CS 4770: Cryptography

CS 6750: Cryptography and Communication Security

Alina Oprea
Associate Professor, CCIS
Northeastern University

March 29 2018

Outline

- ElGamal encryption
  - Based on Diffie-Hellman key exchange
  - CPA secure
- Digital signatures
  - Integrity in public-key world
  - Equivalent of MACs
  - Public verifiability
- Distribution of public keys

The ElGamal system (a modern view)

G: finite cyclic group of order q

We construct a pub-key enc. system (Gen, Enc, Dec):

- Key generation Gen:
  - choose random generator g in G and random x in Z_q
  - output sk = x , pk = (g, $h=g^x$)

Enc(pk=(g,h), m) :

$y \longleftarrow Z_q$, $u \longleftarrow g^y$, $k \longleftarrow h^y$

$c \longleftarrow k \cdot m$

output (u, c)

Dec(sk=x, (u,c)) :

$k \longleftarrow u^x$

$m \longleftarrow k^{-1} \cdot c$

output m

Decisional Diffie-Hellman

Let **G** be a finite cyclic group and **g** generator of G

$$G = \{ 1, g, g^2, g^3, \ldots, g^{q-1} \}$$

q is the order of G

Definition: We say that **DDH is hard in G** if for all PPT adversaries D:

$$| \Pr[D(g^x, g^y, g^{xy}) = 1] - \Pr[D(g^x, g^y, g^z) = 1] | < \text{negligible}$$

G, q and g are public and known to D

x, y, z are chosen uniformly at random in {1,…q-1}

Security

Theorem: Let G be a cyclic group of order q. Assuming that the DDH problem is hard, then El-Gamal encryption is CPA secure.

In particular, for every PPT adversary A attacking the CPA security of El-Gamal:

$$\Pr[\mathrm{Exp}_{\Pi,A}^{\mathrm{CPA}}(n) = 1] = 1/2 + \text{negligible}(n)$$

Proof of security - Intuition

Π **Enc(pk=(g,h), m)**

$y \leftarrow Z_q$, $u \leftarrow g^y$

$c \leftarrow h^y \cdot m$ $(= g^{xy} \cdot m)$

output (u, c)

1. Success of adversary to break **Π** and **Π'** in CPA game is similar

Under the assumption that DDH is hard !

Π' **Enc'(pk=(g,h), m)**

$y \leftarrow Z_q$, $u \leftarrow g^y$, $z \leftarrow Z_q$

$c \leftarrow g^z \cdot m$

output (u, c)

2. Success of adversary to break **Π'** in CPA game is negligible

Malleability of El-Gamal

To encrypt message m:

- $c = (g^y, h^y \cdot m)$, for y random

Multiply second part of ciphertext by α

- $c' = (g^y, h^y \cdot m \cdot \alpha)$ is a valid encryption of $m \cdot \alpha$

