INFS 767
Secure Electronic Commerce
Fall 1999

Lecture 6
Public-Key Certificates
SSL

Prof. Ravi Sandhu

PUBLIC-KEY CERTIFICATES

- **reliable distribution of public-keys**
- **public-key encryption**
  - **sender needs public key of receiver**
- **public-key digital signatures**
  - **receiver needs public key of sender**
- **public-key key agreement**
  - **both need each other's public keys**

© Ravi Sandhu 1999

2

X.509 CERTIFICATE

VERSION
SERIAL NUMBER
SIGNATURE ALGORITHM
ISSUER
VALIDITY
SUBJECT
SUBJECT PUBLIC KEY INFO
SIGNATURE

© Ravi Sandhu 1999

3

X.509 CERTIFICATE

0
1234567891011121314
RSA+MD5, 512
C=US, S=VA, O=GMU, OU=ISSE
5/1/97-5/1/98
C=US, S=VA, O=GMU, OU=ISSE, CN=Ravi Sandhu
RSA, 1024, xxxxxxxxxxxxxxxxxxxxxxxx
SIGNATURE

© Ravi Sandhu 1999

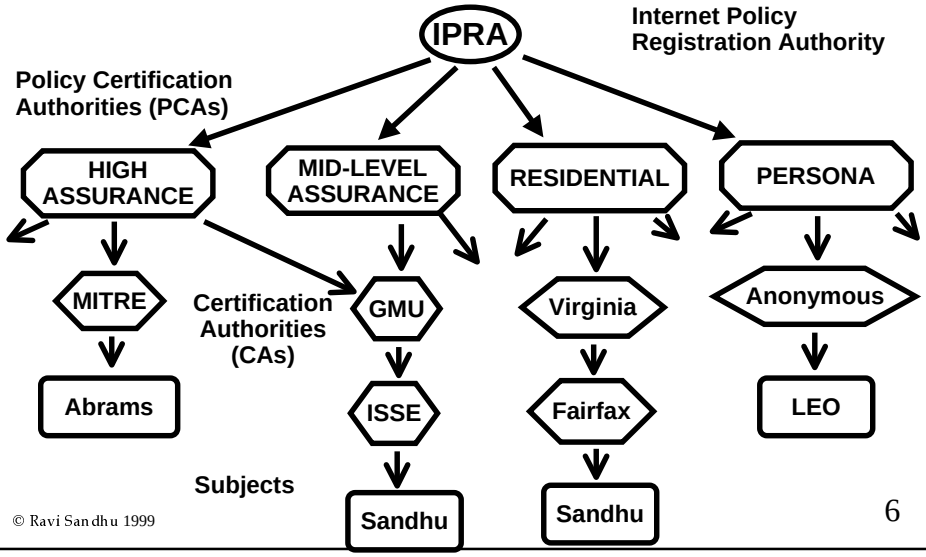
4

CERTIFICATE TRUST

- how to acquire public key of the issuer to verify signature
- whether or not to trust certificates signed by the issuer for this subject

