Pseudorandomness II

601.442/642 Modern Cryptography

5th February 2026

Announcement

- Homework 2 is due **today**.
- Homework 3 will be out today and due next Thursday (12th Feb).

Recap: Pseudorandom Generator

<u>**Pseudorandom Generator**</u>

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

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

Recap: One-Time Computational Security

One-Time Computational Security

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

Recap: One-Time Computational Security

One-Time Computational Security

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

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

Recap: 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.

Constructing PRGs

- Do PRGs exist?

Constructing PRGs

- Do PRGs exist?
    - If P = NP, then PRGs don't exist.

Constructing PRGs

- Do PRGs exist?
  - If P = NP, then PRGs don't exist.
- How are they constructed?

Constructing PRGs

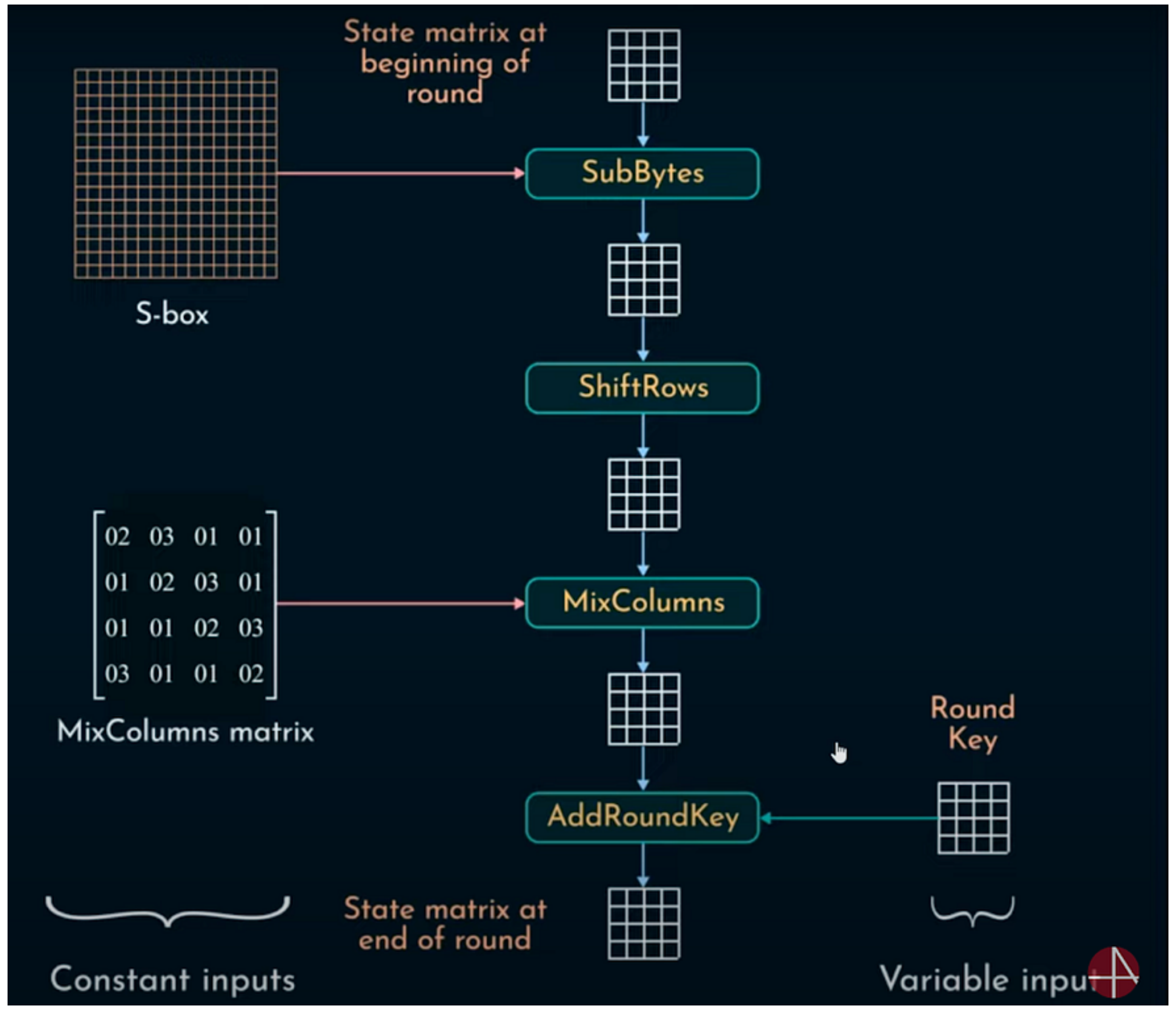
- Do PRGs exist?
  - If P = NP, then PRGs don't exist.
- How are they constructed?
  - **Practical Methodology:** Efficient PRGs for real-world deployment.

Constructing PRGs

- Do PRGs exist?
  - If P = NP, then PRGs don't exist.
- How are they constructed?
  - **Practical Methodology:** Efficient PRGs for real-world deployment.
    - Start from a design framework e.g., a good mixing function composed many times looks random.

Constructing PRGs

- Do PRGs exist?
  - If P = NP, then PRGs don't exist.
- How are they constructed?
  - **Practical Methodology:** Efficient PRGs for real-world deployment.
    - Start from a design framework e.g., a good mixing function composed many times looks random.
    - Come up with a candidate construction.

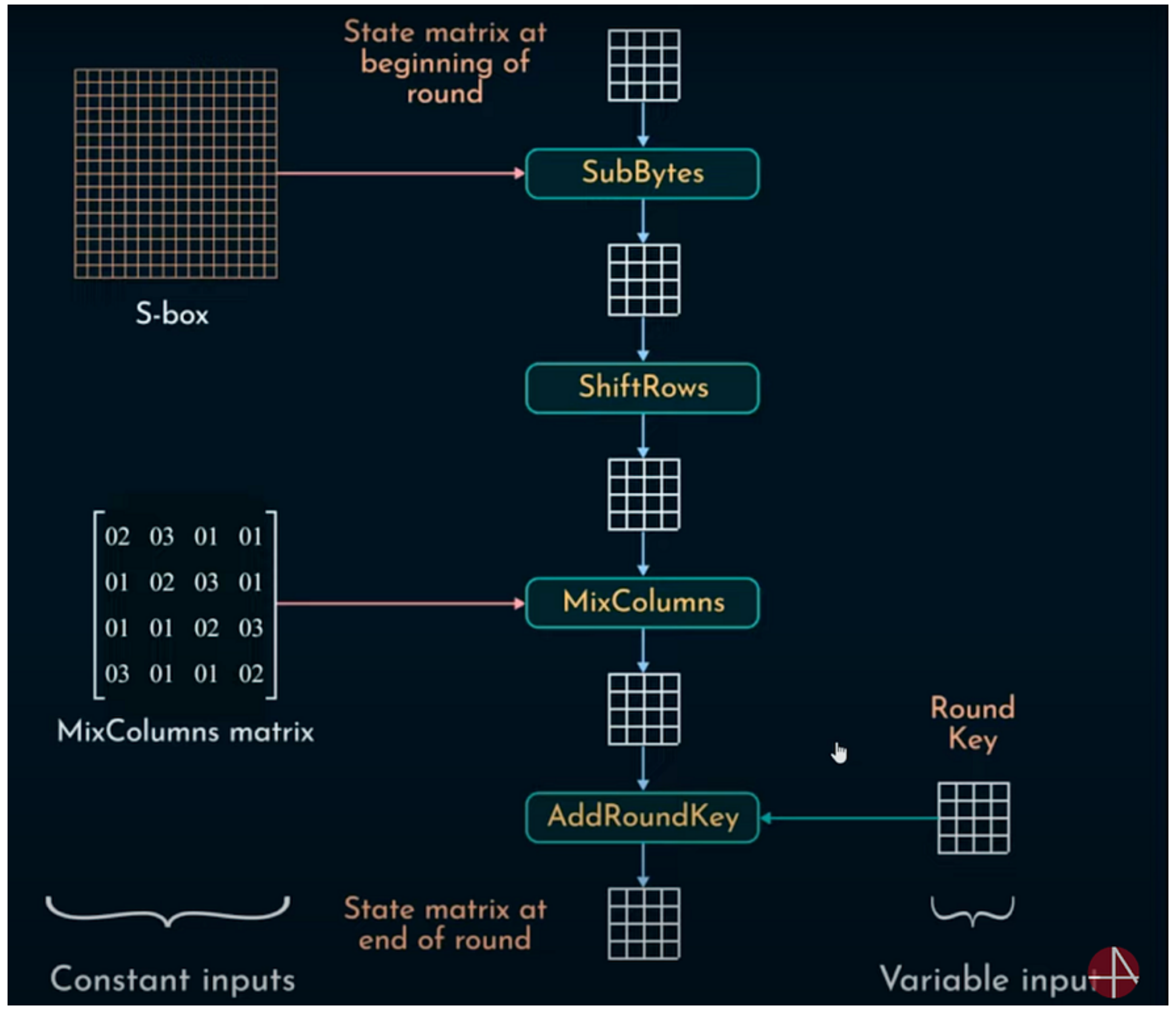


nesoacademy.org

Single round of AES

Constructing PRGs

- Do PRGs exist?
  - If P = NP, then PRGs don’t exist.
- How are they constructed?
  - **Practical Methodology:** Efficient PRGs for real-world deployment.
    - Start from a design framework e.g., a good mixing function composed many times looks random.
    - Come up with a candidate construction.
    - Do extensive cryptanalysis.



nesoacademy.org

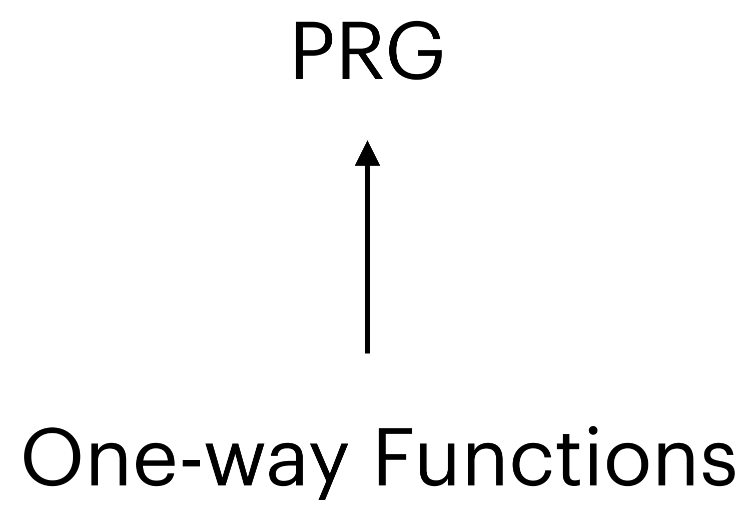
Single round of AES

Constructing PRGs

- Do PRGs exist?
    - If P = NP, then PRGs don't exist.
- How are they constructed?
    - **Practical Methodology:** Efficient PRGs for real-world deployment.
    - **Foundational Methodology:** Build from simpler primitives and well-studied hard problems.

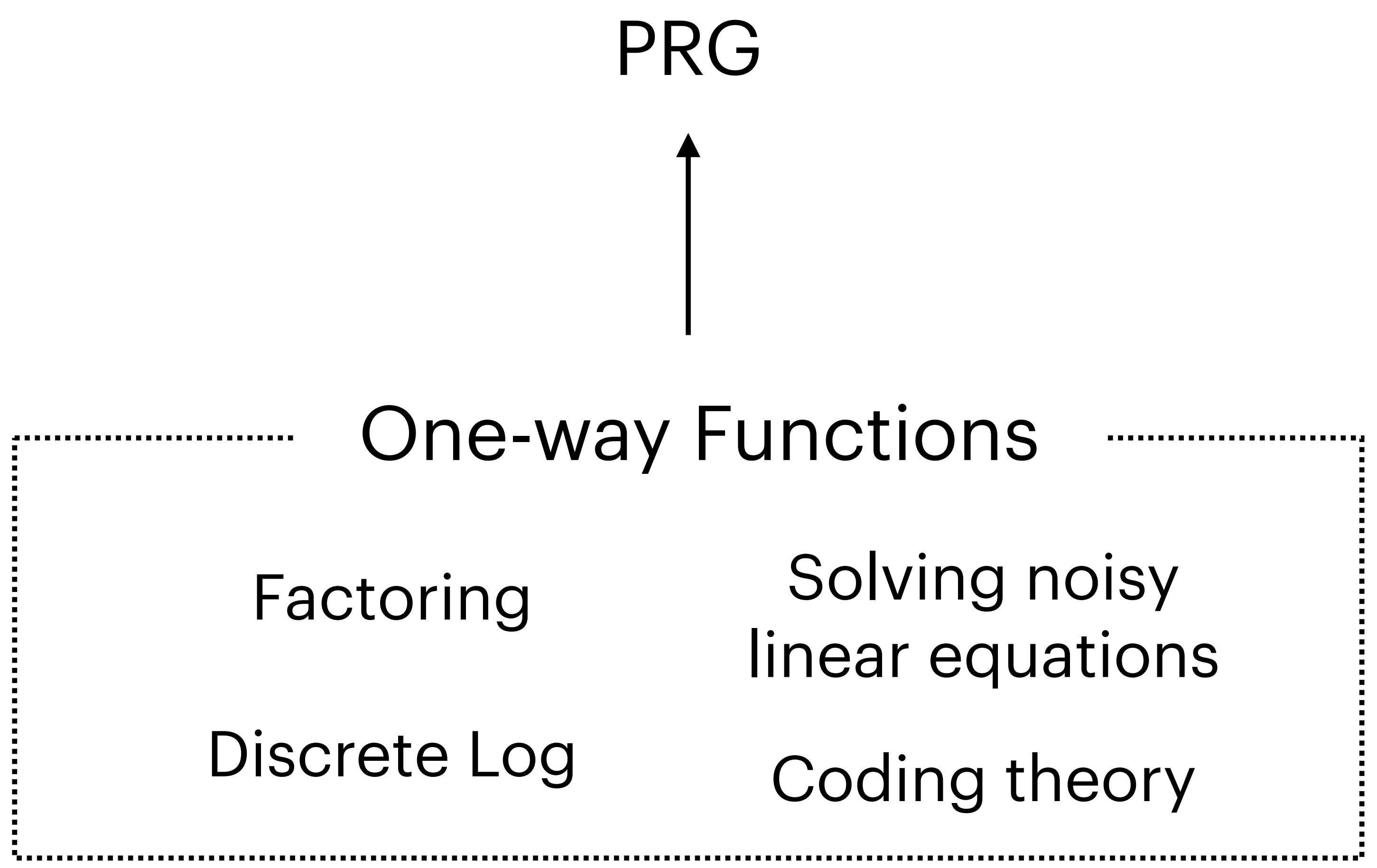
Constructing PRGs

- Do PRGs exist?
  - If P = NP, then PRGs don't exist.
- How are they constructed?
  - **Practical Methodology:** Efficient PRGs for real-world deployment.
  - **Foundational Methodology:** Build from simpler primitives and well-studied hard problems.
    - Helps understand relationship with other primitives.