El-Gamal is malleable and not CCA-secure

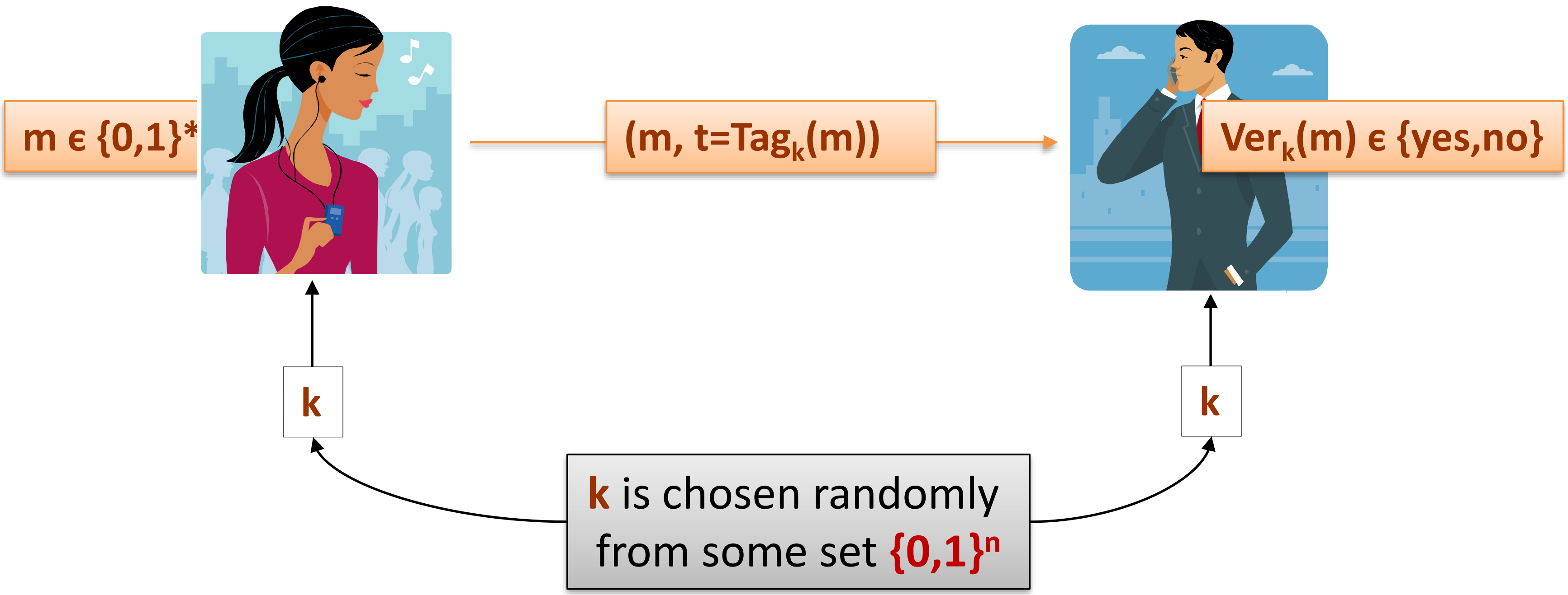
Signature schemes

digital signature schemes

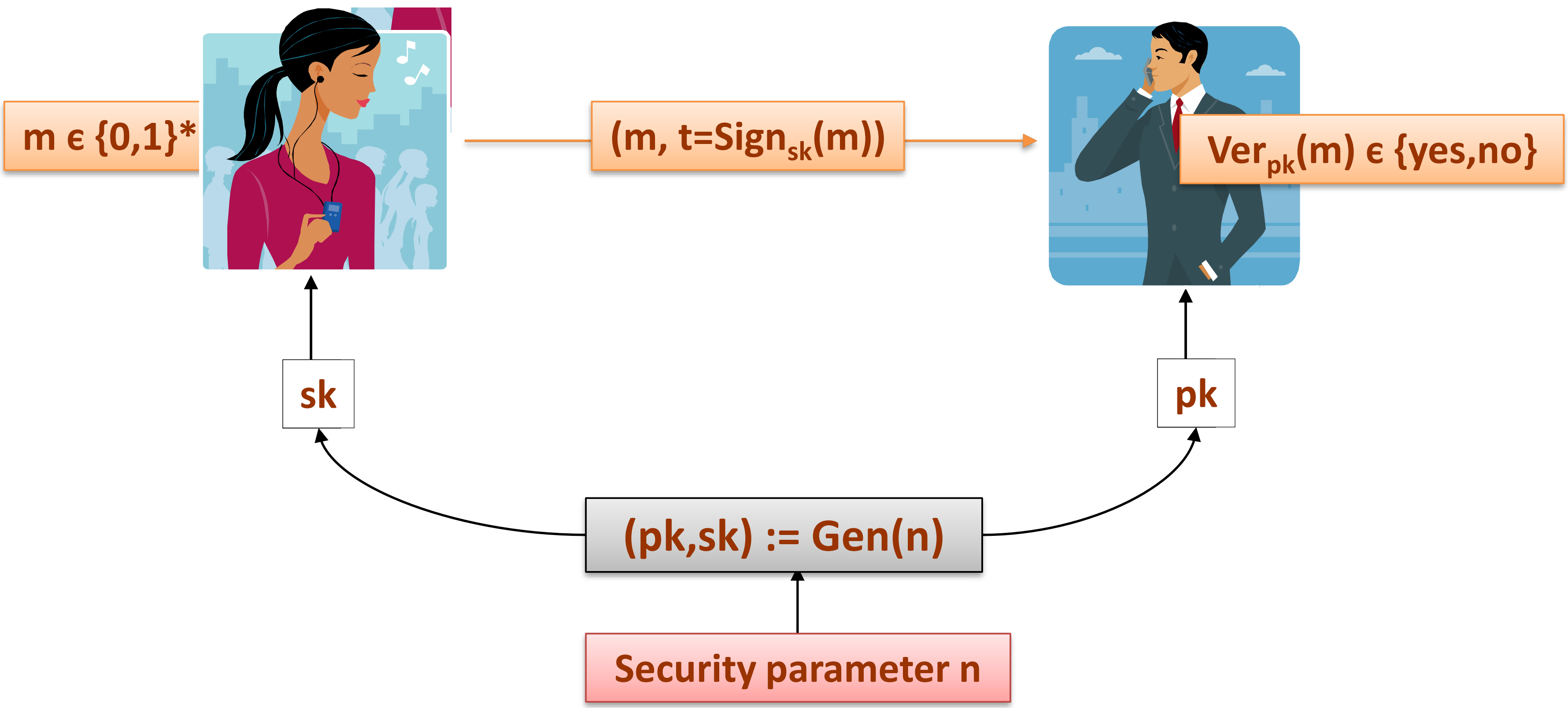
$\approx$

MACs in the public-key setting

