Authenticated Encryption & Digital Signatures

CS 276: Introduction to Cryptography

Sanjam Garg

February 24, 2026

Overview

1. Authenticated Encryption
2. Digital Signatures: Definition
3. One-Time Digital Signatures
4. Universal One-Way Hash Functions (UOWHF)

Authenticated Encryption

Goal

Both **confidentiality** (CPA) and **integrity/authenticity**: Bob can decrypt and verify the message was sent by Alice and unmodified.

Definition 1 (Authenticated Encryption)

Π is an **authenticated encryption scheme** if it is CPA-secure and has **ciphertext integrity (CI)**.

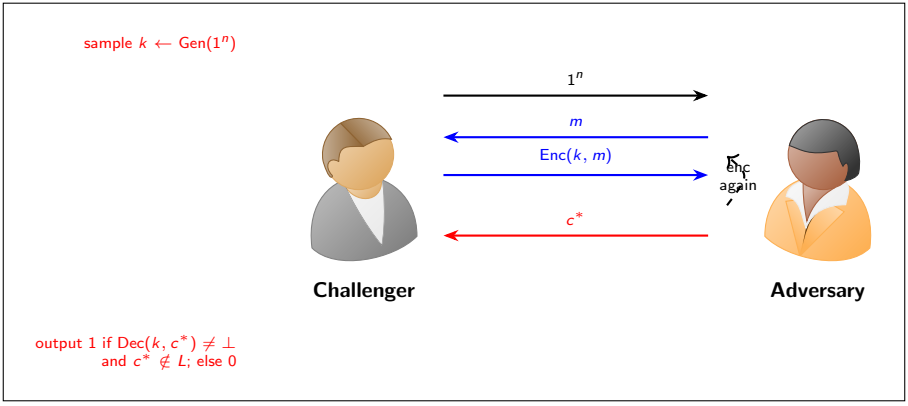
Ciphertext Integrity (CI)

Game $\mathrm{CI}_{\Pi}^{\mathcal{A}}(n)$

1. Challenger samples $k \leftarrow \mathrm{Gen}(1^n)$; $\mathcal{A}$ gets 1^n.
2. **Query**: $\mathcal{A}$ gets oracle $\mathrm{Enc}(k, \cdot)$; may query repeatedly. Let L = list of ciphertexts returned.
3. **Forgery**: $\mathcal{A}$ outputs c^*.
4. Output 1 if $\mathrm{Dec}(k, c^*) \neq \perp$ and $c^* \notin L$; else output 0.

CI: Challenger vs Adversary

$\mathsf{CI}_{\Pi}^{\mathcal{A}}(n)$



Ciphertext Integrity

Definition 2 (Ciphertext Integrity (CI))

Π has **ciphertext integrity** if for all non-uniform PPT $\mathcal{A}$:

$$\Pr[\mathsf{CI}_{\Pi}^{\mathcal{A}}(n) = 1] = \mathsf{negl}(n)$$

Authenticated Encryption $\Rightarrow$ CCA-Secure

Observation

If Π has ciphertext integrity, the adversary cannot produce a *new* valid ciphertext. So on any decryption query $c^* \notin L$, we can return $\perp$. The decryption oracle gives no extra power beyond the encryption oracle.

Authenticated Encryption $\Rightarrow$ CCA-Secure

Observation

If Π has ciphertext integrity, the adversary cannot produce a *new* valid ciphertext. So on any decryption query $c^* \notin L$, we can return $\perp$. The decryption oracle gives no extra power beyond the encryption oracle.

Conclusion

Authenticated encryption $\Rightarrow$ CCA-secure. (CI makes the decryption oracle useless.)