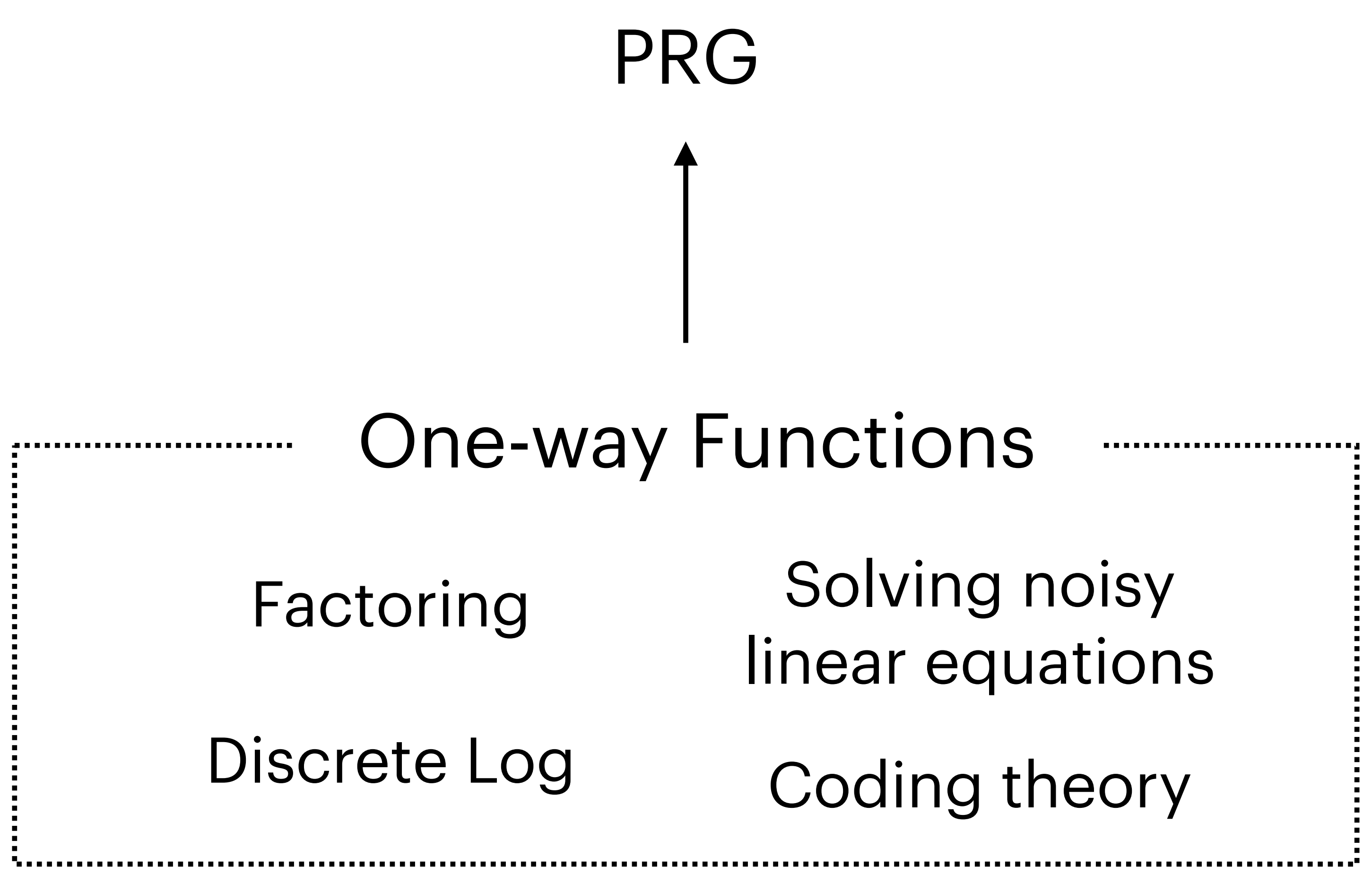
Constructing PRGs

- Do PRGs exist?
  - If P = NP, then PRGs don't exist.
- How are they constructed?
  - **Practical Methodology:** Efficient PRGs for real-world deployment.
  - **Foundational Methodology:** Build from simpler primitives and well-studied hard problems.
    - Helps understand relationship with other primitives.
    - Builds confidence in existence of PRGs.



Constructing PRGs

- Do PRGs exist?
  - If P = NP, then PRGs don't exist.
- How are they constructed?
  - **Practical Methodology:** Efficient PRGs for real-world deployment.
  - **Foundational Methodology:** Build from simpler primitives and well-studied hard problems.
    - Helps understand relationship with other primitives.
    - Builds confidence in existence of PRGs.
  - We will focus on the foundational methodology in this course.



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.

How to encrypt messages longer than $\ell(\lambda)$ bits?

How to encrypt multiple 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.

How to encrypt messages longer than $\ell(\lambda)$ bits?

How to encrypt multiple messages?

A PRG with 1-bit stretch implies a PRG with arbitrary polynomial-bit stretch.

PRG Length Extension

Theorem: For all polynomials $\ell = \ell(\lambda)$, if there exists a PRG with one-bit stretch then there exists a PRG with ℓ-bit stretch.

PRG Length Extension

Theorem: For all polynomials $\ell = \ell(\lambda)$, if there exists a PRG with one-bit stretch then there exists a PRG with ℓ-bit stretch.

Proof:

Let $G : \{0,1\}^\lambda \to \{0,1\}^{\lambda+1}$ be a PRG.

PRG Length Extension

Theorem: For all polynomials $\ell = \ell(\lambda)$, if there exists a PRG with one-bit stretch then there exists a PRG with ℓ-bit stretch.

Proof:

Let $G : \{0,1\}^{\lambda} \to \{0,1\}^{\lambda+1}$ be a PRG. We will construct G_{poly} with ℓ-bit stretch.

$G_{\text{poly}}(s)$:

PRG Length Extension

Theorem: For all polynomials $\ell = \ell(\lambda)$, if there exists a PRG with one-bit stretch then there exists a PRG with ℓ-bit stretch.

Proof:

Let $G : \{0,1\}^\lambda \to \{0,1\}^{\lambda+1}$ be a PRG. We will construct G_{poly} with ℓ-bit stretch.

$G_{\text{poly}}(s)$:

$$x_0 := s$$

PRG Length Extension

Theorem: For all polynomials $\ell = \ell(\lambda)$, if there exists a PRG with one-bit stretch then there exists a PRG with ℓ-bit stretch.

Proof:

Let $G : \{0,1\}^\lambda \to \{0,1\}^{\lambda+1}$ be a PRG. We will construct G_{poly} with ℓ-bit stretch.

$G_{\text{poly}}(s)$:

$$x_0 := s$$

$$x_1 \parallel b_1 := G(x_0)$$

PRG Length Extension

Theorem: For all polynomials $\ell = \ell(\lambda)$, if there exists a PRG with one-bit stretch then there exists a PRG with ℓ-bit stretch.

Proof:

Let $G : \{0,1\}^\lambda \to \{0,1\}^{\lambda+1}$ be a PRG. We will construct G_{poly} with ℓ-bit stretch.

$G_{\text{poly}}(s)$:

$$x_0 := s$$
$$x_1 \parallel b_1 := G(x_0)$$
$$x_2 \parallel b_2 := G(x_1)$$

PRG Length Extension

Theorem: For all polynomials $\ell = \ell(\lambda)$, if there exists a PRG with one-bit stretch then there exists a PRG with ℓ-bit stretch.

Proof:

Let $G : \{0,1\}^\lambda \rightarrow \{0,1\}^{\lambda+1}$ be a PRG. We will construct G_{poly} with ℓ-bit stretch.

$G_{\text{poly}}(s)$:

$$x_0 := s$$
$$x_1 \parallel b_1 := G(x_0)$$
$$x_2 \parallel b_2 := G(x_1)$$
$$\vdots$$
$$x_\ell \parallel b_\ell := G(x_{\ell-1})$$

PRG Length Extension

Theorem: For all polynomials $\ell = \ell(\lambda)$, if there exists a PRG with one-bit stretch then there exists a PRG with ℓ-bit stretch.

Proof:

Let $G : \{0,1\}^\lambda \to \{0,1\}^{\lambda+1}$ be a PRG. We will construct G_{poly} with ℓ-bit stretch.

$G_{\text{poly}}(s)$:

$$x_0 := s$$
$$x_1 \| b_1 := G(x_0)$$
$$x_2 \| b_2 := G(x_1)$$
$$\vdots$$
$$x_\ell \| b_\ell := G(x_{\ell-1})$$

Output $b_1 \| b_2 \| \ldots \| b_\ell$

PRG Length Extension

Theorem: For all polynomials $\ell = \ell(\lambda)$, if there exists a PRG with one-bit stretch then there exists a PRG with ℓ-bit stretch.

Proof:

Let $G : \{0,1\}^\lambda \to \{0,1\}^{\lambda+1}$ be a PRG. We will construct G_{poly} with ℓ-bit stretch.

$G_{\text{poly}}(s)$:

$$x_0 := s$$
$$x_1 \| b_1 := G(x_0)$$
$$x_2 \| b_2 := G(x_1)$$
$$\vdots$$
$$x_\ell \| b_\ell := G(x_{\ell-1})$$

Output $b_1 \| b_2 \| \ldots \| b_\ell$

Security?

PRG Length Extension: Proof

Claim: If G is a PRG then G_{poly} is a PRG.

PRG Length Extension: Proof

Claim: If G is a PRG then G_{poly} is a PRG.

Proof:

G_{poly} can be evaluated in polynomial time. Why?

PRG Length Extension: Proof

Claim: If G is a PRG then G_{poly} is a PRG.

Proof:

G_{poly} can be evaluated in polynomial time. Why?

We need to show $\{G_{\text{poly}}(s) : s \xleftarrow{\$} \{0,1\}^{\lambda}\} \overset{c}{\approx} \{r : r \xleftarrow{\$} \{0,1\}^{\ell(\lambda)}\}$.

PRG Length Extension: Proof

Claim: If G is a PRG then G_{poly} is a PRG.

Proof:

G_{poly} can be evaluated in polynomial time. Why?

We need to show $\{G_{\text{poly}}(s) : s \xleftarrow{\$} \{0,1\}^\lambda\} \overset{c}{\approx} \{r : r \xleftarrow{\$} \{0,1\}^{\ell(\lambda)}\}$.

H_0 :

$G_{\text{poly}}(s)$:

$$x_0 := s$$
$$x_1 \,\|\, b_1 := G(x_0)$$
$$x_2 \,\|\, b_2 := G(x_1)$$
$$\vdots$$
$$x_\ell \,\|\, b_\ell := G(x_{\ell-1})$$

Output $b_1 \,\|\, b_2 \,\|\, \ldots \,\|\, b_\ell$

PRG Length Extension: Proof

Claim: If G is a PRG then G_{poly} is a PRG.

Proof:

G_{poly} can be evaluated in polynomial time. Why?

We need to show $\{G_{\text{poly}}(s) : s \xleftarrow{\$} \{0,1\}^{\lambda}\} \overset{c}{\approx} \{r : r \xleftarrow{\$} \{0,1\}^{\ell(\lambda)}\}$.

H_0 :

$G_{\text{poly}}(s)$:

$$x_0 := s$$
$$x_1 \parallel b_1 := G(x_0)$$
$$x_2 \parallel b_2 := G(x_1)$$
$$\vdots$$
$$x_\ell \parallel b_\ell := G(x_{\ell-1})$$

Output $b_1 \parallel b_2 \parallel \ldots \parallel b_\ell$

H_1 :

$G_{\text{poly}}(s)$:

$$x_0 := s$$
$$x_1 \parallel u_1 := s_1 \parallel u_1$$
$$x_2 \parallel b_2 := G(x_1)$$
$$\vdots$$
$$x_\ell \parallel b_\ell := G(x_{\ell-1})$$

Output $u_1 \parallel b_2 \parallel \ldots \parallel b_\ell$

PRG Length Extension: Proof

Claim: If G is a PRG then G_{poly} is a PRG.

Proof:

G_{poly} can be evaluated in polynomial time. Why?

We need to show $\{G_{\text{poly}}(s) : s \xleftarrow{\$} \{0,1\}^\lambda\} \quad \overset{c}{\approx} \quad \{r : r \xleftarrow{\$} \{0,1\}^{\ell(\lambda)}\}$.

H_0 :

$G_{\text{poly}}(s)$:

$$x_0 := s$$
$$x_1 \parallel b_1 := G(x_0)$$
$$x_2 \parallel b_2 := G(x_1)$$
$$\vdots$$
$$x_\ell \parallel b_\ell := G(x_{\ell-1})$$

Output $b_1 \parallel b_2 \parallel \ldots \parallel b_\ell$

H_1 :

$G_{\text{poly}}(s)$:

$$x_0 := s$$
$$x_1 \parallel u_1 := s_1 \parallel u_1$$
$$x_2 \parallel b_2 := G(x_1)$$
$$\vdots$$
$$x_\ell \parallel b_\ell := G(x_{\ell-1})$$

Output $u_1 \parallel b_2 \parallel \ldots \parallel b_\ell$

…

H_ℓ :

$G_{\text{poly}}(s)$:

$$x_0 := s$$
$$x_1 \parallel u_1 := s_1 \parallel u_1$$
$$x_2 \parallel u_2 := s_2 \parallel u_2$$
$$\vdots$$
$$x_\ell \parallel u_\ell := s_\ell \parallel u_\ell$$

Output $u_1 \parallel u_2 \parallel \ldots \parallel u_\ell$

PRG Length Extension: Proof

Claim: If G is a PRG then G_{poly} is a PRG.

Proof:

G_{poly} can be evaluated in polynomial time. Why?

We need to show $\{G_{\text{poly}}(s) : s \xleftarrow{\$} \{0,1\}^{\lambda}\} \overset{c}{\approx} \{r : r \xleftarrow{\$} \{0,1\}^{\ell(\lambda)}\}$.

Suffices to show that $\forall i \in \{0,\ldots,\ell-1\}$, $H_i \overset{c}{\approx} H_{i+1}$. Why?

H_0 :

$G_{\text{poly}}(s)$:

$$x_0 := s$$
$$x_1 \,\|\, b_1 := G(x_0)$$
$$x_2 \,\|\, b_2 := G(x_1)$$
$$\vdots$$
$$x_\ell \,\|\, b_\ell := G(x_{\ell-1})$$

Output $b_1 \,\|\, b_2 \,\|\, \ldots \,\|\, b_\ell$

H_1 :

$G_{\text{poly}}(s)$:

$$x_0 := s$$
$$x_1 \,\|\, u_1 := s_1 \,\|\, u_1$$
$$x_2 \,\|\, b_2 := G(x_1)$$
$$\vdots$$
$$x_\ell \,\|\, b_\ell := G(x_{\ell-1})$$

Output $u_1 \,\|\, b_2 \,\|\, \ldots \,\|\, b_\ell$

$\ldots$

H_ℓ :

$G_{\text{poly}}(s)$:

$$x_0 := s$$
$$x_1 \,\|\, u_1 := s_1 \,\|\, u_1$$
$$x_2 \,\|\, u_2 := s_2 \,\|\, u_2$$
$$\vdots$$
$$x_\ell \,\|\, u_\ell := s_\ell \,\|\, u_\ell$$

Output $u_1 \,\|\, u_2 \,\|\, \ldots \,\|\, u_\ell$

PRG Length Extension: Proof

Claim: If G is a PRG then for each $i \in \{0, \ldots, \ell - 1\}$, $\quad H_i \overset{c}{\approx} H_{i+1}$.

