

# Cryptography and Network Security

Third Edition

by William Stallings

Lecture slides by Lawrie Brown

# Chapter 13 –Digital Signatures & Authentication Protocols

*To guard against the baneful influence exerted by strangers is therefore an elementary dictate of savage prudence. Hence before strangers are allowed to enter a district, or at least before they are permitted to mingle freely with the inhabitants, certain ceremonies are often performed by the natives of the country for the purpose of disarming the strangers of their magical powers, or of disinfecting, so to speak, the tainted atmosphere by which they are supposed to be surrounded.*

**—The Golden Bough, Sir James George Frazer**

# Digital Signatures

- have looked at message authentication
  - but does not address issues of lack of trust
- digital signatures provide the ability to:
  - verify author, date & time of signature
  - authenticate message contents
  - be verified by third parties to resolve disputes
- hence include authentication function with additional capabilities

# Digital Signature Properties

- must depend on the message signed
- must use information unique to sender
  - to prevent both forgery and denial
- must be relatively easy to produce
- must be relatively easy to recognize & verify
- be computationally infeasible to forge
  - with new message for existing digital signature
  - with fraudulent digital signature for given message
- be practical save digital signature in storage

# Direct Digital Signatures

- involve only sender & receiver
- assumed receiver has sender's public-key
- digital signature made by sender signing entire message or hash with private-key
- can encrypt using receivers public-key
- important that sign first then encrypt message & signature
- security depends on sender's private-key

# Arbitrated Digital Signatures

- involves use of arbiter A
  - validates any signed message
  - then dated and sent to recipient
- requires suitable level of trust in arbiter
- can be implemented with either private or public-key algorithms
- arbiter may or may not see message

# Authentication Protocols

- used to convince parties of each others identity and to exchange session keys
- may be one-way or mutual
- key issues are
  - confidentiality – to protect session keys
  - timeliness – to prevent replay attacks

# Replay Attacks

- where a valid signed message is copied and later resent
  - simple replay
  - repetition that can be logged
  - repetition that cannot be detected
  - backward replay without modification
- countermeasures include
  - use of sequence numbers (generally impractical)
  - timestamps (needs synchronized clocks)
  - challenge/response (using unique nonce)



# Using Symmetric Encryption

- as discussed previously can use a two-level hierarchy of keys
- usually with a trusted Key Distribution Center (KDC)
  - each party shares own master key with KDC
  - KDC generates session keys used for connections between parties
  - master keys used to distribute these to them

# Needham-Schroeder Protocol

- original third-party key distribution protocol
- for session between A B mediated by KDC
- protocol overview is:
  1.  $A \rightarrow KDC: ID_A \parallel ID_B \parallel N_1$
  2.  $KDC \rightarrow A: E_{K_a}[K_s \parallel ID_B \parallel N_1 \parallel E_{K_b}[K_s \parallel ID_A]]$
  3.  $A \rightarrow B: E_{K_b}[K_s \parallel ID_A]$
  4.  $B \rightarrow A: E_{K_s}[N_2]$
  5.  $A \rightarrow B: E_{K_s}[f(N_2)]$

# Needham-Schroeder Protocol

- used to securely distribute a new session key for communications between A & B
- but is vulnerable to a replay attack if an old session key has been compromised
  - then message 3 can be resent convincing B that is communicating with A
- modifications to address this require:
  - timestamps (Denning 81)
  - using an extra nonce (Neuman 93)

# Using Public-Key Encryption

- have a range of approaches based on the use of public-key encryption
- need to ensure have correct public keys for other parties
- using a central Authentication Server (AS)
- various protocols exist using timestamps or nonces

# Denning AS Protocol

- Denning 81 presented the following:
  1.  $A \rightarrow AS: ID_A \parallel ID_B$
  2.  $AS \rightarrow A: E_{KRas}[ID_A \parallel KU_a \parallel T] \parallel E_{KRas}[ID_B \parallel KU_b \parallel T]$
  3.  $A \rightarrow B: E_{KRas}[ID_A \parallel KU_a \parallel T] \parallel E_{KRas}[ID_B \parallel KU_b \parallel T] \parallel E_{KU_b}[E_{KRas}[K_s \parallel T]]$
- note session key is chosen by A, hence AS need not be trusted to protect it
- timestamps prevent replay but require synchronized clocks

# One-Way Authentication

- required when sender & receiver are not in communications at same time (eg. email)
- have header in clear so can be delivered by email system
- may want contents of body protected & sender authenticated

# Using Symmetric Encryption

- can refine use of KDC but can't have final exchange of nonces, vis:
  1.  $A \rightarrow \text{KDC}: ID_A \parallel ID_B \parallel N_1$
  2.  $\text{KDC} \rightarrow A: E_{K_a}[K_s \parallel ID_B \parallel N_1 \parallel E_{K_b}[K_s \parallel ID_A]]$
  3.  $A \rightarrow B: E_{K_b}[K_s \parallel ID_A] \parallel E_{K_s}[M]$
- does not protect against replays
  - could rely on timestamp in message, though email delays make this problematic

# Public-Key Approaches

- have seen some public-key approaches
- if confidentiality is major concern, can use:  
     $A \rightarrow B: E_{K_{Ub}}[K_s] \parallel E_{K_s}[M]$   
    – has encrypted session key, encrypted message
- if authentication needed use a digital signature with a digital certificate:  
     $A \rightarrow B: M \parallel E_{K_{Ra}}[H(M)] \parallel E_{K_{Ra_s}}[T \parallel ID_A \parallel KU_a]$   
    – with message, signature, certificate



# Digital Signature Standard (DSS)

- US Govt approved signature scheme FIPS 186
- uses the SHA hash algorithm
- designed by NIST & NSA in early 90's
- DSS is the standard, DSA is the algorithm
- a variant on ElGamal and Schnorr schemes
- creates a 320 bit signature, but with 512-1024 bit security
- security depends on difficulty of computing discrete logarithms

# DSA Key Generation

- have shared global public key values  $(p, q, g)$ :
  - a large prime  $p = 2^L$ 
    - where  $L = 512$  to  $1024$  bits and is a multiple of  $64$
  - choose  $q$ , a  $160$  bit prime factor of  $p-1$
  - choose  $g = h^{(p-1)/q}$ 
    - where  $h < p-1$ ,  $h^{(p-1)/q} \pmod{p} > 1$
- users choose private & compute public key:
  - choose  $x < q$
  - compute  $y = g^x \pmod{p}$

# DSA Signature Creation

- to **sign** a message  $M$  the sender:
  - generates a random signature key  $k$ ,  $k < q$
  - nb.  $k$  must be random, be destroyed after use, and never be reused
- then computes signature pair:
$$r = (g^k \pmod{p}) \pmod{q}$$
$$s = (k^{-1} \cdot \text{SHA}(M) + x \cdot r) \pmod{q}$$
- sends signature  $(r, s)$  with message  $M$

# DSA Signature Verification

- having received M & signature  $(r, s)$
- to **verify** a signature, recipient computes:  
$$w = s^{-1} \pmod{q}$$
$$u1 = (\text{SHA}(M) \cdot w) \pmod{q}$$
$$u2 = (r \cdot w) \pmod{q}$$
$$v = (g^{u1} \cdot y^{u2} \pmod{p}) \pmod{q}$$
- if  $v=r$  then signature is verified
- see book web site for details of proof why

# Summary

- have considered:
  - digital signatures
  - authentication protocols (mutual & one-way)
  - digital signature standard