Advanced Topics in Network Security

Digital Signatures (Lecture #13)

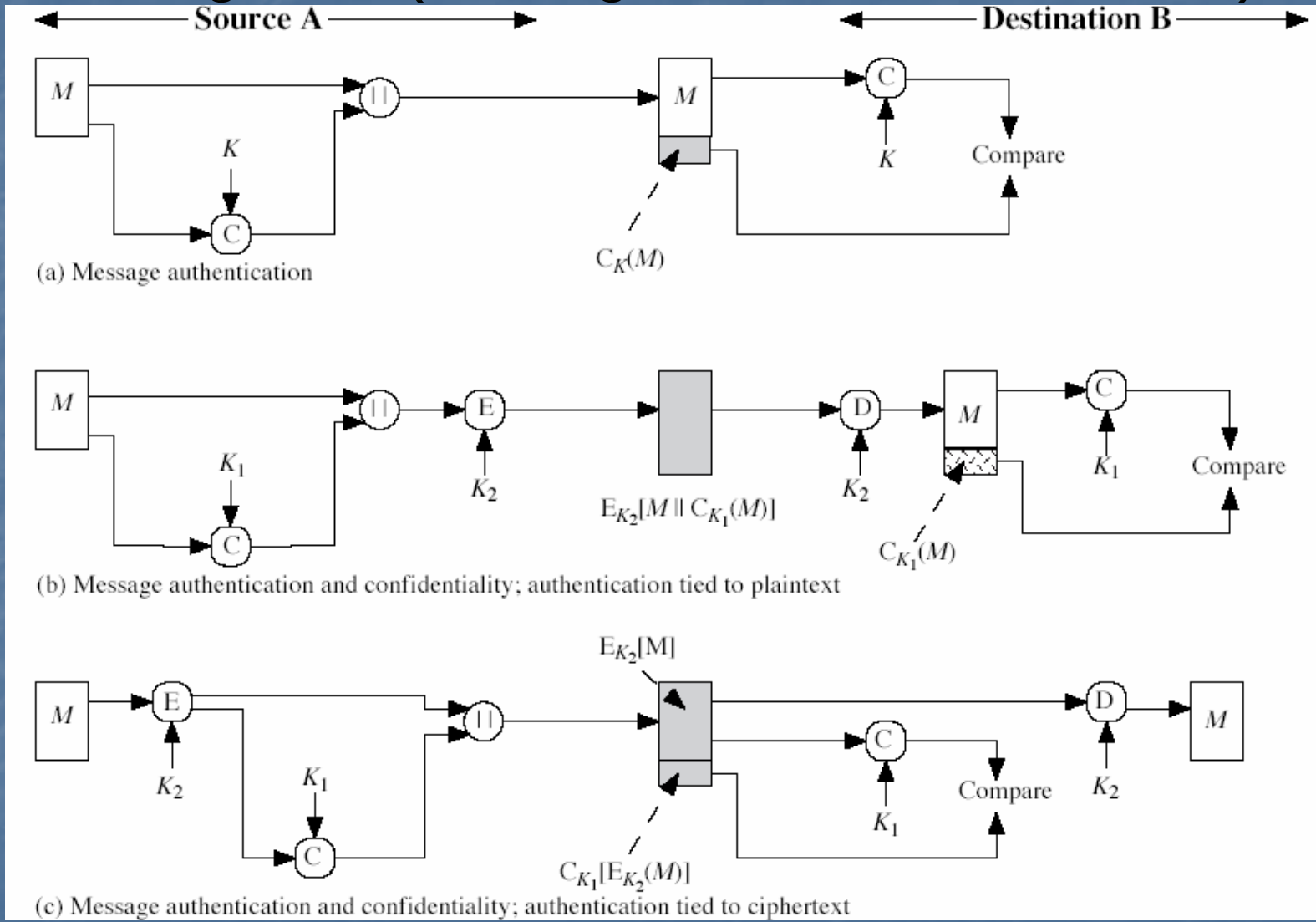
Instructor : Sheau-Dong Lang
lang@cs.ucf.edu

School of Electrical Engineering & Computer Science
University of Central Florida

Message Authentication

- Recall why we have used message authentication
  - Message authentication: a procedure to verify that received messages come from the alleged source and have not been altered.
  - Protects two parties who exchanges from any third party.
  - Trust between two communicating parties

Message Authentication using MAC (Message Authentication Code)



Why Digital Signature?

- What is the difference between message authentication and digital signature?
  - Message authentication does not protect two parties against each other.
  - What if there is no complete trust between sender and receiver?
- Examples (Bob sends an authenticated message to Alice)
  - Alice may forge a message and claim that it came from Bob by simply creating a message and append an authentication code using the key Bob and Alice share.
  - Bob can deny sending the message since it's possible for Alice to forge a message.

