Pseudorandomness

601.442/642 Modern Cryptography

3rd February 2026

Announcement

- Homework 2 is due on **5th Feb**.

Computational Indistinguishability

Computational Indistinguishability

Two probability ensembles $X = \{X_i\}_{i \in \mathbb{N}}$ and $Y = \{Y_i\}_{i \in \mathbb{N}}$ are computationally indistinguishable if

for all $\lambda \in \mathbb{N}$

Computational Indistinguishability

Computational Indistinguishability

Two probability ensembles $X = \{X_i\}_{i \in \mathbb{N}}$ and $Y = \{Y_i\}_{i \in \mathbb{N}}$ are computationally indistinguishable if for all $\lambda \in \mathbb{N}$

$$\left| \Pr_{x \leftarrow X_\lambda}\left[A(1^\lambda, x) = 1\right] - \Pr_{y \leftarrow Y_\lambda}\left[A(1^\lambda, y) = 1\right] \right|$$

Computational Indistinguishability

Computational Indistinguishability

Two probability ensembles $X = \{X_i\}_{i \in \mathbb{N}}$ and $Y = \{Y_i\}_{i \in \mathbb{N}}$ are computationally indistinguishable if every non-uniform PPT adversary A, for all $\lambda \in \mathbb{N}$

$$\left| \Pr_{x \leftarrow X_\lambda}\left[A(1^\lambda, x) = 1\right] - \Pr_{y \leftarrow Y_\lambda}\left[A(1^\lambda, y) = 1\right] \right|$$

Computational Indistinguishability

Computational Indistinguishability

Two probability ensembles $X = \{X_i\}_{i \in \mathbb{N}}$ and $Y = \{Y_i\}_{i \in \mathbb{N}}$ are computationally indistinguishable if every non-uniform PPT adversary A, for all $\lambda \in \mathbb{N}$

$$\left| \Pr_{x \leftarrow X_\lambda}\left[A(1^\lambda, x) = 1\right] - \Pr_{y \leftarrow Y_\lambda}\left[A(1^\lambda, y) = 1\right] \right|$$

Denotes string of λ ones.
Ensures A is polynomial in λ.

Computational Indistinguishability

Computational Indistinguishability

Two probability ensembles $X = \{X_i\}_{i \in \mathbb{N}}$ and $Y = \{Y_i\}_{i \in \mathbb{N}}$ are computationally indistinguishable if every non-uniform PPT adversary A, there exists a negligible function $\mu(\lambda)$ such that for all $\lambda \in \mathbb{N}$

$$\left| \Pr_{x \leftarrow X_\lambda}\left[A(1^\lambda, x) = 1\right] - \Pr_{y \leftarrow Y_\lambda}\left[A(1^\lambda, y) = 1\right] \right| \leq \nu(\lambda),$$

Denotes string of λ ones.

Ensures A is polynomial in λ.

Computational Indistinguishability

Computational Indistinguishability

Two probability ensembles $X = \{X_i\}_{i \in \mathbb{N}}$ and $Y = \{Y_i\}_{i \in \mathbb{N}}$ are computationally indistinguishable if every non-uniform PPT adversary A, there exists a negligible function $\mu(\lambda)$ such that for all $\lambda \in \mathbb{N}$

$$\left| \Pr_{x \leftarrow X_\lambda}\left[A(1^\lambda, x) = 1\right] - \Pr_{y \leftarrow Y_\lambda}\left[A(1^\lambda, y) = 1\right] \right| \leq \nu(\lambda),$$

where the probability is over sampling from the distributions X_λ and Y_λ, and the randomness of A.

Computational Indistinguishability

Computational Indistinguishability

Two probability ensembles $X = \{X_i\}_{i \in \mathbb{N}}$ and $Y = \{Y_i\}_{i \in \mathbb{N}}$ are computationally indistinguishable if every non-uniform PPT adversary A, there exists a negligible function $\mu(\lambda)$ such that for all $\lambda \in \mathbb{N}$

$$\left| \Pr_{x \leftarrow X_\lambda}\left[A(1^\lambda, x) = 1\right] - \Pr_{y \leftarrow Y_\lambda}\left[A(1^\lambda, y) = 1\right] \right| \leq \nu(\lambda),$$

where the probability is over sampling from the distributions X_λ and Y_λ, and the randomness of A.

No efficient test can distinguish between the ensembles X and Y.

Computational Indistinguishability

Computational Indistinguishability

Two probability ensembles $X = \{X_i\}_{i \in \mathbb{N}}$ and $Y = \{Y_i\}_{i \in \mathbb{N}}$ are computationally indistinguishable if every non-uniform PPT adversary A, there exists a negligible function $\mu(\lambda)$ such that for all $\lambda \in \mathbb{N}$

$$\left| \Pr_{x \leftarrow X_\lambda}\left[A(1^\lambda, x) = 1\right] - \Pr_{y \leftarrow Y_\lambda}\left[A(1^\lambda, y) = 1\right] \right| \leq \nu(\lambda),$$

where the probability is over sampling from the distributions X_λ and Y_λ, and the randomness of A.

- We use $X \overset{c}{\approx} Y$ as a shorthand to denote that the two ensembles are computationally indistinguishable.

Computational Indistinguishability

Computational Indistinguishability

Two probability ensembles $X = \{X_i\}_{i \in \mathbb{N}}$ and $Y = \{Y_i\}_{i \in \mathbb{N}}$ are computationally indistinguishable if every non-uniform PPT adversary A, there exists a negligible function $\mu(\lambda)$ such that for all $\lambda \in \mathbb{N}$

$$\left| \Pr_{x \leftarrow X_\lambda} \left[A(1^\lambda, x) = 1 \right] - \Pr_{y \leftarrow Y_\lambda} \left[A(1^\lambda, y) = 1 \right] \right| \leq \nu(\lambda),$$

where the probability is over sampling from the distributions X_λ and Y_λ, and the randomness of A.

- We use $X \overset{c}{\approx} Y$ as a shorthand to denote that the two ensembles are computationally indistinguishable.
- The value

$$\left| \Pr_{x \leftarrow X_\lambda} \left[A(1^\lambda, x) = 1 \right] - \Pr_{y \leftarrow Y_\lambda} \left[A(1^\lambda, y) = 1 \right] \right|$$

is called the adversary's **advantage** in distinguishing between X and Y.

Computational Indistinguishability

Computational Indistinguishability

Two probability ensembles $X = \{X_i\}_{i\in\mathbb{N}}$ and $Y = \{Y_i\}_{i\in\mathbb{N}}$ are computationally indistinguishable if every non-uniform PPT adversary A, there exists a negligible function $\mu(\lambda)$ such that for all $\lambda \in \mathbb{N}$

$$\left| \Pr_{x\leftarrow X_\lambda}\left[A(1^\lambda, x) = 1\right] - \Pr_{y\leftarrow Y_\lambda}\left[A(1^\lambda, y) = 1\right] \right| \leq \nu(\lambda),$$

where the probability is over sampling from the distributions X_λ and Y_λ, and the randomness of A.

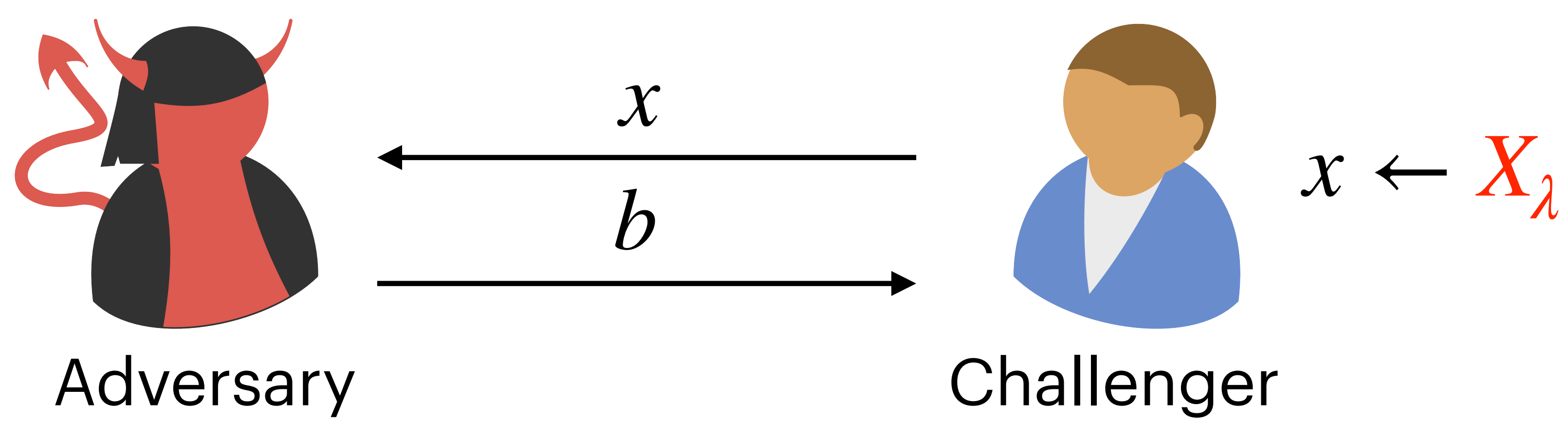
- We use $X \overset{c}{\approx} Y$ as a shorthand to denote that the two ensembles are computationally indistinguishable.
- The value

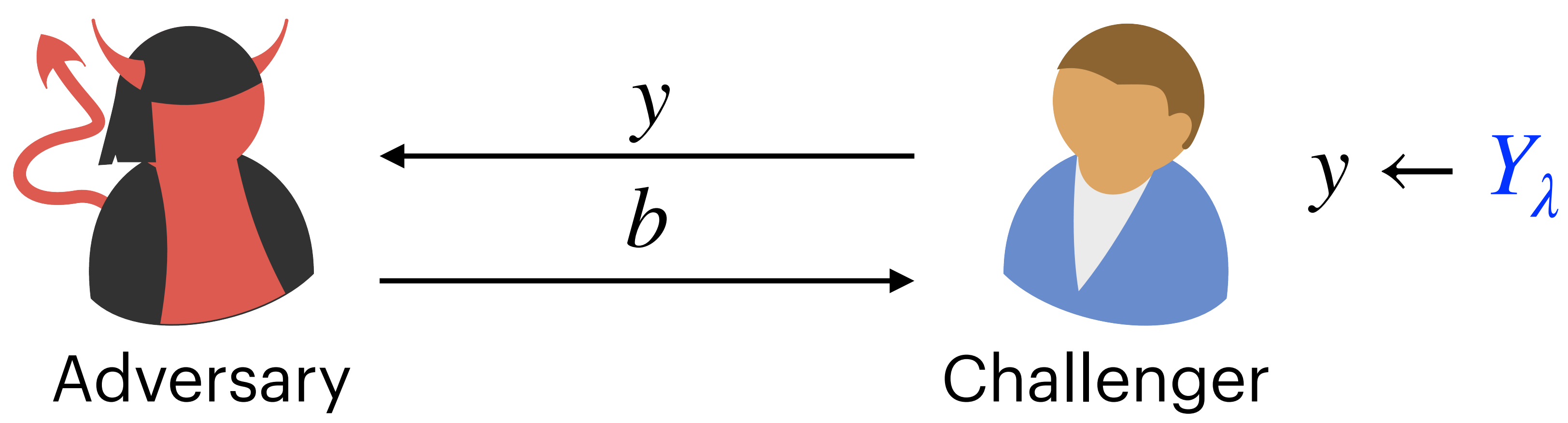
$$\left| \Pr_{x\leftarrow X_\lambda}\left[A(1^\lambda, x) = 1\right] - \Pr_{y\leftarrow Y_\lambda}\left[A(1^\lambda, y) = 1\right] \right|$$

  is called the adversary's **advantage** in distinguishing between X and Y.

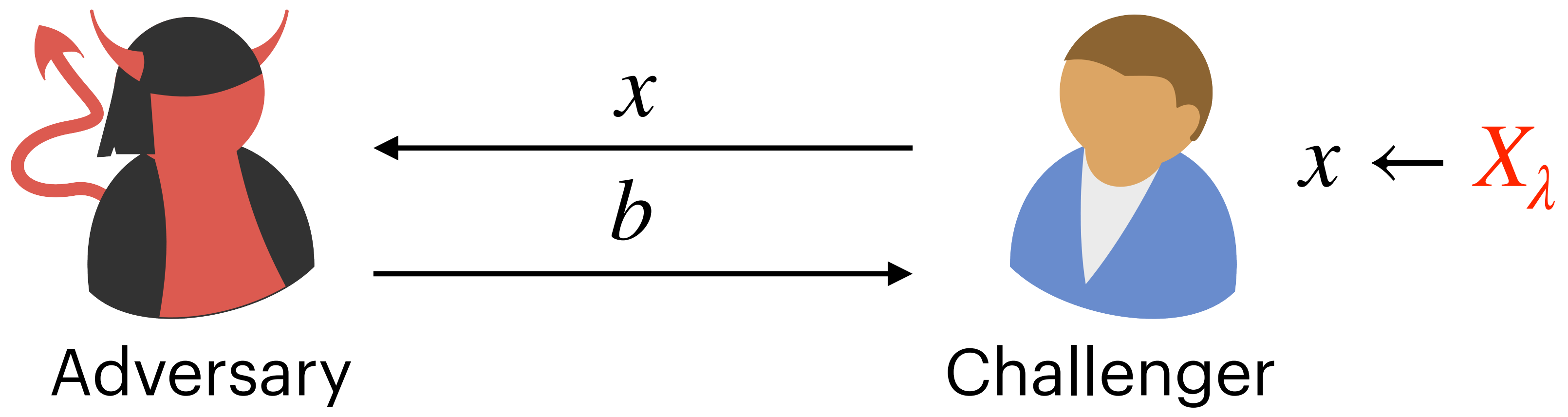
- $X \overset{c}{\approx} Y$ if all non-uniform PPT adversaries have negligible advantage in distinguishing between the two ensembles.

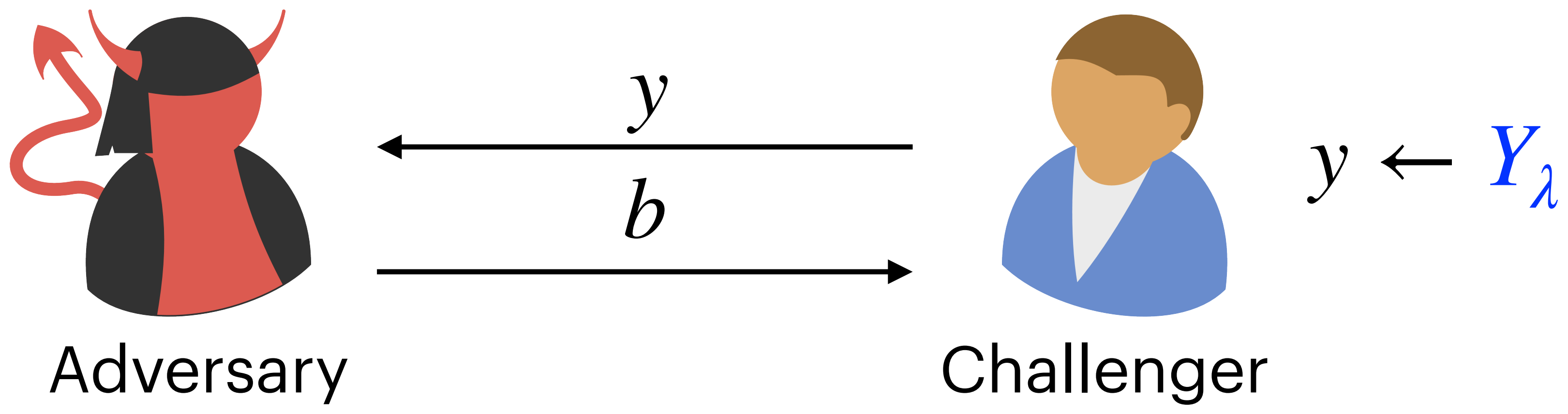
Computational Indistinguishability

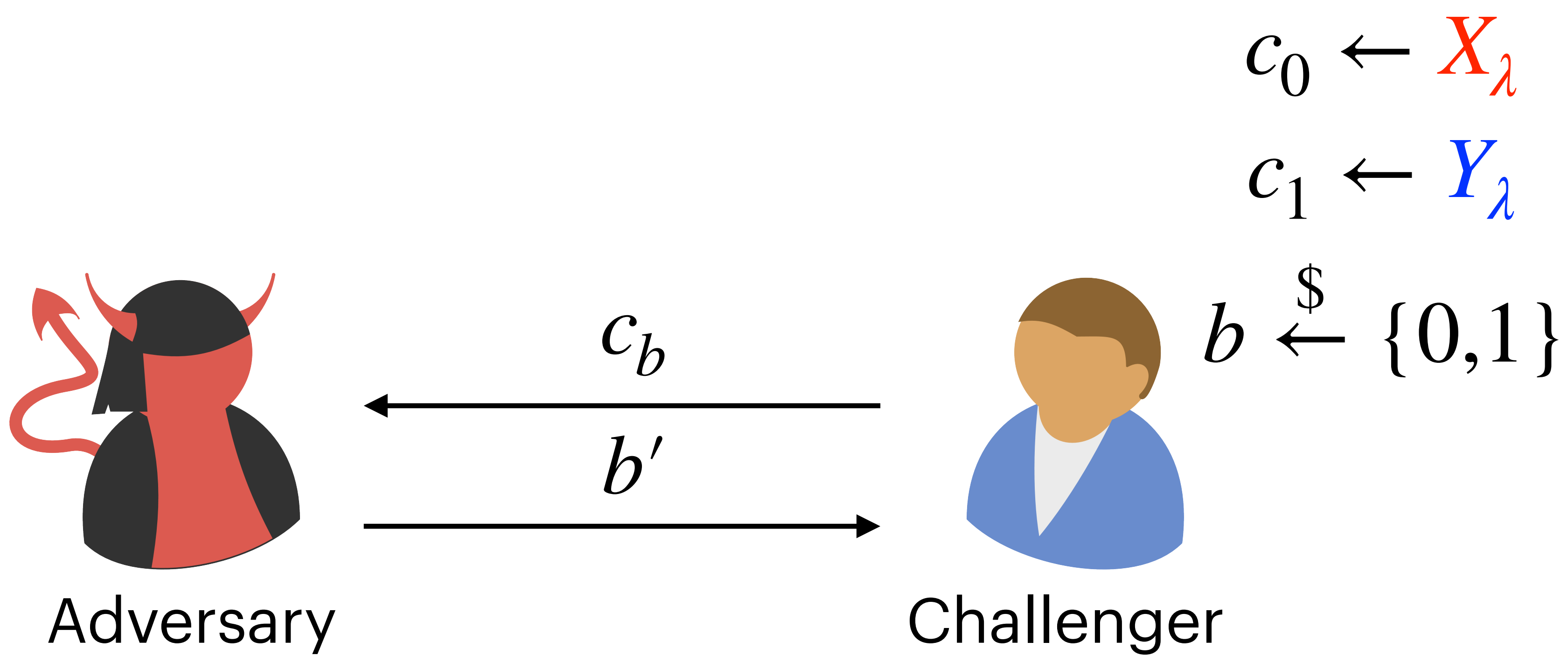




Computational Indistinguishability

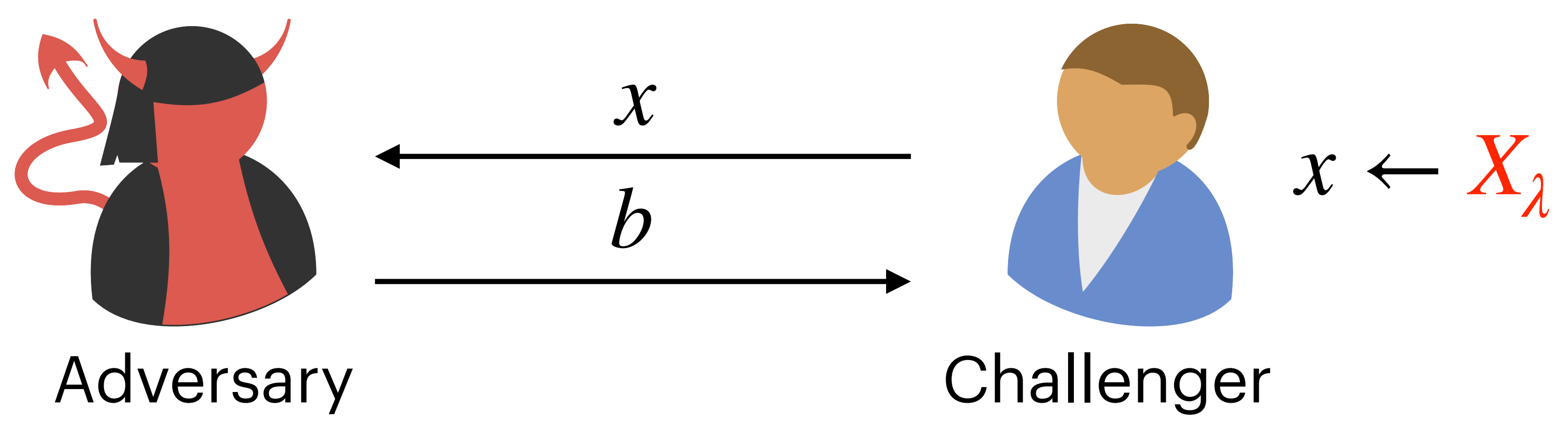


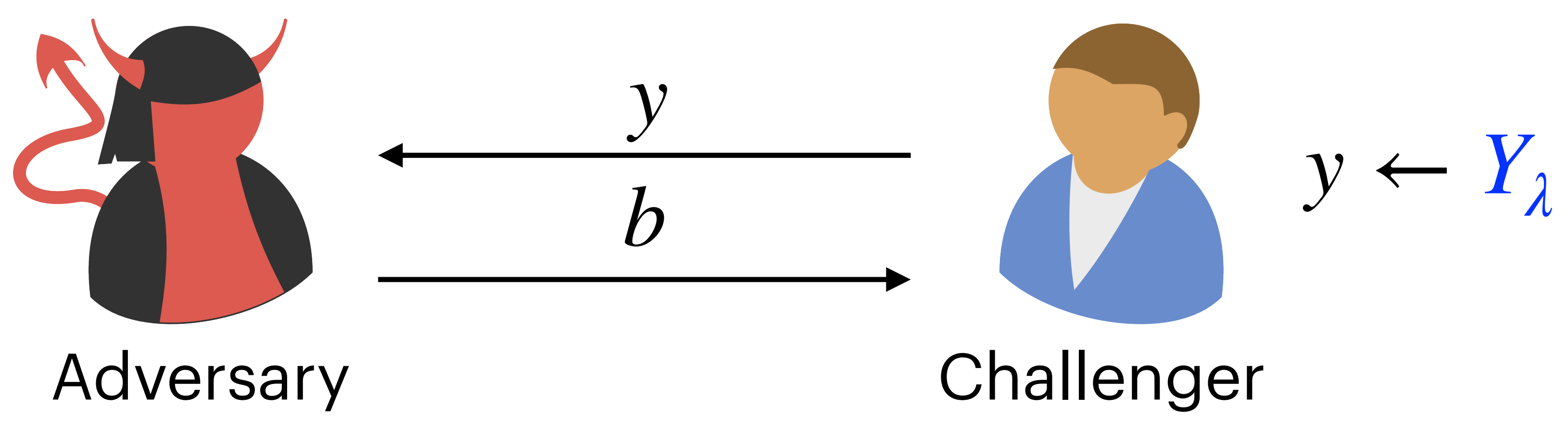


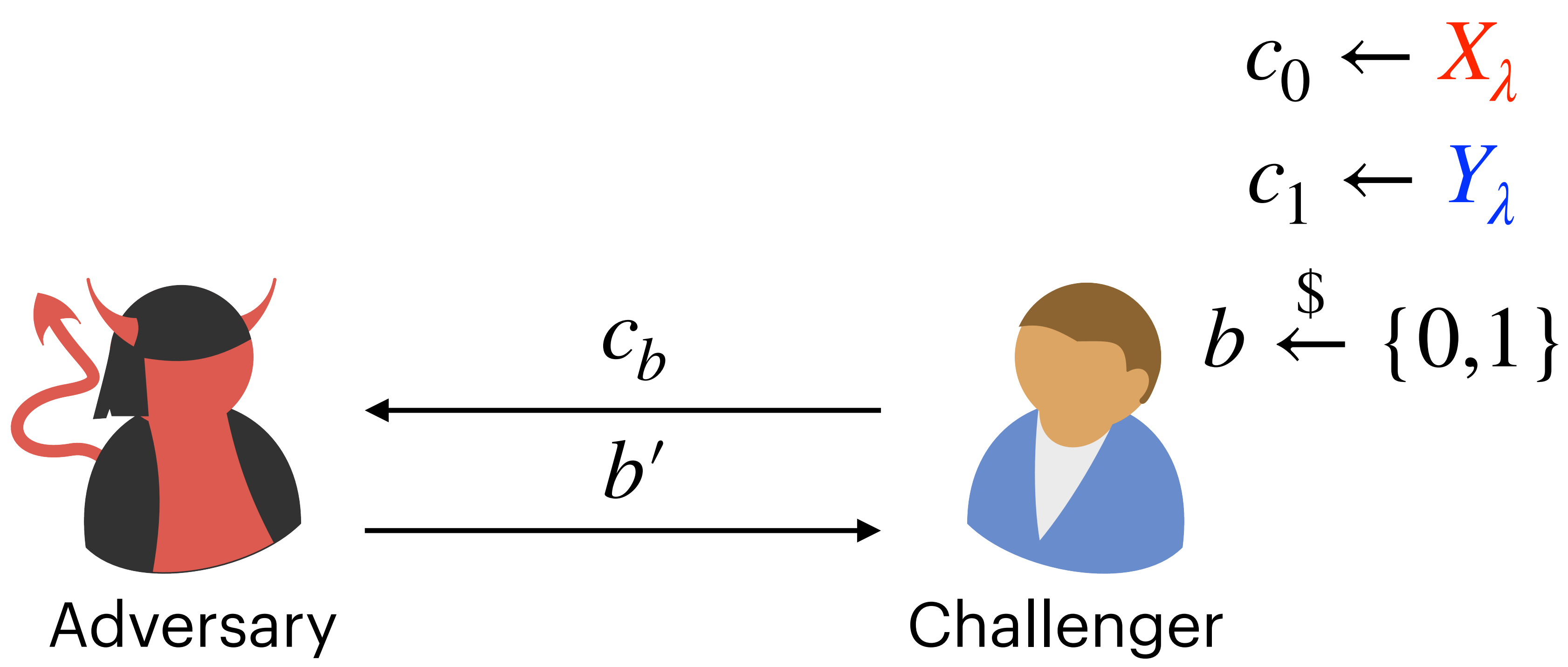


A wins if $b = b'$.

Computational Indistinguishability





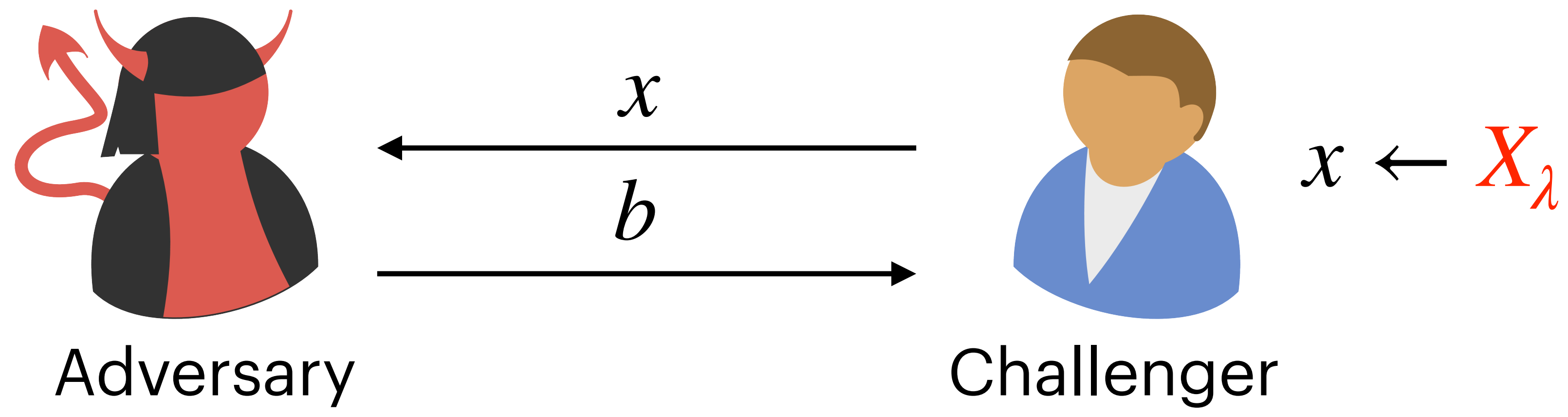


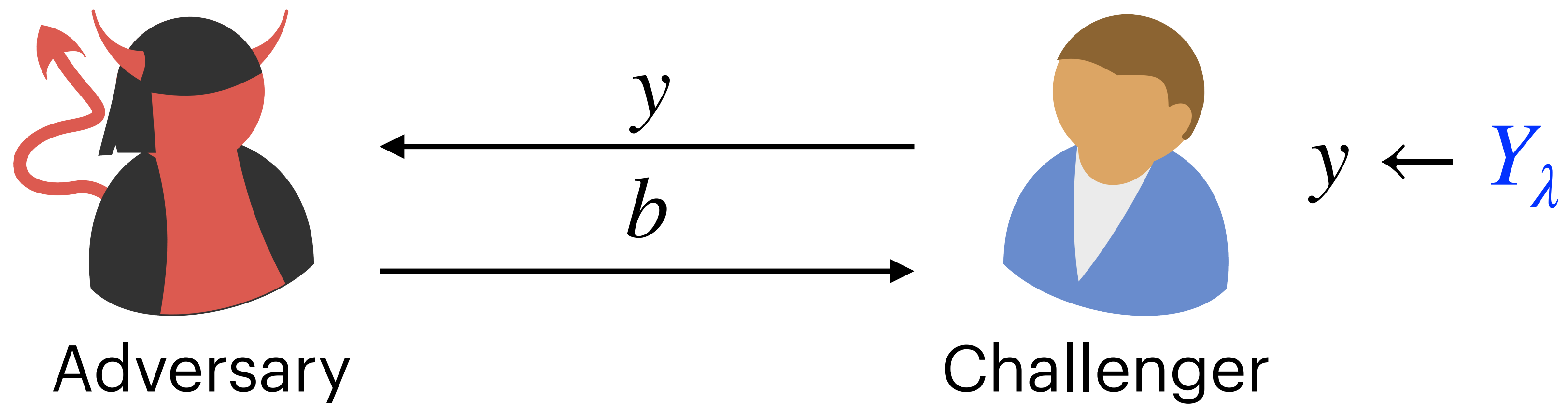
A wins if $b = b'$.

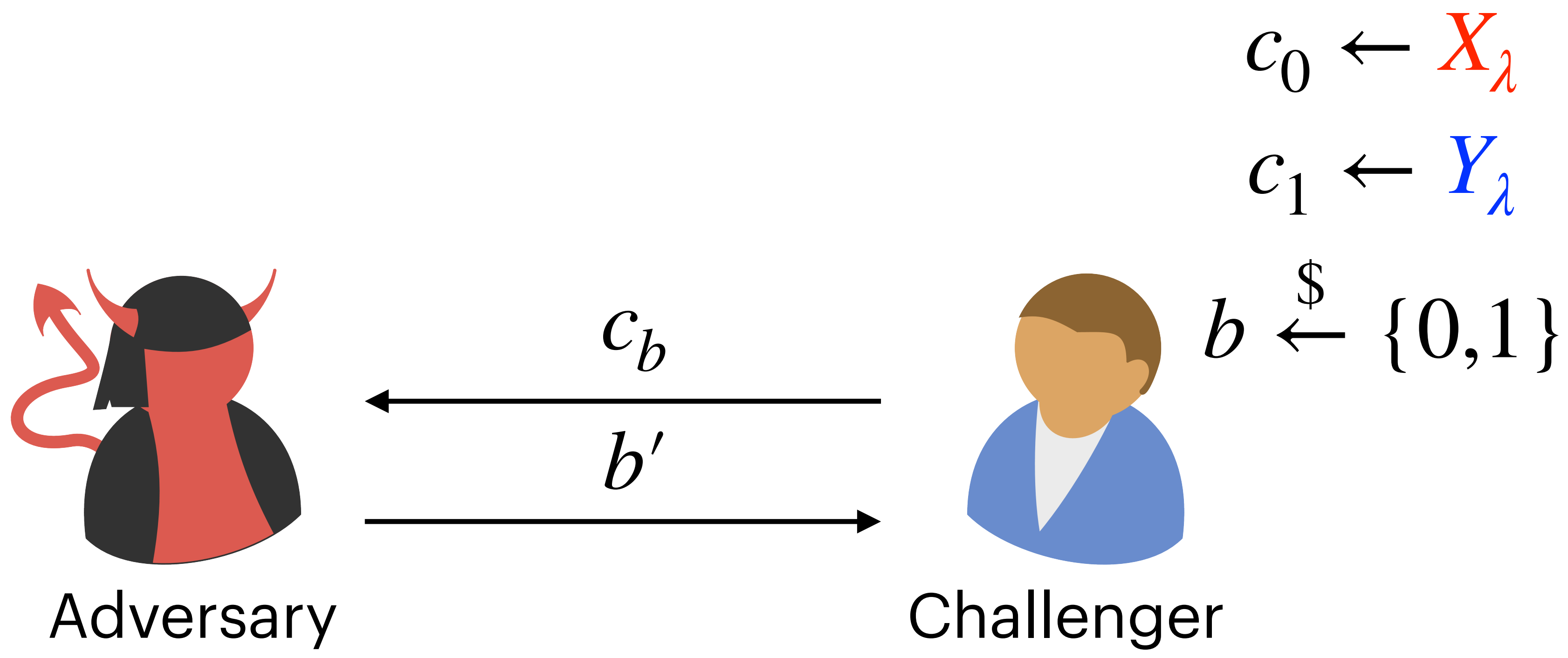
Advantage:

$$\left| \Pr\left[A(1^\lambda, c_b) = 1 \mid b = 0\right] - \Pr\left[A(1^\lambda, c_b) = 1 \mid b = 1\right] \right| \leq \mathsf{negl}(\lambda)$$

Computational Indistinguishability







A wins if $b = b'$.

