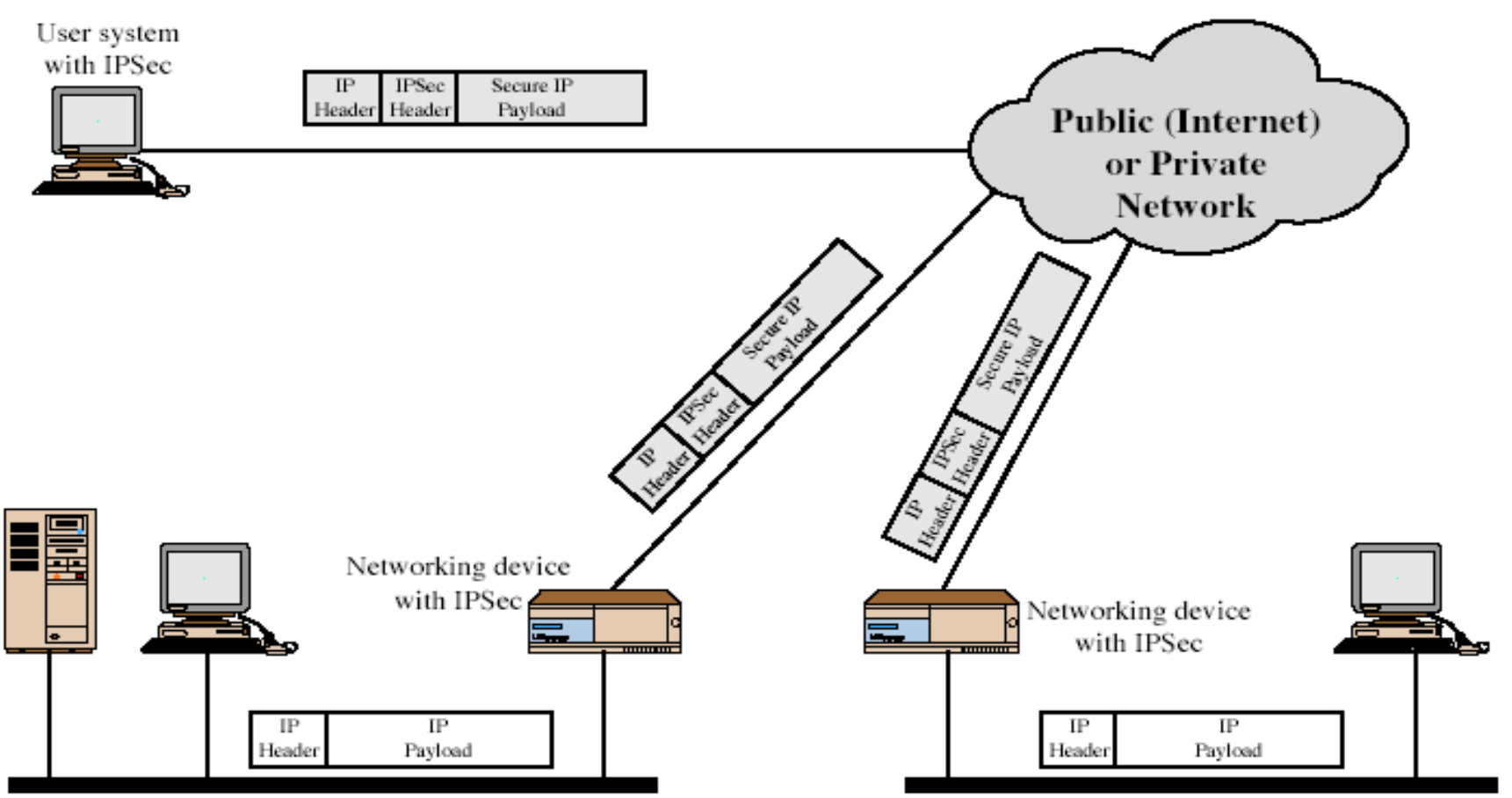
IT 422 Network Security

Web Security

Yasser F. O. Mohammad

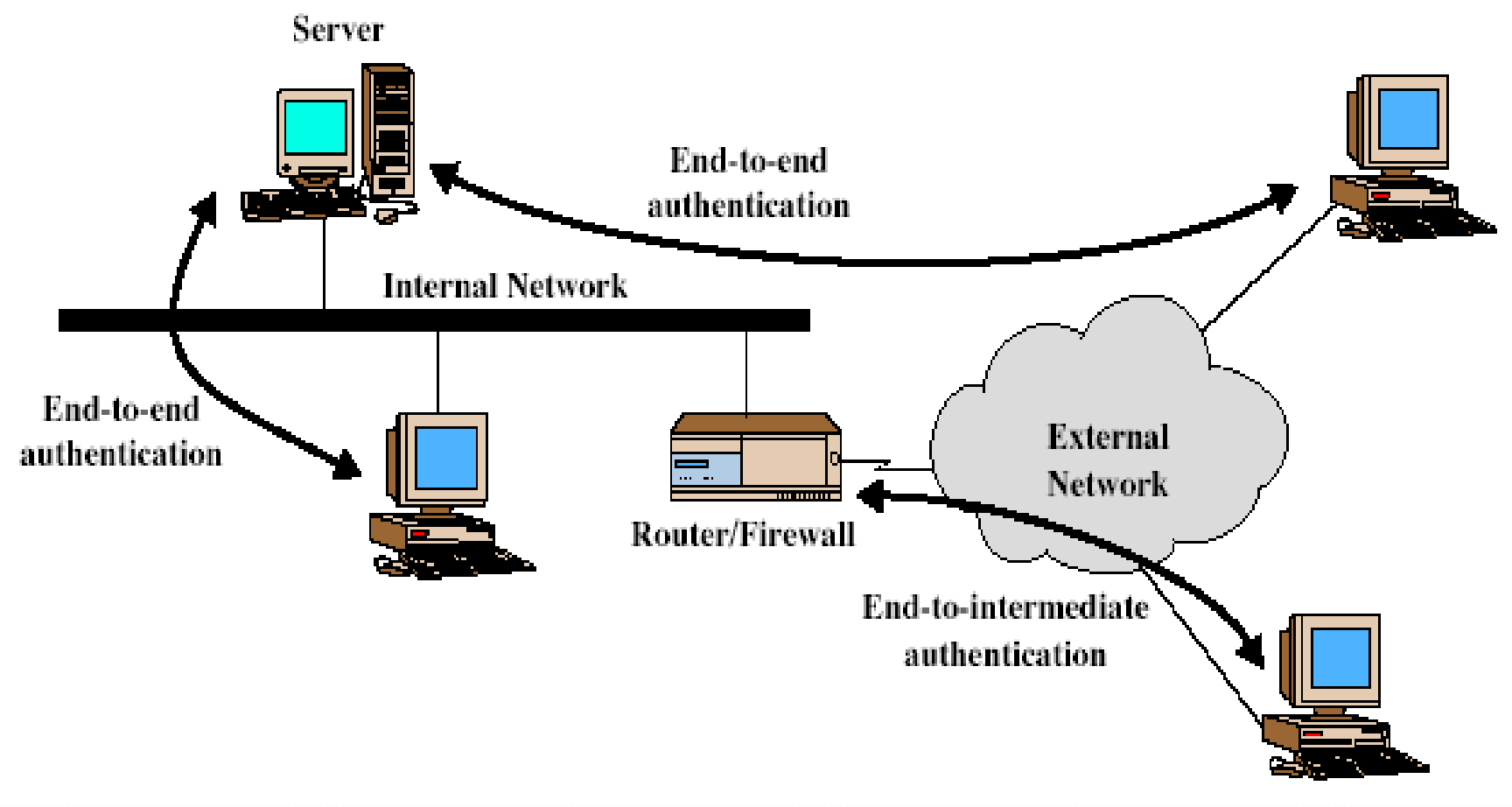
REMIDER 1: IPSec Uses



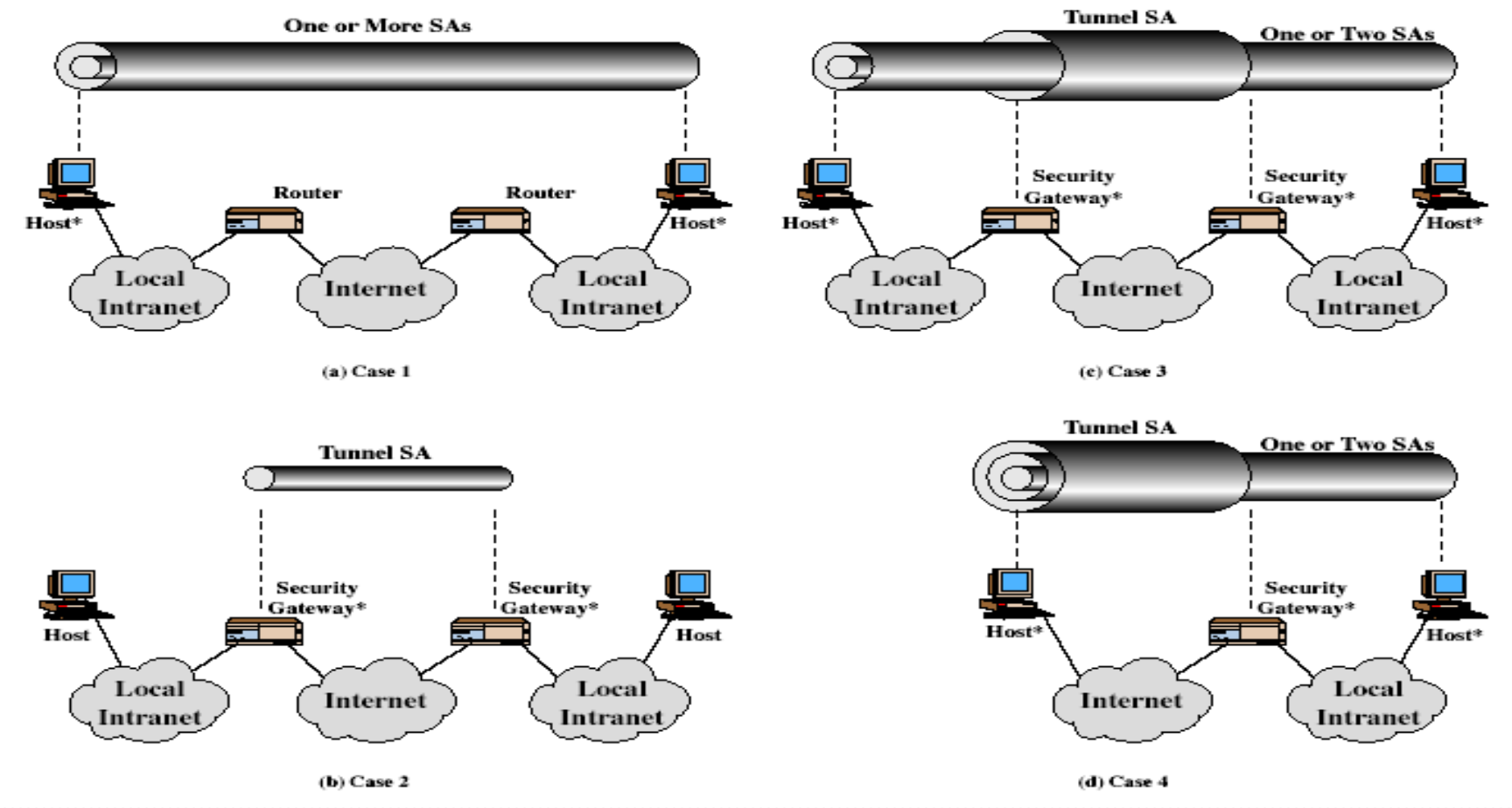
REMINDER 2: IPSec Services

- Access control
- Connectionless integrity
- Data origin authentication
- Rejection of replayed packets
  - a form of partial sequence integrity
- Confidentiality (encryption)
- Limited traffic flow confidentiality

