Hash functions and data integrity

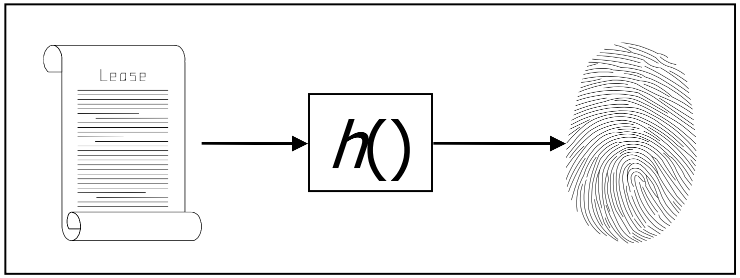
- Manipulation Detection Code (MDC)
- Message Authentication Code (MAC)
- Data integrity and origin authentication

Data integrity and data origin authentication

- **Message integrity** is the property whereby data has not been altered in an *unauthorized* manner since the time it was created, transmitted, or stored by an *authorized* source
- **Message origin authentication** is a type of authentication whereby a party is corroborated as the (original) source of specified data created at some time in the past
- **Data origin authentication includes data integrityand vice versa**

Hash function: informal properties

- The hash (fingerprint, digest) of a message must be
  - "easy" to compute
  - "unique"
  - "difficult" to invert



- The hash of a message can be used to
  - guarantee the integrity and authentication of a message
  - "uniquely" represent the message

Hash function

Nel mezzo del cammin di nostra vita
mi ritrovai per una selva oscura
che' la diritta via era smarrita.

Ahi quanto a dir qual era e` cosa dura
esta selva selvaggia e aspra e forte
che nel pensier rinova la paura!

↓

MD5

↓

d94f329333386d5abef6475313755e94

128 bit

The hash size is fixed, generally smaller than the message size

Basic properties

- A hash function maps bitstrings of arbitrary, finite length into bitstrings of fixed size

$$h \longrightarrow$$

- A hash function is a function *h* which has, as minumum, the following properties
  - **Compression** – *h* maps an input *x* of arbitrary finite lenth to an output *h*(*x*) of fixed bitlength *m*
  - **Ease of computation** – given an input *x*, *h*(*x*) is easy to compute
- A hash function is **many-to-one** and thus implies **collisions**

Additional security properties (MDC)

A **hash function** may have one or more of the following additional security properties

- **Preimage resistance** (**one-way**) – for essentially all pre-specified outputs, it is computationally infeasible to find any input which hashes to that output, i.e., to find x such that $y = h(x)$ given y for which x is not known

- **2nd-preimage resistance** (**weak collision resistance**) – it is computationally infeasible to find any second input which has the same output as any specified input, i.e., given x, to find $x' \neq x$ such that $h(x) = h(x')$

- **Collision resistance** (**strong collision resistance**) – it is computationallyinfeasible to find any two distinct inputs x, x' which hash to the same output, i.e., such that $h(x) = h(x')$

Motivation of properties

- *2nd-preimage resistance*
  - **Digital signature with appendix (*S*, *V*)**
    - ***s* = *S*(*h*(*m*))** is the digital signature for m
  - A trusted third party chooses a message m that Alice signs producing $s = S_A(h(m))$
  - If h is not 2nd-preimage resistant, an adversary (e.g. Alice herself) can
    - determine a 2nd-preimage m' such that h(m') = h(m) and
    - claim that Alice has signed m' instead of m

Motivation of properties

- *Collision resistance*
  - Digital signature with appendix (S, V)
    - s = S(h(m)) is the digital signature for m
  - If h() is not collision resistant, Alice (an untrusted party) can
    - choose m and m' so that h(m) = h(m')
    - compute $s = S_A(h(m))$
    - issue ⟨m, s⟩ to Bob
    - later claim that she actually issued ⟨m', s⟩

Motivation of properties

- *Preimage resistance*
  - Digital signature scheme based on ***RSA***:
    - (n, d) is a private key; (n, e) is a public key
    - A digital signature s for m is $\boldsymbol{s = (h(m))^d \bmod n}$
  - If h is not preimage resistance an adversary can
    - select $z < n$, compute $y = z^e \bmod n$ and find m' such that $h(m') = y$;
    - claim that z is a digital signature for m' (***existential forgery***)

MDC classification

- A **one-way hash function** (**OWHF**) is a hash function *h* with the following properties:
    - preimage resistance
    - 2-nd preimage resistance
- OWHF is also called weak one-way hash function
- A **collision resistant hash function** (**CRHF**) is a hash function *h* with the following properties
    - 2-nd preimage resistance
    - collision resistance
- CRHF is also called strong one-wayhash function

Relationship between properties

- **Collision resistance implies 2-nd preimage resistance**
- **Collision resistance does not imply preimage resistance**
    - However, **in practice**, **CRHF almost always** has the additional property of **preimage resistance**

Objective of adversaries vs MDC

- ***Attack to a OWHF***
  - given a hash value y, find a preimage x such that y = h(x); or
  - given a pair (x, h(x)), find a second preimage x' such that h(x) = h(x')
- ***Attack to a CRHF***
  - find any two inputs x. x', such that h(x) = h(x')

