CS 594 – Advanced Crypto (Spring 2026)

Alex Block

Lecture 20

April 6, 2026

Course Updates

Course Updates

- Final Projects

Course Updates

- Final Projects
  - 6 Groups

Course Updates

- Final Projects
  - 6 Groups
  - 2 Lectures

Course Updates

- Final Projects
  - 6 Groups
  - 2 Lectures
  - **20 min per group presentation** + 3 min for questions.

Course Updates

- Final Projects
  - 6 Groups
  - 2 Lectures
  - **20 min per group presentation** + 3 min for questions.
    - **Hard cut off at 25 min.**

Course Updates

- Final Projects
  - 6 Groups
  - 2 Lectures
  - **20 min per group presentation** + 3 min for questions.
    - **Hard cut off at 25 min.**

- Course Schedule

Course Updates

- Final Projects
  - 6 Groups
  - 2 Lectures
  - **20 min per group presentation** + 3 min for questions.
    - **Hard cut off at 25 min.**

- Course Schedule
  - Today: Full lecture on $i\mathcal{O}$

Course Updates

- Final Projects
  - 6 Groups
  - 2 Lectures
  - **20 min per group presentation** + 3 min for questions.
    - **Hard cut off at 25 min.**

- Course Schedule
  - Today: Full lecture on $i\mathcal{O}$
  - Wed + next Mon: MHFs

Course Updates

- Final Projects
  - 6 Groups
  - 2 Lectures
  - **20 min per group presentation** + 3 min for questions.
    - **Hard cut off at 25 min.**

- Course Schedule
  - Today: Full lecture on $i\mathcal{O}$
  - Wed + next Mon: MHFs
  - Last 3 Lectures: PQ Crypto

More $i\mathcal{O}$ Applications

Last Time: Digital Signatures from $i\mathcal{O}$

Last Time: Digital Signatures from $i\mathcal{O}$

- Let $F = (\mathsf{Eval}, \mathsf{Punc})$ be a PPRF and let $i\mathcal{O}$ be an indistinguishability obfuscator.

Last Time: Digital Signatures from $i\mathcal{O}$

- Let $F = (\mathsf{Eval}, \mathsf{Punc})$ be a PPRF and let $i\mathcal{O}$ be an indistinguishability obfuscator.
- Define circuit $C_k(m, \sigma) = \begin{cases} 1 & \mathsf{Eval}_k(m) = \sigma \\ 0 & \text{otherwise} \end{cases}$.

Last Time: Digital Signatures from $i\mathcal{O}$

- Let $F = (\mathsf{Eval}, \mathsf{Punc})$ be a PPRF and let $i\mathcal{O}$ be an indistinguishability obfuscator.
- Define circuit $C_k(m, \sigma) = \begin{cases} 1 & \mathsf{Eval}_k(m) = \sigma \\ 0 & \text{otherwise} \end{cases}$.
- Define $\Pi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ as follows.

Last Time: Digital Signatures from $i\mathcal{O}$

- Let $F = (\mathsf{Eval}, \mathsf{Punc})$ be a PPRF and let $i\mathcal{O}$ be an indistinguishability obfuscator.
- Define circuit $C_k(m, \sigma) = \begin{cases} 1 & \mathsf{Eval}_k(m) = \sigma \\ 0 & \text{otherwise} \end{cases}$.
- Define $\Pi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ as follows.
  - $\mathsf{Gen}(1^\lambda)$: sample PPRF key $k \xleftarrow{\$} \{0,1\}^\lambda$ uniformly at random.

Last Time: Digital Signatures from $i\mathcal{O}$

- Let $F = (\mathsf{Eval}, \mathsf{Punc})$ be a PPRF and let $i\mathcal{O}$ be an indistinguishability obfuscator.
- Define circuit $C_k(m, \sigma) = \begin{cases} 1 & \mathsf{Eval}_k(m) = \sigma \\ 0 & \text{otherwise} \end{cases}$.
- Define $\Pi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ as follows.
  - $\mathsf{Gen}(1^\lambda)$: sample PPRF key $k \xleftarrow{\$} \{0,1\}^\lambda$ uniformly at random. Define $vk = i\mathcal{O}(C_k)$ and $sk = k$.

Last Time: Digital Signatures from $i\mathcal{O}$

- Let $F = (\mathsf{Eval}, \mathsf{Punc})$ be a PPRF and let $i\mathcal{O}$ be an indistinguishability obfuscator.
- Define circuit $C_k(m, \sigma) = \begin{cases} 1 & \mathsf{Eval}_k(m) = \sigma \\ 0 & \text{otherwise} \end{cases}$.
- Define $\Pi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ as follows.
  - $\mathsf{Gen}(1^\lambda)$: sample PPRF key $k \xleftarrow{\$} \{0,1\}^\lambda$ uniformly at random. Define $vk = i\mathcal{O}(C_k)$ and $sk = k$. Output (sk, vk).

Last Time: Digital Signatures from $i\mathcal{O}$

- Let $F = (\mathsf{Eval}, \mathsf{Punc})$ be a PPRF and let $i\mathcal{O}$ be an indistinguishability obfuscator.
- Define circuit $C_k(m, \sigma) = \begin{cases} 1 & \mathsf{Eval}_k(m) = \sigma \\ 0 & \text{otherwise} \end{cases}$.
- Define $\Pi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ as follows.
  - $\mathsf{Gen}(1^\lambda)$: sample PPRF key $k \xleftarrow{\$} \{0,1\}^\lambda$ uniformly at random. Define $vk = i\mathcal{O}(C_k)$ and $sk = k$. Output (sk, vk).
  - $\mathsf{Sign}_{sk}(m)$: output $\sigma := \mathsf{Eval}_k(m)$.

Last Time: Digital Signatures from $i\mathcal{O}$

- Let $F = (\mathsf{Eval}, \mathsf{Punc})$ be a PPRF and let $i\mathcal{O}$ be an indistinguishability obfuscator.
- Define circuit $C_k(m, \sigma) = \begin{cases} 1 & \mathsf{Eval}_k(m) = \sigma \\ 0 & \text{otherwise} \end{cases}$.
- Define $\Pi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ as follows.
  - $\mathsf{Gen}(1^\lambda)$: sample PPRF key $k \xleftarrow{\$} \{0,1\}^\lambda$ uniformly at random. Define $vk = i\mathcal{O}(C_k)$ and $sk = k$. Output (sk, vk).
  - $\mathsf{Sign}_{sk}(m)$: output $\sigma := \mathsf{Eval}_k(m)$.
  - $\mathsf{Vrfy}_{vk}(m, \sigma)$: output $b := vk(m, \sigma)$.

Last Time: Digital Signatures from $i\mathcal{O}$

- Let $F = (\mathsf{Eval}, \mathsf{Punc})$ be a PPRF and let $i\mathcal{O}$ be an indistinguishability obfuscator.
- Define circuit $C_k(m, \sigma) = \begin{cases} 1 & \mathsf{Eval}_k(m) = \sigma \\ 0 & \text{otherwise} \end{cases}$.
- Define $\Pi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ as follows.
  - $\mathsf{Gen}(1^\lambda)$: sample PPRF key $k \xleftarrow{\$} \{0,1\}^\lambda$ uniformly at random. Define $vk = i\mathcal{O}(C_k)$ and $sk = k$. Output (sk, vk).
  - $\mathsf{Sign}_{sk}(m)$: output $\sigma := \mathsf{Eval}_k(m)$.
  - $\mathsf{Vrfy}_{vk}(m, \sigma)$: output $b := vk(m, \sigma)$.

Goal

Prove that Π above is a selectively secure signature.

Selective Security of Digital Signatures

Selective Security of Digital Signatures

- Reminder: a signature scheme $\Pi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ is *selectively secure/unforgeable* if for all PPT $\mathcal{A}$, there is a negligible function negl such that $\mathcal{A}$ wins the $\mathsf{Sel\text{-}forge}$ game with probability at most $\mathsf{negl}(\lambda)$.

Selective Security of Digital Signatures

- Reminder: a signature scheme $\Pi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ is *selectively secure/unforgeable* if for all PPT $\mathcal{A}$, there is a negligible function negl such that $\mathcal{A}$ wins the $\mathsf{Sel\text{-}forge}$ game with probability at most $\mathsf{negl}(\lambda)$.

$\mathsf{Sel\text{-}forge}_{\mathcal{A},\Pi}(\lambda)$

Selective Security of Digital Signatures

- Reminder: a signature scheme $\Pi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ is *selectively secure/unforgeable* if for all PPT $\mathcal{A}$, there is a negligible function negl such that $\mathcal{A}$ wins the $\mathsf{Sel\text{-}forge}$ game with probability at most $\mathsf{negl}(\lambda)$.

$\mathsf{Sel\text{-}forge}_{\mathcal{A},\Pi}(\lambda)$

1. $m^* \leftarrow \mathcal{A}(1^\lambda)$.

Selective Security of Digital Signatures

- Reminder: a signature scheme $\Pi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ is *selectively secure/unforgeable* if for all PPT $\mathcal{A}$, there is a negligible function negl such that $\mathcal{A}$ wins the $\mathsf{Sel\text{-}forge}$ game with probability at most $\mathsf{negl}(\lambda)$.

