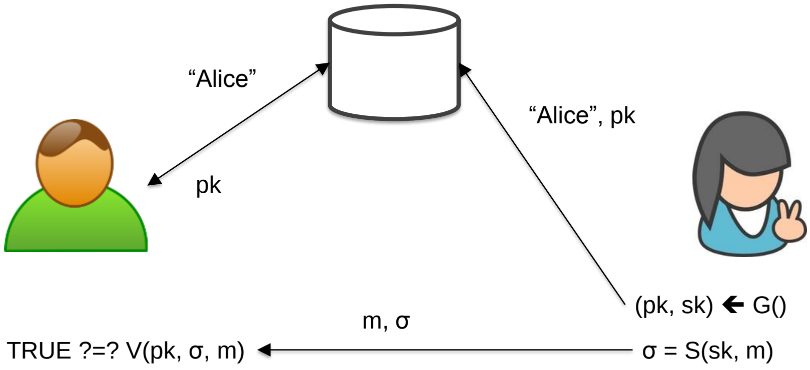
Digital signatures

Digital signatures

UNIVERSITÀ DI PISA

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

Communication model

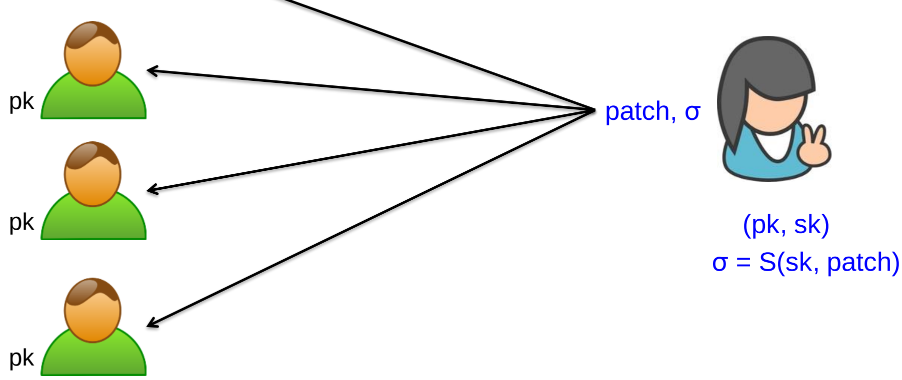


Security

- **DEF (informal)**. Even after observing signatures on multiple messages, an attacker should be unable to *forge* a valid signature on a *new* message

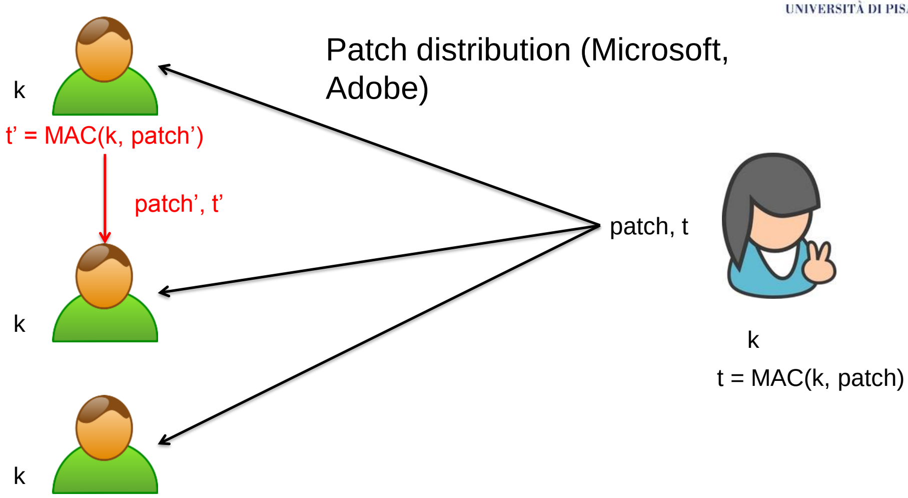
Prototypical application

Patch distribution (Microsoft, Adobe)



