Pseudorandomness (II)

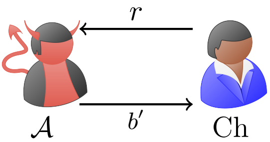
601.642/442: Modern Cryptography

Fall 2020

Recap: Pseudorandom Generator

- A deterministic function $G : \{0,1\}^n \to \{0,1\}^{\ell(n)}$ is a **PRG** if its output distribution is **computationally indistinguishable** from the uniform distribution.
- In other words, the **advantage** of $\mathcal{A}$ in distinguishing between the uniform distribution and the output distribution of G is **negligible**.

Game Based Definition of PRG



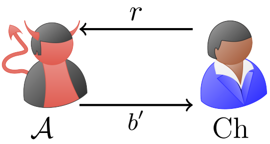
$b \xleftarrow{\$} \{0,1\}$

if $b = 0$: $r \xleftarrow{\$} \{0,1\}^{\ell(n)}$

if $b = 1$: $s \xleftarrow{\$} \{0,1\}^{n}$
$r := G(s)$

$\boxed{\mathcal{A} \text{ wins if } b' = b}$

Game Based Definition of PRG



$$b \xleftarrow{\$} \{0,1\}$$
$$\text{if } b = 0\text{: } r \xleftarrow{\$} \{0,1\}^{\ell(n)}$$
$$\text{if } b = 1\text{: } s \xleftarrow{\$} \{0,1\}^{n}$$
$$r := G(s)$$

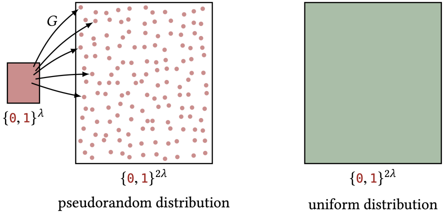
$$\boxed{\mathcal{A} \text{ wins if } b' = b}$$

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

$$\Big| \Pr[b' = 1 | b = 1] - \Pr[b' = 1 | b = 0] \Big| \leqslant \nu(n)$$

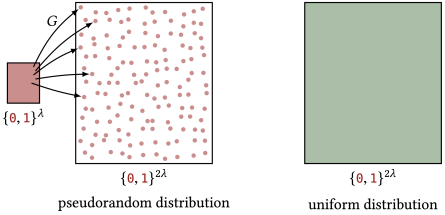
$$\Big| \Pr[\mathcal{A}(1^n, r) = 1 |\ s \xleftarrow{\$} \{0,1\}^n, r := G(s)] - \Pr[\mathcal{A}(1^n, r) = 1 | r \xleftarrow{\$} \{0,1\}^{\ell(n)}] \Big| \leqslant \nu(n)$$

Illustrating PRG $G : \{0,1\}^\lambda \to \{0,1\}^{2\lambda}$



- From a relative perspective, the PRG's output distribution is tiny. Out of the $2^{2\lambda}$ strings in $\{0,1\}^{2\lambda}$, only 2^λ are possible outputs of G. These strings make up $2^\lambda / 2^{2\lambda} = 1/2^\lambda$ fraction of $\{0,1\}^{2\lambda}$ — a **negligible fraction!**

Illustrating PRG $G : \{0,1\}^\lambda \to \{0,1\}^{2\lambda}$



- From a relative perspective, the PRG's output distribution is tiny. Out of the $2^{2\lambda}$ strings in $\{0,1\}^{2\lambda}$, only 2^λ are possible outputs of G. These strings make up $2^\lambda/2^{2\lambda} = 1/2^\lambda$ fraction of $\{0,1\}^{2\lambda}$ — a **negligible fraction!**
- From an absolute perspective, the PRG's output distribution is huge. There are 2^λ possible outputs of G, which is an **exponential amount!** This is large enough that an efficient adversary cannot distinguish it from the set $\{0,1\}^{2\lambda}$.

Pseudorandom OTP

Pseudorandom One-Time Pad

Let n be the security parameter and $\ell(\cdot)$ be a polynomial. Let $G : \{0,1\}^n \rightarrow \{0,1\}^{\ell(n)}$ be a PRG, and let the message space and ciphertext space be $\{0,1\}^{\ell(n)}$.

- $\mathsf{KeyGen}(1^n) := k \leftarrow_{\$} \{0,1\}^n$
- $\mathsf{Enc}(k, m) := c = G(k) \oplus m$
- $\mathsf{Dec}(k, c) := m = G(k) \oplus c$

One-Time Computational Security

We consider the following computational notion of security.

