CS 594 – Advanced Crypto (Spring 2026)

Alex Block

Lecture 19

April 1, 2026

Indistinguishability Obfuscation

Obfuscation, Intuitively

Obfuscation, Intuitively

- Formalization of the idea of "program obfuscation."

Obfuscation, Intuitively

- Formalization of the idea of "program obfuscation."

- An *obfuscator* Obf is a probabilistic algorithm that takes as input a program/circut C and security parameter λ and outputs a new program/circuit $C' = \mathsf{Obf}(1^\lambda, C)$.

Obfuscation, Intuitively

- Formalization of the idea of "program obfuscation."
- An *obfuscator* Obf is a probabilistic algorithm that takes as input a program/circut C and security parameter λ and outputs a new program/circuit $C' = \mathsf{Obf}(1^\lambda, C)$.
- **Efficiency:** Obf runs in time $\mathrm{poly}(|C|, \lambda)$.

Obfuscation, Intuitively

- Formalization of the idea of "program obfuscation."
- An *obfuscator* Obf is a probabilistic algorithm that takes as input a program/circut C and security parameter λ and outputs a new program/circuit $C' = \mathsf{Obf}(1^\lambda, C)$.
- **Efficiency:** Obf runs in time $\mathrm{poly}(|C|, \lambda)$.
- **Correctness:** $C' = \mathsf{Obf}(1^\lambda, C)$ is *functionally equivalent* to C.

Obfuscation, Intuitively

- Formalization of the idea of "program obfuscation."
- An *obfuscator* Obf is a probabilistic algorithm that takes as input a program/circut C and security parameter λ and outputs a new program/circuit $C' = \mathsf{Obf}(1^\lambda, C)$.
- **Efficiency:** Obf runs in time $\mathrm{poly}(|C|, \lambda)$.
- **Correctness:** $C' = \mathsf{Obf}(1^\lambda, C)$ is *functionally equivalent* to C.
  - For all inputs x, $C'(x) = C(x)$.

Obfuscation, Intuitively

- Formalization of the idea of "program obfuscation."
- An *obfuscator* Obf is a probabilistic algorithm that takes as input a program/circut C and security parameter λ and outputs a new program/circuit $C' = \mathsf{Obf}(1^\lambda, C)$.
- **Efficiency:** Obf runs in time $\mathrm{poly}(|C|, \lambda)$.
- **Correctness:** $C' = \mathsf{Obf}(1^\lambda, C)$ is *functionally equivalent* to C.
  - For all inputs x, $C'(x) = C(x)$.
- **Security:** how do we define this?

Virtual Black-box Obfuscation

Virtual Black-box Obfuscation

- Ideal, strongest possible security definition: Virtual Black-box Obfuscation (VBB).

Virtual Black-box Obfuscation

- Ideal, strongest possible security definition: Virtual Black-box Obfuscation (VBB).
  - **Intuitive Definition:** Anything an adversary $\mathcal{A}$ could learn from an *obfuscated* circuit $C' = \mathsf{Obf}(1^\lambda, C)$,

Virtual Black-box Obfuscation

- Ideal, strongest possible security definition: Virtual Black-box Obfuscation (VBB).
  - **Intuitive Definition:** Anything an adversary $\mathcal{A}$ could learn from an *obfuscated* circuit $C' = \mathsf{Obf}(1^\lambda, C)$, $\mathcal{A}$ could have learned if they had *oracle* access to C as a black-box.

Virtual Black-box Obfuscation

- Ideal, strongest possible security definition: Virtual Black-box Obfuscation (VBB).
  - **Intuitive Definition:** Anything an adversary $\mathcal{A}$ could learn from an *obfuscated* circuit $C' = \mathsf{Obf}(1^\lambda, C)$, $\mathcal{A}$ could have learned if they had *oracle* access to C as a black-box.

Definition 1 (VBB Obfuscation)

For all PPT adversaries $\mathcal{A}$ there exists a simulator $\mathcal{S}$ such that for all programs/circuits $\{P_n\}_{n\in\mathbb{N}}$ and all security parameters $\lambda \in \mathbb{N}$, we have

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

Virtual Black-box Obfuscation

- Ideal, strongest possible security definition: Virtual Black-box Obfuscation (VBB).
  - **Intuitive Definition:** Anything an adversary $\mathcal{A}$ could learn from an *obfuscated* circuit $C' = \mathsf{Obf}(1^\lambda, C)$, $\mathcal{A}$ could have learned if they had *oracle* access to C as a black-box.

Definition 1 (VBB Obfuscation)

For all PPT adversaries $\mathcal{A}$ there exists a simulator $\mathcal{S}$ such that for all programs/circuits $\{C_n\}_{n\in\mathbb{N}}$ and all security parameters $\lambda \in \mathbb{N}$, we have

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

- **Pros:** This is a very strong security definition!

Virtual Black-box Obfuscation

- Ideal, strongest possible security definition: Virtual Black-box Obfuscation (VBB).
  - **Intuitive Definition:** Anything an adversary $\mathcal{A}$ could learn from an *obfuscated* circuit $C' = \mathsf{Obf}(1^\lambda, C)$, $\mathcal{A}$ could have learned if they had *oracle* access to C as a black-box.

Definition 1 (VBB Obfuscation)

For all PPT adversaries $\mathcal{A}$ there exists a simulator $\mathcal{S}$ such that for all programs/circuits $\{C_n\}_{n\in\mathbb{N}}$ and all security parameters $\lambda \in \mathbb{N}$, we have

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

- **Pros:** This is a very strong security definition!
- **Cons:** ?

VBB Obfuscation is Impossible

VBB Obfuscation is Impossible

VBB Security is Impossible to achieve in general.

VBB Obfuscation is Impossible

VBB Security is Impossible to achieve in general.

Proof.

VBB Obfuscation is Impossible

VBB Security is Impossible to achieve in general.

Proof.

