Logistics

Logistics

- Homework 1 grades released

Logistics

- Homework 1 grades released
- Homework 3 due on Thursday (the 12th)

Logistics

- Homework 1 grades released
- Homework 3 due on Thursday (the 12th)
  - Hardest homework so far! Definitely start it soon and come to office hours

Logistics

- Homework 1 grades released
- Homework 3 due on Thursday (the 12th)
  - Hardest homework so far! Definitely start it soon and come to office hours
- Lecture notes on pseudorandomness are on the class website

Logistics

- Homework 1 grades released
- Homework 3 due on Thursday (the 12th)
  - Hardest homework so far! Definitely start it soon and come to office hours
- Lecture notes on pseudorandomness are on the class website
- Midterm next Tuesday (the 17th)

Logistics

- Homework 1 grades released
- Homework 3 due on Thursday (the 12th)
    - Hardest homework so far! Definitely start it soon and come to office hours
- Lecture notes on pseudorandomness are on the class website
- Midterm next Tuesday (the 17th)
    - In class

Logistics

- Homework 1 grades released
- Homework 3 due on Thursday (the 12th)
  - Hardest homework so far! Definitely start it soon and come to office hours
- Lecture notes on pseudorandomness are on the class website
- Midterm next Tuesday (the 17th)
  - In class
  - On paper, no laptops or notes

Logistics

- Homework 1 grades released
- Homework 3 due on Thursday (the 12th)
    - Hardest homework so far! Definitely start it soon and come to office hours
- Lecture notes on pseudorandomness are on the class website
- Midterm next Tuesday (the 17th)
    - In class
    - On paper, no laptops or notes
    - Questions a lot like HW3 and the sections 3 and 4 exercises in Boneh-Shoup

Logistics

- Homework 1 grades released
- Homework 3 due on Thursday (the 12th)
  - Hardest homework so far! Definitely start it soon and come to office hours
- Lecture notes on pseudorandomness are on the class website
- Midterm next Tuesday (the 17th)
  - In class
  - On paper, no laptops or notes
  - Questions a lot like HW3 and the sections 3 and 4 exercises in Boneh-Shoup
  - Topics are everything up through PRFs

Logistics

- Homework 1 grades released
- Homework 3 due on Thursday (the 12th)
  - Hardest homework so far! Definitely start it soon and come to office hours
- Lecture notes on pseudorandomness are on the class website
- Midterm next Tuesday (the 17th)
  - In class
  - On paper, no laptops or notes
  - Questions a lot like HW3 and the sections 3 and 4 exercises in Boneh-Shoup
  - Topics are everything up through PRFs
  - Will give all necessary definitions (e.g. the formal definition of a PRG), you don't have to memorize those

Logistics

- Homework 1 grades released
- Homework 3 due on Thursday (the 12th)
    - Hardest homework so far! Definitely start it soon and come to office hours
- Lecture notes on pseudorandomness are on the class website
- Midterm next Tuesday (the 17th)
    - In class
    - On paper, no laptops or notes
    - Questions a lot like HW3 and the sections 3 and 4 exercises in Boneh-Shoup
    - Topics are everything up through PRFs
    - Will give all necessary definitions (e.g. the formal definition of a PRG), you don't have to memorize those
    - Will start Thursday with a review session on whatever you think would be most useful

Proof Techniques

10th February 2026

Proof Techniques

Proof Techniques

- Proving that a construction satisfies a definition

Proof Techniques

- Proving that a construction satisfies a definition
  - e.g. "If $G : \{0,1\}^{\lambda} \rightarrow \{0,1\}^{\lambda+1}$ is a PRG, then $G'(s) = G(G(s)[1...\lambda])$ is a PRG"

Proof Techniques

- Proving that a construction satisfies a definition
  - e.g. "If $G : \{0,1\}^\lambda \to \{0,1\}^{\lambda+1}$ is a PRG, then $G'(s) = G(G(s)[1...\lambda])$ is a PRG"
  - Done via a *hybrid argument*, possibly with *reductions* to prove that two hybrids are indistinguishable

Proof Techniques

- Proving that a construction satisfies a definition
  - e.g. "If $G : \{0,1\}^\lambda \to \{0,1\}^{\lambda+1}$ is a PRG, then $G'(s) = G(G(s)[1...\lambda])$ is a PRG"
  - Done via a *hybrid argument*, possibly with *reductions* to prove that two hybrids are indistinguishable
- Proving that a construction *does not* satisfy a definition

Proof Techniques

- Proving that a construction satisfies a definition
    - e.g. “If $G : \{0,1\}^\lambda \to \{0,1\}^{\lambda+1}$ is a PRG, then $G'(s) = G(G(s)[1...\lambda])$ is a PRG”
    - Done via a *hybrid argument*, possibly with *reductions* to prove that two hybrids are indistinguishable
- Proving that a construction *does not* satisfy a definition
    - e.g. “If G is a PRG, then $G'(s) = G(s)\,||\,s$ is *not* a PRG”

Proof Techniques

- Proving that a construction satisfies a definition
  - e.g. "If $G : \{0,1\}^\lambda \to \{0,1\}^{\lambda+1}$ is a PRG, then $G'(s) = G(G(s)[1...\lambda])$ is a PRG"
  - Done via a *hybrid argument*, possibly with *reductions* to prove that two hybrids are indistinguishable
- Proving that a construction *does not* satisfy a definition
  - e.g. "If G is a PRG, then $G'(s) = G(s) || s$ is *not* a PRG"
  - Done by specifying an adversary and analyzing its advantage

Proof Techniques

- Proving that a construction satisfies a definition
    - e.g. "If $G : \{0,1\}^\lambda \to \{0,1\}^{\lambda+1}$ is a PRG, then $G'(s) = G(G(s)[1...\lambda])$ is a PRG"
    - Done via a *hybrid argument*, possibly with *reductions* to prove that two hybrids are indistinguishable
- Proving that a construction *does not* satisfy a definition
    - e.g. "If G is a PRG, then $G'(s) = G(s) || s$ is *not* a PRG"
    - Done by specifying an adversary and analyzing its advantage

A reduction also involves specifying an adversary!

Proof Example: PRG

Proof Example: PRG

$G : \{0,1\}^{\lambda} \rightarrow \{0,1\}^{\lambda+1}$ is a PRG

Proof Example: PRG

$G : \{0,1\}^\lambda \to \{0,1\}^{\lambda+1}$ is a PRG

$\underline{G'(s)}$:

return $G(G(s)[1\ldots\lambda])$

Proof Example: PRG

$G : \{0,1\}^{\lambda} \to \{0,1\}^{\lambda+1}$ is a PRG

Given:

$$\left\{ G(s) : s \xleftarrow{\$} \{0,1\}^{\lambda} \right\} \overset{c}{\approx} \left\{ r : r \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

$\underline{G'(s)}$:
return $G(G(s)[1\ldots\lambda])$

Proof Example: PRG

$G : \{0,1\}^\lambda \rightarrow \{0,1\}^{\lambda+1}$ is a PRG

Given:

$$\left\{ G(s) : s \xleftarrow{\$} \{0,1\}^\lambda \right\} \overset{c}{\approx} \left\{ r : r \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

Must show that:

$$\left\{ G'(s) : s \xleftarrow{\$} \{0,1\}^\lambda \right\} \overset{c}{\approx} \left\{ r : r \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

```
G'(s) :
  return G(G(s)[1...λ])
```

Proof Example: PRG

$G : \{0,1\}^\lambda \to \{0,1\}^{\lambda+1}$ is a PRG

Given:

$$\left\{ G(s) : s \xleftarrow{\$} \{0,1\}^\lambda \right\} \overset{c}{\approx} \left\{ r : r \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

Must show that:

$$\left\{ G'(s) : s \xleftarrow{\$} \{0,1\}^\lambda \right\} \overset{c}{\approx} \left\{ r : r \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

$$H_0 : \left\{ G(G(s)[1\ldots\lambda]) : s \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

$\underline{G'(s)}$:

return $G(G(s)[1\ldots\lambda])$

Proof Example: PRG

$G : \{0,1\}^\lambda \to \{0,1\}^{\lambda+1}$ is a PRG

Given:

$$\left\{ G(s) : s \xleftarrow{\$} \{0,1\}^\lambda \right\} \overset{c}{\approx} \left\{ r : r \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

Must show that:

$$\left\{ G'(s) : s \xleftarrow{\$} \{0,1\}^\lambda \right\} \overset{c}{\approx} \left\{ r : r \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

$\underline{G'(s)}$:

return $G(G(s)[1\ldots\lambda])$

$$H_0 : \left\{ G(G(s)[1\ldots\lambda]) : s \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

$$H_1 : \left\{ G(r[1\ldots\lambda]) : r \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

Proof Example: PRG

$G : \{0,1\}^\lambda \rightarrow \{0,1\}^{\lambda+1}$ is a PRG

Given:

$$\left\{ G(s) : s \xleftarrow{\$} \{0,1\}^\lambda \right\} \overset{c}{\approx} \left\{ r : r \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

Must show that:

$$\left\{ G'(s) : s \xleftarrow{\$} \{0,1\}^\lambda \right\} \overset{c}{\approx} \left\{ r : r \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

$\underline{G'(s)}$:
  return $G(G(s)[1\ldots\lambda])$

$$H_0 : \left\{ G(G(s)[1\ldots\lambda]) : s \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

$$H_1 : \left\{ G(r[1\ldots\lambda]) : r \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

If $\exists$ an adversary A_{H_0,H_1} that distinguishes between H_0 and H_1 with probability $\nu(\lambda)$, then $\exists$ an adversary that distinguishes between $G(s)$ and $r \xleftarrow{\$} \{0,1\}^\lambda$ with probability $\nu(\lambda)$

Proof Example: PRG

$G : \{0,1\}^{\lambda} \rightarrow \{0,1\}^{\lambda+1}$ is a PRG

Given:

$$\left\{ G(s) : s \xleftarrow{\$} \{0,1\}^{\lambda} \right\} \overset{c}{\approx} \left\{ r : r \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

Must show that:

$$\left\{ G'(s) : s \xleftarrow{\$} \{0,1\}^{\lambda} \right\} \overset{c}{\approx} \left\{ r : r \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

$\underline{G'(s)}$:
return $G(G(s)[1\ldots\lambda])$

$$H_0 : \left\{ G(G(s)[1\ldots\lambda]) : s \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

$$H_1 : \left\{ G(r[1\ldots\lambda]) : r \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

C_G

If $\exists$ an adversary A_{H_0,H_1} that distinguishes between H_0 and H_1 with probability $\nu(\lambda)$, then $\exists$ an adversary that distinguishes between $G(s)$ and $r \xleftarrow{\$} \{0,1\}^{\lambda}$ with probability $\nu(\lambda)$

Proof Example: PRG

$G : \{0,1\}^{\lambda} \rightarrow \{0,1\}^{\lambda+1}$ is a PRG

Given:

$$\left\{ G(s) : s \xleftarrow{\$} \{0,1\}^{\lambda} \right\} \overset{c}{\approx} \left\{ r : r \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

Must show that:

$$\left\{ G'(s) : s \xleftarrow{\$} \{0,1\}^{\lambda} \right\} \overset{c}{\approx} \left\{ r : r \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

$\underline{G'(s)}$:

return $G(G(s)[1\ldots\lambda])$

$$H_0 : \left\{ G(G(s)[1\ldots\lambda]) : s \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

$$H_1 : \left\{ G(r[1\ldots\lambda]) : r \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

C_G

A_G

If $\exists$ an adversary A_{H_0,H_1} that distinguishes between H_0 and H_1 with probability $\nu(\lambda)$, then $\exists$ an adversary that distinguishes between $G(s)$ and $r \xleftarrow{\$} \{0,1\}^{\lambda}$ with probability $\nu(\lambda)$

Proof Example: PRG

$G : \{0,1\}^\lambda \rightarrow \{0,1\}^{\lambda+1}$ is a PRG

Given:

$$\left\{ G(s) : s \xleftarrow{\$} \{0,1\}^\lambda \right\} \overset{c}{\approx} \left\{ r : r \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

Must show that:

$$\left\{ G'(s) : s \xleftarrow{\$} \{0,1\}^\lambda \right\} \overset{c}{\approx} \left\{ r : r \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

$\underline{G'(s)}$:

return $G(G(s)[1\ldots\lambda])$

$$H_0 : \left\{ G(G(s)[1\ldots\lambda]) : s \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

$$H_1 : \left\{ G(r[1\ldots\lambda]) : r \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

C_G

If $\exists$ an adversary A_{H_0,H_1} that distinguishes between H_0 and H_1 with probability $\nu(\lambda)$, then $\exists$ an adversary that distinguishes between $G(s)$ and $r \xleftarrow{\$} \{0,1\}^\lambda$ with probability $\nu(\lambda)$

Proof Example: PRG

$G : \{0,1\}^\lambda \to \{0,1\}^{\lambda+1}$ is a PRG

Given:

$$\left\{ G(s) : s \xleftarrow{\$} \{0,1\}^\lambda \right\} \stackrel{c}{\approx} \left\{ r : r \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

Must show that:

$$\left\{ G'(s) : s \xleftarrow{\$} \{0,1\}^\lambda \right\} \stackrel{c}{\approx} \left\{ r : r \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

$\underline{G'(s)}$:
return $G(G(s)[1\ldots\lambda])$

$$H_0 : \left\{ G(G(s)[1\ldots\lambda]) : s \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

$$H_1 : \left\{ G(r[1\ldots\lambda]) : r \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

If $\exists$ an adversary A_{H_0,H_1} that distinguishes between H_0 and H_1 with probability $\nu(\lambda)$, then $\exists$ an adversary that distinguishes between $G(s)$ and $r \xleftarrow{\$} \{0,1\}^\lambda$ with probability $\nu(\lambda)$

C_G

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

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

$r_0 := G(s)$

$r_1 \xleftarrow{\$} \{0,1\}^{\lambda+1}$

Proof Example: PRG

$G : \{0,1\}^\lambda \to \{0,1\}^{\lambda+1}$ is a PRG

$\underline{G'(s)}$:

return $G(G(s)[1\ldots\lambda])$

Given:

$$\left\{ G(s) : s \xleftarrow{\$} \{0,1\}^\lambda \right\} \overset{c}{\approx} \left\{ r : r \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

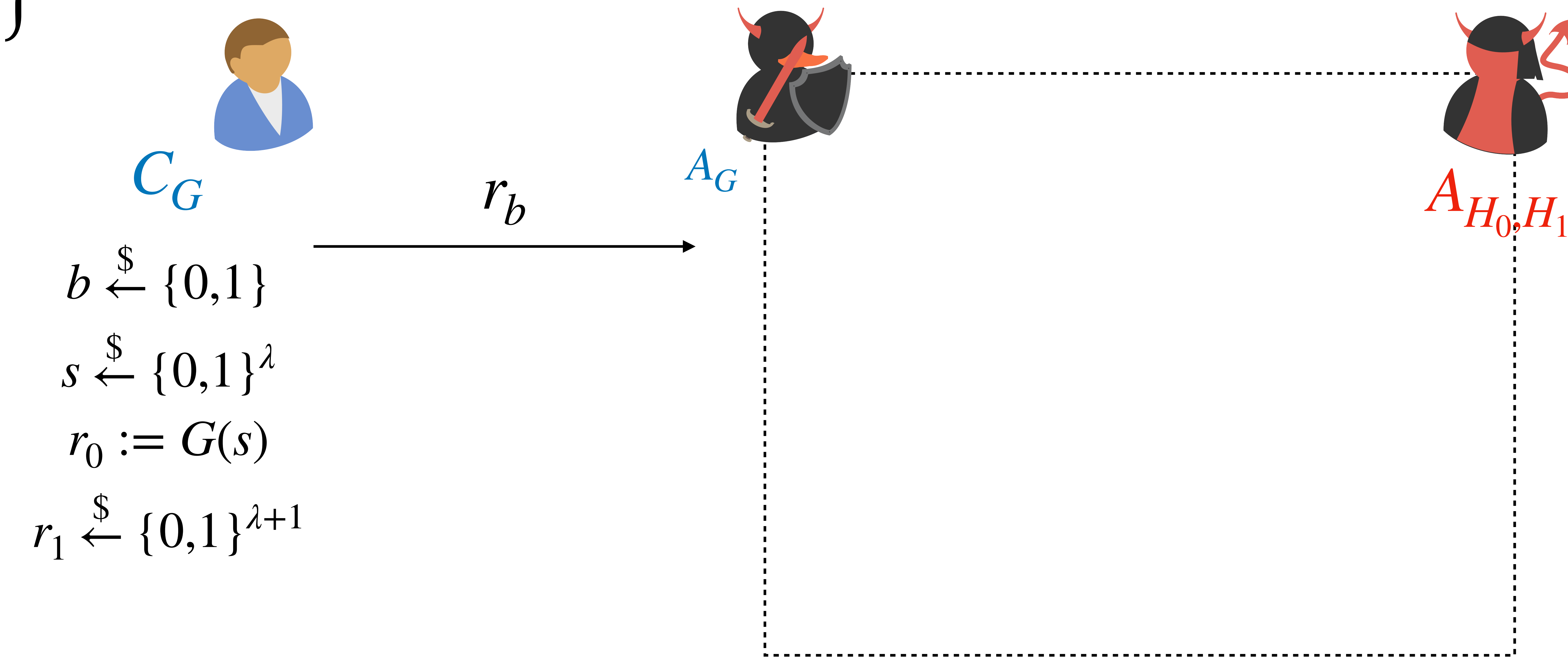
Must show that:

$$\left\{ G'(s) : s \xleftarrow{\$} \{0,1\}^\lambda \right\} \overset{c}{\approx} \left\{ r : r \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

$$H_0 : \left\{ G(G(s)[1\ldots\lambda]) : s \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

$$H_1 : \left\{ G(r[1\ldots\lambda]) : r \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

