Digital signatures

GIANLUCA DINI
Dept. of Ingegneria dell'Informazione
University of Pisa
email: gianluca.dini@unipi.it

Version: 2021-04-26

Digital Signatures

OVERVIEW

The problem

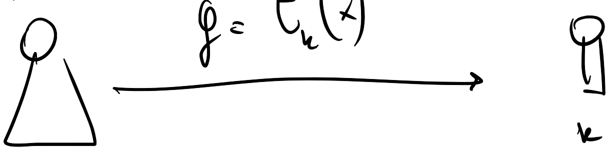
- Alice and Bob share a secret key k
- Alice receives and decrypts a message which makes semantic sense
- Alice concludes that the message comes from Bob
  - Message origin authetication ➔ message integrity
    - Beware, we know that ciphers are malleable!
    - MDC / MAC do not change the reasoning

Apr-21 Digital signatures 3

A

$y = E_k(x)$

B



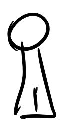
k

k

$x = D_x(y)$

X HA UN SIGNIFICATO SEMANTICO

X VIENE DA ALICE



TRENT

(TRUSTED THIRD PARTY)

– HA BISOGNO DI LEI PER ESPRIMERE UN GIUDIZIO

MA COMUNQUE NON PUO' SAPERE SE IL MESSAGGIO VIENE DA ALICE O BOB PERCHE' CONDIVIDONO LA CHIAVE SEGRETA

⇓

VA BENE SE HO MUTUAL TRUST

1) NON RIVELA SEGRETI

2) HA COMPETENZE PER NON RIVELARE I SEGRETI

The problem →

- The reasoning above works under the assumption of mutual trust
  - If a dispute arise, Alice cannot prove to a third party that Bob generated the message
- There are practical cases in which Alice and Bob wish to securely communicate but they don't trust each other
  - E.g.: e-commerce: customer and merchant have conflicting interests

The problem

- Provability/verifiability requirement
  - If a dispute arises an unbiased third party must be able to solve the dispute equitably, without requiring access to the signer's secret
- Symmetric cryptography is of little help
  - Alice and Bob have the same knowledge and capabilities
- Public-key cryptography is the solution
  - Make it possible to distinguish the actions performed by who knows the private key

Digital signature scheme

UNIVERSITÀ DI PISA

- A signature scheme is defined by three algorithms
- Key generation algorithm G
  - takes as input 1^n and outputs (pubk, privk)
- Signature generation algorithm S
  - takes as input a private key privk and a message x and outputs a signature σ = S(privk, x)
- Signature verification algorithm V
  - takes as input a public key pubk, a signature σ and (optionally) a message x and outputs True o False

Apr-21 Digital signatures 6

Occasionally, I will denote S(privK, x) as $S_{privK}(x)$.

The verification algorithm V(pubK, σ, [x]) returns true if σ is the digital signature of x by means of the private key privK corresponding to the public key pubK specified as argument. V returns false otherwise. Message x is specified in brackets to mean that it is an optional parameters. There exist digital signatures scheme that returns message x as a side-product of a successful signature verification. This means that V(pubK, σ) returns (true, x) in case of successful verification or (false, -).

[PRIVATE KEY, PUBLIC KEY]

SIGNATURE

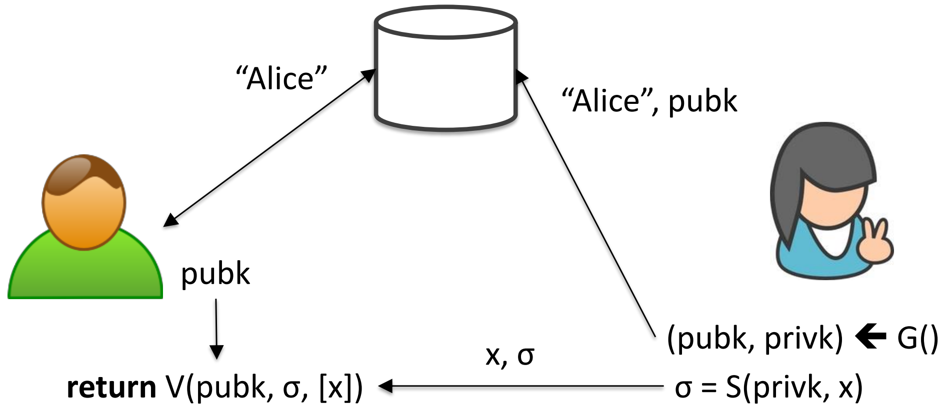
S = S(PRIVK, x)

VERIFICATION

V{pubk, s, (x)} → {TRUE, FALSE} O ALSO V(PUBK, S): {(TRUE, x) / (FALSE)}

OPTIONAL

Communication model



[*optional field*]

Security model

- Threat model
  - Adaptive chosen-message attack
    - Assume the attacker can induce the sender to sign *messages of the attacker's choice*
    - The attacker knows the public key
  - Security goal: existential unforgeability
    - Attacker should be *unable* to forge valid signature on *any* message not signed by the sender

Properties

- Consistency Property
  - For all x and (pubk, privk), V(pubk, [x] S(privk, x)) = TRUE
- Security property (informal)
  - Even after observing signatures on multiple messages, an attacker should be unable to forge a valid signature on a new message

Comments

- Security property implies
  - Integrity
  - Verifiability
  - Non-repudiation
  - No confidentiality
    - Use a cipher (AES, 3DES,…) if confidentiality is a requirement

Integrity

Ad adversary who doesn't know the privK cannot generate a dig sig on a new message.

