Digital Signatures

CS 276: Introduction to Cryptography

Sanjam Garg

March 2, 2026

Overview

1. One-Shot to Multi-Shot Signatures
2. Collision-Resistant Hash Functions
3. CRHF-Based Many-Time Signatures

UOWHF: Formal Definition

Definition 1 (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 1 (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.

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**,

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**, so the adversary sees s before choosing the message. UOWHF security does not apply (first preimage must be chosen before seeing s).

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**, so the adversary sees s before choosing the message. UOWHF security does not apply (first preimage must be chosen before seeing s).

Correct construction

Let the length-restricted one-time scheme sign messages of length $\ell(n) = |s| + n$ (e.g. $|s| = k(n)$, so $\ell(n) = k(n) + n$).

- $\mathrm{Gen}(1^n)$: run $(pk_\ell, sk_\ell) \leftarrow \mathrm{Gen}_\ell(1^n)$; output (pk_ℓ, sk_ℓ).
- $\mathrm{Sign}(sk_\ell, m)$: sample $s \leftarrow I(1^n)$; set $m_\ell = s \| h_s(m)$; compute $\sigma_\ell \leftarrow \mathrm{Sign}_\ell(sk_\ell, m_\ell)$; output (σ_ℓ, s).
- $\mathrm{Verify}(pk_\ell, m, (\sigma_\ell, s))$: output $\mathrm{Verify}_\ell(pk_\ell, s \| h_s(m), \sigma_\ell)$.

Key point: the seed s is chosen **at signing time** (and sent with the signature), not with the public key.

Proof: One-Time for Long Messages (reduction)

$\mathcal{B}$ breaks one-time security of $(\mathsf{Gen}_\ell, \mathsf{Sign}_\ell, \mathsf{Verify}_\ell)$ using $\mathcal{A}$ that breaks the long-message scheme.

1. $\mathcal{B}$ receives pk_ℓ from challenger; gives pk_ℓ to $\mathcal{A}$.
2. **Signing query** m: $\mathcal{B}$ samples $s \leftarrow I(1^n)$, sets $m_\ell = s \| h_s(m)$, queries its one-time oracle on m_ℓ to get σ_ℓ, returns (σ_ℓ, s) to $\mathcal{A}$.
3. **Forgery** $(m^*, \sigma^* = (s^*, \sigma_\ell^*))$: $\mathcal{B}$ outputs $m_\ell^* = s^* \| h_{s^*}(m^*)$ and σ_ℓ^*.

$\mathcal{B}$ wins if $\mathcal{A}$ forges and $m_\ell^* \neq m_\ell$ (the one-time query).

Proof: One-Time for Long Messages (analysis)

Two cases:

- **Case 1:** $s^* \neq s$. Then $m^*_\ell = s^* \| h_{s^*}(m^*) \neq s \| h_s(m) = m_\ell$. So $\mathcal{B}$ wins whenever $\mathcal{A}$ forges in this case.
- **Case 2:** $s^* = s$. Then $m^*_\ell \neq m_\ell$ only if $h_s(m^*) \neq h_s(m)$. But $\mathcal{A}$ forged with $m^* \neq m$; if $h_s(m^*) = h_s(m)$ then we have a UOWHF collision (same seed s, two distinct preimages m^*, m with same hash). So except with negligible probability, $h_s(m^*) \neq h_s(m)$ and $m^*_\ell \neq m_\ell$; $\mathcal{B}$ wins.

$\mathcal{A}$ has non-negligible success in one of the two cases; in both, $\mathcal{B}$ succeeds with non-negligible probability. So $\mathcal{B}$ breaks one-time security. $\square$

From One-Time to Many-Time Signatures

Goal

Upgrade one-time, arbitrary-length scheme $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ to a full (many-time) digital signature using a **PRF**.

From One-Time to Many-Time Signatures

Goal

Upgrade one-time, arbitrary-length scheme $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ to a full (many-time) digital signature using a **PRF**.

