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

Internal Assessment Test 1 – November 2025

Sub:	Cryptography and Network Security					Sub Code:	BCS703	Branch:	CSE	
Date:	02.12.2025	Duration:	90 mins	Max Marks:	50	Sem / Sec:	VII (A, B & C)		OBE	

Answer any FIVE FULL Questions

		MARKS	CO	RBT
1 (a)	Define Hash Function. With the help of neat diagrams, explain the applications of Hash Function	6	CO3	L2

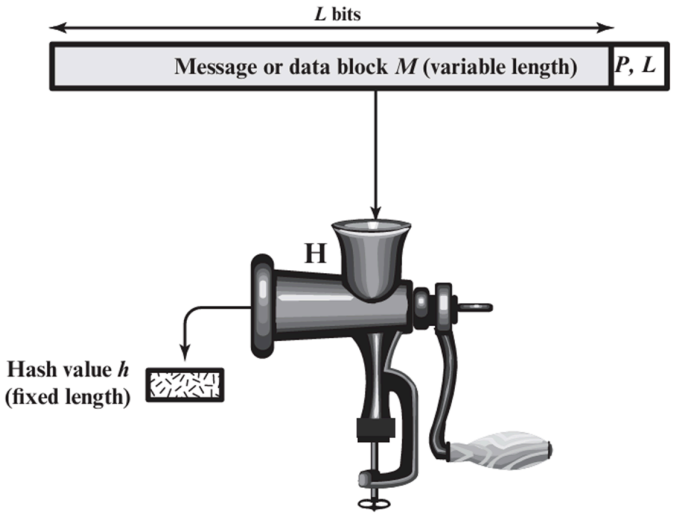
SCHEME & SOLUTION
Definition -2 Marks, Applications - 4 Marks

Hash Functions

- **Hash Function *H*** accepts a variable-length block of data ***M*** as input and produces a fixed-size hash value.

$$h = \mathrm{H}(M)$$

- Provides data integrity.
- A change to any bit or bits in ***M*** results in a change in the hash value.
- Hash function needed for security applications is referred to as a **cryptographic hash function**.



P, L = padding plus length field

Figure 11.1 Cryptographic Hash Function; $h = \mathrm{H}(M)$

Applications of Cryptographic Hash Functions:

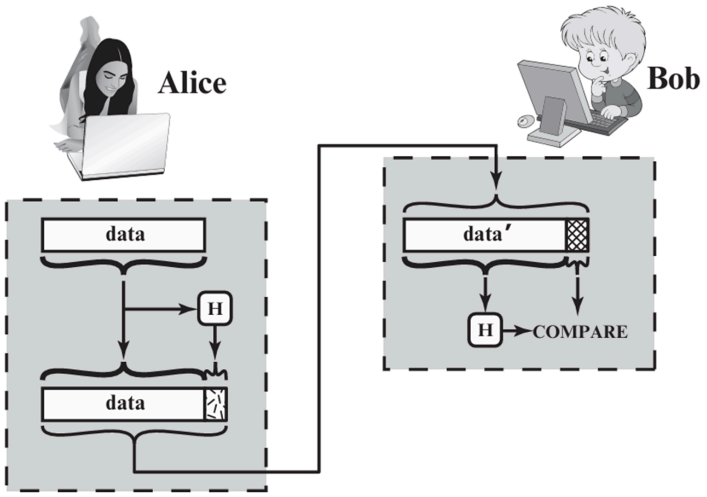
Message Authentication

•It is a mechanism or service used to verify the integrity of a message.

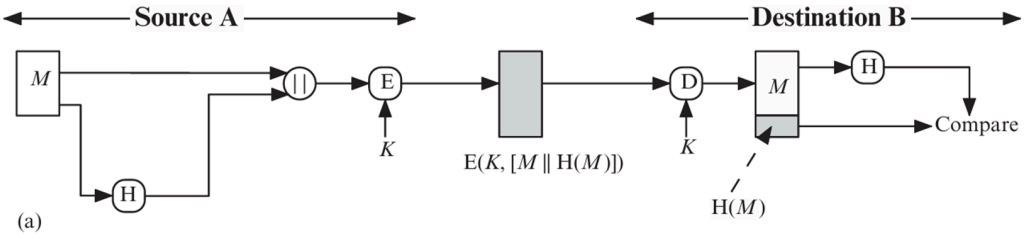
•Assures that data received is exactly as sent.

•When hash function is used for authentication, it is referred as **message digest**.

•The sender computes a hash function of the bits in the message and transmits both the hash value and the message. The receiver performs the same hash calculation on the message bits and compares this value with the incoming has value. If there is a mismatch, the receiver knows that the message (or hash value) has been altered.



(a) Use of hash function to check data integrity



Digital Signatures

•The hash value of the message is encrypted with a user's private key.

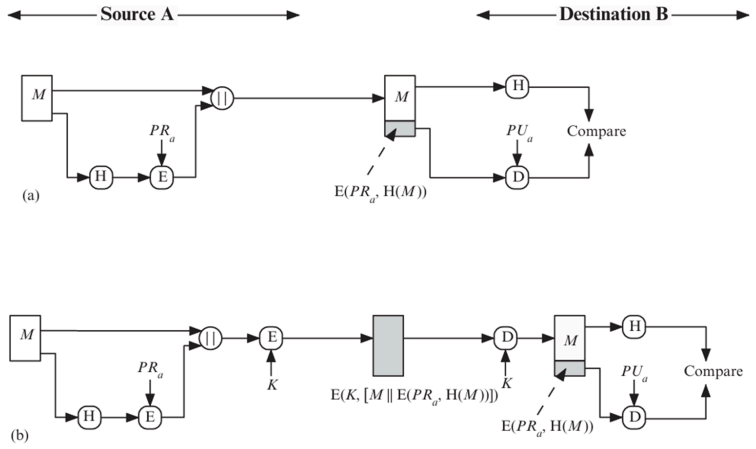


Figure 11.4 Simplified Examples of Digital Signatures

One-way Password file

•Hash of the passwords are stored by an operating system rather than password itself.

•When a user enters a password, hash of that password is compared to the stored has value for verification.