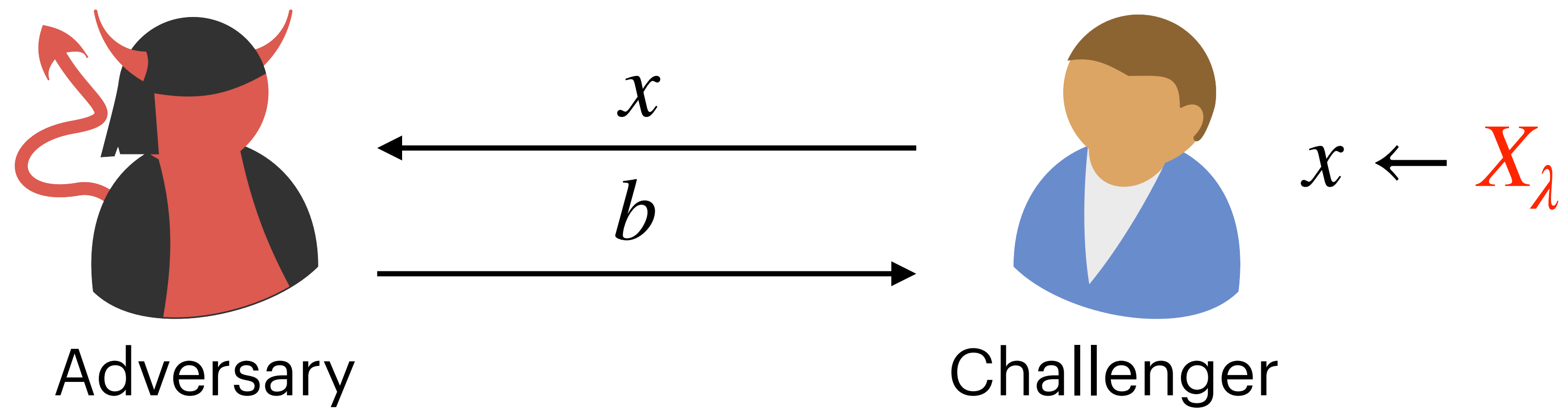
Advantage:

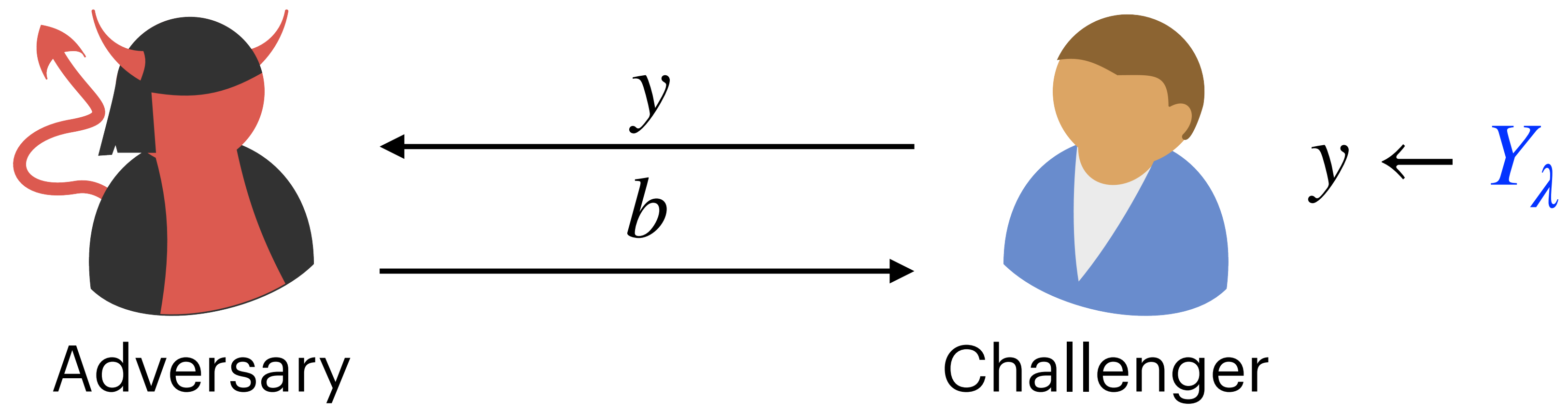
$$\left| \Pr\left[A(1^\lambda, c_b) = 1 \mid b = 0\right] - \Pr\left[A(1^\lambda, c_b) = 1 \mid b = 1\right] \right| \leq \mathsf{negl}(\lambda)$$

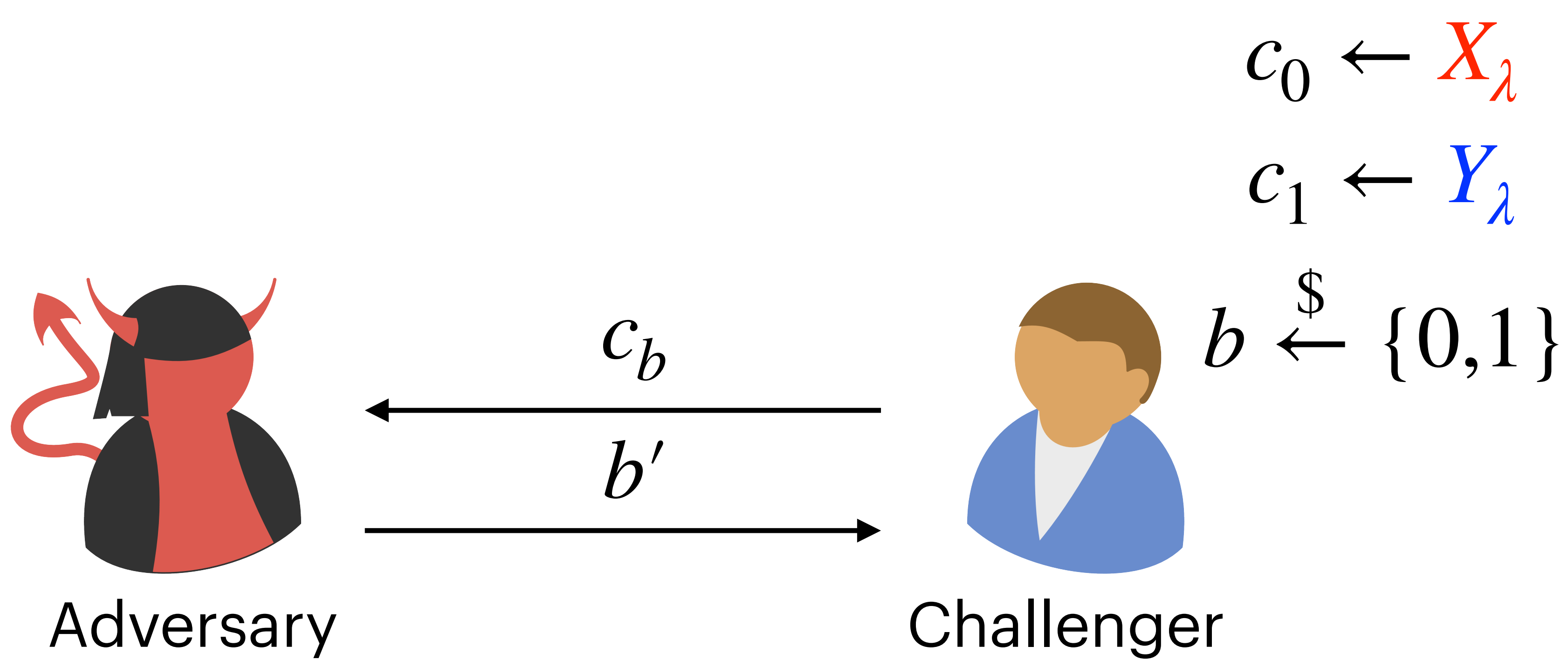
Bit-Guessing:

$$\Pr\left[b' = b : \begin{array}{c} c_0 \leftarrow X_\lambda \\ c_1 \leftarrow Y_\lambda \\ b \xleftarrow{\$} \{0,1\} \\ b' \leftarrow A(1^\lambda, c_b) \end{array} \right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)\,.$$

Computational Indistinguishability







A wins if $b = b'$.

Advantage:

$$\left| \Pr\left[A(1^\lambda, c_b) = 1 \mid b = 0\right] - \Pr\left[A(1^\lambda, c_b) = 1 \mid b = 1\right] \right| \leq \mathsf{negl}(\lambda)$$

Bit-Guessing:

$$\Pr\left[b' = b : \begin{array}{c} c_0 \leftarrow X_\lambda \\ c_1 \leftarrow Y_\lambda \\ b \xleftarrow{\$} \{0,1\} \\ b' \leftarrow A(1^\lambda, c_b) \end{array} \right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)\,.$$

Exercise: Prove both definitions are equivalent.

Summary of Types of Indistinguishability

- Let $X = \{X_i\}_{i\in\mathbb{N}}$ and $Y = \{Y_i\}_{i\in\mathbb{N}}$ be two probability ensembles.

Summary of Types of Indistinguishability

- Let $X = \{X_i\}_{i \in \mathbb{N}}$ and $Y = \{Y_i\}_{i \in \mathbb{N}}$ be two probability ensembles.

- **Perfect Indistinguishability (Identical):** $X \equiv Y$ if for all non-uniform (possibly inefficient) adversaries A and all $\lambda \in \mathbb{N}$

Summary of Types of Indistinguishability

- Let $X = \{X_i\}_{i \in \mathbb{N}}$ and $Y = \{Y_i\}_{i \in \mathbb{N}}$ be two probability ensembles.

- **Perfect Indistinguishability (Identical):** $X \equiv Y$ if for all non-uniform (possibly inefficient) adversaries A and all $\lambda \in \mathbb{N}$

$$\left| \Pr_{x \leftarrow X_\lambda} \left[A(x) = 1 \right] - \Pr_{y \leftarrow Y_\lambda} \left[A(y) = 1 \right] \right| = 0 .$$

Summary of Types of Indistinguishability

- Let $X = \{X_i\}_{i \in \mathbb{N}}$ and $Y = \{Y_i\}_{i \in \mathbb{N}}$ be two probability ensembles.

- **Perfect Indistinguishability (Identical):** $X \equiv Y$ if for all non-uniform (possibly inefficient) adversaries A and all $\lambda \in \mathbb{N}$

$$\left| \Pr_{x \leftarrow X_\lambda} \left[A(x) = 1 \right] - \Pr_{y \leftarrow Y_\lambda} \left[A(y) = 1 \right] \right| = 0 .$$

- **Statistical Indistinguishability:** $X \overset{s}{\approx} Y$ if for all non-uniform (possibly inefficient) adversaries A, there exists a negligible function $\nu(\cdot)$ such that for all $\lambda \in \mathbb{N}$

Summary of Types of Indistinguishability

- Let $X = \{X_i\}_{i\in\mathbb{N}}$ and $Y = \{Y_i\}_{i\in\mathbb{N}}$ be two probability ensembles.

- **Perfect Indistinguishability (Identical):** $X \equiv Y$ if for all non-uniform (possibly inefficient) adversaries A and all $\lambda \in \mathbb{N}$

$$\left| \Pr_{x\leftarrow X_\lambda}\left[A(x) = 1\right] - \Pr_{y\leftarrow Y_\lambda}\left[A(y) = 1\right] \right| = 0 .$$

- **Statistical Indistinguishability:** $X \overset{s}{\approx} Y$ if for all non-uniform (possibly inefficient) adversaries A, there exists a negligible function $\nu(\cdot)$ such that for all $\lambda \in \mathbb{N}$

$$\left| \Pr_{x\leftarrow X_\lambda}\left[A(x) = 1\right] - \Pr_{y\leftarrow Y_\lambda}\left[A(y) = 1\right] \right| \leq \nu(\lambda) .$$

Summary of Types of Indistinguishability

- Let $X = \{X_i\}_{i\in\mathbb{N}}$ and $Y = \{Y_i\}_{i\in\mathbb{N}}$ be two probability ensembles.

- **Perfect Indistinguishability (Identical):** $X \equiv Y$ if for all non-uniform (possibly inefficient) adversaries A and all $\lambda \in \mathbb{N}$

$$\left|\Pr_{x\leftarrow X_\lambda}\left[A(x) = 1\right] - \Pr_{y\leftarrow Y_\lambda}\left[A(y) = 1\right]\right| = 0\,.$$

- **Statistical Indistinguishability:** $X \overset{s}{\approx} Y$ if for all non-uniform (possibly inefficient) adversaries A, there exists a negligible function $\nu(\,\cdot\,)$ such that for all $\lambda \in \mathbb{N}$

$$\left|\Pr_{x\leftarrow X_\lambda}\left[A(x) = 1\right] - \Pr_{y\leftarrow Y_\lambda}\left[A(y) = 1\right]\right| \leq \nu(\lambda)\,.$$

- **Computational Indistinguishability:** $X \overset{c}{\approx} Y$ if for all non-uniform PPT adversaries A, there exists a negligible function $\nu(\,\cdot\,)$ such that for all $\lambda \in \mathbb{N}$

Summary of Types of Indistinguishability

- Let $X = \{X_i\}_{i\in\mathbb{N}}$ and $Y = \{Y_i\}_{i\in\mathbb{N}}$ be two probability ensembles.

- **Perfect Indistinguishability (Identical):** $X \equiv Y$ if for all non-uniform (possibly inefficient) adversaries A and all $\lambda \in \mathbb{N}$

$$\left| \Pr_{x\leftarrow X_\lambda}\left[A(x)=1\right] - \Pr_{y\leftarrow Y_\lambda}\left[A(y)=1\right] \right| = 0\,.$$

- **Statistical Indistinguishability:** $X \overset{s}{\approx} Y$ if for all non-uniform (possibly inefficient) adversaries A, there exists a negligible function $\nu(\cdot)$ such that for all $\lambda \in \mathbb{N}$

$$\left| \Pr_{x\leftarrow X_\lambda}\left[A(x)=1\right] - \Pr_{y\leftarrow Y_\lambda}\left[A(y)=1\right] \right| \le \nu(\lambda)\,.$$

- **Computational Indistinguishability:** $X \overset{c}{\approx} Y$ if for all non-uniform PPT adversaries A, there exists a negligible function $\nu(\cdot)$ such that for all $\lambda \in \mathbb{N}$

$$\left| \Pr_{x\leftarrow X_\lambda}\left[A(1^\lambda, x)=1\right] - \Pr_{y\leftarrow Y_\lambda}\left[A(1^\lambda, y)=1\right] \right| \le \nu(\lambda)\,.$$

Computational Indistinguishability: Properties

Computational Indistinguishability: Properties

- **Closure Property:** Transforming computationally indistinguishable ensembles through an efficient operation preserves computational indistinguishability.

Computational Indistinguishability: Properties

- **Closure Property:** Transforming computationally indistinguishable ensembles through an efficient operation preserves computational indistinguishability.

> **Lemma:** If $\{X_i\}_i \overset{c}{\approx} \{Y_i\}_i$ then for any efficient algorithm M, $\{M(X_i)\}_i \overset{c}{\approx} \{M(Y_i)\}_i$.

Computational Indistinguishability: Properties

- **Closure Property:** Transforming computationally indistinguishable ensembles through an efficient operation preserves computational indistinguishability.

Lemma: If $\{X_i\}_i \overset{c}{\approx} \{Y_i\}_i$ then for any efficient algorithm M, $\{M(X_i)\}_i \overset{c}{\approx} \{M(Y_i)\}_i$.

Intuition: If not, an adversary can use M to distinguish between the two ensembles.

Computational Indistinguishability: Properties

- **Closure Property:** Transforming computationally indistinguishable ensembles through an efficient operation preserves computational indistinguishability.

Lemma: If $\{X_i\}_i \overset{c}{\approx} \{Y_i\}_i$ then for any efficient algorithm M, $\{M(X_i)\}_i \overset{c}{\approx} \{M(Y_i)\}_i$.

Intuition: If not, an adversary can use M to distinguish between the two ensembles.

- **Transitivity:** If X is computationally indistinguishable from Y and Y is computationally indistinguishable from Z, then X is computationally indistinguishable from Z.

Computational Indistinguishability: Properties

- **Closure Property:** Transforming computationally indistinguishable ensembles through an efficient operation preserves computational indistinguishability.

Lemma: If $\{X_i\}_i \overset{c}{\approx} \{Y_i\}_i$ then for any efficient algorithm M, $\{M(X_i)\}_i \overset{c}{\approx} \{M(Y_i)\}_i$.

Intuition: If not, an adversary can use M to distinguish between the two ensembles.

- **Transitivity:** If X is computationally indistinguishable from Y and Y is computationally indistinguishable from Z, then X is computationally indistinguishable from Z.

Lemma: If $X \overset{c}{\approx} Y$ and $Y \overset{c}{\approx} Z$ then $X \overset{c}{\approx} Z$.

Computational Indistinguishability: Properties

- **Closure Property:** Transforming computationally indistinguishable ensembles through an efficient operation preserves computational indistinguishability.

> **Lemma:** If $\{X_i\}_i \overset{c}{\approx} \{Y_i\}_i$ then for any efficient algorithm M, $\{M(X_i)\}_i \overset{c}{\approx} \{M(Y_i)\}_i$.

Intuition: If not, an adversary can use M to distinguish between the two ensembles.

- **Transitivity:** If X is computationally indistinguishable from Y and Y is computationally indistinguishable from Z, then X is computationally indistinguishable from Z.

> **Lemma:** If $X \overset{c}{\approx} Y$ and $Y \overset{c}{\approx} Z$ then $X \overset{c}{\approx} Z$.
>
> **Proof:** Let A be an efficient adversary.
>
> $$X \overset{c}{\approx} Y \implies \left| \Pr_{x \leftarrow X_\lambda}\left[A(1^\lambda, x) = 1\right] - \Pr_{y \leftarrow Y_\lambda}\left[A(1^\lambda, y) = 1\right] \right| \leq \nu_1(\lambda)\,.$$

Computational Indistinguishability: Properties

- **Closure Property:** Transforming computationally indistinguishable ensembles through an efficient operation preserves computational indistinguishability.

Lemma: If $\{X_i\}_i \overset{c}{\approx} \{Y_i\}_i$ then for any efficient algorithm M, $\{M(X_i)\}_i \overset{c}{\approx} \{M(Y_i)\}_i$.

Intuition: If not, an adversary can use M to distinguish between the two ensembles.

- **Transitivity:** If X is computationally indistinguishable from Y and Y is computationally indistinguishable from Z, then X is computationally indistinguishable from Z.

Lemma: If $X \overset{c}{\approx} Y$ and $Y \overset{c}{\approx} Z$ then $X \overset{c}{\approx} Z$.

Proof: Let A be an efficient adversary.

$$X \overset{c}{\approx} Y \quad \Longrightarrow \quad \left| \Pr_{x \leftarrow X_\lambda}\left[A(1^\lambda, x) = 1\right] - \Pr_{y \leftarrow Y_\lambda}\left[A(1^\lambda, y) = 1\right] \right| \leq \nu_1(\lambda)\,.$$

$$Y \overset{c}{\approx} Z \quad \Longrightarrow \quad \left| \Pr_{y \leftarrow Y_\lambda}\left[A(1^\lambda, y) = 1\right] - \Pr_{z \leftarrow Z_\lambda}\left[A(1^\lambda, z) = 1\right] \right| \leq \nu_2(\lambda)\,.$$

Computational Indistinguishability: Properties

- **Closure Property:** Transforming computationally indistinguishable ensembles through an efficient operation preserves computational indistinguishability.

Lemma: If $\{X_i\}_i \overset{c}{\approx} \{Y_i\}_i$ then for any efficient algorithm M, $\{M(X_i)\}_i \overset{c}{\approx} \{M(Y_i)\}_i$.

Intuition: If not, an adversary can use M to distinguish between the two ensembles.

- **Transitivity:** If X is computationally indistinguishable from Y and Y is computationally indistinguishable from Z, then X is computationally indistinguishable from Z.

Lemma: If $X \overset{c}{\approx} Y$ and $Y \overset{c}{\approx} Z$ then $X \overset{c}{\approx} Z$.

Proof: Let A be an efficient adversary.

$$X \overset{c}{\approx} Y \implies \left| \Pr_{x \leftarrow X_\lambda}\left[A(1^\lambda, x) = 1\right] - \Pr_{y \leftarrow Y_\lambda}\left[A(1^\lambda, y) = 1\right] \right| \leq \nu_1(\lambda).$$

$$Y \overset{c}{\approx} Z \implies \left| \Pr_{y \leftarrow Y_\lambda}\left[A(1^\lambda, y) = 1\right] - \Pr_{z \leftarrow Z_\lambda}\left[A(1^\lambda, z) = 1\right] \right| \leq \nu_2(\lambda).$$

Thus $\left| \Pr_{x \leftarrow X_\lambda}\left[A(1^\lambda, x) = 1\right] - \Pr_{z \leftarrow Z_\lambda}\left[A(1^\lambda, z) = 1\right] \right|$

Computational Indistinguishability: Properties

- **Closure Property:** Transforming computationally indistinguishable ensembles through an efficient operation preserves computational indistinguishability.

Lemma: If $\{X_i\}_i \overset{c}{\approx} \{Y_i\}_i$ then for any efficient algorithm M, $\{M(X_i)\}_i \overset{c}{\approx} \{M(Y_i)\}_i$.

Intuition: If not, an adversary can use M to distinguish between the two ensembles.

- **Transitivity:** If X is computationally indistinguishable from Y and Y is computationally indistinguishable from Z, then X is computationally indistinguishable from Z.

Lemma: If $X \overset{c}{\approx} Y$ and $Y \overset{c}{\approx} Z$ then $X \overset{c}{\approx} Z$.

Proof: Let A be an efficient adversary.

$$X \overset{c}{\approx} Y \implies \left| \Pr_{x \leftarrow X_\lambda}\left[A(1^\lambda, x) = 1\right] - \Pr_{y \leftarrow Y_\lambda}\left[A(1^\lambda, y) = 1\right] \right| \leq \nu_1(\lambda) .$$

$$Y \overset{c}{\approx} Z \implies \left| \Pr_{y \leftarrow Y_\lambda}\left[A(1^\lambda, y) = 1\right] - \Pr_{z \leftarrow Z_\lambda}\left[A(1^\lambda, z) = 1\right] \right| \leq \nu_2(\lambda) .$$

$$\text{Thus} \quad \left| \Pr_{x \leftarrow X_\lambda}\left[A(1^\lambda, x) = 1\right] - \Pr_{z \leftarrow Z_\lambda}\left[A(1^\lambda, z) = 1\right] \right| \leq \nu_1(\lambda) + \nu_2(\lambda)$$

Computational Indistinguishability: Properties

- **Closure Property:** Transforming computationally indistinguishable ensembles through an efficient operation preserves computational indistinguishability.

> **Lemma:** If $\{X_i\}_i \overset{c}{\approx} \{Y_i\}_i$ then for any efficient algorithm M, $\{M(X_i)\}_i \overset{c}{\approx} \{M(Y_i)\}_i$.

Intuition: If not, an adversary can use M to distinguish between the two ensembles.

- **Transitivity:** If X is computationally indistinguishable from Y and Y is computationally indistinguishable from Z, then X is computationally indistinguishable from Z.

> **Lemma:** If $X \overset{c}{\approx} Y$ and $Y \overset{c}{\approx} Z$ then $X \overset{c}{\approx} Z$.
>
> **Proof:** Let A be an efficient adversary.
>
> $$X \overset{c}{\approx} Y \implies \left| \Pr_{x \leftarrow X_\lambda}\left[A(1^\lambda, x) = 1\right] - \Pr_{y \leftarrow Y_\lambda}\left[A(1^\lambda, y) = 1\right] \right| \leq \nu_1(\lambda)\,.$$
>
> $$Y \overset{c}{\approx} Z \implies \left| \Pr_{y \leftarrow Y_\lambda}\left[A(1^\lambda, y) = 1\right] - \Pr_{z \leftarrow Z_\lambda}\left[A(1^\lambda, z) = 1\right] \right| \leq \nu_2(\lambda)\,.$$
>
> Thus $$\left| \Pr_{x \leftarrow X_\lambda}\left[A(1^\lambda, x) = 1\right] - \Pr_{z \leftarrow Z_\lambda}\left[A(1^\lambda, z) = 1\right] \right| \leq \nu_1(\lambda) + \nu_2(\lambda) \leq \nu(\lambda)\,.$$

Generalizing Transitivity: Hybrid Lemma

Lemma (Hybrid Lemma): Let λ be the security parameter and $n := n(\lambda)$ be a polynomial. If $X_1, \ldots, X_n$ be probability ensembles such that for all $i \in \{0, \ldots, n-1\}$ $X_i \overset{c}{\approx} X_{i+1}$, then $X_1 \overset{c}{\approx} X_n$.

Generalizing Transitivity: Hybrid Lemma

Lemma (Hybrid Lemma): Let λ be the security parameter and $n := n(\lambda)$ be a polynomial. If $X_1, \ldots, X_n$ be probability ensembles such that for all $i \in \{0,\ldots,n-1\}$ $X_i \overset{c}{\approx} X_{i+1}$, then $X_1 \overset{c}{\approx} X_n$.

This is the hybrid technique, stated more generally, in the computational setting.

Generalizing Transitivity: Hybrid Lemma

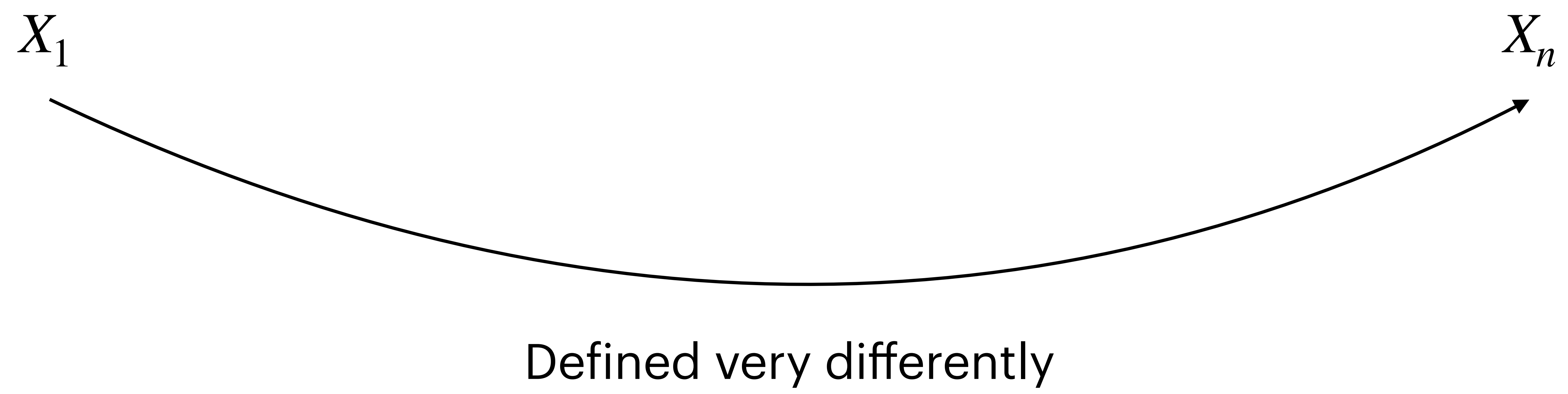
Lemma (Hybrid Lemma): Let λ be the security parameter and $n := n(\lambda)$ be a polynomial. If $X_1, \ldots, X_n$ be probability ensembles such that for all $i \in \{0,\ldots,n-1\}$ $X_i \overset{c}{\approx} X_{i+1}$, then $X_1 \overset{c}{\approx} X_n$.

This is the hybrid technique, stated more generally, in the computational setting.

Used in most crypto proofs!

Generalizing Transitivity: Hybrid Lemma

Lemma (Hybrid Lemma): Let λ be the security parameter and $n := n(\lambda)$ be a polynomial. If $X_1, \ldots, X_n$ be probability ensembles such that for all $i \in \{0, \ldots, n-1\}$ $X_i \overset{c}{\approx} X_{i+1}$, then $X_1 \overset{c}{\approx} X_n$.



Generalizing Transitivity: Hybrid Lemma

Lemma (Hybrid Lemma): Let λ be the security parameter and $n := n(\lambda)$ be a polynomial. If $X_1, \ldots, X_n$ be probability ensembles such that for all $i \in \{0, \ldots, n-1\}$ $X_i \overset{c}{\approx} X_{i+1}$, then $X_1 \overset{c}{\approx} X_n$.

Isolate each conceptual change

Generalizing Transitivity: Hybrid Lemma

Lemma (Hybrid Lemma): Let λ be the security parameter and $n := n(\lambda)$ be a polynomial. If $X_1, \ldots, X_n$ be probability ensembles such that for all $i \in \{0, \ldots, n-1\}$ $X_i \overset{c}{\approx} X_{i+1}$, then $X_1 \overset{c}{\approx} X_n$.

$$X_1 \quad \overset{c}{\approx} \quad X_2 \quad \overset{c}{\approx} \quad X_3 \qquad \cdots \qquad X_{n-1} \quad \overset{c}{\approx} \quad X_n$$

Isolate each conceptual change
and show that it preserves indistinguishability

Generalizing Transitivity: Hybrid Lemma

Lemma (Hybrid Lemma): Let λ be the security parameter and $n := n(\lambda)$ be a polynomial. If $X_1, \ldots, X_n$ be probability ensembles such that for all $i \in \{0, \ldots, n-1\}$ $X_i \overset{c}{\approx} X_{i+1}$, then $X_1 \overset{c}{\approx} X_n$.

$$X_1 \overset{c}{\approx} X_2 \overset{c}{\approx} X_3 \quad \cdots \quad X_{n-1} \overset{c}{\approx} X_n$$

Aggregate of all changes preserves indistinguishability

Generalizing Transitivity: Hybrid Lemma

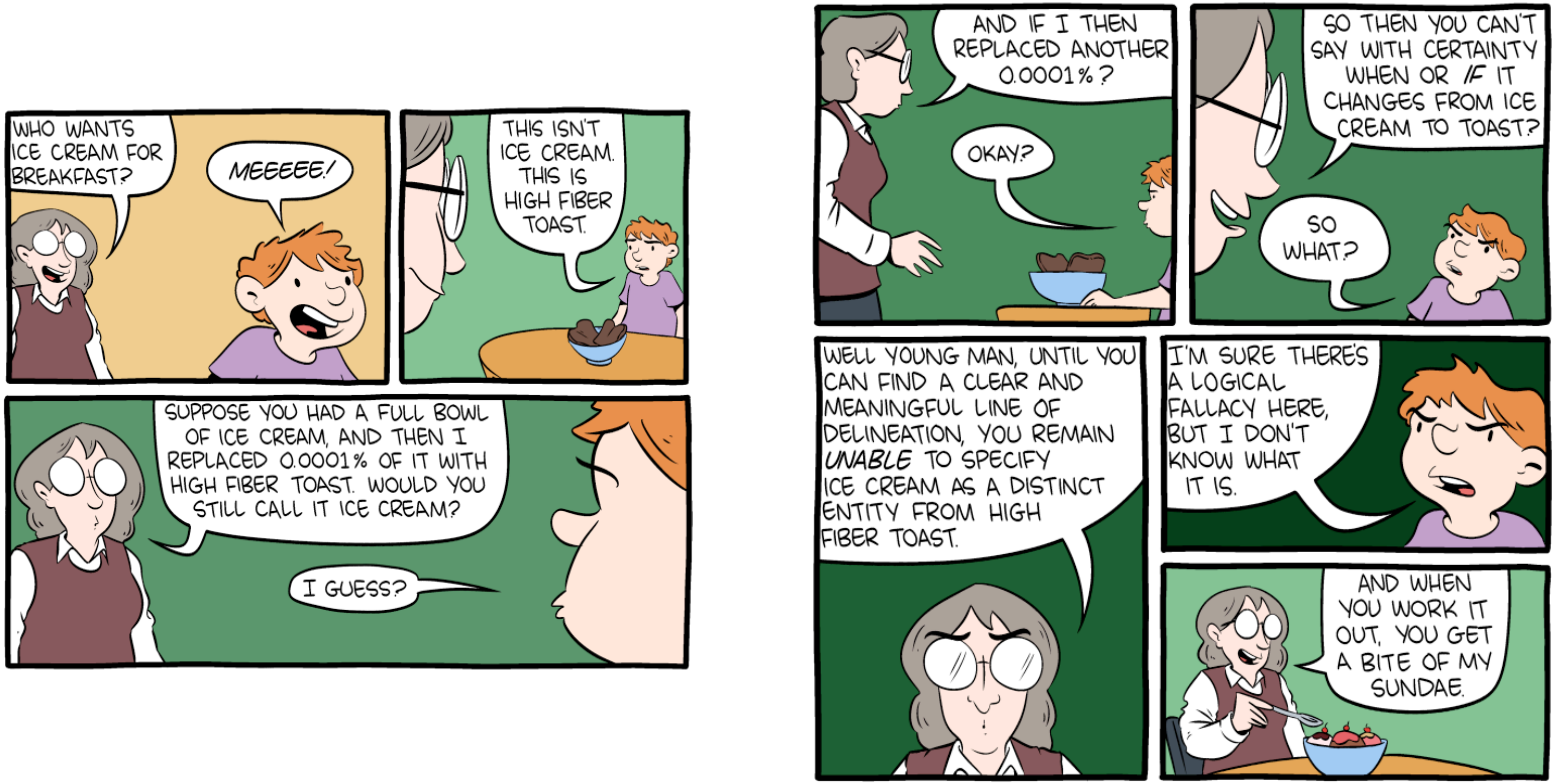
Lemma (Hybrid Lemma): Let λ be the security parameter and $n := n(\lambda)$ be a polynomial. If $X_1, \ldots, X_n$ be probability ensembles such that for all $i \in \{0, \ldots, n-1\}$ $X_i \overset{c}{\approx} X_{i+1}$, then $X_1 \overset{c}{\approx} X_n$.

$$X_1 \quad \overset{c}{\approx} \quad X_2 \quad \overset{c}{\approx} \quad X_3 \quad \cdots \quad X_{n-1} \quad \overset{c}{\approx} \quad X_n$$

Aggregate of all changes preserves indistinguishability

Number of hybrid ensembles must be polynomial.

Generalizing Transitivity: Hybrid Lemma



Overcoming Shannon's Bound

- Let's take another look at OTP:

One-Time Pad

Let λ be the security parameter.

