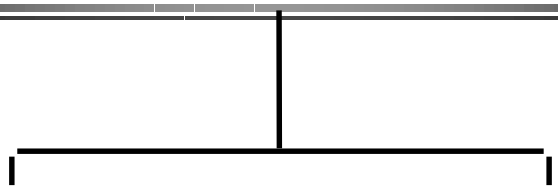
INFS 767
Secure Electronic Commerce
Fall 1999

Lecture 1
Cryptography Review

Prof. Ravi Sandhu

CRYPTOGRAPHIC TECHNOLOGY



SECRET KEY

Symmetric Key
Single Key
Conventional

PUBLIC KEY

Asymmetric Key
Two Key

© Ravi Sandhu 1999

2

CRYPTOGRAPHIC TECHNOLOGY

- Secret-key encryption
- Public-key encryption
- Public-key digital signatures
- Public-key key agreement
- Message digests
- Message authentication codes
- Public-key certificates
- Challenge-response authentication
- Secure Sockets Layer (SSL)

© Ravi Sandhu 1999
3

CRYPTOGRAPHIC SERVICES

- confidentiality
  - traffic flow confidentiality
- integrity
- authentication
- non-repudiation
- management of security

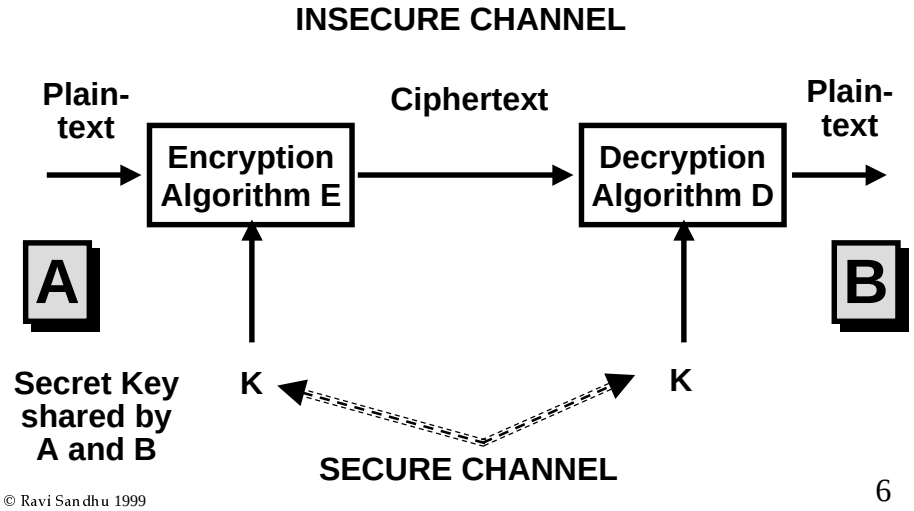
© Ravi Sandhu 1999
4

ATTACKS

- **passive attacks**
  - **observe but do not modify traffic**
  - **threat for confidentiality**
- **active attacks**
  - **delete, add, and replay traffic**
  - **threat for confidentiality, integrity, authentication and non-repudiation**

© Ravi Sandhu 1999
5

SECRET KEY CRYPTOSYSTEM



© Ravi Sandhu 1999
6

SECRET KEY CRYPTOSYSTEM

- confidentiality depends only on secrecy of the key
- attacker is assumed to know E and D
- secret key systems do not scale well
  - with N parties we need to generate and distribute N*(N-1)/2 keys
- A and B can be people or computers

© Ravi Sandhu 1999
7

SECRET KEY CRYPTOSYSTEM

- long-term keys
  - prolonged use increases exposure
- session keys
  - short-term keys communicated by means of
    - long-term secret keys
    - public key technology

© Ravi Sandhu 1999
8

SECRET KEY CRYPTOSYSTEMS

- 64 bit data block size
  - DES: 56 bit key
  - Triple DES: 112 bit key
  - IDEA: 128 bit key
  - RC2: variable size key: 1 byte to 128 bytes
  - many others

© Ravi Sandhu 1999
9

SECRET KEY CRYPTOSYSTEMS

- variable block size
  - RC5
    - 32, 64 or 128 block size
    - variable key size
    - variable number of rounds
  - new Advanced Encryption Standard under development
    - must support key-block combinations of 128-128, 192-128, 256-128
    - may support other combinations

© Ravi Sandhu 1999
10

CRYPTANALYSIS

- **ciphertext only**
  - **cryptanalyst only knows ciphertext**
- **known plaintext**
  - **cryptanalyst knows some plaintext-ciphertext pairs**
- **chosen plaintext**
- **chosen ciphertext**

© Ravi Sandhu 1999
11

KNOWN PLAINTEXT ATTACK

- **40 bit key requires $2^{39} \approx 5 * 10^{11}$ trials on average (exportable from USA)**
- **trials/second — time required**

trials/second	time required
1	20,000 years
10^3	20 years
10^6	6 days
10^9	9 minutes
10^{12}	0.5 seconds

© Ravi Sandhu 1999
12

KNOWN PLAINTEXT ATTACK

- 56 bit key requires $2^{55} \approx 3.6 * 10^{\wedge 16}$ trials on average (DES)
- trials/second — time required

trials/second	time required
1	10^9 years
10^3	10^6 years
10^6	10^3 years
10^9	1 year
10^{12}	10 hours

© Ravi Sandhu 1999
13

KNOWN PLAINTEXT ATTACK

- 128 bit key requires $2^{127} \approx 2 * 10^{38}$ trials on average (IDEA)
- trials/second — time required

trials/second	time required
1	10^{30} years
10^3	10^{27} years
10^6	10^{24} years
10^9	10^{21} years
10^{12}	10^{18} years

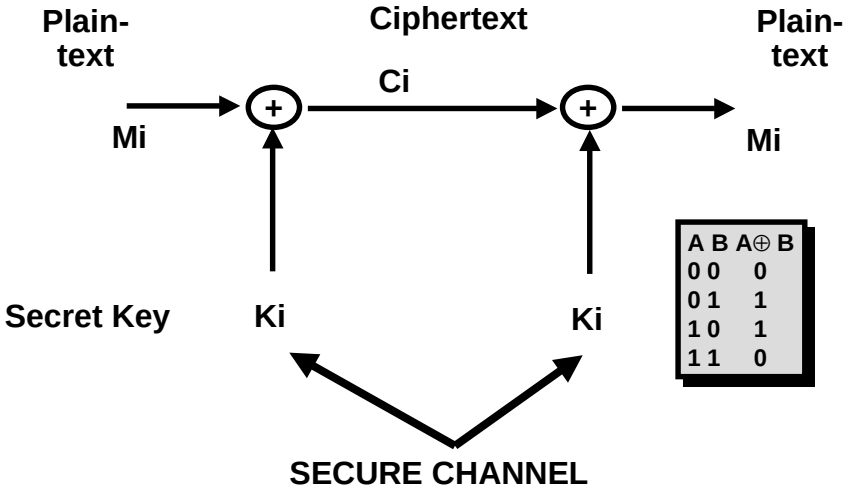
© Ravi Sandhu 1999
14

DICTIONARY ATTACKS

- if keys are poorly chosen known plaintext attacks can be very simple
- often the user's password is the key
  - in a dictionary attack the cryptanalyst tries passwords from a dictionary, rather than all possible keys
  - for a 20,000 word dictionary, 1 trial/second will crack a poor password in less than 3 hours