Encrypt-then-MAC

Construction

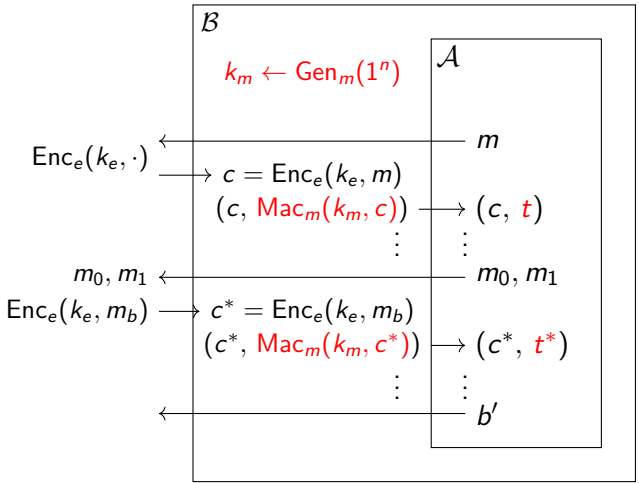
Let $\Pi_e = (\mathsf{Gen}_e, \mathsf{Enc}_e, \mathsf{Dec}_e)$ be CPA-secure and $\Pi_m = (\mathsf{Gen}_m, \mathsf{Mac}_m, \mathsf{Verify}_m)$ be EUF-CMA-secure.

- $\mathsf{Gen}(1^n)$: $k_e \leftarrow \mathsf{Gen}_e(1^n)$, $k_m \leftarrow \mathsf{Gen}_m(1^n)$; return (k_e, k_m)
- $\mathsf{Enc}((k_e, k_m), m)$: $c \leftarrow \mathsf{Enc}_e(k_e, m)$, $t \leftarrow \mathsf{Mac}_m(k_m, c)$; return (c, t)
- $\mathsf{Dec}((k_e, k_m), (c, t))$: if $\mathsf{Verify}_m(k_m, c, t) = 0$ return $\perp$; else return $\mathsf{Dec}_e(k_e, c)$

Claim

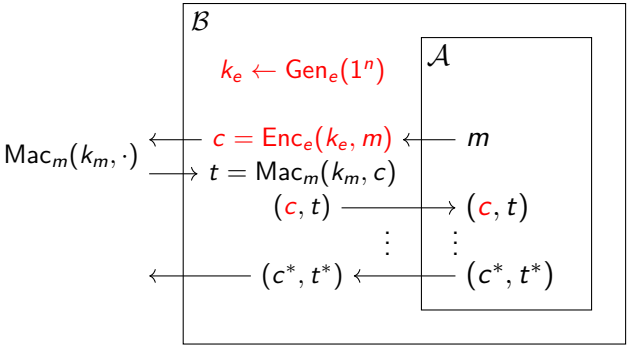
This Π is an authenticated encryption scheme (CPA + CI).

Encrypt-then-MAC: CPA Reduction



$\mathcal{B}$ samples k_m, uses Π_e-oracle for encryption and computes MACs itself. If $\mathcal{A}$ breaks CPA of Π, $\mathcal{B}$ breaks CPA of Π_e.

Encrypt-then-MAC: CI Reduction



$\mathcal{B}$ samples k_e, encrypts with $\mathsf{Enc}_e(k_e, \cdot)$ and gets MAC tags from Π_m-oracle. If $\mathcal{A}$ forges valid (c^*, t^*), then (c^*, t^*) is a MAC forgery for Π_m.

AES-GCM

Example: AES-GCM

Most widely used authenticated encryption scheme.

- Uses **counter-mode** encryption and a MAC (Carter–Wegman style).
- Can authenticate **additional data** (AAD): encryption only on the message; MAC is on (ciphertext + associated data). Useful for authenticating headers.
- More efficient than generic Encrypt-then-MAC (optimized construction).

Digital Signature Scheme

Definition 3 (Digital Signature Scheme)

A **digital signature scheme** is a tuple $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$:

1. $\mathsf{Gen}(1^n) \to (vk, sk)$: outputs verification key vk and signing key sk.
2. $\mathsf{Sign}(sk, m) \to \sigma$: on input sk and message $m \in \{0,1\}^n$, outputs signature σ.
3. $\mathsf{Verify}(vk, m, \sigma) \to \{0,1\}$: outputs 1 if valid, 0 otherwise.