Verifiability

A digital signature is a number dependent on some secret *known only* to the signer. Thus a signed message can be unambiguously traced back to its originator since a valid dig sig can only be computed with the unique signer's private key. Only the signer has the ability to generate a signature on his behalf. Hence, we can prove that the signing party has actually generated the message.

If a dispute arises an unbiased third-party can solve the dispute equitably, without requiring access to the signer's secret

Non-repudiation

The signer cannot deny it signed the message.

Algorithm families

- Integer factorization
  - RSA
- Discrete logarithm
  - ElGamal, DSA
- Elliptic curves
  - ECDSA

Digital signatures

NON-REPUDIATION VS AUTHENTICATION

Non-repudiation

- Non-repudiation prevents a signer from signing a document and subsequently being able to successfully deny having done so.

Non-repudiation vs authentication

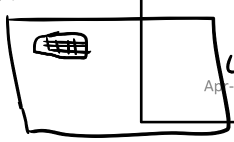
- Authentication
  - Based on symmetric cryptography
  - Allows a party to convince itself or a mutually trusted party of the integrity/authenticity of a given message at a given time t_0
- Non-repudiation
  - based on public-key cryptography
  - allows a party to convince others at any time $t_1 \geq t_0$ of the integrity/authenticity of a given message at time t_0

Dig sig vs non-repudiation

UNIVERSITÀ DI PISA

- Data origin authentication as provided by a digital signature is valid only while the secrecy of the signer's private key is maintained
- A threat that must be addressed is a signer who intentionally discloses his private key, and thereafter claims that a previously valid signature was forged
- This threat may be addressed by
  - Prevent direct access to the key → MANTENGO LA CHIAVE PRIVATA CIFRATA
    $z = E_k(\text{PRIV K})$ DOVE $k = H(\text{PWD})_{128}$
  - Use of a trusted timestamp agent
  - Use of a trusted notary agent

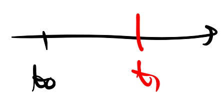
PW



MALWARE ACCEDE AL SISTEMA E RUBA LA PASSWORD CIFRATA PER UN OFFLINE ATTACCO.

t_0 : $\sigma = S(\text{PRIVATE KEY}, x)$

PRIVK = ALICE PRIVATE KEY

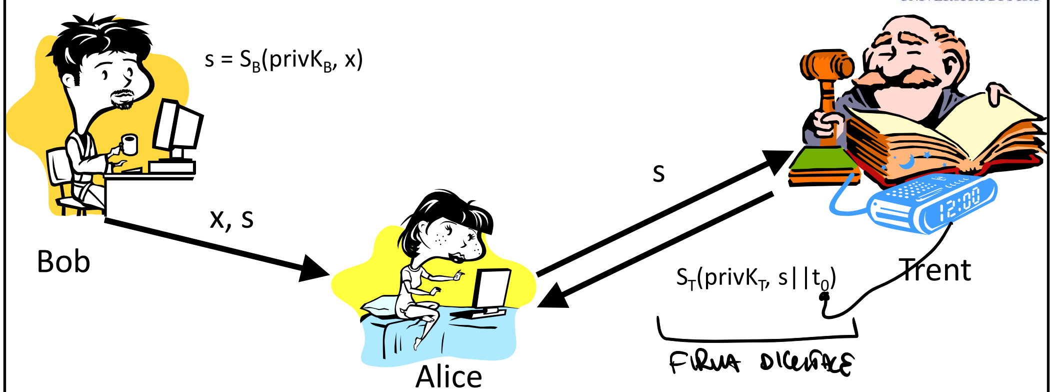


ALICE DICE CHE LA CHIAVE PRIVATA È STATA COMPROMESSA

POTREBBE ESSERE RIPUDIATO TUTTO QUELLO FIRMATO

Trusted Timestamping Service

UNIVERSITÀ DI PISA



- Trent certifies that digital signature *s exists* at time t_0
- If Bob's $privk_B$ is compromised at $t_1 > t_0$, then s is valid

s COMPROMESSA

t_0 t_1

MA A t_0 ERA VALIDA!

Trusted Notary Service (teorico)

UNIVERSITÀ DI PISA

- TNS generalize the TTS
- Trent certifies that a certain statement on the digital signature s is true at a certain time t0
- Examples of statements
  - Signature s exists at time t0
  - Signature s is valid at time t0
- Trent may certify the existence of a certain document
  - s = S(privKT, H(documents) || timestamp)
  - Document remains secret
- Trent is trusted to verify the statement before issuing it