Idea: tree of keys

- For each $\alpha \in \{\epsilon\} \cup \{0,1\}^{\leq n}$, set $(pk_\alpha, sk_\alpha) = \mathsf{Gen}(1^n; \mathsf{PRF}_s(\alpha 10 \cdots 0))$ (randomness from PRF).
- Root: pk_ϵ is the public key; s is the PRF seed.
- To sign m: pick random path $r \leftarrow \{0,1\}^n$. For each level i, sign children's public keys along the path; at leaf sign m. Output path + all signatures.

Tree Construction: GEN and SIGN

GEN(1^n)

Output (pk_ϵ, s) where $s \leftarrow \{0,1\}^n$ is the PRF seed;
$(pk_\epsilon, sk_\epsilon) = \mathsf{Gen}(1^n; \mathsf{PRF}_s(\epsilon 10 \cdots 0))$. Public key is $pk = pk_\epsilon$.

SIGN(sk, m)

1. Draw $r \leftarrow \{0,1\}^n$ (path to leaf).
2. For $i = 0, \ldots, n-1$: let $\alpha_i = r_1 \cdots r_i$, $m_i = pk_{\alpha_i \| 0} \| pk_{\alpha_i \| 1}$, $\sigma_i \leftarrow \mathsf{Sign}(sk_{\alpha_i}, m_i)$.
3. $\sigma_n \leftarrow \mathsf{Sign}(sk_r, m)$ (sign message at leaf).
4. Output $\Sigma = (r, m_0, \sigma_0, \ldots, m_{n-1}, \sigma_{n-1}, \sigma_n)$.

Tree Construction: VERIFY

VERIFY(pk, m, Σ)

Parse $\Sigma = (r, m_0, \sigma_0, \ldots, m_{n-1}, \sigma_{n-1}, \sigma_n)$. From r and the m_i, recompute pk_{α_i} along the path (each m_i contains $pk_{\alpha_i\|0}$ and $pk_{\alpha_i\|1}$; pk_ϵ is from pk). For $i = 0, \ldots, n$, check Verify($pk_{\alpha_i}, m_i, \sigma_i$) (with $m_n = m$). Accept iff all checks pass.

Trust propagates: root pk_ϵ is trusted; each σ_i certifies the next level's keys; leaf signature certifies m.

Security Proof: Hybrids

Assume $\mathcal{A}$ breaks $(\mathsf{GEN}, \mathsf{SIGN}, \mathsf{VERIFY})$ with probability $\varepsilon(n)$. Call the real game H_0.

H_1: PRF $\to$ random function

Replace PRF_s with a truly random function. By PRF security, $\mathcal{A}$ still succeeds with probability $\varepsilon(n) - \mathrm{negl}(n) = \text{non-negl}(n)$.

Security Proof: Hybrids

Assume $\mathcal{A}$ breaks $(\mathsf{GEN}, \mathsf{SIGN}, \mathsf{VERIFY})$ with probability $\varepsilon(n)$. Call the real game H_0.

H_1: PRF $\to$ random function

Replace PRF_s with a truly random function. By PRF security, $\mathcal{A}$ still succeeds with probability $\varepsilon(n) - \mathrm{negl}(n) = \text{non-negl}(n)$.

H_2: Abort if path collision

If for two distinct sign queries $j \neq j'$ we have $r_j = r_{j'}$, abort. Collision probability $\leq q^2/2^n = \mathrm{negl}(n)$. So in H_2, $\mathcal{A}$ still succeeds with non-negligible probability and all queried paths $r_1, \ldots, r_q$ are **distinct**.

Security Proof: Two Cases

$\mathcal{A}$ forges (M^*, Σ^*) with path r^*.

Case 1: $r^* = r_j$ for some $j \in [q]$

Same path as the j-th query. Signatures along the path are deterministic (keys fixed by r^*), so Σ^* agrees with Σ_j until the leaf. At the leaf, $\mathcal{A}$ must have forged a signature on $M^* \neq m_j$ under $pk_{r_j} \Rightarrow$ breaks one-time security at the leaf.

Security Proof: Two Cases

$\mathcal{A}$ forges (M^*, Σ^*) with path r^*.

