Lecture 5

Digital Signatures

Fall 2023

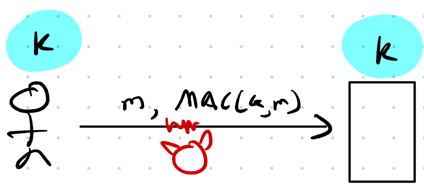
Corrigan-Gibbs & Zeldovich

MIT

Plan

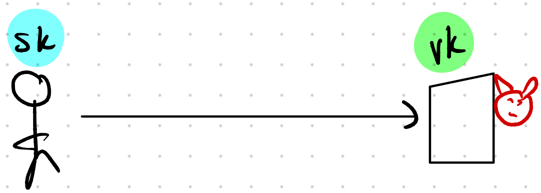
- Definition
- One-time signatures from OWF
- Many-time

Last time: MAC



Being able to authenticate gives you ability to produce new MAC'd msgs

This time: Signatures



Can verify/authenticate msgs without being able to sign new ones!

First, recap:

$$\mathrm{MAC} : \mathcal{K} \times \mathcal{M} \to \mathcal{T}$$

MAC Security: EUF-CMA

Defined by security game we saw last time

MAC for short msgs:

$$\text{PRF } F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$$ AES is example of PRF

$$\mathrm{MAC}(k, m) := F(k, m)$$ (msg space $\mathcal{X}$)

* PRF Security relies on key being completely random. *

~~$F(\text{"0000"}, \text{"0000"})$~~

MAC for long msgs:

$$\text{UHF } \mathrm{Hash} : \mathcal{K} \times \mathcal{X}^{\ell} \to \mathcal{X}$$ { "secret key version of CRHF" }

(msg space $\mathcal{X}^{\ell}$)

$$\mathrm{MAC}((k_1, k_2), m) := F(k_1, \mathrm{Hash}(k_2, m))$$

UHF is a fast "non-cryptographic" hash fn.

Intuition: Until attacker finds two msgs that hash to same value, has no info that can help it forge.
[See Book Thm 7.7]

Digital signatures

- Used everywhere on the web for authentication
  ↳ certificates, HTTPS, SSH, ...
- Unlike a pen-and-paper signature, can't cut & paste
  ↳ Sig is bound to data signed.
- Unlike MAC, there are two keys

  private signing key sk
  public verification key vk/pk

The idea of using two keys (public & private) was <u>the</u> revolutionary idea in cryptography in the 20th century (Diffie & Hellman)

↳ Followed thousands of years of shared secrets

Syntax

$\text{Gen}() \to (sk, pk)$ — { In theory, Gen takes the sec param as input

$\text{Sign}(sk, m) \to \sigma$ — $m \in \mathcal{M}$ = msg space

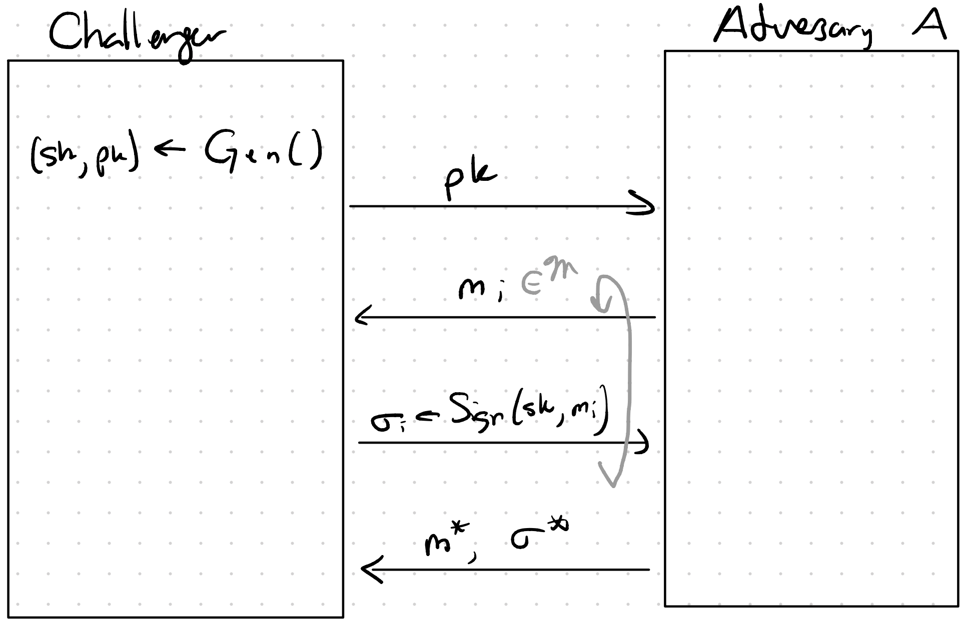
$\text{Verify}(pk, m, \sigma) \to \{0,1\}$

