

# Computer Security

## Network Security 2

1

# Web Security

- most commercial and governmental organisations have web sites
- significant volumes of transactions are being carried out over the web
- as a consequence demand for secure web services is growing

2

# World Wide Web

- a client/server application
- runs over the internet, using TCP/IP
- the underlying software is complex and any complex software may contain security flaws
- once a web server is subverted it may be used as an entry into the organisation's network

3

# Security Threats

- has the same needs as other computer communication
  - confidentiality
  - integrity
  - authentication
  - protection against denial of service

4

## Provision of Security

- could use IPsec
  - network level
- other possible locations for the security are above the transport level (SSL/TLS) or at the application level (Kerberos, PGP, SET, etc)

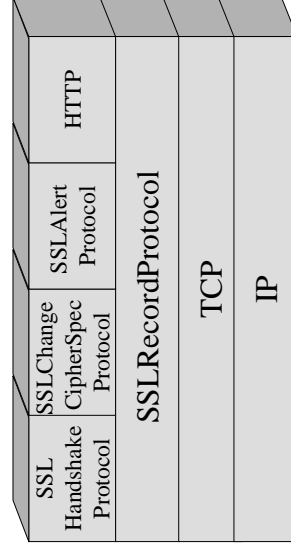
6

## SSL/TLS

- Secure Socket Layer (SSL)/Transport Layer Security (TLS)
- SSL originated by Netscape
- can be placed in protocol stack or incorporated in browser
- TLS is internet-standardised version

6

## SSL Protocol Stack



7

## SSL Architecture

- runs over TCP
- provides security to higher level protocols, such as HTTP
- connection
- session

8

## Connection

- transport layer connection
- peer to peer relationship
- transient
- associated with a single session

9

## Session

- an association between client and server
- created by handshake protocol
- define a set of cryptographic parameters
- govern multiple connections

10

## Session State

- session identifier
- peer certificate (X.509v3)
- compression method
- cipher spec (encryption and integrity)
- master secret
- is resumable

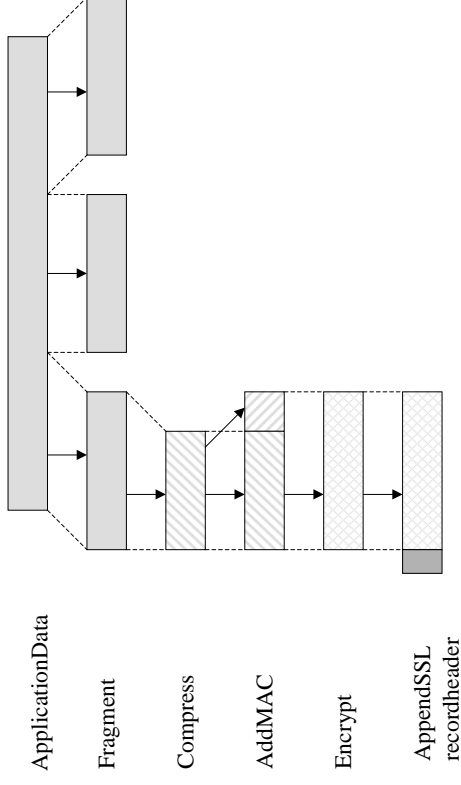
11

## Connection State

- server and client random
- keys
- initialisation vectors
- sequence numbers

12

## SSL Record Protocol



13

## SSL Record Protocol

- fragmentation splits user data into blocks of  $2^{14}$  bytes or less
- compression is optional and, if used, must be lossless and increase length by no more than 1024 bytes
- MAC generated using a version of HMAC (see text)

14

## SSL Record Protocol Supported Ciphers

- IDEA (128 bit)
- RC2-40 (40 bit)
- DES-40 (40 bit)
- DES (56 bit)
- 3DES (168 bit)
- Fortezza (80 bit)
- RC4-40 (40 bit, stream cipher)
- RC4-128 (128 bit, stream cipher)

15

## Change Cipher Spec and Alert Protocols

- see text

16

## Handshake Protocol

- authenticates client and server
- allows them to negotiate encryption and MAC algorithms and keys
- most complex part of SSL
- used before any application data transmitted