Case 1: $r^* = r_j$ for some $j \in [q]$

Same path as the j-th query. Signatures along the path are deterministic (keys fixed by r^*), so Σ^* agrees with Σ_j until the leaf. At the leaf, $\mathcal{A}$ must have forged a signature on $M^* \neq m_j$ under $pk_{r_j} \Rightarrow$ breaks one-time security at the leaf.

Case 2: $r^* \neq r_j$ for all j

Path r^* diverges from every r_j. Let i be the first level where r^* differs from all r_j. Then $\mathcal{A}$ forged a signature on $m_i^* = pk_{\alpha_i^* \| 0} \| pk_{\alpha_i^* \| 1}$ under $pk_{\alpha_i^*}$, never signed by the one-time scheme for that node $\Rightarrow$ breaks one-time security at that node.

Security Proof: Reduction $\mathcal{B}$

$\mathcal{B}$ has one-time challenger: gets pk, one signing query, must forge for a different message.

1. Guess $j^* \leftarrow [q]$ (which sign query will be embedded).
2. Build tree: pick path $r_{j^*} \leftarrow \{0,1\}^n$. For every node $\alpha \neq r_{j^*}$, run $\mathsf{Gen}(1^n; R_\alpha)$ to get (pk_α, sk_α). Set $pk_{r_{j^*}} = pk$ (challenger's key at the **leaf**). Give pk_ϵ (root, generated by $\mathcal{B}$) to $\mathcal{A}$.
3. On sign query m_j: if $j \neq j^*$, sign honestly with tree keys. If $j = j^*$, sign internal nodes along r_{j^*} with sk_{α_i}; query one-time oracle on m_{j^*} for the leaf; output Σ_{j^*}.

Security Proof: Reduction $\mathcal{B}$ (cont.)

When $\mathcal{A}$ forges

If $\mathcal{A}$ outputs (M^*, Σ^*) with $r^* = r_{j^*}$, then $M^* \neq m_{j^*}$ (valid forgery). The leaf signature in Σ^* is a one-time forgery under $pk \Rightarrow \mathcal{B}$ forwards it and wins.

Security Proof: Reduction $\mathcal{B}$ (cont.)

When $\mathcal{A}$ forges

If $\mathcal{A}$ outputs (M^*, Σ^*) with $r^* = r_{j^*}$, then $M^* \neq m_{j^*}$ (valid forgery). The leaf signature in Σ^* is a one-time forgery under $pk \Rightarrow \mathcal{B}$ forwards it and wins.

Success probability

$\mathcal{B}$ wins if $\mathcal{A}$ forges and $r^* = r_{j^*}$. Since j^* is uniform in $[q]$ and independent of $\mathcal{A}$'s view (in H_2), $\Pr[r^* = r_{j^*}] \geq 1/q$. So $\Pr[\mathcal{B} \text{ wins}] \geq \frac{1}{q} \cdot \varepsilon'(n)$ where $\varepsilon'(n)$ is $\mathcal{A}$'s success in H_2, hence non-negligible. $\square$

CRHF: Definition

Definition 2 (Collision-resistant hash family)

A family $\{H_n = \{h_i : D_n \to R_n\}_{i \in I_n}\}_n$ with $|D_n| > |R_n|$ is **collision-resistant** if there exist PPT $(\text{Sampler}, \text{Eval})$:

- $\text{Sampler}(1^n) \to i \in I_n$
- $\text{Eval}(i, x) = h_i(x)$
- For every PPT $\mathcal{A}$:

$$\Pr[i \leftarrow \text{Sampler}(1^n), (x, x') \leftarrow \mathcal{A}(1^n, i) : x \neq x' \wedge h_i(x) = h_i(x')] = \text{negl}(n)$$

Adversary sees the key i before finding a collision (stronger than UOWHF).

CRHF from Discrete Log

Assumption

Discrete-log assumption: given (g, g^x) in group $\mathbb{G}_n$, computing x is hard.

CRHF from Discrete Log

Assumption

Discrete-log assumption: given (g, g^x) in group $\mathbb{G}_n$, computing x is hard.