Comparison to MACs

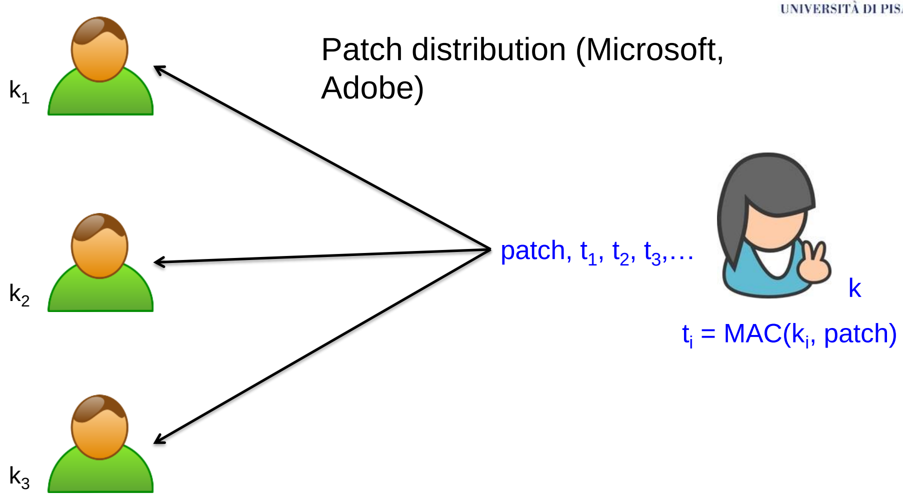
Patch distribution (Microsoft, Adobe)



Comparison to MACs

UNIVERSITÀ DI PISA

Patch distribution (Microsoft, Adobe)



Comparison to MACs

UNIVERSITÀ DI PISA

- Single shared key k
  - A client may forge the tag
  - Unfeasible if clients are not trusted
- Point-to-point key k_i
  - Computing and network overhead
  - Prohibitive key management overhead
  - Unmanageable!

Comparison to MACs

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

Non-repudiation

- 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!

Informal properties

UNIVERSITÀ DI PISA

- **DEF.** A digital signature is a number dependent on some secret known only to the signer and, additionally, on the content of the message being signed
- **Property.** A digital signature must be verifiable
  - If a dispute arises an unbiased third party must be able to solve the dispute equitably, without requiring access to the signer's secret

9 maggio 2018 Digital signatures 11

Digital signature scheme

UNIVERSITÀ DI PISA

- A **signature scheme** is defined by three PPT algorithms (G, S, V):
- **Key generation algorithm** G takes as input 1^n and outputs (pk, sk)
- **Signature generation algorithm** S takes as input a private key sk and a message m and outputs a signature σ = S(sk, M)
- **Signature verification algorithm** V takes as input a public key pk, a signature σ and (optionally) a message m and outputs True o False
- **Consistency**. For all m and (pk, sk), V(pk, [m], S(sk, m)) = TRUE

9 maggio 2018 Digital signatures 12

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 gets the public key
- **Security goal**
  - **Existential unforgeability**
    - Attacker should be *unable* to forge valid signature on *any* message not signed by the sender

Digital signatures

THE RSA SIGNATURE SCHEME

Plain RSA

- **Key generation**
  - (e, n) public key; (d, n) private key
    - *Same algorithm ad PKE*
- **Signing operation**
  - $\sigma = m^d \bmod n$
- **Verification operation**
  - $m == \sigma^e \bmod n$

Properties

- Computational aspects
  - *The same considerations as PKE*
  - The re-blocking problem
- Security
  - Algorithmic attacks
  - Existential forgery
  - Malleability

The re-blocking problem

- The problem (theoretical)
  - If Alice wants to send a secret and signed message to Bob then it must be $n_A < n_B$
- Possible solutions
  - **Reordering**: the operation with the smaller modulus is performed first
    - CONS: The preferred order is always to sign first and encrypt later
  - **Two moduli for every entity**
    - Every entity has two moduli
    - Moduli for signing (e.g., t-bits) is always smaller of all possible moduli for encryption (e.g., t+1-bits)

Algorithmic attacks

- The verifier must have the correct public key
- Attempt to break RSA by computing d
  - The most general attack tries to factor modulus n
  - The modulus must be sufficient large (1024 bits or more are recommended)