REMINDER 3: Transport & Tunnel Modes



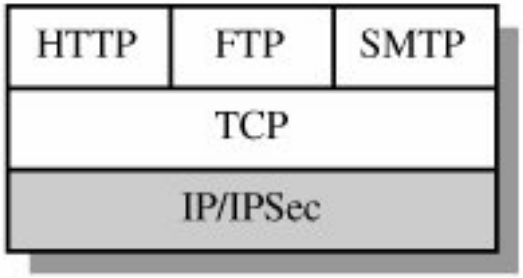
REMINDER 4: Combining Security Associations



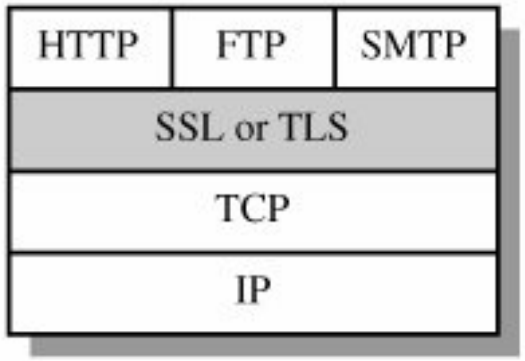
Why Web is Special?

- Two way
  - I can attack you!!
- Highly visible outlet
  - They cannot even protect their server!!
- Complex Underlying Software
  - Many Achill's ankles
- Web Servers are connected to the database
  - A window to the home
- Everyone uses it
  - The new employee does not know about certificates!!

