Web Security

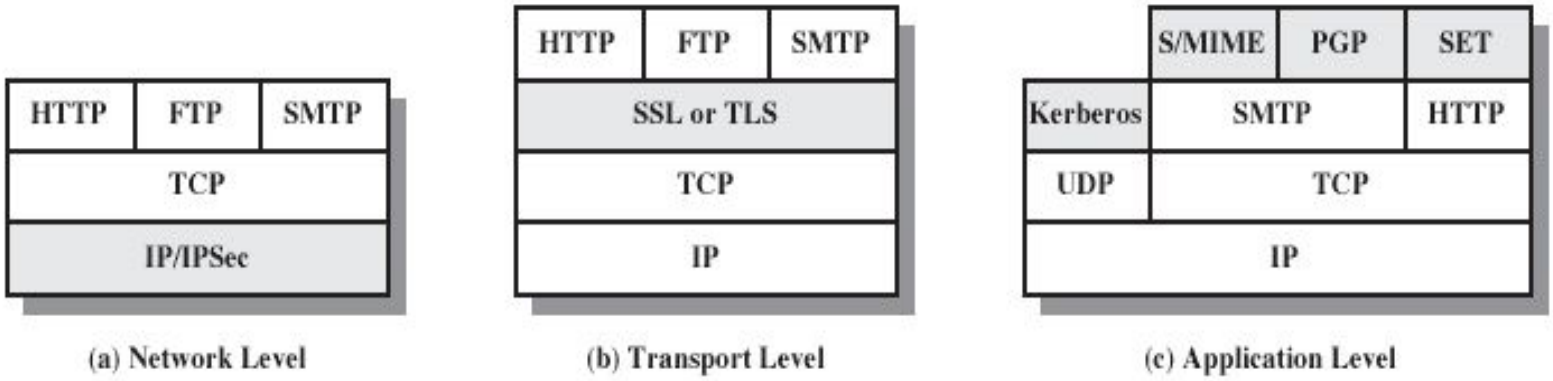
Web Security

- Web now widely used by business, government, individuals
- but Internet & Web are vulnerable
- have a variety of threats
  - integrity
  - confidentiality
  - denial of service
  - authentication
- need added security mechanisms

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
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



IPSec: The advantage of using IPSec is that it is transparent to end users and applications and provides a general-purpose solution.

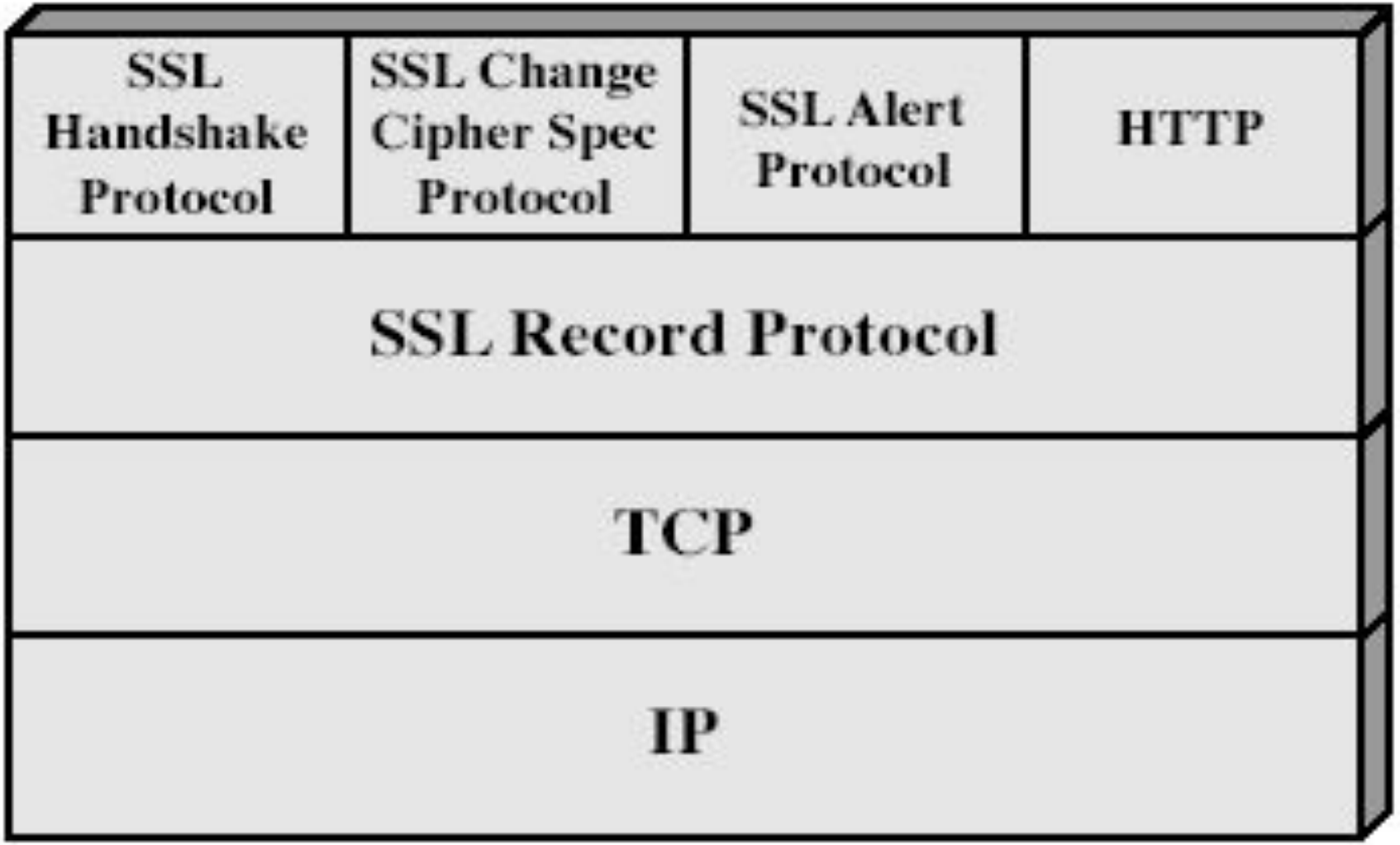
Transport Level: SSL or TLS could be provided as part of the underlying protocol suite and therefore be transparent to applications. Alternatively, SSL can be embedded in specific packages. (e.g. Netscape and Microsoft Explorer browsers come equipped with SSL, and most Web servers have implemented the protocol.)