Digital Signature

- 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

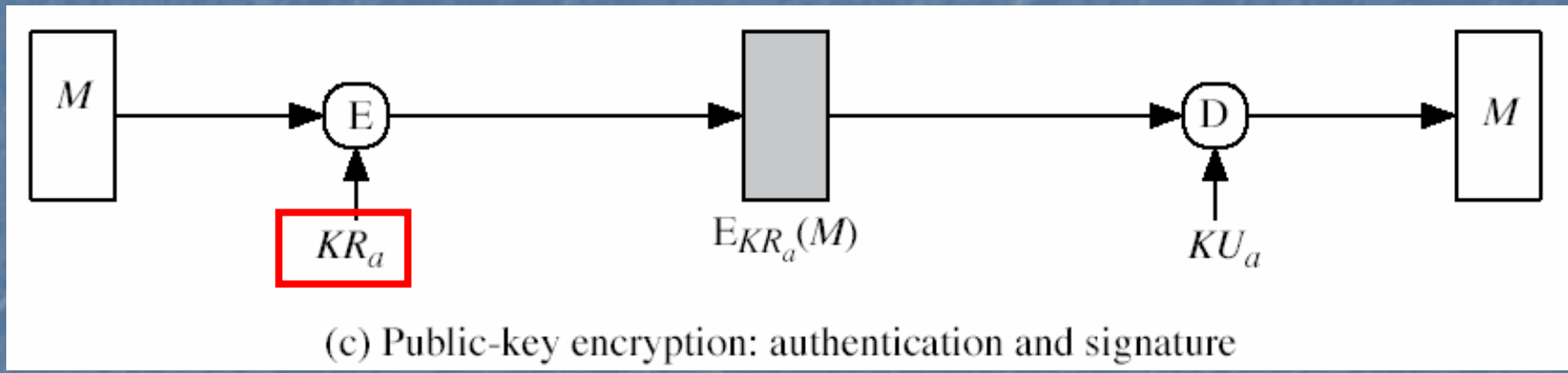
Digital Signature Properties

- 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

- Most of digital signature approaches can be classified into two categories
  - Direct vs. Arbitrated

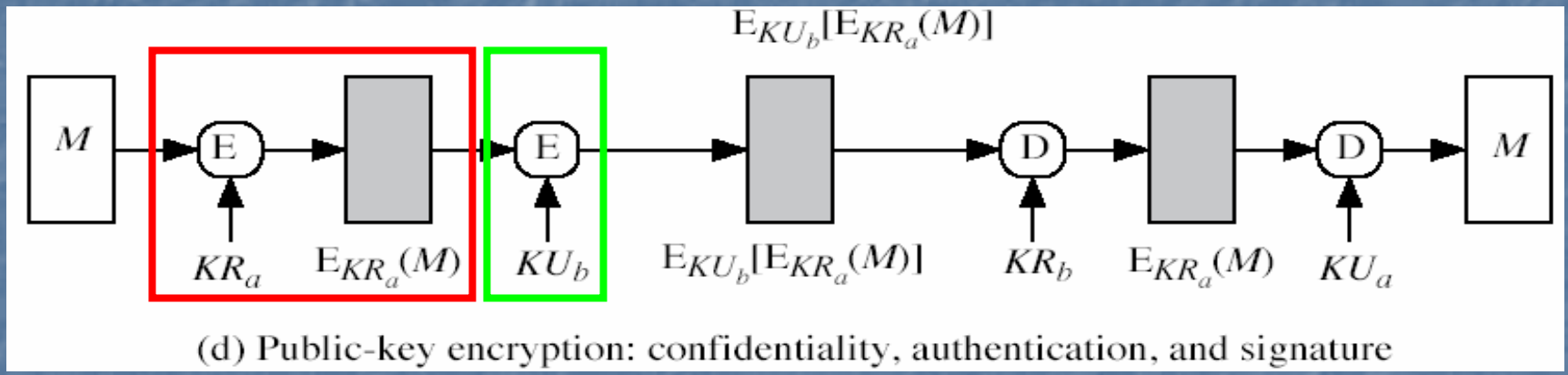
Direct Digital Signatures

- Involve only sender & receiver
- Assumed receiver has sender's public-key
- Digital signature made by sender signing entire message or hash with private-key