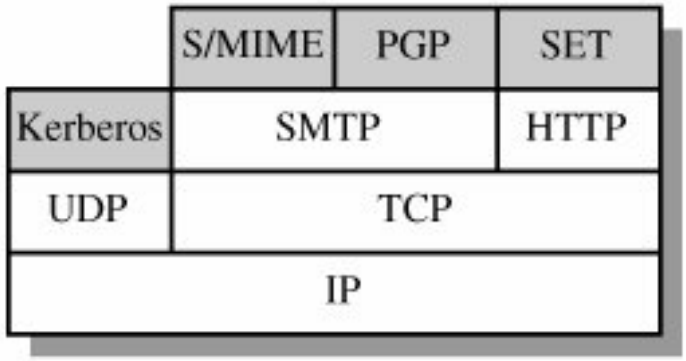
Approaches



(a) Network level



(b) Transport level



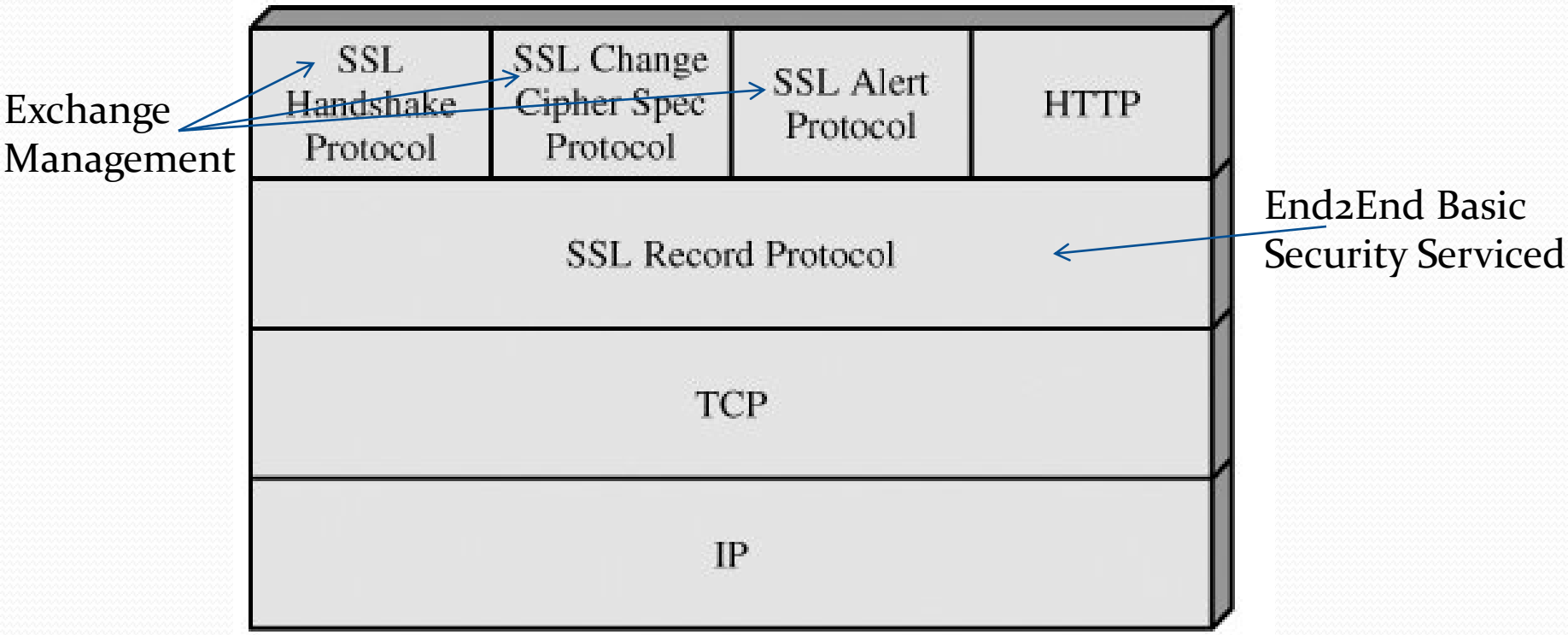
(c) Application level

SSL and TLS

- SSL
  - Secure Socket Layer
  - Netscape originated
  - Version 3 was published as an Internet Draft
- TLS (Transport Layer Security)
  - Formed within IETF
  - Can be considered SSL version 3.1

SSL

- Two layers of protocols



Connection and Session

- 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.

- Sessions live longer than connections
- Multiple concurrent connections are common
- Multiple concurrent sessions are possible but not usually used

Session States

- 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:
- Cipher spec:
- 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.

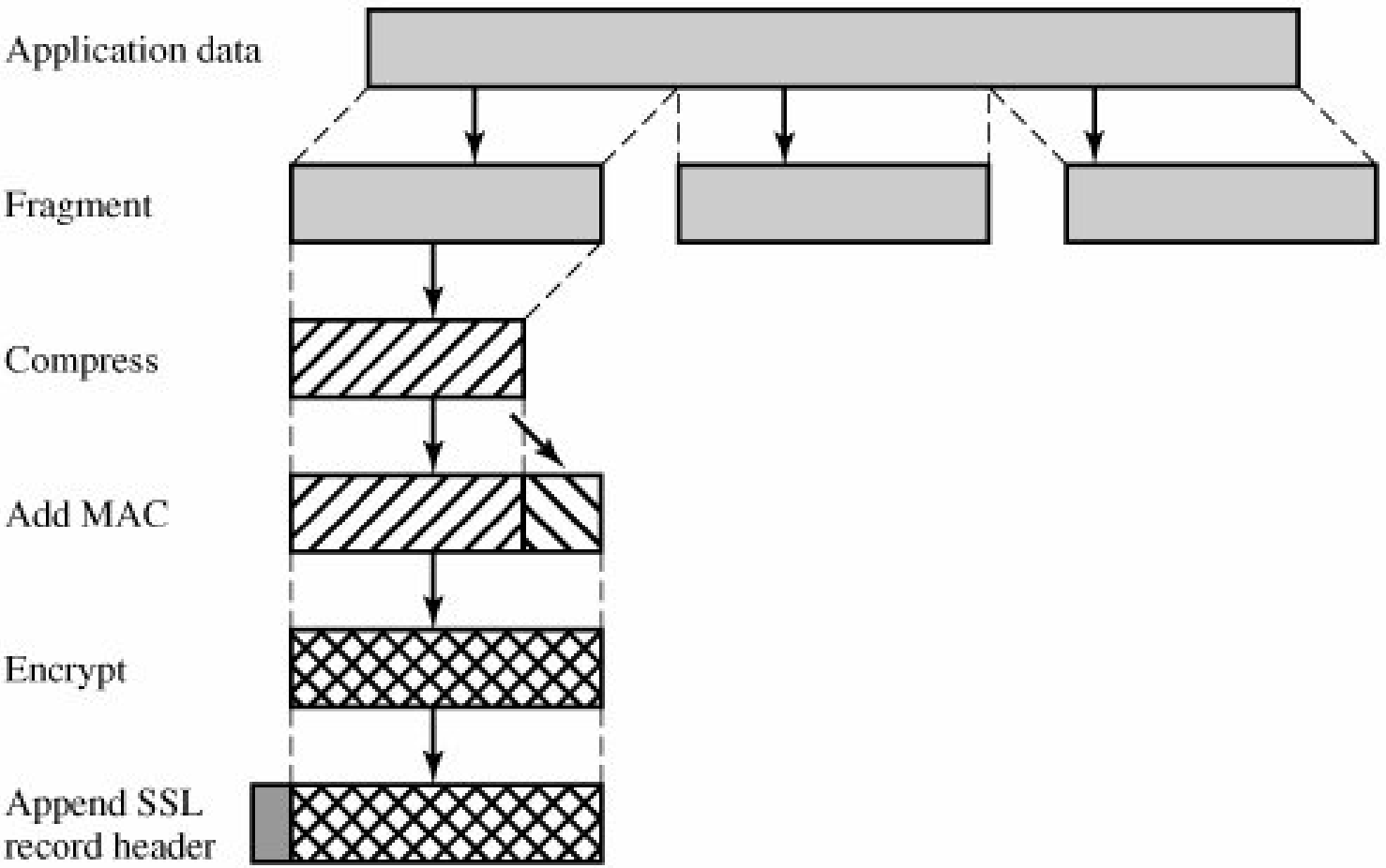
Connection States

- 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 Vector
- Sequence numbers:
  - Separate sequence numbers for transmitted and received messages for each connection. When a party sends or receives a change cipher spec message, the appropriate sequence number is set to zero. Sequence numbers may not exceed $2^{64} - 1$.