Application-specific security: Services are embedded within the particular application. The advantage of this approach is that the service can be tailored to the specific needs of a given application. (e.g. SET).

SSL (Secure Socket Layer)

- **transport layer security** service
- originally developed by Netscape
- version3 designed with public review & input
- subsequently became Internet standard known as TLS (Transport Layer Security)
- uses TCP to provide a reliable end-to-end service
- SSL has two layers of protocols

SSL Architecture



SSL Architecture

- **SSL session**
  - an association between client & server
  - created by the Handshake Protocol
  - define a set of cryptographic parameters
  - may be shared by multiple SSL connections
- **SSL connection**
  - a transient, peer-to-peer, communications link
  - associated with one SSL session

SSL Architecture

A session state is defined by the following parameters (definitions from SSL specification):

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. This element of the state may be null.

Compression method: The algorithm used to compress data prior to encryption.

Cipher spec: Specifies the bulk data encryption algorithm (such as null, DES etc.) and a hash algorithm (such as MD5 or SHA-l) used for MAC calculation. IIt also defines cryptographic attributes such as the hash size.

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.

SSL Architecture

A 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 secret key used in MAC operations on data sent by the client.

Server write key: The conventional encryption key for data encrypted by the server and decrypted by the client.

Client write key: The conventional encryption key for data encrypted by the client and decrypted by the server.

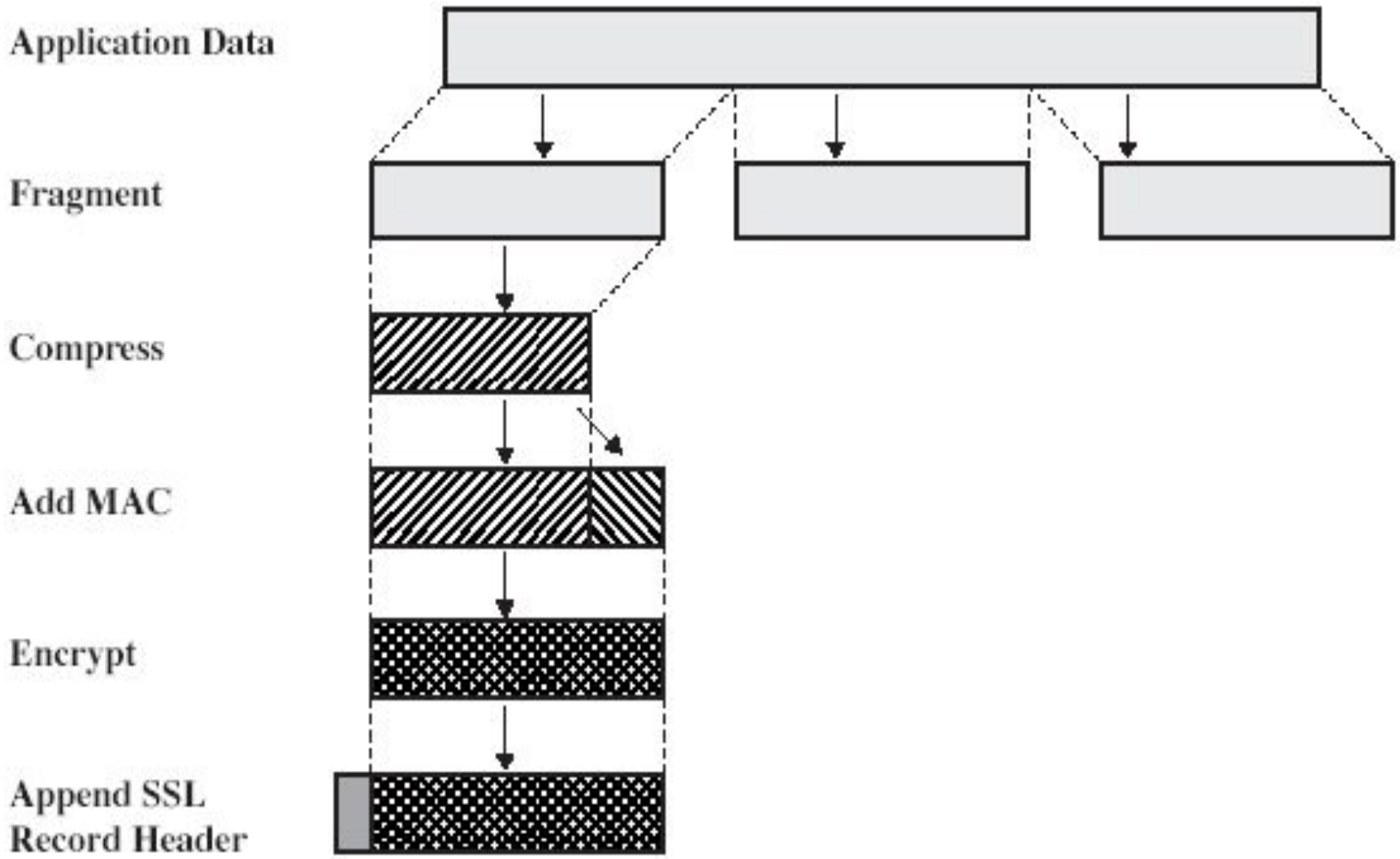
Initialization vectors: When a block cipher in CBC mode is used, an initialization vector (IV) is maintained for each key- initialized by the SSL Handshake Protocol.

Sequence numbers: Each party maintains separate sequence numbers for transmitted and received messages for each connection.