- Let $\alpha, \beta, \gamma \xleftarrow{\$} \{0,1\}^\lambda$.

VBB Obfuscation is Impossible

VBB Security is Impossible to achieve in general.

Proof.

- Let $\alpha, \beta, \gamma \xleftarrow{\$} \{0,1\}^\lambda$.
- Define the following program

VBB Obfuscation is Impossible

VBB Security is Impossible to achieve in general.

Proof.

- Let $\alpha, \beta, \gamma \xleftarrow{\$} \{0,1\}^\lambda$.
- Define the following program

$$P_{\alpha,\beta,\gamma}(x) = \begin{cases} \beta & x = \alpha \\ \gamma & x(\alpha) = \beta \\ \bot & \text{otherwise} \end{cases}.$$

VBB Obfuscation is Impossible

VBB Security is Impossible to achieve in general.

Proof.

- Let $\alpha, \beta, \gamma \xleftarrow{\$} \{0,1\}^\lambda$.
- Define the following program

$$P_{\alpha,\beta,\gamma}(x) = \begin{cases} \beta & x = \alpha \\ \gamma & x(\alpha) = \beta \ . \\ \bot & \text{otherwise} \end{cases}$$

- Here, we interpret the string x as the *description* of a program (e.g., a Turing machine), and $x(\alpha)$ denotes the output of running this program on input x.

VBB Obfuscation is Impossible

VBB Obfuscation is Impossible

- **Observation 1: blackbox queries hide α, β, γ.**

VBB Obfuscation is Impossible

- **Observation 1: blackbox queries hide α, β, γ.**
  - Recall that α, β, γ are uniformly random in $\{0,1\}^\lambda$.

VBB Obfuscation is Impossible

- **Observation 1: blackbox queries hide α, β, γ.**
  - Recall that α, β, γ are uniformly random in $\{0,1\}^\lambda$.
  - Let $\mathcal{S}$ be the simulator given oracle access to $P_{\alpha,\beta,\gamma}$.

VBB Obfuscation is Impossible

- **Observation 1: blackbox queries hide α, β, γ.**
  - Recall that α, β, γ are uniformly random in $\{0,1\}^\lambda$.
  - Let $\mathcal{S}$ be the simulator given oracle access to $P_{\alpha,\beta,\gamma}$.
  - If $\mathcal{S}$ makes at most q queries to its oracle, then all queries made by $\mathcal{S}$ will be answered by $\perp$ except with at most $2q/2^\lambda$ probability.

VBB Obfuscation is Impossible

- **Observation 1: blackbox queries hide α, β, γ.**
  - Recall that α, β, γ are uniformly random in $\{0,1\}^\lambda$.
  - Let $\mathcal{S}$ be the simulator given oracle access to $P_{\alpha,\beta,\gamma}$.
  - If $\mathcal{S}$ makes at most q queries to its oracle, then all queries made by $\mathcal{S}$ will be answered by $\perp$ except with at most $2q/2^\lambda$ probability.
  - Proof sketch:

VBB Obfuscation is Impossible

- **Observation 1: blackbox queries hide α, β, γ.**
  - Recall that α, β, γ are uniformly random in $\{0,1\}^\lambda$.
  - Let $\mathcal{S}$ be the simulator given oracle access to $P_{\alpha,\beta,\gamma}$.
  - If $\mathcal{S}$ makes at most q queries to its oracle, then all queries made by $\mathcal{S}$ will be answered by $\perp$ except with at most $2q/2^\lambda$ probability.
  - Proof sketch:
    - $\Pr[x_1 = \alpha] \leqslant 2^{-\lambda}$,
      $\Pr[x_i = \alpha | P_{\alpha,\beta,\gamma}(x_1) = \cdots = P_{\alpha,\beta,\gamma}(x_{i-1}) = \perp] \leqslant 2^{-\lambda}$.

VBB Obfuscation is Impossible

- **Observation 1: blackbox queries hide α, β, γ.**
  - Recall that α, β, γ are uniformly random in $\{0,1\}^\lambda$.
  - Let $\mathcal{S}$ be the simulator given oracle access to $P_{\alpha,\beta,\gamma}$.
  - If $\mathcal{S}$ makes at most q queries to its oracle, then all queries made by $\mathcal{S}$ will be answered by $\perp$ except with at most $2q/2^\lambda$ probability.
  - Proof sketch:
    - $\Pr[x_1 = \alpha] \leqslant 2^{-\lambda}$,
      $\Pr[x_i = \alpha | P_{\alpha,\beta,\gamma}(x_1) = \cdots = P_{\alpha,\beta,\gamma}(x_{i-1}) = \perp] \leqslant 2^{-\lambda}$.
    - $\Pr[x_1(\alpha) = \beta] \leqslant 2^{-\lambda}$,
      $\Pr[x_i(\alpha) = \beta | P_{\alpha,\beta,\gamma}(x_1) = \cdots = P_{\alpha,\beta,\gamma}(x_{i-1}) = \perp] \leqslant 2^{-\lambda}$.

VBB Obfuscation is Impossible

- **Observation 1: blackbox queries hide α, β, γ.**
  - Recall that α, β, γ are uniformly random in $\{0,1\}^\lambda$.
  - Let $\mathcal{S}$ be the simulator given oracle access to $P_{\alpha,\beta,\gamma}$.
  - If $\mathcal{S}$ makes at most q queries to its oracle, then all queries made by $\mathcal{S}$ will be answered by $\perp$ except with at most $2q/2^\lambda$ probability.
  - Proof sketch:
    - $\Pr[x_1 = \alpha] \leqslant 2^{-\lambda}$,
      $\Pr[x_i = \alpha | P_{\alpha,\beta,\gamma}(x_1) = \cdots = P_{\alpha,\beta,\gamma}(x_{i-1}) = \perp] \leqslant 2^{-\lambda}$.
    - $\Pr[x_1(\alpha) = \beta] \leqslant 2^{-\lambda}$,
      $\Pr[x_i(\alpha) = \beta | P_{\alpha,\beta,\gamma}(x_1) = \cdots = P_{\alpha,\beta,\gamma}(x_{i-1}) = \perp] \leqslant 2^{-\lambda}$.
    - Union bound over all queries $x_1, \ldots, x_q$ gives the stated probability.