$\mathsf{Sel\text{-}forge}_{\mathcal{A},\Pi}(\lambda)$

1. $m^* \leftarrow \mathcal{A}(1^\lambda)$.
2. $(sk, vk) \leftarrow \mathsf{Gen}(1^\lambda)$.

Selective Security of Digital Signatures

- Reminder: a signature scheme $\Pi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ is *selectively secure/unforgeable* if for all PPT $\mathcal{A}$, there is a negligible function negl such that $\mathcal{A}$ wins the $\mathsf{Sel\text{-}forge}$ game with probability at most $\mathsf{negl}(\lambda)$.

$\mathsf{Sel\text{-}forge}_{\mathcal{A},\Pi}(\lambda)$

1. $m^* \leftarrow \mathcal{A}(1^\lambda)$.
2. $(sk, vk) \leftarrow \mathsf{Gen}(1^\lambda)$.
3. $\sigma^* \leftarrow \mathcal{A}^{\mathsf{Sign}_{sk}(\cdot)}(1^\lambda, vk)$.

Selective Security of Digital Signatures

- Reminder: a signature scheme $\Pi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ is *selectively secure/unforgeable* if for all PPT $\mathcal{A}$, there is a negligible function negl such that $\mathcal{A}$ wins the $\mathsf{Sel\text{-}forge}$ game with probability at most $\mathsf{negl}(\lambda)$.

$\mathsf{Sel\text{-}forge}_{\mathcal{A},\Pi}(\lambda)$

1. $m^* \leftarrow \mathcal{A}(1^\lambda)$.
2. $(sk, vk) \leftarrow \mathsf{Gen}(1^\lambda)$.
3. $\sigma^* \leftarrow \mathcal{A}^{\mathsf{Sign}_{sk}(\cdot)}(1^\lambda, vk)$. $(m^*, \sigma^*) \notin Q$
4. Output 1 if $\mathsf{Vrfy}_{vk}(m^*, \sigma^*) = 1$ and $m^* \notin \mathcal{Q}$, where $\mathcal{Q}$ is the set of all messages that $\mathcal{A}$ sent to oracle $\mathsf{Sign}_{sk}(\cdot)$.

Selective Security of Digital Signatures

- Reminder: a signature scheme $\Pi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ is *selectively secure/unforgeable* if for all PPT $\mathcal{A}$, there is a negligible function negl such that $\mathcal{A}$ wins the $\mathsf{Sel\text{-}forge}$ game with probability at most $\mathsf{negl}(\lambda)$.

$\mathsf{Sel\text{-}forge}_{\mathcal{A},\Pi}(\lambda)$

1. $m^* \leftarrow \mathcal{A}(1^\lambda)$.
2. $(sk, vk) \leftarrow \mathsf{Gen}(1^\lambda)$.
3. $\sigma^* \leftarrow \mathcal{A}^{\mathsf{Sign}_{sk}(\cdot)}(1^\lambda, vk)$.
4. Output 1 if $\mathsf{Vrfy}_{vk}(m^*, \sigma^*) = 1$ and $m^* \notin \mathcal{Q}$, where $\mathcal{Q}$ is the set of all messages that $\mathcal{A}$ sent to oracle $\mathsf{Sign}_{sk}(\cdot)$. Otherwise, output 0.

Selective Security Proof: Hybrid Argument

Selective Security Proof: Hybrid Argument

- Our security proof will go through a hybrid argument, and we will argue indistinguishability of these hybrids.

Selective Security Proof: Hybrid Argument

- Our security proof will go through a hybrid argument, and we will argue indistinguishability of these hybrids.

Hybrid H_0

Selective Security Proof: Hybrid Argument

- Our security proof will go through a hybrid argument, and we will argue indistinguishability of these hybrids.

Hybrid H_0

In this hybrid, the game Sel-forge is played with $\mathcal{A}$ and our specified Π.

Selective Security Proof: Hybrid Argument

- Our security proof will go through a hybrid argument, and we will argue indistinguishability of these hybrids.

Hybrid H_0

In this hybrid, the game Sel-forge is played with $\mathcal{A}$ and our specified Π.

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

Selective Security Proof: Hybrid Argument

- Our security proof will go through a hybrid argument, and we will argue indistinguishability of these hybrids.

Hybrid H_0

In this hybrid, the game Sel-forge is played with $\mathcal{A}$ and our specified Π.

- Let $f : \{0,1\}^\lambda \to \{0,1\}^\lambda$ be a one-way permutation.
- Define new circuit $C_{f,k}(\sigma, m) = \begin{cases} 1 & f(\mathsf{Eval}_k(m)) = f(\sigma) \\ 0 & \text{otherwise} \end{cases}$.

Selective Security Proof: Hybrid Argument

- Our security proof will go through a hybrid argument, and we will argue indistinguishability of these hybrids.

Hybrid H_0

In this hybrid, the game Sel-forge is played with $\mathcal{A}$ and our specified Π.

- Let $f : \{0,1\}^\lambda \to \{0,1\}^\lambda$ be a one-way permutation.
- Define new circuit $C_{f,k}(\sigma, m) = \begin{cases} 1 & f(\mathsf{Eval}_k(m)) = f(\sigma) \\ 0 & \text{otherwise} \end{cases}$.
- **Observation:** C_k and $C_{f,k}$ are *equivalent circuits.*

Selective Security Proof: Hybrid Argument

- Let $f : \{0,1\}^\lambda \to \{0,1\}^\lambda$ be a one-way permutation.
- Define new circuit $C_{f,k}(\sigma, m) = \begin{cases} 1 & f(\mathsf{Eval}_k(m)) = f(\sigma) \\ 0 & \text{otherwise} \end{cases}$.
- **Observation:** C_k and $C_{f,k}$ are *equivalent circuits*.

Selective Security Proof: Hybrid Argument

- Let $f : \{0,1\}^\lambda \to \{0,1\}^\lambda$ be a one-way permutation.
- Define new circuit $C_{f,k}(\sigma, m) = \begin{cases} 1 & f(\mathsf{Eval}_k(m)) = f(\sigma) \\ 0 & \text{otherwise} \end{cases}$.
- **Observation:** C_k and $C_{f,k}$ are *equivalent circuits*.

Hybrid H_1

Selective Security Proof: Hybrid Argument

- Let $f\colon \{0,1\}^\lambda \to \{0,1\}^\lambda$ be a one-way permutation.
- Define new circuit $C_{f,k}(\sigma, m) = \begin{cases} 1 & f(\mathsf{Eval}_k(m)) = f(\sigma) \\ 0 & \text{otherwise} \end{cases}$.
- **Observation:** C_k and $C_{f,k}$ are *equivalent circuits*.

Hybrid H_1

In this hybrid, the game Sel-forge is played with $\mathcal{A}$ and new $\Pi^{(1)} = (\mathsf{Gen}^{(1)}, \mathsf{Sign}^{(1)}, \mathsf{Vrfy}^{(1)})$.

Selective Security Proof: Hybrid Argument

- Let $f : \{0,1\}^\lambda \to \{0,1\}^\lambda$ be a one-way permutation.
- Define new circuit $C_{f,k}(\sigma, m) = \begin{cases} 1 & f(\mathsf{Eval}_k(m)) = f(\sigma) \\ 0 & \text{otherwise} \end{cases}$.
- **Observation:** C_k and $C_{f,k}$ are *equivalent circuits*.

Hybrid H_1

In this hybrid, the game Sel-forge is played with $\mathcal{A}$ and new $\Pi^{(1)} = (\mathsf{Gen}^{(1)}, \mathsf{Sign}^{(1)}, \mathsf{Vrfy}^{(1)})$.

- $\mathsf{Gen}^{(1)}(1^\lambda)$: sample PPRF key $k \xleftarrow{\$} \{0,1\}^\lambda$ and OWP f.

Selective Security Proof: Hybrid Argument

- Let $f : \{0,1\}^\lambda \to \{0,1\}^\lambda$ be a one-way permutation.
- Define new circuit $C_{f,k}(\sigma, m) = \begin{cases} 1 & f(\mathsf{Eval}_k(m)) = f(\sigma) \\ 0 & \text{otherwise} \end{cases}$.
- **Observation:** C_k and $C_{f,k}$ are *equivalent circuits*.

Hybrid H_1

In this hybrid, the game Sel-forge is played with $\mathcal{A}$ and new $\Pi^{(1)} = (\mathsf{Gen}^{(1)}, \mathsf{Sign}^{(1)}, \mathsf{Vrfy}^{(1)})$.

- $\mathsf{Gen}^{(1)}(1^\lambda)$: sample PPRF key $k \xleftarrow{\$} \{0,1\}^\lambda$ and OWP f. Define $vk = i\mathcal{O}(C_{f,k})$ and $sk = k$.

Selective Security Proof: Hybrid Argument

- Let $f : \{0,1\}^\lambda \to \{0,1\}^\lambda$ be a one-way permutation.
- Define new circuit $C_{f,k}(\sigma, m) = \begin{cases} 1 & f(\mathsf{Eval}_k(m)) = f(\sigma) \\ 0 & \text{otherwise} \end{cases}$.
- **Observation:** C_k and $C_{f,k}$ are *equivalent circuits*.