© Ravi Sandhu 1999
15

PERFECT SECRECY VERNAM ONE-TIME PAD



© Ravi Sandhu 1999
16

PERFECT SECRECY VERNAM ONE-TIME PAD

- known plaintext reveals the portion of the key that has been used, but does not reveal anything about the future bits of the key
- has been used
- can be approximated

© Ravi Sandhu 1999
17

DATA ENCRYPTION STANDARD (DES)

- 56 bit key
- 64 bit block size E and D are public
- US Federal standard for sensitive but unclassified information
  - adopted as ANSI DEA (Data Encryption Algorithm)
  - considered by ISO as a standard but abandoned (under US pressure)

© Ravi Sandhu 1999
18

DES

- has not been broken by sustained public cryptanalysis since 1977
  - differential and linear cryptanalysis not considered practical attacks
- major weakness is key size of 56 bits
  - 56 hours known plaintext on a special purpose supercomputer
  - couple of months on a network of workstations
- triple DES gives effective key size of 112 bits

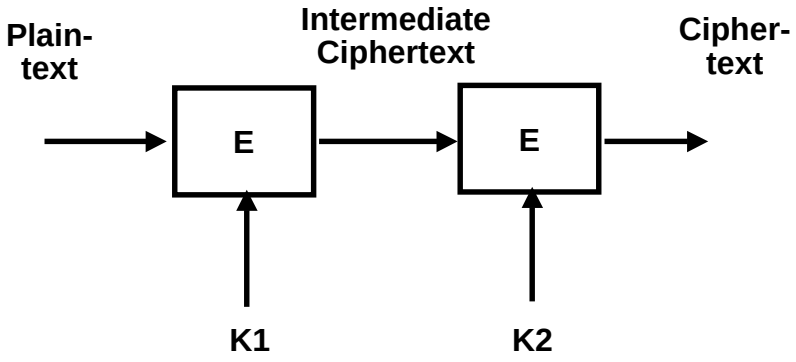
© Ravi Sandhu 1999
19

MULTIPLE ENCRYPTION

- is it useful?
  - published evidence suggests that multiple encryption with DES is effective
- how to do it?