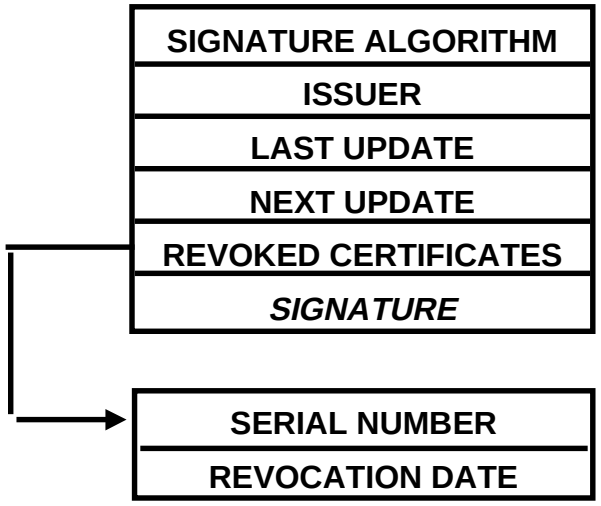
© Ravi Sandhu 1999
5

PEM CERTIFICATION GRAPH



© Ravi Sandhu 1999
6

CRL FORMAT



© Ravi Sandhu 1999

7

PGP BOTTOM UP TRUST MODEL

- **How does Alice get Bob's public key**
  - **directly from Bob through some secure channel (e.g., post, phone, floppy)**
  - **from Chuck, who is known to both Alice and Bob and introduces Bob to Alice**
  - **from a trusted certifying authority**
- **PGP has mechanisms to support these, and related, alternatives**

© Ravi Sandhu 1999

8

X.509 CERTIFICATES

- **X.509v1**
  - **very basic**
- **X.509v2**
  - **adds unique identifiers to prevent against reuse of X.500 names**
- **X.509v3**
  - **adds many extensions**
  - **can be further extended**

© Ravi Sandhu 1999
9

SEPARATE KEYS FOR SEPARATE PURPOSES

- **RSA is the only known public-key cryptosystem in which the same public-private key pair can be used for**
  - **digital signatures**
  - **encryption**
- **perceived as a major advantage**

© Ravi Sandhu 1999
10

SIGNATURE KEYS

- **private key: must be private for entire life, may never leave smart card**
  - **needs to be securely destroyed after lifetime**
  - **no need for backup or archiving (would conflict with above)**
  - **no need to weaken or escrow due to law**
- **public key: must be archive possibly for a long time**

© Ravi Sandhu 1999
11

ENCRYPTION KEY

- **private key: backup or archive required for recovery**
  - **should not be destroyed after lifetime**
  - **may be weakened/escrowed due to law**
- **public key:**
  - **no need to backup RSA or other encryption keys**
  - **need to backup Diffie-Hellman key agreement keys**

© Ravi Sandhu 1999
12

X.509 INNOVATIONS

- distinguish various certificates
  - signature, encryption, key-agreement
- identification info in addition to X.500 name
- name other than X.500 name
  - email address
- issuer can state policy and usage
  - good enough for casual email but not good enough for signing checks
- limits on use of signature keys for further certification

© Ravi Sandhu 1999
13

X.509v3 EXTENSIONS

- X.509v3 same as X.509v2 but adds extensions
- provides a general extension mechanism
  - extension type: registered just like an algorithm is registered
  - standard extension types: needed for interoperability

© Ravi Sandhu 1999
14

X.509v3 EXTENSIONS CRITICALITY

- non-critical: extension can be ignored by certificate user
  - alternate name can be non-critical
- critical : extension should not be ignored by certificate user
  - limit on use of signatures for further certification

© Ravi Sandhu 1999
15

X.509v3 EXTENSIONS CRITICALITY

- criticality is flagged by certificate issuer
  - certificate user may consider non-critical extensions more important than critical ones
  - certificate user may refuse to use certificate if some extensions are missing
- critical extensions should be few and should be standard

© Ravi Sandhu 1999
16

X.509v3 NAMES

- internet email address
- internet domain name
- web uri (url's are subset of uri)
- IP address
- X.400 email address
- X.500 directory name
- registered identifier
- other name

© Ravi Sandhu 1999
17

X.509v3 STANDARD EXTENSIONS

- Key and policy information
- Subject and issuer attributes
- Certification path constraints
- Extensions related to CRLs
  - will be discussed with CRLs

© Ravi Sandhu 1999
18

KEY AND POLICY INFORMATION

- key usage
  - critical: intended only for that purpose, limits liability of CA
  - non-critical: advisory to help find the correct key, no liability implication
- private-key usage period
  - certificate valid for 2 years for verifying signature
  - key valid only for one year for signing
- certificate policies
  - for CAs

© Ravi Sandhu 1999
19

SUBJECT AND ISSUER ATTRIBUTES

- Subject alternative names
- Issuer alternative names
- Subject directory attributes
  - whatever you like
  - position, phone, address etc.