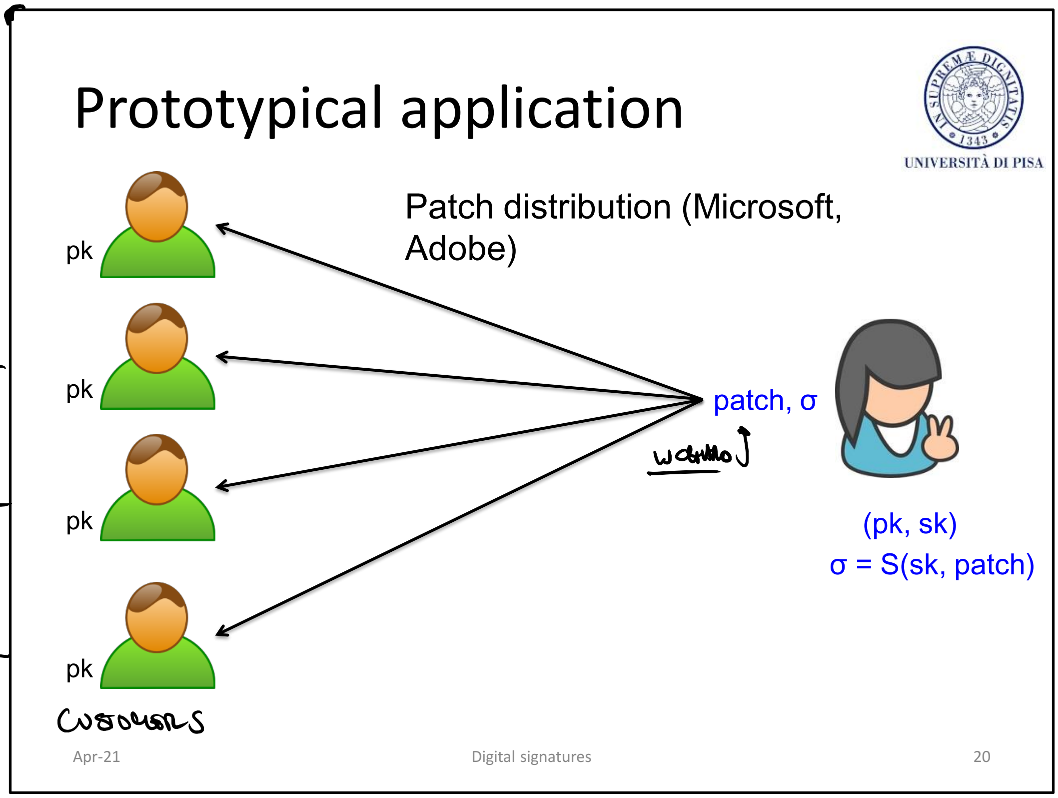
Digital Signatures

COMPARISON TO MAC

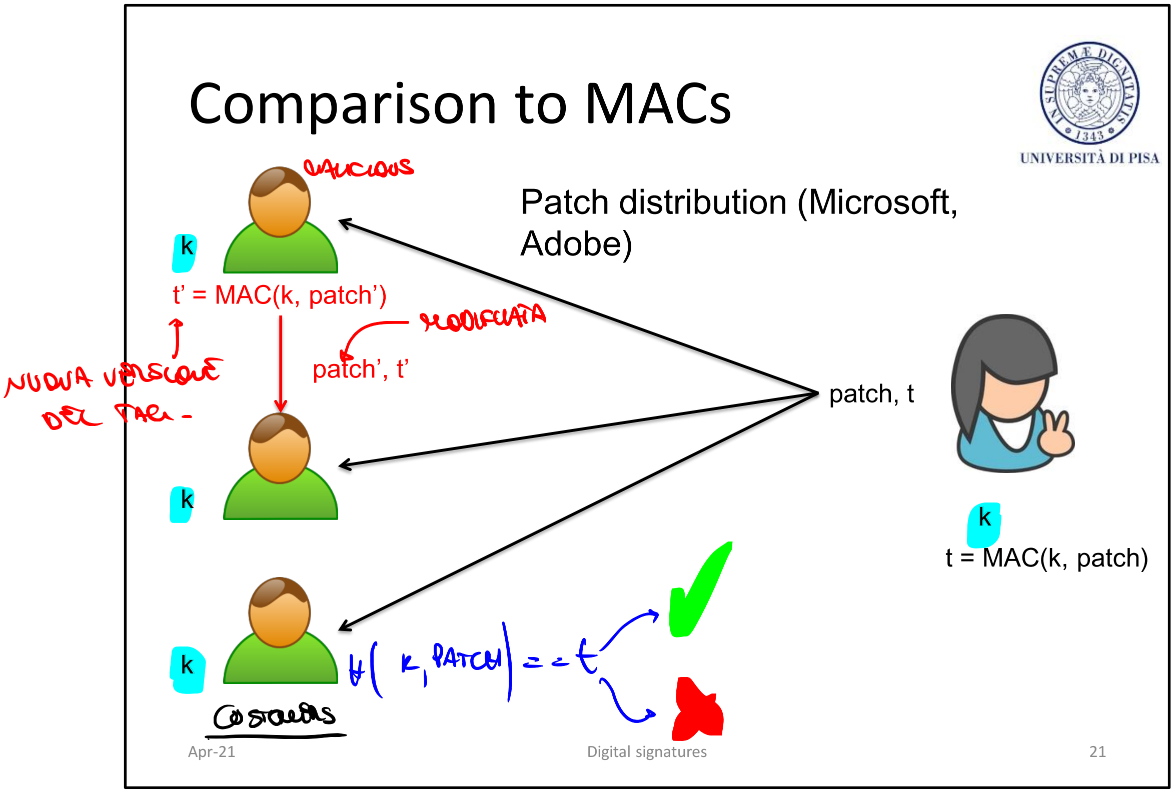
Digital signatures

- Provide *integrity* in the public-key setting
- Analogous to message authentication codes (MACs) but some key differences…