© Ravi Sandhu 1999
20

DOUBLE DES

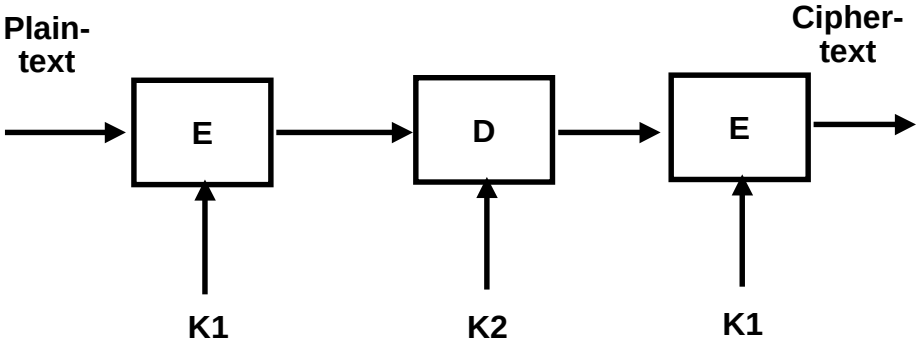


- effective key size is only 57 bits due to meet-in-the-middle attack

© Ravi Sandhu 1999

21

TRIPLE DES

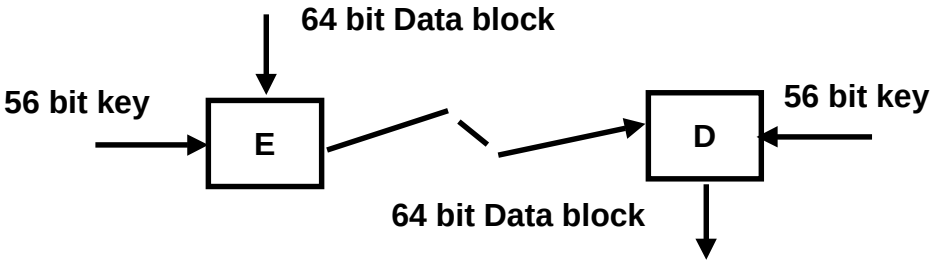


- effective key size is 112 bits due to meet-in-the-middle attack

© Ravi Sandhu 1999

22

ELECTRONIC CODE BOOK (ECB) MODE

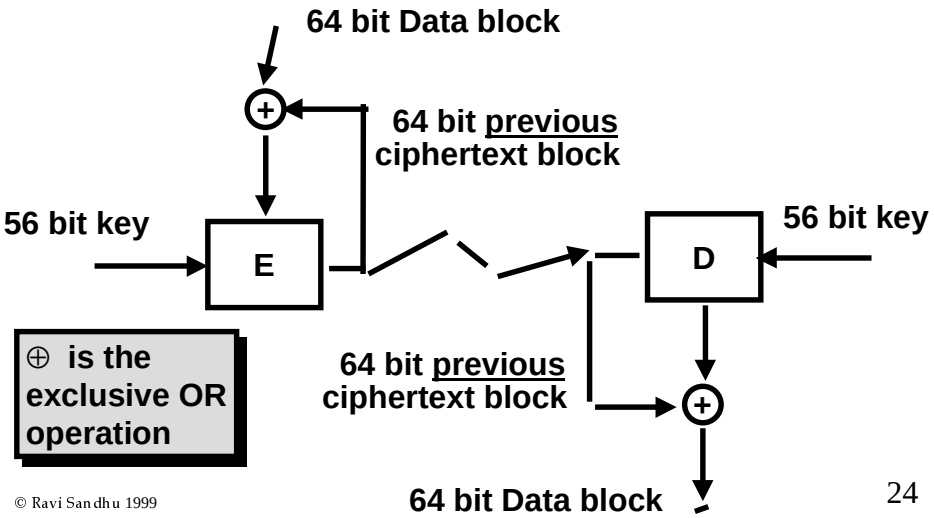


- OK for small messages
- identical data blocks will be identically encrypted

© Ravi Sandhu 1999

23

CIPHER BLOCK CHAINING (CBC) MODE



© Ravi Sandhu 1999

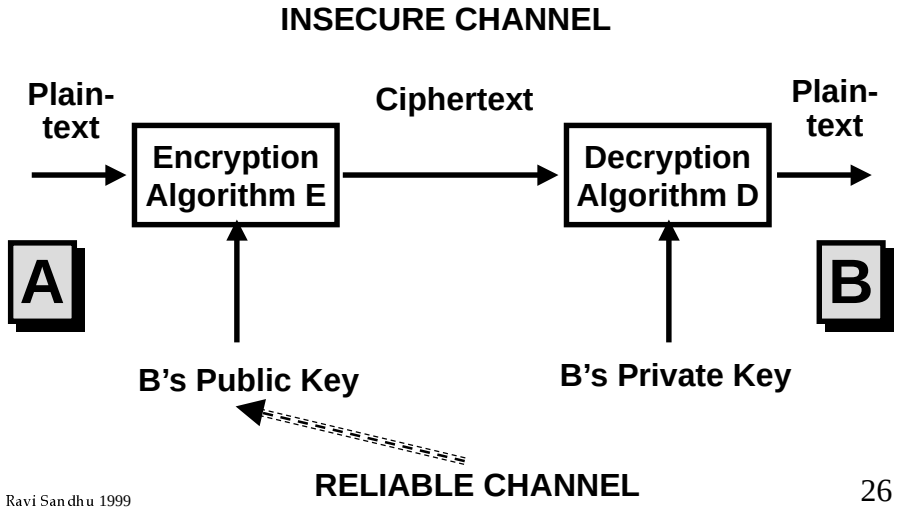
24

CIPHER BLOCK CHAINING (CBC) MODE

- Needs an Initialization Vector (IV) to serve as the first feedback block
- IV need not be secret or random
- Integrity of the IV is important, otherwise first data block can be arbitrarily changed.
- IV should be changed from message to message, or first block of every message should be distinct

© Ravi Sandhu 1999
25

PUBLIC KEY ENCRYPTION



© Ravi Sandhu 1999
26

PUBLIC KEY CRYPTOSYSTEM

- solves the key distribution problem provided there is a reliable channel for communication of public keys
- requires reliable dissemination of 1 public key/party
- scales well for large-scale systems

© Ravi Sandhu 1999
27

PUBLIC KEY ENCRYPTION

- confidentiality based on infeasibility of computing B's private key from B's public key
- key sizes are large (512 bits and above) to make this computation infeasible

© Ravi Sandhu 1999
28

RSA

- public key is (n,e)
- private key is d
- encrypt: $C = M^e \bmod n$
- decrypt: $M = C^d \bmod n$

© Ravi Sandhu 1999
29

GENERATION OF RSA KEYS

- choose 2 large (100 digit) prime numbers p and q
- compute n = p * q
- pick e relatively prime to (p-1)*(q-1)
- compute d, e*d = 1 mod (p-1)*(q-1)
- publish (n,e)
- keep d secret (and discard p, q)

© Ravi Sandhu 1999
30

PROTECTION OF RSA KEYS

- compute d, e*d = 1 mod (p-1)*(q-1)
  - if factorization of n into p*q is known, this is easy to do
- security of RSA is no better than the difficulty of factoring n into p, q

© Ravi Sandhu 1999
31

RSA VERSUS DES

- RSA encrypts at kilobits/second
- DES encrypts at megabits/second
- This 1000-fold difference in speed is likely to remain independent of technology advances

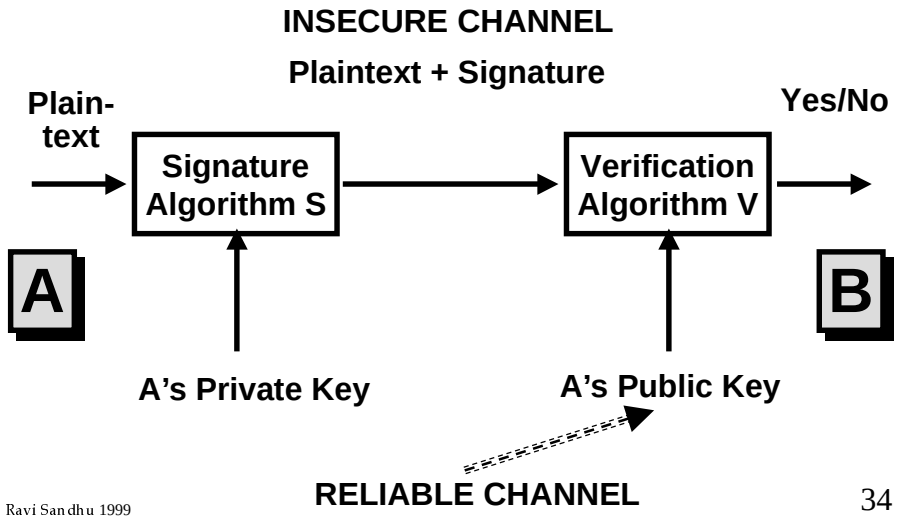
© Ravi Sandhu 1999
32

RSA KEY SIZE

- **key size of RSA is selected by the user**
  - **casual** **512 bits**
  - **"commercial"** **1024 bits**
  - **"military"** **2048 bits**

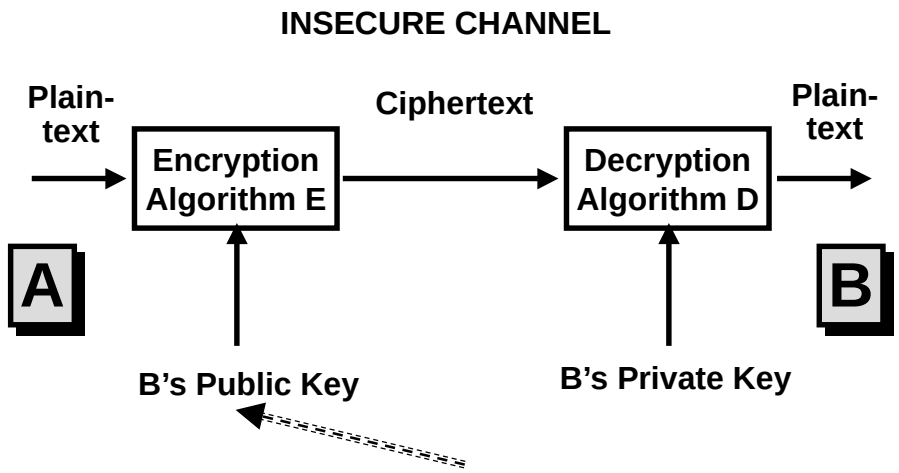
© Ravi Sandhu 1999
33

DIGITAL SIGNATURES



© Ravi Sandhu 1999
34

COMPARE PUBLIC KEY ENCRYPTION



© Ravi Sandhu 1999

35

DIGITAL SIGNATURES IN RSA

- **RSA has a unique property, not shared by other public key systems**
- **Encryption and decryption commute**
  - **$(M^e \bmod n)^d \bmod n = M$ encryption**
  - **$(M^d \bmod n)^e \bmod n = M$ signature**
- **Same public key can be use for encryption and signature**

