USN | | | | | | | | | | |

CMRIT — CMR INSTITUTE OF TECHNOLOGY, BENGALURU. CELEBRATING 25 YEARS. ACCREDITED WITH A+ GRADE BY NAAC

Internal Assessment Test 1 – May. 2022

Sub:	Network Security				Sub Code:	18EC821	Branch:	ECE
Date:	14-05-22	Duration:	90 min's	Max Marks: 50	Sem / Sec:	8 – A, B, C, D		OBE

Answer any FIVE FULL Questions

		MARKS	CO	RBT
1.	Explain the need for security, and also write a note on modern nature of attacks.	[10]	CO1	L2
2.	Describe the following: Confidentiality, Authentication, Integrity, Non-repudiation, Availability as applied to network security	[10]	CO1	L2
3.	Discuss the Passive and Active attacks in detail.	[10]	CO1	L2
4.	Infer on web traffic security approaches.	[10]	CO2	L2
5.	Differentiate between SSL Connection and SSL Session?	[10]	CO2	L2
6.	Illustrate SSL Handshake Protocol Action with neat diagram.	[10]	CO2	L3
7.	Explain Pseudorandom Function and Alert Codes in TLS.	[10]	CO2	L2
8.	Briefly explain HTTPS, connection initiation and connection closure in HTTPS.	[10]	CO2	L1

USN | | | | | | | | | | |

CMRIT — CMR INSTITUTE OF TECHNOLOGY, BENGALURU. CELEBRATING 25 YEARS. ACCREDITED WITH A+ GRADE BY NAAC

Internal Assessment Test 1 – May. 2022

Sub:	Network Security				Sub Code:	18EC821	Branch:	ECE
Date:	14-05-22	Duration:	90 min's	Max Marks: 50	Sem / Sec:	8 – A, B, C, D		OBE

Answer any FIVE FULL Questions

		MARKS	CO	RBT
1.	Explain the need for security, and also write a note on modern nature of attacks.	[10]	CO1	L2
2.	Describe the following: Confidentiality, Authentication, Integrity, Non-repudiation, Availability as applied to network security	[10]	CO1	L2
3.	Discuss the Passive and Active attacks in detail.	[10]	CO1	L2
4.	Infer on web traffic security approaches.	[10]	CO2	L2
5.	Differentiate between SSL Connection and SSL Session?	[10]	CO2	L2
6.	Illustrate SSL Handshake Protocol Action with neat diagram.	[10]	CO2	L3
7.	Explain Pseudorandom Function and Alert Codes in TLS.	[10]	CO2	L2
8.	Briefly explain HTTPS, connection initiation and connection closure in HTTPS.	[10]	CO2	L1

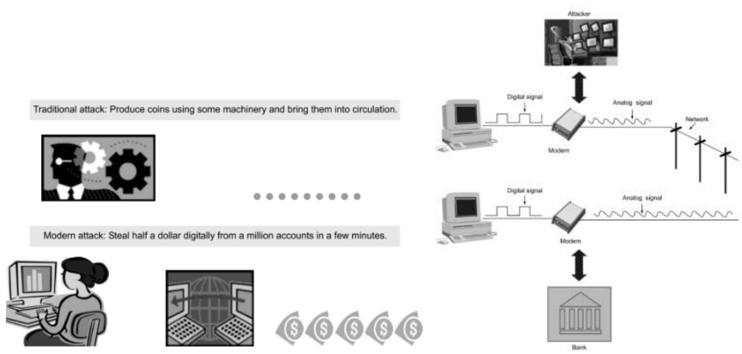
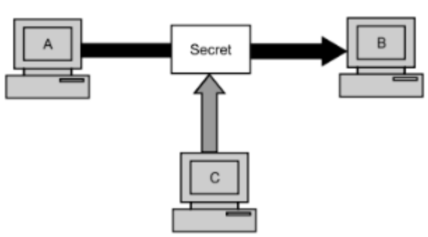
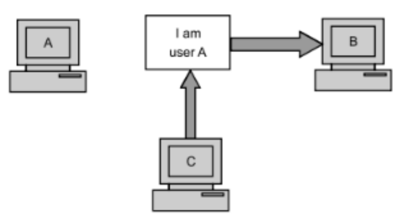
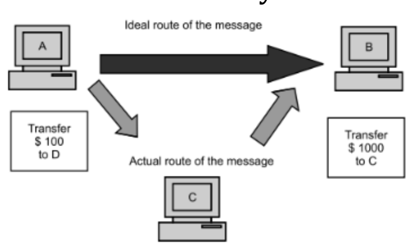
CMR
INSTITUTE OF
TECHNOLOGY