Hybrid H_1

In this hybrid, the game Sel-forge is played with $\mathcal{A}$ and new $\Pi^{(1)} = (\mathsf{Gen}^{(1)}, \mathsf{Sign}^{(1)}, \mathsf{Vrfy}^{(1)})$.

- $\mathsf{Gen}^{(1)}(1^\lambda)$: sample PPRF key $k \xleftarrow{\$} \{0,1\}^\lambda$ and OWP f. Define $vk = i\mathcal{O}(C_{f,k})$ and $sk = k$. Output (sk, vk).

Selective Security Proof: Hybrid Argument

- Let $f : \{0,1\}^\lambda \to \{0,1\}^\lambda$ be a one-way permutation.
- Define new circuit $C_{f,k}(\sigma, m) = \begin{cases} 1 & f(\mathsf{Eval}_k(m)) = f(\sigma) \\ 0 & \text{otherwise} \end{cases}$.
- **Observation:** C_k and $C_{f,k}$ are *equivalent circuits*.

Hybrid H_1

In this hybrid, the game Sel-forge is played with $\mathcal{A}$ and new $\Pi^{(1)} = (\mathsf{Gen}^{(1)}, \mathsf{Sign}^{(1)}, \mathsf{Vrfy}^{(1)})$.

- $\mathsf{Gen}^{(1)}(1^\lambda)$: sample PPRF key $k \xleftarrow{\$} \{0,1\}^\lambda$ and OWP f. Define $vk = i\mathcal{O}(C_{f,k})$ and $sk = k$. Output (sk, vk).
- $\mathsf{Sign}^{(1)} = \mathsf{Sign}$. $\mathsf{Sign}^{(1)}_{sk}(m) = \mathsf{Eval}_{sk}(m)$

Selective Security Proof: Hybrid Argument

- Let $f : \{0,1\}^\lambda \to \{0,1\}^\lambda$ be a one-way permutation.
- Define new circuit $C_{f,k}(\sigma, m) = \begin{cases} 1 & f(\mathsf{Eval}_k(m)) = f(\sigma) \\ 0 & \text{otherwise} \end{cases}$.
- **Observation:** C_k and $C_{f,k}$ are *equivalent circuits*.

Hybrid H_1

In this hybrid, the game Sel-forge is played with $\mathcal{A}$ and new $\Pi^{(1)} = (\mathsf{Gen}^{(1)}, \mathsf{Sign}^{(1)}, \mathsf{Vrfy}^{(1)})$.

- $\mathsf{Gen}^{(1)}(1^\lambda)$: sample PPRF key $k \xleftarrow{\$} \{0,1\}^\lambda$ and OWP f. Define $vk = i\mathcal{O}(C_{f,k})$ and $sk = k$. Output (sk, vk).
- $\mathsf{Sign}^{(1)} = \mathsf{Sign}$.
- $\mathsf{Vrfy}^{(1)} = \mathsf{Vrfy}.(m, \sigma) = vk(m, \sigma)$

Selective Security Proof: Hybrid Argument

- Let $f : \{0,1\}^\lambda \to \{0,1\}^\lambda$ be a one-way permutation.
- Define new circuit $C_{f,k}(\sigma, m) = \begin{cases} 1 & f(\mathsf{Eval}_k(m)) = f(\sigma) \\ 0 & \text{otherwise} \end{cases}$.
- **Observation:** C_k and $C_{f,k}$ are *equivalent circuits*.

Hybrid H_1

In this hybrid, the game Sel-forge is played with $\mathcal{A}$ and new $\Pi^{(1)} = (\mathsf{Gen}^{(1)}, \mathsf{Sign}^{(1)}, \mathsf{Vrfy}^{(1)})$.

- $\mathsf{Gen}^{(1)}(1^\lambda)$: sample PPRF key $k \xleftarrow{\$} \{0,1\}^\lambda$ and OWP f. Define $vk = i\mathcal{O}(C_{f,k})$ and $sk = k$. Output (sk, vk).
- $\mathsf{Sign}^{(1)} = \mathsf{Sign}$.
- $\mathsf{Vrfy}^{(1)} = \mathsf{Vrfy}$.

- **Claim:** H_0 and H_1 are indistinguishable by $i\mathcal{O}$ security since C_k and $C_{f,k}$ are equivalent circuits!

Selective Security Proof: Hybrid Argument

Selective Security Proof: Hybrid Argument

- Define new circuit $C_{f,k\{x\},z}(\sigma, m) = \begin{cases} 1 & (f(\sigma), m) = (z, x) \\ 1 & \mathsf{Eval}_{k\{x\}}(m) = \sigma \\ 0 & \text{otherwise} \end{cases}$

Selective Security Proof: Hybrid Argument

- Define new circuit $C_{f,k\{x\},z}(\sigma, m) = \begin{cases} 1 & (f(\sigma), m) = (z, x) \\ 1 & \mathsf{Eval}_{k\{x\}}(m) = \sigma \\ 0 & \text{otherwise} \end{cases}$
- **Observation:** $C_{f,k}$ and $C_{f,k\{x\},z}$ are *equivalent circuits*.

$f(\mathsf{Eval}_K(m)) = f(\sigma)$ $\forall\, m \neq x$

$(f(\sigma), m) = (z, x)$

for $m = x$

$f(\mathsf{Eval}_K(x)) = f(\sigma)$

Selective Security Proof: Hybrid Argument

- Define new circuit $C_{f,k\{x\},z}(\sigma, m) = \begin{cases} 1 & (f(\sigma), m) = (z, x) \\ 1 & \mathsf{Eval}_{k\{x\}}(m) = \sigma \\ 0 & \text{otherwise} \end{cases}$
- **Observation:** $C_{f,k}$ and $C_{f,k\{x\},z}$ are *equivalent circuits.*

Hybrid H_2

Selective Security Proof: Hybrid Argument

- Define new circuit $C_{f,k\{x\},z}(\sigma, m) = \begin{cases} 1 & (f(\sigma), m) = (z, x) \\ 1 & \mathsf{Eval}_{k\{x\}}(m) = \sigma \\ 0 & \text{otherwise} \end{cases}$
- **Observation:** $C_{f,k}$ and $C_{f,k\{x\},z}$ are *equivalent circuits*.

Hybrid H_2

In this hybrid, the game Sel-forge is played with $\mathcal{A}$ and new $\Pi^{(2)} = (\mathsf{Gen}^{(2)}, \mathsf{Sign}^{(2)}, \mathsf{Vrfy}^{(2)})$.

Selective Security Proof: Hybrid Argument

- Define new circuit $C_{f,k\{x\},z}(\sigma, m) = \begin{cases} 1 & (f(\sigma), m) = (z, x) \\ 1 & \mathsf{Eval}_{k\{x\}}(m) = \sigma \\ 0 & \text{otherwise} \end{cases}$
- **Observation:** $C_{f,k}$ and $C_{f,k\{x\},z}$ are *equivalent circuits.*

Hybrid H_2

In this hybrid, the game $\mathsf{Sel\text{-}forge}$ is played with $\mathcal{A}$ and new $\Pi^{(2)} = (\mathsf{Gen}^{(2)}, \mathsf{Sign}^{(2)}, \mathsf{Vrfy}^{(2)})$.

- $\mathsf{Gen}^{(2)}(1^\lambda, m^*)$: sample PPRF key $k \xleftarrow{\$} \{0,1\}^\lambda$ and OWP f.

Selective Security Proof: Hybrid Argument

- Define new circuit $C_{f,k\{x\},z}(\sigma, m) = \begin{cases} 1 & (f(\sigma), m) = (z, x) \\ 1 & \mathsf{Eval}_{k\{x\}}(m) = \sigma \\ 0 & \text{otherwise} \end{cases}$
- **Observation:** $C_{f,k}$ and $C_{f,k\{x\},z}$ are *equivalent circuits*.

Hybrid H_2

In this hybrid, the game Sel-forge is played with $\mathcal{A}$ and new $\Pi^{(2)} = (\mathsf{Gen}^{(2)}, \mathsf{Sign}^{(2)}, \mathsf{Vrfy}^{(2)})$.

- $\mathsf{Gen}^{(2)}(1^\lambda, m^*)$: sample PPRF key $k \xleftarrow{\$} \{0,1\}^\lambda$ and OWP f. Define $vk = i\mathcal{O}(C_{f,k\{m^*\},z^*})$ and $sk = k$, where $z^* = f(\mathsf{Eval}_{k\{m^*\}}(m^*))$ and $m^* \xleftarrow{\$} \mathcal{A}(1^\lambda)$.

Selective Security Proof: Hybrid Argument

- Define new circuit $C_{f,k\{x\},z}(\sigma, m) = \begin{cases} 1 & (f(\sigma), m) = (z, x) \\ 1 & \mathsf{Eval}_{k\{x\}}(m) = \sigma \\ 0 & \text{otherwise} \end{cases}$
- **Observation:** $C_{f,k}$ and $C_{f,k\{x\},z}$ are *equivalent circuits.*

Hybrid H_2

In this hybrid, the game Sel-forge is played with $\mathcal{A}$ and new $\Pi^{(2)} = (\mathsf{Gen}^{(2)}, \mathsf{Sign}^{(2)}, \mathsf{Vrfy}^{(2)})$.

