Message authentication and digital signatures

• Message authentication
  – verify that the message is from the right sender, and not modified (incl message sequence)

• Digital signatures
  – in addition, non–repudiation

• Two levels:
  – authentication function
  – authentication protocol (using auth. function)
Authentication functions

- Message encryption
  - the whole ciphertext is the authenticator
- Message Authentication Code (MAC)
  - $C_k(m) \Rightarrow$ fix length value (the MAC)
- Hash function
  - $H(m) \Rightarrow$ fix length hash value
Authentication by encryption

• Conventional encryption
  – B receives \( c = E_k(m) \) from A, where \( k \) is secret
    • confidentiality: only A and B know \( k \)
    • authentication: only A could have sent it, cannot have been altered
    • but B can forge messages, and A can deny them
  – If arbitrary data is sent, how do we know a plaintext?
    • add a checksum to the message
      – \( E_k(m + f(m)) \) – internal error control
      – \( E_k(m) + f(E_k(m)) \) – external error control
        • can be forged!
Authentication by encryption

• Public–key encryption
  – $c = E_{dB}(m)$ gives confidentiality but no authentication
  – $c = E_{eA}(m)$ gives authentication but no confidentiality
  – $c = E_{dB}(E_{eA}(m))$ gives both
  – B cannot forge messages, and A cannot deny them
  – Still needs checksum for arbitrary data
Message Authentication Code

- Cryptographic checksum
  - $\text{MAC} = C_k(m)$, where $k$ shared secret key
  - send both $m$ and MAC
  - recipient computes $C_k(m)$ and compares with MAC
  - confidentiality:
    - $E_r(m+C_k(m))$ – plaintext authenticated
    - $E_r(m)+C_k(E_r(m))$ – ciphertext authenticated
- $C_k$ need not be reversible
  - many $m$ may have same MAC
MAC (cont)

- Advantages to encryption
  - faster
  - broadcast msgs can be checked at only one place
  - random tests possible
  - MAC can be kept and checked again any number of times
  - can give authentication without confidentiality
  - conf. and auth. can be handled at different levels
  - decryption loses authentication

- Fraud possible: A and B share $k$
MAC attacks

• $C$ maps $m$ of arbitrary length and $2^m$ $m$–bit keys to $2^n$ $n$–bit MAC values: collisions possible (likely)

• Brute force attack to find $k$ is no less difficult than finding a decryption key of same length
Requirements on a MAC fcn

- given $m$ and $C_k(m)$, infeasible to construct $m'$ s.t. $C_k(m') = C_k(m)$
  - cannot fake a MAC

- $C_k(m)$ uniformly distributed: random $m$ collide with probability $1/2^n$
  - thwarts brute-force chosen-plaintext attack

- For random $m$, $C_k(m) = C_k(f(m))$ with probability $1/2^n$
  - no weak spots
MAC based on DES

- Data Authentication Algorithm (DAA)
  - ANSI standard
- CBC with initialization vector 0
  - pad last plaintext block with zeros
  - MAC is leftmost 16–64 bits of last cipherblock
Hash functions

- One-way hash function takes variable-length $m$ and produces fixed-length hash value $H(m)$, a "fingerprint" of $m$.

- Requirements
  - one-way: given $x$, can’t find $m$ s.t. $x=H(m)$
    - difficulty $2^n$
  - weak collision resistance: given $x$, can’t find $y \neq x$ s.t. $H(x)=H(y)$
    - difficulty $2^n$
  - strong collision resistance: can’t find pair $(x,y)$ s.t. $H(x)=H(y)$
    - difficulty $2^{n/2}$
Hash usage

1. $m+H(m)$ – no confidentiality or authentication
2. $E_k(m+H(m))$ – auth&conf
3. $m+E_k(H(m))$ – same as MAC
4. $m+E_{eA}(H(m))$ – authentication (digital signature)
5. $E_k(m+E_{eA}(H(m)))$ – and confidentiality
6. $m+H(m+k)$ – authentication without encryption
7. $E_k(m+H(m+k))$ – and confidentiality
Hash algorithms