© Ravi Sandhu 1999
20

CERTIFICATION PATH CONSTRAINTS

- **Basic Constraints**
  - **can or cannot act as CA**
  - **if can act as CA limit on certification path**
    - **limit=1 means cannot certify other CAs**
- **Name Constraints**
  - **limits names of subjects that this CA can issue certificates for**
- **Policy Constraints**
  - **concerned with CA policies**

© Ravi Sandhu 1999
21

CERTIFICATION PATH CONSTRAINTS

- **Basic Constraints**
  - **can or cannot act as CA**
  - **if can act as CA limit on certification path extending from here**
  - **limit=1 means cannot certify other CAs**
- **b. Name Constraints**
- **limits names of subjects that this CA can issue certificates for**

© Ravi Sandhu 1999
22

CERTIFICATE REVOCATION LISTS

- CRLs issued periodically as per CA policy
  - off-cycle CRLs may also be needed
  - blank CRLs can be issued

© Ravi Sandhu 1999
23

CERTIFICATE REVOCATION LISTS

- CRL distribution
  - pull method
  - push method
- DMS example
  - pull method with push for compromised key list (CKL) which is broadcast via secure email, single CKL for entire system

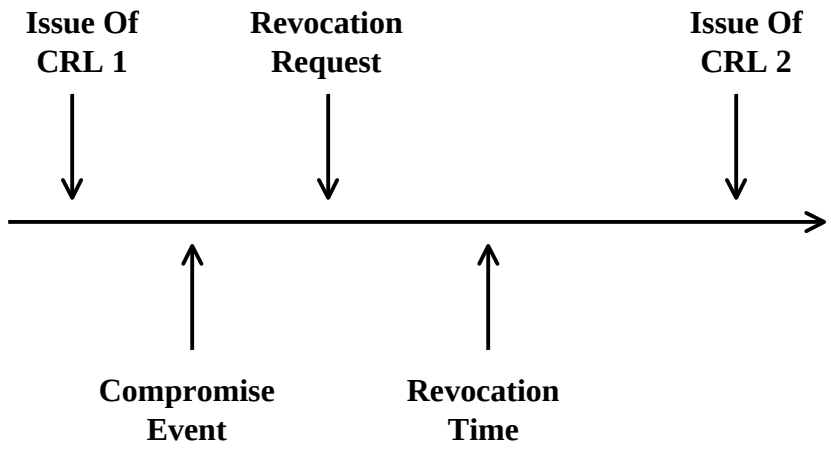
© Ravi Sandhu 1999
24

CERTIFICATE REVOCATION LISTS

- **immediate or real-time revocation**
  - **needs query to CA on every certificate use**
  - **maybe ok for small closed communities**

© Ravi Sandhu 1999
25

REVOCATION TIME-LINE



© Ravi Sandhu 1999
26

OCSP

ON-LINE CERTIFICATE STATUS PROTOCOL

- **Alternative to CRLs**
- **consult authoritative server**

© Ravi Sandhu 1999

27

SHORT-LIVED CERTIFICATES

- **Authorization certificates can be short lived**
  - **minutes, hours, days instead of**
  - **months, years**

© Ravi Sandhu 1999

28

X.509 CRL EXTENSIONS

- General Extensions
- CRL distribution points
- Delta-CRLs
- Indirect-CRLs
- Certificate Suspension

© Ravi Sandhu 1999
29

GENERAL EXTENSIONS

- Reason Code
  - Key Compromise
  - CA Compromise
  - Affiliation changed
  - Superseded
  - Cessation of operation
  - Remove from CRL: defer till Delta-CRL
  - Certificate hold: defer
- Invalidity Date