VBB Obfuscation is Impossible

VBB Obfuscation is Impossible

- **Observation 2: easy to extract γ from any obfuscated $P_{\alpha,\beta,\gamma}$.**

VBB Obfuscation is Impossible

- **Observation 2: easy to extract γ from any obfuscated $P_{\alpha,\beta,\gamma}$.**
  - This is a *diagonalization* trick.

VBB Obfuscation is Impossible

- **Observation 2: easy to extract γ from any obfuscated $P_{\alpha,\beta,\gamma}$.**
  - This is a *diagonalization* trick.
  - Let $P \leftarrow \mathsf{Obf}(1^\lambda, P_{\alpha,\beta,\gamma})$.

VBB Obfuscation is Impossible

- **Observation 2: easy to extract γ from any obfuscated $P_{\alpha,\beta,\gamma}$.**
  - This is a *diagonalization* trick.
  - Let $P \leftarrow \mathsf{Obf}(1^\lambda, P_{\alpha,\beta,\gamma})$.
  - What happens if we run $P(P)$?

VBB Obfuscation is Impossible

- **Observation 2: easy to extract γ from any obfuscated $P_{\alpha,\beta,\gamma}$.**
  - This is a *diagonalization* trick.
  - Let $P \leftarrow \mathsf{Obf}(1^\lambda, P_{\alpha,\beta,\gamma})$.
  - What happens if we run $P(P)$?

    $P(P) = P_{\alpha,\beta,\gamma}(P)$ Correctness of Obf

VBB Obfuscation is Impossible

- **Observation 2: easy to extract γ from any obfuscated $P_{\alpha,\beta,\gamma}$.**
  - This is a *diagonalization* trick.
  - Let $P \leftarrow \mathsf{Obf}(1^\lambda, P_{\alpha,\beta,\gamma})$.
  - What happens if we run $P(P)$?

$$\begin{aligned} P(P) &= P_{\alpha,\beta,\gamma}(P) && \text{Correctness of } \mathsf{Obf} \\ &= \gamma && \text{Since } P(\alpha) = P_{\alpha,\beta,\gamma}(\alpha) \text{ (Correctness of } \mathsf{Obf}), \end{aligned}$$

VBB Obfuscation is Impossible

- **Observation 2: easy to extract γ from any obfuscated $P_{\alpha,\beta,\gamma}$.**
  - This is a *diagonalization* trick.
  - Let $P \leftarrow \mathsf{Obf}(1^\lambda, P_{\alpha,\beta,\gamma})$.
  - What happens if we run $P(P)$?

$$\begin{aligned} P(P) &= P_{\alpha,\beta,\gamma}(P) && \text{Correctness of } \mathsf{Obf} \\ &= \gamma && \text{Since } P(\alpha) = P_{\alpha,\beta,\gamma}(\alpha) \text{ (Correctness of } \mathsf{Obf}\text{),} \\ & && \quad \text{and } P_{\alpha,\beta,\gamma}(\alpha) = \beta \text{ by definition.} \end{aligned}$$

VBB Obfuscation is Impossible

VBB Obfuscation is Impossible

- The above counter-example works for *programs*, but not for *circuits*.

VBB Obfuscation is Impossible

- The above counter-example works for *programs*, but not for *circuits*.
  - Intuition/challenge: cannot give a circuit itself as input.

VBB Obfuscation is Impossible

- The above counter-example works for *programs*, but not for *circuits*.
  - Intuition/challenge: cannot give a circuit itself as input.

- Assuming FHE exists, the impossibility result still holds for circuits.

VBB Obfuscation is Impossible

- The above counter-example works for *programs*, but not for *circuits*.
  - Intuition/challenge: cannot give a circuit itself as input.
- Assuming FHE exists, the impossibility result still holds for circuits.
- Let $\alpha, \beta, \gamma \xleftarrow{\$} \{0,1\}^\lambda$.

VBB Obfuscation is Impossible

- The above counter-example works for *programs*, but not for *circuits*.
  - Intuition/challenge: cannot give a circuit itself as input.
- Assuming FHE exists, the impossibility result still holds for circuits.
- Let $\alpha, \beta, \gamma \xleftarrow{\$} \{0,1\}^\lambda$. Define circuit $C_{\alpha,\beta,\gamma}$ as follows.

VBB Obfuscation is Impossible

- The above counter-example works for *programs*, but not for *circuits*.
  - Intuition/challenge: cannot give a circuit itself as input.

- Assuming FHE exists, the impossibility result still holds for circuits.

- Let $\alpha, \beta, \gamma \xleftarrow{\$} \{0,1\}^\lambda$. Define circuit $C_{\alpha,\beta,\gamma}$ as follows.

$$C_{\alpha,\beta,\gamma}(x) = \begin{cases} \mathrm{Enc}_{\mathrm{pk}}(\alpha) & x = 0^\lambda \\ \beta & x = \alpha \\ \gamma & \mathrm{Dec}_{\mathrm{sk}}(x) = \beta \\ \bot & \text{otherwise} \end{cases}.$$

VBB Obfuscation is Impossible

- The above counter-example works for *programs*, but not for *circuits*.
  - Intuition/challenge: cannot give a circuit itself as input.

- Assuming FHE exists, the impossibility result still holds for circuits.