Hash type	Design goal	Ideal strength
OWHF	preimage resistance 2nd-premage resistance	2^m 2^m
CRHF	collisione resistance	$2^{m/2}$

Severity of practical consequences of an attack

- **Severity of practical consequences of an attack depends on the degree of control an adversary has over the message *x*** for which an MDC may be forged
- **selective forgery**: the adversary has complete or partial control over x
- **existential forgery**: the adversary has no control over ***x***

Algorithm independent attacks

- Assumptions
  1. Treat an hash functions as a "***black box***";
  2. Only consider the ***output bitlength m***;
  3. ***hash approximates a random variable***
- Specific attacks
  - **Guessing attack**: find a preimage ($O(2^m)$)
  - **Birthday attack**: find a collision ($O(2^{m/2})$)
  - **Precomputation of hash values**: if ***r*** pairs of a OWHF are precomputed and tabulated the probability of finding a second preimage increases to ***r*** times its original value
  - **Long-message attack for 2nd preimage**: for "long" messages, a 2nd preimage is generally easier to find than a preimage

Guessing attack

Problem: given (x, h(x)), find a 2nd-preimage x'

Algorithm
repeat
$x' \leftarrow$ random(); // guessing
until h(x) = h(x')

- Every step requires an hash computation and a random number generation that are efficient operations
- Storage and data complexity is negligible

Assumption 3 implies that, on average $O(2^m)$ "guesses" are necessary to determine a 2nd-preimage

The birthday paradox

- In a room of 23 people, the probability that at least a person is born on 25 december is 23/365 = **0.063**
  - **Proof**. P = 1/365 + … + 1/365 (23 times) = 0.063
- In a room of 23 people, the probability that at least 2 people have the same birthday is **0.507**
  - **Proof**. Let P be the probability we want to calculate. Let Q be the probability of the complementary event, Q = 1 – P.

    $Q = (364/365) \times (363/365) \times \ldots \times (343/365) = 0.493$

    $P = 0.507$

The birthday paradox

- An urn has m balls numbered *1* to m. Suppose that n balls are drawn from the urn one at a time, with replacement, and their numbers are listed.
- The probability of at least one coincidence (i.e., a ball drawn at least twice) is

$$1 - \exp(-n^2/2m), \text{ if } m \to \infty \text{ and } n = O(\mathrm{SQRT}(m))$$

- As $m \to \infty$, the expected number of draws before a coincidence is

$$\mathrm{SQRT}(\Pi m/2).$$

The Yuval's attack

Objective

Let x_1 be the *legitimate message* and
x_2 be a *fraudulent message*.

By applying "small" variations to x_1 and x_2 find x'_1 and x'_2 s.t. $h(x'_1) = h(x'_2)$

An adversary signs or lets someone sign x'_1 and later claims that x'_2 has been signed instead

The Yuval's attack

- Generate t variations x_1' of x_1 and store the couple $(x, h(x_1'))$ in table T *(time and storage complexity $O(t)$)*
- repeat
  generate a new variation x'_2 for x_2
  until $h(x'_2)$ is in the table T;
  return the corresponding variation x_1' for x_1

If $t = 2^m$, we can obtain a collision after $N = H/t$ trials with probability equal to 1

(if $t = 2^{m/2}$, then $N = 2^{m/2}$)