- KeyGen(1^λ): $k \xleftarrow{\$} \{0,1\}^\lambda$.
- Enc(k, m): ct $:= k \oplus m$.
- Dec(k, ct): $m := k \oplus$ ct.

Overcoming Shannon's Bound

- Let's take another look at OTP:

One-Time Pad

Let λ be the security parameter.

- KeyGen(1^λ): $k \xleftarrow{\$} \{0,1\}^\lambda$.
- Enc(k, m): ct $:= k \oplus m$.
- Dec(k, ct): $m := k \oplus$ ct.

- Why did we need a key as long as the message?

Overcoming Shannon's Bound

- Let's take another look at OTP:

One-Time Pad

Let λ be the security parameter.

- KeyGen(1^λ): $k \xleftarrow{\$} \{0,1\}^\lambda$.
- Enc(k, m): ct $:= k \oplus m$.
- Dec(k, ct): $m := k \oplus$ ct.

- Why did we need a key as long as the message?
  - To mask every bit of the message.

Overcoming Shannon's Bound

- Let's take another look at OTP:

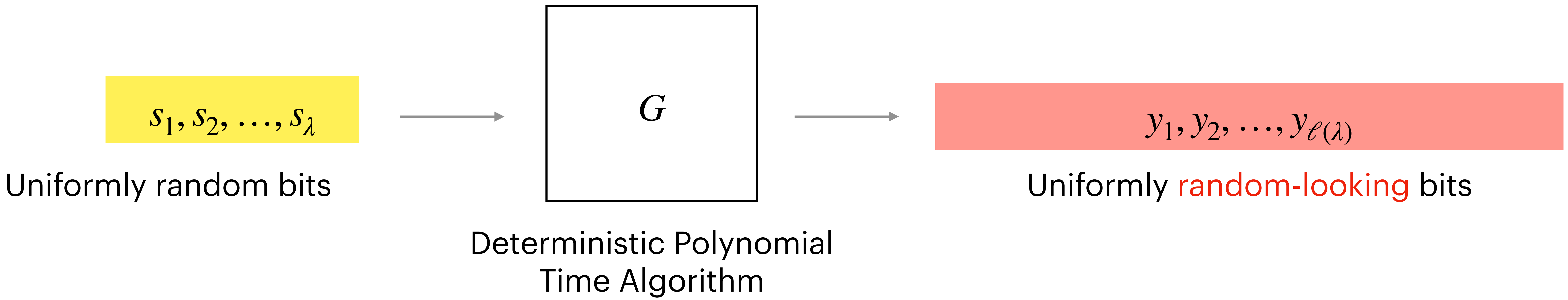
<u>One-Time Pad</u>

Let λ be the security parameter.

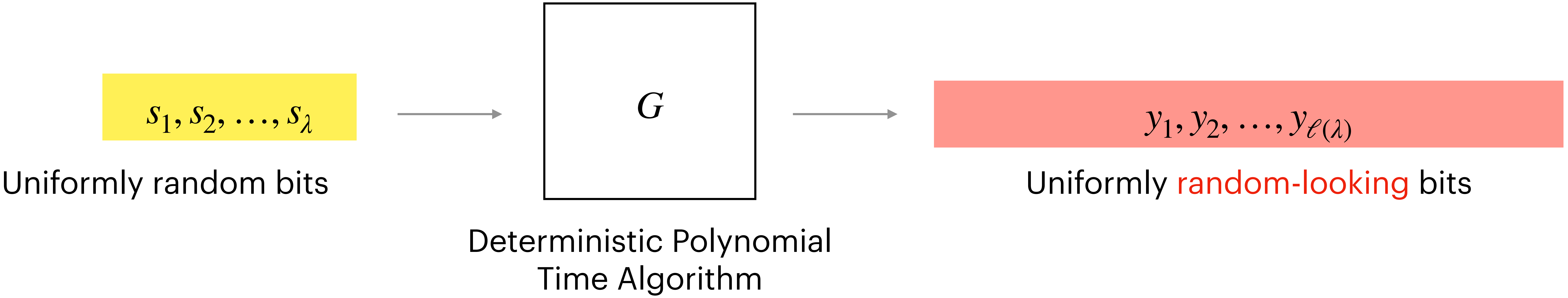
- KeyGen(1^λ): $k \xleftarrow{\$} \{0,1\}^\lambda$.
- Enc(k, m): ct $:= k \oplus m$.
- Dec(k, ct): $m := k \oplus$ ct.

- Why did we need a key as long as the message?
  - To mask every bit of the message.
  - What if we can expand a few random bits into many random "looking" bits?

Pseudorandom Generator

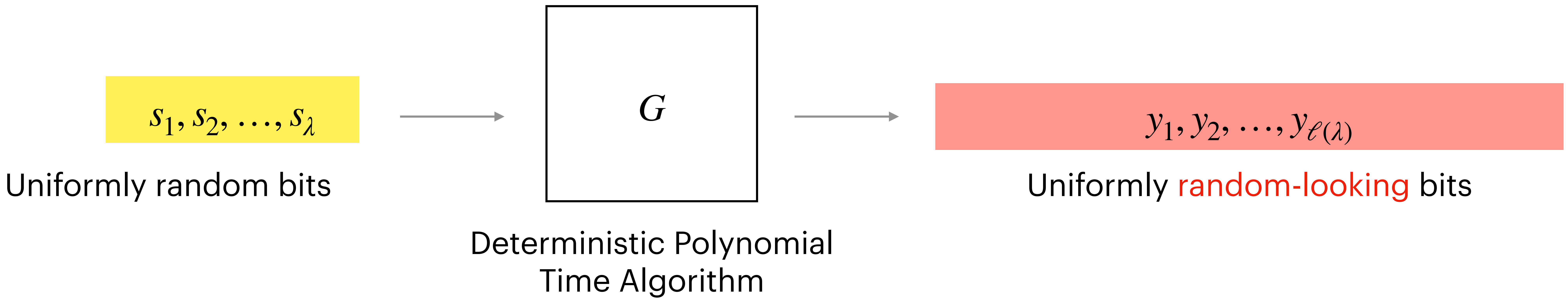


Pseudorandom Generator



- A **pseudorandom generator** $G : \{0,1\}^\lambda \rightarrow \{0,1\}^{\ell(\lambda)}$ takes a short, uniformly random seed $s \in \{0,1\}^\lambda$ and outputs a longer pseudorandom string $y \in \{0,1\}^{\ell(\lambda)}$.

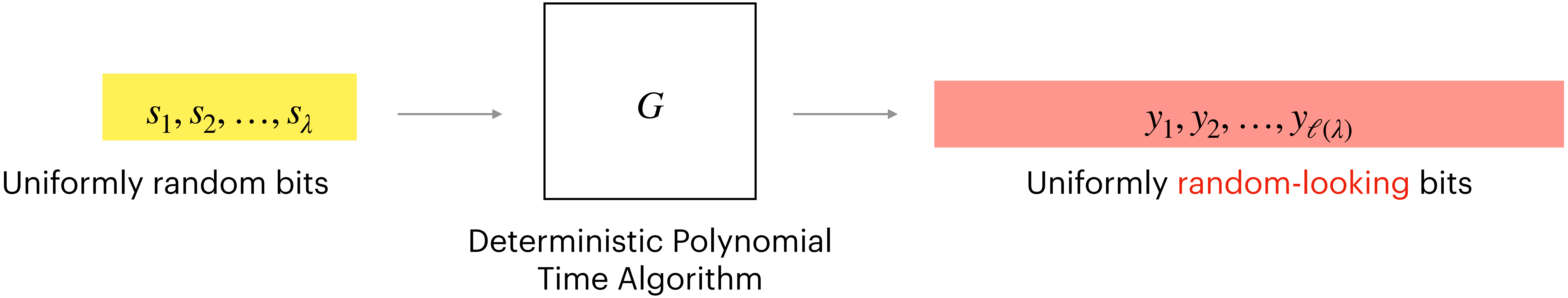
Pseudorandom Generator



- A **pseudorandom generator** $G : \{0,1\}^\lambda \to \{0,1\}^{\ell(\lambda)}$ takes a short, uniformly random seed $s \in \{0,1\}^\lambda$ and outputs a longer pseudorandom string $y \in \{0,1\}^{\ell(\lambda)}$.

- **Pseudorandom:** The output of G is as good as a uniformly random $\ell(\lambda)$-bit string to any efficient distinguisher i.e.,

Pseudorandom Generator



- A **pseudorandom generator** $G : \{0,1\}^\lambda \to \{0,1\}^{\ell(\lambda)}$ takes a short, uniformly random seed $s \in \{0,1\}^\lambda$ and outputs a longer pseudorandom string $y \in \{0,1\}^{\ell(\lambda)}$.

- **Pseudorandom:** The output of G is as good as a uniformly random $\ell(\lambda)$-bit string to any efficient distinguisher i.e.,

$$\{G(s) : s \xleftarrow{\$} \{0,1\}^\lambda\} \quad \overset{c}{\approx} \quad \{r : r \xleftarrow{\$} \{0,1\}^{\ell(\lambda)}\}$$

Pseudorandom Generator

Pseudorandom Generator

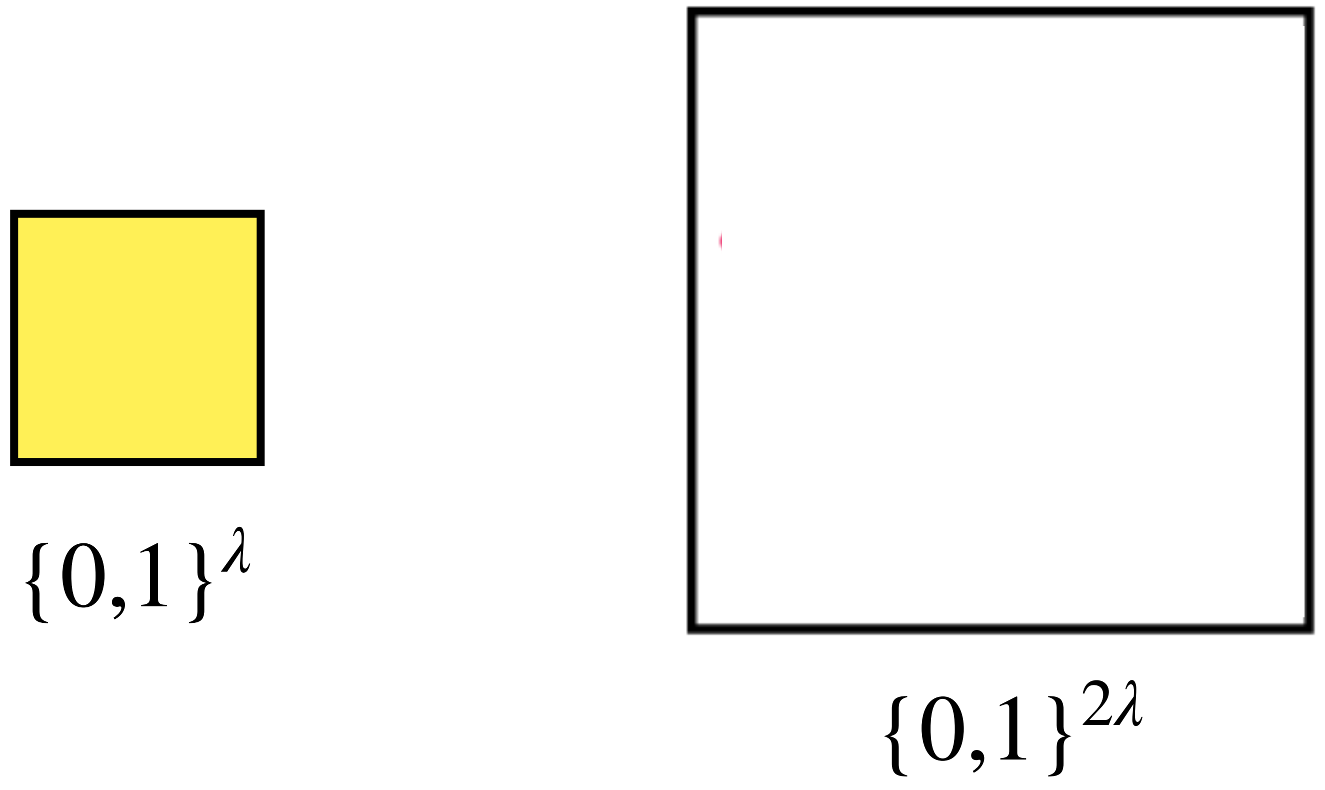
A deterministic algorithm G is called a pseudorandom generator (PRG) if:

- G can be computed in polynomial time,
- On input any $s \in \{0,1\}^\lambda$, G outputs a $\ell(\lambda)$-bit string such that $\ell(\lambda) > \lambda$,
- $\{G(s) : s \stackrel{\$}{\leftarrow} \{0,1\}^\lambda\} \quad \stackrel{c}{\approx} \quad \{r : r \stackrel{\$}{\leftarrow} \{0,1\}^{\ell(\lambda)}\}$

The stretch of G is defined as $\ell(\lambda) - \lambda$.

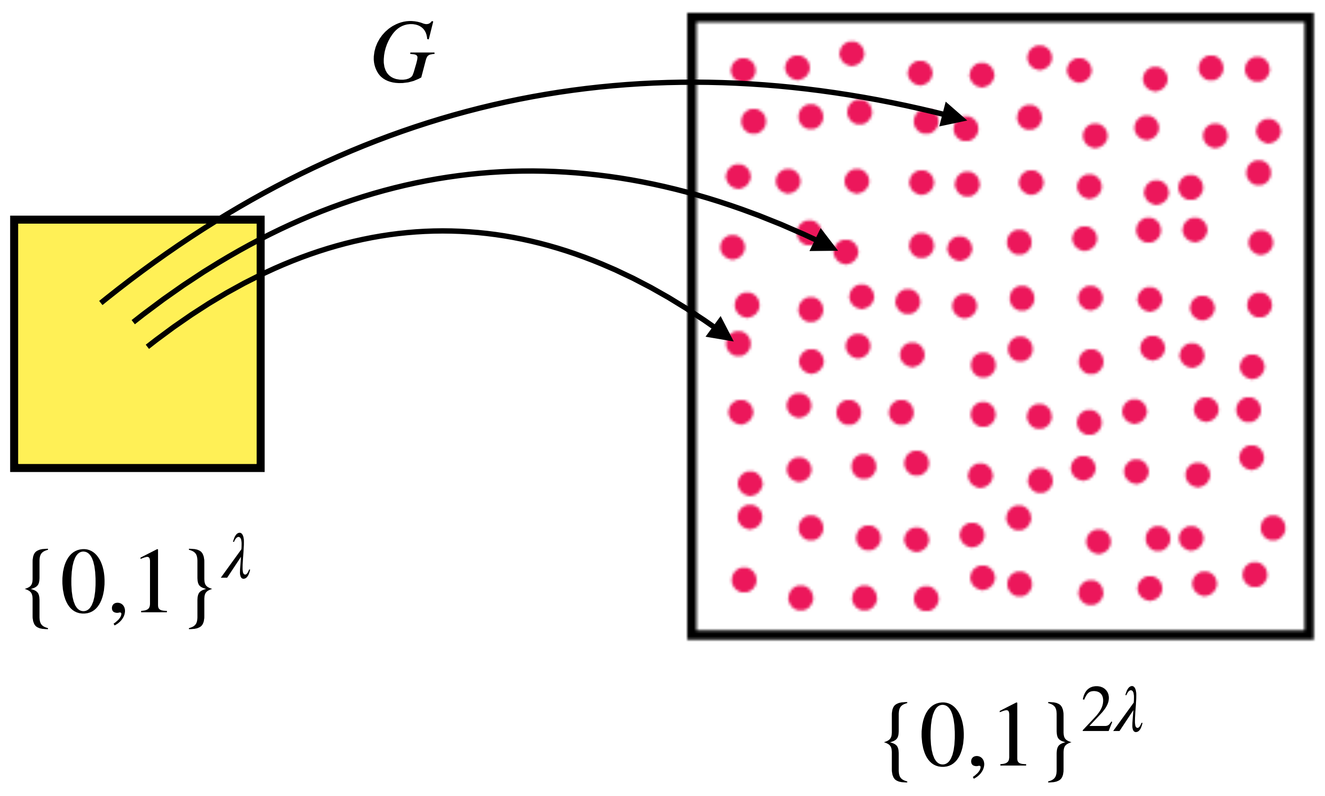
Why <u>Pseudo</u>random?

Consider a PRG G with λ-bit stretch.



Why Pseudorandom?

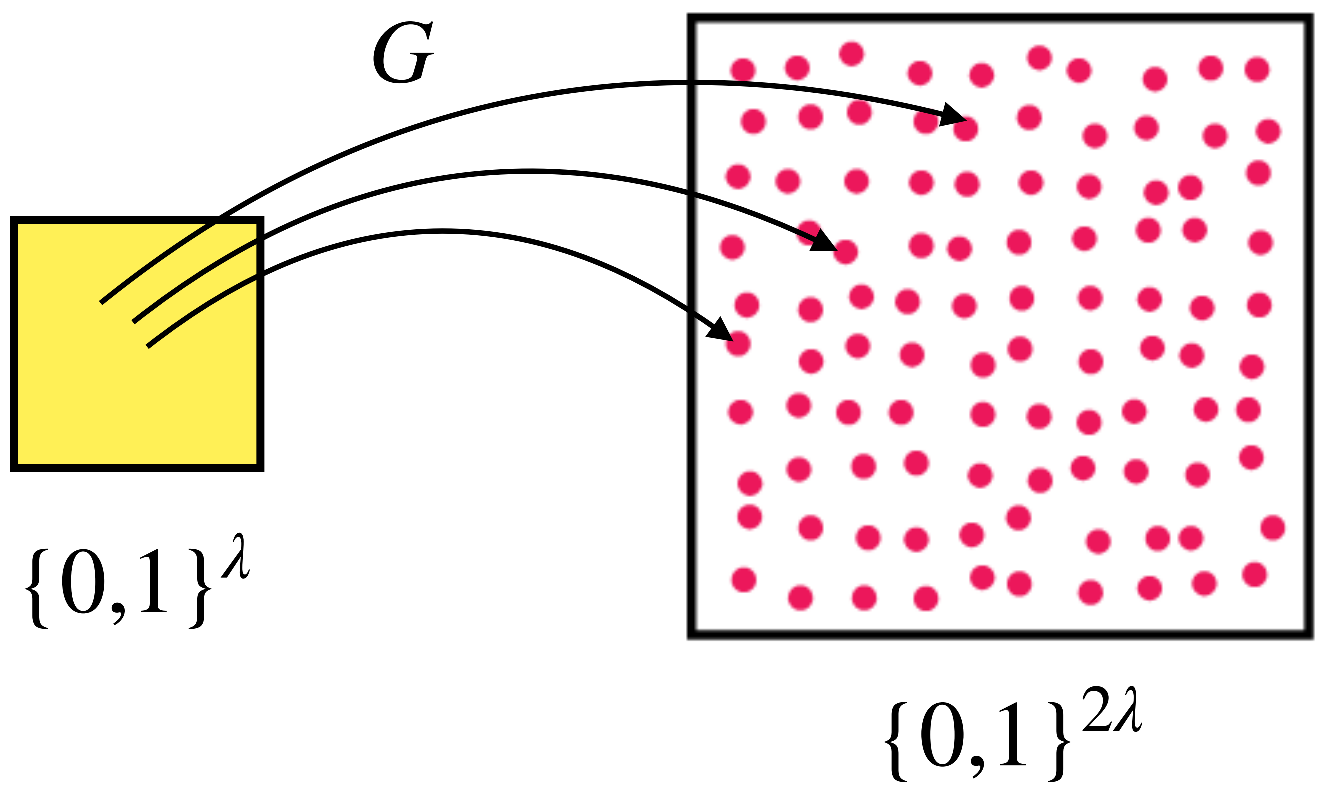
Consider a PRG G with λ-bit stretch.



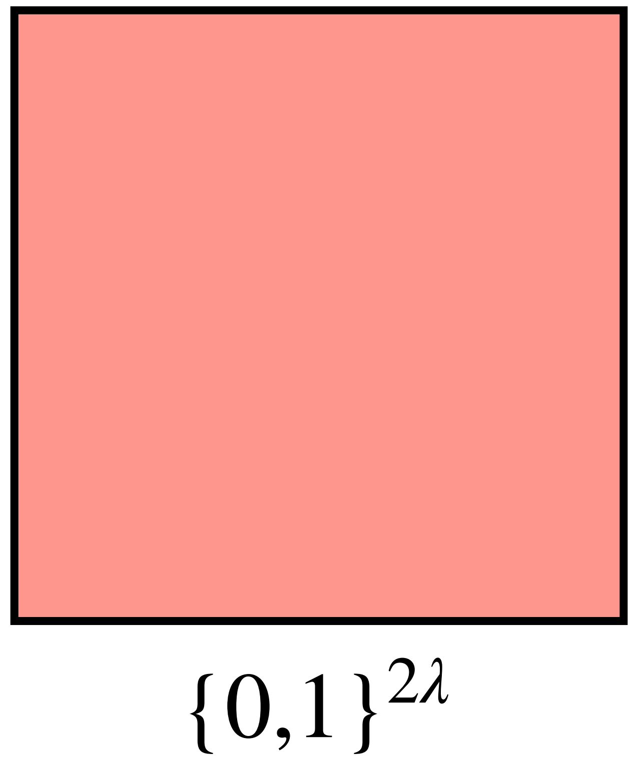
Pseudorandom Distribution

Why Pseudorandom?

Consider a PRG G with λ-bit stretch.



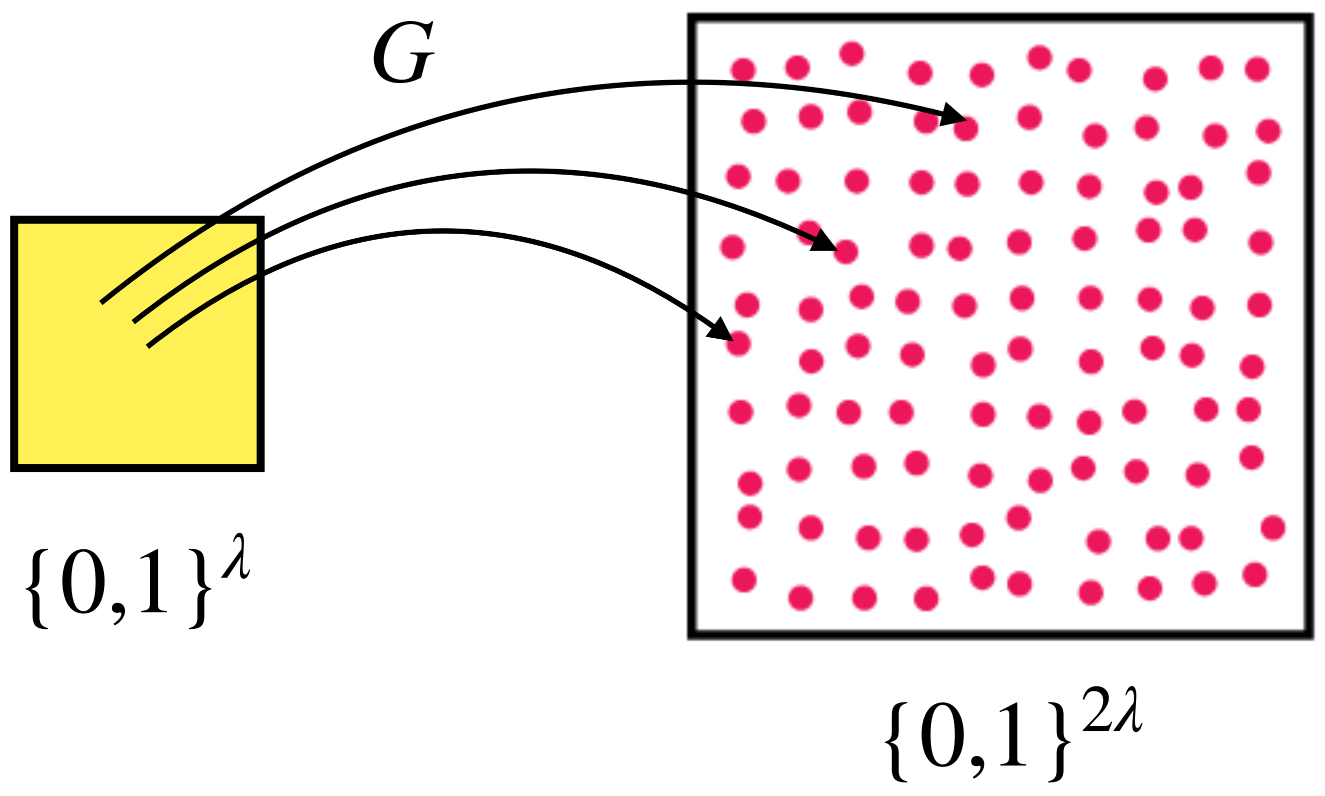
Pseudorandom Distribution

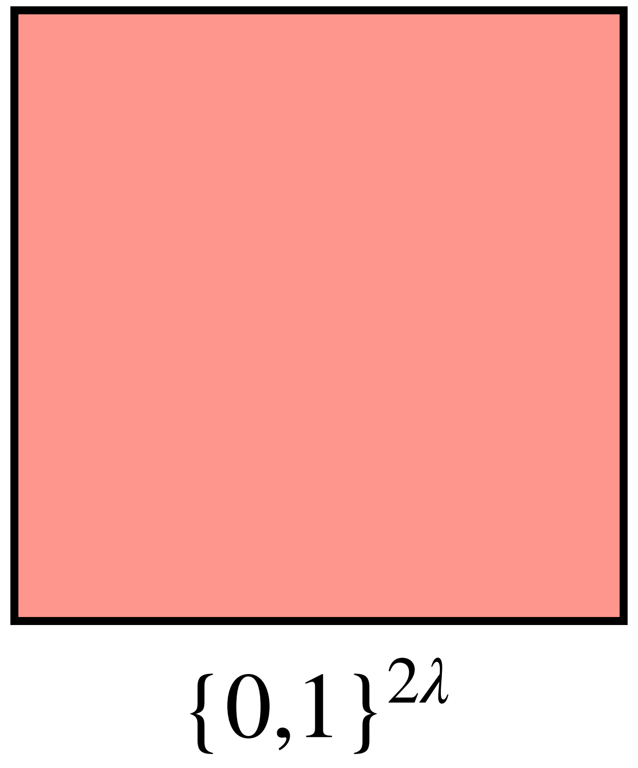


Uniform Distribution

Why Pseudorandom?

Consider a PRG G with λ-bit stretch.

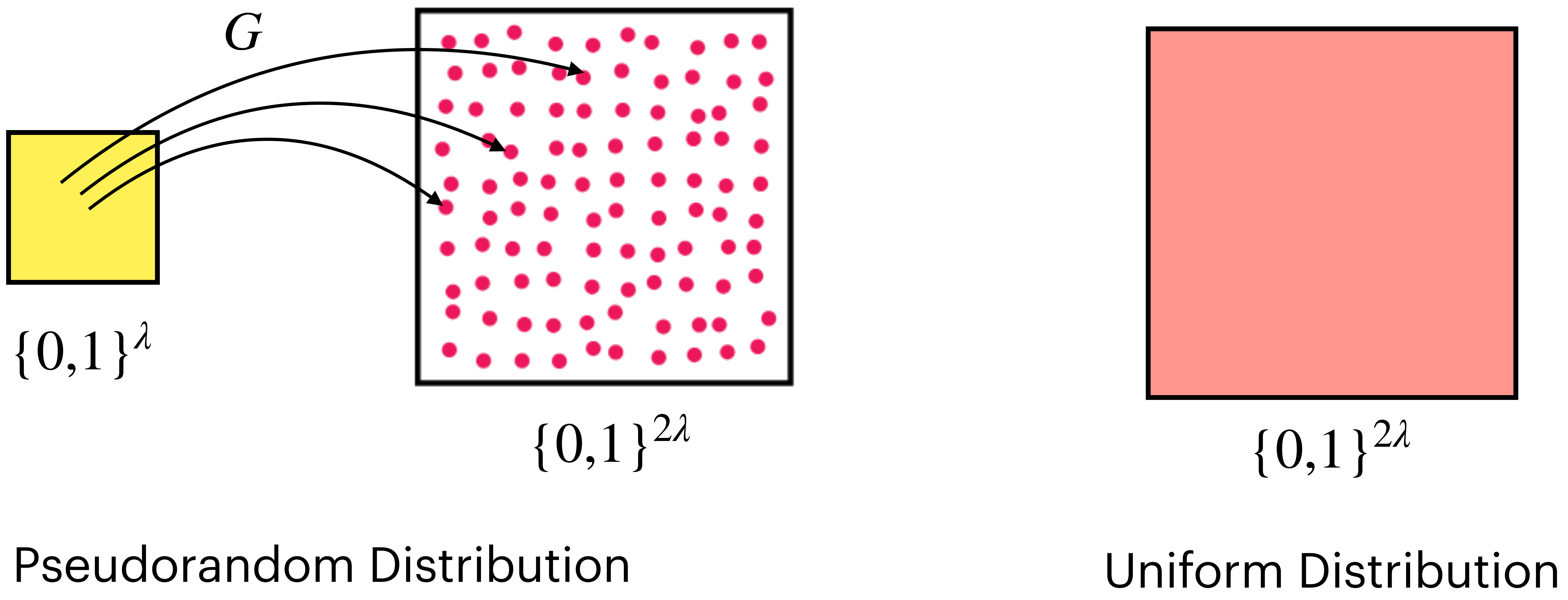




- Relatively, the PRG's output space is tiny. The output space makes up a negligible fraction (i.e, $2^{-\lambda}$) of all 2λ-bit strings.

Why Pseudorandom?

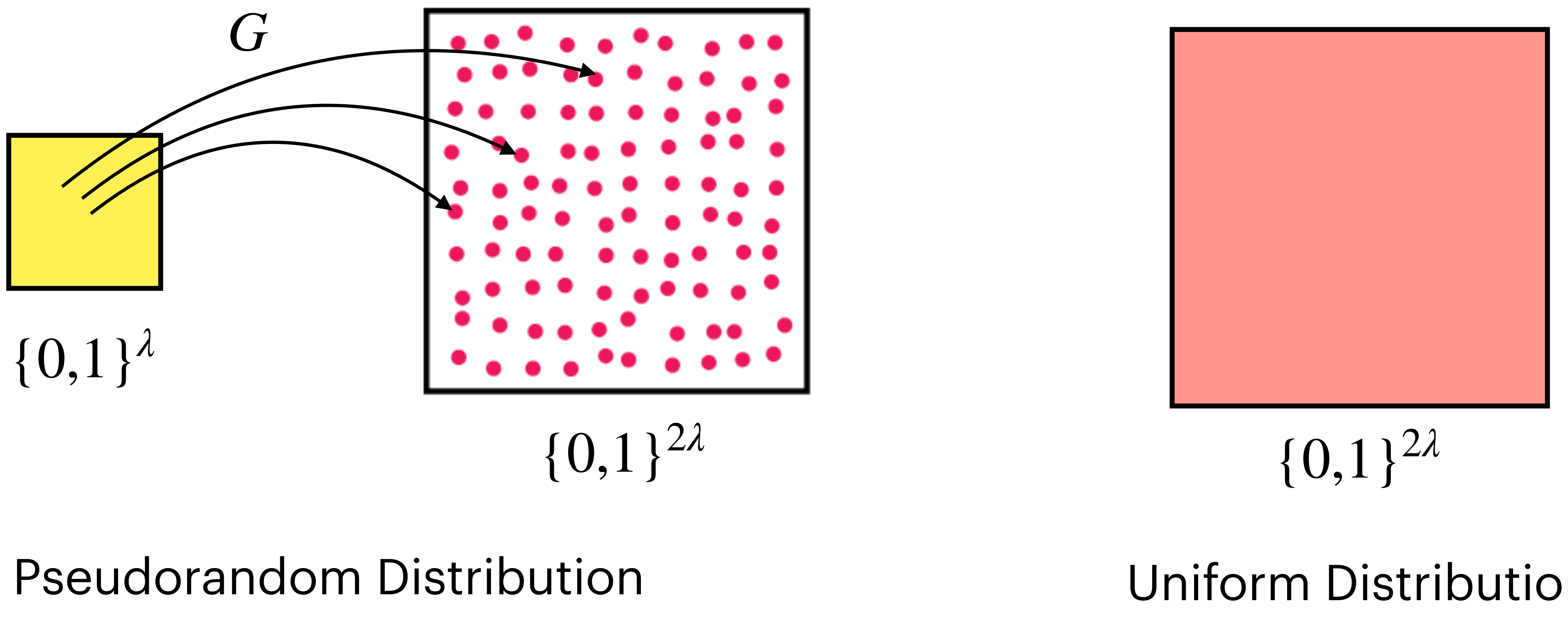
Consider a PRG G with λ-bit stretch.



- Relatively, the PRG's output space is tiny. The output space makes up a negligible fraction (i.e, $2^{-\lambda}$) of all 2λ-bit strings.
  - How can an efficient adversary break PRG security with negligible probability?

Why Pseudorandom?