- $\mathsf{Gen}^{(2)}(1^\lambda, m^*)$: sample PPRF key $k \xleftarrow{\$} \{0,1\}^\lambda$ and OWP f. Define $vk = i\mathcal{O}(C_{f,k\{m^*\},z^*})$ and $sk = k$, where $z^* = f(\mathsf{Eval}_{k\{m^*\}}(m^*))$ and $m^* \xleftarrow{\$} \mathcal{A}(1^\lambda)$. Output (sk, vk).

Selective Security Proof: Hybrid Argument

- Define new circuit $C_{f,k\{x\},z}(\sigma, m) = \begin{cases} 1 & (f(\sigma), m) = (z, x) \\ 1 & \mathsf{Eval}_{k\{x\}}(m) = \sigma \\ 0 & \text{otherwise} \end{cases}$
- **Observation:** $C_{f,k}$ and $C_{f,k\{x\},z}$ are *equivalent circuits.*

Hybrid H_2

In this hybrid, the game Sel-forge is played with $\mathcal{A}$ and new $\Pi^{(2)} = (\mathsf{Gen}^{(2)}, \mathsf{Sign}^{(2)}, \mathsf{Vrfy}^{(2)})$.

- $\mathsf{Gen}^{(2)}(1^\lambda, m^*)$: sample PPRF key $k \xleftarrow{\$} \{0,1\}^\lambda$ and OWP f. Define $vk = i\mathcal{O}(C_{f,k\{m^*\},z^*})$ and $sk = k$, where $z^* = f(\mathsf{Eval}_{k\{m^*\}}(m^*))$ and $m^* \xleftarrow{\$} \mathcal{A}(1^\lambda)$. Output (sk, vk).
- $\mathsf{Sign}^{(2)} = \mathsf{Sign}$.

Selective Security Proof: Hybrid Argument

- Define new circuit $C_{f,k\{x\},z}(\sigma, m) = \begin{cases} 1 & (f(\sigma), m) = (z, x) \\ 1 & \mathsf{Eval}_{k\{x\}}(m) = \sigma \\ 0 & \text{otherwise} \end{cases}$
- **Observation:** $C_{f,k}$ and $C_{f,k\{x\},z}$ are *equivalent circuits*.

Hybrid H_2

In this hybrid, the game Sel-forge is played with $\mathcal{A}$ and new $\Pi^{(2)} = (\mathsf{Gen}^{(2)}, \mathsf{Sign}^{(2)}, \mathsf{Vrfy}^{(2)})$.

- $\mathsf{Gen}^{(2)}(1^\lambda, m^*)$: sample PPRF key $k \xleftarrow{\$} \{0,1\}^\lambda$ and OWP f. Define $vk = i\mathcal{O}(C_{f,k\{m^*\},z^*})$ and $sk = k$, where $z^* = f(\mathsf{Eval}_{k\{m^*\}}(m^*))$ and $m^* \xleftarrow{\$} \mathcal{A}(1^\lambda)$. Output (sk, vk).
- $\mathsf{Sign}^{(2)} = \mathsf{Sign}$.
- $\mathsf{Vrfy}^{(2)} = \mathsf{Vrfy}$.

Selective Security Proof: Hybrid Argument

- Define new circuit $C_{f,k\{x\},z}(\sigma, m) = \begin{cases} 1 & (f(\sigma), m) = (z, x) \\ 1 & \mathsf{Eval}_{k\{x\}}(m) = \sigma \\ 0 & \text{otherwise} \end{cases}$
- **Observation:** $C_{f,k}$ and $C_{f,k\{x\},z}$ are *equivalent circuits.*

Hybrid H_2

In this hybrid, the game Sel-forge is played with $\mathcal{A}$ and new $\Pi^{(2)} = (\mathsf{Gen}^{(2)}, \mathsf{Sign}^{(2)}, \mathsf{Vrfy}^{(2)})$.

- $\mathsf{Gen}^{(2)}(1^\lambda, m^*)$: sample PPRF key $k \xleftarrow{\$} \{0,1\}^\lambda$ and OWP f. Define $vk = i\mathcal{O}(C_{f,k\{m^*\},z^*})$ and $sk = k$, where $z^* = f(\mathsf{Eval}_{k\{m^*\}}(m^*))$ and $m^* \xleftarrow{\$} \mathcal{A}(1^\lambda)$. Output (sk, vk).
- $\mathsf{Sign}^{(2)} = \mathsf{Sign}$.
- $\mathsf{Vrfy}^{(2)} = \mathsf{Vrfy}$.

- **Claim:** H_1 and H_2 are indistinguishable by $i\mathcal{O}$ security!

Selective Security Proof: Hybrid Argument

Selective Security Proof: Hybrid Argument

- Keep the same circuit $C_{f,k\{x\},z}(\sigma, m)$.

Selective Security Proof: Hybrid Argument

- Keep the same circuit $C_{f,k\{x\},z}(\sigma, m)$.

Hybrid H_3

Selective Security Proof: Hybrid Argument

- Keep the same circuit $C_{f,k\{x\},z}(\sigma, m)$.

Hybrid H_3

In this hybrid, the game Sel-forge is played with $\mathcal{A}$ and new $\Pi^{(3)} = (\mathsf{Gen}^{(2)}, \mathsf{Sign}^{(3)}, \mathsf{Vrfy}^{(3)})$.

Selective Security Proof: Hybrid Argument

- Keep the same circuit $C_{f,k\{x\},z}(\sigma, m)$.

Hybrid H_3

In this hybrid, the game $\mathsf{Sel\text{-}forge}$ is played with $\mathcal{A}$ and new $\Pi^{(3)} = (\mathsf{Gen}^{(2)}, \mathsf{Sign}^{(3)}, \mathsf{Vrfy}^{(3)})$.

- $\mathsf{Gen}^{(3)}(1^\lambda, m^*)$: sample PPRF key $k \xleftarrow{\$} \{0,1\}^\lambda$, OWP f, and $r \xleftarrow{\$} \{0,1\}^\lambda$.

Selective Security Proof: Hybrid Argument

- Keep the same circuit $C_{f,k\{x\},z}(\sigma, m)$.

Hybrid H_3

In this hybrid, the game Sel-forge is played with $\mathcal{A}$ and new $\Pi^{(3)} = (\mathsf{Gen}^{(2)}, \mathsf{Sign}^{(3)}, \mathsf{Vrfy}^{(3)})$.

- $\mathsf{Gen}^{(3)}(1^\lambda, m^*)$: sample PPRF key $k \xleftarrow{\$} \{0,1\}^\lambda$, OWP f, and $r \xleftarrow{\$} \{0,1\}^\lambda$. Define $vk = i\mathcal{O}(C_{f,k\{m^*\},R})$ and $sk = k$, where $R = f(r)$ and $m^* \xleftarrow{\$} \mathcal{A}(1^\lambda)$.

Selective Security Proof: Hybrid Argument

- Keep the same circuit $C_{f,k\{x\},z}(\sigma, m)$.

Hybrid H_3

In this hybrid, the game Sel-forge is played with $\mathcal{A}$ and new $\Pi^{(3)} = (\mathsf{Gen}^{(2)}, \mathsf{Sign}^{(3)}, \mathsf{Vrfy}^{(3)})$.

- $\mathsf{Gen}^{(3)}(1^\lambda, m^*)$: sample PPRF key $k \xleftarrow{\$} \{0,1\}^\lambda$, OWP f, and $r \xleftarrow{\$} \{0,1\}^\lambda$. Define $vk = i\mathcal{O}(C_{f,k\{m^*\},R})$ and $sk = k$, where $R = f(r)$ and $m^* \xleftarrow{\$} \mathcal{A}(1^\lambda)$. Output (sk, vk).

Selective Security Proof: Hybrid Argument

- Keep the same circuit $C_{f,k\{x\},z}(\sigma, m)$.

Hybrid H_3

In this hybrid, the game Sel-forge is played with $\mathcal{A}$ and new $\Pi^{(3)} = (\mathsf{Gen}^{(2)}, \mathsf{Sign}^{(3)}, \mathsf{Vrfy}^{(3)})$.

- $\mathsf{Gen}^{(3)}(1^\lambda, m^*)$: sample PPRF key $k \xleftarrow{\$} \{0,1\}^\lambda$, OWP f, and $r \xleftarrow{\$} \{0,1\}^\lambda$. Define $vk = i\mathcal{O}(C_{f,k\{m^*\},R})$ and $sk = k$, where $R = f(r)$ and $m^* \xleftarrow{\$} \mathcal{A}(1^\lambda)$. Output (sk, vk).
- $\mathsf{Sign}^{(3)} = \mathsf{Sign}$.

Selective Security Proof: Hybrid Argument

- Keep the same circuit $C_{f,k\{x\},z}(\sigma, m)$.

Hybrid H_3

In this hybrid, the game Sel-forge is played with $\mathcal{A}$ and new $\Pi^{(3)} = (\mathsf{Gen}^{(2)}, \mathsf{Sign}^{(3)}, \mathsf{Vrfy}^{(3)})$.

