REVA UNIVERSITY

Established as per the Section 2(f) of the UGC Act, 1956
Approved by AICTE, COA and BCI, New Delhi

Lecture 1.6

Authentication Applications

School of Computing and Information Technology

Mr. Raghavendra Nayaka P
raghavendra.nayak@reva.edu.in

AY: 2020-2021

Introduction Class

Recap of previous Lecture

TOPICS TO BE DISCUSSED

Authentication Applications: Kerberos, X.509 Directory Authentication Service.

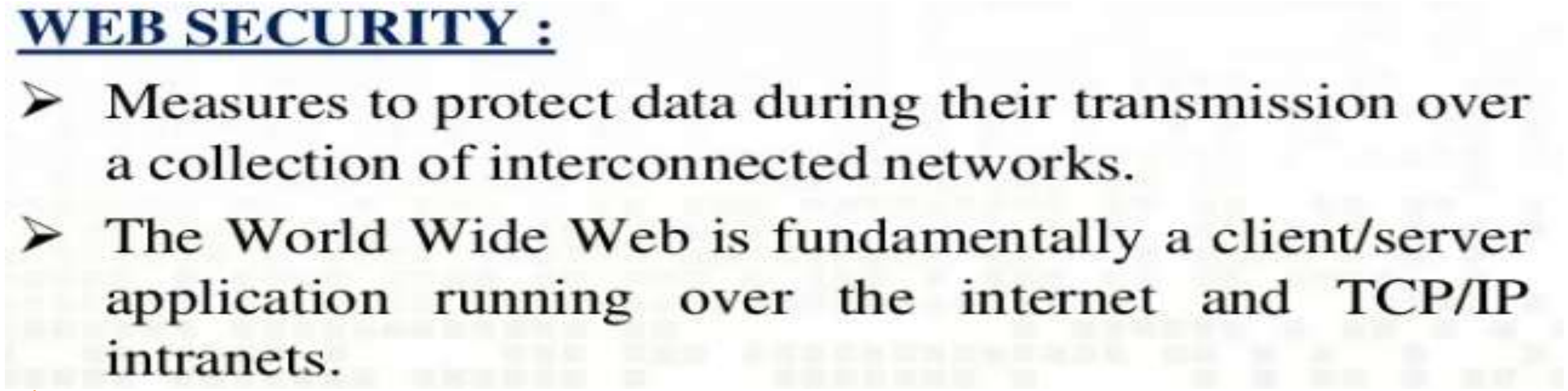
Electronic Mail Security: Pretty Good Privacy (PGP); S/MIME.

Transport level Security, : Web Security Considerations,Web Security Threats

Web Traffic Security Approaches, SSL Architecture, SSL Record Protocol

WEB SECURITY THREATS

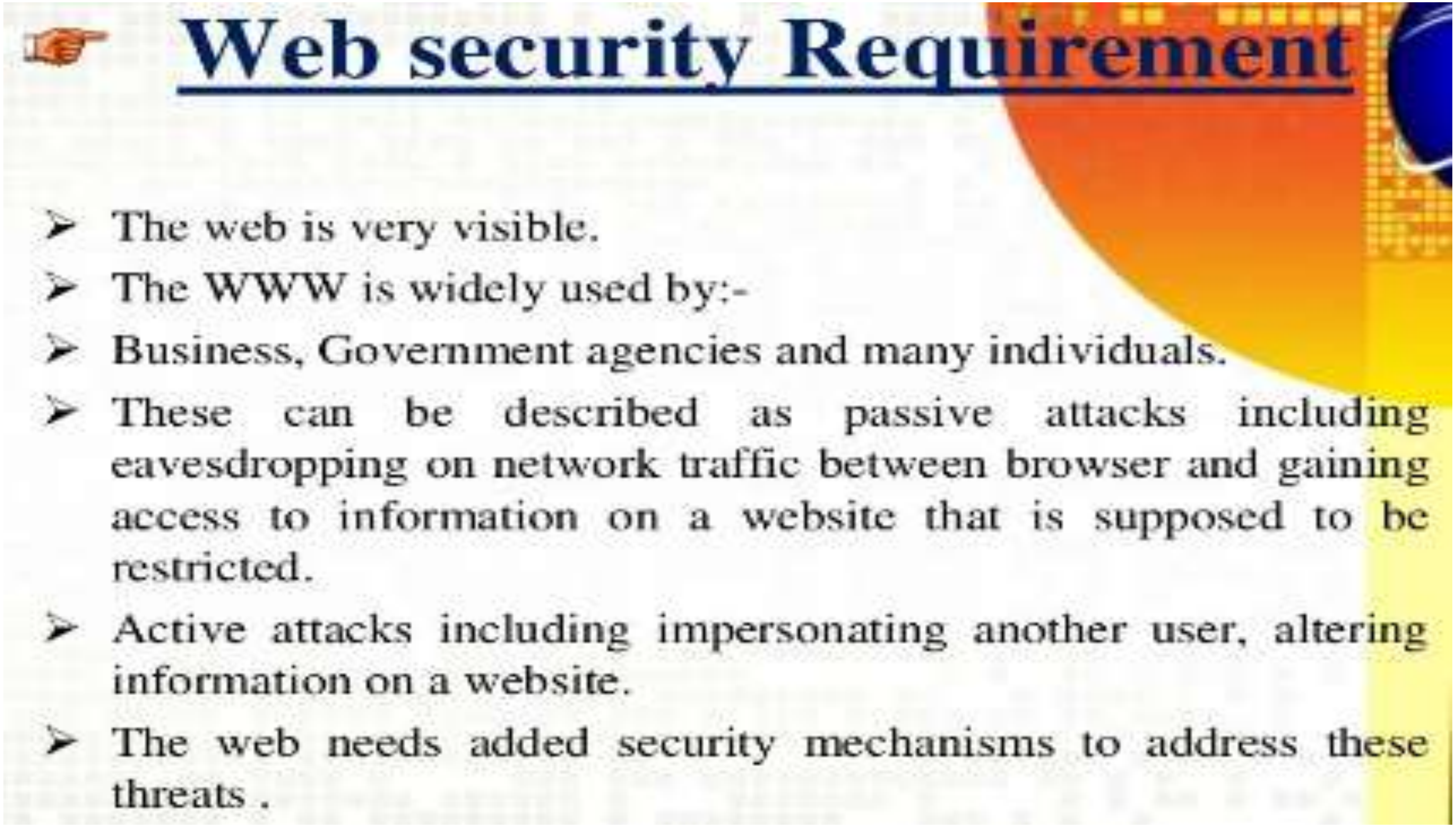
Web security is also known as “Cybersecurity”. It basically means protecting a **website** or **web** application by detecting, preventing and responding to cyber threats. Websites and **web** applications are just as prone to **security** breaches as physical homes, stores, and government locations.



WEB SECURITY :

- Measures to protect data during their transmission over a collection of interconnected networks.
- The World Wide Web is fundamentally a client/server application running over the internet and TCP/IP intranets.

WEB SECURITY THREATS



Web security Requirement

- The web is very visible.
- The WWW is widely used by:-
- Business, Government agencies and many individuals.
- These can be described as passive attacks including eavesdropping on network traffic between browser and gaining access to information on a website that is supposed to be restricted.
- Active attacks including impersonating another user, altering information on a website.
- The web needs added security mechanisms to address these threats .

WEB SECURITY THREATS

✓ Table 16.1 provides a summary of the types of security threats faced when using the Web.