© Ravi Sandhu 1999

36

EL GAMAL AND VARIANTS

- encryption only
- signature only
  - 1000's of variants
  - including NIST's DSA

© Ravi Sandhu 1999
37

NIST DIGITAL SIGNATURE STANDARD

- System-wide constants
  - p 512-1024 bit prime
  - q 160 bit prime divisor of p-1
  - g $g = h^{((p-1)/q)} \bmod p$, $1<h<p-1$
- El-Gamal variant
  - separate algorithms for digital signature and public-key encryption

© Ravi Sandhu 1999
38

NIST DIGITAL SIGNATURE STANDARD

- to sign message m: private key x
  - choose random r
  - compute v = (g^r mod p) mod q
  - compute s = (m+xv)/k mod q
  - signature is (s,v,m)
- to verify signature: public key y
  - compute u1 = m/s mod q
  - compute u2 = v/s mod q
  - verify that v = ($g^{u1}*y^{u2}$ mod p) mod q

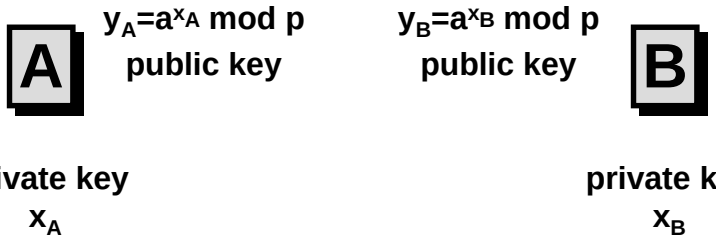
© Ravi Sandhu 1999

NIST DIGITAL SIGNATURE STANDARD

- signature does not repeat, since r will be different on each occasion
- if same random number r is used for two messages, the system is broken
- message expands by a factor of 2
- RSA signatures do repeat, and there is no message expansion

© Ravi Sandhu 1999

DIFFIE-HELLMAN KEY ESTABLISHMENT



$$k = y_B^{x_A} \bmod p = y_A^{x_B} \bmod p = a^{x_A * x_B} \bmod p$$

system constants: p: prime number, a: integer

© Ravi Sandhu 1999

DIFFIE-HELLMAN KEY ESTABLISHMENT

- security depends on difficulty of computing x given $y=a^x \bmod p$

  called the discrete logarithm problem

© Ravi Sandhu 1999

BASIC SECURITY PRINCIPLE FOR SESSION KEYS

- **compromise of a session key**
  - **does not permit reuse of the compromised session key**
  - **does not compromise**
    - **future session keys**
    - **long-term keys**

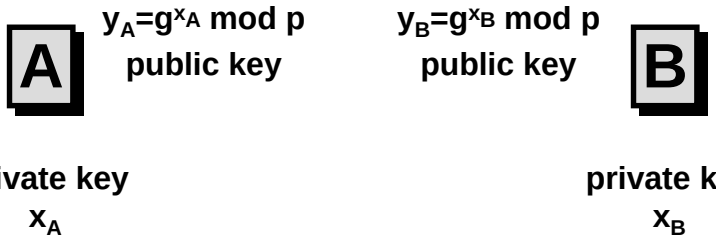
© Ravi Sandhu 1999
43

PERFECT FORWARD SECRECY

- **compromise of a long-term key does not compromise**
  - **past session keys**
- **concern for encryption keys but not for authentication keys**
- **not really "perfect" in same sense as perfect secrecy for one-time pad**

© Ravi Sandhu 1999
44

DIFFIE-HELLMAN KEY ESTABLISHMENT



$$K_{AB} = y_B^{x_A} \bmod p = y_A^{x_B} \bmod p = g^{x_A * x_B} \bmod p$$

system constants: p: prime number, g: integer

© Ravi Sandhu 1999

EPHEMERAL DIFFIE-HELLMAN

- A -> B: A's ephemeral DH public-key
- B -> A: B's ephemeral DH public-key
- session key is computed on basis of ephemeral DH public-private keys
- exchange of these ephemeral public keys requires authentication and integrity
  - digital signatures
  - message authentication codes

© Ravi Sandhu 1999

EPHEMERAL DIFFIE-HELLMAN

- heavyweight, so cannot be the only protocol in our suite
- supply of ephemeral public keys can be computed in advance
- need to discard ephemeral private keys after one-time use

© Ravi Sandhu 1999
47

ELLIPTIC CURVE CRYPTOGRAPHY

- 160 bit ECC public key is claimed to be as secure as 1024 bit RSA or DH key
- ECDSA: Elliptic Curve digital signature algorithm based on NIST Digital Signature Standard
- ECSVA: Elliptic Curve key agreement algorithm based on Diffie-Hellman
- ECES: Elliptic Curve encryption algorithm based on El-Gamal

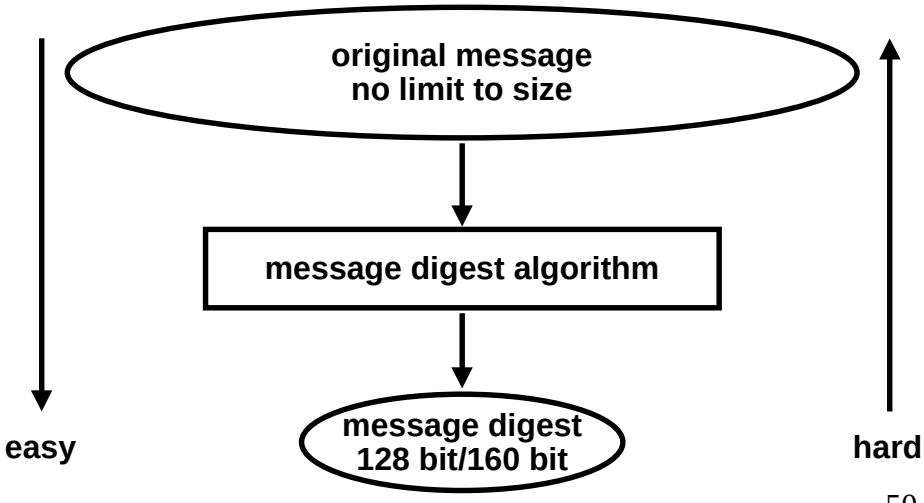
© Ravi Sandhu 1999
48

MESSAGE DIGESTS

- **Public-key technology is very slow**
- **Public-key encryption**
  - **use public-key encryption to send a secret key with confidentiality**
  - **actual traffic is encrypted using secret key**
- **Public-key digital signatures**
  - **cannot sign big messages**

© Ravi Sandhu 1999
49

MESSAGE DIGEST



© Ravi Sandhu 1999
50

MESSAGE DIGEST

- for performance reasons
  - sign the message digest
  - not the message
- one way function
  - m=H(M) is easy to compute
  - $M=H^{-1}(m)$ is hard to compute