© Ravi Sandhu 1999
30

CRL DISTRIBUTION POINTS

- **CRLs can get very big**
  - **version 1 CRL (1988, 1993)**
    - **each CA has two CRLs: one for end users, one for CAs**
    - **end user CRL can still be very big**
  - **version 2 CRL**
    - **can partition certificates, each partition associated with one CRL**
    - **distribution point**
    - **also can have different distribution points for different revocation reasons**

© Ravi Sandhu 1999
31

CRL DISTRIBUTION POINTS

- **certificate extension field, says where to look**
- **CRL extension field**
  - **distribution point for this CRL and limits on scope and reason of revocation**
  - **protects against substitution of a CRL from one distribution point to another**

© Ravi Sandhu 1999
32

DELTA-CRLs

- **Delta CRL indicator**
  - **only carries changes from previous CRL**
- **Remove from CRL reason code causes purge from base CRL (stored at certificate user)**
- **removal due to expiry of validity period or restoration of suspension**

© Ravi Sandhu 1999
33

INDIRECT-CRL

- **CRL can be issued by different CA than issuer of certificate**
  - **allows all compromise revocations to be one list**
  - **allows all CA revocations to be on one list (simplify certificate chasing)**

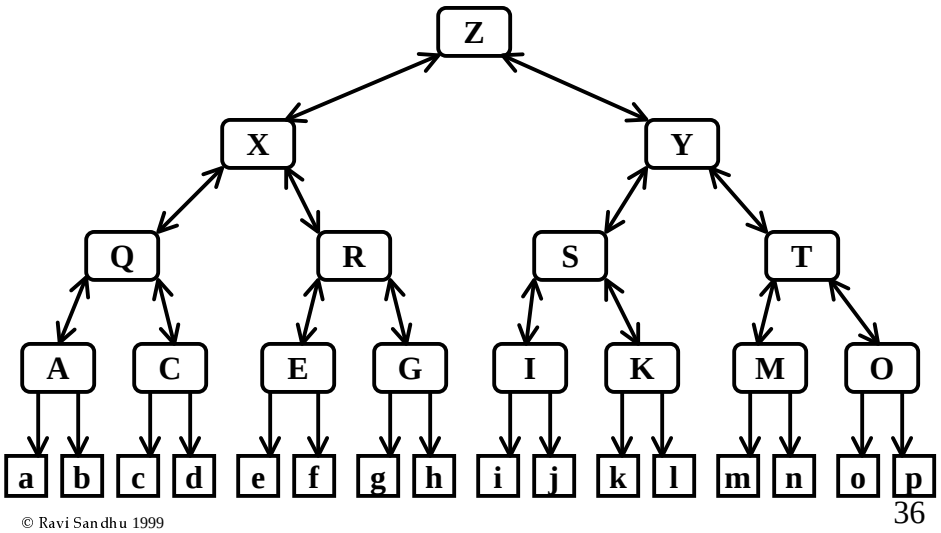
© Ravi Sandhu 1999
34

CERTIFICATE SUSPENSION

- **Certificate hold reason code in CRL**
- **Supporting CRL entry extension**
  - **Instruction code: instructions on what to do with held certificate**
    - **call CA, repossess token**