- $\mathsf{Gen}^{(3)}(1^\lambda, m^*)$: sample PPRF key $k \xleftarrow{\$} \{0,1\}^\lambda$, OWP f, and $r \xleftarrow{\$} \{0,1\}^\lambda$. Define $vk = i\mathcal{O}(C_{f,k\{m^*\},R})$ and $sk = k$, where $R = f(r)$ and $m^* \xleftarrow{\$} \mathcal{A}(1^\lambda)$. Output (sk, vk).
- $\mathsf{Sign}^{(3)} = \mathsf{Sign}$.
- $\mathsf{Vrfy}^{(3)} = \mathsf{Vrfy}$.

Selective Security Proof: Hybrid Argument

- Keep the same circuit $C_{f,k\{x\},z}(\sigma, m)$.

Hybrid H_3

In this hybrid, the game Sel-forge is played with $\mathcal{A}$ and new $\Pi^{(3)} = (\mathsf{Gen}^{(2)}, \mathsf{Sign}^{(3)}, \mathsf{Vrfy}^{(3)})$.

- $\mathsf{Gen}^{(3)}(1^\lambda, m^*)$: sample PPRF key $k \xleftarrow{\$} \{0,1\}^\lambda$, OWP f, and $r \xleftarrow{\$} \{0,1\}^\lambda$. Define $vk = i\mathcal{O}(C_{f,k\{m^*\},R})$ and $sk = k$, where $R = f(r)$ and $m^* \xleftarrow{\$} \mathcal{A}(1^\lambda)$. Output (sk, vk).
- $\mathsf{Sign}^{(3)} = \mathsf{Sign}$.
- $\mathsf{Vrfy}^{(3)} = \mathsf{Vrfy}$.

- **Claim:** H_2 and H_3 are indistinguishable by PPRF security!

Selective Security Proof: Hybrid Argument

- Keep the same circuit $C_{f,k\{x\},z}(\sigma, m)$.

Hybrid H_3

In this hybrid, the game Sel-forge is played with $\mathcal{A}$ and new $\Pi^{(3)} = (\mathsf{Gen}^{(2)}, \mathsf{Sign}^{(3)}, \mathsf{Vrfy}^{(3)})$.

- $\mathsf{Gen}^{(3)}(1^\lambda, m^*)$: sample PPRF key $k \xleftarrow{\$} \{0,1\}^\lambda$, OWP f, and $r \xleftarrow{\$} \{0,1\}^\lambda$. Define $vk = i\mathcal{O}(C_{f,k\{m^*\},R})$ and $sk = k$, where $R = f(r)$ and $m^* \xleftarrow{\$} \mathcal{A}(1^\lambda)$. Output (sk, vk).
- $\mathsf{Sign}^{(3)} = \mathsf{Sign}$.
- $\mathsf{Vrfy}^{(3)} = \mathsf{Vrfy}$.

- **Claim:** H_2 and H_3 are indistinguishable by PPRF security!
  - $R = f(r)$ uniform in $\{0,1\}^\lambda$ since r is random.

Selective Security Proof: Hybrid Argument

- Keep the same circuit $C_{f,k\{x\},z}(\sigma, m)$.

Hybrid H_3

In this hybrid, the game Sel-forge is played with $\mathcal{A}$ and new $\Pi^{(3)} = (\mathsf{Gen}^{(2)}, \mathsf{Sign}^{(3)}, \mathsf{Vrfy}^{(3)})$.

- $\mathsf{Gen}^{(3)}(1^\lambda, m^*)$: sample PPRF key $k \xleftarrow{\$} \{0,1\}^\lambda$, OWP f, and $r \xleftarrow{\$} \{0,1\}^\lambda$. Define $vk = i\mathcal{O}(C_{f,k\{m^*\},R})$ and $sk = k$, where $R = f(r)$ and $m^* \xleftarrow{\$} \mathcal{A}(1^\lambda)$. Output (sk, vk).
- $\mathsf{Sign}^{(3)} = \mathsf{Sign}$.
- $\mathsf{Vrfy}^{(3)} = \mathsf{Vrfy}$.

- **Claim:** H_2 and H_3 are indistinguishable by PPRF security!
  - $R = f(r)$ uniform in $\{0,1\}^\lambda$ since r is random.
  - PPRF security: $\mathsf{Eval}_{k\{m^*\}}(m^*)$ is indistinguishable from random, even if $\mathcal{A}$ is given r!

Selective Security Proof: Hybrid Argument

Selective Security Proof: Hybrid Argument

- So far we have shown that $H_0 \approx_c H_1 \approx_c H_2 \approx_c H_3$.

Selective Security Proof: Hybrid Argument

- So far we have shown that $H_0 \approx_c H_1 \approx_c H_2 \approx_c H_3$.
- Now, we need to argue that $\mathcal{A}$ wins Sel-forge with only negligible probability with respect to Π (the original signature scheme).

Selective Security Proof: Hybrid Argument

- So far we have shown that $H_0 \approx_c H_1 \approx_c H_2 \approx_c H_3$.
- Now, we need to argue that $\mathcal{A}$ wins Sel-forge with only negligible probability with respect to Π (the original signature scheme).
- To do so, we argue that $\mathcal{A}$ wins the forgery game in H_3 with only negligible probability.

Selective Security Proof: Hybrid Argument

- So far we have shown that $H_0 \approx_c H_1 \approx_c H_2 \approx_c H_3$.
- Now, we need to argue that $\mathcal{A}$ wins Sel-forge with only negligible probability with respect to Π (the original signature scheme).
- To do so, we argue that $\mathcal{A}$ wins the forgery game in H_3 with only negligible probability.
  - Then, since $H_3 \approx_c H_0$, this shows selective security.

Selective Security Proof: Hybrid Argument

Selective Security Proof: Hybrid Argument

- Recall in H_3, we have $vk = i\mathcal{O}(C_{f,\{m^*\},R})$ for $R = f(r)$ and $m^* \xleftarrow{\$} \mathcal{A}(1^\lambda)$, where $r \xleftarrow{\$} \{0,1\}^\lambda$.

Selective Security Proof: Hybrid Argument

- Recall in H_3, we have $vk = i\mathcal{O}(C_{f,K\{m^*\},R})$ for $R = f(r)$ and $m^* \xleftarrow{\$} \mathcal{A}(1^\lambda)$, where $r \xleftarrow{\$} \{0,1\}^\lambda$.
  - To forge a signature, $\mathcal{A}$ must output σ^* such that $f(r) = f(\sigma^*)$.

Selective Security Proof: Hybrid Argument

- Recall in H_3, we have $vk = i\mathcal{O}(C_{f,K\{m^*\},R})$ for $R = f(r)$ and $m^* \xleftarrow{\$} \mathcal{A}(1^\lambda)$, where $r \xleftarrow{\$} \{0,1\}^\lambda$.
  - To forge a signature, $\mathcal{A}$ must output σ^* such that $f(r) = f(\sigma^*)$.
  - Security follows by security of the OWP f!

Selective Security Proof: Hybrid Argument

- Recall in H_3, we have $vk = i\mathcal{O}(C_{f,K\{m^*\},R})$ for $R = f(r)$ and $m^* \xleftarrow{\$} \mathcal{A}(1^\lambda)$, where $r \xleftarrow{\$} \{0,1\}^\lambda$.
  - To forge a signature, $\mathcal{A}$ must output σ^* such that $f(r) = f(\sigma^*)$.
  - Security follows by security of the OWP f!
- Therefore, $\mathcal{A}$ wins H_3 with negligible probability, which shows $\mathcal{A}$ wins H_0 with negligible probability.

Short Signatures from $i\mathcal{O}$

Short Signatures from $i\mathcal{O}$

- In our signature construction, the signature is only the output of a PPRF, which has λ-bits of output.

Short Signatures from $i\mathcal{O}$

- In our signature construction, the signature is only the output of a PPRF, which has λ-bits of output.
- These are very short signatures!

Short Signatures from $i\mathcal{O}$

- In our signature construction, the signature is only the output of a PPRF, which has λ-bits of output.
- These are very short signatures!
- This is one advantage of the $i\mathcal{O}$ based construction.

Short Signatures from $i\mathcal{O}$

- In our signature construction, the signature is only the output of a PPRF, which has λ-bits of output.
- These are very short signatures!
- This is one advantage of the $i\mathcal{O}$ based construction.
- Drawbacks:

Short Signatures from $i\mathcal{O}$

- In our signature construction, the signature is only the output of a PPRF, which has λ-bits of output.
- These are very short signatures!
- This is one advantage of the $i\mathcal{O}$ based construction.
- Drawbacks:
  - $|vk|$ is huge (it is an obfuscated circuit).

Short Signatures from $i\mathcal{O}$

- In our signature construction, the signature is only the output of a PPRF, which has λ-bits of output.
- These are very short signatures!
- This is one advantage of the $i\mathcal{O}$ based construction.
- Drawbacks:
  - $|vk|$ is huge (it is an obfuscated circuit).
  - Only theoretical construction, not practically efficient (currently).

Signature Lengths of Other Constructions

Signature Lengths of Other Constructions

- $i\mathcal{O}$ gives us λ-bit signatures.