If $\exists$ an adversary A_{H_0,H_1} that distinguishes between H_0 and H_1 with probability $\nu(\lambda)$, then $\exists$ an adversary that distinguishes between $G(s)$ and $r \xleftarrow{\$} \{0,1\}^\lambda$ with probability $\nu(\lambda)$



Proof Example: PRG

$G : \{0,1\}^{\lambda} \rightarrow \{0,1\}^{\lambda+1}$ is a PRG

$\underline{G'(s)}$:

return $G(G(s)[1\ldots\lambda])$

Given:

$$\left\{ G(s) : s \xleftarrow{\$} \{0,1\}^{\lambda} \right\} \overset{c}{\approx} \left\{ r : r \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

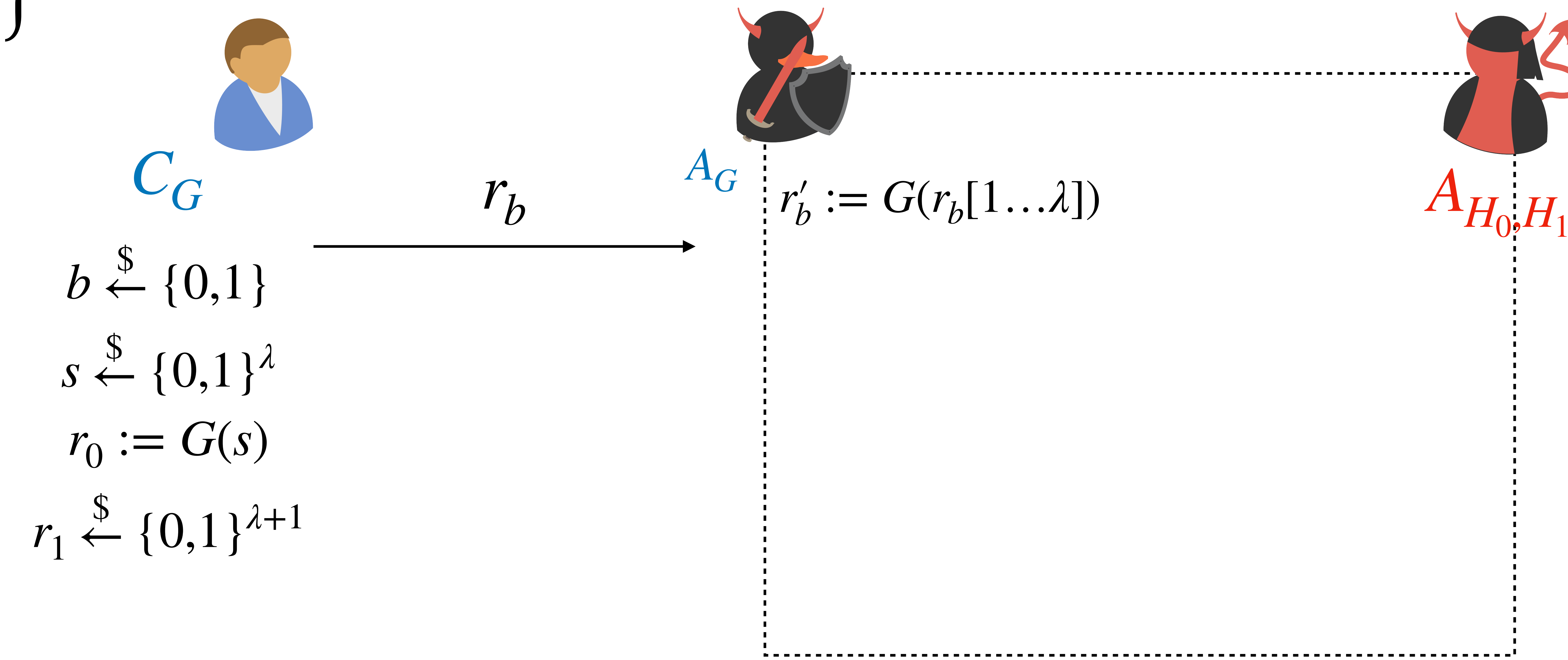
Must show that:

$$\left\{ G'(s) : s \xleftarrow{\$} \{0,1\}^{\lambda} \right\} \overset{c}{\approx} \left\{ r : r \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

$$H_0 : \left\{ G(G(s)[1\ldots\lambda]) : s \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

$$H_1 : \left\{ G(r[1\ldots\lambda]) : r \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

If $\exists$ an adversary A_{H_0,H_1} that distinguishes between H_0 and H_1 with probability $\nu(\lambda)$, then $\exists$ an adversary that distinguishes between $G(s)$ and $r \xleftarrow{\$} \{0,1\}^{\lambda}$ with probability $\nu(\lambda)$



Proof Example: PRG

$G : \{0,1\}^\lambda \to \{0,1\}^{\lambda+1}$ is a PRG

Given:

$$\left\{ G(s) : s \xleftarrow{\$} \{0,1\}^\lambda \right\} \overset{c}{\approx} \left\{ r : r \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

Must show that:

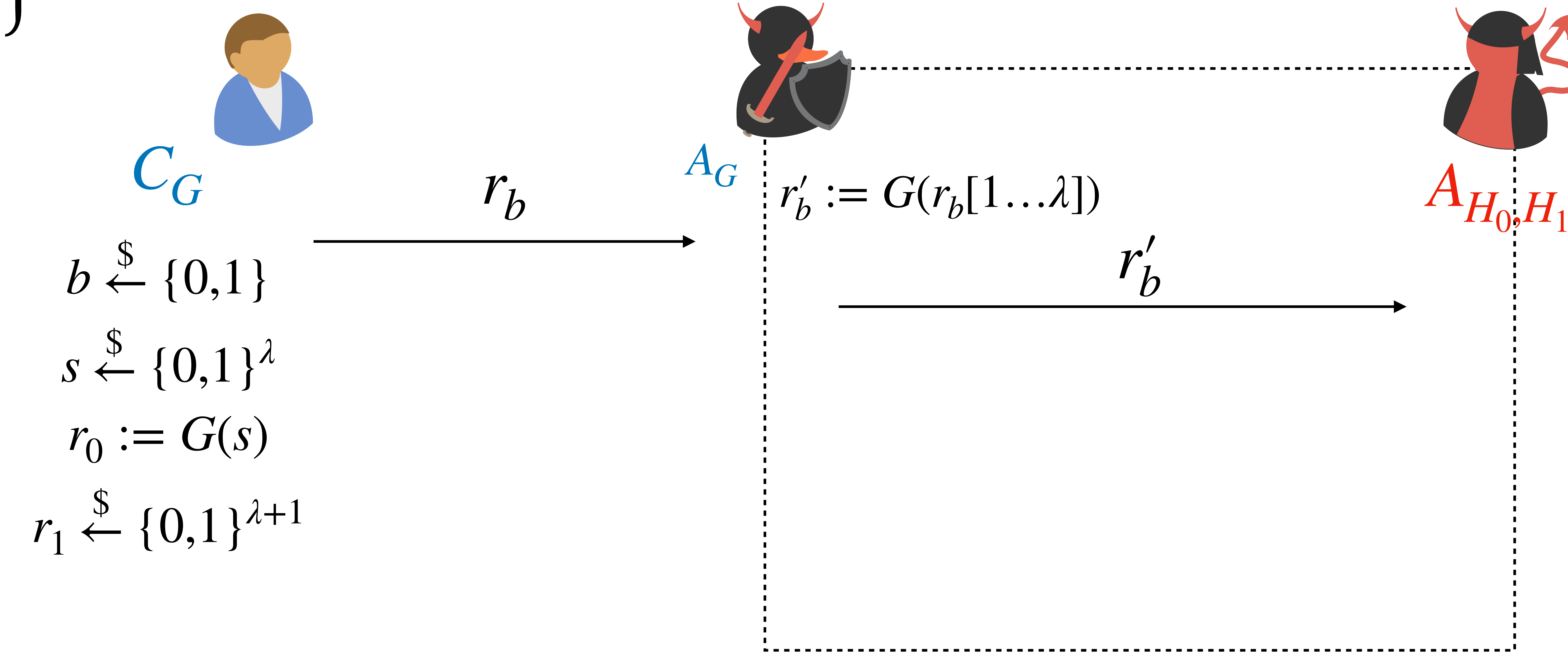
$$\left\{ G'(s) : s \xleftarrow{\$} \{0,1\}^\lambda \right\} \overset{c}{\approx} \left\{ r : r \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

$\underline{G'(s)}$:
return $G(G(s)[1\ldots\lambda])$

$$H_0 : \left\{ G(G(s)[1\ldots\lambda]) : s \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

$$H_1 : \left\{ G(r[1\ldots\lambda]) : r \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

If ∃ an adversary A_{H_0,H_1} that distinguishes between H_0 and H_1 with probability $\nu(\lambda)$, then ∃ an adversary that distinguishes between $G(s)$ and $r \xleftarrow{\$} \{0,1\}^\lambda$ with probability $\nu(\lambda)$



Proof Example: PRG

$G : \{0,1\}^\lambda \to \{0,1\}^{\lambda+1}$ is a PRG

$$\underline{G'(s)}:$$
return $G(G(s)[1\ldots\lambda])$

Given:

$$\left\{ G(s) : s \xleftarrow{\$} \{0,1\}^\lambda \right\} \stackrel{c}{\approx} \left\{ r : r \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

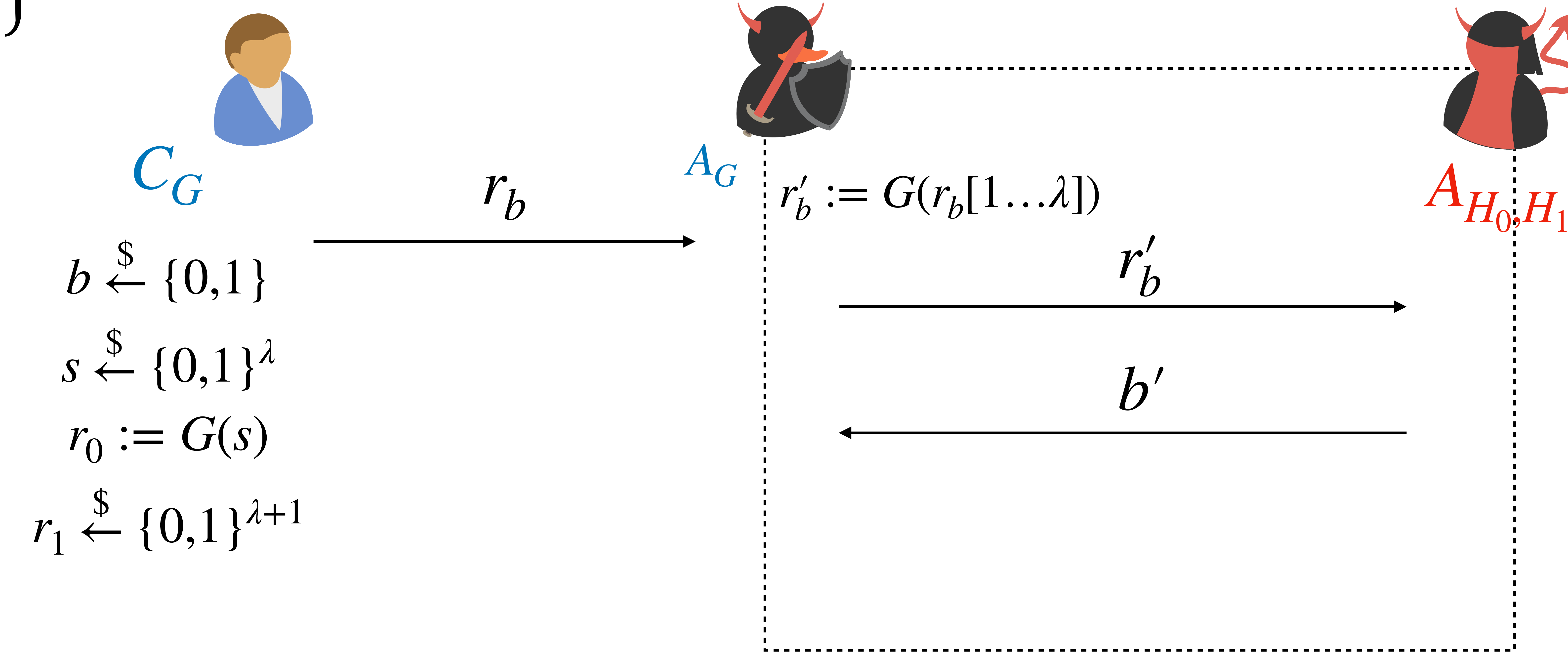
Must show that:

$$\left\{ G'(s) : s \xleftarrow{\$} \{0,1\}^\lambda \right\} \stackrel{c}{\approx} \left\{ r : r \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

$$H_0 : \left\{ G(G(s)[1\ldots\lambda]) : s \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

$$H_1 : \left\{ G(r[1\ldots\lambda]) : r \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

If $\exists$ an adversary A_{H_0,H_1} that distinguishes between H_0 and H_1 with probability $\nu(\lambda)$, then $\exists$ an adversary that distinguishes between $G(s)$ and $r \xleftarrow{\$} \{0,1\}^\lambda$ with probability $\nu(\lambda)$



Proof Example: PRG

$G : \{0,1\}^{\lambda} \rightarrow \{0,1\}^{\lambda+1}$ is a PRG

$\underline{G'(s)}$:

return $G(G(s)[1\ldots\lambda])$

Given:

$$\left\{ G(s) : s \xleftarrow{\$} \{0,1\}^{\lambda} \right\} \overset{c}{\approx} \left\{ r : r \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

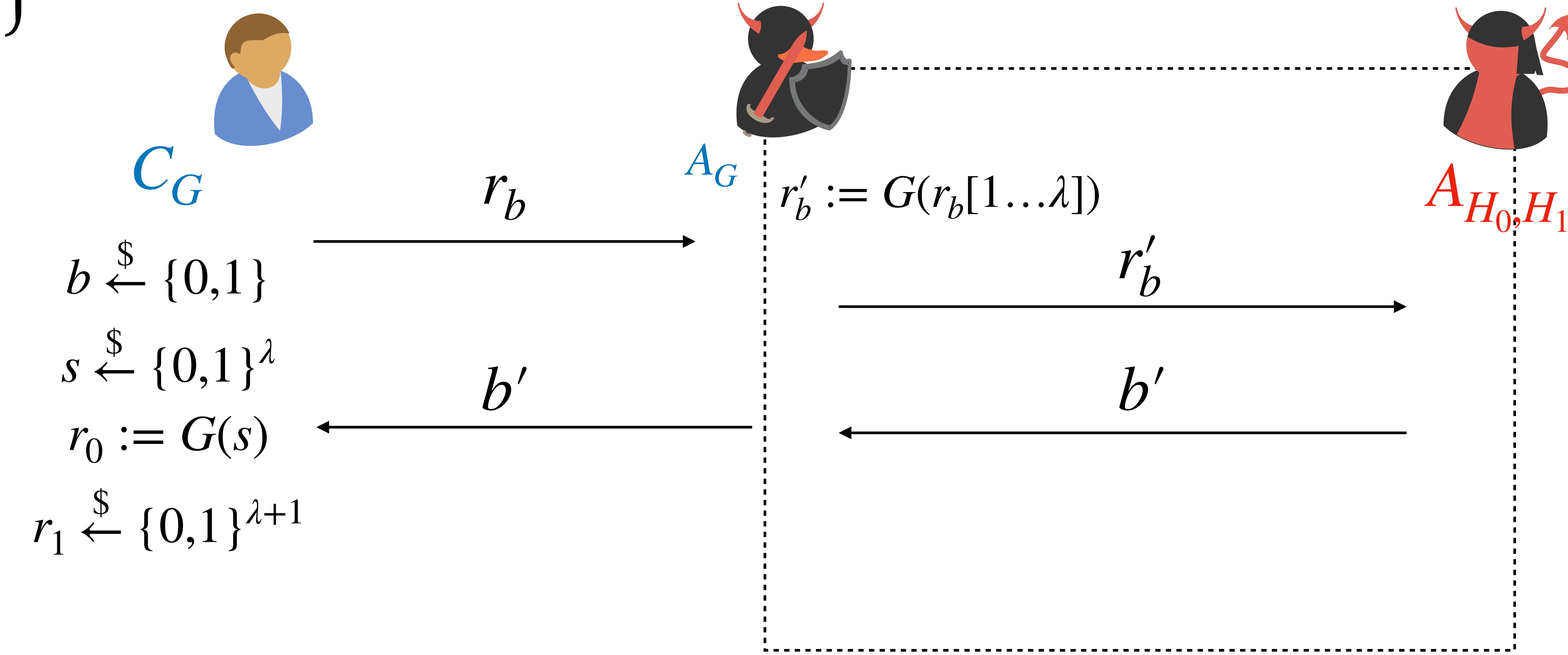
Must show that:

$$\left\{ G'(s) : s \xleftarrow{\$} \{0,1\}^{\lambda} \right\} \overset{c}{\approx} \left\{ r : r \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

$$H_0 : \left\{ G(G(s)[1\ldots\lambda]) : s \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

$$H_1 : \left\{ G(r[1\ldots\lambda]) : r \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

If $\exists$ an adversary A_{H_0,H_1} that distinguishes between H_0 and H_1 with probability $\nu(\lambda)$, then $\exists$ an adversary that distinguishes between $G(s)$ and $r \xleftarrow{\$} \{0,1\}^{\lambda}$ with probability $\nu(\lambda)$