Digital Signature Scheme

Definition 3 (Digital Signature Scheme)

A **digital signature scheme** is a tuple $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$:

1. $\mathsf{Gen}(1^n) \to (vk, sk)$: outputs verification key vk and signing key sk.
2. $\mathsf{Sign}(sk, m) \to \sigma$: on input sk and message $m \in \{0,1\}^n$, outputs signature σ.
3. $\mathsf{Verify}(vk, m, \sigma) \to \{0,1\}$: outputs 1 if valid, 0 otherwise.

Correctness

For all $m \in \{0,1\}^n$:

$$\Pr[(vk, sk) \leftarrow \mathsf{Gen}(1^n),\ \sigma \leftarrow \mathsf{Sign}(sk, m) : \mathsf{Verify}(vk, m, \sigma) = 1] = 1$$

Security Game (EUF-CMA)

Game: adversary vs challenger

1. Challenger samples $(vk, sk) \leftarrow \mathsf{Gen}(1^n)$ and gives vk to $\mathcal{A}$.
2. **Signing oracle**: $\mathcal{A}$ can query $m^{(i)}$ and receives $\sigma^{(i)} \leftarrow \mathsf{Sign}(sk, m^{(i)})$.
3. **Forgery**: $\mathcal{A}$ outputs (m^*, σ^*) with $m^* \notin \{m^{(1)}, \ldots, m^{(q)}\}$.
4. $\mathcal{A}$ **wins** if $\mathsf{Verify}(vk, m^*, \sigma^*) = 1$.

Security Game (EUF-CMA)

Game: adversary vs challenger

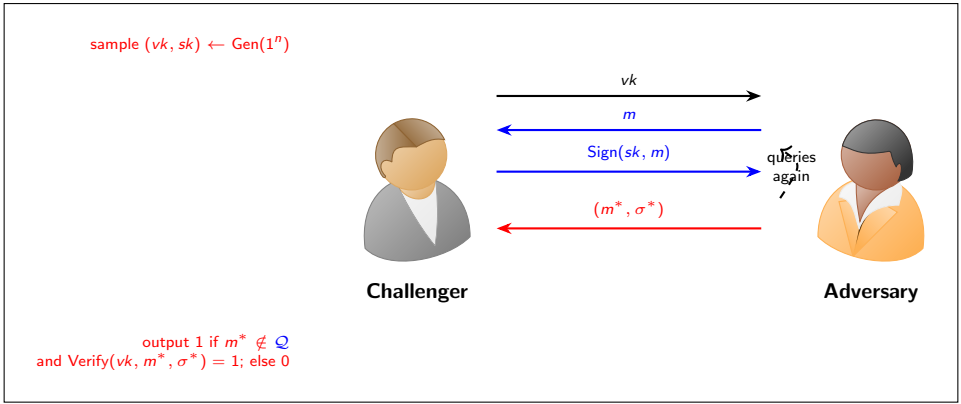
1. Challenger samples $(vk, sk) \leftarrow \mathsf{Gen}(1^n)$ and gives vk to $\mathcal{A}$.
2. **Signing oracle**: $\mathcal{A}$ can query $m^{(i)}$ and receives $\sigma^{(i)} \leftarrow \mathsf{Sign}(sk, m^{(i)})$.
3. **Forgery**: $\mathcal{A}$ outputs (m^*, σ^*) with $m^* \notin \{m^{(1)}, \ldots, m^{(q)}\}$.
4. $\mathcal{A}$ **wins** if $\mathsf{Verify}(vk, m^*, \sigma^*) = 1$.

Definition 4 (Security)

The scheme is **secure** (EUF-CMA) if for every non-uniform PPT $\mathcal{A}$, the probability that $\mathcal{A}$ wins is $\mathsf{negl}(n)$.

