Random Oracles, FDH, Schnorr, BLS

CS 276: Introduction to Cryptography

Sanjam Garg

March 2, 2026

Overview

1. Trapdoor Permutations and RSA
2. Random Oracle Model
3. Full Domain Hash (RSA-FDH)
4. Schnorr Signatures
5. BLS Signatures

Trapdoor Permutations

Definition 1 (Trapdoor permutation)

A family $\{f_s : D_s \to D_s\}_s$ is a **one-way trapdoor permutation** if there exist PPT (I, D, F, F^{-1}):

- $(s, \tau) \leftarrow I(1^n)$: public index s, trapdoor τ.
- $D(s)$: samples uniform element of D_s.
- $F(s, x) = f_s(x)$; $F^{-1}(\tau, y) = f_s^{-1}(y)$ (inversion with trapdoor).
- f_s is one-way without τ.

RSA Trapdoor Permutation

Parameters

$N = PQ$ (primes $2^{n-1} \leq P < Q \leq 2^n$); e coprime to $\phi(N) = (P-1)(Q-1)$; $d = e^{-1} \bmod \phi(N)$. Public: $s = (N, e)$; trapdoor: $\tau = (N, d)$. Domain: $D_s = \{1, \ldots, N-1\}$.

RSA Trapdoor Permutation

Parameters

$N = PQ$ (primes $2^{n-1} \leq P < Q \leq 2^n$); e coprime to $\phi(N) = (P-1)(Q-1)$; $d = e^{-1} \bmod \phi(N)$. Public: $s = (N, e)$; trapdoor: $\tau = (N, d)$. Domain: $D_s = \{1, \ldots, N-1\}$.

RSA function

- $F_{\text{RSA}}(s, x) = x^e \bmod N$.
- $F^{-1}_{\text{RSA}}(\tau, y) = y^d \bmod N = x$ (since $x^{ed} \equiv x \pmod{N}$).

RSA-FDH Construction

Idea

Sign by applying the **trapdoor inverse** to a hash of the message. The hash maps into the permutation domain ("full domain"), so $f^{-1}(H(m))$ is a valid signature.

RSA-FDH Construction

Idea

Sign by applying the **trapdoor inverse** to a hash of the message. The hash maps into the permutation domain ("full domain"), so $f^{-1}(H(m))$ is a valid signature.

Construction (RSA-FDH)

- Hash $H : \{0,1\}^* \to \mathbb{Z}_N^*$ (e.g. random oracle; output interpreted in $\{1, \ldots, N-1\}$).
- Sign(sk, m): $\sigma = H(m)^d \bmod N$ (invert RSA on $H(m)$).
- Verify(pk, m, σ): accept iff $\sigma^e \equiv H(m) \pmod N$.

RSA-FDH Construction

Idea

Sign by applying the **trapdoor inverse** to a hash of the message. The hash maps into the permutation domain ("full domain"), so $f^{-1}(H(m))$ is a valid signature.

Construction (RSA-FDH)

- Hash $H : \{0,1\}^* \to \mathbb{Z}_N^*$ (e.g. random oracle; output interpreted in $\{1, \ldots, N-1\}$).
- Sign(sk, m): $\sigma = H(m)^d \bmod N$ (invert RSA on $H(m)$).
- Verify(pk, m, σ): accept iff $\sigma^e \equiv H(m) \pmod N$.

Why hash?

Signing raw messages would be insecure (e.g. multiplicative relations). Hashing breaks algebraic structure; security is proved in the **random oracle model**.

Why Random Oracle?

RSA-FDH and standard assumptions

Collision resistance of H is *not* enough for RSA-FDH: e.g. if $H(m_1)H(m_2) \equiv H(m_3) \pmod{N}$, then $\sigma_1\sigma_2 = f^{-1}(H(m_3))$ gives a forgery. No standard-model proof is known.

Why Random Oracle?

RSA-FDH and standard assumptions

Collision resistance of H is *not* enough for RSA-FDH: e.g. if $H(m_1)H(m_2) \equiv H(m_3) \pmod{N}$, then $\sigma_1\sigma_2 = f^{-1}(H(m_3))$ gives a forgery. No standard-model proof is known.