Proof Example: PRG

$G : \{0,1\}^\lambda \to \{0,1\}^{\lambda+1}$ is a PRG

$\underline{G'(s)}$:

return $G(G(s)[1\ldots\lambda])$

Given:

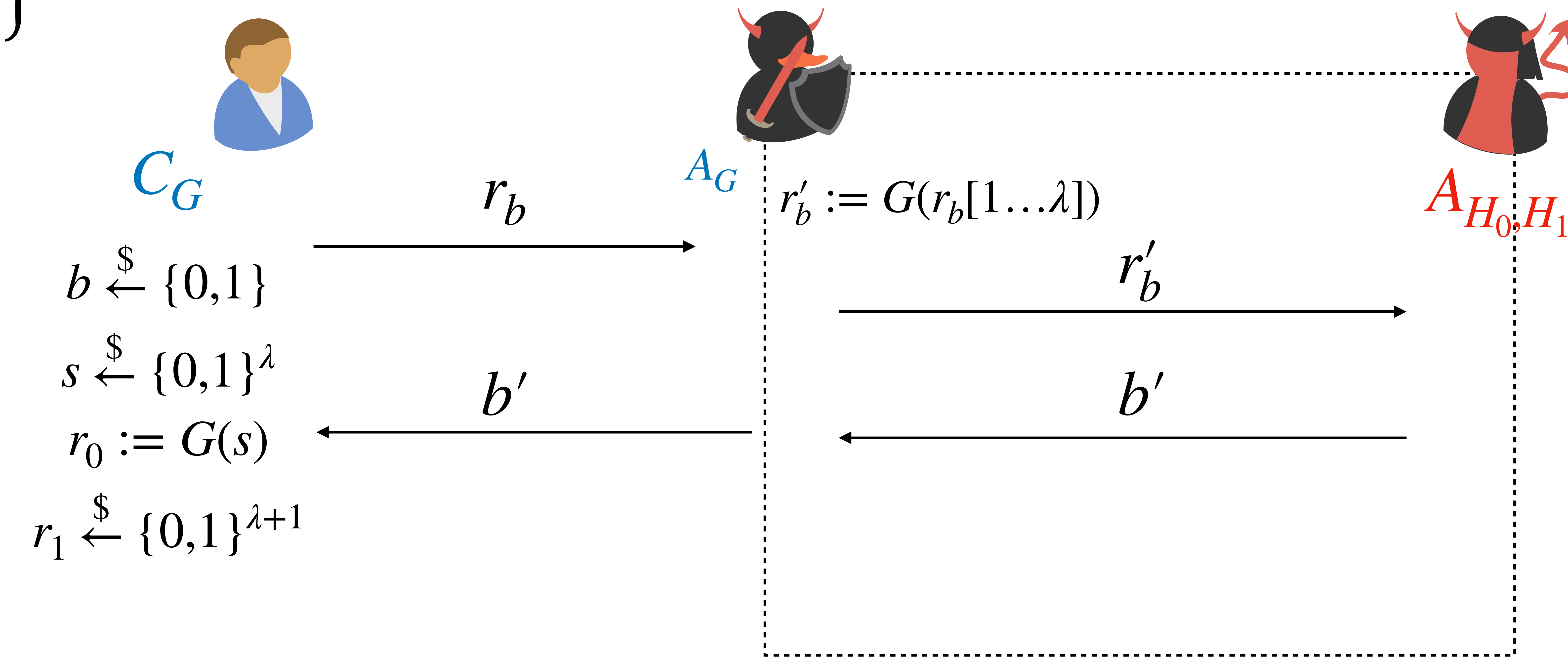
$$\left\{ G(s) : s \xleftarrow{\$} \{0,1\}^\lambda \right\} \overset{c}{\approx} \left\{ r : r \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

Must show that:

$$\left\{ G'(s) : s \xleftarrow{\$} \{0,1\}^\lambda \right\} \overset{c}{\approx} \left\{ r : r \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

$$H_0 : \left\{ G(G(s)[1\ldots\lambda]) : s \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

$$H_1 : \left\{ G(r[1\ldots\lambda]) : r \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$



Proof Example: PRG

$G : \{0,1\}^{\lambda} \to \{0,1\}^{\lambda+1}$ is a PRG

Given:

$$\left\{ G(s) : s \xleftarrow{\$} \{0,1\}^{\lambda} \right\} \overset{c}{\approx} \left\{ r : r \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

Must show that:

$$\left\{ G'(s) : s \xleftarrow{\$} \{0,1\}^{\lambda} \right\} \overset{c}{\approx} \left\{ r : r \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

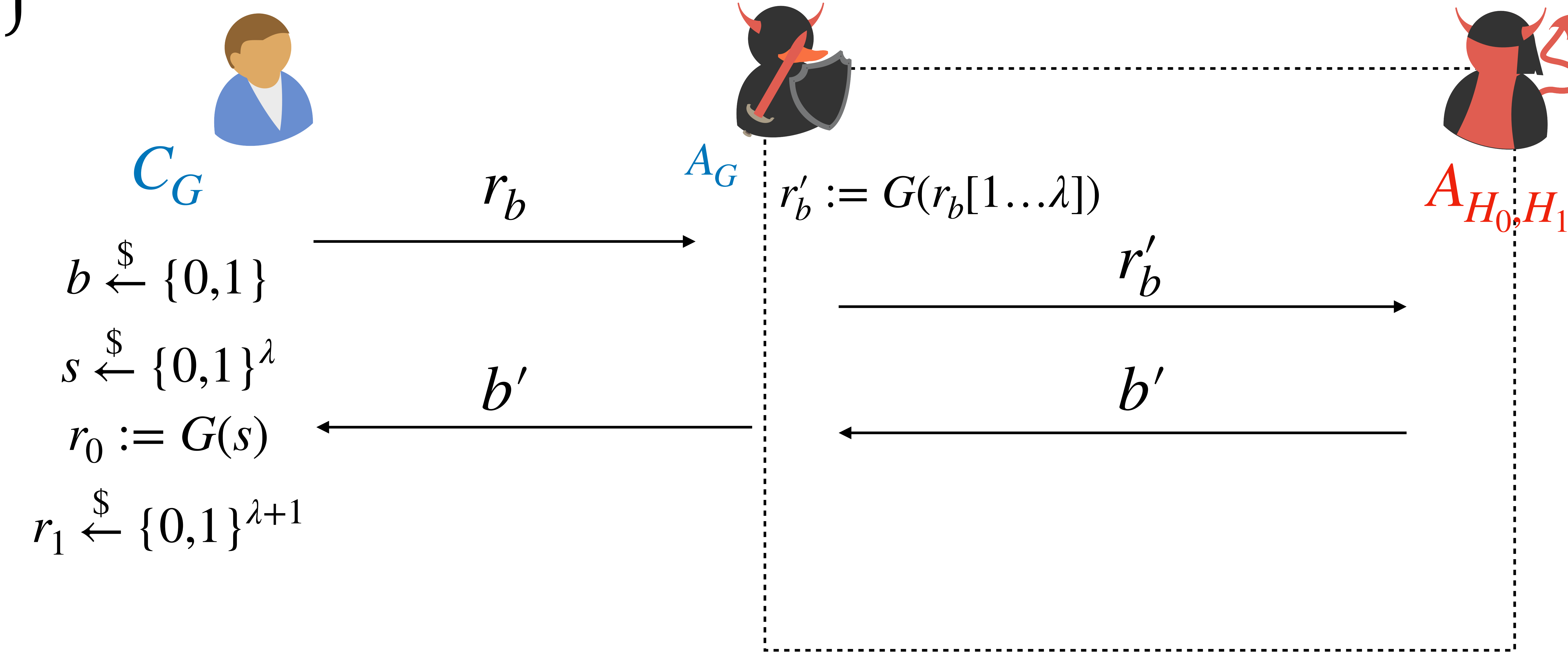
$\underline{G'(s)}$:

return $G(G(s)[1\ldots\lambda])$

$$H_0 : \left\{ G(G(s)[1\ldots\lambda]) : s \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

$$H_1 : \left\{ G(r[1\ldots\lambda]) : r \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

When $b = 0$, A_{H_0,H_1} sees $G(G(s)[1\ldots\lambda])$, the same as in H_0!



Proof Example: PRG

$G : \{0,1\}^\lambda \to \{0,1\}^{\lambda+1}$ is a PRG

Given:

$$\left\{ G(s) : s \xleftarrow{\$} \{0,1\}^\lambda \right\} \overset{c}{\approx} \left\{ r : r \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

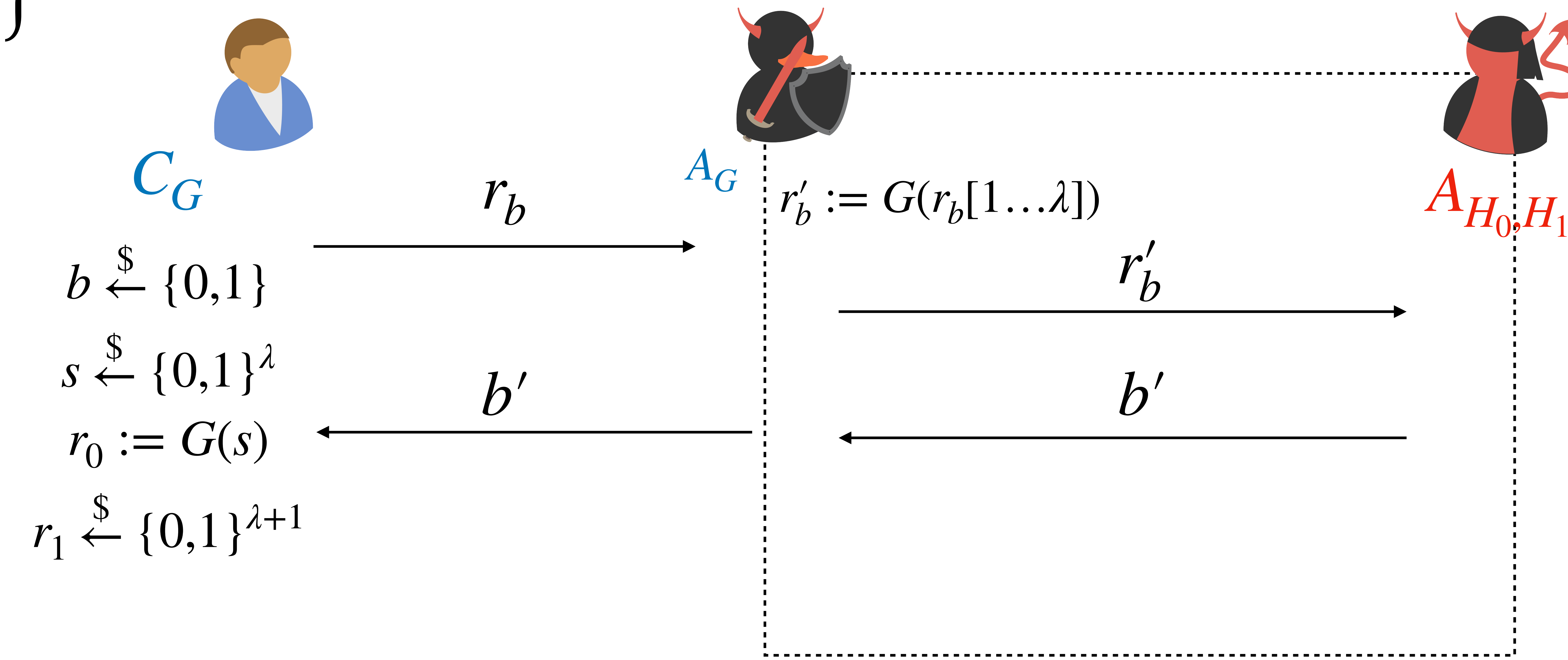
Must show that:

$$\left\{ G'(s) : s \xleftarrow{\$} \{0,1\}^\lambda \right\} \overset{c}{\approx} \left\{ r : r \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

$\underline{G'(s)}$:
return $G(G(s)[1\ldots\lambda])$

$$H_0 : \left\{ G(G(s)[1\ldots\lambda]) : s \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

$$H_1 : \left\{ G(r[1\ldots\lambda]) : r \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$



When $b = 0$, A_{H_0,H_1} sees $G(G(s)[1\ldots\lambda])$, the same as in H_0!

When $b = 1$, A_{H_0,H_1} sees $G(r[1\ldots\lambda])$, the same as in H_1!

Proof Example: PRG

$G : \{0,1\}^\lambda \to \{0,1\}^{\lambda+1}$ is a PRG

$G'(s)$:
  return $G(G(s)[1\ldots\lambda])$

Given:

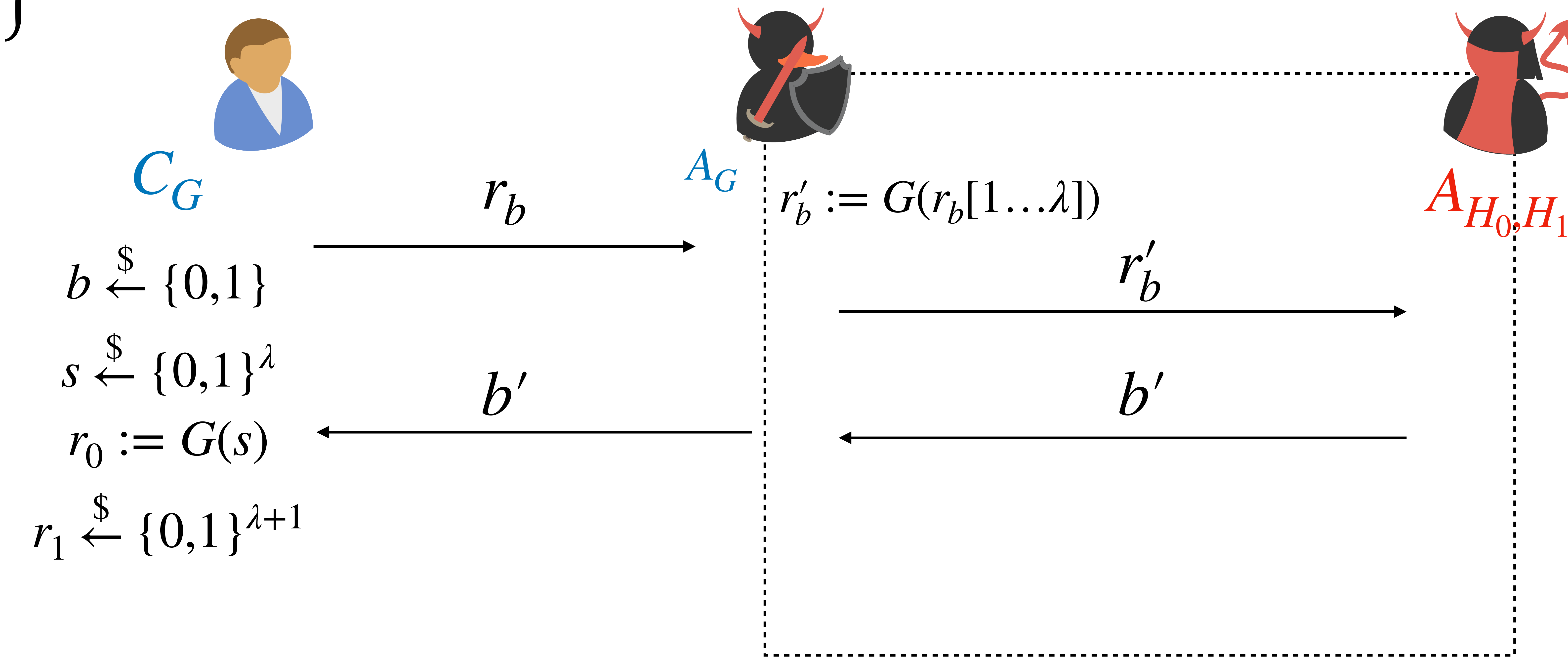
$$\left\{ G(s) : s \xleftarrow{\$} \{0,1\}^\lambda \right\} \stackrel{c}{\approx} \left\{ r : r \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

Must show that:

$$\left\{ G'(s) : s \xleftarrow{\$} \{0,1\}^\lambda \right\} \stackrel{c}{\approx} \left\{ r : r \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

$$H_0 : \left\{ G(G(s)[1\ldots\lambda]) : s \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

$$H_1 : \left\{ G(r[1\ldots\lambda]) : r \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$



When $b = 0$, A_{H_0,H_1} sees $G(G(s)[1\ldots\lambda])$, the same as in H_0!

When $b = 1$, A_{H_0,H_1} sees $G(r[1\ldots\lambda])$, the same as in H_1!