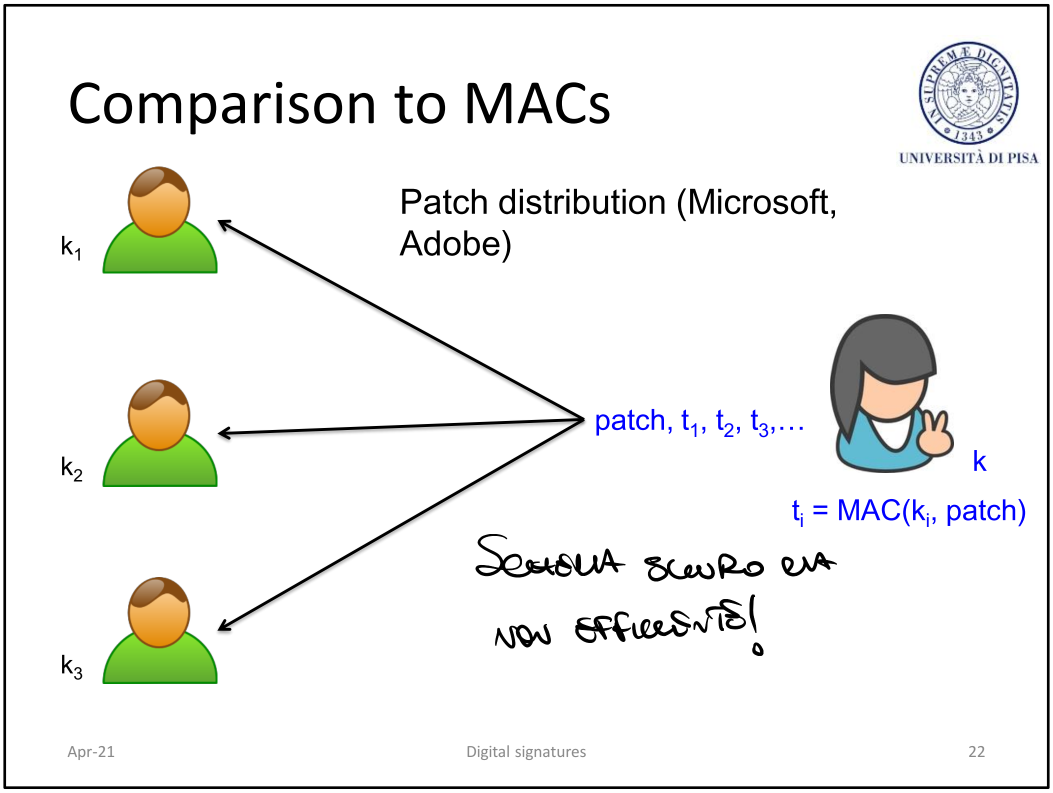
VERIFICANO LA FIRMA DIGITALE SULLA PATCH E SE E' VALIDA LA APPLICANO



The pk may be embedded in the software the client gets.



Customers share the symmetric key necessary to verify the tag t. A malicious customer may modify the patch, recalculate the tag and distribute it to the other users. This can be solved by means of MACs by distributing a different key to every different customer.



Each customer has its own key and therefore receives a personalized tag. A malicious customer cannot forge the tag of other customers. This solution has several drawbacks. From a computational point of view, you have to compute one tag for each customer. From a network viewpoint you must transmit one tag for each customer. Finally, you have to distribute one key to every customer.

Comparison to MACs

- Single shared key k
  - A client may forge the tag
  - Unfeasible if clients are not trusted

NON SICURA

- Point-to-point key k_i
  - Computing and network overhead
  - Prohibitive key management overhead
  - Unmanageable!

NON FATTIBILI

Comparison to MACs

- Public verifiability
  - Dig Sig: anyone can verify the signature
  - MAC: Only a holder of the key can verify a MAC tag
- Transferability
  - Dig Sig can forward a signature to someone else
  - MAC cannot

%

Comparison to MACs

- Nonrepudiability
  - Signer cannot (easily) deny issuing a signature
    - Crucial for legal application
    - Judge can verify signature using a copy of pK
  - MACs cannot provide this functionality
    - Without access to the key, no way to verify a tag
    - Even if receiver leaks key to judge, how can the judge verify the key is correct?
    - Even if the key is correct, receiver could have generated the tag!

Digital signatures

THE RSA SIGNATURE SCHEME

Plain RSA

- Key generation
  - (e, n) public key; (d, n) private key
- Signing operation
  - $\sigma = x^d \bmod n$
- Verification operation
  - Return ($x == \sigma^e \bmod n$)

Apr-21 Digital signatures 27

Proof of consistency was given for the cipher.

The role of encryption and decryption is swapped. This is valid only for RSA.

Properties

- Computational aspects
  - *The same considerations as PKE*
- Security
  - Algorithmic attacks
    - Factoring
  - Existential forgery
  - Malleability

The acceleration techniques can be applied to dig sig. In particular, short public keys make verification a ver fast operation. (sign once, verify many times: e.g., certificates)

Existential forgery

ABILITÀ DI GENERARE UNA FIRMA DIGITALE SENZA CHIAVE PRIVATA

UNIVERSITÀ DI PISA

- Given public key (n, e), generate a valid signature for a random message x
  - Choose a signature σ
  - Compute x = σ^e mod n
  - Output x, σ   σ È LA FIRMA DIGITALE DI x
  - Message x is random and may have no application meaning.
  - However, this property is highly undesirable

$$x^d = (\sigma^e)^d = \sigma^{e \cdot d} = \sigma \bmod n$$

Apr-21 Digital signatures 29

Malleability

- Combine two signatures to obtain a third (existential forgery)
  - Exploit the homorphic property of RSA
- The attack
  - Given $\sigma_1 = x_1^d \bmod n$
  - Given $\sigma_2 = x_2^d \bmod n$
  - Output $\sigma_3 \equiv (\sigma_1 \cdot \sigma_2) \bmod n$ that is a valid signature of $x_3 \equiv (x_1 \cdot x_2) \bmod n$
    - PROOF.