PRG Length Extension: Proof

Claim: If G is a PRG then for each $i \in \{0,\ldots,\ell-1\}$, $\quad H_i \overset{c}{\approx} H_{i+1}$.

Proof:

$H_i:$

$G_{\text{poly}}(s):$

$$x_0 := s$$
$$x_1 \parallel u_1 := s_1 \parallel u_1$$
$$\vdots$$
$$x_i \parallel u_i := s_i \parallel u_i$$
$$x_{i+1} \parallel b_{i+1} := G(x_i)$$
$$x_{i+2} \parallel b_{i+2} := G(x_{i+1})$$
$$\vdots$$
$$x_\ell \parallel b_\ell := G(x_{\ell-1})$$

Output

$$u_1 \parallel \ldots \parallel u_i \parallel b_{i+1} \parallel b_{i+2} \parallel \ldots \parallel b_\ell$$

$H_{i+1}:$

$G_{\text{poly}}(s):$

$$x_0 := s$$
$$x_1 \parallel u_1 := s_1 \parallel u_1$$
$$\vdots$$
$$x_i \parallel u_i := s_i \parallel u_i$$
$$x_{i+1} \parallel u_{i+1} := s_{i+1} \parallel u_{i+1}$$
$$x_{i+2} \parallel b_{i+2} := G(x_{i+1})$$
$$\vdots$$
$$x_\ell \parallel b_\ell := G(x_{\ell-1})$$

Output

$$u_1 \parallel \ldots \parallel u_i \parallel u_{i+1} \parallel b_{i+2} \parallel \ldots \parallel b_\ell$$

PRG Length Extension: Proof

Claim: If G is a PRG then for each $i \in \{0,\ldots,\ell-1\}$, $\quad H_i \overset{c}{\approx} H_{i+1}$.

Proof:

$H_i :$

$G_{\text{poly}}(s) :$

$$x_0 := s$$
$$x_1 \parallel u_1 := s_1 \parallel u_1$$
$$\vdots$$
$$x_i \parallel u_i := s_i \parallel u_i$$
$$x_{i+1} \parallel b_{i+1} := G(x_i)$$
$$x_{i+2} \parallel b_{i+2} := G(x_{i+1})$$
$$\vdots$$
$$x_\ell \parallel b_\ell := G(x_{\ell-1})$$

Output

$u_1 \parallel \ldots \parallel u_i \parallel b_{i+1} \parallel b_{i+2} \parallel \ldots \parallel b_\ell$

$H_{i+1} :$

$G_{\text{poly}}(s) :$

$$x_0 := s$$
$$x_1 \parallel u_1 := s_1 \parallel u_1$$
$$\vdots$$
$$x_i \parallel u_i := s_i \parallel u_i$$
$$x_{i+1} \parallel u_{i+1} := s_{i+1} \parallel u_{i+1}$$
$$x_{i+2} \parallel b_{i+2} := G(x_{i+1})$$
$$\vdots$$
$$x_\ell \parallel b_\ell := G(x_{\ell-1})$$

Output

$u_1 \parallel \ldots \parallel u_i \parallel u_{i+1} \parallel b_{i+2} \parallel \ldots \parallel b_\ell$

Intuition:

PRG Length Extension: Proof

Claim: If G is a PRG then for each $i \in \{0,\ldots,\ell-1\}$, $\quad H_i \overset{c}{\approx} H_{i+1}$.

Proof:

H_i :

$G_{\text{poly}}(s)$:

$$x_0 := s$$
$$x_1 \| u_1 := s_1 \| u_1$$
$$\vdots$$
$$x_i \| u_i := s_i \| u_i$$
$$x_{i+1} \| b_{i+1} := G(x_i)$$
$$x_{i+2} \| b_{i+2} := G(x_{i+1})$$
$$\vdots$$
$$x_\ell \| b_\ell := G(x_{\ell-1})$$

Output

$u_1 \| \ldots \| u_i \| b_{i+1} \| b_{i+2} \| \ldots \| b_\ell$

H_{i+1} :

$G_{\text{poly}}(s)$:

$$x_0 := s$$
$$x_1 \| u_1 := s_1 \| u_1$$
$$\vdots$$
$$x_i \| u_i := s_i \| u_i$$
$$x_{i+1} \| u_{i+1} := s_{i+1} \| u_{i+1}$$
$$x_{i+2} \| b_{i+2} := G(x_{i+1})$$
$$\vdots$$
$$x_\ell \| b_\ell := G(x_{\ell-1})$$

Output

$u_1 \| \ldots \| u_i \| u_{i+1} \| b_{i+2} \| \ldots \| b_\ell$

Intuition:

b_{i+1} is indistinguishable from u_{i+1} if G is a PRG.

PRG Length Extension: Proof

Claim: If G is a PRG then for each $i \in \{0,\ldots,\ell-1\}$, $\quad H_i \overset{c}{\approx} H_{i+1}$.

Proof:

H_i :

$G_{\text{poly}}(s)$:

$$x_0 := s$$
$$x_1 \,\|\, u_1 := s_1 \,\|\, u_1$$
$$\vdots$$
$$x_i \,\|\, u_i := s_i \,\|\, u_i$$
$$x_{i+1} \,\|\, b_{i+1} := G(x_i)$$
$$x_{i+2} \,\|\, b_{i+2} := G(x_{i+1})$$
$$\vdots$$
$$x_\ell \,\|\, b_\ell := G(x_{\ell-1})$$

Output

$u_1 \,\|\, \ldots \,\|\, u_i \,\|\, b_{i+1} \,\|\, b_{i+2} \,\|\, \ldots \,\|\, b_\ell$

H_{i+1} :

$G_{\text{poly}}(s)$:

$$x_0 := s$$
$$x_1 \,\|\, u_1 := s_1 \,\|\, u_1$$
$$\vdots$$
$$x_i \,\|\, u_i := s_i \,\|\, u_i$$
$$x_{i+1} \,\|\, u_{i+1} := s_{i+1} \,\|\, u_{i+1}$$
$$x_{i+2} \,\|\, b_{i+2} := G(x_{i+1})$$
$$\vdots$$
$$x_\ell \,\|\, b_\ell := G(x_{\ell-1})$$

Output

$u_1 \,\|\, \ldots \,\|\, u_i \,\|\, u_{i+1} \,\|\, b_{i+2} \,\|\, \ldots \,\|\, b_\ell$

Intuition:

b_{i+1} is indistinguishable from u_{i+1} if G is a PRG.

How will we formally prove this?

PRG Length Extension: Proof

Claim: If G is a PRG then for each $i \in \{0,\ldots,\ell-1\}$, $\quad H_i \overset{c}{\approx} H_{i+1}$.

Proof:

$H_i :$

$G_{\text{poly}}(s)$:

$$x_0 := s$$
$$x_1 \parallel u_1 := s_1 \parallel u_1$$
$$\vdots$$
$$x_i \parallel u_i := s_i \parallel u_i$$
$$x_{i+1} \parallel b_{i+1} := G(x_i)$$
$$x_{i+2} \parallel b_{i+2} := G(x_{i+1})$$
$$\vdots$$
$$x_\ell \parallel b_\ell := G(x_{\ell-1})$$

Output

$u_1 \parallel \ldots \parallel u_i \parallel b_{i+1} \parallel b_{i+2} \parallel \ldots \parallel b_\ell$

$H_{i+1} :$

$G_{\text{poly}}(s)$:

$$x_0 := s$$
$$x_1 \parallel u_1 := s_1 \parallel u_1$$
$$\vdots$$
$$x_i \parallel u_i := s_i \parallel u_i$$
$$x_{i+1} \parallel u_{i+1} := s_{i+1} \parallel u_{i+1}$$
$$x_{i+2} \parallel b_{i+2} := G(x_{i+1})$$
$$\vdots$$
$$x_\ell \parallel b_\ell := G(x_{\ell-1})$$

Output

$u_1 \parallel \ldots \parallel u_i \parallel u_{i+1} \parallel b_{i+2} \parallel \ldots \parallel b_\ell$

Intuition:

b_{i+1} is indistinguishable from u_{i+1} if G is a PRG.

How will we formally prove this?

Reduction!

PRG Length Extension: Proof

Claim: If G is a PRG then for each $i \in \{0,\ldots,\ell-1\}, \quad H_i \overset{c}{\approx} H_{i+1}$.

Proof:

PRG Length Extension: Proof

Claim: If G is a PRG then for each $i \in \{0, \ldots, \ell - 1\}$, $\quad H_i \overset{c}{\approx} H_{i+1}$.

Proof:

PRG Length Extension: Proof

Claim: If G is a PRG then for each $i \in \{0, \ldots, \ell - 1\}$, $H_i \overset{c}{\approx} H_{i+1}$.

Proof:

$A_{H_i, H_{i+1}}$

PRG Length Extension: Proof

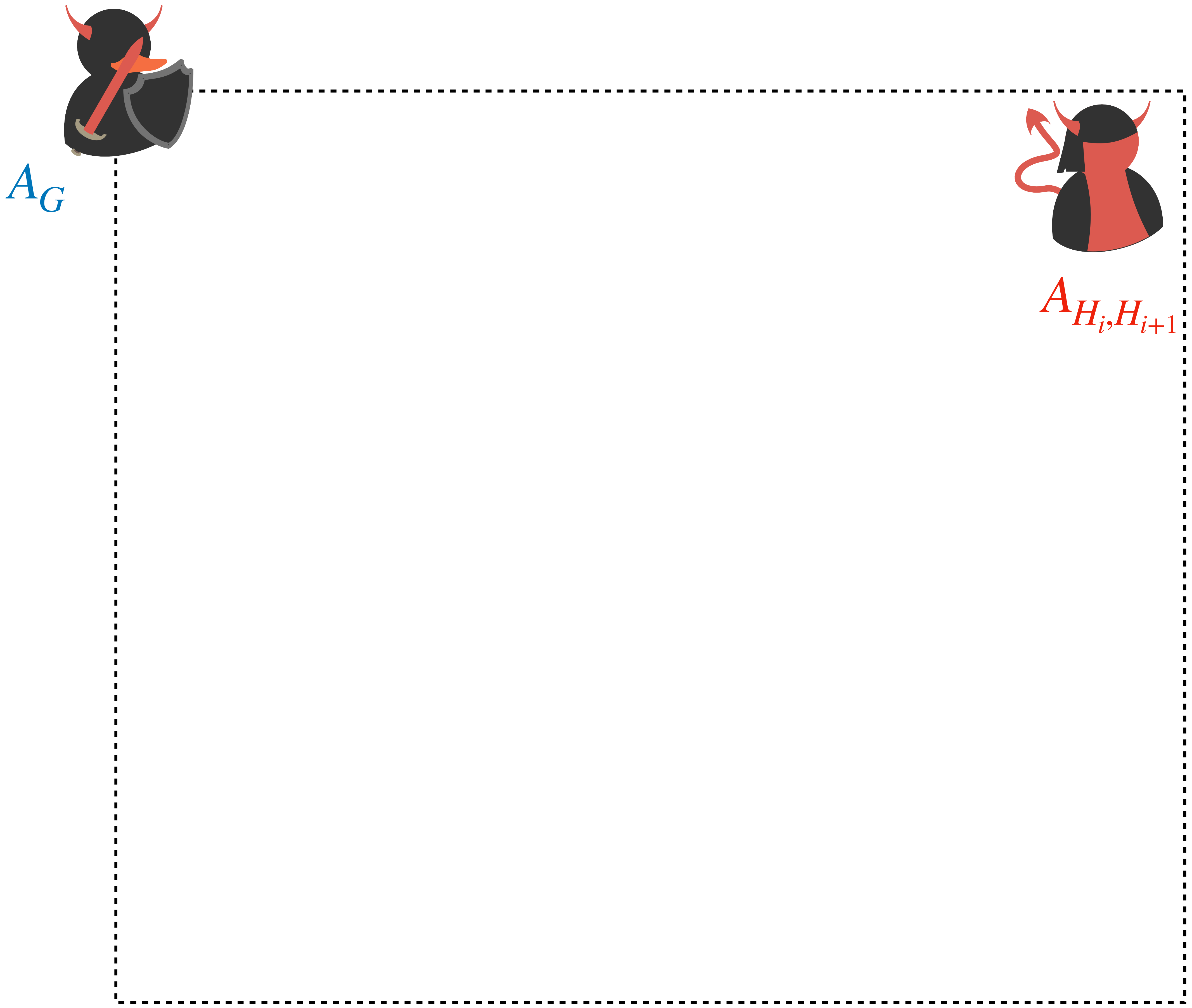
Claim: If G is a PRG then for each $i \in \{0,\ldots,\ell-1\}$, $\quad H_i \overset{c}{\approx} H_{i+1}$.

Proof:

PRG Length Extension: Proof

Claim: If G is a PRG then for each $i \in \{0, \ldots, \ell - 1\}$, $\quad H_i \overset{c}{\approx} H_{i+1}$.

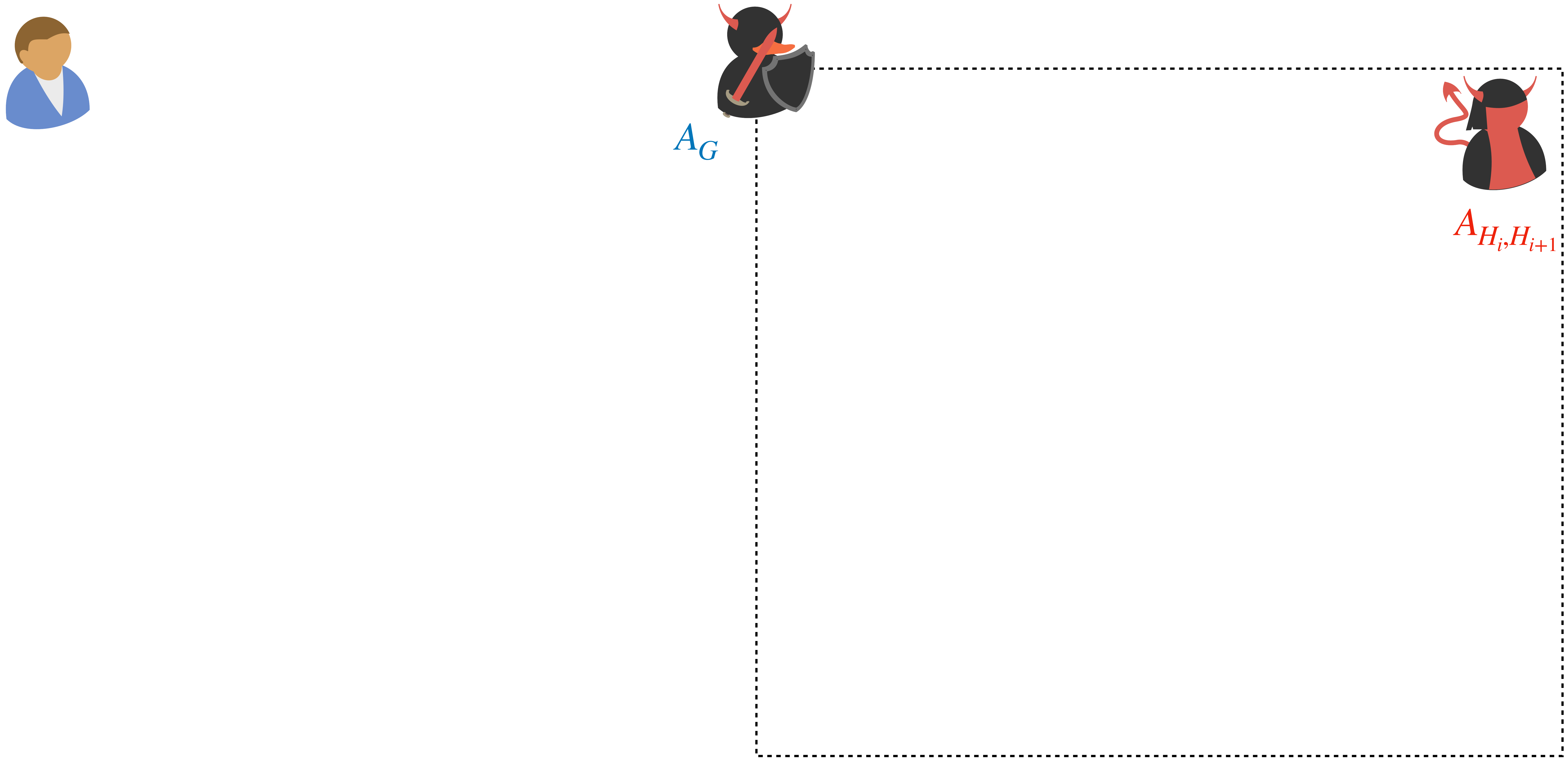
Proof:



PRG Length Extension: Proof

Claim: If G is a PRG then for each $i \in \{0,\ldots,\ell-1\}$, $\quad H_i \overset{c}{\approx} H_{i+1}$.