Ideal security

- **Design goal**

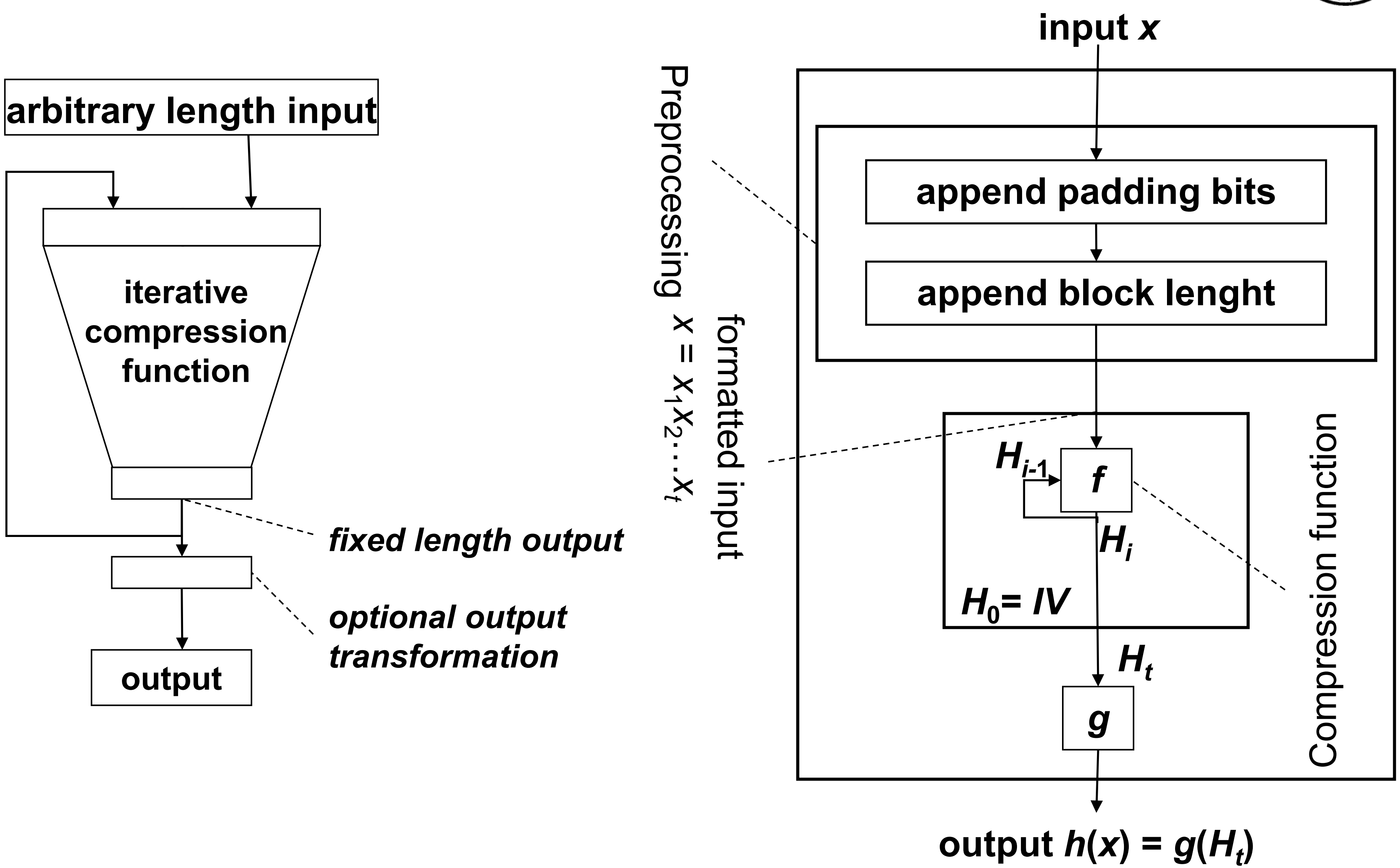
  The best possible attacks should require no less than $O(2^m)$ to find a preimage and $O(2^{m/2})$ to find a collision

- **Ideal security**

  given y, producing a preimage or a 2nd-preimage requires 2^m operations

  given x, producing a collision requires $2^{m/2}$ operations

General model of iterated hash functions



Classification of MDC

MDC may be categorized based on the nature of the operations comprising their internal compression functions

- **Hash functions based on block ciphers**
- **Ad-hoc hash functions**
- **Hash functions based on modular arithmetic**

Upper bounds of strength

Hash Function	*n*	*m*	Preimage	Collision	Comments
Matyas-Meyer-Oseas	*n*	*m*	2^n	$2^{n/2}$	cifrario
MDC-2 (con DES)	64	128	2×2^{82}	2×2^{54}	cifrario
MDC-4 (con DES)	64	128	2^{109}	2×2^{54}	cifrario
Merkle (con DES)	106	128	2^{112}	2^{56}	cifrario
MD4*	512	128	2^{128}	2^{20}	ad-hoc
MD5	512	128	2^{128}	2^{64}	ad-hoc
RIPEMD-128	512	128	2^{128}	2^{64}	ad-hoc
SHA-1, RIPEMD-160	512	160	2^{160}	2^{80}	ad-hoc