      $x_3 = \sigma_3^e \equiv (\sigma_1 \cdot \sigma_2)^e \equiv \sigma_1^e \cdot \sigma_2^e \equiv x_1^{de} \cdot x_2^{ed} \equiv x_1 \cdot x_2 \bmod n$

Remember that $x \equiv x^{ed} \bmod n$ (see proof of consistency of the cipher)

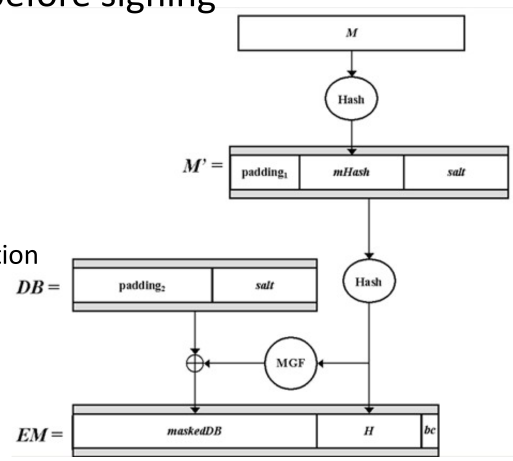
RSA Padding

- Plain RSA is never used
  - Because of existential forgery and malleability,
- Padding
  - Padding allows only certain message formats
    - It must be difficult to choose a signature whose corresponding message has that format
  - Probabilistic Signature Scheme in PKCS#1
    - Encoding Method for Signature with Appendix (EMSA)

We don’t sign the message x but the encoded message EM

PSS

- The message is encoded before signing
  - $s = EM^d \bmod n$ where
    - M = message
    - EM = encoded message
    - salt : random value
      - Makes s probabilistic
    - MGF: mask generation function
    - fixed values:
      - bc, padding1, padding2



Digital Signatures

DIGITAL SIGNATURES VS HASH FUNCTIONS

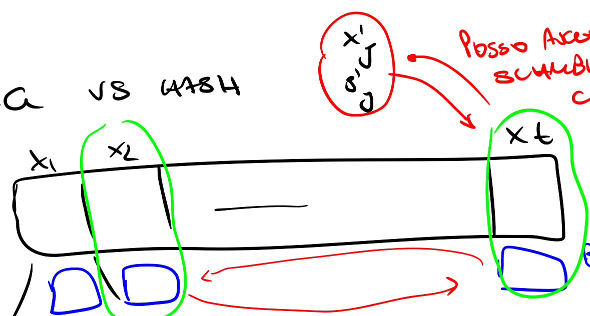
Signing long messages

- Consider RSA digsig
  - Message $0 \leq x < n$
    - E.g., n = 1024–3072 bits (128–384 bytes)
  - What if $x > n$?
  - An ECB-like approach is not recommended
    1. High-computational load (performance)
    2. Message overhead (performance)
    3. Block reordering and substitution (security)
- We would like to have a short signature for messages on any length
- The solution of this problem is hash functions

Apr-21 Digital signatures 34

DIGSIG VS HASH

Posso anche scambiare con blocchi confezionati



~ una sorta di ECB

Blocchi firmati digitalmente

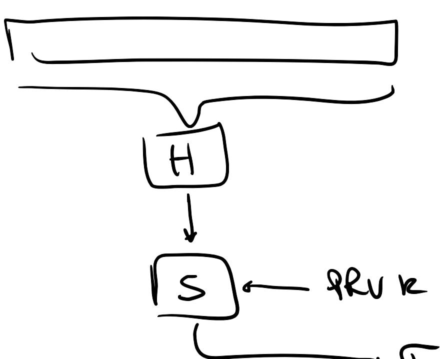
$0 \leq x_i \leq n$ → firmo digitalmente ogni messaggio molto lungo a blocchi

1) Computational: devo fare t firme digitali

2) Communication: la dimensione dei messaggi + firma digitale raddoppia

3) Security: posso scambiare blocchi senza problemi

LARGE MESSAGE



firmato digitalmente

Dig sig vs hash properties

- Hash functions properties
  - Pre-image resistance
  - Second pre-image resistance
  - Collision resistance
- These properties are crucial for digital signatures security

Dig sig vs hash properties

$A \xrightarrow{x, s} B$

- Pre-image Resistance $t = H(x)$, $s = S(prvk_A, t)$
  - Digital signature scheme based on (school-book) RSA
    - (n, d) is Alice's private key;
    - (n, e) is Alice's public key
    - $s = (H(x))^d \pmod n$
  - If H is not pre-image resistant, then existential forgery is possible
    - Select z < n
    - Compute $y = z^e \pmod n$
    - Find x' such that H(x') = y (🡐)
    - Claim that z is the digital signature of m' Q.E.D

Dig sig vs hash properties

UNIVERSITÀ DI PISA

- 2nd preimage resistance
  - The protocol
    - Bob → Alice: x
    - Alice → Bob: x, s = S(privK_A, H(x))
  - If H is not 2nd-preimage resistant, the following attack is possible
    - An adversary (e.g., Alice herself) can determine a 2nd-preimage x' of x and then (🡨) and then
    - claim that Alice has signed x’ instead of x Q.E.D

Dig sig vs hash properties

- Collision-resistance
  - If H is not collision resistant, the following attack is possible
    - Alice chooses x and x’ s.t. H(x) = H(x’) (🡐)
    - computes s = S(privK$_A$, H(x))
    - Sends (x, s) to Bob
    - later claims that she actually sent (x', s) Q.E.D

