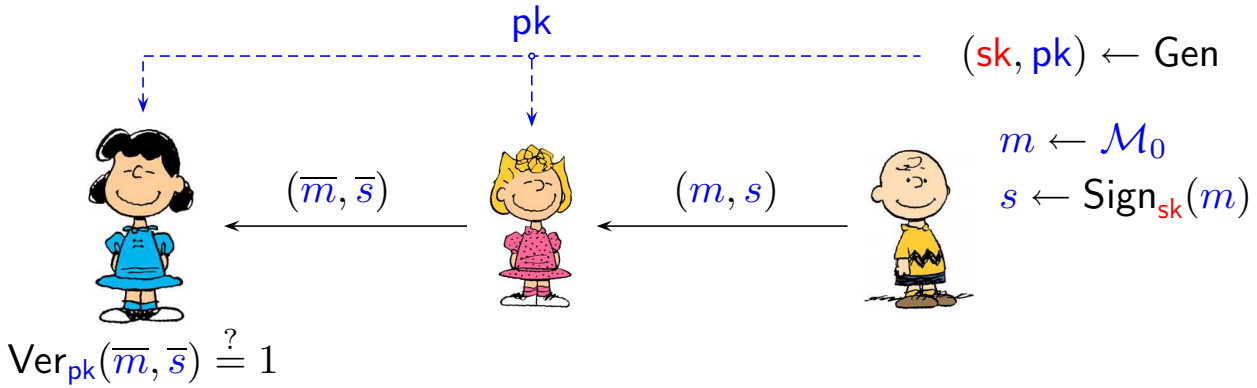
# Digital Signatures

Sven Laur
swen@math.ut.ee

University of Tartu

# Formal Syntax

# Digital signature scheme




- To establish electronic identity, Charlie must generate $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}$ and convinces others that the public information pk represents him.
- A keyed hash function $\mathsf{Sign}_{\mathsf{sk}} : \mathcal{M} \rightarrow \mathcal{S}$ takes in a plaintext and outputs a corresponding digital signature.
- A public verification algorithm $\mathsf{Ver}_{\mathsf{pk}} : \mathcal{M} \times \mathcal{S} \rightarrow \{0, 1\}$ tries to distinguish between altered and original message pairs.
- The signature scheme is functional if for all $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}$ and $m \in \mathcal{M}$:

$$\mathsf{Ver}_{\mathsf{pk}}(m, \mathsf{Sign}_{\mathsf{sk}}(m)) = 1 \ .$$

# Example. RSA-1024 signature scheme

**Key generation** Gen:

1. Choose uniformly $512$-bit prime numbers $p$ and $q$.
2. Compute $N = p \cdot q$ and $\phi(N) = (p-1)(q-1)$.
3. Choose uniformly $e \leftarrow \mathbb{Z}^*_{\phi(N)}$ and set $d = e^{-1} \mod \phi(N)$.
4. Output $\mathsf{sk} = (p, q, e, d)$ and $\mathsf{pk} = (N, e)$.

**Signing and verification:**

$$\mathcal{M} = \mathbb{Z}_N, \quad \mathcal{S} = \mathbb{Z}_N, \quad \mathcal{R} = \emptyset$$

$$\mathsf{Sign}_{\mathsf{sk}}(m) = m^d \mod N$$

$$\mathsf{Ver}_{\mathsf{pk}}(m, s) = 1 \quad \Leftrightarrow \quad m = s^e \mod N \ .$$

# Security definitions

## Attack scenarios

- ▷ Key only attack. The adversary has access only to the public key pk.
- ▷ Chosen message attack. Besides the public key pk, the adversary can adaptively query a list of valid signatures $(m_1, s_1), \ldots, (m_n, s_n)$.