Signature Lengths of Other Constructions

- $i\mathcal{O}$ gives us λ-bit signatures.
- Comparison with other signatures for security parameter λ:

Signature Lengths of Other Constructions

- $i\mathcal{O}$ gives us λ-bit signatures.
- Comparison with other signatures for security parameter λ:
  - **RSA-FDH**: $\omega(\lambda)$-bits.

Signature Lengths of Other Constructions

- $i\mathcal{O}$ gives us λ-bit signatures.
- Comparison with other signatures for security parameter λ:
  - **RSA-FDH**: $\omega(\lambda)$-bits.
  - **EC-DSA**: 4λ-bits.

Signature Lengths of Other Constructions

- $i\mathcal{O}$ gives us λ-bit signatures.
- Comparison with other signatures for security parameter λ:
  - **RSA-FDH**: $\omega(\lambda)$-bits.
  - **EC-DSA**: 4λ-bits.
  - **Schnorr**: 4λ-bits.

Signature Lengths of Other Constructions

- $i\mathcal{O}$ gives us λ-bit signatures.
- Comparison with other signatures for security parameter λ:
  - **RSA-FDH**: $\omega(\lambda)$-bits.
  - **EC-DSA**: 4λ-bits.
  - **Schnorr**: 4λ-bits.
  - **Short Schnorr**: 3λ-bits.

Signature Lengths of Other Constructions

- $i\mathcal{O}$ gives us λ-bit signatures.
- Comparison with other signatures for security parameter λ:
  - **RSA-FDH**: $\omega(\lambda)$-bits.
  - **EC-DSA**: 4λ-bits.
  - **Schnorr**: 4λ-bits.
  - **Short Schnorr**: 3λ-bits.
    - Suggested in Schnorr's original paper.

Signature Lengths of Other Constructions

- $i\mathcal{O}$ gives us λ-bit signatures.
- Comparison with other signatures for security parameter λ:
  - **RSA-FDH**: $\omega(\lambda)$-bits.
  - **EC-DSA**: 4λ-bits.
  - **Schnorr**: 4λ-bits.
  - **Short Schnorr**: 3λ-bits.
    - Suggested in Schnorr's original paper.
    - Eurocrypt 2022: shown to have λ-bits of security in the GGM+RO.

Signature Lengths of Other Constructions

- $i\mathcal{O}$ gives us λ-bit signatures.
- Comparison with other signatures for security parameter λ:
  - **RSA-FDH**: $\omega(\lambda)$-bits.
  - **EC-DSA**: 4λ-bits.
  - **Schnorr**: 4λ-bits.
  - **Short Schnorr**: 3λ-bits.
    - Suggested in Schnorr's original paper.
    - Eurocrypt 2022: shown to have λ-bits of security in the GGM+RO.
  - **BLS** 2λ-bits

Signature Lengths of Other Constructions

- $i\mathcal{O}$ gives us λ-bit signatures.
- Comparison with other signatures for security parameter λ:
  - **RSA-FDH**: $\omega(\lambda)$-bits.
  - **EC-DSA**: 4λ-bits.
  - **Schnorr**: 4λ-bits.
  - **Short Schnorr**: 3λ-bits.
    - Suggested in Schnorr's original paper.
    - Eurocrypt 2022: shown to have λ-bits of security in the GGM+RO.
  - **BLS** 2λ-bits
    - Requires bilinear pairings for verification.

Signature Lengths of Other Constructions

- $i\mathcal{O}$ gives us λ-bit signatures.
- Comparison with other signatures for security parameter λ:
  - **RSA-FDH**: $\omega(\lambda)$-bits.
  - **EC-DSA**: 4λ-bits.
  - **Schnorr**: 4λ-bits.
  - **Short Schnorr**: 3λ-bits.
    - Suggested in Schnorr's original paper.
    - Eurocrypt 2022: shown to have λ-bits of security in the GGM+RO.
  - **BLS** 2λ-bits
    - Requires bilinear pairings for verification.
    - Shorter signatures but higher computational overhead.

Witness Encryption

Witness Encryption

- Idea: fix an **NP** language L.

Witness Encryption

- Idea: fix an **NP** language L.
  - Encryption occurs with respect to message m, instance x, and security parameter λ.

Witness Encryption

- Idea: fix an **NP** language L.
  - Encryption occurs with respect to message m, instance x, and security parameter λ.
  - Decryption only recovers message m *if decrypter knows witness* w certifying $x \in L$.

Witness Encryption

- Idea: fix an **NP** language L.
  - Encryption occurs with respect to message m, instance x, and security parameter λ.
  - Decryption only recovers message m *if decrypter knows witness* w certifying $x \in L$.

Definition 1 (Witness Encryption)

Witness Encryption

- Idea: fix an **NP** language L.
  - Encryption occurs with respect to message m, instance x, and security parameter λ.
  - Decryption only recovers message m *if decrypter knows witness* w certifying $x \in L$.

Definition 1 (Witness Encryption)

Let $L \in \mathbf{NP}$.

Witness Encryption

- Idea: fix an **NP** language L.
  - Encryption occurs with respect to message m, instance x, and security parameter λ.
  - Decryption only recovers message m *if decrypter knows witness* w certifying $x \in L$.

Definition 1 (Witness Encryption)

Let $L \in \mathbf{NP}$. A *witness encryption scheme with respect to* L $\Pi_L = (\mathsf{Enc}, \mathsf{Dec})$ consists of the following PPT algorithms.

Witness Encryption

- Idea: fix an **NP** language L.
  - Encryption occurs with respect to message m, instance x, and security parameter λ.
  - Decryption only recovers message m *if decrypter knows witness* w certifying $x \in L$.

Definition 1 (Witness Encryption)

Let $L \in \mathbf{NP}$. A *witness encryption scheme with respect to* L $\Pi_L = (\mathsf{Enc}, \mathsf{Dec})$ consists of the following PPT algorithms.

- $\mathsf{Enc}(1^\lambda, x, m)$: the encryption algorithm takes as input security parameter 1^λ, an unbounded-length string x, and a message $m \in \{0,1\}$, and outputs a ciphertext c. $= \mathrm{poly}(\lambda, |x|)$ bits

Witness Encryption

- Idea: fix an **NP** language L.
  - Encryption occurs with respect to message m, instance x, and security parameter λ.
  - Decryption only recovers message m *if decrypter knows witness* w certifying $x \in L$.

Definition 1 (Witness Encryption)

Let $L \in \mathbf{NP}$. A *witness encryption scheme with respect to* L $\Pi_L = (\mathsf{Enc}, \mathsf{Dec})$ consists of the following PPT algorithms.

- $\mathsf{Enc}(1^\lambda, x, m)$: the encryption algorithm takes as input security parameter 1^λ, an unbounded-length string x, and a message $m \in \{0,1\}$, and outputs a ciphertext c.
- $\mathsf{Dec}(c, w)$: takes as input a ciphertext c and an unbounded-length string w and outputs a message $m \in \{0,1\}$ or the (abort) symbol $\perp$.

Witness Encryption

Witness Encryption

Definition 2 (Witness Encryption: Correctness)

Witness Encryption

Definition 2 (Witness Encryption: Correctness)

Let Π_L be a witness encryption scheme for $L \in \mathbf{NP}$.

Witness Encryption

Definition 2 (Witness Encryption: Correctness)

Let Π_L be a witness encryption scheme for $L \in \mathbf{NP}$. We require that for all λ, any $M \in \{0,1\}$, and any $(x,w) \in R_L$, it holds that

Witness Encryption

Definition 2 (Witness Encryption: Correctness)

Let Π_L be a witness encryption scheme for $L \in \mathbf{NP}$. We require that for all λ, any $M \in \{0,1\}$, and any $(x,w) \in R_L$, it holds that

$$\Pr\left[\mathsf{Dec}(\mathsf{Enc}(1^\lambda, x, m), w) = m\right] = 1.$$

Witness Encryption

Definition 2 (Witness Encryption: Correctness)

Let Π_L be a witness encryption scheme for $L \in \mathbf{NP}$. We require that for all λ, any $M \in \{0,1\}$, and any $(x,w) \in R_L$, it holds that

$$\Pr\left[\mathsf{Dec}(\mathsf{Enc}(1^\lambda, x, m), w) = m\right] = 1.$$

Definition 3 (Witness Encryption: Security)

Witness Encryption

Definition 2 (Witness Encryption: Correctness)

Let Π_L be a witness encryption scheme for $L \in \mathbf{NP}$. We require that for all λ, any $M \in \{0,1\}$, and any $(x,w) \in R_L$, it holds that

$$\Pr\left[\mathsf{Dec}(\mathsf{Enc}(1^\lambda, x, m), w) = m\right] = 1.$$

Definition 3 (Witness Encryption: Security)

Let Π_L be a witness encryption scheme for $L \in \mathbf{NP}$.

