CRYPTOGRAPHY AND NETWORK SECURITY
UNIT-5

Syllabus: User Authentication, Transport Layer Security & Email Security
User Authentication: Remote user authentication principles, Kerberos,
Transport Level Security: Web Security Requirements, Secure Socket Layer (SSL) and Transport Layer Security(TLS), Secure Shell(SSH), **Electronic Mail Security:** Pretty Good Privacy (PGP) and S/MIME.

5.1. User Authentication:

In most computer security contexts, user authentication is the fundamental building block and the primary line of defense. User authentication is the basis for most type of access control and for user accountability. RFC 4949 (Internet Security Glossary) defines user authentication.

A typical item of authentication information associated with this user ID is a password, which is kept secret (known only to Alice and to the system). If no one is able to obtain or guess Alice's password, then the combination of Alice's user ID and password enables administrators to set up Alice's access permissions and audit her activity. Because Alice's ID is not secret, system users can send her e-mail, but because her password is secret, no one can pretend to be Alice.

The process of verifying an identity claimed by or for a system entity is called authentication.
An authentication process consists of two steps:

- **Identification step:** Presenting an identifier to the security system.
- **Verification step:** Presenting or generating authentication information that corroborates the binding between the entity and the identifier.

In essence, identification is the means by which a user provides a claimed identity to the system; user authentication is the means of establishing the validity of the claim.

Means of User Authentication:

There are four general means of authenticating a user's identity, which can be used alone or in combination:

- **Something the individual knows**: Examples include a password, a personal identification number (PIN), or answers to a prearranged set of questions.
- **Something the individual possesses**: Examples include cryptographic keys, electronic keycards, smart cards, and physical keys. This type of authenticator is referred to as a token.
- **Something the individual is** (static biometrics): Examples include recognition by fingerprint, retina, and face.
- **Something the individual does (dynamic biometrics):** Examples include recognition by voice pattern, handwriting characteristics, and typing rhythm.

All of these methods, properly implemented and used, can provide secure user authentication. However, each method has problems.

- An adversary may be able to guess or steal a password.
- Similarly, an adversary may be able to forge or steal a token.

- A user may forget a password or lose a token.
- Furthermore, there is a significant administrative overhead for managing password and token information on systems and securing such information on systems.
- With respect to biometric authenticators, there are a variety of problems, including dealing with false positives and false negatives, user acceptance, cost, and convenience.

Mutual Authentication:

Protocols which enable communicating parties to satisfy themselves mutually about each other's identity and to exchange session keys.

Central to the problem of authenticated key exchange are **two issues:**

- **confidentiality** and
- **timeliness.**

Confidentiality: To prevent **masquerade** and to prevent compromise of session keys, essential identification and session-key information must be communicated in encrypted form. This requires the prior existence of secret or public keys that can be used for this purpose.

Timeliness, is important because of the threat of message **replays**. Such replays, at worst, could allow an opponent to compromise a session key or successfully impersonate another party. At minimum, a successful replay can disrupt operations by presenting parties with messages that appear genuine but are not.

Replay Attacks:

lists the following examples of replay attacks:

1. The simplest replay attack is one in which the opponent simply copies a message and replays it later.
2. An opponent can replay a timestamped message within the valid time window. If both the original and the replay arrive within then time window, this incident can be logged.
3. As with example (2), an opponent can replay a timestamped message within the valid time window, but in addition, the opponent suppresses the original message. Thus, the repetition cannot be detected.
4. Another attack involves a backward replay without modification. This is a replay back to the message sender. This attack is possible if symmetric encryption is used and the sender cannot easily recognize the difference between messages sent and messages received on the basis of content.

Approaches to Coping with Replay Attacks:

- Attach a **sequence number** to each message used in an authentication exchange
  - A new message is accepted only if its sequence number is in the proper order
  - Difficulty with this approach is that it requires each party to keep track of the last sequence number for each claimant it has dealt with
  - Generally, not used for authentication and key exchange because of overhead
- **Timestamps**
  - Requires that clocks among the various participants be synchronized
  - Party A accepts a message as fresh only if the message contains a timestamp that, in A's judgment, is close enough to A's knowledge of current time

- **Challenge/response**
  - Party A, expecting a fresh message from B, first sends B a **nonce** (challenge) and requires that the subsequent message (response) received from B contain the correct nonce value

One-Way Authentication:

One application for which encryption is growing in popularity is electronic mail (e-mail). The very nature of electronic mail, and its chief benefit, is that it is not necessary for the sender and receiver to be online at the same time. Instead, the e-mail message is forwarded to the receiver's electronic mailbox, where it is buffered until the receiver is available to read it.

The "envelope" or header of the e-mail message must be in the clear, so that the message can be handled by the store-and-forward e-mail protocol, such as the Simple Mail Transfer Protocol (SMTP). However, it is often desirable that the mail-handling protocol not require access to the plaintext form of the message, because that would require trusting the mail-handling mechanism. Accordingly, the e-mail message should be encrypted such that the mail-handling system is not in possession of the decryption key. A second requirement is that of authentication. Typically, the recipient wants some assurance that the message is from the alleged sender.

5.2. Kerberos:

Kerberos is an authentication service developed as part of Project Athena at MIT, and is one of the best known and most widely implemented **trusted third party** key distribution systems.

A workstation cannot be trusted to identify its users correctly to network services

- A user may **gain access** to a particular workstation and pretend to be another user operating from that workstation
- A user may alter the **network address** of a workstation so that the requests sent from the altered workstation appear to come from the impersonated workstation
- A user may eavesdrop on exchanges and use a **replay attack** to gain entrance to a server or to disrupt operations

Kerberos provides a centralized authentication server whose function is to authenticate users to servers and servers to users. Unlike most other authentication schemes, Kerberos relies exclusively on **symmetric encryption**, making no use of public-key encryption. Two versions of Kerberos are in common use: **version 4 & version 5.**

Kerberos Requirements:

- **Secure:** A network eavesdropper should not be able to obtain the necessary information to impersonate a user. More generally, Kerberos should be strong enough that a potential opponent does not find it to be the weak link.
- **Reliable:** For all services that rely on Kerberos for access control, lack of availability of the Kerberos service means lack of availability of the supported services. Hence, Kerberos should be highly reliable and should employ a distributed server architecture with one system able to back up another.
- **Transparent:** Ideally, the user should not be aware that authentication is taking place beyond the requirement to enter a password.
- **Scalable:** The system should be capable of supporting large numbers of clients and servers.

