ENEE 457: Computer Systems Security
10/5/16

Lecture 10
Digital Signatures

Charalampos (Babis) Papamanthou

Department of Electrical and Computer Engineering
University of Maryland, College Park

- Slides adjusted from:
  - http://dziembowski.net/Teaching/BISS09/

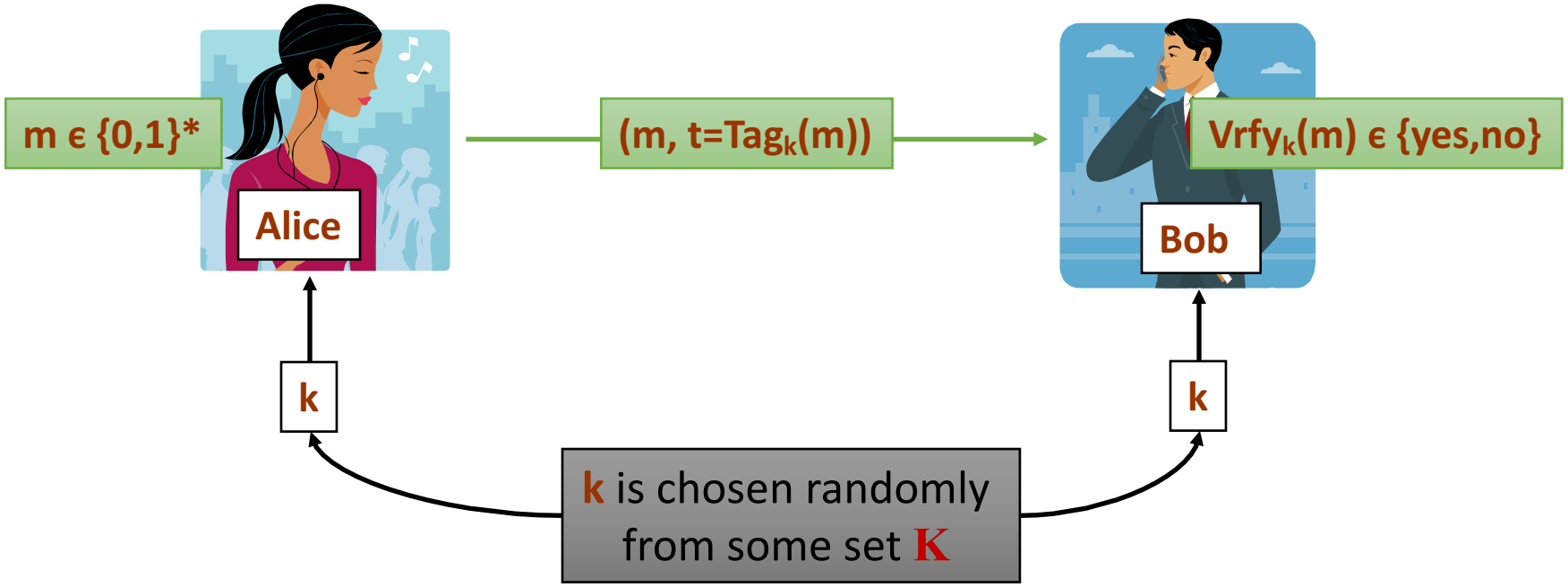
Signature schemes

digital signature schemes

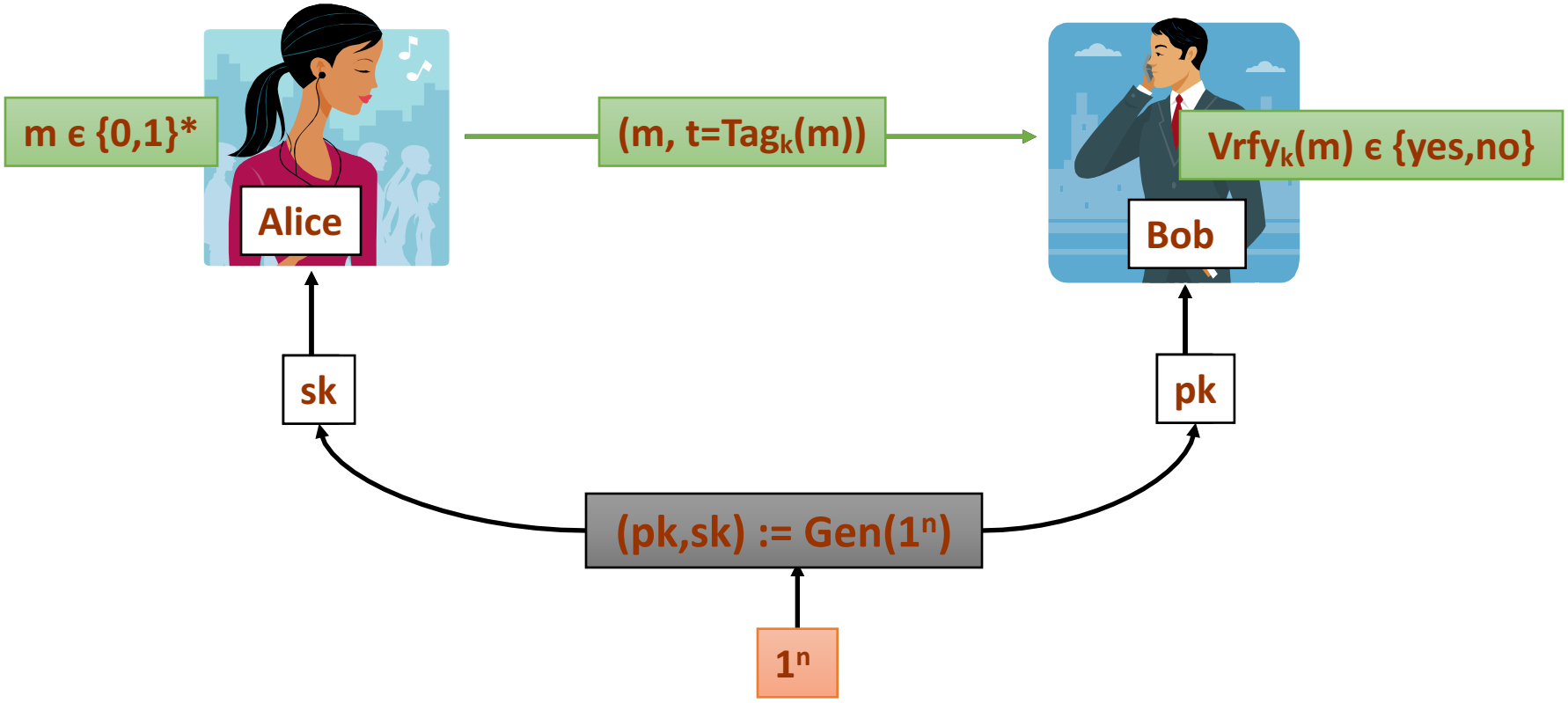
≈

MACs in the public-key setting