Witness Encryption

Definition 2 (Witness Encryption: Correctness)

Let Π_L be a witness encryption scheme for $L \in \mathbf{NP}$. We require that for all λ, any $M \in \{0,1\}$, and any $(x,w) \in R_L$, it holds that

$$\Pr\left[\mathsf{Dec}(\mathsf{Enc}(1^\lambda, x, m), w) = m\right] = 1.$$

Definition 3 (Witness Encryption: Security)

Let Π_L be a witness encryption scheme for $L \in \mathbf{NP}$. We say that Π_L is *secure* if for all PPT adversaries $\mathcal{A}$, there exists a negligible function negl such that for any $x \notin L$, we have

Witness Encryption

Definition 2 (Witness Encryption: Correctness)

Let Π_L be a witness encryption scheme for $L \in \mathbf{NP}$. We require that for all λ, any $M \in \{0,1\}$, and any $(x,w) \in R_L$, it holds that

$$\Pr\left[\mathsf{Dec}(\mathsf{Enc}(1^\lambda, x, m), w) = m\right] = 1.$$

Definition 3 (Witness Encryption: Security)

Let Π_L be a witness encryption scheme for $L \in \mathbf{NP}$. We say that Π_L is *secure* if for all PPT adversaries $\mathcal{A}$, there exists a negligible function negl such that for any $x \notin L$, we have

$$\left|\Pr\left[\mathcal{A}(\mathsf{Enc}(1^\lambda, x, 0)) = 1\right] - \Pr\left[\mathcal{A}(\mathsf{Enc}(1^\lambda, x, 1)) = 1\right]\right| \leqslant \mathsf{negl}(\lambda).$$

Witness Encryption is Powerful

Witness Encryption is Powerful

- Let L be an **NP**-complete language.

Witness Encryption is Powerful

- Let L be an **NP**-complete language.
- Suppose that Π_L is a witness encryption scheme for L.

Witness Encryption is Powerful

- Let L be an **NP**-complete language.
- Suppose that Π_L is a witness encryption scheme for L.
- Then, we can use Π_L to build the following primitives.

Witness Encryption is Powerful

- Let L be an **NP**-complete language.
- Suppose that Π_L is a witness encryption scheme for L.
- Then, we can use Π_L to build the following primitives.
  - **Public-key Encryption**: with Witness Encryption and a PRG, we can build a PKE scheme.

Witness Encryption is Powerful

- Let L be an **NP**-complete language.
- Suppose that Π_L is a witness encryption scheme for L.
- Then, we can use Π_L to build the following primitives.
  - **Public-key Encryption**: with Witness Encryption and a PRG, we can build a PKE scheme.
  - **Identity-based Encryption**: with Witness Encryption and a *unique* signature scheme (every message has a unique signature), we can build IBE.

Witness Encryption is Powerful

- Let L be an **NP**-complete language.
- Suppose that Π_L is a witness encryption scheme for L.
- Then, we can use Π_L to build the following primitives.

Witness Encryption is Powerful

- Let L be an **NP**-complete language.
- Suppose that Π_L is a witness encryption scheme for L.
- Then, we can use Π_L to build the following primitives.
  - **Attribute-based Encryption for Circuits**: with Witness Encryption, perfectly binding non-interactive commitments (OWFs), and non-interactive ZAPs (witness-indistinguishable proofs; can obtain from bilinear maps), we can construct ABE for circuits.

Witness Encryption is Powerful

- Let L be an **NP**-complete language.
- Suppose that Π_L is a witness encryption scheme for L.
- Then, we can use Π_L to build the following primitives.
  - **Attribute-based Encryption for Circuits**: with Witness Encryption, perfectly binding non-interactive commitments (OWFs), and non-interactive ZAPs (witness-indistinguishable proofs; can obtain from bilinear maps), we can construct ABE for circuits.
  - **Fully secure IBE**: with Witness Encryption, perfectly binding (non-interactive) commitments, non-interactive ZAPs, and PRFs, we can build fully secure IBE.

Witness Encryption is Powerful

- Let L be an **NP**-complete language.
- Suppose that Π_L is a witness encryption scheme for L.
- Then, we can use Π_L to build the following primitives.
  - **Attribute-based Encryption for Circuits**: with Witness Encryption, perfectly binding non-interactive commitments (OWFs), and non-interactive ZAPs (witness-indistinguishable proofs; can obtain from bilinear maps), we can construct ABE for circuits.
  - **Fully secure IBE**: with Witness Encryption, perfectly binding (non-interactive) commitments, non-interactive ZAPs, and PRFs, we can build fully secure IBE.
- See https://eprint.iacr.org/2013/258.pdf for details.

Witness Encryption from $i\mathcal{O}$

Witness Encryption from $i\mathcal{O}$

- Let $L =$ Circuit-SAT.

Witness Encryption from $i\mathcal{O}$

- Let $L = \mathsf{Circuit\text{-}SAT}$.
  - Note that L is **NP**-complete.

Witness Encryption from $i\mathcal{O}$

- Let $L = \mathsf{Circuit\text{-}SAT}$.
  - Note that L is **NP**-complete.

- We'll use $i\mathcal{O}$ to build a witness encryption scheme with respect to L.

Witness Encryption from $i\mathcal{O}$

- Let $L = \mathsf{Circuit\text{-}SAT}$.
  - Note that L is **NP**-complete.
- We'll use $i\mathcal{O}$ to build a witness encryption scheme with respect to L.
- We'll consider all circuits C with a single bit of output (this is still Circuit-SAT).

Witness Encryption from $i\mathcal{O}$

- Let $L = \mathsf{Circuit\text{-}SAT}$.
  - Note that L is **NP**-complete.
- We'll use $i\mathcal{O}$ to build a witness encryption scheme with respect to L.
- We'll consider all circuits C with a single bit of output (this is still $\mathsf{Circuit\text{-}SAT}$).
- Fix $C\colon \{0,1\}^n \to \{0,1\}$.

Witness Encryption from $i\mathcal{O}$

- Let $L = \mathsf{Circuit\text{-}SAT}$.
  - Note that L is **NP**-complete.
- We'll use $i\mathcal{O}$ to build a witness encryption scheme with respect to L.
- We'll consider all circuits C with a single bit of output (this is still Circuit-SAT).
- Fix $C\colon \{0,1\}^n \to \{0,1\}$.
- **Idea:** Use C as the public key and witness x as the secret key.

Witness Encryption from $i\mathcal{O}$

- Let $L = \mathsf{Circuit\text{-}SAT}$.
  - Note that L is **NP**-complete.
- We'll use $i\mathcal{O}$ to build a witness encryption scheme with respect to L.
- We'll consider all circuits C with a single bit of output (this is still Circuit-SAT).
- Fix $C\colon \{0,1\}^n \to \{0,1\}$.
- **Idea:** Use C as the public key and witness x as the secret key.
  - $\mathsf{Enc}(C, m) = c$ for $m \in \{0,1\}$.

Witness Encryption from $i\mathcal{O}$

- Let $L = \mathsf{Circuit\text{-}SAT}$.
  - Note that L is **NP**-complete.
- We'll use $i\mathcal{O}$ to build a witness encryption scheme with respect to L.
- We'll consider all circuits C with a single bit of output (this is still Circuit-SAT).
- Fix $C\colon \{0,1\}^n \to \{0,1\}$.
- **Idea:** Use C as the public key and witness x as the secret key.
  - $\mathsf{Enc}(C, m) = c$ for $m \in \{0,1\}$.
  - $\mathsf{Dec}(x, c) = m$ if $C(x) = 1$; $\perp$ otherwise.

Witness Encryption from $i\mathcal{O}$

- **Idea:** Use C as the public key and witness x as the secret key.
  - $\mathsf{Enc}(C, m) = c$ for $m \in \{0, 1\}$.
  - $\mathsf{Dec}(x, c) = m$ if $C(x) = 1$; $\perp$ otherwise.

Witness Encryption from $i\mathcal{O}$

- **Idea:** Use C as the public key and witness x as the secret key.
  - $\mathsf{Enc}(C, m) = c$ for $m \in \{0, 1\}$.
  - $\mathsf{Dec}(x, c) = m$ if $C(x) = 1$; $\perp$ otherwise.

- Any party can encrypt using C.

Witness Encryption from $i\mathcal{O}$

- **Idea:** Use C as the public key and witness x as the secret key.
  - $\mathsf{Enc}(C, m) = c$ for $m \in \{0, 1\}$.
  - $\mathsf{Dec}(x, c) = m$ if $C(x) = 1$; $\perp$ otherwise.
- Any party can encrypt using C.
- Can only decrypt if we know x such that $C(x) = 1$.

Witness Encryption from $i\mathcal{O}$

- **Idea:** Use C as the public key and witness x as the secret key.
  - $\mathsf{Enc}(C, m) = c$ for $m \in \{0, 1\}$.
  - $\mathsf{Dec}(x, c) = m$ if $C(x) = 1$; $\perp$ otherwise.
- Any party can encrypt using C.
- Can only decrypt if we know x such that $C(x) = 1$.
- If C is *not satisfiable*, then m is permanently locked in c.

Witness Encryption from $i\mathcal{O}$

- **Idea:** Use C as the public key and witness x as the secret key.
  - $\mathsf{Enc}(C, m) = c$ for $m \in \{0, 1\}$.
  - $\mathsf{Dec}(x, c) = m$ if $C(x) = 1$; $\perp$ otherwise.
- Any party can encrypt using C.
- Can only decrypt if we know x such that $C(x) = 1$.
- If C is *not satisfiable*, then m is permanently locked in c.
  - I.e., attacker cannot distinguish between $\mathsf{Enc}(C, m)$ and $\mathsf{Enc}(C, m')$.