SSL Record Protocol

- Services
  - Confidentiality:
    - The Handshake Protocol defines a shared secret key that is used for conventional encryption of SSL payloads.
  - Message Integrity:
    - The Handshake Protocol also defines a shared secret key that is used to form a message authentication code (MAC).

SSL Record Protocol Overview



MACing

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

pad_1	=	the byte 0x36 (0011 0110) repeated 48 times (384 bits) for MD5 and 40 times (320 bits) for SHA-1
pad_2	=	the byte 0x5C (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)

Encryption

Block Cipher		Stream Cipher	
Algorithm	**Key Size**	**Algorithm**	**Key Size**
AES	128,256	RC4-40	40
IDEA	128	RC4-128	128
RC2-40	40		
DES-40	40		
DES	56		
3DES	168		
Fortezza	80		

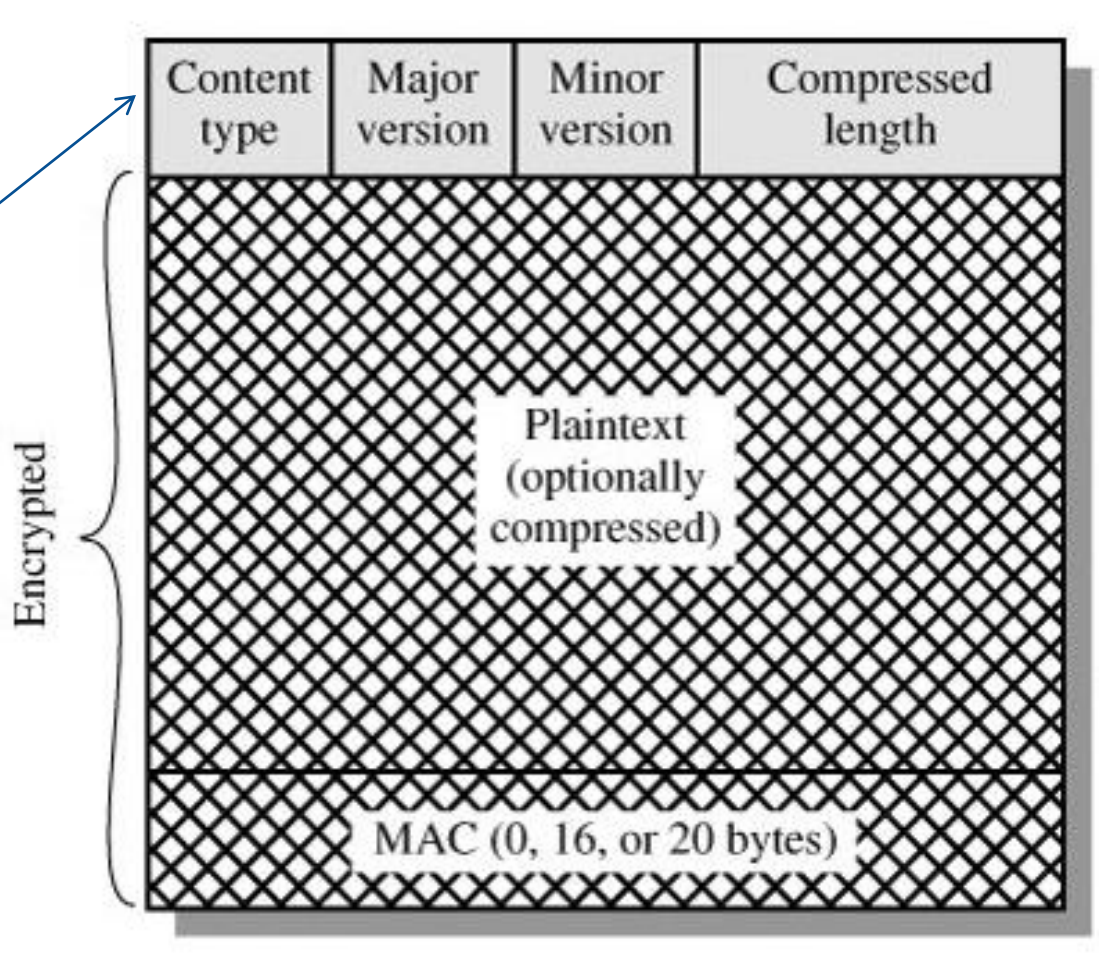
SSL Record Format

Change_cipher_spec

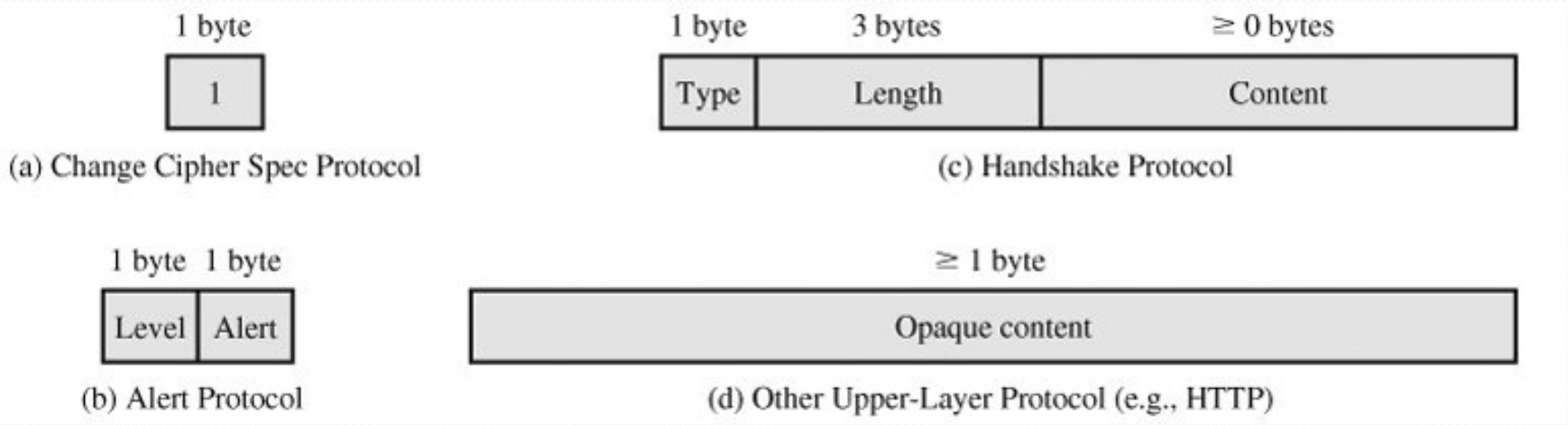
Alert

Handshake

Application_data

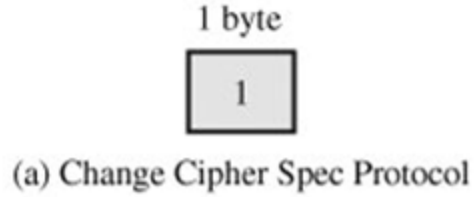


Higher Level Protocols



Change Cipher Spec Protocol

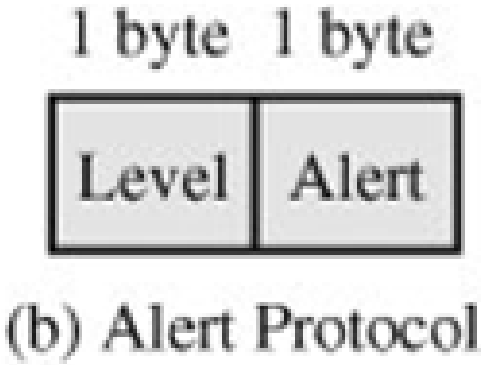
- Causes pending state to be copied into current state
- This updates the parameters of the cryptographic system



(a) Change Cipher Spec Protocol

Alert Protocol

- Convey important events
- First byte (severity)
  - 1=warning
  - 2=fatal
- Second byte (event)
  - unexpected_message:
  - bad_record_mac:
  - decompression_failure:
  - handshake_failure:
  - etc