Random oracle methodology

Replace hash H with an **oracle** O for a truly random function. Prove security of Π^O; use this as *evidence* for the real scheme Π^H. Strong assumption: ROM secure $\not\Rightarrow$ secure with every concrete H.

Observability and Programmability

Observability

In ROM, the adversary must *query* the oracle; the reduction **sees** all hash queries. In the standard model, H is fixed and we do not see internal calls.

Observability and Programmability

Observability

In ROM, the adversary must *query* the oracle; the reduction **sees** all hash queries. In the standard model, H is fixed and we do not see internal calls.

Programmability

Oracle answers must look random. We can **program** answers: e.g. set $O(m^*) = y^*$ for a challenge $y^* = f(x^*)$ and use the adversary's forgery to invert the OWP. Not possible with a fixed hash.

RSA-FDH

Scheme

Let $\{f_s\}$ be a trapdoor permutation (e.g. RSA). Hash $H : \{0,1\}^* \to D_s$ (full domain).

- Gen(1^n): $(s, \tau) \leftarrow I(1^n)$; pk $= s$, sk $= \tau$.
- Sign(sk, m): $\sigma = f_\tau^{-1}(H(m))$.
- Verify(pk, m, σ): accept iff $f_s(\sigma) = H(m)$.

RSA-FDH

Scheme

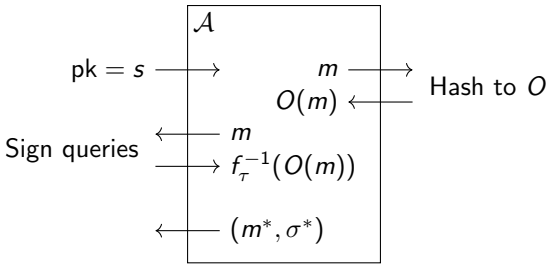
Let $\{f_s\}$ be a trapdoor permutation (e.g. RSA). Hash $H : \{0,1\}^* \to D_s$ (full domain).

- Gen(1^n): $(s, \tau) \leftarrow I(1^n)$; pk $= s$, sk $= \tau$.
- Sign(sk, m): $\sigma = f_\tau^{-1}(H(m))$.
- Verify(pk, m, σ): accept iff $f_s(\sigma) = H(m)$.

Theorem 2

RSA-FDH is EUF-CMA secure in the ROM, assuming $\{f_s\}$ is a secure trapdoor permutation family.

RSA-FDH: Reduction (sketch)



RSA-FDH: Reduction (more detail)

Reduction $\mathcal{B}$ (more detail)

- **Input**: $\mathcal{B}$ receives (s, y^*) from the TDP challenger; goal: output $x^* = f^{-1}(y^*)$.
- **Setup**: Give $\mathsf{pk} = s$ to $\mathcal{A}$. Simulate random oracle O and signing.
- **Guess**: Pick $i^* \leftarrow [q_h]$ (guess that $\mathcal{A}$ will forge on the i^*-th *hash* query m_{i^*}).
- **Oracle**: For the i^*-th hash query m_{i^*}, set $O(m_{i^*}) = y^*$ (program the challenge). For every other hash query m, sample $x \leftarrow D_s$, set $O(m) = f_s(x)$, store (m, x).
- **Sign queries**: On Sign(m), if $m = m_{i^*}$ then abort (can't sign without trapdoor). Else look up x with $O(m) = f_s(x)$ and return $\sigma = x$.
- **Forgery**: If $\mathcal{A}$ outputs (m^*, σ^*) with $m^* = m_{i^*}$, then $f_s(\sigma^*) = O(m_{i^*}) = y^*$, so $\sigma^* = x^*$; $\mathcal{B}$ outputs σ^* and wins.

Success probability: $\Pr[\mathcal{B} \text{ inverts } y^*] \geq \varepsilon(n)/q_h$ (loss by guessing the forged message among hash queries).

Schnorr Signature Scheme

Scheme

Group G of prime order q; hash $H : \{0,1\}^* \to \mathbb{Z}_q$.