Witness Encryption from $i\mathcal{O}$

Witness Encryption from $i\mathcal{O}$

- Let $C\colon \{0,1\}^n \to \{0,1\}$.

Witness Encryption from $i\mathcal{O}$

- Let $C\colon \{0,1\}^n \to \{0,1\}$.
- Define circuit $D_{C,b}$ for $b \in \{0,1\}$ as follows.

$$D_{C,b}(x) = \begin{cases} b & C(x) = 1 \\ \bot & \text{otherwise} \end{cases}.$$

Witness Encryption from $i\mathcal{O}$

- Let $C\colon \{0,1\}^n \to \{0,1\}$.
- Define circuit $D_{C,b}$ for $b \in \{0,1\}$ as follows.

$$D_{C,b}(x) = \begin{cases} b & C(x) = 1 \\ \bot & \text{otherwise} \end{cases}.$$

- Witness Encryption Construction

Witness Encryption from $i\mathcal{O}$

- Let $C\colon \{0,1\}^n \to \{0,1\}$.
- Define circuit $D_{C,b}$ for $b \in \{0,1\}$ as follows.

$$D_{C,b}(x) = \begin{cases} b & C(x) = 1 \\ \bot & \text{otherwise} \end{cases}.$$

- Witness Encryption Construction
  - $\mathsf{Enc}(1^\lambda, C, m) = i\mathcal{O}(1^\lambda, D_{C,m})$.

Witness Encryption from $i\mathcal{O}$

- Let $C\colon \{0,1\}^n \to \{0,1\}$.
- Define circuit $D_{C,b}$ for $b \in \{0,1\}$ as follows.

$$D_{C,b}(x) = \begin{cases} b & C(x) = 1 \\ \bot & \text{otherwise} \end{cases}.$$

- Witness Encryption Construction
  - $\mathsf{Enc}(1^\lambda, C, m) = i\mathcal{O}(1^\lambda, D_{C,m})$.
    - I.e., ciphertext is an obfuscated circuit.

Witness Encryption from $i\mathcal{O}$

- Let $C\colon \{0,1\}^n \to \{0,1\}$.
- Define circuit $D_{C,b}$ for $b \in \{0,1\}$ as follows.

$$D_{C,b}(x) = \begin{cases} b & C(x) = 1 \\ \bot & \text{otherwise} \end{cases}.$$

- Witness Encryption Construction
  - $\mathsf{Enc}(1^\lambda, C, m) = i\mathcal{O}(1^\lambda, D_{C,m})$.
    - I.e., ciphertext is an obfuscated circuit.
  - $\mathsf{Dec}(x, c) = c(x)$.

Witness Encryption from $i\mathcal{O}$

- Let $C\colon \{0,1\}^n \to \{0,1\}$.
- Define circuit $D_{C,b}$ for $b \in \{0,1\}$ as follows.

$$D_{C,b}(x) = \begin{cases} b & C(x) = 1 \\ \bot & \text{otherwise} \end{cases}.$$

- Witness Encryption Construction
  - $\mathsf{Enc}(1^\lambda, C, m) = i\mathcal{O}(1^\lambda, D_{C,m})$.
    - I.e., ciphertext is an obfuscated circuit.
  - $\mathsf{Dec}(x, c) = c(x)$.
    - Ciphertext should be obfuscated circuit.

Witness Encryption from $i\mathcal{O}$

- Let $C\colon \{0,1\}^n \to \{0,1\}$.
- Define circuit $D_{C,b}$ for $b \in \{0,1\}$ as follows.

$$D_{C,b}(x) = \begin{cases} b & C(x) = 1 \\ \bot & \text{otherwise} \end{cases}.$$

- Witness Encryption Construction
  - $\mathsf{Enc}(1^\lambda, C, m) = i\mathcal{O}(1^\lambda, D_{C,m})$.
    - I.e., ciphertext is an obfuscated circuit.
  - $\mathsf{Dec}(x, c) = c(x)$.
    - Ciphertext should be obfuscated circuit.
    - Evaluate the obfuscated circuit at x.

functionally equivalent

Witness Encryption from $i\mathcal{O}$

Witness Encryption from $i\mathcal{O}$

- Now we need to show that $\Pi = (\mathsf{Enc}, \mathsf{Dec})$ is a correct and secure Witness Encryption Scheme.

Witness Encryption from $i\mathcal{O}$

- Now we need to show that $\Pi = (\mathsf{Enc}, \mathsf{Dec})$ is a correct and secure Witness Encryption Scheme.

- Correctness

Witness Encryption from $i\mathcal{O}$

- Now we need to show that $\Pi = (\mathsf{Enc}, \mathsf{Dec})$ is a correct and secure Witness Encryption Scheme.

- Correctness
  - Follows by correctness of $i\mathcal{O}$!

Witness Encryption from $i\mathcal{O}$

- Now we need to show that $\Pi = (\mathsf{Enc}, \mathsf{Dec})$ is a correct and secure Witness Encryption Scheme.

- Correctness
  - Follows by correctness of $i\mathcal{O}$!

- Security (Hybrid Argument)

Witness Encryption from $i\mathcal{O}$

- Now we need to show that $\Pi = (\mathsf{Enc}, \mathsf{Dec})$ is a correct and secure Witness Encryption Scheme.

- Correctness
  - Follows by correctness of $i\mathcal{O}$!

- Security (Hybrid Argument)
  - Suppose that $C(x) = 0$ for all x (it is unsatisfiable).

Witness Encryption from $i\mathcal{O}$

- Now we need to show that $\Pi = (\mathsf{Enc}, \mathsf{Dec})$ is a correct and secure Witness Encryption Scheme.

- Correctness
  - Follows by correctness of $i\mathcal{O}$!

- Security (Hybrid Argument)
  - Suppose that $C(x) = 0$ for all x (it is unsatisfiable).
  - In this case, $D_{C,b}(x)$ is equivalent to the circuit $D(x) := \bot$.

functionally

Pad(D)

Witness Encryption from $i\mathcal{O}$

- Now we need to show that $\Pi = (\mathsf{Enc}, \mathsf{Dec})$ is a correct and secure Witness Encryption Scheme.

- Correctness
  - Follows by correctness of $i\mathcal{O}$!

- Security (Hybrid Argument)
  - Suppose that $C(x) = 0$ for all x (it is unsatisfiable).
  - In this case, $D_{C,b}(x)$ is equivalent to the circuit $D(x) := \bot$.
  - Security of $i\mathcal{O}$:

Witness Encryption from $i\mathcal{O}$

- Now we need to show that $\Pi = (\mathsf{Enc}, \mathsf{Dec})$ is a correct and secure Witness Encryption Scheme.

- Correctness
  - Follows by correctness of $i\mathcal{O}$!

- Security (Hybrid Argument)
  - Suppose that $C(x) = 0$ for all x (it is unsatisfiable).
  - In this case, $D_{C,b}(x)$ is equivalent to the circuit $D(x) := \bot$.
  - Security of $i\mathcal{O}$:
    - $\mathsf{Enc}(C, m) = i\mathcal{O}(1^\lambda, D_{C,m})$ is indistinguishable from $i\mathcal{O}(1^\lambda, D)$.

pad(D)

Witness Encryption from $i\mathcal{O}$

- Now we need to show that $\Pi = (\mathsf{Enc}, \mathsf{Dec})$ is a correct and secure Witness Encryption Scheme.

- Correctness
  - Follows by correctness of $i\mathcal{O}$!

- Security (Hybrid Argument)
  - Suppose that $C(x) = 0$ for all x (it is unsatisfiable).
  - In this case, $D_{C,b}(x)$ is equivalent to the circuit $D(x) := \bot$.
  - Security of $i\mathcal{O}$:
    - $\mathsf{Enc}(C, m) = i\mathcal{O}(1^\lambda, D_{C,m})$ is indistinguishable from $i\mathcal{O}(1^\lambda, D)$.
    - $\mathsf{Enc}(C, m')$ for $m' \neq m$ is also indistinguishable from $i\mathcal{O}(1^\lambda, D)$.

$\mathsf{Enc}(C, m) \approx_c i\mathcal{O}(1^\lambda, \mathsf{Pad}(D)) \approx_c \mathsf{Enc}(C, m')$

$m \neq m'$

Witness Encryption from $i\mathcal{O}$

- Now we need to show that $\Pi = (\mathsf{Enc}, \mathsf{Dec})$ is a correct and secure Witness Encryption Scheme.

- Correctness
  - Follows by correctness of $i\mathcal{O}$!

- Security (Hybrid Argument)
  - Suppose that $C(x) = 0$ for all x (it is unsatisfiable).
  - In this case, $D_{C,b}(x)$ is equivalent to the circuit $D(x) := \bot$.
  - Security of $i\mathcal{O}$:
    - $\mathsf{Enc}(C, m) = i\mathcal{O}(1^\lambda, D_{C,m})$ is indistinguishable from $i\mathcal{O}(1^\lambda, D)$.
    - $\mathsf{Enc}(C, m')$ for $m' \neq m$ is also indistinguishable from $i\mathcal{O}(1^\lambda, D)$.
    - Therefore, $\mathsf{Enc}(C, m) \approx_c \mathsf{Enc}(C, m')$.

Next Time: Memory-hard Functions