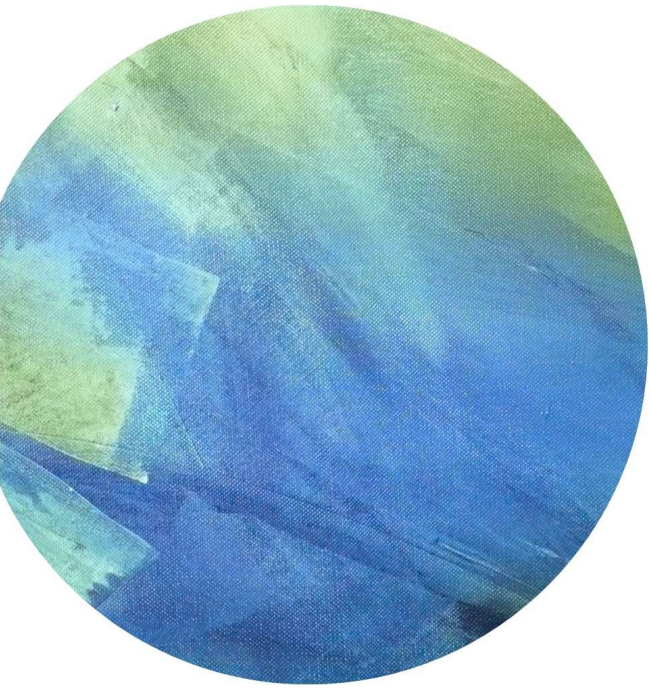
Lecture 12



Outline

Digital Signatures

More on random oracles

Digital Signatures

Digital Signatures

Public-key version of "MACs"

$$\mathrm{Gen}(1^k; r) \longrightarrow (\underset{\text{public}}{vk}, \underline{sign\,k})$$

$$\mathrm{Sign}(m, sk; r) \longrightarrow \sigma$$

$$\mathrm{Verify}(m, \sigma, vk) \longrightarrow \underset{\text{reject}}{0} \,/\, \underset{\text{accept.}}{1}$$

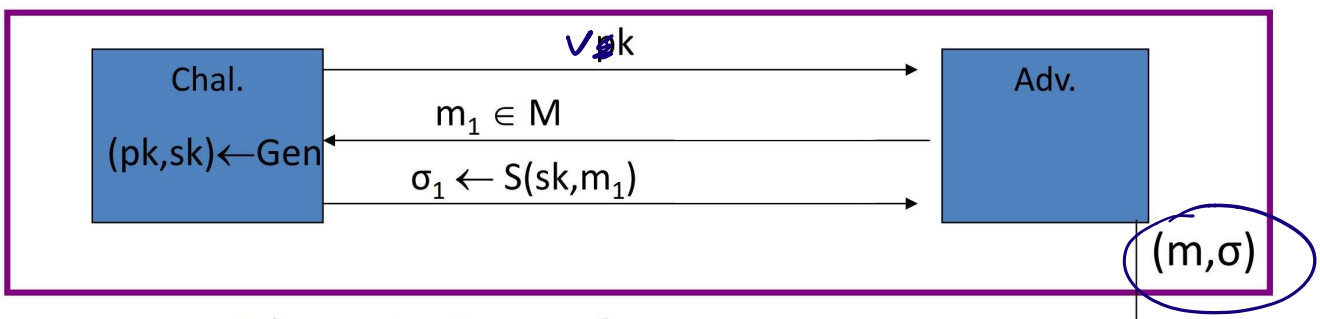
Correctness : $\forall m,\ \mathrm{Verify}(m, \mathrm{Sign}(m, sk; r), vk) = 1$

$$\left(\begin{array}{l} pk := vk \\ sk := signk \end{array} \right)$$

One-Time Signatures

- Definition of security

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



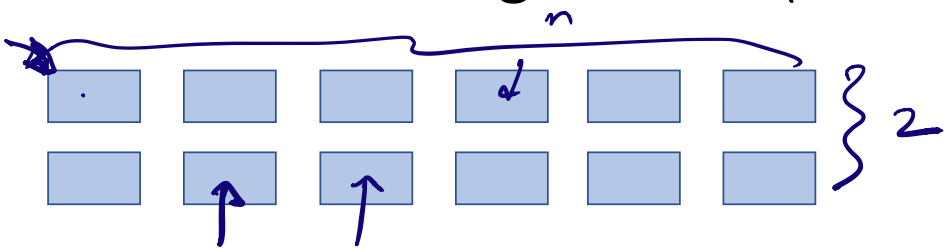
Adv. wins if **V(vk,m,σ) = `accept'** and $m \neq m_1$

Stronger: and $(m,\sigma) \neq (m_1,\sigma_1)$

(Strong one-time signature)

