Information Systems Security

Dr. Ayman Abdel-Hamid

College of Computing and Information Technology

Arab Academy for Science & Technology and Maritime Transport

Digital Signatures and Authentication Protocols

Outline

- Digital Signatures
  - Direct
  - Arbitrated
- Authentication Techniques
  - Mutual Authentication
  - One-way Authentication

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 Requirements

- 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 to 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 receiver's public-key
- important that sign first then encrypt message & signature
- security depends on sender's private-key

Arbitrated Digital Signatures 1/2

- 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

Arbitrated Digital Signatures 2/2

(a) Conventional Encryption, Arbiter Sees Message
(1) $X \rightarrow A: \; M \| E_{K_{xa}}[ID_X \| H(M)]$
(2) $A \rightarrow Y: \; E_{K_{ay}}[ID_X \| M \| E_{K_{xa}}[ID_X \| H(M)] \| T]$
(b) Conventional Encryption, Arbiter Does Not See Message
(1) $X \rightarrow A: \; ID_X \| E_{K_{xy}}[M] \| E_{K_{xa}}[ID_X \| H(E_{K_{xy}}[M])]$
(2) $A \rightarrow Y: \; E_{K_{ay}}[ID_X \| E_{K_{xy}}[M] \| E_{K_{xa}}[ID_X \| H(E_{K_{xy}}[M])] \| T]$
(c) Public-Key Encryption, Arbiter Does Not See Message
(1) $X \rightarrow A: \; ID_X \| E_{KR_x}[ID_X \| E_{KU_y}(E_{KR_x}[M])]$
(2) $A \rightarrow Y: \; E_{KR_a}[ID_X \| E_{KU_y}[E_{KR_x}[M]] \| T]$

Notation:

X = sender
Y = recipient
A = Arbiter

M = message
T = timestamp

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 1/2

- where a valid signed message is copied and later resent
  - simple replay
    - Opponent copies a message and replays it later
  - repetition that can be logged
    - Replay a timestamped message within valid time window
  - repetition that cannot be detected
    - Original message could have been suppressed
    - Only replay message arrives
  - backward replay without modification
    - Replay back to the sender (possible with symmetric encryption and sender does not know difference between sent and received based on content)

Replay Attacks 2/2

- countermeasures include
  - use of sequence numbers (generally impractical)
    - Keep track of last sequence number for each entity
  - timestamps (needs synchronized clocks)
    - Not very suitable for connection-oriented protocols
  - challenge/response (using unique nonce)
    - Not very suitable for connectionless protocols

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 1/5

- original third-party key distribution protocol
- for session between A & B mediated by KDC
- protocol overview is [NEED 78]:

1. A→KDC: $ID_A \| ID_B \| N_1$

2. KDC→A: $E_{Ka}[Ks \| ID_B \| N_1 \| E_{Kb}[Ks \| ID_A]]$

3. A→B: $E_{Kb}[Ks \| ID_A]$

4. B→A: $E_{Ks}[N_2]$

5. A→B: $E_{Ks}[f(N_2)]$

Needham-Schroeder Protocol 2/5

- 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 it is communicating with A
- modifications to address this require:
  - timestamps (Denning 81)
  - using an extra nonce (Neuman 93)

Needham-Schroeder Protocol 3/5

- modifications to address this require:
  - timestamps (Denning 81, Denning 82)
    1. A→KDC: $ID_A \| ID_B$
    2. KDC→A: $E_{Ka}[Ks \| ID_B \| T \| E_{Kb}[Ks \| ID_A \| T]]$
    3. A→B: $E_{Kb}[Ks \| ID_A \| T]$
    4. B→A: $E_{Ks}[N_1]$
    5. A→B: $E_{Ks}[f(N_1)]$
  - Verify timeliness if $|clock - T| < \Delta t_1 + \Delta t_2$
    - Δt_1: estimated normal discrepancy between KDC's clock and local clock at A or B
    - Δt_2: expected network delay
  - What happens if clocks become unsynchronized and the sender's clock is ahead of the intended recipient's clock? (can cause a suppress-replay attack)

Needham-Schroeder Protocol 4/5

- modifications to address this require:
  - using an extra nonce (Neuman 93)

    1. A→B : $ID_A \| N_a$

    2. B→KDC : $ID_B \| N_b \| E_{Kb}[ID_A \| N_a \| T_b]$]

    3. KDC→A: $E_{Ka}[ID_B \| N_a \| K_s \| T_b] \| E_{Kb}[ID_A \| K_s \| T_b] \| N_b$

    4. A→B: $E_{Kb}[ID_A \| K_s \| T_b] \| E_{Ks}[N_b]$

  - T_b is a suggested expiration time sent by B
  - Step 3 provides A with a ticket for future communication with B without having to go through the KDC again

Needham-Schroeder Protocol 5/5

- For Future Communication

  1. A→B: $E_{Kb}[ID_A||K_s||T_b]$, N'_a

  2. B→A: N'_b , $E_{Ks}[N'_a]$

  3. A→B: $E_{Ks}[N'_b]$

  – T_b is relative to B's clock → no synchronized clocks required

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→AS: $ID_A \| ID_B$
  2. AS→A: $E_{KRas}[ID_A \| KU_a \| T] \| E_{KRas}[ID_B \| KU_b \| T]$
  3. A→B: $E_{KRas}[ID_A \| KU_a \| T] \| E_{KRas}[ID_B \| KU_b \| T] \| E_{KUb}[E_{KRa}[K_s \| T]]$
- 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 (e.g., email)
- have header in clear so can be delivered by email system
- may want contents of body protected & sender authenticated