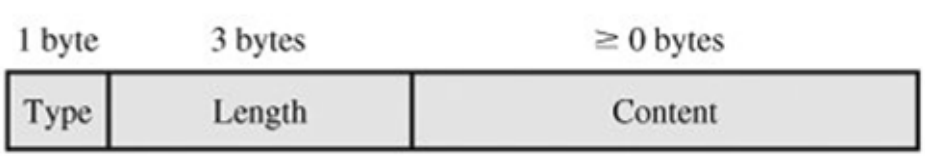
(b) Alert Protocol

Handshake Protocol

- Mutual Authentication
- Cryptographic parameters and key agreement

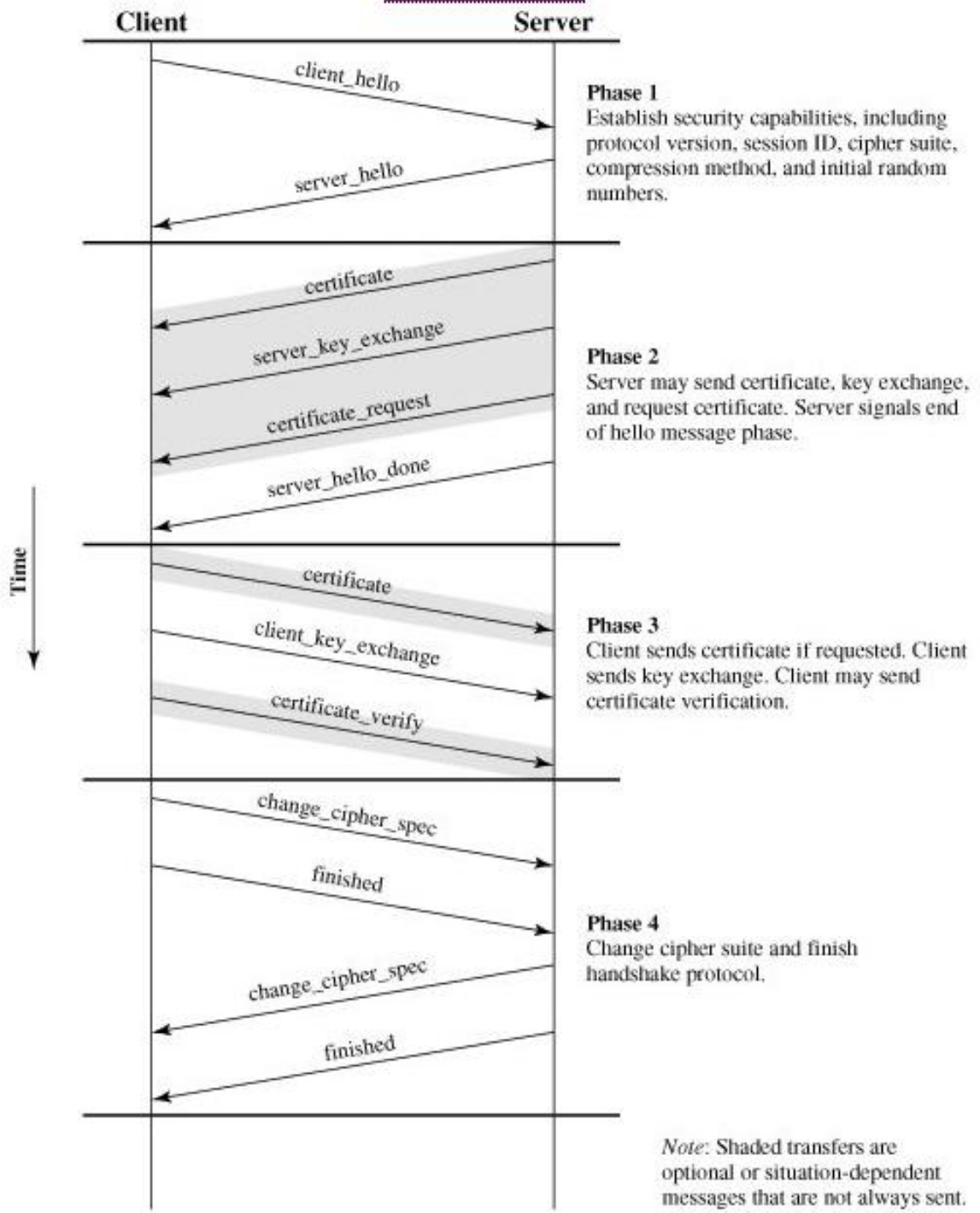
- Most complex part of SSL

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

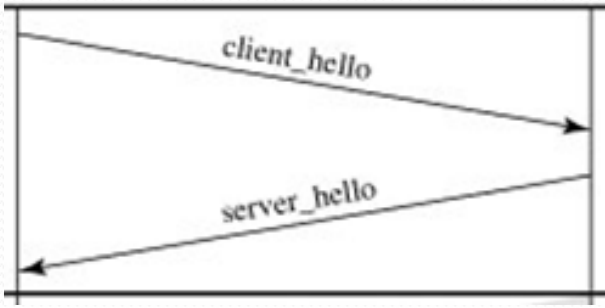


(c) Handshake Protocol

Handshake Protocol



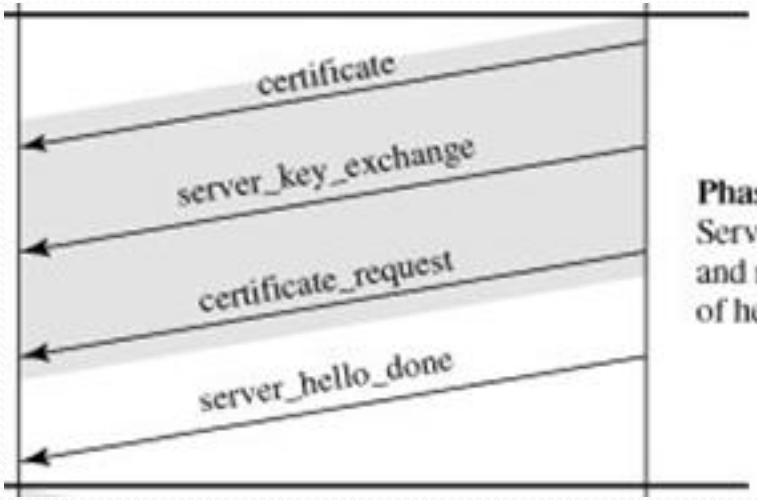
Phase 1: Establish Security Capabilities



- Initiates a connection
- Establishes security capabilities

- Content of *_***hello*** messages
  - Version:
    - The highest SSL version understood by the client.
  - Random:
    - 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 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. 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.

Phase 2:
Server Authentication and Key Exchange



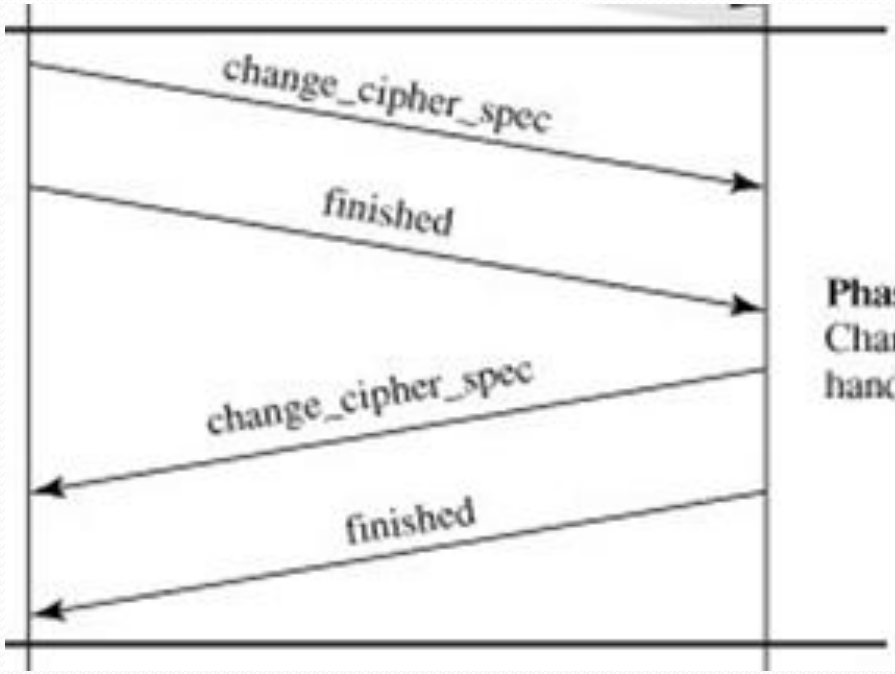
Phase 2
Server may send certificate, key exchange, and request certificate. Server signals end of hello message phase.

Phase 3:
Client Authentication and Key Exchange

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

```
CertificateVerify.signature.md5_hash
     MD5(master_secret || pad_2 || MD5(handshake_messages ||