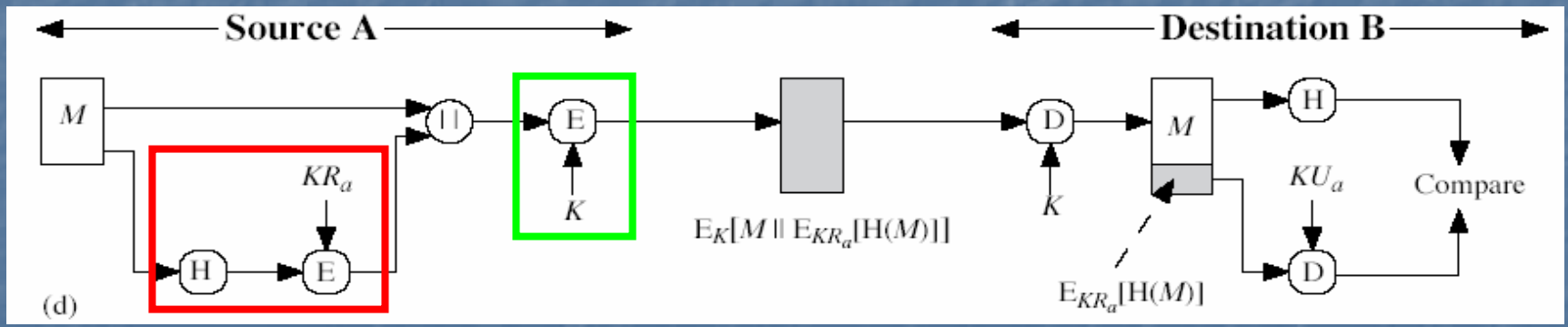


Direct Digital Signatures

- Confidentiality can be provided by further encrypting using receivers public-key or a shared secret key
- Important that sign first then encrypt message & signature



(d) Public-key encryption: confidentiality, authentication, and signature



Direct Digital Signatures

- Problems and weakness of direct digital signatures
  - Security depends on the sender's private key

    → could be lost and even the sender may deny sending the message claiming that the private key was stolen
  - Some private keys can be actually stolen and the opponent can replay the message at later time

Arbitrated Digital Signatures

- Try to improve the weaknesses of direct digital signatures
- Involves use of an arbiter A
  - Every signed message from the sender first goes to the arbiter
  - The arbiter validates the signed message
  - Then, the arbiter dates and sends it to the recipient
- Requires suitable level of trust in arbiter
- Can be implemented with either private or public-key algorithms

Arbitrated Digital Signatures

(a) Conventional Encryption, Arbiter Sees Message

(1) $X \rightarrow A$: $M \| E_{K_{xa}}[ID_X \| H(M)]$ signature

(2) $A \rightarrow Y$: $E_{K_{ay}}[ID_X \| M \| E_{K_{xa}}[ID_X \| H(M)] \| T]$

- Arbiter sees the message
  - Any eavesdropper can see the message
- Y cannot directly check the signature of X
- In the future disputes between X and Y, Y can present the signature received from the arbiter
- X and Y trust A such that A never releases the secrecy of K_{xa} and K_{ay}

Arbitrated Digital Signatures

(b) Conventional Encryption, Arbiter Does Not See Message

(1) $X \rightarrow A: ID_X \| E_{K_{xy}}[M] \| E_{K_{xa}}\left[ID_X \| H\left(E_{K_{xy}}[M]\right)\right]$ signature

(2) $A \rightarrow Y: E_{K_{ay}}\left[ID_X \| E_{K_{xy}}[M] \| E_{K_{xa}}\left[ID_X \| H\left(E_{K_{xy}}[M]\right)\right] \| T\right]$

- Arbiter cannot see the message
  - Provide confidentiality by using secret key K_{xy} between X and Y
- The rest is the same as the previous approach
- Problems
  - The arbiter could deny a signed message in alliance with the sender
  - The arbiter could forge the sender's signature in alliance with the receiver

Arbitrated Digital Signatures

(c) Public-Key Encryption, Arbiter Does Not See Message

(1) $X \rightarrow A$: $ID_X \| \mathrm{E}_{KR_x}\left[ID_X \| \mathrm{E}_{KU_y}\left(\mathrm{E}_{KR_x}[M]\right)\right]$ signature

(2) $A \rightarrow Y$: $\mathrm{E}_{KR_a}\left[ID_X \| \mathrm{E}_{KU_y}\left[\mathrm{E}_{KR_x}[M]\right] \| T\right]$

- X double encrypts a message M with X's private key, KU_x, and Y's public key, KU_y
- Could solve the previous problems

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 and to prevent masquerade
  - Timeliness – to prevent replay attacks

Types of Replay Attacks

- Where a valid signed message is copied and later resent
  - Simple replay
  - Repetition that can be logged
    - Replay a timstamped messages within the valid time window
  - Repetition that cannot be detected
    - The original message is suppressed and only the replay message arrives at the destination
  - Backward replay without modification
    - Reply back to the sender
    - Possible if symmetric encryption is used and the sender cannot easily recognize the difference between the sent message and the received message

Counter Replay Attacks

- Countermeasures include
  - Use of sequence numbers
    - Keep track of the last sequence number
    - Associated overhead
    - Generally impractical
  - Timestamps
    - Needs synchronized clocks
    - Accept a message as fresh only if the message contains a timestamp that is close enough to A's knowledge of current time
  - Challenge/response
    - Using unique nonce (challenge)
    - Receiver first sends a nonce to the sender and requires the sender to use the correct nonce value (response)

