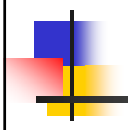


# SIGNATURES

Digital signatures: classical and public key

1

## Handwritten Signature

- Used everyday – in a letter, on a check, sign a contract
- A signature on a signed paper document specifies the person responsible for that document
- Signature becomes physically a part of the document
- Signature can be verified by comparing it to known authentic signatures (e.g., signature on a check, credit card)
- A copy of a signed paper document can usually be distinguished from the original

2

## Digital Signatures

- A digital signature scheme is a method of signing a message stored in electronic form
- A digital signature is not attached physically to the message, so the scheme must somehow "bind" the signature to the message
- A digital signature can be verified by a publicly known verification algorithm (so anyone can verify a digital signature)
- A copy of a signed message cannot be distinguished from the original (so we must add some information such as a date to ensure that the signed message cannot be reused)

3

## Digital Signature

- Construct that authenticated origin and/or contents of message in a manner provable to a disinterested third party ("judge")
- Sender cannot deny having sent message (service is "nonrepudiation")
  - Limited to *technical* proofs
    - Inability to deny one's cryptographic key was used to sign
  - One could claim the cryptographic key was stolen or compromised
    - Legal proofs, *etc.,* probably required; not dealt with here

4

## Digital Signature Scheme

Consists of two components

- A signing algorithm
  - Given a message, produces a signature
- A verification algorithm
  - Given a pair (m, s), returns true if s is the signature of the message m; false otherwise

5

## Common Error

- Classical: Alice, Bob share key $k$
  - Alice sends $m$ || { $m$ } $k$ to Bob

This is a digital signature

***WRONG***

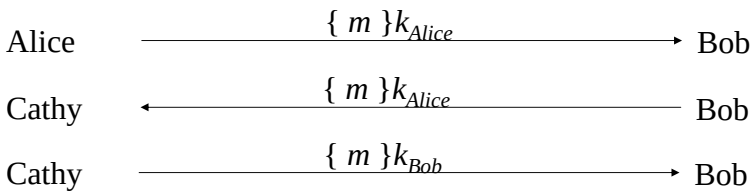
**This is not a digital signature**

- Why? Third party cannot determine whether Alice or Bob generated message

6

## Classical Digital Signatures

- Require trusted third party
  - Alice, Bob each share keys with trusted party Cathy
- To resolve dispute, judge gets $\{ m \} k_{Alice}$, $\{ m \} k_{Bob}$, and has Cathy decipher them; if messages matched, contract was signed




7

## Public Key Digital Signatures

- Alice's keys are $d_{Alice}$, $e_{Alice}$
- Alice sends Bob

$$m \,||\, \{ m \} d_{Alice}$$

- In case of dispute, judge computes

$$\{ \{ m \} d_{Alice} \} e_{Alice}$$

- and if it is $m$, Alice signed message
  - She is the only one who knows $d_{Alice}$!

8

## Digital Signatures in RSA

- RSA has an important property, not shared by other public key systems: encryption and decryption are commutative
  - Encryption followed by decryption yields the original message
    $$(M^e \bmod n)^d \bmod n = M$$
  - Decryption followed by encryption yields the original message
    $$(M^d \bmod n)^e \bmod n = M$$

9

## RSA Digital Signatures

- Use private key to encipher message
  - Protocol for use is *critical*
- Key points:
  - Never sign random documents, and when signing, always sign hash and never document
    - Mathematical properties can be turned against signer
  - Sign message first, then encipher
    - Changing public keys causes forgery

10

## Attack #1

- Example: Alice, Bob communicating
  - $n_A = 95$, $e_A = 59$, $d_A = 11$
  - $n_B = 77$, $e_B = 53$, $d_B = 17$
- 26 contracts, numbered 00 to 25
  - Alice has Bob sign 05 and 17:
    - $c = m^{d_B} \bmod n_B = 05^{17} \bmod 77 = 3$
    - $c = m^{d_B} \bmod n_B = 17^{17} \bmod 77 = 19$
  - Alice computes $05 \times 17 \bmod 77 = 08$; corresponding signature is $03 \times 19 \bmod 77 = 57$; claims Bob signed 08
  - Judge computes $c^{e_B} \bmod n_B = 57^{53} \bmod 77 = 08$
    - Signature validated; Bob is toast

11

## Attack #2: Bob's Revenge

- Bob, Alice agree to sign contract 06
- Alice enciphers, then signs:
  $(m^{e_B} \bmod 77)^{d_A} \bmod n_A = (06^{53} \bmod 77)^{11} \bmod 95 = 63$
- Bob now changes his public key
  - Computes $r$ such that $13^r \bmod 77 = 6$; say, $r = 59$
  - Computes $re_B \bmod \phi(n_B) = 59 \times 53 \bmod 60 = 7$
  - Replace public key $e_B$ with 7, private key $d_B = 43$
- Bob claims contract was 13. Judge computes:
  - $(63^{59} \bmod 95)^{43} \bmod 77 = 13$
  - Verified; now Alice is toast

12

## El Gamal Digital Signature

- Relies on discrete log problem
- Choose $p$ prime, $g$, $d < p$ , compute $y = g^d \bmod p$
- Public key: ($y$, $g$, $p$); private key: $d$
- To sign contract m:
  - Choose $k$ relatively prime to $p$–1, and **not yet used**
  - Compute $a = g^k \bmod p$
  - Find $b$ such that $m = (da + kb) \bmod p$–1
  - Signature is ($a$, $b$)
- To validate, check that
  - $y^a a^b \bmod p = g^m \bmod p$