Kerberos Version 4:

Makes use of DES to provide the authentication service

There are different approaches to security are:

1 **SIMPLE AUTHENTICATION DIALOGUE**

2 **MORE SECURE AUTHENTICATION DIALOGUE.**

A SIMPLE AUTHENTICATION DIALOGUE:

- For a secure transaction, server should confirm the client and its request.
- In unprotected network it creates burden on server, therefore an Authentication Server(AS) is used.
- **Authentication server (AS)**
  - Knows the passwords of all users and stores these in a centralized database
  - Shares a unique secret key with each server
- **Ticket**
  - Created once the AS accepts the user as authentic; contains the user's ID and network address and the server's ID
  - Encrypted using the secret key shared by the AS and the server

Consider the following hypothetical dialogue

$$\textbf{(1)}\ \mathrm{C} \rightarrow \mathrm{AS}: \quad ID_C \| P_C \| ID_V$$
$$\textbf{(2)}\ \mathrm{AS} \rightarrow \mathrm{C}: \quad Ticket$$
$$\textbf{(3)}\ \mathrm{C} \rightarrow \mathrm{V}: \quad ID_C \| Ticket$$
$$Ticket = E(K_v, [ID_C \| AD_C \| ID_V])$$

where

C	= client	ID_V	= identifier of V
AS	= authentication server	P_C	= password of user on C
V	= server	AD_C	= network address of C
ID_C	= identifier of user on C	K_v	= secret encryption key shared by AS and V

Problem: An opponent could capture the ticket transmitted in message (2), then use the name ID_C and transmit a message of form (3) another workstation. The server would receive a valid ticket that matches the user ID and grant access to the user on that other workstation. To prevent this attack, the AS includes in the ticket the network address from which the original request came.

A MORE SECURE AUTHENTICATION DIALOGUE:

- The main problem in *A SIMPLE AUTHENTICATION DIALOGUE, the user* must enter password for every individual service.
- Kerberos overcome this by using a new server, known as **Ticket granting server (TGS).**
- Now in Kerberos we have two servers; **AS and TGS.**

Once per user logon session:

(1) $C \rightarrow AS: \quad ID_C \| ID_{tgs}$

(2) $AS \rightarrow C: \quad E(K_c, Ticket_{tgs})$

Once per type of service:

(3) $C \rightarrow TGS: \quad ID_C \| ID_V \| Ticket_{tgs}$

(4) $TGS \rightarrow C: \quad Ticket_v$

Once per service session:

(5) $C \rightarrow V: \quad ID_C \| Ticket_v$

$$Ticket_{tgs} = E(K_{tgs}, [ID_C \| AD_C \| ID_{tgs} \| TS_1 \| Lifetime_1])$$
$$Ticket_v = E(K_v, [ID_C \| AD_C \| ID_v \| TS_2 \| Lifetime_2])$$

The new service, TGS, issues tickets to users who have been authenticated to AS. Thus, the user first requests a ticket-granting ticket from the AS. The client module in the user workstation saves this ticket. Each time the user requires access to a new service, the client applies to the TGS, using the ticket to authenticate itself. The TGS then grants a ticket for the particular service. The client saves each service-granting ticket and uses it to authenticate its user to a server each time a particular service is requested.

Let us look at the details of this scheme:

1. The client requests a ticket-granting ticket on behalf of the user by sending its user's ID to the AS, together with the TGS ID, indicating a request to use the TGS service.
2. The AS responds with a ticket that is encrypted with a key that is derived from the user's password, which is already stored at the AS. When this response arrives at the client, the client prompts the user for his or her password, generates the key, and attempts to decrypt the incoming message. If the correct password is supplied, the ticket is successfully recovered.
3. The client requests a service-granting ticket on behalf of the user. For this purpose, the client transmits a message to the TGS containing the user's ID, the ID of the desired service, and the ticket-granting ticket.
4. The TGS decrypts the incoming ticket using a key shared only by the AS and the TGS and verifies the success of the decryption by the presence of its ID. It checks to make sure that the lifetime has not expired. Then it compares the user ID and network address with the incoming information to authenticate the user. If the user is permitted access to the server V, the TGS issues a ticket to grant access to the requested service.
5. The client requests access to a service on behalf of the user. For this purpose, the client transmits a message to the server containing the user's ID and the service-granting ticket. The server authenticates by using the contents of the ticket.