© Ravi Sandhu 1999
51

DESIRED CHARACTERISTICS

- weak hash function
  - difficult to find M' such that H(M')=H(M)
  - try messages at random to find M' with H(M')=m
    - 2^k trials on average, k=64 to be safe
- strong hash function
  - difficult to find any two M and M' such that H(M')=H(M)
  - try pairs of messages at random to find M and M' such that H(M')=H(M)
    - $2^{k/2}$ trials on average, k=128 to be safe
    - k=160 is better

© Ravi Sandhu 1999
52

MESSAGE DIGESTS

- **MD5**
  - **proposed by Ron Rivest (of RSA)**
  - **improved version of MD4**
  - **128 bit digest**
  - **simple, compact and fast**
- **NIST SHA**
  - **160 bit digest**
  - **similar to MD5**

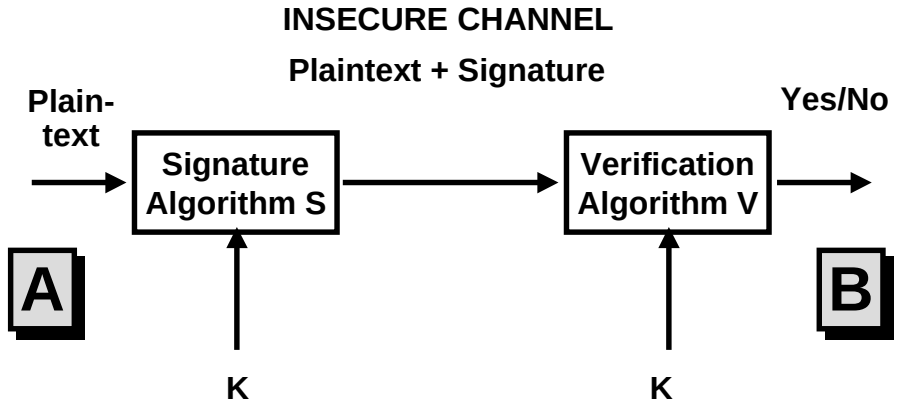
© Ravi Sandhu 1999
53

MESSAGE AUTHENTICATION CODES

- **secret-key technique to provide efficient**
  - **authentication**
  - **integrity**
- **does not provide**
  - **non-repudiation**

© Ravi Sandhu 1999
54

MESSAGE AUTHENTICATION CODES



Signature = MD of plaintext + K

© Ravi Sandhu 1999

55

HMAC

- **HMAC computation**
- **$HMAC_K(M) = h(K \oplus opad \,||\, h(K \oplus ipad \,||\, M))$**
  - **h is any message digest function**
  - **M message**
  - **K secret key**
  - **opad, ipad: fixed outer and inner padding**
- **HMAC-MD5, HMAC-SHA**