13

## Example

- Alice chooses $p = 29$, $g = 3$, $d = 6$

  $y = 3^6 \bmod 29 = 4$
- Alice wants to send Bob signed contract 23
  - Chooses $k = 5$ (relatively prime to 28)
  - This gives $a = g^k \bmod p = 3^5 \bmod 29 = 11$
  - Then solving $23 = (6 \times 11 + 5b) \bmod 28$ gives $b = 25$
  - Alice sends message 23 and signature (11, 25)
- Bob verifies signature: $g^m \bmod p = 3^{23} \bmod 29 = 8$ and $y^a a^b \bmod p = 4^{11} 11^{25} \bmod 29 = 8$
  - They match, so Alice signed

14

## Attack

- Eve learns $k$, corresponding message $m$ and signature ($a$, $b$)
  - Extended Euclidean Algorithm gives $d$, the private key
- Example from above: Eve learned Alice signed last message with $k = 5$

  $m = (da + kb) \bmod (p–1) = (11d + 5\times25) \bmod 28$

  so Alice's private key is $d = 6$

15

## Digital Signature Standard (DSS)

- Developed by NSA
- Proposed in 1991, adopted as a standard in 1994
- Modification of El Gamal signature scheme
- NIST Digital Signature Standard
  - System-wide constants
    - p  512-1024 bit prime (approx.170 digits)
    - q  160 bit prime divisor of p-1
    - a  $= h^{((p-1)/q)} \bmod p$, for chosen h with1<h<p-1
  - x  selected random between 1 and q-1 inclusive
  - y  $= a^{x} \bmod p$
  - The public key is (p, q, a, y)

16

## NIST Digital Signature Standard

- To sign a message m
  - choose a random r
  - compute v = ($a^r$ mod p) mod q
  - compute s = (m+xv)/r mod q
  - signature on the message m is (v,s)
- To verify a signature (v,s) on m
  - compute u1 = m/s mod q
  - compute u2 = v/s mod q
  - verify that v = ($a^{u1}$ * $y^{u2}$ mod p) mod q

17

## Crypto Key Infrastructure

- Goal: bind identity to key
- Classical: not possible as all keys are shared
  - Use protocols to agree on a shared key (seen earlier)
- Public key: bind identity to public key
  - Crucial as people will use key to communicate with principal whose identity is bound to key
  - Erroneous binding means no secrecy between principals
  - Assume principal identified by an acceptable name

18

# Key Distribution Solutions

**Certificate Authority:**

**Centralized Approach**

- Has a very well publicized public key, so that clients can be sure that they're talking to the CA
- This is the public key version of the Kerberos protocol

19

# Digital Certificates

- A digital certificate is an assertion
  - Digitally signed by a "certificate authority", "famous" and with known public key
- An assertion
  - Typically an identity assertion, sometimes a list of authorizations
- Create token (message) containing
  - Identity of principal (here, Alice)
  - Corresponding public key
  - Timestamp (when issued)
  - Other information (perhaps identity of signer)

signed by trusted authority (here, Cathy)

$$C_A = \{ e_A \,||\, \text{Alice} \,||\, T \}\, d_C$$

20

## Use

- Bob gets Alice's certificate
  - If he knows Cathy's public key, he can decipher the certificate
    - When was certificate issued?
    - Is the principal Alice?
  - Now Bob has Alice's public key
- Problem: Bob needs Cathy's public key to validate certificate
  - Problem pushed "up" a level
  - Two approaches: Merkle's tree, signature chains

21

## Certificate Signature Chains

- Create certificate
  - Generate hash of certificate
  - Encipher hash with issuer's private key
- Validate
  - Obtain issuer's public key
  - Decipher enciphered hash
  - Recompute hash from certificate and compare
- Problem: getting issuer's public key

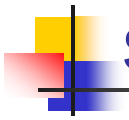
22

# Key Distribution Solutions

**Certificate Authority:**
**Distributed Approach**

- PGP “web of trust”
  - Each client maintains a list of
    - Who you know
    - Transitive set of people that they have introduced you to
  - Confidence ratings on
    - Their identity
    - Their veracity

23



# Storing Keys

- Multi-user or networked systems: attackers may defeat access control mechanisms
  - Encipher file containing key
    - Attacker can monitor keystrokes to decipher files
    - Key will be resident in memory that attacker may be able to read
  - Use physical devices like “smart card”
    - Key never enters system
    - Card can be stolen, so have 2 devices combine bits to make single key

24

## Key Revocation

- Certificates invalidated *before* expiration
  - Usually due to compromised key
  - May be due to change in circumstance (*e.g.,* someone leaving company)
- Problems
  - Entity revoking certificate authorized to do so
  - Revocation information circulates to everyone fast enough
    - Network delays, infrastructure problems may delay information

25

## Key secrecy

- There are problems with truly secret keys:
  - What if someone loses or forgets a key?
  - What if the holder of the key resigns or is killed?
  - What if the user is a criminal?
- On the other hand simply divulging the key to anybody (even - or perhaps especially! - the government) is very insecure
- Encryption Dilemma
  - Public's need for secure communication
  - Government's need for lawful access to information

26

# Key Escrow

- A proposed solution is *Key Escrow:*
  - The private key is broken into pieces, which can be verified to be correct
  - Each piece is given to some authority
  - The whole key can only be reconstructed if all the authorities agree
- This is the basis of the US Clipper Chip
  - http://www.cosc.georgetown.edu/~denning
  - http://www.cpsr.org/program/clipper/clipper.html

27