Message Authentication Codes – the idea



Signature Schemes

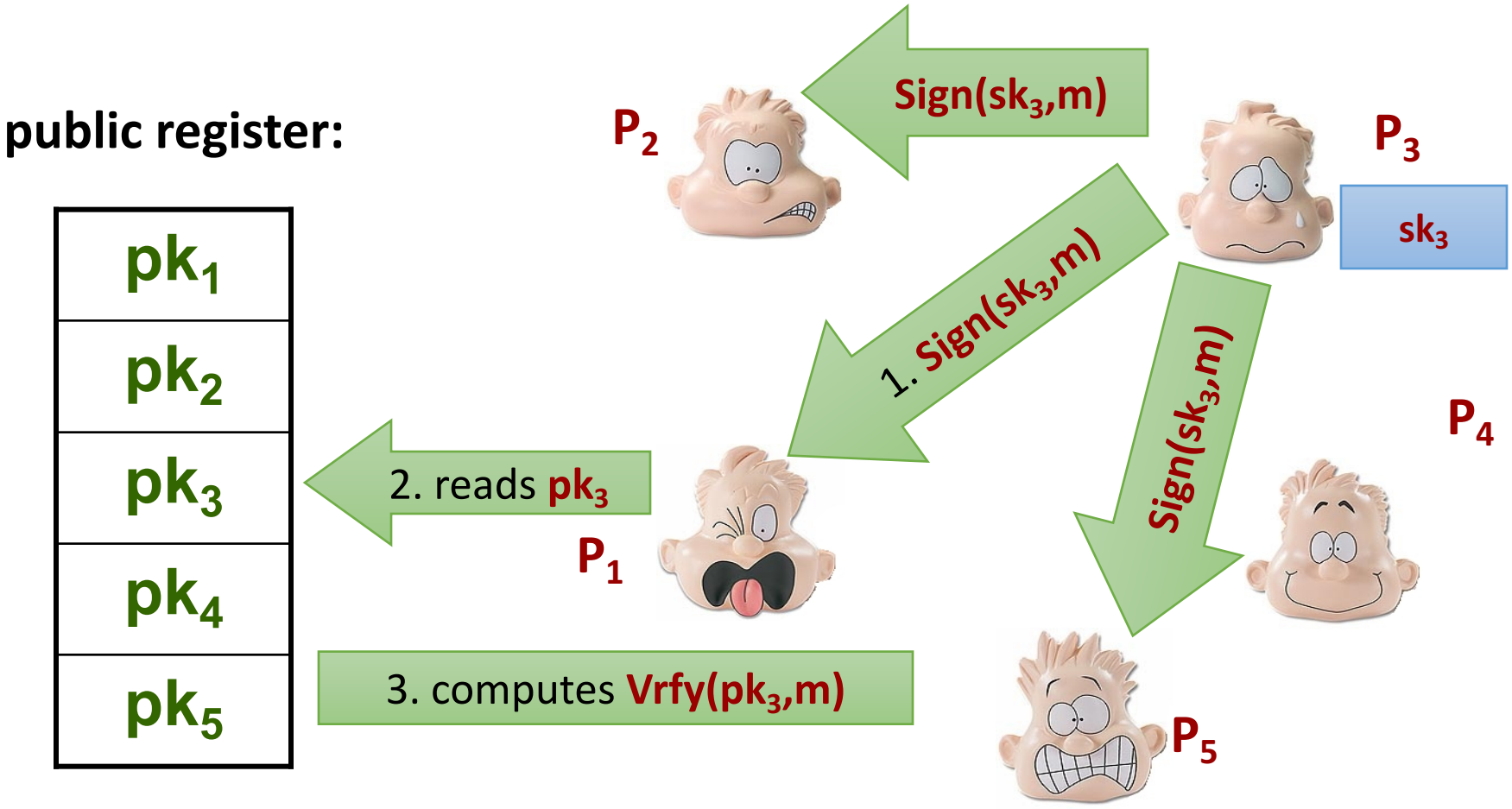


Advantages of the signature schemes

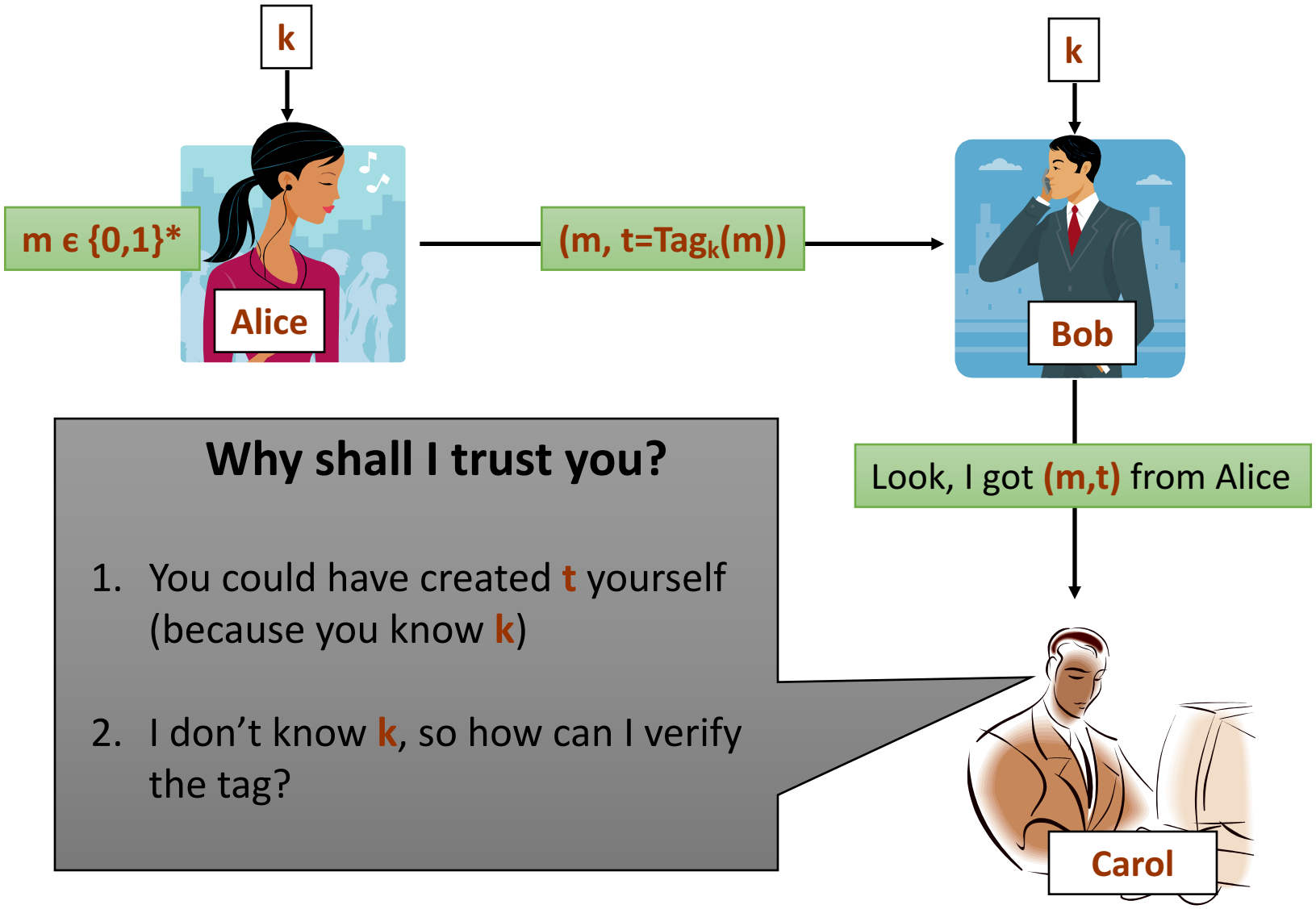
Digital signatures are:

1. **publicly verifiable**
2. **transferable**
3. provide **non-repudiation**

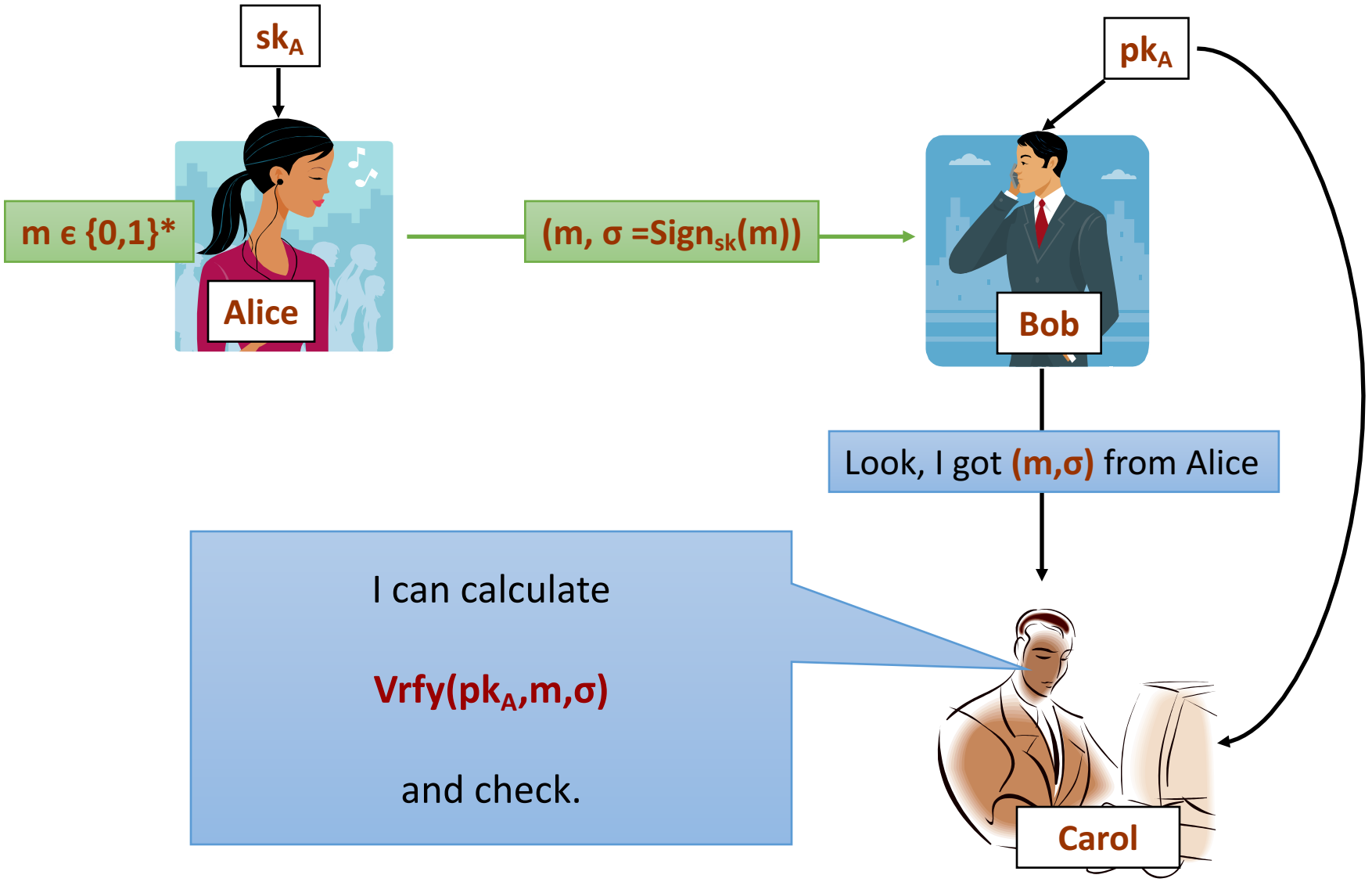
Anyone can verify the signatures



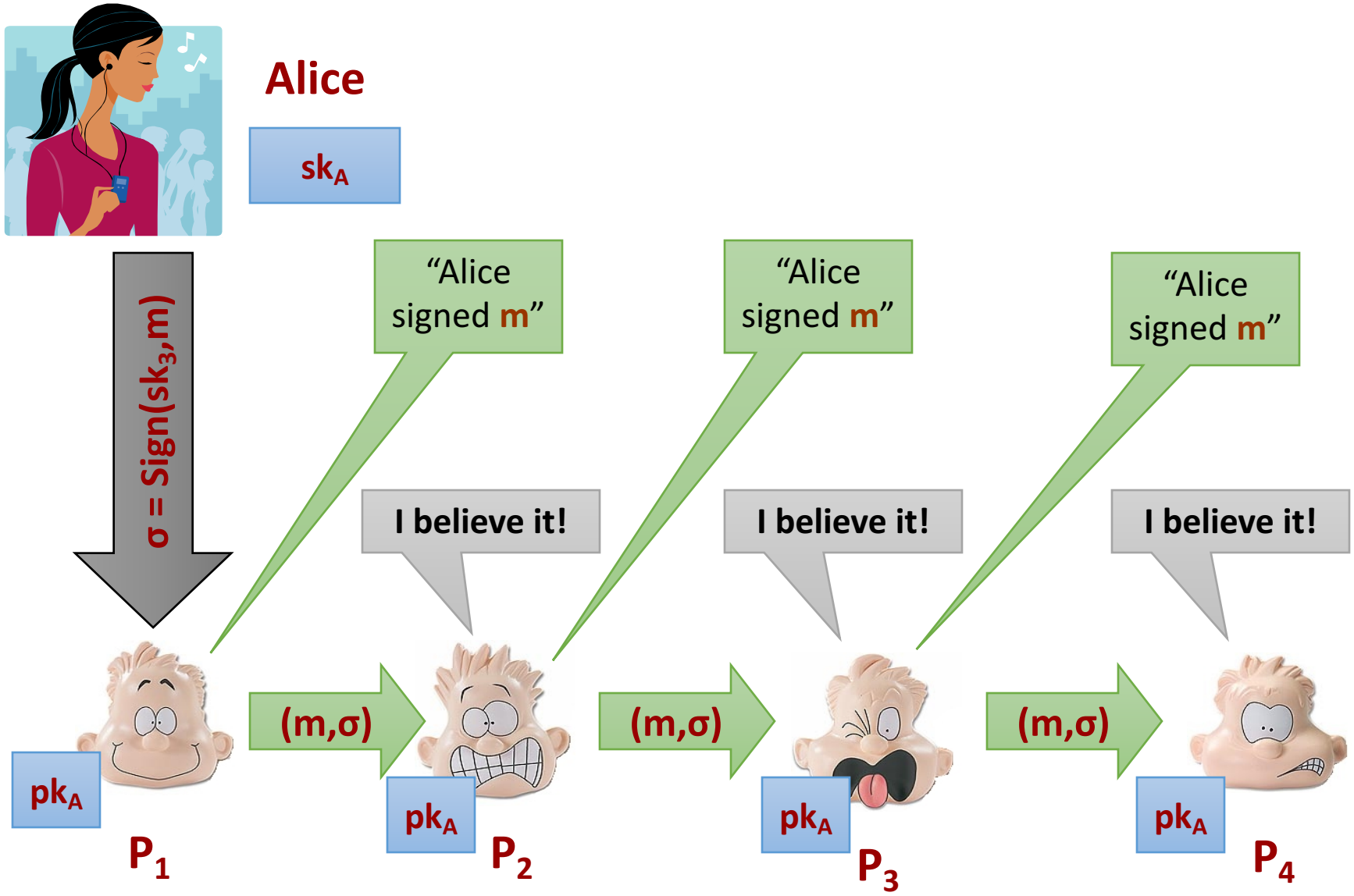
Look at the MACs...



