# Network Security

Asim Rasheed

# Digital Signatures

# Digital Signatures

- Message Authentication does not address issues of lack of trust between communicating parties
- Digital Signatures

–Cryptographic technique, analogous to hand-written signatures.

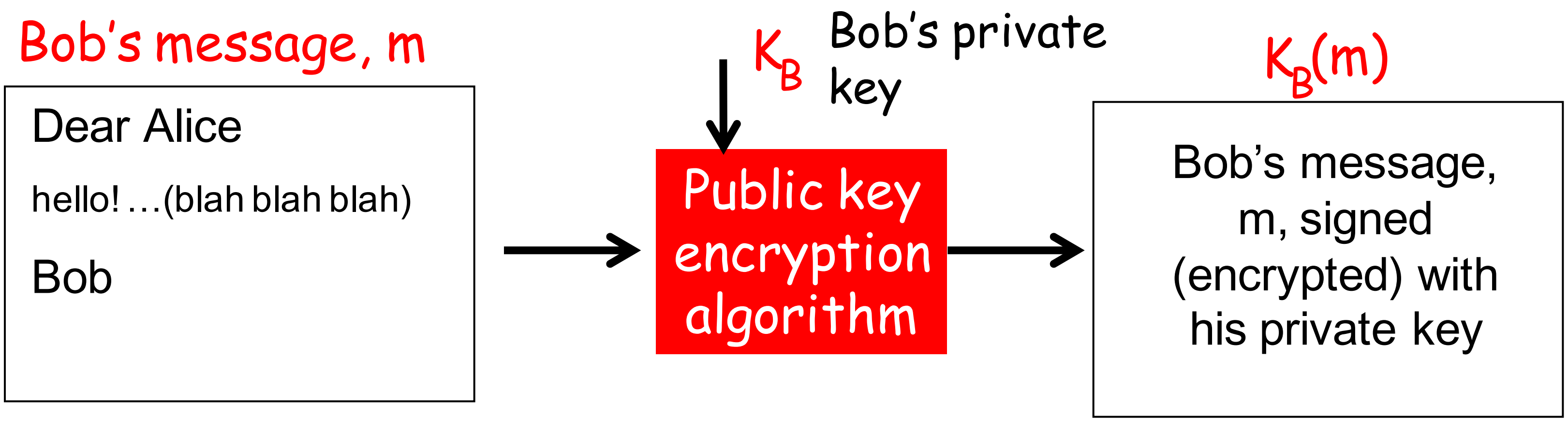
–sender (Bob) digitally signs document, establishing he is document owner/creator.

–verifiable, non-forgeable: recipient (Alice) can prove to someone that Bob, and no one else (including Alice), must have signed the document

# Digital Signatures

## Simple digital signature for message m:

- Bob signs m by encrypting with his private key $K_B$, creating "signed" message, $K_B(m)$



# Digital Signature Provides…

- Verification of author, date & time of signature
- Authentication of message contents
- Verification to 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
- Computationally infeasible to forge
  - with new message for existing digital signature
  - with fraudulent digital signature for given message
- Practical to save digital signature in storage

# Direct Digital Signature

- Involves only sender & receiver
- Assumption: 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 signs first, then encrypt message & signature

# Direct Digital Signature Security

- Depends on sender's private-key
- Sender may
  - later deny sending a particular message
  - claim that the private key was lost or lost
- One way to stop is use timestamps with signed message

# Arbitrated Digital Signatures

- Involves use of an arbiter that
  - Validates any signed message
  - Then the message is dated and sent to recipient
- Sender cannot disown the message
- 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 Signature Technique I

- Conventional Encryption, Arbiter sees message
  - $X \rightarrow A: M \| E_{K_{xa}}[ID_x \| H(M)]$
  - $A \rightarrow Y: E_{K_{ay}}[ID_x \| M \| E_{K_{xa}}[ID_x \| H(M)] \| T]$

