UnitelmaSapienza
Università degli Studi di Roma

Sapienza
Università di Roma
Dipartimento di Informatica

Signatures

Digital signatures: classical and public key

F. Parisi Presicce

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

Digital Signature

- 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)

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

Digital Signature Scheme

Consists of two components

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

Common Error

Classical: Alice, Bob share key k

- Alice sends $m \| \{ m \} k$ to Bob

  Is this a digital signature?

NO

This is not a digital signature

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

Classical Digital Signatures

Require trusted third party

- Alice, Bob each share keys with trusted party Cathy

To resolve dispute

- judge gets $\{ m \}\, k_{Alice}$ and $\{ m \}\, k_{Bob}$, and has Cathy decipher them;
- if messages matched, contract was signed

$$\text{Alice} \xrightarrow{\{ m \} k_{Alice}} \text{Bob}$$

$$\text{Cathy} \xleftarrow{\{ m \} k_{Alice}} \text{Bob}$$

$$\text{Cathy} \xrightarrow{\{ m \} k_{Bob}} \text{Bob}$$

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 result is m, Alice signed message

- She is the only one who knows d_{Alice} !

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
  - $(M^d \bmod n)^e \bmod n = M$

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

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*17 mod 77 = 08; corresponding signature is 03*19 mod 77 = 57; claims Bob signed 08
- Judge computes $c^{e_B} \bmod n_B = 57^{53} \bmod 77 = 08$
  - Signature validated; Bob is toast !

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 $re_B \bmod \phi(n_B) = 59*53 \bmod 60 = 7$
- 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

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 r relatively prime to p–1, and **not yet used**
- Compute $a = g^r \bmod p$
- Find b such that $m = (d*a + r*b) \bmod (p-1)$
- Signature is (a, b)

To validate, check that

- $Y^a * a^b \bmod p = g^m \bmod p$

Example

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

$$y = 3^6 \bmod 29 = 4$$

Alice wants to send Bob signed contract 23

- Chooses r = 5 (relatively prime to 28)
- This gives $a = g^r \bmod p = 3^5 \bmod 29 = 11$
- Then solving $23 = (6*11 + 5*b) \bmod 28$ 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

Attack

Eve learns r, 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 $r = 5$

$$m = (d*a + r*b) \bmod (p-1) = (11*d + 5*25) \bmod 28$$

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

Digital Signature Standard (DSS)

Developed by NSA

Proposed in 1991, adopted in 1994, latest standard 2013

Modification of El Gamal signature scheme

NIST Digital Signature Standard FIPS186-4 2013

- System-wide constants
  - p at least 1024 bit prime
  - q at least 224 bit prime divisor of p-1
  - g generator $1<g<p$ of subgroup of GF(p) of order q
- x private key (pseudo)randomly generated $1<x<q$
- $y = g^x \mod p$
- The public key is (p, q, g, y)

NIST Digital Signing

To sign the hash h(m) of a message m

- Convert h(m) into an integer
- choose random k unique to each message
- compute $r = (g^k \bmod p) \bmod q$
- compute $s = k^{-1} * (h(m)+x*r) \bmod q$
- If either r=0 or s=0, generate new k
- signature on the message m is (r , s)

NIST Digital Signature verifying

To verify a signature (r',s') on m

- Verify that 0< r', s' <q : if not, discard
- compute $u1 = s'^{-1} * h(m) \bmod q$
- compute $u2 = s'^{-1} * r' \bmod q$
- verify that $r' = (g^{u1} * y^{u2} \bmod p) \bmod q$
  - If verified, signature valid;
  - if not, either signing process incorrect or imposter attempt at forging signature

NIST Digital Signature Algorithm (DSA)

Security based on two DLP, one mod p and one mod q

Increasing the size of one without increasing the other one does not improve security

the standard specifies appropriate lengths for p and q

- 1024, 160
- 2048, 224
- 2048, 256
- 3072, 256

Signature in RSA

Traditionally, RSA has been used for signatures by encrypting with private key the result of hashing the message, possibly after some fixed padding

- Why padding? To make full use of RSA with 2048 bits with SHA-256, for example

But commonly used hash functions such as MD5, SHA-1 have been "compromised" rendering the signatures not full-proof

- SHA-2 family still secure

In 1996, Bellare and Rogaway proposed a Probabilistic Signature Scheme (PSS) which provides "provable security"