SSL Record Protocol

- **confidentiality**
  - using symmetric encryption with a shared secret key defined by Handshake Protocol
  - IDEA, RC2-40, DES-40, DES, 3DES, Fortezza, RC4-40, RC4-128
  - message is compressed before encryption
- **message integrity**
  - using a MAC with shared secret key
  - similar to HMAC but with different padding

SSL Record Protocol Steps



SSL Record Protocol Operation

SSL Record Protocol Steps

Fragmentation: Each upper-layer message is fragmented into blocks of 2^{14} bytes (16384 bytes) or less.
Compression: compression is optionally applied. Compression must be lossless.
Compute MAC: a shared secret key is used. The calculation is defined as:

hash(MAC_write_secret II pad_2 II hash(MAC_write_secret II pad_1 II seq_num II SSLCompressed.type II SSLCompressed.length II SSLCompressed.fragment))

where

II = concatenation

MAC- write_secret = shared secret key

Hash = cryptographic hash algorithm; either MD5 or SHA1

pad_1 = the byte 006 (0011 0110) repeated 48 times (384bits) for MD5 and 40 times (320 bits) for SHA-l

pad_2 = the byte Ox5C (0101 1100) repeated 48 times for MD5 and 40 times for SHA-1

Seq_num = the sequence number for this message

SSLCompressed.type = the higher-level protocol used to process this fragment

SSLCompressed.length = the length of the compressed fragment

SSLCompressed.fragment = the compressed fragment (if compression is not used, the plaintext fragment)

SSL Record Protocol Steps

Encryption: Compressed message plus MAC is encrypted by symmetric algorithm. Algorithms used are IDEA, RC2-40, DES-40, DES, 3DES, Fortezza. Fortezza can be used in a smart card encryption scheme.

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

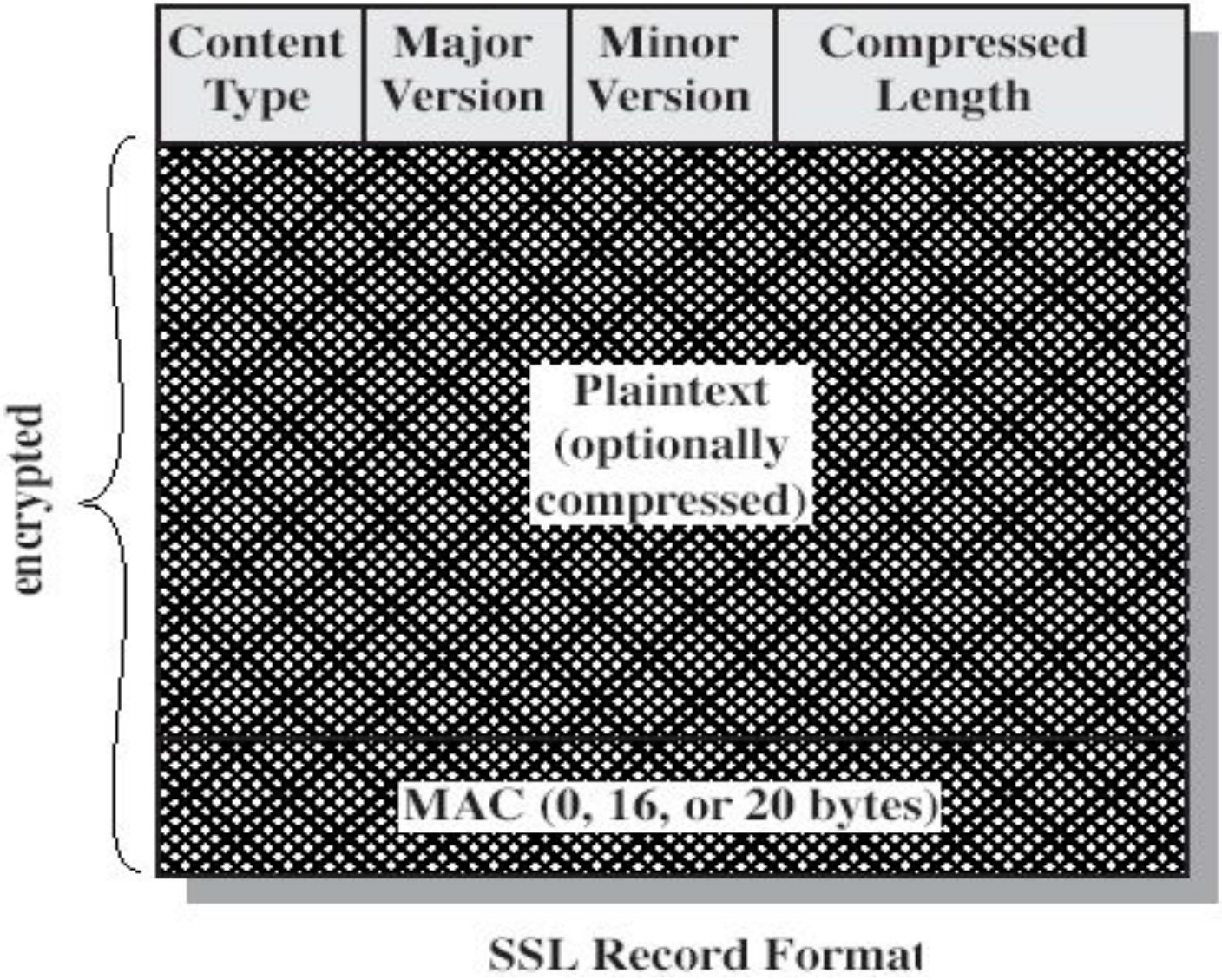
Content Type (8 bits): The higher layer protocol used to process the enclosed fragment.

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 O.

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.

SSL Record Format

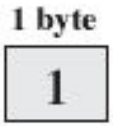


SSL Record Format

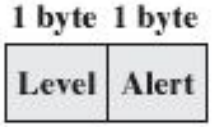
SSL Change Cipher Spec Protocol

- One of three SSL specific protocols which use the SSL Record protocol
- Consists 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.