One-Time Computational Security

We say that an encryption scheme is one-time ~~perfectly~~ **computationally** secure if $\forall m_0, m_1 \in \mathcal{M}$ chosen by an adversary, the following distributions are ~~identical~~ **computationally indistinguishable**:

1. $\mathcal{D}_1 := \{c := \mathsf{Enc}(k, m_0); k \leftarrow \mathsf{KeyGen}(1^n)\}$
2. $\mathcal{D}_2 := \{c := \mathsf{Enc}(k, m_1); k \leftarrow \mathsf{KeyGen}(1^n)\}$

Security of Pseudorandom OTP

Lemma

Pseudorandom OTP satisfies one-time computational security.

Proof. We need to show that $\forall m_0, m_1 \in \{0,1\}^{\ell(n)}$ chosen by an adversary, the following two distrbutions are computationally indistinguishable:

1. $\mathcal{D}_1 := \{c := m_0 \oplus G(k);\ k \leftarrow \{0,1\}^n\}$
2. $\mathcal{D}_2 := \{c := m_1 \oplus G(k);\ k \leftarrow \{0,1\}^n\}$

Security of Pseudorandom OTP

Lemma

Pseudorandom OTP satisfies one-time computational security.

Proof. We need to show that $\forall m_0, m_1 \in \{0,1\}^{\ell(n)}$ chosen by an adversary, the following two distrbutions are computationally indistinguishable:

1. $\mathcal{D}_1 := \{c := m_0 \oplus G(k);\ k \leftarrow \{0,1\}^n\}$
2. $\mathcal{D}_2 := \{c := m_1 \oplus G(k);\ k \leftarrow \{0,1\}^n\}$

Consider the following hybrids:

1. $\mathcal{H}_1 := \left\{c := m_0 \oplus G(k);\ k \xleftarrow{\$} \{0,1\}^n\right\}$

Security of Pseudorandom OTP

Lemma

Pseudorandom OTP satisfies one-time computational security.

Proof. We need to show that $\forall m_0, m_1 \in \{0,1\}^{\ell(n)}$ chosen by an adversary, the following two distrbutions are computationally indistinguishable:

1. $\mathcal{D}_1 := \{c := m_0 \oplus G(k);\ k \leftarrow \{0,1\}^n\}$
2. $\mathcal{D}_2 := \{c := m_1 \oplus G(k);\ k \leftarrow \{0,1\}^n\}$

Consider the following hybrids:

1. $\mathcal{H}_1 := \left\{c := m_0 \oplus G(k);\ k \xleftarrow{\$} \{0,1\}^n\right\}$
2. $\mathcal{H}_2 := \left\{c := m_0 \oplus r;\ r \xleftarrow{\$} \{0,1\}^{\ell(n)}\right\}$

Security of Pseudorandom OTP

Lemma

Pseudorandom OTP satisfies one-time computational security.

Proof. We need to show that $\forall m_0, m_1 \in \{0,1\}^{\ell(n)}$ chosen by an adversary, the following two distrbutions are computationally indistinguishable:

1. $\mathcal{D}_1 := \{c := m_0 \oplus G(k);\ k \leftarrow \{0,1\}^n\}$
2. $\mathcal{D}_2 := \{c := m_1 \oplus G(k);\ k \leftarrow \{0,1\}^n\}$

Consider the following hybrids:

1. $\mathcal{H}_1 := \left\{c := m_0 \oplus G(k);\ k \xleftarrow{\$} \{0,1\}^n\right\}$
2. $\mathcal{H}_2 := \left\{c := m_0 \oplus r;\ r \xleftarrow{\$} \{0,1\}^{\ell(n)}\right\}$
3. $\mathcal{H}_3 := \left\{c := m_1 \oplus r;\ r \xleftarrow{\$} \{0,1\}^{\ell(n)}\right\}$

Security of Pseudorandom OTP

Lemma

Pseudorandom OTP satisfies one-time computational security.