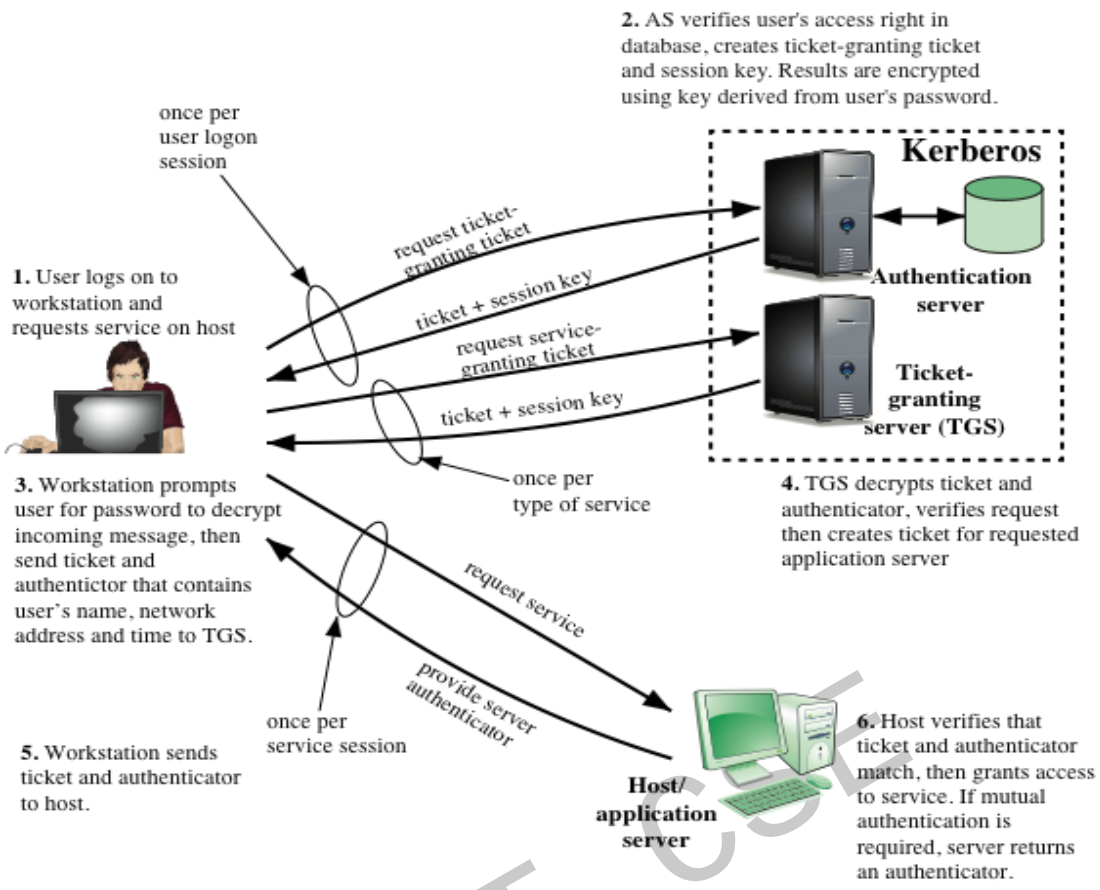


Figure 15.1 Overview of Kerberos

KERBEROS REALMS:

A full-service Kerberos environment consisting of a Kerberos server, a number of clients, and a number of application servers requires the following:

1. The Kerberos server must have the user ID and hashed passwords of all participating users in its database. All users are registered with the Kerberos server.
2. The Kerberos server must share a secret key with each server. All servers are registered with the Kerberos server

A **Kerberos realm is a set of managed nodes that share the same Kerberos database.** The Kerberos database resides on the Kerberos master computer system, which should be kept in a physically secure room. A read-only copy of the Kerberos database might also reside on other Kerberos computer systems. However, all changes to the database must be made on the master computer system. Changing or accessing the contents of a Kerberos database requires the Kerberos master password.

Kerberos Version 5:

Kerberos version 5 is specified in RFC 4120 and provides a number of improvements over version 4. Kerberos version 4 was developed for use within the Project Athena environment and, accordingly, did not fully address the need to be of general purpose.

Kerberos version 5 Authentication Dialogue:

The Kerberos version 5 message exchange involves three sessions, these are

a) Authentication service exchange

b) Ticket-granting exchange

c) Client/server authentication exchange.

Each session has two steps.

(1) C → AS *Options* ‖ *IDc* ‖ *Realmc* ‖ *IDtgs* ‖ *Times* ‖ *Nonce*1

(2) AS → C *Realmc* ‖ *IDC* ‖ *Tickettgs* ‖ E(*Kc*, [*Kc,tgs* ‖ *Times* ‖ *Nonce*1 ‖ *Realmtgs* ‖ *IDtgs*])

Tickettgs = E(*Ktgs*, [*Flags* ‖ *Kc,tgs* ‖ *Realmc* ‖ *IDC* ‖ *ADC* ‖ *Times*])

(a) Authentication Service Exchange to obtain ticket-granting ticket

(3) C → TGS *Options* ‖ *IDv* ‖ *Times* ‖ ‖ *Nonce*2 ‖ *Tickettgs* ‖ *Authenticatorc*

(4) TGS → C *Realmc* ‖ *IDC* ‖ *Ticketv* ‖ E(*Kc,tgs*, [*Kc,v* ‖ *Times* ‖ *Nonce*2 ‖ *Realmv* ‖ *IDv*])

Tickettgs = E(*Ktgs*, [*Flags* ‖ *Kc,tgs* ‖ *Realmc* ‖ *IDC* ‖ *ADC* ‖ *Times*])

Ticketv = E(*Kv*, [*Flags* ‖ *Kc,v* ‖ *Realmc* ‖ *IDC* ‖ ADC ‖ *Times*])

Authenticatorc = E(*Kc,tgs*, [*IDC* ‖ *Realmc* ‖ *TS*1])

(b) Ticket-Granting Service Exchange to obtain service-granting ticket

(5) C → V $Options \| Ticket_v \| Authenticator_c$

(6) V → C $E_{Kc,v}$ [$TS_2 \| Subkey \| Seq\#$]

Ticketv = E(*Kv*, [Flags ‖ *Kc,v* ‖ *Realmc* ‖ *IDC* ‖ *ADC* ‖ *Times*])

Authenticatorc = E(*Kc,v*, [*IDC* ‖ *Realmc* ‖ *TS*2 ‖ *Subkey* ‖ *Seq#*])

(c) Client/Server Authentication Exchange to obtain service

Environmental differences between Kerberos version 4 and version 5:

S.No	Parameters	Version 4	Version 5
1	Encryption system dependence	Version 4 requires the use of DES.	In version 5, ciphertext is tagged with an encryption-type identifier so that any encryption technique may be used.
2	Internet protocol dependence	Version 4 requires the use of Internet Protocol (IP) addresses.	Version 5 network addresses are tagged with type and length, allowing any network address type to be used.
3	Message byte ordering	In version 4, the sender of a message employs a byte ordering of its own choosing and tags the message to indicate least significant byte in lowest address or most significant byte in lowest address.	In version 5, all message structures are defined using Abstract Syntax Notation One (ASN.1) and Basic Encoding Rules (BER), which provide an unambiguous byte ordering.
4	Ticket lifetime	Lifetime values in version 4 are encoded in an 8-bit quantity in units of five minutes. Thus, the	In version 5, tickets include an explicit start time and end time,

		maximum lifetime that can be expressed is 2^8*5=1280 minutes. This may be inadequate for some applications.	allowing tickets with arbitrary lifetimes.
5	Authentication forwarding	Version 4 does not allow credentials issued to one client to be forwarded to some other host and used by some other client. This capability would enable a client to access a server and have that server access another server on behalf of the client.	Version 5 provides this capability authentication forwarding.
6	Inter-realm authentication	In version 4, interoperability among N realms requires on the order of N^2 Kerberos-to-Kerberos relationships, as described earlier.	Version 5 supports a method that requires fewer relationships.