Hash-and-Sign paradigm

- Given a signature scheme Σ = (G, S, V) for "short" messages of length n
- Given a Hash function H: $\{0, 1\}^* \rightarrow \{0, 1\}^n$
- Construct a signature scheme Σ' = (G, S', V') for messages of any length
  - σ = S'(privK, m) = S(privk, H(m))
  - V'(m, pubK, σ) = V(H(m), pubK, σ)

Hash-and-sign paradigm

UNIVERSITÀ DI PISA

- THM. If Σ is secure and H is collision-resistant then Σ' is secure
  - Proof (by contradiction)
    - Assume that the sender authenticates m_1, m_2,…and manages to forge (m’, σ’), m’ ≠ m_i, for all i
    - Let h_i = H(mi). Then, we have two cases
      - If H(m’) = h_i for some i, then collision in H (contradiction)
      - If H(m') ≠ hi, for all i, then forgery in Σ (contradiction)

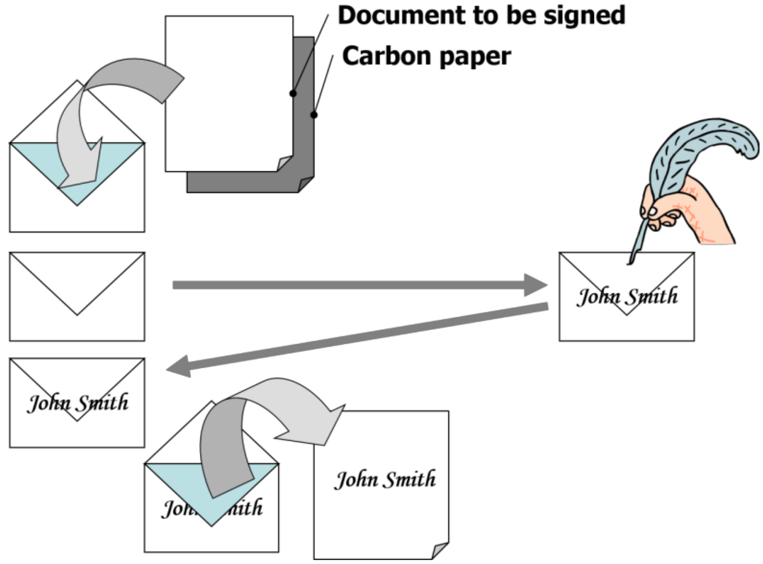
Digital signatures

RSA-BASED BLIND SIGNATURES (NO)

Blind signatures

- Intuition
  - In a blind signature scheme, the signer can’t see what it is signig
- Unlinkabiliy
  - The signer is not able to link the signature to the act of signing

The metaphor



Blind signatures

- The protocol
  - Alice
    - Randomly chooses b s.t. gcd(b, n) = 1
    - Computes $x' \equiv x \cdot b^e \pmod n$
    - Sends x' to Bob (signer)
    - Inviare m' al signer
  - Bob
    - Receives x'
    - Computes $s' \equiv (x')^d \pmod n$
    - Returns s' Alice

Blind signatures →

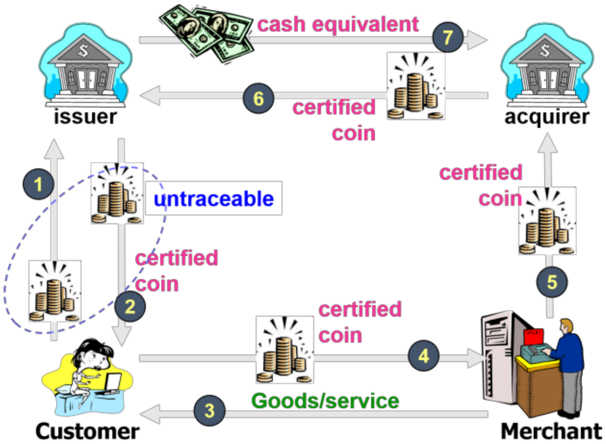
UNIVERSITÀ DI PISA

- The protocol
  - Alice
    - Receives s'
    - Computes $s \equiv s' \cdot b^{-1} \pmod{n}$
      - s is digital signature of s
- Proof
  - $s' \cdot b^{-1} \equiv (x')^d \cdot b^{-1} \equiv (x \cdot b^e)^d \cdot b^{-1} \equiv x^d \cdot b^{ed} \cdot b^{-1} \equiv$
    $\equiv x^d \cdot b \cdot b^{-1} \equiv x^d \equiv s \bmod n$ QED

Applications

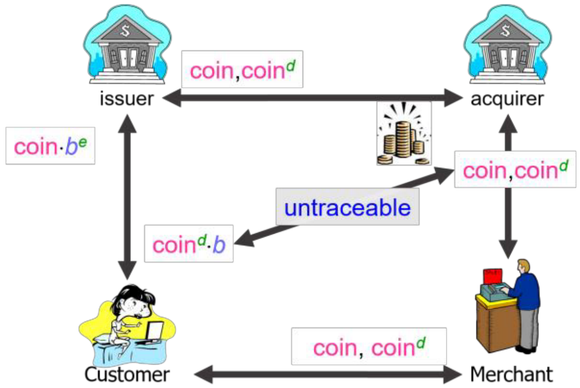
- Privacy related applications
  - Digital cash (David Chaum, 1983)
  - Electronic voting

Digital cash



- coin: a random number
- coin·b^e: blinded coin
- coin, $coin^d$: certified coin
- $d_{10€}$: a 10€ worth bank's private key