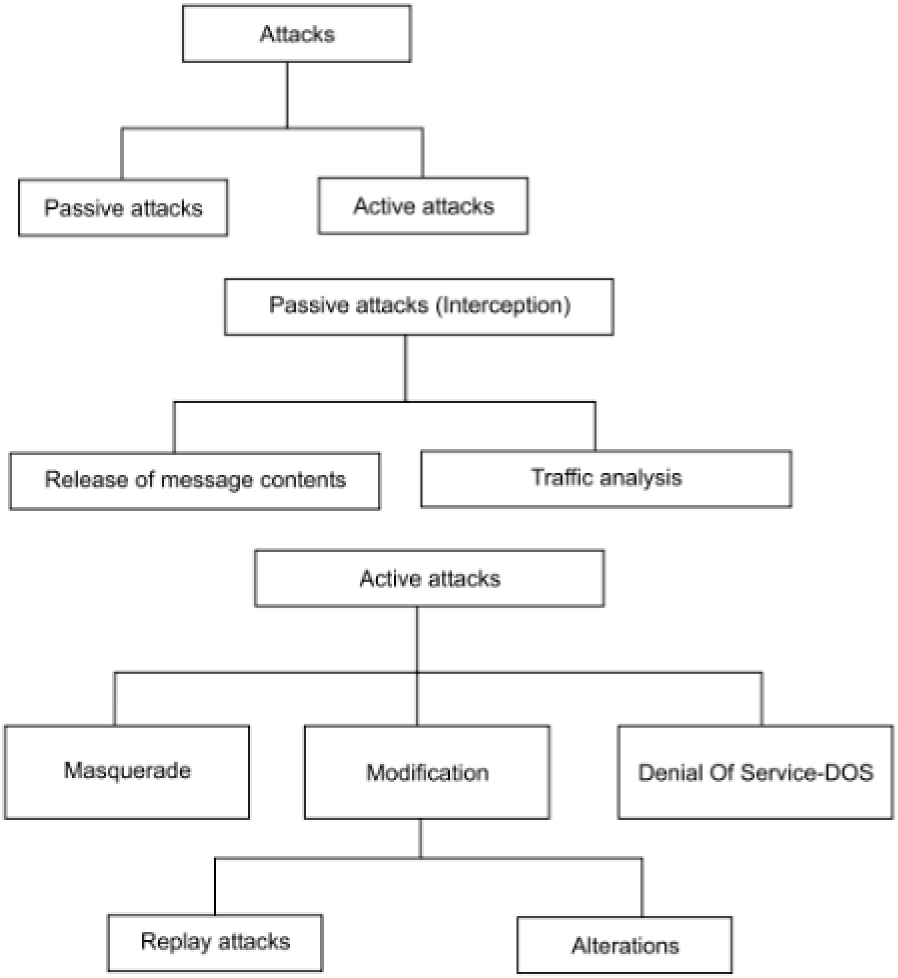
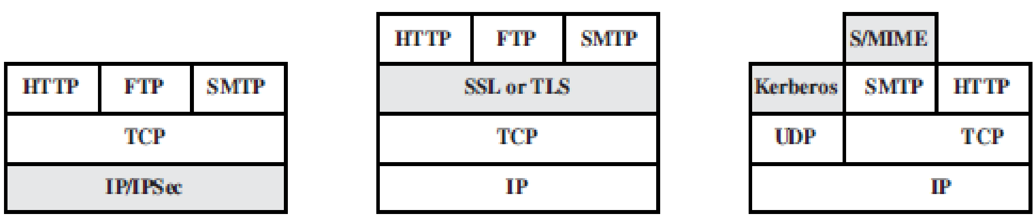
Internal Assesment Test – I MAY 2022

Sub:	Network Security							Code:	18EC821
Date:	14/05/2022	Duration:	90 mins	Max Marks:	50	Sem:	8	Branch:	ECE