Correctness: Honest verifier accepts honest sigs:

$$\forall\ (sk, pk) \leftarrow \text{Gen}()$$
$$\forall\ m \in \mathcal{M}$$
$$\forall\ \sigma \leftarrow \text{Sign}(sk, m)$$
$$\text{Verify}(pk, m, \sigma) = 1.$$

Security: Almost identical to MAC security

EUF-CMA: Attacker sees sigs on many msgs of its choosing. Can't forge a sig on new msg of its choice.



$$\begin{cases} 1 & \text{if } \mathrm{Ver}(pk, m^*, \sigma^*) = 1 \\ 0 & \text{o.w.} \end{cases}$$

For sig scheme $\Sigma = (\mathrm{Gen}, \mathrm{Sign}, \mathrm{Ver})$, adv A, let

$$\mathrm{SigAdv}[A, \Sigma] = \Pr[\text{game output is } 1].$$

Sig scheme $\Sigma = (\mathrm{Gen}, \mathrm{Sign}, \mathrm{Ver})$ is secure if $\forall$ eff adv A

$$\mathrm{SigAdv}[A, \Sigma] \leq \text{"negl"}$$

Subtle point:

This defn admits schemes in which it is easy to cook up new valid sigs on msg m given one sig on m. [Has led to attacks!]

→ ECDSA has this malleability property

→ Think: How can you tweak defn to prevent?

Efficiency

We typically care about

* Signature size
* Size of pk
* Signing time
* Verification time
* Assumptions (Factoring/RSA? PQ?)

No one signature scheme dominates all others in all metrics ☹

Recall: One-way function

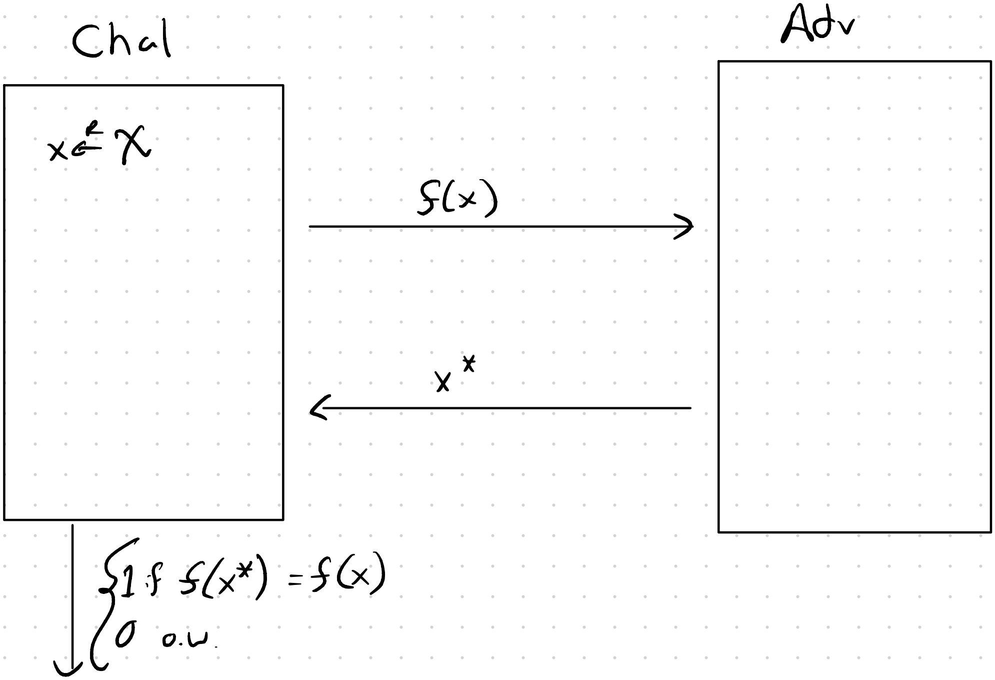
A fn $f: \mathcal{X} \to \mathcal{Y}$ is one way if

$\forall$ eff advs A

$$\Pr\left[f(x^*) = x : \begin{array}{l} x \xleftarrow{R} \mathcal{X} \\ x^* \leftarrow A(f(x)) \end{array} \right] \le \text{negl.}$$

(Reminder: IS $P = NP \Rightarrow \nexists$ OWF.)