EUF-CMA: Challenger vs Adversary

$\text{EUF-CMA}_{\mathcal{A},\Pi}(n)$



One-Time Digital Signature

Weaker notion

Same syntax and correctness as a full signature scheme, but in the security game the adversary may call the **signing oracle only once**.

One-Time Digital Signature

Weaker notion

Same syntax and correctness as a full signature scheme, but in the security game the adversary may call the **signing oracle only once**.

Construction from OWF

Let $f : \{0,1\}^n \to \{0,1\}^n$ be a one-way function.

- $\mathsf{Gen}(1^n)$: choose $x_{i,b} \leftarrow \{0,1\}^n$ for $i \in [n]$, $b \in \{0,1\}$. Set $vk = (f(x_{1,0}), \ldots, f(x_{n,0}); f(x_{1,1}), \ldots, f(x_{n,1}))$, $sk = (x_{i,b})_{i,b}$.
- $\mathsf{Sign}(sk, m)$: output $\sigma = x_{1,m_1} \| \cdots \| x_{n,m_n}$.
- $\mathsf{Verify}(vk, m, \sigma)$: parse σ, check $f(x_{i,m_i}) = vk_{i,m_i}$ for all i.

One-Time Security: Why Only One Query?

Attack with two queries

If $\mathcal{A}$ gets signatures on 0^n and 1^n, then $\mathcal{A}$ receives $x_{i,0}$ and $x_{i,1}$ for all $i \Rightarrow$ **entire secret key** is revealed. Then $\mathcal{A}$ can sign any message.

One-Time Security: Reduction $\mathcal{B}$ (full)

Let $\mathcal{A}$ break one-time security with probability $\mu(n)$. $\mathcal{B}$ gets OWF challenge $y = f(x^*)$ and inverts it.

1. Sample $i^* \leftarrow [n]$, $b^* \leftarrow \{0,1\}$. Set $vk_{i^*,b^*} = y$.
2. For $(i,b) \neq (i^*,b^*)$ sample $x_{i,b} \leftarrow \{0,1\}^n$ and set $vk_{i,b} = f(x_{i,b})$. Send vk to $\mathcal{A}$.
3. **Signing query** m: if $m_{i^*} = b^*$ then **abort** (abort_1). Else sign using $x_{i,b}$ for $(i,b) \neq (i^*,b^*)$ (we don't know x_{i^*,b^*}).
4. **Forgery** (m^*, σ^*): if $m^*_{i^*} = m_{i^*}$ then **abort** (abort_2). Else parse $\sigma^* = x_{1,m^*_1} \| \cdots \| x_{n,m^*_n}$ and output x_{i^*,b^*} (the preimage of y).

One-Time Security: Probability Analysis

When $\mathcal{B}$ does not abort

Conditioned on no abort_1 or abort_2, the view of $\mathcal{A}$ is as in the real game and $\mathcal{B}$'s output is a valid preimage of y whenever $\mathcal{A}$ forges. So $\Pr[\mathcal{B} \text{ inverts } y \mid \text{no abort}] = \mu(n)$.

One-Time Security: Probability Analysis

When $\mathcal{B}$ does not abort

Conditioned on no abort_1 or abort_2, the view of $\mathcal{A}$ is as in the real game and $\mathcal{B}$'s output is a valid preimage of y whenever $\mathcal{A}$ forges. So $\Pr[\mathcal{B} \text{ inverts } y \mid \text{no abort}] = \mu(n)$.

Abort probability

- abort_1: query m has $m_{i^*} = b^* \Rightarrow \Pr[\text{no abort}_1] = 1/2$.
- abort_2: conditioned on no abort_1, forgery has $m^*_{i^*} \neq m_{i^*}$; we need $m^*_{i^*} = b^*$ to extract. For random i^*, $\Pr[m^*_{i^*} = b^* \mid \text{no abort}_1] \geq 1/n$.