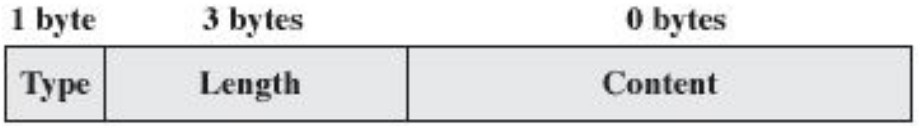
SSL Record Protocol Payload



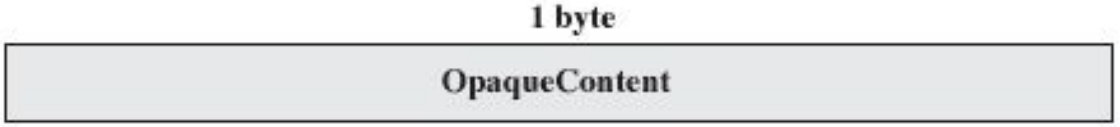
(a) Change Cipher Spec Protocol



(b) Alert Protocol



(c) Handshake Protocol



(d) Other Upper-Layer Protocol (e.g., HTTP)

SSL Record Protocol Payload

SSL Alert Protocol

- conveys SSL-related alerts to peer entity
- severity
    - warning or fatal
    - If the level is fatal, SSL immediately terminates the connection.
- specific alert
  - unexpected_message: An inappropriate message was received.
  - bad_record_mac: An incorrect MAC was received.
  - decompression_failure: The decompression function received improper input (e.g., unable to decompress or decompress to greater than maximum allowable length).
  - handshake_failure: Sender was unable to negotiate an acceptable set of security parameters given the options available.
  - Illegal_parameter: A field in a handshake message was out of range or inconsistent with other fields.

- compressed & encrypted like all SSL data

SSL Alert Protocol

The remainder of the alerts are the following

- **close_notify:** Notifies the recipient that the sender will not send any more messages on this connection.
- **no_certificate:** May be sent in response to a certificate request if no appropriate certificate is available.
- **bad_certificate**: A received certificate was corrupt (e.g., contained a signature that did not verify).
- **unsupported_certificate:** The type of the received certificate is not supported.
- **certificate_revoked:** A certificate has been revoked by its signer.
- **certificate_expired:** A certificate has expired.
- **certificate_unknown:** Some other unspecified issue arose in processing the certificate, rendering it unacceptable.

SSL Handshake Protocol

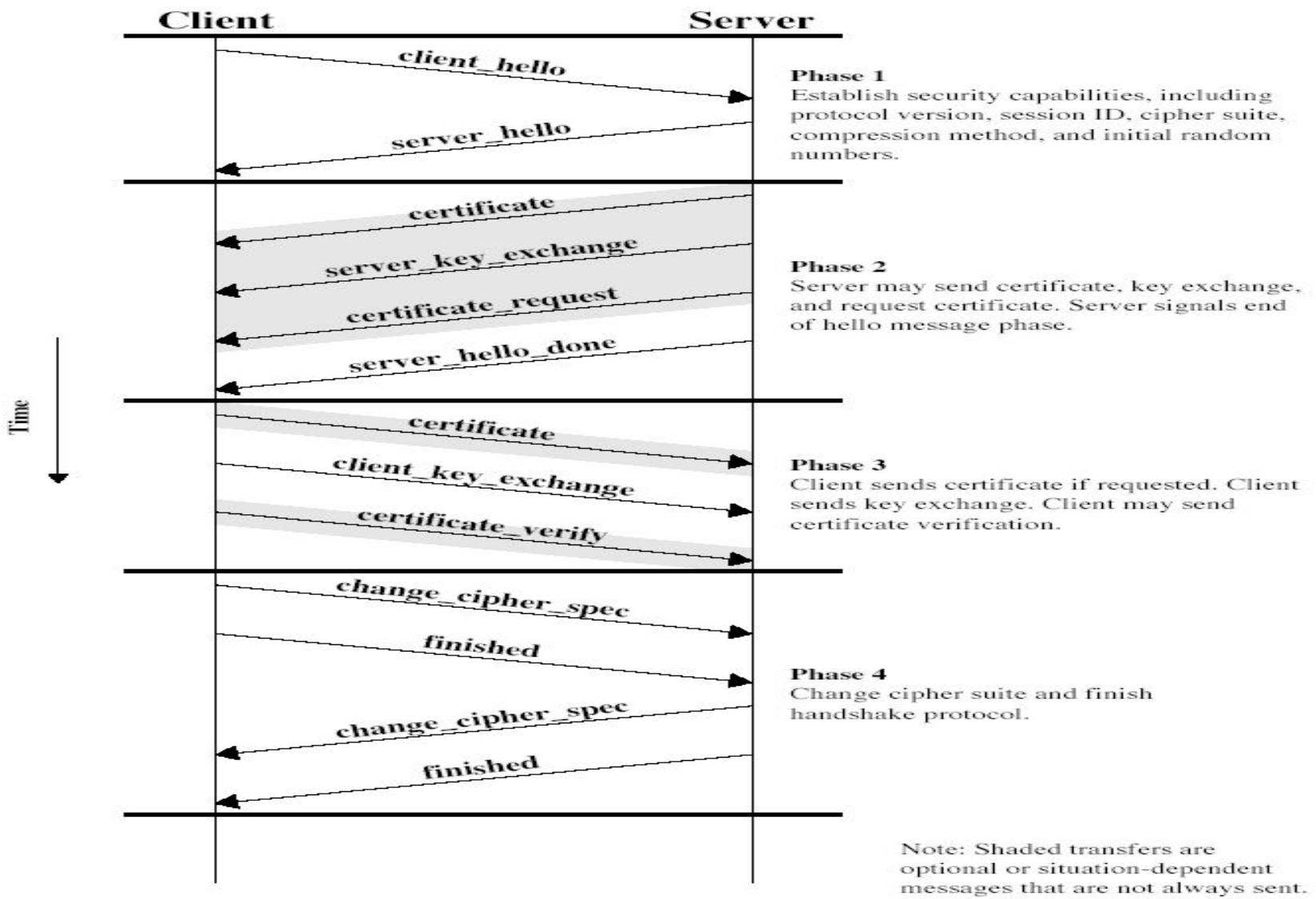
- allows server & client to:
  - authenticate each other
  - to negotiate encryption & MAC algorithms
  - to negotiate cryptographic keys to be used
- comprises a series of messages in phases
  - Phase 1: Establish Security Capabilities
  - Phase 2: Server Authentication and Key Exchange
  - Phase 3: Client Authentication and Key Exchange
  - Phase 4: Finish