Or, in game form...



Example candidate OWFs:

$$f_{SHA}(x) := SHA256(x)$$

$$f_{PRF}(x) := PRF(k, \text{"000000"})$$

$$f_{MAC}(x) := MAC(k, \text{"000000"})$$

⋮

OWFs are essentially the weakest/simplest crypto tool.

↳ Surprise? That possible to use them to get signatures

★ OWF only hard to invert on a randomly sampled input. ★

If $f: \{0,1\}^n \to \{0,1\}^n$ is OWF, no guarantee that

$$f(\underbrace{00000 \cdots 0}_{n/2}1\underbrace{1 \quad x \quad}_{n/2})$$ is hard to invert

$$f(01\underbrace{1 \quad x \quad}_{n-1})$$ is hard to invert

One-time-secure signatures (Lamport)

[One-time secure = secure if adv only sees one sig under a given sk]

Msg space $\mathcal{M} = \{0,1\}^n$
Sec param $\lambda = 128$.
O.W.F $f : \{0,1\}^\lambda \to \{0,1\}^\lambda$

Gen() → (sk, vk)

Choose $2n$ random λ-bit strings....

$$sk \leftarrow \begin{pmatrix} x_{01} & x_{02} & x_{03} & \cdots & x_{0n} \\ x_{11} & x_{12} & x_{13} & \cdots & x_{1n} \end{pmatrix}$$

$$vk \leftarrow \begin{pmatrix} f(x_{01}) & f(x_{02}) & f(x_{03}) & \cdots & f(x_{0n}) \\ f(x_{11}) & f(x_{12}) & f(x_{13}) & & f(x_{1n}) \end{pmatrix} = \begin{pmatrix} y_{01}, \cdots, y_{0n} \\ y_{11}, \cdots, y_{1n} \end{pmatrix}$$

Sign$(sk, m_1 \| m_2 \| \cdots \| m_n \in \{0,1\}^n) \to \sigma$

Output sk values corresponding to bits of msg

$$\sigma = (x_{m_1,1}, x_{m_2,2}, \cdots, x_{m_n,n})$$

Verify$(pk, m_1 \| \cdots \| m_n \in \{0,1\}^n, \sigma) \to \{0,1\}$

Check $\forall i \in [n]$. $y_{m_i i} = f(\sigma_i)$
Output "1" if all accept.

Why are Lamport sigs only one-time secure?

$$sk \leftarrow \begin{pmatrix} x_{01} & x_{02} & x_{03} & \cdots & x_{0n} \\ x_{11} & x_{12} & x_{13} & \cdots & x_{1n} \end{pmatrix}$$

Each sig gives away 1/2 of secret key.

With 2 sigs can recover all.

Why is it secure for one-time use?

Intuition: To sign m^*, need to invert f at at least one point.

Beware: Intuition is often wrong! Devil in details.

↳ Security proofs are a useful tool, more or less essential in modern crypto.

Properties

Correctness — By construction.

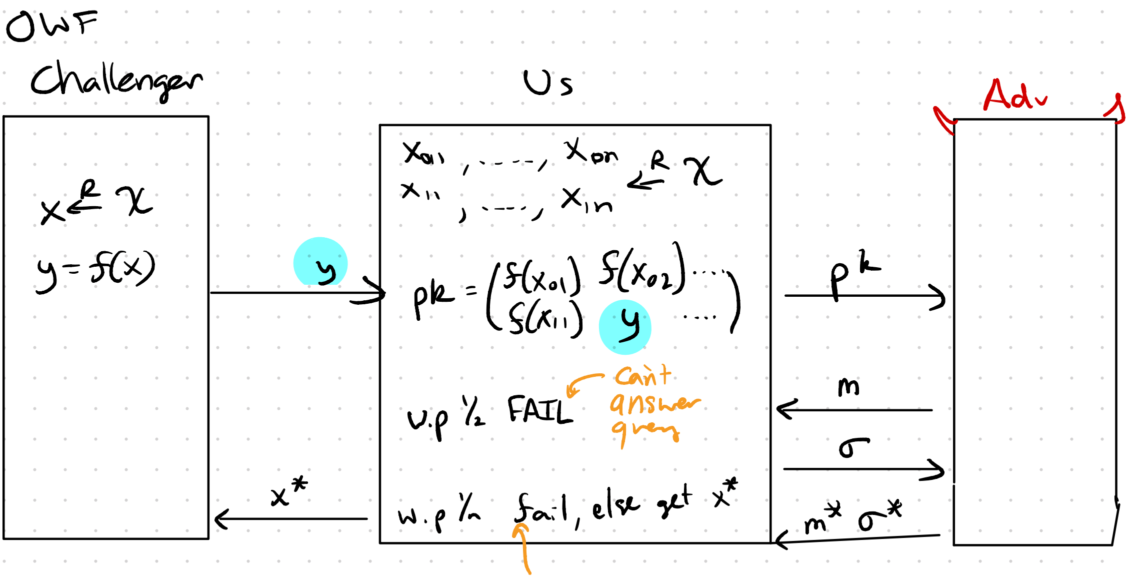
Security — ONE-TIME secure.