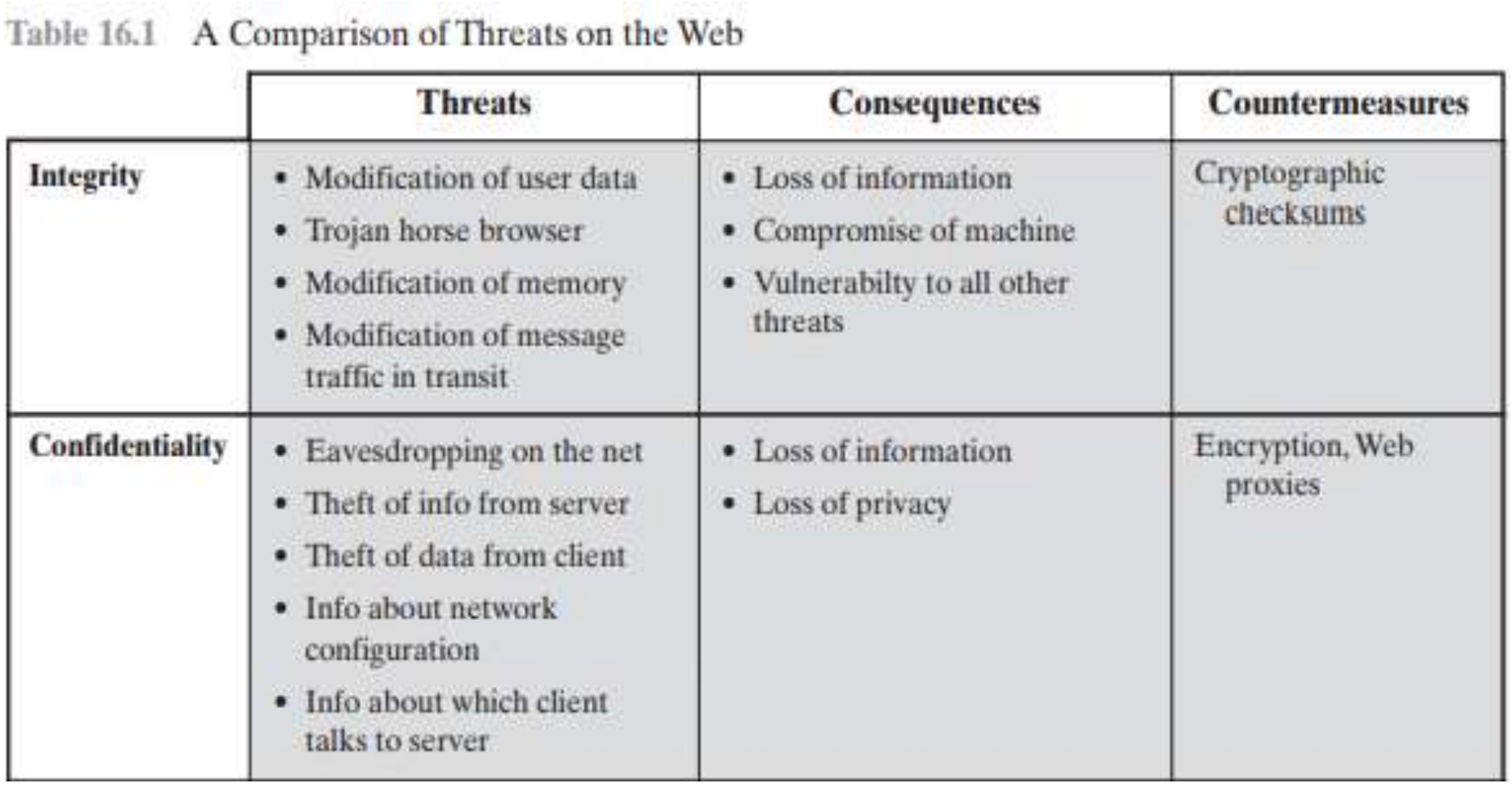
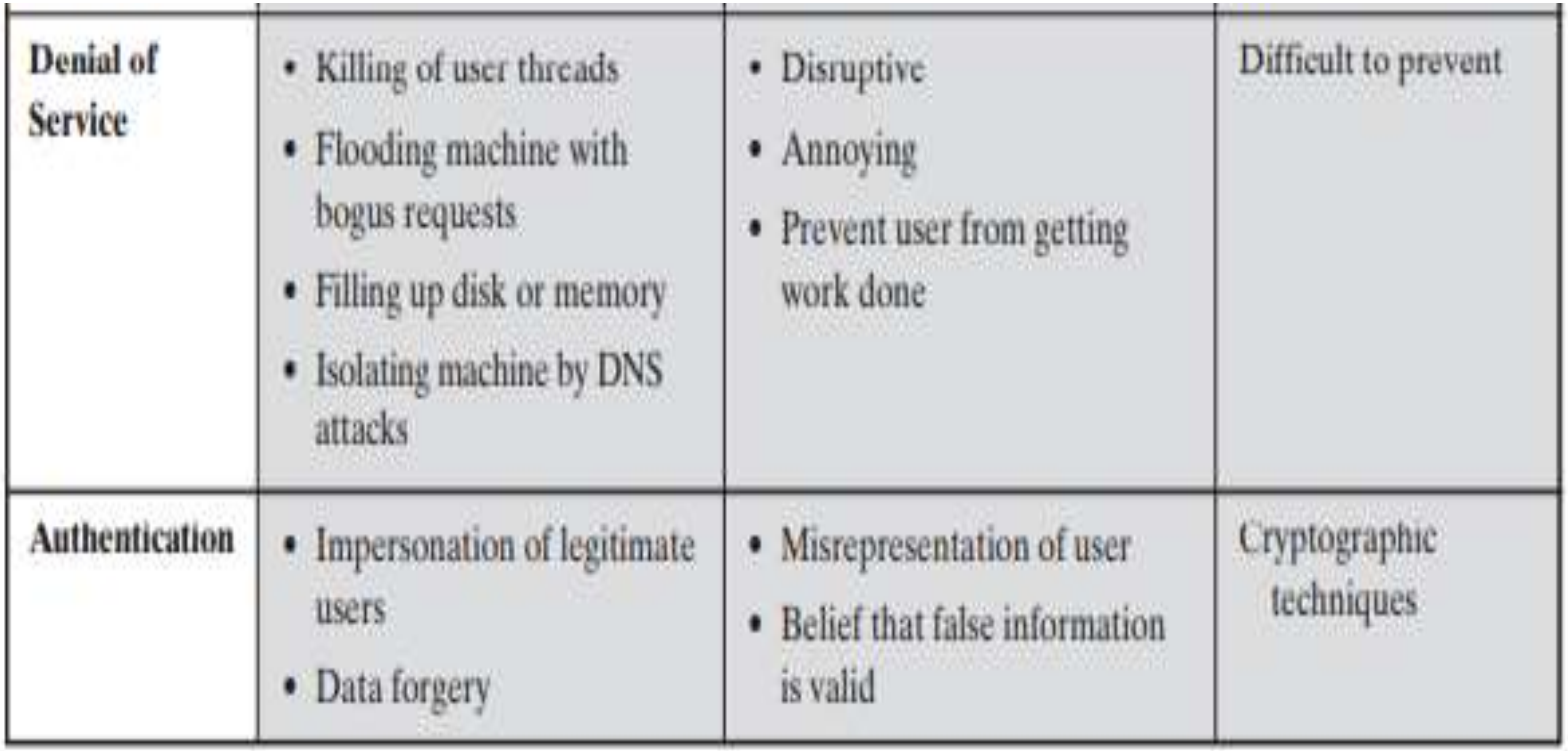


Table 16.1 A Comparison of Threats on the Web

	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

WEB SECURITY THREATS

Table 16.1 provides a summary of the types of security threats faced when using the Web.