© Ravi Sandhu 1999

56

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

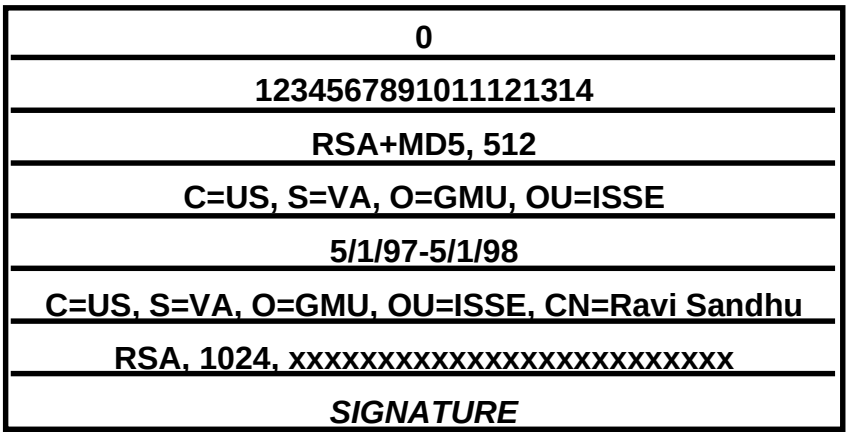
© Ravi Sandhu 1999
57

X.509 CERTIFICATE

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

© Ravi Sandhu 1999
58

X.509 CERTIFICATE



© Ravi Sandhu 1999

59

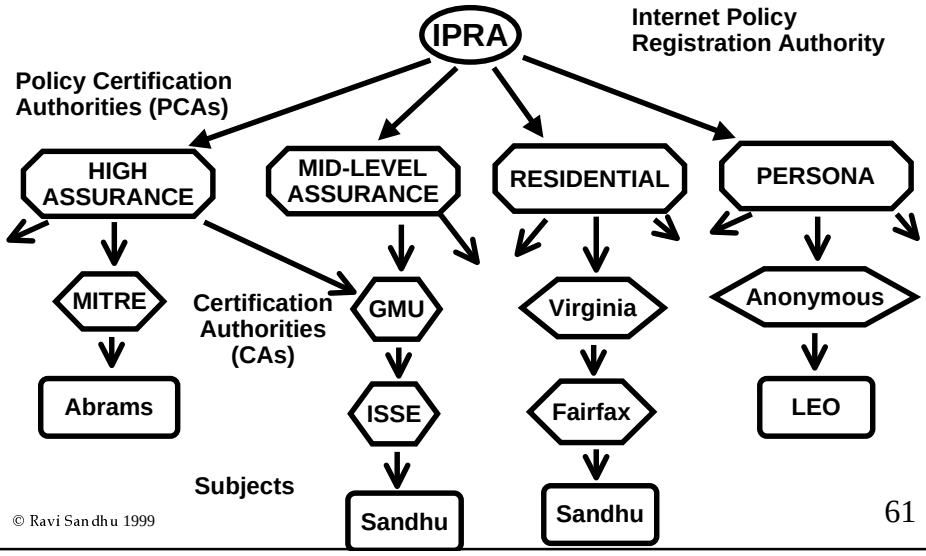
CERTIFICATE TRUST

- 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

60

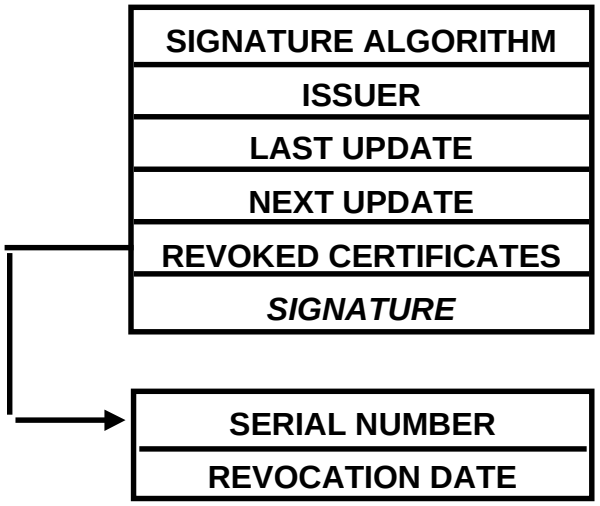
PEM CERTIFICATION GRAPH



© Ravi Sandhu 1999

61

CRL FORMAT



© Ravi Sandhu 1999

62

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
63

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
64

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
65

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
66

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
67

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
68

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
69

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
70

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
71

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
72

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
73

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
74

SUBJECT AND ISSUER ATTRIBUTES

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

© Ravi Sandhu 1999

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

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
77

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
78

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**

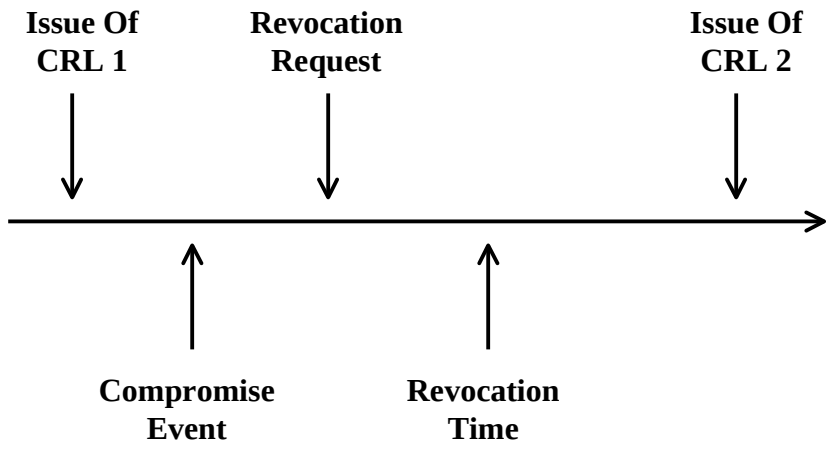
© Ravi Sandhu 1999
79

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
80

REVOCATION TIME-LINE



© Ravi Sandhu 1999

81

OCSP

ON-LINE CERTIFICATE STATUS PROTOCOL

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

© Ravi Sandhu 1999

82

SHORT-LIVED CERTIFICATES

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

© Ravi Sandhu 1999

X.509 CRL EXTENSIONS

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

© Ravi Sandhu 1999

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
85

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
86

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
87

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
88

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)