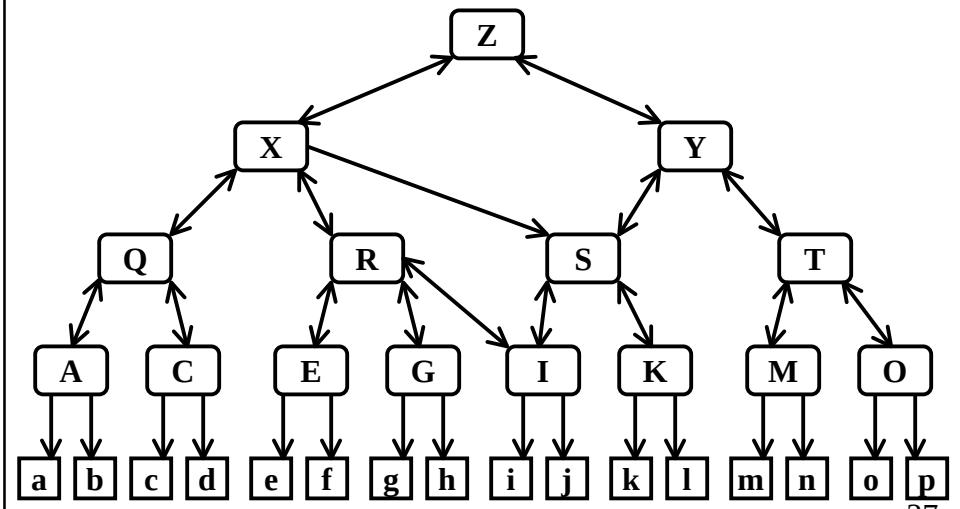
© Ravi Sandhu 1999
35

GENERAL HIERARCHICAL STRUCTURE



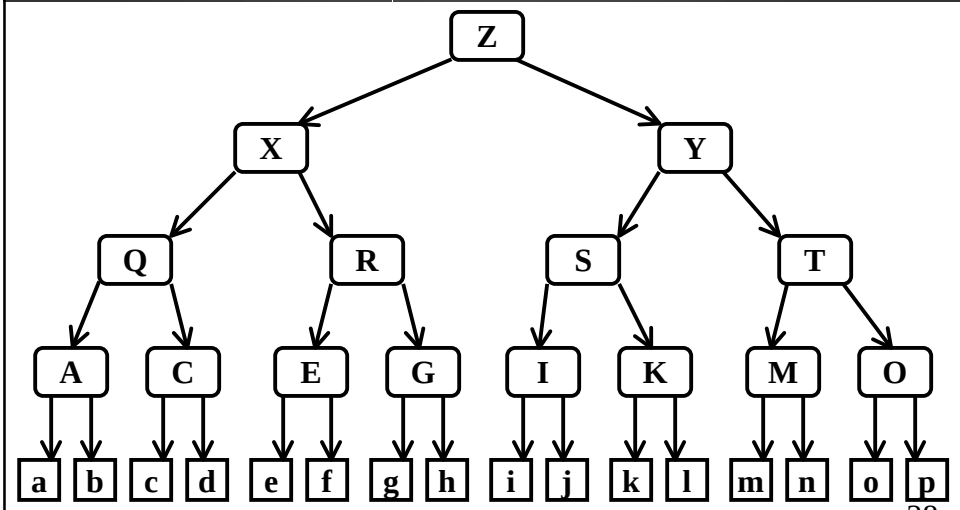
© Ravi Sandhu 1999
36

GENERAL HIERARCHICAL STRUCTURE WITH ADDED LINKS



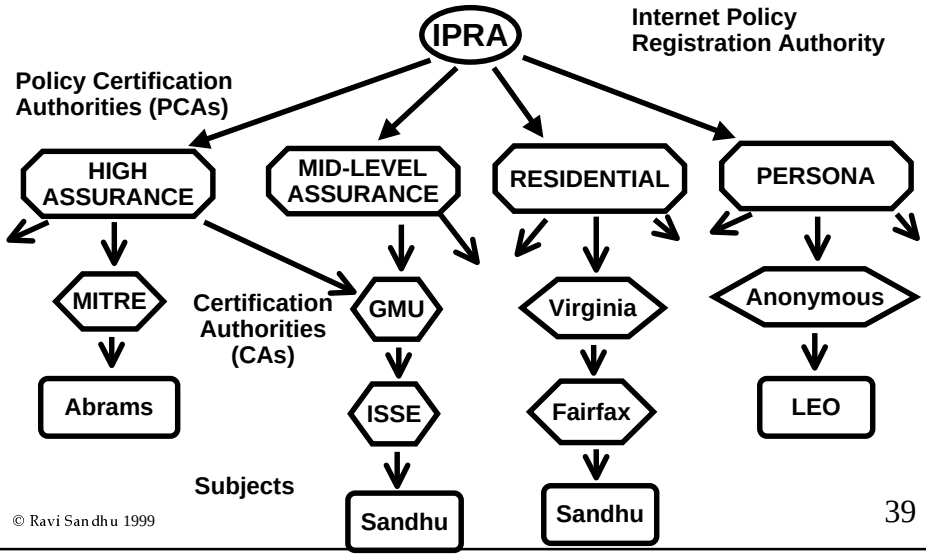
© Ravi Sandhu 1999
37

TOP-DOWN HIERARCHICAL STRUCTURE



© Ravi Sandhu 1999
38

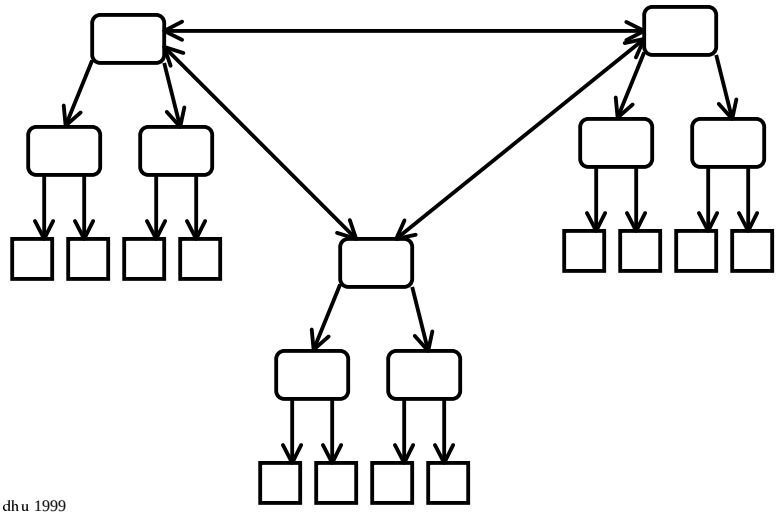
PEM CERTIFICATION GRAPH



© Ravi Sandhu 1999

39

FOREST OF HIERARCHIES



© Ravi Sandhu 1999

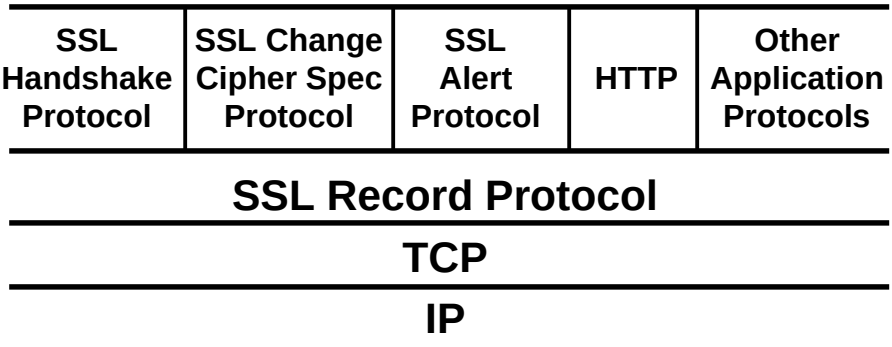
40

TRANSPORT LAYER SECURITY (TLS)

- based on Netscape's SSL (secure sockets layer)
  - SSL versions 1.0, 2.0, 3.0, 3.1
  - TLS 1.0 very close to SSL 3.1
- layered on top of TCP

© Ravi Sandhu 1999
41

SSL ARCHITECTURE



© Ravi Sandhu 1999
42

SSL SERVICES

- peer entity authentication
- data confidentiality
- data authentication and integrity
- compression/decompression
- generation/distribution of session keys
  - integrated into protocol
- security parameter negotiation