Proof. We need to show that $\forall m_0, m_1 \in \{0,1\}^{\ell(n)}$ chosen by an adversary, the following two distrbutions are computationally indistinguishable:

1. $\mathcal{D}_1 := \{c := m_0 \oplus G(k);\ k \leftarrow \{0,1\}^n\}$
2. $\mathcal{D}_2 := \{c := m_1 \oplus G(k);\ k \leftarrow \{0,1\}^n\}$

Consider the following hybrids:

1. $\mathcal{H}_1 := \left\{c := m_0 \oplus G(k);\ k \xleftarrow{\$} \{0,1\}^n\right\}$
2. $\mathcal{H}_2 := \left\{c := m_0 \oplus r;\ r \xleftarrow{\$} \{0,1\}^{\ell(n)}\right\}$
3. $\mathcal{H}_3 := \left\{c := m_1 \oplus r;\ r \xleftarrow{\$} \{0,1\}^{\ell(n)}\right\}$
4. $\mathcal{H}_4 := \left\{c := m_1 \oplus G(k);\ k \xleftarrow{\$} \{0,1\}^n\right\}$

Security of Pseudorandom OTP

1. $\mathcal{H}_1 := \left\{ c := m_0 \oplus G(k);\ k \xleftarrow{\$} \{0,1\}^n \right\}$
2. $\mathcal{H}_2 := \left\{ c := m_0 \oplus r;\ r \xleftarrow{\$} \{0,1\}^{\ell(n)} \right\}$
3. $\mathcal{H}_3 := \left\{ c := m_1 \oplus r;\ r \xleftarrow{\$} \{0,1\}^{\ell(n)} \right\}$
4. $\mathcal{H}_4 := \left\{ c := m_1 \oplus G(k);\ k \xleftarrow{\$} \{0,1\}^n \right\}$

Security of Pseudorandom OTP

1. $\mathcal{H}_1 := \left\{c := m_0 \oplus G(k);\ k \xleftarrow{\$} \{0,1\}^n\right\}$
2. $\mathcal{H}_2 := \left\{c := m_0 \oplus r;\ r \xleftarrow{\$} \{0,1\}^{\ell(n)}\right\}$
3. $\mathcal{H}_3 := \left\{c := m_1 \oplus r;\ r \xleftarrow{\$} \{0,1\}^{\ell(n)}\right\}$
4. $\mathcal{H}_4 := \left\{c := m_1 \oplus G(k);\ k \xleftarrow{\$} \{0,1\}^n\right\}$

- $\mathcal{H}_1 \approx_c \mathcal{H}_2$: From the security of PRG, we know that

$$\{G(k);\ k \xleftarrow{\$} \{0,1\}^n\} \approx_c \{r;\ r \xleftarrow{\$} \{0,1\}^{\ell(n)}\}$$

From closure property of computational indistinguishability, we get

$$\{m_0 \oplus G(k);\ k \xleftarrow{\$} \{0,1\}^n\} \approx_c \{m_0 \oplus r;\ r \xleftarrow{\$} \{0,1\}^{\ell(n)}\}$$

Security of Pseudorandom OTP

1. $\mathcal{H}_1 := \left\{ c := m_0 \oplus G(k);\ k \xleftarrow{\$} \{0,1\}^n \right\}$
2. $\mathcal{H}_2 := \left\{ c := m_0 \oplus r;\ r \xleftarrow{\$} \{0,1\}^{\ell(n)} \right\}$
3. $\mathcal{H}_3 := \left\{ c := m_1 \oplus r;\ r \xleftarrow{\$} \{0,1\}^{\ell(n)} \right\}$
4. $\mathcal{H}_4 := \left\{ c := m_1 \oplus G(k);\ k \xleftarrow{\$} \{0,1\}^n \right\}$

Security of Pseudorandom OTP

1. $\mathcal{H}_1 := \left\{ c := m_0 \oplus G(k);\ k \xleftarrow{\$} \{0,1\}^n \right\}$
2. $\mathcal{H}_2 := \left\{ c := m_0 \oplus r;\ r \xleftarrow{\$} \{0,1\}^{\ell(n)} \right\}$
3. $\mathcal{H}_3 := \left\{ c := m_1 \oplus r;\ r \xleftarrow{\$} \{0,1\}^{\ell(n)} \right\}$
4. $\mathcal{H}_4 := \left\{ c := m_1 \oplus G(k);\ k \xleftarrow{\$} \{0,1\}^n \right\}$

- $\mathcal{H}_2 \equiv \mathcal{H}_3$: $\mathcal{H}_2$ is an OTP encryption of m_0 and $\mathcal{H}_3$ is an OTP encryption of m_1. Therefore, $\mathcal{H}_2$ and $\mathcal{H}_3$ are identical because of the one-time perfect security of OTP.