Technical differences between Kerberos version 4 and version 5:

S.No	Parameters	Version 4	Version 5
1	Double encryption	In version 4, that tickets provided to clients are encrypted twice—once with the secret key of the target server and then again with a secret key known to the client.	In version 5, the ticket which ate issued to the clients are encrypted with only one key.
2	PCBC encryption	Encryption in version 4 makes use of a nonstandard mode of DES known as propagating cipher block chaining (PCBC).	Version 5 provides explicit integrity mechanisms, allowing the standard CBC mode to be used for encryption
3	Session keys	The session keys are included in each ticket that can be later used by client and server.	Each time connection is established a different sub session key is used by both client and server.
4	Password attacks	It does not provide any mechanism to prevent attacks on passwords	Version 5 provides a mechanism known as pre authentication but it does not prevent password attacks

5.3 Transport Level Security:

5.3.1 Web security considerations:

The World Wide Web is fundamentally a client/server application running over the Internet and TCP/IP intranets

The following characteristics of Web usage suggest the need for tailored security tools:

- Web servers are relatively easy to configure and manage
- Web content is increasingly easy to develop
- The underlying software is extraordinarily complex
    - May hide many potential security flaws
- A Web server can be exploited as a launching pad into the corporation's or agency's entire computer complex
- Casual and untrained (in security matters) users are common clients for Web-based services
    - Such users are not necessarily aware of the security risks that exist and do not have the tools or knowledge to take effective countermeasures

Web security Threats:

	Threats	Consequences	Countermeasures
Integrity	•Modification of user data •Trojan horse browser •Modification of memory •Modification of message traffic in transit	•Loss of information •Compromise of machine •Vulnerabilty to all other threats	Cryptographic checksums
Confidentiality	•Eavesdropping on the net •Theft of info from server •Theft of data from client •Info about network configuration •Info about which client talks to server	•Loss of information •Loss of privacy	Encryption, Web proxies
Denial of Service	•Killing of user threads •Flooding machine with bogus requests •Filling up disk or memory •Isolating machine by DNS attacks	•Disruptive •Annoying •Prevent user from getting work done	Difficult to prevent
Authentication	•Impersonation of legitimate users •Data forgery	•Misrepresentation of user •Belief that false information is valid	Cryptographic techniques

Table. A Comparison of Threats on the Web

Web Traffic Security Approaches:

A number of approaches to providing Web security are possible.

1. One way to provide Web security is to use **IP security** (IPsec) (Figure(a)). The advantage of using IPsec is that it is transparent to end users and applications and provides a general-purpose solution.
2. Another relatively general-purpose solution is to implement security just above TCP (Figure (b)). The foremost example of this approach is the **Secure Sockets Layer (SSL)** and the follow-on Internet standard known as **Transport Layer Security (TLS).** At this level, there are two implementation choices. For full generality, SSL (or TLS) could be provided as part of the underlying protocol suite and therefore be transparent to applications. Alternatively, SSL can be embedded in specific packages. For example, Netscape and Microsoft Explorer browsers come equipped with SSL, and most Web servers have implemented the protocol.

3. Application-specific security services are embedded within the particular application. Figure (c) shows examples of this architecture. The advantage of this approach is that the service can be tailored to the specific needs of a given application.

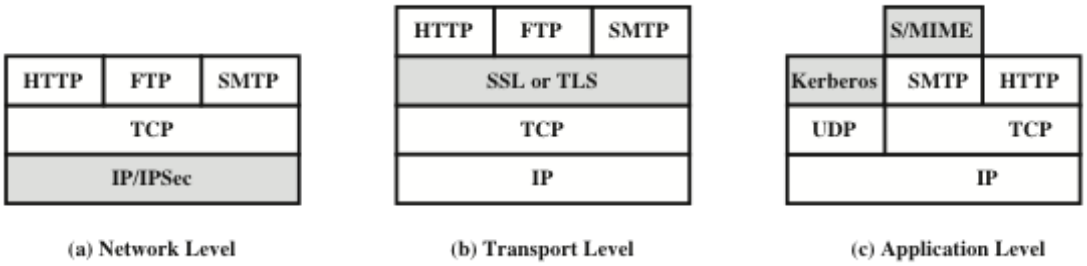


Figure: relative location of security facilities in the TCP/IP Protocol stack

5.4. SSL (Secure Socket Layer):

- SSL probably most widely used Web security mechanism, and it is implemented at the Transport layer.
- SSL is designed to make use of TCP to provide a reliable end-to-end secure service.
- Netscape originated SSL. Version 3 of the protocol was designed with public review and input from industry and was published as an Internet draft document. Subsequently, became Internet standard known as TLS (Transport Layer Security)

SSL Architecture:

- SSL is designed to make use of TCP to provide a reliable end-to-end secure service.
- SSL is not a single protocol but rather two layers of protocols.

Two important SSL concepts are the SSL session and the SSL connection, which are defined in the specification as follows.

1. **Connection:** A connection is a transport that provides a suitable type of service. For SSL, such connections are peer-to-peer relationships. Every connection is associated with one session.
2. **Session:** An SSL session is an association between a client and a server. Sessions are created by the Handshake Protocol. Sessions define a set of cryptographic security parameters which can be shared among multiple connections.

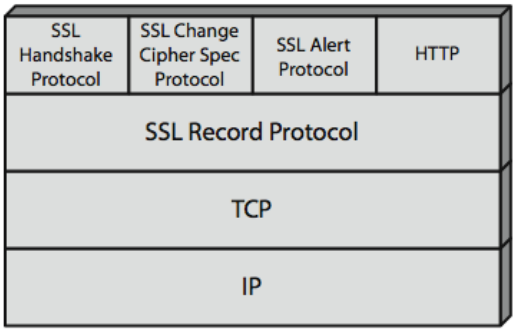


Figure: SSL Protocol stack

SSL Record Protocol:

SSL Record Protocol defines two services for SSL connections:

1. **Confidentiality:** The Handshake Protocol defines a shared secret key that is used for conventional encryption of SSL payloads. The message is compressed before being

concatenated with the MAC and encrypted, with a range of ciphers being supported as shown.