- $\mathsf{Gen}(1^n)$: $x \leftarrow \mathbb{Z}_q$, $\mathsf{pk} = g^x$, $\mathsf{sk} = x$.
- $\mathsf{Sign}(\mathsf{sk}, m)$: $k \leftarrow \mathbb{Z}_q$, $r = g^k$, $h = H(m \| r)$, $s = k + hx$; output $\sigma = (h, s)$.
- $\mathsf{Verify}(\mathsf{pk}, m, (h, s))$: accept iff $h = H(m \| (g^s / \mathsf{pk}^h))$.

Schnorr: Security in ROM

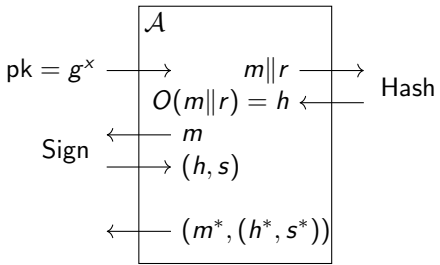
Theorem 3

Schnorr is EUF-CMA secure in the ROM, assuming the discrete log problem is hard in G.

Schnorr: Security in ROM

Theorem 3

Schnorr is EUF-CMA secure in the ROM, assuming the discrete log problem is hard in G.



Schnorr: Reduction (proof details)

Goal

$\mathcal{B}$ gets $(G, q, g, \mathsf{pk} = g^x)$; goal: compute x (discrete log). Uses forger $\mathcal{A}$.

Programming the oracle (sign without sk)

To answer Sign(m) without knowing x: pick $h, s \leftarrow \mathbb{Z}_q$, set $r = g^s/\mathsf{pk}^h$. Then (h, s) is a valid signature: $g^s/\mathsf{pk}^h = r$ and we **define** $O(m\|r) = h$. So $\mathcal{B}$ never needs sk to simulate signing.

Schnorr: Forking and extraction

Forking

Guess $i^* \in [q_h]$ (the hash query that will be used in the forgery). Run $\mathcal{A}$ **twice** with the same random tape; at the i^*-th hash query answer h in the first run and $h' \neq h$ in the second. Same tape $\Rightarrow$ same r^* at that query; two forgeries (h, s) and (h', s') for the same m^*, r^*.

Schnorr: Forking and extraction

Forking

Guess $i^* \in [q_h]$ (the hash query that will be used in the forgery). Run $\mathcal{A}$ **twice** with the same random tape; at the i^*-th hash query answer h in the first run and $h' \neq h$ in the second. Same tape $\Rightarrow$ same r^* at that query; two forgeries (h, s) and (h', s') for the same m^*, r^*.

Extract discrete log

Valid forgeries: $g^s = r^* \cdot \mathsf{pk}^h$ and $g^{s'} = r^* \cdot \mathsf{pk}^{h'}$. So $g^s / g^{s'} = \mathsf{pk}^{h-h'}$, i.e. $g^{s-s'} = g^{x(h-h')}$. Hence $x = (s - s')/(h - h') \bmod q$. $\mathcal{B}$ outputs x.

Schnorr: Forking and extraction

Forking

Guess $i^* \in [q_h]$ (the hash query that will be used in the forgery). Run $\mathcal{A}$ **twice** with the same random tape; at the i^*-th hash query answer h in the first run and $h' \neq h$ in the second. Same tape $\Rightarrow$ same r^* at that query; two forgeries (h, s) and (h', s') for the same m^*, r^*.

Extract discrete log

Valid forgeries: $g^s = r^* \cdot \mathrm{pk}^h$ and $g^{s'} = r^* \cdot \mathrm{pk}^{h'}$. So $g^s / g^{s'} = \mathrm{pk}^{h-h'}$, i.e. $g^{s-s'} = g^{x(h-h')}$. Hence $x = (s - s')/(h - h') \bmod q$. $\mathcal{B}$ outputs x.

Success probability

By the **rewinding/forking lemma**, $\mathcal{B}$ extracts x with probability $\approx \varepsilon^2 / q_h$ (quadratic loss in ROM).