Message Authentication Codes



Signature Schemes

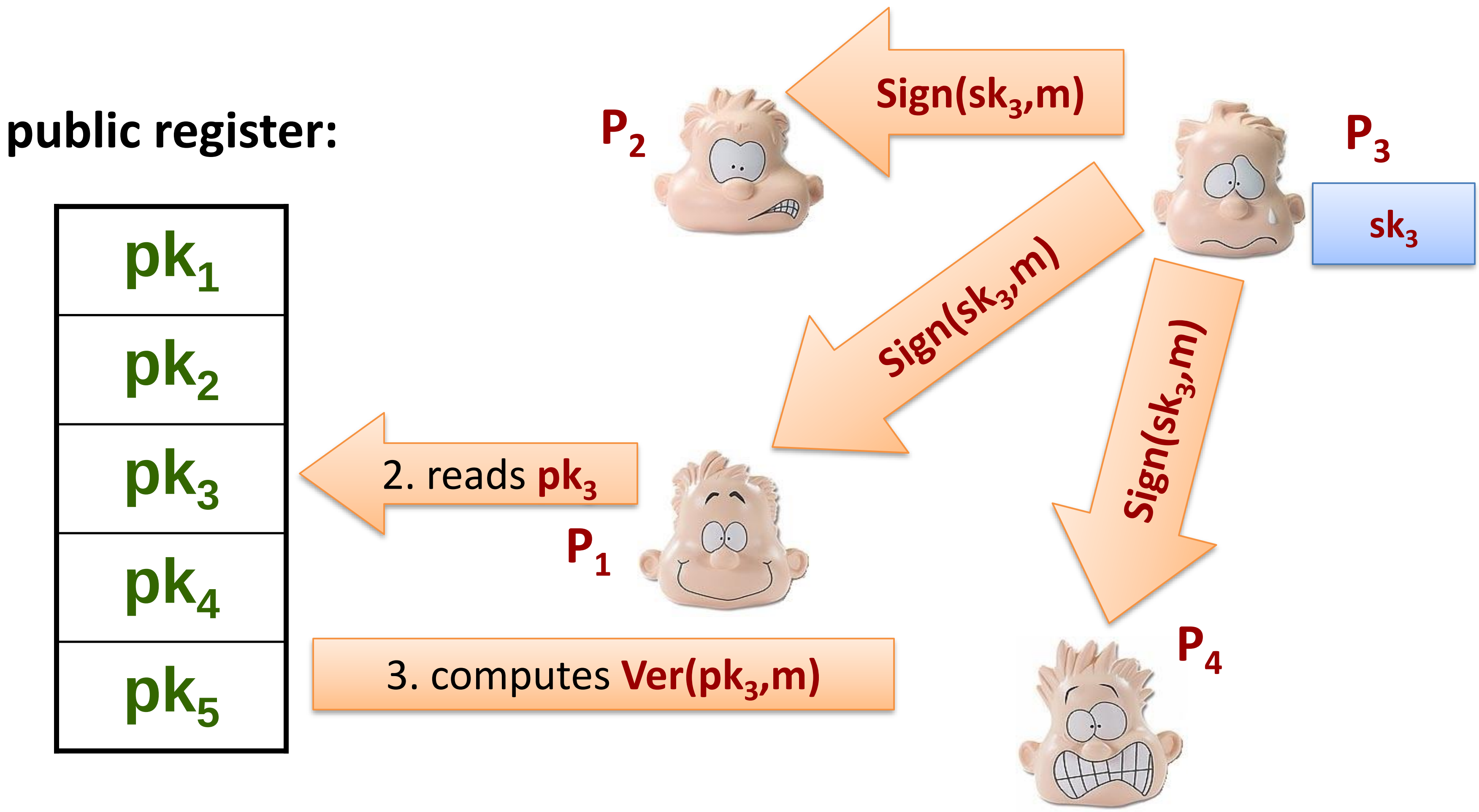


Advantages of signature schemes

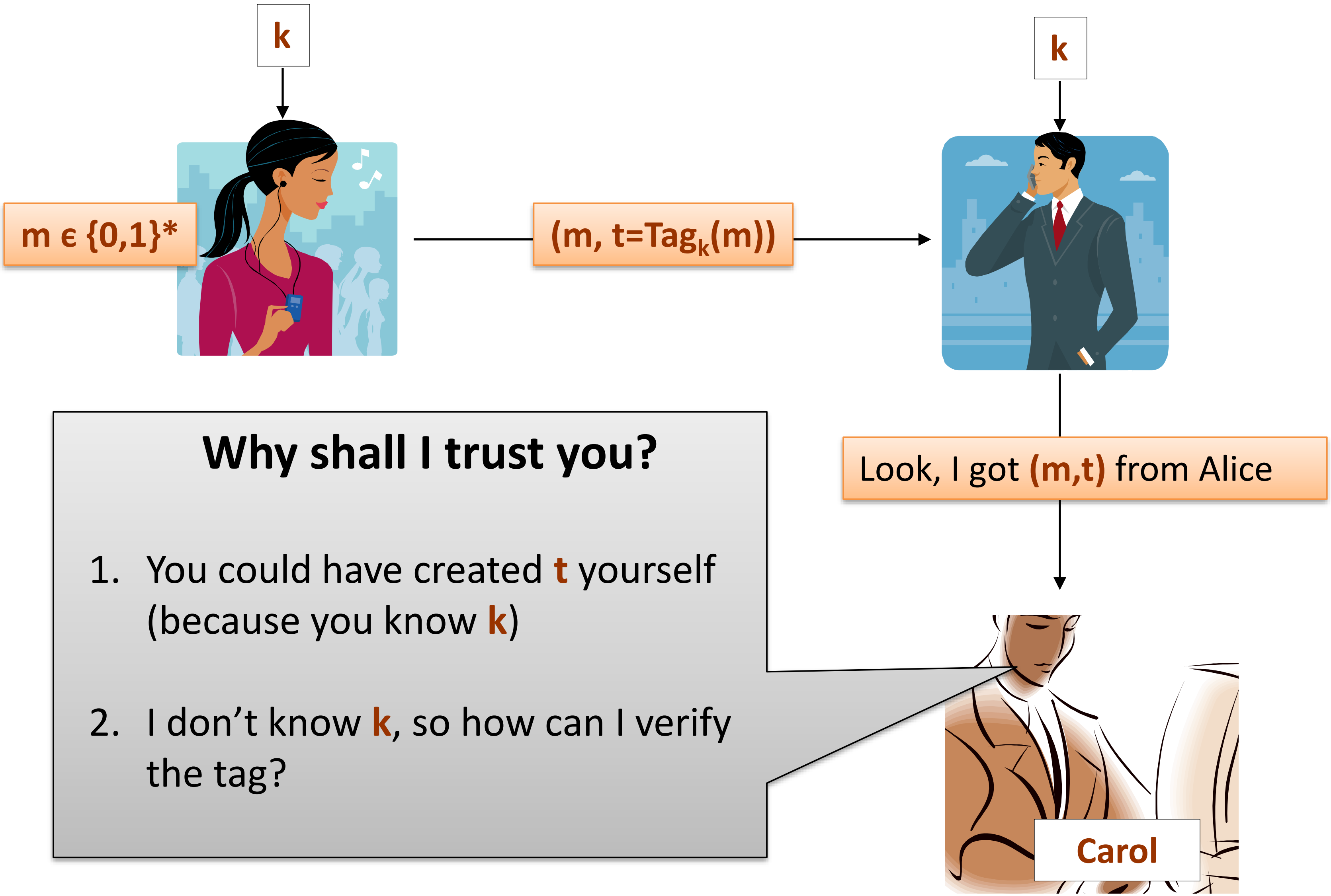
Digital signatures are:

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

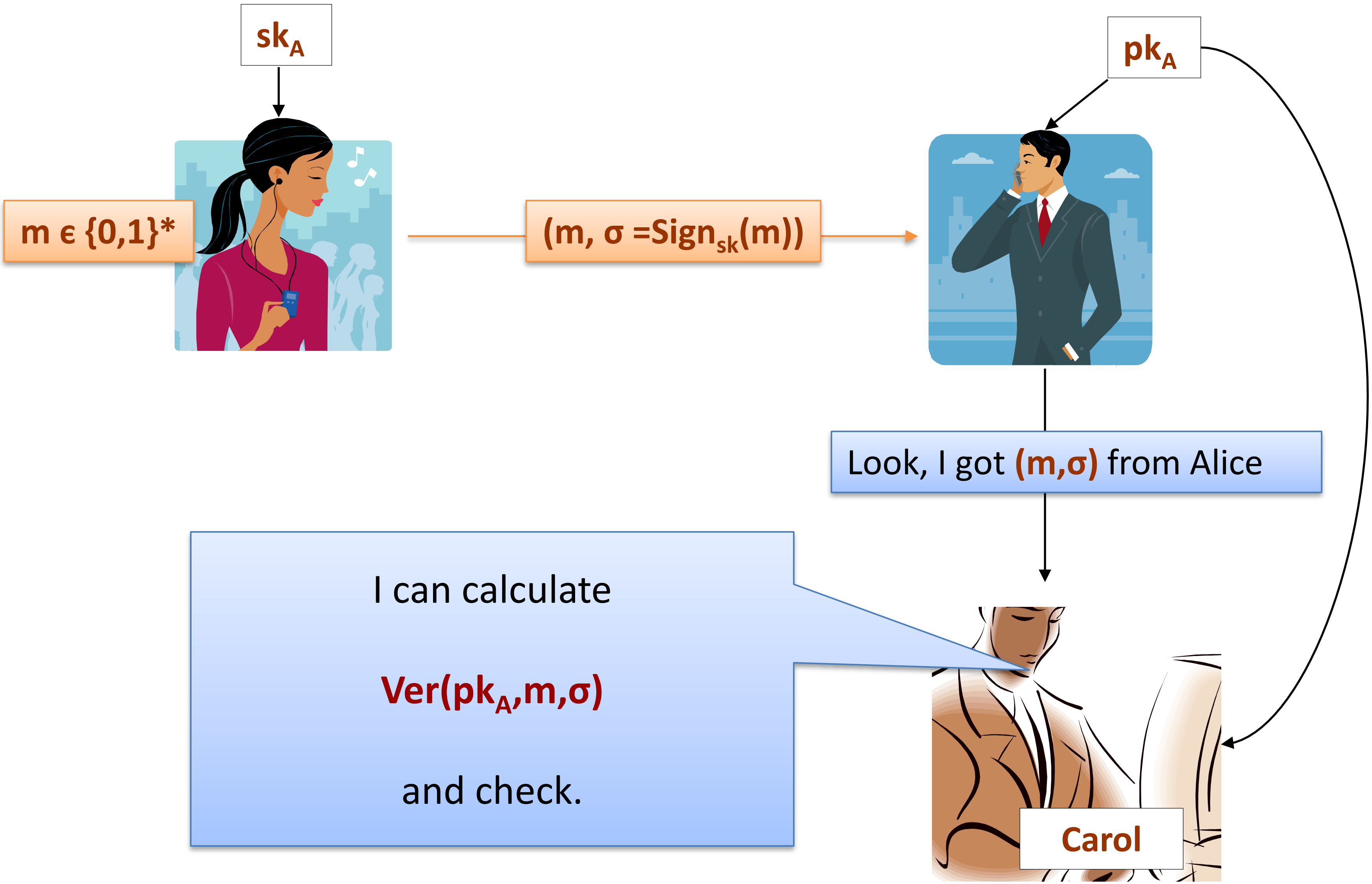
Anyone can verify the signatures



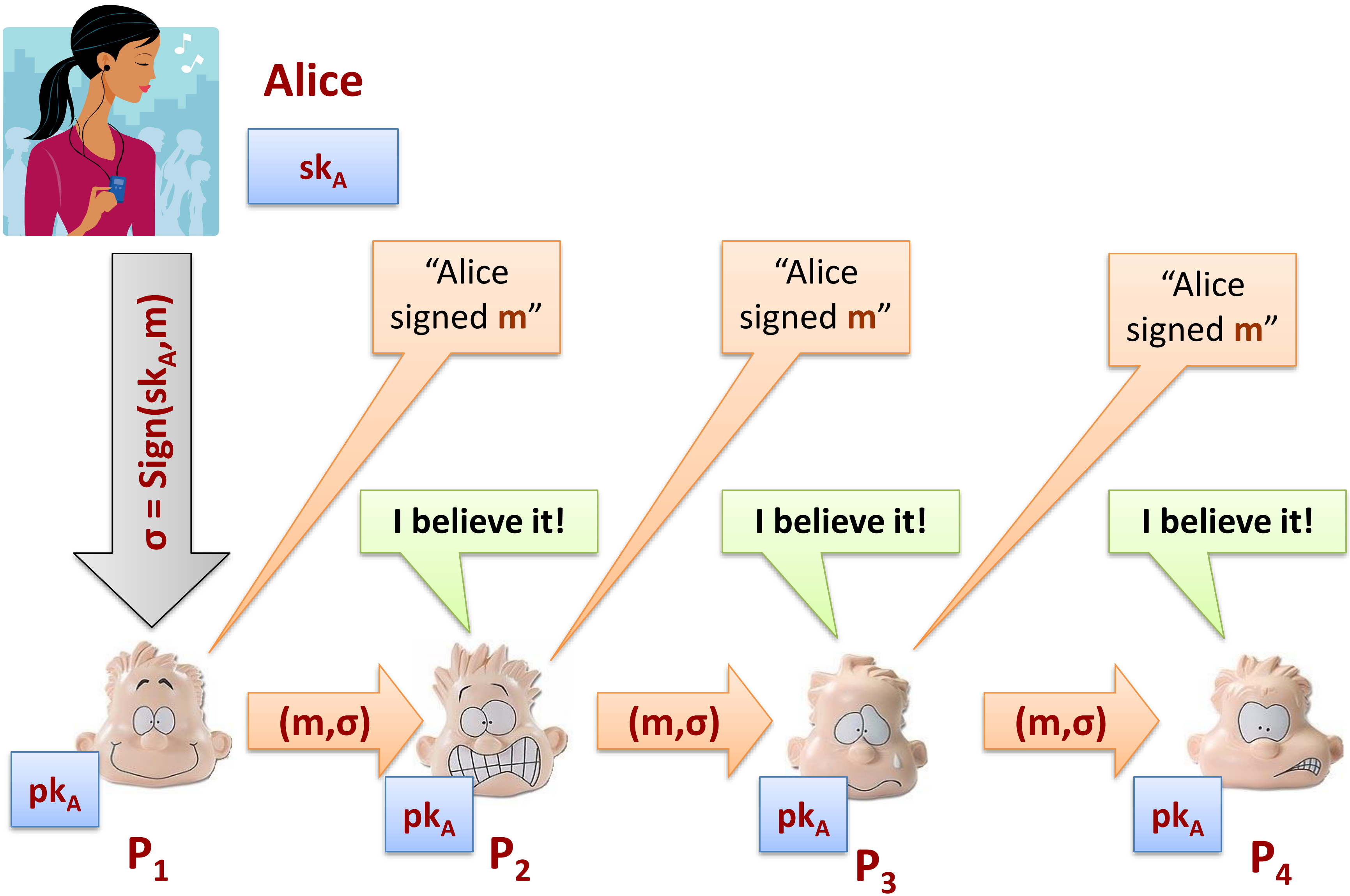
Look at the MACs...



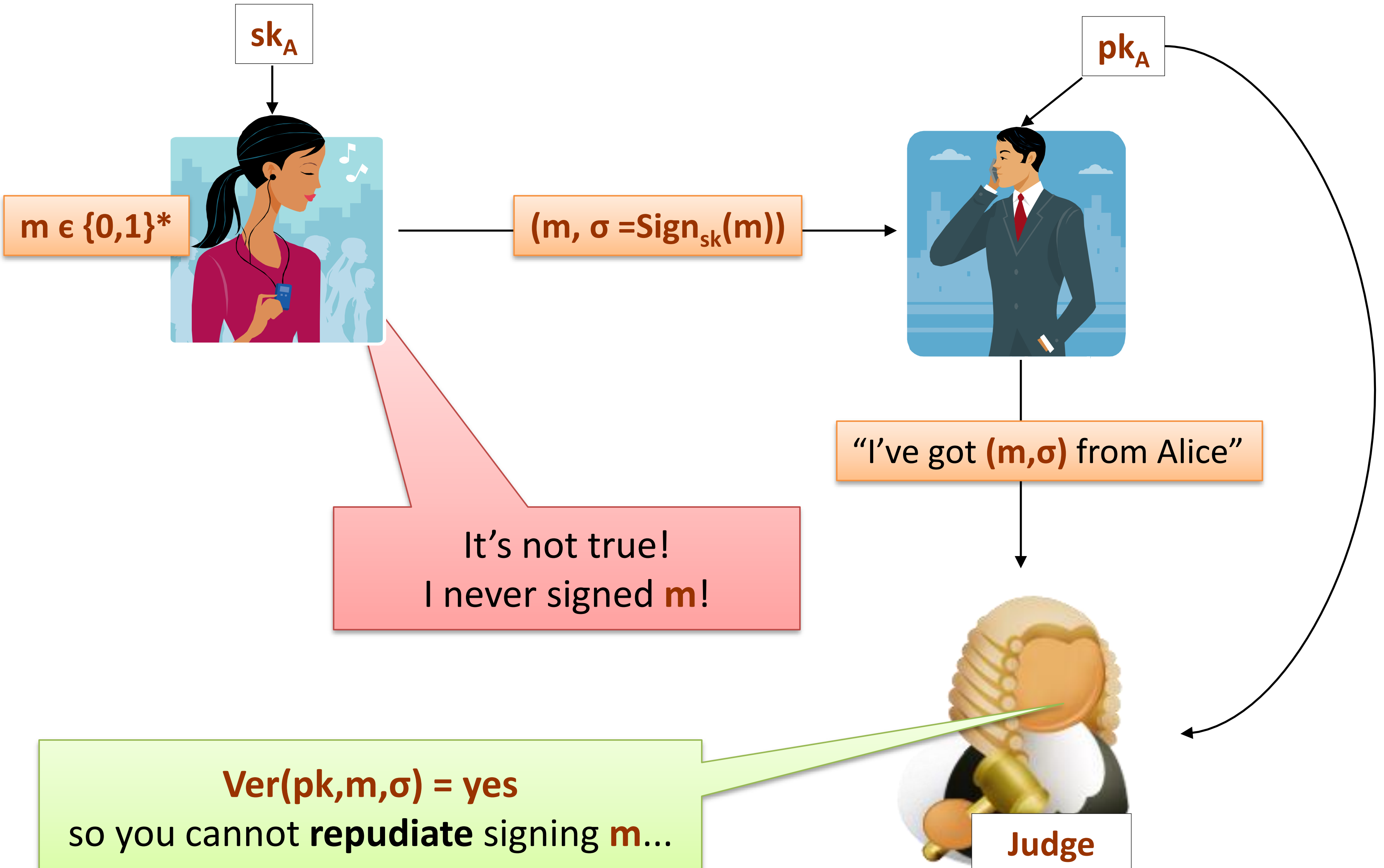
Signatures are publicly-verifiable!



So, the signatures are transferable



Non-repudiation



Digital Signature Schemes

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

- the **key-generation** algorithm **Gen** takes as input a security parameter **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 **Ver** takes as input a key **pk**, a message **m** and a signature **σ**, and outputs a bit **$b \in$ {yes, no}.**

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

Correctness

We require that it always holds that:

$Ver_{pk}(m, Sign_{sk}(m))$ = Yes with high probability

What remains is to define **security**.