- Sender X and arbiter A share a secret key and so do arbiter A and receiver Y
- Timestamp informs Y that message is not a replay
- Y stores M and the signature; in case of disputes, Y can send it to Arbiter

# Requirements for Arbiter

- X must trust A not to reveal their shared key and not to generate false signatures

- Y must trust A to send $E_{K_{ay}}[ID_x||M||E_{K_{xa}}[ID_x||H(M)]||T]$, only if the hash value is correct and signature was generated by X

- Both sides must trust A to resolve disputes

# Arbitrated Digital Signature Technique II

- Conventional Encryption, Arbiter does not see message
  - $X \rightarrow A$: $ID_x \| E_{K_{xy}}[M] \| E_{K_{xa}}[ID_x \| H(E_{K_{xy}}[M])]$
  - $A \rightarrow Y$: $E_{K_{ay}}[ID_x \| E_{K_{xy}}[M] \| E_{K_{xa}}[ID_x \| H(E_{K_{xy}}[M])] \| T]$

- Ensures confidentiality
- X and Y also share secret key
- A could form alliance with X to deny!

# Arbitrated Digital Signature Technique III

- Public key encryption, arbiter does not see message
  - X -> A: $ID_x || E_{KR_x}[ID_x || E_{KU_Y}(E_{KR_x}[M])]$
  - A -> Y: $E_{KR_a}[ID_x || E_{KU_y}[E_{KR_x}[M]] || T]$

- Secure technique
- Confidentiality is kept
- A only verifies the originator and Y can only decrypt the original 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 prevent masquerade and compromise of session;
    - Requires prior existence of secret or public keys
  - Timeliness – to prevent replay attacks
    - These attacks can allow an opponent to compromise a session key (at worst)
    - Or can disrupt operations by presenting messages that appear genuine (at minimum)

# Replay Attacks

- A valid signed message is copied and later resent
- Examples:
  - Simple replay: opponent simply copies and replays the message
  - Repetition that can be logged: replay a timestamped message within the valid time window
  - Repetition that cannot be detected: original message suppressed and only replay message arrived
  - Backward replay without modification: replay back to the sender

# Countermeasures

- Use of sequence numbers
  - Message accepted only if in proper order
  - Each party keeps track of the last sequence number (overhead)
- Timestamps
  - Message is accepted as fresh, only if timestamp is close to receiver's current time
  - Need synchronized clocks
- Challenge/Response (using unique nonce)
  - One party sends a challenge (nonce) and requires the subsequent message to contain correct nonce value
  - Unsuitable for connectionless applications

# Using Symmetric Encryption

- 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
  - Example: Kerberos

# Needham-Schroeder Protocol

- Original third-party key distribution protocol
- For session between A & B mediated by KDC
- Protocol overview:

  **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
- Vulnerable to 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
  - using an extra nonce

# 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 presented the following:

1. $A \rightarrow AS$: $ID_A || ID_B$
2. $AS \rightarrow A$: $E_{KR_{as}}[ID_A || KU_a || T] || E_{KR_{as}}[ID_B || KU_b || T]$
3. $A \rightarrow B$: $E_{KR_{as}}[ID_A || KU_a || T] || E_{KR_{as}}[ID_B || KU_b || T] || E_{KU_b}[E_{KR_a}[K_S || T]]$

- Note session key is chosen by A, hence AS need not be trusted to protect it
- Timestamps prevent replay but require synchronized clocks

# Public-Key Encryption using Nonce

- To avoid synchronization Woo and Lam proposed the following:

1. $A \rightarrow KDC$: $ID_A \| ID_B$
2. $KDC \rightarrow A$: $E_{KR_{auth}}[ID_B \| KU_b]$
3. $A \rightarrow B$: $E_{KU_b}[N_a \| ID_A]$
4. $B \rightarrow KDC$: $ID_B \| ID_A \| E_{KU_{auth}}[N_a]$
5. $KDC \rightarrow B$: $E_{KR_{auth}}[ID_A \| KU_a] \| E_{KU_b}[E_{KR_{auth}}[N_a \| K_s \| ID_B]]$
6. $B \rightarrow A$: $E_{KU_a}[E_{KR_{auth}}[N_a \| K_s \| ID_B] \| N_b]$
7. $A \rightarrow B$: $E_{K_s}[N_b]$

Binds $K_S$ to the two communicating parties

# Revision by Woo & Lam

1. $A \rightarrow KDC$: $ID_A \| ID_B$
2. $KDC \rightarrow A$: $E_{KR_{auth}}[ID_B \| KU_b]$
3. $A \rightarrow B$: $E_{KU_b}[N_a \| ID_A]$
4. $B \rightarrow KDC$: $ID_B \| ID_A \| E_{KU_{auth}}[N_a]$
5. $KDC \rightarrow B$: $E_{KR_{auth}}[ID_A \| KU_a] \| E_{KU_b}[E_{KR_{auth}}[N_a \| K_s \| ID_A \| ID_B]]$
6. $B \rightarrow A$: $E_{KU_a}[E_{KR_{auth}}[N_a \| K_s \| ID_A \| ID_B] \| N_b]$
7. $A \rightarrow B$: $E_{K_s}[N_b]$

# 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

# Using Symmetric Encryption

- Decentralized approach is impractical. Exchange of nonces not possible
- Can refine use of KDC but can't have final exchange of nonces:
  1. A→KDC: $ID_A \parallel ID_B \parallel N_1$
  2. KDC→A: $E_{Ka}[Ks \parallel ID_B \parallel N_1 \parallel E_{Kb}[Ks \parallel ID_A]\ ]$
  3. A→B: $E_{Kb}[Ks \parallel ID_A] \parallel E_{Ks}[M]$
- Does not protect against replays

–Could rely on timestamp in message, though email delays make this problematic

# Public-Key Approaches

- Public-key approaches require that sender knows the recipient's private key (authentication) or public key (confidentiality)
- If confidentiality is major concern, can use:

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

–Has encrypted session key, encrypted message

- If authentication is needed use a digital signature with a digital certificate:

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

–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
- Used to provide digital signatures
- Creates a 320 bit signature, but with 512-1024 bit security
- Security depends on difficulty of computing discrete logarithms