2. **Message Integrity:** The Handshake Protocol also defines a shared secret key that is used to form a message authentication code (MAC).

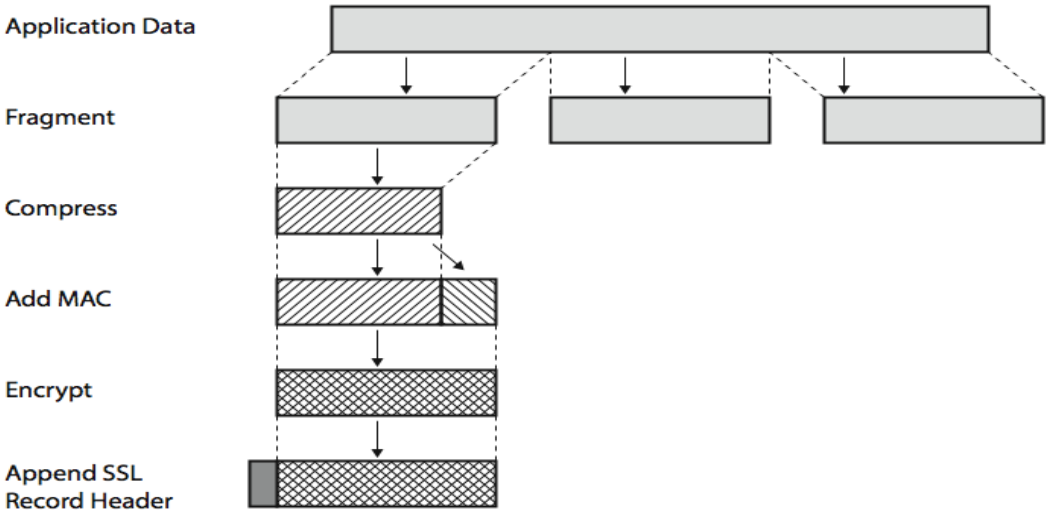


Figure: SSL Record Protocol Operation

Figure shows the overall operation of the SSL Record Protocol. The Record Protocol takes an application message to be transmitted, fragments the data into manageable blocks, optionally compresses the data, applies a MAC, encrypts, adds a header, and transmits the resulting unit in a TCP segment. Received data are decrypted, verified, decompressed, and reassembled before being delivered to higher-level users.

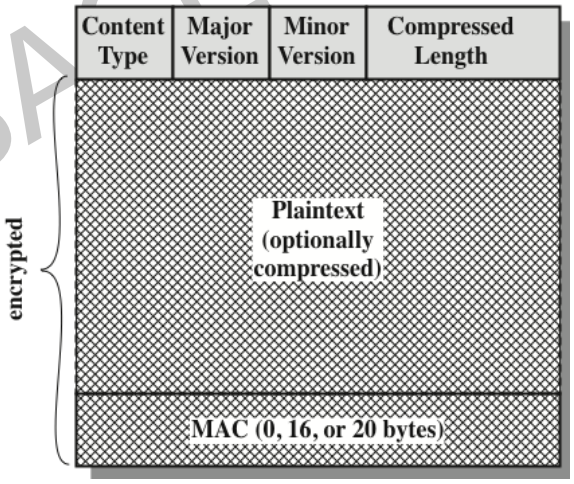


Figure: SSL Record Format

The final step of SSL Record Protocol processing is to prepare a header consisting of the following fields:

- **Content Type (8 bits):** The higher-layer protocol used to process the enclosed fragment.
- **Major Version (8 bits):** Indicates major version of SSL in use. For SSLv3, the value is 3.
- **Minor Version (8 bits):** Indicates minor version in use. For SSLv3, the value is 0.
- **Compressed Length (16 bits):** The length in bytes of the plaintext fragment (or compressed fragment if compression is used). The maximum value is 2^{14} + 2048.

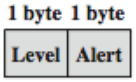
ChangeCipherSpec Protocol:

The Change Cipher Spec Protocol is one of the three SSL-specific protocols that use the SSL Record Protocol. It is the simplest, consisting of a single message, which consists of a single byte with the value 1. The sole purpose of this message is to cause the pending state to be copied into the current state, which updates the cipher suite to be used on this connection.

(a) Change Cipher Spec Protocol

SSL Alert Protocol:

The Alert Protocol is used to convey SSL-related alerts to the peer entity. As with other applications that use SSL, alert messages are compressed and encrypted, as specified by the current state. Each message in this protocol consists of two bytes, the first takes the value warning (1) or fatal (2) to convey the severity of the message. The second byte contains a code that indicates the specific alert.



(b) Alert Protocol

SSL Handshake Protocol:

The most complex part of SSL is the Handshake Protocol. This protocol allows the server and client to authenticate each other and to negotiate an encryption and MAC algorithm and cryptographic keys to be used to protect data sent in an SSL record. The Handshake Protocol is used before any application data is transmitted. The Handshake Protocol consists of a series of messages exchanged by client and server.

(c) Handshake Protocol

The exchange can be viewed in 4 phases:

- **Phase 1. Establish Security Capabilities** - this phase is used by the client to initiate a logical connection and to establish the security capabilities that will be associated with it
- **Phase 2. Server Authentication and Key Exchange** - the server begins this phase by sending its certificate if it needs to be authenticated.
- **Phase 3. Client Authentication and Key Exchange** - the client should verify that the server provided a valid certificate if required and check that the server_hello parameters are acceptable
- **Phase 4. Finish** - this phase completes the setting up of a secure connection. The client sends a change_cipher_spec message and copies the pending CipherSpec into the current CipherSpec. At this point the handshake is complete and the client and server may begin to exchange application layer data.