SSL Handshake Protocol Message Types

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

SSL Handshake Protocol



Phase 1. Establish Security Capabilities

- 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_hellow 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
  - **Session ID:** A variable-length session identifier. A nonzero value indicates that the client wishes to update the parameters of an existing connection or create a new connection on this session. A zero value indicates that the client wishes to establish a new connection on a new session.
  - **CipherSuite:** This is a list that contains the combinations of cryptographic algorithms supported by--the client, in decreasing order of preference.
  - **Compression Method:** This is a list of the compression methods the client supports.

Phase 1. Establish Security Capabilities

- After sending the client_hellow message, the client waits for the server_hello message, which contains the same parameters as the client_hellow message.
- The first element of the Cipher Suite parameter is the key exchange method The following key exchange methods are supported:
  - **RSA:** The secret key is encrypted with the receiver's RSA public key. A public-key certificate for the receiver's key must be made available.
  - **Fixed Diffie-Hellman:** This is a Diffie-Hellman key exchange in which the server's certificate contains the Diffie-Hellman public parameters signed by the certificate authority (CA).
  - **Ephemeral Diffie-Hellman:** This technique is used to create ephemeral (temporary, one-time) secret keys.
  - **Anonymous Diffie-Hellman:** The base Diffie-Hellman algorithm is used, with no authentication- vulnerable to man-in-middle-attack
  - **Fortezza:** The client's Fortezza parameters are sent.

Phase 2. Server Authentication and Key Exchange

- The server begins this phase by sending its certificate, if it needs to be authenticated; the message contains one or a chain of X.509 certificates.
- It is not required in two instances:

  (1) The server has sent a certificate with fixed Diffie-Hellman parameters, or

  (2) RSA key exchange is to be used. The server_keyexchange message is needed for the following:
- The certificate_request message includes two parameters: certificate_type and certificate_authorities. The certificate type indicates the public-key algorithm and its use:.
  - RSA, signature only
  - DSS, signature only
  - RSA for fixed Diffie-Hellman;
  - DSS for fixed Diffie-Hellman
  - RSA for ephemeral Diffie-Hellman.
  - DSS for ephemeral Diffie-Hellman.
  - Fortezza

Phase 3. Client Authentication and Key Exchange

- Server provided a valid certificate and check that the receipt of the server_done message, the client should verify that the server_hello parameters are acceptable.
- The client generates a 48-byte *pre-master secret* and encrypts with the public key from the server's certificate or temporary RSA key from a server_key _exchange message.
- Its used to compute a master secret .

Phase 4. Finish

- 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.
- 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.

TLS (Transport Layer Security)

- IETF standard RFC 2246 similar to SSLv3
- with minor differences
  - in record format version number
  - uses HMAC for MAC
  - a pseudo-random function expands secrets
  - has additional alert codes
  - some changes in supported ciphers
  - changes in certificate negotiations
  - changes in use of padding

Secure Electronic Transactions (SET)

- open encryption & security specification
- to protect Internet credit card transactions
- developed in 1996 by Mastercard, Visa etc
- not a payment system
- rather a set of security protocols & formats
  - secure communications amongst parties
  - trust from use of X.509v3 certificates
  - privacy by restricted info to those who need it

SET Requirements

- Provide confidentiality of payment and ordering information
- Ensure the integrity of all transmitted data
- Provide authentication that a cardholder is a legitimate user of a credit card account
- Provide authentication that a merchant can accept credit card transactions through its relationship with a financial institution.
- Ensure the use of the best security practices and system design techniques to protect all legitimate parties in an electronic commerce transaction
- Create a protocol that neither depends on transport security mechanisms nor prevents their use
- Facilitate and encourage interoperability among software and network providers.

Key Features of SET

❑ Confidentiality of information:

✔ Cardholder account and payment information is secured as it travels across the network

✔. An interesting and important feature of SET is that it prevents the merchant from learning the cardholder's credit card number; this is only provided to the issuing bank.

❑ Integrity of data:

✔Payment information sent from cardholders to merchants includes order information, personal data, and payment instructions.

✔ SET guarantees that these message contents are not altered in transit.

✔RSA digital signatures, using SHA-1 hash codes, provide message integrity.

❑ Cardholder account authentication:

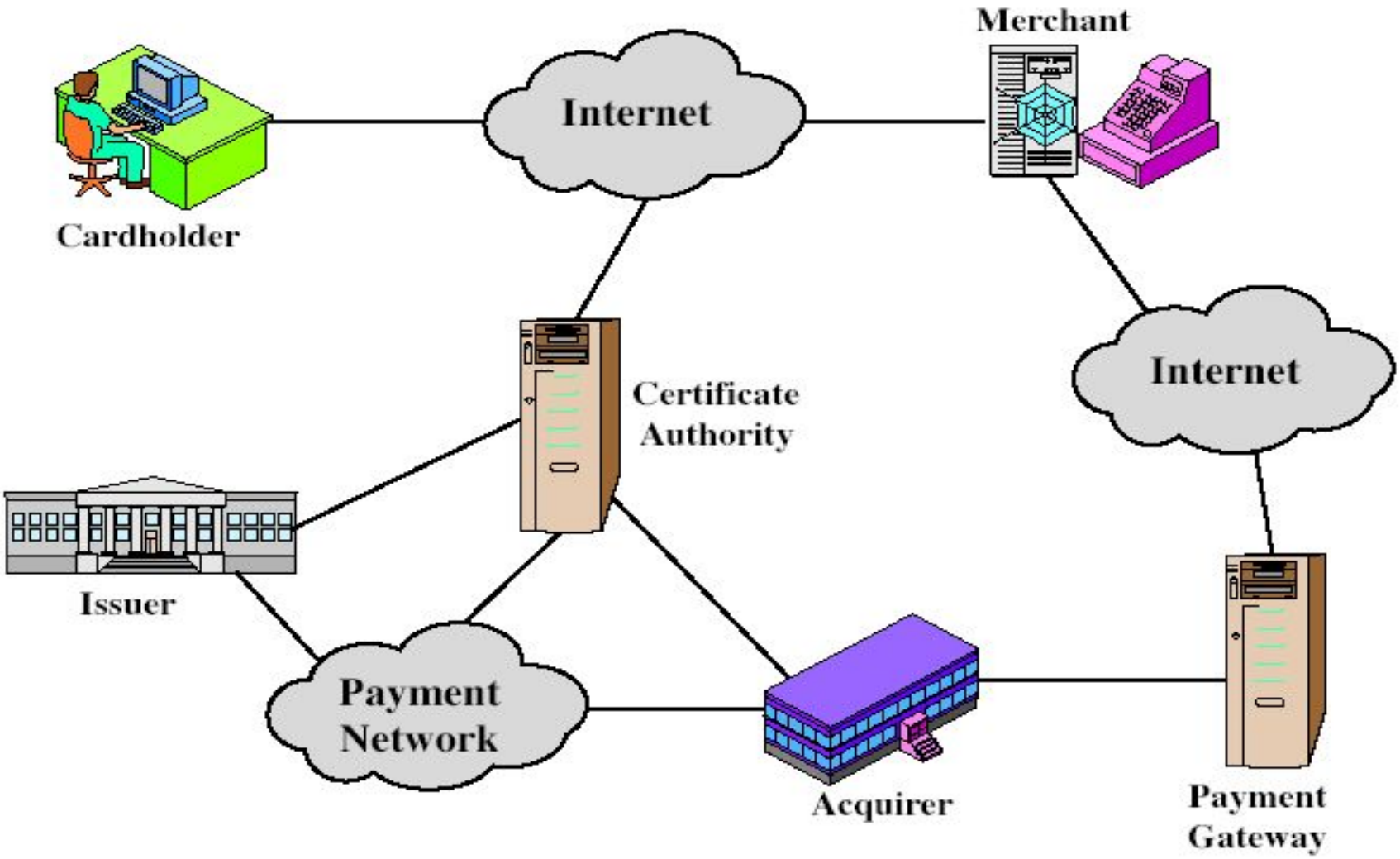
✔ SET enables merchants to verify that a cardholder is a legitimate user of a valid card account number

✔ SET usesx509.v3 digital certificates with RSA signatures for this purpose

❑ Merchant authentication:

✔ SET enables cardholders to verify that a merchant has relationship with a financial institution allowing it to accept payment cards.

SET Components



SET Parcipants

- **Cardholder:** In the electronic environment, consumers and corporate purchasers interact with merchants from personal computers over the Internet. A cardholder is an authorized holder of a payment card (e.g., MasterCard, Visa) that has been issued by an issuer.
- **Merchant:** A merchant is a person or organization that has goods or services to sell to the cardholder.
- **Issuer:** This is a financial institution, such as a bank, that provides the cardholder with the payment card.
- **Acquirer:**
  - ✔ financial institution that establishes an account with a merchant and processes payment card authorizations and payments.
  - ✔provides authorization to the merchant that a given card account is active and that the proposed purchase does not exceed the credit limit.
  - ✔acquirer also provides electronic transfer of payments to the merchant's account.
- **Payment gateway:**
  - ✔ a function operated by the acquirer or a designated third party that processes merchant payment messages.
  - ✔ payment gateway interfaces between SET and the existing bankcard payment networks for authorization and payment functions.
- **Certification authority (CA):** an entity that is trusted to issue X.509v3 public-key certificates for cardholders, merchants, and payment gateways.

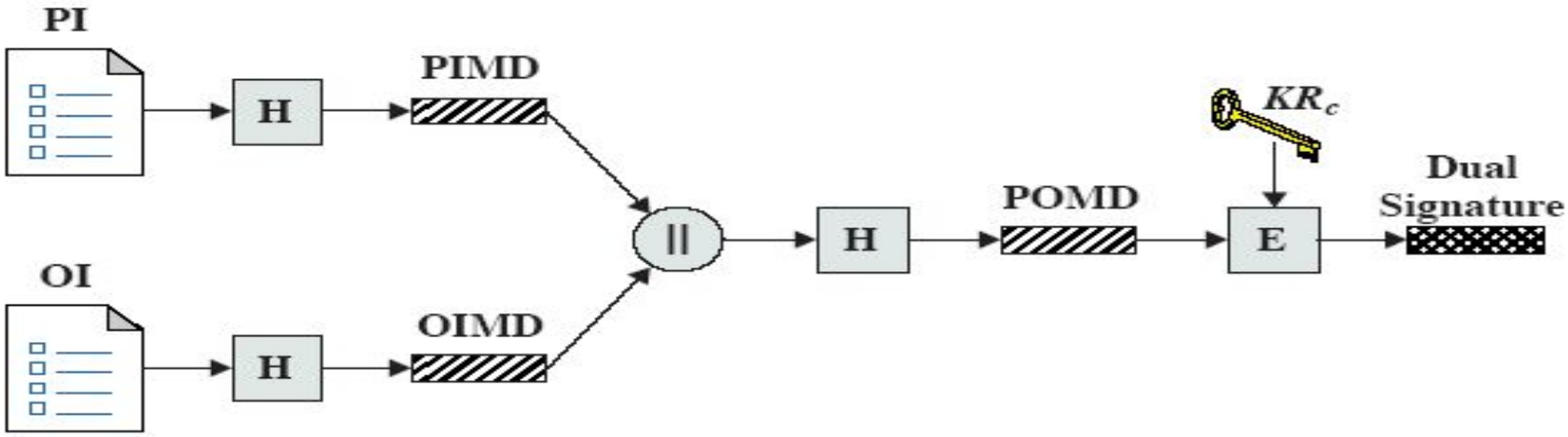
SET Transaction

1. customer opens account
2. customer receives a certificate
3. merchants have their own certificates
4. customer places an order
5. merchant is verified
6. order and payment are sent
7. merchant requests payment authorization
8. merchant confirms order
9. merchant provides goods or service
10. merchant requests payment