© Ravi Sandhu 1999
43

SSL KEY EXCHANGE ALGORITHMS

- RSA
- Fixed DH
- Ephemeral DH
- Anonymous DH
- Fortezza

© Ravi Sandhu 1999
44

SSL RECORD PROTOCOL

- 4 steps by sender (reversed by receiver)
  - Fragmentation
  - Compression
  - MAC
  - Encryption

© Ravi Sandhu 1999
45

SSL SESSION

- SSL session negotiated by handshake protocol
  - session ID
    - chosen by server
  - X.509 public-key certificate of peer
    - possibly null
  - compression algorithm
  - cipher spec
    - encryption algorithm
    - message digest algorithm
  - master secret
    - 48 byte shared secret
  - is resumable flag
    - can be used to initiate new connections

© Ravi Sandhu 1999
46

SSL CONNECTION

- Every connection is associated with one session
- Session can be reused across multiple secure connections
- Handshake protocol
  - establishes new session and connection together
  - uses existing session for new connection

© Ravi Sandhu 1999
47

SSL CONNECTION STATE

- 4 parts to state
  - current read state
  - current write state
  - pending read state
  - pending write state
- handshake protocol
  - initially current state is empty
  - either pending state can be made current and reinitialized to empty

© Ravi Sandhu 1999
48

SSL CONNECTION STATE

- connection end: client or server
- algorithms: encryption, message digest, compression
- master secret: 48 byte
- client and server random: 32 bytes each
- keys generated from master secret, client/server random
  - client_write_MAC_secret server_write_MAC_secret
  - client_write_key server_write_key
  - client_write_IV server_write_IV
- compression state
- cipher state: initially IV, subsequently next feedback block
- sequence number: starts at 0, max $2^{64}-1$

© Ravi Sandhu 1999

SSL RECORD PROTOCOL

- each SSL record contains
  - content type: 8 bits
  - protocol version number: 8 bits major, 8 bits minor
  - length: max 16K bytes
  - data payload
    - optionally compressed and encrypted
  - message authentication code (MAC)
    - MAC computed before encryption

© Ravi Sandhu 1999

SSL HANDSHAKE PROTOCOL

- initially SSL session has null compression and encryption algorithms
- both are set by the handshake protocol at beginning of session
- handshake protocol may be repeated during the session

© Ravi Sandhu 1999

51

SSL HANDSHAKE PROTOCOL

```
Client                                              Server

ClientHello                  -------->
                                                ServerHello
                                               Certificate*
                                         ServerKeyExchange*
                                        CertificateRequest*
                             <--------      ServerHelloDone
Certificate*
ClientKeyExchange
CertificateVerify*
[ChangeCipherSpec]
Finished                     -------->
                                          [ChangeCipherSpec]
                             <--------             Finished
Application Data             <------->     Application Data
```

Fig. 1 - Message flow for a full handshake

* Indicates optional or situation-dependent messages that are not always sent.

© Ravi Sandhu 1999

52

SSL HANDSHAKE PROTOCOL

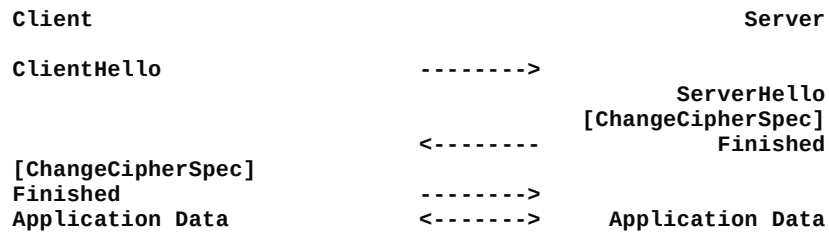


Fig. 2 - Message flow for an abbreviated handshake

© Ravi Sandhu 1999

SSL HANDSHAKE PROTOCOL

- **client hello**
  - **4 byte timestamp, 28 byte random value**
  - **session ID: if reuse existing session**
  - **cipher_suite list: ordered list**
  - **compression list: ordered list**
  - **client version: highest version**
- **server hello**
  - **32 byte random value**
  - **session ID: new or reuse**
  - **cipher_suite, compression, version: select one each**