- Let $\alpha, \beta, \gamma \xleftarrow{\$} \{0,1\}^\lambda$. Define circuit $C_{\alpha,\beta,\gamma}$ as follows.

$$C_{\alpha,\beta,\gamma}(x) = \begin{cases} \mathrm{Enc}_{\mathrm{pk}}(\alpha) & x = 0 \\ \beta & x = \alpha \\ \gamma & \mathrm{Dec}_{\mathrm{sk}}(x) = \beta \\ \bot & \text{otherwise} \end{cases}.$$

- **Observation 1:** Oracle access to $C_{\alpha,\beta,\gamma}$ still hides α, β, γ by security of FHE.

VBB Obfuscation is Impossible

VBB Obfuscation is Impossible

- **Observation 2:** Can extract γ easily from any obfuscated circuit.

VBB Obfuscation is Impossible

- **Observation 2:** Can extract γ easily from any obfuscated circuit.
  - Let $C \leftarrow \mathsf{Obf}(1^\lambda, C_{\alpha,\beta,\gamma})$.

VBB Obfuscation is Impossible

- **Observation 2:** Can extract γ easily from any obfuscated circuit.
  - Let $C \leftarrow \mathsf{Obf}(1^\lambda, C_{\alpha,\beta,\gamma})$.
  - Let $a = C(0)$.

VBB Obfuscation is Impossible

- **Observation 2:** Can extract γ easily from any obfuscated circuit.
  - Let $C \leftarrow \mathsf{Obf}(1^\lambda, C_{\alpha,\beta,\gamma})$.
  - Let $a = C(0)$. Correctness of Obf implies $a = \mathrm{Enc}_{\mathrm{pk}}(\alpha)$.

VBB Obfuscation is Impossible

- **Observation 2:** Can extract γ easily from any obfuscated circuit.
  - Let $C \leftarrow \mathsf{Obf}(1^\lambda, C_{\alpha,\beta,\gamma})$.
  - Let $a = C(0)$. Correctness of Obf implies $a = \mathrm{Enc}_{\mathrm{pk}}(\alpha)$.
  - Use FHE to evaluate C on α *homomorphically*.

VBB Obfuscation is Impossible

- **Observation 2:** Can extract γ easily from any obfuscated circuit.
  - Let $C \leftarrow \mathsf{Obf}(1^\lambda, C_{\alpha,\beta,\gamma})$.
  - Let $a = C(0)$. Correctness of Obf implies $a = \mathrm{Enc}_{\mathrm{pk}}(\alpha)$.
  - Use FHE to evaluate C on α *homomorphically*.
    - I.e., obtain $B = \mathsf{FHE.Eval}(\mathrm{Enc}_{\mathrm{pk}}(\alpha), C)$.

VBB Obfuscation is Impossible

- **Observation 2:** Can extract γ easily from any obfuscated circuit.
  - Let $C \leftarrow \mathsf{Obf}(1^\lambda, C_{\alpha,\beta,\gamma})$.
  - Let $a = C(0)$. Correctness of Obf implies $a = \mathrm{Enc}_{\mathrm{pk}}(\alpha)$.
  - Use FHE to evaluate C on α *homomorphically*.
    - I.e., obtain $B = \mathsf{FHE.Eval}(\mathrm{Enc}_{\mathrm{pk}}(\alpha), C)$.
    - Correctness of FHE: $B = \mathrm{Enc}_{\mathrm{pk}}(C(\alpha))$.

VBB Obfuscation is Impossible

- **Observation 2:** Can extract γ easily from any obfuscated circuit.
  - Let $C \leftarrow \mathsf{Obf}(1^\lambda, C_{\alpha,\beta,\gamma})$.
  - Let $a = C(0)$. Correctness of Obf implies $a = \mathrm{Enc}_{\mathrm{pk}}(\alpha)$.
  - Use FHE to evaluate C on α *homomorphically*.
    - I.e., obtain $B = \mathsf{FHE.Eval}(\mathrm{Enc}_{\mathrm{pk}}(\alpha), C)$.
    - Correctness of FHE: $B = \mathrm{Enc}_{\mathrm{pk}}(C(\alpha))$.
    - Correctness of Obf: $C(\alpha) = \beta$.

VBB Obfuscation is Impossible

- **Observation 2:** Can extract γ easily from any obfuscated circuit.
  - Let $C \leftarrow \mathsf{Obf}(1^\lambda, C_{\alpha,\beta,\gamma})$.
  - Let $a = C(0)$. Correctness of Obf implies $a = \mathrm{Enc}_{\mathrm{pk}}(\alpha)$.
  - Use FHE to evaluate C on α *homomorphically*.
    - I.e., obtain $B = \mathsf{FHE.Eval}(\mathrm{Enc}_{\mathrm{pk}}(\alpha), C)$.
    - Correctness of FHE: $B = \mathrm{Enc}_{\mathrm{pk}}(C(\alpha))$.
    - Correctness of Obf: $C(\alpha) = \beta$.
  - Run $d = C(\mathrm{Enc}_{pk}(\beta))$.

VBB Obfuscation is Impossible

- **Observation 2:** Can extract γ easily from any obfuscated circuit.
  - Let $C \leftarrow \mathsf{Obf}(1^\lambda, C_{\alpha,\beta,\gamma})$.
  - Let $a = C(0)$. Correctness of Obf implies $a = \mathrm{Enc}_{\mathrm{pk}}(\alpha)$.
  - Use FHE to evaluate C on α *homomorphically*.
    - I.e., obtain $B = \mathsf{FHE.Eval}(\mathrm{Enc}_{\mathrm{pk}}(\alpha), C)$.
    - Correctness of FHE: $B = \mathrm{Enc}_{\mathrm{pk}}(C(\alpha))$.
    - Correctness of Obf: $C(\alpha) = \beta$.
  - Run $d = C(\mathrm{Enc}_{pk}(\beta))$. Correctness of Obf implies that $d = \gamma$.

Indistinguishability Obfuscation

Indistinguishability Obfuscation

- Since VBB Obfuscation is impossible in the general case, we opt for a weaker indistinguishability definition.

