EE309 Advanced Programming Techniques for EE

Lecture 23: Public key cryptography

INSU YUN (윤인수)

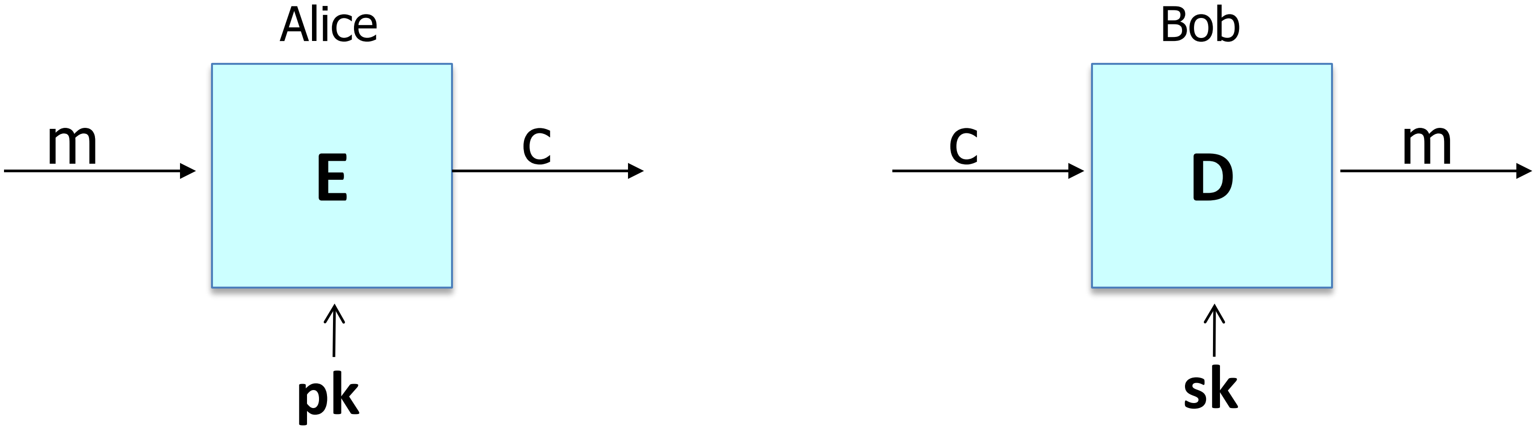
School of Electrical Engineering, KAIST

[Slides from Cryptography at Coursera by Dan boneh]

Public key encryption: definitions and security

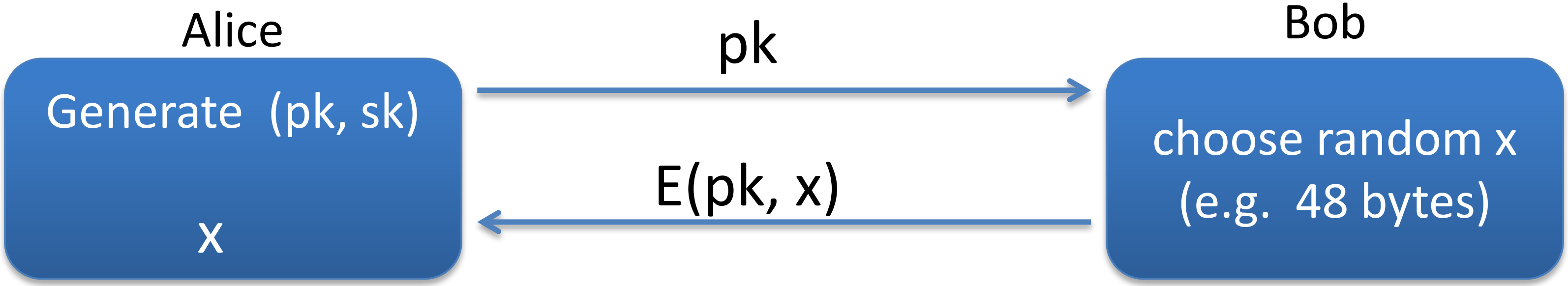
Public key encryption

Bob: generates (PK, SK) and gives PK to Alice



Applications

Session setup (for now, only eavesdropping security)



Alice

Generate (pk, sk)

x

pk

E(pk, x)

Bob

choose random x

(e.g. 48 bytes)

Non-interactive applications: (e.g. Email)

- Bob sends email to Alice encrypted using pk_{alice}
- Note: Bob needs pk_{alice} (public key management)

Public key encryption

Def: a public-key encryption system is a triple of algs. (G, E, D)

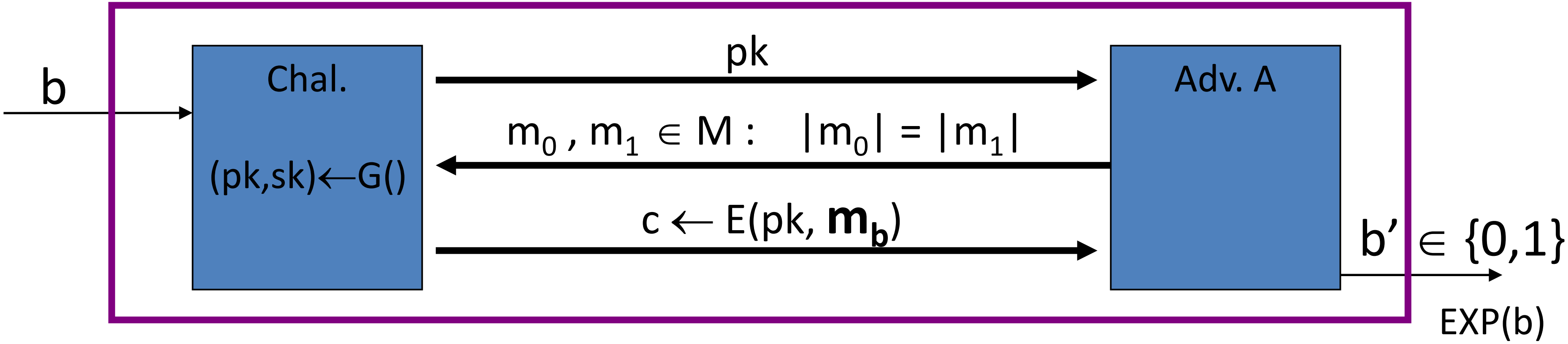
- G(): randomized alg. outputs a key pair (pk, sk)
- E(pk, m): randomized alg. that takes $m \in M$ and outputs $c \in C$
- D(sk,c): det. alg. that takes $c \in C$ and outputs $m \in M$ or $\perp$

Consistency: $\forall$(pk, sk) output by G :

$$\forall m \in M: \quad D(sk, E(pk, m)) = m$$

Security: eavesdropping

For b=0,1 define experiments EXP(0) and EXP(1) as:



Def: $\mathbb{E}$ =(G,E,D) is sem. secure (a.k.a IND-CPA) if for all efficient A:

$$\mathrm{Adv}_{SS}[A,\mathbb{E}] = \Big| \Pr[\mathrm{EXP}(0)=1] - \Pr[\mathrm{EXP}(1)=1] \Big| < \text{negligible}$$

Relation to symmetric cipher security

Recall: for symmetric ciphers we had two security notions:

- One-time security and many-time security (CPA)
- We showed that one-time security $\not\Rightarrow$ many-time security

For public key encryption:

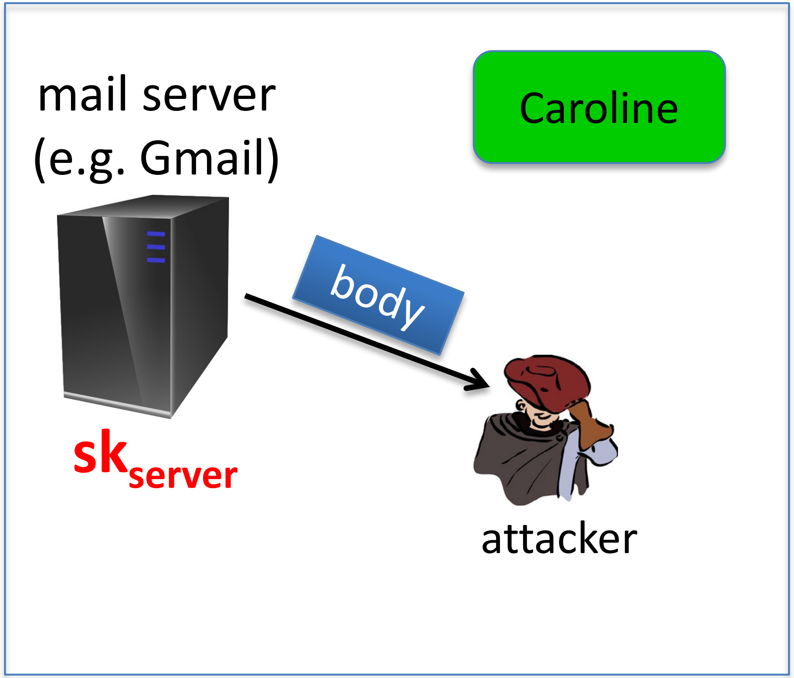
- One-time security $\Rightarrow$ many-time security (CPA)