Lamport One-Time Signatures (from OWF)

- vk =



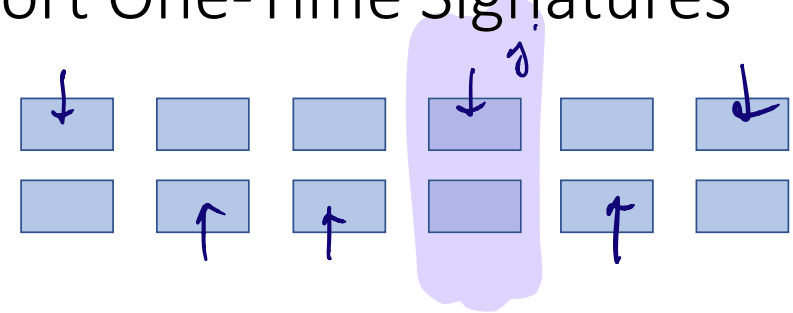
- sk =

$Gen(1^k; r) = \left(\begin{array}{cccc} \square & \overset{vk}{\square} & \cdots & \square \\ \square & \square & \cdots & \square \end{array} \right)$, $sk = \left(\begin{array}{ccc} \overset{\text{opening}}{\square} & \cdots & \square \\ \square & \cdots & \square \end{array} \right)$

To sign $m = \underline{m_1 m_2} \ldots m_n$, the signer sends openings of a few boxes.

Lamport One-Time Signatures

- vk =



- One-time security (definition):

Adv is given (m_1, σ_1) for m_1 of his choice

vk

must not be able to output

(m, σ) for $m \neq m_1$.

$m[j] \neq m_1[j]$ ∴ Adv doesn't know how to open j^{th} box for m

$f : X \to Y$ easy.

$f^{-1} : Y \to X$ hard

Eg: Adv obtained r_{30} must output r_{31}

Lamport One-Time Signatures

- vk = $\begin{pmatrix} y_{10} & y_{20} & y_{30} & \square & \square & \square \\ y_{11} & y_{21} & y_{31} & \square & \square & \square \end{pmatrix}$

y_{i0} for $i \in [n]$ $= f(r_{i0})$

- One-time security (proof):

$vk = \{y_{i0}, y_{i1}\}_{i \in [n]}$

Adv$(vk) \to m_1$. Gets σ_1 on m_1

$\sigma_1 = \{r_{i(m_1)_i}\}_{i \in [n]}$

Must output $\sigma = \{r_{i m_i}\}_{i \in [n]}$. $\exists i \in [n]$ s.t. $m_i \neq (m_1)_i$

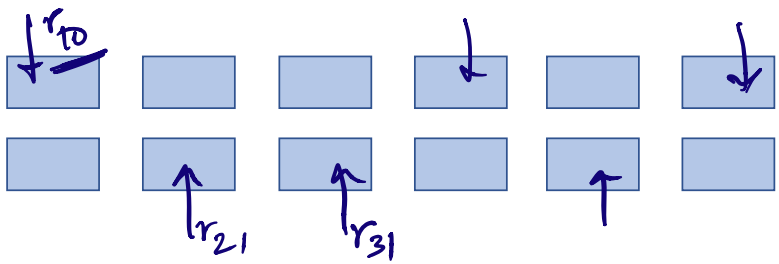
$\Rightarrow$ Adv cannot find $r_{i m_i}$ unless it inverts f on $y_{i m_i}$

Adv given $\boxed{y_{10}}$, r_{10} s.t. $f(r_{10}) = y_{10}$

Can it find r'_{10} s.t. $f(r'_{10}) = y_{10}$

ONE-WAY PERMUTATION

Lamport One-Time Signatures



- What about two-time security?

What about "strong" signature?

Given m, $\{r_{i m_i}\}_{i \in [n]}$

If Adv finds $\left(m, \{r'_{i m_i}\}_{i \in [n]}\right)$

Alternative (Stronger) Game:

Challenger $\xrightarrow{vk}$ Adv

$\xleftarrow{m_1}$

$\xrightarrow{\sigma_1}$

$\xleftarrow{m_2}$

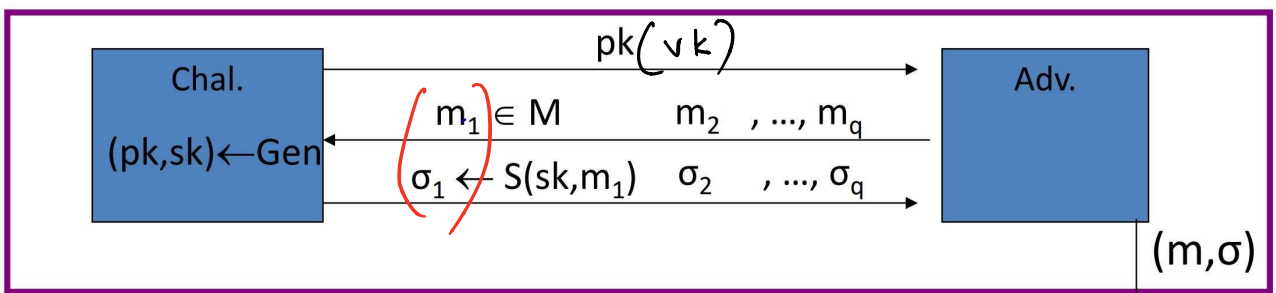
$\xrightarrow{\sigma_2}$

$\vdots$

$\xrightarrow{(m,\sigma)}$

Many-Time Signatures

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



Adv. wins if **V(pk,m,σ) = `accept'** and **m ∉ {m₁, … , m_q}**

"Strong": Adv wins if $V(pk, m, \sigma) = acc$, & $(m,\sigma) \notin \{(m_1,\sigma_1) \ldots (m_q,\sigma_q)\}$

Applications?

- Authenticating software updates

Server

sk

Initial software, **vk** →

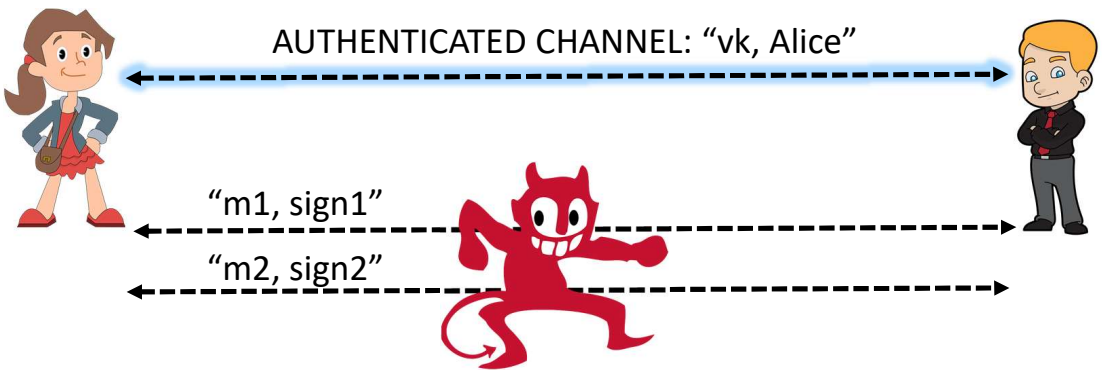
$update_1$, $sign_1$ →

$update_2$, $sign_2$ →

Client

Accepts update if Verify(update$_i$, sign$_i$, vk) = 1

One-time => Many-time authenticated channel



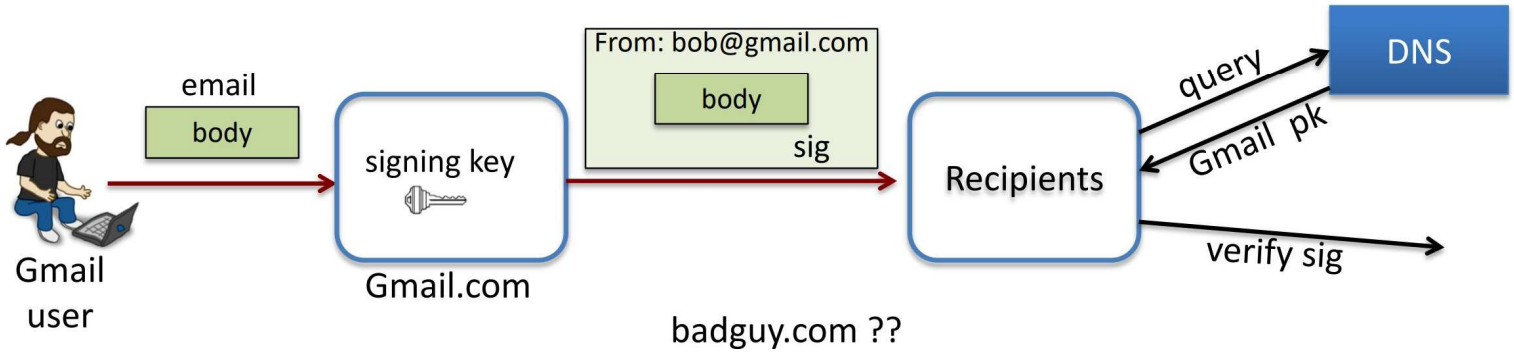
Signing email: DKIM (domain key identified mail)