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

WEB TRAFFIC SECURITY APPROACHES

- A number of approaches to providing Web security are possible.

- One way to provide Web security is to use IP security (IPsec).The advantage of using IPsec is that it is transparent to end users and applications and provides a general-purpose solution.

- Furthermore, IPsec includes a filtering capability so that only selected traffic need incur the overhead of IPsec processing.

- Figure 16.1c 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.

WEB TRAFFIC SECURITY APPROACHES

A number of approaches to providing web security are possible figure illustrate this difference.

1. Network Level.
2. Transport Level.
3. Application Level.

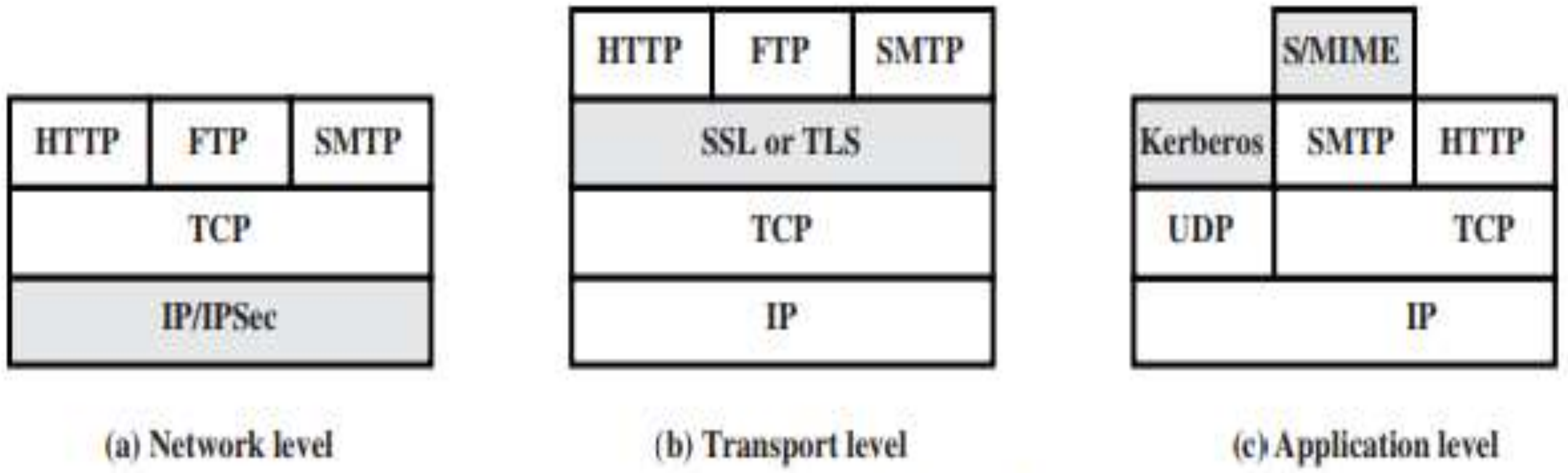


Figure 16.1 Relative Location of Security Facilities in the TCP/IP Protocol Stack

WEB TRAFFIC SECURITY APPROACHES

- ✓ IP/IPSec : The functional areas general IP Security mechanisms provides
- ✓ Authentication: This provides source authentication.
- ✓ Confidentiality: encryption
- ✓ key management: transfer of keys securely.
- ✓ applicable to use over LANs, across public & private WANs, & for the Internet.

SSL(SECURE SOCKET LAYER) 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
- ✓ As illustrated in Figure However, it is feasible to prevent the success of these attacks, usually by means of encryption. Thus, the emphasis in dealing with passive attacks is on prevention rather than detection.
- ✓ The SSL Record Protocol provides basic security 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 SSL.

- ✓ Three higher-layer protocols are defined as part of SSL:
- ➢ SSL Record Protocol
- ➢ The Handshake Protocol.
- ➢ The Change Cipher Spec Protocol
- ➢ The Alert Protocol.

SSL(SECURE SOCKET LAYER) ARCHITECTURE

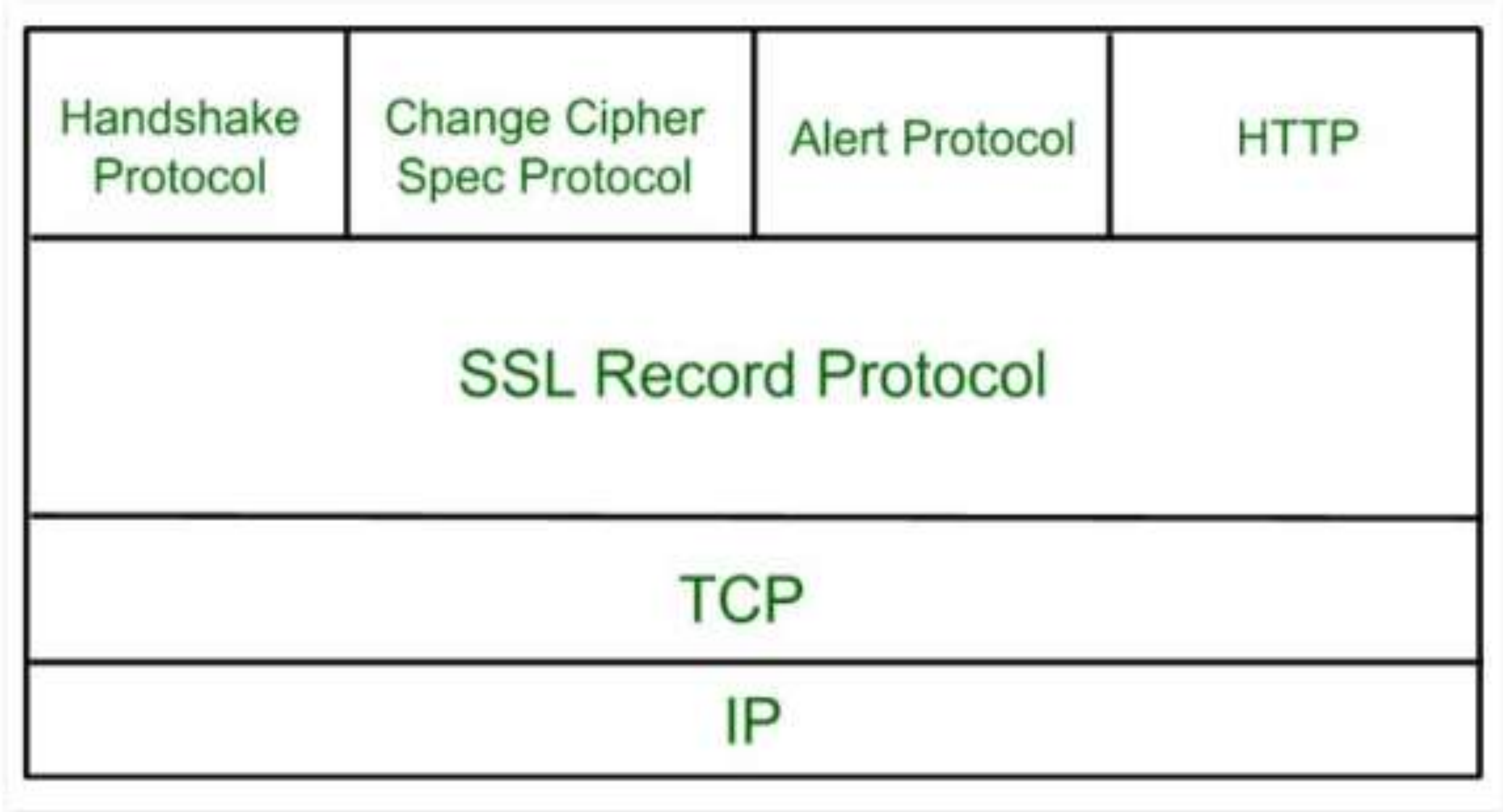
Two important SSL concepts are the SSL session and the SSL connection, which are defined in the specification as follows.