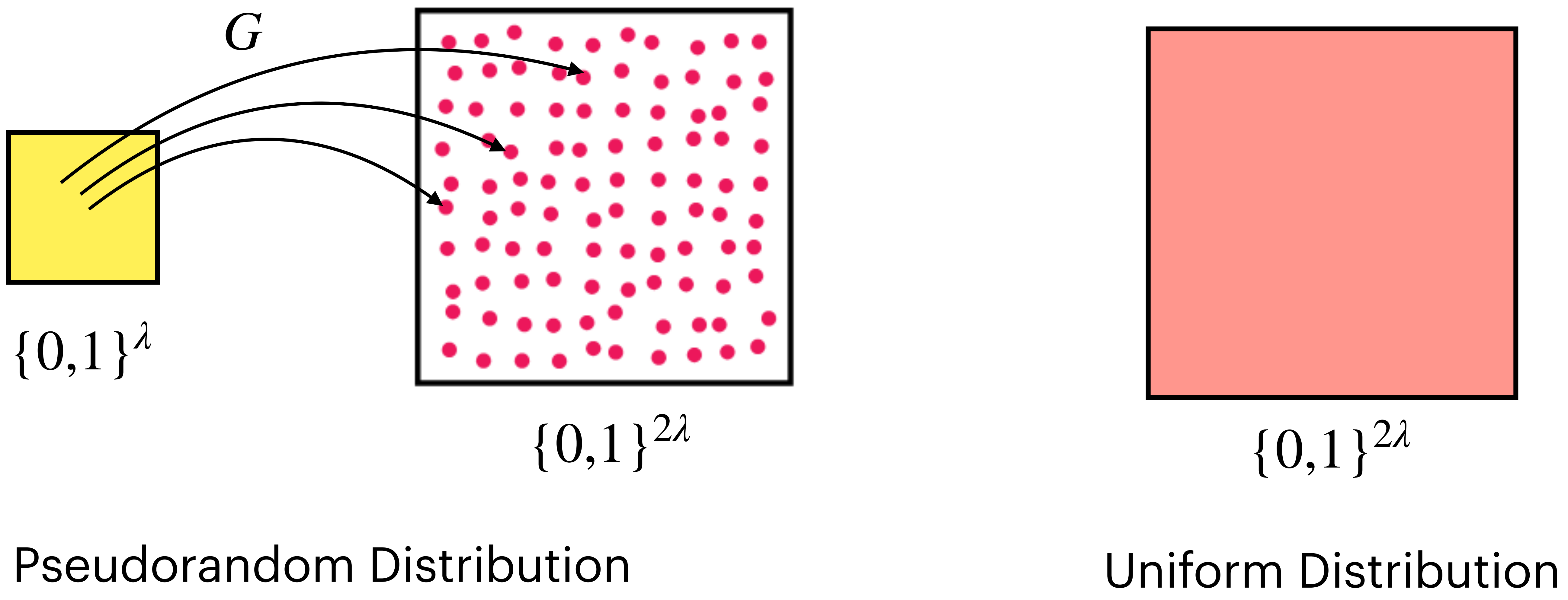
Consider a PRG G with λ-bit stretch.



- Relatively, the PRG's output space is tiny. The output space makes up a negligible fraction (i.e, $2^{-\lambda}$) of all 2λ-bit strings.
  - How can an efficient adversary break PRG security with negligible probability?
  - How can a brute-force attack break PRG security?

Why Pseudorandom?

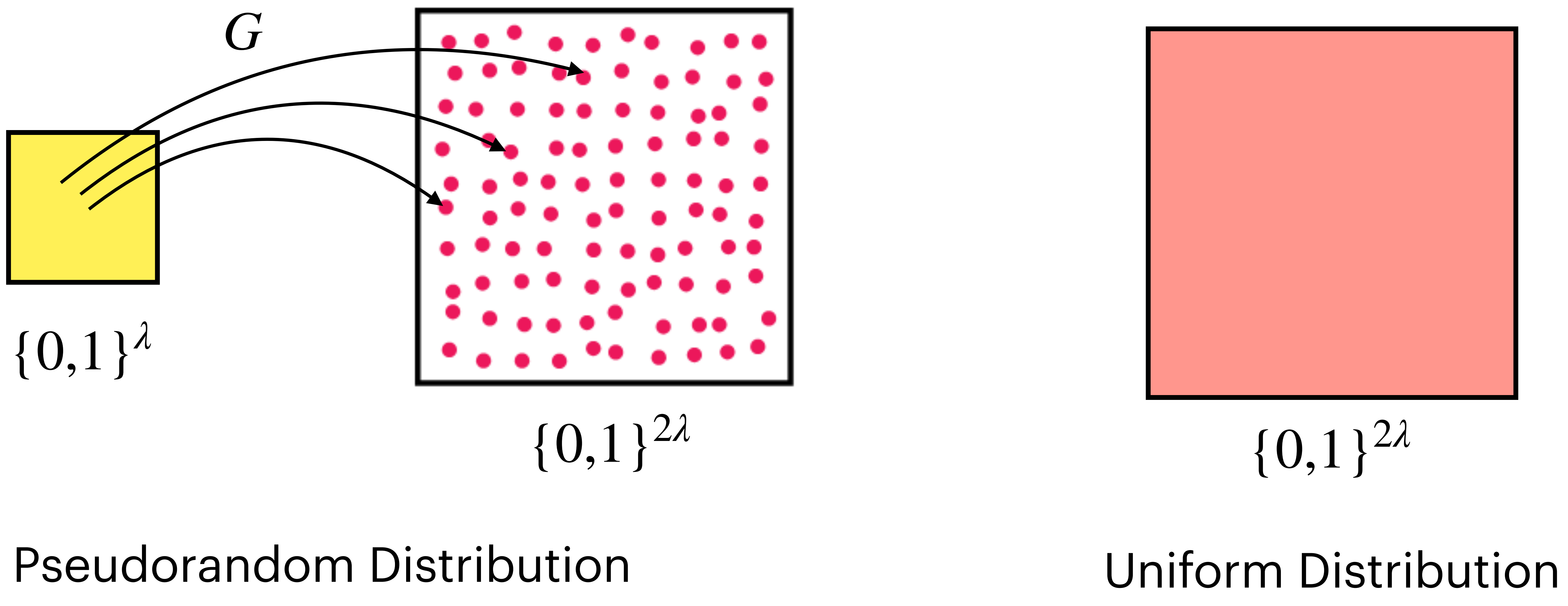
Consider a PRG G with λ-bit stretch.



- Relatively, the PRG's output space is tiny. The output space makes up a negligible fraction (i.e, $2^{-\lambda}$) of all 2λ-bit strings.
    - How can an efficient adversary break PRG security with negligible probability?
    - How can a brute-force attack break PRG security?
- From an absolute perspective, the PRGs output space is exponentially large!

Why Pseudorandom?

Consider a PRG G with λ-bit stretch.



- Relatively, the PRG's output space is tiny. The output space makes up a negligible fraction (i.e, $2^{-\lambda}$) of all 2λ-bit strings.
  - How can an efficient adversary break PRG security with negligible probability?
  - How can a brute-force attack break PRG security?
- From an absolute perspective, the PRGs output space is exponentially large!

Any poly-time algorithm that requires a long random string can instead be fed pseudorandomness generated using a short random seed.

Pseudorandom OTP

One-Time Pad

Let λ be the security parameter and $\ell(\lambda)$ be a polynomial.

- KeyGen(1^λ): $k \xleftarrow{\$} \{0,1\}^{\ell(\lambda)}$.
- Enc(k, m): ct $:= k \oplus m$.
- Dec(k, ct): $m := k \oplus$ ct.

Pseudorandom OTP

Pseudorandom One-Time Pad

Let λ be the security parameter and $\ell(\lambda)$ be a polynomial. Let G be a PRG with stretch $\ell(\lambda) - \lambda$.

- KeyGen(1^λ): $k \xleftarrow{\$} \{0,1\}^{\ell(\lambda)}$.
- Enc(k, m): ct $:= k \oplus m$.
- Dec(k, ct): $m := k \oplus$ ct.

Pseudorandom OTP

Pseudorandom One-Time Pad

Let λ be the security parameter and $\ell(\lambda)$ be a polynomial. Let G be a PRG with stretch $\ell(\lambda) - \lambda$.

- KeyGen(1^λ): $k \xleftarrow{\$} \{0,1\}^\lambda$.
- Enc(k, m): ct $:= k \oplus m$.
- Dec(k, ct): $m := k \oplus \text{ct}$.

Pseudorandom OTP

Pseudorandom One-Time Pad

Let λ be the security parameter and $\ell(\lambda)$ be a polynomial. Let G be a PRG with stretch $\ell(\lambda) - \lambda$.

- KeyGen(1^λ): $k \xleftarrow{\$} \{0,1\}^\lambda$.
- Enc(k, m): ct $:= G(k) \oplus m$.
- Dec(k, ct): $m := k \oplus \text{ct}$.

Pseudorandom OTP

Pseudorandom One-Time Pad

Let λ be the security parameter and $\ell(\lambda)$ be a polynomial. Let G be a PRG with stretch $\ell(\lambda) - \lambda$.

- KeyGen(1^λ): $k \xleftarrow{\$} \{0,1\}^\lambda$.
- Enc(k, m): ct $:= G(k) \oplus m$.
- Dec(k, ct): $m := G(k) \oplus$ ct.

Pseudorandom OTP

Pseudorandom One-Time Pad

Let λ be the security parameter and $\ell(\lambda)$ be a polynomial. Let G be a PRG with stretch $\ell(\lambda) - \lambda$.

- KeyGen(1^λ): $k \xleftarrow{\$} \{0,1\}^\lambda$.
- Enc(k, m): ct $:= G(k) \oplus m$.
- Dec(k, ct): $m := G(k) \oplus$ ct.

Keys are shorter than the message: λ-bit keys and $\ell(\lambda)$-bit messages.

Pseudorandom OTP

Pseudorandom One-Time Pad

Let λ be the security parameter and $\ell(\lambda)$ be a polynomial. Let G be a PRG with stretch $\ell(\lambda) - \lambda$.

- KeyGen(1^λ): $k \xleftarrow{\$} \{0,1\}^\lambda$.
- Enc(k, m): ct $:= G(k) \oplus m$.
- Dec(k, ct): $m := G(k) \oplus$ ct.

Keys are shorter than the message: λ-bit keys and $\ell(\lambda)$-bit messages.

Security? Is it perfectly secure?

One-Time Computational Security

One-Time Computational Security

An encryption scheme with message length $\ell := \ell(\lambda)$ is one-time computationally secure if $\forall m_0, m_1 \in \{0,1\}^\ell$

One-Time Computational Security

One-Time Computational Security

An encryption scheme with message length $\ell := \ell(\lambda)$ is one-time computationally secure if $\forall m_0, m_1 \in \{0,1\}^\ell$

$$D_0 = \left\{ \mathsf{ct} : \begin{array}{c} k \leftarrow \mathsf{KeyGen}(1^\lambda) \\ \mathsf{ct} \leftarrow \mathsf{Enc}(k, m_0) \end{array} \right\} \quad \overset{c}{\approx} \quad D_1 = \left\{ \mathsf{ct} : \begin{array}{c} k \leftarrow \mathsf{KeyGen}(1^\lambda) \\ \mathsf{ct} \leftarrow \mathsf{Enc}(k, m_1) \end{array} \right\}$$

Security of Pseudorandom OTP

Theorem: Pseudorandom OTP is one-time computational secure.

Security of Pseudorandom OTP

Theorem: Pseudorandom OTP is one-time computational secure.

Intuition: We need to show that for any given $m_0, m_1 \in \{0,1\}^{\ell(\lambda)}$

$$\left\{ G(k) \oplus m_0 \ : \ k \xleftarrow{\$} \{0,1\}^\lambda \right\}$$

$$\stackrel{c}{\approx}$$

$$\left\{ G(k) \oplus m_1 \ : \ k \xleftarrow{\$} \{0,1\}^\lambda \right\}$$

Security of Pseudorandom OTP

Theorem: Pseudorandom OTP is one-time computational secure.

Intuition:

$$H_0 = \left\{ G(k) \oplus m_0 \ : \ k \xleftarrow{\$} \{0,1\}^\lambda \right\}$$

$$\left\{ G(k) \oplus m_1 \ : \ k \xleftarrow{\$} \{0,1\}^\lambda \right\}$$

Security of Pseudorandom OTP

Theorem: Pseudorandom OTP is one-time computational secure.

Intuition:

$$H_0 = \left\{ G(k) \oplus m_0 \ : \ k \stackrel{\$}{\leftarrow} \{0,1\}^{\lambda} \right\}$$

$$H_1 = \left\{ r \oplus m_0 \ : \ r \stackrel{\$}{\leftarrow} \{0,1\}^{\ell(\lambda)} \right\}$$

$$\left\{ G(k) \oplus m_1 \ : \ k \stackrel{\$}{\leftarrow} \{0,1\}^{\lambda} \right\}$$

Security of Pseudorandom OTP

Theorem: Pseudorandom OTP is one-time computational secure.

Intuition:

$$H_0 = \left\{ G(k) \oplus m_0 \ : \ k \xleftarrow{\$} \{0,1\}^\lambda \right\}$$

From security of PRG, we know that

$$\{G(k) : k \xleftarrow{\$} \{0,1\}^\lambda\} \quad \overset{c}{\approx} \quad \{r : r \xleftarrow{\$} \{0,1\}^{\ell(\lambda)}\}$$

$$H_1 = \left\{ r \oplus m_0 \ : \ r \xleftarrow{\$} \{0,1\}^{\ell(\lambda)} \right\}$$

$$\left\{ G(k) \oplus m_1 \ : \ k \xleftarrow{\$} \{0,1\}^\lambda \right\}$$

Security of Pseudorandom OTP

Theorem: Pseudorandom OTP is one-time computational secure.

Intuition:

$$H_0 = \left\{ G(k) \oplus m_0 \ : \ k \xleftarrow{\$} \{0,1\}^\lambda \right\}$$

From security of PRG, we know that

$$\{G(k) : k \xleftarrow{\$} \{0,1\}^\lambda\} \quad \overset{c}{\approx} \quad \{r : r \xleftarrow{\$} \{0,1\}^{\ell(\lambda)}\}$$

$$H_1 = \left\{ r \oplus m_0 \ : \ r \xleftarrow{\$} \{0,1\}^{\ell(\lambda)} \right\}$$

From closure property of computational indistinguishability, we get

$$\{G(k) \oplus m_0 : k \xleftarrow{\$} \{0,1\}^\lambda\} \quad \overset{c}{\approx} \quad \{r \oplus m_0 : r \xleftarrow{\$} \{0,1\}^{\ell(\lambda)}\}$$

$$\left\{ G(k) \oplus m_1 \ : \ k \xleftarrow{\$} \{0,1\}^\lambda \right\}$$

Security of Pseudorandom OTP

Theorem: Pseudorandom OTP is one-time computational secure.

Intuition:

$$H_0 = \left\{ G(k) \oplus m_0 \ : \ k \overset{\$}{\leftarrow} \{0,1\}^\lambda \right\}$$

$$H_0 \overset{c}{\approx} H_1$$

$$H_1 = \left\{ r \oplus m_0 \ : \ r \overset{\$}{\leftarrow} \{0,1\}^{\ell(\lambda)} \right\}$$

From security of PRG, we know that

$$\{G(k) : k \overset{\$}{\leftarrow} \{0,1\}^\lambda\} \quad \overset{c}{\approx} \quad \{r : r \overset{\$}{\leftarrow} \{0,1\}^{\ell(\lambda)}\}$$

From closure property of computational indistinguishability, we get

$$\{G(k) \oplus m_0 : k \overset{\$}{\leftarrow} \{0,1\}^\lambda\} \quad \overset{c}{\approx} \quad \{r \oplus m_0 : r \overset{\$}{\leftarrow} \{0,1\}^{\ell(\lambda)}\}$$

$$\left\{ G(k) \oplus m_1 \ : \ k \overset{\$}{\leftarrow} \{0,1\}^\lambda \right\}$$

Security of Pseudorandom OTP

Theorem: Pseudorandom OTP is one-time computational secure.

Intuition:

$$H_0 = \left\{ G(k) \oplus m_0 \ : \ k \xleftarrow{\$} \{0,1\}^{\lambda} \right\}$$

$$H_0 \overset{c}{\approx} H_1$$

$$H_1 = \left\{ r \oplus m_0 \ : \ r \xleftarrow{\$} \{0,1\}^{\ell(\lambda)} \right\}$$

$$\left\{ G(k) \oplus m_1 \ : \ k \xleftarrow{\$} \{0,1\}^{\lambda} \right\}$$

Security of Pseudorandom OTP

Theorem: Pseudorandom OTP is one-time computational secure.

Intuition:

$$H_0 = \left\{ G(k) \oplus m_0 \ : \ k \xleftarrow{\$} \{0,1\}^{\lambda} \right\}$$

$$H_0 \overset{c}{\approx} H_1$$

$$H_1 = \left\{ r \oplus m_0 \ : \ r \xleftarrow{\$} \{0,1\}^{\ell(\lambda)} \right\}$$

$$H_2 = \left\{ r \oplus m_1 \ : \ r \xleftarrow{\$} \{0,1\}^{\ell(\lambda)} \right\}$$

$$\left\{ G(k) \oplus m_1 \ : \ k \xleftarrow{\$} \{0,1\}^{\lambda} \right\}$$

Security of Pseudorandom OTP

Theorem: Pseudorandom OTP is one-time computational secure.

Intuition:

$$H_0 = \left\{ G(k) \oplus m_0 \ : \ k \xleftarrow{\$} \{0,1\}^{\lambda} \right\}$$

$$H_0 \overset{c}{\approx} H_1$$

$$H_1 = \left\{ r \oplus m_0 \ : \ r \xleftarrow{\$} \{0,1\}^{\ell(\lambda)} \right\}$$

OTP is perfectly secure.

$$H_2 = \left\{ r \oplus m_1 \ : \ r \xleftarrow{\$} \{0,1\}^{\ell(\lambda)} \right\}$$

$$\left\{ G(k) \oplus m_1 \ : \ k \xleftarrow{\$} \{0,1\}^{\lambda} \right\}$$

Security of Pseudorandom OTP

Theorem: Pseudorandom OTP is one-time computational secure.

Intuition:

$$H_0 = \left\{ G(k) \oplus m_0 \ : \ k \xleftarrow{\$} \{0,1\}^{\lambda} \right\}$$

$$H_0 \overset{c}{\approx} H_1$$

$$H_1 = \left\{ r \oplus m_0 \ : \ r \xleftarrow{\$} \{0,1\}^{\ell(\lambda)} \right\}$$

$$H_1 \equiv H_2$$

OTP is perfectly secure.

$$H_2 = \left\{ r \oplus m_1 \ : \ r \xleftarrow{\$} \{0,1\}^{\ell(\lambda)} \right\}$$

$$\left\{ G(k) \oplus m_1 \ : \ k \xleftarrow{\$} \{0,1\}^{\lambda} \right\}$$

Security of Pseudorandom OTP

Theorem: Pseudorandom OTP is one-time computational secure.

Intuition:

$$H_0 = \left\{ G(k) \oplus m_0 \ : \ k \xleftarrow{\$} \{0,1\}^\lambda \right\}$$

$$H_0 \overset{c}{\approx} H_1$$

$$H_1 = \left\{ r \oplus m_0 \ : \ r \xleftarrow{\$} \{0,1\}^{\ell(\lambda)} \right\}$$

$$H_1 \equiv H_2$$

$$H_2 = \left\{ r \oplus m_1 \ : \ r \xleftarrow{\$} \{0,1\}^{\ell(\lambda)} \right\}$$

$$H_3 = \left\{ G(k) \oplus m_1 \ : \ k \xleftarrow{\$} \{0,1\}^\lambda \right\}$$

Security of Pseudorandom OTP

Theorem: Pseudorandom OTP is one-time computational secure.

Intuition:

$$H_0 = \left\{ G(k) \oplus m_0 \;:\; k \xleftarrow{\$} \{0,1\}^{\lambda} \right\}$$

$$H_0 \overset{c}{\approx} H_1$$

$$H_1 = \left\{ r \oplus m_0 \;:\; r \xleftarrow{\$} \{0,1\}^{\ell(\lambda)} \right\}$$

$$H_1 \equiv H_2$$

$$H_2 = \left\{ r \oplus m_1 \;:\; r \xleftarrow{\$} \{0,1\}^{\ell(\lambda)} \right\}$$

$$H_2 \overset{c}{\approx} H_3$$

Similar to $H_0 \overset{c}{\approx} H_1$.

$$H_3 = \left\{ G(k) \oplus m_1 \;:\; k \xleftarrow{\$} \{0,1\}^{\lambda} \right\}$$

Security of Pseudorandom OTP

Theorem: Pseudorandom OTP is one-time computational secure.

Intuition:

$$H_0 = \left\{ G(k) \oplus m_0 \ : \ k \xleftarrow{\$} \{0,1\}^{\lambda} \right\}$$

$H_0 \overset{c}{\approx} H_1$

$$H_1 = \left\{ r \oplus m_0 \ : \ r \xleftarrow{\$} \{0,1\}^{\ell(\lambda)} \right\}$$

$H_1 \equiv H_2$

$$H_2 = \left\{ r \oplus m_1 \ : \ r \xleftarrow{\$} \{0,1\}^{\ell(\lambda)} \right\}$$

$H_2 \overset{c}{\approx} H_3$

$$H_3 = \left\{ G(k) \oplus m_1 \ : \ k \xleftarrow{\$} \{0,1\}^{\lambda} \right\}$$

$H_0 \overset{c}{\approx} H_3$ Hybrid Lemma

Security of Pseudorandom OTP

Theorem: Pseudorandom OTP is one-time computational secure.

Intuition:

$$H_0 = \left\{ G(k) \oplus m_0 \ : \ k \xleftarrow{\$} \{0,1\}^{\lambda} \right\}$$

$$H_0 \overset{c}{\approx} H_1$$

$$H_1 = \left\{ r \oplus m_0 \ : \ r \xleftarrow{\$} \{0,1\}^{\ell(\lambda)} \right\}$$

- How do we formally establish that $H_0 \overset{c}{\approx} H_1$?

Security of Pseudorandom OTP

Theorem: Pseudorandom OTP is one-time computational secure.

Intuition:

$$H_0 = \left\{ G(k) \oplus m_0 \ : \ k \xleftarrow{\$} \{0,1\}^{\lambda} \right\}$$

$$H_0 \overset{c}{\approx} H_1$$

$$H_1 = \left\{ r \oplus m_0 \ : \ r \xleftarrow{\$} \{0,1\}^{\ell(\lambda)} \right\}$$

- How do we formally establish that $H_0 \overset{c}{\approx} H_1$?
- **Goal:**

Security of Pseudorandom OTP

Theorem: Pseudorandom OTP is one-time computational secure.

Intuition:

$$H_0 = \left\{ G(k) \oplus m_0 \ : \ k \xleftarrow{\$} \{0,1\}^{\lambda} \right\}$$

$$H_0 \overset{c}{\approx} H_1$$

$$H_1 = \left\{ r \oplus m_0 \ : \ r \xleftarrow{\$} \{0,1\}^{\ell(\lambda)} \right\}$$

- How do we formally establish that $H_0 \overset{c}{\approx} H_1$?
- **Goal:**

  If no efficient A_{PRG} can distinguish $G(k)$ from r

Security of Pseudorandom OTP

Theorem: Pseudorandom OTP is one-time computational secure.

Intuition:

$$H_0 = \left\{ G(k) \oplus m_0 \ : \ k \stackrel{\$}{\leftarrow} \{0,1\}^\lambda \right\}$$

$$H_0 \stackrel{c}{\approx} H_1$$

$$H_1 = \left\{ r \oplus m_0 \ : \ r \stackrel{\$}{\leftarrow} \{0,1\}^{\ell(\lambda)} \right\}$$

- How do we formally establish that $H_0 \stackrel{c}{\approx} H_1$?
- **Goal:**
  If no efficient A_{PRG} can distinguish $G(k)$ from r
  then
  no efficient A can distinguish between H_0 and H_1.

Security of Pseudorandom OTP

Theorem: Pseudorandom OTP is one-time computational secure.

Intuition:

$$H_0 = \left\{ G(k) \oplus m_0 \ : \ k \xleftarrow{\$} \{0,1\}^\lambda \right\}$$

$$H_0 \overset{c}{\approx} H_1$$

$$H_1 = \left\{ r \oplus m_0 \ : \ r \xleftarrow{\$} \{0,1\}^{\ell(\lambda)} \right\}$$

- How do we formally establish that $H_0 \overset{c}{\approx} H_1$?
- **Goal:**

  If no efficient A_{PRG} can distinguish $G(k)$ from r

  then

  no efficient A can distinguish between H_0 and H_1.

- **Contrapositive:**

Security of Pseudorandom OTP

Theorem: Pseudorandom OTP is one-time computational secure.

Intuition:

$$H_0 = \left\{ G(k) \oplus m_0 \ : \ k \xleftarrow{\$} \{0,1\}^{\lambda} \right\}$$

$$H_0 \overset{c}{\approx} H_1$$

$$H_1 = \left\{ r \oplus m_0 \ : \ r \xleftarrow{\$} \{0,1\}^{\ell(\lambda)} \right\}$$

- How do we formally establish that $H_0 \overset{c}{\approx} H_1$?
- **Goal:**
  If no efficient A_{PRG} can distinguish $G(k)$ from r
  then
  no efficient A can distinguish between H_0 and H_1.
- **Contrapositive:**
  If an efficient A can distinguish between H_0 and H_1
  then
  there there is an efficient A_{PRG} that can distinguish between $G(k)$ and r.

Security of Pseudorandom OTP

Theorem: Pseudorandom OTP is one-time computational secure.

Intuition:

$$H_0 = \left\{ G(k) \oplus m_0 \ : \ k \xleftarrow{\$} \{0,1\}^{\lambda} \right\}$$

$$H_0 \overset{c}{\approx} H_1$$

$$H_1 = \left\{ r \oplus m_0 \ : \ r \xleftarrow{\$} \{0,1\}^{\ell(\lambda)} \right\}$$

- How do we formally establish that $H_0 \overset{c}{\approx} H_1$?
- **Goal:**
  If no efficient A_{PRG} can distinguish $G(k)$ from r
  then
  no efficient A can distinguish between H_0 and H_1.
- **Contrapositive:**
  If an efficient A can distinguish between H_0 and H_1
  then
  there there is an efficient A_{PRG} that can distinguish between $G(k)$ and r.
  - But existence of such a A_{PRG} contradicts PRG security.

Security of Pseudorandom OTP

Theorem: Pseudorandom OTP is one-time computational secure.

Intuition:

$$H_0 = \left\{ G(k) \oplus m_0 \ : \ k \xleftarrow{\$} \{0,1\}^\lambda \right\}$$

$$H_0 \overset{c}{\approx} H_1$$

$$H_1 = \left\{ r \oplus m_0 \ : \ r \xleftarrow{\$} \{0,1\}^{\ell(\lambda)} \right\}$$

- How do we formally establish that $H_0 \overset{c}{\approx} H_1$?
- **Goal:**
  If no efficient A_{PRG} can distinguish $G(k)$ from r
  then
  no efficient A can distinguish between H_0 and H_1.
- **Contrapositive:**
  If an efficient A can distinguish between H_0 and H_1
  then
  there there is an efficient A_{PRG} that can distinguish between $G(k)$ and r.
  - But existence of such a A_{PRG} contradicts PRG security.
  - Thus, such an efficient A cannot exist.

Security of Pseudorandom OTP

Theorem: Pseudorandom OTP is one-time computational secure.

Intuition:

$$H_0 = \left\{ G(k) \oplus m_0 \ : \ k \xleftarrow{\$} \{0,1\}^\lambda \right\}$$

$$H_0 \overset{c}{\approx} H_1$$

$$H_1 = \left\{ r \oplus m_0 \ : \ r \xleftarrow{\$} \{0,1\}^{\ell(\lambda)} \right\}$$

We will use A to build A_{PRG}.

- How do we formally establish that $H_0 \overset{c}{\approx} H_1$?
- **Goal:**
  If no efficient A_{PRG} can distinguish $G(k)$ from r
  then
  no efficient A can distinguish between H_0 and H_1.
- **Contrapositive:**
  If an efficient A can distinguish between H_0 and H_1
  then
  there there is an efficient A_{PRG} that can distinguish between $G(k)$ and r.
  - But existence of such a A_{PRG} contradicts PRG security.
  - Thus, such an efficient A cannot exist.

Security of Pseudorandom OTP

Theorem: Pseudorandom OTP is one-time computational secure.

Intuition:

$$H_0 = \left\{ G(k) \oplus m_0 \ : \ k \xleftarrow{\$} \{0,1\}^\lambda \right\}$$

$$H_0 \overset{c}{\approx} H_1$$

$$H_1 = \left\{ r \oplus m_0 \ : \ r \xleftarrow{\$} \{0,1\}^{\ell(\lambda)} \right\}$$

We will use A to build A_{PRG}.

Reduction: Use a solution to one problem to solve another related problem.

In crypto, reductions are the standard tool to "transfer" the hardness of one problem to another.

- How do we formally establish that $H_0 \overset{c}{\approx} H_1$?
- **Goal:**
  If no efficient A_{PRG} can distinguish $G(k)$ from r
  then
  no efficient A can distinguish between H_0 and H_1.
- **Contrapositive:**
  If an efficient A can distinguish between H_0 and H_1
  then
  there there is an efficient A_{PRG} that can distinguish between $G(k)$ and r.
  - But existence of such a A_{PRG} contradicts PRG security.
  - Thus, such an efficient A cannot exist.

Security of Pseudorandom OTP: Reduction

Security of Pseudorandom OTP: Reduction

Claim: If G is a PRG then $H_0 = \left\{ G(k) \oplus m_0 \ : \ k \stackrel{\$}{\leftarrow} \{0,1\}^{\lambda} \right\} \stackrel{c}{\approx} H_1 = \left\{ r \oplus m_0 \ : \ r \stackrel{\$}{\leftarrow} \{0,1\}^{\ell(\lambda)} \right\}.$

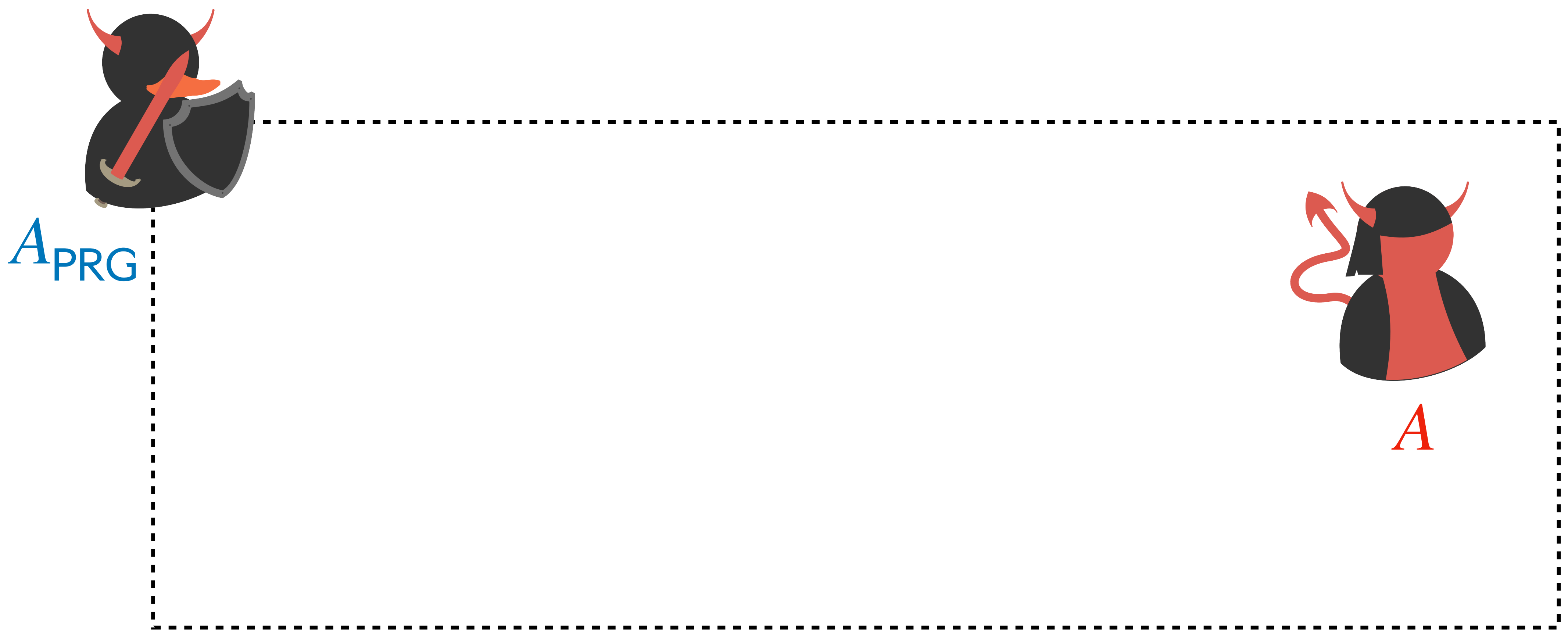
Security of Pseudorandom OTP: Reduction

Claim: If G is a PRG then $H_0 = \left\{ G(k) \oplus m_0 \;:\; k \xleftarrow{\$} \{0,1\}^{\lambda} \right\} \overset{c}{\approx} H_1 = \left\{ r \oplus m_0 \;:\; r \xleftarrow{\$} \{0,1\}^{\ell(\lambda)} \right\}$.

- We are given an adversary A that distinguishes between H_0 and H_1 with probability ϵ.

Security of Pseudorandom OTP: Reduction

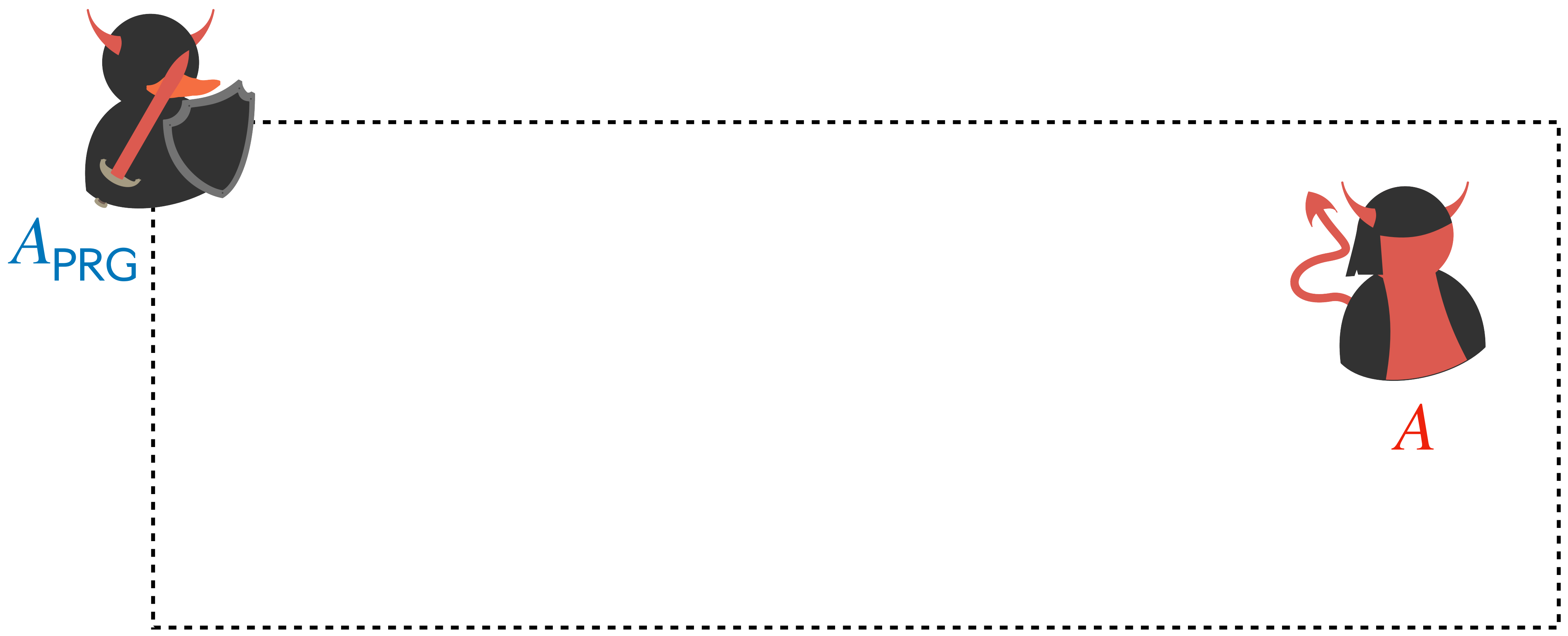
Claim: If G is a PRG then $H_0 = \left\{ G(k) \oplus m_0 \ : \ k \xleftarrow{\$} \{0,1\}^{\lambda} \right\} \overset{c}{\approx} H_1 = \left\{ r \oplus m_0 \ : \ r \xleftarrow{\$} \{0,1\}^{\ell(\lambda)} \right\}$.