•Intrusion detection and virus detection

•Store **H(F)** for each file on a system and secure the hash value.

•One can determine if a file has been modified by recomputing **H(F).**

1 (b)	Explain Public Key Authority and Public Key Certificate techniques to distribute the public keys.	4	CO4	L2

SCHEME & SOLUTION
Public Key Authority - 2 Marks, Public Key Certificate - 2 Marks

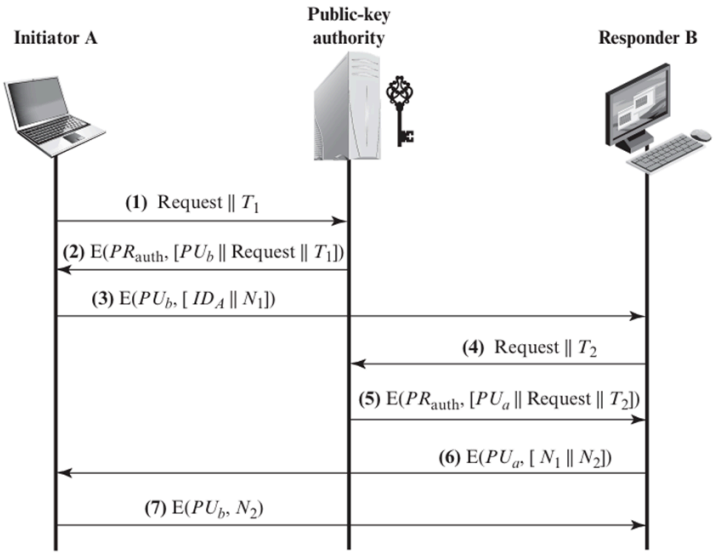


Figure 14.12 Public-Key Distribution Scenario

Public-Key Authority

Stronger security for public-key distribution can be achieved by providing tighter control over the distribution of public keys from the directory. A typical scenario is illustrated in Figure 14.12, which is based on a figure in [POPE79]. As before, the scenario assumes that a central authority maintains a dynamic directory of public keys of all participants. In addition, each participant reliably knows a public key for the authority, with only the authority knowing the corresponding private key. The following steps (matched by number to Figure 14.12) occur.

1. A sends a timestamped message to the public-key authority containing a request for the current public key of B.
2. The authority responds with a message that is encrypted using the authority's private key, PR_{auth}. Thus, A is able to decrypt the message using the authority's public key. Therefore, A is assured that the message originated with the authority. The message includes the following:
   - B's public key, PU_b, which A can use to encrypt messages destined for B
   - The original request used to enable A to match this response with the corresponding earlier request and to verify that the original request was not altered before reception by the authority
   - The original timestamp given so A can determine that this is not an old message from the authority containing a key other than B's current public key
3. A stores B's public key and also uses it to encrypt a message to B containing an identifier of A (ID_A) and a nonce (N_1), which is used to identify this transaction uniquely.

4, 5. B retrieves A's public key from the authority in the same manner as A retrieved B's public key.

At this point, public keys have been securely delivered to A and B, and they may begin their protected exchange. However, two additional steps are desirable:

6. B sends a message to A encrypted with PU_a and containing A's nonce (N_1) as well as a new nonce generated by B (N_2). Because only B could have decrypted message (3), the presence of N_1 in message (6) assures A that the correspondent is B.
7. A returns N_2, which is encrypted using B's public key, to assure B that its correspondent is A.

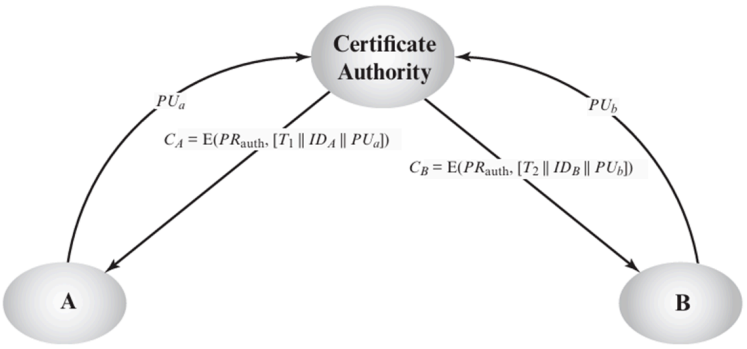
Public Key Certificates

•Suggested by Kohnfelder, to exchange keys without contacting a public-key authority.

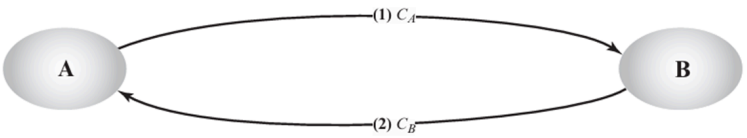
•A certificate consists of a public key, an identifier of the key owner, and the whole block signed by a trusted third party.

•Typically, the third party is a certificate authority, such as a government agency or a financial institution.

•A user can present his or her public key to the authority in a secure manner and obtain a certificate.



(a) Obtaining certificates from CA



(b) Exchanging certificates

Figure 14.13 Exchange of Public-Key Certificates

2 (a)	Describe briefly the Public Key Infrastructure (PKI) model with the help of a neat diagram. **SCHEME & SOLUTION** **Diagram - 2 Marks, Explanation - 3 Marks** **PUBLIC KEY INFRASTRUCTURE**	5	CO4	L2

