker needs to know the client's session ID information (obtained by user clicks on the malicious link that contains a prepared session ID)
  - The attacker can take over the targeted session by using the stolen session ID in their own browser session.
  - Eventually, the server is tricked into thinking that the attacker's connection is the same as the real user's original session.
  - To protect against SSL hijacking, one must avoid connecting to non-secure (HTTP) urls, be careful while connecting to the public wi-fi, use secure cookie flag, use anti-malware on clients as well as server machines, and time-out inactive sessions.

Attacks on SSL/TLS

- **HEARTBLEED Attack:**
  - The heartbeat protocol is used to keep a connection alive as long as both parties are still there.
  - In typical heartbeat protocol, the client sends the heartbeat message with the data and size to the server.
  - The server then responds back with the client's data received and size data.
  - The Heartbleed vulnerability was aimed to exploit the fact that if the client sends a fake data length to the server, then the server would respond back with some random data from its memory to meet the length requirement specified by the client.
  - The random unencrypted data from the server's memory may contain critical information, such as private keys, credit card details, and other sensitive information.
  - The attacker may misuse these information.

HTTPS

HTTPS

- **HTTPS (HTTP over SSL) - combination of HTTP and SSL.**
- Used to implement secure communication between a Web browser and a Web server.
- The HTTPS capability is built into all modern Web browsers.
- Its use depends on the Web server supporting HTTPS communication.
- From user's point of view
  - URL (uniform resource locator) addresses begin with https:// rather than http://.
  - A normal HTTP connection uses port **80**.
  - If HTTPS is specified, port **443** is used, which invokes SSL.

HTTPS

- When HTTPS is used, the following elements of the communication are encrypted:
  - **URL of the requested document**
  - **Contents of the document**
  - **Contents of browser forms (filled in by browser user)**
  - **Cookies sent from browser to server and from server to browser**
  - **Contents of HTTP header**
- There is no fundamental change in using HTTP over either SSL or TLS, and both implementations are referred to as HTTPS.

HTTPS: Connection Initiation

- For HTTPS, the agent acting as the HTTP client also acts as the TLS client.
- The client initiates a connection to the server on the appropriate port and then sends the TLS **ClientHello** to begin the TLS handshake.
- When the TLS handshake has finished, the client may then initiate the first HTTP request.
- All HTTP data is to be sent as TLS application data.
- More specific
  - HTTP request flows to TLS level and then to TCP level.
  - At the level of TLS, a session is established between a TLS client and a TLS server.
  - This session can support one or more connections at any time.

HTTPS: Connection Closure

- An HTTP client/server can close a connection by sending HTTP record: **Connection: close**.
- This indicates that the connection will be closed after this record is delivered.
- The closure of an HTTPS connection requires that TLS close the connection with the peer TLS entity.
- At the TLS level, the **TLS alert protocol sends a close_notify alert.**