block size: *n*
output size: *m*

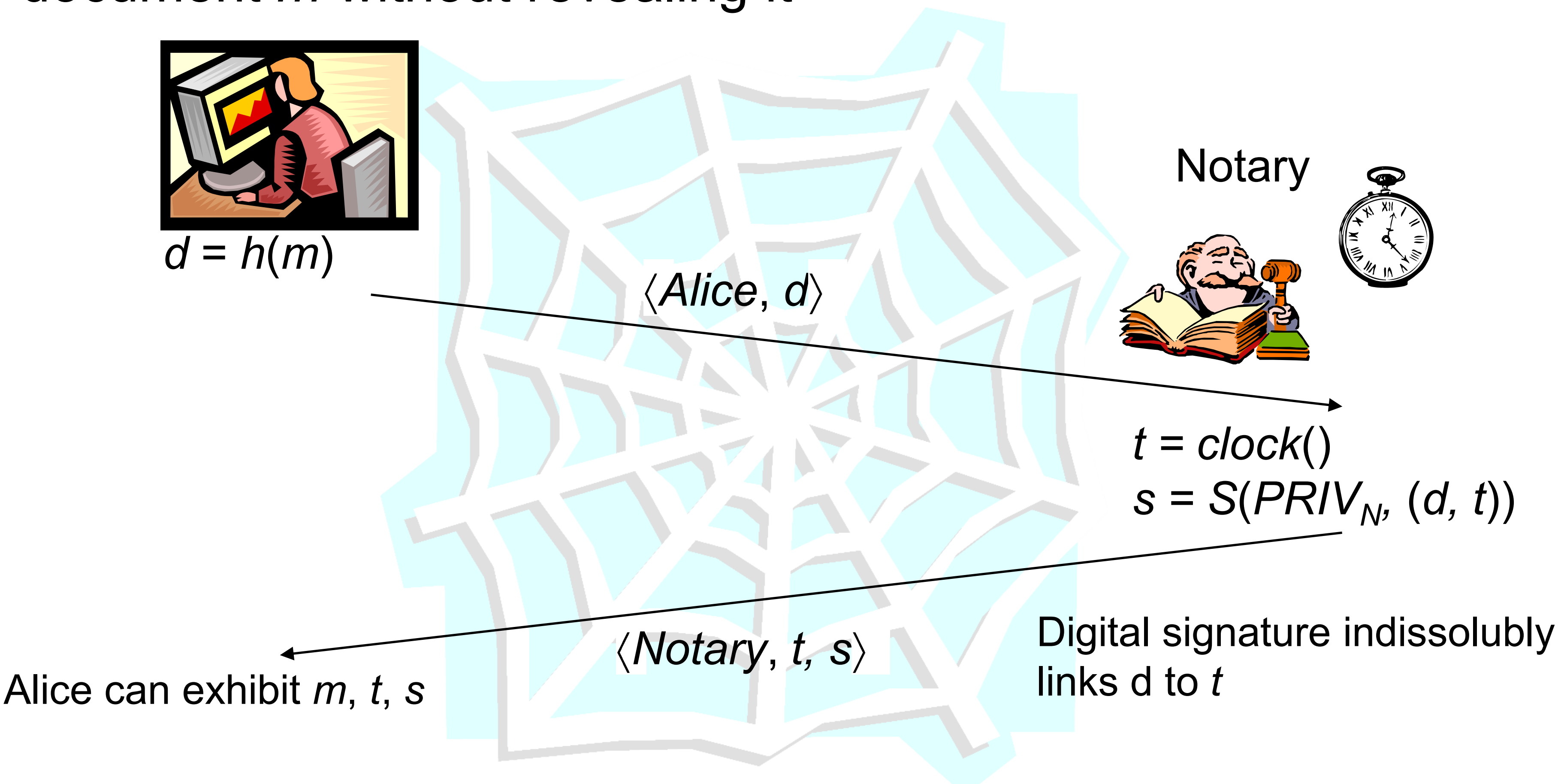
bitsize for practical security

OWHF: m ≥ 80

CRHF: m ≥ 160

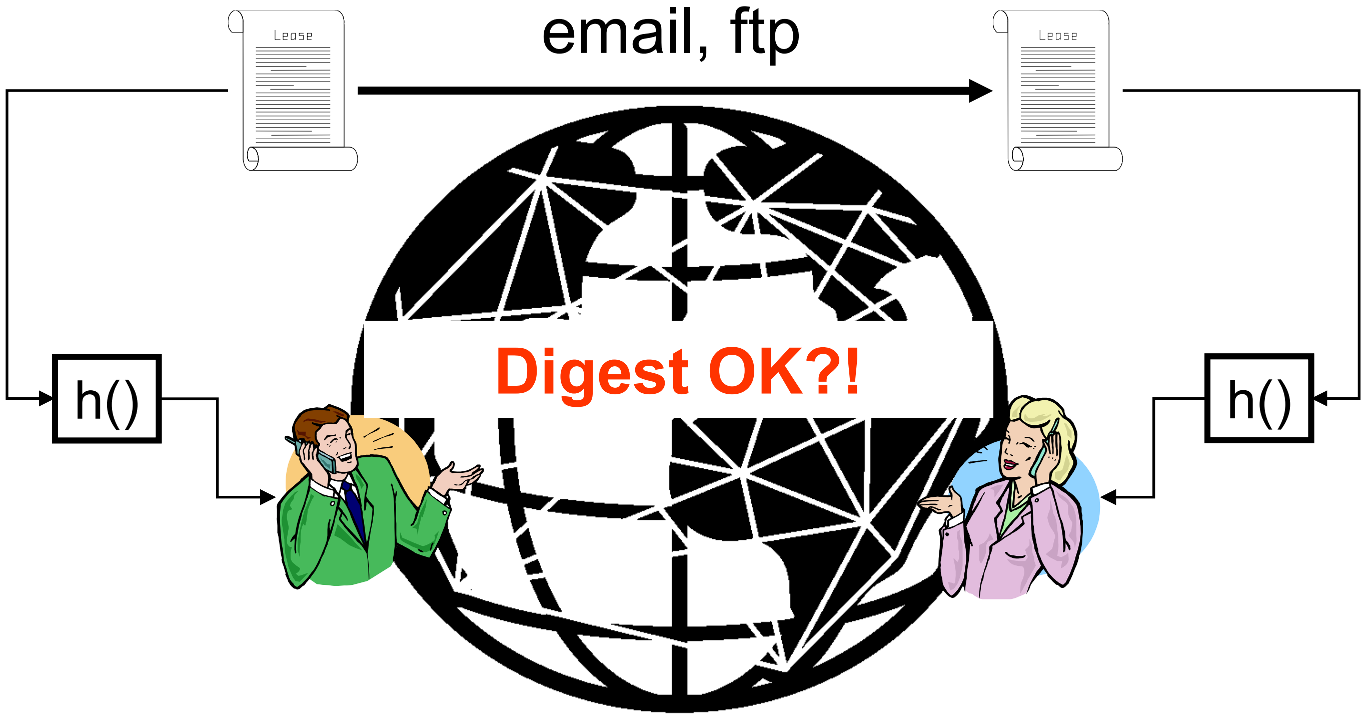
An example

Alice wants to be able to proof that, at a given time *t*, she held a document *m* without revealing it