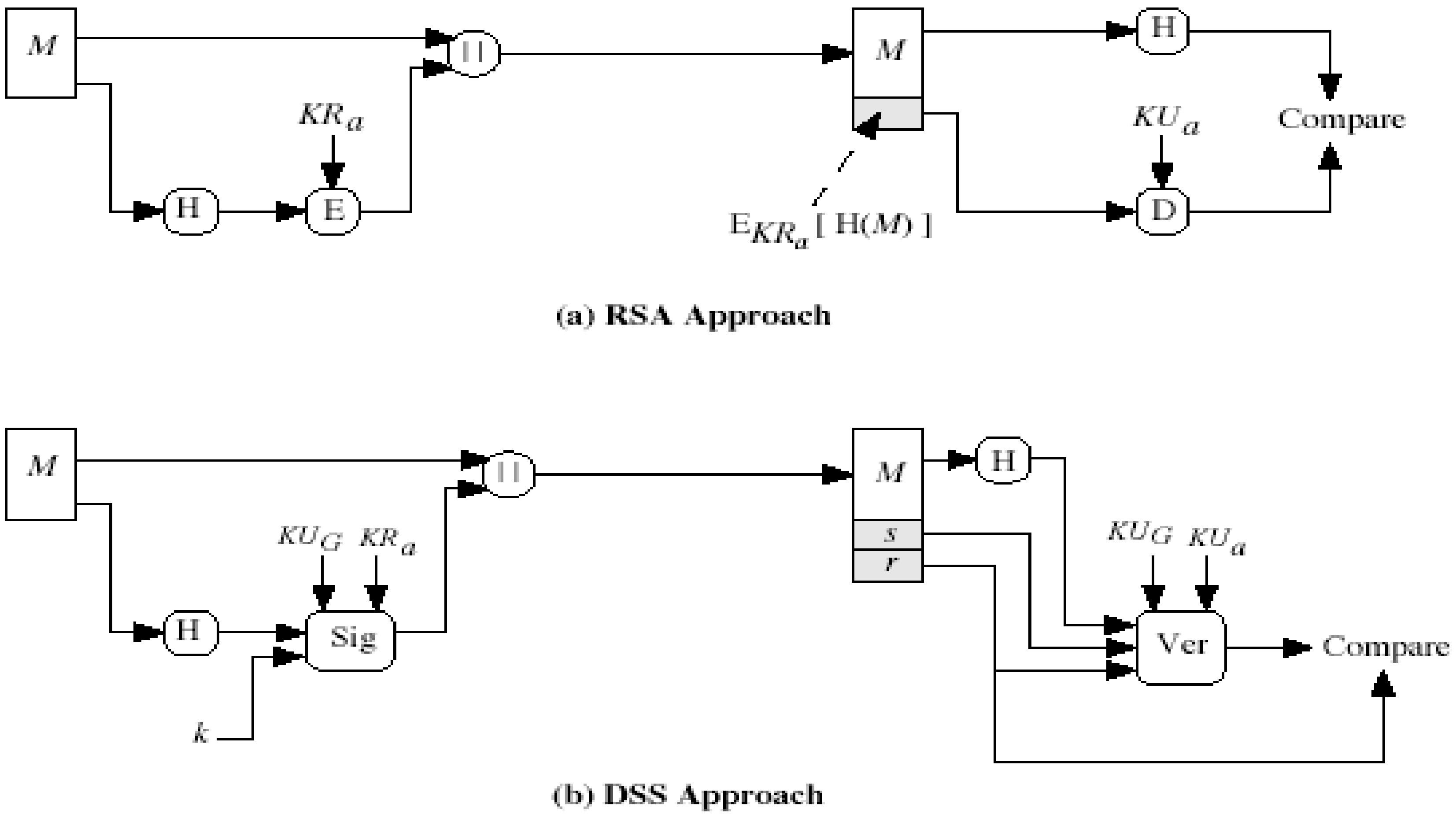
# Digital Signature Approaches
# RSA approach

- Message is input to a hash function
- Then encrypted using sender's private key
- Recipient produces a hash code of the message
- Decrypts the signature and compares the hashes

# Digital Signature Approaches DSS approach

- Message is input to a hash function
- Then hash is input to signature function with random number k
- Signature function also depends upon sender's private key
- s and r are then sent with the message
- Recipient produces a hash code of the message
- Hash, s and r are used to compute the signature and then compared

# Digital Signature Approaches



# 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 $g = h^{(p-1)/q} \bmod p$
    - where $1 < h < p-1$ & $g > 1$
- Each user chooses private key 'x' & computes public key 'y':
  - where $1 \le x \le q-1$
  - 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 and should be unique for each signing
- 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

# The Digital Signature Algorithm

**Global Public Key Components**

$p$ prime number where $2^{L-1} < p < 2^{L}$
for $512 \le L \le 1024$ and $L$ a multiple of 64
i.e., bit length of between 512 and 1024 bits in increments of 64 bits

$q$ prime divisor of $(p-1)$, where $2^{159} < q < 2^{160}$
i.e., bit length of 160 bits

$g$ $= h^{(p-1)/q} \bmod p$
where $h$ is any integer with $1 < h < (p-1)$
such that $h^{(p-1)/q} \bmod p > 1$