  (follows from the fact that attacker can encrypt by himself)

- Public key encryption **must** be randomized

Security against active attacks

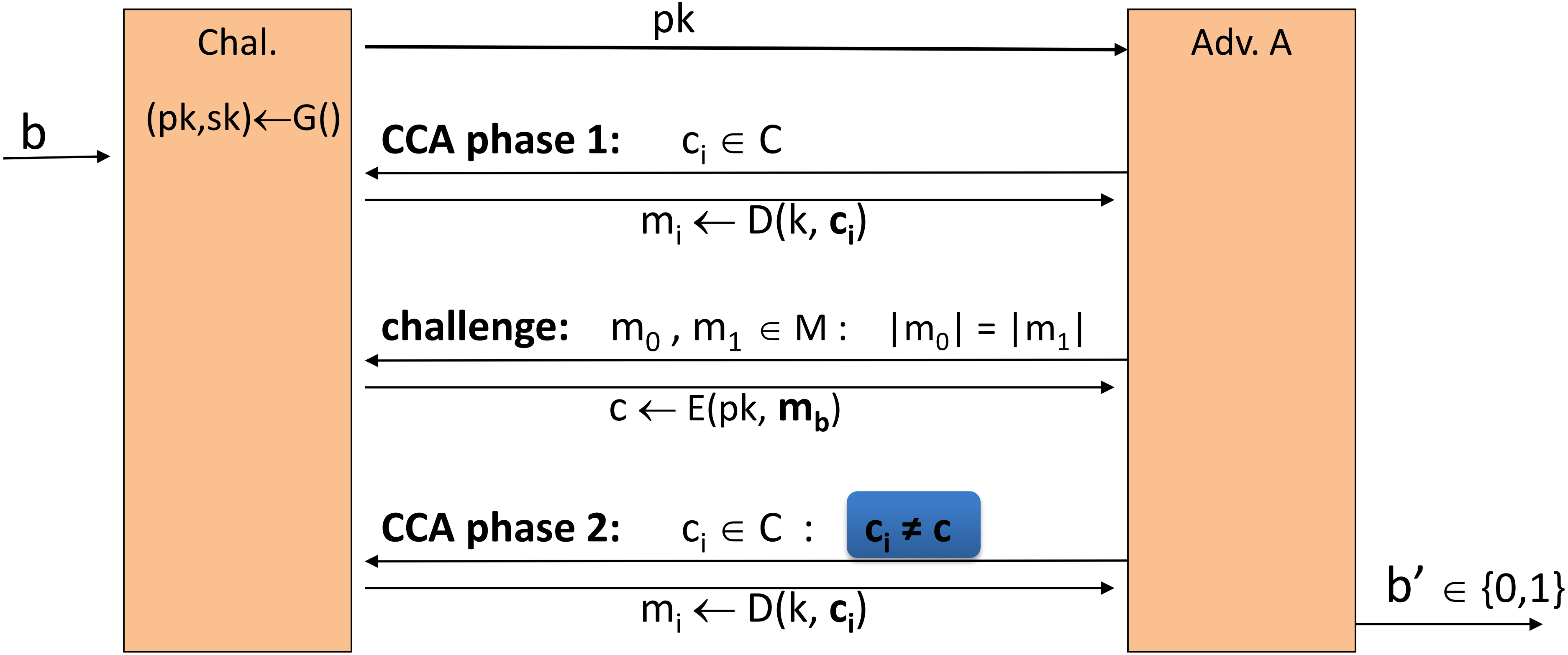
What if attacker can tamper with ciphertext?



Attacker is given decryption of msgs
that start with **"to: attacker"**

(pub-key) Chosen Ciphertext Security: definition

$\mathbb{E}$ = (G,E,D) public-key enc. over (M,C). For b=0,1 define EXP(b):



Chal.

(pk,sk)⟵G()

b

pk

Adv. A

CCA phase 1: $c_i \in C$

$m_i \leftarrow D(k, \mathbf{c_i})$

challenge: $m_0, m_1 \in M : \quad |m_0| = |m_1|$

$c \leftarrow E(pk, \mathbf{m_b})$

CCA phase 2: $c_i \in C$: $\mathbf{c_i \neq c}$

$m_i \leftarrow D(k, \mathbf{c_i})$

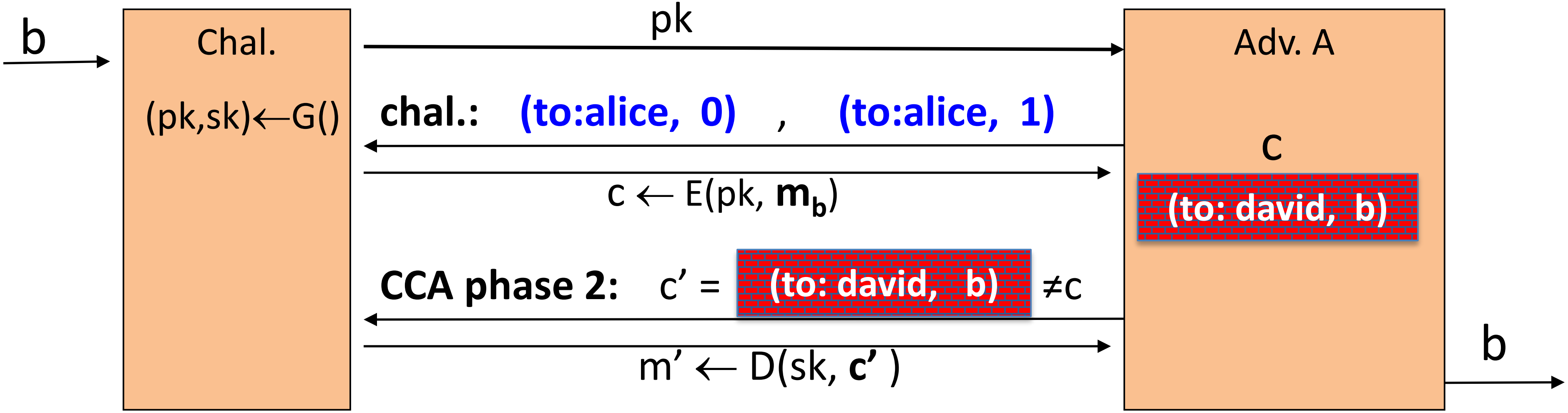
$b' \in \{0,1\}$

Chosen ciphertext security: definition

Def: $\mathbb{E}$ is CCA secure (a.k.a IND-CCA) if for all efficient A:

$\mathrm{Adv}_{\mathrm{CCA}}[A,\mathbb{E}] = \left|\Pr[\mathrm{EXP}(0)=1] - \Pr[\mathrm{EXP}(1)=1]\right|$ is negligible.

Example: Suppose **(to: alice, body)** ⟶ **(to: david, body)**



b

Chal.