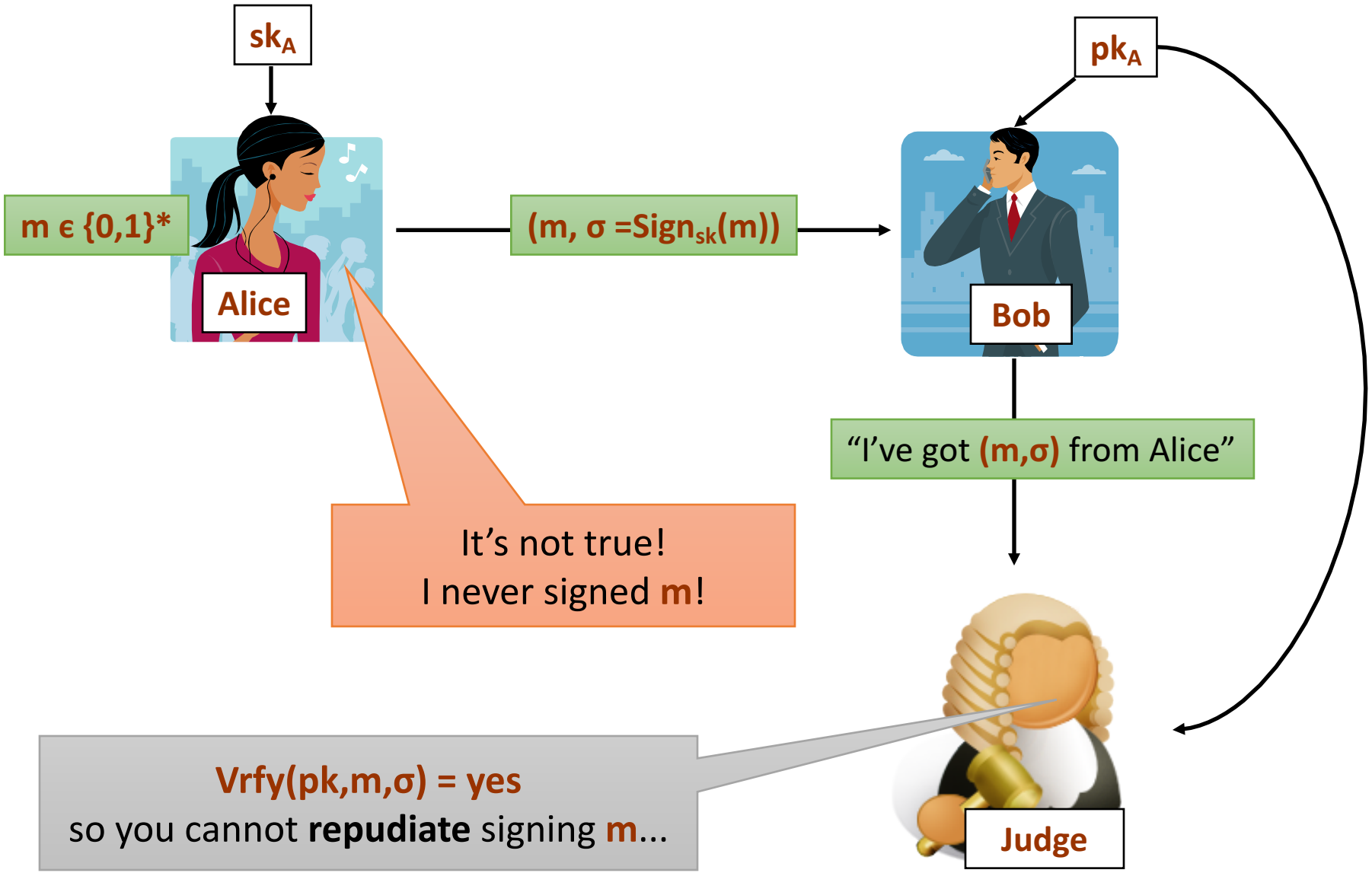
Signatures are publicly-verifiable!



So, the signatures are transferable



Non-repudiation



Digital Signature Schemes

A **digital signature scheme** is a tuple **(Gen,Sign,Vrfy)** of poly-time algorithms, such that:

- the **key-generation** algorithm **Gen** takes as input a security parameter 1^n and outputs a pair **(pk,sk)**,
- the **signing** algorithm **Sign** takes as input a key **sk** and a message $m \in \{0,1\}^*$ and outputs a signature **σ**,
- the **verification** algorithm **Vrfy** takes as input a key **pk**, a message **m** and a signature **σ**, and outputs a bit **b ϵ {yes, no}.**

If $\text{Vrfy}_{pk}(m,\sigma) = \text{yes}$ then we say that **σ** is a **valid signature on the message m.**

Correctness

We require that it always holds that:

$$\mathbf{Vrfy_{pk}(m,Sign_{sk}(m)) = yes}$$

What remains is to define **security** of a **MAC**.

How to define security?