So $\Pr[\text{no abort}] \geq 1/(2n)$. Hence $\Pr[\mathcal{B} \text{ inverts } y] \geq \mu(n)/(2n) = \text{non-negl}(n)$. $\square$

UOWHF ("Woof")

Comparison

- **Universal hash**: adversary outputs both colliding inputs *before* seeing the key.
- **Collision-resistant**: adversary outputs both *after* seeing the key.
- **UOWHF**: adversary outputs *one* input before the key, *second* after the key.

UOWHF ("Woof")

Comparison

- **Universal hash**: adversary outputs both colliding inputs *before* seeing the key.
- **Collision-resistant**: adversary outputs both *after* seeing the key.
- **UOWHF**: adversary outputs *one* input before the key, *second* after the key.

Use

UOWHFs are stronger than universal hashes but weaker than CRHFs. They can be built from OWFs and are used to extend one-time signatures to arbitrary-length messages.

UOWHF: Formal Definition

Definition 5 (UOWHF)

A collection $\mathcal{H} = \{h_s : \{0,1\}^* \to \{0,1\}^{\ell(|s|)}\}_s$ is a **universal one-way hash function** if:

1. **Sampling**: PPT $I(1^n)$ samples index s.
2. **Evaluation**: deterministic $h_s(x)$ is computable in poly time.
3. **Security**: $\Pr[\mathrm{UOWHF}_{\mathcal{A}}^{\mathcal{H}}(n) = 1] = \mathrm{negl}(n)$.

UOWHF: Formal Definition

Definition 5 (UOWHF)

A collection $\mathcal{H} = \{h_s : \{0,1\}^* \to \{0,1\}^{\ell(|s|)}\}_s$ is a **universal one-way hash function** if:

1. **Sampling**: PPT $I(1^n)$ samples index s.
2. **Evaluation**: deterministic $h_s(x)$ is computable in poly time.
3. **Security**: $\Pr[\mathrm{UOWHF}_{\mathcal{A}}^{\mathcal{H}}(n) = 1] = \mathrm{negl}(n)$.

Game $\mathrm{UOWHF}_{\mathcal{A}}^{\mathcal{H}}(n)$

1. $\mathcal{A}(1^n)$ outputs $(state, x)$ (first preimage, before seeing the key).
2. Challenger samples $s \leftarrow I(1^n)$ and sends s to $\mathcal{A}$.
3. $\mathcal{A}(state, s)$ outputs y (second preimage).
4. Output 1 if $x \neq y$ and $h_s(x) = h_s(y)$; else output 0.

UOWHF from One-Way Permutation: Step I

$(d, d-1)$-UOWHF

Let $f : \{0,1\}^d \to \{0,1\}^d$ be a one-way permutation; $a, b \in \mathrm{GF}(2^d)$. Define

$$h_{s=(a,b)}(x) = \mathrm{chop}(a \cdot f(x) + b)$$

where chop removes the first bit (output length $d-1$).

UOWHF from One-Way Permutation: Step I

$(d, d-1)$-UOWHF

Let $f : \{0,1\}^d \to \{0,1\}^d$ be a one-way permutation; $a, b \in \mathrm{GF}(2^d)$. Define

$$h_{s=(a,b)}(x) = \mathrm{chop}(a \cdot f(x) + b)$$

where chop removes the first bit (output length $d-1$).

Key fact

$x \mapsto a \cdot f(x) + b$ is one-to-one (for $a \neq 0$); chop is two-to-one. So every image of h_s has **exactly two preimages**.

Step I: Full Proof

$\mathcal{B}$ breaks OWP given $\mathcal{A}$ breaking UOWHF. $\mathcal{B}$ receives $y = f(x^*)$.