- We are given an adversary A that distinguishes between H_0 and H_1 with probability ϵ.
- We will construct A_{PRG} (using A) that distinguishes between $G(k)$ and r with probability ϵ_{PRG}.

Security of Pseudorandom OTP: Reduction

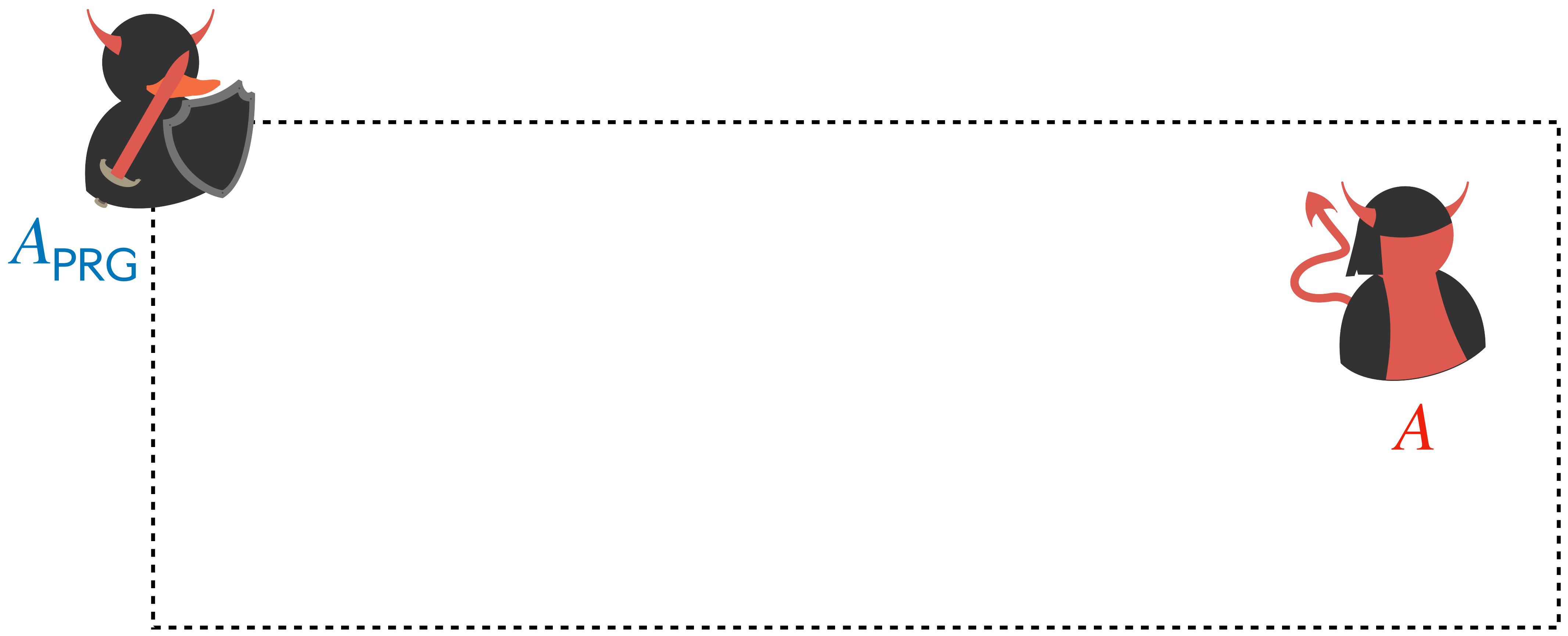
Claim: If G is a PRG then $H_0 = \left\{ G(k) \oplus m_0 \ : \ k \xleftarrow{\$} \{0,1\}^{\lambda} \right\} \overset{c}{\approx} H_1 = \left\{ r \oplus m_0 \ : \ r \xleftarrow{\$} \{0,1\}^{\ell(\lambda)} \right\}$.



- We are given an adversary A that distinguishes between H_0 and H_1 with probability ϵ.
- We will construct A_{PRG} (using A) that distinguishes between $G(k)$ and r with probability ϵ_{PRG}.
  - We will show that $\epsilon_{\mathsf{PRG}} = \epsilon$.

Security of Pseudorandom OTP: Reduction

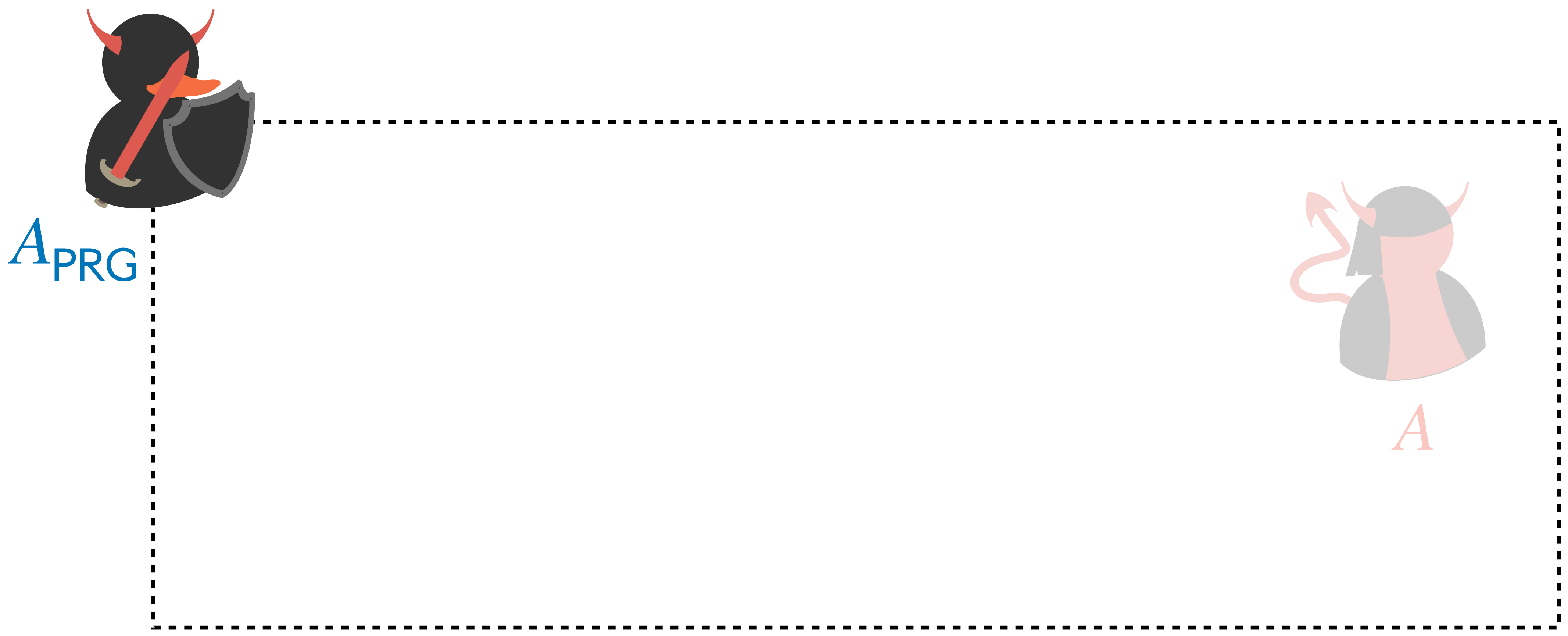
Claim: If G is a PRG then $H_0 = \left\{ G(k) \oplus m_0 \ : \ k \xleftarrow{\$} \{0,1\}^\lambda \right\} \overset{c}{\approx} H_1 = \left\{ r \oplus m_0 \ : \ r \xleftarrow{\$} \{0,1\}^{\ell(\lambda)} \right\}$.



- We are given an adversary A that distinguishes between H_0 and H_1 with probability ϵ.
- We will construct A_{PRG} (using A) that distinguishes between $G(k)$ and r with probability ϵ_{PRG}.
  - We will show that $\epsilon_{\text{PRG}} = \epsilon$.
  - Since ϵ_{PRG} is negligible due to PRG security, ϵ is also negligible.

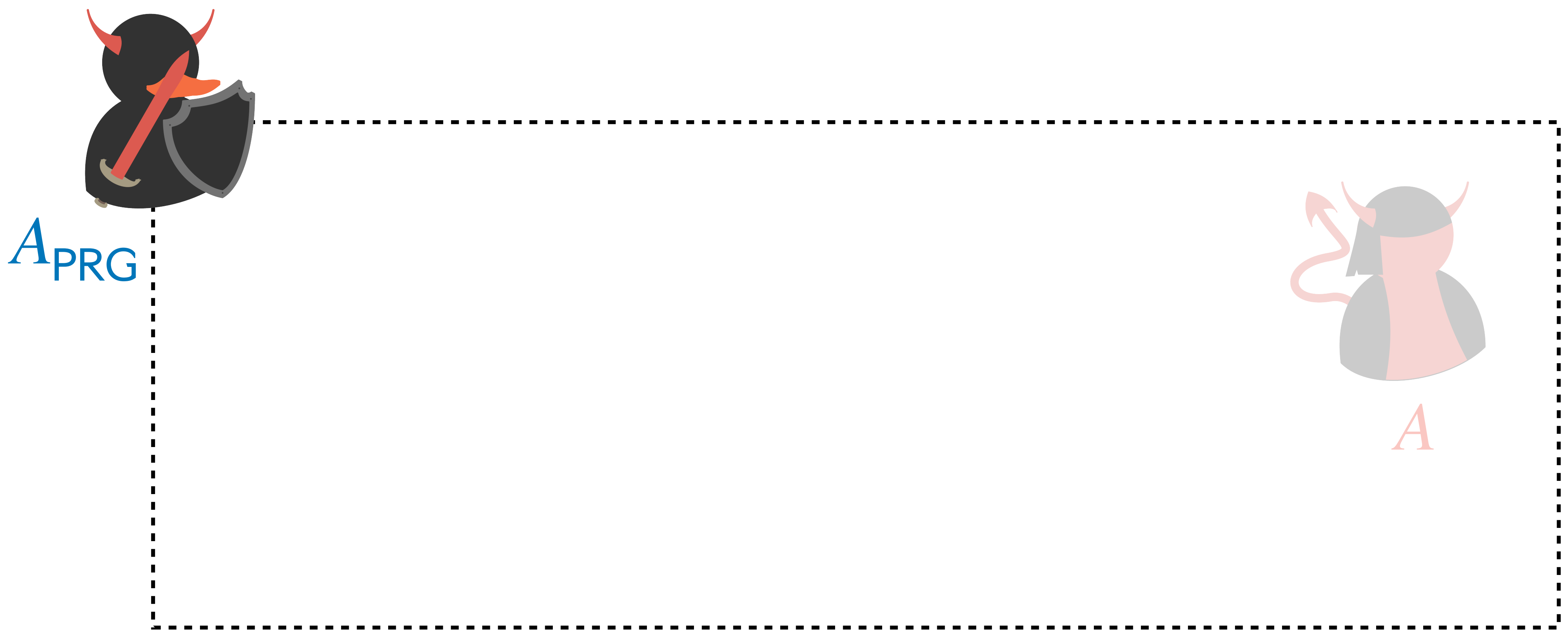
Security of Pseudorandom OTP: Reduction

Claim: If G is a PRG then $H_0 = \left\{ G(k) \oplus m_0 \ : \ k \xleftarrow{\$} \{0,1\}^{\lambda} \right\} \overset{c}{\approx} H_1 = \left\{ r \oplus m_0 \ : \ r \xleftarrow{\$} \{0,1\}^{\ell(\lambda)} \right\}$.



Security of Pseudorandom OTP: Reduction

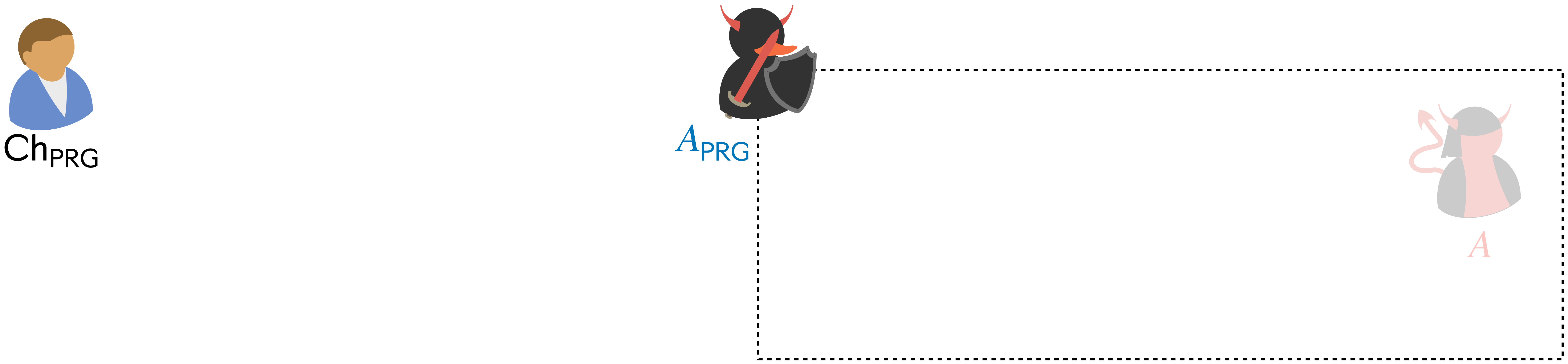
Claim: If G is a PRG then $H_0 = \left\{ G(k) \oplus m_0 \ : \ k \xleftarrow{\$} \{0,1\}^{\lambda} \right\} \overset{c}{\approx} H_1 = \left\{ r \oplus m_0 \ : \ r \xleftarrow{\$} \{0,1\}^{\ell(\lambda)} \right\}.$



- A_{PRG} is playing the PRG indistinguishability game.

Security of Pseudorandom OTP: Reduction

Claim: If G is a PRG then $H_0 = \left\{ G(k) \oplus m_0 \ :\ k \xleftarrow{\$} \{0,1\}^{\lambda} \right\} \overset{c}{\approx} H_1 = \left\{ r \oplus m_0 \ :\ r \xleftarrow{\$} \{0,1\}^{\ell(\lambda)} \right\}$.



- A_{PRG} is playing the PRG indistinguishability game.

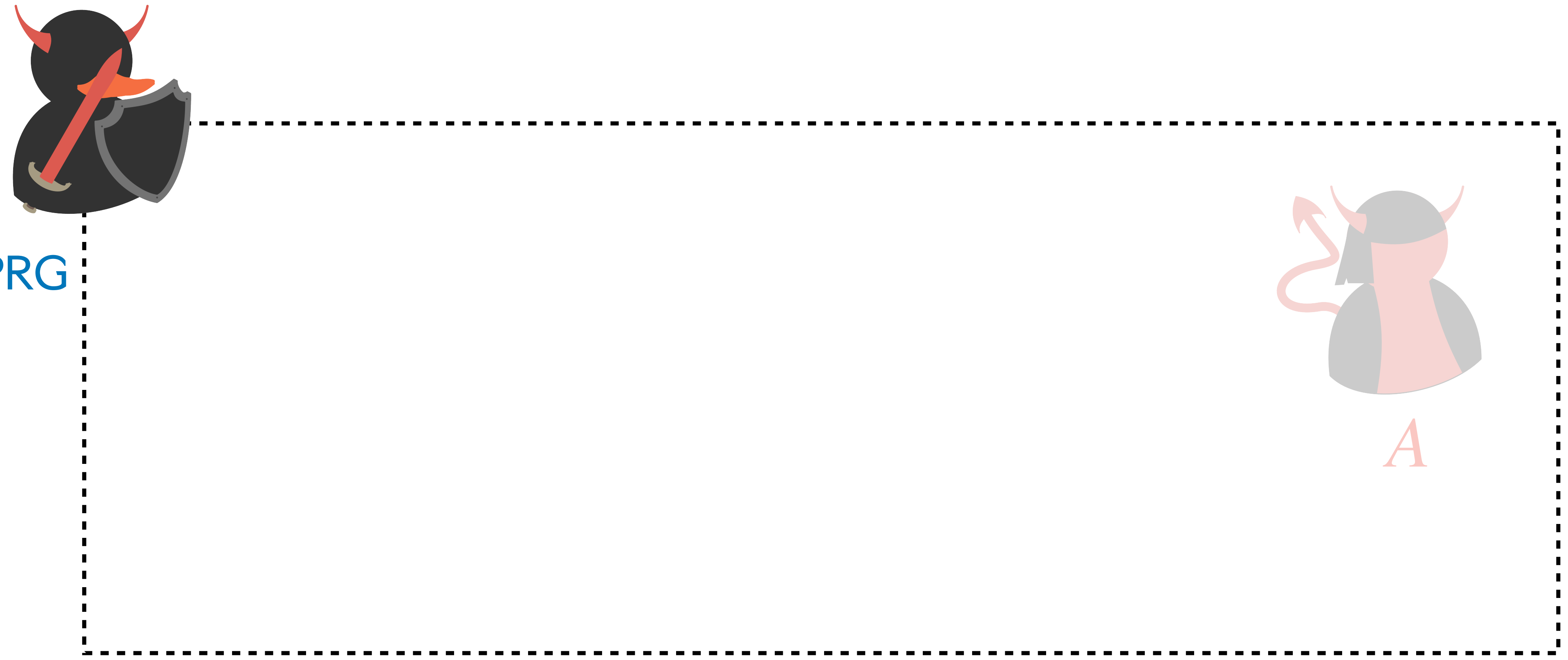
Security of Pseudorandom OTP: Reduction

Claim: If G is a PRG then $H_0 = \left\{ G(k) \oplus m_0 \ : \ k \xleftarrow{\$} \{0,1\}^{\lambda} \right\} \overset{c}{\approx} H_1 = \left\{ r \oplus m_0 \ : \ r \xleftarrow{\$} \{0,1\}^{\ell(\lambda)} \right\}$.

$k \xleftarrow{\$} \{0,1\}^{\lambda}$

$r_0 := G(k)$

$\mathsf{Ch}_{\mathsf{PRG}}$



A_{PRG}

A

- A_{PRG} is playing the PRG indistinguishability game.

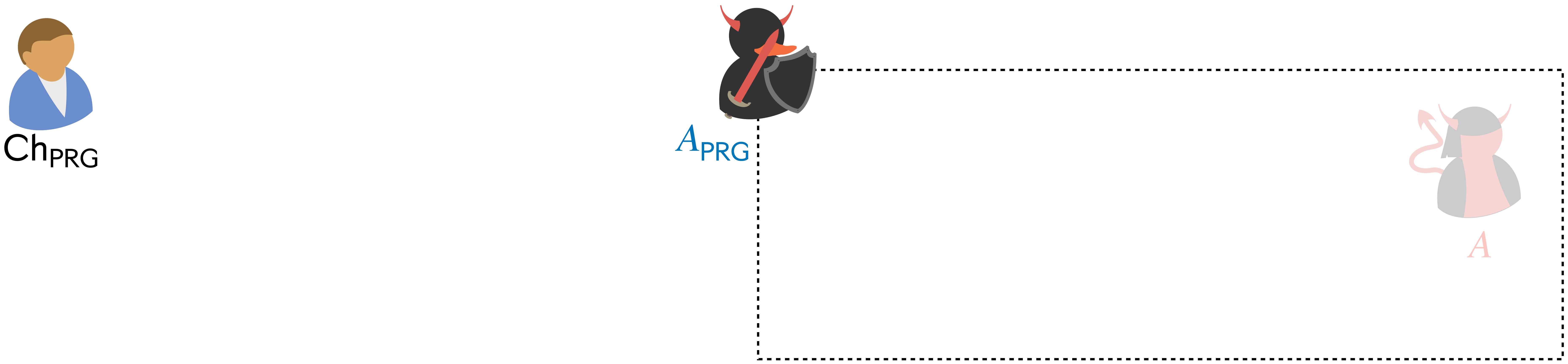
Security of Pseudorandom OTP: Reduction

Claim: If G is a PRG then $H_0 = \left\{ G(k) \oplus m_0 \;:\; k \xleftarrow{\$} \{0,1\}^{\lambda} \right\} \overset{c}{\approx} H_1 = \left\{ r \oplus m_0 \;:\; r \xleftarrow{\$} \{0,1\}^{\ell(\lambda)} \right\}$.

$k \xleftarrow{\$} \{0,1\}^{\lambda}$

$r_0 := G(k)$

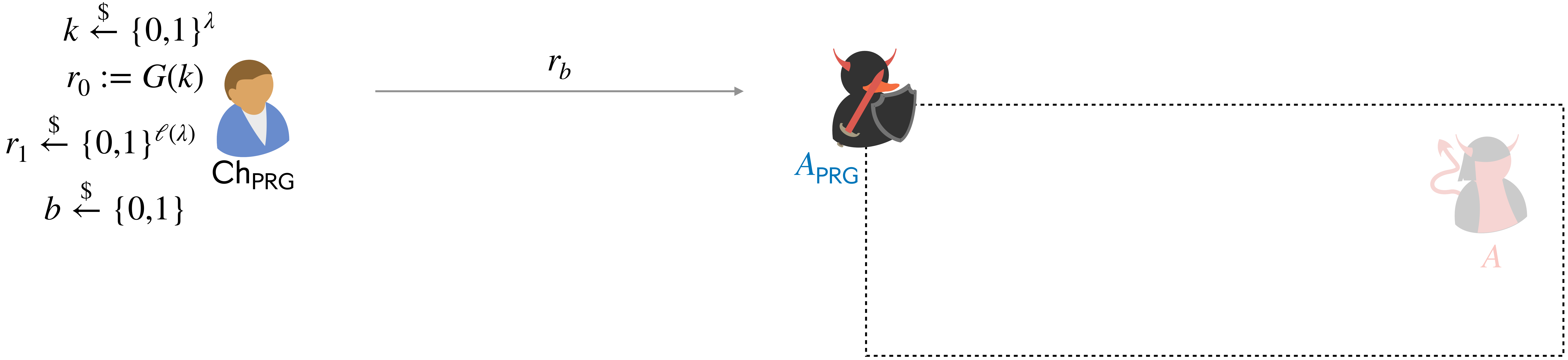
$r_1 \xleftarrow{\$} \{0,1\}^{\ell(\lambda)}$



- A_{PRG} is playing the PRG indistinguishability game.

Security of Pseudorandom OTP: Reduction

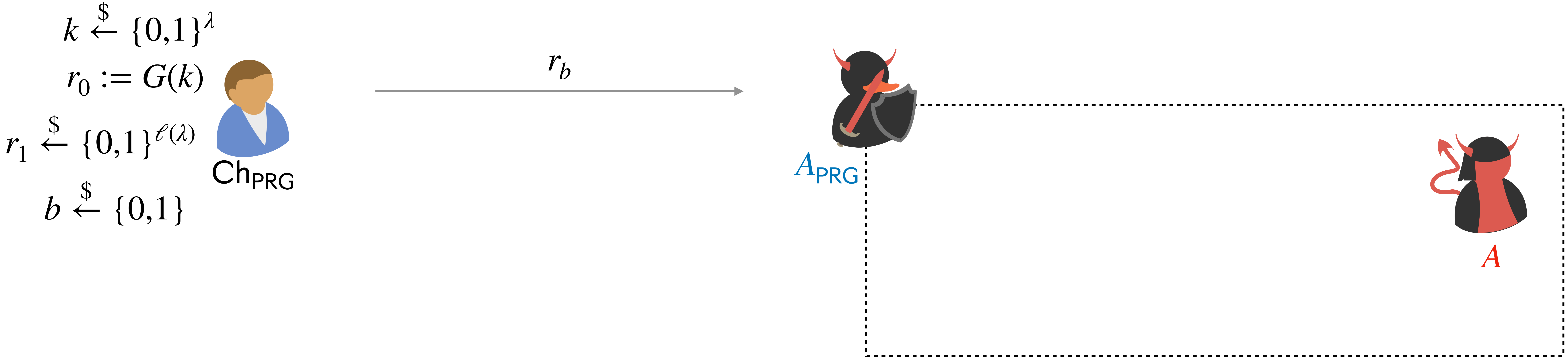
Claim: If G is a PRG then $H_0 = \left\{ G(k) \oplus m_0 \ : \ k \xleftarrow{\$} \{0,1\}^\lambda \right\} \overset{c}{\approx} H_1 = \left\{ r \oplus m_0 \ : \ r \xleftarrow{\$} \{0,1\}^{\ell(\lambda)} \right\}.$



- A_{PRG} is playing the PRG indistinguishability game.

Security of Pseudorandom OTP: Reduction

Claim: If G is a PRG then $H_0 = \left\{ G(k) \oplus m_0 \; : \; k \xleftarrow{\$} \{0,1\}^\lambda \right\} \overset{c}{\approx} H_1 = \left\{ r \oplus m_0 \; : \; r \xleftarrow{\$} \{0,1\}^{\ell(\lambda)} \right\}$.



- A_{PRG} is playing the PRG indistinguishability game.

Security of Pseudorandom OTP: Reduction

Claim: If G is a PRG then $H_0 = \left\{ G(k) \oplus m_0 \ : \ k \xleftarrow{\$} \{0,1\}^{\lambda} \right\} \overset{c}{\approx} H_1 = \left\{ r \oplus m_0 \ : \ r \xleftarrow{\$} \{0,1\}^{\ell(\lambda)} \right\}$.

- A_{PRG} is playing the PRG indistinguishability game.
- A is playing the indistinguishability game between H_0 and H_1.

Security of Pseudorandom OTP: Reduction

Claim: If G is a PRG then $H_0 = \left\{ G(k) \oplus m_0 \ : \ k \xleftarrow{\$} \{0,1\}^\lambda \right\} \overset{c}{\approx} H_1 = \left\{ r \oplus m_0 \ : \ r \xleftarrow{\$} \{0,1\}^{\ell(\lambda)} \right\}$.

- A_{PRG} is playing the PRG indistinguishability game.
- A is playing the indistinguishability game between H_0 and H_1.

Security of Pseudorandom OTP: Reduction

Claim: If G is a PRG then $H_0 = \left\{ G(k) \oplus m_0 \ : \ k \xleftarrow{\$} \{0,1\}^\lambda \right\} \overset{c}{\approx} H_1 = \left\{ r \oplus m_0 \ : \ r \xleftarrow{\$} \{0,1\}^{\ell(\lambda)} \right\}$.

- A_{PRG} is playing the PRG indistinguishability game.
- A is playing the indistinguishability game between H_0 and H_1.
  - We should construct A_{PRG} in such a way that A thinks its talking to Ch_{H_0,H_1}.

Security of Pseudorandom OTP: Reduction

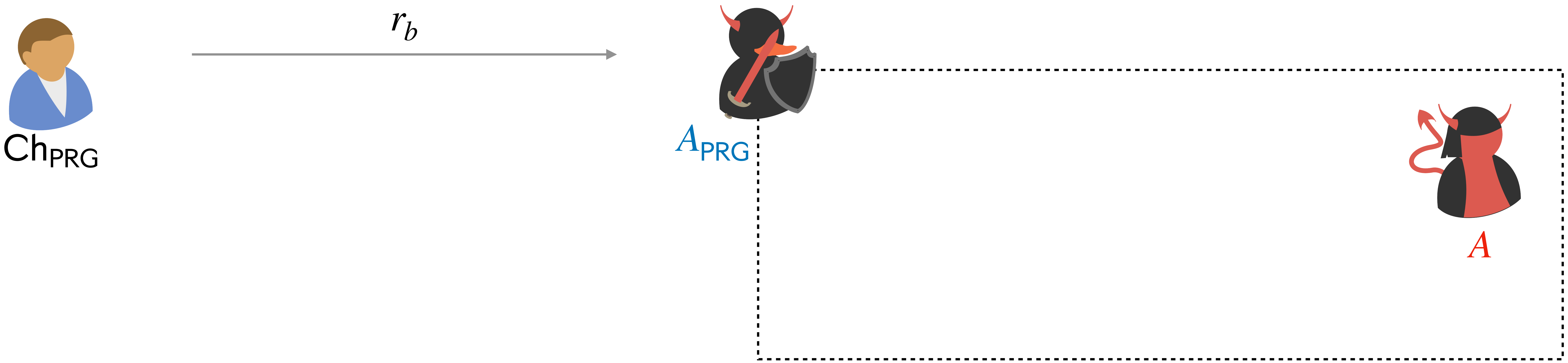
Claim: If G is a PRG then $H_0 = \left\{ G(k) \oplus m_0 \ : \ k \xleftarrow{\$} \{0,1\}^\lambda \right\} \overset{c}{\approx} H_1 = \left\{ r \oplus m_0 \ : \ r \xleftarrow{\$} \{0,1\}^{\ell(\lambda)} \right\}$.

$k \xleftarrow{\$} \{0,1\}^\lambda$

$r_0 := G(k)$

$r_1 \xleftarrow{\$} \{0,1\}^{\ell(\lambda)}$

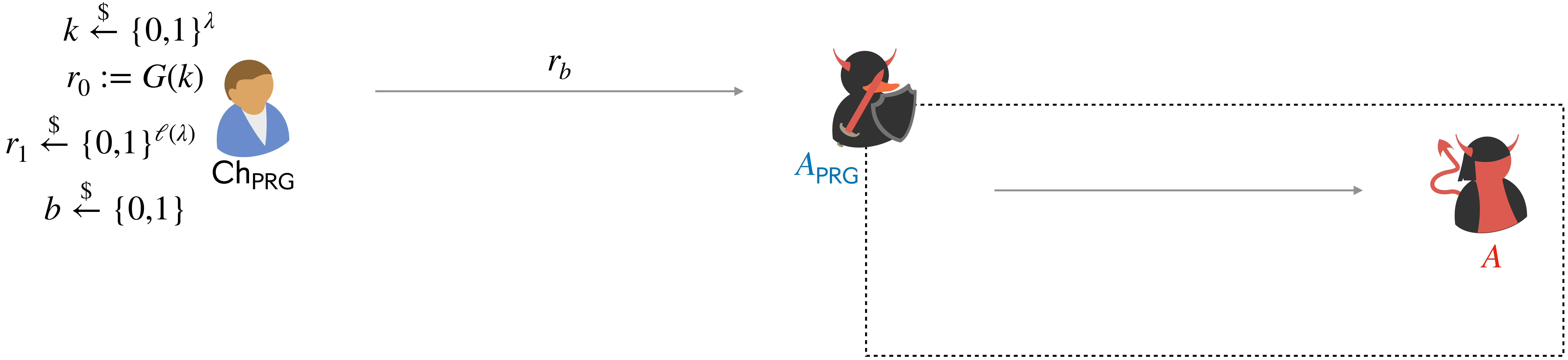
$b \xleftarrow{\$} \{0,1\}$



- A_{PRG} is playing the PRG indistinguishability game.
- A is playing the indistinguishability game between H_0 and H_1.
  - We should construct A_{PRG} in such a way that A thinks its talking to Ch_{H_0,H_1}.

Security of Pseudorandom OTP: Reduction

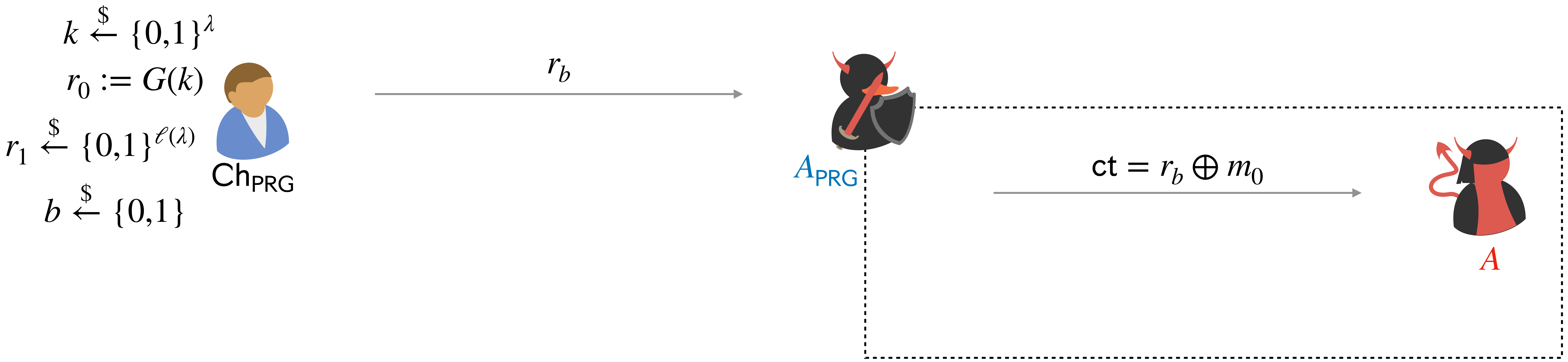
Claim: If G is a PRG then $H_0 = \left\{ G(k) \oplus m_0 \;:\; k \xleftarrow{\$} \{0,1\}^\lambda \right\} \overset{c}{\approx} H_1 = \left\{ r \oplus m_0 \;:\; r \xleftarrow{\$} \{0,1\}^{\ell(\lambda)} \right\}$.