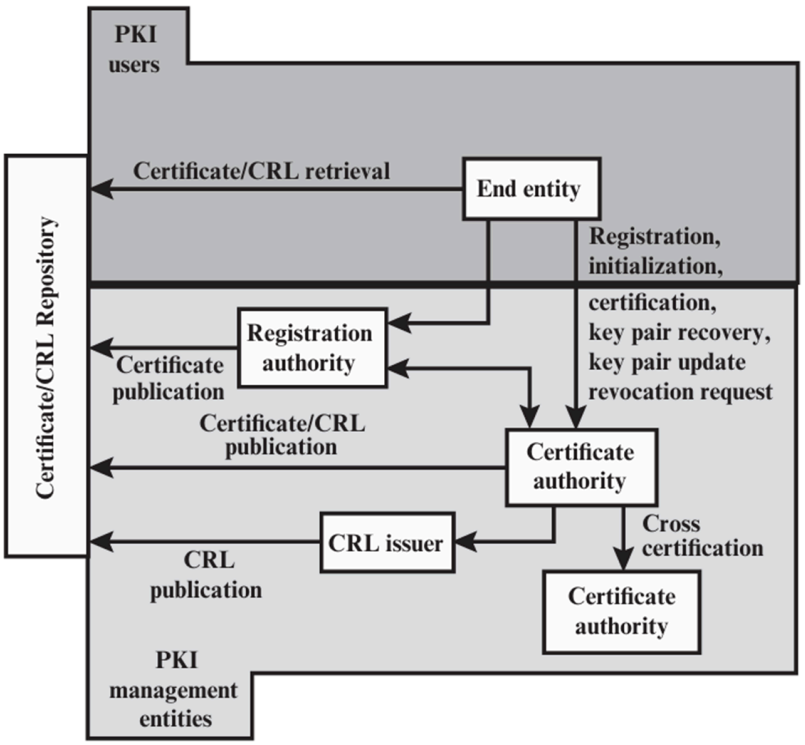
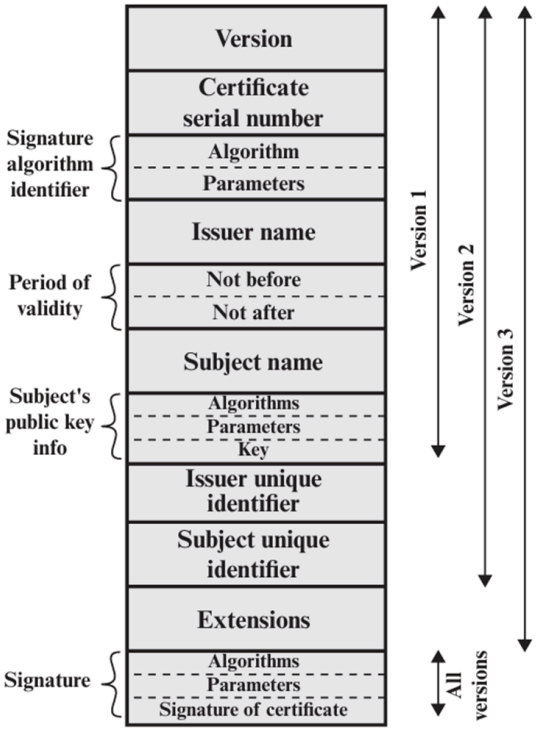


Figure 14.17 PKIX Architectural Model

	The principal objective for developing a PKI is to enable secure, convenient, and efficient acquisition of public keys. The Internet Engineering Task Force (IETF) Public Key Infrastructure X.509 (PKIX) set up the model based on X.509. •End entity: A generic term used to denote end users, devices. •Certification authority (CA): The issuer of certificates and certificate revocation lists (CRLs). •Registration authority (RA): Often associated with the end entity registration process but can assist in a number of other areas as well. •CRL issuer: An optional component that a CA can delegate to publish CRLs. •Repository: A generic term used to denote any method for storing certificates and CRLs so that they can be retrieved by end entities. •Registration: This is the process whereby a user first makes itself known to a CA. •Initialization: Before a client system can operate securely, it is necessary to install key materials. •Certification: CA issues a certificate for a user's public key, returns that certificate to the user's client system, and/or posts that certificate in a repository. •Key pair recovery: Key pair recovery allows end entities to restore their encryption/decryption key pair from an authorized key backup facility. •Key Pair Update: All key pairs need to be updated regularly and new certificates issued. •Revocation request: An authorized person advises a CA of an abnormal situation requiring certificate revocation. •Cross certification: Two CAs exchange information used in establishing Cross-certificate.			
2(c)	With the help of a neat diagram explain various fields in X.509 Certificate format. **SCHEME & SOLUTION** **Diagram - 2 Marks, Explanation - 3 Marks**  (a) X.509 certificate	5	CO4	L2

The standard uses the following notation to define a certificate:

$$CA \ll A \gg = CA \{V, SN, AI, CA, UCA, A, UA, Ap, T^A\}$$

where

$Y \ll X \gg$ = the certificate of user X issued by certification authority Y

$Y \{I\}$ = the signing of I by Y. It consists of I with an encrypted hash code appended

V = version of the certificate

SN = serial number of the certificate

AI = identifier of the algorithm used to sign the certificate

CA = name of certificate authority

UCA = optional unique identifier of the CA

A = name of user A

UA = optional unique identifier of the user A

Ap = public key of user A

T^A = period of validity of the certificate

- **Version:** Differentiates among successive versions of the certificate format; the default is version 1. If the *issuer unique identifier* or *subject unique identifier* are present, the value must be version 2. If one or more extensions are present, the version must be version 3. Although the X.509 specification is currently at version 7, no changes have been made to the fields that make up the certificate since version 3.
- **Serial number:** An integer value unique within the issuing CA that is unambiguously associated with this certificate.
- **Signature algorithm identifier:** The algorithm used to sign the certificate together with any associated parameters. Because this information is repeated in the signature field at the end of the certificate, this field has little, if any, utility.
- **Issuer name:** X.500 name of the CA that created and signed this certificate.
- **Period of validity:** Consists of two dates: the first and last on which the certificate is valid.
- **Subject name:** The name of the user to whom this certificate refers. That is, this certificate certifies the public key of the subject who holds the corresponding private key.
- **Subject's public-key information:** The public key of the subject, plus an identifier of the algorithm for which this key is to be used, together with any associated parameters.
- **Issuer unique identifier:** An optional-bit string field used to identify uniquely the issuing CA in the event the X.500 name has been reused for different entities.
- **Subject unique identifier:** An optional-bit string field used to identify uniquely the subject in the event the X.500 name has been reused for different entities.
- **Extensions:** A set of one or more extension fields. Extensions were added in version 3 and are discussed later in this section.
- **Signature:** Covers all of the other fields of the certificate. One component of this field is the digital signature applied to the other fields of the certificate. This field includes the signature algorithm identifier.