Existential forgery

- Generate a valid signature for a random message x
  - Given Alice's public key (n, e)
  - Choose a signature σ
  - Compute $x = \sigma^e \bmod n$
  - Output x, σ
  - Message m is random and may have no application meaning. However, this property is undesirable

Malleability

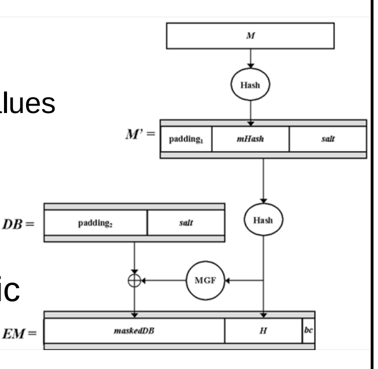
- **Goal**. Combine two signatures to obtain a third (existential forgery)
- **Attack**
  - Given $\sigma_1 = m_1^d \bmod n$
  - Given $\sigma_2 = m_2^d \bmod n$
  - Output $\sigma_3 = (\sigma_1 \bullet \sigma_2) \bmod n$ that is a valid signature of $m_3 = (m_1 \bullet m_2) \bmod n$
    - PROOF.
      - $\sigma_3^e = (\sigma_1 \bullet \sigma_2)^e = \sigma_1^e \bullet \sigma_2^e = m_1 \bullet m_2 \bmod n$

RSA Padding

- Because of existential forgery and malleability, plain RSA is never used
- Padding scheme allows only certain message formats
  - It must be difficult to choose a signature whose corresponding message has that format
- Padding schemes
  - Probabilistic Signature Scheme (PSS) in PKCS#1
  - Full Domain Hash (RSA-FDH)
  - ISO/IEC 9776

Probabilistic Signature Standard (PSS)

- The message is encoded before signing
  - M = message
  - EM = encoded message
  - Salt : random value
  - MGF: mask generation function
  - bc, $padding_1$, $padding_2$: fixed values
  - s = EM^d mod n
- PROS
  - Verifiable secure
  - Salting makes EM probabilistic



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 if Z_p^*
- Choose a random number d in $\{2, 3, \ldots, p - 2\}$
- Compute $\beta = \alpha^d \bmod p$
- Let (p, α, β) be the public key and d the private key

Signature generation

- Digital signature of message x
- Choose an ephemeral key ke in $\{0, 1, 2, p-2\}$ such that $\gcd(ke, p - 1) = 1$
- Compute the signature parameters
  - $r = \alpha^{ke} \bmod p$
  - $s = (x - d \bullet r) ke^{-1} \bmod p - 1$
  - (r, s) is the digital signature
- Send x, (r, s)

Signature verification

- Upon verification of x, (r, s)
- Compute $t = \beta^{r} \bullet r^{s}$
- If $t = \alpha^{x} \bmod p$ ➔ valid signature; otherwise invalid signature

Proof

1. Let $\beta^{r} \bullet r^{s} = (\alpha^{d})^{r}(\alpha^{ke})^{s} = \alpha^{d \bullet r + ke \bullet s} \bmod p$
2. If $\beta^{r} \bullet r^{s} = \alpha^{x} \bmod p$ then $\alpha^{x} = \alpha^{d \bullet r + ke \bullet s} \bmod p$
3. According to Fermat's little theorem Eq.2 holds if $x = d \bullet r + ke \bullet s \bmod p - 1$
4. From which the construction of parameter $s = (x - d \bullet r)ke^{-1} \bmod p - 1$

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 ke^{-1} mod p by extended Eulero algorithm (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 ke 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

The Digital Signature Algorithm (DSA)

- 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 w.r.t. Elgamal
  - Signature is only 320 bits
  - Some attacks against to Elgamal are not applicable to DSA

Elliptic Curve DSA (ECDSA)

- ECDSA was standardized in US by ANSI in 1998
- **Pros**
  - ECC allow 160-256-bit lengths which provide security equivalent to 1024-3072-bit RSA/DL
- **Cons**
  - Finding EC with good cryptographic properties in nontrivial
  - Standardize curves by NIST or Brainpool consortium