17

## Handshake Protocol Format

- Type (1 byte) - indicates one of 10 messages
- length (3 bytes) - length of message in bytes
- content ( $\geq 0$  bytes) - parameters associated with message

18

## SSL Handshake Protocol

```

C → S : client_hello
S → C : server_hello
S → C : certificate (optional)
S → C : server_key_exchange (optional)
S → C : certificate_request (optional)
S → C : server_hello_done
C → S : certificate (optional)
C → S : client_key_exchange
C → S : certificate_verify (optional)
C → S : change_cipher_spec
C → S : finished
S → C : change_cipher_spec
S → C : finished

```

19

## SSL Handshake Protocol

- client, server\_hello : version, nonce (random), session id, cipher suite, compression method
- certificate : chain of X.509v3 certificates
- client,server\_key\_exchange : parameters, signature
- certificate\_verify : signature on handshake messages,etc
- certificate\_request : type, authorities

20

## Key Exchange Methods

- RSA
- Diffie Hellman
- Fortezza

21

## Key Generation

- if using RSA, client generates 48 byte pre\_master\_secret and sends it in client\_key\_exchange
- pre\_master\_secret can be viewed as seed for pseudo-random function
- nonces as salt to make cyptanalysis harder

22

## Key Generation

Master secret creation

```

master_secret = MD5(pre_master_secret || SHA ('A' ||
pre_master_secret || ClientHello.nonce || ServerHello.nonce)) ||
MD5(pre_master_secret || SHA ('BB' ||
pre_master_secret || ClientHello.nonce || ServerHello.nonce)) ||
MD5(pre_master_secret || SHA ('CCC' ||
pre_master_secret || ClientHello.nonce || ServerHello.nonce))

```

23

## Cryptographic Parameters

- need
  - client write key
  - server write key
  - client write MAC key
  - server write MAC key
  - client write IV
  - server write IV

24

## Cryptographic Parameters

- generated using same format as master\_secret (master\_secret replaces pre\_master\_secret)
- if necessary extended ('DDDD', 'EEEE', ...) until output is large enough to create all necessary values

25

## SET

- Secure Electronic Transactions
- developed by Visa and Mastercard
- its objective is to allow secure credit card transactions over the web

26

## Basic Outline

- customer and vendor have certificates, issued by bank
- all messages are encrypted
- bank checks and authorises payment

27

## In More Detail

- customer browses, either electronically or via paper catalog, and selects items for purchase
- customer fills in electronic order form and selects means of payment
- order and payment instructions are signed and sent to vendor

28

## Cont.

- merchant requests authorisation from customer's financial institution
- merchant send confirmation of order
- merchant ships goods or performs service
- merchant requests payment from customer's financial institution

29

## Problems

- in a single message client wants to specify
  - goods ordered
  - bank account details
- customer wants merchant and bank to each see one part of the message only

30

## So...

- the designers of SET could simply have had the customer sign each part of the two part of the message separately
- but what would stop someone mixing and matching orders and payment info?

31

## Solution

- a **dual signature**
- doesn't involve two signing keys
- signature is created as follows
  - create digests of the two messages
  - concatenate result and compute digest of that
  - sign it (ie, encrypt with private key)

32

## Checking Signature

- signer sends one message and digest of other
- verifier can create digest of message they were sent
- then redo concatenation and digest computation
- then check signature using public key

33

## In SET

- both message sections are initially sent to the merchant
- however, the part intended for the bank is (can be) encrypted under the public key of the bank

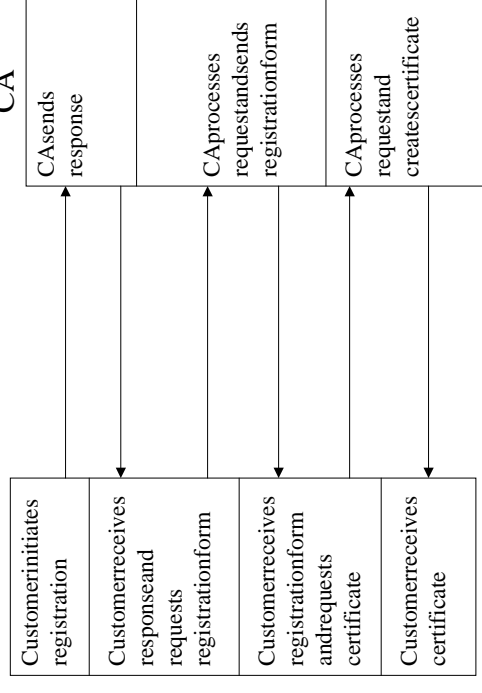
34

## Registration

- both customer and merchant must be registered with a CA
- a bank may act as a CA