Digital cash



- coin: a random number
- coin·b^e: blinded coin
- coin, coin^d: certified coin
- $d_{10€}$: a 10€ worth bank's private key

Double spending →

- The protocol does not prevent
  - the customer from spending the digital coin multiple times
  - The merchant from depositing the digital coin multiple times
- Partial countermeasure
  - The issuer maintains the list of spent digital coins
    - Protect the bank from frauds
    - Don’t allow issuer to identify the fraudster

Double spending

- Purely criptographic solution based on
  - Secret splitting
  - Bit commitment
  - Cut-and-choose
- Inefficient but great impulse to cryptography

Digital signatures

THE ELGAMAL SIGNATURE SCHEME

Elgamal in a nutshell

UNIVERSITÀ DI PISA

- Invented in 1985
- Based on difficulty of discrete logarithm
- Digital signature operations are different from the cipher operations

Key generation

- Choose a large prime *p*
- Choose a primitive element α of (a subgroup of) Zp*
- Choose a random number $d \in \{2, 3, \ldots, p - 2\}$
- Compute $\beta = \alpha^d \bmod p$
- pubK = (p, α, β) is the public key and
- privK = d *is* the private key

Signature generation

- Message x
- Choose an ephemeral key k_E in {0, 1, 2, p – 2} such that $\gcd(k_E, p-1) = 1$
- Compute the signature parameters
  - $r \equiv \alpha^{kE} \bmod p$
  - $s \equiv (x - d \cdot r)k_E^{-1} \bmod p - 1$
  - (r, s) is the digital signature
- Send $\langle x, (r, s)\rangle$

Signature verification

- Verification of $\langle x, (r, s)\rangle$
- Compute $t \equiv \beta^r \cdot r^s \bmod p$
- If ($t \equiv \alpha^x \bmod p$) ➔ valid signature;
  otherwise ➔ invalid signature

Proof

1. Let $t \equiv \beta^r \cdot r^s \equiv (\alpha^d)^r (\alpha^{kE})^s \equiv \alpha^{d \cdot r + kE \cdot s}$ mod p
2. If $\beta^r \cdot r^s \equiv \alpha^x$ mod p then $\alpha^x \equiv \alpha^{d \cdot r + kE \cdot s}$ mod p [a]
3. According to Fermat's Little Theorem Eq.[a] holds if $x \equiv d \cdot r + k_E \cdot s$ mod p – 1
4. from which the construction of parameter $s = (x - d \cdot r) k_E^{-1}$ mod p – 1

Computational aspects

- Key generation
  - Generation of a large prime (1024 bits)
  - True random generator for the private key
  - Exponentiation by square-and-multiply
- Signature generation
  - $| s | = | r | = | p |$ thus $|x, (r, s)| = 3 \, | x |$ (*msg expansion*)
  - One exponentiation by square-and-multiply
  - One inverse k_E^{-1} mod p by EEA (pre-computation)
- Signature verification
  - Two exponentiations by square-and-multiply
  - One multiplication

Security aspects

- The verifier must have the correct public key
- The DLP must be intractable
- Ephemeral key cannot be reused
  - If k_E is reused the adversary can compute the private key *d* and impersonate the signer
- Existential forgery for a random message x unless it is hashed

Reuse of ephemeral key

- If the ephemeral key k_E is reused, an attacker can easily compute the private key d
  - Proof
    - Message x_1 and x_2 and the reused ephemeral key k_E reused
    - $(x_1, (s_1, r))$ and $(x_2, (s_2, r))$ where
      - $r \equiv \alpha^{kE} \bmod p$
      - $s_1 \equiv (x_1 - d \cdot r) \cdot k_E^{-1} \bmod p - 1$ [a]
      - $s_2 \equiv (x_2 - d \cdot r) \cdot k_E^{-1} \bmod p - 1$ [b]
        - » [a], [b] is a system in two unknowns and two equations
      - $s_1 - s_2 \equiv (x_1 - x_2) \cdot k_E^{-1} \bmod p - 1$
      - $k_E \equiv (x_1 - x_2) \cdot (s_1 - s_2)^{-1} \bmod p - 1$
      - $d \equiv (x_1 - s_1 \cdot k_E) \cdot r^{-1} \bmod p - 1$ Q.E.D.

Existential Forgery Attack

- The attack

Alice **Adversary** **Bob**

privK = d, pubK = (p, α, β)

< ---------------(p, α, β)-------------------

1. select i, j, s.t. gcd(j, p – 1) = 1
2. compute the signature
   $r \equiv \alpha^{i} \cdot \beta^{j} \bmod p$
   $s \equiv -r \cdot j^{-1} \bmod p - 1$
3. compute the message
   $x \equiv s \cdot i \bmod p - 1$

verification < -----------------(x, (r, s))---------------

$t \equiv \beta^{r} \cdot r^{s} \bmod p$ since

$t \equiv \alpha x \bmod p$ ➔ valid signature!