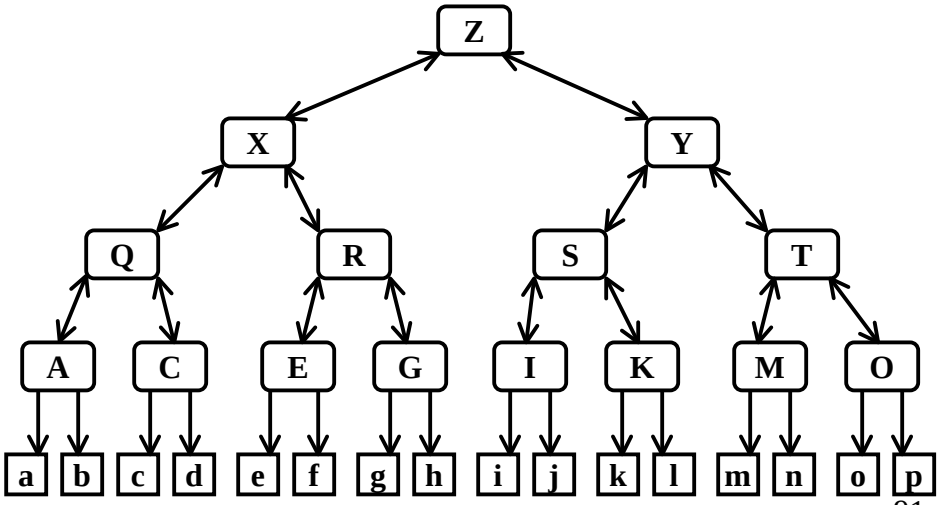
© Ravi Sandhu 1999
89

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

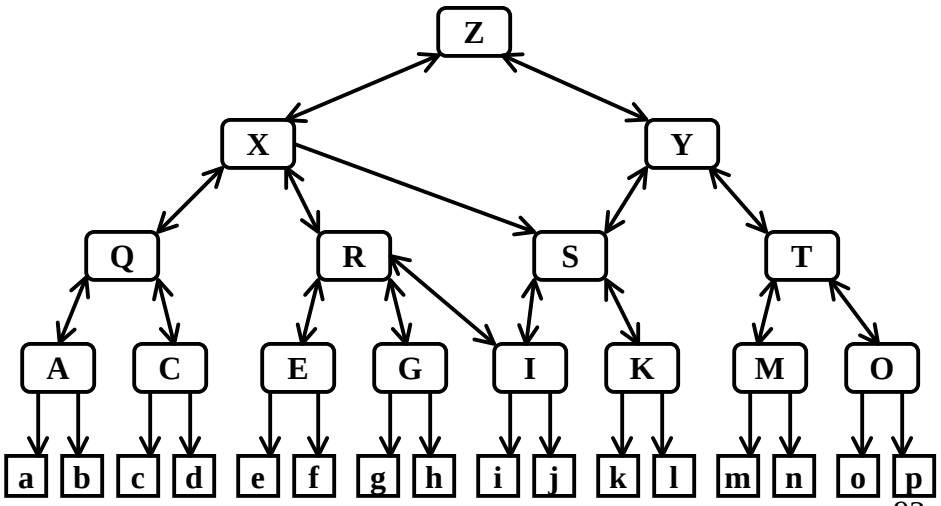
© Ravi Sandhu 1999
90

GENERAL HIERARCHICAL STRUCTURE



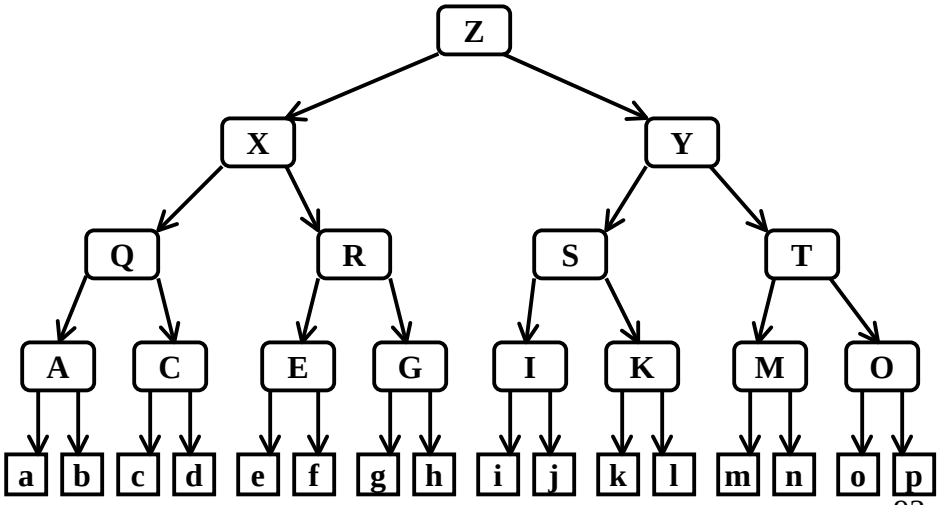
© Ravi Sandhu 1999
91

GENERAL HIERARCHICAL STRUCTURE WITH ADDED LINKS



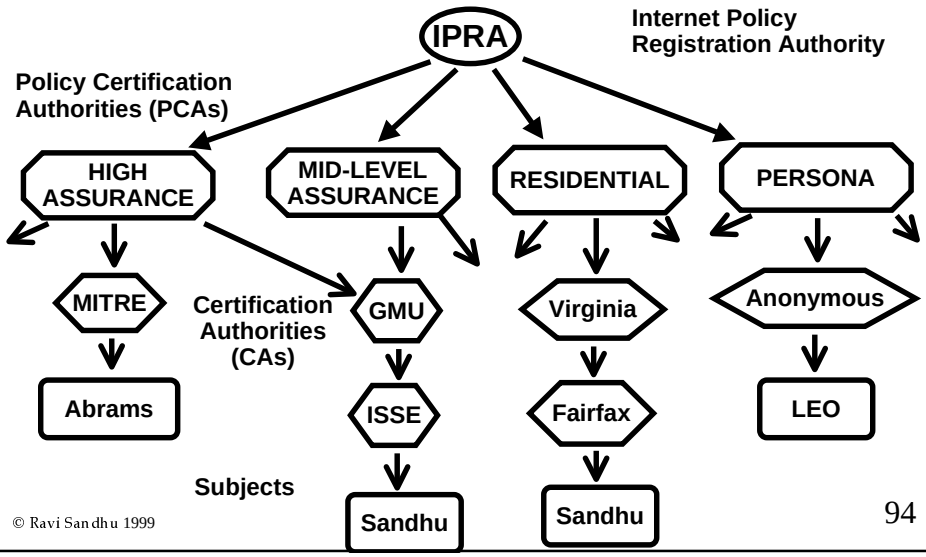
© Ravi Sandhu 1999
92

TOP-DOWN HIERARCHICAL STRUCTURE



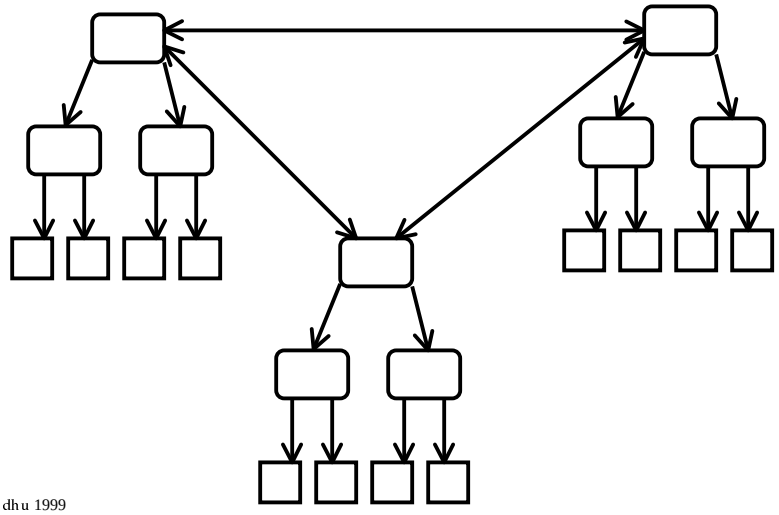
© Ravi Sandhu 1999
93

PEM CERTIFICATION GRAPH



© Ravi Sandhu 1999
94

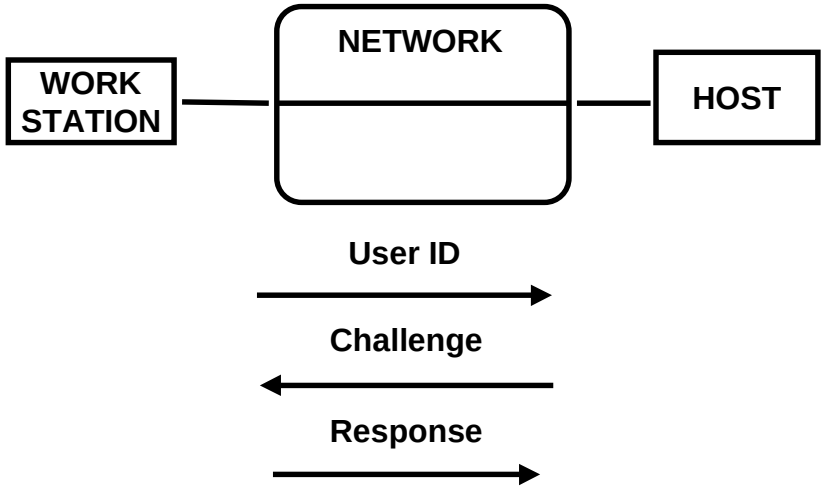
FOREST OF HIERARCHIES



© Ravi Sandhu 1999

95

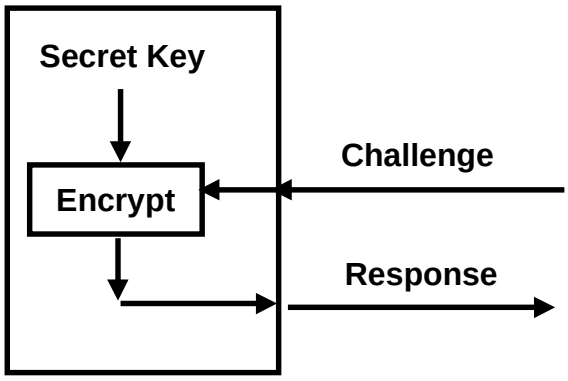
CHALLENGE RESPONSE



© Ravi Sandhu 1999

96

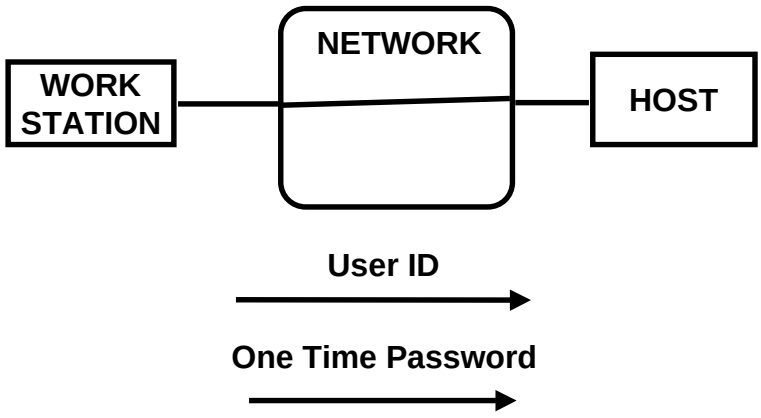
CHALLENGE RESPONSE



© Ravi Sandhu 1999

97

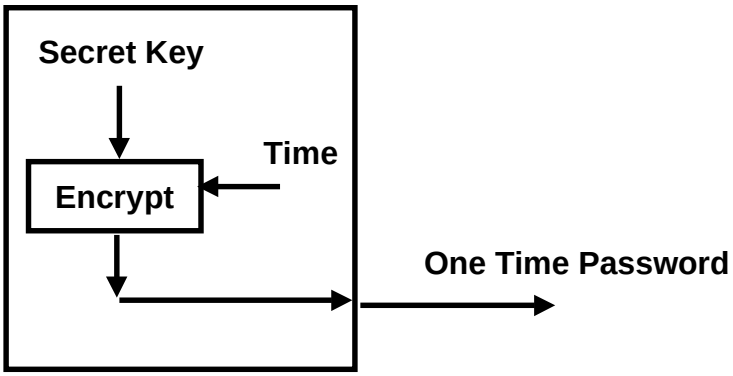
TIME SYNCHRONIZED



© Ravi Sandhu 1999

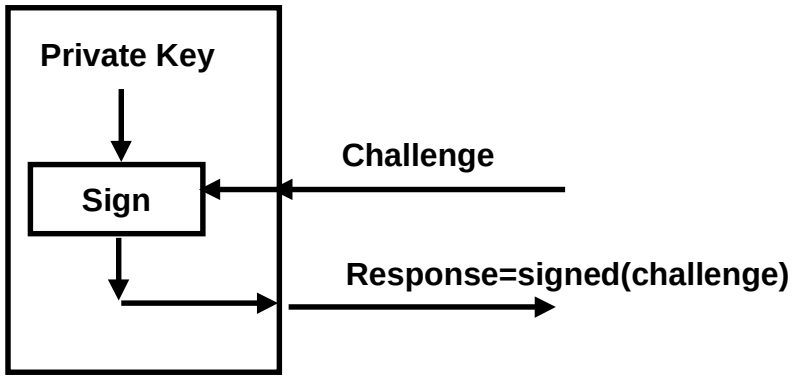
98

TIME SYNCHRONIZED



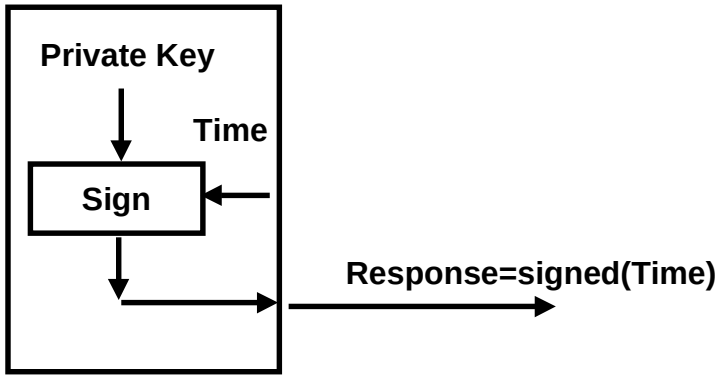
© Ravi Sandhu 1999
99

PUBLIC KEY BASED



© Ravi Sandhu 1999
100

PUBLIC KEY BASED



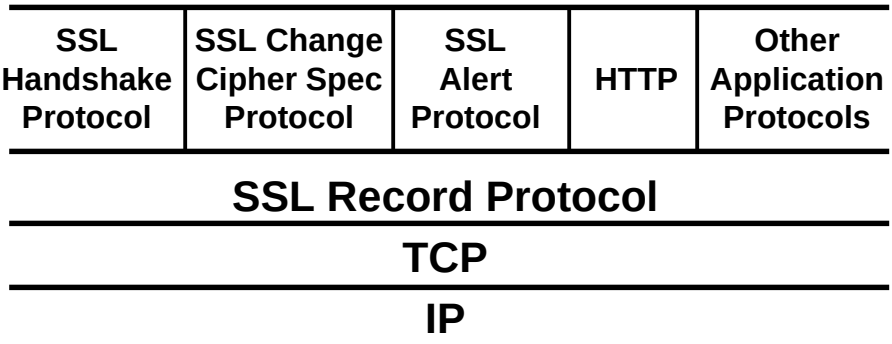
© Ravi Sandhu 1999
101

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
102

SSL ARCHITECTURE



© Ravi Sandhu 1999
103

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
104

SSL KEY EXCHANGE ALGORITHMS

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

© Ravi Sandhu 1999

SSL RECORD PROTOCOL