3 (a)	With the help of a neat diagram, explain the NIST Model for Electronic User Authentication **SCHEME & SOLUTION** **Diagram - 2 Marks, Explanation - 3 Marks**	5	CO3	L2

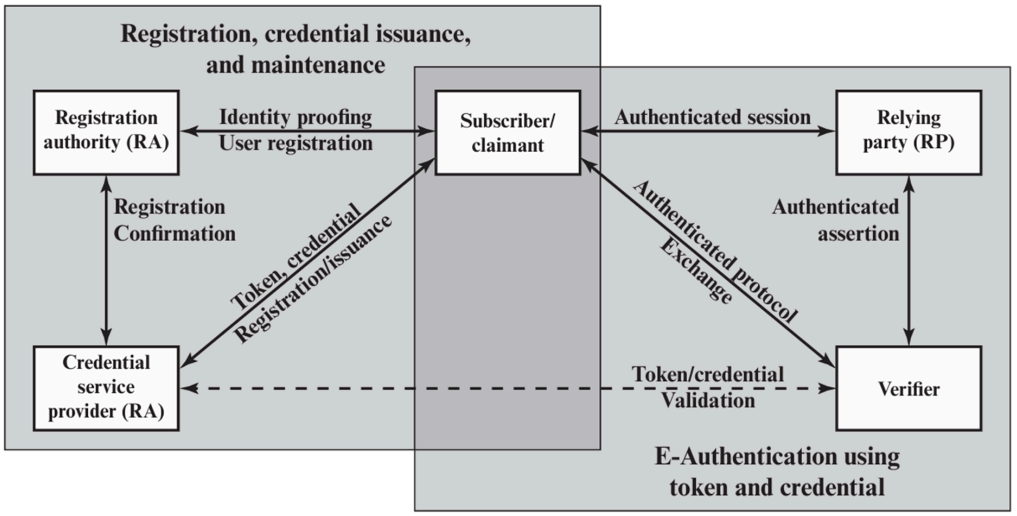


Figure 15.1 The NIST SP 800-63-2 E-Authentication Architectural Model

NIST Model for Electronic User Authentication

- **NIST SP 800-63-2** defines electronic user authentication.
- Process of establishing confidence in user identity that is provided electronically to the system.
- Authentication is used to determine if the user can be authorized to perform particular functions.
- Ex: Database transactions, system resources.

The initial requirement for performing user authentication is that the user must be registered with the system. The following is a typical sequence for registration. An applicant applies to a **registration authority (RA)** to become a **subscriber**

of a **credential service provider (CSP)**. In this model, the RA is a trusted entity that establishes and vouches for the identity of an applicant to a CSP. The CSP then engages in an exchange with the subscriber. Depending on the details of the overall authentication system, the CSP issues some sort of electronic credential to the subscriber. The **credential** is a data structure that authoritatively binds an identity and additional attributes to a token possessed by a subscriber, and can be verified when presented to the verifier in an authentication transaction. The token could be an encryption key or an encrypted password that identifies the subscriber. The token may be issued by the CSP, generated directly by the subscriber, or provided by a third party. The token and credential may be used in subsequent authentication events.

Once a user is registered as a subscriber, the actual authentication process can take place between the subscriber and one or more systems that perform authentication and, subsequently, authorization. The party to be authenticated is called a **claimant** and the party verifying that identity is called a **verifier**. When a claimant successfully demonstrates possession and control of a token to a verifier through an authentication protocol, the verifier can verify that the claimant is the subscriber named in the corresponding credential. The verifier passes on an assertion about the identity of the subscriber to the **relying party (RP)**. That assertion includes identity information about a subscriber, such as the subscriber name, an identifier assigned at registration, or other subscriber attributes that were verified in the registration process. The RP can use the authenticated information provided by the verifier to make access control or authorization decisions.

An implemented system for authentication will differ from or be more complex than this simplified model, but the model illustrates the key roles and functions needed for a secure authentication system.

3 (b)	Differentiate Kerberos Version 4 and Version 5. **SCHEME & SOLUTION** **Differences - 5 Marks**	5	CO4	L3

Kerberos Version 5

- Kerberos Version 5 is specified in RFC 4120.
- Provides a number of improvements over Version 4.

Version 5 is intended to address the limitations of version 4 in two aspects:

- **Environment Shortcomings**
- **Technical Deficiencies**

Environmental Shortcomings

Encryption System Dependence:

- **Version 4** requires the use of DES. The strength of the DES is a concern. (Ticket, Message, Key)
- **Version 5** was redesigned to be encryption independent. It works with any symmetric encryption.
- Encryption keys are tagged with a type and a length, allowing the same key to be used in different algorithms.

Internet Protocol Dependence:

- **Version 4** requires the use of Internet Protocol (IPV4).
- **Version 5** was redesigned to be accommodate IPV6 and OSI addresses.
- Network addresses are tagged with type and length allowing any network address type to be used.

Message byte Ordering:

- **Version 4** sender choses a byte ordering of his/her own, which is ambiguous.
- **Version 5** all the message structures are defined using ASN.1 (Abstract Syntax Notation One) and Basic Encoding Rules (BER).

Ticket Life time:

- **Version 4** encoded in an 8-bit quantity in units of 5 minutes. Maximum life time is expressed as

$$2^8 \times 5 = 1280 \text{ minutes}$$

- **Version 5** includes an explicit start time and end time, allowing tickets with arbitrary lifetimes.

Authentication Forwarding:

- **Version 4** tickets are tied to the client's machine. It cannot be forwarded to another server.
- **Version 5** allows forwardable tickets. The client can allow the server to temporarily act on their behalf.

Inter-realm Authentication:

- **Version 4** interoperability among N realms require N^2 Kerberos-to-Kerberos relationship.
- **Version 5** requires a fewer relationships.

Technical Deficiencies

Double Encryption:

- **Version 4** tickets provided are encrypted twice- once with secret key of Server and again with the secret key known to the client.
- It is computationally expensive.