Strategy

$\exists$ adversary that forges w.p. ε $\Longrightarrow$ $\exists$ adv that inverts o.w.f wp $\approx \varepsilon$.

not exactly

$\Rightarrow$ Contradicts OWF security.



m and m^* both "avoid" the challenge y

Efficiency

* OWFs are very fast $\approx$ 60 m/s for AES
* Sigs are large-ish $\approx \lambda^2$ bits for $\lambda = 256$.

Extension: Signing long msgs: "Hash & sign"

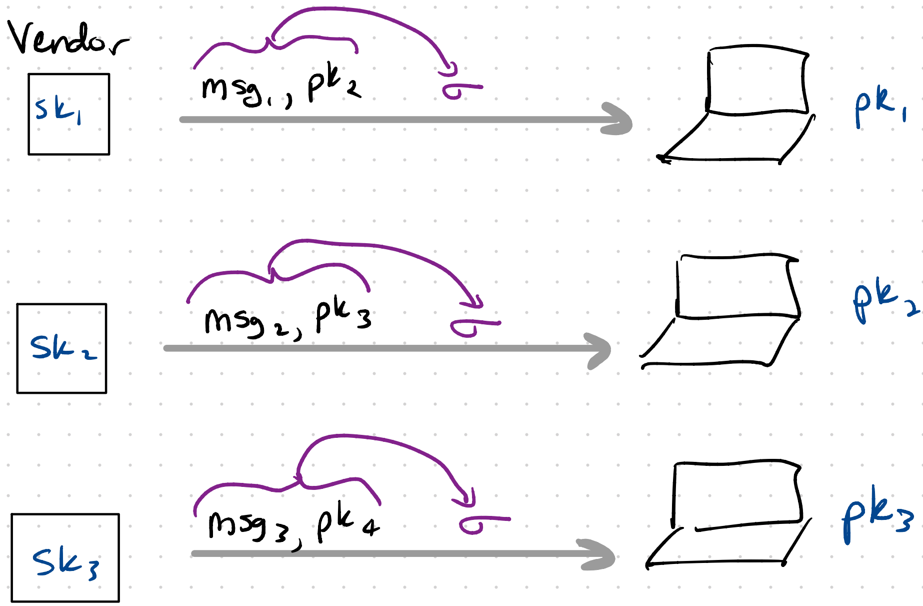
Use $H: \{0,1\}^* \to \{0,1\}^{\lambda=256}$ CRHF

$Sig'(sk, m) := Sig(sk, H(m))$ Lamport sig for λ-bit msgs.

$Ver'(pk, m, \sigma) := Ver(pk, H(m), \sigma)$

Q: Why not use a (keyed) universal hash fn?

Application: Software Updates



* Vendor signs each update with a fresh sk.

* Only need one-time security
  [Not typically used in practice.]