Security of Pseudorandom OTP

1. $\mathcal{H}_1 := \left\{ c := m_0 \oplus G(k);\ k \xleftarrow{\$} \{0,1\}^n \right\}$
2. $\mathcal{H}_2 := \left\{ c := m_0 \oplus r;\ r \xleftarrow{\$} \{0,1\}^{\ell(n)} \right\}$
3. $\mathcal{H}_3 := \left\{ c := m_1 \oplus r;\ r \xleftarrow{\$} \{0,1\}^{\ell(n)} \right\}$
4. $\mathcal{H}_4 := \left\{ c := m_1 \oplus G(k);\ k \xleftarrow{\$} \{0,1\}^n \right\}$

- $\mathcal{H}_2 \equiv \mathcal{H}_3$: $\mathcal{H}_2$ is an OTP encryption of m_0 and $\mathcal{H}_3$ is an OTP encryption of m_1. Therefore, $\mathcal{H}_2$ and $\mathcal{H}_3$ are identical because of the one-time perfect security of OTP.
- $\mathcal{H}_3 \approx_c \mathcal{H}_4$: Similar to $\mathcal{H}_1 \approx_c \mathcal{H}_2$.

Security of Pseudorandom OTP

1. $\mathcal{H}_1 := \left\{c := m_0 \oplus G(k);\ k \xleftarrow{\$} \{0,1\}^n\right\}$
2. $\mathcal{H}_2 := \left\{c := m_0 \oplus r;\ r \xleftarrow{\$} \{0,1\}^{\ell(n)}\right\}$
3. $\mathcal{H}_3 := \left\{c := m_1 \oplus r;\ r \xleftarrow{\$} \{0,1\}^{\ell(n)}\right\}$
4. $\mathcal{H}_4 := \left\{c := m_1 \oplus G(k);\ k \xleftarrow{\$} \{0,1\}^n\right\}$

- $\mathcal{H}_2 \equiv \mathcal{H}_3$: $\mathcal{H}_2$ is an OTP encryption of m_0 and $\mathcal{H}_3$ is an OTP encryption of m_1. Therefore, $\mathcal{H}_2$ and $\mathcal{H}_3$ are identical because of the one-time perfect security of OTP.
- $\mathcal{H}_3 \approx_c \mathcal{H}_4$: Similar to $\mathcal{H}_1 \approx_c \mathcal{H}_2$.

$$\boxed{\mathcal{H}_1 \approx_c \mathcal{H}_2 \equiv \mathcal{H}_3 \approx_c \mathcal{H}_4}$$

By hybrid lemma, $\mathcal{H}_1$ is computationally indistinguishable to $\mathcal{H}_4$.

How to construct PRGs?

One-bit stretch PRG $\implies$ Poly-bit stretch PRG

- We will now show that once you can construct a PRG with tiny stretch (even 1 bit), you can also construct arbitrary polynomial stretch PRG.

One-bit stretch PRG $\implies$ Poly-bit stretch PRG

- We will now show that once you can construct a PRG with tiny stretch (even 1 bit), you can also construct arbitrary polynomial stretch PRG.
- Intuition: Iterate the one-bit stretch PRG poly times

One-bit stretch PRG $\implies$ Poly-bit stretch PRG

- We will now show that once you can construct a PRG with tiny stretch (even 1 bit), you can also construct arbitrary polynomial stretch PRG.
- Intuition: Iterate the one-bit stretch PRG poly times

One-bit stretch PRG $\implies$ Poly-bit stretch PRG

- We will now show that once you can construct a PRG with tiny stretch (even 1 bit), you can also construct arbitrary polynomial stretch PRG.
- Intuition: Iterate the one-bit stretch PRG poly times

Construction of $G_{poly} : \{0,1\}^n \to \{0,1\}^{\ell(n)}$

Let $G : \{0,1\}^n \to \{0,1\}^{n+1}$ be a one-bit stretch PRG.

$$\begin{aligned} s &= x_0 \\ G(x_0) &= x_1 \| b_1 \\ &\vdots \\ G(x_{\ell(n)-1}) &= x_{\ell(n)} \| b_{\ell(n)} \end{aligned}$$