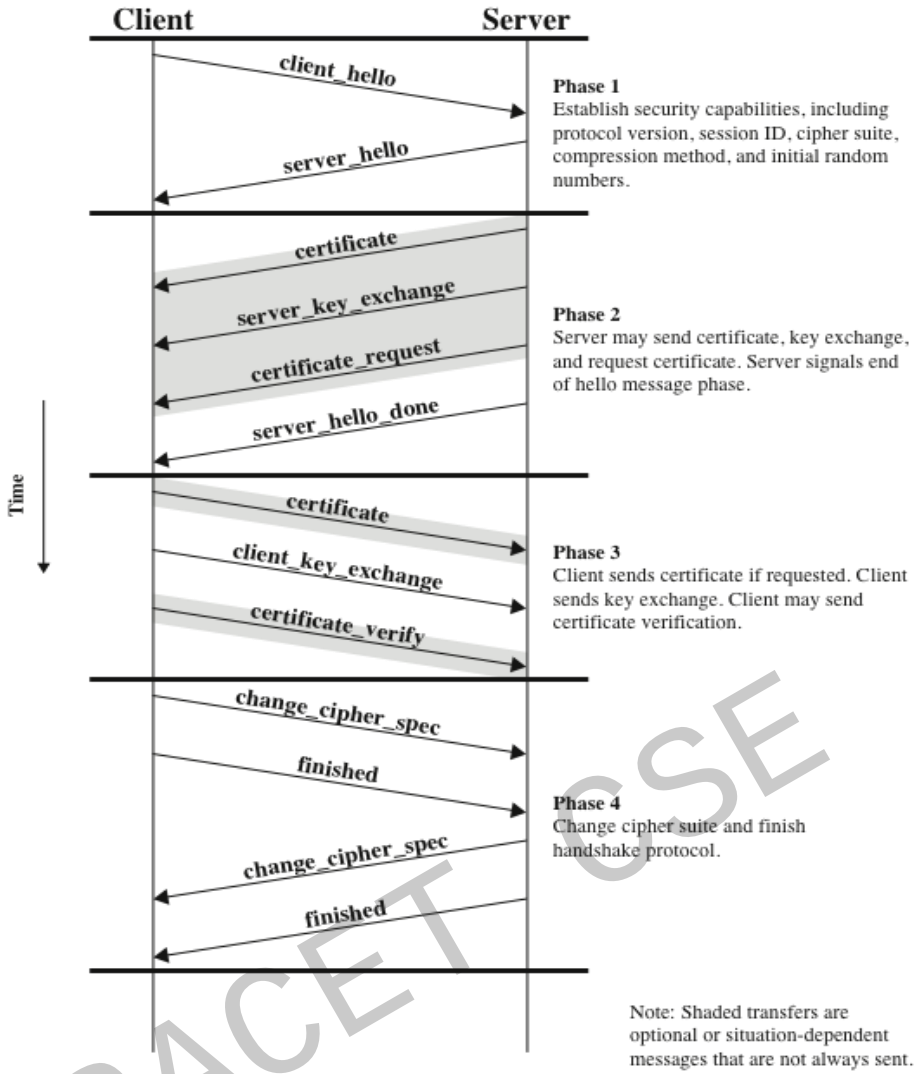


Figure 17.6 Handshake Protocol Action

5.5. Transport Layer Security (TLS) Protocol

In order to provide an open Internet standard of SSL, Internet Engineering Task Force(IETF) released The Transport Layer Security (TLS) protocol in January 1999. TLS is defined as a proposed Internet Standard in RFC 5246.

Salient 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.
- The architecture of TLS protocol is similar to SSLv3 protocol. It has two sub protocols: the TLS Record protocol and the TLS Handshake protocol.
- Though SSLv3 and TLS protocol have similar architecture, several changes were made in architecture and functioning particularly for the handshake protocol.

Comparison of TLS and SSL Protocols:

1. **Protocol Version** – The header of TLS protocol segment carries the version number 3.1 to differentiate between number 3 carried by SSL protocol segment header.
2. **Message Authentication** – TLS employs a keyed-hash message authentication code (H-MAC). Benefit is that H-MAC operates with any hash function, not just MD5 or SHA, as explicitly stated by the SSL protocol.
3. **Session Key Generation** – There are two differences between TLS and SSL protocol for generation of key material.
    1. Method of computing pre-master and master secrets is similar. But in TLS protocol, computation of master secret uses the HMAC standard and pseudorandom function (PRF) output instead of ad-hoc MAC.
    2. The algorithm for computing session keys and initiation values (IV) is different in TLS than SSL protocol.
4. **Alert Protocol Message –**
    1. TLS protocol supports all the messages used by the Alert protocol of SSL, except *No certificate* alert message being made redundant. The client sends empty certificate in case client authentication is not required.
    2. Many additional Alert messages are included in TLS protocol for other error conditions such as *record_overflow, decode_error* etc.
5. **Supported Cipher Suites** – SSL supports RSA, Diffie-Hellman and Fortezza cipher suites. TLS protocol supports all suits except Fortezza.
6. **Client Certificate Types** – TLS defines certificate types to be requested in a *certificate_request* message. SSLv3 support all of these. Additionally, SSL support certain other types of certificate such as Fortezza.
7. **CertificateVerify and Finished Messages** –
    1. In SSL, complex message procedure is used for the *certificate_verify* message. With TLS, the verified information is contained in the handshake messages itself thus avoiding this complex procedure.
    2. Finished message is computed in different manners in TLS and SSLv3.
8. **Padding of Data** – In SSL protocol, the padding added to user data before encryption is the minimum amount required to make the total data-size equal to a multiple of the cipher's block length. In TLS, the padding can be any amount that results in data-size that is a multiple of the cipher's block length, up to a maximum of 255 bytes.

5.6. Secure Shell Protocol (SSH):

The salient features of SSH are as follows –

- SSH is a network protocol that runs on top of the TCP/IP layer. It is designed to replace the TELNET which provided unsecure means of remote logon facility.
- SSH provides a secure client/server communication and can be used for tasks such as file transfer and e-mail.
- SSH2 is a prevalent protocol which provides improved network communication security over earlier version SSH1.