Many-Time Secure Sigs [See Book for citations]

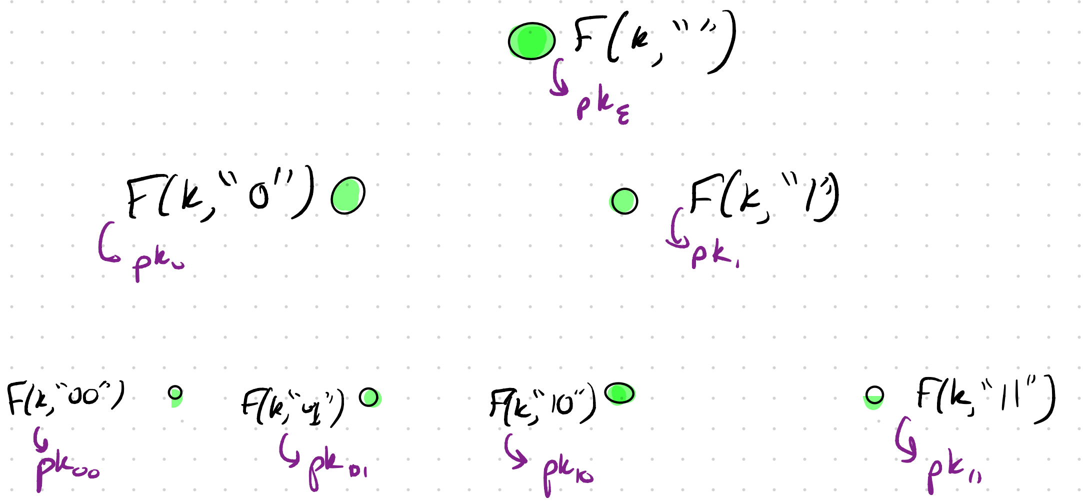
We will show how to construct a full-blown (many-time secure) sig scheme from any one-time secure scheme + PRF.

$\Rightarrow$ Sig from OWF. [Since $\exists$ OWF $\Rightarrow$ $\exists$ PRF]

Observation: Can use one PRF secret key to implicitly generate a gigantic tree of sig sks (tree has 2^n leaves)

$\boxed{\text{PRF: } F : \mathcal{K} \times \{0,1\}^{\leq n} \to sk}$

All keys "as good as random" by PRF security.



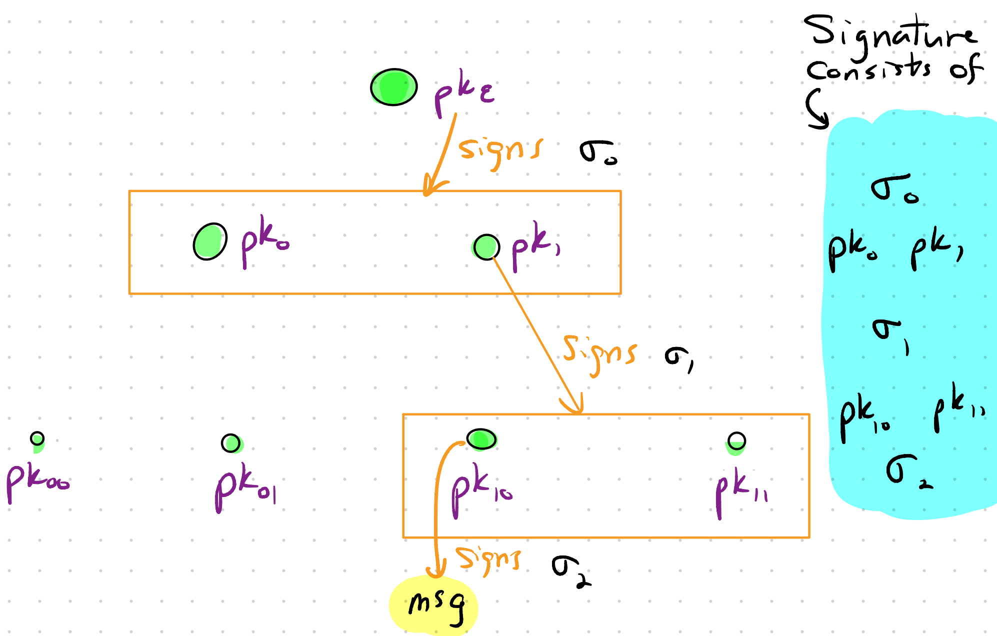
From sk, can generate pk for each node.

Many-time Sigs ... Will be informal. See Book for full description

$$\text{Gen}() \to \begin{cases} pk \leftarrow pk_\varepsilon \text{ (at root)} \\ sk \leftarrow \text{PRF key } k \end{cases}$$

$\text{Sign}(sk, m) \to \sigma$

* Pick a leaf ℓ at random.
* Sign using sk_ℓ
* Sign each pair of vks using sk of "parent node" in tree.
* Publish all sigs and vks on path to root + siblings.



Many-time Sigs (cont'd)

Ver(pk, m, σ):

Verify chain of sigs down to leaf