Indistinguishability Obfuscation

- Since VBB Obfuscation is impossible in the general case, we opt for a weaker indistinguishability definition.
- First, we need to define when two circuits C, C' are *equivalent*.

Indistinguishability Obfuscation

- Since VBB Obfuscation is impossible in the general case, we opt for a weaker indistinguishability definition.
- First, we need to define when two circuits C, C' are *equivalent*.

Definition 2

Two (Boolean) circuits C, C' are *equivalent* if

Indistinguishability Obfuscation

- Since VBB Obfuscation is impossible in the general case, we opt for a weaker indistinguishability definition.
- First, we need to define when two circuits C, C' are *equivalent*.

Definition 2

Two (Boolean) circuits C, C' are *equivalent* if

1. $|C| = |C'|$, and

Indistinguishability Obfuscation

- Since VBB Obfuscation is impossible in the general case, we opt for a weaker indistinguishability definition.
- First, we need to define when two circuits C, C' are *equivalent*.

Definition 2

Two (Boolean) circuits C, C' are *equivalent* if

1. $|C| = |C'|$, and
2. For all inputs x, $C(x) = C'(x)$.

Indistinguishability Obfuscation

- Since VBB Obfuscation is impossible in the general case, we opt for a weaker indistinguishability definition.
- First, we need to define when two circuits C, C' are *equivalent*.

Definition 2

Two (Boolean) circuits C, C' are *equivalent* if

1. $|C| = |C'|$, and
2. For all inputs x, $C(x) = C'(x)$.

Definition 3

Indistinguishability Obfuscation

Indistinguishability Obfuscation

- Since VBB Obfuscation is impossible in the general case, we opt for a weaker indistinguishability definition.
- First, we need to define when two circuits C, C' are *equivalent*.

Definition 2

Two (Boolean) circuits C, C' are *equivalent* if

1. $|C| = |C'|$, and
2. For all inputs x, $C(x) = C'(x)$.

Definition 3

Indistinguishability Obfuscation Let $i\mathcal{O}$ be an obfuscator.

Indistinguishability Obfuscation

- Since VBB Obfuscation is impossible in the general case, we opt for a weaker indistinguishability definition.
- First, we need to define when two circuits C, C' are *equivalent*.

Definition 2

Two (Boolean) circuits C, C' are *equivalent* if

1. $|C| = |C'|$, and
2. For all inputs x, $C(x) = C'(x)$.

Definition 3

Indistinguishability Obfuscation Let $i\mathcal{O}$ be an obfuscator. Then, we say that $i\mathcal{O}$ is an *indistinguishability obfuscator* (or just *secure*) if for all pairs of equivalent circuits C, C', all PPT $\mathcal{A}$, and all $\lambda \in \mathbb{N}$, we have

Indistinguishability Obfuscation

- Since VBB Obfuscation is impossible in the general case, we opt for a weaker indistinguishability definition.
- First, we need to define when two circuits C, C' are *equivalent*.

Definition 2

Two (Boolean) circuits C, C' are *equivalent* if

1. $|C| = |C'|$, and
2. For all inputs x, $C(x) = C'(x)$.

Definition 3

Indistinguishability Obfuscation Let $i\mathcal{O}$ be an obfuscator. Then, we say that $i\mathcal{O}$ is an *indistinguishability obfuscator* (or just *secure*) if for all pairs of equivalent circuits C, C', all PPT $\mathcal{A}$, and all $\lambda \in \mathbb{N}$, we have

$$\left|\Pr\left[\mathcal{A}(i\mathcal{O}(1^\lambda, C)) = 1\right] - \Pr\left[\mathcal{A}(i\mathcal{O}(1^\lambda, C')) = 1\right]\right| \leqslant \mathsf{negl}(\lambda).$$

Best Possible $i\mathcal{O}$

Best Possible $i\mathcal{O}$

- Pros of $i\mathcal{O}$: constructions exist, primitive is very useful!

Best Possible $i\mathcal{O}$

- Pros of $i\mathcal{O}$: constructions exist, primitive is very useful!
- Cons of $i\mathcal{O}$: weaker security definition, current constructions are not practical.

Best Possible $i\mathcal{O}$

- Pros of $i\mathcal{O}$: constructions exist, primitive is very useful!
- Cons of $i\mathcal{O}$: weaker security definition, current constructions are not practical.

- "On Best Possible Obfuscation" [TCC'07]

Best Possible $i\mathcal{O}$

- Pros of $i\mathcal{O}$: constructions exist, primitive is very useful!
- Cons of $i\mathcal{O}$: weaker security definition, current constructions are not practical.

- "On Best Possible Obfuscation" [TCC'07]
  - Let $i\mathcal{O}$ be a secure obfuscator.

Best Possible $i\mathcal{O}$

- Pros of $i\mathcal{O}$: constructions exist, primitive is very useful!
- Cons of $i\mathcal{O}$: weaker security definition, current constructions are not practical.

- "On Best Possible Obfuscation" [TCC'07]
  - Let $i\mathcal{O}$ be a secure obfuscator.
  - Let Obf be a different obfuscator satisfying some other security definition.

Best Possible $i\mathcal{O}$

- Pros of $i\mathcal{O}$: constructions exist, primitive is very useful!
- Cons of $i\mathcal{O}$: weaker security definition, current constructions are not practical.

- "On Best Possible Obfuscation" [TCC'07]
  - Let $i\mathcal{O}$ be a secure obfuscator.
  - Let Obf be a different obfuscator satisfying some other security definition.
  - The following properties hold.

Best Possible $i\mathcal{O}$

- Pros of $i\mathcal{O}$: constructions exist, primitive is very useful!
- Cons of $i\mathcal{O}$: weaker security definition, current constructions are not practical.

- "On Best Possible Obfuscation" [TCC'07]
  - Let $i\mathcal{O}$ be a secure obfuscator.
  - Let Obf be a different obfuscator satisfying some other security definition.
  - The following properties hold.
    1. $\mathsf{Obf}'(C) := i\mathcal{O}(\mathsf{Obf}(C))$ cannot be *weaker* than Obf.