As in the case of MACs, we need to specify:

1. how the messages $m_1,\ldots,m_t$ are chosen,

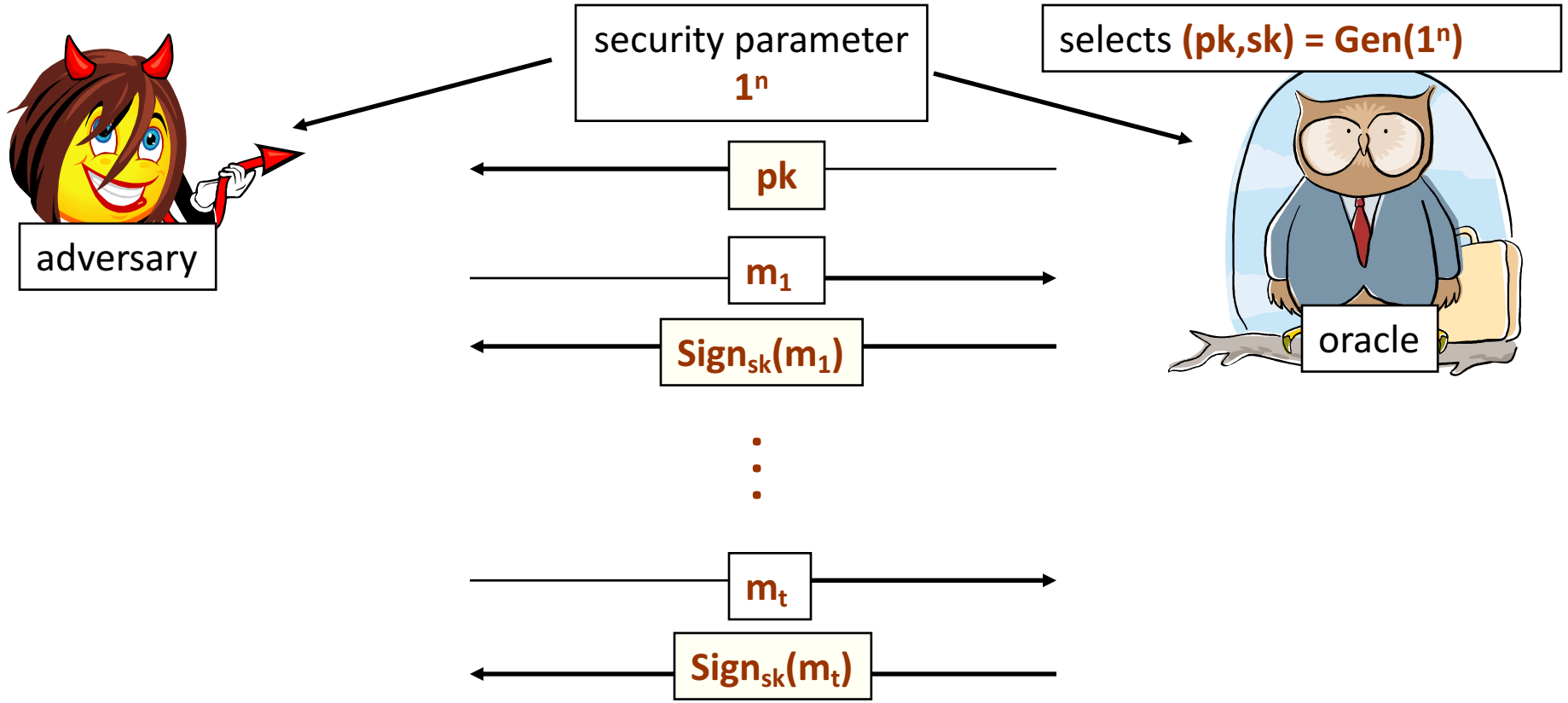
2. what is the goal of the adversary.

Good tradition: be as pessimistic as possible!

Therefore we assume that

1. The adversary is allowed to chose $m_1,\ldots,m_t$.

2. The **goal of the adversary** is to produce a valid signature on some m' such that $m' \neq m_1,\ldots,m_t$.



We say that the adversary **breaks the signature scheme** if at the end she outputs (m', σ') such that

1. $Vrfy(m', \sigma') = yes$
2. $m' \neq m_1,\ldots,m_t$

The security definition

sometimes we just say: **unforgeable** (if the context is clear)

We say that **(Gen,Sign,Vrfy)** is **existentially unforgeable under an adaptive chosen-message attack** if

$\forall$ polynomial-time adversary **A**

P(A breaks it) is negligible (in **n**)

The "handbook RSA signatures"

N = pq - **RSA** modulus

e is such that **gcd(e, ϕ(N)) = 1**,
d is such that **ed = 1 (mod ϕ(N))**

$$\mathbf{Sign}_{(d,N)}(c) = c^d \bmod N$$

and

$$\mathbf{Vrfy}_{(e,N)}(m, \sigma) = \mathbf{yes} \text{ iff } \sigma^e = m \bmod$$

Correctness:

$\sigma^e = (m^d)^e$
$= m^{de}$
$= m^1$
$= m$

Problems with the "handbook RSA" [1/2]

A "no-message attack":

The adversary can forge a signature on a "random" message **m**.

Given the public key **(N,e)**:

he just selects a random **σ** and computes

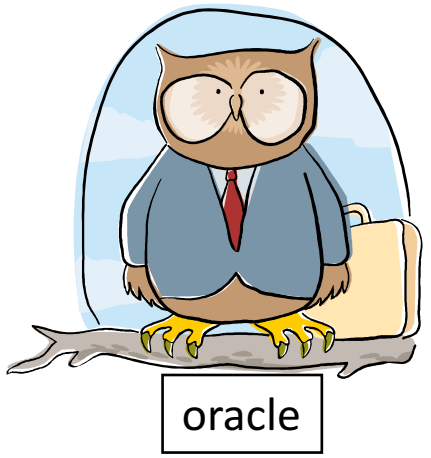
$$m = \sigma^e \bmod N.$$

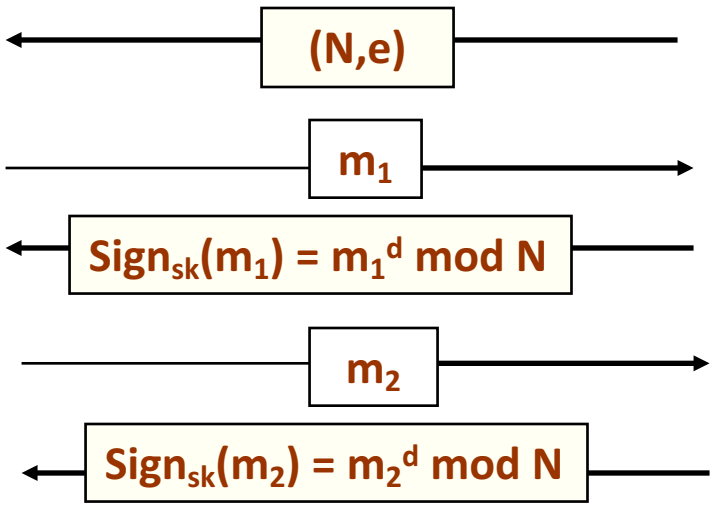
Trivially, **σ** is a valid signature on **m.**

Problems with the "handbook RSA" (2/2)

How to forge a signature on an arbitrary message **m**?

Use the homomorphic properties of RSA.





chooses:

1. random m_1
2. $m_2 := m / m_1 \bmod N$

computes (**mod N**):

$$m_1^d \cdot m_2^d = (m_1 \cdot m_2)^d = m^d$$

this is a valid signature on **m**

Is it a problem?

In many applications – probably not.

But we would like to have schemes that are not application-dependent...

Solution

Before computing the RSA function – apply some function **H**.

N = pq, such that **p** and **q** are large random primes
e is such that **gcd(e, ϕ(N)) = 1**
d is such that **ed = 1 (mod ϕ(N))**

$Sign_d$: Z_N^* → Z_N^* is defined as:

$$Sign(m) = H(m)^d \bmod N.$$

$Vrfy_e$ is defined as:

$$Vrfy_e(m,\sigma) = \text{yes} \text{ iff } \sigma^e = H(m) \pmod N$$

How to choose such H?

A minimal requirement:

it should be collision-resistant.

(because if the adversary can find two messages **m,m'** such that

H(m) = H(m')

then he can forge a signature on **m'** by asking the oracle for a signature on **m**)

Hash-and-Sign [1/5]

Hash and sign is a generic construction that takes as input:

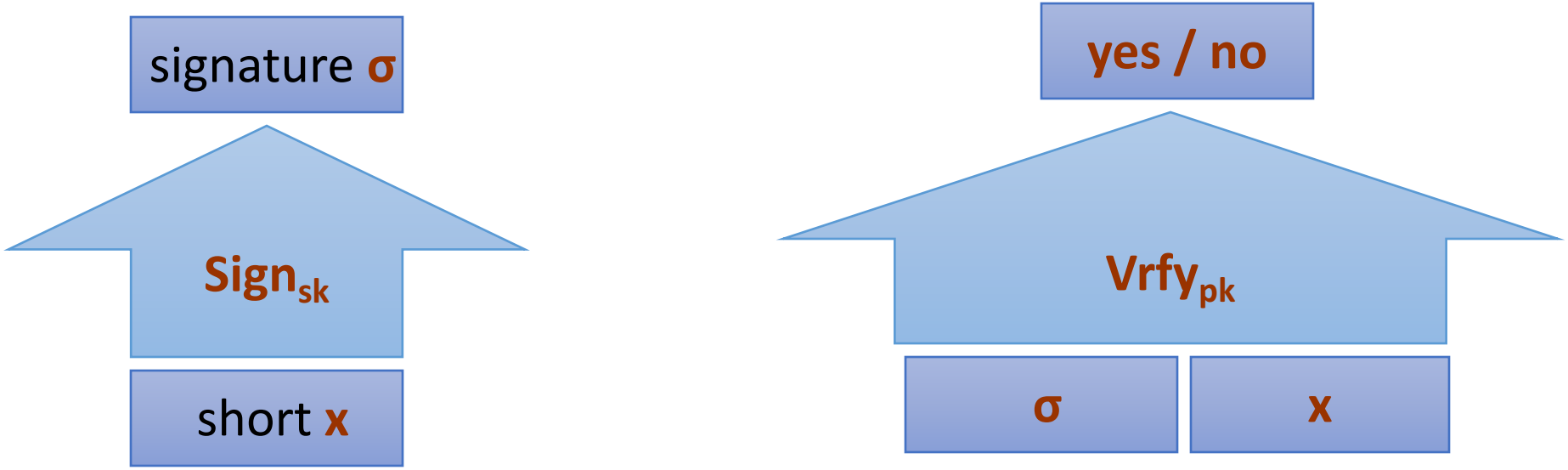
- a signature scheme that works on "**short messages**", and
- a **hash function**,