(pk,sk)←G()

pk

chal.: **(to:alice, 0)** , **(to:alice, 1)**

$c \leftarrow E(pk, \mathbf{m_b})$

CCA phase 2: c′ = **(to: david, b)** ≠c

$m' \leftarrow D(sk, \mathbf{c'})$

Adv. A

c

(to: david, b)

b

Active attacks: symmetric vs. pub-key

Recall: secure symmetric cipher provides **authenticated encryption**

[chosen plaintext security & ciphertext integrity **]**

- Roughly speaking: **attacker cannot create new ciphertexts**
- Implies security against chosen ciphertext attacks

In public-key settings:

- Attacker **can** create new ciphertexts using pk !!
- So instead: we directly require chosen ciphertext security

Public Key Encryption from trapdoor permutations: Constructions

Trapdoor functions (TDF)

Def: a trapdoor func. $X \longrightarrow Y$ is a triple of efficient algs. (G, F, F^{-1})

- $G()$: randomized alg. outputs a key pair (pk, sk)
- $F(pk,\cdot)$: det. alg. that defines a function $X \longrightarrow Y$
- $F^{-1}(sk,\cdot)$: defines a function $Y \longrightarrow X$ that inverts $F(pk,\cdot)$

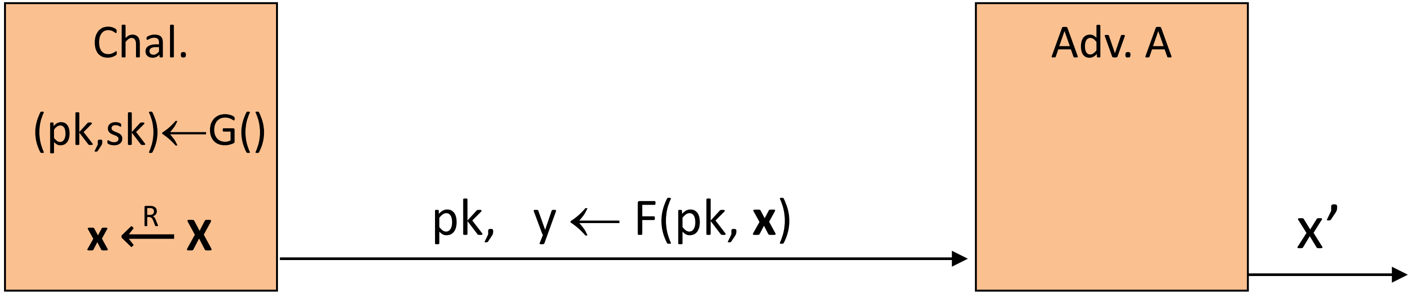
More precisely: $\forall (pk, sk)$ output by G

$$\forall x \in X: \quad F^{-1}(sk, F(pk, x)) = x$$

Secure Trapdoor Functions (TDFs)

(G, F, F^{-1}) is secure if $F(pk, \cdot)$ is a "one-way" function:

can be evaluated, but cannot be inverted without sk



Chal.

$(pk, sk) \leftarrow G()$

$\mathbf{x} \xleftarrow{R} \mathbf{X}$

$pk, \quad y \leftarrow F(pk, \mathbf{x})$

Adv. A

x'

Def: (G, F, F^{-1}) is a secure TDF if for all efficient A:

$$\mathrm{Adv}_{OW}[A,F] = \mathbf{Pr[\ x = x'\ ]} < \text{negligible}$$

Public-key encryption from TDFs

- (G, F, F^{-1}): secure TDF $X \longrightarrow Y$
- (E_s, D_s) : symmetric auth. encryption defined over (K,M,C)
- $H: X \longrightarrow K$ a hash function

We construct a pub-key enc. system (G, E, D):

Key generation G: same as G for TDF

Public-key encryption from TDFs

- (G, F, F^{-1}): secure TDF $X \longrightarrow Y$
- (E_s, D_s) : symmetric auth. encryption defined over (K,M,C)
- $H: X \longrightarrow K$ a hash function

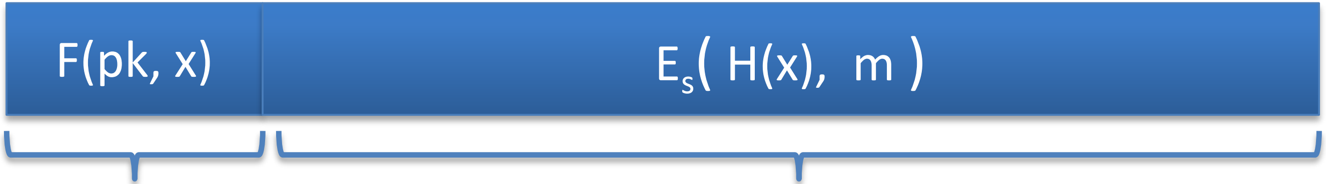
E(pk, m) :

```
    $x \xleftarrow{R} X$,    $y \longleftarrow F(pk, x)$
    $k \longleftarrow H(x)$,    $c \longleftarrow E_s(k, m)$
    output $(y, c)$
```

D(sk, (y,c)) :

```
    $x \longleftarrow F^{-1}(sk, y)$,
    $k \longleftarrow H(x)$,    $m \longleftarrow D_s(k, c)$
    output $m$
```

In pictures:



$F(pk, x)$ $E_s(\, H(x), \; m \,)$

header body

Security Theorem:

If **(G, F, F^{-1})** is a secure TDF, **(E_s, D_s)** provides auth. enc.

and **H:** $X \longrightarrow K$ is a “random oracle”

then **(G,E,D)** is CCA^{ro} secure.

Incorrect use of a Trapdoor Function (TDF)

Never encrypt by applying F directly to plaintext:

E(pk, m) :

output $c \longleftarrow F(pk, m)$

D(sk, c) :

output $F^{-1}(sk, c)$

Problems:

- Deterministic: cannot be semantically secure !!
- Many attacks exist (next segment)

The RSA trapdoor permutation

Review: trapdoor permutations

Three algorithms: (G, F, F^{-1})

- G: outputs pk, sk. pk defines a function $F(pk, \cdot): X \to X$
- $F(pk, x)$: evaluates the function at x
- $F^{-1}(sk, y)$: inverts the function at y using sk

Secure trapdoor permutation:

The function $F(pk, \cdot)$ is one-way without the trapdoor sk

Review: arithmetic mod composites

Let $N = p \cdot q$ where p,q are prime

$$Z_N = \{0,1,2,\ldots,N-1\} \quad ; \quad (Z_N)^* = \{\text{invertible elements in } Z_N\}$$

<u>Facts</u>: $x \in Z_N$ is invertible $\Leftrightarrow$ $\gcd(x,N) = 1$

- Number of elements in $(Z_N)^*$ is $\varphi(N) = (p-1)(q-1) = N-p-q+1$

<u>Euler's thm</u>: $\boxed{\forall\, x \in (Z_N)^* \ : \ x^{\varphi(N)} = 1}$

The RSA trapdoor permutation

First published: Scientific American, Aug. 1977.

Very widely used:

- SSL/TLS: certificates and key-exchange
- Secure e-mail and file systems

… many others

The RSA trapdoor permutation

G(): choose random primes $p,q \approx 1024$ bits. Set **$N=pq$**.

choose integers **e , d** s.t. **$e \cdot d = 1 \pmod{\varphi(N)}$**

output $pk = (N, e)$, $sk = (N, d)$

F(pk, x): $\mathbb{Z}_N^* \to \mathbb{Z}_N^*$; **$\mathrm{RSA}(x) = x^e$** (in Z_N)

F^{-1}(sk, y) $= y^d$; $y^d = \mathbf{RSA(x)^d} = x^{ed} = x^{k\varphi(N)+1} = \left(x^{\varphi(N)}\right)^k \cdot \mathbf{x} = x$

The RSA assumption

RSA assumption: RSA is one-way permutation

For all efficient algs. A:

$$\Pr\left[\ A(N,e,y) = y^{1/e}\ \right] < \text{negligible}$$

where $p,q \xleftarrow{R}$ n-bit primes, $N \leftarrow pq$, $y \xleftarrow{R} Z_N^*$

Review: RSA pub-key encryption (ISO std)

(E_s, D_s): symmetric enc. scheme providing auth. encryption.

H: $Z_N \to K$ where K is key space of (E_s, D_s)

- **G**(): generate RSA params: pk = (N,e), sk = (N,d)
- **E**(pk, m):
  (1) choose random x in Z_N
  (2) $y \leftarrow RSA(x) = x^e$, $k \leftarrow H(x)$
  (3) output $(y, E_s(k,m))$
- **D**(sk, (y, c)): output $D_s\big(H\big(RSA^{-1}(y)\big), c\big)$

Textbook RSA is insecure

Textbook RSA encryption:

- public key: **(N,e)** Encrypt: $\mathbf{c \longleftarrow m^e}$ (in Z_N)
- secret key: **(N,d)** Decrypt: $\mathbf{c^d \longrightarrow m}$

Insecure cryptosystem !!