- A_{PRG} is playing the PRG indistinguishability game.
- A is playing the indistinguishability game between H_0 and H_1.
  - We should construct A_{PRG} in such a way that A thinks its talking to Ch_{H_0,H_1}.

Security of Pseudorandom OTP: Reduction

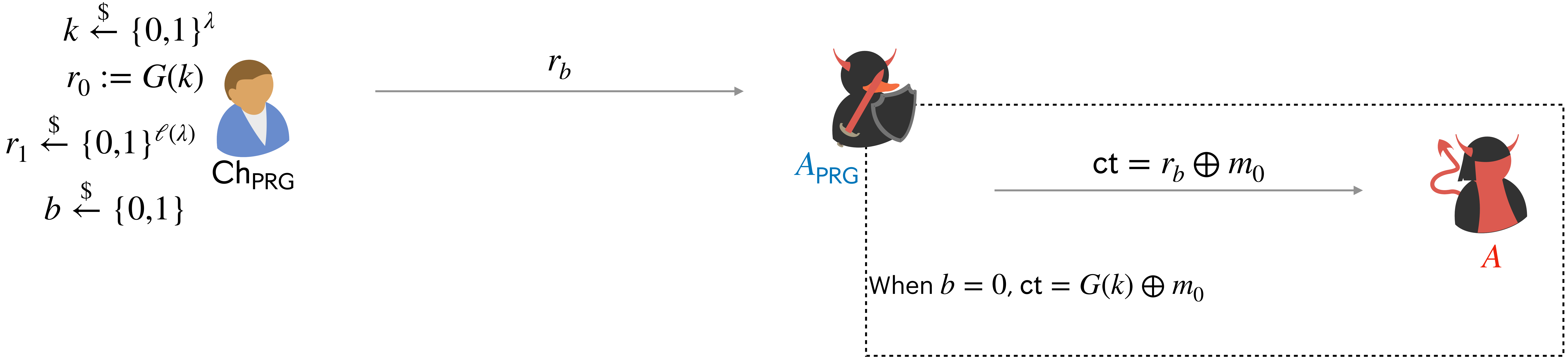
Claim: If G is a PRG then $H_0 = \left\{ G(k) \oplus m_0 \ : \ k \xleftarrow{\$} \{0,1\}^\lambda \right\} \overset{c}{\approx} H_1 = \left\{ r \oplus m_0 \ : \ r \xleftarrow{\$} \{0,1\}^{\ell(\lambda)} \right\}$.



- A_{PRG} is playing the PRG indistinguishability game.
- A is playing the indistinguishability game between H_0 and H_1.
    - We should construct A_{PRG} in such a way that A thinks its talking to Ch_{H_0,H_1}.

Security of Pseudorandom OTP: Reduction

Claim: If G is a PRG then $H_0 = \left\{ G(k) \oplus m_0 \ : \ k \xleftarrow{\$} \{0,1\}^\lambda \right\} \overset{c}{\approx} H_1 = \left\{ r \oplus m_0 \ : \ r \xleftarrow{\$} \{0,1\}^{\ell(\lambda)} \right\}$.



- A_{PRG} is playing the PRG indistinguishability game.
- A is playing the indistinguishability game between H_0 and H_1.
    - We should construct A_{PRG} in such a way that A thinks its talking to Ch_{H_0,H_1}.

Security of Pseudorandom OTP: Reduction

Claim: If G is a PRG then $H_0 = \left\{ G(k) \oplus m_0 \ : \ k \xleftarrow{\$} \{0,1\}^\lambda \right\} \overset{c}{\approx} H_1 = \left\{ r \oplus m_0 \ : \ r \xleftarrow{\$} \{0,1\}^{\ell(\lambda)} \right\}$.

$k \xleftarrow{\$} \{0,1\}^\lambda$

$r_0 := G(k)$

$r_1 \xleftarrow{\$} \{0,1\}^{\ell(\lambda)}$

$b \xleftarrow{\$} \{0,1\}$

$\mathsf{Ch}_{\mathsf{PRG}}$

r_b →

A_{PRG}

$\mathsf{ct} = r_b \oplus m_0$ →

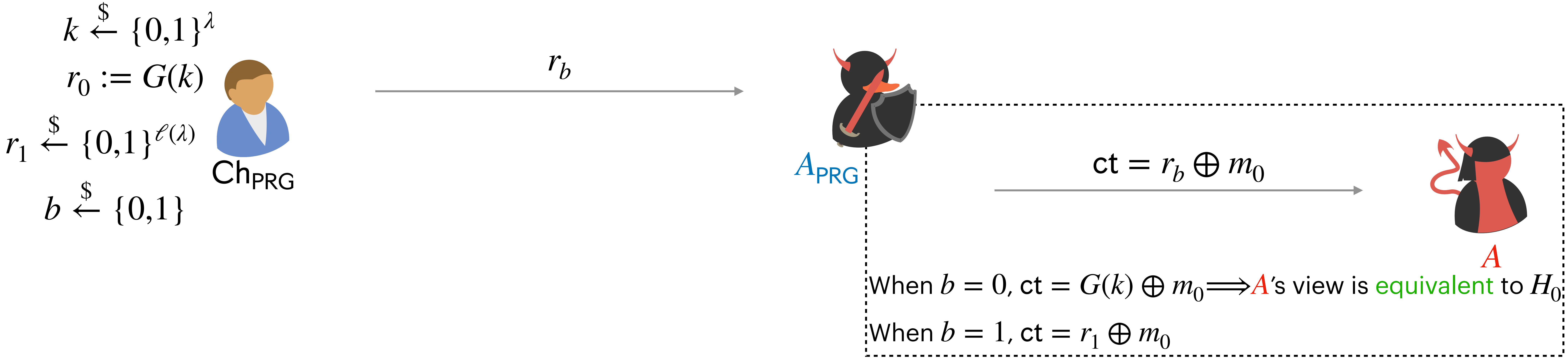
A

When $b = 0$, $\mathsf{ct} = G(k) \oplus m_0 \Longrightarrow A$'s view is equivalent to H_0

- A_{PRG} is playing the PRG indistinguishability game.
- A is playing the indistinguishability game between H_0 and H_1.
  - We should construct A_{PRG} in such a way that A thinks its talking to Ch_{H_0,H_1}.

Security of Pseudorandom OTP: Reduction

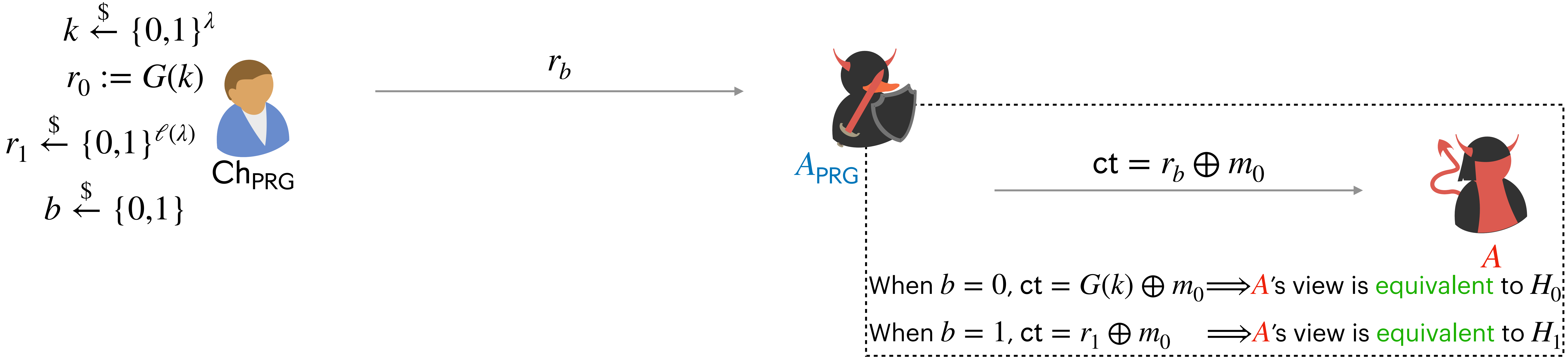
Claim: If G is a PRG then $H_0 = \left\{ G(k) \oplus m_0 \ : \ k \xleftarrow{\$} \{0,1\}^\lambda \right\} \overset{c}{\approx} H_1 = \left\{ r \oplus m_0 \ : \ r \xleftarrow{\$} \{0,1\}^{\ell(\lambda)} \right\}$.



- A_{PRG} is playing the PRG indistinguishability game.
- A is playing the indistinguishability game between H_0 and H_1.
  - We should construct A_{PRG} in such a way that A thinks its talking to Ch_{H_0,H_1}.

Security of Pseudorandom OTP: Reduction

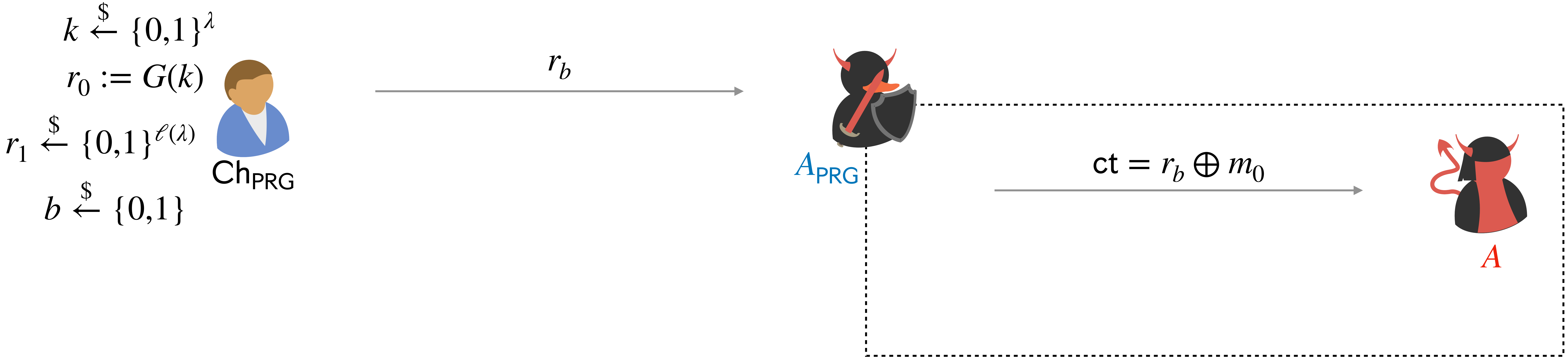
Claim: If G is a PRG then $H_0 = \left\{ G(k) \oplus m_0 \ : \ k \xleftarrow{\$} \{0,1\}^\lambda \right\} \overset{c}{\approx} H_1 = \left\{ r \oplus m_0 \ : \ r \xleftarrow{\$} \{0,1\}^{\ell(\lambda)} \right\}$.



- A_{PRG} is playing the PRG indistinguishability game.
- A is playing the indistinguishability game between H_0 and H_1.
  - We should construct A_{PRG} in such a way that A thinks its talking to Ch_{H_0,H_1}.

Security of Pseudorandom OTP: Reduction

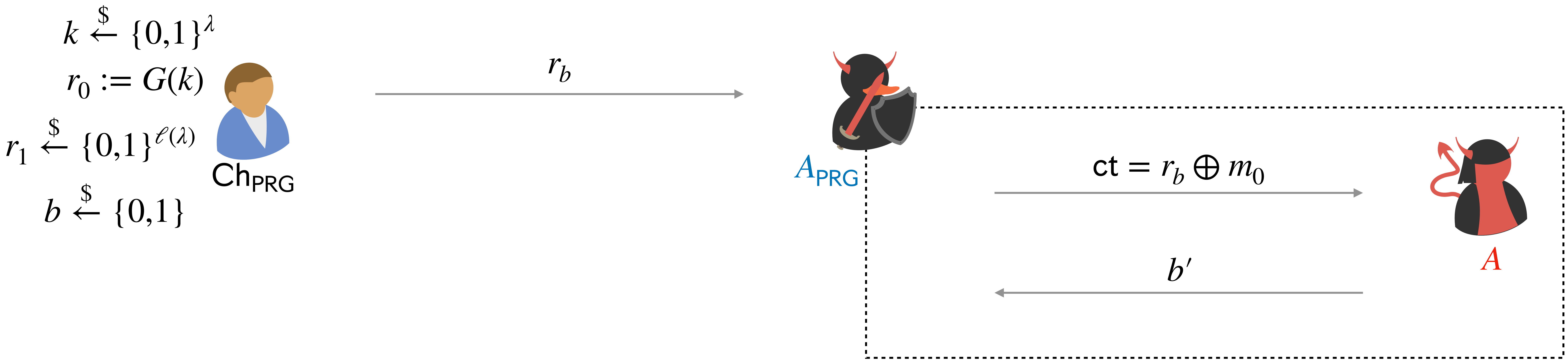
Claim: If G is a PRG then $H_0 = \left\{ G(k) \oplus m_0 \ : \ k \xleftarrow{\$} \{0,1\}^{\lambda} \right\} \overset{c}{\approx} H_1 = \left\{ r \oplus m_0 \ : \ r \xleftarrow{\$} \{0,1\}^{\ell(\lambda)} \right\}$.



- A_{PRG} is playing the PRG indistinguishability game.
- A is playing the indistinguishability game between H_0 and H_1.
  - We should construct A_{PRG} in such a way that A thinks its talking to Ch_{H_0,H_1}.

Security of Pseudorandom OTP: Reduction

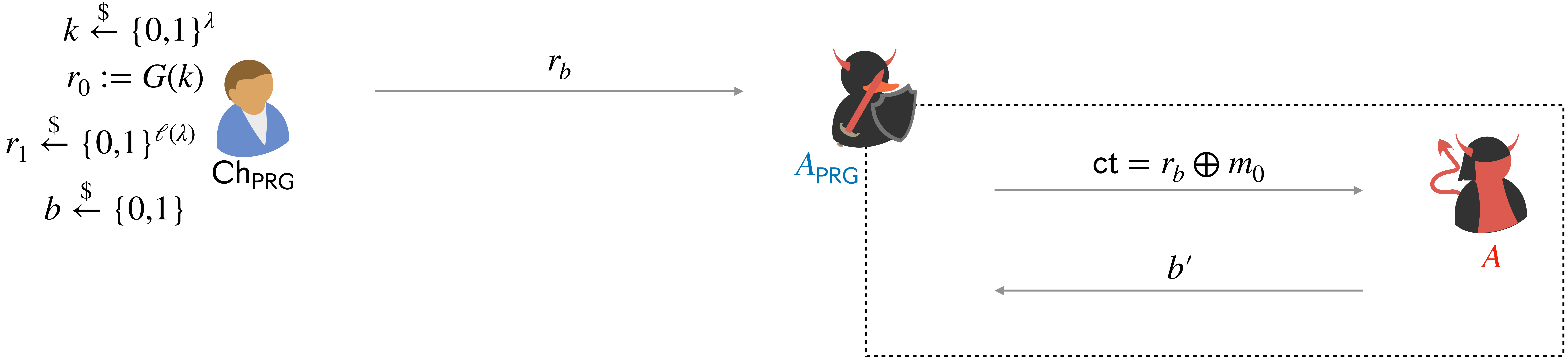
Claim: If G is a PRG then $H_0 = \left\{ G(k) \oplus m_0 \ : \ k \xleftarrow{\$} \{0,1\}^\lambda \right\} \overset{c}{\approx} H_1 = \left\{ r \oplus m_0 \ : \ r \xleftarrow{\$} \{0,1\}^{\ell(\lambda)} \right\}$.



- A_{PRG} is playing the PRG indistinguishability game.
- A is playing the indistinguishability game between H_0 and H_1.
  - We should construct A_{PRG} in such a way that A thinks its talking to Ch_{H_0,H_1}.

Security of Pseudorandom OTP: Reduction

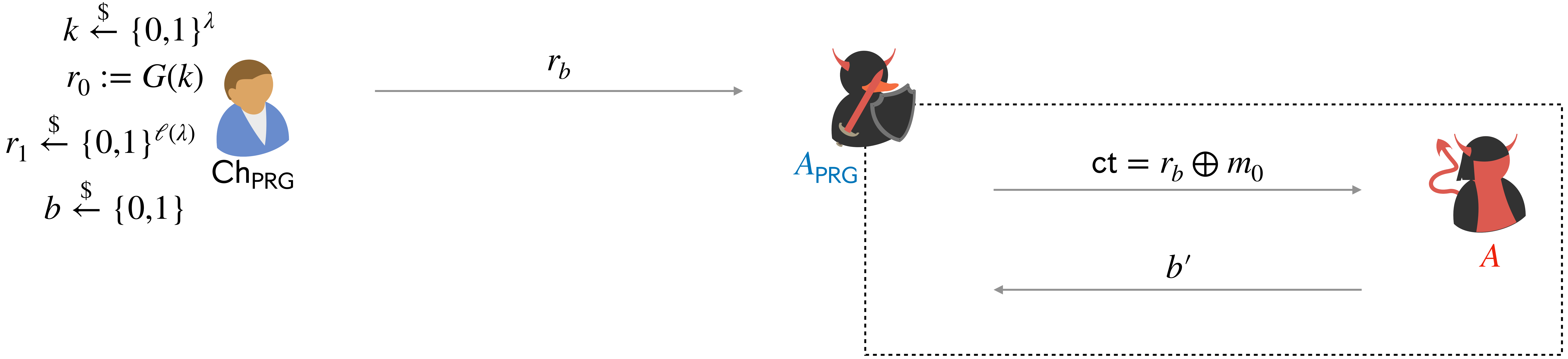
Claim: If G is a PRG then $H_0 = \left\{ G(k) \oplus m_0 \ : \ k \xleftarrow{\$} \{0,1\}^\lambda \right\} \overset{c}{\approx} H_1 = \left\{ r \oplus m_0 \ : \ r \xleftarrow{\$} \{0,1\}^{\ell(\lambda)} \right\}$.



- A_{PRG} is playing the PRG indistinguishability game.
- A is playing the indistinguishability game between H_0 and H_1.
  - We should construct A_{PRG} in such a way that A thinks its talking to Ch_{H_0,H_1}.
  - We should construct A_{PRG} to leverage A's distinguishing advantage.

Security of Pseudorandom OTP: Reduction

Claim: If G is a PRG then $H_0 = \left\{ G(k) \oplus m_0 \; : \; k \xleftarrow{\$} \{0,1\}^\lambda \right\} \overset{c}{\approx} H_1 = \left\{ r \oplus m_0 \; : \; r \xleftarrow{\$} \{0,1\}^{\ell(\lambda)} \right\}$.

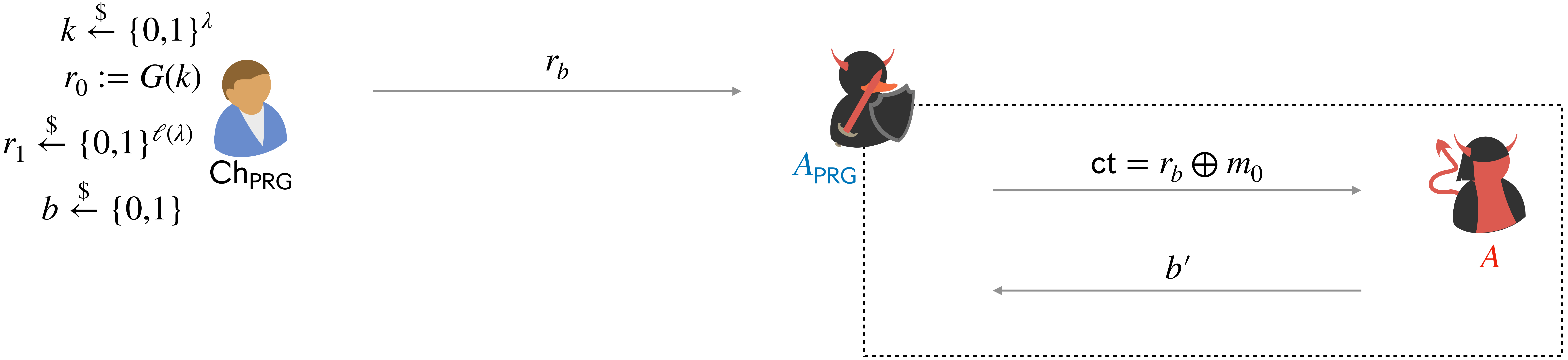


If $b' = 0$, then A believes $\mathsf{ct} \leftarrow H_0$

- A_{PRG} is playing the PRG indistinguishability game.
- A is playing the indistinguishability game between H_0 and H_1.
  - We should construct A_{PRG} in such a way that A thinks its talking to Ch_{H_0,H_1}.
  - We should construct A_{PRG} to leverage A's distinguishing advantage.

Security of Pseudorandom OTP: Reduction

Claim: If G is a PRG then $H_0 = \left\{ G(k) \oplus m_0 \ : \ k \xleftarrow{\$} \{0,1\}^\lambda \right\} \overset{c}{\approx} H_1 = \left\{ r \oplus m_0 \ : \ r \xleftarrow{\$} \{0,1\}^{\ell(\lambda)} \right\}$.

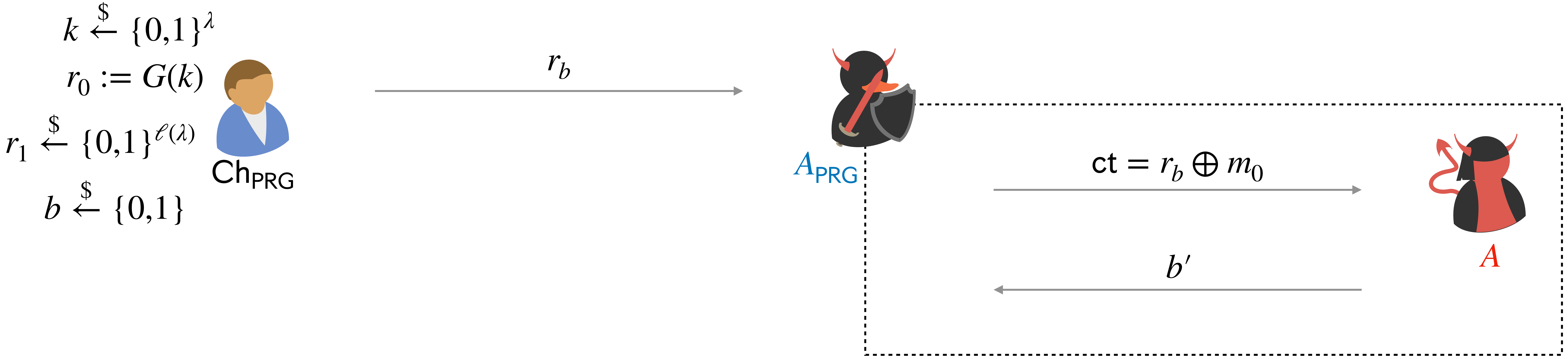


If $b' = 0$, then A believes ct $\leftarrow H_0 \implies r_b := G(k)$

- A_{PRG} is playing the PRG indistinguishability game.
- A is playing the indistinguishability game between H_0 and H_1.
  - We should construct A_{PRG} in such a way that A thinks its talking to Ch_{H_0,H_1}.
  - We should construct A_{PRG} to leverage A's distinguishing advantage.

Security of Pseudorandom OTP: Reduction

Claim: If G is a PRG then $H_0 = \left\{ G(k) \oplus m_0 \ : \ k \xleftarrow{\$} \{0,1\}^\lambda \right\} \overset{c}{\approx} H_1 = \left\{ r \oplus m_0 \ : \ r \xleftarrow{\$} \{0,1\}^{\ell(\lambda)} \right\}$.



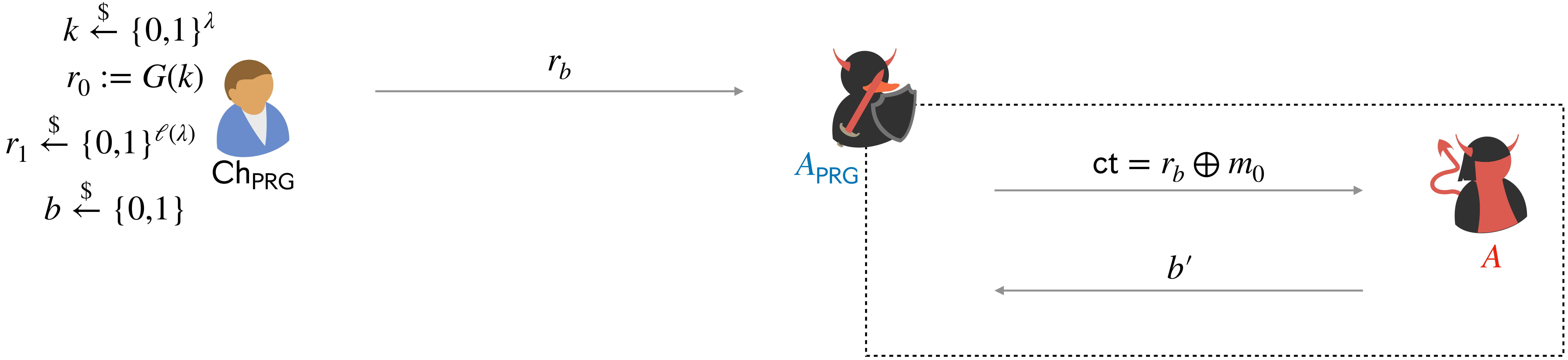
If $b' = 0$, then A believes $\mathsf{ct} \leftarrow H_0$ $\implies r_b := G(k)$

If $b' = 1$, then A believes $\mathsf{ct} \leftarrow H_1$

- A_{PRG} is playing the PRG indistinguishability game.
- A is playing the indistinguishability game between H_0 and H_1.
  - We should construct A_{PRG} in such a way that A thinks its talking to Ch_{H_0,H_1}.
  - We should construct A_{PRG} to leverage A's distinguishing advantage.

Security of Pseudorandom OTP: Reduction

Claim: If G is a PRG then $H_0 = \left\{ G(k) \oplus m_0 \ : \ k \xleftarrow{\$} \{0,1\}^\lambda \right\} \overset{c}{\approx} H_1 = \left\{ r \oplus m_0 \ : \ r \xleftarrow{\$} \{0,1\}^{\ell(\lambda)} \right\}$.



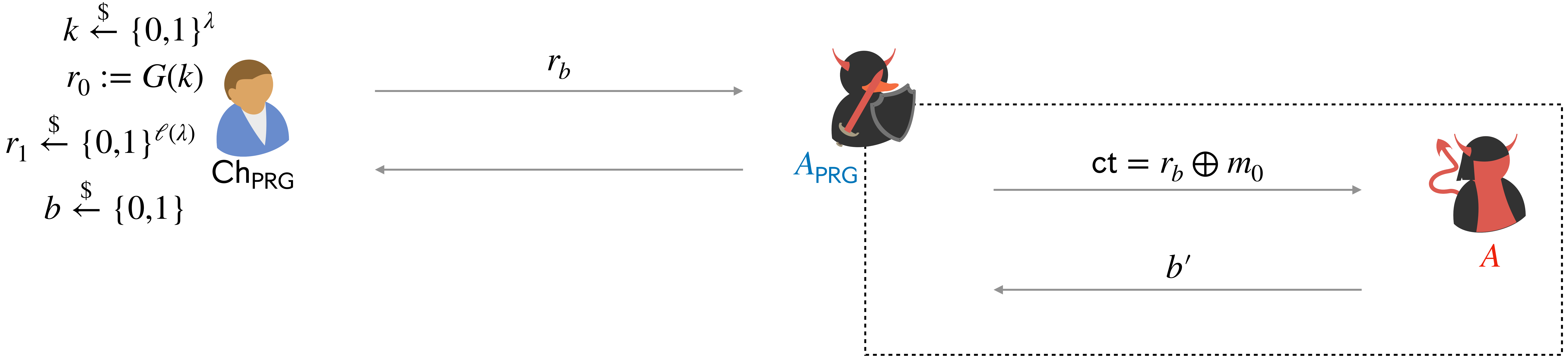
If $b' = 0$, then A believes $\mathsf{ct} \leftarrow H_0 \implies r_b := G(k)$

If $b' = 1$, then A believes $\mathsf{ct} \leftarrow H_1 \implies r_b \leftarrow \{0,1\}^\lambda$

- A_{PRG} is playing the PRG indistinguishability game.
- A is playing the indistinguishability game between H_0 and H_1.
  - We should construct A_{PRG} in such a way that A thinks its talking to Ch_{H_0,H_1}.
  - We should construct A_{PRG} to leverage A's distinguishing advantage.

Security of Pseudorandom OTP: Reduction

Claim: If G is a PRG then $H_0 = \left\{ G(k) \oplus m_0 \ : \ k \xleftarrow{\$} \{0,1\}^\lambda \right\} \overset{c}{\approx} H_1 = \left\{ r \oplus m_0 \ : \ r \xleftarrow{\$} \{0,1\}^{\ell(\lambda)} \right\}$.



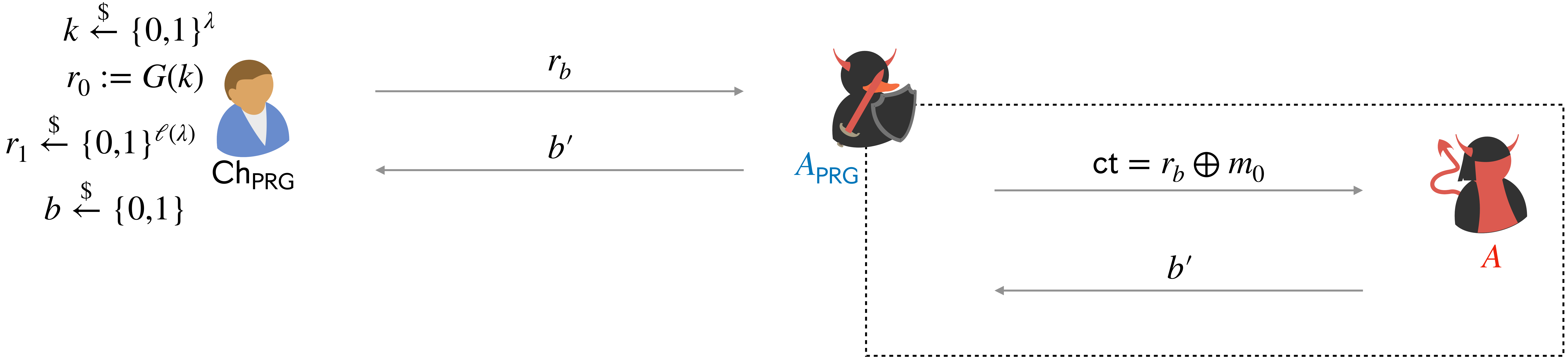
If $b' = 0$, then A believes $\mathsf{ct} \leftarrow H_0 \implies r_b := G(k)$

If $b' = 1$, then A believes $\mathsf{ct} \leftarrow H_1 \implies r_b \leftarrow \{0,1\}^\lambda$

- A_{PRG} is playing the PRG indistinguishability game.
- A is playing the indistinguishability game between H_0 and H_1.
    - We should construct A_{PRG} in such a way that A thinks its talking to Ch_{H_0,H_1}.
    - We should construct A_{PRG} to leverage A's distinguishing advantage.

Security of Pseudorandom OTP: Reduction

Claim: If G is a PRG then $H_0 = \left\{ G(k) \oplus m_0 \ :\ k \xleftarrow{\$} \{0,1\}^\lambda \right\} \stackrel{c}{\approx} H_1 = \left\{ r \oplus m_0 \ :\ r \xleftarrow{\$} \{0,1\}^{\ell(\lambda)} \right\}$.



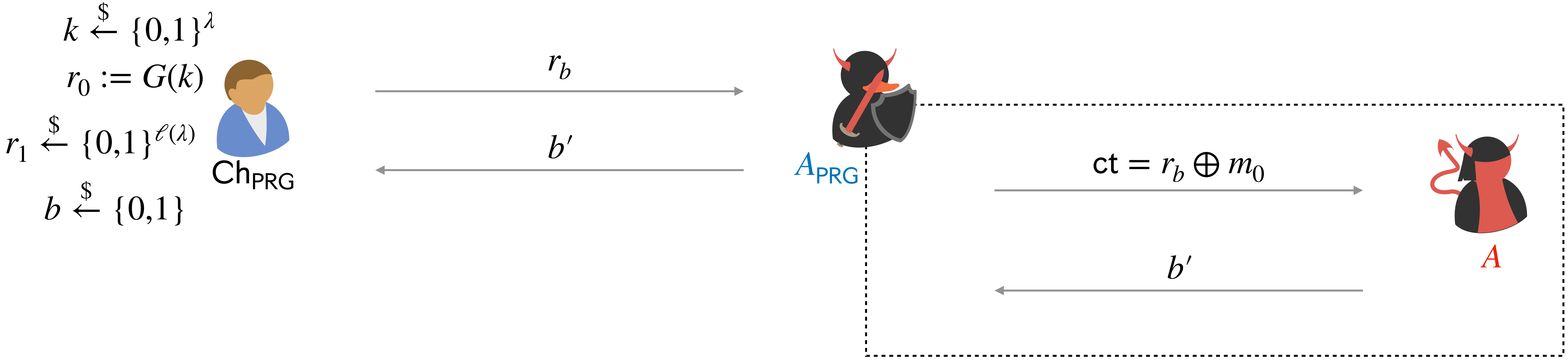
If $b' = 0$, then A believes $\mathsf{ct} \leftarrow H_0 \implies r_b := G(k)$

If $b' = 1$, then A believes $\mathsf{ct} \leftarrow H_1 \implies r_b \leftarrow \{0,1\}^\lambda$

- A_{PRG} is playing the PRG indistinguishability game.
- A is playing the indistinguishability game between H_0 and H_1.
  - We should construct A_{PRG} in such a way that A thinks its talking to Ch_{H_0,H_1}.
  - We should construct A_{PRG} to leverage A's distinguishing advantage.

Security of Pseudorandom OTP: Reduction

Claim: If G is a PRG then $H_0 = \left\{ G(k) \oplus m_0 \;:\; k \xleftarrow{\$} \{0,1\}^\lambda \right\} \overset{c}{\approx} H_1 = \left\{ r \oplus m_0 \;:\; r \xleftarrow{\$} \{0,1\}^{\ell(\lambda)} \right\}$.