Best Possible $i\mathcal{O}$

- Pros of $i\mathcal{O}$: constructions exist, primitive is very useful!
- Cons of $i\mathcal{O}$: weaker security definition, current constructions are not practical.

- "On Best Possible Obfuscation" [TCC'07]
  - Let $i\mathcal{O}$ be a secure obfuscator.
  - Let Obf be a different obfuscator satisfying some other security definition.
  - The following properties hold.
    1. $\mathsf{Obf}'(C) := i\mathcal{O}(\mathsf{Obf}(C))$ cannot be *weaker* than Obf.
    2. $C' = \mathsf{Obf}(C)$ is functionally equivalent to C.

Best Possible $i\mathcal{O}$

- Pros of $i\mathcal{O}$: constructions exist, primitive is very useful!
- Cons of $i\mathcal{O}$: weaker security definition, current constructions are not practical.

- "On Best Possible Obfuscation" [TCC'07]
  - Let $i\mathcal{O}$ be a secure obfuscator.
  - Let Obf be a different obfuscator satisfying some other security definition.
  - The following properties hold.
    1. $\mathsf{Obf}'(C) := i\mathcal{O}(\mathsf{Obf}(C))$ cannot be *weaker* than Obf.
    2. $C' = \mathsf{Obf}(C)$ is functionally equivalent to C.
    3. C' is equivalent to $\mathsf{Pad}(C)$.

Obf' has same security as Obf

$|C'| > |C|$

(1) $\mathsf{Pad}(C)(x) = C(x)\ \forall x$

(2) $|\mathsf{Pad}(C)| = |C'|$

Best Possible $i\mathcal{O}$

- Pros of $i\mathcal{O}$: constructions exist, primitive is very useful!
- Cons of $i\mathcal{O}$: weaker security definition, current constructions are not practical.

- "On Best Possible Obfuscation" [TCC'07]
  - Let $i\mathcal{O}$ be a secure obfuscator.
  - Let Obf be a different obfuscator satisfying some other security definition.
  - The following properties hold.
    1. $\mathsf{Obf}'(C) := i\mathcal{O}(\mathsf{Obf}(C))$ cannot be *weaker* than Obf.
    2. $C' = \mathsf{Obf}(C)$ is functionally equivalent to C.
    3. C' is equivalent to $\mathsf{Pad}(C)$.
    4. $\mathsf{Obf}'(C)$ is indistinguishable from $i\mathcal{O}(\mathsf{Pad}(C))$ (by $i\mathcal{O}$ security).

Results on $i\mathcal{O}$

Results on $i\mathcal{O}$

- Unlike other crypto primitives, $i\mathcal{O}$ does not imply that $\mathbf{P} \neq \mathbf{NP}$.

Results on $i\mathcal{O}$

- Unlike other crypto primitives, $i\mathcal{O}$ does not imply that $\mathbf{P} \neq \mathbf{NP}$.

Lemma 1

If $\mathbf{P} = \mathbf{NP}$, *then* $i\mathcal{O}$ *exists.*

$coNP = \{L \mid \bar{L} \in NP\}$

$P \subseteq NP \cap coNP$

$P = NP \Rightarrow NP = coNP$

Results on $i\mathcal{O}$

- Unlike other crypto primitives, $i\mathcal{O}$ does not imply that $\mathbf{P} \neq \mathbf{NP}$.

Lemma 1

If $\mathbf{P} = \mathbf{NP}$, *then* $i\mathcal{O}$ *exists.*

Proof.

□

Results on $i\mathcal{O}$

- Unlike other crypto primitives, $i\mathcal{O}$ does not imply that $\mathbf{P} \neq \mathbf{NP}$.

Lemma 1

If $\mathbf{P} = \mathbf{NP}$, *then* $i\mathcal{O}$ *exists.*

Proof.

Given any circuit C, the obfuscator $i\mathcal{O}$ simply outputs $\widetilde{C}$ which is the smallest circuit that is equivalent to C (if there are multiple, pick the lexicographically first one). functionally □

Results on $i\mathcal{O}$

- Unlike other crypto primitives, $i\mathcal{O}$ does not imply that $\mathbf{P} \neq \mathbf{NP}$.

Lemma 1

If $\mathbf{P} = \mathbf{NP}$, *then* $i\mathcal{O}$ *exists.*

Proof.

Given any circuit C, the obfuscator $i\mathcal{O}$ simply outputs $\widetilde{C}$ which is the smallest circuit that is equivalent to C (if there are multiple, pick the lexicographically first one). Then for all circuits C' equivalent to C (same size and functionally equivalent), $i\mathcal{O}$ outputs the same $\widetilde{C}$. □

Results on $i\mathcal{O}$

- Unlike other crypto primitives, $i\mathcal{O}$ does not imply that $\mathbf{P} \neq \mathbf{NP}$.

Lemma 1

If $\mathbf{P} = \mathbf{NP}$, *then* $i\mathcal{O}$ *exists.*

Proof.

Given any circuit C, the obfuscator $i\mathcal{O}$ simply outputs $\widetilde{C}$ which is the smallest circuit that is equivalent to C (if there are multiple, pick the lexicographically first one). Then for all circuits C' equivalent to C (same size and functionally equivalent), $i\mathcal{O}$ outputs the same $\widetilde{C}$. □

Corollary 1

The existence of $i\mathcal{O}$ *does not imply the existence of one-way functions.*

Results on $i\mathcal{O}$

- Unlike other crypto primitives, $i\mathcal{O}$ does not imply that $\mathbf{P} \neq \mathbf{NP}$.

Lemma 1

If $\mathbf{P} = \mathbf{NP}$, *then* $i\mathcal{O}$ *exists.*

Proof.