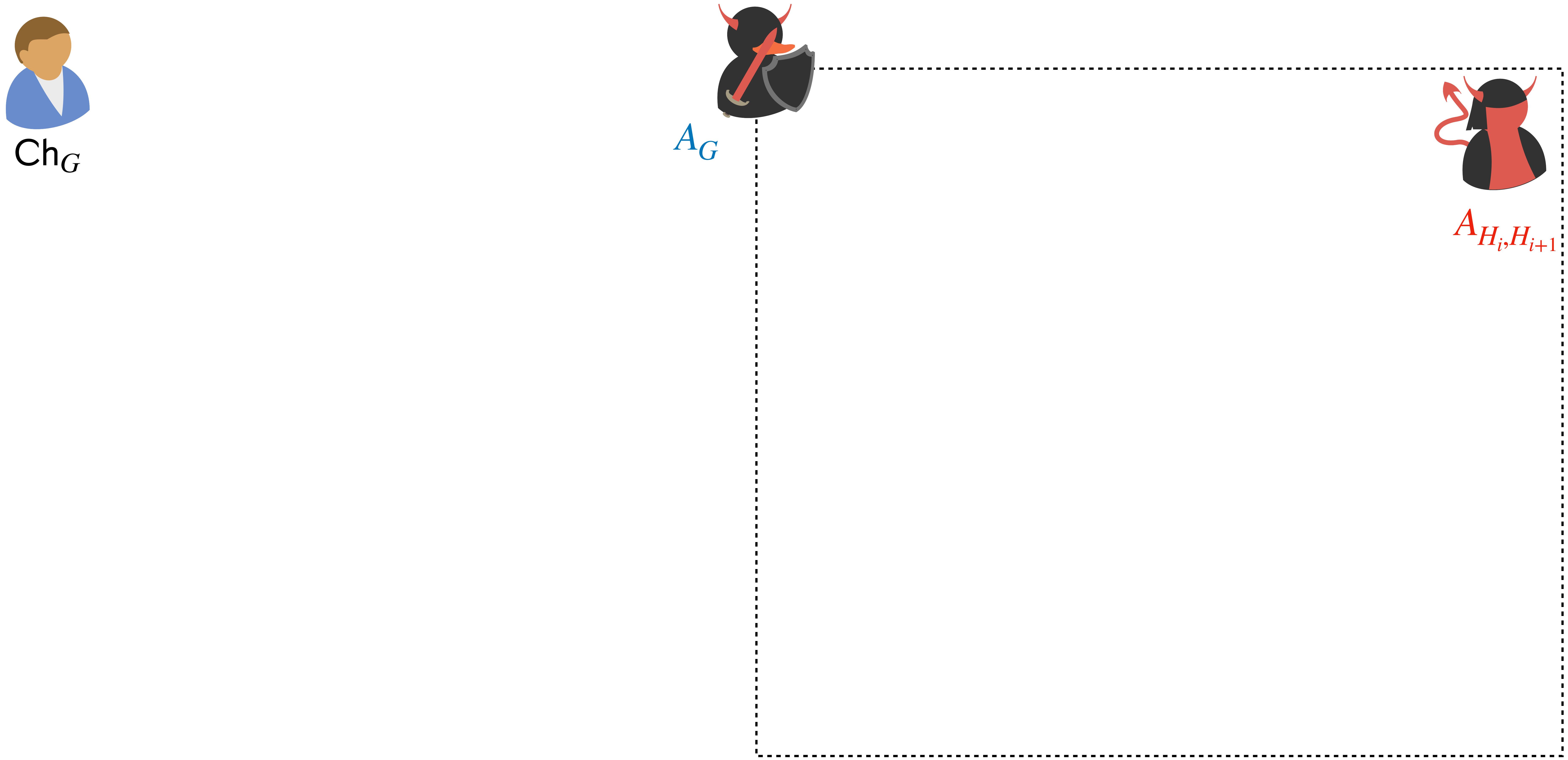
Proof:



PRG Length Extension: Proof

Claim: If G is a PRG then for each $i \in \{0, \ldots, \ell - 1\}$, $\quad H_i \overset{c}{\approx} H_{i+1}$.

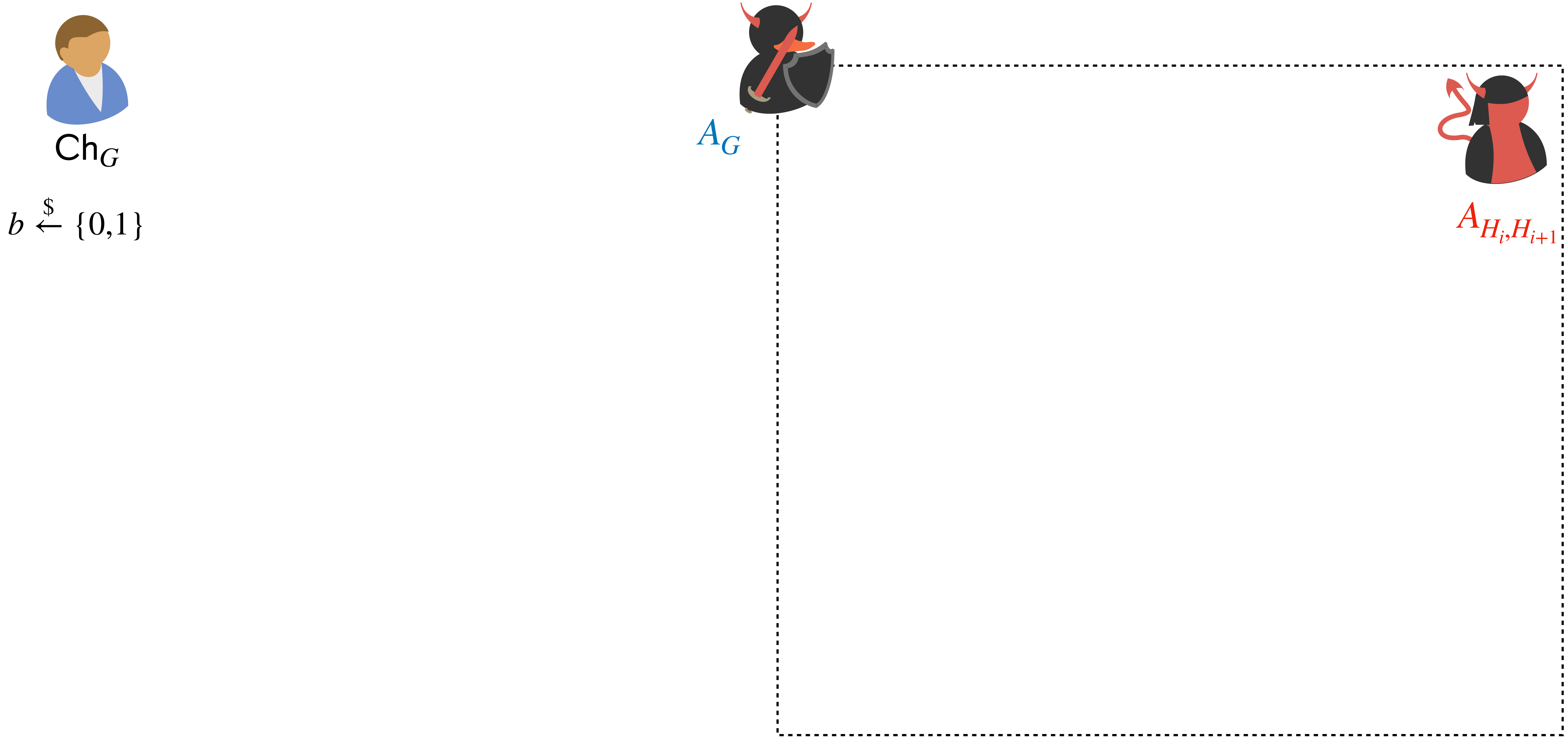
Proof:



PRG Length Extension: Proof

Claim: If G is a PRG then for each $i \in \{0, \ldots, \ell - 1\}$, $\quad H_i \overset{c}{\approx} H_{i+1}$.

Proof:



PRG Length Extension: Proof

Claim: If G is a PRG then for each $i \in \{0, \ldots, \ell - 1\}$, $\quad H_i \overset{c}{\approx} H_{i+1}$.

Proof:

Ch_G

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

If $b = 0$:

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

$r := G(s)$

A_G

$A_{H_i, H_{i+1}}$

PRG Length Extension: Proof

Claim: If G is a PRG then for each $i \in \{0,\ldots,\ell-1\}$, $\quad H_i \stackrel{c}{\approx} H_{i+1}$.

Proof:

Ch_G

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

If $b = 0$:

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

$r := G(s)$

If $b = 1$:

$r \stackrel{\$}{\leftarrow} \{0,1\}^{\lambda+1}$

PRG Length Extension: Proof

Claim: If G is a PRG then for each $i \in \{0,\ldots,\ell-1\}$, $H_i \overset{c}{\approx} H_{i+1}$.

Proof:

Ch_G

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

If $b = 0$:

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

$r := G(s)$

If $b = 1$:

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

$r = r_1 \parallel r_2 \parallel \ldots \parallel r_{\lambda+1}$ →

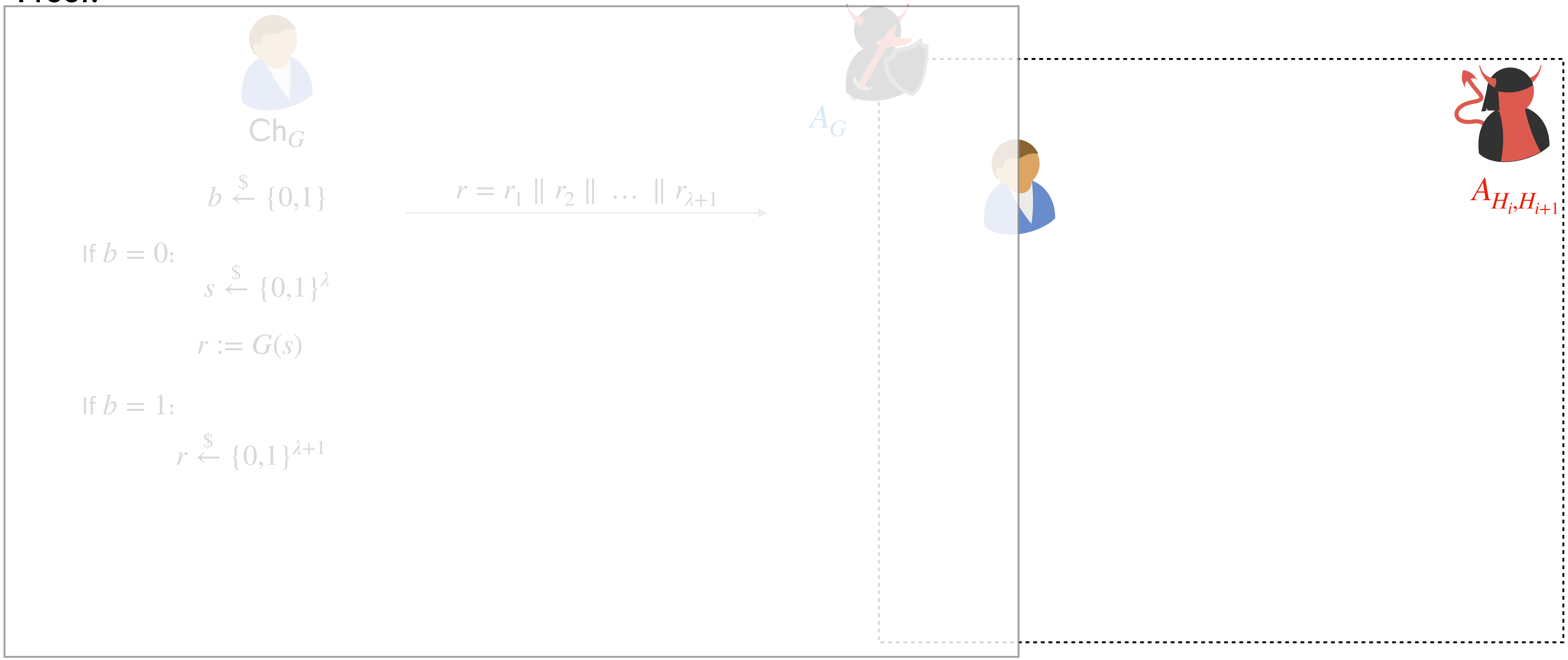
A_G

$A_{H_i,H_{i+1}}$

PRG Length Extension: Proof

Claim: If G is a PRG then for each $i \in \{0,\ldots,\ell-1\}$, $\quad H_i \overset{c}{\approx} H_{i+1}$.