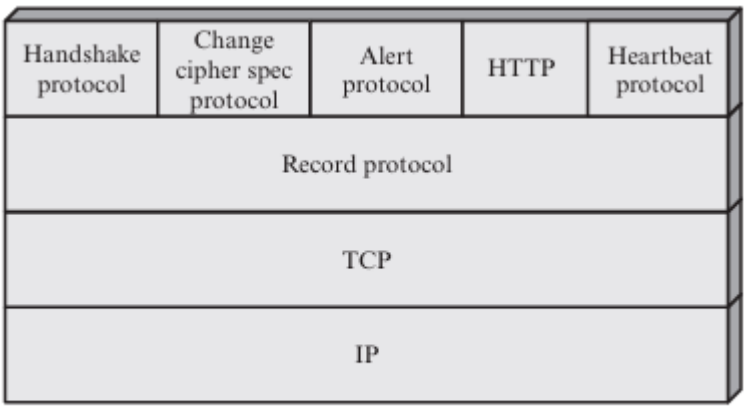
PCBC Encryption:

- **Version 4** makes use of non-standard mode of DES known as Propagating cipher block chaining (PCBC). It is vulnerable to attacks.
- **Version 5** provides cipher block chaining (CBC) mode for encryption.

Session Keys:

- **Version 4** singe session key is used for all types of communication.
- **Version 5** provides a sub session key, which is used only for that one connection. A new access by the client would result in the use of a new sub session key.

	Password attacks: • **Both versions are vulnerable to a password attack.** • **Version 5** provides a mechanism called pre-authentication, making the password attacks difficult, but it does not prevent them.			

4	With the help of suitable diagrams, explain the architecture and phases of TLS (Transport Layer Security). **SCHEME & SOLUTION** **Diagram - 2 Marks, Phases - 8 Marks**  Figure 17.2 TLS Protocol Stack Two important TLS concepts: • TLS Session : Association between a client and a server. Sessions are created by the Handshake Protocol. • Define cryptographic security parameters, which can be shared among multiple connections. • TLS Connection: A connection is a transport that provides a suitable type of service. Every connection is associated with one session. • Master secret: 48-byte secret shared between the client and server. • Is resumable: A flag indicating whether the session can be used to initiate new connections	10	CO4	L2

Connection state is defined by the following parameters:

- Server and client random: Byte sequences that are chosen by the server and client for each connection.
- Server write MAC secret: The secret key used in MAC operations on data sent by the server.
- Client write MAC secret: The symmetric key used in MAC operations on data sent by the client.

A session state is defined by the following parameters:

- Session identifier: An arbitrary byte sequence chosen by the server to identify an active or resumable session state.
- Peer certificate: An X509.v3 certificate of the peer.
- Compression method: The algorithm used to compress data prior to encryption.
- Cipher spec: Specifies the bulk data encryption algorithm and a hash algorithm. It also defines cryptographic attributes such as the hash_size.

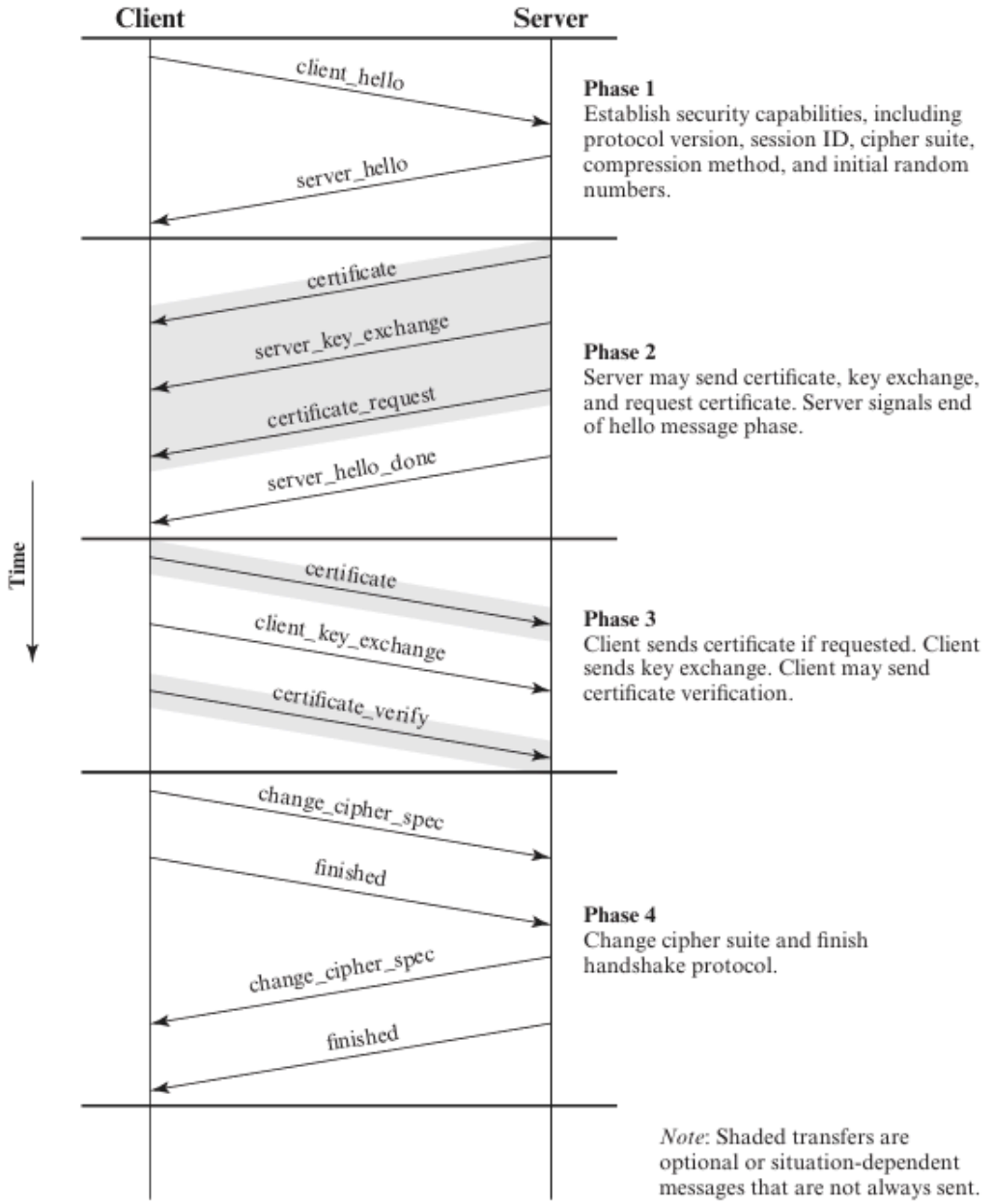


Figure 17.6 Handshake Protocol Action

Phase 1:

- The first element of the Ciphersuite parameter is the key exchange method.