Construction

- Sampler(1^n): pick $x \leftarrow |\mathbb{G}_n|$, set $h = g^x$; output index (g, h).
- Eval$((g, h), (r, s))$: output $g^r h^s$ (two group elements $\rightarrow$ one; compression).

Collision $(r_1, s_1) \neq (r_2, s_2)$ with $g^{r_1} h^{s_1} = g^{r_2} h^{s_2} \Rightarrow r_1 + x s_1 = r_2 + x s_2 \Rightarrow$
$x = (r_1 - r_2)/(s_2 - s_1)$ (discrete log of h).

Merkle–Damgård (2-to-1 to arbitrary length)

Setup

Let $h : \{0,1\}^{2n} \to \{0,1\}^n$ be a 2-to-1 CRHF (same sampler as before). Goal: hash arbitrary-length x to n bits.

Construction

1. Pad x so its length is $m = k \cdot n$ for some k; then pad so k is a power of 2 (e.g. with 0s).
2. Partition into k blocks of n bits: $x = x_1 \| x_2 \| \cdots \| x_k$. View these as the **leaves** of a complete binary tree of depth $\ell = \log_2 k$.
3. Assign to leaf i: $\text{tag}_i = x_i$ (the i-th block).
4. Each internal node (with children labeled by strings $y\|0$ and $y\|1$) gets $\text{tag}_y = h(\text{tag}_{y\|0} \| \text{tag}_{y\|1})$.
5. Output = tag at the **root** (single n-bit value).

Merkle–Damgård: Security and use

Security

If an adversary finds $x \neq x'$ with $H(x) = H(x')$, then the root tags are equal. By induction from the root toward the leaves, there must be some internal node y where the two computations first differ in a child: $\text{tag}_{y\|0}\|\text{tag}_{y\|1} \neq \text{tag}'_{y\|0}\|\text{tag}'_{y\|1}$ but $h(\text{tag}_{y\|0}\|\text{tag}_{y\|1}) = h(\text{tag}'_{y\|0}\|\text{tag}'_{y\|1})$. So we get a collision for h. Thus CR of $h \Rightarrow$ CR of the construction.

Merkle–Damgård: Security and use

Security

If an adversary finds $x \neq x'$ with $H(x) = H(x')$, then the root tags are equal. By induction from the root toward the leaves, there must be some internal node y where the two computations first differ in a child: $\text{tag}_{y\|0}\|\text{tag}_{y\|1} \neq \text{tag}'_{y\|0}\|\text{tag}'_{y\|1}$ but $h(\text{tag}_{y\|0}\|\text{tag}_{y\|1}) = h(\text{tag}'_{y\|0}\|\text{tag}'_{y\|1})$. So we get a collision for h. Thus CR of $h \Rightarrow$ CR of the construction.

One-time signatures for long messages

Sample one-time scheme (sk, vk) for n-bit messages. To sign long m: compute $y = H(m)$ with Merkle–Damgård, then $\sigma \leftarrow \text{Sign}(sk, y)$. Verify by hashing m and checking $\text{Verify}(vk, H(m), \sigma)$. Security: a forgery on $m^* \neq m$ with valid σ^* would give $H(m^*) = H(m)$ (collision) or break one-time security.

Many-Time Signatures from CRHF + One-Time

Two steps

1. **One-time for long messages**: Merkle–Damgård CRHF + one-time sig for n-bit messages $\Rightarrow$ sign long m by signing $H(m)$.
2. **Many-time**: Use PRF to derive a tree of one-time keys (as before); at each node sign children's public keys or the message. Same tree idea, but one-time scheme now supports long messages via CRHF.

Many-Time Signatures from CRHF + One-Time

Two steps

1. **One-time for long messages**: Merkle–Damgård CRHF + one-time sig for n-bit messages $\Rightarrow$ sign long m by signing $H(m)$.
2. **Many-time**: Use PRF to derive a tree of one-time keys (as before); at each node sign children's public keys or the message. Same tree idea, but one-time scheme now supports long messages via CRHF.

Security: PRF security + one-time security; reduction similar to UOWHF-based tree (guess which node is forged).