• Connection: A connection is a transport (in the OSI layering model definition) that provides a suitable type of service. For SSL, such connections are peer-to-peer relationships. The connections are transient. Every connection is associated with one session.

- 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. Sessions are used to avoid the expensive negotiation of new security parameters for each connection.

SSL(SECURE SOCKET LAYER) ARCHITECTURE

- **SSL Protocol Stack:**



SSL(SECURE SOCKET LAYER) ARCHITECTURE

SSL Record Protocol:

SSL Record provides two services to SSL connection.

- Confidentiality
- Message Integrity

In the SSL Record Protocol application data is divided into fragments.
The fragment is compressed and then encrypted MAC (Message Authentication Code) generated by algorithms like SHA (Secure Hash Protocol) and MD5 (Message Digest) is appended.

After that encryption of the data is done and in last SSL header is appended to the data.

SSL(SECURE SOCKET LAYER) ARCHITECTURE

Fragments : A small part broken off or detached(separate).

.

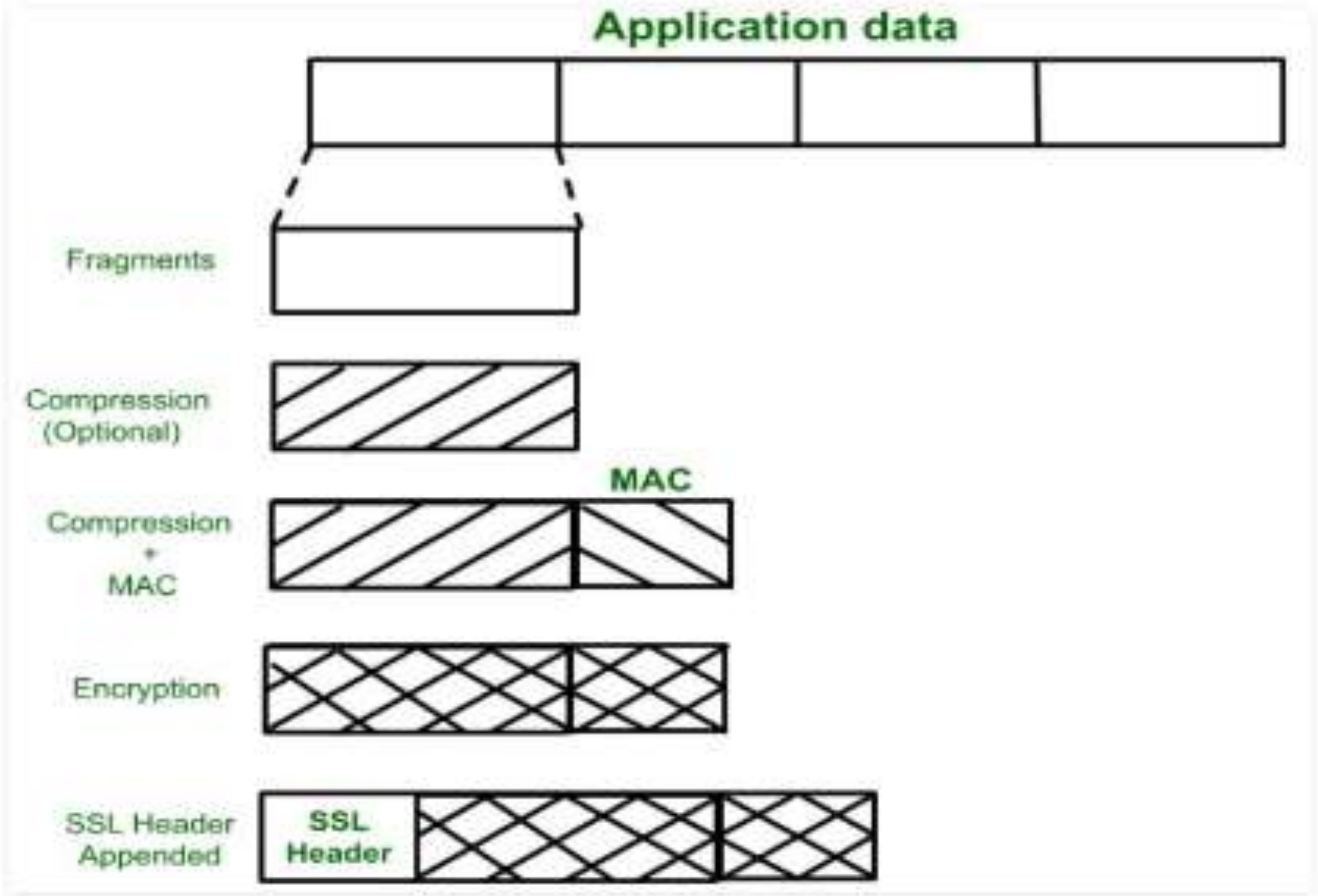
MAC (Message Authentication Code)(Data link layer) : a message authentication code (MAC), sometimes known as a tag, is a short piece of information used for authenticating a message. In other words, to confirm that the message came from the stated sender (its authenticity) and has not been changed.

SHA (Secure Hash Protocol) : The SSH protocol was developed by SSH communication security Ltd to safely communicate with the remote machine. Secure communication provides a strong password authentication and encrypted communication with a public key over an insecure channel.

MD5 (Message Digest) : The message digest is a hashing algorithm used to protect data when files are conveyed via insecure channels.

SSL(SECURE SOCKET LAYER) ARCHITECTURE

SSL Record Protocol Operation



SSL(SECURE SOCKET LAYER) ARCHITECTURE

Handshake Protocol:

Handshake Protocol is used to establish sessions. This protocol allows the client and server to authenticate each other by sending a series of messages to each other. Handshake protocol uses four phases to complete its cycle.

Phase-1: In Phase-1 - Both Client and Server send hello-packets to each other. In this IP session, cipher suite and protocol version are exchanged for security purposes.

Phase-2: Server sends his certificate and Server-key-exchange. The server end phase-2 by sending the Server-hello-end packet.

Phase-3: In this phase, Client replies to the server by sending his certificate and Client-exchange-key.

Phase-4: In Phase-4 Change-cipher suite occurred and after this Handshake Protocol ends.

SSL(SECURE SOCKET LAYER) ARCHITECTURE

Change-cipher Protocol:

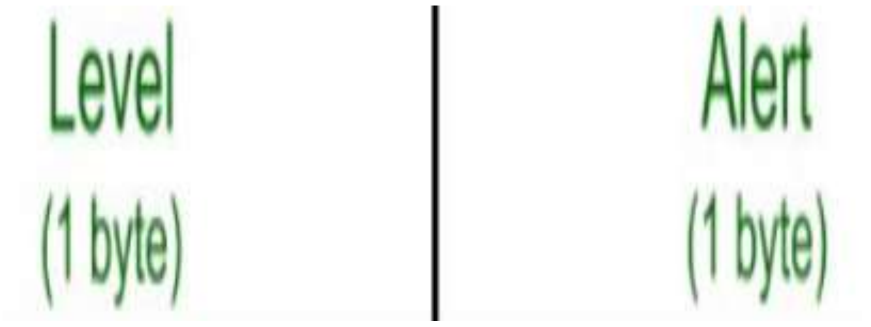
This protocol uses the SSL record protocol. Unless Handshake Protocol is completed, the SSL record Output will be in a pending state. After handshake protocol, the Pending state is converted into the current state.