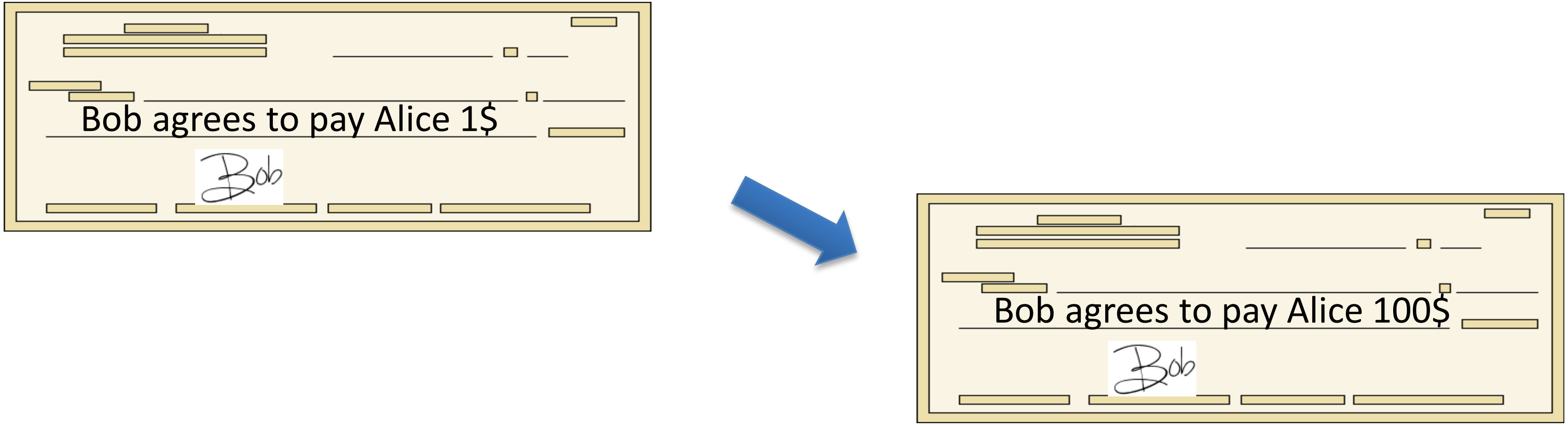
- Is not semantically secure and many attacks exist

⇒ The RSA trapdoor permutation is not an encryption scheme !

What is a digital signature?

Physical signatures

Goal: bind document to author



Bob agrees to pay Alice 1$

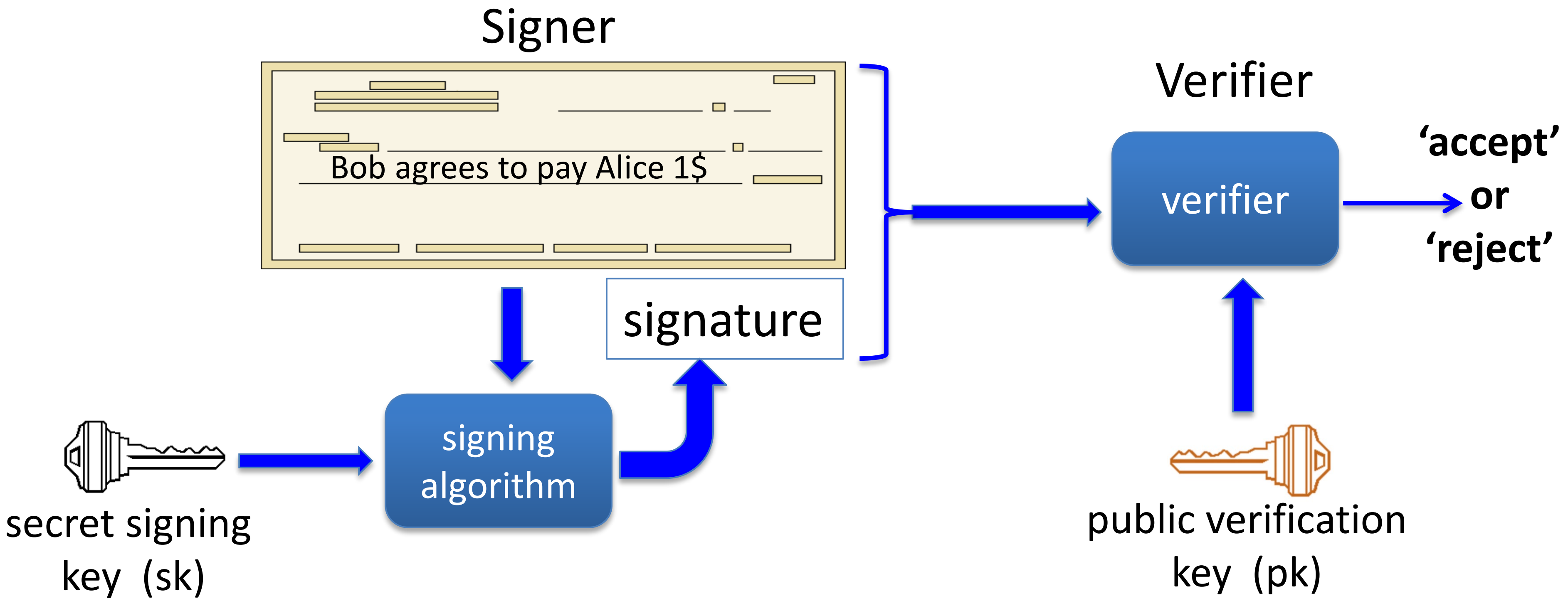
Bob agrees to pay Alice 100$

Problem in the digital world:

anyone can copy Bob’s signature from one doc to another

Digital signatures

Solution: make signature depend on document



Signer

Bob agrees to pay Alice 1$

signature

signing algorithm

secret signing key (sk)

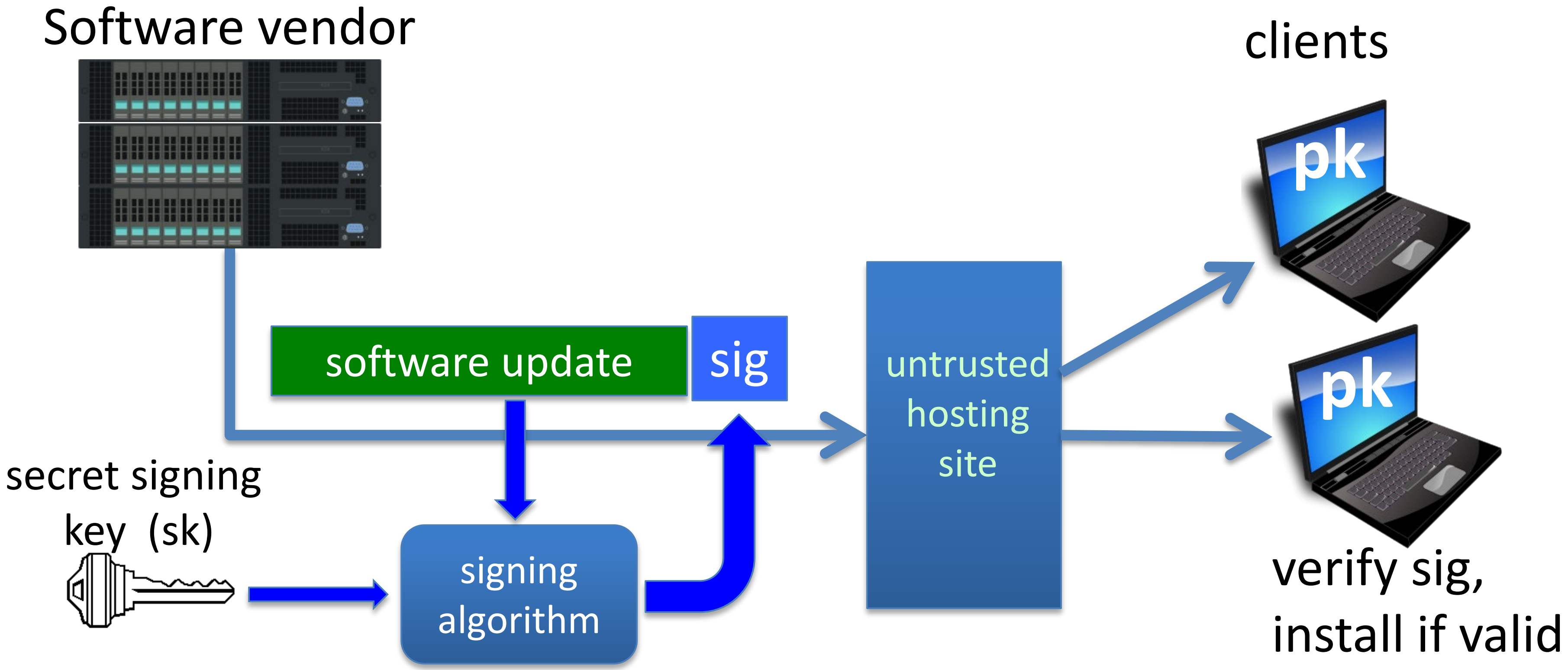
Verifier

verifier

‘accept’
or
‘reject’

public verification key (pk)