35

## Customer Registration Outline



36

## Notes

- CA response includes CA's certificate
- after receiving CA certificate customer generates symmetric session key
- encrypts account details to CA encrypted under session key
- sends with this session key, encrypted under CA's public key

37

## Notes

- customer fills in registration form
- generates key pair, if necessary
- generates random number to be used by CA in creating certificate
- generates two symmetric keys
- uses one to encrypt message to CA

38

## Notes

- message includes other key (used by CA to encrypt response) and filled in registration form
- CA generates random number
- combines this with customer random number to generate secret value
- this secret value is used to protect account information in certificate

39

## Notes

- account information, expiration date and secret value are hashed and placed in certificate
- CA creates and signs certificate
- CA sends response, including its random number, encrypted by other key generated by customer

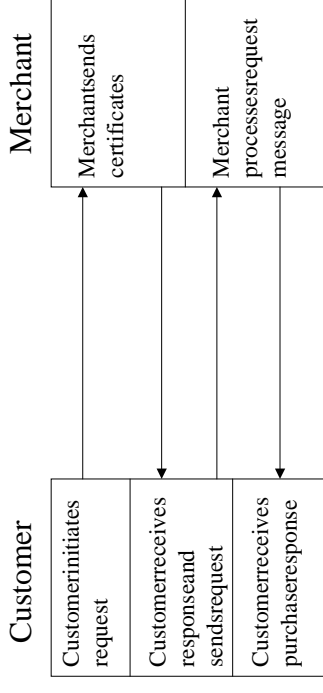
40

## Merchant Registration

- similar, but four messages only
- begins with request for registration form

41

## Purchase Request



42

## Customer Initiates Request, Merchant Replies

- customer has completed order
- merchant assigns unique transaction id to order
- merchant sends its certificate(s) and certificate(s) of bank or payment gateway and transaction id

43

## Customer Sends Request

- customer generates order information (OI) and payment information (PI)
- includes merchant identifier in both transaction identifier in both
- customer generates dual signature for OI and PI
- customer generates random symmetric key for encryption

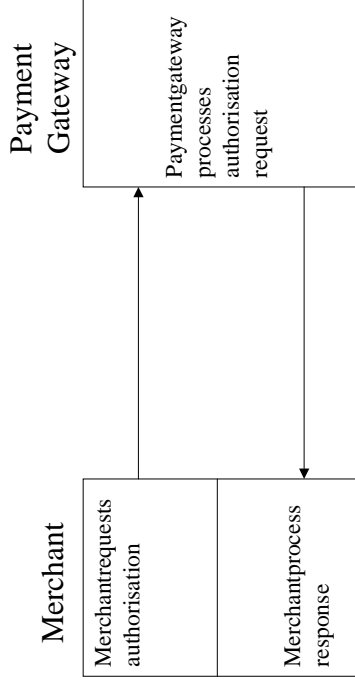
44

## Merchant Process Request

- checks signatures
- processes order, including authorising payment
- generates response, signs it and includes certificate
- customer verifies signatures

45

## Payment Authorisation



46

## Payment Authorisation

- merchant generates and signs authorisation request
- includes amount to be authorised and transaction id from OI
- encrypted using randomly generated symmetric key, which is encrypted under public key of payment gateway

47

## Payment Authorisation

- gateway decrypts symmetric key then decrypts authorisation request
- decrypts symmetric key of PI and then decrypts PI
- verifies signatures at each step
- Uses digest of OI (included in PI) to check PI has not been altered (ie, checks dual signature)

48

## Payment Authorisation

- gateway sends an authorisation request to customer's bank
- on reply Gateway generates and signs authorisation response message
- response is encrypted under randomly generated key, which is encrypted under merchant's public key

49

## SET

- Full protocol definition about 260 pages
- Available from Visa, Mastercard and SETco

50

## Viruses

51

## Security in Mobile and Wireless Networks

52