## Attack types

- ▷ Universal forgery. The adversary must create a valid signature for a prescribed message $m$ that is chosen from a distribution $\mathcal{M}_0$.
- ▷ Existential forgery. The adversary must create a valid signature for some message $m$, i.e., there are no limitations on the choice of message.

| One more signature attack = Chosen message attack + Existential forgery |
|---|

# Security against one more signature attack

A signature scheme is $(t, q, \varepsilon)$-secure against one more signature attack if $\mathsf{Adv}^{\mathsf{forge}}(\mathcal{A}) = \Pr\left[\mathcal{G}^{\mathcal{A}} = 1\right] \leq \varepsilon$ for any $t$-time adversary $\mathcal{A}$ where

$\mathcal{G}^{\mathcal{A}}$

$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}$

For $i \in \{1, \ldots, q\}$ do

$m_i \leftarrow \mathcal{A}(s_{i-1})$

$s_i \leftarrow \mathsf{Sign}_{\mathsf{sk}}(m_i)$

$(m, s) \leftarrow \mathcal{A}$

if $(m, s) \in \{(m_1, s_1), \ldots, (m_q, s_q)\}$ return $0$

return $\mathsf{Ver}_{\mathsf{pk}}(m, s)$

# A conceptual description of digital signatures

A digital signature is a non-interactive and transferable counterpart of the following extended identification protocol:

1. Verifier sends a message $m$ to a prover.
2. The prover accepts the message $m$.
3. The prover authenticates him or herself by using sk.

Transferability means that signature must be verifiable by other parties that did not participate in the creation of the signature.

As a result, digital signature is either

- ▷ a non-interactive proof of possession that is linked to the message $m$.
- ▷ a non-interactive proof of knowledge that is linked to the message $m$.

# Digital Signatures Based on Proofs of Possession

# Trapdoor one-way permutations

A collection of trapdoor permutations $\mathcal{F}_{\mathrm{tp}}$ is determined by three algorithms $(\mathsf{Gen}, \mathrm{Map}, \mathrm{Inv})$ such that

$$\forall(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{Gen}, \quad \forall m \in \mathcal{M}_{\mathsf{pk}} : \qquad \mathrm{Inv}_{\mathsf{sk}}(\mathrm{Map}_{\mathsf{pk}}(m)) = m$$

and both algorithms $\mathrm{Map}_{\mathsf{pk}}(\cdot)$ and $\mathrm{Inv}_{\mathsf{sk}}(\cdot)$ are deterministic.

# OW-CPA security

A collection of trapdoor permutations $\mathcal{F}_{\mathrm{tp}}$ is $(t, \varepsilon)$-secure if for any $t$-time adversary $\mathcal{A}$

$$\mathsf{Adv}^{\mathsf{inv\text{-}cpa}}(\mathcal{A}) = \Pr\left[\mathcal{G}^{\mathcal{A}} = 1\right] \leq \varepsilon$$

where

$$\mathcal{G}^{\mathcal{A}}$$

$$\left[\begin{array}{l} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen} \\ m \xleftarrow{u} \mathcal{M}_{\mathsf{pk}} \\ y \leftarrow \mathrm{Map}_{\mathsf{pk}}(m) \\ \mathsf{return}\ [\mathcal{A}(y) \stackrel{?}{=} m] \end{array}\right.$$

# Full Domain Hash

## Setup

Run $(\mathsf{pk}, \mathsf{sk}) \leftarrow$ Gen to create an instance of trapdoor permutation. Choose $h : \mathcal{M} \rightarrow \mathcal{M}_{\mathsf{pk}}$ from a collision resistant function family $\mathcal{H}$.

## Signing

To sign $m \in \mathcal{M}$, compute $x \leftarrow h(m)$ and output $s \leftarrow \mathrm{Inv}_{\mathsf{sk}}(x)$.

## Verification

A pair $(m, s)$ is a valid signature if $h(m) = \mathrm{Map}_{\mathsf{pk}}(s)$.

# Random oracle model

Let us model the hash function by an oracle $\mathcal{O}_{\mathsf{pk}}(\cdot)$ that evaluates $h \xleftarrow{u} \mathcal{H}_{\mathrm{all}}$ where $\mathcal{H}_{\mathrm{all}} = \{h : \mathcal{M} \to \mathcal{M}_{\mathsf{pk}}\}$ is a set of all functions.