A more realistic example



Digital signatures: syntax

Def: a signature scheme (Gen,S,V) is a triple of algorithms:

- Gen(): randomized alg. outputs a key pair (pk, sk)
- S(sk, $m \in M$) outputs sig. σ
- V(pk, m, σ) outputs 'accept' or 'reject'

Consistency: for all (pk, sk) output by Gen :

$$\forall m \in M: \quad V\big(pk,\ m,\ S(sk, m)\big) = \text{'accept'}$$

Digital signatures: security

Attacker's power: **chosen message attack**

- for $m_1, m_2, \ldots, m_q$ attacker is given $\sigma_i \leftarrow S(sk, m_i)$

Attacker's goal: **existential forgery**

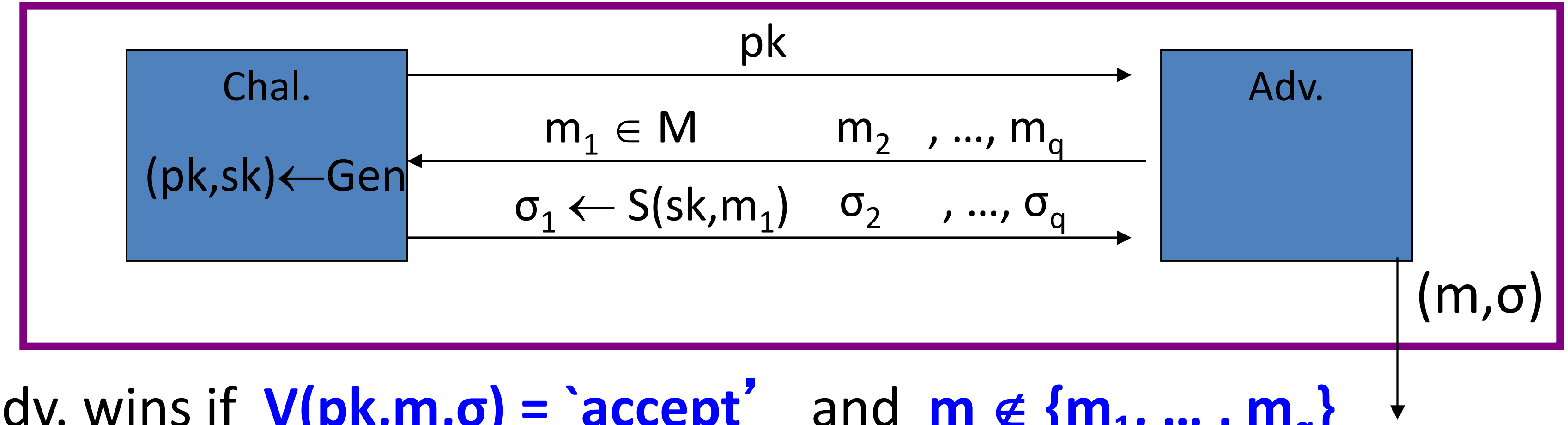
- produce some **new** valid message/sig pair (m, σ).

$$m \notin \{ m_1, \ldots, m_q \}$$

⇒ attacker cannot produce a valid sig. for a new message

Secure signatures

For a sig. scheme (Gen,S,V) and adv. A define a game as:



Chal.

$(pk,sk) \leftarrow Gen$

pk

$m_1 \in M$ $\quad m_2 \quad , \dots, m_q$

$\sigma_1 \leftarrow S(sk,m_1) \quad \sigma_2 \quad , \dots, \sigma_q$

Adv.

(m,σ)

Adv. wins if **V(pk,m,σ) = `accept'** and **$m \notin \{m_1, \dots, m_q\}$**

<u>Def</u>: SS=(Gen,S,V) is **secure** if for all "efficient" A:

$$Adv_{SIG}[A,SS] = Pr[\text{ A wins}] \quad \text{is "negligible"}$$

Let (Gen,S,V) be a signature scheme.

Suppose an attacker is able to find $m_0 \neq m_1$ such that

$$\mathbf{V(pk, m_0, \sigma) = V(pk, m_1, \sigma)} \quad \text{for all } \sigma \text{ and keys } (pk, sk) \leftarrow \text{Gen}$$

Can this signature be secure?

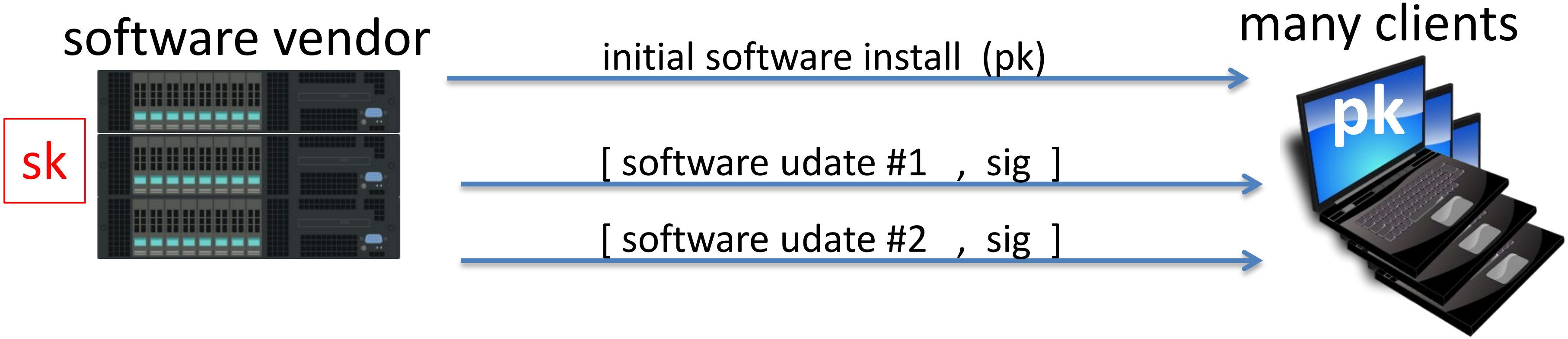
- ○ Yes, the attacker cannot forge a signature for either m_0 or m_1
- ○ No, signatures can be forged using a chosen msg attack
- ○ It depends on the details of the scheme

Applications

Applications

Code signing:

- Software vendor signs code
- Clients have vendor's pk.   Install software if signature verifies.