1. Run $\mathcal{A}(1^n)$; $\mathcal{A}$ outputs $(state, x_0)$.
2. Choose $s = (a, b)$ so that $h_s(x^*) = h_s(x_0)$: require

$$a \cdot y + b = a \cdot f(x_0) + b + (1 \| 0^{d-1}),$$

   i.e. $a \cdot (y - f(x_0)) = (1 \| 0^{d-1})$. Pick $a \neq 0$ and solve for b (over $\mathrm{GF}(2^d)$). Send s to $\mathcal{A}$.
3. $\mathcal{A}(state, s)$ outputs x_1. $\mathcal{B}$ outputs x_1.

The two preimages of $h_s(x_0) = h_s(x^*)$ are x_0 and x^* (w.o.p. $x^* \neq x_0$). To break UOWHF, $\mathcal{A}$ must output $x_1 \neq x_0$ with $h_s(x_1) = h_s(x_0)$, so $x_1 = x^*$. Thus $\mathcal{B}$ inverts f.
□

UOWHF from OWP: Step II $(2n, n)$

$(2n, n)$-UOWHF

For $d \in \{n+1, \ldots, 2n\}$, let $\mathcal{H}^{d,d-1}$ be a $(d, d-1)$-UOWHF. Compose n layers (from the right, shrinking by 1 each time):

$$H^{2n,n}_{s_1 \cdots s_n}(x) = h^{n+1,n}_{s_1}(h^{n+2,n+1}_{s_2}(\cdots h^{2n,2n-1}_{s_n}(x) \cdots))$$

UOWHF from OWP: Step II $(2n, n)$

$(2n, n)$-UOWHF

For $d \in \{n+1, \ldots, 2n\}$, let $\mathcal{H}^{d,d-1}$ be a $(d, d-1)$-UOWHF. Compose n layers (from the right, shrinking by 1 each time):

$$H^{2n,n}_{s_1 \cdots s_n}(x) = h^{n+1,n}_{s_1}(h^{n+2,n+1}_{s_2}(\cdots h^{2n,2n-1}_{s_n}(x) \cdots))$$

Partial evaluation

$H^i_{\text{partial}}(x)$ = output after the first i layers (from the right). So $H^i_{\text{partial}}(x)$ is the **input** to the i-th layer.

Step II: Full Proof

$\mathcal{B}$ breaks $(d, d-1)$-UOWHF using $\mathcal{A}$ that breaks H.

1. Sample $i \leftarrow [n]$. For $i' \neq i$ sample $s_{i'}$. (Layer i will be the challenge.)
2. Run $\mathcal{A}(1^n)$; get first preimage x_0. Compute $x_0' = H^{i-1}_{\text{partial}}(x_0)$ (input to layer i). Send x_0' to UOWHF challenger.
3. Challenger returns s. Set $s_i = s$; send $(s_1, \ldots, s_n)$ to $\mathcal{A}$.
4. $\mathcal{A}$ returns x_1. Compute $x_1' = H^{i-1}_{\text{partial}}(x_1)$; send x_1' to challenger.

If $\mathcal{A}$ finds collision $x_0 \neq x_1$ with $H(x_0) = H(x_1)$, then there is some layer i^* where $H^{i^*-1}_{\text{partial}}(x_0) \neq H^{i^*-1}_{\text{partial}}(x_1)$ but $H^{i^*}_{\text{partial}}(x_0) = H^{i^*}_{\text{partial}}(x_1) \Rightarrow$ collision in $h_{s_{i^*}}$. With probability $1/n$, $i = i^*$. So $\Pr[\mathcal{B} \text{ wins}] \geq \frac{1}{n} \Pr[\mathcal{A} \text{ wins}]$. $\square$

UOWHF from OWP: Steps III & IV

Step III: $(2*, *)$-UOWHF

Let $\mathcal{H}^{2n,n} = \{h_s : \{0,1\}^{2n} \to \{0,1\}^n\}$ be a $(2n, n)$-UOWHF. Chunk x into blocks of length $2n$ (pad last block with $10\cdots0$); $H_s(x) = h_s(x_1)\| \cdots \| h_s(x_t)$. Same seed s for all chunks.