**User's Private Key**

$x$ random or pseudorandom integer with $0 < x < q$

**User's Public Key**

$y$ $= g^{x} \bmod p$

**User's Per-Message Secret Number**

$k$ = random or pseudorandom integer with $0 < k < q$

**Signing**

$r = (g^{k} \bmod p) \bmod q$

$s = \left[k^{-1}\left(\mathrm{H}(M) + xr\right)\right] \bmod q$

Signature $= (r, s)$

**Verifying**

$w = (s')^{-1} \bmod q$

$u_1 = \left[\mathrm{H}(M')w\right] \bmod q$

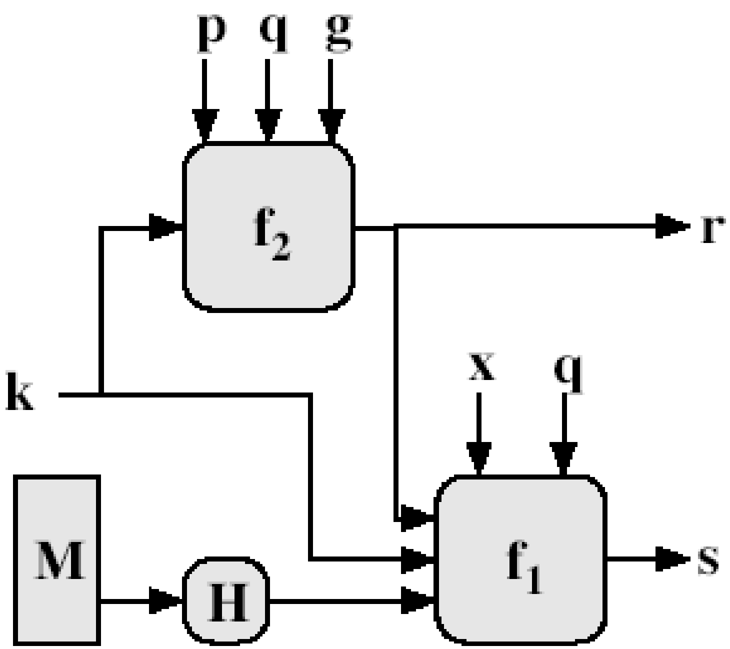
$u_2 = (r')w \bmod q$

$v = \left[\left(g^{u1} y^{u2}\right) \bmod p\right] \bmod q$

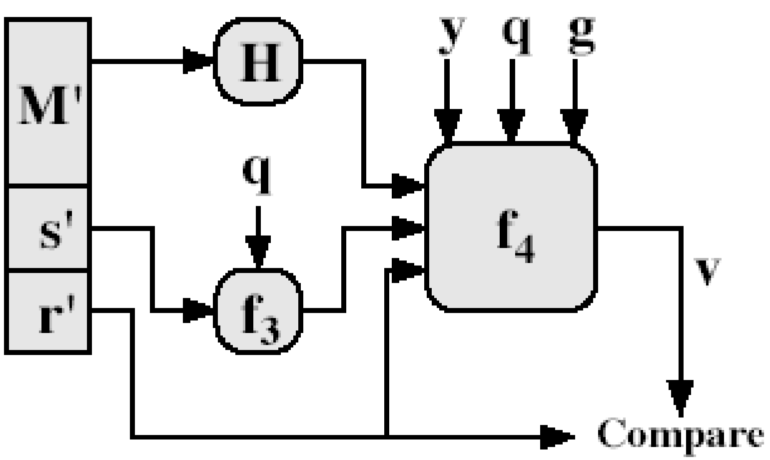
TEST: $v = r'$

$M$ = message to be signed
$\mathrm{H}(M)$ = hash of M using SHA-1
$M', r', s'$ = received versions of $M, r, s$

# Signature Verification




Signing




Verification

# Summary

- Have considered:
  - Digital signatures
  - Authentication protocols (mutual & one-way)
  - Digital Signature Standard

# Any question ?