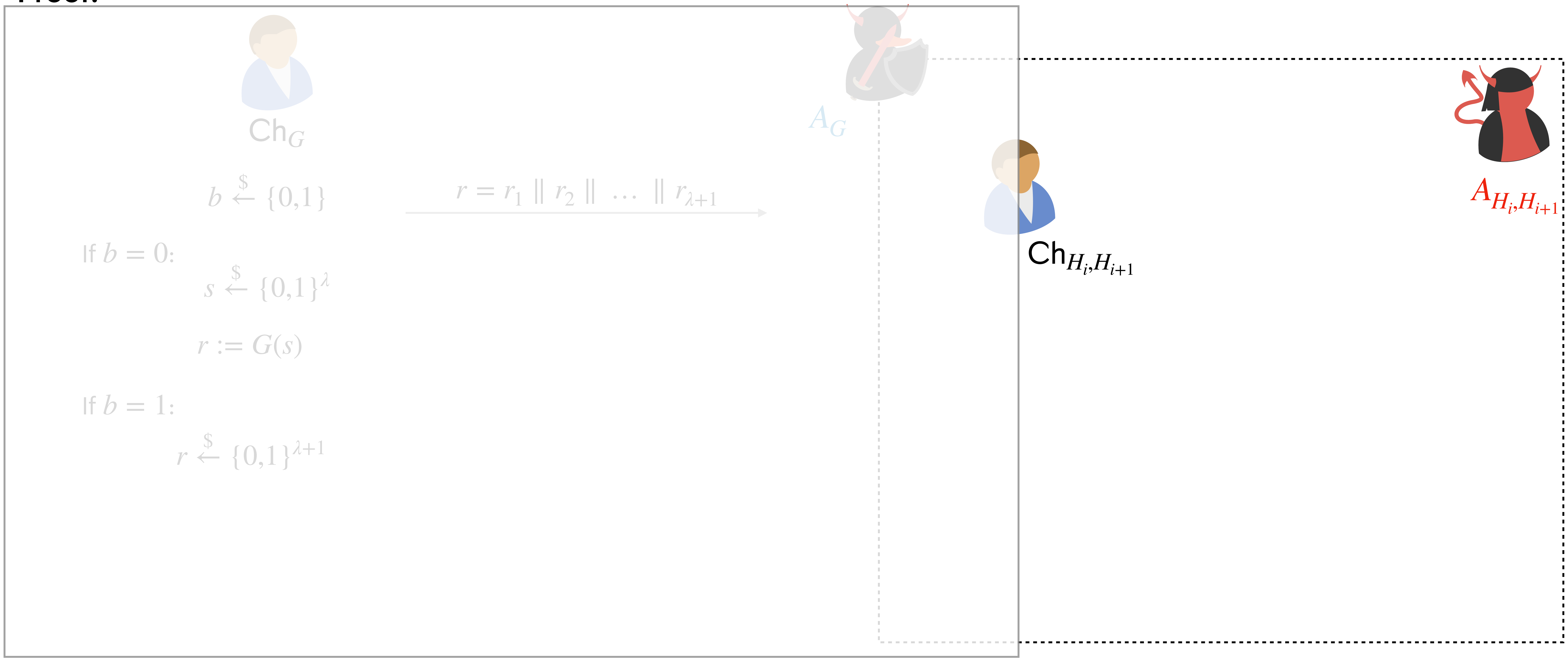
Proof:



PRG Length Extension: Proof

Claim: If G is a PRG then for each $i \in \{0,\ldots,\ell-1\}$, $\quad H_i \overset{c}{\approx} H_{i+1}$.

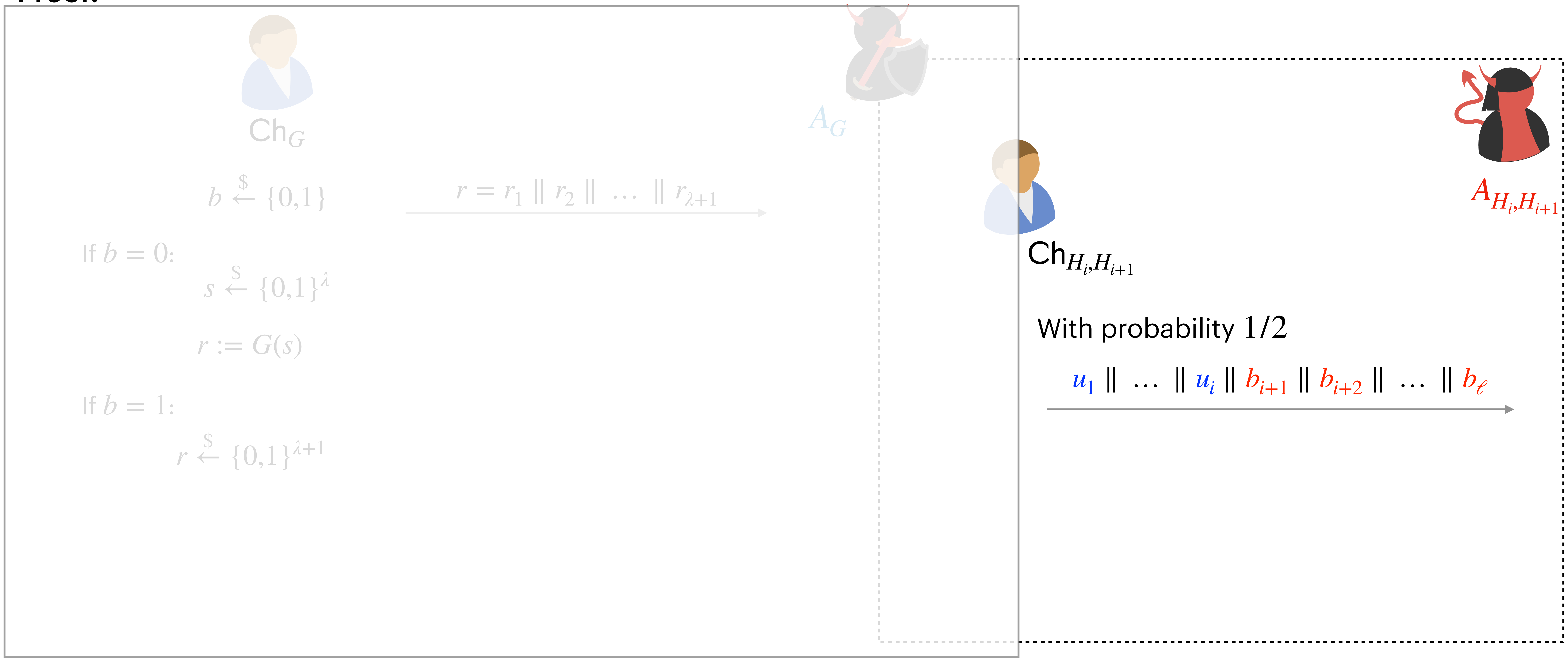
Proof:



PRG Length Extension: Proof

Claim: If G is a PRG then for each $i \in \{0,\ldots,\ell-1\}$, $\quad H_i \overset{c}{\approx} H_{i+1}$.

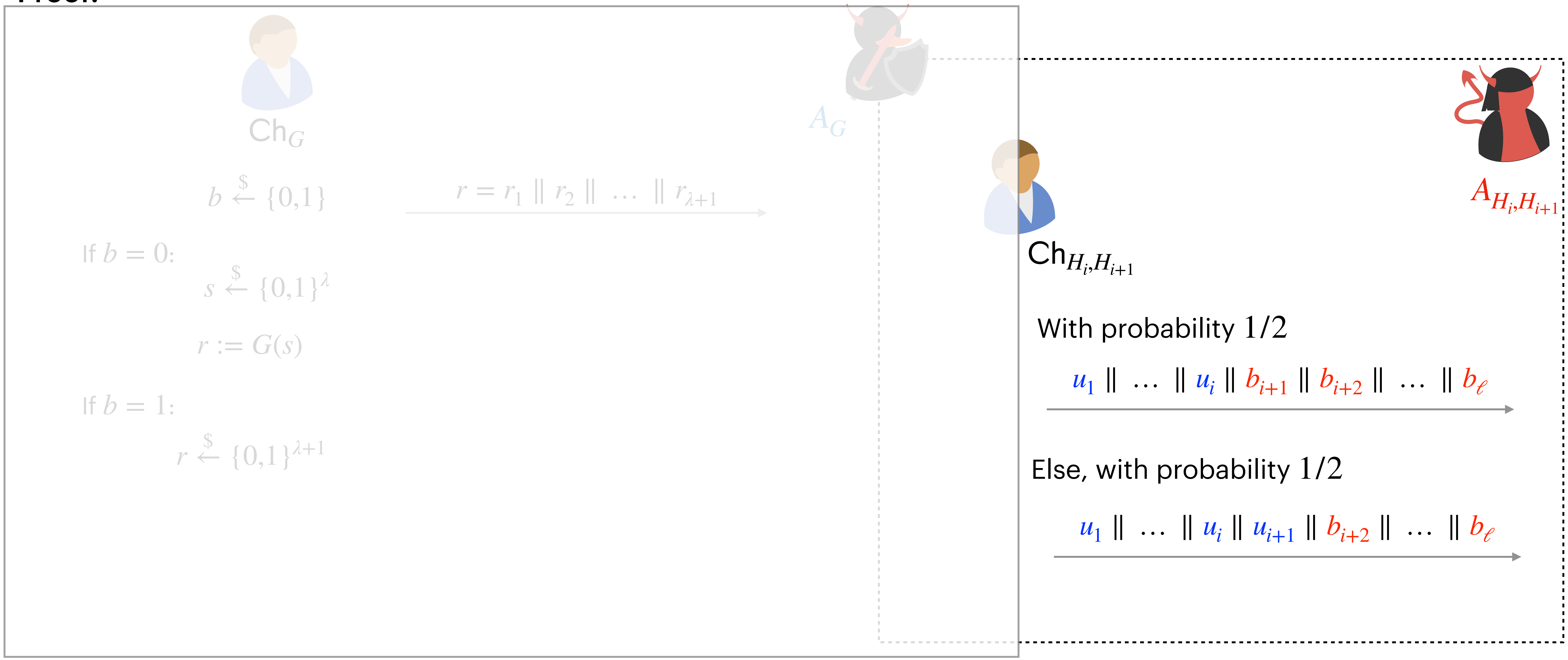
Proof:



PRG Length Extension: Proof

Claim: If G is a PRG then for each $i \in \{0,\ldots,\ell-1\}$, $\quad H_i \stackrel{c}{\approx} H_{i+1}$.

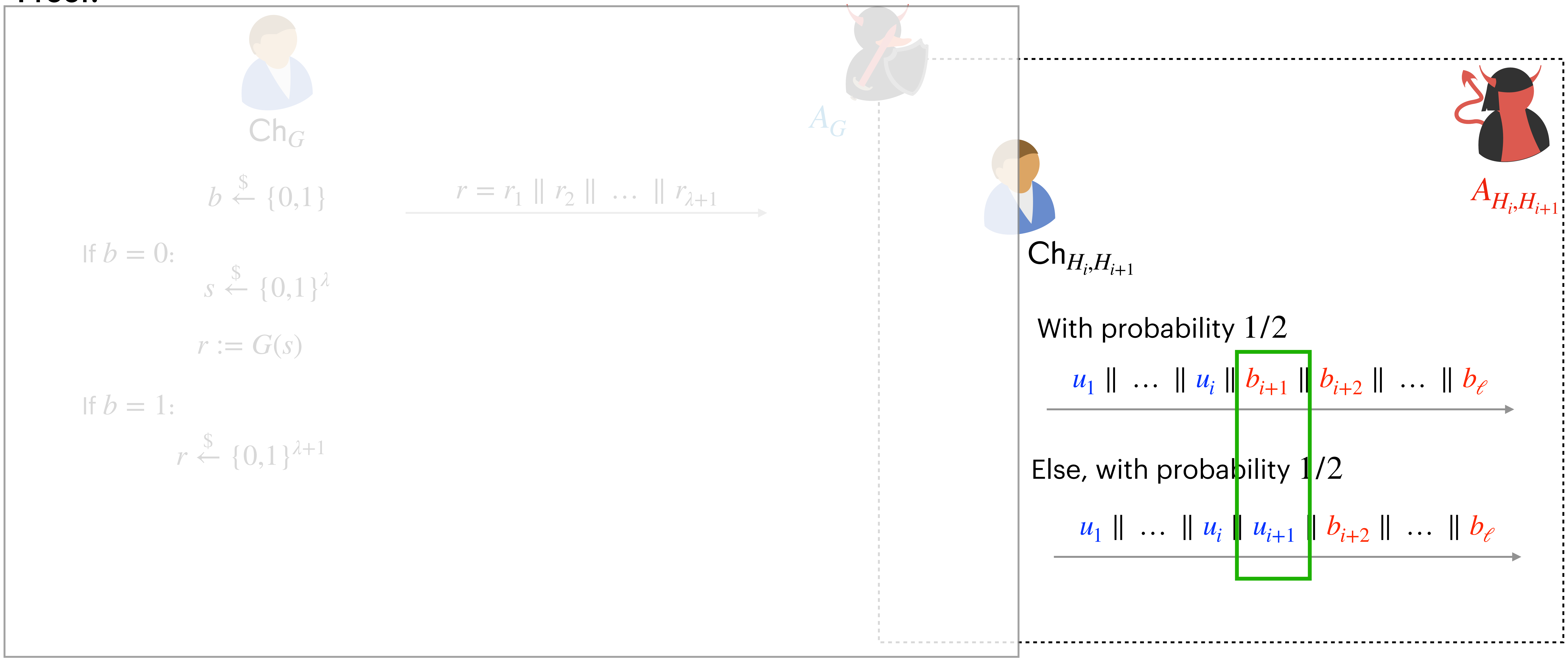
Proof:



PRG Length Extension: Proof

Claim: If G is a PRG then for each $i \in \{0,\ldots,\ell-1\}$, $H_i \overset{c}{\approx} H_{i+1}$.

Proof:



PRG Length Extension: Proof

Claim: If G is a PRG then for each $i \in \{0,\ldots,\ell-1\}$, $H_i \overset{c}{\approx} H_{i+1}$.

Proof:

Ch_G

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

If $b = 0$:

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

$r := G(s)$

If $b = 1$:

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

$r = r_1 \parallel r_2 \parallel \ldots \parallel r_{\lambda+1}$ →

A_G

$A_{H_i,H_{i+1}}$

PRG Length Extension: Proof

Claim: If G is a PRG then for each $i \in \{0,\ldots,\ell-1\}$, $H_i \overset{c}{\approx} H_{i+1}$.

Proof:

Ch_G

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

If $b = 0$:

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

$r := G(s)$

If $b = 1$:

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

$r = r_1 \parallel r_2 \parallel \ldots \parallel r_{\lambda+1}$ →

A_G

$u_1, u_2, \ldots, u_i \overset{\$}{\leftarrow} \{0,1\}$

$A_{H_i, H_{i+1}}$

PRG Length Extension: Proof

Claim: If G is a PRG then for each $i \in \{0,\ldots,\ell - 1\}$, $\quad H_i \overset{c}{\approx} H_{i+1}$.

Proof:

Ch_G

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

If $b = 0$:

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

$r := G(s)$

If $b = 1$:

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

$r = r_1 \parallel r_2 \parallel \ \ldots \ \parallel r_{\lambda+1}$ →

A_G

$A_{H_i,H_{i+1}}$

$u_1, u_2, \ldots, u_i \xleftarrow{\$} \{0,1\}$

$x_{i+1} := r_1 \parallel r_2 \parallel \ \ldots \ \parallel r_\lambda$

PRG Length Extension: Proof

Claim: If G is a PRG then for each $i \in \{0,\ldots,\ell-1\}$, $\quad H_i \overset{c}{\approx} H_{i+1}$.

Proof:

Ch_G

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

If $b = 0$:

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

$r := G(s)$

If $b = 1$:

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

$r = r_1 \parallel r_2 \parallel \ldots \parallel r_{\lambda+1}$ →

A_G

$A_{H_i,H_{i+1}}$

$u_1, u_2, \ldots, u_i \overset{\$}{\leftarrow} \{0,1\}$

$x_{i+1} := r_1 \parallel r_2 \parallel \ldots \parallel r_\lambda$

$x_{i+2} \parallel b_{i+2} := G(x_{i+1})$

PRG Length Extension: Proof

Claim: If G is a PRG then for each $i \in \{0,\ldots,\ell-1\}$, $\quad H_i \overset{c}{\approx} H_{i+1}$.