Thus, the advantage is now computed as average

$$\mathsf{Adv}^{\mathsf{forge}}_{\mathcal{H}_{\mathrm{all}}}(\mathcal{A}) = \frac{1}{|\mathcal{H}_{\mathrm{all}}|} \cdot \sum_{h \in \mathcal{H}_{\mathrm{all}}} \mathsf{Adv}^{\mathsf{forge}}_{h}(\mathcal{A}) \ .$$

In reality, we substitute $\mathcal{H}_{\mathrm{all}}$ with a suitable hash family $\mathcal{H}$ and hope

$$\mathsf{Adv}^{\mathsf{forge}}_{\mathcal{H}}(\mathcal{A}) = \frac{1}{|\mathcal{H}|} \cdot \sum_{h \in \mathcal{H}_{\mathrm{all}}} \mathsf{Adv}^{\mathsf{forge}}_{h}(\mathcal{A}) \approx \mathsf{Adv}^{\mathsf{forge}}_{\mathcal{H}_{\mathrm{all}}}(\mathcal{A})$$

for all relevant adversaries $\mathcal{A} \in \mathfrak{A}$ humans devise.

# Security in the random oracle model

**Theorem.** Let $\mathcal{F}_{\mathrm{tp}}$ is $(t, \varepsilon)$-secure collection of trapdoor permutations such that $\mathrm{Map}_{\mathsf{pk}}(\cdot)$ is always $\tau$-time computable. Then the FDH signature scheme is $(t - (q_h + q_s + 1) \cdot \tau, q_s, (q_h + q_s + 1) \cdot \varepsilon)$-secure against one more signature attack provided that the adversary can do up to $q_h$ hash queries.

## Drawbacks

- ▷ The result holds only in random oracle model.
- ▷ The security bound increases linearly with $q_h$ and $d_s$.
- ▷ PSS signature scheme achieves a "sublinear" bound.
- ▷ The proof cannot be generalised to the standard model [Pallier2007].

# The corresponding proof

- ▷ We can always assume that $\mathcal{A}$ queries hash for the forgery.
- ▷ The following evaluation strategies lead to identical results

$\mathcal{O}_{\mathsf{pk}}(m)$

if $H[m] = \perp$ then
  $H[m] \xleftarrow{u} \mathcal{M}_{\mathsf{pk}}$
return $H[m]$