Answer any five questions out of eight questions

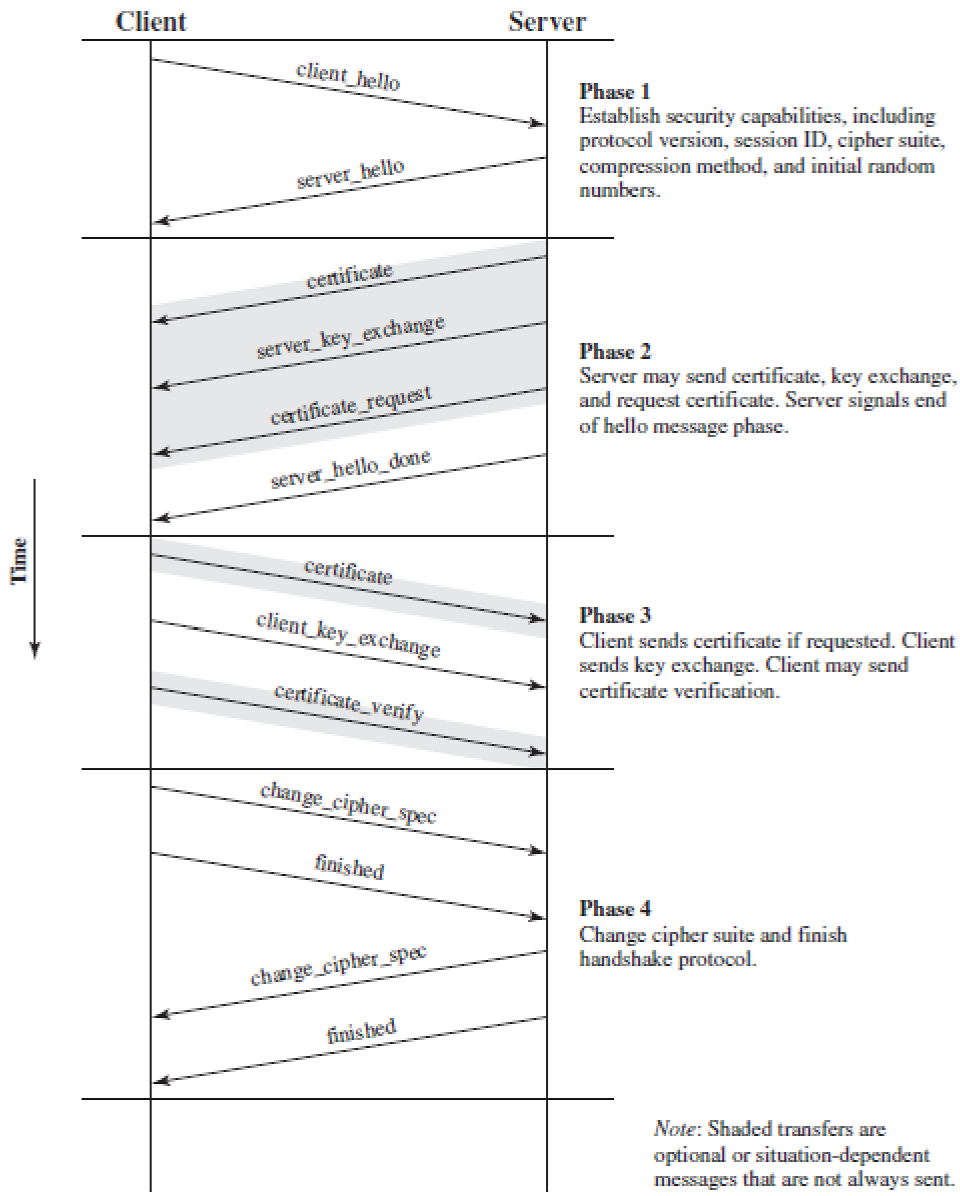
SCHEME OF VALUATION

Q. no.	Questions	Marks
1.	Most previous computer applications had *no*, or at best, *very little* security. This continued for a number of years until the importance of data was truly realized. Until then, computer data was considered to be useful, but not something to be protected. When computer applications were developed to handle financial and personal data, the real need for security was felt like never before. People realized that data on computers is an extremely important aspect of modern life. Therefore, various areas in security began to gain prominence. Two typical examples of such security mechanisms were as follows: ● Provide a user identification and password to every user, and use that information to authenticate a user. ● Encode information stored in the databases in some fashion, so that it is not visible to users who do not have the right permission.	4M
	a. Automating Attacks b. Privacy Concerns c. Distance Does not Matter 	6M
2.	Confidentiality, Authentication, Integrity, Non-repudiation  Confidentiality Authentication  Integrity Non-repudiation	4x2=8M
	Confidentiality, Authentication, Integrity, Non-repudiation (Explaination)	2M