Therefore, $Pr[A_G \text{ wins}] = \Pr[A_{H_0,H_1} \text{ wins}]$

Proof Example: PRG

$G : \{0,1\}^\lambda \to \{0,1\}^{\lambda+1}$ is a PRG

$\underline{G'(s)}$:
return $G(G(s)[1\ldots\lambda])$

Given:

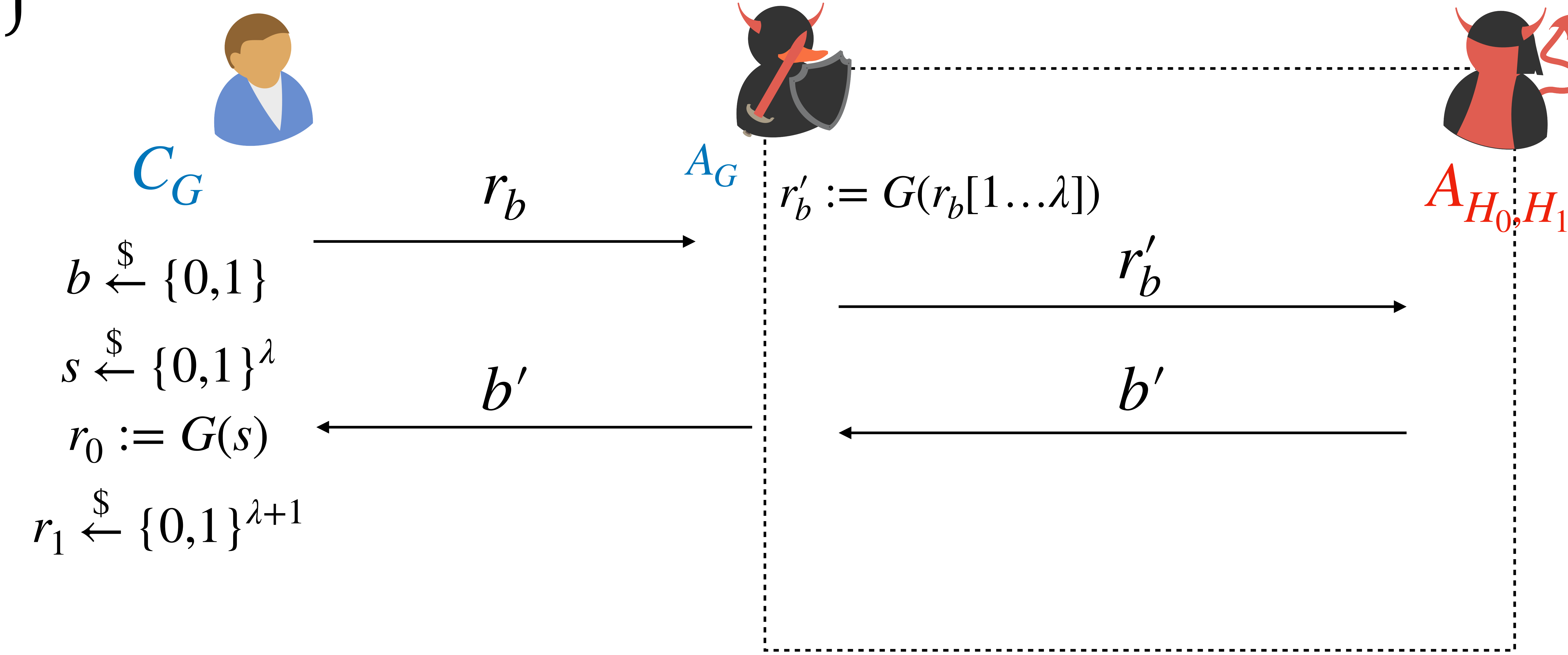
$$\left\{G(s) : s \xleftarrow{\$} \{0,1\}^\lambda\right\} \stackrel{c}{\approx} \left\{r : r \xleftarrow{\$} \{0,1\}^{\lambda+1}\right\}$$

Must show that:

$$\left\{G'(s) : s \xleftarrow{\$} \{0,1\}^\lambda\right\} \stackrel{c}{\approx} \left\{r : r \xleftarrow{\$} \{0,1\}^{\lambda+1}\right\}$$

$$H_0 : \left\{G(G(s)[1\ldots\lambda]) : s \xleftarrow{\$} \{0,1\}^{\lambda+1}\right\}$$

$$H_1 : \left\{G(r[1\ldots\lambda]) : r \xleftarrow{\$} \{0,1\}^{\lambda+1}\right\}$$



When $b = 0$, A_{H_0,H_1} sees $G(G(s)[1\ldots\lambda])$, the same as in H_0!

When $b = 1$, A_{H_0,H_1} sees $G(r[1\ldots\lambda])$, the same as in H_1!

Therefore, $\text{negl}(\lambda) = \Pr[A_{H_0,H_1}$ wins$]$

Proof Example: PRG

$G : \{0,1\}^\lambda \to \{0,1\}^{\lambda+1}$ is a PRG

Given:

$$\left\{ G(s) : s \xleftarrow{\$} \{0,1\}^\lambda \right\} \overset{c}{\approx} \left\{ r : r \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

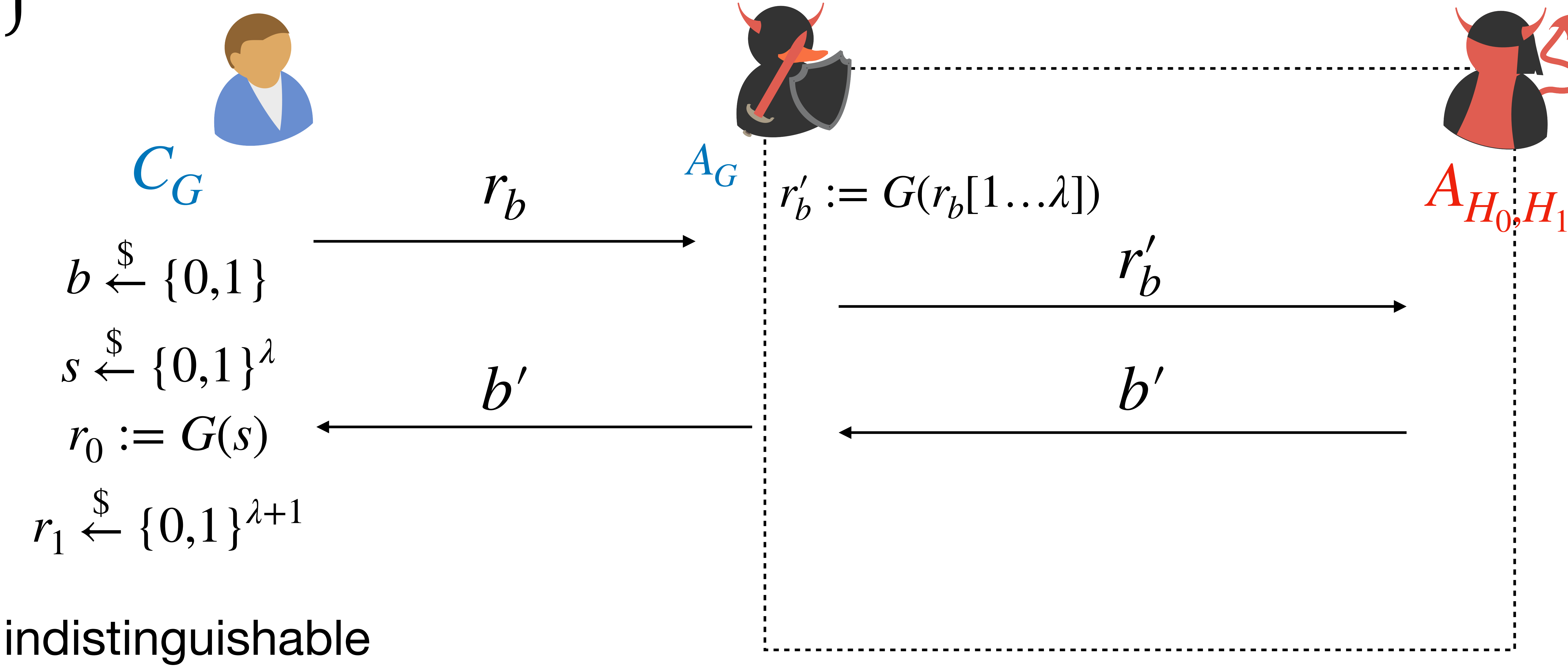
Must show that:

$$\left\{ G'(s) : s \xleftarrow{\$} \{0,1\}^\lambda \right\} \overset{c}{\approx} \left\{ r : r \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

$\underline{G'(s)}$:
return $G(G(s)[1\ldots\lambda])$

$$H_0 : \left\{ G(G(s)[1\ldots\lambda]) : s \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

$$H_1 : \left\{ G(r[1\ldots\lambda]) : r \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$



When $b = 0$, A_{H_0,H_1} sees $G(G(s)[1\ldots\lambda])$, the same as in H_0!

When $b = 1$, A_{H_0,H_1} sees $G(r[1\ldots\lambda])$, the same as in H_1!

Therefore, H_0 and H_1 are computationally indistinguishable

Proof Example: PRG

$G : \{0,1\}^\lambda \rightarrow \{0,1\}^{\lambda+1}$ is a PRG

Given:

$$\left\{ G(s) : s \xleftarrow{\$} \{0,1\}^\lambda \right\} \stackrel{c}{\approx} \left\{ r : r \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

Must show that:

$$\left\{ G'(s) : s \xleftarrow{\$} \{0,1\}^\lambda \right\} \stackrel{c}{\approx} \left\{ r : r \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

$$H_0 : \left\{ G(G(s)[1\ldots\lambda]) : s \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

$$H_1 : \left\{ G(r[1\ldots\lambda]) : r \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

$\underline{G'(s)}$:

return $G(G(s)[1\ldots\lambda])$

Proof Example: PRG

$G : \{0,1\}^\lambda \to \{0,1\}^{\lambda+1}$ is a PRG

Given:

$$\left\{ G(s) : s \xleftarrow{\$} \{0,1\}^\lambda \right\} \overset{c}{\approx} \left\{ r : r \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

Must show that:

$$\left\{ G'(s) : s \xleftarrow{\$} \{0,1\}^\lambda \right\} \overset{c}{\approx} \left\{ r : r \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

$\underline{G'(s)}$:
return $G(G(s)[1\ldots\lambda])$

$$H_0 : \left\{ G(G(s)[1\ldots\lambda]) : s \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

$$H_1 : \left\{ G(r[1\ldots\lambda]) : r \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

$$H_2 : \left\{ r : r \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

Proof Example: PRG

$G : \{0,1\}^\lambda \to \{0,1\}^{\lambda+1}$ is a PRG

Given:

$$\left\{ G(s) : s \xleftarrow{\$} \{0,1\}^\lambda \right\} \overset{c}{\approx} \left\{ r : r \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

Must show that:

$$\left\{ G'(s) : s \xleftarrow{\$} \{0,1\}^\lambda \right\} \overset{c}{\approx} \left\{ r : r \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

$\underline{G'(s)}$:

return $G(G(s)[1\ldots\lambda])$

$$H_0 : \left\{ G(G(s)[1\ldots\lambda]) : s \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

$$H_1 : \left\{ G(r[1\ldots\lambda]) : r \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

$$H_2 : \left\{ r : r \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

Prove indistinguishable via a reduction

Proof Example: PRG

$G : \{0,1\}^\lambda \to \{0,1\}^{\lambda+1}$ is a PRG

Given:

$$\left\{ G(s) : s \xleftarrow{\$} \{0,1\}^\lambda \right\} \overset{c}{\approx} \left\{ r : r \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

Must show that:

$$\left\{ G'(s) : s \xleftarrow{\$} \{0,1\}^\lambda \right\} \overset{c}{\approx} \left\{ r : r \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

$\underline{G'(s)}$:

return $G(G(s)[1\ldots\lambda])$

$$H_0 : \left\{ G(G(s)[1\ldots\lambda]) : s \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

$$H_1 : \left\{ G(r[1\ldots\lambda]) : r \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

$$H_2 : \left\{ r : r \xleftarrow{\$} \{0,1\}^{\lambda+1} \right\}$$

Prove indistinguishable via a reduction

By the hybrid lemma $H_0 \overset{c}{\approx} H_2$, and so G' is a PRG

Proof Example: Not a PRG

Proof Example: Not a PRG

$G : \{0,1\}^\lambda \rightarrow \{0,1\}^{\lambda+1}$ is a PRG

Proof Example: Not a PRG

$G : \{0,1\}^{\lambda} \to \{0,1\}^{\lambda+1}$ is a PRG

$\underline{G'(s)}$:

return $G(s) || s$

Proof Example: Not a PRG

$G : \{0,1\}^\lambda \to \{0,1\}^{\lambda+1}$ is a PRG

$\underline{G'(s)}$:
return $G(s) || s$

$G' : \{0,1\}^\lambda \to \{0,1\}^{2\lambda+1}$

Proof Example: Not a PRG

$G : \{0,1\}^{\lambda} \rightarrow \{0,1\}^{\lambda+1}$ is a PRG

$\underline{A_{G'}(r)}$:

$\underline{G'(s)}$:

return $G(s) \,||\, s$

$G' : \{0,1\}^{\lambda} \rightarrow \{0,1\}^{2\lambda+1}$

Proof Example: Not a PRG

$G : \{0,1\}^\lambda \to \{0,1\}^{\lambda+1}$ is a PRG

$\underline{G'(s)}$:

return $G(s) || s$

$G' : \{0,1\}^\lambda \to \{0,1\}^{2\lambda+1}$

$\underline{A_{G'}(r)}$:

$x = r[1 \ldots \lambda + 1]$

$y = r[\lambda + 2 \ldots 2\lambda + 1]$

Proof Example: Not a PRG

$G : \{0,1\}^\lambda \to \{0,1\}^{\lambda+1}$ is a PRG

$\underline{A_{G'}(r)}$:

$x = r[1\ldots\lambda+1]$

$y = r[\lambda+2\ldots 2\lambda+1]$

if $G(y) = x$: return 0

else return 1

$\underline{G'(s)}$:

return $G(s)||s$

$G' : \{0,1\}^\lambda \to \{0,1\}^{2\lambda+1}$

Proof Example: Not a PRG

$G : \{0,1\}^\lambda \rightarrow \{0,1\}^{\lambda+1}$ is a PRG

$\underline{G'(s)}$:

return $G(s) \,||\, s$

$G' : \{0,1\}^\lambda \rightarrow \{0,1\}^{2\lambda+1}$

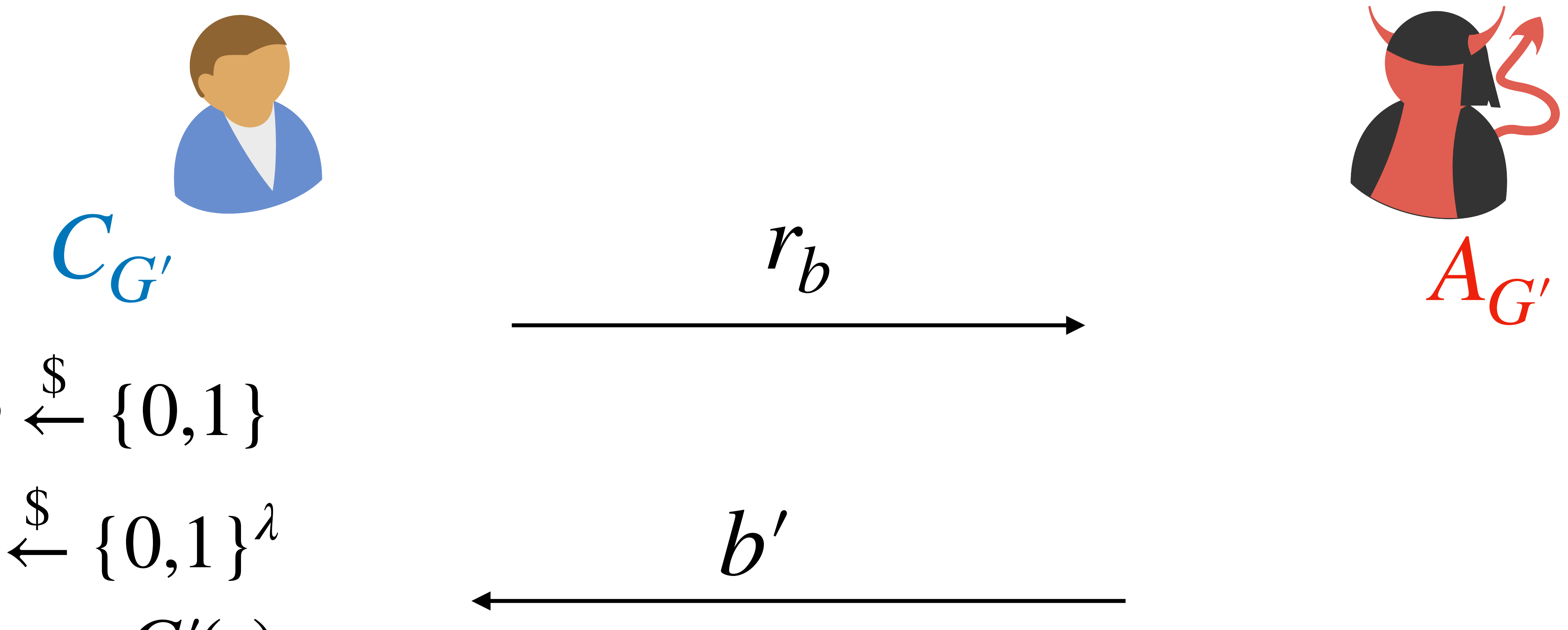
$\underline{A_{G'}(r)}$:

$x = r[1\ldots\lambda+1]$

$y = r[\lambda+2\ldots 2\lambda+1]$

if $G(y) = x$: return 0

else return 1



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

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

$r_0 := G'(s)$

$r_1 \xleftarrow{\$} \{0,1\}^{2\lambda+1}$

Proof Example: Not a PRG

$G : \{0,1\}^{\lambda} \rightarrow \{0,1\}^{\lambda+1}$ is a PRG

$\underline{A_{G'}(r)}$:

$x = r[1\ldots\lambda + 1]$

$y = r[\lambda + 2\ldots2\lambda + 1]$

if $G(y) = x$: return 0

else return 1

$\underline{G'(s)}$:

return $G(s)||s$

$G' : \{0,1\}^{\lambda} \rightarrow \{0,1\}^{2\lambda+1}$

$C_{G'}$

$A_{G'}$

r_b →

← b'