Following key methods are supported.

- **RSA**
- **Fixed Diffie Hellman**
- **Ephemeral Diffie Hellman**
- **Anonymous Diffie Hellman**

- **CipherAlgorithm:** Any of the algorithms mentioned earlier: RC4, RC2, DES, 3DES, DES40, or IDEA
- **MACAlgorithm:** MD5 or SHA-1
- **CipherType:** Stream or Block
- **IsExportable:** True or False
- **HashSize:** 0, 16 (for MD5), or 20 (for SHA-1) bytes
- **Key Material:** A sequence of bytes that contain data used in generating the write keys
- **IV Size:** The size of the Initialization Value for Cipher Block Chaining (CBC) encryption

Phase 2:

- Sever begins this phase by sending its certificate for authentication.
- Message contains one or more X.509 certificates.
- The certificate message is required for any key exchange except anonymous Diffie-Hellman.
- Server_Key_Exchange message may be sent if required. IT needs:
- Anonymous Diffie Hellman
- Ephemeral Diffie Hellman
- RSA Key Exchange

- Signature is created by taking the hash of the message and encrypting with sender's private key.

- Hash value has Diffie-Hellman or RSA parameters and also two nonces for initial hello messages.
- This ensures against replay attacks and misrepresentation.

- If server is not using anonymous Diffie-Hellman can request certificate from client.

The certificate_request message includes two parameters:

- Certificate_type (RSA, DSS,
- Certificate_authorities

- Final message in phase2 is server_done message sent by server to indicate the end of the server hello and associated messages.

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.
- 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.

```
CertificateVerify.signature.md5_hash
    MD5(handshake_messages);
Certificate.signature.sha_hash
    SHA(handshake_messages);
```

Phase 4:

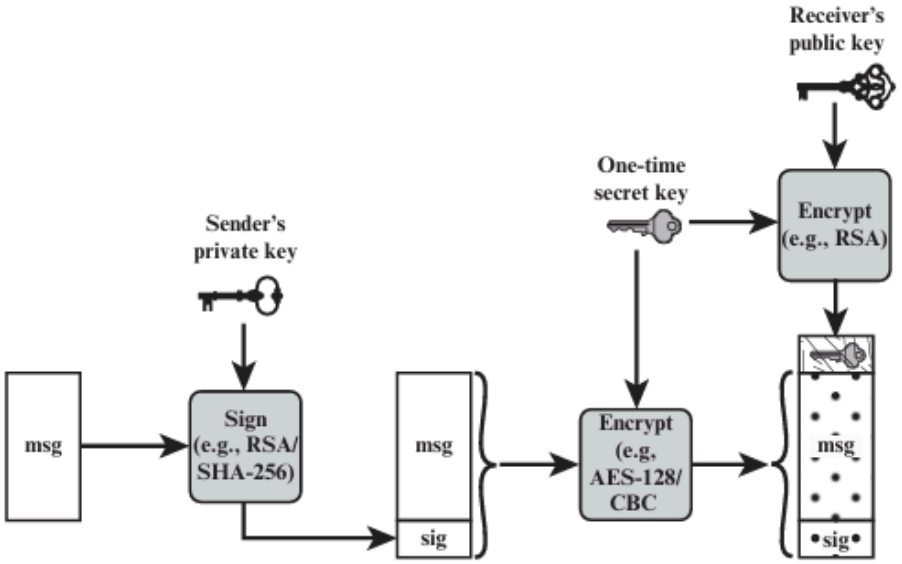
Finish

- Phase 4 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.
- 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.

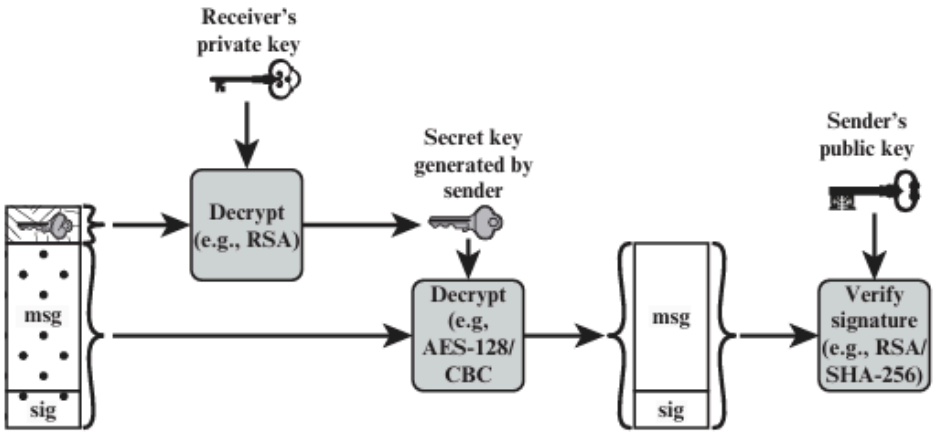
- The content of finished message is:

- Finished_label is the string “client finished” for the client and “server finished” for the server.
- At this point, the handshake is complete and the client and server may begin to exchange application-layer data.

5 (a)	Write a note on the following: 1. S/MIME Functionality 2. Types of S/MIME Messages **SCHEME & SOLUTION** **Functionality - 2 Marks, Types - 3 Marks**	5	CO5	L2



(a) Sender signs, then encrypts message



(b) Receiver decrypts message, then verifies sender's signature

Figure 19.5 Simplified S/MIME Functional Flow

S/MIME Message Content Types

S/MIME uses the following message content types, which are defined in RFC 5652, Cryptographic Message Syntax:

- **Data:** Refers to the inner MIME-encoded message content, which may then be encapsulated in a SignedData, EnvelopedData, or CompressedData content type.
- **SignedData:** Used to apply a digital signature to a message.
- **EnvelopedData:** This consists of encrypted content of any type and encrypted-content encryption keys for one or more recipients.
- **CompressedData:** Used to apply data compression to a message.