$\mathcal{O}_{\mathsf{pk}}(m)$

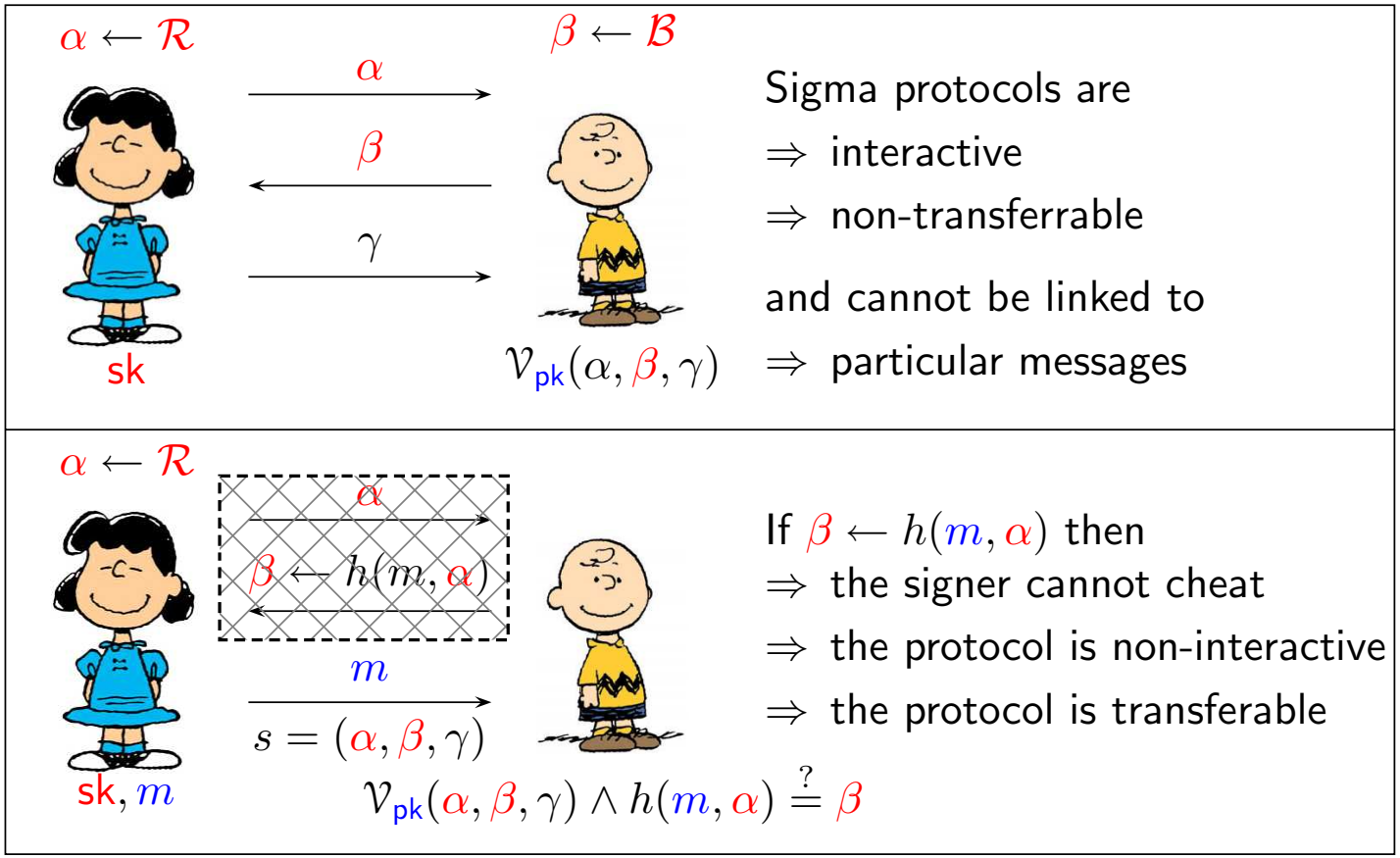
if $H[m] = \perp$ then
  $S[m] \xleftarrow{u} \mathcal{M}_{\mathsf{pk}}$
  $H[m] \leftarrow \mathrm{Map}_{\mathsf{pk}}(S[m])$
return $H[m]$

- ▷ Let $\varepsilon_i$ be the probability that forgery was successful and the $i$th hash query corresponded to the message.
- ▷ Then $\varepsilon_i \leq \varepsilon$, since otherwise we would have a too efficient inverter.

# Digital Signatures Based on Proofs of Knowledge

# Fiat-Shamir heuristics

Fiat-Shamir heuristics converts any sigma-protocol to a signature scheme.



# Generic signature schemes

Let $\pi$ be a sigma protocol specified by the prover and verifier algorithms $\mathcal{P}_{\mathsf{sk}}$ and $\mathcal{V}_{\mathsf{pk}}$. Then the corresponding generic signature scheme is following.

**Setup**

Use output $(\mathsf{pk}, \mathsf{sk}) \leftarrow$ Gen of the sigma protocol setup.