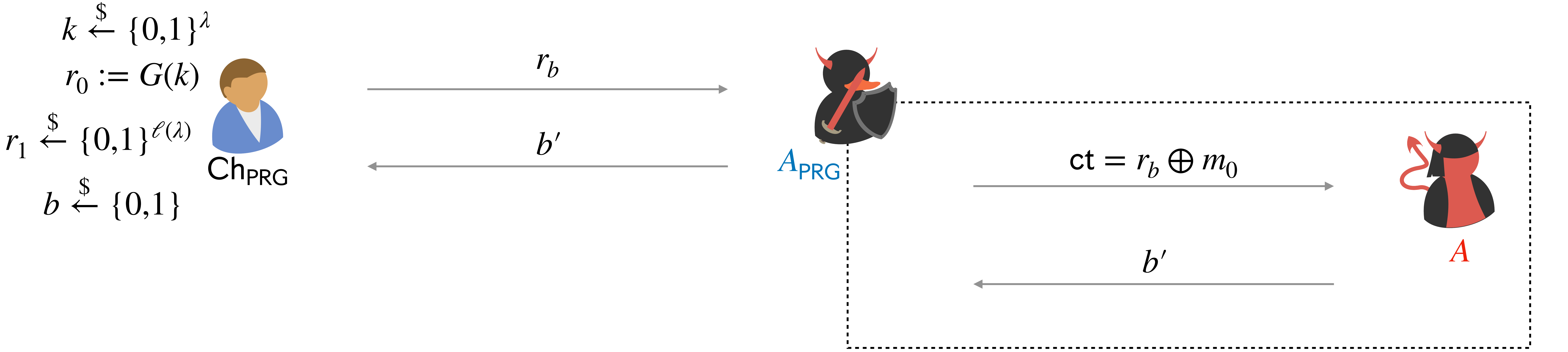


If $b' = 0$, then A believes ct $\leftarrow H_0 \implies r_b := G(k)$

If $b' = 1$, then A believes ct $\leftarrow H_1 \implies r_b \leftarrow \{0,1\}^\lambda$

Security of Pseudorandom OTP: Reduction

Claim: If G is a PRG then $H_0 = \left\{ G(k) \oplus m_0 \ : \ k \xleftarrow{\$} \{0,1\}^\lambda \right\} \overset{c}{\approx} H_1 = \left\{ r \oplus m_0 \ : \ r \xleftarrow{\$} \{0,1\}^{\ell(\lambda)} \right\}$.



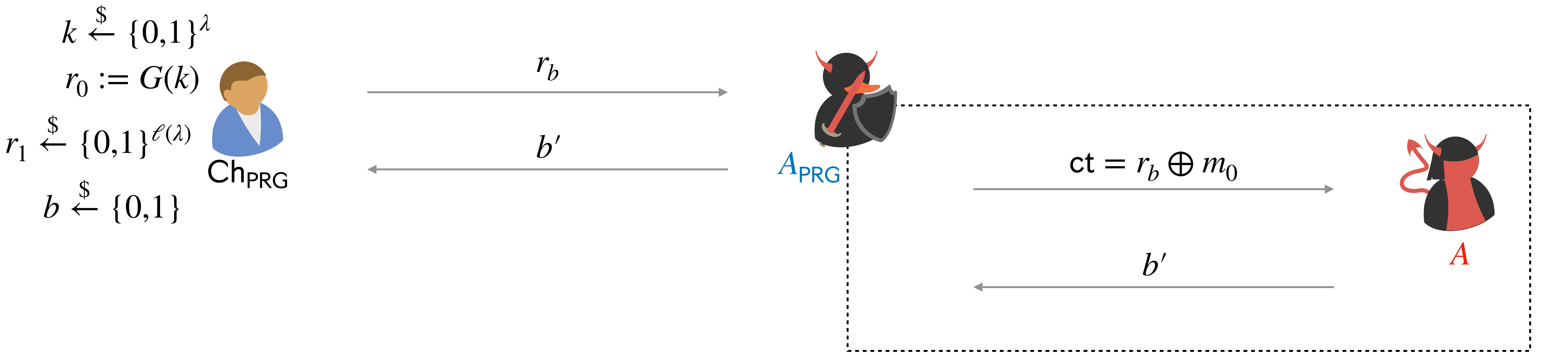
If $b' = 0$, then A believes ct $\leftarrow H_0$ $\implies r_b := G(k)$

If $b' = 1$, then A believes ct $\leftarrow H_1$ $\implies r_b \leftarrow \{0,1\}^\lambda$

$$\left| \Pr\left[A(\text{ct}) = 1 \,|\, b = 0\right] - \Pr\left[A(\text{ct}) = 1 \,|\, b = 1\right] \right| = \epsilon$$

Security of Pseudorandom OTP: Reduction

Claim: If G is a PRG then $H_0 = \left\{ G(k) \oplus m_0 \ : \ k \xleftarrow{\$} \{0,1\}^\lambda \right\} \overset{c}{\approx} H_1 = \left\{ r \oplus m_0 \ : \ r \xleftarrow{\$} \{0,1\}^{\ell(\lambda)} \right\}$.



If $b' = 0$, then A believes $\mathsf{ct} \leftarrow H_0 \implies r_b := G(k)$

If $b' = 1$, then A believes $\mathsf{ct} \leftarrow H_1 \implies r_b \leftarrow \{0,1\}^\lambda$

$$\left| \Pr\left[A(\mathsf{ct}) = 1 \,|\, b = 0\right] - \Pr\left[A(\mathsf{ct}) = 1 \,|\, b = 1\right] \right| = \epsilon$$

$$\left| \Pr\left[A_{\mathsf{PRG}}(r_b) = 1 \,|\, b = 0\right] - \Pr\left[A_{\mathsf{PRG}}(r_b) = 1 \,|\, b = 1\right] \right| = \epsilon_{\mathsf{PRG}}$$

Security of Pseudorandom OTP: Reduction

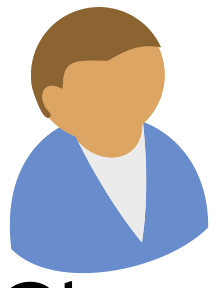
Claim: If G is a PRG then $H_0 = \left\{ G(k) \oplus m_0 \;:\; k \xleftarrow{\$} \{0,1\}^{\lambda} \right\} \overset{c}{\approx} H_1 = \left\{ r \oplus m_0 \;:\; r \xleftarrow{\$} \{0,1\}^{\ell(\lambda)} \right\}$.

$k \xleftarrow{\$} \{0,1\}^{\lambda}$

$r_0 := G(k)$

$r_1 \xleftarrow{\$} \{0,1\}^{\ell(\lambda)}$

$b \xleftarrow{\$} \{0,1\}$



$\mathsf{Ch}_{\mathsf{PRG}}$

r_b →

← b'

A_{PRG}

$\mathsf{ct} = r_b \oplus m_0$ →

← b'

A

If $b' = 0$, then A believes $\mathsf{ct} \leftarrow H_0 \implies r_b := G(k)$

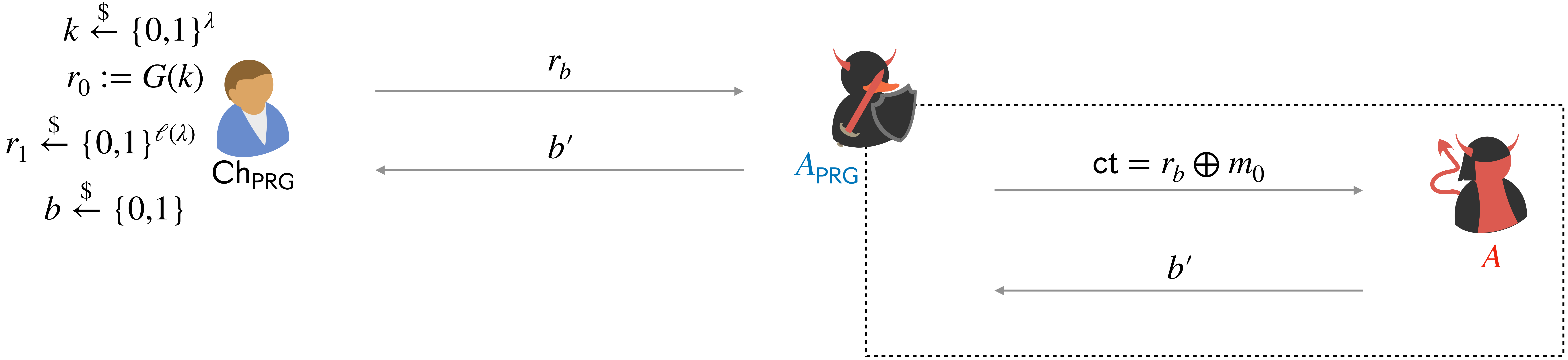
If $b' = 1$, then A believes $\mathsf{ct} \leftarrow H_1 \implies r_b \leftarrow \{0,1\}^{\lambda}$

$\left| \Pr\left[A(\mathsf{ct}) = 1 \mid b = 0\right] - \Pr\left[A(\mathsf{ct}) = 1 \mid b = 1\right] \right| = \epsilon$

$\left| \Pr\left[A_{\mathsf{PRG}}(r_b) = 1 \mid b = 0\right] - \Pr\left[A_{\mathsf{PRG}}(r_b) = 1 \mid b = 1\right] \right| = \epsilon_{\mathsf{PRG}}$

Security of Pseudorandom OTP: Reduction

Claim: If G is a PRG then $H_0 = \left\{ G(k) \oplus m_0 \;:\; k \xleftarrow{\$} \{0,1\}^\lambda \right\} \overset{c}{\approx} H_1 = \left\{ r \oplus m_0 \;:\; r \xleftarrow{\$} \{0,1\}^{\ell(\lambda)} \right\}$.



If $b' = 0$, then A believes ct $\leftarrow H_0 \implies r_b := G(k)$

If $b' = 1$, then A believes ct $\leftarrow H_1 \implies r_b \leftarrow \{0,1\}^\lambda$

$$\left| \Pr\left[A(\mathsf{ct}) = 1 \mid b = 0\right] - \Pr\left[A(\mathsf{ct}) = 1 \mid b = 1\right] \right| = \epsilon$$

$$=$$

$$\left| \Pr\left[A_{\mathsf{PRG}}(r_b) = 1 \mid b = 0\right] - \Pr\left[A_{\mathsf{PRG}}(r_b) = 1 \mid b = 1\right] \right| = \epsilon_{\mathsf{PRG}}$$

Security of Pseudorandom OTP: Reduction

Claim: If G is a PRG then $H_0 = \left\{ G(k) \oplus m_0 \ : \ k \xleftarrow{\$} \{0,1\}^\lambda \right\} \overset{c}{\approx} H_1 = \left\{ r \oplus m_0 \ : \ r \xleftarrow{\$} \{0,1\}^{\ell(\lambda)} \right\}$.

$k \xleftarrow{\$} \{0,1\}^\lambda$

$r_0 := G(k)$

$r_1 \xleftarrow{\$} \{0,1\}^{\ell(\lambda)}$

$b \xleftarrow{\$} \{0,1\}$

$\mathsf{Ch}_{\mathsf{PRG}}$ → r_b → A_{PRG}

A_{PRG} → b' → $\mathsf{Ch}_{\mathsf{PRG}}$

A_{PRG} → $\mathsf{ct} = r_b \oplus m_0$ → A

A → b' → A_{PRG}

If $b' = 0$, then A believes $\mathsf{ct} \leftarrow H_0 \implies r_b := G(k)$

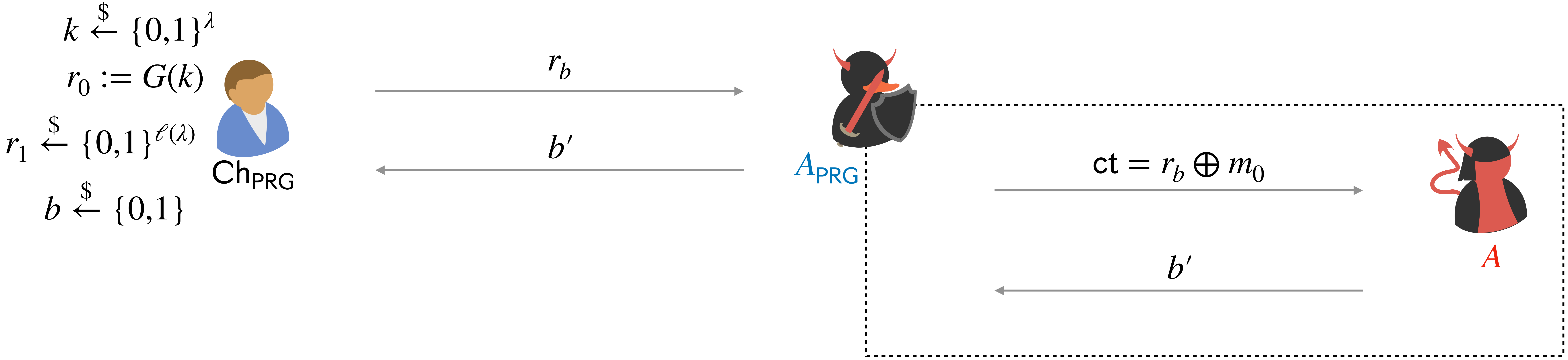
If $b' = 1$, then A believes $\mathsf{ct} \leftarrow H_1 \implies r_b \leftarrow \{0,1\}^\lambda$

$\left| \Pr\left[A(\mathsf{ct}) = 1 \mid b = 0\right] - \Pr\left[A(\mathsf{ct}) = 1 \mid b = 1\right] \right| = \epsilon$

$\left| \Pr\left[A_{\mathsf{PRG}}(r_b) = 1 \mid b = 0\right] - \Pr\left[A_{\mathsf{PRG}}(r_b) = 1 \mid b = 1\right] \right| = \epsilon_{\mathsf{PRG}}$

Security of Pseudorandom OTP: Reduction

Claim: If G is a PRG then $H_0 = \left\{ G(k) \oplus m_0 \ : \ k \xleftarrow{\$} \{0,1\}^\lambda \right\} \overset{c}{\approx} H_1 = \left\{ r \oplus m_0 \ : \ r \xleftarrow{\$} \{0,1\}^{\ell(\lambda)} \right\}$.



If $b' = 0$, then A believes ct $\leftarrow H_0 \implies r_b := G(k)$

If $b' = 1$, then A believes ct $\leftarrow H_1 \implies r_b \leftarrow \{0,1\}^\lambda$

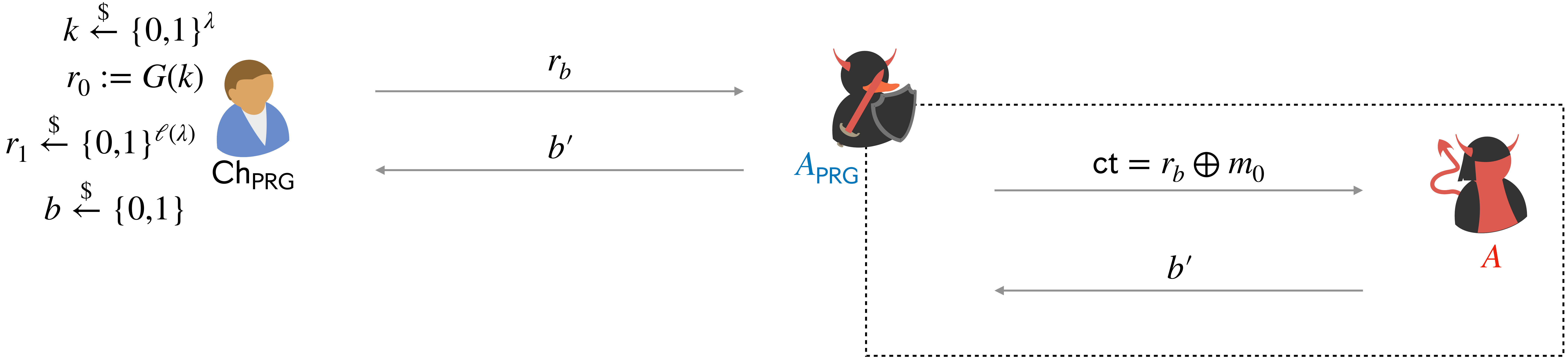
$$\left| \Pr\left[A(\text{ct}) = 1 \mid b = 0\right] - \Pr\left[A(\text{ct}) = 1 \mid b = 1\right] \right| = \epsilon$$

$$=$$

$$\left| \Pr\left[A_{\text{PRG}}(r_b) = 1 \mid b = 0\right] - \Pr\left[A_{\text{PRG}}(r_b) = 1 \mid b = 1\right] \right| = \epsilon_{\text{PRG}}$$

Security of Pseudorandom OTP: Reduction

Claim: If G is a PRG then $H_0 = \left\{ G(k) \oplus m_0 \ : \ k \xleftarrow{\$} \{0,1\}^\lambda \right\} \overset{c}{\approx} H_1 = \left\{ r \oplus m_0 \ : \ r \xleftarrow{\$} \{0,1\}^{\ell(\lambda)} \right\}$.



If $b' = 0$, then A believes $\mathsf{ct} \leftarrow H_0 \implies r_b := G(k)$

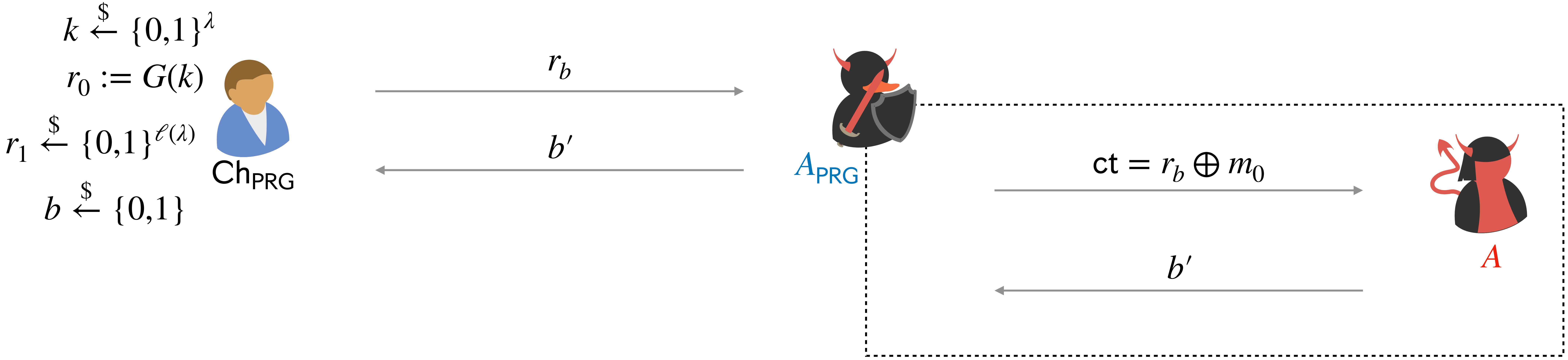
If $b' = 1$, then A believes $\mathsf{ct} \leftarrow H_1 \implies r_b \leftarrow \{0,1\}^\lambda$

$$\left| \Pr\left[A(\mathsf{ct}) = 1 \mid b = 0\right] - \Pr\left[A(\mathsf{ct}) = 1 \mid b = 1\right] \right| = \epsilon$$

$$\left| \Pr\left[A_{\mathsf{PRG}}(r_b) = 1 \mid b = 0\right] - \Pr\left[A_{\mathsf{PRG}}(r_b) = 1 \mid b = 1\right] \right| = \epsilon_{\mathsf{PRG}}$$

Security of Pseudorandom OTP: Reduction

Claim: If G is a PRG then $H_0 = \left\{ G(k) \oplus m_0 \ : \ k \xleftarrow{\$} \{0,1\}^\lambda \right\} \overset{c}{\approx} H_1 = \left\{ r \oplus m_0 \ : \ r \xleftarrow{\$} \{0,1\}^{\ell(\lambda)} \right\}$.



If $b' = 0$, then A believes $\mathsf{ct} \leftarrow H_0 \implies r_b := G(k)$

If $b' = 1$, then A believes $\mathsf{ct} \leftarrow H_1 \implies r_b \leftarrow \{0,1\}^\lambda$

$$\left| \Pr\left[A(\mathsf{ct}) = 1 \mid b = 0\right] - \Pr\left[A(\mathsf{ct}) = 1 \mid b = 1\right] \right| = \epsilon$$

$$\left| \Pr\left[A_{\mathsf{PRG}}(r_b) = 1 \mid b = 0\right] - \Pr\left[A_{\mathsf{PRG}}(r_b) = 1 \mid b = 1\right] \right| = \epsilon_{\mathsf{PRG}}$$

Therefore, $\epsilon = \epsilon_{\mathsf{PRG}} = \mathsf{negl}(\lambda)$.

Security of Pseudorandom OTP

Theorem: Pseudorandom OTP is one-time computational secure.

Intuition:

$$H_0 = \left\{ G(k) \oplus m_0 \ : \ k \xleftarrow{\$} \{0,1\}^{\lambda} \right\}$$

$$H_0 \overset{c}{\approx} H_1$$

$$H_1 = \left\{ r \oplus m_0 \ : \ r \xleftarrow{\$} \{0,1\}^{\ell(\lambda)} \right\}$$

$$H_1 \equiv H_2$$

$$H_2 = \left\{ r \oplus m_1 \ : \ r \xleftarrow{\$} \{0,1\}^{\ell(\lambda)} \right\}$$

$$H_2 \overset{c}{\approx} H_3$$

$$H_3 = \left\{ G(k) \oplus m_1 \ : \ k \xleftarrow{\$} \{0,1\}^{\lambda} \right\}$$

$$H_0 \overset{c}{\approx} H_3$$

Reductions

Reductions

Claim: If $X_0 \overset{c}{\approx} X_1$ then $Y_0 \overset{c}{\approx} Y_1$.

Reductions

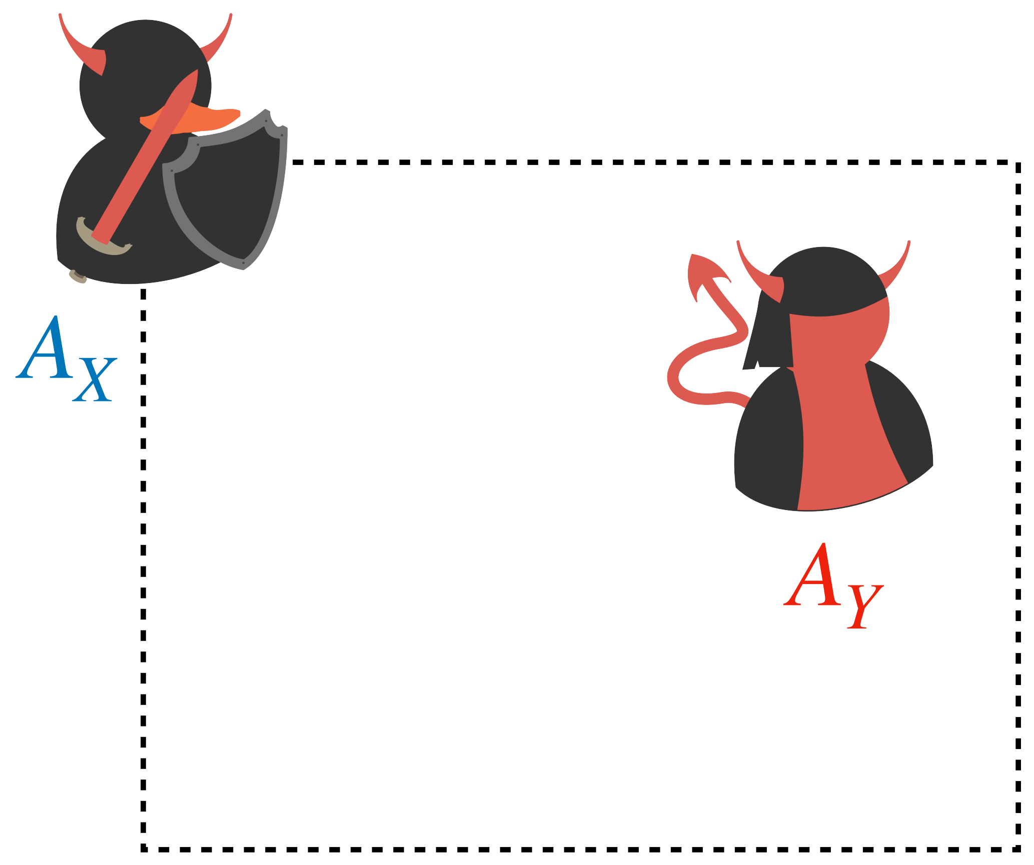
Claim: If $X_0 \overset{c}{\approx} X_1$ then $Y_0 \overset{c}{\approx} Y_1$.

- Given A_Y that distinguishes between Y_0 and Y_1 with probability ϵ_Y.

Reductions

Claim: If $X_0 \overset{c}{\approx} X_1$ then $Y_0 \overset{c}{\approx} Y_1$.

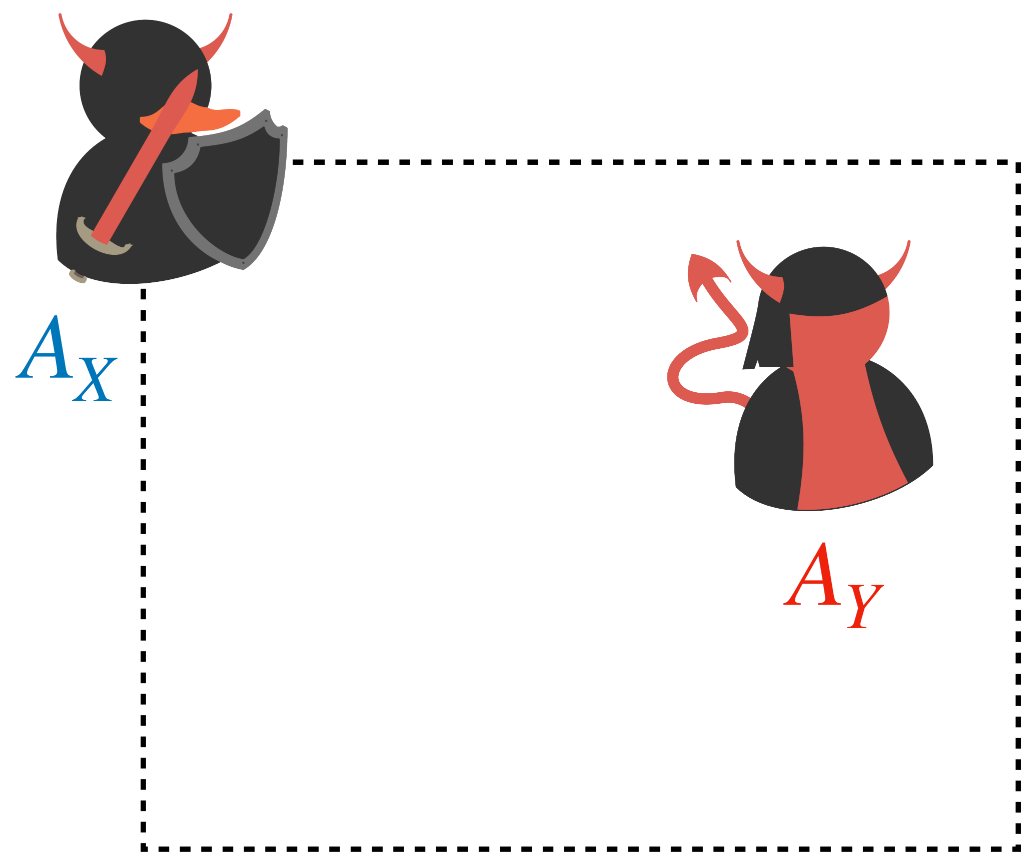
- Given A_Y that distinguishes between Y_0 and Y_1 with probability ϵ_Y.
- We construct A_X that distinguishes between X_0 and X_1 with probability ϵ_X.



Reductions

Claim: If $X_0 \overset{c}{\approx} X_1$ then $Y_0 \overset{c}{\approx} Y_1$.

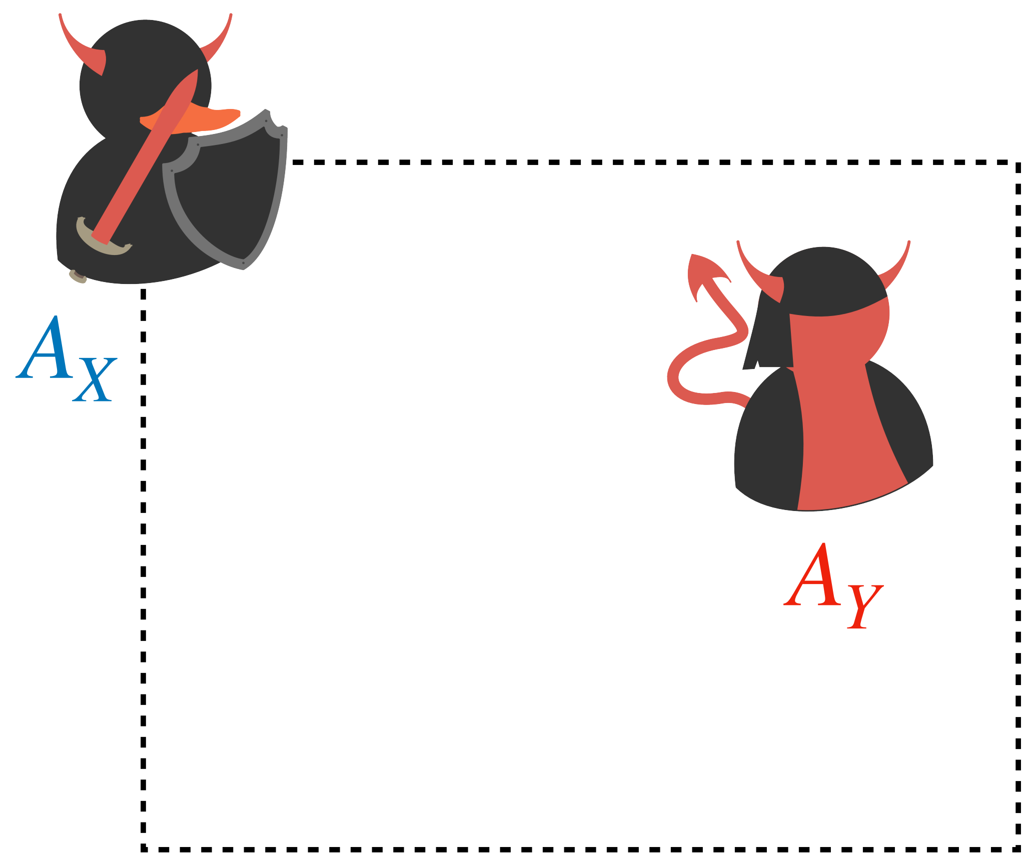
- Given A_Y that distinguishes between Y_0 and Y_1 with probability ϵ_Y.
- We construct A_X that distinguishes between X_0 and X_1 with probability ϵ_X.
  - ϵ_X will be negligibly close to ϵ_Y i.e., $\epsilon_Y = \epsilon_X + \mathsf{negl}(\lambda)$.



Reductions

Claim: If $X_0 \overset{c}{\approx} X_1$ then $Y_0 \overset{c}{\approx} Y_1$.

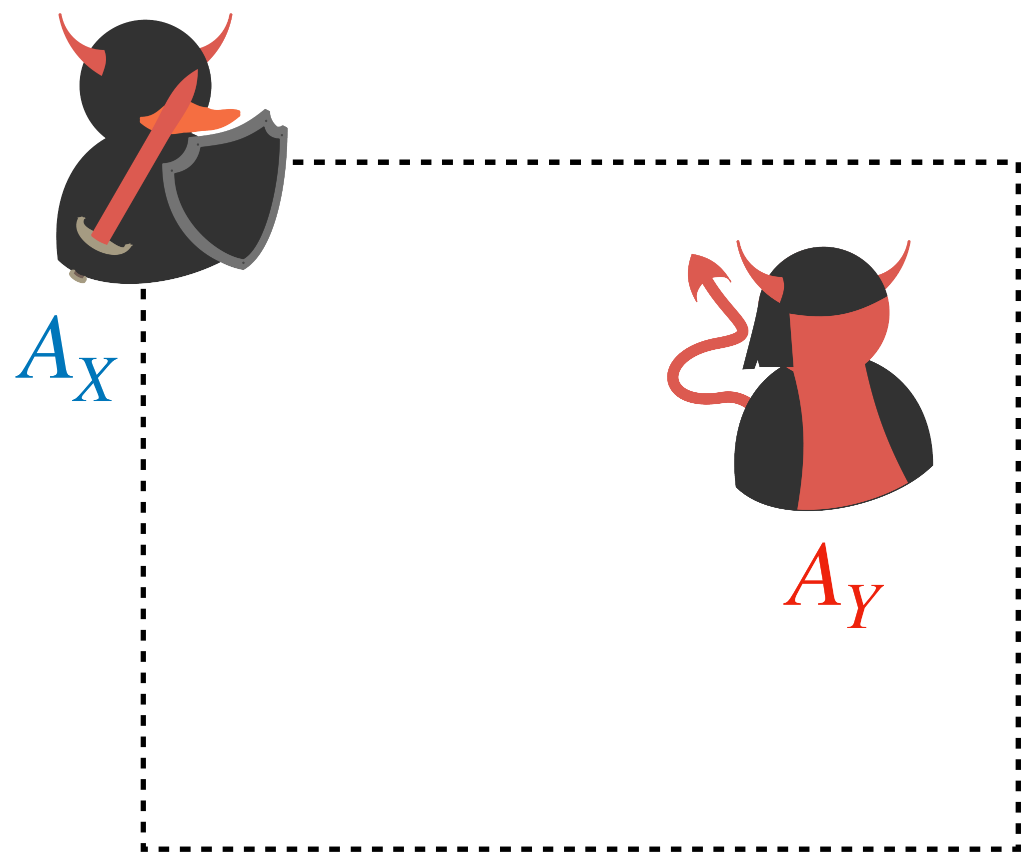
- Given A_Y that distinguishes between Y_0 and Y_1 with probability ϵ_Y.
- We construct A_X that distinguishes between X_0 and X_1 with probability ϵ_X.
  - ϵ_X will be negligibly close to ϵ_Y i.e., $\epsilon_Y = \epsilon_X + \text{negl}(\lambda)$.
  - Since $X_0 \overset{c}{\approx} X_1$, $\epsilon_X = \text{negl}'(\lambda)$.