Problem: bad email claiming to be from **someuser@gmail.com**

but in reality, mail is coming from domain **baguy.com**

⇒ Incorrectly makes gmail.com look like a bad source of email

Solution: **gmail.com** (and other sites) sign every outgoing mail

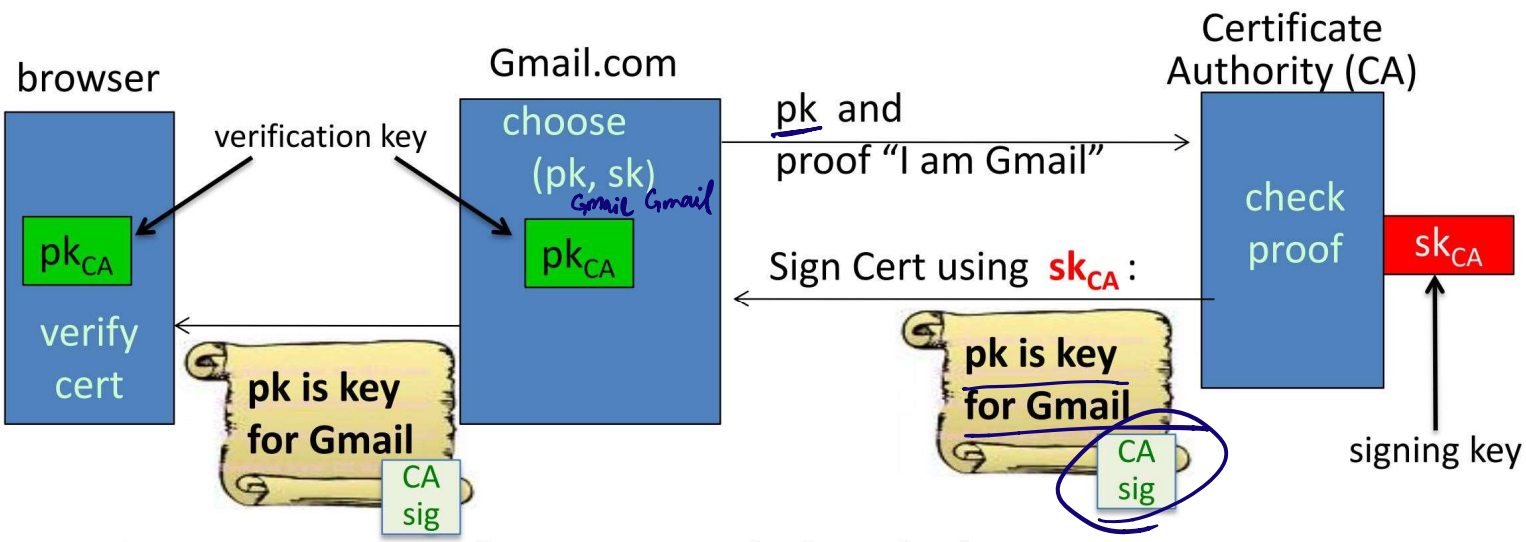


$\boxed{vk_{CA}}$ "public knowledge"

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)

MACs versus signatures

- Use MAC when

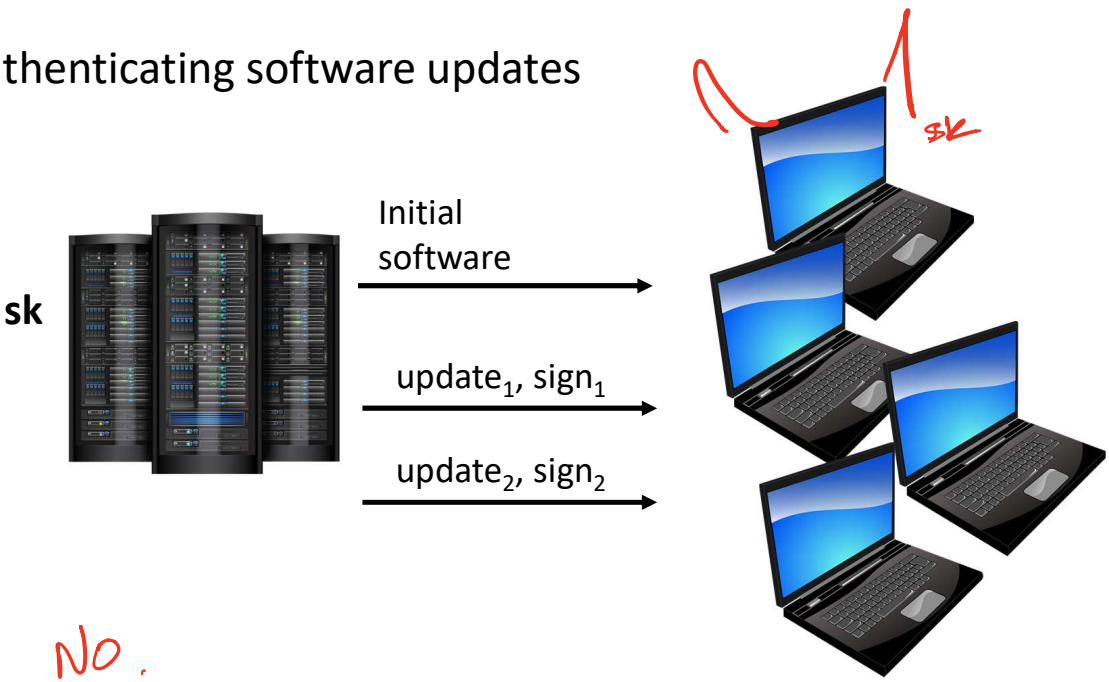
  1. signer & verifier are same
  2. single signer & single verifier