**Signing**

To sign a message $m$, output $(\alpha, \beta, \gamma)$ as a signature where

$$\alpha \leftarrow \mathcal{P}_{\mathsf{sk}}, \quad \beta \leftarrow h(m, \alpha), \quad \gamma \leftarrow \mathcal{P}_{\mathsf{sk}}(\beta) \ .$$

**Verification**

A tuple $(m, \alpha, \beta, \gamma)$ is a valid signature if

$$\beta = h(m, \alpha) \wedge \mathsf{Ver}_{\mathsf{pk}}(\alpha, \beta, \gamma) = 1 \ .$$

# Famous examples

Schnorr authentication protocol

- ◇ Schnorr signature scheme
- ◇ DSA algorithm
- ◇ ElGamal signature scheme
- ◇ Nyberg-Rueppel signature scheme

Fiat-Shamir identification protocol

- ◇ Feige-Fiat-Shamir signature scheme

Guillou-Quisquater identification protocol

- ◇ Guillou-Quisquater signature scheme

# Security in the random oracle model

**Forking Lemma** (Basic form). Let $\kappa$ be the knowledge error of the sigma protocol. If for a particular public key pk, a $t$-time adversary $\mathcal{A}$ manages to output valid signature by making at most $n$ queries to the random oracle and no queries to the signing oracle with probability $\varepsilon$, then there exist an extraction algorithm $\mathcal{K}$ for that runs in expected time

$$\mathbf{E}[\tau] = \Theta\left(\frac{(n+2)\cdot t}{\varepsilon - \kappa}\right)$$

and returns the corresponding secret key sk.

**Simulatability Lemma.** Oracle calls to the signing oracle can be replaced with the runs of the sigma protocol simulator. The probability of simulation failures due to the contradicting assignments for $\mathcal{O}(\cdot)$ are negligible.

# Yet Another Matrix Game

# An encoding of the extraction task

Assume that $\mathcal{A}$ never queries the same value $h(m_i, \alpha_i)$ twice and that $\mathcal{A}$ itself verifies the validity of the candidate signature $(m_{n+1}, s_{n+1})$.

Let $\omega_0$ denote the randomness used by $\mathcal{A}$ and let $\omega_1, \ldots \omega_{n+1}$ be the replies for the hash queries $h(m_i, \alpha_i)$. Now define

$$A(\omega_0, \omega_1, \ldots, \omega_{n+1}) = \begin{cases} i, & \text{if the } i^{\text{th}} \text{ reply } \omega_i \text{ is used in forgery} \\ 0, & \text{if } \mathcal{A} \text{ fails} \end{cases},$$

- ▷ For any $\overline{\boldsymbol{\omega}} = (\omega_0, \ldots, \omega_{i-1}, \overline{\omega}_i, \ldots, \overline{\omega}_{n+1})$, $\mathcal{A}$ behaves identically up to the $i^{\text{th}}$ query as with the randomness $\boldsymbol{\omega}$.
- ▷ To extract the secret key sk, we must find $\boldsymbol{\omega}$ and $\overline{\boldsymbol{\omega}}$ such that $A(\boldsymbol{\omega}) = i$ and $A(\overline{\boldsymbol{\omega}}) = i$ and $\omega_i \neq \overline{\omega}_i$.

# Extended classical algorithm

Rewind:

1. Probe random entries $A(\boldsymbol{\omega})$ until $A(r, c) \neq 0$.
2. Store the matrix location $\boldsymbol{\omega}$ and the rewinding point $i \leftarrow A(\boldsymbol{\omega})$.
3. Probe random entries $A(\overline{\boldsymbol{\omega}})$ until $A(\overline{\boldsymbol{\omega}}) = i$.
4. Output the location tuple $(\boldsymbol{\omega}, \overline{\boldsymbol{\omega}})$.

Rewind-Exp:

1. Repeat the procedure Rewind until $\omega_i \neq \overline{\omega}_i$.
2. Use the Knowledge extraction lemma to extract sk.