         master_secret || pad_1));
Certificate.signature.sha_hash
      SHA(master_secret || pad_2 || SHA(handshake_messages ||
          master_secret || pad_1));
```

Phase 4: Finish



Phase 4
Change cipher suite and finish handshake protocol.

```
MD5(master_secret || pad2 || MD5(handshake_messages ||
    Sender || master_secret || pad1))
SHA(master_secret || pad2 || SHA(handshake_messages ||
    Sender || master_secret || pad1))
```

Differences between SSL and TLS

- SELF READ

Secure Electronic Transaction

- Provides a secure communications channel among all parties involved in a transaction
- Provides trust by the use of X.509v3 digital certificates
- Ensures privacy because the information is only available to parties in a transaction when and where necessary

SET Requirements

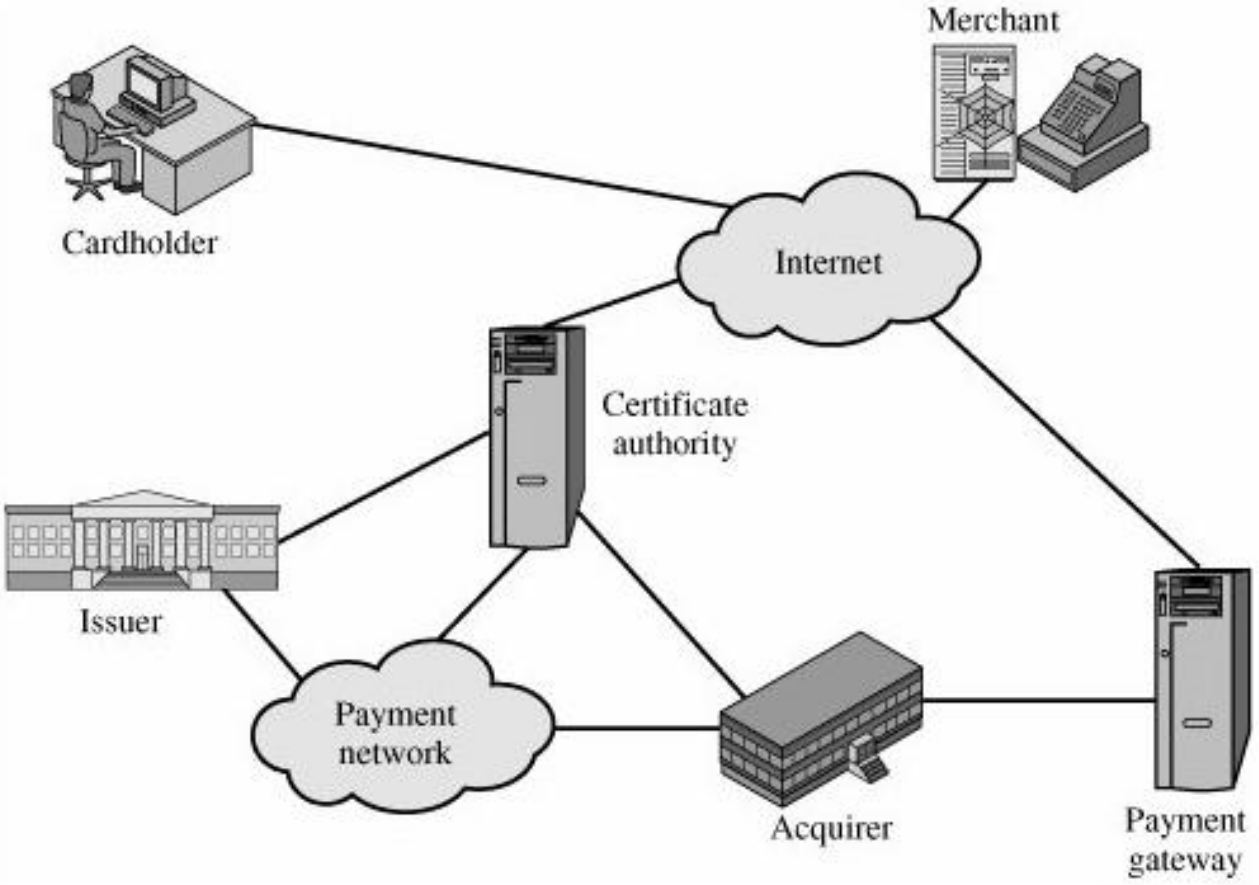
1. Provide confidentiality of payment and ordering information
2. Ensure the integrity of all transmitted data
3. Provide authentication that a cardholder is a legitimate user of a credit card account
4. Provide authentication that a **merchant** can accept credit card transactions through its relationship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