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

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

$r_0 := G'(s)$

$r_1 \xleftarrow{\$} \{0,1\}^{2\lambda+1}$

$Pr[b = b'] =$

Proof Example: Not a PRG

$G : \{0,1\}^\lambda \rightarrow \{0,1\}^{\lambda+1}$ is a PRG

$\underline{A_{G'}(r)}$:

$x = r[1\ldots\lambda + 1]$

$y = r[\lambda + 2\ldots 2\lambda + 1]$

if $G(y) = x$: return 0

else return 1

$\underline{G'(s)}$:

return $G(s)||s$

$G' : \{0,1\}^\lambda \rightarrow \{0,1\}^{2\lambda+1}$

$C_{G'}$

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

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

$r_0 := G'(s)$

$r_1 \xleftarrow{\$} \{0,1\}^{2\lambda+1}$

r_b →

← b'

$A_{G'}$

$$Pr[b = b'] = \frac{1}{2}Pr[b = b'|b = 0] + \frac{1}{2}Pr[b = b'|b = 1]$$

Proof Example: Not a PRG

$G : \{0,1\}^{\lambda} \rightarrow \{0,1\}^{\lambda+1}$ is a PRG

$\underline{A_{G'}(r)}$:

$x = r[1 \ldots \lambda + 1]$

$y = r[\lambda + 2 \ldots 2\lambda + 1]$

if $G(y) = x$: return 0

else return 1

$\underline{G'(s)}$:

return $G(s) || s$

$G' : \{0,1\}^{\lambda} \rightarrow \{0,1\}^{2\lambda+1}$

$$Pr[b = b'] = \frac{1}{2}Pr[b = b' | b = 0] + \frac{1}{2}Pr[b = b' | b = 1]$$

$C_{G'}$

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

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

$r_0 := G'(s)$

$r_1 \xleftarrow{\$} \{0,1\}^{2\lambda+1}$

r_b →

← b'

$A_{G'}$

$$Pr[b = b' | b = 0]$$

Proof Example: Not a PRG

$G : \{0,1\}^\lambda \rightarrow \{0,1\}^{\lambda+1}$ is a PRG

$\underline{A_{G'}(r)}$:

$x = r[1\ldots\lambda + 1]$

$y = r[\lambda + 2\ldots 2\lambda + 1]$

if $G(y) = x$: return 0

else return 1

$\underline{G'(s)}$:

return $G(s)||s$

$G' : \{0,1\}^\lambda \rightarrow \{0,1\}^{2\lambda+1}$

$C_{G'}$

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

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

$r_0 := G'(s)$

$r_1 \xleftarrow{\$} \{0,1\}^{2\lambda+1}$

r_b →

← b'

$A_{G'}$

$$Pr[b = b'] = \frac{1}{2}Pr[b = b' | b = 0] + \frac{1}{2}Pr[b = b' | b = 1]$$

$$Pr[b = b' | b = 0]$$

$$Pr[0 = b' | b = 0]$$

Proof Example: Not a PRG

$G : \{0,1\}^{\lambda} \rightarrow \{0,1\}^{\lambda+1}$ is a PRG

$\underline{A_{G'}(r)}$:

$x = r[1\ldots\lambda + 1]$

$y = r[\lambda + 2\ldots 2\lambda + 1]$

if $G(y) = x$: return 0

else return 1

$\underline{G'(s)}$:

return $G(s)||s$

$G' : \{0,1\}^{\lambda} \rightarrow \{0,1\}^{2\lambda+1}$

$C_{G'}$

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

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

$r_0 := G'(s)$

$r_1 \xleftarrow{\$} \{0,1\}^{2\lambda+1}$

r_b →

← b'

$A_{G'}$

$$Pr[b = b'] = \frac{1}{2}Pr[b = b' | b = 0] + \frac{1}{2}Pr[b = b' | b = 1]$$

$$Pr[b = b' | b = 0]$$

$$Pr[0 = b' | b = 0]$$

$$Pr[G(s) = G(s)]$$

Proof Example: Not a PRG

$G:\{0,1\}^{\lambda}\rightarrow\{0,1\}^{\lambda+1}$ is a PRG

$\underline{A_{G'}(r)}:$

$x = r[1\ldots\lambda+1]$

$y = r[\lambda+2\ldots2\lambda+1]$

if $G(y) = x$: return 0

else return 1

$\underline{G'(s)}:$

return $G(s)||s$

$G':\{0,1\}^{\lambda}\rightarrow\{0,1\}^{2\lambda+1}$

$C_{G'}$

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

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

$r_0 := G'(s)$

$r_1 \xleftarrow{\$} \{0,1\}^{2\lambda+1}$

r_b →

← b'

$A_{G'}$

$$Pr[b=b'] = \frac{1}{2}Pr[b=b'|b=0] + \frac{1}{2}Pr[b=b'|b=1]$$

$Pr[b=b'|b=0]$

$Pr[0=b'|b=0]$

$Pr[G(s)=G(s)]$

$=1$

Proof Example: Not a PRG

$G : \{0,1\}^\lambda \to \{0,1\}^{\lambda+1}$ is a PRG

$\underline{A_{G'}(r)}$:

$x = r[1 \ldots \lambda + 1]$

$y = r[\lambda + 2 \ldots 2\lambda + 1]$

if $G(y) = x$: return 0

else return 1

$\underline{G'(s)}$:
return $G(s) \| s$

$G' : \{0,1\}^\lambda \to \{0,1\}^{2\lambda+1}$

$C_{G'}$

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

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

$r_0 := G'(s)$

$r_1 \xleftarrow{\$} \{0,1\}^{2\lambda+1}$

r_b →

← b'

$A_{G'}$

$$Pr[b = b'] = \frac{1}{2}Pr[b = b' \mid b = 0] + \frac{1}{2}Pr[b = b' \mid b = 1]$$

$Pr[b = b' \mid b = 0]$

$Pr[0 = b' \mid b = 0]$

$Pr[G(s) = G(s)]$

$= 1$

$Pr[b = b' \mid b = 1]$

Proof Example: Not a PRG

$G : \{0,1\}^{\lambda} \rightarrow \{0,1\}^{\lambda+1}$ is a PRG

$\underline{A_{G'}(r)}$:

$x = r[1\ldots\lambda + 1]$

$y = r[\lambda + 2\ldots 2\lambda + 1]$

if $G(y) = x$: return 0

else return 1

$\underline{G'(s)}$:

return $G(s) || s$

$G' : \{0,1\}^{\lambda} \rightarrow \{0,1\}^{2\lambda+1}$

$C_{G'}$

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

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

$r_0 := G'(s)$

$r_1 \xleftarrow{\$} \{0,1\}^{2\lambda+1}$

r_b →

← b'

$A_{G'}$

$$Pr[b = b'] = \frac{1}{2}Pr[b = b' | b = 0] + \frac{1}{2}Pr[b = b' | b = 1]$$

$Pr[b = b' | b = 0]$

$Pr[0 = b' | b = 0]$

$Pr[G(s) = G(s)]$

$= 1$

$Pr[b = b' | b = 1]$

$Pr[1 = b' | b = 0]$

Proof Example: Not a PRG

$G : \{0,1\}^{\lambda} \rightarrow \{0,1\}^{\lambda+1}$ is a PRG

$\underline{G'(s)}$:

return $G(s) || s$

$G' : \{0,1\}^{\lambda} \rightarrow \{0,1\}^{2\lambda+1}$

$\underline{A_{G'}(r)}$:

$x = r[1\ldots\lambda + 1]$

$y = r[\lambda + 2\ldots 2\lambda + 1]$

if $G(y) = x$: return 0

else return 1

$C_{G'}$

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

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

$r_0 := G'(s)$

$r_1 \xleftarrow{\$} \{0,1\}^{2\lambda+1}$

$\xrightarrow{\quad r_b \quad}$

$\xleftarrow{\quad b' \quad}$

$A_{G'}$

$$Pr[b = b'] = \frac{1}{2}Pr[b = b' | b = 0] + \frac{1}{2}Pr[b = b' | b = 1]$$

$Pr[b = b' | b = 0]$

$Pr[0 = b' | b = 0]$

$Pr[G(s) = G(s)]$

$= 1$

$Pr[b = b' | b = 1]$

$Pr[1 = b' | b = 0]$

$Pr[G(y) \neq x | x \xleftarrow{\$} \{0,1\}^{\lambda+1}]$

Proof Example: Not a PRG

$G : \{0,1\}^{\lambda} \rightarrow \{0,1\}^{\lambda+1}$ is a PRG

$\underline{G'(s)}$:

return $G(s) || s$

$G' : \{0,1\}^{\lambda} \rightarrow \{0,1\}^{2\lambda+1}$

$\underline{A_{G'}(r)}$:

$x = r[1 \ldots \lambda + 1]$

$y = r[\lambda + 2 \ldots 2\lambda + 1]$

if $G(y) = x$: return 0

else return 1

$C_{G'}$

$b \overset{\$}{\leftarrow} \{0,1\}$

$s \overset{\$}{\leftarrow} \{0,1\}^{\lambda}$

$r_0 := G'(s)$

$r_1 \overset{\$}{\leftarrow} \{0,1\}^{2\lambda+1}$

r_b →

← b'

$A_{G'}$

$$Pr[b = b'] = \frac{1}{2} Pr[b = b' | b = 0] + \frac{1}{2} Pr[b = b' | b = 1]$$

$Pr[b = b' | b = 0]$

$Pr[0 = b' | b = 0]$

$Pr[G(s) = G(s)]$

$= 1$

$Pr[b = b' | b = 1]$

$Pr[1 = b' | b = 0]$

$Pr[G(y) \neq x | x \overset{\$}{\leftarrow} \{0,1\}^{\lambda+1}]$

$= 1 - \frac{1}{2^{\lambda+1}}$

Proof Example: Not a PRG

$G : \{0,1\}^{\lambda} \rightarrow \{0,1\}^{\lambda+1}$ is a PRG

$\underline{A_{G'}(r)}$:

$x = r[1\ldots\lambda + 1]$

$y = r[\lambda + 2\ldots 2\lambda + 1]$

if $G(y) = x$: return 0

else return 1

$\underline{G'(s)}$:

return $G(s)\,||\,s$

$G' : \{0,1\}^{\lambda} \rightarrow \{0,1\}^{2\lambda+1}$

$$Pr[b = b'] = \frac{1}{2} + \frac{1}{2} - \frac{1}{2^{\lambda+2}} = 1 - \frac{1}{2^{\lambda+2}}$$

$C_{G'}$

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

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

$r_0 := G'(s)$

$r_1 \xleftarrow{\$} \{0,1\}^{2\lambda+1}$

r_b →

← b'

$A_{G'}$

$$Pr[b = b' \,|\, b = 0]$$
$$Pr[0 = b' \,|\, b = 0]$$
$$Pr[G(s) = G(s)]$$
$$= 1$$

$$Pr[b = b' \,|\, b = 1]$$
$$Pr[1 = b' \,|\, b = 0]$$
$$Pr[G(y) \neq x \,|\, x \xleftarrow{\$} \{0,1\}^{\lambda+1}]$$
$$= 1 - \frac{1}{2^{\lambda+1}}$$

Proof Example: Not a PRG

$G : \{0,1\}^{\lambda} \rightarrow \{0,1\}^{\lambda+1}$ is a PRG

$\underline{A_{G'}(r)}$:

$x = r[1\ldots\lambda+1]$

$y = r[\lambda+2\ldots 2\lambda+1]$

if $G(y) = x$: return 0

else return 1

$\underline{G'(s)}$:

return $G(s)\,||\,s$

$G' : \{0,1\}^{\lambda} \rightarrow \{0,1\}^{2\lambda+1}$

$C_{G'}$

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

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

$r_0 := G'(s)$

$r_1 \xleftarrow{\$} \{0,1\}^{2\lambda+1}$

r_b →

← b'

$A_{G'}$

$$Pr[b = b'] = \frac{1}{2} + \frac{1}{2} - \frac{1}{2^{\lambda+2}} = 1 - \frac{1}{2^{\lambda+2}}$$

$Pr[b = b'$

$Pr[0 = b'$

$Pr[G(s) =$

$= 1$

NOT negligibly close to $\frac{1}{2}$, and so G' is *not* a PRG

$= 1 - \frac{\quad}{2^{\lambda+1}}$

Proof Techniques

Proof Techniques

- Proving that a construction satisfies a definition

Proof Techniques

- Proving that a construction satisfies a definition
- Proving that a construction *does not* satisfy a definition

Proof Techniques

- Proving that a construction satisfies a definition
- Proving that a construction *does not* satisfy a definition
  - This may be harder than the example we just gave! It might require "specific" schemes.

Proof Techniques

- Proving that a construction satisfies a definition
- Proving that a construction *does not* satisfy a definition
  - This may be harder than the example we just gave! It might require "specific" schemes.
  - For example: to disprove the following statement "$\forall G$, if G is a PRG, then G' is a PRG" (where G' uses G in its construction), you need to prove: "$\exists G$ that is a PRG, such that G' is *not* a PRG."

Proof Techniques

- Proving that a construction satisfies a definition
- Proving that a construction *does not* satisfy a definition
  - This may be harder than the example we just gave! It might require "specific" schemes.
  - For example: to disprove the following statement "$\forall G$, if G is a PRG, then G' is a PRG" (where G' uses G in its construction), you need to prove: "$\exists G$ that is a PRG, such that G' is *not* a PRG."
  - This gives you a lot of freedom! You can choose G to be *whatever you want*, as long as it is a PRG!

Proof Techniques

Proof Techniques

- Proving that a construction satisfies a definition

Proof Techniques

- Proving that a construction satisfies a definition
- Proving that a construction *does not* satisfy a definition

Proof Techniques

- Proving that a construction satisfies a definition
- Proving that a construction *does not* satisfy a definition
- Proving that a definition implies another definition

Proof Techniques

- Proving that a construction satisfies a definition
- Proving that a construction *does not* satisfy a definition
- Proving that a definition implies another definition
    - Done via a reduction (see uniform ciphertext security -> perfect security)

Proof Techniques

- Proving that a construction satisfies a definition
- Proving that a construction *does not* satisfy a definition
- Proving that a definition implies another definition
    - Done via a reduction (see uniform ciphertext security -> perfect security)
- Proving that a definition *does not* imply another definition

Proof Techniques

- Proving that a construction satisfies a definition
- Proving that a construction *does not* satisfy a definition
- Proving that a definition implies another definition
    - Done via a reduction (see uniform ciphertext security -> perfect security)
- Proving that a definition *does not* imply another definition
    - Done via a “pathological” construction. Define a construction that satisfies one definition, while trivially not satisfying another, then *define an adversary* that attacks the other construction.

Proof Techniques

- Proving that a construction satisfies a definition
- Proving that a construction *does not* satisfy a definition
- Proving that a definition implies another definition
  - Done via a reduction (see uniform ciphertext security -> perfect security)
- Proving that a definition *does not* imply another definition
  - Done via a “pathological” construction. Define a construction that satisfies one definition, while trivially not satisfying another, then *define an adversary* that attacks the other construction.
  - See perfect security not implying uniform ciphertext security, or key privacy questions on homework.

Proof Techniques

When approaching a proof, first ask: "Which type is it going to be?"

- Proving that a construction satisfies a definition
- Proving that a construction *does not* satisfy a definition
- Proving that a definition implies another definition
  - Done via a reduction (see uniform ciphertext security -> perfect security)
- Proving that a definition *does not* imply another definition
  - Done via a "pathological" construction. Define a construction that satisfies one definition, while trivially not satisfying another, then *define an adversary* that attacks the other construction.
  - See perfect security not implying uniform ciphertext security, or key privacy questions on homework.