- Use signature when

  * many verifiers for the same signer
  * "public" verification

Can you use MACs here?

- Authenticating software updates



No.

Signatures from Trapdoor Permutations

<u>Trapdoor Permutations</u>

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

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

Secure trapdoor permutation:

The function F (pk, ·) is one-way without the trapdoor sk

Very simply:

Sign: $f^{-1}(sk, m) \to \sigma$

verify: $f(pk, \sigma) \stackrel{?}{=} m$

Signatures from Trapdoor Permutations

- First stab:
  - Gen outputs (vk = pk, sign key = sk)

    public secret
  - Sign (m, sign key) = F^{-1} (sk,m) = σ
  - Verify (σ, vk, m) : Check if $\boxed{F(pk, \sigma)} = \boxed{= m}$ $= F(pk, F^{-1}(sk, m)) = m.$

    pk
- Is this secure?

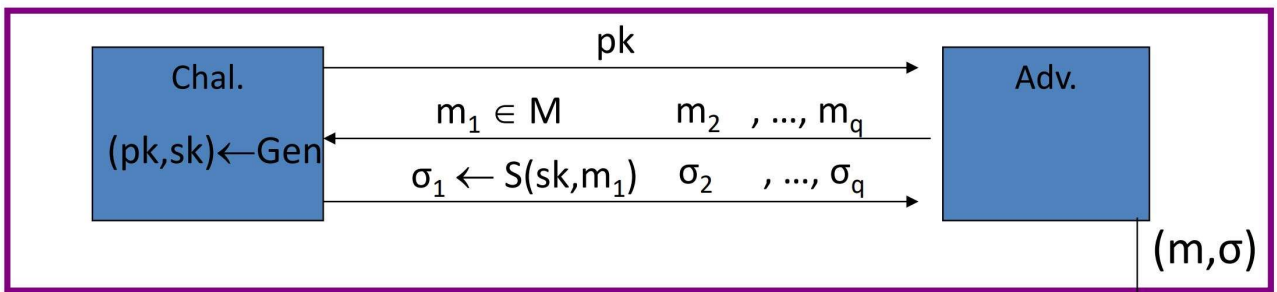
zero-time setting: Adv : (pk)

$f(pk, x)$ on x of its choice.

$[f(pk, x) = y]$ Set $(m = y, \sigma = x)$

Recall:

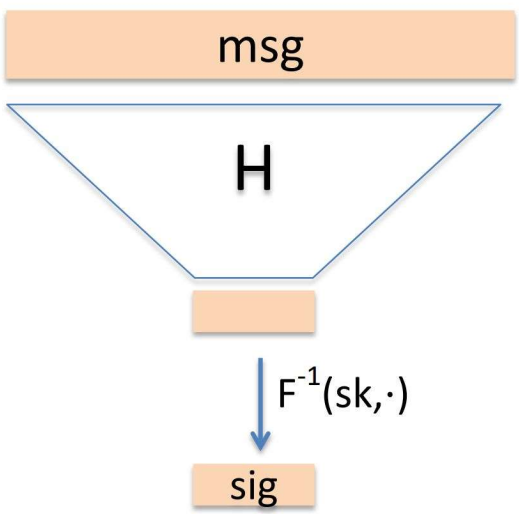
For a sig. scheme (Gen,S,V) and adv. A define a game as:



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

So how should we modify it?

S(sk, msg):



$= F^{-1}(sk, \underline{\underline{H(m)}})$

So how should we modify it?