Change-cipher protocol consists of a single message which is 1 byte in length and can have only one value. This protocol's purpose is to cause the pending state to be copied into the current state.

1 Byte

Alert Protocol:

This protocol is used to convey SSL-related alerts to the peer entity. Each message in this protocol contains 2 bytes.



SILENT FEATURES OF SECURE SOCKET LAYER:

- The advantage of this approach is that the service can be tailored to the specific needs of the given application.
- Secure Socket Layer was originated by Netscape.
- SSL is designed to make use of TCP to provide reliable end-to-end secure service.
- This is a two-layered protocol.

HANDSHAKE PROTOCOL(**TCP 3-WAY HANDSHAKE PROCESS**)

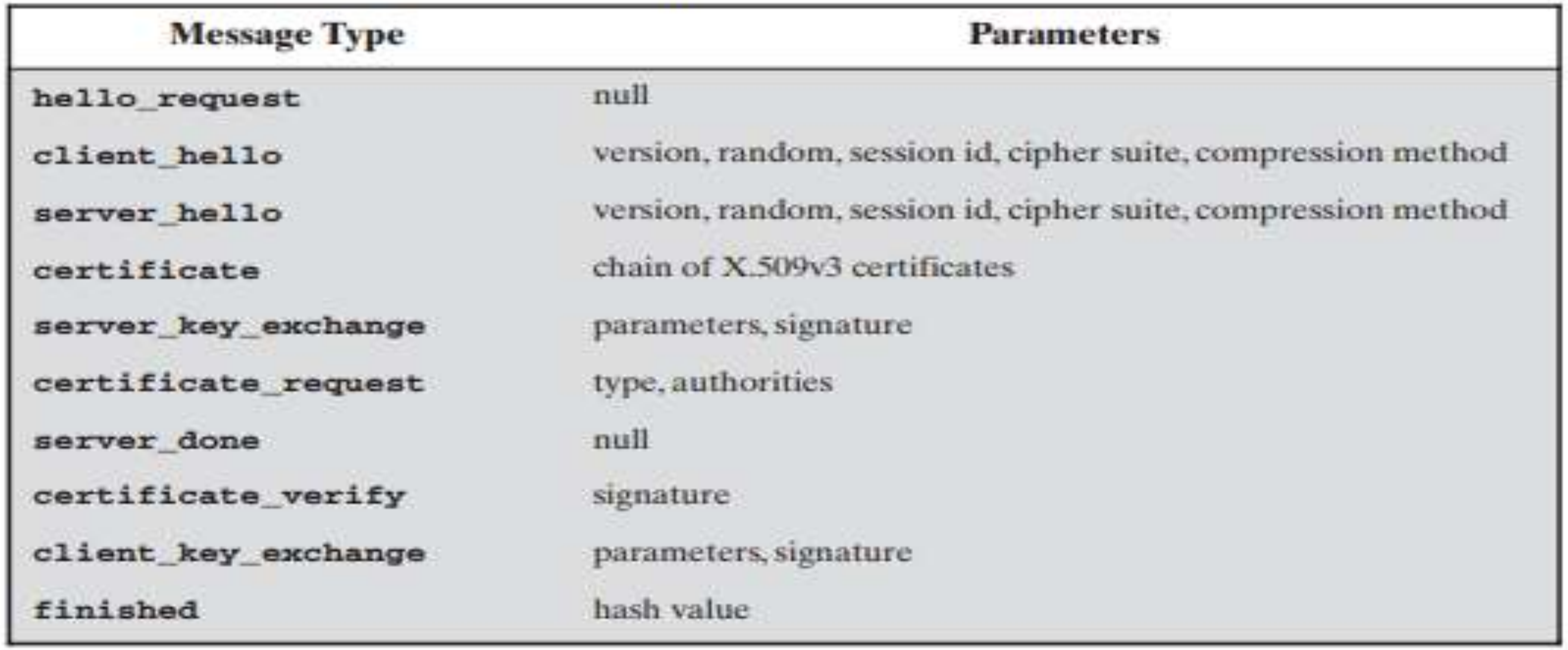
- Definition:
- A more complex handshaking protocol might allow the sender to ask the receiver if it is ready to receive or for the receiver to reply with a negative acknowledgement meaning "I did not receive your last message correctly, please resend it" (e.g., if the data was corrupted en route).
- 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.

HANDSHAKE PROTOCOL(**TCP 3-WAY HANDSHAKE PROCESS**)

Each message has three fields:

- Type (1 byte): Indicates one of 10 messages. Table 16.2 lists the defined message types.
- Length (3 bytes): The length of the message in bytes.
- Content (bytes): The parameters associated with this message these are listed in Table 16.2

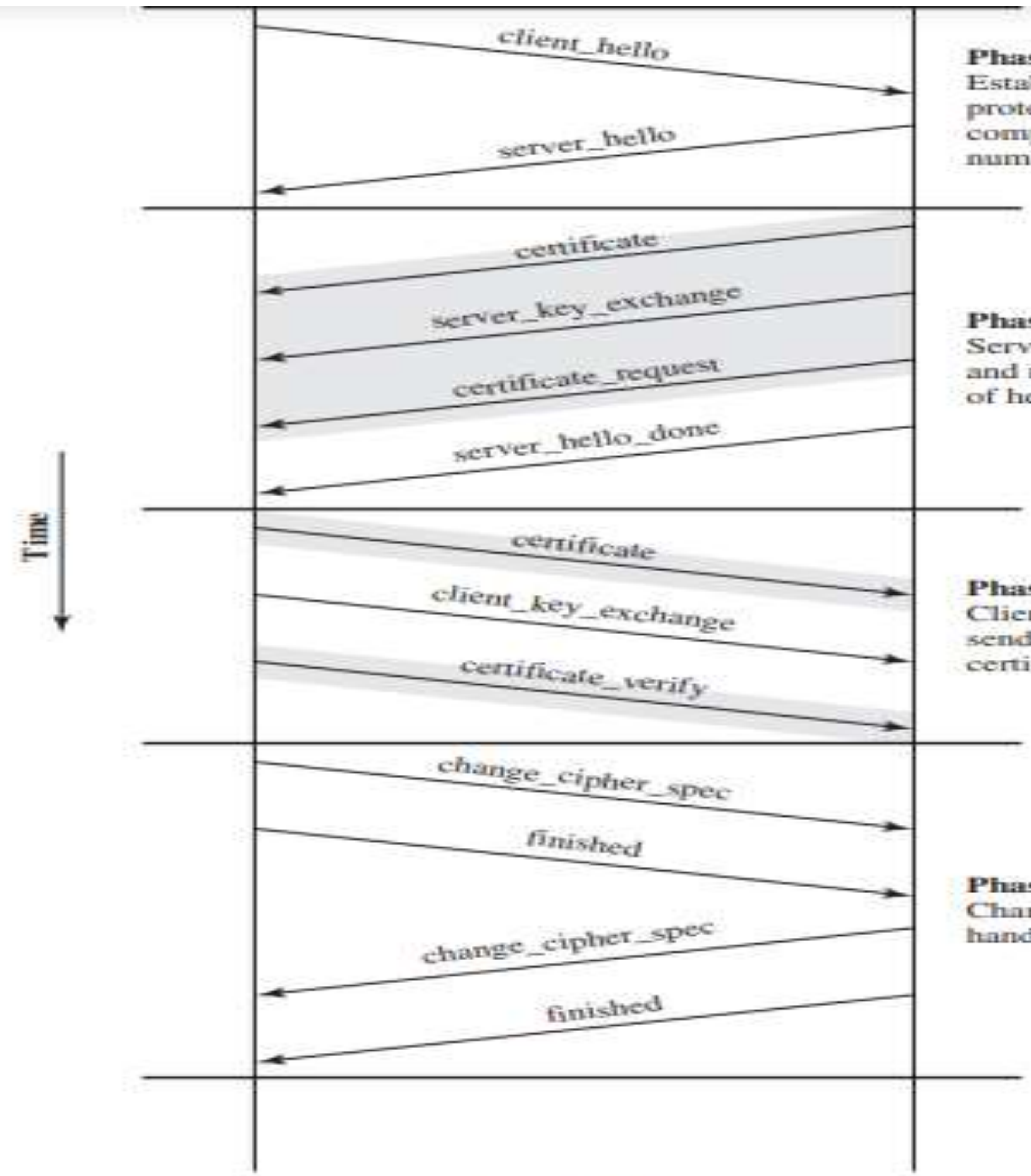
Table 16.2 SSL Handshake Protocol Message Types