Pseudorandomness II

601.442/642 Modern Cryptography

5th February 2026

Multi-Message Security

One-Time Computational Security

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

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

Multi-Message Security

One-Time Computational Security

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

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

Multi-Message Security

One-Time Computational Security

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

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

Multi-Message Security

One-Time Computational Security

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

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

$$b \overset{\$}{\leftarrow} \{0,1\}$$
$$k \leftarrow \text{KeyGen}(1^\lambda)$$
$$c \leftarrow \text{Enc}(k, m_b)$$

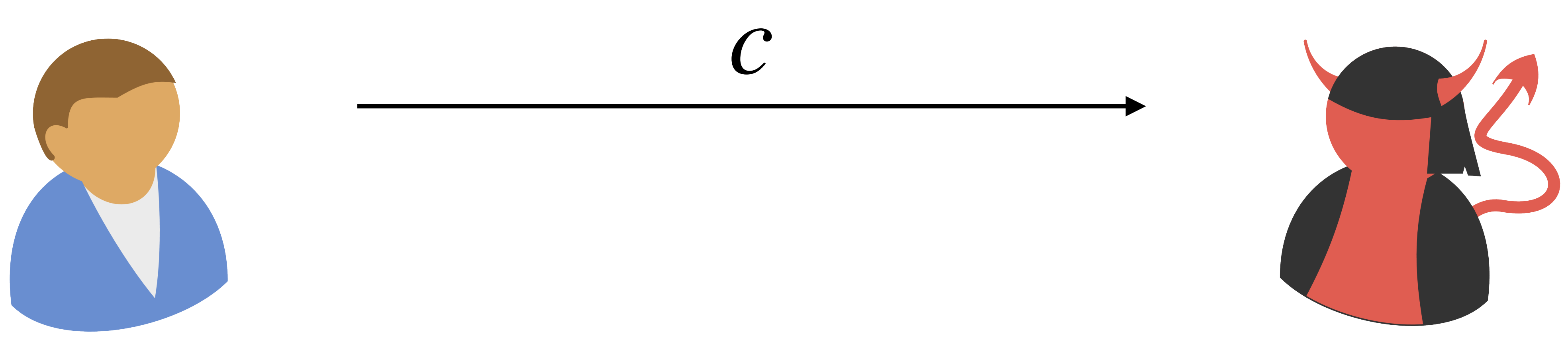
Multi-Message Security

One-Time Computational Security

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

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

$$b \overset{\$}{\leftarrow} \{0,1\}$$
$$k \leftarrow \mathsf{KeyGen}(1^\lambda)$$
$$c \leftarrow \mathsf{Enc}(k, m_b)$$



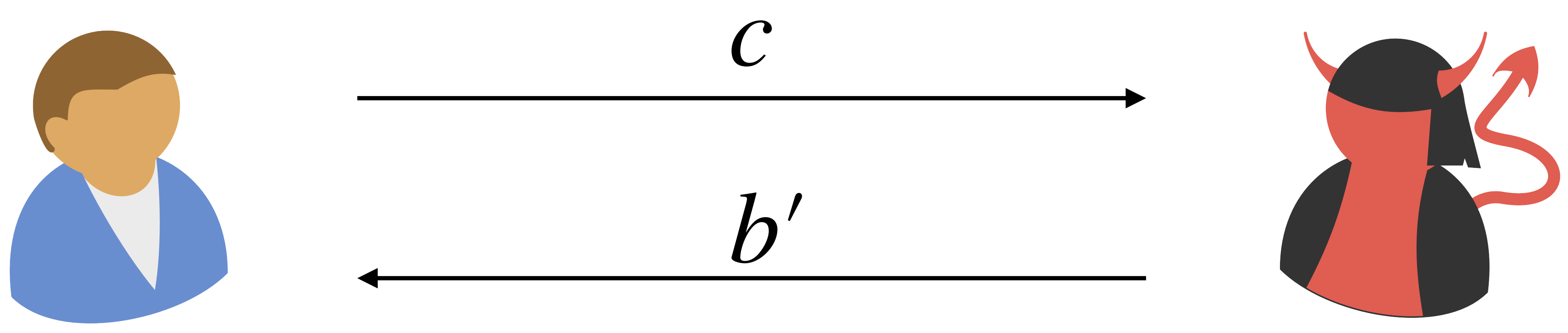
Multi-Message Security

One-Time Computational Security

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

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

$$b \overset{\$}{\leftarrow} \{0,1\}$$
$$k \leftarrow \mathsf{KeyGen}(1^{\lambda})$$
$$c \leftarrow \mathsf{Enc}(k, m_b)$$



Multi-Message Security

One-Time Computational Security

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

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

$$b \xleftarrow{\$} \{0,1\}$$
$$k \leftarrow \mathsf{KeyGen}(1^\lambda)$$
$$c \leftarrow \mathsf{Enc}(k, m_b)$$

c →

← b'

Wins if $b' = b$

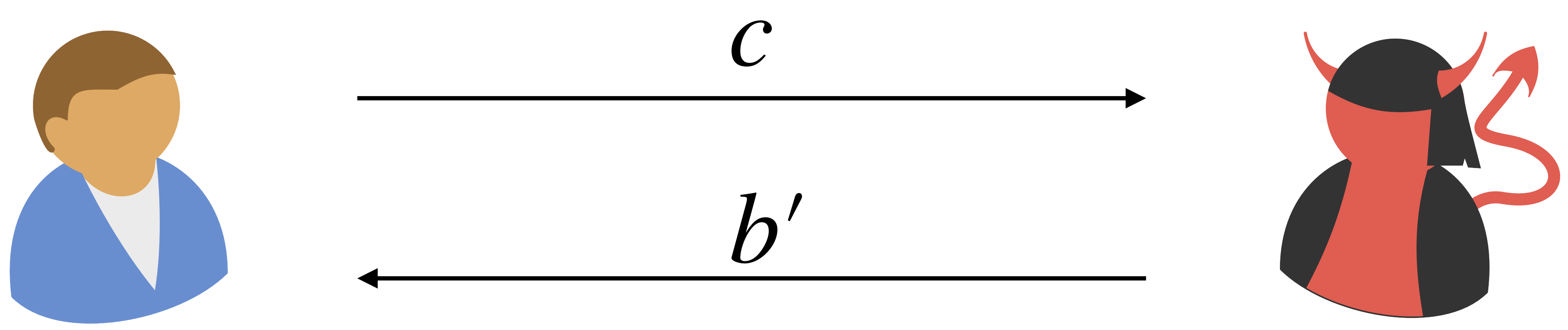
Multi-Message Security

One-Time Computational Security

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

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

$$b \xleftarrow{\$} \{0,1\}$$
$$k \leftarrow \mathsf{KeyGen}(1^\lambda)$$
$$c \leftarrow \mathsf{Enc}(k, m_b)$$



Wins if $b' = b$

$$\forall \mathscr{A}, \forall m_0, m_1,$$
$$\Pr[b' = b] \leq \frac{1}{2} + \nu(\lambda)$$

Multi-Message Security

Multi-Message Security

An encryption scheme with message length $\ell := \ell(\lambda)$ is multi-message secure if $\forall \{(m_0^i, m_1^i)\}_{i=1}^{q(\lambda)}$ where $q(\lambda)$ is a polynomial

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

Multi-Message Security

$q(\lambda)$ ***pairs*** of messages

Multi-Message Security

An encryption scheme with message length $\ell := \ell(\lambda)$ is multi-message secure if $\forall \{(m_0^i, m_1^i)\}_{i=1}^{q(\lambda)}$ where $q(\lambda)$ is a polynomial

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

Multi-Message Security

$q(\lambda)$ *pairs* of messages

Multi-Message Security

An encryption scheme with message length $\ell := \ell(\lambda)$ is multi-message secure if $\forall \{(m_0^i, m_1^i)\}_{i=1}^{q(\lambda)}$ where $q(\lambda)$ is a polynomial

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

Multi-Message Security

$q(\lambda)$ *pairs* of messages

Multi-Message Security

An encryption scheme with message length $\ell := \ell(\lambda)$ is multi-message secure if $\forall \{(m_0^i, m_1^i)\}_{i=1}^{q(\lambda)}$ where $q(\lambda)$ is a polynomial

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

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

$$k \leftarrow \text{KeyGen}(1^\lambda)$$

for $i = 1 \ldots q(\lambda)$:

Multi-Message Security

$q(\lambda)$ *pairs* of messages

Multi-Message Security

An encryption scheme with message length $\ell := \ell(\lambda)$ is multi-message secure if $\forall \{(m_0^i, m_1^i)\}_{i=1}^{q(\lambda)}$ where $q(\lambda)$ is a polynomial

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

$$b \overset{\$}{\leftarrow} \{0,1\}$$

$$k \leftarrow \text{KeyGen}(1^\lambda)$$

for $i = 1 \ldots q(\lambda)$:

$$c_i = \text{Enc}(k, m_b^i)$$

Multi-Message Security

Multi-Message Security

An encryption scheme with message length $\ell := \ell(\lambda)$ is multi-message secure if $\forall \{(m_0^i, m_1^i)\}_{i=1}^{q(\lambda)}$ where $q(\lambda)$ is a polynomial

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

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

$$k \leftarrow \text{KeyGen}(1^\lambda)$$

for $i = 1 \ldots q(\lambda)$:

$$c_i = \text{Enc}(k, m_b^i)$$

$\{c_i\}_{i=1}^{q(\lambda)}$ →

Multi-Message Security

$q(\lambda)$ *pairs* of messages

Multi-Message Security

An encryption scheme with message length $\ell := \ell(\lambda)$ is multi-message secure if $\forall \{(m_0^i, m_1^i)\}_{i=1}^{q(\lambda)}$ where $q(\lambda)$ is a polynomial

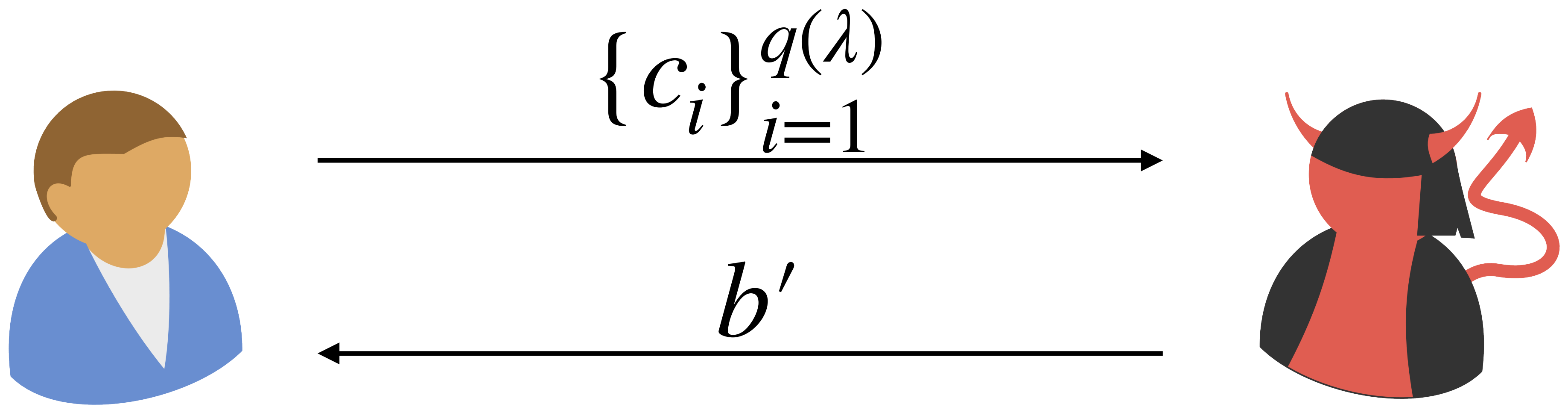
$$D_0 = \left\{ \mathsf{ct} : \begin{array}{c} k \leftarrow \mathsf{KeyGen}(1^\lambda) \\ c \leftarrow \{\mathsf{Enc}(k, m_0^i)\}_{i=1}^{q(\lambda)} \end{array} \right\} \overset{c}{\approx} D_1 = \left\{ \mathsf{ct} : \begin{array}{c} k \leftarrow \mathsf{KeyGen}(1^\lambda) \\ c \leftarrow \{\mathsf{Enc}(k,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

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

$$k \leftarrow \mathsf{KeyGen}(1^\lambda)$$

$$\text{for } i = 1 \ldots q(\lambda):$$

$$c_i = \mathsf{Enc}(k, m_b^i)$$



Multi-Message Security

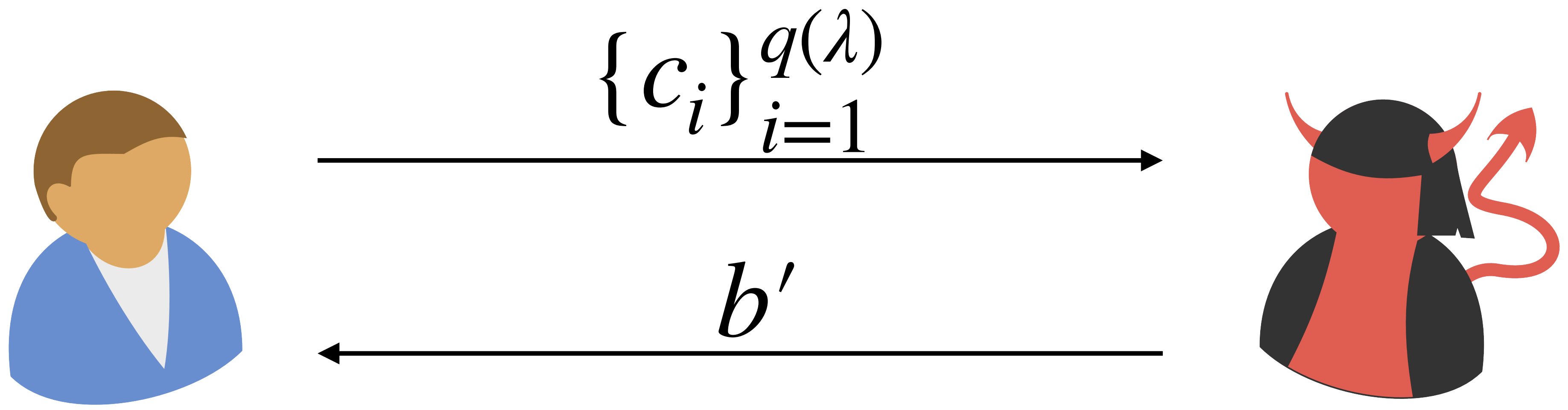
$q(\lambda)$ pairs of messages

Multi-Message Security

An encryption scheme with message length $\ell := \ell(\lambda)$ is multi-message secure if $\forall \{(m_0^i, m_1^i)\}_{i=1}^{q(\lambda)}$ where $q(\lambda)$ is a polynomial

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

$$b \overset{\$}{\leftarrow} \{0,1\}$$
$$k \leftarrow \text{KeyGen}(1^\lambda)$$
$$\text{for } i = 1 \ldots q(\lambda):$$
$$c_i = \text{Enc}(k, m_b^i)$$



Wins if $b' = b$

Multi-Message Security

$q(\lambda)$ *pairs* of messages

Multi-Message Security

An encryption scheme with message length $\ell := \ell(\lambda)$ is multi-message secure if $\forall \{(m_0^i, m_1^i)\}_{i=1}^{q(\lambda)}$ where $q(\lambda)$ is a polynomial

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

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

$$k \leftarrow \mathsf{KeyGen}(1^\lambda)$$

for $i = 1 \ldots q(\lambda)$:

$$c_i = \mathsf{Enc}(k, m_b^i)$$

$\{c_i\}_{i=1}^{q(\lambda)}$ →

← b'

$$\Pr[b' = b] \leq \frac{1}{2} + \nu(\lambda)$$

Wins if $b' = b$

Multi-Message Security

Multi-Message Security

An encryption scheme with message length $\ell := \ell(\lambda)$ is multi-message secure if $\forall \{(m_0^i, m_1^i)\}_{i=1}^{q(\lambda)}$ where $q(\lambda)$ is a polynomial

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

Does Pseudorandom OTP satisfy this definition?

Pseudorandom OTP

$$\mathsf{KeyGen}(1^\lambda) : k \overset{\$}{\leftarrow} \{0,1\}$$

$$\mathsf{Enc}(k, m) : G(k) \oplus m$$

Multi-Message Security

Multi-Message Security

An encryption scheme with message length $\ell := \ell(\lambda)$ is multi-message secure if $\forall \{(m_0^i, m_1^i)\}_{i=1}^{q(\lambda)}$ where $q(\lambda)$ is a polynomial

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

Does Pseudorandom OTP satisfy this definition?

No! Same problem as regular OTP

Pseudorandom OTP

$\text{KeyGen}(1^\lambda) : k \xleftarrow{\$} \{0,1\}$

$\text{Enc}(k, m) : G(k) \oplus m$

Multi-Message Security

Multi-Message Security

An encryption scheme with message length $\ell := \ell(\lambda)$ is multi-message secure if $\forall \{(m_0^i, m_1^i)\}_{i=1}^{q(\lambda)}$ where $q(\lambda)$ is a polynomial

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

Does Pseudorandom OTP satisfy this definition?

No! Same problem as regular OTP

Idea: Can we design a multi-message secure encryption scheme that is **stateful**?

Pseudorandom OTP

$\text{KeyGen}(1^\lambda) : k \xleftarrow{\$} \{0,1\}$

$\text{Enc}(k, m) : G(k) \oplus m$

Stateful Multi-Message Secure Encryption

Stateful Multi-Message Secure Encryption

We just saw a PRG that can output a *polynomial* number of pseudorandom bits.

Stateful Multi-Message Secure Encryption

We just saw a PRG that can output a *polynomial* number of pseudorandom bits.

What if we use one chunk at a time?

Stateful Multi-Message Secure Encryption

We just saw a PRG that can output a *polynomial* number of pseudorandom bits.

What if we use one chunk at a time?

$$G(s) = 00010101...11100010...01011010...$$

Stateful Multi-Message Secure Encryption

We just saw a PRG that can output a *polynomial* number of pseudorandom bits.

What if we use one chunk at a time?

$$G(s) = 00010101...11100010...01011010... \rightarrow \text{poly}(\lambda)$$

Stateful Multi-Message Secure Encryption

We just saw a PRG that can output a *polynomial* number of pseudorandom bits.

What if we use one chunk at a time?

$$G(s) = 00010101...11100010...01011010... \rightarrow \text{poly}(\lambda)$$

Stateful Multi-Message Secure Encryption

We just saw a PRG that can output a *polynomial* number of pseudorandom bits.

What if we use one chunk at a time?

$$G(s) = 00010101...11100010...01011010... \rightarrow \text{poly}(\lambda)$$

Stateful Multi-Message Secure Encryption

We just saw a PRG that can output a *polynomial* number of pseudorandom bits.

What if we use one chunk at a time?

$$G(s) = 00010101 \ldots 11100010 \ldots 01011010 \ldots \rightarrow \text{poly}(\lambda)$$

Stateful Multi-Message Secure Encryption

We just saw a PRG that can output a *polynomial* number of pseudorandom bits.

What if we use one chunk at a time?

$G(s)[0]$

$$G(s) = 00010101\ldots 11100010\ldots 01011010\ldots \rightarrow \text{poly}(\lambda)$$

Stateful Multi-Message Secure Encryption

We just saw a PRG that can output a *polynomial* number of pseudorandom bits.

What if we use one chunk at a time?

$G(s)[0]$ $G(s)[1]$

$$G(s) = 00010101...11100010...01011010... \rightarrow \text{poly}(\lambda)$$

Stateful Multi-Message Secure Encryption

We just saw a PRG that can output a *polynomial* number of pseudorandom bits.