Proof:

Ch_G

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

If $b = 0$:

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

$r := G(s)$

If $b = 1$:

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

$r = r_1 \parallel r_2 \parallel \ldots \parallel r_{\lambda+1}$ →

A_G

$A_{H_i,H_{i+1}}$

$u_1, u_2, \ldots, u_i \overset{\$}{\leftarrow} \{0,1\}$

$x_{i+1} := r_1 \parallel r_2 \parallel \ldots \parallel r_\lambda$

$x_{i+2} \parallel b_{i+2} := G(x_{i+1})$

$x_{i+3} \parallel b_{i+3} := G(x_{i+2})$

$\vdots$

$x_\ell \parallel b_\ell := G(x_{\ell-1})$

PRG Length Extension: Proof

Claim: If G is a PRG then for each $i \in \{0,\ldots,\ell-1\}$, $\quad H_i \overset{c}{\approx} H_{i+1}$.

Proof:

Ch_G

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

If $b = 0$:

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

$r := G(s)$

If $b = 1$:

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

$r = r_1 \parallel r_2 \parallel \ldots \parallel r_{\lambda+1}$ →

A_G

$A_{H_i,H_{i+1}}$

$u_1, u_2, \ldots, u_i \overset{\$}{\leftarrow} \{0,1\}$

$x_{i+1} := r_1 \parallel r_2 \parallel \ldots \parallel r_\lambda$

$x_{i+2} \parallel b_{i+2} := G(x_{i+1})$

$x_{i+3} \parallel b_{i+3} := G(x_{i+2})$

$\vdots$

$x_\ell \parallel b_\ell := G(x_{\ell-1})$

$u_1 \parallel u_2 \parallel \ldots \parallel u_i \parallel$ →

PRG Length Extension: Proof

Claim: If G is a PRG then for each $i \in \{0,\ldots,\ell-1\}$, $\quad H_i \overset{c}{\approx} H_{i+1}$.

Proof:

Ch_G

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

If $b = 0$:

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

$r := G(s)$

If $b = 1$:

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

$r = r_1 \parallel r_2 \parallel \ldots \parallel r_{\lambda+1}$ →

A_G

$A_{H_i,H_{i+1}}$

$u_1, u_2, \ldots, u_i \xleftarrow{\$} \{0,1\}$

$x_{i+1} := r_1 \parallel r_2 \parallel \ldots \parallel r_\lambda$

$x_{i+2} \parallel b_{i+2} := G(x_{i+1})$

$x_{i+3} \parallel b_{i+3} := G(x_{i+2})$

$\vdots$

$x_\ell \parallel b_\ell := G(x_{\ell-1})$

$u_1 \parallel u_2 \parallel \ldots \parallel u_i \parallel r_{\lambda+1} \parallel b_{i+2} \parallel \ldots \parallel b_\ell$ →

PRG Length Extension: Proof

Claim: If G is a PRG then for each $i \in \{0, \ldots, \ell - 1\}$, $\quad H_i \overset{c}{\approx} H_{i+1}$.

Proof:

Ch_G

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

If $b = 0$:

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

$r := G(s)$

If $b = 1$:

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

$r = r_1 \| r_2 \| \ldots \| r_{\lambda+1}$ →

A_G

$A_{H_i, H_{i+1}}$

$u_1, u_2, \ldots, u_i \overset{\$}{\leftarrow} \{0,1\}$

$x_{i+1} := r_1 \| r_2 \| \ldots \| r_\lambda$

$x_{i+2} \| b_{i+2} := G(x_{i+1})$

$x_{i+3} \| b_{i+3} := G(x_{i+2})$

$\vdots$

$x_\ell \| b_\ell := G(x_{\ell-1})$

$u_1 \| u_2 \| \ldots \| u_i \| r_{\lambda+1} \| b_{i+2} \| \ldots \| b_\ell$ →

Input mapping:

PRG Length Extension: Proof

Claim: If G is a PRG then for each $i \in \{0,\ldots,\ell-1\}$, $\quad H_i \overset{c}{\approx} H_{i+1}$.

Proof:

Ch_G

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

If $b = 0$:

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

$r := G(s)$

If $b = 1$:

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

$r = r_1 \parallel r_2 \parallel \ldots \parallel r_{\lambda+1}$ →

A_G

$A_{H_i,H_{i+1}}$

$u_1, u_2, \ldots, u_i \overset{\$}{\leftarrow} \{0,1\}$

$x_{i+1} := r_1 \parallel r_2 \parallel \ldots \parallel r_\lambda$

$x_{i+2} \parallel b_{i+2} := G(x_{i+1})$

$x_{i+3} \parallel b_{i+3} := G(x_{i+2})$

$\vdots$

$x_\ell \parallel b_\ell := G(x_{\ell-1})$

$u_1 \parallel u_2 \parallel \ldots \parallel u_i \parallel r_{\lambda+1} \parallel b_{i+2} \parallel \ldots \parallel b_\ell$ →

Input mapping:

If $b = 0$, $r_{\lambda+1}$ is output of G and $A_{H_i,H_{i+1}}$'s view is identical to H_i.

PRG Length Extension: Proof

Claim: If G is a PRG then for each $i \in \{0,\ldots,\ell-1\}$, $H_i \overset{c}{\approx} H_{i+1}$.

Proof:

Ch_G

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

If $b = 0$:

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

$r := G(s)$

If $b = 1$:

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

$r = r_1 \parallel r_2 \parallel \ldots \parallel r_{\lambda+1}$ →

A_G

$A_{H_i,H_{i+1}}$

$u_1, u_2, \ldots, u_i \overset{\$}{\leftarrow} \{0,1\}$

$x_{i+1} := r_1 \parallel r_2 \parallel \ldots \parallel r_\lambda$

$x_{i+2} \parallel b_{i+2} := G(x_{i+1})$

$x_{i+3} \parallel b_{i+3} := G(x_{i+2})$

$\vdots$

$x_\ell \parallel b_\ell := G(x_{\ell-1})$

$u_1 \parallel u_2 \parallel \ldots \parallel u_i \parallel r_{\lambda+1} \parallel b_{i+2} \parallel \ldots \parallel b_\ell$ →

Input mapping:

If $b = 0$, $r_{\lambda+1}$ is output of G and $A_{H_i,H_{i+1}}$'s view is identical to H_i.

If $b = 1$, $r_{\lambda+1}$ is uniformly random and $A_{H_i,H_{i+1}}$'s view is identical to H_{i+1}.

PRG Length Extension: Proof

Claim: If G is a PRG then for each $i \in \{0,\ldots,\ell-1\}$, $\quad H_i \overset{c}{\approx} H_{i+1}$.

Proof:

Ch_G

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

If $b = 0$:

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

$r := G(s)$

If $b = 1$:

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

$r = r_1 \,\|\, r_2 \,\|\, \ldots \,\|\, r_{\lambda+1}$ →

A_G

$A_{H_i,H_{i+1}}$

$u_1, u_2, \ldots, u_i \xleftarrow{\$} \{0,1\}$

$x_{i+1} := r_1 \,\|\, r_2 \,\|\, \ldots \,\|\, r_\lambda$

$x_{i+2} \,\|\, b_{i+2} := G(x_{i+1})$

$x_{i+3} \,\|\, b_{i+3} := G(x_{i+2})$

$\vdots$

$x_\ell \,\|\, b_\ell := G(x_{\ell-1})$

$u_1 \,\|\, u_2 \,\|\, \ldots \,\|\, u_i \,\|\, r_{\lambda+1} \,\|\, b_{i+2} \,\|\, \ldots \,\|\, b_\ell$ →

PRG Length Extension: Proof

Claim: If G is a PRG then for each $i \in \{0,\ldots,\ell-1\}$, $H_i \overset{c}{\approx} H_{i+1}$.

Proof:

Ch_G

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

If $b = 0$:

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

$r := G(s)$

If $b = 1$:

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

$r = r_1 \| r_2 \| \ldots \| r_{\lambda+1}$ →

A_G

$A_{H_i,H_{i+1}}$

$u_1, u_2, \ldots, u_i \overset{\$}{\leftarrow} \{0,1\}$

$x_{i+1} := r_1 \| r_2 \| \ldots \| r_\lambda$

$x_{i+2} \| b_{i+2} := G(x_{i+1})$

$x_{i+3} \| b_{i+3} := G(x_{i+2})$

$\vdots$

$x_\ell \| b_\ell := G(x_{\ell-1})$

$u_1 \| u_2 \| \ldots \| u_i \| r_{\lambda+1} \| b_{i+2} \| \ldots \| b_\ell$ →

← b'

PRG Length Extension: Proof

Claim: If G is a PRG then for each $i \in \{0,\ldots,\ell-1\}$, $H_i \overset{c}{\approx} H_{i+1}$.

Proof:

Ch_G

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

If $b = 0$:

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

$r := G(s)$

If $b = 1$:

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

$r = r_1 \| r_2 \| \ldots \| r_{\lambda+1}$ →

A_G

$A_{H_i, H_{i+1}}$

$u_1, u_2, \ldots, u_i \xleftarrow{\$} \{0,1\}$

$x_{i+1} := r_1 \| r_2 \| \ldots \| r_\lambda$

$x_{i+2} \| b_{i+2} := G(x_{i+1})$

$x_{i+3} \| b_{i+3} := G(x_{i+2})$

$\vdots$

$x_\ell \| b_\ell := G(x_{\ell-1})$

$u_1 \| u_2 \| \ldots \| u_i \| r_{\lambda+1} \| b_{i+2} \| \ldots \| b_\ell$ →

← b'

← b'

PRG Length Extension: Proof

Claim: If G is a PRG then for each $i \in \{0,\ldots,\ell-1\}$, $\quad H_i \overset{c}{\approx} H_{i+1}$.

Proof:

Ch_G

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

If $b = 0$:

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

$r := G(s)$

If $b = 1$:

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

$r = r_1 \parallel r_2 \parallel \ldots \parallel r_{\lambda+1}$ →

A_G

$A_{H_i,H_{i+1}}$

$u_1, u_2, \ldots, u_i \overset{\$}{\leftarrow} \{0,1\}$

$x_{i+1} := r_1 \parallel r_2 \parallel \ldots \parallel r_\lambda$

$x_{i+2} \parallel b_{i+2} := G(x_{i+1})$

$x_{i+3} \parallel b_{i+3} := G(x_{i+2})$

$\vdots$

$x_\ell \parallel b_\ell := G(x_{\ell-1})$

$u_1 \parallel u_2 \parallel \ldots \parallel u_i \parallel r_{\lambda+1} \parallel b_{i+2} \parallel \ldots \parallel b_\ell$ →

← b'

Output mapping:

← b'

PRG Length Extension: Proof

Claim: If G is a PRG then for each $i \in \{0,\ldots,\ell-1\}$, $\quad H_i \overset{c}{\approx} H_{i+1}$.

Proof:

Ch_G

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

If $b = 0$:

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

$r := G(s)$

If $b = 1$:

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

$r = r_1 \parallel r_2 \parallel \ldots \parallel r_{\lambda+1}$ →

A_G

$A_{H_i,H_{i+1}}$

$u_1, u_2, \ldots, u_i \overset{\$}{\leftarrow} \{0,1\}$

$x_{i+1} := r_1 \parallel r_2 \parallel \ldots \parallel r_\lambda$

$x_{i+2} \parallel b_{i+2} := G(x_{i+1})$

$x_{i+3} \parallel b_{i+3} := G(x_{i+2})$

$\vdots$

$x_\ell \parallel b_\ell := G(x_{\ell-1})$

$u_1 \parallel u_2 \parallel \ldots \parallel u_i \parallel r_{\lambda+1} \parallel b_{i+2} \parallel \ldots \parallel b_\ell$ →

← b'

← b'

Output mapping:

A_G has the same advantage as $A_{H_i,H_{i+1}}$.

PRG Length Extension

Claim: If G is a PRG then G_{poly} is a PRG.

Proof:

PRG Length Extension

Claim: If G is a PRG then G_{poly} is a PRG.

Proof:

Claim: If G is a PRG then for each $i \in \{0, \ldots, \ell - 1\}$, $\quad H_i \overset{c}{\approx} H_{i+1}$.

PRG Length Extension

Claim: If G is a PRG then G_{poly} is a PRG.

Proof:

Claim: If G is a PRG then for each $i \in \{0, \ldots, \ell - 1\}$, $H_i \overset{c}{\approx} H_{i+1}$.

Therefore, from the hybrid lemma

If for all efficient adversaries A $\left| \Pr_{k \overset{\$}{\leftarrow} \{0,1\}^\lambda} \left[A(1^\lambda, G(k)) = 1 \right] - \Pr_{r \overset{\$}{\leftarrow} \{0,1\}^{\lambda+1}} \left[A(1^\lambda, r) = 1 \right] \right| \leq \mathsf{negl}(\lambda)$

PRG Length Extension

Claim: If G is a PRG then G_{poly} is a PRG.

Proof:

Claim: If G is a PRG then for each $i \in \{0,\ldots,\ell-1\}$, $\quad H_i \overset{c}{\approx} H_{i+1}$.

Therefore, from the hybrid lemma

If for all efficient adversaries A $\quad \left| \Pr_{k \overset{\$}{\leftarrow} \{0,1\}^\lambda} \left[A(1^\lambda, G(k)) = 1\right] - \Pr_{r \overset{\$}{\leftarrow} \{0,1\}^{\lambda+1}} \left[A(1^\lambda, r) = 1\right] \right| \leq \text{negl}(\lambda)$

Then, for all efficient adversaries A $\quad \left| \Pr_{k \overset{\$}{\leftarrow} \{0,1\}^\lambda} \left[A(1^\lambda, G_{\text{poly}}(k)) = 1\right] - \Pr_{r \overset{\$}{\leftarrow} \{0,1\}^{\ell(\lambda)}} \left[A(1^\lambda, r) = 1\right] \right| \leq$

PRG Length Extension

Claim: If G is a PRG then G_{poly} is a PRG.

Proof:

Claim: If G is a PRG then for each $i \in \{0,\ldots,\ell-1\}$, $\quad H_i \overset{c}{\approx} H_{i+1}$.

Therefore, from the hybrid lemma

If for all efficient adversaries A $\quad \left| \Pr_{k \overset{\$}{\leftarrow} \{0,1\}^\lambda}\left[A(1^\lambda, G(k)) = 1\right] - \Pr_{r \overset{\$}{\leftarrow} \{0,1\}^{\lambda+1}}\left[A(1^\lambda, r) = 1\right] \right| \leq \text{negl}(\lambda)$

Then, for all efficient adversaries A $\quad \left| \Pr_{k \overset{\$}{\leftarrow} \{0,1\}^\lambda}\left[A(1^\lambda, G_{\text{poly}}(k)) = 1\right] - \Pr_{r \overset{\$}{\leftarrow} \{0,1\}^{\ell(\lambda)}}\left[A(1^\lambda, r) = 1\right] \right| \leq \ell(\lambda) \cdot \text{negl}(\lambda)$

PRG Length Extension

Claim: If G is a PRG then G_{poly} is a PRG.

Proof:

Claim: If G is a PRG then for each $i \in \{0,\ldots,\ell-1\}$, $\quad H_i \overset{c}{\approx} H_{i+1}$.