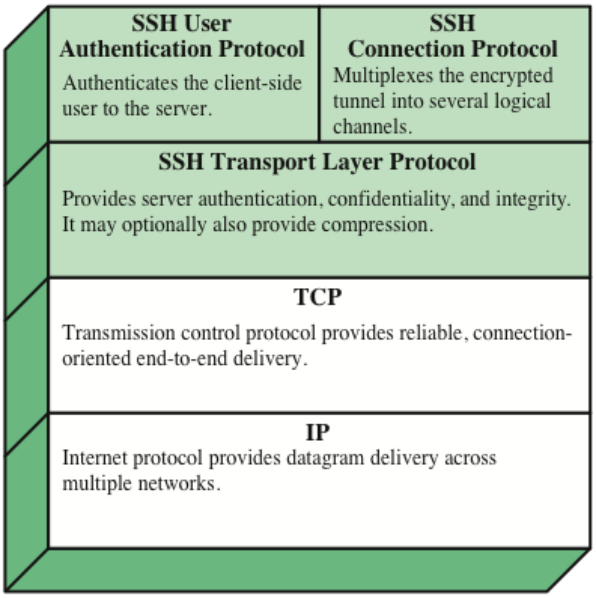


Figure: SSH Protocol stack

Transport Layer Protocol:

In this part of SSH protocol provides data confidentiality, server (host) authentication, and data integrity. It may optionally provide data compression as well.

- **Server Authentication** – Host keys are asymmetric like public/private keys. A server uses a public key to prove its identity to a client. The client verifies that contacted server is a "known" host from the database it maintains. Once the server is authenticated, session keys are generated.
- **Session Key Establishment** – After authentication, the server and the client agree upon cipher to be used. Session keys are generated by both the client and the server. Session keys are generated before user authentication so that usernames and passwords can be sent encrypted. These keys are generally replaced at regular intervals (say, every hour) during the session and are destroyed immediately after use.
- **Data Integrity** – SSH uses Message Authentication Code (MAC) algorithms to for data integrity check. It is an improvement over 32 bit CRC used by SSH1.

User Authentication Protocol:

In this part of SSH authenticates the user to the server. The server verifies that access is given to intended users only. Many authentication methods are currently used such as, typed passwo rds, Kerberos, public-key authentication, etc.

Connection Protocol:

This provides multiple logical channels over a single underlying SSH connection

SSH Services:

SSH provides three main services that enable provision of many secure solutions. These services are briefly described as follows –

- **Secure Command-Shell (Remote Logon)** – It allows the user to edit files, view the contents of directories, and access applications on connected device. Systems administrators can remotely start/view/stop services and processes, create user accounts, and change file/directories permissions and so on. All tasks that are feasible at a machine's command prompt can now be performed securely from the remote machine using secure remote logon.

- **Secure File Transfer** – SSH File Transfer Protocol (SFTP) is designed as an extension for SSH-2 for secure file transfer. In essence, it is a separate protocol layered over the Secure Shell protocol to handle file transfers. SFTP encrypts both the username/password and the file data being transferred. It uses the same port as the Secure Shell server, i.e. system port no 22.
- **Port Forwarding (Tunneling)** – It allows data from unsecured TCP/IP based applications to be secured. After port forwarding has been set up, Secure Shell reroutes traffic from a program (usually a client) and sends it across the encrypted tunnel to the program on the other side (usually a server). Multiple applications can transmit data over a single multiplexed secure channel, eliminating the need to open many ports on a firewall or router.

5.7. Electronic Mail Security:

Email is one of the most widely used and regarded network services. Currently message contents are not secure, may be inspected either in transit or by suitably privileged users on destination system.

Email Security Enhancements:

- confidentiality
  - protection from disclosure
- authentication
  - of sender of message
- message integrity
  - protection from modification
- non-repudiation of origin
  - protection from denial by sender

5.8. Pretty Good Privacy (PGP):

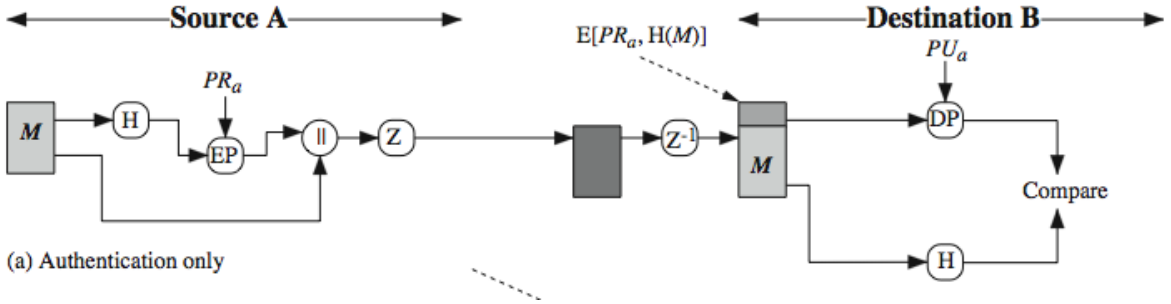
- Provides a confidentiality and authentication service that can be used for electronic mail and file storage applications
- Developed by Phil Zimmermann
  - Selected the best available cryptographic algorithms as building blocks
  - Integrated these algorithms into a general-purpose application that is independent of operating system and processor and that is based on a small set of easy-to-use commands
  - Made the package and its documentation, including the source code, freely available via the Internet, bulletin boards, and commercial networks
  - Entered into an agreement with a company to provide a fully compatible, low-cost commercial version of PGP

PGP Growth:

- It is available free worldwide in versions that run on a variety of platforms
- The commercial version satisfies users who want a product that comes with vendor support
- It is based on algorithms that have survived extensive public review and are considered extremely secure
- It has a wide range of applicability
- It was not developed by, nor is it controlled by, any governmental or standards organization
- Is now on an Internet standards track, however it still has an aura of an antiestablishment endeavor

PGP Operation – Authentication:

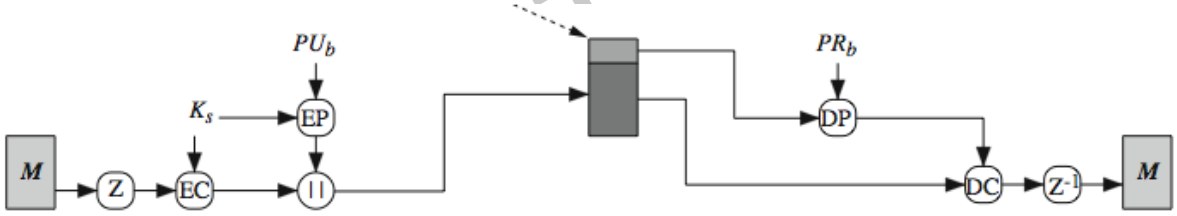
1. sender creates a message
2. SHA-1 used to generate 160-bit hash code of message
3. hash code is encrypted with RSA using the sender's private key, and result is attached to message
4. receiver uses RSA or DSS with sender's public key to decrypt and recover hash code
5. receiver generates new hash code for message and compares with decrypted hash code, if match, message is accepted as authentic