1-Way Auth: Using Symmetric Encryption

- can refine use of KDC but can't have final exchange of nonces:
  1. A→KDC: $ID_A \| ID_B \| N_1$
  2. KDC→A: $E_{Ka}[Ks \| ID_B \| N_1 \| E_{Kb}[Ks \| ID_A]\ ]$
  3. A→B: $E_{Kb}[Ks \| ID_A] \| E_{Ks}[M]$
- does not protect against replays
  – could rely on timestamp in message, though email delays make this problematic

1-Way Auth: Public-Key Approaches

- some public-key approaches
  - Approaches require that sender knows recipient's public key (confidentiality) or recipient knows sender's public key (authentication)
- if confidentiality is major concern, can use:

  $A \rightarrow B: E_{KUb}[Ks] \| E_{Ks}[M]$
  - Use a one time-secret key K_s. Has encrypted session key, encrypted message

Public-Key Approaches

- if authentication needed use a digital signature with a digital certificate:

  $A \rightarrow B: M \| E_{KRa}[H(M)]$ What is the problem here?

  $A \rightarrow B: E_{KUb}\ [M \| E_{KRa}[H(M)]]$ What is the problem here?

  $A \rightarrow B: M \| E_{KRa}[H(M)] \| E_{KRas}[T \| ID_A \| KU_a]$

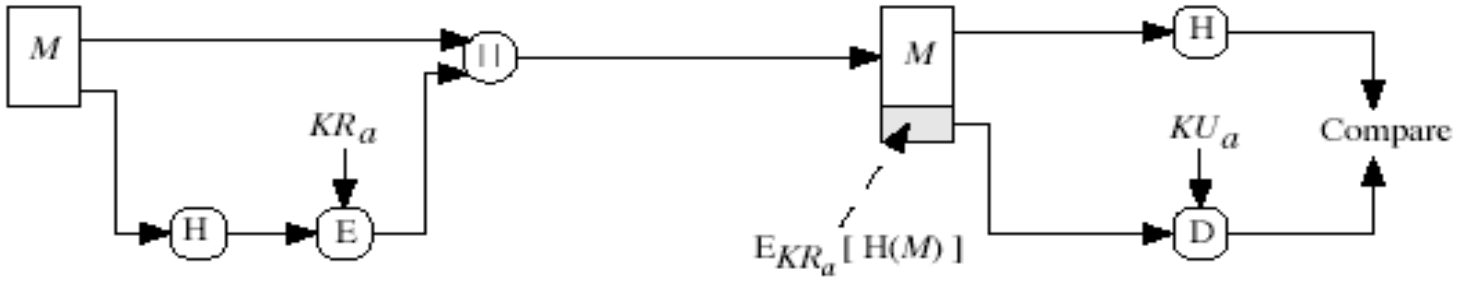
  – with message, signature, certificate

  – If confidentiality required, entire message encrypted with KU_b

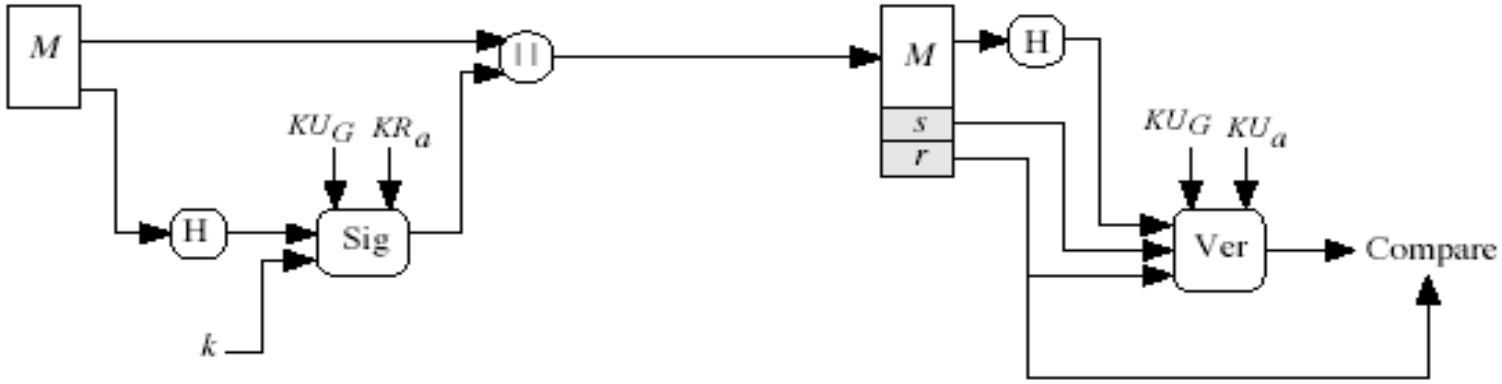
Digital Signature Standard (DSS)

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

Digital Signature Standard (DSS)



(a) RSA Approach



(b) DSS Approach

DSA Key Generation

- have shared global public values (p, q, g):
  - a large prime $2^{L-1} < p < 2^{L}$
    - 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`
  - `k` must be random, be destroyed after use, and never be reused
- then computes signature pair:

$$r = (g^k \pmod p) \pmod q$$

$$s = (k^{-1}.SHA(M) + x.r) \pmod q$$

- sends signature `(r,s)` with message `M`

DSA Signature Verification

- having received M & signature (r,s)
- to **verify** a signature, recipient computes:

  $w = s^{-1} \pmod q$

  $u1 = (SHA(M).w) \pmod q$

  $u2 = (r.w) \pmod q$

  $v = (g^{u1}.y^{u2} \pmod p) \pmod q$
- if v=r then signature is verified