# Average case complexity I

Assume that $\mathcal{A}$ succeeds with probability $\varepsilon$. Then the results proved for the simplified case (discussed in the last lecture) imply

$$\mathbf{E}[\mathsf{probes}_1] = \tfrac{1}{\varepsilon} \qquad \text{and} \qquad \mathbf{E}[\mathsf{probes}_2 | A(\boldsymbol{\omega}) = i] = \tfrac{1}{\varepsilon_i}$$

where $\varepsilon_i$ is the fraction of entries labelled with $i$. Thus

$$\mathbf{E}[\mathsf{probes}_2] = \sum_{i=1}^{n+1} \Pr\left[A(\boldsymbol{\omega}) = i\right] \cdot \mathbf{E}[\mathsf{probes}_2 | A(\boldsymbol{\omega}) = i]$$

$$\mathbf{E}[\mathsf{probes}_2] = \sum_{i=1}^{n+1} \frac{\varepsilon_i}{\varepsilon} \cdot \frac{1}{\varepsilon_i} = \frac{n+1}{\varepsilon} \ .$$

# Average case complexity II

As a result we obtain that the Rewind algorithm does on average

$$\mathbf{E}[\text{probes}] = \tfrac{n+2}{\varepsilon}$$

probes. Since the Rewind algorithm fails with probability

$$\Pr[\text{failure}] = \frac{\Pr[\text{halting} \wedge \omega_i = \overline{\omega}_i]}{\Pr[\text{halting}]} \leq \frac{\kappa}{\varepsilon}$$

we make on average

$$\mathbf{E}[\text{probes}^*] = \frac{1}{\Pr[\text{success}]} \cdot \mathbf{E}[\text{probes}] \leq \frac{\varepsilon}{\varepsilon - \kappa} \cdot \frac{n+2}{\varepsilon} = \frac{n+2}{\varepsilon - \kappa} \ .$$

# Interpretation of the Results

# Objective and subjective security

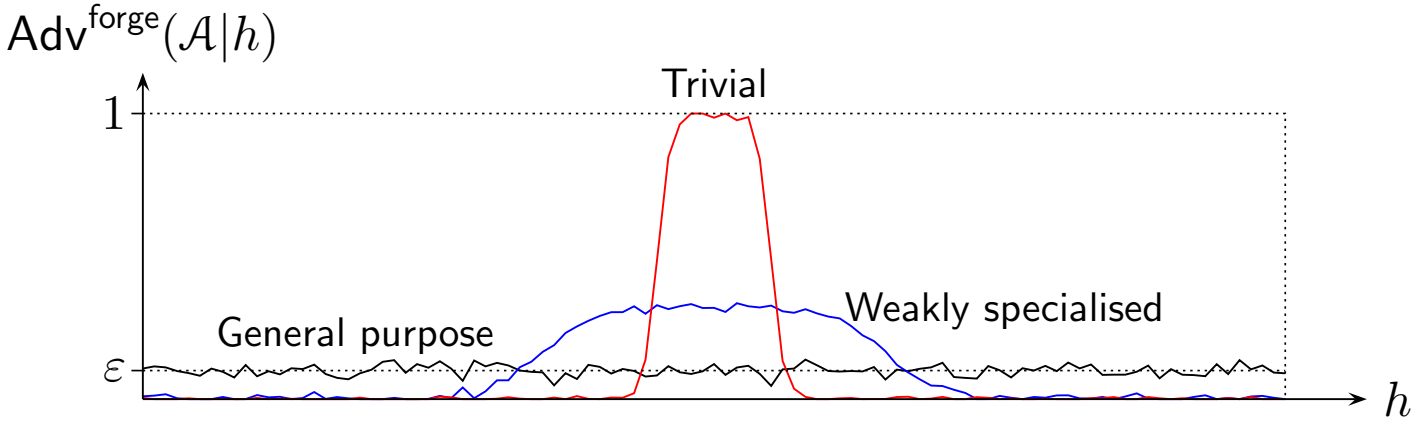
All forms of the Forking Lemma are stated with respect to a fixed key and thus give only soundness guarantees.

If we believe that no human can devise an algorithm that for a particular public key pk computes the corresponding secret key, then we also get the corresponding subjective security guarantee.

For objective security guarantees, we have to consider average success probability over all public keys. In this situation, the knowledge extraction with high probability is overkill. Fixed number of probes together with Jensen's inequality give more tight bounds.