(a) Authentication only

PGP Operation – Confidentiality:

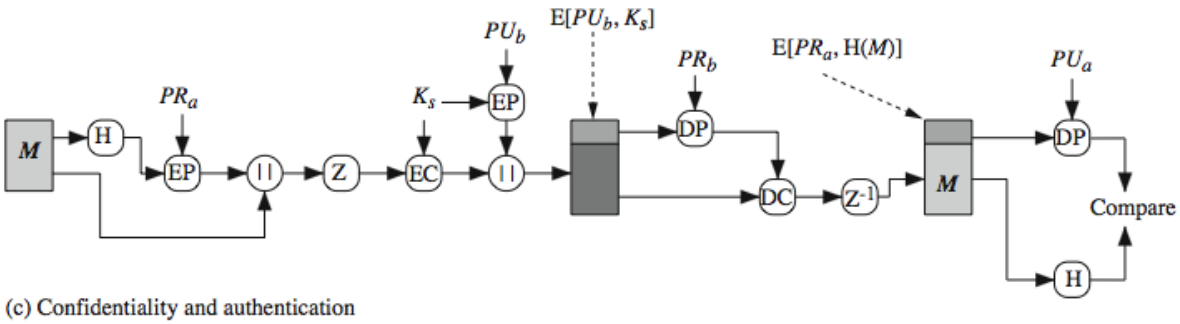
1. sender generates message and random 128-bit number to be used as session key for this message only
2. message is encrypted, using CAST-128 / IDEA/3DES with session key
3. session key is encrypted using RSA with recipient's public key, then attached to message
4. receiver uses RSA with its private key to decrypt and recover session key
5. session key is used to decrypt message



(b) Confidentiality only

PGP Operation – Confidentiality & Authentication:

- uses both services on same message
  - create signature & attach to message
  - encrypt both message & signature
  - attach RSA encrypted session key



(c) Confidentiality and authentication

PGP Operation – Compression:

- by default, PGP compresses message after signing but before encrypting

- so can store uncompressed message & signature for later verification
- & because compression is non deterministic

- uses ZIP compression algorithm

PGP Operation – Email Compatibility:

- when using PGP will have binary data to send (encrypted message etc)
- however, email was designed only for text
- hence PGP must encode raw binary data into printable ASCII characters
- uses radix-64 algorithm
  - maps 3 bytes to 4 printable chars
  - also appends a CRC
- PGP also segments messages if too big

S/MIME (Secure/Multipurpose Internet Mail Extensions):

- It is a security enhancement to MIME Internet e-mail format standard based on technology from RSA Data Security
- Defined in:
  - RFCs 3370, 3850, 3851, 3852
- S/MIME support in many mail agents
  - eg MS Outlook, Mozilla, Mac Mail etc
- To understand S/MIME, we need first to have a general understanding of the underlying e-mail format that it uses, namely MIME. We have to learn about RFC5322(internet Message Format)

RFC 5322:

- Defines a format for text messages that are sent using electronic mail
- Messages are viewed as having an envelope and contents
  - The envelope contains whatever information is needed to accomplish transmission and delivery
  - The contents compose the object to be delivered to the recipient
  - RFC 5322 standard applies only to the contents
- The content standard includes a set of header fields that may be used by the mail system to create the envelope

Multipurpose Internet Mail Extensions (MIME):

- An extension to the RFC 5322 framework that is intended to address some of the problems and limitations of the use of Simple Mail Transfer Protocol (SMTP)
- Is intended to resolve these problems in a manner that is compatible with existing RFC 5322 implementations
- The specification is provided in RFCs 2045 through 2049

The MIME specification includes the following elements.

1. **Five new message header fields** are defined, which may be included in an RFC 5322 header. These fields provide information about the body of the message.
2. A number of content formats are defined, thus standardizing representations that support multimedia electronic mail.
3. Transfer encodings are defined that enable the conversion of any content format into a form that is protected from alteration by the mail system.

The Five Header Fields Defined in MIME:

The five header fields defined in MIME are

- **MIME-Version:** Must have the parameter value 1.0. This field indicates that the message conforms to RFCs 2045 and 2046.
- **Content-Type:** Describes the data contained in the body with sufficient detail that the receiving user agent can pick an appropriate agent or mechanism to represent the data to the user or otherwise deal with the data in an appropriate manner.
- **Content-Transfer-Encoding:** Indicates the type of transformation that has been used to represent the body of the message in a way that is acceptable for mail transport.
- **Content-ID:** Used to identify MIME entities uniquely in multiple contexts.
- **Content-Description:** A text description of the object with the body; this is useful when the object is not readable (e.g., audio data).

S/MIME Functionality:

S/MIME provides the following functions.

- **Enveloped data:** This consists of encrypted content of any type and encrypted content encryption keys for one or more recipients.
- **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.
- **Clear-signed data:** As with signed data, a digital signature of the content is formed. However, in this case, 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.
- **Signed and 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 Messages:

- S/MIME secures a MIME entity with a signature, encryption, or both.
- forming a MIME wrapped **Public-Key Cryptography Standards(**PKCS) object
- have a range of content-types:
    - enveloped data
    - signed data
    - clear-signed data
    - registration request
    - certificate only message

S/MIME Certificate Processing:

- S/MIME uses public-key certificates that conform to version 3 of X.509
- The key-management scheme used by S/MIME is in some ways a hybrid between a strict X.509 certification hierarchy and PGP's web of trust
- S/MIME managers and/or users must configure each client with a list of trusted keys and with certificate revocation lists
    - The responsibility is local for maintaining the certificates needed to verify incoming signatures and to encrypt outgoing messages
- The certificates are signed by certification authorities