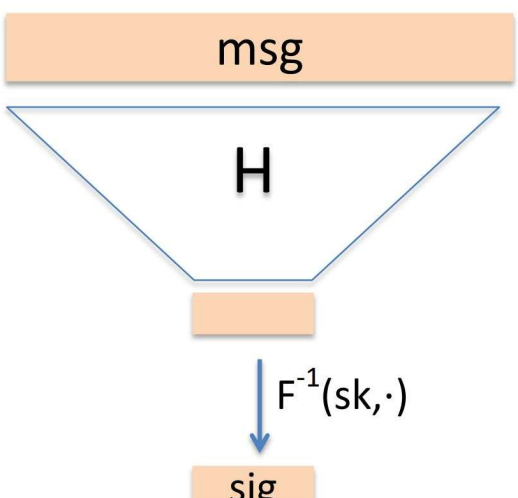
Attack strategy

$F(pk, x) = y$ → $H(m)$ → make it hard to find m.

σ

(m, σ)

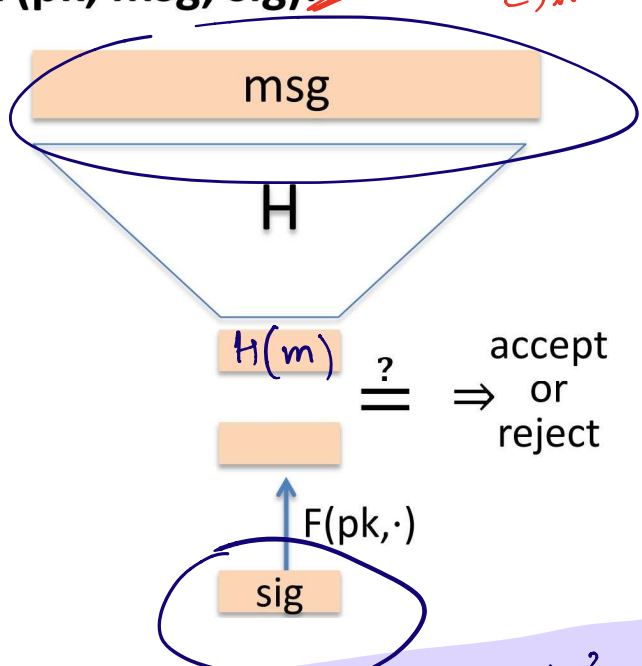
S(sk, msg):



$Sign(sk, m) = F^{-1}(sk, H(m))$

$\sigma = F^{-1}(sk, H(m))$

V(pk, msg, sig):



$Verify(m, \sigma, pk) = 1$ if $H(m) \stackrel{?}{=} F(pk, \sigma)$

0 otherwise

Full-Domain Hash

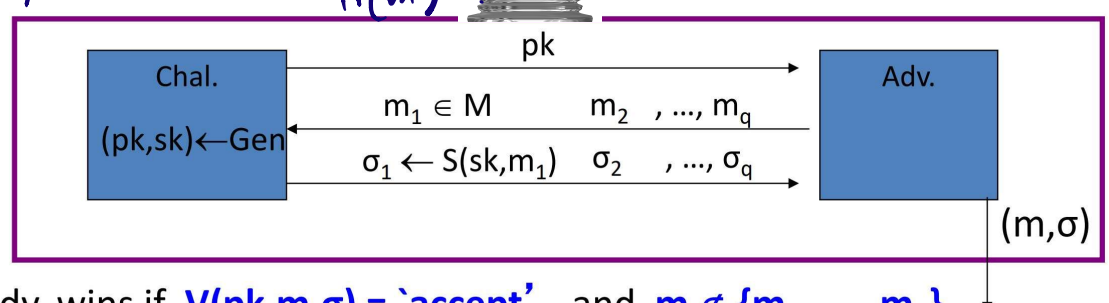
- Gen outputs (vk = pk, sign key = sk)
- Sign (m, sign key) = F^{-1} (sk,**H(m)**) = σ
- Verify (σ, vk, m) : Check if F (pk, σ) = H(m)
- Is this secure? Yes, assuming H is a *random oracle*