Message Type	Parameters
`hello_request`	null
`client_hello`	version, random, session id, cipher suite, compression method
`server_hello`	version, random, session id, cipher suite, compression method
`certificate`	chain of X.509v3 certificates
`server_key_exchange`	parameters, signature
`certificate_request`	type, authorities
`server_done`	null
`certificate_verify`	signature
`client_key_exchange`	parameters, signature
`finished`	hash value

HANDSHAKE PROTOCOL

)



Phase 1
Establish security capabilities, including protocol version, session ID, cipher suite, compression method, and initial random numbers.

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

Phase 3
Client sends certificate if requested. Client sends key exchange. Client may send certificate verification.

Phase 4
Change cipher suite and finish handshake protocol.

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

Figure 16.6 Handshake Protocol Action

HANDSHAKE PROTOCOL(**TCP 3-WAY HANDSHAKE PROCESS**)

- ✓ PHASE 1:
- ✓ ESTABLISH SECURITY CAPABILITIES This phase is used to initiate a logical connection and to establish the security capabilities that will be associated with it. The exchange is initiated by the client, which sends a client_hello message with the following parameters:
- ✓ Version: The highest SSL version understood by the client.
- ✓ Random: A client-generated random structure consisting of a 32-bit timestamp and 28 bytes generated by a secure random number generator. These values serve as nonces and are used during key exchange to prevent replay attacks
- ✓ Session ID: A variable-length session identifier. A nonzero value indicates that the client wishes to update the parameters of an existing connection or to create a new connection on this session. A zero value indicates that the client wishes to establish a new connection on a new session

HANDSHAKE PROTOCOL(**TCP 3-WAY HANDSHAKE PROCESS**)

- ✓ PHASE 2. SERVER AUTHENTICATION AND KEY EXCHANGE
- ✓ The server begins this phase by sending its certificate if it needs to be authenticated
- ✓ The certificate message is required for any agreed-on key exchange method except anonymous Diffie-Hellman.
- ✓ A server_key_exchange message may be sent if it is required.

  It is not required in two instances:
- ✓ (1) The server has sent a certificate with fixed Diffie-Hellman parameters or
- ✓ (2) a RSA key exchange is to be used.

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. If the server has requested a certificate, the client begins this phase by sending a certificate message. If no suitable certificate is available, the client sends a no_certificate alert instead

HANDSHAKE PROTOCOL(**TCP 3-WAY HANDSHAKE PROCESS**)

- ✓ 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.

- ✓ Note that this message is not considered part of the Handshake Protocol but is sent using the Change Cipher Spec Protocol. 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. The content of the finished message is the concatenation of two hash values

- ✓ Change-cipher Protocol: This protocol uses the SSL record protocol. Unless Handshake Protocol is completed, the SSL record Output will be in a pending state. After handshake protocol, the Pending state is converted into the current state. Change-cipher protocol consists of a single message which is 1 byte in length and can have only one value.

HANDSHAKE PROTOCOL(**TCP 3-WAY HANDSHAKE PROCESS**)

- 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 these are discussed subsequently.
- Compression Method: This is a list of the compression methods the client supports

The following key exchange methods are supported.

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

KERBEROS

- Kerberos is an authentication service developed as part of Project Athena at MIT
- The problem that Kerberos addresses is this: Assume an open distributed environment in which users at workstations wish to access services on servers distributed throughout the network. We would like for servers to be able to restrict access to authorized users and to be able to authenticate requests for service.
- In this environment, a workstation cannot be trusted to identify its users correctly to network services.
- In particular, the following three threats exist:
- 1. A user may gain access to a particular workstation and pretend to be another user operating from that workstation.
- 2. 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.
- 3. A user may eavesdrop on exchanges and use a replay attack to gain entrance to a server or to disrupt operations

KERBEROS

- In any of these cases, an unauthorized user may be able to gain access to services and data that he or she is not authorized to access. Rather than building in elaborate authentication protocols at each server, Kerberos provides a centralized authentication server whose function is to authenticate users to servers and servers to users.

In this environment, three approaches to security can be envisioned :

- 1. Rely on each individual client workstation to assure the identity of its user or users and rely on each server to enforce a security policy based on user identification (ID).
- 2. Require that client systems authenticate themselves to servers, but trust the client system concerning the identity of its user.
- 3. Require the user to prove his or her identity for each service invoked. Also require that servers prove their identity to clients

KERBEROS

- ✓ The first published report on Kerberos [STEI88] listed the following 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. This suggests a modular, distributed architecture.

KERBEROS VERSION 4

- Version 4 of Kerberos makes use of DES, in a rather elaborate protocol, to provide the authentication service.
- A SIMPLE AUTHENTICATION DIALOGUE In an unprotected network environment, any client can apply to any server for service. The obvious security risk is that of impersonation.

- An opponent can pretend to be another client and obtain unauthorized privileges on server machines. To counter this threat, servers must be able to confirm the identities of clients who request service. Each server can be required to undertake this task for each client/server interaction, but in an open environment, this places a substantial burden on each server.

- An alternative is to use an authentication server (AS) that knows the passwords of all users and stores these in a centralized database. In addition, the AS shares a unique secret key with each server. These keys have been distributed physically or in some other secure manner.

KERBEROS VERSION 4

Consider the following hypothetical dialogue:

(1) C → AS: $ID_C \| P_C \| ID_V$

(2) AS → C: $Ticket$

(3) C → V: $ID_C \| Ticket$

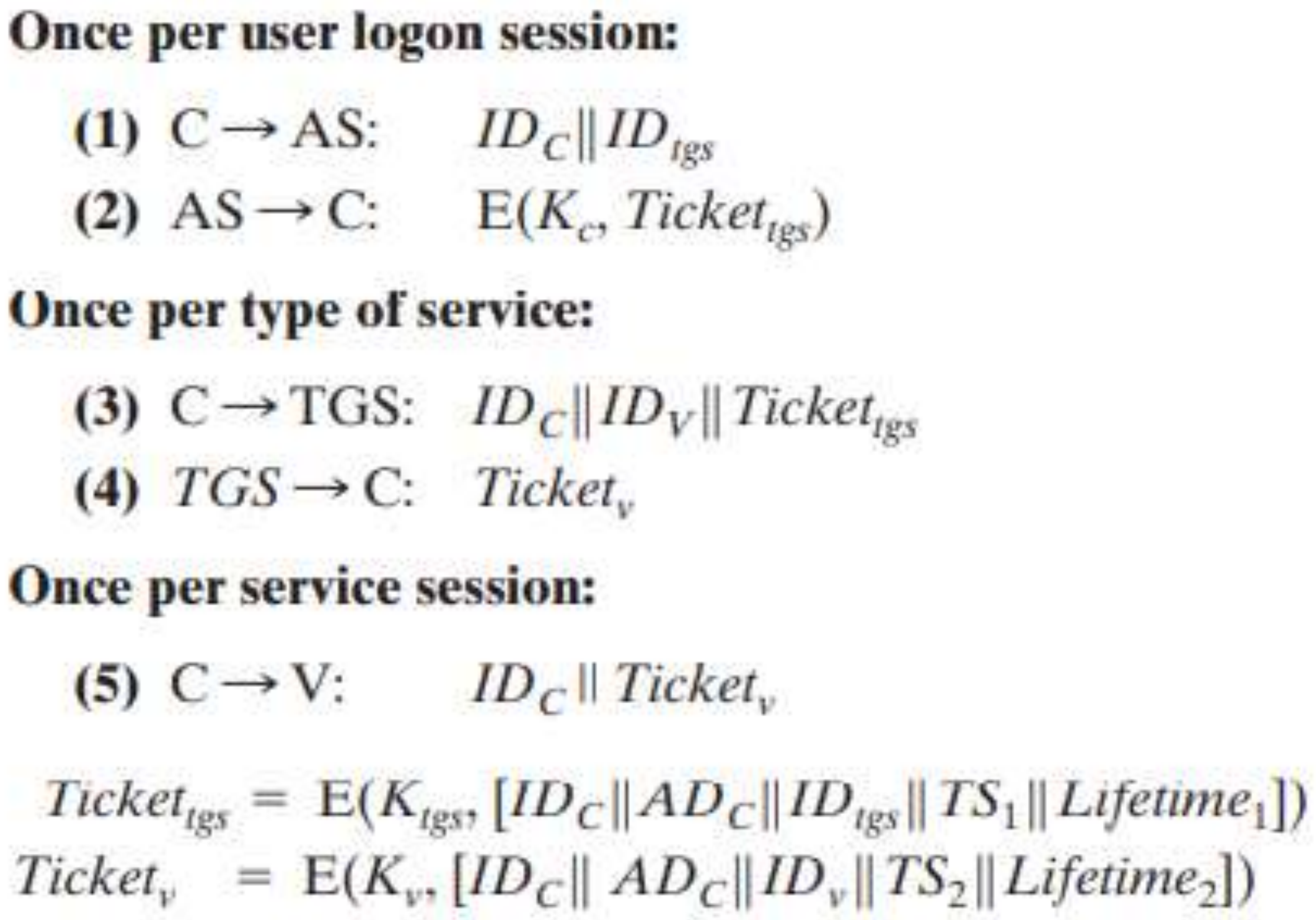
$$Ticket = E(K_v, [ID_C \| AD_C \| ID_V])$$

where

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

KERBEROS VERSION 4

The problem is that the earlier scenario involved a plaintext transmission of the password. An eavesdropper could capture the password and use any service accessible to the victim. To solve these additional problems, we introduce a scheme for avoiding plaintext passwords and a new server, known as the ticket-granting server (TGS).



Once per user logon session:

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

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

Once per type of service:

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

(4) $TGS \rightarrow \text{C}$: $Ticket_v$

Once per service session:

(5) $\text{C} \rightarrow \text{V}$: $ID_C \| Ticket_v$

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

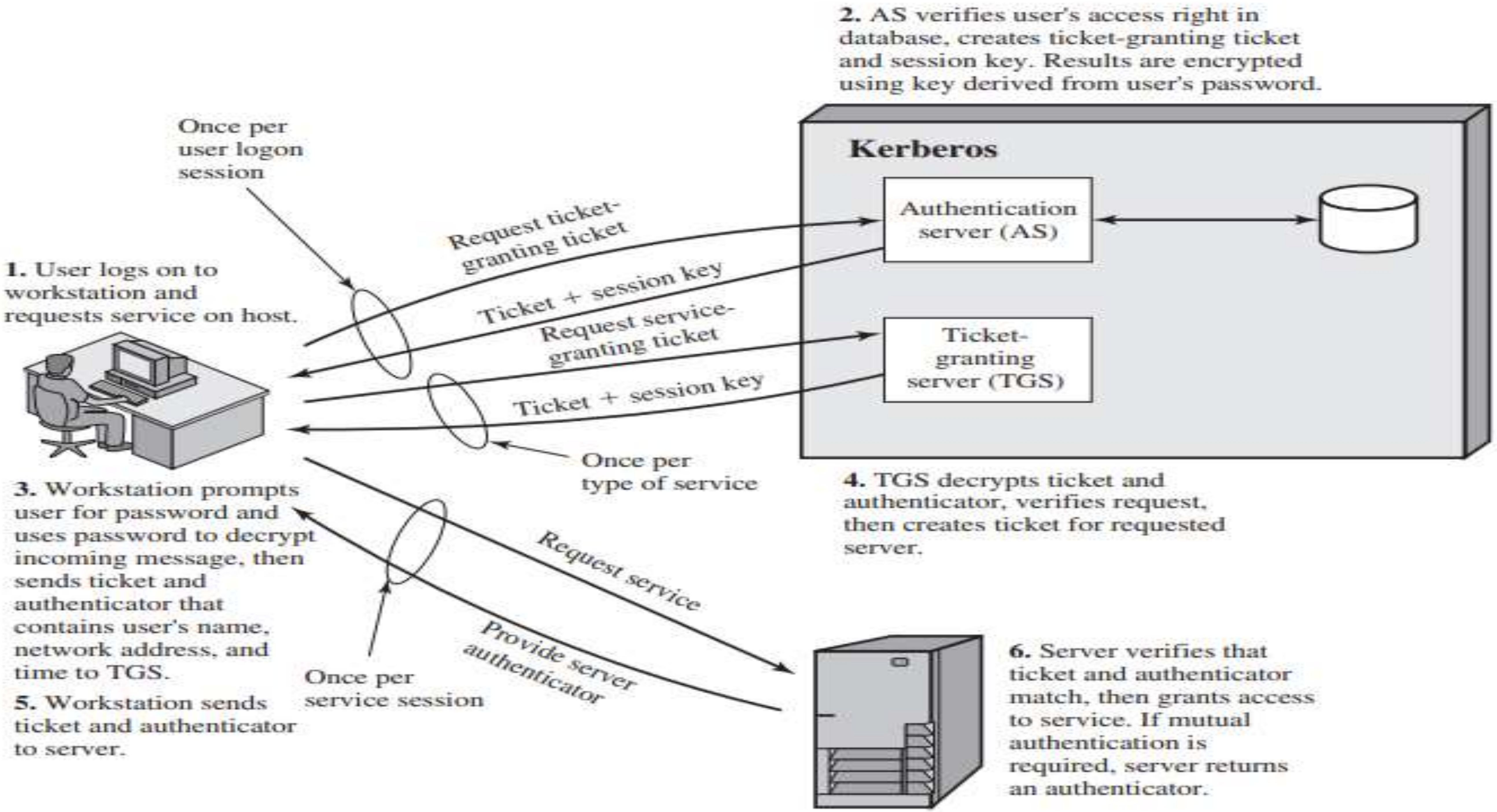
KERBEROS VERSION 4

- 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. 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

OVER VIEW OF KERBEROS


Once per user logon session
Kerberos
Authentication server (AS)
Request ticket-granting ticket
Ticket + session key
Request service-granting ticket
Ticket + session key
Ticket-granting server (TGS)
Once per type of service
Request service
Provide server authenticator
Once per service session

1. User logs on to workstation and requests service on host.

2. AS verifies user's access right in database, creates ticket-granting ticket and session key. Results are encrypted using key derived from user's password.