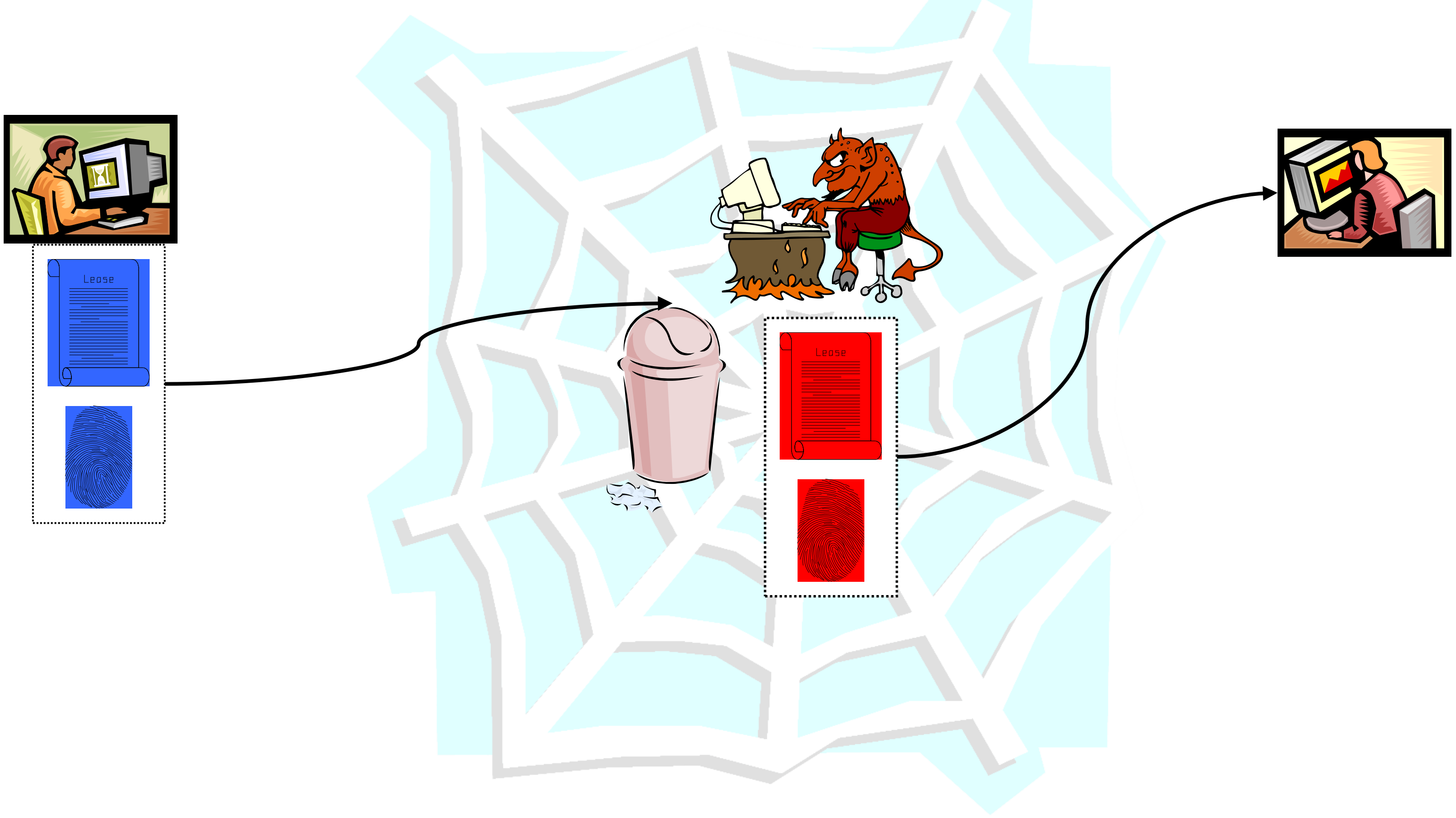


Manipulation Detection Code

The purpose of **MDC**, **in conjunction with other mechanisms** (authentic channel, encryption, digital signature), is to provide **message integrity**