BLS Signatures

Setting

Boneh–Lynn–Shacham: signatures in **pairing-based** groups. Let $e : G_1 \times G_2 \to G_T$ be a bilinear map; $H : \{0,1\}^* \to G_1$ (hash to curve).

BLS Signatures

Setting

Boneh–Lynn–Shacham: signatures in **pairing-based** groups. Let $e : G_1 \times G_2 \to G_T$ be a bilinear map; $H : \{0,1\}^* \to G_1$ (hash to curve).

Scheme

- Gen: secret x, public $g_2^x \in G_2$.
- Sign(x, m): $\sigma = H(m)^x \in G_1$.
- Verify(pk, m, σ): check $e(\sigma, g_2) = e(H(m), \mathrm{pk})$.

Short signatures; security (e.g. co-CDH) in the random oracle model. Used in blockchain and threshold settings.

BLS: Security in ROM

Theorem 4

BLS is EUF-CMA secure in the ROM, assuming the **co-CDH** *problem is hard in* (G_1, G_2).

BLS: Security in ROM

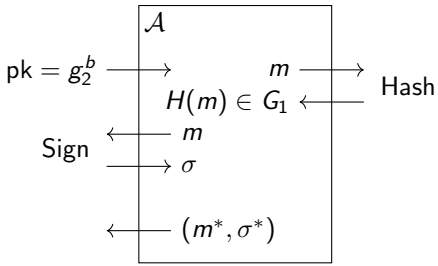
Theorem 4

BLS is EUF-CMA secure in the ROM, assuming the **co-CDH** *problem is hard in* (G_1, G_2).

co-CDH assumption

Given $(g_1, g_2, g_1^a, g_2^b, g_1^b)$ (and e), compute g_1^{ab}. Hard: no PPT can succeed with non-negligible probability.

BLS: Reduction (sketch)



BLS: Reduction (proof details)

Input and setup

$\mathcal{B}$ receives $(g_1, g_2, g_1^a, g_2^b, g_1^b)$; goal: output g_1^{ab}. Set $\mathsf{pk} = g_2^b$ and give it to $\mathcal{A}$. Simulate H and Sign.

Programming H and signing

For most m, set $H(m) = g_1^r$ with $r \leftarrow \mathbb{Z}_q$; then $\sigma = H(m)^b = (g_1^b)^r$ ($\mathcal{B}$ can compute this). For one randomly chosen hash query m_{i^*}, set $H(m_{i^*}) = g_1^a$ (embed the co-CDH challenge). If $\mathcal{A}$ later asks $\mathsf{Sign}(m_{i^*})$, abort.

BLS: Reduction (extraction and success)

Extraction

If $\mathcal{A}$ forges on $m^* = m_{i^*}$, then $\sigma^* = H(m_{i^*})^x = (g_1^a)^b = g_1^{ab}$ (since the "secret key" in the simulation is b). $\mathcal{B}$ outputs σ^* and wins the co-CDH game.

BLS: Reduction (extraction and success)

Extraction

If $\mathcal{A}$ forges on $m^* = m_{i^*}$, then $\sigma^* = H(m_{i^*})^x = (g_1^a)^b = g_1^{ab}$ (since the "secret key" in the simulation is b). $\mathcal{B}$ outputs σ^* and wins the co-CDH game.

Success probability

$\mathcal{B}$ guesses the forged message among the hash queries: $\Pr[i^* \text{ correct}] \geq 1/q_h$. With probability $\approx (1 - 1/q_h)^{q_s}$ the adversary never asks $\text{Sign}(m_{i^*})$. So $\Pr[\mathcal{B} \text{ wins}] \geq \varepsilon \cdot (1 - 1/q_h)^{q_s} \cdot (1/q_h)$, which is non-negligible if ε is.

Summary

- **Random oracle model**: hash as oracle O; observability + programmability enable reductions (e.g. RSA-FDH, Schnorr).
- **RSA-FDH**: sign by inverting f on $H(m)$; EUF-CMA in ROM from trapdoor OWP.
- **Schnorr**: $\sigma = (h, s)$ with $s = k + hx$; EUF-CMA in ROM from discrete log.
- **BLS**: pairing-based short signatures; $H(m)^x$; used in practice.