$G_{poly}(s) := b_1 \ldots b_{\ell(n)}$

Pseudorandomnes of G_{poly}

- We want to show $\left\{G_{poly}(s);\ s \xleftarrow{\$} \{0,1\}^n\right\} \approx_c \left\{r \xleftarrow{\$} \{0,1\}^{\ell(n)}\right\}$

Pseudorandomnes of G_{poly}

- We want to show $\left\{G_{poly}(s);\ s \xleftarrow{\$} \{0,1\}^n\right\} \approx_c \left\{r \xleftarrow{\$} \{0,1\}^{\ell(n)}\right\}$
- Consider the following hybrid experiments:

Pseudorandomnes of G_{poly}

- We want to show $\left\{G_{poly}(s);\ s \xleftarrow{\$} \{0,1\}^n\right\} \approx_c \left\{r \xleftarrow{\$} \{0,1\}^{\ell(n)}\right\}$
- Consider the following hybrid experiments:

Pseudorandomnes of G_{poly}

- We want to show $\left\{G_{poly}(s);\ s \xleftarrow{\$} \{0,1\}^n\right\} \approx_c \left\{r \xleftarrow{\$} \{0,1\}^{\ell(n)}\right\}$
- Consider the following hybrid experiments:

Experiment $\mathcal{H}_1$	Experiment $\mathcal{H}_2$	Experiment $\mathcal{H}_{\ell(n)}$
$s = x_0$	$s = x_0$	$s = X_0$
$G(x_0) = x_1 \| b_1$	$s_1 \| u_1 = x_1 \| u_1$	$s_1 \| u_1 = x_1 \| u_1$
$G(x_1) = x_2 \| b_2$	$G(x_1) = x_2 \| b_2$	$s_2 \| u_2 = x_2 \| u_2$
$\ldots$	$\ldots$	$\ldots$
$\ldots$	$\ldots$	$\ldots$
$G(x_{\ell(n)-1}) = x_{\ell(n)} \| b_{\ell(n)}$	$G(X_{\ell(n)-1}) = x_{\ell(n)} \| b_{\ell(n)}$	$s_{\ell(n)} \| u_{\ell(n)} = x_{\ell(n)} \| u_{\ell(n)}$
$\boxed{\text{Output } G(s) := b_1 b_2 \ldots b_{\ell(n)}}$	$\boxed{\text{Output } G(s) := u_1 b_2 \ldots b_{\ell(n)}}$	$\boxed{\text{Output } G(s) := u_1 u_2 \ldots u_{\ell(n)}}$

Pseudorandomnes of G_{poly}

- We want to show $\left\{G_{poly}(s);\ s \xleftarrow{\$} \{0,1\}^n\right\} \approx_c \left\{r \xleftarrow{\$} \{0,1\}^{\ell(n)}\right\}$
- Consider the following hybrid experiments:

<u>Experiment $\mathcal{H}_1$</u>

$$\begin{aligned} s &= x_0 \\ G(x_0) &= x_1 || b_1 \\ G(x_1) &= x_2 || b_2 \\ &\dots \\ &\dots \\ G(x_{\ell(n)-1}) &= x_{\ell(n)} || b_{\ell(n)} \end{aligned}$$

$$\boxed{\text{Output } G(s) := b_1 b_2 \dots b_{\ell(n)}}$$

<u>Experiment $\mathcal{H}_2$</u>

$$\begin{aligned} s &= x_0 \\ s_1 || u_1 &= x_1 || u_1 \\ G(x_1) &= x_2 || b_2 \\ &\dots \\ &\dots \\ G(X_{\ell(n)-1}) &= x_{\ell(n)} || b_{\ell(n)} \end{aligned}$$

$$\boxed{\text{Output } G(s) := u_1 b_2 \dots b_{\ell(n)}}$$

<u>Experiment $\mathcal{H}_{\ell(n)}$</u>

$$\begin{aligned} s &= X_0 \\ s_1 || u_1 &= x_1 || u_1 \\ s_2 || u_2 &= x_2 || u_2 \\ &\dots \\ &\dots \\ s_{\ell(n)} || u_{\ell(n)} &= x_{\ell(n)} || u_{\ell(n)} \end{aligned}$$

$$\boxed{\text{Output } G(s) := u_1 u_2 \dots u_{\ell(n)}}$$

- In order to show that G_{poly} is a PRG, it suffices to show that $\mathcal{H}_1 \approx_c \mathcal{H}_{\ell(n)}$.