How to define security?

We have to assume that the adversary can see some pairs

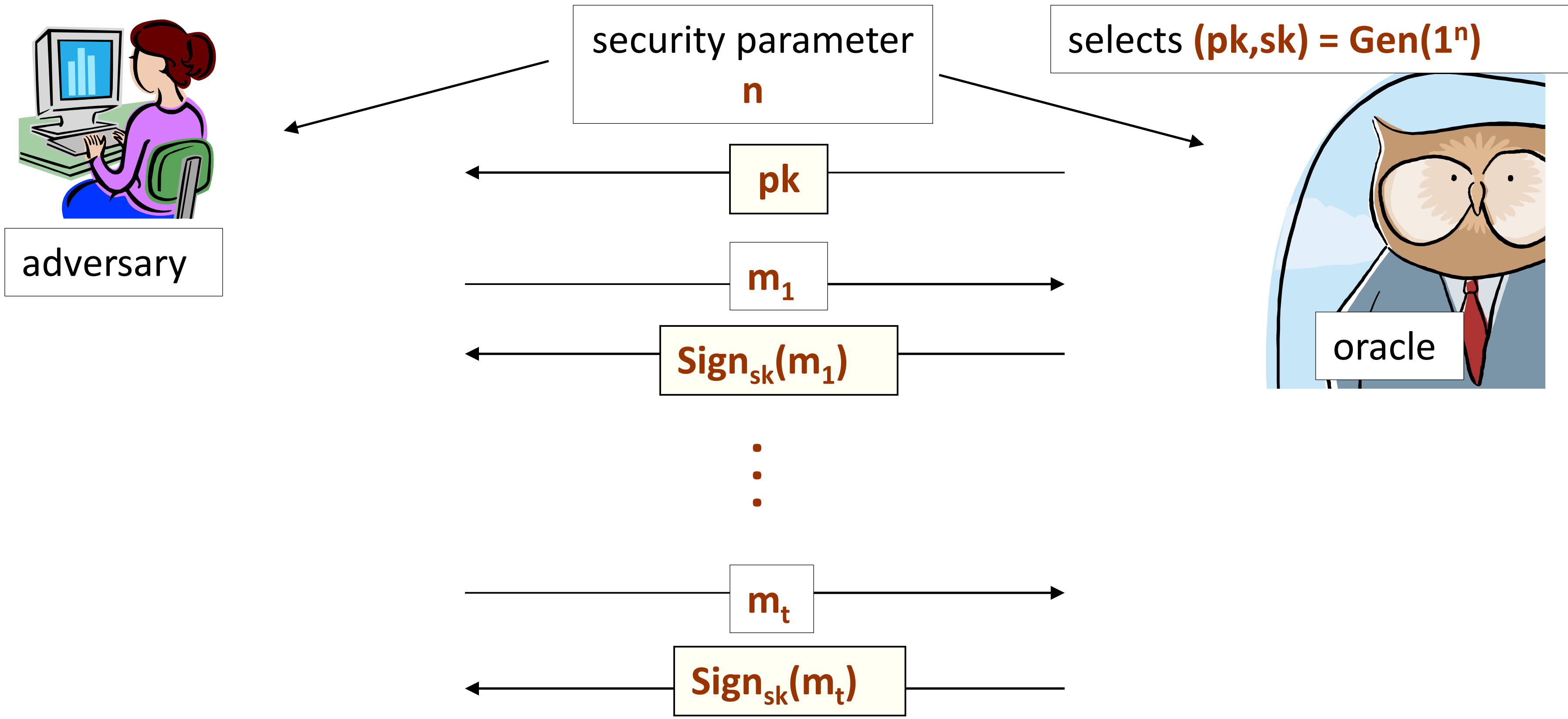
$(m_1,\sigma_1),\ldots,(m_t,\sigma_t)$

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.

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. **Ver(m’, σ’) = 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,Ver)** is **existentially unforgeable under an adaptive chosen-message attack** if

∀ polynomial-time adversary **A**   **P(A breaks it)** is negligible (in **n**)

Security experiment for Signatures

- Experiment $\mathrm{Exp}_{\Pi,A}^{\mathrm{Sign}}(n)$:
  1. Choose $(pk,sk) \leftarrow Gen(n)$
  2. $m,\sigma \leftarrow A^{Sign_{sk}()}(pk)$
  3. Output 1 if $Ver_{pk}(m, \sigma) = 1$ and *m* was not queried to the Sign() oracle
  4. Output 0 otherwise

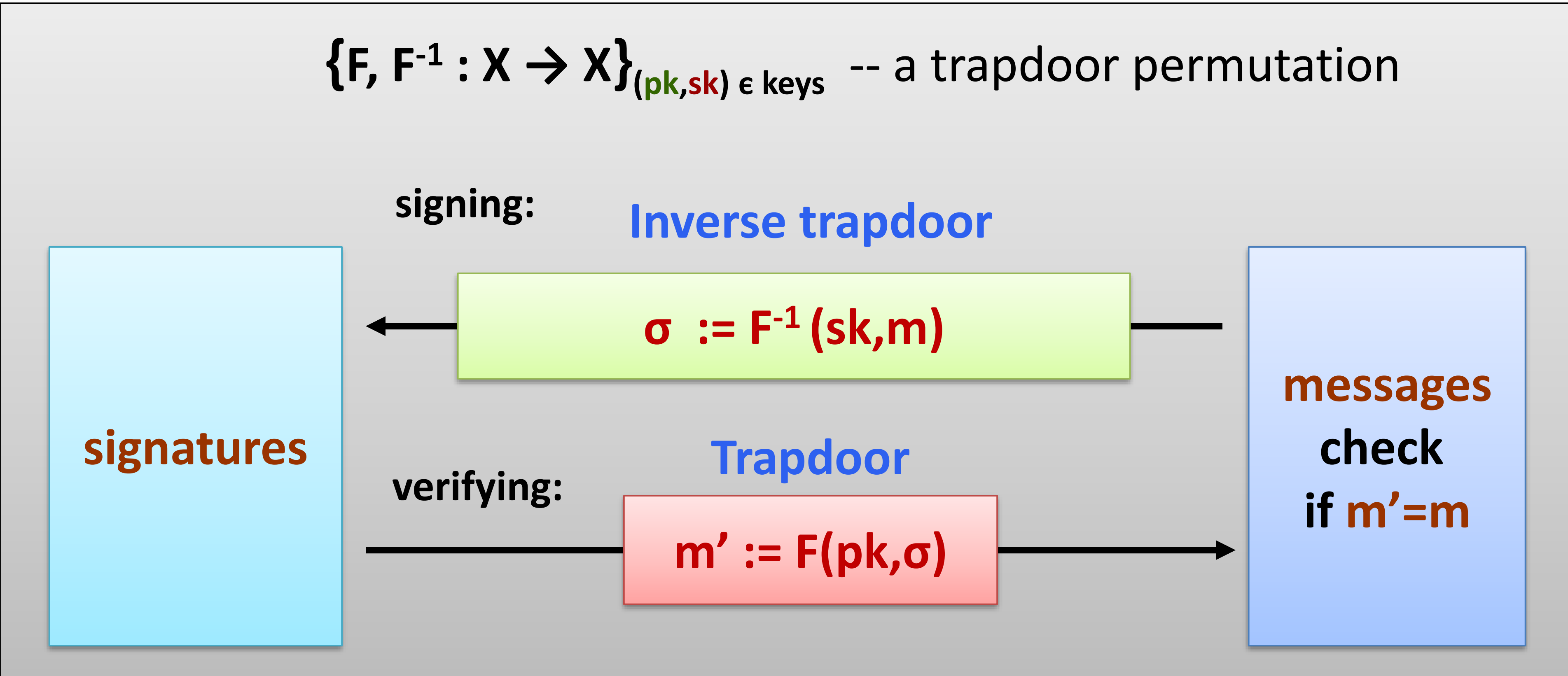
(Gen,Tag,Ver) is **a secure (existential unforgeable)** signature if:

For every **PPT** adversary A:

Pr[$\mathrm{Exp}_{\Pi,A}^{\mathrm{Sign}}(n)$ = 1] is negligible in n

How to design secure signature schemes?