3.		4M
	Explanation of passive and active attacks	6M
4	 (a) Network level (b) Transport level (c) Application level	6M
	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.	4M
5.	**SSL Connection:** A transient peer-to-peer communications link. Each connection is associated with one SSL session. **SSL Session:** A session is an association between client and server. It is created by the Handshake Protocol.	2x5=10M

It defines a set of security parameters.
It may be shared by multiple SSL connections.
It is useful to avoid expensive negotiations of security parameters for each connection. Single session has many connections. Every connection has a different key

6.



Handshake Protocol Action

6M

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

4M

7.	**Alert Codes** TLS supports all of the alert codes defined in SSLv3 with the exception of `no_certificate`. A number of additional codes are defined in TLS; of these, the following are always fatal. • **record_overflow:** A TLS record was received with a payload (ciphertext) whose length exceeds $2^{14}+2048$ bytes, or the ciphertext decrypted to a length of greater than $2^{14}+1024$ bytes. • **unknown_ca:** A valid certificate chain or partial chain was received, but the certificate was not accepted because the CA certificate could not be located or could not be matched with a known, trusted CA. • **access_denied:** A valid certificate was received, but when access control was applied, the sender decided not to proceed with the negotiation. • **decode_error:** A message could not be decoded, because either a field was out of its specified range or the length of the message was incorrect. • **protocol_version:** The protocol version the client attempted to negotiate is recognized but not supported. • **insufficient_security:** Returned instead of `handshake_failure` when a negotiation has failed specifically because the server requires ciphers more secure than those supported by the client. • **unsupported_extension:** Sent by clients that receive an extended server hello containing an extension not in the corresponding client hello. • **internal_error:** An internal error unrelated to the peer or the correctness of the protocol makes it impossible to continue. • **decrypt_error:** A handshake cryptographic operation failed, including being unable to verify a signature, decrypt a key exchange, or validate a finished message. • **user_canceled:** This handshake is being canceled for some reason unrelated to a protocol failure. • **no_renegotiation:** Sent by a client in response to a hello request or by the server in response to a client hello after initial handshaking. Either of these messages would normally result in renegotiation, but this alert indicates that the sender is not able to renegotiate. This message is always a warning.	10M
8.	HTTPS (HTTP over SSL) refers to the combination of HTTP and SSL 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. For example, search engines do not support HTTPS. The principal difference seen by a user of a Web browser is that 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. 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	2M

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. Normal HTTP behavior, including retained connections, should be followed.

We need to be clear that there are three levels of awareness of a connection in HTTPS. At the HTTP level, an HTTP client requests a connection to an HTTP server by sending a connection request to the next lowest layer. Typically, the next lowest layer is TCP, but it also may be TLS/SSL. 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. As we have seen, a TLS request to establish a connection begins with the establishment of a TCP connection between the TCP entity on the client side and the TCP entity on the server side.

4M

Connection Closure

An HTTP client or server can indicate the closing of a connection by including the following line in an 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 on the remote side, which will involve closing the underlying TCP connection. At the TLS level, the proper way to close a connection is for each side to use the TLS alert protocol to send a `close_notify` alert. TLS implementations must initiate an exchange of closure alerts before closing a connection. A TLS implementation may, after sending a closure alert, close the connection without waiting for the peer to send its closure alert, generating an "incomplete close". Note that an implementation that does this may choose to reuse the session. This should only be done when the application knows (typically through detecting HTTP message boundaries) that it has received all the message data that it cares about.

HTTP clients also must be able to cope with a situation in which the underlying TCP connection is terminated without a prior `close_notify` alert and without a `Connection: close` indicator. Such a situation could be due to a programming

4M