Therefore, from the hybrid lemma

If for all efficient adversaries A $\quad \left| \Pr_{k \xleftarrow{\$} \{0,1\}^\lambda} \left[A(1^\lambda, G(k)) = 1\right] - \Pr_{r \xleftarrow{\$} \{0,1\}^{\lambda+1}} \left[A(1^\lambda, r) = 1\right] \right| \leq \text{negl}(\lambda)$

Then, for all efficient adversaries A $\quad \left| \Pr_{k \xleftarrow{\$} \{0,1\}^\lambda} \left[A(1^\lambda, G_{\text{poly}}(k)) = 1\right] - \Pr_{r \xleftarrow{\$} \{0,1\}^{\ell(\lambda)}} \left[A(1^\lambda, r) = 1\right] \right| \leq \ell(\lambda) \cdot \text{negl}(\lambda)$

- **Concrete Security:** If G is (T, ϵ)-secure, then G_{poly} is $(T, \ell(\lambda) \cdot \epsilon)$ secure.

PRG Length Extension

Claim: If G is a PRG then G_{poly} is a PRG.

Proof:

Claim: If G is a PRG then for each $i \in \{0,\ldots,\ell-1\}$, $H_i \overset{c}{\approx} H_{i+1}$.

Therefore, from the hybrid lemma

If for all efficient adversaries A $\left|\Pr_{k \xleftarrow{\$} \{0,1\}^\lambda}\left[A(1^\lambda, G(k)) = 1\right] - \Pr_{r \xleftarrow{\$} \{0,1\}^{\lambda+1}}\left[A(1^\lambda, r) = 1\right]\right| \leq \text{negl}(\lambda)$

Then, for all efficient adversaries A $\left|\Pr_{k \xleftarrow{\$} \{0,1\}^\lambda}\left[A(1^\lambda, G_{\text{poly}}(k)) = 1\right] - \Pr_{r \xleftarrow{\$} \{0,1\}^{\ell(\lambda)}}\left[A(1^\lambda, r) = 1\right]\right| \leq \ell(\lambda) \cdot \text{negl}(\lambda)$

- **Concrete Security:** If G is (T, ϵ)-secure, then G_{poly} is $(T, \ell(\lambda) \cdot \epsilon)$ secure.
  - **Example:** If $\epsilon = 2^{-40}$ then encrypting a 131 KB message using G_{poly} leads to $\epsilon_{\text{poly}} = 2^{-20}$.

PRG Length Extension

Claim: If G is a PRG the

Proof:

Claim: If G is

Therefore, from the

If for all effic

Then, for all

Probability (ϵ)	Event
2^{-10}	Full house in 5-card poker
2^{-20}	Royal flush in 5-card poker
2^{-28}	Winning this week's Powerball jackpot
2^{-40}	Royal flush in two consecutive poker games
2^{-60}	Next meteorite that hits Earth lands on this slide

$]\big| \leq \mathsf{negl}(\lambda)$

$, r) = 1]\big| \leq \ell(\lambda) \cdot \mathsf{negl}(\lambda)$

- **Concrete Security:** If G is (T, ϵ)-secure, then G_{poly} is $(T, \ell(\lambda) \cdot \epsilon)$ secure.
  - **Example:** If $\epsilon = 2^{-40}$ then encrypting a 131 KB message using G_{poly} leads to $\epsilon_{\text{poly}} = 2^{-20}$.

PRG Length Extension

Claim: If G is a PRG the

Proof:

Claim: If G i

Therefore, from the

If for all effic

Then, for all

Probability (ϵ)	Event
2^{-10}	Full house in 5-card poker
2^{-20}	Royal flush in 5-card poker
2^{-28}	Winning this week's Powerball jackpot
2^{-40}	Royal flush in two consecutive poker games
2^{-60}	Next meteorite that hits Earth lands on this slide

$] \Big| \leq \text{negl}(\lambda)$

$, r) = 1] \Big| \leq \ell(\lambda) \cdot \text{negl}(\lambda)$

- **Concrete Security:** If G is (T, ϵ)-secure, then G_{poly} is $(T, \ell(\lambda) \cdot \epsilon)$ secure.
  - **Example:** If $\epsilon = 2^{-40}$ then encrypting a 131 KB message using G_{poly} leads to $\epsilon_{\text{poly}} = 2^{-20}$.
- In this course, we will not worry about the security loss or the running time of the reduction.

PRG Length Extension

Claim: If G is a PRG the

Proof:

Claim: If G i

Therefore, from the

If for all effic ... $]\Big| \leq \text{negl}(\lambda)$

Then, for all ... $^{\lambda}, r) = 1]\Big| \leq \ell(\lambda) \cdot \text{negl}(\lambda)$

Probability (ϵ)	**Event**
2^{-10}	Full house in 5-card poker
2^{-20}	Royal flush in 5-card poker
2^{-28}	Winning this week's Powerball jackpot
2^{-40}	Royal flush in two consecutive poker games
2^{-60}	Next meteorite that hits Earth lands on this slide

- **Concrete Security:** If G is (T, ϵ)-secure, then G_{poly} is $(T, \ell(\lambda) \cdot \epsilon)$ secure.
  - **Example:** If $\epsilon = 2^{-40}$ then encrypting a 131 KB message using G_{poly} leads to $\epsilon_{\text{poly}} = 2^{-20}$.
- In this course, we will not worry about the security loss or the running time of the reduction.
  - This is fine for an asymptotic approach.

PRG Length Extension

Claim: If G is a PRG the

Proof:

Claim: If G i

Therefore, from the

If for all effic

$] \Big| \leq \text{negl}(\lambda)$

Then, for all

$^{\lambda}, r) = 1] \Big| \leq \ell(\lambda) \cdot \text{negl}(\lambda)$

Probability (ϵ)	**Event**
2^{-10}	Full house in 5-card poker
2^{-20}	Royal flush in 5-card poker
2^{-28}	Winning this week's Powerball jackpot
2^{-40}	Royal flush in two consecutive poker games
2^{-60}	Next meteorite that hits Earth lands on this slide

- **Concrete Security:** If G is (T, ϵ)-secure, then G_{poly} is $(T, \ell(\lambda) \cdot \epsilon)$ secure.
  - **Example:** If $\epsilon = 2^{-40}$ then encrypting a 131 KB message using G_{poly} leads to $\epsilon_{\text{poly}} = 2^{-20}$.
- In this course, we will not worry about the security loss or the running time of the reduction.
  - This is fine for an asymptotic approach.
  - However, the loss matters in practice for concrete security.

Multi-Message Security

One-Time Computational Security

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

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

Multi-Message Security

One-Time Computational Security

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

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

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 $\forall m_0, m_1 \in \{0,1\}^\ell$

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

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

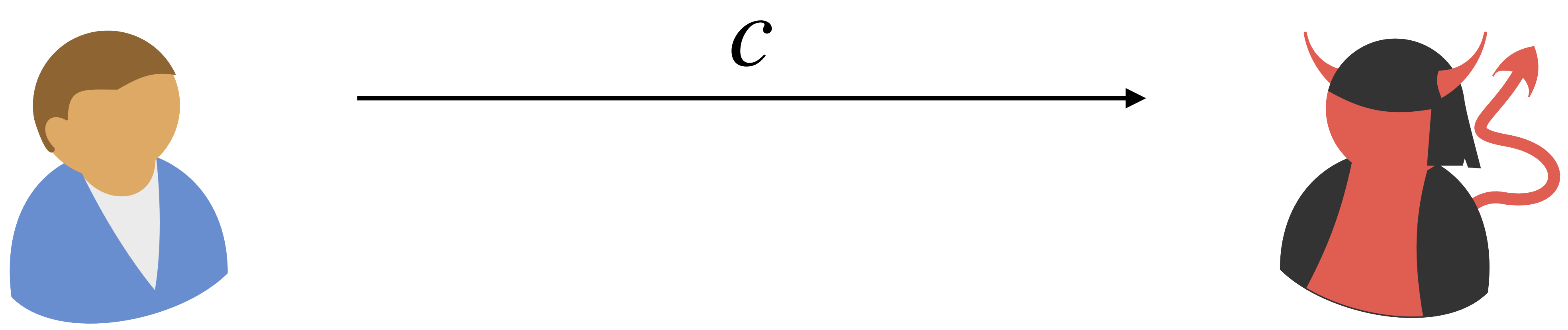
Multi-Message Security

One-Time Computational Security

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

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

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



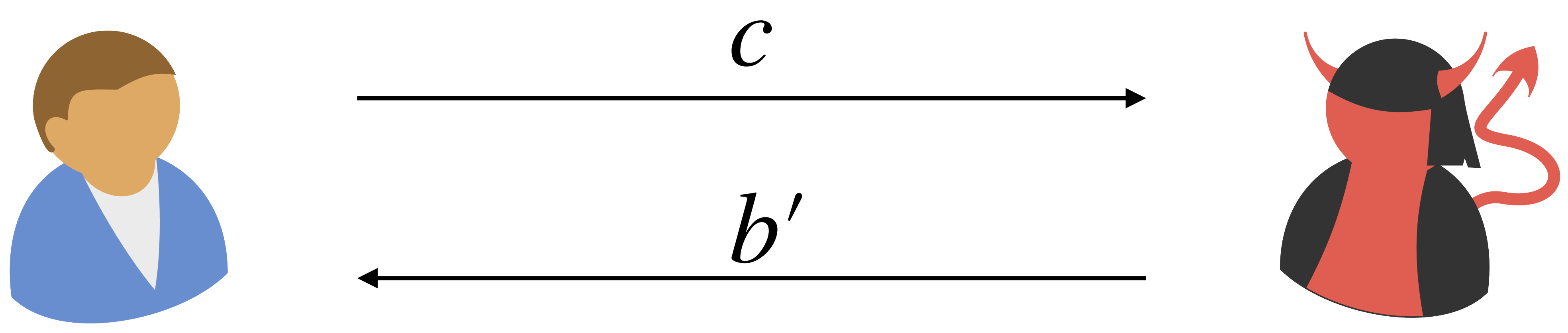
Multi-Message Security

One-Time Computational Security

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

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

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



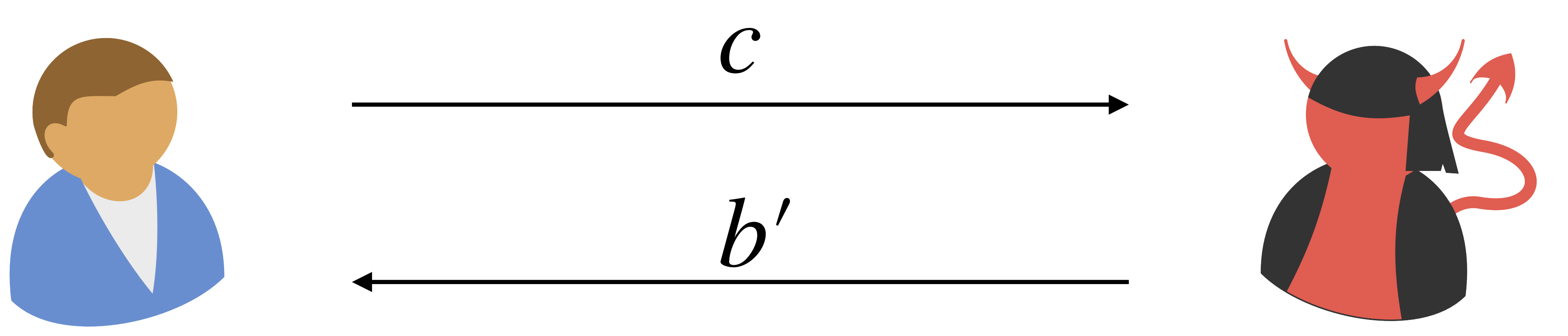
Multi-Message Security

One-Time Computational Security

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

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

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



Wins if $b' = b$

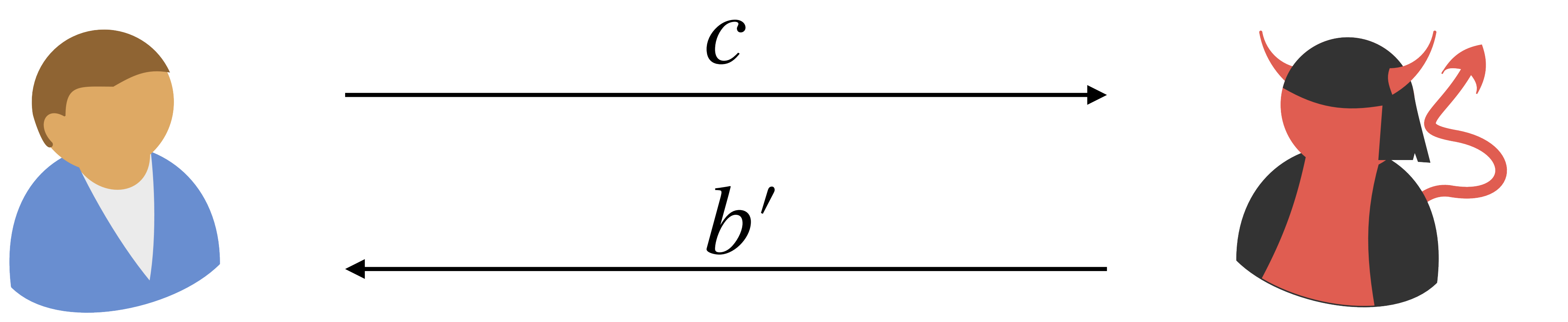
Multi-Message Security

One-Time Computational Security

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

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

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



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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$\{c_i\}_{i=1}^{q(\lambda)}$ →

Multi-Message Security

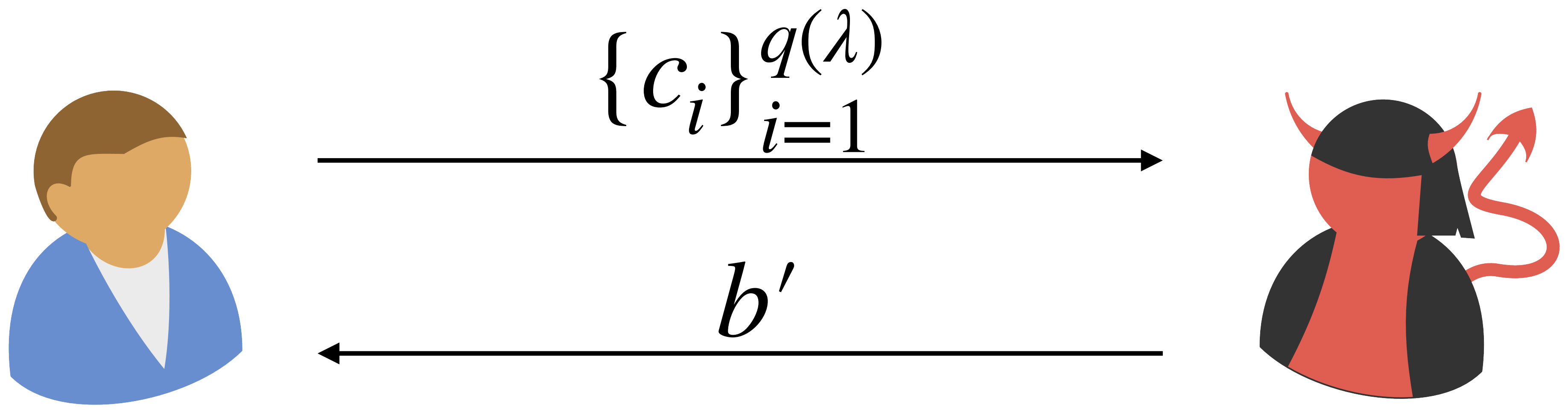
$q(\lambda)$ *pairs* of messages

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

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

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

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

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

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

$\{c_i\}_{i=1}^{q(\lambda)}$ →

← b'

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

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

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

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

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

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

$\{c_i\}_{i=1}^{q(\lambda)}$ →

← 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?