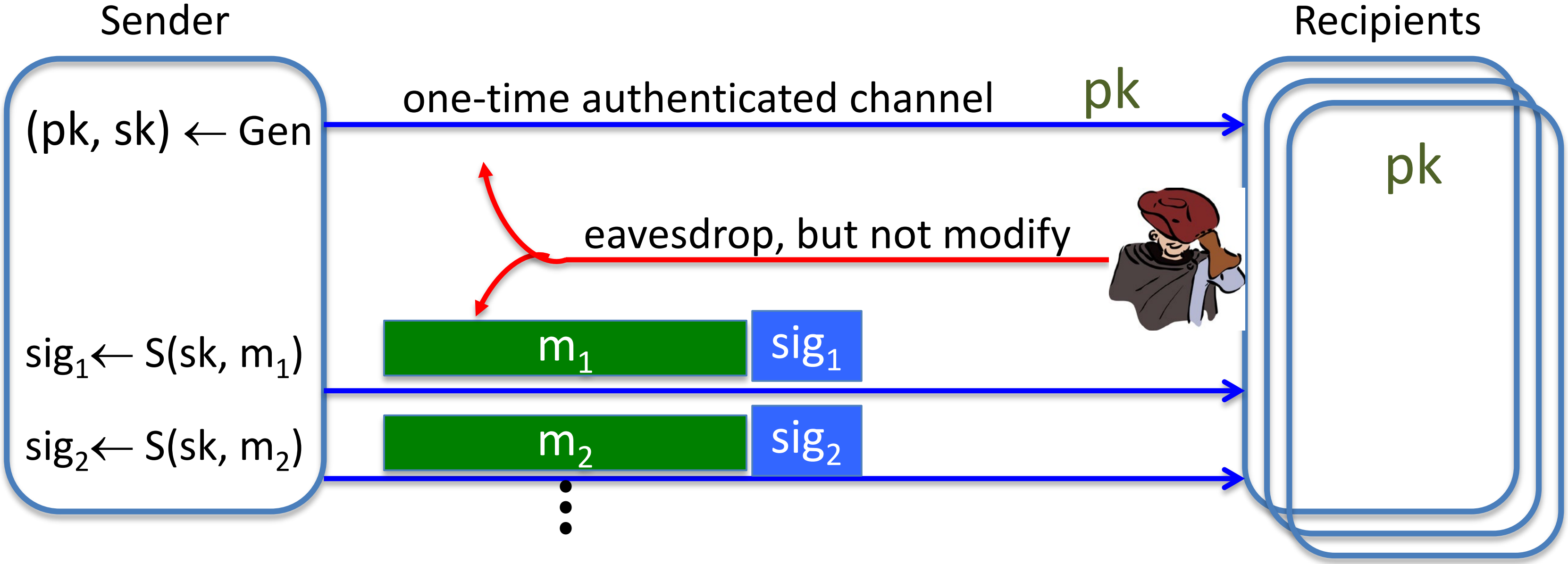


More generally:

One-time authenticated channel (non-private, one-directional)

⟹ many-time authenticated channel

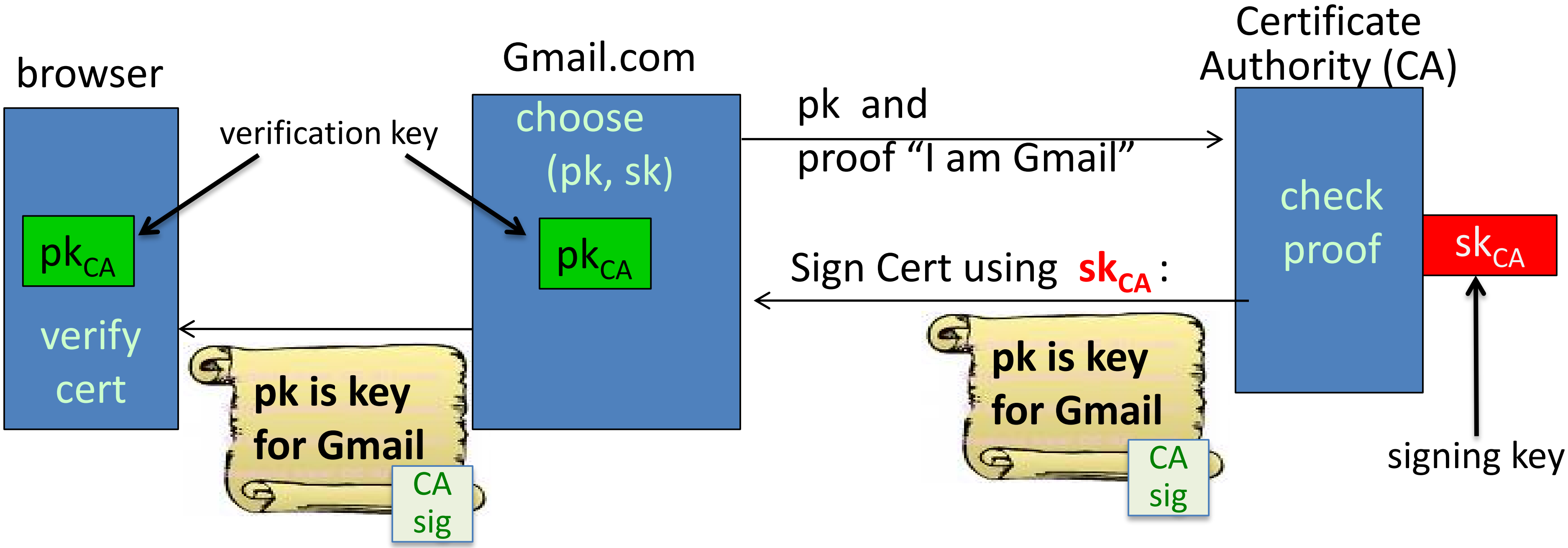
Initial software install is authenticated, but not private



Important application: Certificates

Problem: browser needs server's public-key to setup a session key

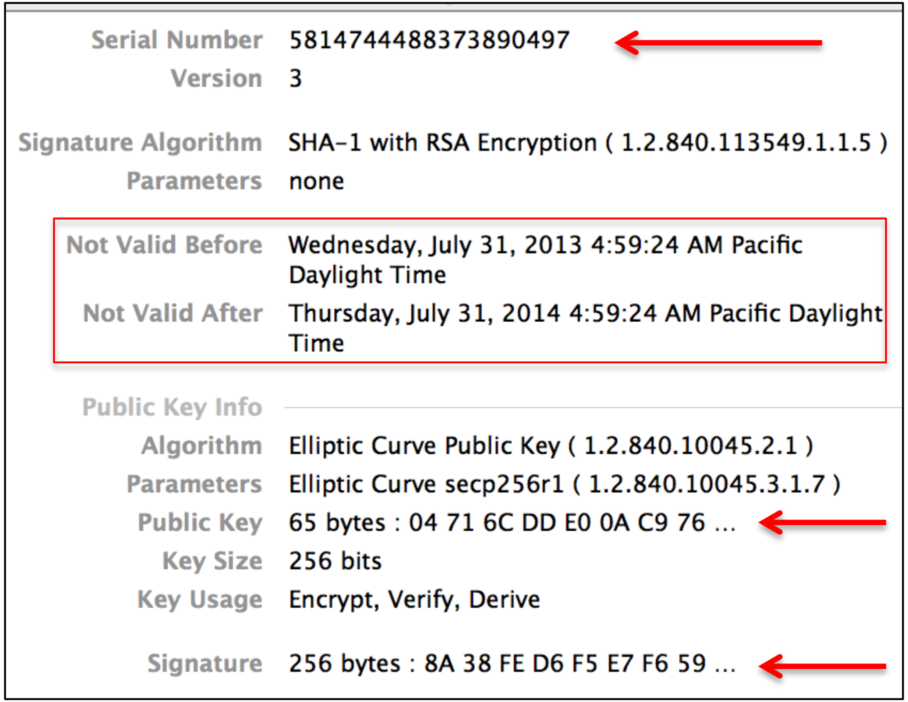
Solution: server asks trusted 3rd party (CA) to sign its public-key pk