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→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

- 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
  - The opponent doesn't know K_a and K_b but simply replays it
- Modifications to address this require:
  - Timestamps (Denning 81)
  - Using an extra nonce (Neuman 93)

Denning's Protocol

- Protocol overview is:
  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: $E_{Ks}[f(N_1)]$

- A and B can verify timeliness by checking

  $|Clock - T| < \Delta t_1 + \Delta t_2$

  Δt_1 : estimated normal discrepancy between the KDC's clock and the local clocks

  Δt_2 : expected network delay time
- Replaying an old session can be verified at Step 3

Denning's Protocol

- Suppressed replay attacks
  - Sabotage or faults in the clocks of the synchronization of the clocks
  - If the sender's clock is ahead of the intended recipient's clock, then the message can be intercepted and replayed later as if it's current
  - (Solution) Clocks need resynchronization regularly

Improved Protocol

- Protocol overview is:
  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: $E_{Kb}[ID_A \,||\, K_s \,||\, T_b] \,||\, E_{Ks}[N_b]$

- Handshaking protocol
  - Alternative to avoid clock resynchronization for suppress-replay attack
  - Use of nonces for handshaking
    - Nounce N_a is returned back to A
    - Nounce N_b is also returned back to B
    - Therefore, they don't need to synchronize the clock

Using Public-Key Encryption

- Have a range of approaches based on the use of public-key encryption
- Need to ensure that both parties have the correct public keys for the other party
- Using a central Authentication Server (AS)
  - Provides public key certificates
- Various protocols exist using timestamps or nonces

Denning AS Protocol

- Assumption
  - A and B don't have each other's public key
  - AS has the public keys
- Denning presented the following:

1. $A \rightarrow AS: ID_A \,||\, ID_B$

2. $AS \rightarrow A: E_{KRas}[ID_A||KU_a||T] \,||\, E_{KRas}[ID_B||KU_b||T]$

3. $A \rightarrow B: E_{KRas}[ID_A||KU_a||T] \,||\, E_{KRas}[ID_B||KU_b||T] \,||\, E_{KUb}[E_{KRa}[K_s||T]]$

- Note session key is chosen by A, hence AS need not be trusted to protect it
- Timestamps prevent replay but require synchronized clocks
  - Look at T in each step: to use the same T, A and B must have the synchronized clock

One-Way Authentication

- The sender and the receiver need not be on-line at same time (eg. email)
- Have the E-mail header in clear so can be delivered by email system
- May want contents of body protected by encryption and sender authenticated

One Way Authentication : Using Symmetric Encryption

- Can refine use of KDC but can't have final exchange of nonces: so that B doesn't have to be on-line
  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

One Way Authentication : Public-Key Approaches

- Have seen some public-key approaches
- If confidentiality is major concern, can use:

  $A \rightarrow B: E_{KUb}[Ks] \,||\, E_{Ks}[M]$

  - Has encrypted session key, encrypted message
- If authentication needed use a digital signature with a digital certificate:

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

  - 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
- Creates a 320 bit signature, but with 512-1024 bit security
- Security depends on difficulty of computing discrete logarithms

Mathematical Background

- A primitive prime root of a prime number
  - q : a prime number
  - α : a primitive prime root of q
  - $\alpha \bmod q$, $\alpha^2 \bmod q$, $\alpha^3 \bmod q$, ..., $\alpha^{q-1} \bmod q$
  - Successive powers of α can generate all the numbers between 1 and $p-1$ such that $\alpha^i \bmod q$, where $1 \le i \le q-1$, will be distinct and consists of the integers 1 to $q-1$.
- Discrete logarithm
  - For any integer b less than q and a prime root α of prime number q, we can find a unique exponent i such that

    $b = \alpha^i \bmod q$ where $1 \le i \le q-1$

    exponent i is the discrete logarithm of b for the base α and mod q

DSA Key Generation

- Have shared global public key values (p,q,g):
  - a large prime $2^{L-1} < p < 2^{L}$
    - where $512 \leq L \leq 1024$ bits and is a multiple of 64
  - choose q, a 160 bit prime divisor of p-1
    - $2^{159} < q < 2^{160}$
  - choose $g = h^{(p-1)/q} \bmod p$
    - where $1<h<p-1$, $h^{(p-1)/q} \pmod p > 1$
- Users choose private & compute public key:
  - choose $0<x<q$
  - compute $y = g^{x} \pmod p$
  - It is computationally infeasible to determine x, which is the discrete logarithm of y to the base g, mod 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:

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

  $s = (k^{-1}.SHA(M) + x.r) \pmod q$
- 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
- see book web site for details of proof why

Summary

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