SET features

- 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. Conventional encryption by DES is used to provide confidentiality.
- 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. Certain messages are also protected by HMAC using SHA-1.
- Cardholder account authentication:
  - SET enables merchants to verify that a cardholder is a legitimate user of a valid card account number. SET uses X.509v3 digital certificates with RSA signatures for this purpose.
- Merchant authentication:
  - SET enables cardholders to verify that a merchant has a relationship with a financial institution allowing it to accept payment cards. SET uses X.509v3 digital certificates with RSA signatures for this purpose.

- Note that unlike IPSec and SSL/TLS, SET provides only one choice for each cryptographic algorithm

SET Parties

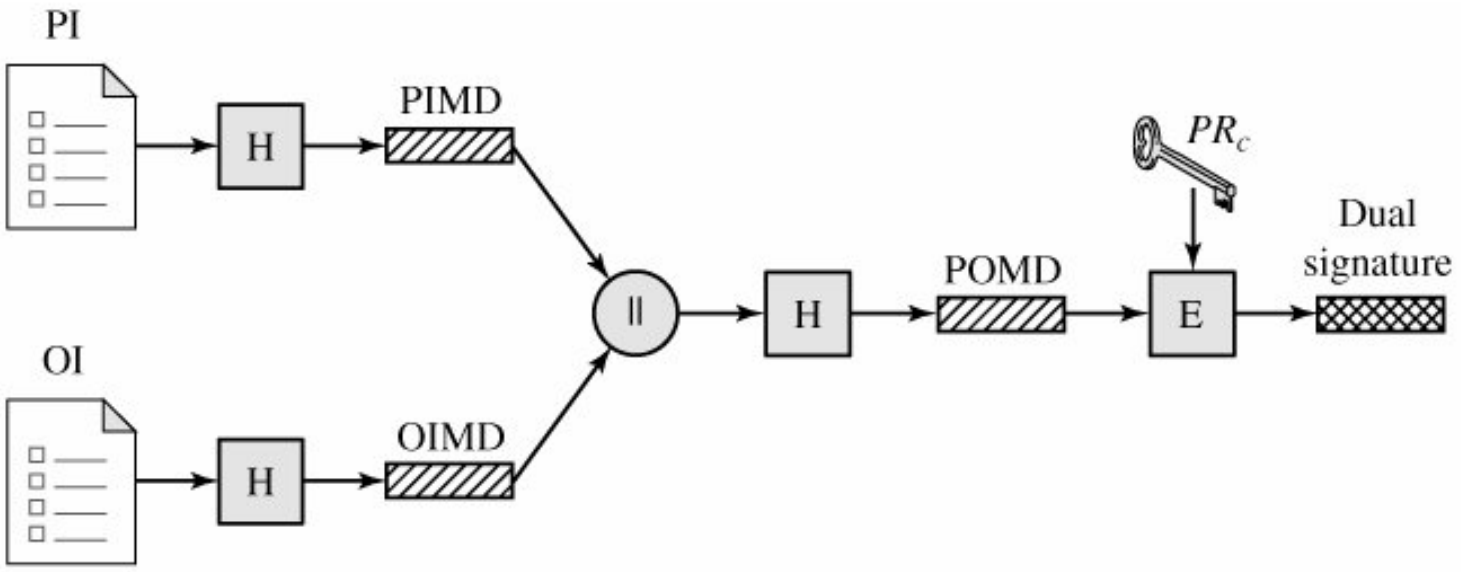

Merchant
Cardholder
Internet
Certificate
authority
Issuer
Payment
network
Acquirer
Payment
gateway

SET sequence of events

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

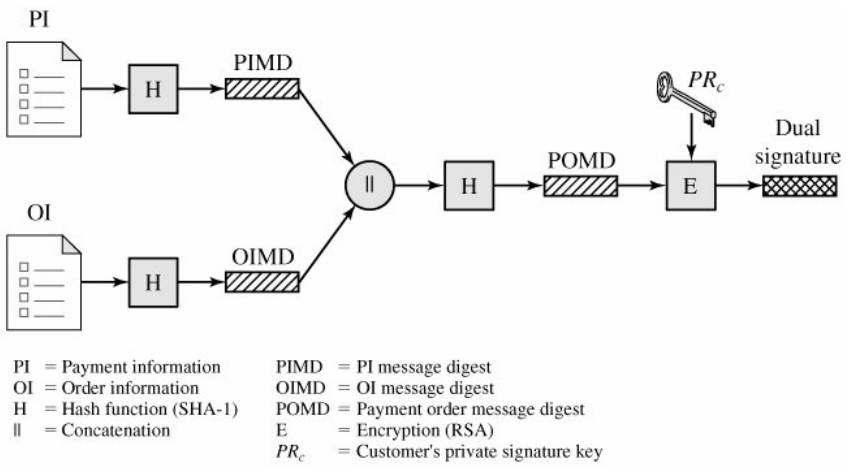
Dual Signature

- Links two messages that are sent to two different destinations (Order to merchant and Payment to bank)