Pseudorandomnes of G_{poly}

Experiment $\mathcal{H}_1$	Experiment $\mathcal{H}_2$	Experiment $\mathcal{H}_{\ell(n)}$
$s = x_0$	$s = x_0$	$s = X_0$
$G(x_0) = x_1 \| b_1$	$s_1 \| u_1 = x_1 \| u_1$	$s_1 \| u_1 = x_1 \| u_1$
$G(x_1) = x_2 \| b_2$	$G(x_1) = x_2 \| b_2$	$s_2 \| u_2 = x_2 \| u_2$
...	...	...
...	...	...
$G(x_{\ell(n)-1}) = x_{\ell(n)} \| b_{\ell(n)}$	$G(X_{\ell(n)-1}) = x_{\ell(n)} \| b_{\ell(n)}$	$s_{\ell(n)} \| u_{\ell(n)} = x_{\ell(n)} \| u_{\ell(n)}$
$\boxed{\text{Output } G(s) := b_1 b_2 \ldots b_{\ell(n)}}$	$\boxed{\text{Output } G(s) := u_1 b_2 \ldots b_{\ell(n)}}$	$\boxed{\text{Output } G(s) := u_1 u_2 \ldots u_{\ell(n)}}$

- $\mathcal{H}_1 \approx_c \mathcal{H}_2$: From the security of PRG, we know that

$$\{G(s); s \xleftarrow{\$} \{0,1\}^n\} \approx_c \{s_1||u_1 \xleftarrow{\$} \{0,1\}^{n+1}\}$$

Indistinguishability of $\mathcal{H}_1$ and $\mathcal{H}_2$ follows from the closure property of computational indistinguishability.

Pseudorandomnes of G_{poly}

Experiment $\mathcal{H}_1$

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

Output $G(s) := b_1 b_2 \ldots b_{\ell(n)}$

Experiment $\mathcal{H}_2$

$$s = x_0$$
$$s_1||u_1 = x_1||u_1$$
$$G(x_1) = x_2||b_2$$
$$\ldots$$
$$\ldots$$
$$G(X_{\ell(n)-1}) = x_{\ell(n)}||b_{\ell(n)}$$

Output $G(s) := u_1 b_2 \ldots b_{\ell(n)}$

Experiment $\mathcal{H}_{\ell(n)}$

$$s = X_0$$
$$s_1||u_1 = x_1||u_1$$
$$s_2||u_2 = x_2||u_2$$
$$\ldots$$
$$\ldots$$
$$s_{\ell(n)}||u_{\ell(n)} = x_{\ell(n)}||u_{\ell(n)}$$

Output $G(s) := u_1 u_2 \ldots u_{\ell(n)}$

- $\mathcal{H}_1 \approx_c \mathcal{H}_2$: From the security of PRG, we know that

$$\{G(s); s \xleftarrow{\$} \{0,1\}^n\} \approx_c \{s_1||u_1 \xleftarrow{\$} \{0,1\}^{n+1}\}$$

  Indistinguishability of $\mathcal{H}_1$ and $\mathcal{H}_2$ follows from the closure property of computational indistinguishability.
- Similarly, $\forall i \in [\ell(n) - 1]$, $\mathcal{H}_i \approx_c \mathcal{H}_{i+1}$.

Pseudorandomnes of G_{poly}

Experiment $\mathcal{H}_1$	Experiment $\mathcal{H}_2$	Experiment $\mathcal{H}_{\ell(n)}$
$s = x_0$	$s = x_0$	$s = X_0$
$G(x_0) = x_1 \| b_1$	$s_1 \| u_1 = x_1 \| u_1$	$s_1 \| u_1 = x_1 \| u_1$
$G(x_1) = x_2 \| b_2$	$G(x_1) = x_2 \| b_2$	$s_2 \| u_2 = x_2 \| u_2$
$\ldots$	$\ldots$	$\ldots$
$\ldots$	$\ldots$	$\ldots$
$G(x_{\ell(n)-1}) = x_{\ell(n)} \| b_{\ell(n)}$	$G(X_{\ell(n)-1}) = x_{\ell(n)} \| b_{\ell(n)}$	$s_{\ell(n)} \| u_{\ell(n)} = x_{\ell(n)} \| u_{\ell(n)}$
Output $G(s) := b_1 b_2 \ldots b_{\ell(n)}$	Output $G(s) := u_1 b_2 \ldots b_{\ell(n)}$	Output $G(s) := u_1 u_2 \ldots u_{\ell(n)}$