© Ravi Sandhu 1999

SSL HANDSHAKE PROTOCOL: MASTER SECRET

```
master_secret = PRF(pre_master_secret, "master secret",
                    ClientHello.random + ServerHello.random)
[0..47];

pre_master_secret: 48 bytes
```

© Ravi Sandhu 1999
55

SSL HANDSHAKE AUTHENTICATION MODES

- **authentication of both parties**
- **server authentication with unauthenticated client**
- **total anonymity**

© Ravi Sandhu 1999
56

SSL HANDSHAKE
ANONYMOUS KEY EXCHANGE

- **RSA**
  - **client uses server's uncertified RSA public key (from key exchange message) to encrypt pre_master_secret and sends to server**
- **DH**
  - **DH public keys are exchanged in key exchange messages and both parties compute pre_master_secret**

© Ravi Sandhu 1999
57

SSL HANDSHAKE
RSA KEY EXCHANGE & AUTHENTICATION

- **RSA public key is**
  - **in server certificate or**
  - **temporary key in key exchange message signed by server's private key**
- **client encrypts pre_master_secret and sends to server**
- **for client authentication**
  - **certificate verify message includes client signature or prior handshake messages**

© Ravi Sandhu 1999
58

SSL HANDSHAKE
DH KEY EXCHANGE & AUTHENTICATION

- **server has fixed DH certificate**
  - **if client has fixed DH certificate then pre_master_secret computed from it**
  - **otherwise client sends temporary DH parameters (possibly authenticated by client signature)**
- **server uses temporary DH key signed by itself (hashed with client/server random)**
  - **as above**

© Ravi Sandhu 1999
59

SSL HANDSHAKE
CHANGE CIPHER SPEC PROTOCOL

- **1 byte message protected by current state**
- **copies pending state to current state**
  - **sender copies write pending state to write current state**
  - **receiver copies read pending state to read current state**
- **immediately send finished message under new current state**

© Ravi Sandhu 1999
60

SSL HANDSHAKE PROTOCOL: FINISHED MESSAGE

```
verify_data
    PRF(master_secret, finished_label, MD5(handshake_messages)+
    SHA-1(handshake_messages)) [0..11];

finished_label
    For Finished messages sent by the client, the string "client
    finished". For Finished messages sent by the server, the
    string "server finished".

handshake_messages
    All of the data from all handshake messages up to but not
    including this message. This is only data visible at the
    handshake layer and does not include record layer headers.
```

© Ravi Sandhu 1999
61

SSL ALERT MESSAGES

```
Warning or fatal

          close_notify(0),
          unexpected_message(10),
          bad_record_mac(20),
          decryption_failed(21),
          record_overflow(22),
          decompression_failure(30),
          handshake_failure(40),
          bad_certificate(42),
          unsupported_certificate(43),
          certificate_revoked(44),
          certificate_expired(45),
          certificate_unknown(46),
          illegal_parameter(47),
          unknown_ca(48),
          access_denied(49),
          decode_error(50),
          decrypt_error(51),
          export_restriction(60),
          protocol_version(70),
          insufficient_security(71),
          internal_error(80),
          user_canceled(90),
          no_renegotiation(100),
```

© Ravi Sandhu 1999
62

APPLICATIONS AND SSL

- use dedicated port numbers for every application that uses SSL
  - de facto what is happening
- use normal application port and negotiate security options as part of application protocol
- negotiate use of SSL during normal TCP/IP connection establishment

© Ravi Sandhu 1999
63

APPLICATION PORTS OFFICIAL AND UNOFFICIAL

- https 443
- ssmtp 465
- snntp 563
- sldap 636
- spop3 995
- ftp-data 889
- ftps 990
- imaps 991
- telnets 992
- ircs 993

© Ravi Sandhu 1999
64