Remember this idea?



In general it’s not that simple.

The "handbook RSA signatures"

$N = pq$ - **RSA** modulus

e is such that **$\gcd(e, \phi(N)) = 1$**,
d is such that **$ed = 1 \pmod{\phi(N)}$**

Correctness:

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

$$\mathrm{Sign}_{(d,N)}(m) = m^d \bmod N$$

and

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

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

$$\mathbf{m = \sigma^e \bmod N.}$$

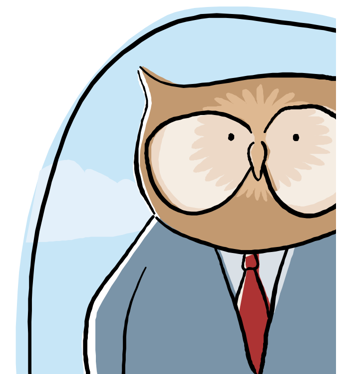
Trivially, **σ** is a valid signature on **m.**

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

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

Use the homomorphic properties of RSA.

adversary



oracle

← **(N,e)**

m_1 →

← **$Sign_{sk}(m_1) = m_1^d \bmod N$**

m_2 →

← **$Sign_{sk}(m_2) = m_2^d \bmod N$**

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**

Solution

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

$N = pq$, such that **p** and **q** are large random primes
e is such that **$\gcd(e, \phi(N)) = 1$**
d is such that **$ed = 1 \pmod{\phi(N)}$**

$Sign_d$: $Z_N^* \rightarrow Z_N^*$ is defined as:

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

Ver_e is defined as:

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

Hash-and-sign paradigm

Fact (security of the **Full Domain Hash**)

- Let **$H : \{0,1\}^* \rightarrow Z_N^*$** be a hash function modeled as a **random function.**
- Suppose the **RSA assumption** holds

Then the "**hashed RSA**" is existentially unforgeable signature

hashed RSA

$N = pq$, such that **p** and **q** are large random primes
e is such that **$\gcd(e, \phi(N)) = 1$**
d is such that **$ed = 1 \pmod{\phi(N)}$**

$Sign_d: Z_N^* \rightarrow Z_N^*$ is defined as:
$Sign(m) = H(m)^d \bmod N.$

Ver_e is defined as:
$Ver_e(m,\sigma) = yes$
iff $\sigma^e = H(m) \pmod N$

Other popular signature schemes

- **Rabin** signatures (based on squaring mod N=pq)

Based on discrete log:

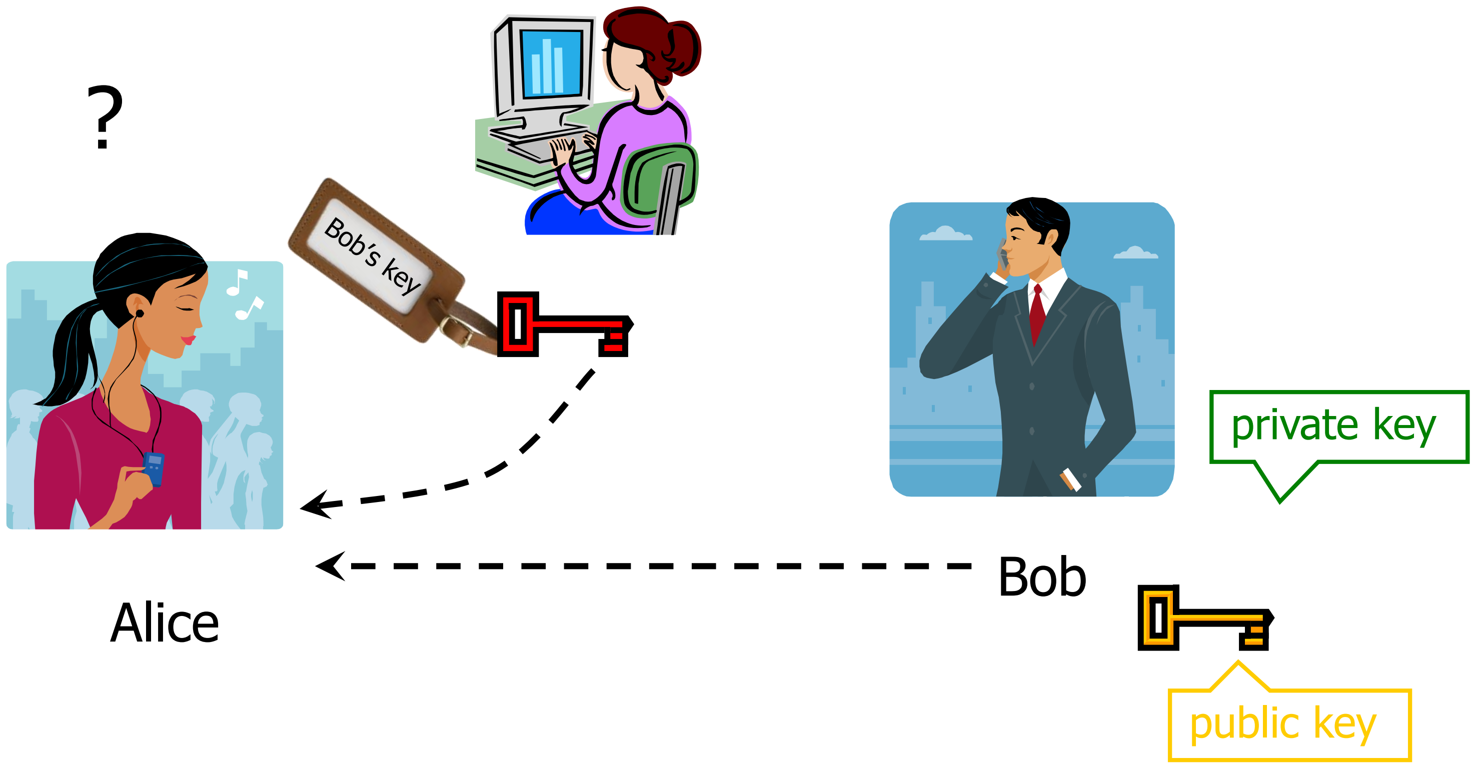
- **ElGamal** signatures
- Digital Signature Standard (**DSS**)
- **Schnorr** signatures

(also based on other groups – **elliptic curves**)

Secure communication on the Internet

- Generate public key, secret key pair
  - Using Miller-Rabin primality testing
- Distribute the Public Key
  - Using digital signatures and PKI
- Generate and share secret key
  - Using Public Key CCA secure encryption
- Communicate securely
  - Using symmetric-key authenticated encryption