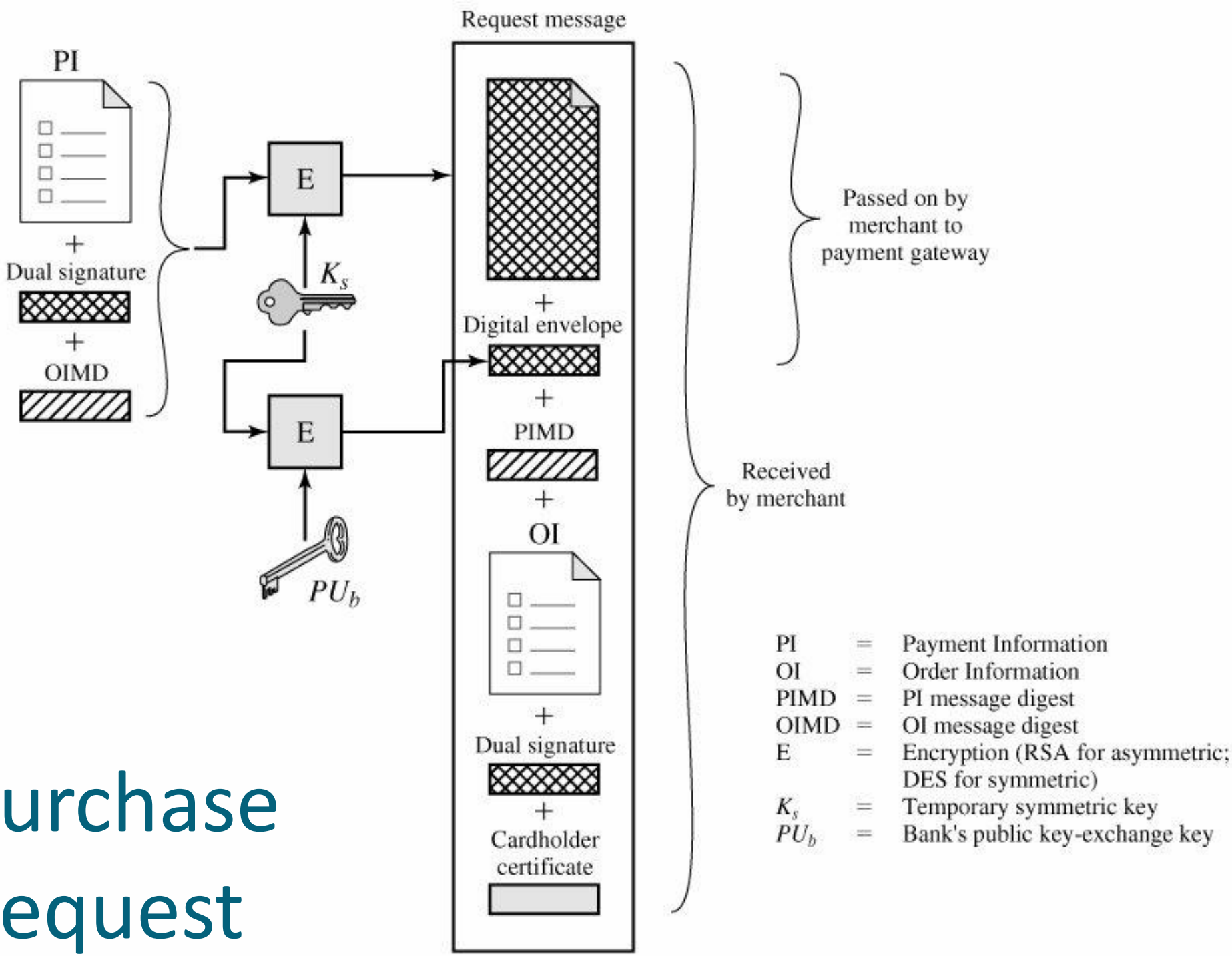


Verifying DS

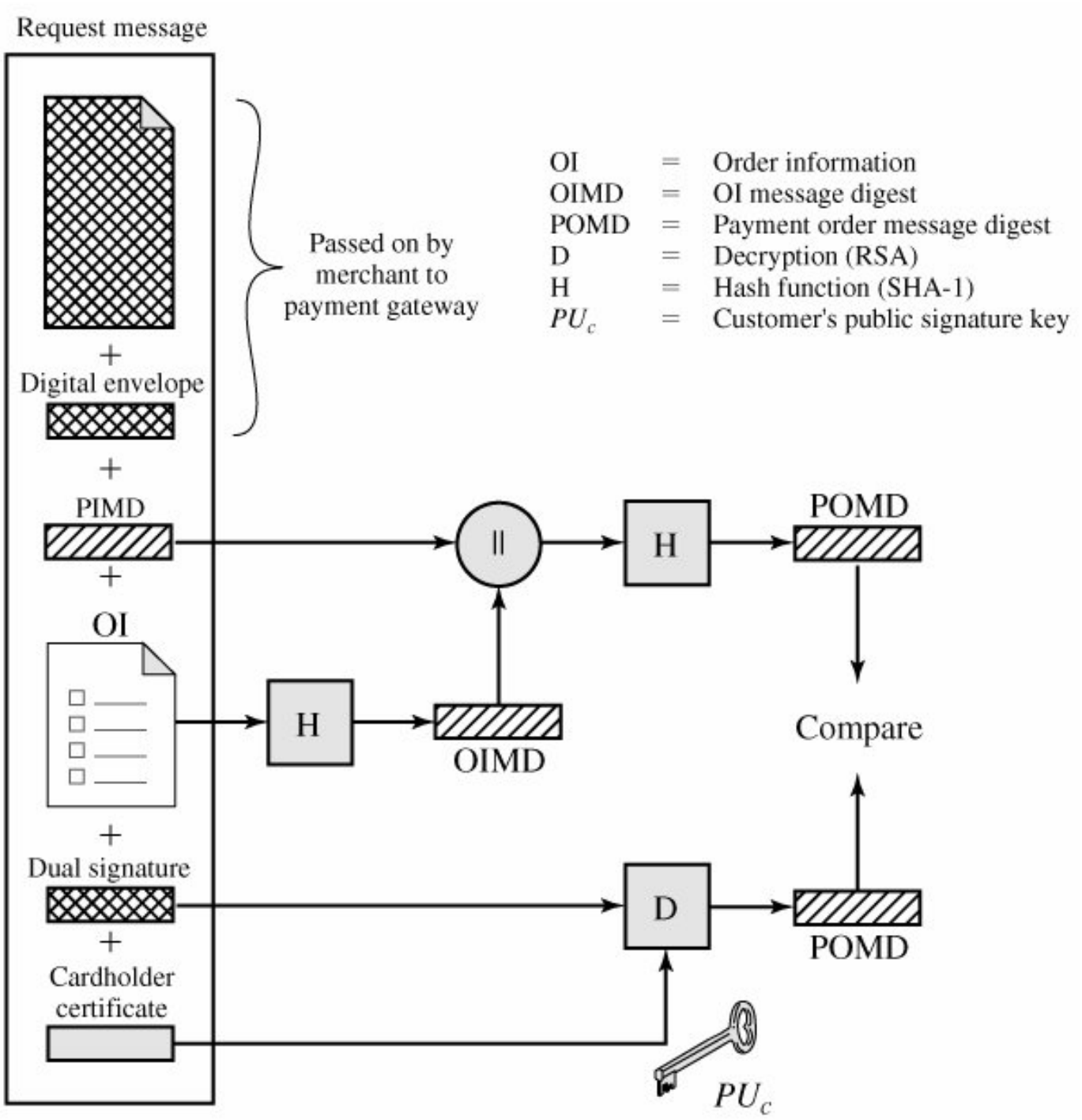
- Merchant
  - Input: PIMD, OI, DS
  - Compute h1= PIMD||H(OI)
  - Compute h2=D(PU_c, DS)
  - h1 must equal h2
- Bank
  - Input: OIMD, PI, DS
  - Compute h1= H(PI)||OIMD
  - Compute h2=D(PU_c, DS)
  - h1 must equal h2



Purchase Request



Merchant Verification of Request



Payment Authorization

- SELF READ