Existential forgery

- Proof

  $t \equiv \beta^r \cdot r^s \equiv (\alpha^d)^r \cdot (\alpha^i \cdot \beta^j)^s \equiv (\alpha^d)^r \cdot (\alpha^i \cdot \alpha^{d \cdot j})^s \equiv \alpha^{d \cdot r} \cdot (\alpha^{i+d \cdot j})^s$

  $\equiv \alpha^{d \cdot r} \cdot (\alpha^{i+d \cdot j})^s \equiv \alpha^{d \cdot r} \cdot \alpha^{(i+d \cdot j) \cdot (-r \cdot j^{-1})} \equiv$

  $\equiv \alpha^{d \cdot r} \cdot \alpha^{-d \cdot r} \cdot \alpha^{-r \cdot i \cdot j^{-1}} \equiv \alpha^{s \cdot i} \bmod p$ [a]

  - As the message was constructed as $x \equiv s \cdot i \bmod p$ then equation [a] $\alpha^{s \cdot i} \equiv \alpha^x \bmod p$ which is the condition to accept the signature as valid
  - The adversay computes in step 3 the message x whose semantics (s) cannot control
  - The attack is not feasible if the message is hashed
    - $s \equiv (H(x) - d \cdot r) k_E^{-1} \bmod p - 1$

Digital Signatures

DIGITAL SIGNATURE ALGORITHM (DSA)

Introduction

UNIVERSITÀ DI PISA

- The Elgamal scheme is rarely used in practice
- DSA is a more popular variant
  - It's a federal US government standard for digital signatures (DSS)
  - It was proposed by NIST
- Advantages of DSA w.r.t. Elgamal
  - Signature is only 320 bits
  - Some attacks against to Elgamal are not applicable to DSA

Key Generation

1. Generate a prime p with $2^{1023} < p < 2^{1024}$.
2. Find a prime divisor q of p−1 with $2^{159} < q < 2^{160}$.
3. Find an element α with ord(α) = q, i.e., α generates the subgroup with q elements.
4. Choose a random integer d with 0 < d < q.
5. Compute $\beta \equiv \alpha^d \bmod p$.
6. The keys are now:
   1. pubK = (p,q,α,β)
   2. privK = (d)

Central idea

- DSA uses two cyclic groups
  - Zp*, the order of which has bit lenght 2014 bit
  - 160-bit subgroup of Zp*
  - This setup yields shorter signatures
- Other combinations are possible

p	q	signature
1024	160	320
2048	224	448
3072	256	512

Signature Generation

1. Choose an integer as random ephemeral key k_E with $0 < k_E < q$.
2. Compute $r \equiv (\alpha^{kE} \bmod p) \bmod q$.
3. Compute $s \equiv (SHA(x) + d \cdot r)k_E^{-1} \bmod q$.
   – SHA-1(·) produces a 160-bit value
4. Digital signature is the pair (r, s)
   – 160 + 160 = 320 bit long

Signature Verification

1. Compute auxiliary value $w \equiv s^{-1} \bmod q$.
2. Compute auxiliary value $u_1 \equiv w \cdot SHA(x) \bmod q$.
3. Compute auxiliary value $u_2 \equiv w \cdot r \bmod q$.
4. Compute $v \equiv (\alpha^{u1} \cdot \beta^{u2} \bmod p) \bmod q$.
5. The verification follows from:
    1. If $v \equiv r \bmod q$ ➔ valid signature
    2. Otherwise ➔ invalid signature

Proof %

- We show that a signature (r, s) satisfies the verification condition v ≡ r mod q.
  - $s \equiv (SHA(x)+d\ r)k_E^{-1}$ mod q which is equivalent to $k_E \equiv s^{-1}$ SHA(x)+d s^{-1} r mod q.
  - The right-hand side can be expressed in terms of the auxiliary values u1 and u2: $k_E \equiv u_1+d\ u_2$ mod q.
  - We can raise α to either side of the equation if we reduce modulo p: α^{kE} mod p ≡ $\alpha^{u1 + d\ u2}$ mod p.

Proof

- Since the public key value β was computed as $\beta \equiv \alpha^{d} \bmod p$, we can write: $\alpha^{kE} \equiv \alpha^{u1} \beta^{u2} \bmod p$.
- We now reduce both sides of the equation modulo q: $(\alpha^{kE} \bmod p) \bmod q \equiv (\alpha^{u1}\beta^{u2} \bmod p) \bmod q$.
- Since r was constructed as $r \equiv (\alpha^{kE} \bmod p) \bmod q$ and $v \equiv (\alpha^{u1}\beta^{u2} \bmod p) \bmod q$,
- this expression is identical to the condition for verifying a signature as valid:
  - $r \equiv v \bmod q$.

Computational aspects %

- Key Generation
  - The most challenging phase
    - Find a Z_p* with 1024-bit prime *p* and a subgroup in the range of 2^{160}
      - This condition is fulfilled if $|Zp^*| = |p – 1|$ has a prime factor q of 160 bit
  - General approch:
    - To find q first and then p

Computational aspects %

- Signing
  - Computing r requires exponentiation
    - Operands are on 1024 bit
    - Exponent q is on 160 bit
      - On average 160 + 80 = 240 SQs and MULTs
    - Result is reduced mod q
    - Does not depend on x so can be precomputed
  - Computing s
    - Involve 160-bit operands
    - The most costly operation is inverse

Computational aspects

- Verification
  - Computing the auxiliary parameters w, u_1 and u_2 involves 160-bit operands
  - This is relatively fast

Security

- We have to protect from two different DLPs
  1. d = $\log_{\alpha} \beta$ mod p.
     - Index calcolus attack
       - Prime p must be on 1024 bits for 80-bit security level
  2. α generates a subgroup a subgroup of order q
     - Index calculus attack cannot be applied
     - Only generic DLP attacks can be used
       - Square-root attacks: Baby-step giant-step, Pollard's rho
       - Running time: $\sqrt{q} = \sqrt{2^{160}} = 80$
- Vulerable to k_E reuse
  - Analalogue to ElGamal

Apr-21 Digital signatures 74