Authenticity of Public Keys

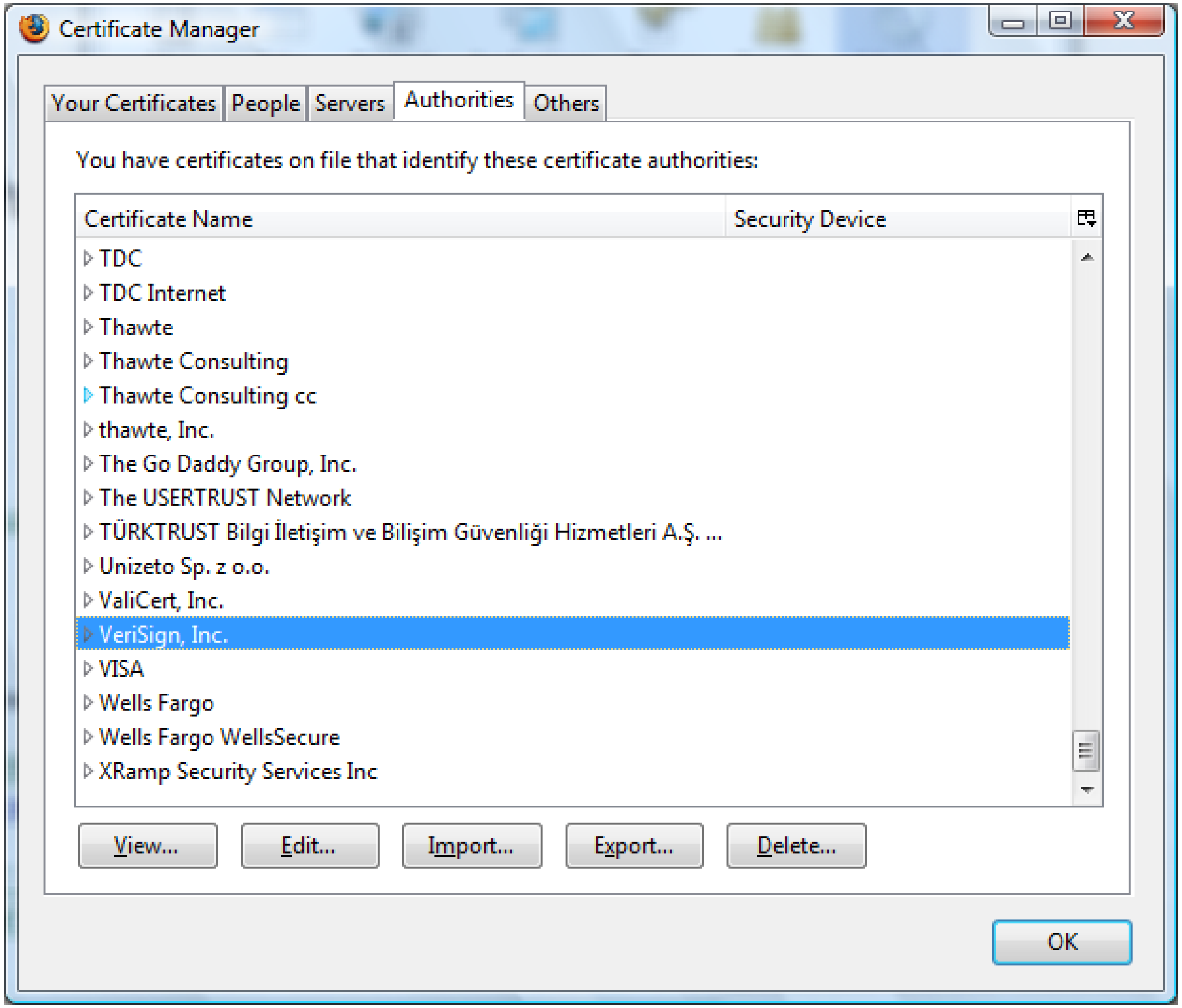


Problem: How does Alice know that the public key she received is really Bob's public key?

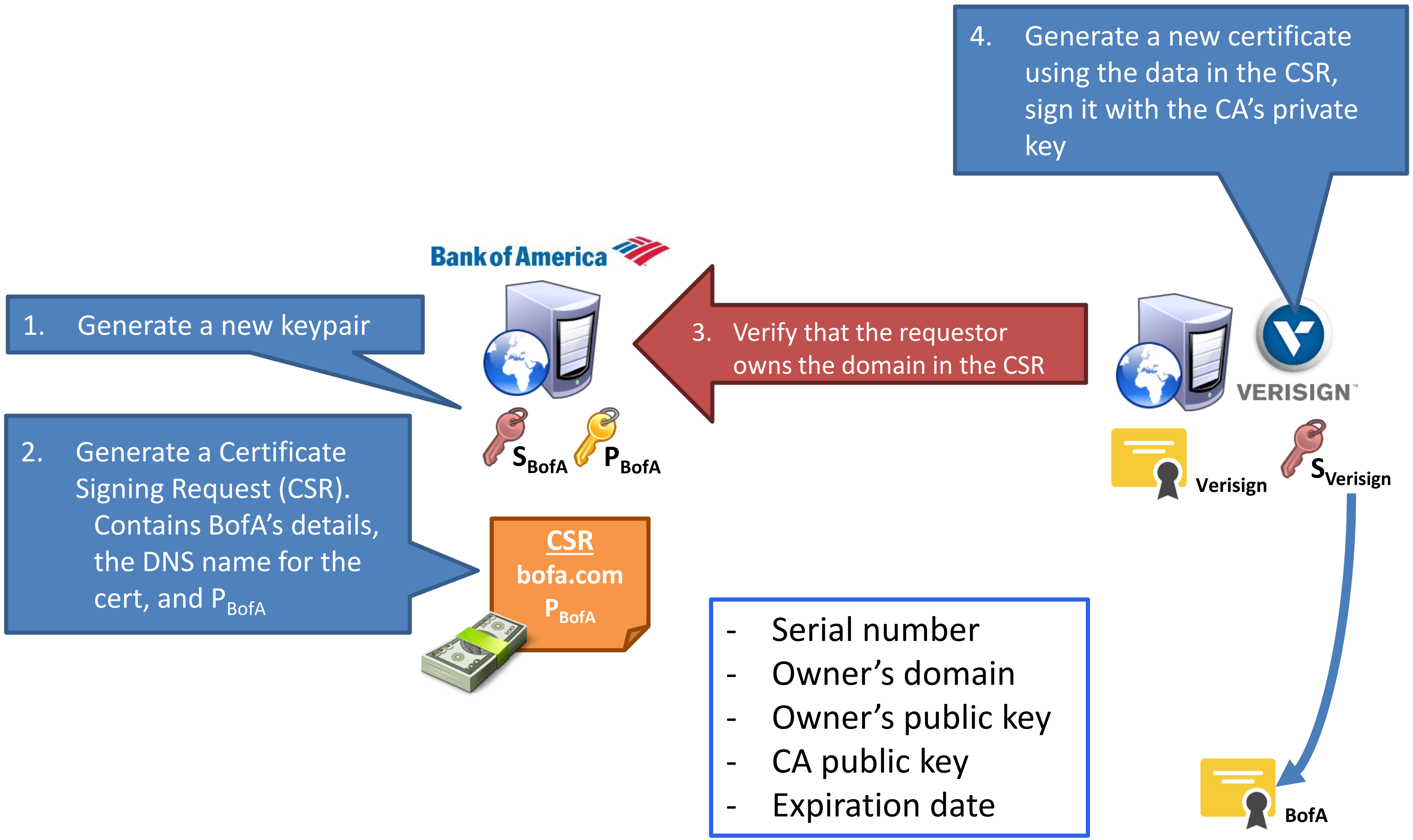
Distribution of Public Keys

- Public announcement or public directory
  - Risks: forgery and tampering
- Public-key certificate
  - Signed statement specifying the key and identity
    - Sig_{Alice}(“Bob”, PK_{Bob})
  - Could Bob sign his own certificate?
- Common approach: certificate authority (CA)
  - An agency responsible for certifying public keys
  - It generates certificates for domain names (example.com) on the web