Reductions

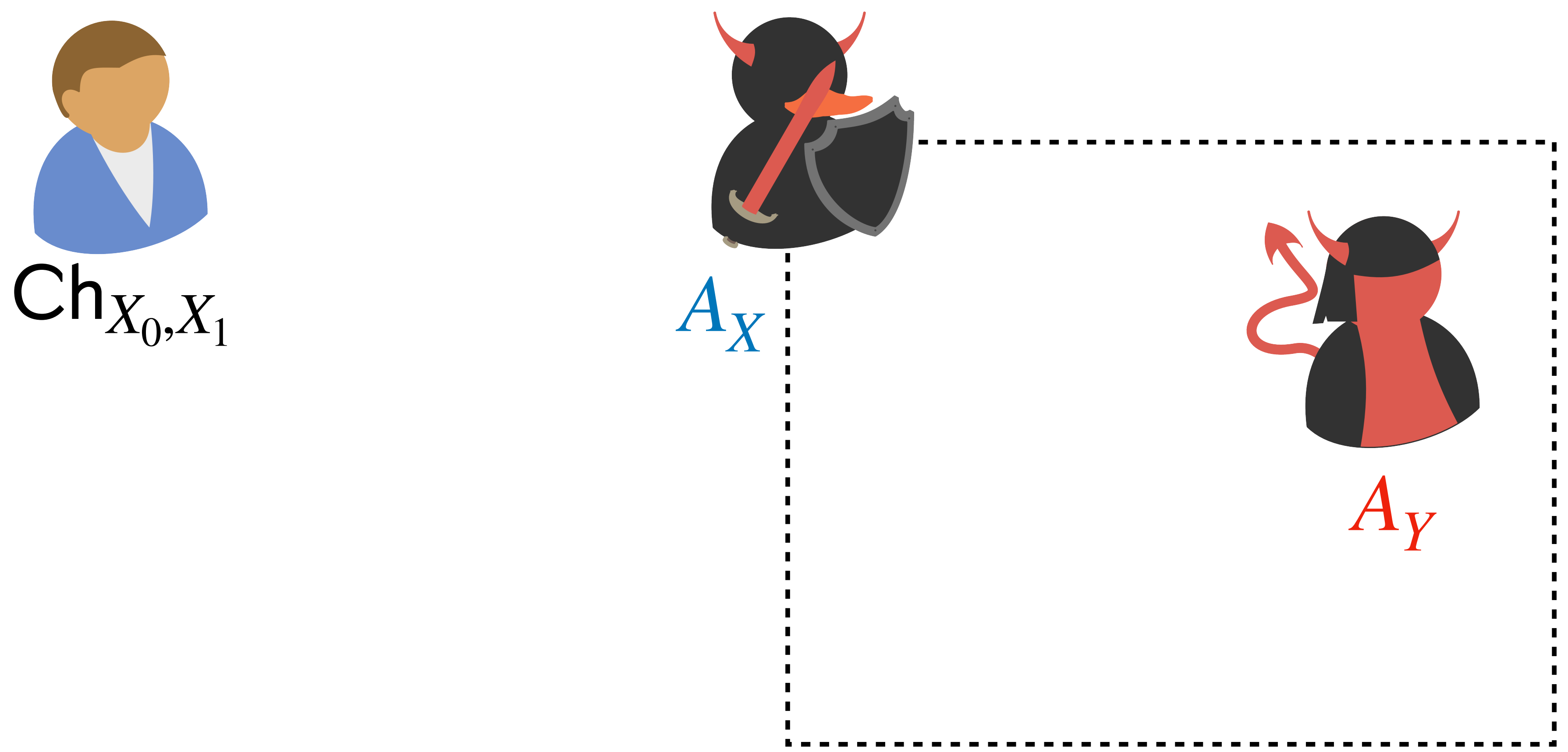
Claim: If $X_0 \overset{c}{\approx} X_1$ then $Y_0 \overset{c}{\approx} Y_1$.

- Given A_Y that distinguishes between Y_0 and Y_1 with probability ϵ_Y.
- We construct A_X that distinguishes between X_0 and X_1 with probability ϵ_X.
  - ϵ_X will be negligibly close to ϵ_Y i.e., $\epsilon_Y = \epsilon_X + \mathsf{negl}(\lambda)$.
  - Since $X_0 \overset{c}{\approx} X_1$, $\epsilon_X = \mathsf{negl}'(\lambda)$.
  - Therefore, $\epsilon_Y = \mathsf{negl}(\lambda) + \mathsf{negl}'(\lambda) = \mathsf{negl}''(\lambda)$.



Reductions: Key Points

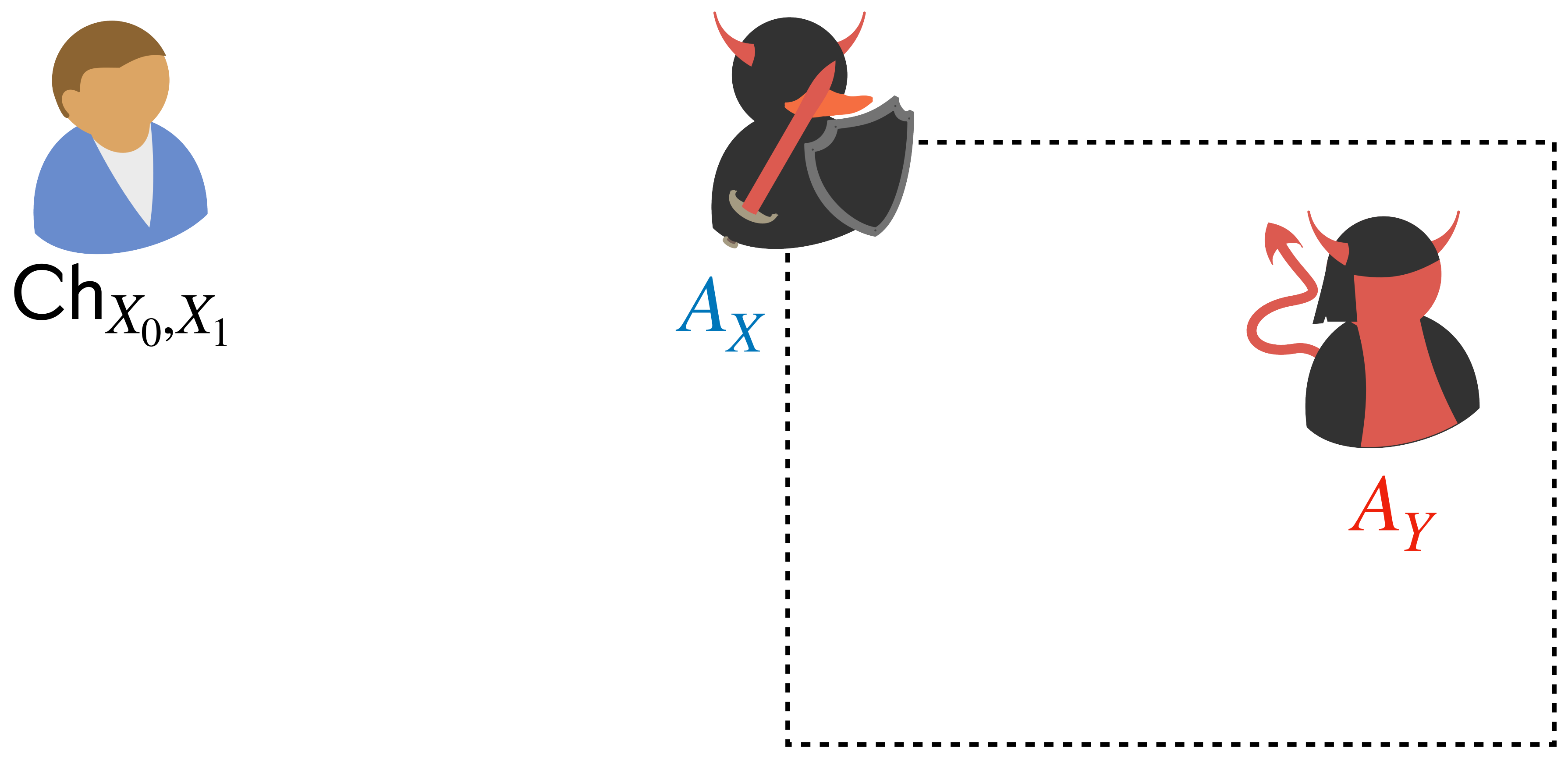
Claim: If $X_0 \overset{c}{\approx} X_1$ then $Y_0 \overset{c}{\approx} Y_1$.



Reductions: Key Points

Claim: If $X_0 \overset{c}{\approx} X_1$ then $Y_0 \overset{c}{\approx} Y_1$.

Two things to keep in mind when working through a reduction.

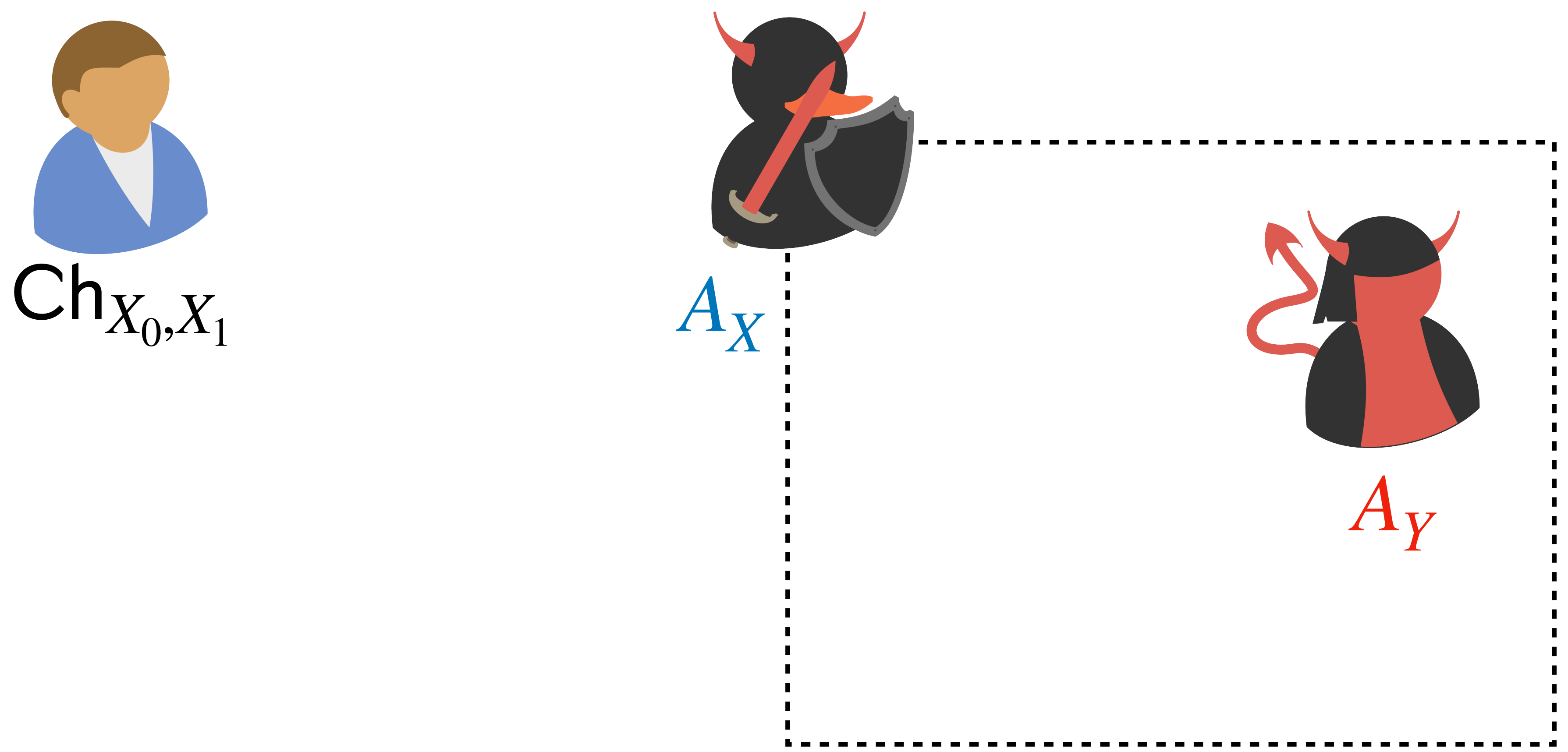


Reductions: Key Points

Claim: If $X_0 \overset{c}{\approx} X_1$ then $Y_0 \overset{c}{\approx} Y_1$.

Two things to keep in mind when working through a reduction.

- **Input Mapping:** How can A_X map the input it receives from Ch_{X_0,X_1} to an input for A_Y?

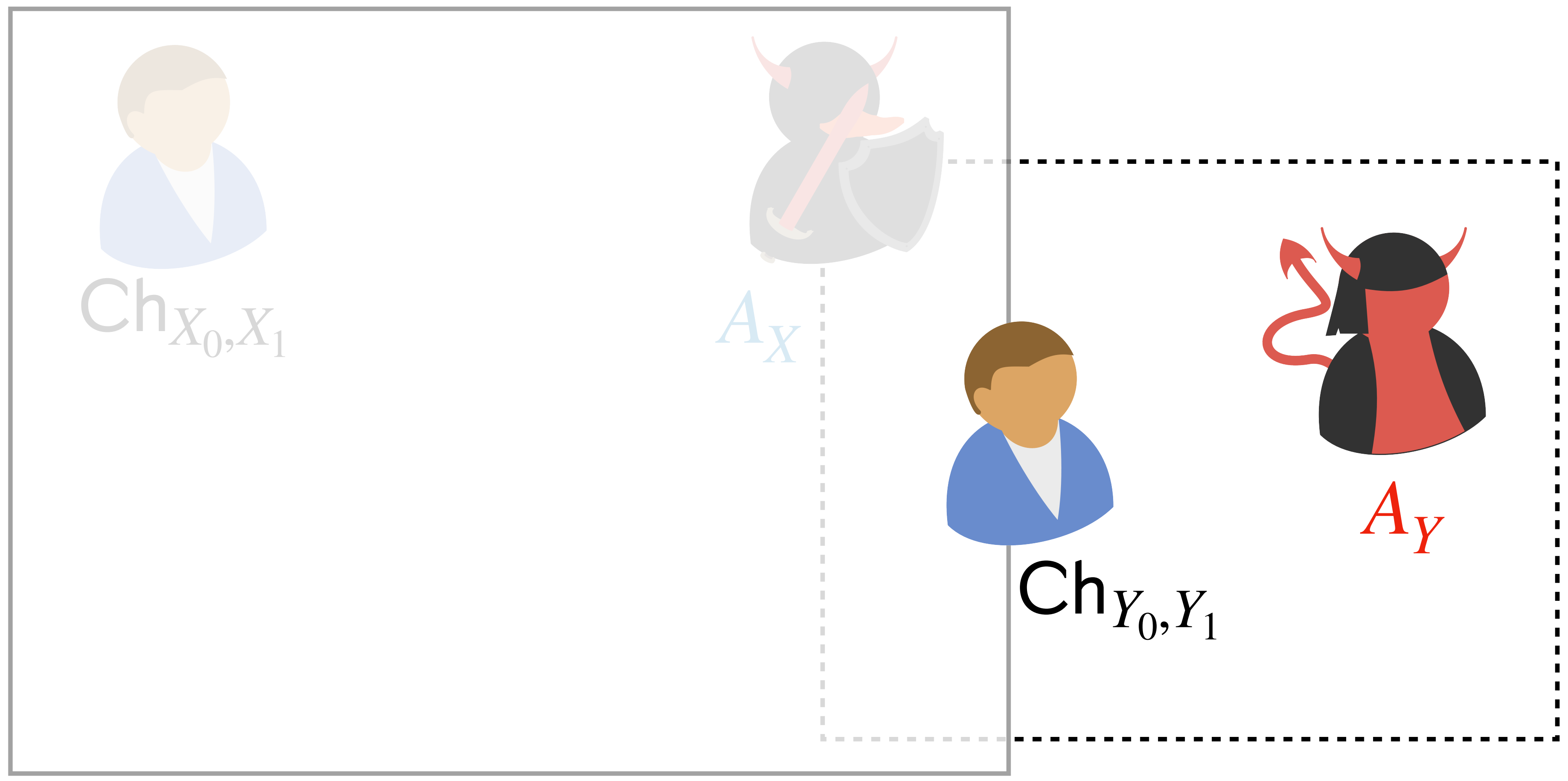


Reductions: Key Points

Claim: If $X_0 \overset{c}{\approx} X_1$ then $Y_0 \overset{c}{\approx} Y_1$.

Two things to keep in mind when working through a reduction.

- **Input Mapping:** How can A_X map the input it receives from Ch_{X_0,X_1} to an input for A_Y?
  - A_Y's view must be identical to the indistinguishability game between Y_0 and Y_1.

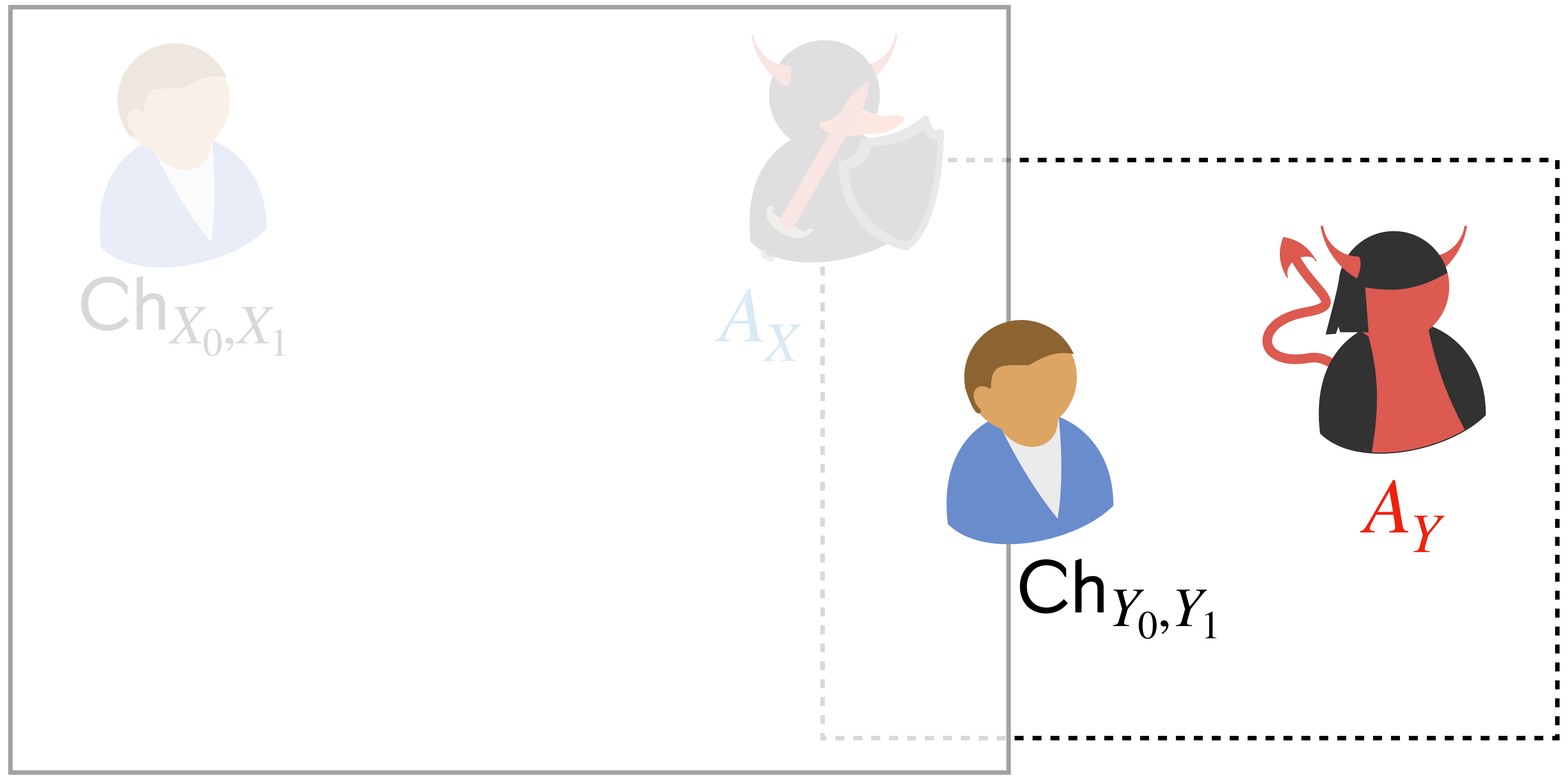


Reductions: Key Points

Claim: If $X_0 \overset{c}{\approx} X_1$ then $Y_0 \overset{c}{\approx} Y_1$.

Two things to keep in mind when working through a reduction.

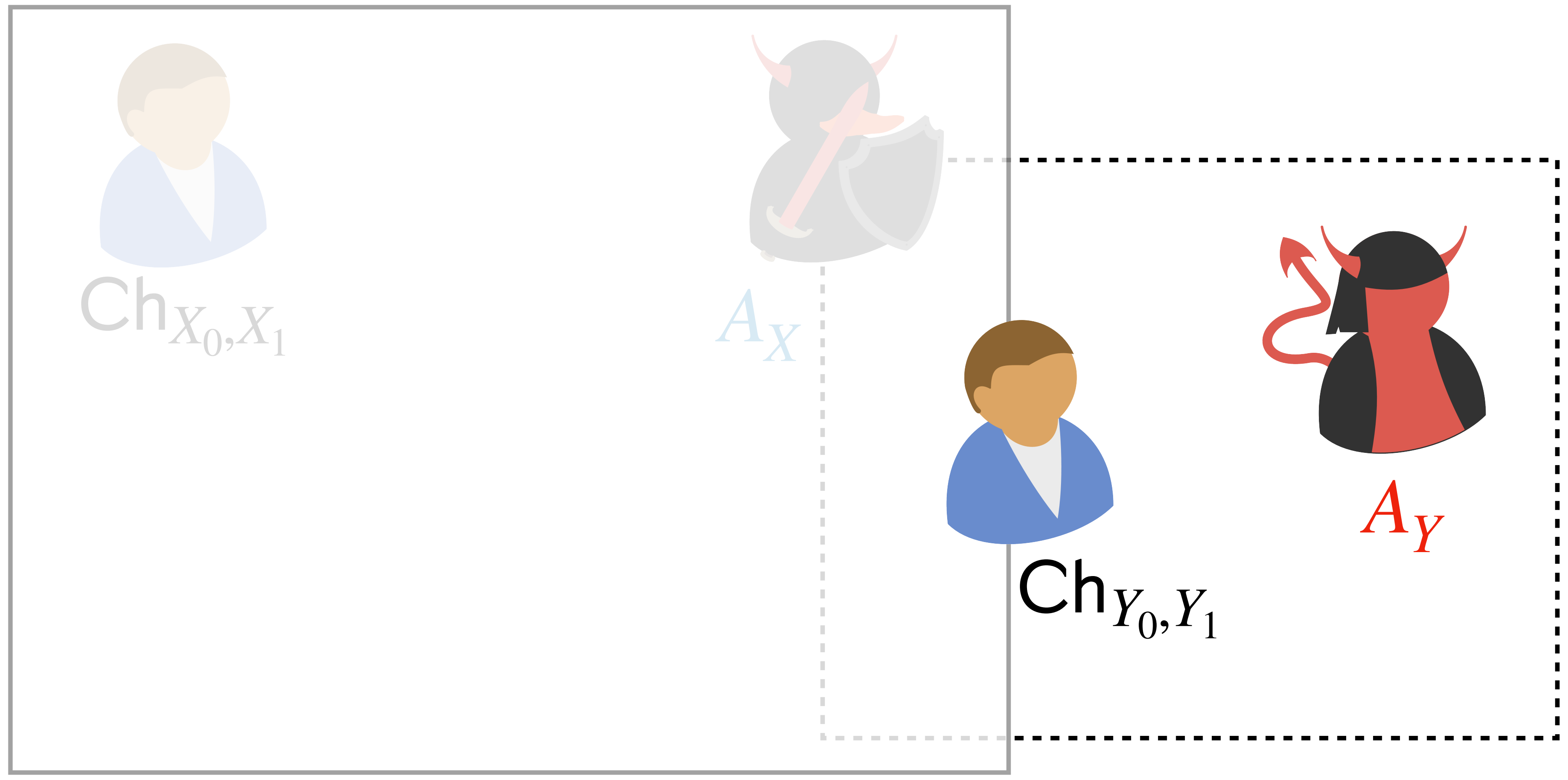
- **Input Mapping:** How can A_X map the input it receives from Ch_{X_0,X_1} to an input for A_Y?
  - A_Y's view must be identical to the indistinguishability game between Y_0 and Y_1. Why?



Reductions: Key Points

Claim: If $X_0 \overset{c}{\approx} X_1$ then $Y_0 \overset{c}{\approx} Y_1$.

Two things to keep in mind when working through a reduction.



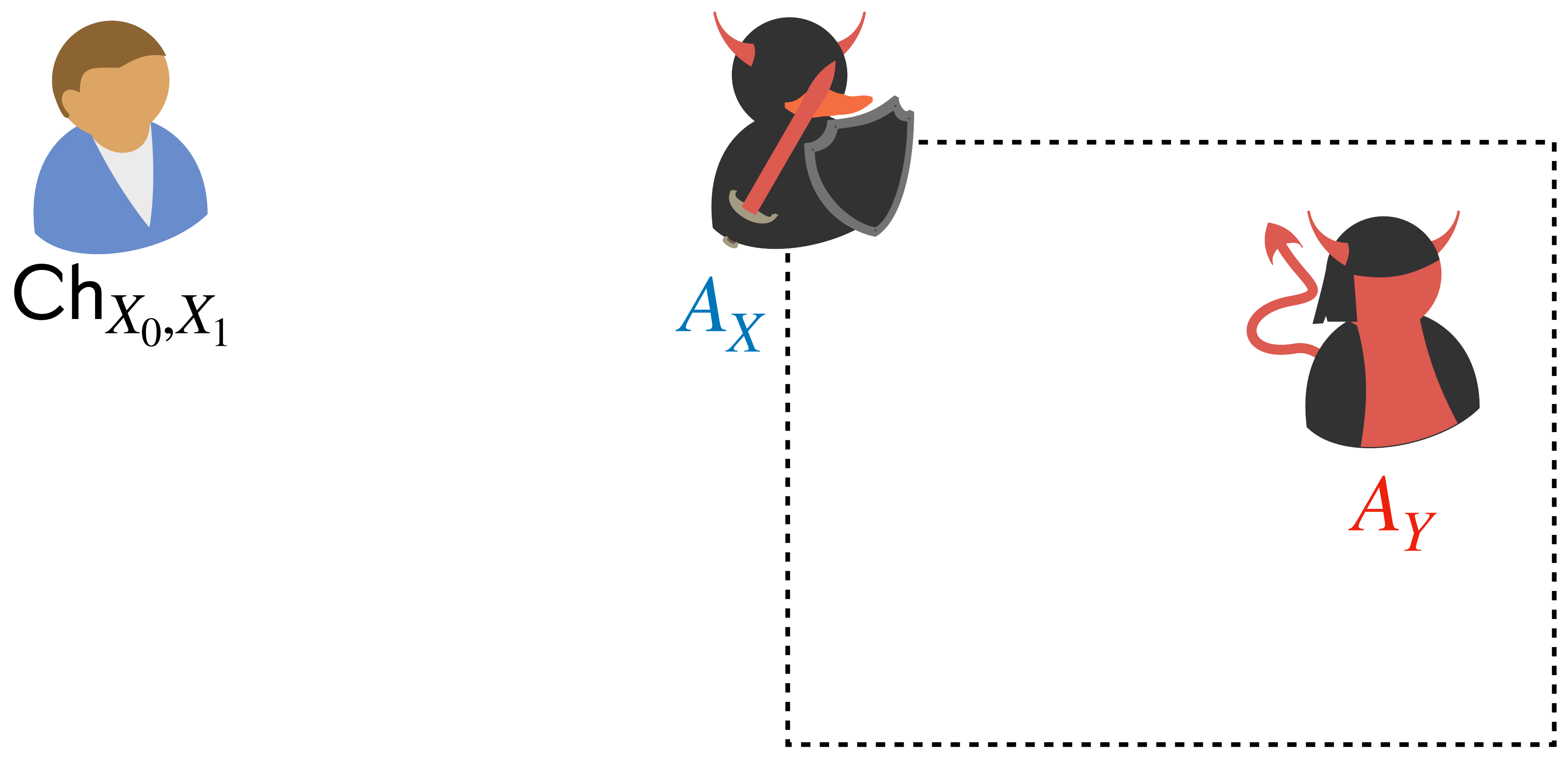
- **Input Mapping:** How can A_X map the input it receives from Ch_{X_0,X_1} to an input for A_Y?
  - A_Y's view must be identical to the indistinguishability game between Y_0 and Y_1. Why?
  - We cannot assume anything about A_Y. All we know is that it succeeds with probability ϵ_Y if we give the input distribution it expects.

Reductions: Key Points

Claim: If $X_0 \overset{c}{\approx} X_1$ then $Y_0 \overset{c}{\approx} Y_1$.

Two things to keep in mind when working through a reduction.

- **Input Mapping:** How can A_X map the input it receives from Ch_{X_0,X_1} to an input for A_Y?
  - A_Y's view must be identical to the indistinguishability game between Y_0 and Y_1. Why?
  - We cannot assume anything about A_Y. All we know is that it succeeds with probability ϵ_Y if we give the input distribution it expects.
- **Output Mapping:** How can A_X map the output it receives from A_Y to an output for Ch_{X_0,X_1}?

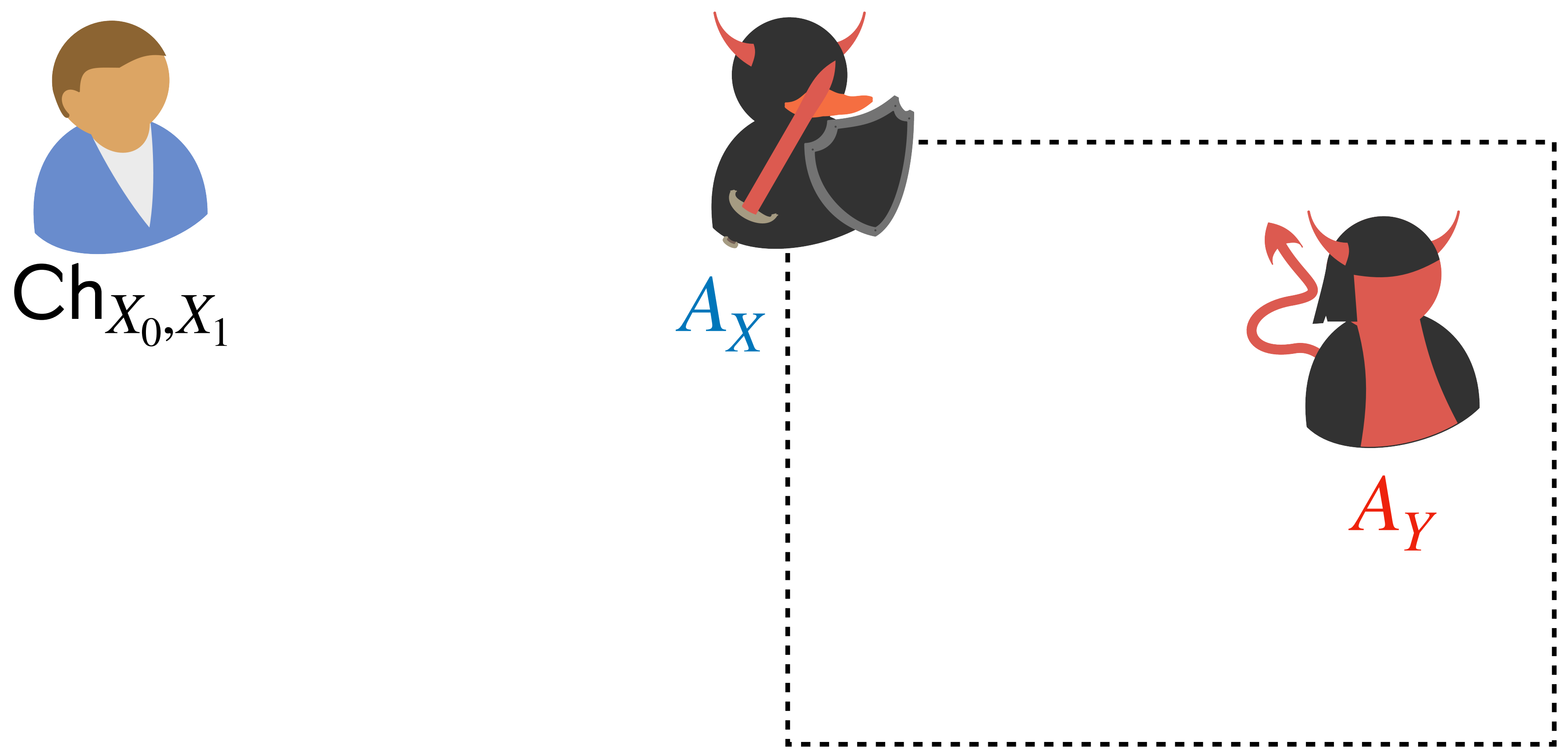


Reductions: Key Points

Claim: If $X_0 \overset{c}{\approx} X_1$ then $Y_0 \overset{c}{\approx} Y_1$.

Two things to keep in mind when working through a reduction.

- **Input Mapping:** How can A_X map the input it receives from Ch_{X_0,X_1} to an input for A_Y?
  - A_Y's view must be identical to the indistinguishability game between Y_0 and Y_1. Why?
  - We cannot assume anything about A_Y. All we know is that it succeeds with probability ϵ_Y if we give the input distribution it expects.
- **Output Mapping:** How can A_X map the output it receives from A_Y to an output for Ch_{X_0,X_1}?
  - A_X should leverage A_Y's distinguishing advantage.



Reductions: Key Points

Claim: If $X_0 \overset{c}{\approx} X_1$ then $Y_0 \overset{c}{\approx} Y_1$.

Two things to keep in mind when working through a reduction.

- **Input Mapping:** How can A_X map the input it receives from Ch_{X_0,X_1} to an input for A_Y?
  - A_Y's view must be identical to the indistinguishability game between Y_0 and Y_1. Why?
  - We cannot assume anything about A_Y. All we know is that it succeeds with probability ϵ_Y if we give the input distribution it expects.
- **Output Mapping:** How can A_X map the output it receives from A_Y to an output for Ch_{X_0,X_1}?
  - A_X should leverage A_Y's distinguishing advantage.
  - We need to relate ϵ_X with ϵ_Y to then argue that ϵ_Y is negligible.