- **MD5**
  - widely used (e.g. PGP)
  - 128-bit hash values: collisions found "in 24 days"

- **SHA–1 and RIPEMD–160**
  - 160-bit hash values
  - now preferred over MD5 (e.g. in PGP)

- (see chapter 9)
Digital signatures

• MAC is not enough
  – recipient can fake it since he knows $k$
  – sender can therefore deny messages

• Digital signatures
  – verify the author, time and date
  – authenticates the contents
  – verifiable by third party
Varieties of digital signatures

- **Direct**
  - only source and destination involved
  - ex: use PKS—encrypted hash values
    - problem: sender can claim private key stolen (cf. credit card loss), even with timestamp

- **Arbitrated**
  - signed messages sent through trusted server
    - X sends $id_x + E_{eX}(id_x + E_{dy}(E_{eX}(m)))$ to arbitrer A
    - A checks X’s keys and sends $E_{eA}(id_x + E_{dy}(E_{eX}(m))+T)$ to Y
    - Y can find $id_x$ encrypted with A’s private key
    - A doesn’t see the message m
Digital Signature Standard

- DSS uses
  - SHA–1 for hash value
  - Digital Signature Algorithm (DSA)
    - based on ElGamal
    - can be fast: possible to precalculate slow things
- DSS can be used in PGP
Authentication protocols

- Mutual authentication
  - both parties ensure each other’s identities and, e.g., exchange session keys

- One-way authentication
  - recipient ensures sender is authentic e.g. for email
Mutual authentication

- Confidentiality and timeliness important
  - replay attacks could break confidentiality and/or authenticity
  - use timestamps or nonces (use–once random values)
- Conventional encryption
  - requires trusted Key Distribution Center
  - each user has a secret Master Key, shared with KDC
- Public–key encryption
  - possible with or without KDC
One–way authentication

- Desirable to avoid handshake protocols
- Conventional encryption: use KDC
- Public–key
  - encrypt whole message twice for conf & auth
  - faster: combine PK and conventional
    - send $E_{dB}(k_S)+E_{kS}(m)$ – confidentiality
    - send $m+E_{eA}(m)$ – "authenticity" (cf. man–in–the–middle)
    - send $E_{dB}(k_S)+E_{kS}(m+E_{eA}(m))$ – auth+conf (PGP)
Key management for PKS

• Distribution of public keys
  – Public announcement
    • forgery possible
  – Public directory run by trusted authority
    • keys submitted in secure+authentic way
    • keys retrieved from directory
      – using authentic paper directory
      – electronically from authority using PKS
Public–key certificates

• Avoid bottleneck at directory authority
  – Use Certificate Authority (CA)

• Requirements
  – anyone can find the name and public key of the certificate owner in the certificate
  – anyone can verify that the certificate was made by CA
  – anyone can verify the certificate is current
  – only the CA can create/update certificates
Certificates

- A certificate consists of the owner’s name, public key, and a timestamp, encrypted with the CA’s private key
  \[ C_A = E_{eCA}(id_A,d_A,T) \]
- To start communication, A sends his cert to B
  - B can decrypt using CA’s public key, validate the timestamp, check \( id_A \), and use \( d_A \)
X.509 Certificate Standard

- Used in SSL/TLS, S/MIME, SET, Ipsec,...
- Uses PKS and digital signatures
  - doesn’t specify which algorithms (but recommends)
- Kernel
  - format of certificates (fig 11.3)
  - CA hierarchy (fig 11.4)
  - revocation of certificates
    - CA has list of revoked certificates
  - one–, two–, and three–way authentication procedures
PGP key management

- Each user has two key-rings
  - private key ring
    - private keys (encrypted), public key ID,...
  - public key ring
    - public keys (own and others), user id, trust, signatures,...

- Key trust and validity: distributed
  - keys signed to certify their validity
  - a key is valid if signed by $n$ (1) fully trusted user,
    or by $m$ (3) semi–trusted users

- Keys distributed by key servers