UOWHF from OWP: Steps III & IV

Step III: $(2*, *)$-UOWHF

Let $\mathcal{H}^{2n,n} = \{h_s : \{0,1\}^{2n} \to \{0,1\}^n\}$ be a $(2n, n)$-UOWHF. Chunk x into blocks of length $2n$ (pad last block with $10\cdots0$); $H_s(x) = h_s(x_1)\| \cdots \| h_s(x_t)$. Same seed s for all chunks.

Step IV: full UOWHF

For $x \in \{0,1\}^{2^t \cdot n}$ define $H_{s_1,\ldots,s_t}(x) = (t,\ h_{s_t}(h_{s_{t-1}}(\cdots h_{s_1}(x)\cdots)))$. Iterate $(2*, *)$ with different seeds; output length fixed. Same seed reused per layer.

Step III: Full Proof

$\mathcal{B}$ breaks $(2n, n)$-UOWHF using $\mathcal{A}$ that breaks H.

1. Run $\mathcal{A}(1^n)$; $\mathcal{A}$ outputs $(state, x^0)$. Let $t = \lceil |x^0|/(2n) \rceil$ (number of chunks after padding).
2. Sample $i \leftarrow [t]$. Send first preimage $x'^0 = x_i^0$ (the i-th chunk of x^0) to the $(2n, n)$-UOWHF challenger.
3. Challenger returns s. Send s to $\mathcal{A}$.
4. $\mathcal{A}(state, s)$ outputs x^1. Send second preimage $x'^1 = x_i^1$ (the i-th chunk of x^1) to the challenger.

If $\mathcal{A}$ finds a collision $x^0 \neq x^1$ with $H_s(x^0) = H_s(x^1)$, then $h_s(x_{i'}^0) = h_s(x_{i'}^1)$ for every chunk i'. Since $x^0 \neq x^1$, there is some i^* with $x_{i^*}^0 \neq x_{i^*}^1$. With probability $\geq 1/t$, $i = i^*$, so $\mathcal{B}$ wins. Hence $\Pr[\mathcal{B} \text{ wins}] \geq \frac{1}{t} \Pr[\mathcal{A} \text{ wins}]$. $\square$

Step IV: Full Proof

$\mathcal{B}$ breaks $(2*, *)$-UOWHF using $\mathcal{A}$ that breaks the full $H_{s_1,\ldots,s_t}$.

1. Sample layer $i \leftarrow [t]$. For $i' \neq i$ sample $s_{i'}$. (Layer i will use the challenge seed.)
2. Run $\mathcal{A}(1^n)$; $\mathcal{A}$ outputs $(state, x^0)$. Compute $x^0_{\text{in}} = h_{s_{i-1}}(\cdots h_{s_1}(x^0)\cdots)$ (input to layer i). Send x^0_{in} to $(2*, *)$-UOWHF challenger.
3. Challenger returns s. Set $s_i = s$; send $(s_1, \ldots, s_t)$ to $\mathcal{A}$.
4. $\mathcal{A}(state, s_1, \ldots, s_t)$ outputs x^1. Compute $x^1_{\text{in}} = h_{s_{i-1}}(\cdots h_{s_1}(x^1)\cdots)$; send x^1_{in} to challenger.

A collision in $H_{s_1,\ldots,s_t}$ implies a collision at some layer i^*. With probability $1/t$, $i = i^*$, so $\Pr[\mathcal{B} \text{ wins}] \geq \frac{1}{t} \Pr[\mathcal{A} \text{ wins}]$. $\square$

One-Time Signatures for Long Messages (UOWHF)

Bad idea

Gen outputs $pk = (pk_\ell, s)$ with $s \leftarrow I(1^n)$; sign m by $\sigma = \mathrm{Sign}_\ell(sk_\ell, h_s(m))$. The seed s is part of the **public key**,