All these security guarantees hold in the random oracle model!

# Average case nature of advantages




The limit on the average advantage over all functions means:

- ▷ An attack algorithm $\mathcal{A}$ can be successful on few functions
- ▷ For randomly chosen function family $\mathcal{H}$ the corresponding average advantage is comparable with high probability over the choice of $\mathcal{H}$.

Such argumentation does not rule out possibility that one can choose adaptively a specialised attack algorithm $\mathcal{A}$ based on the description of $h$.

# Security against generic attacks

An adaptive choice of a specialised attack algorithm implies that the attack depends on the description of the hash function and not the family $\mathcal{H}$.

Often, it is advantageous to consider only generic attacks that depend on the description of function family $\mathcal{H}$ and use only black-box access to the function $h$. Therefore, we can consider two oracles $\mathcal{O}_{\mathcal{H}_{\mathrm{all}}}$ and $\mathcal{O}_{\mathcal{H}}$.

If $\mathcal{H}$ is pseudorandom function family then for any generic attack, we can substitute $\mathcal{H}$ with the $\mathcal{H}_{\mathrm{all}}$ and the success decreases marginally.

**Theorem.** Security in the random oracle model implies security against generic attacks if $\mathcal{H}$ is a pseudorandom function family.

- ▷ The assumption that attackers use only generic attacks is subjective.
- ▷ Such an assumption are not universal, i.e., there are settings where this assumption is clearly irrational (various non-instantiability results).

# Why the random oracle argument truly fails?

All currently used hash functions are iterative

$$h(m_1, \ldots, m_k) = f(\cdots f(f(\mathsf{iv}, m_1), m_2), \ldots, m_k) \ .$$

Consequently, the class of generic attack $\mathfrak{A}_{\mathrm{bb}}$ that treats $h(\cdot)$ as a black box without internal structure is sub-optimal.

Instead, we should consider a wider class of attacks that treat $f$ as a black box, but still try to exploit iterative structure of the hash function.

However, the corresponding black box adversaries $\mathcal{A}^{f(\cdot),h(\cdot)}$ can distinguish $\mathcal{H}$ form $\mathcal{H}_{\mathrm{all}}$ with high probability.

As a result, there might be a successful attack strategy $\mathcal{A}^{f(\cdot),h(\cdot)}$ that works for all possible iterative hash functions, although the signature scheme is secure in the random oracle model.

# Other Interesting Examples

# Generic group signatures

**Setup**

Participants set up a joint public parameters pk such that each of them knows one of the corresponding sub-secrets $\mathsf{sk}_1, \ldots, \mathsf{sk}_n$

**Signing**

Signing procedure is based on sigma protocol that proves that a signer knows a sub-secret $\mathsf{sk}_i$. Again, the challenge is replaced with $h(m, \alpha)$.

**Verification**

A tuple $(m, \alpha, \beta, \gamma)$ is a valid signature if

$$\beta = h(m, \alpha) \wedge \mathsf{Ver}_{\mathsf{pk}}(\alpha, \beta, \gamma) = 1 \ .$$

**Extra property:** Verifier cannot learn who exactly signed the document.

# Forward secure signatures

## Setup

A participant chooses a master secret key $s \xleftarrow{u} \mathbb{Z}_n$ for an RSA modulus and outputs $v = s^{2^n}$ as a public key.

## Signing

The signer can uses $s^{2^{n-1}}, \ldots, s^2, s$ as a secret keys and uses a generic signature scheme based on the Fiat-Shamir identification scheme.

## Verification

A tuple $(m, \alpha, \beta, \gamma)$ is a valid signature if

$$\beta = h(m, \alpha) \wedge \mathsf{Ver}_{\mathsf{pk}}(\alpha, \beta, \gamma) = 1 \ .$$

**Extra property:** Single public key $v$ corresponds to many secret keys $s^{2^{n-1}}, \ldots, s^2, s$ that can be revoked one by one.