5 (b)	With the help of a relevant diagram, explain the functional flow of DKIM . **SCHEME & SOLUTION** **Diagram - 2 Marks, Explanation - 3 Marks**	5	CO5	L2

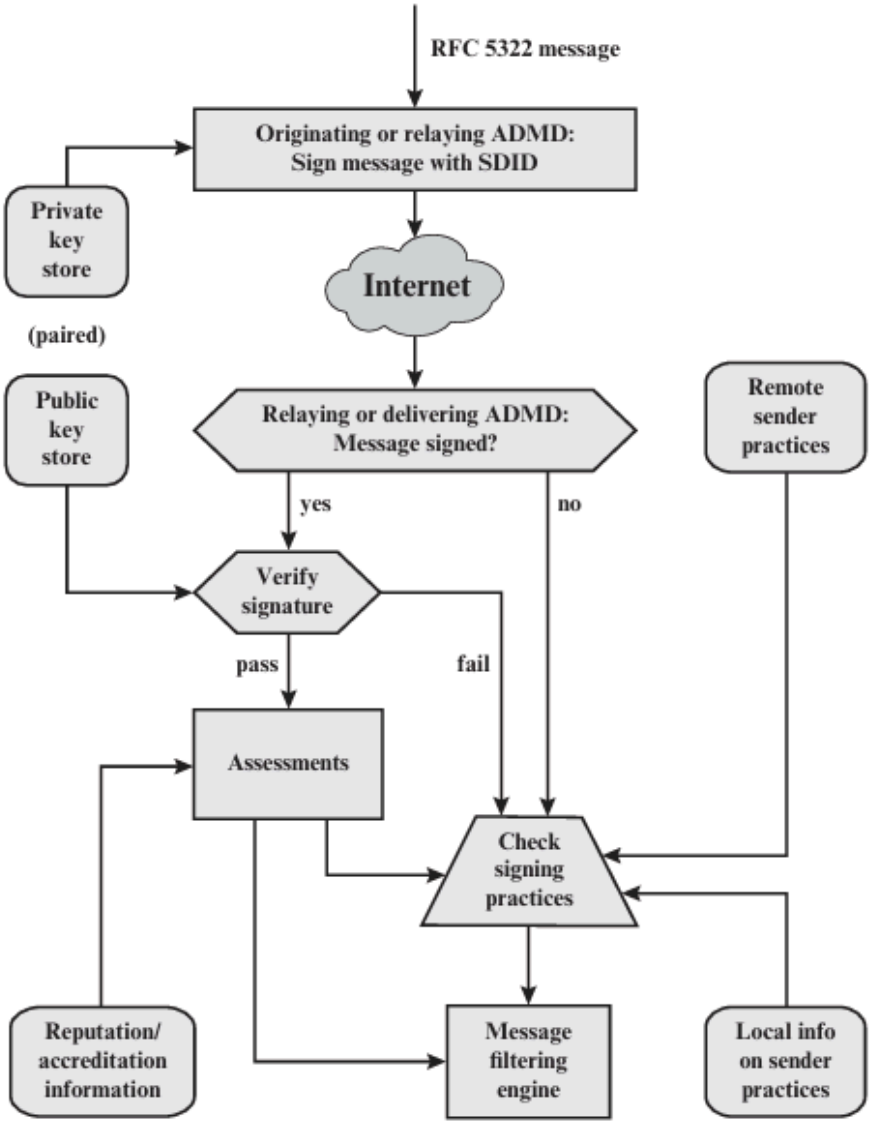
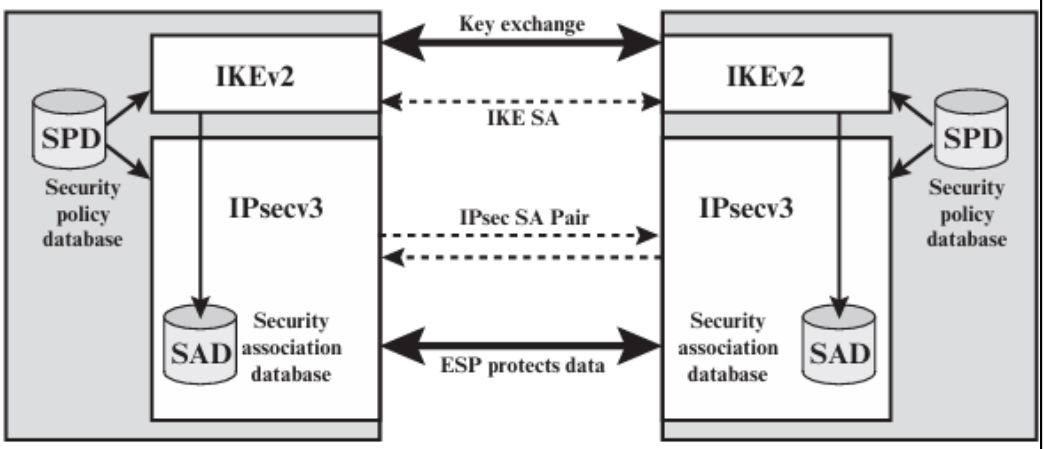