Dual Signature

- customer creates dual messages
  - order information (OI) for merchant
  - payment information (PI) for bank
- neither party needs details of other
- but **must** know they are linked
- use a dual signature for this
  - signed concatenated hashes of OI & PI

Dual Signature



PI = Payment Information
OI = Order Information
H = Hash function (SHA-1)
|| = Concatenation

PIMD = PI message digest
OIMD = OI message digest
POMD = Payment Order message digest
E = Encryption (RSA)
KR_c = Customer's private signature key

Construction of Dual Signature

In summary,

1. The merchant has received OI and verified the signature.
2. The bank has received PI and verified the signature.
3. The customer has linked the OI and PI and can prove the linkage.

SET Transaction Types (Table 17.3)

Following transactions occurs :

- Purchase request
- Payment authorization
- Payment capture

Purchase Request

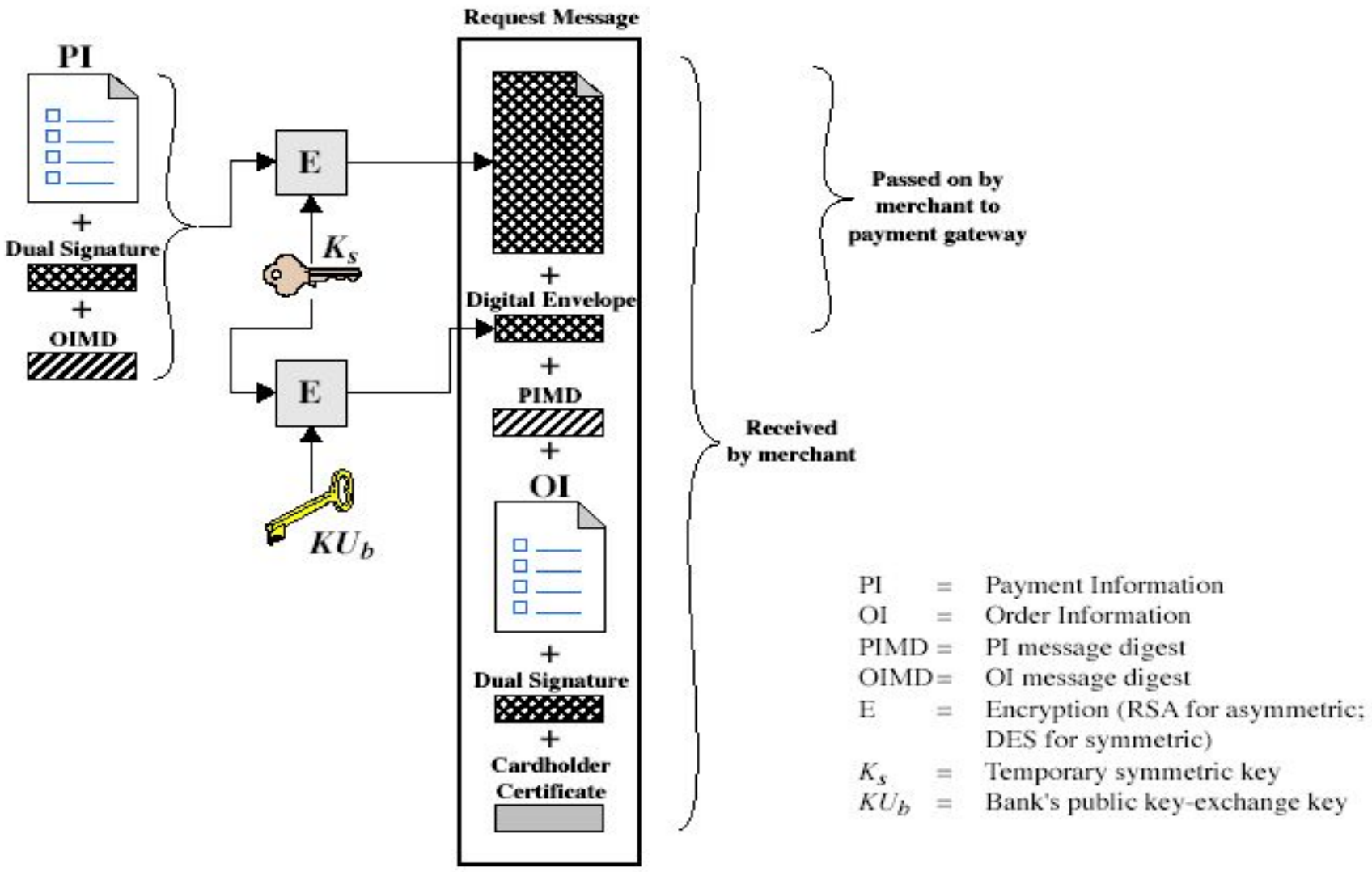
Purchase request exchange consists of four messages:

- Initiate Request
- Initiate Response
- Purchase Request and
- Purchase Response