Digital Signatures

HASH FUNCTIONS

Properties

UNIVERSITÀ DI PISA

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

Pre-image resistance

- Digital signature scheme based on (school-book) RSA
  - (n, d) is a Alice's private key;
  - (n, e) is a Alice's public key
  - $\sigma = (h(m))^d \pmod n$
- **THM -** If h() is not pre-image resistant => ***existential forgery***
  - Select $z < n$
  - Compute $y = z^e \pmod n$
  - Find m' such that $h(m') = y$
  - Claims that z is the digital signature of m'

2nd preimage resistance

- Let (G, S, V) be a signature scheme
- A trusted third party chooses a message x that Alice signs producing $s = S(d_A, h(x))$
- If h() is not 2nd-preimage resistant, an adversary (e.g. Alice herself) can claim that Alice has signed x' instead of x
  - Adversary determines a 2nd-preimage x' of x
  - Adversary claims that Alice has signed x' instead of x

Collision resistance

- Let (G, S, V) be a signature scheme
- If h() is not collision resistant, Alice (an untrusted party) can
  - choose x and x' so that h(x) = h(x')
  - compute $s = S(d_A, h(x))$
  - Issue (m, s) to Bob
  - later claim that she actually issued (x', s)

Digital signatures

NON-REPUDIATION VS AUTHENTICATION

Non-repudiation vs authentication

UNIVERSITÀ DI PISA

- **DEF.** Non-repudiation prevents a signer from signing a document and subsequently being able to successfully deny having done so.
- **Non-repudiation vs authentication of origin**
  - **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 $t0$
  - **Non-repudiation** (based on public-key cryptography) allows a party to convince others at any time $t1 \geq t0$ of the integrity/authenticity of a given message at time $t0$

Dig sig vs non-repudiation

UNIVERSITÀ DI PISA

- Alice's digital signature for a given message depends on the message and a secret known to Alice only (the private key)
- Bob verifies the digital signature by means of another, different value: the public key

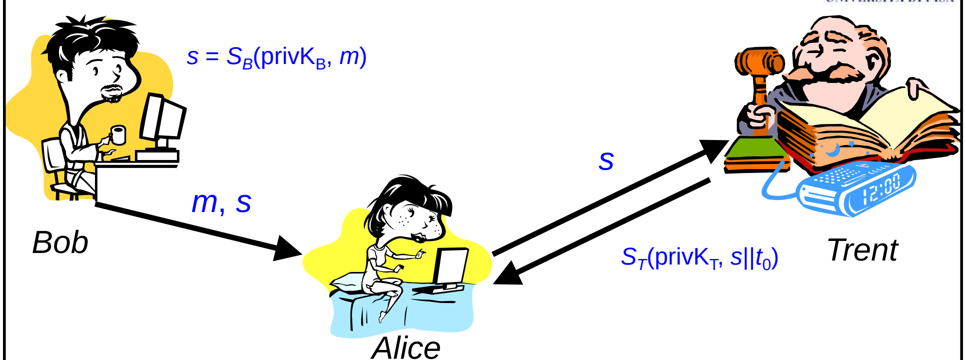
Dig sig vs non-repudiation

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
  - Use of a trusted timestamp agent
  - Use of a trusted notary agent

Trusted timestamping service

UNIVERSITÀ DI PISA



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

Trusted Notary Service

UNIVERSITÀ DI PISA

- TNS generalize the TTS
  - Trent certifies that a certain statement σ on the digital signature s (is true at t_0
    - s exists at t_0
    - s is valid at t_0
  - Trent may certify the existence of a certain document doc
    - s = S(privK_T, H(*doc*) || *timestamp*)
    - Document *doc* remains secret
- Trent is trusted to verify the statement before issuing it

9 maggio 2018 Digital signatures 43

Digital signatures

SOME ADVANCED CONCEPTS

9 maggio 2018 Digital signatures 44

Classification

- **Dig sig with message recovery**
  - does not require the original message as input to the verification algorithm. In this case, the original message is recovered from the signature itself
  - Examples: RSA, Rabin, Nyberg-Rueppel
- **Dig sig with appendix**
  - requires the original message as input to the verification algorithm
  - uses hash functions
  - Examples: ElGamal, DSA, DSS, Schnorr