3. Workstation prompts user for password and uses password to decrypt incoming message, then sends ticket and authenticator that contains user's name, network address, and time to TGS.

4. TGS decrypts ticket and authenticator, verifies request, then creates ticket for requested server.

5. Workstation sends ticket and authenticator to server.

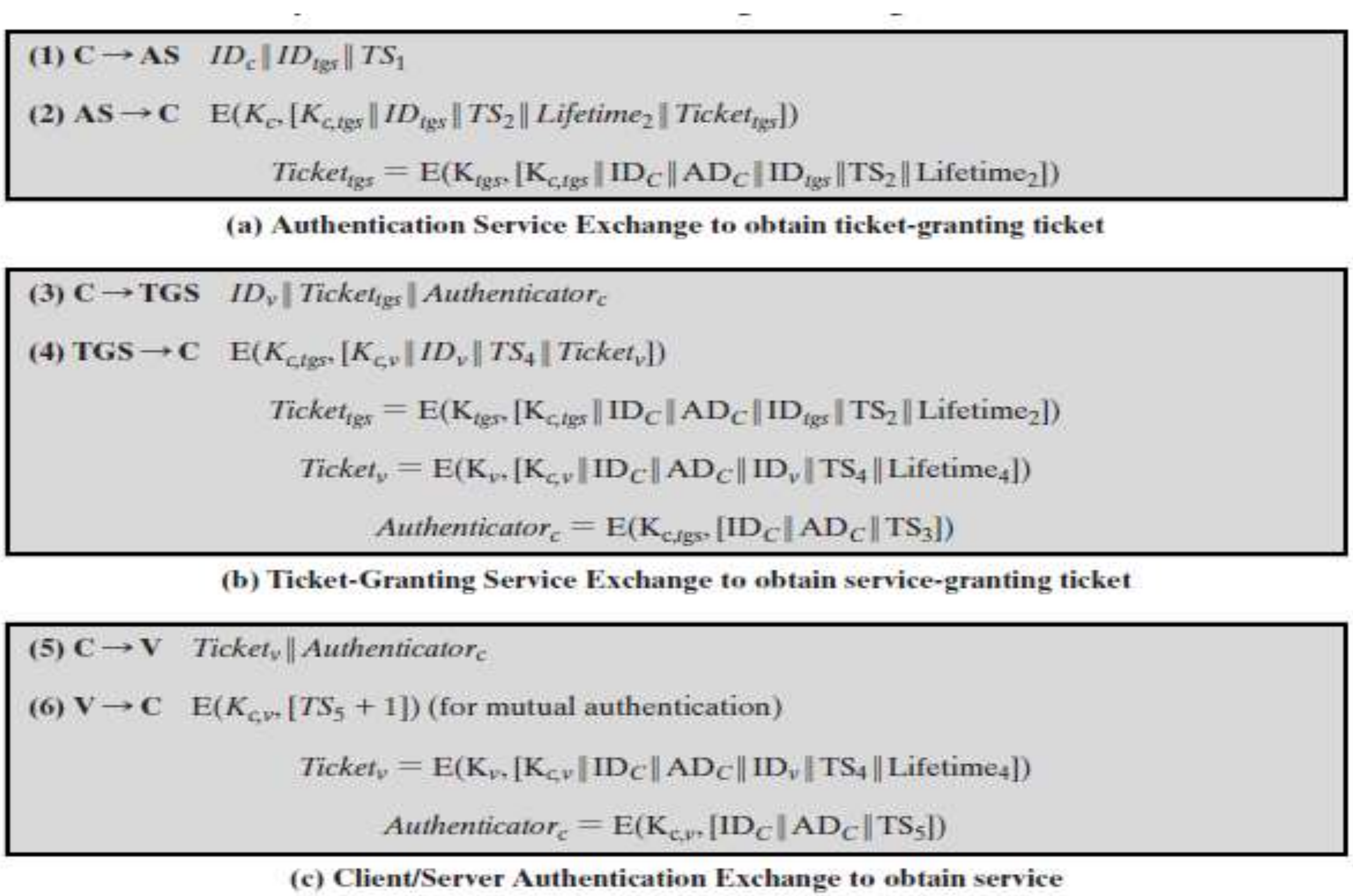
6. Server verifies that ticket and authenticator match, then grants access to service. If mutual authentication is required, server returns an authenticator.

Figure 15.1 Overview of Kerberos

OVER VIEW OF KERBEROS STEPS

- **Step-1:**
  User login and request services on the host. Thus user requests for ticket-granting service.
- **Step-2:**
  Authentication Server verifies user's access right using database and then gives ticket-granting-ticket and session key. Results are encrypted using the Password of the user.
- **Step-3:**
  The decryption of the message is done using the password then send the ticket to Ticket Granting Server. The Ticket contains authenticators like user names and network addresses.
- **Step-4:**
  Ticket Granting Server decrypts the ticket sent by User and authenticator verifies the request then creates the ticket for requesting services from the Server.
- **Step-5:**
  The user sends the Ticket and Authenticator to the Server.
- **Step-6:**
  The server verifies the Ticket and authenticators then generate access to the service. After this User can access the services.

THE VERSION 4 AUTHENTICATION DIALOGUE



(1) C → AS $ID_c \| ID_{tgs} \| TS_1$

(2) AS → C $E(K_c, [K_{c,tgs} \| ID_{tgs} \| TS_2 \| Lifetime_2 \| Ticket_{tgs}])$

$Ticket_{tgs} = E(K_{tgs}, [K_{c,tgs} \| ID_C \| AD_C \| ID_{tgs} \| TS_2 \| Lifetime_2])$

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

(3) C → TGS $ID_v \| Ticket_{tgs} \| Authenticator_c$

(4) TGS → C $E(K_{c,tgs}, [K_{c,v} \| ID_v \| TS_4 \| Ticket_v])$

$Ticket_{tgs} = E(K_{tgs}, [K_{c,tgs} \| ID_C \| AD_C \| ID_{tgs} \| TS_2 \| Lifetime_2])$

$Ticket_v = E(K_v, [K_{c,v} \| ID_C \| AD_C \| ID_v \| TS_4 \| Lifetime_4])$

$Authenticator_c = E(K_{c,tgs}, [ID_C \| AD_C \| TS_3])$

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

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

(6) V → C $E(K_{c,v}, [TS_5 + 1])$ (for mutual authentication)

$Ticket_v = E(K_v, [K_{c,v} \| ID_C \| AD_C \| ID_v \| TS_4 \| Lifetime_4])$

$Authenticator_c = E(K_{c,v}, [ID_C \| AD_C \| TS_5])$

(c) Client/Server Authentication Exchange to obtain service

SUMMARY OF THE LECTURE

Authentication Applications: Kerberos, X.509 Directory Authentication Service.

Transport level Security,
Web Security Considerations:
Web Security Threats,
Web Traffic Security Approaches,
SSL Architecture,
SSL Record Protocol

Change Cipher Spec Protocol, Alert Protocol, Handshake Protocol
Electronic Mail Security: Pretty Good Privacy (PGP); S/MIME.

DISCUSSION

discussion
students
group
questions
class
person
time
way
participants
conversation
minutes
comment
people
think
one
Discussion
question
know
comments
groups
critical
tolerance
newsprint
ideas
new
another
different
detectives
responses
good
made
DISCUSSION
others
Students
someone
practice
contributions
storyteller
learning
make
just
things
course
work
give
first
get
two
possible
team
view
Marcuse
room
around
ask
also
power
small
assumptions
point
want
asked
help
repressive
speak
thinking
something
important
next
feel
really
like
quote
use
John
see
rule
words
talk
might
specific
exercise
circle