What if we use one chunk at a time?

$G(s)[0]$ $G(s)[1]$ $G(s)[2]$

$$G(s) = 00010101 \ldots 11100010 \ldots 01011010 \ldots \rightarrow \text{poly}(\lambda)$$

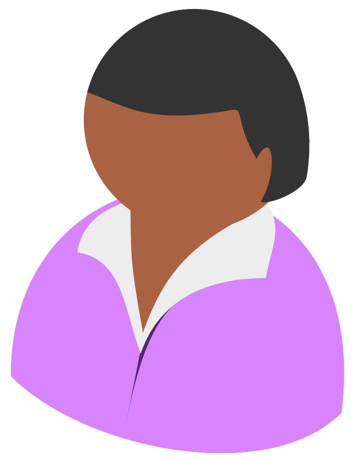
Stateful Multi-Message Secure Encryption

We just saw a PRG that can output a *polynomial* number of pseudorandom bits.

What if we use one chunk at a time?

$G(s)[0]$ $\quad$ $G(s)[1]$ $\quad$ $G(s)[2]$

$$G(s) = 00010101\ldots11100010\ldots01011010\ldots \rightarrow \text{poly}(\lambda)$$



Stateful Multi-Message Secure Encryption

We just saw a PRG that can output a *polynomial* number of pseudorandom bits.

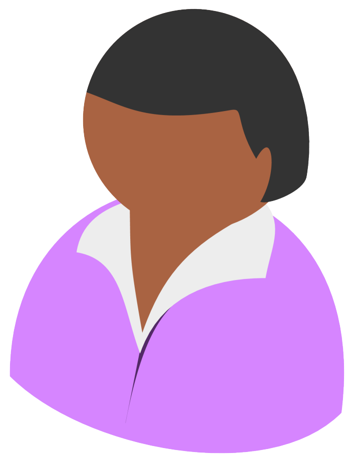
What if we use one chunk at a time?

$G(s)[0]$ $G(s)[1]$ $G(s)[2]$

$$G(s) = 00010101 \ldots 11100010 \ldots 01011010 \ldots \rightarrow \text{poly}(\lambda)$$

k

k



Stateful Multi-Message Secure Encryption

We just saw a PRG that can output a *polynomial* number of pseudorandom bits.

What if we use one chunk at a time?

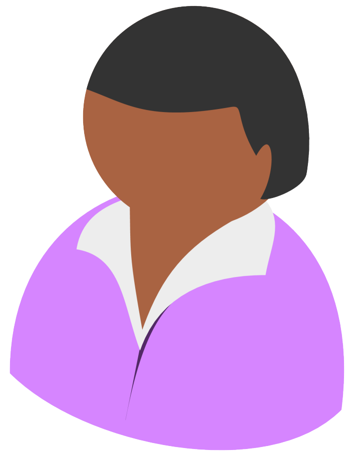
$G(s)[0]$ $G(s)[1]$ $G(s)[2]$

$$G(s) = 00010101\ldots11100010\ldots01011010\ldots \rightarrow \text{poly}(\lambda)$$

k

$i = 0$

k



Stateful Multi-Message Secure Encryption

We just saw a PRG that can output a *polynomial* number of pseudorandom bits.

What if we use one chunk at a time?

$G(s)[0]$ $G(s)[1]$ $G(s)[2]$

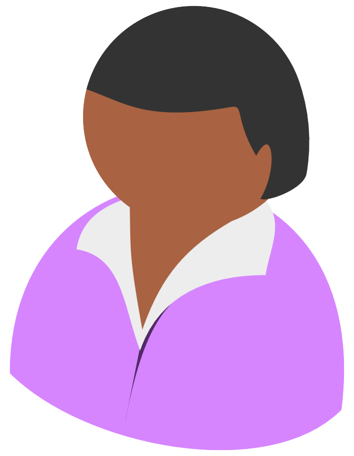
$$G(s) = 00010101 \ldots 11100010 \ldots 01011010 \ldots \to \text{poly}(\lambda)$$

k

$i = 0$

$\xrightarrow{c_0 = m_0 \oplus G(k)[0]}$

k



Stateful Multi-Message Secure Encryption

We just saw a PRG that can output a *polynomial* number of pseudorandom bits.

What if we use one chunk at a time?

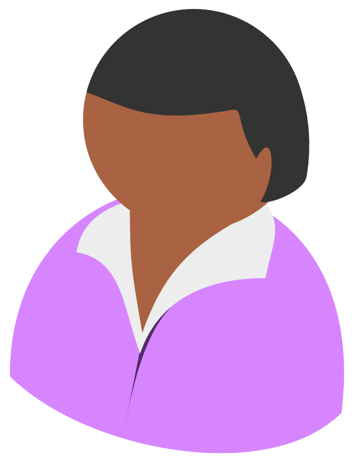
$G(s)[0]$ $G(s)[1]$ $G(s)[2]$

$$G(s) = 00010101\ldots11100010\ldots01011010\ldots \rightarrow \text{poly}(\lambda)$$

k

$i = 0$

$c_0 = m_0 \oplus G(k)[0]$ $\longrightarrow$



k $m_0 = c_0 \oplus G(k)[0]$

Stateful Multi-Message Secure Encryption

We just saw a PRG that can output a *polynomial* number of pseudorandom bits.

What if we use one chunk at a time?

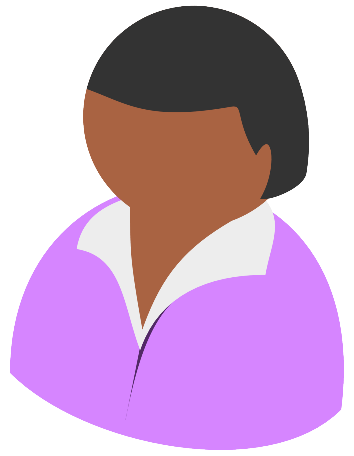
$G(s)[0]$ $\quad$ $G(s)[1]$ $\quad$ $G(s)[2]$

$$G(s) = 00010101 \ldots 11100010 \ldots 01011010 \ldots \rightarrow \text{poly}(\lambda)$$

k

$i = 1$

$c_0 = m_0 \oplus G(k)[0]$ $\longrightarrow$



k $\quad$ $m_0 = c_0 \oplus G(k)[0]$

Stateful Multi-Message Secure Encryption

We just saw a PRG that can output a *polynomial* number of pseudorandom bits.

What if we use one chunk at a time?

$G(s)[0]$ $G(s)[1]$ $G(s)[2]$

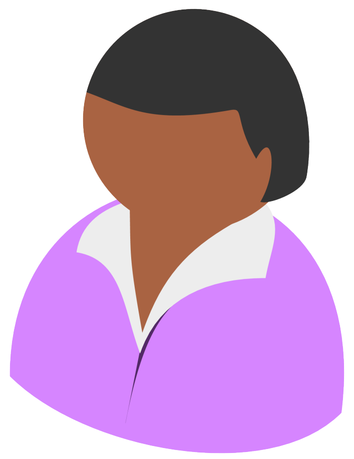
$G(s) = 00010101 \ldots 11100010 \ldots 01011010 \ldots \rightarrow \text{poly}(\lambda)$

k

$i = 1$

$c_0 = m_0 \oplus G(k)[0]$

$c_1 = m_1 \oplus G(k)[1]$



k $m_0 = c_0 \oplus G(k)[0]$

Stateful Multi-Message Secure Encryption

We just saw a PRG that can output a *polynomial* number of pseudorandom bits.

What if we use one chunk at a time?

$G(s)[0]$ $G(s)[1]$ $G(s)[2]$

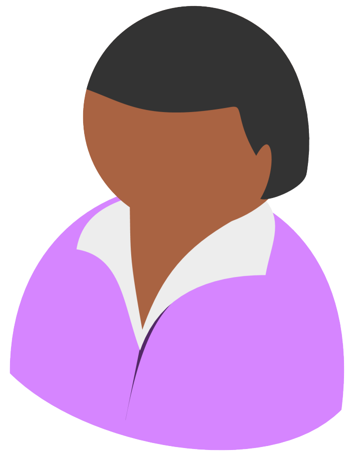
$$G(s) = 00010101 \ldots 11100010 \ldots 01011010 \ldots \to \text{poly}(\lambda)$$

k

$i = 1$

$c_0 = m_0 \oplus G(k)[0]$

$c_1 = m_1 \oplus G(k)[1]$



k

$m_0 = c_0 \oplus G(k)[0]$

$m_1 = c_1 \oplus G(k)[1]$

Stateful Multi-Message Secure Encryption

We just saw a PRG that can output a *polynomial* number of pseudorandom bits.

What if we use one chunk at a time?

What are the downsides of keeping state?

$G(s)[0]$ $G(s)[1]$ $G(s)[2]$

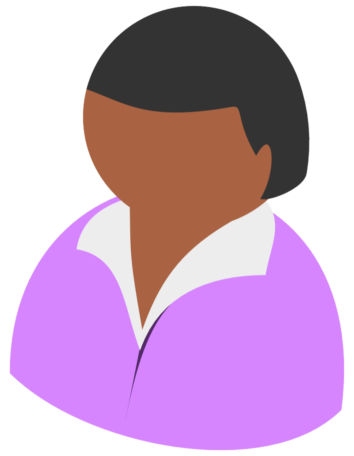
$$G(s) = 00010101\ldots11100010\ldots01011010\ldots \rightarrow \text{poly}(\lambda)$$

k

$i = 1$

$c_0 = m_0 \oplus G(k)[0]$

$c_1 = m_1 \oplus G(k)[1]$



k

$m_0 = c_0 \oplus G(k)[0]$

$m_1 = c_1 \oplus G(k)[1]$

Stateful Multi-Message Secure Encryption

We just saw a PRG that can output a *polynomial* number of pseudorandom bits.

What if we use one chunk at a time?

What are the downsides of keeping state?

$G(s)[0]$ $G(s)[1]$ $G(s)[2]$

$$G(s) = 00010101 \ldots 11100010 \ldots 01011010 \ldots \rightarrow \text{poly}(\lambda)$$

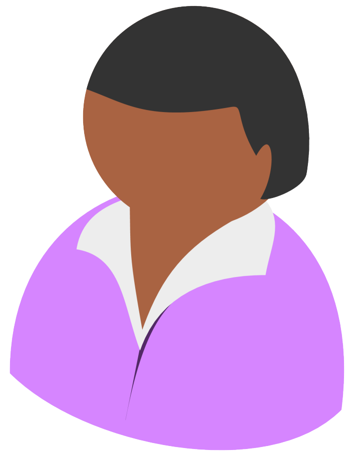
Losing it!

k

$i = 1$

$c_0 = m_0 \oplus G(k)[0]$

$c_1 = m_1 \oplus G(k)[1]$



k

$m_0 = c_0 \oplus G(k)[0]$

$m_1 = c_1 \oplus G(k)[1]$

Stateless Multi-Message Secure Encryption

Stateless Multi-Message Secure Encryption

What if we remove state by randomly sampling the chunk index?

Stateless Multi-Message Secure Encryption

What if we remove state by randomly sampling the chunk index?

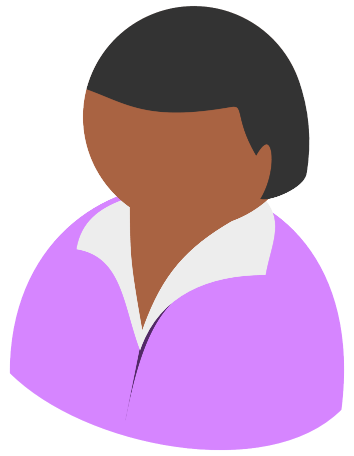
k

$i_0 \xleftarrow{\$} [0\ldots\text{poly}(\lambda)]$

$i_1 \xleftarrow{\$} [0\ldots\text{poly}(\lambda)]$

$c_0 = (m_0 \oplus G(k)[i_0], i_0)$ →

$c_1 = (m_1 \oplus G(k)[i_1], i_1)$ →



k

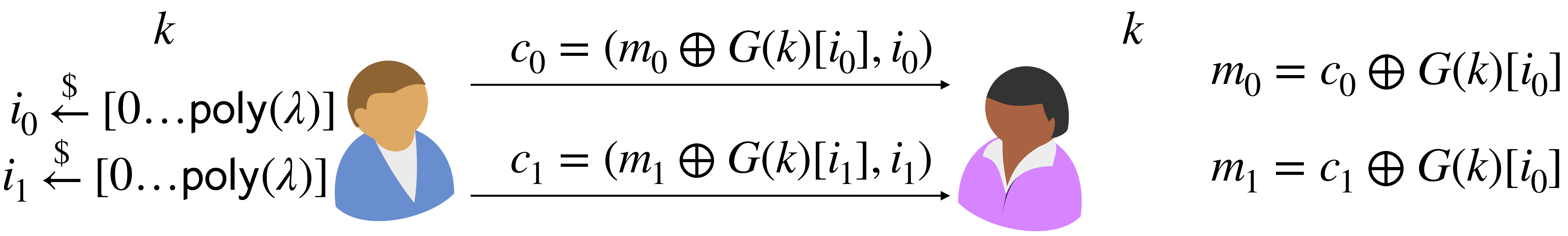
$m_0 = c_0 \oplus G(k)[i_0]$

$m_1 = c_1 \oplus G(k)[i_0]$

Stateless Multi-Message Secure Encryption

What if we remove state by randomly sampling the chunk index?

What happens if $i_0 = i_1$?

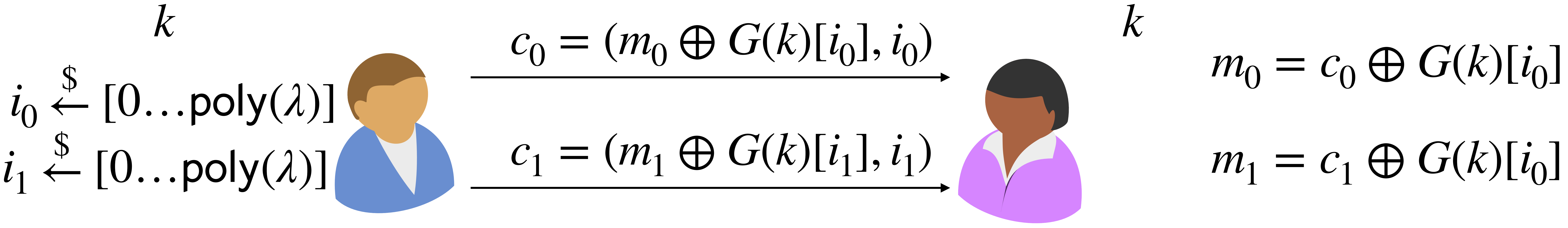


Stateless Multi-Message Secure Encryption

What if we remove state by randomly sampling the chunk index?

What happens if $i_0 = i_1$?

What is the probability that $i_0 = i_1$?



Stateless Multi-Message Secure Encryption

What if we remove state by randomly sampling the chunk index?

What happens if $i_0 = i_1$?

What is the probability that $i_0 = i_1$?

This is totally insecure! There is a non-negligible chance of sampling the same index, and so a non-negligible chance of reusing a chunk!

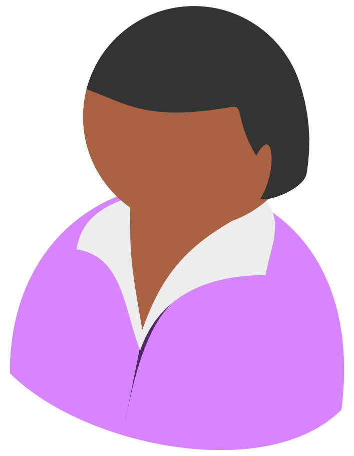
k

$i_0 \xleftarrow{\$} [0\ldots\text{poly}(\lambda)]$

$i_1 \xleftarrow{\$} [0\ldots\text{poly}(\lambda)]$

$c_0 = (m_0 \oplus G(k)[i_0], i_0)$

$c_1 = (m_1 \oplus G(k)[i_1], i_1)$



k

$m_0 = c_0 \oplus G(k)[i_0]$

$m_1 = c_1 \oplus G(k)[i_0]$

Stateless Multi-Message Secure Encryption

What if we remove state by randomly sampling the chunk index?

What happens if $i_0 = i_1$?

What is the probability that $i_0 = i_1$?

This is totally insecure! There is a non-negligible chance of sampling the same index, and so a non-negligible chance of reusing a chunk!

Idea: What if we could index into an *exponential* amount of randomness?

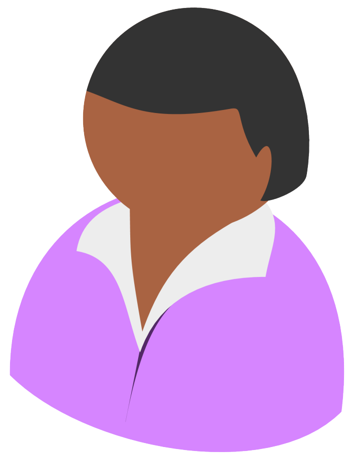
k

$i_0 \xleftarrow{\$} [0\ldots\text{poly}(\lambda)]$

$i_1 \xleftarrow{\$} [0\ldots\text{poly}(\lambda)]$

$c_0 = (m_0 \oplus G(k)[i_0], i_0)$

$c_1 = (m_1 \oplus G(k)[i_1], i_1)$



k

$m_0 = c_0 \oplus G(k)[i_0]$

$m_1 = c_1 \oplus G(k)[i_0]$

Stateless Multi-Message Secure Encryption

What if Alice and Bob shared an *exponential* amount of randomness?

Stateless Multi-Message Secure Encryption

What if Alice and Bob shared an *exponential* amount of randomness?