Figure 19.11 DKIM Functional Flow

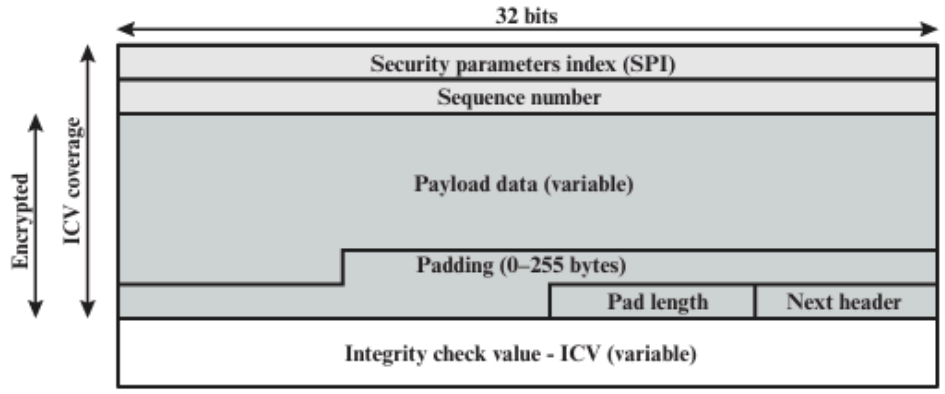
DKIM Functional Flow

Figure 19.11 provides a more detailed look at the elements of DKIM operation. Basic message processing is divided between a signing Administrative Management Domain (ADMD) and a verifying ADMD. At its simplest, this is between the originating ADMD and the delivering ADMD, but it can involve other ADMDs in the handling path.

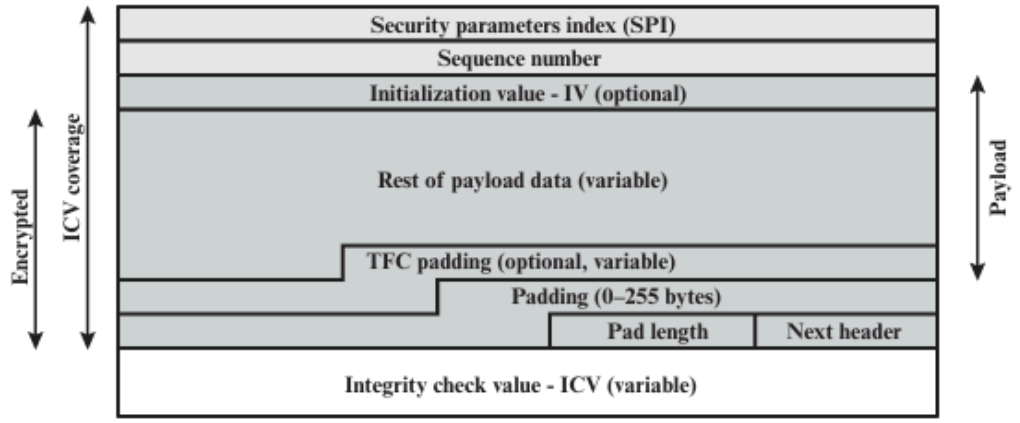
Signing is performed by an authorized module within the signing ADMD and uses private information from a Key Store. Within the originating ADMD,

	this might be performed by the MUA, MSA, or an MTA. Verifying is performed by an authorized module within the verifying ADMD. Within a delivering ADMD, verifying might be performed by an MTA, MDA or MUA. The module verifies the signature or determines whether a particular signature was required. Verifying the signature uses public information from the Key Store. If the signature passes, reputation information is used to assess the signer and that information is passed to the message filtering system. If the signature fails or there is no signature using the author's domain, information about signing practices related to the author can be retrieved remotely and/or locally, and that information is passed to the message filtering system. For example, if the sender (e.g., gmail) uses DKIM but no DKIM signature is present, then the message may be considered fraudulent.			
6(a)	Depict and explain IPSec architecture and explain the parameters for security association **SCHEME & SOLUTION** **Diagram - 2 Marks, Explanation - 4 Marks**  Figure 20.2 IPsec Architecture	6	CO5	L2

	In each IPsec implementation, there is a nominal[2] Security Association Database that defines the parameters associated with each SA. A security association is normally defined by the following parameters in an SAD entry. ■ **Security Parameter Index:** A 32-bit value selected by the receiving end of an SA to uniquely identify the SA. In an SAD entry for an outbound SA, the SPI is used to construct the packet's AH or ESP header. In an SAD entry for an inbound SA, the SPI is used to map traffic to the appropriate SA. ■ **Sequence Number Counter:** A 32-bit value used to generate the Sequence Number field in AH or ESP headers, described in Section 20.3 (required for all implementations). ■ **Sequence Counter Overflow:** A flag indicating whether overflow of the Sequence Number Counter should generate an auditable event and prevent further transmission of packets on this SA (required for all implementations). ■ **Anti-Replay Window:** Used to determine whether an inbound AH or ESP packet is a replay, described in Section 20.3 (required for all implementations). ■ **AH Information:** Authentication algorithm, keys, key lifetimes, and related parameters being used with AH (required for AH implementations). ■ **ESP Information:** Encryption and authentication algorithm, keys, initialization values, key lifetimes, and related parameters being used with ESP (required for ESP implementations). ■ **Lifetime of this Security Association:** A time interval or byte count after which an SA must be replaced with a new SA (and new SPI) or terminated, plus an indication of which of these actions should occur (required for all implementations). ■ **IPsec Protocol Mode:** Tunnel, transport, or wildcard. ■ **Path MTU:** Any observed path maximum transmission unit (maximum size of a packet that can be transmitted without fragmentation) and aging variables (required for all implementations).			
6 (b)	With the help of a neat diagram explain the ESP packet format. **SCHEME & SOLUTION** **Diagram - 2 Marks, Explanation - 2 Marks**	4	CO5	L2



(a) Top-level format of an ESP Packet



(b) Substructure of payload data

Figure 20.5 ESP Packet Format

- **Security Parameters Index (32 bits):** Identifies a security association.
- **Sequence Number (32 bits):** A monotonically increasing counter value; this provides an anti-replay function, as discussed for AH.
- **Payload Data (variable):** This is a transport-level segment (transport mode) or IP packet (tunnel mode) that is protected by encryption.
- **Padding (0–255 bytes):** The purpose of this field is discussed later.
- **Pad Length (8 bits):** Indicates the number of pad bytes immediately preceding this field.
- **Next Header (8 bits):** Identifies the type of data contained in the payload data field by identifying the first header in that payload (e.g., an extension header in IPv6, or an upper-layer protocol such as TCP).
- **Integrity Check Value (variable):** A variable-length field (must be an integral number of 32-bit words) that contains the Integrity Check Value computed over the ESP packet minus the Authentication Data field.