An insecure system made of secure components



MDC alone is not sufficient to provide data integrity

Integrity with MDC

MDC and an authentic channel

- physically authentic channel
- digital signature

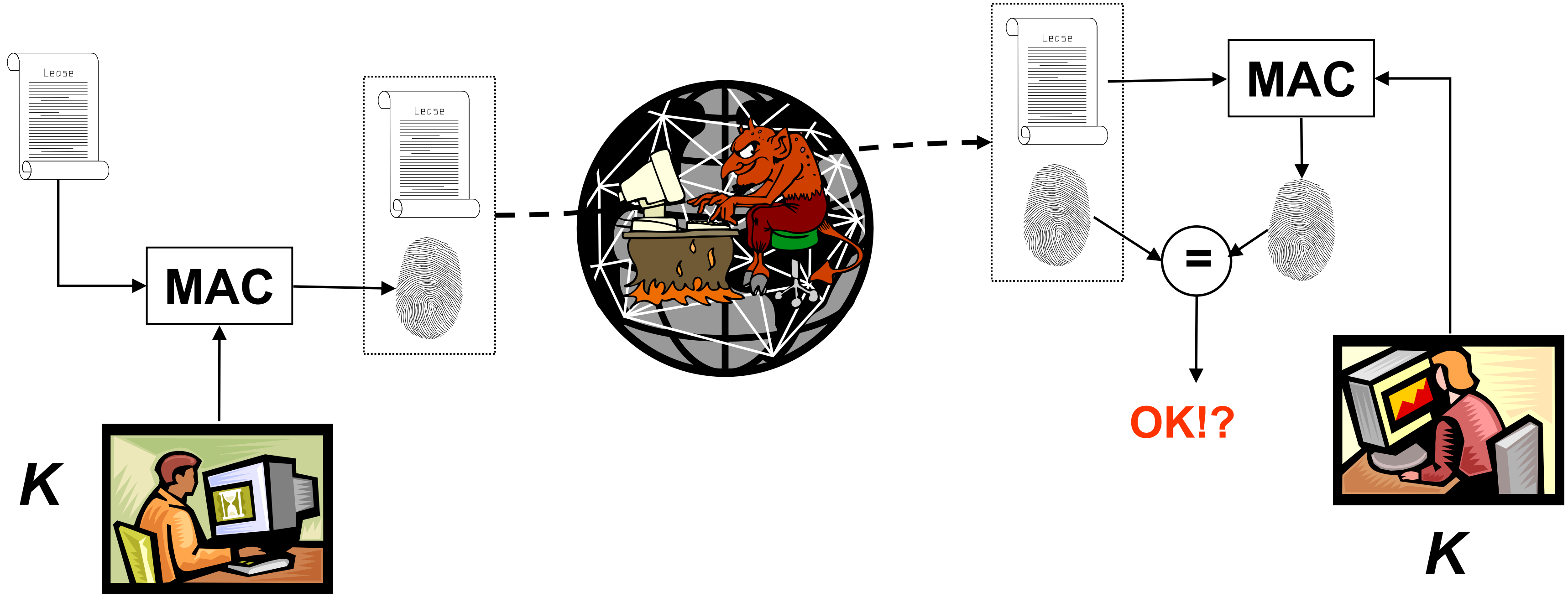
MDC and encryption

- $E_k(x, h(x))$
  - confidentiality and integrity
  - *h* may be weaker
  - as secure as *E*
- x, $E_k(h(x))$
  - *h* must be collision resistant
  - k must be used only for integrity (*risk of selective forgery*)
- $E_k(x), h(x)$
  - h must be collision resistant
  - *h* can be used to check a guessed *x*

Message Authentication Code (MAC)

Message Authentication Code

The purpose of **MAC** is to provide **message authentication by symmetric techniques** (without the use of any additional mechanism)



Alice and Bob share a secret key

Message Authentication Code

Definition. A MAC algorithm is a famility of functions h_k, parametrized by a **secret** key k, with the following properties:

ease of computation – Given a function $\boldsymbol{h_k}$, a key $\boldsymbol{k}$ and an input $\boldsymbol{x}$, $\boldsymbol{h_k(x)}$ is ***easy to compute***

compression – $\boldsymbol{h_k}$ maps an input $\boldsymbol{x}$ of arbitrary finite bitlength into an output $\boldsymbol{h_k(x)}$ of fixed length $\boldsymbol{n}$.

computation-resistance – for each key $\boldsymbol{k}$, given zero o more $\boldsymbol{(x_i, h_k(x_i))}$ pairs, it is **computationally infeasible** to compute $\boldsymbol{(x, h_k(x))}$ for any new input $\boldsymbol{x \neq x_i}$ (including possible $\boldsymbol{h_k(x) = h_k(x_i)}$ for some $\boldsymbol{i}$).