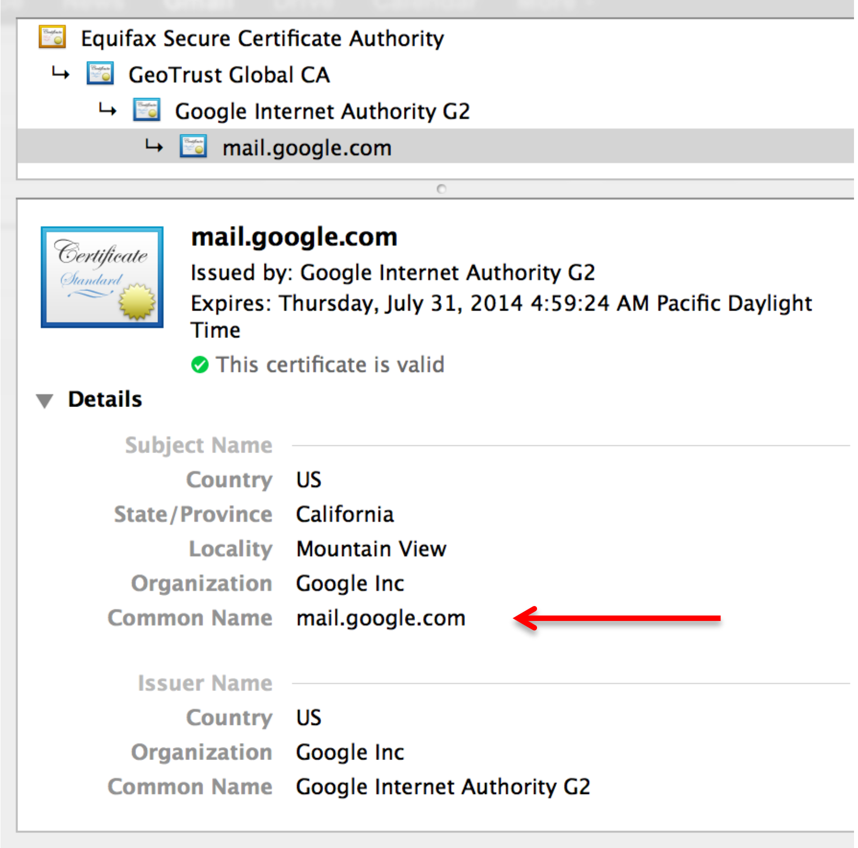
Server uses Cert for an extended period (e.g. one year)

Certificates: example

Important fields:



Serial Number	5814744488373890497
Version	3
Signature Algorithm	SHA-1 with RSA Encryption (1.2.840.113549.1.1.5)
Parameters	none
Not Valid Before	Wednesday, July 31, 2013 4:59:24 AM Pacific Daylight Time
Not Valid After	Thursday, July 31, 2014 4:59:24 AM Pacific Daylight Time
Public Key Info	
Algorithm	Elliptic Curve Public Key (1.2.840.10045.2.1)
Parameters	Elliptic Curve secp256r1 (1.2.840.10045.3.1.7)
Public Key	65 bytes : 04 71 6C DD E0 0A C9 76 …
Key Size	256 bits
Key Usage	Encrypt, Verify, Derive
Signature	256 bytes : 8A 38 FE D6 F5 E7 F6 59 …



- Equifax Secure Certificate Authority
  - ↳ GeoTrust Global CA
    - ↳ Google Internet Authority G2
      - ↳ mail.google.com

mail.google.com

Issued by: Google Internet Authority G2

Expires: Thursday, July 31, 2014 4:59:24 AM Pacific Daylight Time

✓ This certificate is valid

▼ **Details**

Subject Name	
Country	US
State/Province	California
Locality	Mountain View
Organization	Google Inc
Common Name	mail.google.com
Issuer Name	
Country	US
Organization	Google Inc
Common Name	Google Internet Authority G2

What entity generates the CA's secret key sk_{CA} ?

- ○ the browser
- ○ Gmail
- ○ the CA
- ○ the NSA

Constructions overview

Review: digital signatures

Def: a signature scheme (Gen,S,V) is a triple of algorithms:

- Gen(): randomized alg. outputs a key pair (pk, sk)
- S(sk, $m \in M$) outputs sig. σ
- V(pk, m, σ) outputs 'yes' or 'no'

Security:

- Attacker's power: chosen message attack
- Attacker's goal: existential forgery

Extending the domain with CRHF

Let **Sig**=(Gen, S, V) be a sig scheme for short messages, say $M = \{0,1\}^{256}$

Let $H: M^{big} \rightarrow M$ be a hash function (s.g. SHA-256)

Def: $\mathbf{Sig}^{big}$ = (Gen, S^{big} , V^{big}) for messages in M^{big} as:

$$S^{big}(sk, m) = S(sk, H(m)) \quad ; \quad V^{big}(pk, m, \sigma) = V(pk, H(m), \sigma)$$

Thm: If **Sig** is a secure sig scheme for M and H is collision resistant then $\mathbf{Sig}^{big}$ is a secure sig scheme for M^{big}

⟹ suffices to construct signatures for short 256-bit messages

Suppose an attacker finds two distinct messages m_0, m_1 such that $H(m_0) = H(m_1)$. Can she use this to break $\mathbf{Sig}^{\mathbf{big}}$?

○ No, $\mathbf{Sig}^{\mathbf{big}}$ is secure because the underlying scheme **Sig** is

○ It depends on what underlying scheme **Sig** is used

○ Yes, she would ask for a signature on m_0 and obtain an existential forgery for m_1

Primitives that imply signatures: TDP

Recall: $f: X \longrightarrow X$ is a **trapdoor permutation** (TDP) if:

- easy: for all $x \in X$ compute $f(x)$
- inverting f is hard, **unless one has a trapdoor**

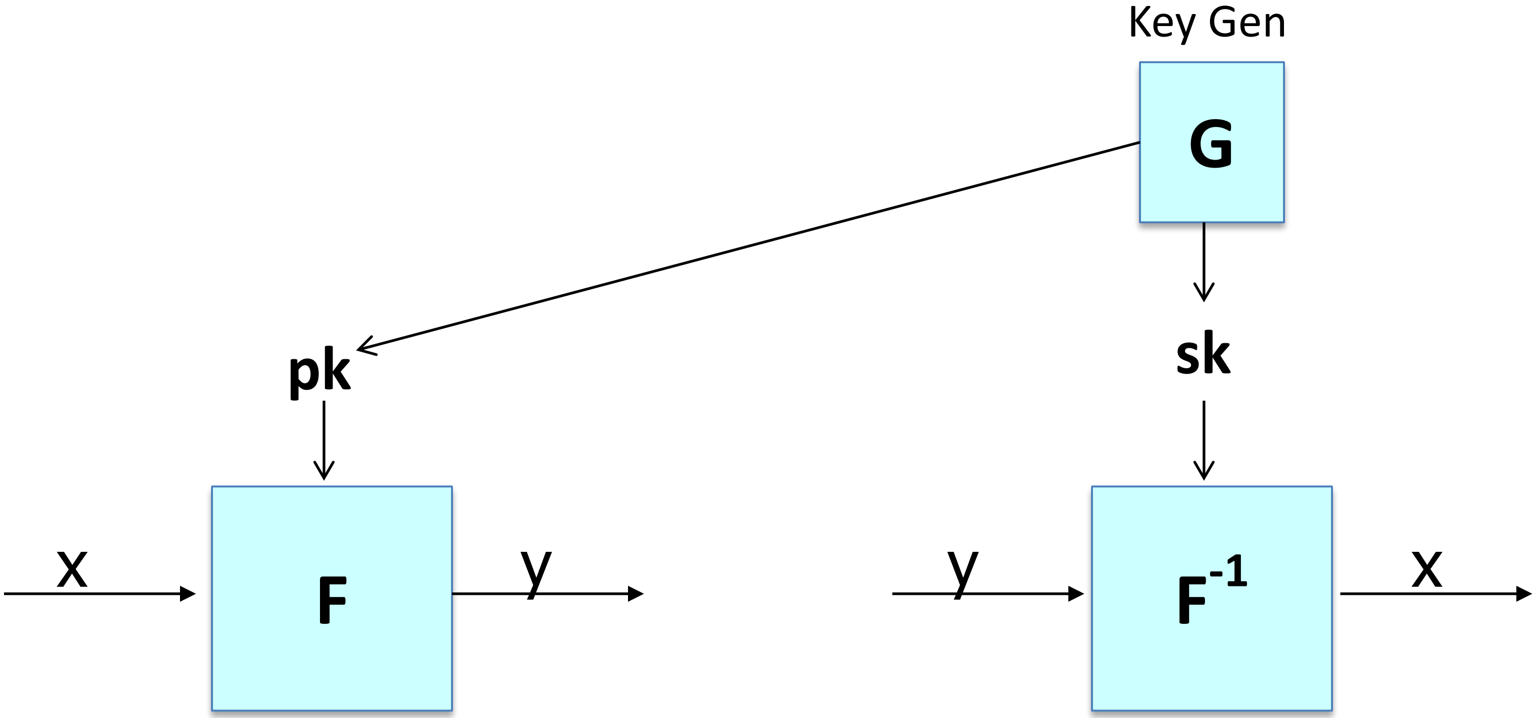
Example: RSA

Signatures from TDP: very simple and practical (next segment)