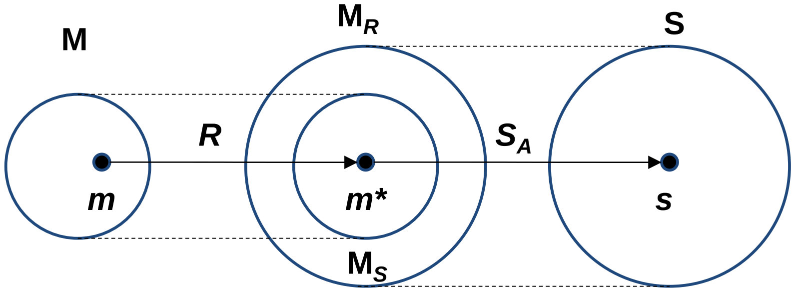
RSA-based dig sig

- Digital signature with message recovery
  - Redundancy function
    - A suitable redundancy function is necessary in order to avoid existential forgery
    - **IOS/IEC 9796** (1991) defines a mapping that takes a k-bit integer and maps it into a 2k-bits integer
- Digital signature scheme with appendix
  - MD5 (128 bit)
  - **PKCS#1** specifies a redundancy function mapping 128-bit integer to a k-bit integer, where k is the modulus size (k>512, k = 768, 1024)

Dig sig with message recovery (1)

- **Definitions**
  - **M** is the message space
  - $\mathbf{M_S}$ is the signing space
  - **S** is the signature space
- **Key generation**
  - **A** selects a private key d_A defining a ***signing algorithm*** S_A which is a one-to-one mapping S_A: $M_S \rightarrow S$
  - **A** defines the corresponding public key defining the ***verification algorithm*** V_A such that $V_A \times S_A$ is identity map on M_S.

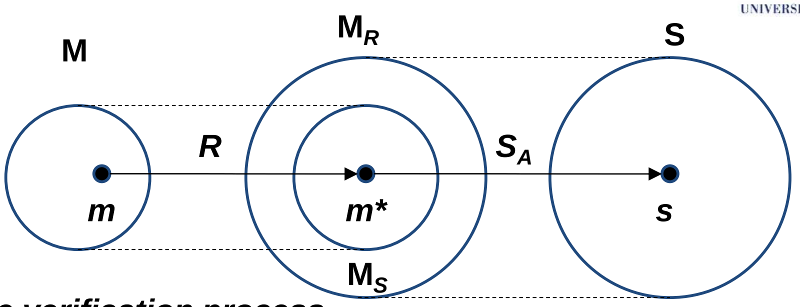
Dig sig with message recovery (2)



The signing process

- Compute ***m* = R(m)***, ***R*** is a ***redundancy function*** (invertible)
- Compute $s = S_A(m^*)$

Dig sig with message recovery (3)



The verification process

- Obtain authentic public key V_A
- Compute $m^* = V(s)$

► Verify if $m^* \in M_S$ (if not, reject the signature)

- Recover the message $m = R^{-1}(m^*)$

Dig sig with message recovery (4)

- ***Properties of S_A and V_A***
- (**efficiency**) S_A should be efficient to compute
- (**efficiency**) V_A should be efficient to compute
- (**security**) It should be **computationally infeasible** for an entity other than A to find an s $\in$S such that $V_A(s) \in M_S$

Dig sig with message recovery (5)

- **The redundancy function**
  - R and R^{-1} are publicly known
  - Selecting an appropriate R is critical to the security of the system
- **A bad redundancy function may lead to existential forgery**
  - Let us suppose that $M_R \equiv M_S$
  - R and S_A are bijections, therefore M and S have the same number of elements
  - Therefore, for all $s \in S$, $V_A(s) \in M_R$. Hence, it is "easy" to find an m for which s is the signature, $m = R^{-1}(V_A(s))$
  - s is a valid signature for m (**existential forgery**)
  - **Plain RSA dig sig suffers from existential forgery**

Dig signatures with message recovery (6)

- **A good redundancy function although too redundant**
  - Example
    - $M = \{m : m \in \{0, 1\}^n\}$, $M_S = \{m : m \in \{0, 1\}^{2n}\}$
    - R: $M \rightarrow M_S$, $R(m) = m||m$ (concatenation)
    - $M_R \subseteq M_S$
    - When *n* is large, $|M_R|/|M_S| = (1/2)^n$ is small. Therefore, for an adversary it is unlikely to choose an s that yields $V_A(s) \in M_R$

Redundancy function for RSA

- **ISO/IEC 9776** is an international standard that defines a redundancy function for **RSA** and **Rabin**
- Multiplicative property(*) of RSA
  - **Requirement on R**: a ***necessary condition*** for avoiding existential forgery is that **R** must not satisfy the multiplicative property.