Message Authentication Code

- ***MAC forgery*** occurs if computation-resistance does not hold
- **Computation resistance implies key non-recovery** (but not vice versa)
- **MAC definition says nothing about preimage and 2nd-preimage for parties knowing *k***
- **For an adversary not knowing *k***
  - **h_k must be 2nd-preimage and collision resistant;**
  - **h_k must be preimage resistant w.r.t. a chosen-text attack;**

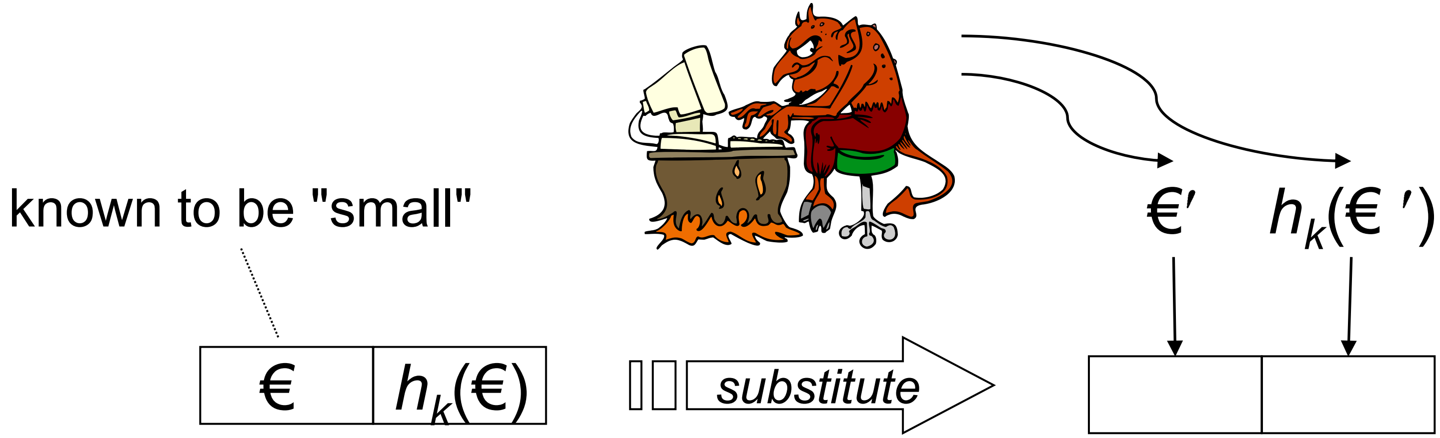
Attacks to MAC

- **Adversary's objective**
  - without prior knowledge of k, compute a new text-MAC pair $(x, h_k(x))$, for some $x \neq x_i$, given one or more pairs $(x_i, h_k(x_i))$
- **Attack scenarios for adversaries with increasing strenght**:
  - known-text attack
  - chosen-text attack
  - adaptive chosen-text attack
- A MAC algorithm should withstand adaptive chosen-text attack regardless of whether such an attack may actually be mounted in a particular environment

Types of forgery

- Forgery allows an adversary to have a forged text accepted as authentic
- Classification of forgeries
  - *Selective forgeries*: an adversary is able to produce text-MAC pairs of text of his choice
  - *Existential forgeries*: an adversary is able to produce text-MAC pairs, but with no control over the value of that text
- Comments
  - Key recovery allows both selective and existential forgery
  - Even an existential forgery may have severe consequences

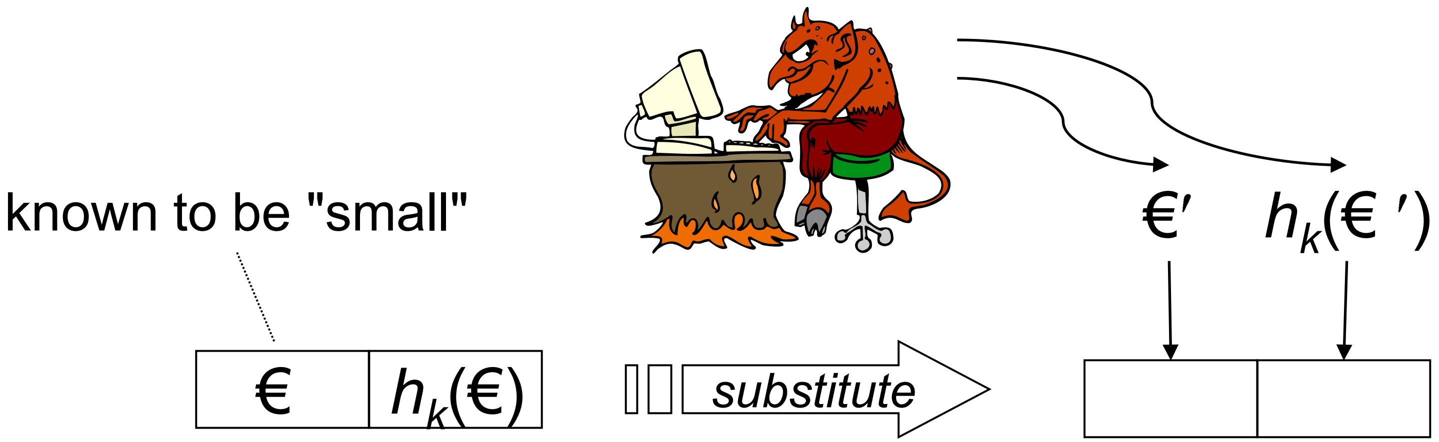
An example of existential forgery



Mr. Lou Cipher

- knows that € is a small number
- esistentially forges a pair (€′, h_k(€ ′)) with €′ uniformly distributed in [0, 2^{32} – 1] ($P_{forgery}$ = 1 – €/2^{32})
- substitutes (€, h_k(€)) with (€′, h_k(€ ′))

An example of existential forgery



Countermeasure

Messages whose integrity or authenticity has to be verified are constrained to have pre-determined structure or a high degree of verifiable redundancy

For example: change € into €⚐€

Relationship between properties

Let h_k be a MAC algorithm.