Given any circuit C, the obfuscator $i\mathcal{O}$ simply outputs $\widetilde{C}$ which is the smallest circuit that is equivalent to C (if there are multiple, pick the lexicographically first one). Then for all circuits C' equivalent to C (same size and functionally equivalent), $i\mathcal{O}$ outputs the same $\widetilde{C}$. □

Corollary 1

The existence of $i\mathcal{O}$ *does not imply the existence of one-way functions.*

- Interestingly, combining $i\mathcal{O}$ and OWFs will yield powerful crypto primitives.

Applications of $i\mathcal{O}$

Digital Signatures from $i\mathcal{O}$

Digital Signatures from $i\mathcal{O}$

- We can easily construct *digital signatures* from $i\mathcal{O}$ and PPRFs.

Digital Signatures from $i\mathcal{O}$

- We can easily construct *digital signatures* from $i\mathcal{O}$ and PPRFs.

Definition 4 (Digital Signatures)

A *digital signature scheme* consists of three PPT algorithms $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ such that the following hold.

Digital Signatures from $i\mathcal{O}$

- We can easily construct *digital signatures* from $i\mathcal{O}$ and PPRFs.

Definition 4 (Digital Signatures)

A *digital signature scheme* consists of three PPT algorithms $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ such that the following hold.

- The *key generation* algorithm Gen takes as input security parameter 1^λ and outputs a pair of keys (sk, vk) each of length at least λ, the private *signing key* and public *verification key*.

Digital Signatures from $i\mathcal{O}$

- We can easily construct *digital signatures* from $i\mathcal{O}$ and PPRFs.

Definition 4 (Digital Signatures)

A *digital signature scheme* consists of three PPT algorithms $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ such that the following hold.

- The *key generation* algorithm Gen takes as input security parameter 1^λ and outputs a pair of keys (sk, vk) each of length at least λ, the private *signing key* and public *verification key*.
- The *signing* algorithm Sign takes as input sk and message m and outputs a signature $\sigma \leftarrow \mathsf{Sign}_{sk}(m)$.

Digital Signatures from $i\mathcal{O}$

- We can easily construct *digital signatures* from $i\mathcal{O}$ and PPRFs.

Definition 4 (Digital Signatures)

A *digital signature scheme* consists of three PPT algorithms $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ such that the following hold.

- The *key generation* algorithm Gen takes as input security parameter 1^λ and outputs a pair of keys (sk, vk) each of length at least λ, the private *signing key* and public *verification key*.
- The *signing* algorithm Sign takes as input sk and message m and outputs a signature $\sigma \leftarrow \mathsf{Sign}_{sk}(m)$.
- The *verification* algorithm Vrfy is deterministic and takes as input vk, message m, and signature σ and outputs a bit $b = \mathsf{Vrfy}_{vk}(m, \sigma)$.

Digital Signatures from $i\mathcal{O}$

- We can easily construct *digital signatures* from $i\mathcal{O}$ and PPRFs.

Definition 4 (Digital Signatures)

A *digital signature scheme* consists of three PPT algorithms $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ such that the following hold.

- The *key generation* algorithm Gen takes as input security parameter 1^λ and outputs a pair of keys (sk, vk) each of length at least λ, the private *signing key* and public *verification key*.
- The *signing* algorithm Sign takes as input sk and message m and outputs a signature $\sigma \leftarrow \mathsf{Sign}_{sk}(m)$.
- The *verification* algorithm Vrfy is deterministic and takes as input vk, message m, and signature σ and outputs a bit $b = \mathsf{Vrfy}_{vk}(m, \sigma)$. $b = 1$ denotes that σ is a *valid* signature for m, and $b = 0$ denotes that it is *invalid*.

Digital Signatures from $i\mathcal{O}$

Digital Signatures from $i\mathcal{O}$

Definition 5 (Digital Signatures: Correctness)

We say that a digital signature scheme $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ is *(perfectly) correct* if for all $\lambda \in \mathbb{N}$, $(sk, vk) \leftarrow \mathsf{Gen}(1^\lambda)$, and all legal messages m, we have $\Pr[\mathsf{Vrfy}_{vk}(m, \mathsf{Sign}_{sk}(m))] = 1$.

Digital Signatures from $i\mathcal{O}$

Definition 5 (Digital Signatures: Correctness)

We say that a digital signature scheme $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ is *(perfectly) correct* if for all $\lambda \in \mathbb{N}$, $(sk, vk) \leftarrow \mathsf{Gen}(1^\lambda)$, and all legal messages m, we have $\Pr[\mathsf{Vrfy}_{pk}(m, \mathsf{Sign}_{sk}(m))] = 1$.

Definition 6 (Digital Signatures: Selective Security)

We say that a digital signature scheme $\Pi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ is *selectively unforgeable* if for all PPT adversaries $\mathcal{A}$, there is a negligible function such that

Digital Signatures from $i\mathcal{O}$

Definition 5 (Digital Signatures: Correctness)

We say that a digital signature scheme $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ is *(perfectly) correct* if for all $\lambda \in \mathbb{N}$, $(sk, vk) \leftarrow \mathsf{Gen}(1^\lambda)$, and all legal messages m, we have $\Pr[\mathsf{Vrfy}_{pk}(m, \mathsf{Sign}_{sk}(m))] = 1$.

Definition 6 (Digital Signatures: Selective Security)

We say that a digital signature scheme $\Pi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ is *selectively unforgeable* if for all PPT adversaries $\mathcal{A}$, there is a negligible function such that

$$\Pr[\mathsf{Sel\text{-}forge}_{\mathcal{A},\Pi}(\lambda) = 1] \leqslant \mathsf{negl}(\lambda),$$

Digital Signatures from $i\mathcal{O}$

Definition 5 (Digital Signatures: Correctness)

We say that a digital signature scheme $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ is *(perfectly) correct* if for all $\lambda \in \mathbb{N}$, $(sk, vk) \leftarrow \mathsf{Gen}(1^\lambda)$, and all legal messages m, we have $\Pr[\mathsf{Vrfy}_{pk}(m, \mathsf{Sign}_{sk}(m))] = 1$.