- Commonly used for signing certificates

Signatures From Trapdoor Permutations

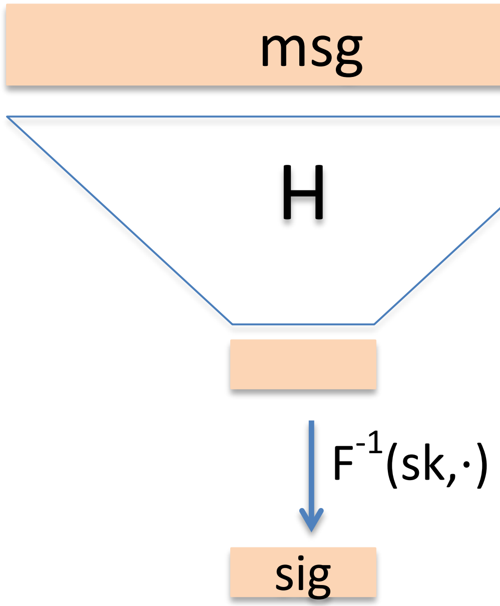
Review: Trapdoor permutation (G, F, F^{-1})



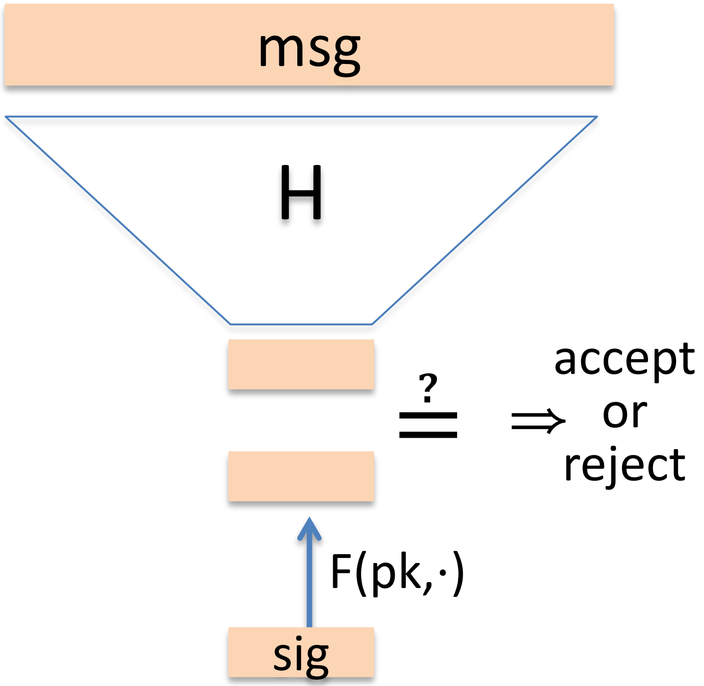
f(x) = F(pk, x) is one-to-one (X ⟶ X) and is a **one-way function.**

Full Domain Hash Signatures: pictures

S(sk, msg):



V(pk, msg, sig):



Full Domain Hash (FDH) Signatures

(G_{TDP}, F, F^{-1}): Trapdoor permutation on domain X

$H: M \longrightarrow X$ hash function (FDH)

(Gen, S, V) signature scheme:

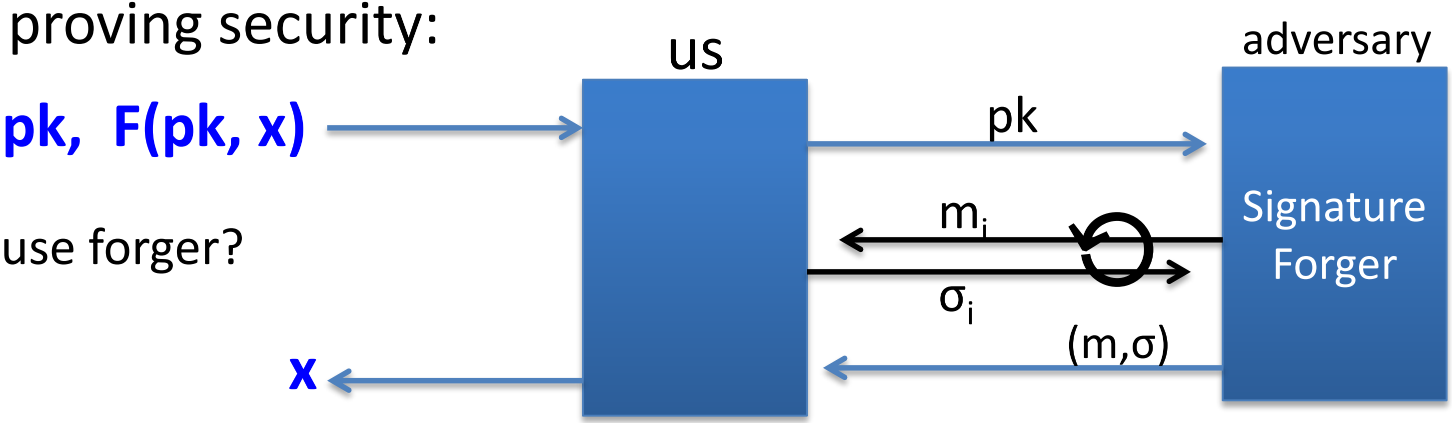
- **Gen**: run G_{TDP} and output pk, sk
- **$S(sk, m \in M)$**: output $\sigma \longleftarrow F^{-1}(sk, H(m))$
- **$V(pk, m, \sigma)$**: output $\begin{cases} \text{‘accept’ if } F(pk, \sigma) = H(m) \\ \text{‘reject’ otherwise} \end{cases}$

Security

Thm [BR]: **(G_{TDP}, F, F^{-1})** secure TDP ⇒ **(Gen, S, V)** secure signature

when **H:** $M \longrightarrow X$ is modeled as an "ideal" hash function

Difficulty in proving security:



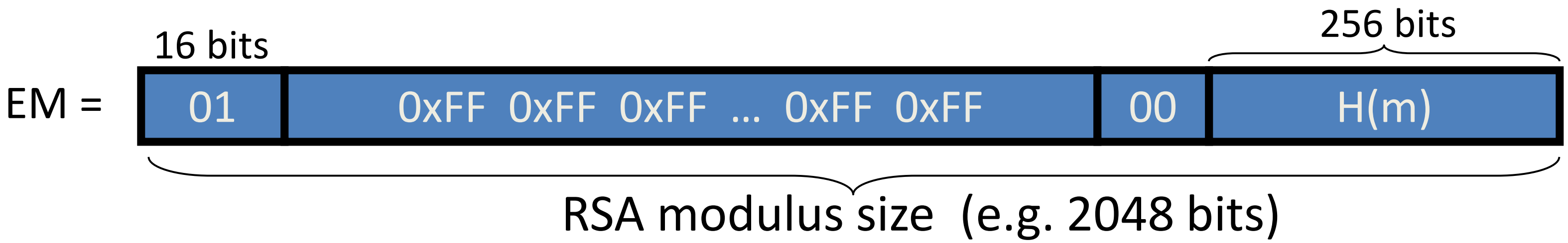
How can use use forger?

Solution: "we" will know sig. on **all-but-one** of m where adv. queries H().
Hope adversary gives forgery for that single message.

PKCS1 v1.5 signatures

RSA trapdoor permutation: pk = (N,e) , sk = (N,d)

- **S(sk, m∈M):**



16 bits

256 bits

EM = | 01 | 0xFF 0xFF 0xFF … 0xFF 0xFF | 00 | H(m) |

RSA modulus size (e.g. 2048 bits)

output: $\sigma \longleftarrow (EM)^d \bmod N$

- **V(pk, m∈M, σ):** verify that $\sigma^e \bmod N$ has the correct format

Security: no security analysis, not even with ideal hash functions

Many more topics to cover …

- Elliptic Curve Crypto
- Quantum computing
- New key management paradigms:
  - identity based encryption and functional encryption
- Anonymous digital cash
- Private voting and auction systems
- Computing on ciphertexts: fully homomorphic encryption
- Lattice-based crypto
- Two party and multi-party computation