and transforms it into a

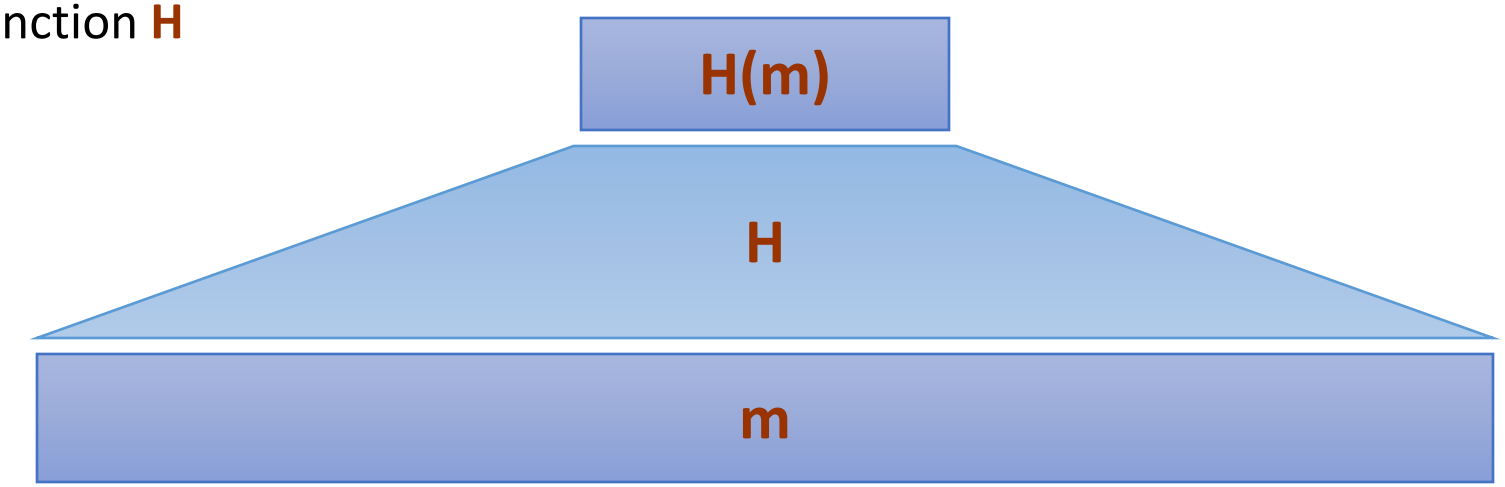
- a signature scheme that works on "**long messages**".

Hash-and-Sign [2/5]

1. **(Gen,Sign,Vrfy)** – a signature scheme "for short messages"

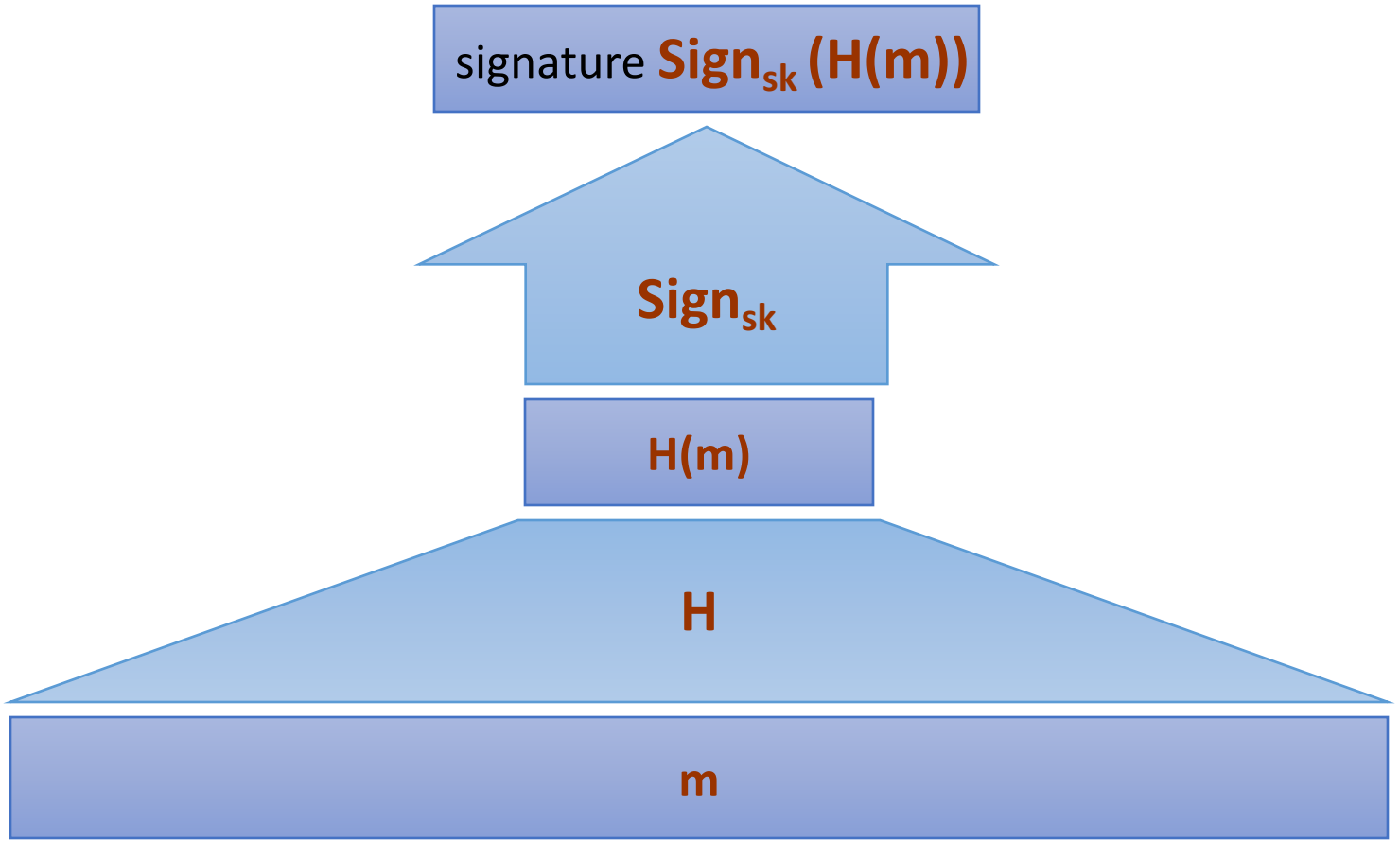


2. a hash function **H**



Hash-and-Sign [3/5]

How to sign a message **m**?



Hash-and-Sign [4/5]

How to verify?