Then h_k is, against a chosen-text attack by an adversary not knowing key k,

- 2nd-preimage and collision resistance, and
  - PROOF. Computation resistance implies that MAC cannot be even computed without the knowledge of k
- preimage resistant
  - PROOF BY CONTRADICTION.

Let us suppose that h is not preimage resistance. Then, given a randomly-selected hash value y it is possible to recover the preimage x. But this violates computation resistance

Security objectives

Let h_k be a MAC algorithm with a *t*-bit key and an *m*-bit output

Design Goal	Ideal strength	Adversary's Goal
key non-recovery	2^t	deduce *k*
computational resistance	$P_f = \max(2^{-t}, 2^{-m})$	produce new (text, MAC)

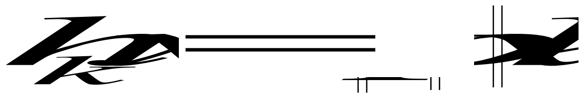
P_f is the probability of forgery by correctly guessing a MAC

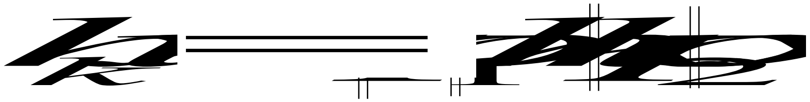
bitsize for practical security

- $m \geq 64$ bit
- $t \geq 64 \div 80$ bit

Implementation

- MAC based on block-cipher
  - CBC-based MAC
- MAC based on MDC
  - The MAC key should be involved at both the start and the end of the MAC computation

 envelope method with padding

 hash-based MAC

- Customized MAC (MAA, MD5-MAC)
- MAC for stream ciphers

Data integrity

Data integrity using MAC alone

- $x, h_k(x)$

Data integrity using an MDC and an authentic channel

- message x is transmitted over an insecure channel
- MDC is transmitted over the authentic channel (telephone, daily newspaper,…)

Data integrity

Data integrity combined with encryption **(...)**

- **Encryption alone does not guarantee data integrity**
  - reordering of ECB blocks
  - encryption of random data
  - bit manipulation in additive stream cipher and DES ciphertext blocks
- **Data integrity using encryption and an MDC (…)**
  - $C = E_k(x, h(x))$
    - h(x) deve soddisfare proprietà più deboli rispetto a quelle necessarie per la firma digitale
    - La sicurezza del meccanismo di integrità è pari al più a quella cifrario

Data integrity combined with encryption

- **Data integrity using encryption and an MDC** ***soluzioni sconsigliabili***
  - **(x, $E_k(h(x))$)** h must be collision resistant, otherwise pairs (x, x') with colliding outputs can be verifiably pre-determined without the knowledge of k
  - **$E_k(x)$, $h(x)$** – little computational savings with respect to encrypt x and $h(x)$; h must be collision resistant; correct guesses of x can be confirmed

Data integrity using encryption and a MAC

- $C = E_{k1}(x, h_{k2}(x))$
  - Pros w.r.t. MDC
    - Should *E* be defeated, *h* still guarantees integrity
    - *E* precludes an exhaustive key search attack on *h*
  - Cons w.r.t. MDC
    - Two keys instead of one
  - Recommendations
    - *k*1 and *k*2 should be different
    - E and h should be different

Data integrity

Data integrity using encryption and a MAC

Alternatives

- $E_{k1}(x), h_{k2}(E_{k1}(x))$
  - allow authentication without knowledge of plaintext
  - no guarantee that the party creating MAC knew the plaintext
- $E_{k1}(x), h_{k2}(x).$
  - *E* and *h* cannot compromise each other

Comments

- Data origin mechanisms based on shared keys (e.g., MACs) *do not provide non-repudiation* of data origin
- While MAC (and digital signatures) provide data origin authentication, they *provide no inherent uniqueness or timeliness guarantees*

  To provide these guarantees, data origin mechanisms can be augmented with **time variant parameters**

  - timestamps
  - sequence numbers
  - random numbers

Resistance properties

Resistance properties required for specified data integrity applications

Hash properties required → Integrity application ↓	Preimage resistant	2nd-preimage resistant	Collision resistant
MDC + asymmetric signature	yes	yes	yes†
MDC + authentic channel		yes	yes†
MDC + symmetric encryption			
Hash for one-way password file	yes		
MAC (key unknown to attacker)	yes	yes	yes†
MAC (key known to attacker)		yes‡	

† Resistance required if chosen message attack

‡ Resistance required in the rare case of multi-cast authentication