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

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?

$G(s)[0]$

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

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?

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

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

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?

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

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

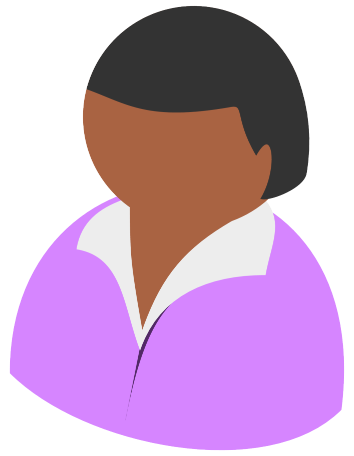
Stateful Multi-Message Secure Encryption

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

What if we use one chunk at a time?

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

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



Stateful Multi-Message Secure Encryption

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

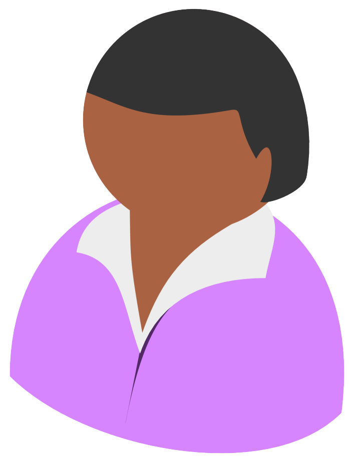
What if we use one chunk at a time?

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

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

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?

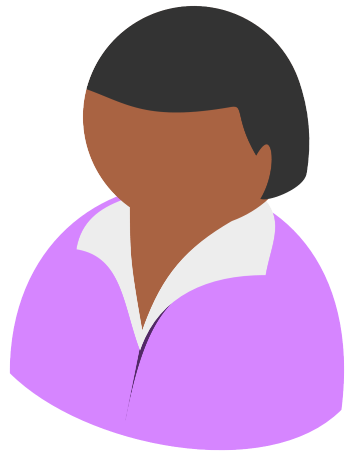
$G(s)[0]$ $G(s)[1]$ $G(s)[2]$

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

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?

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

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

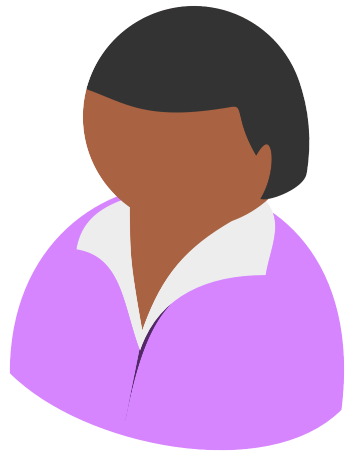
k

$i = 0$

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

$\longrightarrow$

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?

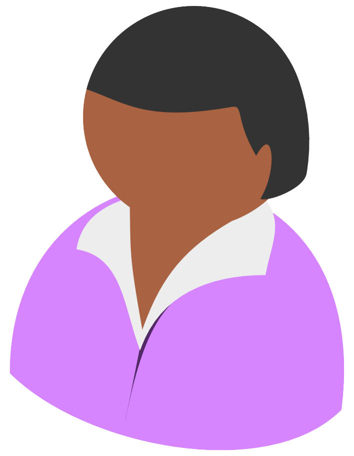
$G(s)[0]$ $G(s)[1]$ $G(s)[2]$

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

k

$i = 0$

$c_0 = m_0 \oplus G(k)[0]$ →



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?

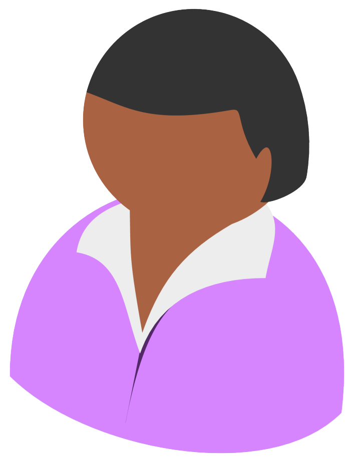
$G(s)[0]$ $G(s)[1]$ $G(s)[2]$

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

k

$i = 1$

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



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?

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

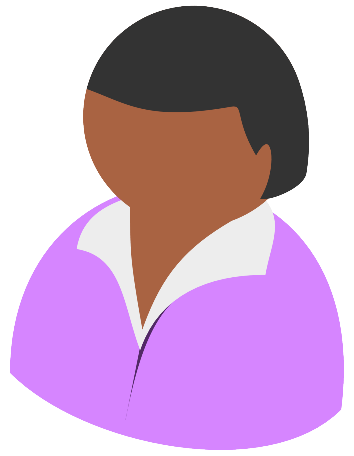
$$G(s) = 00010101 \ldots 11100010 \ldots 01011010 \ldots \rightarrow \text{poly}(\lambda)$$

k

$i = 1$

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

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



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

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?

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

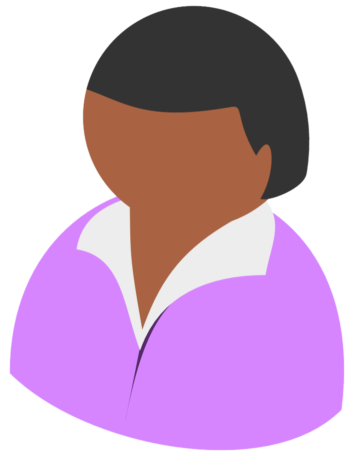
$$G(s) = 00010101 \ldots 11100010 \ldots 01011010 \ldots \rightarrow \text{poly}(\lambda)$$

k

$i = 1$

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

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



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?

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

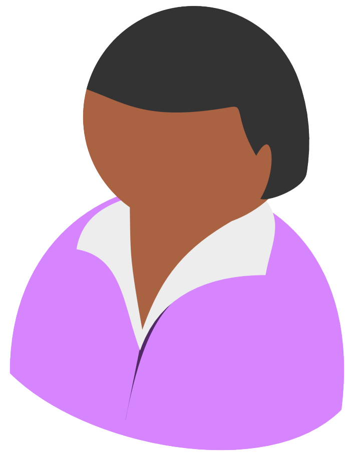
$$G(s) = 00010101 \ldots 11100010 \ldots 01011010 \ldots \rightarrow \text{poly}(\lambda)$$

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?

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

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

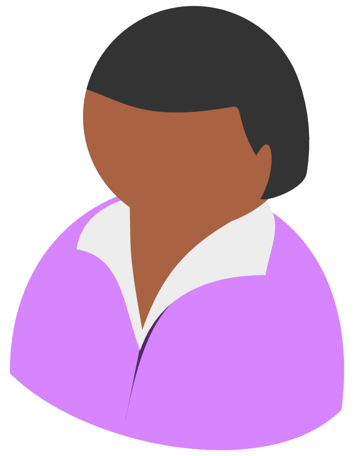
Losing it!

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?

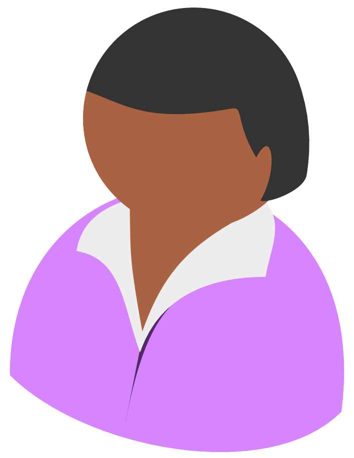
k

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

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

$c_0 = (m_0 \oplus G(k)[i_0], i_0)$ →

$c_1 = (m_1 \oplus G(k)[i_1], i_1)$ →



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 $i_0 = i_1$?

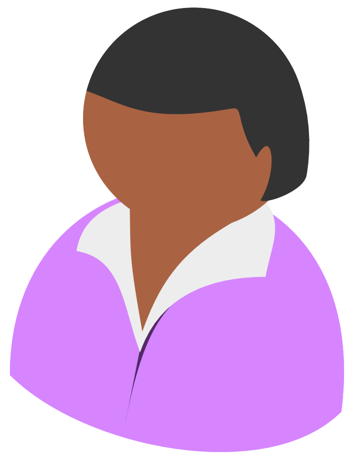
k

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

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

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

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



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 $i_0 = i_1$?

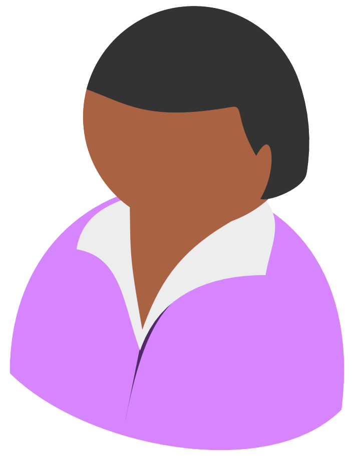
k

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

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

$c_0 = (m_0 \oplus G(k)[i_0], i_0)$ →

$c_1 = (m_1 \oplus G(k)[i_1], i_1)$ →



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!

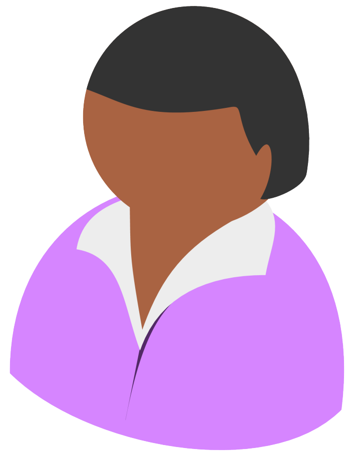
k

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

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

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

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



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?

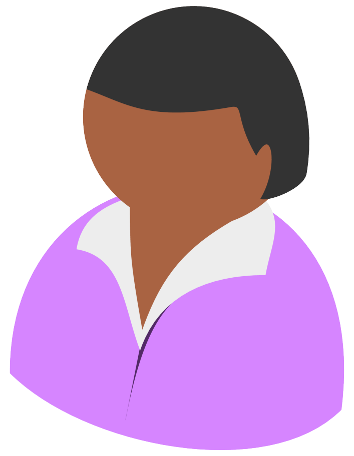
k

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

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

$c_0 = (m_0 \oplus G(k)[i_0], i_0)$ →

$c_1 = (m_1 \oplus G(k)[i_1], i_1)$ →



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

$\cdots$

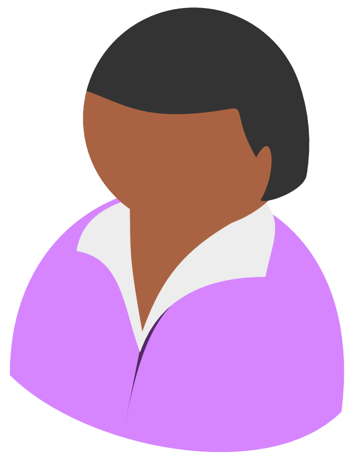
Stateless Multi-Message Secure Encryption

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

$\cdots$



Stateless Multi-Message Secure Encryption

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

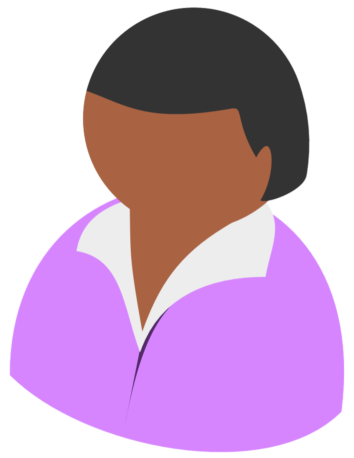
$F =$

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?

$F =$

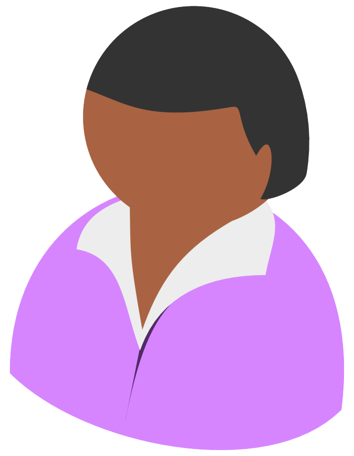
x	r
000…000	11001010
000…001	10011111
000…010	10010010
000…011	10111111

$\cdots$

F

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

F



Stateless Multi-Message Secure Encryption

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

$F =$

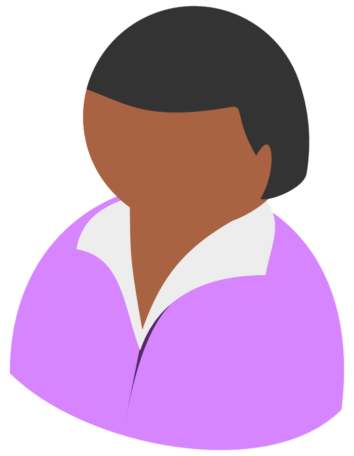
x	r
000…000	11001010
000…001	10011111
000…010	10010010
000…011	10111111

$\cdots$

F

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

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



F

Stateless Multi-Message Secure Encryption

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

$F =$

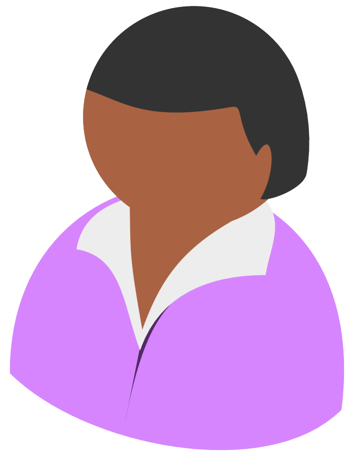
x	r
000…000	11001010
000…001	10011111
000…010	10010010
000…011	10111111

$\cdots$

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

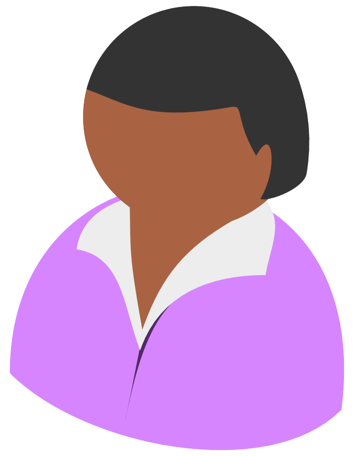
$\cdots$

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

$\cdots$

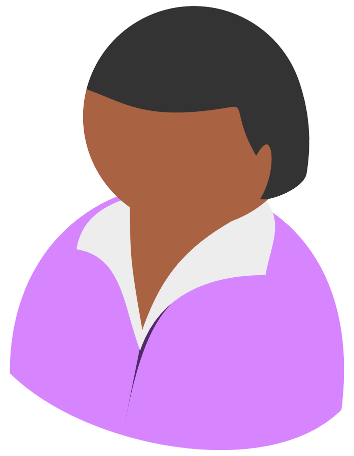
F is a *random function*

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

$m_0 = c_0 \oplus F[i_0]$