- $\mathcal{H}_1 \approx_c \mathcal{H}_2$: From the security of PRG, we know that

  $$\{G(s); s \xleftarrow{\$} \{0,1\}^n\} \approx_c \{s_1||u_1 \xleftarrow{\$} \{0,1\}^{n+1}\}$$

  Indistinguishability of $\mathcal{H}_1$ and $\mathcal{H}_2$ follows from the closure property of computational indistinguishability.
- Similarly, $\forall i \in [\ell(n) - 1]$, $\mathcal{H}_i \approx_c \mathcal{H}_{i+1}$.
- By Hybrid lemma, $\mathcal{H}_1 \approx_c \mathcal{H}_{\ell(n)}$.

Contrapositive Point of View

- So far, we have only considered security proofs in the "forward" direction.
- A more classical (although initially potentially confusing) way is to prove security by arriving at a contradiction.
- First, we establish the following definitions.

Contrapositive Point of View

- So far, we have only considered security proofs in the "forward" direction.
- A more classical (although initially potentially confusing) way is to prove security by arriving at a contradiction.
- First, we establish the following definitions.

Definition (Non-Negligible Functions)

A function $\nu(n)$ is non-negligible if $\exists c$, such that $\forall n_0$, $\exists n > n_0$, $\nu(n) \geqslant \frac{1}{n^c}$.

Contrapositive Point of View

- So far, we have only considered security proofs in the "forward" direction.
- A more classical (although initially potentially confusing) way is to prove security by arriving at a contradiction.
- First, we establish the following definitions.

Definition (Non-Negligible Functions)

A function $\nu(n)$ is non-negligible if $\exists c$, such that $\forall n_0, \exists n > n_0$, $\nu(n) \geqslant \frac{1}{n^c}$.

Lemma (Alternate way to state Hybrid Lemma)

Let $X^1, \ldots, X^m$ be distribution ensembles for $m = \mathsf{poly}(n)$. Suppose there exists a distinguisher/adversary $\mathcal{A}$ that distinguishes between X^1 and X^m with probability μ. Then $\exists i \in [m-1]$, such that $\mathcal{A}$ distinguishes between X^i and X^{i+1} with advantage at least μ/m.

Contrapositive Point of View

- So far, we have proved statements of the following form.

  "If G is a one-bit stretch PRG, then G_{poly} is a poly-bit stretch PRG."

Contrapositive Point of View

- So far, we have proved statements of the following form.

  "If G is a one-bit stretch PRG, then G_{poly} is a poly-bit stretch PRG."

- Let's now think about the **contrapositve** of these statements.

  *"If G_{poly} is a **not** poly-bit stretch PRG, then G is **not** a one-bit stretch PRG."*

Contrapositive Point of View

- So far, we have proved statements of the following form.

  "If G is a one-bit stretch PRG, then G_{poly} is a poly-bit stretch PRG."

- Let's now think about the **contrapositve** of these statements.

  *"If G_{poly} is a **not** poly-bit stretch PRG, then G is **not** a one-bit stretch PRG."*

- If G_{poly} is not a PRG, then there exists a n.u. PPT adversary $\mathcal{A}$ who can distinguish between its output on a random input and a uniformly sampled string with some **non-negligible** advantage μ.

Contrapositive Point of View

- So far, we have proved statements of the following form.

  "If G is a one-bit stretch PRG, then G_{poly} is a poly-bit stretch PRG."

- Let's now think about the **contrapositve** of these statements.

  *"If G_{poly} is a **not** poly-bit stretch PRG, then G is **not** a one-bit stretch PRG."*

- If G_{poly} is not a PRG, then there exists a n.u. PPT adversary $\mathcal{A}$ who can distinguish between its output on a random input and a uniformly sampled string with some **non-negligible** advantage μ.

Contrapositive Point of View

- So far, we have proved statements of the following form.

  "If G is a one-bit stretch PRG, then G_{poly} is a poly-bit stretch PRG."

- Let's now think about the **contrapositve** of these statements.

  *"If G_{poly} is a **not** poly-bit stretch PRG, then G is **not** a one-bit stretch PRG."*

- If G_{poly} is not a PRG, then there exists a n.u. PPT adversary $\mathcal{A}$ who can distinguish between its output on a random input and a uniformly sampled string with some **non-negligible** advantage μ.

Can we somehow use $\mathcal{A}$ to also break security of G?