Definition 6 (Digital Signatures: Selective Security)

We say that a digital signature scheme $\Pi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ is *selectively unforgeable* if for all PPT adversaries $\mathcal{A}$, there is a negligible function such that

$$\Pr[\mathsf{Sel\text{-}forge}_{\mathcal{A},\Pi}(\lambda) = 1] \leqslant \mathsf{negl}(\lambda),$$

where $\mathsf{Sel\text{-}forge}_{\mathcal{A},\Pi}$ is defined on the next slide.

Digital Signatures: Selective Forgery Game

Digital Signatures: Selective Forgery Game

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

Digital Signatures: Selective Forgery Game

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

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

Digital Signatures: Selective Forgery Game

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

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

Digital Signatures: Selective Forgery Game

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

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

Digital Signatures: Selective Forgery Game

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

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

Digital Signatures: Selective Forgery Game

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

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

Digital Signatures: Selective Forgery Game

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

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

- Note: stronger security definition than standard *existentially unforgeability (EUF) under an adaptive chosen-message attack.*

Digital Signatures: Selective Forgery Game

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

1. ~~$m^* \leftarrow \mathcal{A}(1^\lambda)$.~~
2. $(sk, vk) \leftarrow \mathsf{Gen}(1^\lambda)$.
3. $(m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sign}_{sk}(\cdot)}(1^\lambda, pk)$.
4. Output 1 if $\mathsf{Vrfy}_{pk}(m^*, \sigma^*) = 1$ 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.

- Note: stronger security definition than standard *existentially unforgeability (EUF) under an adaptive chosen-message attack.*
  - To obtain EUF security, modify the above game by removing (1) and having $\mathcal{A}$ output a pair (m^*, σ^*) in (3).

Digital Signatures: Selective Forgery Game

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

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

- Note: stronger security definition than standard *existentially unforgeability (EUF) under an adaptive chosen-message attack.*
  - To obtain EUF security, modify the above game by removing (1) and having $\mathcal{A}$ output a pair (m^*, σ^*) in (3).
- Selective security implies *universal unforgeability.*

Digital Signatures: Selective Forgery Game

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

1. ~~$m^* \leftarrow \mathcal{A}(1^\lambda)$.~~
2. $(sk, vk) \leftarrow \mathsf{Gen}(1^\lambda)$.
3. $\sigma^* \leftarrow \mathcal{A}^{\mathsf{Sign}_{sk}(\cdot)}(1^\lambda, pk)$.
4. Output 1 if $\mathsf{Vrfy}_{pk}(m^*, \sigma^*) = 1$ 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.

- Note: stronger security definition than standard *existentially unforgeability (EUF) under an adaptive chosen-message attack.*
  - To obtain EUF security, modify the above game by removing (1) and having $\mathcal{A}$ output a pair (m^*, σ^*) in (3).
- Selective security implies *universal unforgeability.*
  - Above game holds for *all messages* m^* (given as input to the game).

Digital Signatures: Selective Forgery Game

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

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

- Note: stronger security definition than standard *existentially unforgeability (EUF) under an adaptive chosen-message attack.*
  - To obtain EUF security, modify the above game by removing (1) and having $\mathcal{A}$ output a pair (m^*, σ^*) in (3).
- Selective security implies *universal unforgeability.*
  - Above game holds for *all messages* m^* (given as input to the game).
  - Take Union bound over all possible target messages in above game.

Digital Signatures: Selective Forgery Game

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

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

- Note: stronger security definition than standard *existentially unforgeability (EUF) under an adaptive chosen-message attack.*
  - To obtain EUF security, modify the above game by removing (1) and having $\mathcal{A}$ output a pair (m^*, σ^*) in (3).
- Selective security implies *universal unforgeability.*
  - Above game holds for *all messages* m^* (given as input to the game).
  - Take Union bound over all possible target messages in above game.
  - Requires sub-exponentially secure $i\mathcal{O}$ and PPRFs.

Digital Signatures from $i\mathcal{O}$: Construction

Digital Signatures from $i\mathcal{O}$: Construction

- We can construct a selectively secure signature scheme using $i\mathcal{O}$ and a PPRF.

Digital Signatures from $i\mathcal{O}$: Construction

- We can construct a selectively secure signature scheme using $i\mathcal{O}$ and a PPRF.
- Let $F = (\mathsf{Eval}, \mathsf{Punc})$ be a PPRF and let $i\mathcal{O}$ be an indistinguishability obfuscator.

Digital Signatures from $i\mathcal{O}$: Construction

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

Digital Signatures from $i\mathcal{O}$: Construction

- We can construct a selectively secure signature scheme using $i\mathcal{O}$ and a PPRF.
- 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.

Digital Signatures from $i\mathcal{O}$: Construction

- We can construct a selectively secure signature scheme using $i\mathcal{O}$ and a PPRF.
- 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.

Digital Signatures from $i\mathcal{O}$: Construction

- We can construct a selectively secure signature scheme using $i\mathcal{O}$ and a PPRF.
- 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 $sk = k$.

Digital Signatures from $i\mathcal{O}$: Construction

- We can construct a selectively secure signature scheme using $i\mathcal{O}$ and a PPRF.
- 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).

Digital Signatures from $i\mathcal{O}$: Construction

- We can construct a selectively secure signature scheme using $i\mathcal{O}$ and a PPRF.
- 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 $\sigma := \mathsf{Eval}_k(m)$.

Digital Signatures from $i\mathcal{O}$: Construction

- We can construct a selectively secure signature scheme using $i\mathcal{O}$ and a PPRF.
- 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 $b := vk(m, \sigma)$.

$C_k(m,\sigma) = 1 \longleftrightarrow \mathsf{Eval}_k(m) = \sigma$

Next Time: Security Proof, then Memory-hard Functions