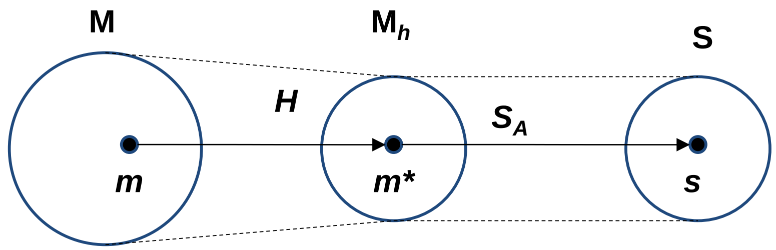
(*) Homomorphism property

Dig sig with appendix (1)

- **Definitions**
  - *M* is the message space
  - *H* is a hash function with domain *M*
  - M_h is the image of *h*
  - *S* is the signature space
- **Key generation**
  - Alice selects a private key $\boldsymbol{d_A}$ which defines a ***signing algorithm*** $\boldsymbol{S_A}$ which is a **one-to-one** mapping $S_A: M_h \rightarrow S$
  - Alice defines the corresponding public key $\boldsymbol{e_A}$ defining the ***verification algorithm*** $\mathbf{V_A}$ such that $\mathbf{V_A(m^*, s) = true}$ if $\mathbf{S_A(m^*) = s}$ and false otherwise, for all $m^* \in M_h$ and $s \in S$, where $\mathbf{m^* = H(m)}$ for $m \in M$.

Dig sig with appendix (2)

UNIVERSITÀ DI PISA

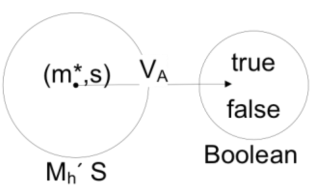


Signature generation process

- Compute **$m^* = h(m)$**, **$s = S_A(m^*)$**
- Send **(m, s)**

Dig sig with appendix (3)

UNIVERSITÀ DI PISA



- **Verification process**
  - Obtain A's public key **V_A**
  - Compute **$m^* = H(m)$**, **$u = V_A(m^*, s)$**
  - Accept the signature iff **u == true**

Dig sig with appendix (4)

- ***Properties of S_A and V_A***
  - (**efficiency**) S_A should be efficient to compute
  - (**efficiency**) V_A should be efficient to compute
  - (**security**) It should be **computationally infeasible** for an entity other than *A* to find an $m \in M$ and an $s \in S$ such that $V_A(m^*, s) = true$, where $m^* = h(m)$

Dig sig with appendix from message recovery

- **Signature generation**
  - Compute $m^* = R(h(m))$, $s = S_A(m^*)$
  - *A*'s digital signature for *m* is *s*
  - *m*, *s* are made available to anyone who may wish to verify the signature
- **Signature verification**
  - Obtain A's public key VA
  - Compute $m^* = R(h(m))$, $m' = V_A(s)$, and $u = (m' == m^*)$
  - Accept the signature iff u = true
- **Comment**
  - R is not security critical anymore and can be any one-to-one mapping

Hash-and-sign paradigm

UNIVERSITÀ DI PISA

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

9 maggio 2018 Digital signatures 59

Hash-and-sign paradigm

UNIVERSITÀ DI PISA

- **THM**. If π is secure and H is collision-resistant then π' is secure
  - **Proof (by contradiction)**
  - Let us assume that the sender authenticates $m_1, m_2, \ldots$ and the adversary manages to forge (m', σ'), $m' \neq m_i$, for all i
  - Let $h_i = H(m_i)$. Then, we have two cases
    - If $H(m') = h_i$ for some i, then collision in H (contradiction)
    - If $H(m') \neq h_i$, for all i, then forgery of π (contradiction)

9 maggio 2018 Digital signatures 60