Many-time Signature Security of

Full-Domain Hash: Proof of Security

$\text{Sign}(sk, m) = F^{-1}(sk, H(m))$

$\text{Verify}(\sigma, m, pk): F(pk, \sigma) \stackrel{?}{=} H(m)$



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

Chall. pk Adv

$\sigma_1 = F^{-1}(sk, H(m_1))$

$\xleftarrow{m_1}$

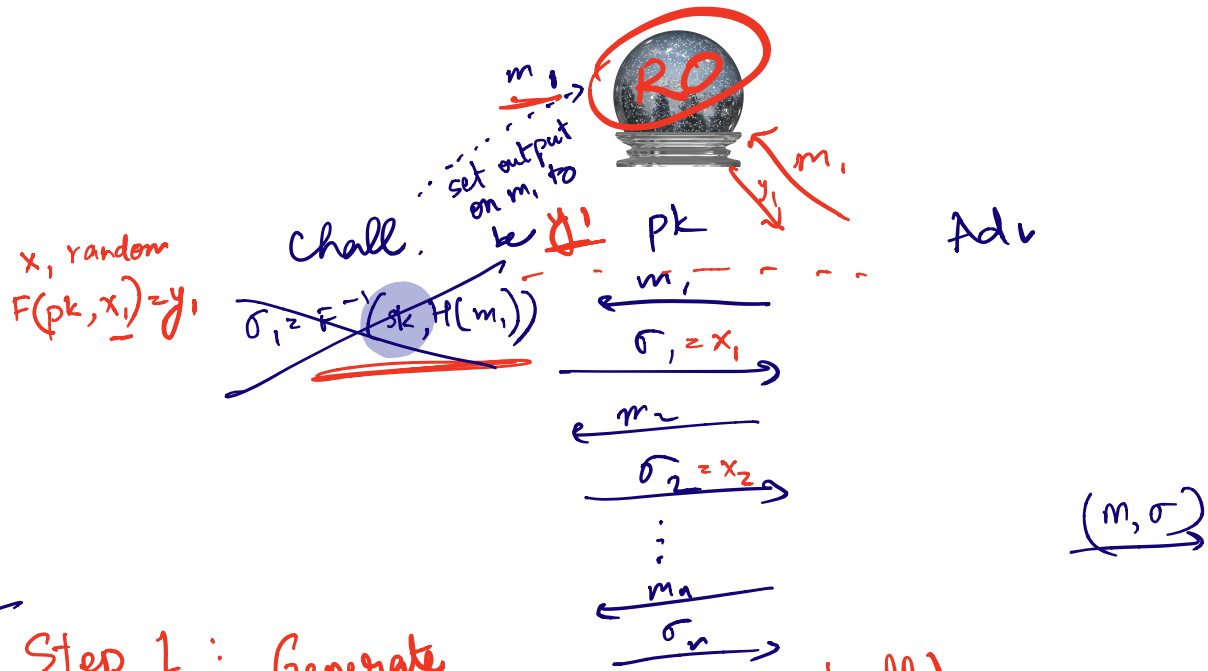
$\xrightarrow{\sigma_1}$

$\xleftarrow{m_2}$

$\xrightarrow{\sigma_2}$

$\vdots$

$\xrightarrow{(m, \sigma)}$

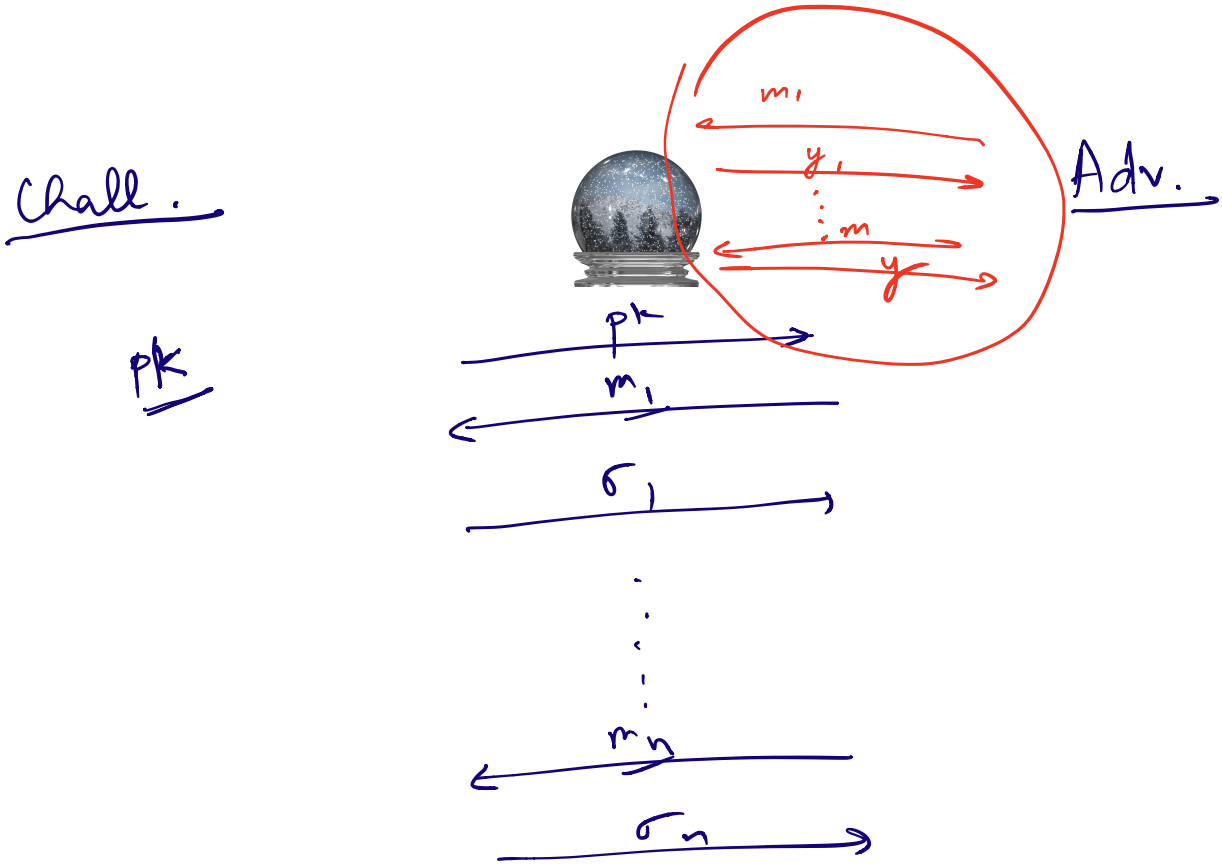


Step 1 : Generate σ_1 without access to sk at all!

$\sigma_1 = x_1$.

Step 2 : Generate (σ_1, σ_2) without access to sk ...

Step n : Generate $(\sigma_1 \ldots \sigma_n)$ without access to sk ...



Case 1: Adv never queried R.O. on m!

$\Rightarrow$ Adv gave us σ s.t. $F(pk,\sigma) = H(m)$.

$\Rightarrow$ contradicts "programmability"

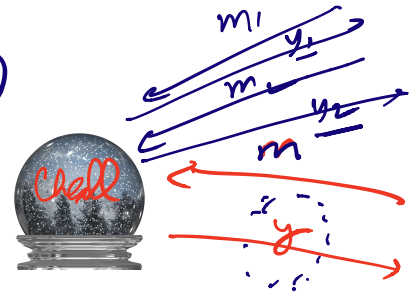
(m, σ) s.t.

$H(m) = F(pk, \sigma)$.

$\Rightarrow$ Build wrapper B around Adv