Trusted Certificate Authorities



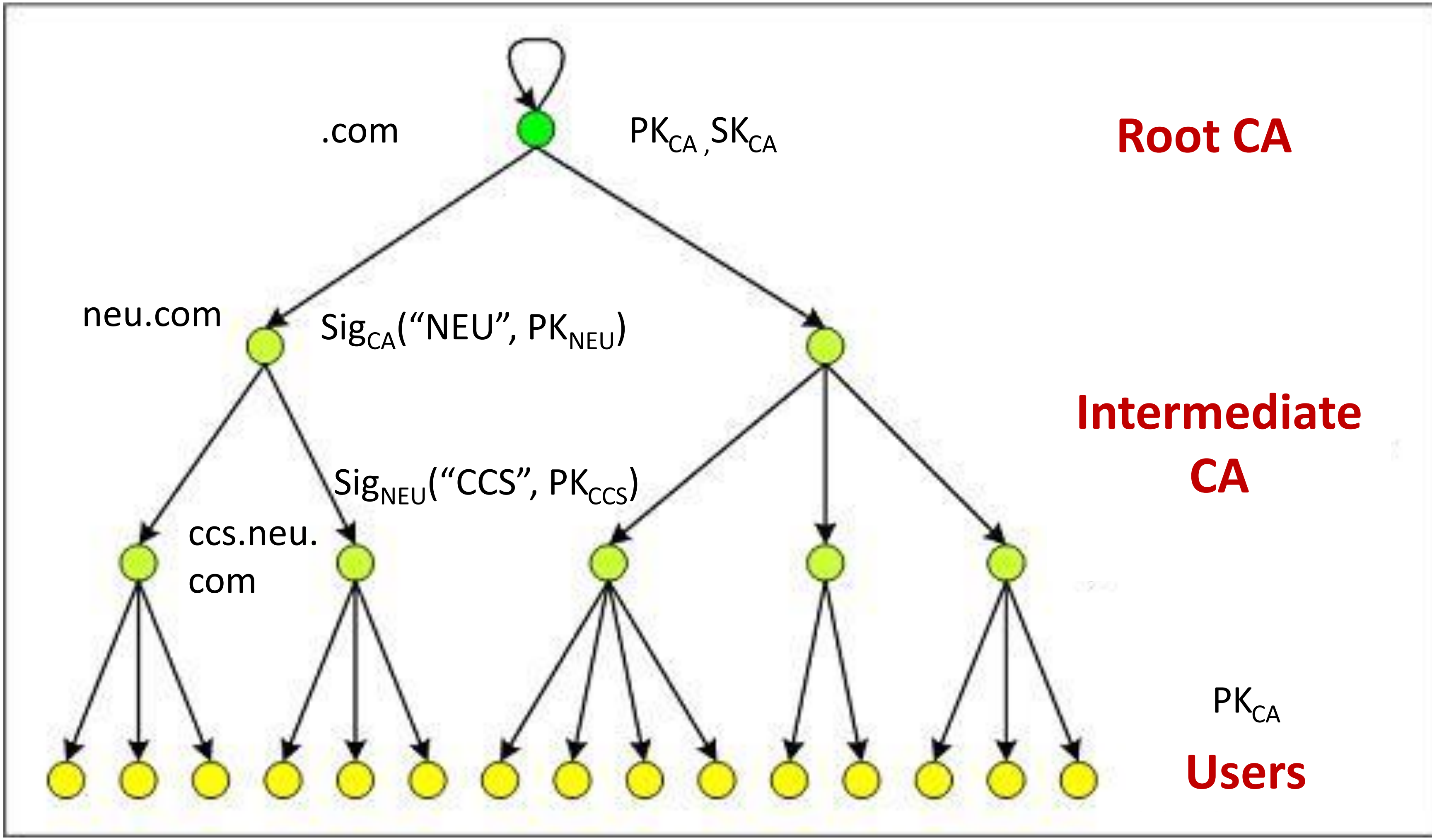
Acquiring a Certificate



CA Hierarchy or PKI

- Browsers, operating systems, etc. have trusted root certificate authorities
  - Firefox 3 includes certificates of 135 trusted root CAs
- A Root CA signs certificates for intermediate CAs, they sign certificates for lower-level CAs, etc.
  - Certificate "chain of trust"
    - $Sig_{Verisign}$("NEU", PK_{NEU}), Sig_{NEU}("CCS", PK_{CCS})
- CA is responsible for verifying the identities of certificate requestors, domain ownership

Certificate Hierarchy - PKI



Comodo

Independent Iranian hacker claims responsibility for Comodo hack

Posts claiming to be from an Iranian hacker responsible for the Comodo hack …

by **Peter Bright** - Mar 28 2011, 11:15am EDT

65

```
Hello

I'm writing this to the world, so you'll know more about me..

At first I want to give some points, so you'll be sure I'm the hacker:

I hacked Comodo from InstantSSL.it, their CEO's e-mail address mfpenco@mfpenco.com
Their Comodo username/password was: user: gtadmin password: [trimmed]
Their DB name was: globaltrust and instantsslcms
```

The alleged hacker's claim of responsibility on pastebin.com

The hack that resulted in Comodo creating certificates for popular e-mail providers including Google Gmail, Yahoo Mail, and Microsoft Hotmail has been claimed as the work of an independent Iranian patriot. A post made to data sharing site pastebin.com by a person going by the handle "comodohacker" claimed responsibility for the hack and described details of the attack. A second post provided source code apparently reverse-engineered as one of the parts of the attack.

What if CA secret key is compromised?

Recover from secret key compromise

- Revocation is <u>very</u> important
- Many valid reasons to revoke a certificate
  - Private key corresponding to the certified public key has been compromised
  - User stopped paying his certification fee to the CA and the CA no longer wishes to certify him
  - CA's certificate has been compromised!
- Methods
  - Certificate expiration
  - Certificate revocation
    - Certificate Revocation Lists (CRL)
    - Online Certificate Status Protocol (OCSP)

Key insights

- Digital signature schemes
  - Analogs of MACs in public-key setting
  - Public verifiability
  - Transferability
  - Non-repudiation
- Constructions
  - Hash-and-sign: Full-Domain Hash RSA
- PKI infrastructure
  - Distribute public keys
  - Hierarchical CA model
  - Single CA compromise can result in breaches
  - Revocation has a number of issues in practice

Acknowledgement

Some of the slides and slide contents are taken from
http://www.crypto.edu.pl/Dziembowski/teaching
and fall under the following:

We have also used slides from Prof. Dan Boneh online cryptography course at Stanford University:

http://crypto.stanford.edu/~dabo/courses/OnlineCrypto/