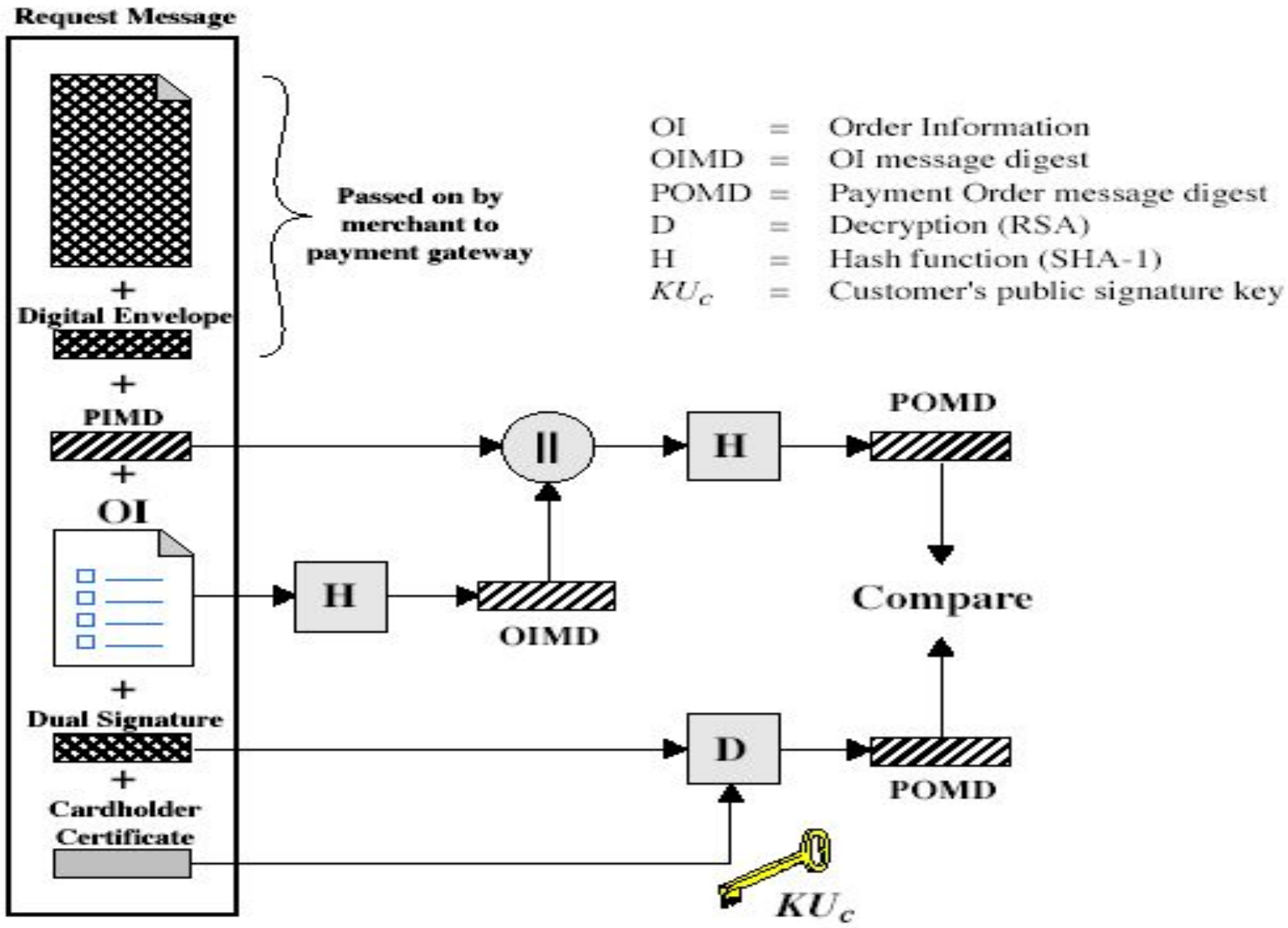
Initiate Request: The customer requests the certificates in the Initiate Request message, sent to the merchant. Also contains nonce to ensure timeliness

Initiate Response: The merchant generates a response and signs it with its private signature key. The response includes the nonce from the customer, another nonce for the customer to return in the next message, and a transaction ID for this purchase transaction.,merchant's signature certificate and the payment gateway's key exchange certificate.

Purchase Request – Customer



Purchase Request – Merchant



Purchase Request – Merchant

1. verifies cardholder certificates using CA signatures
2. verifies dual signature using customer's public signature key to ensure order has not been tampered with in transit & that it was signed using cardholder's private signature key
3. processes order and forwards the payment information to the payment gateway for authorization (described later)
4. sends a purchase response to cardholder
   - The Purchase Response message includes a response block that acknowledges the order and references the corresponding transaction number.
   - This block is signed by the merchant using its private signature key.

Payment Gateway Authorization

1. verifies all certificates
2. decrypts digital envelope of authorization block to obtain symmetric key & then decrypts authorization block
3. verifies merchant's signature on authorization block
4. decrypts digital envelope of payment block to obtain symmetric key & then decrypts payment block
5. verifies dual signature on payment block
6. verifies that transaction ID received from merchant matches that in PI received (indirectly) from customer
7. requests & receives an authorization from issuer
8. sends authorization response back to merchant

Payment Capture

- merchant sends payment gateway a payment capture request
- gateway checks request
- then causes funds to be transferred to merchants account
- notifies merchant using capture response

SSL	TLS
Secure Socket Layer.	Transport Layer Security.
Supports the Fortezza algorithm (Encryption, Hashing, DSA)	Does not support the Fortezza algorithm.
Common in the 3.0 version.	Common in the 1.0 version.
Message digest is used to create a master secret.	IPseudo-random function is used to create a master secret.
MAC based protocol is used.	HMAC based protocol is used.
More complex than TLS.	simple.
Less secured as compared to TLS.	provides high security.
Less reliable and slower.	Highly reliable and upgraded. It provides less latency.
SSL has been depreciated.	TLS is still widely used.
Uses port to set up explicit connection.	Uses protocol to set up implicit connection.

Summary

- need for web security
- SSL/TLS transport layer security protocols and their comparison
- SET secure credit card payment protocols
- SET and Dual Signature