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

© Ravi Sandhu 1999

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
107

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
108

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
109

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
110

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
111

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
112

SSL HANDSHAKE PROTOCOL

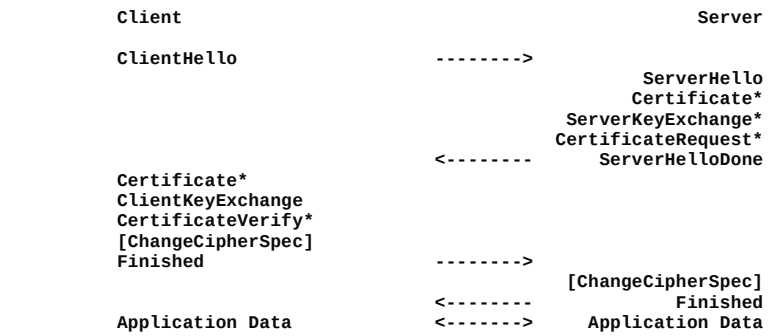


Fig. 1 - Message flow for a full handshake

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

© Ravi Sandhu 1999

SSL HANDSHAKE PROTOCOL

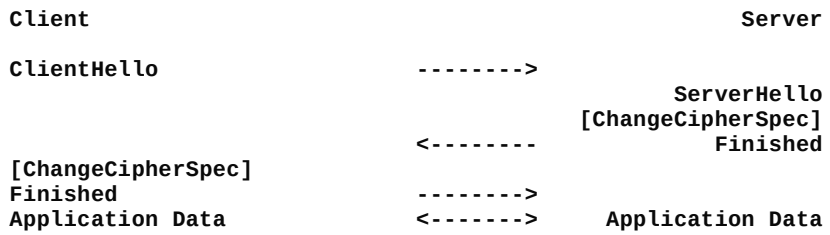


Fig. 2 - Message flow for an abbreviated handshake

© 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
115

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
116

SSL HANDSHAKE AUTHENTICATION MODES

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

© Ravi Sandhu 1999
117

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
118

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
119

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
120

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
121

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
122

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

123

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

124

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

125