that outputs $(m, f(pk, \sigma)) = (m, H(m))$ without querying RO on m!

$\underline{x}$ $\quad y = F(pk, x)$



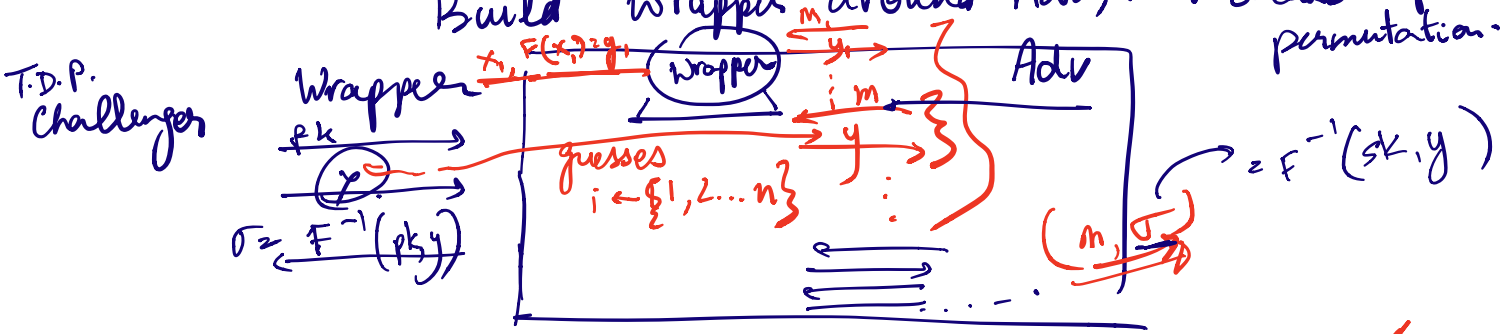
Case 2: Adv queried R.O. on m.

Then outputs (m, σ) s.t. $\boxed{\underbrace{H(m)}_{=y} = F(pk, \sigma)}$

Attacker outputs $\boxed{\sigma = F^{-1}(sk, y).}$

Take Adv that breaks MT Sign. Sec. game.

Build wrapper around Adv, that breaks trapdoor permutation.



Given Adv that breaks MT Sign. Sec, w.p. $\varepsilon = \frac{1}{poly(k)}$

$\Rightarrow \exists$ Wrapper that breaks TDP w.p. $\left(\frac{\varepsilon}{n}\right)$

$n \to poly(k)$

$= \frac{1}{poly'(k) \cdot poly(k)}$

$= \frac{1}{poly(k)}$

Signature lengths

Goal: best existential forgery attack time $\geq 2^{128}$

algorithm	signature size
RSA	2048-3072 bits
EC-DSA	512 bits
Schnorr	384 bits
BLS	256 bits

Open problem: practical 128-bit signatures

Time-permitting, cover:

- Domain Extension
- Aggregatable Signatures