$F =$

x	r
000…000	11001010
000…001	10011111
000…010	10010010
000…011	10111111

$\dots$

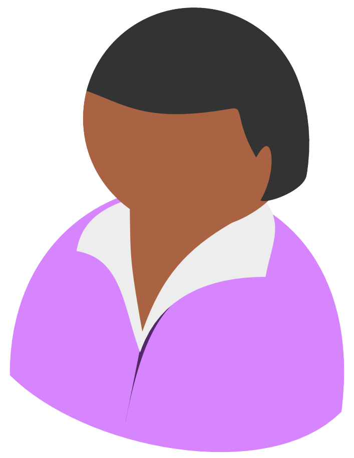
Stateless Multi-Message Secure Encryption

What if Alice and Bob shared an *exponential* amount of randomness?

$F =$

x	r
000…000	11001010
000…001	10011111
000…010	10010010
000…011	10111111

$\cdots$



Stateless Multi-Message Secure Encryption

What if Alice and Bob shared an *exponential* amount of randomness?

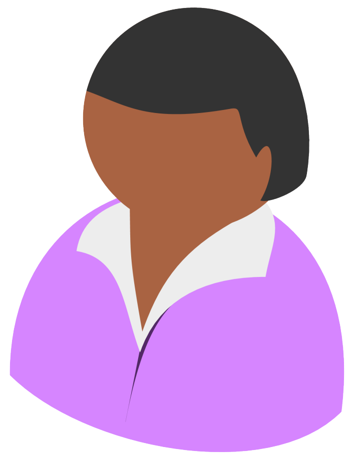
$F =$

x	r
000…000	11001010
000…001	10011111
000…010	10010010
000…011	10111111

$\cdots$

F

F



Stateless Multi-Message Secure Encryption

What if Alice and Bob shared an *exponential* amount of randomness?

$F =$

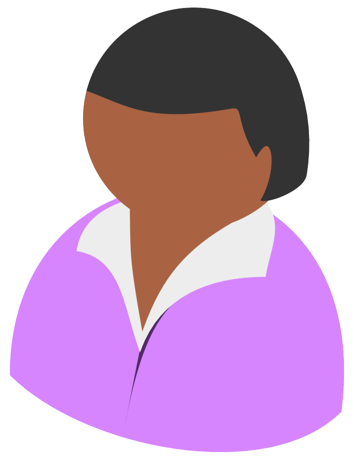
x	r
000...000	11001010
000...001	10011111
000...010	10010010
000...011	10111111

$\cdots$

F

$i_0 \xleftarrow{\$} \{0,1\}^\lambda$

F



Stateless Multi-Message Secure Encryption

What if Alice and Bob shared an *exponential* amount of randomness?

$F =$

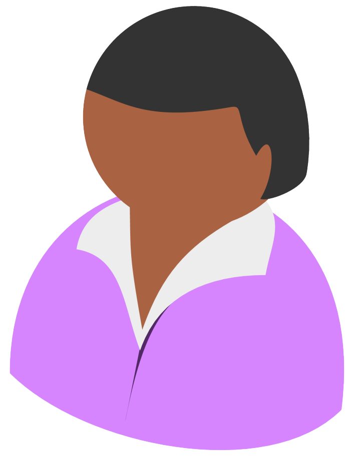
x	r
000…000	11001010
000…001	10011111
000…010	10010010
000…011	10111111

$\cdots$

F

$i_0 \xleftarrow{\$} \{0,1\}^\lambda$

$c_0 = (m_0 \oplus F[i_0], i_0)$ →



F

Stateless Multi-Message Secure Encryption

What if Alice and Bob shared an *exponential* amount of randomness?

$F =$

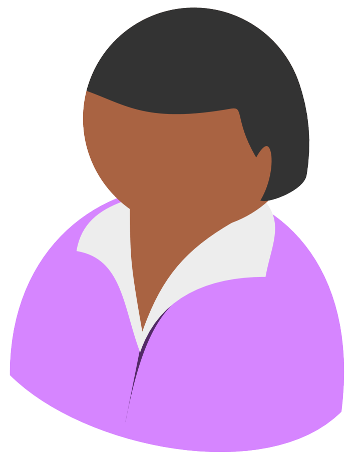
x	r
000…000	11001010
000…001	10011111
000…010	10010010
000…011	10111111

$\cdots$

F

$i_0 \xleftarrow{\$} \{0,1\}^\lambda$

$c_0 = (m_0 \oplus F[i_0], i_0)$ →



F

$m_0 = c_0 \oplus F[i_0]$

Stateless Multi-Message Secure Encryption

What if Alice and Bob shared an *exponential* amount of randomness?

$F =$

x	r
000…000	11001010
000…001	10011111
000…010	10010010
000…011	10111111

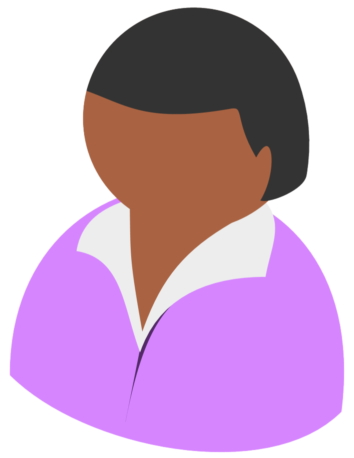
$\cdots$

The probability of sampling the same index is *negligible*, so this is secure!

F

$i_0 \xleftarrow{\$} \{0,1\}^\lambda$

$c_0 = (m_0 \oplus F[i_0], i_0)$ →



F

$m_0 = c_0 \oplus F[i_0]$

Stateless Multi-Message Secure Encryption

What if Alice and Bob shared an *exponential* amount of randomness?

$F =$

x	r
000…000	11001010
000…001	10011111
000…010	10010010
000…011	10111111

$\cdots$

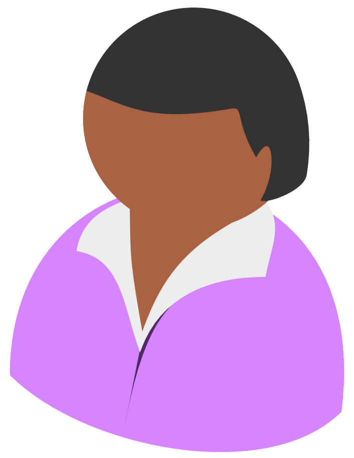
F is a *random function*

The probability of sampling the same index is *negligible*, so this is secure!

F

$i_0 \xleftarrow{\$} \{0,1\}^\lambda$

$c_0 = (m_0 \oplus F[i_0], i_0)$



F

$m_0 = c_0 \oplus F[i_0]$

Random Functions

x	y
000…000	11001010
000…001	10011111
000…010	10010010
000…011	10111111

. . .

Random Functions

$$F : \{0,1\}^\lambda \rightarrow \{0,1\}^\lambda$$

x	y
000…000	11001010
000…001	10011111
000…010	10010010
000…011	10111111

. . .

Random Functions

$$F : \{0,1\}^\lambda \to \{0,1\}^\lambda$$

How many possible functions are there?

x	y
000…000	11001010
000…001	10011111
000…010	10010010
000…011	10111111

. . .

Random Functions

$F : \{0,1\}^\lambda \rightarrow \{0,1\}^\lambda$

How many possible functions are there?

As a table F has 2^λ rows

x	y
000…000	11001010
000…001	10011111
000…010	10010010
000…011	10111111

$\cdots$

Random Functions

$F : \{0,1\}^\lambda \to \{0,1\}^\lambda$

How many possible functions are there?

As a table F has 2^λ rows

Each output is λ bits

x	y
000…000	11001010
000…001	10011111
000…010	10010010
000…011	10111111

$\cdots$

Random Functions

$$F : \{0,1\}^\lambda \rightarrow \{0,1\}^\lambda$$

How many possible functions are there?

x	y
000…000	11001010
000…001	10011111
000…010	10010010
000…011	10111111

$\cdots$

As a table F has 2^λ rows

Each output is λ bits

So there are $2^\lambda \cdot \lambda$ total bits to fill in the table

Random Functions

$F : \{0,1\}^\lambda \rightarrow \{0,1\}^\lambda$

How many possible functions are there?

As a table F has 2^λ rows

Each output is λ bits

So there are $2^\lambda \cdot \lambda$ total bits to fill in the table

$\Rightarrow 2^{\lambda 2^\lambda}$ possible functions!

x	y
000…000	11001010
000…001	10011111
000…010	10010010
000…011	10111111

$\cdots$

Random Functions

Random Functions

We will view random functions in two ways

Random Functions

We will view random functions in two ways

(1) As a large table with each possible output

x	y
000…000	11001010
000…001	10011111
000…010	10010010
000…011	10111111

…

Random Functions

We will view random functions in two ways

(1) As a large table with each possible output

x	y
000…000	11001010
000…001	10011111
000…010	10010010
000…011	10111111

. . .

(2) As a large table that is *filled in as it is queried*

Random Functions

We will view random functions in two ways

(1) As a large table with each possible output

x	y
000…000	11001010
000…001	10011111
000…010	10010010
000…011	10111111

. . .

(2) As a large table that is *filled in as it is queried*

x	y

Random Functions

We will view random functions in two ways

(1) As a large table with each possible output

x	y
000…000	11001010
000…001	10011111
000…010	10010010
000…011	10111111

$\cdots$

(2) As a large table that is *filled in as it is queried*

$F(101)?$

x	y

Random Functions

We will view random functions in two ways

(1) As a large table with each possible output

x	y
000…000	11001010
000…001	10011111
000…010	10010010
000…011	10111111

$\cdots$

(2) As a large table that is *filled in as it is queried*

$F(101)?$

$$101010 \xleftarrow{\$} \{0,1\}^6$$

x	y

Random Functions

We will view random functions in two ways

(1) As a large table with each possible output

x	y
000…000	11001010
000…001	10011111
000…010	10010010
000…011	10111111

$\cdots$

(2) As a large table that is *filled in as it is queried*

$F(101)?$

$$101010 \xleftarrow{\$} \{0,1\}^6$$

x	y
101	

Random Functions

We will view random functions in two ways

(1) As a large table with each possible output

x	y
000…000	11001010
000…001	10011111
000…010	10010010
000…011	10111111

$\cdots$

(2) As a large table that is *filled in as it is queried*

$F(101)?$

$$101010 \xleftarrow{\$} \{0,1\}^6$$

x	y
101	101010

Random Functions

We will view random functions in two ways

(1) As a large table with each possible output

x	y
000…000	11001010
000…001	10011111
000…010	10010010
000…011	10111111

$\cdots$

(2) As a large table that is *filled in as it is queried*

$F(101)?$

$101010 \xleftarrow{\$} \{0,1\}^6$

x	y
101	101010

101010

Random Functions

We will view random functions in two ways

(1) As a large table with each possible output

x	y
000…000	11001010
000…001	10011111
000…010	10010010
000…011	10111111

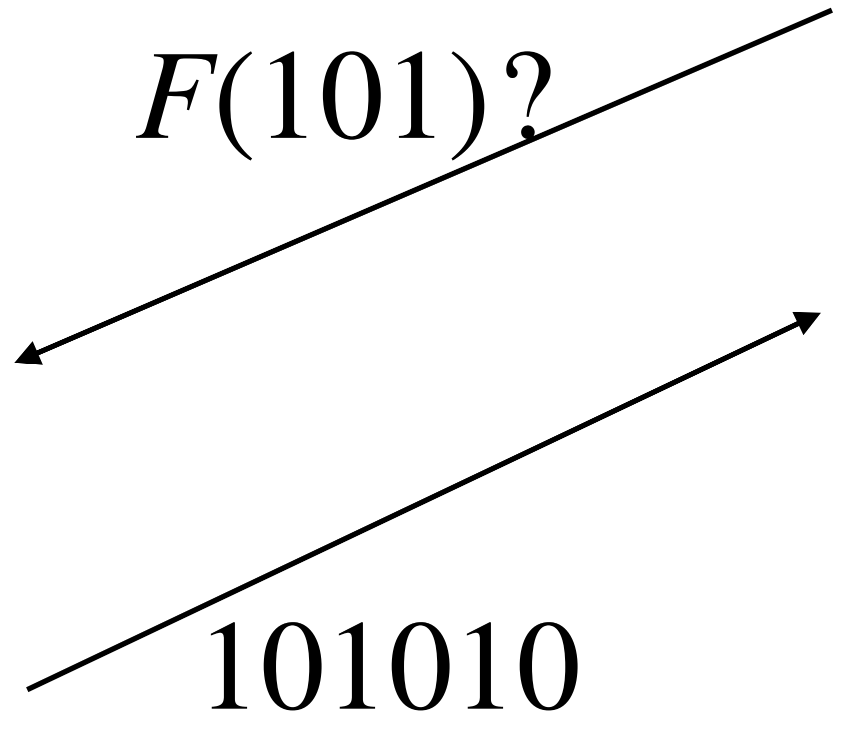
$\cdots$

(2) As a large table that is *filled in as it is queried*

If all you have is oracle access to F (i.e. you can get outputs, but don't have the function's *description*), these two are *identical*. They have the same output distribution

$$101010 \xleftarrow{\$} \{0,1\}^6$$

x	y
101	101010



Random Functions

Random Functions

Random functions are very *useful.*

Random Functions

Random functions are very *useful.*

But we can't actually use them. They require way too many bits to store.

Random Functions

Random functions are very *useful.*

But we can't actually use them. They require way too many bits to store.

Goal: use crypto to create a function that *looks like* a random function, but can be described with polynomial bits! A *Pseudorandom Function* (PRF)

Random Functions

Random functions are very *useful.*

But we can't actually use them. They require way too many bits to store.

Goal: use crypto to create a function that *looks like* a random function, but can be described with polynomial bits! A *Pseudorandom Function* (PRF)

Computational Indistinguishability!

Pseudorandom Functions

Pseudorandom Functions

But, what are we distinguishing between?

Pseudorandom Functions

But, what are we distinguishing between?

Descriptions of the functions wouldn't work. We just said that random functions are way bigger than what we're building

Pseudorandom Functions

But, what are we distinguishing between?

Descriptions of the functions wouldn't work. We just said that random functions are way bigger than what we're building

Idea: Get to *query* the function and try and distinguish by its *outputs*

Pseudorandom Functions

But, what are we distinguishing between?

Descriptions of the functions wouldn't work. We just said that random functions are way bigger than what we're building

Idea: Get to *query* the function and try and distinguish by its *outputs*

Problem: We can't keep the *description* of the PRF secret (Kerckoff's principal)

Pseudorandom Functions

But, what are we distinguishing between?

Descriptions of the functions wouldn't work. We just said that random functions are way bigger than what we're building

Idea: Get to *query* the function and try and distinguish by its *outputs*

Problem: We can't keep the *description* of the PRF secret (Kerckoff's principal)

Solution: PRFs will be *keyed*, and we'll keep the key secret!

Pseudorandom Functions

But, what are we distinguishing between?

Descriptions of the functions wouldn't work. We just said that random functions are way bigger than what we're building

Idea: Get to *query* the function and try and distinguish by its *outputs*

$$F : \{0,1\}^\lambda \times \{0,1\}^\lambda \rightarrow \{0,1\}^\lambda$$

Problem: We can't keep the *description* of the PRF secret (Kerckoff's principal)

Solution: PRFs will be *keyed*, and we'll keep the key secret!

Pseudorandom Functions

But, what are we distinguishing between?

Descriptions of the functions wouldn't work. We just said that random functions are way bigger than what we're building

Idea: Get to *query* the function and try and distinguish by its *outputs*

$$F : \{0,1\}^\lambda \times \{0,1\}^\lambda \rightarrow \{0,1\}^\lambda$$

$$F(k, x) \rightarrow y$$

Problem: We can't keep the *description* of the PRF secret (Kerckoff's principal)

Solution: PRFs will be *keyed*, and we'll keep the key secret!

Pseudorandom Functions

But, what are we distinguishing between?

Descriptions of the functions wouldn't work. We just said that random functions are way bigger than what we're building

Idea: Get to *query* the function and try and distinguish by its *outputs*

Problem: We can't keep the *description* of the PRF secret (Kerckoff's principal)

Solution: PRFs will be *keyed*, and we'll keep the key secret!

$$F : \{0,1\}^\lambda \times \{0,1\}^\lambda \to \{0,1\}^\lambda$$

$$F(k, x) \to y$$

Can also view this as a *family* of functions

Pseudorandom Functions

But, what are we distinguishing between?

Descriptions of the functions wouldn't work. We just said that random functions are way bigger than what we're building

Idea: Get to *query* the function and try and distinguish by its *outputs*

$$F : \{0,1\}^\lambda \times \{0,1\}^\lambda \to \{0,1\}^\lambda$$

$$F(k, x) \to y$$

Problem: We can't keep the *description* of the PRF secret (Kerckoff's principal)

Can also view this as a *family* of functions

Solution: PRFs will be *keyed*, and we'll keep the key secret!

$$\{F_k\}_{k \in \{0,1\}^\lambda}$$

Pseudorandom Functions

But, what are we distinguishing between?

Descriptions of the functions wouldn't work. We just said that random functions are way bigger than what we're building

Idea: Get to *query* the function and try and distinguish by its *outputs*

$$F : \{0,1\}^\lambda \times \{0,1\}^\lambda \to \{0,1\}^\lambda$$

$$F(k, x) \to y$$

Problem: We can't keep the *description* of the PRF secret (Kerckoff's principal)

Can also view this as a *family* of functions

Solution: PRFs will be *keyed*, and we'll keep the key secret!

$$\{F_k\}_{k \in \{0,1\}^\lambda} \qquad F_k : \{0,1\}^\lambda \to \{0,1\}^\lambda$$