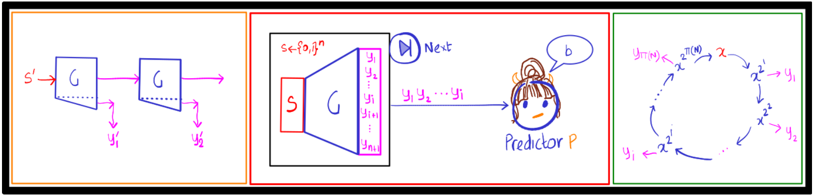
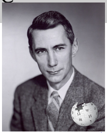


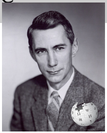
CS783: Theoretical Foundations of Cryptography

Lecture 4 (09/Aug/24)

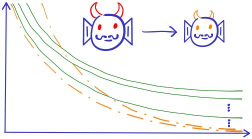
Instructor: Chethan Kamath

- We started off with Shannon's impossibility *

*Credit: Tekniska Museet

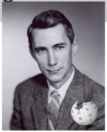


- We started off with Shannon's impossibility *
- One way around Shannon's impossibility is to settle for *computational* secrecy
  - Needed two new notions: PPT and negligible

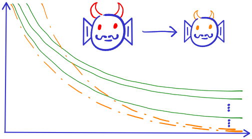


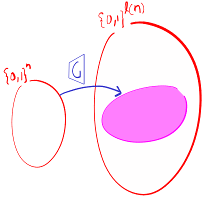
*Credit: Tekniska Museet

Recall from Last Lecture



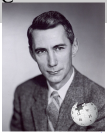
- We started off with Shannon's impossibility *
- One way around Shannon's impossibility is to settle for *computational* secrecy
  - Needed two new notions: PPT and negligible



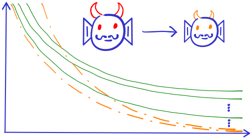


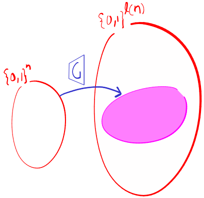
- Defined pseudo-random generators (PRGs)

*Credit: Tekniska Museet

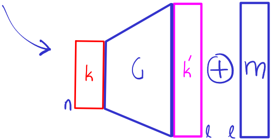


- We started off with Shannon's impossibility *
- One way around Shannon's impossibility is to settle for *computational* secrecy
  - Needed two new notions: PPT and negligible





- Defined pseudo-random generators (PRGs)
- Saw construction of computational OTP from PRG
  - First security reduction!



*Credit: Tekniska Museet

Applications of PRG

- Already saw application of PRG: constructing SKE that allows encrypting longer messages

- Already saw application of PRG: constructing SKE that allows encrypting longer messages
- Several other applications
  - Helps reduce the amount of uniform random bits required: crucial to cryptography since most algorithms are randomised
  - *Derandomisation*, i.e., convert a randomised algorithm into a deterministic algorithm
    - Yao: if "strong" PRGs exist, then $\mathsf{BPP} = \mathsf{P}$


*

*Credit: Kyoto-prize.org

- Already saw application of PRG: constructing SKE that allows encrypting longer messages
- Several other applications
  - Helps reduce the amount of uniform random bits required: crucial to cryptography since most algorithms are randomised
  - *Derandomisation*, i.e., convert a randomised algorithm into a deterministic algorithm
    - Yao: if "strong" PRGs exist, then $\mathsf{BPP} = \mathsf{P}$


*

- Non-cryptographic PRGs (e.g., LFSR): physics simulation

  ⚠ But not pseudorandom in cryptographic sense

*Credit: Kyoto-prize.org

Do PRGs Exist?

- Recall from Lecture 3: they don't if $\mathsf{P} = \mathsf{NP}$.
  - Thus PRGs only exist *conditioned* on $\mathsf{P} \neq \mathsf{NP}$.

Do PRGs Exist?

- Recall from Lecture 3: they don't if $\mathsf{P} = \mathsf{NP}$.
  - Thus PRGs only exist *conditioned* on $\mathsf{P} \neq \mathsf{NP}$.

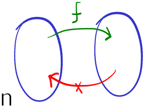
$$\begin{array}{ccccccccc} x & \to & x^2 & \to & x^{2^2} & \to & x^{2^3} & \to & \cdots \\ & & \downarrow & & \downarrow & & \downarrow & & \\ & & b_1 & & b_2 & & b_3 & & \end{array}$$

- This lecture: *Unpredictable sequences* $\to$ PRG
  - Theoretical e.g.: Based on hardness of *factoring* integers
  - Practical e.g.: stream ciphers like Salsa20 and ChaCha

- Recall from Lecture 3: they don't if $\mathsf{P} = \mathsf{NP}$.
  - Thus PRGs only exist *conditioned* on $\mathsf{P} \neq \mathsf{NP}$.

$$\begin{tikzcd} x \arrow[r] & x^2 \arrow[r] \arrow[d] & x^{2^2} \arrow[r] \arrow[d] & x^{2^3} \arrow[r] \arrow[d] & \cdots \\ & b_1 & b_2 & b_3 & \end{tikzcd}$$

- This lecture: *Unpredictable sequences* $\to$ PRG
  - Theoretical e.g.: Based on hardness of *factoring* integers
  - Practical e.g.: stream ciphers like Salsa20 and ChaCha
- Lecture 6: *Hard functions* $\to$ PRG
  - E.g.: one-way function (OWF) and one-way permutation
  - OWF is the *minimal* assumption required for cryptography

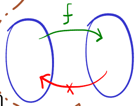


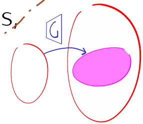
Do PRGs Exist?

- Recall from Lecture 3: they don't if $\mathsf{P} = \mathsf{NP}$.
    - Thus PRGs only exist *conditioned* on $\mathsf{P} \neq \mathsf{NP}$.

$$x \to x^2 \to x^{2^2} \to x^{2^3} \to \cdots$$
$$b_1 \quad b_2 \quad b_3$$

- This lecture: *Unpredictable sequences* $\to$ PRG
    - Theoretical e.g.: Based on hardness of *factoring* integers
    - Practical e.g.: stream ciphers like Salsa20 and ChaCha
- Lecture 6: *Hard functions* $\to$ PRG
    - E.g.: one-way function (OWF) and one-way permutation
    - OWF is the *minimal* assumption required for cryptography
- Thus, seemingly different notions of pseudorandomness, unpredictability and hardness are the same!



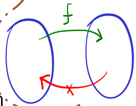


Do PRGs Exist?

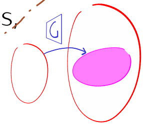
- Recall from Lecture 3: they don't if $\mathsf{P} = \mathsf{NP}$.
  - Thus PRGs only exist *conditioned* on $\mathsf{P} \neq \mathsf{NP}$.

$$x \to x^2 \to x^{2^2} \to x^{2^3} \to \cdots$$
$$\downarrow \quad b_1 \qquad \downarrow \quad b_2 \qquad \downarrow \quad b_3$$

- This lecture: *Unpredictable sequences* → PRG
  - Theoretical e.g.: Based on hardness of *factoring* integers
  - Practical e.g.: stream ciphers like Salsa20 and ChaCha
- Lecture 6: *Hard functions* → PRG
  - E.g.: one-way function (OWF) and one-way permutation
  - OWF is the *minimal* assumption required for cryptography



- Thus, seemingly different notions of pseudorandomness, unpredictability and hardness are the same!



- Note. If PRGs against *fixed-poly.* distinguishers suffices, then: *complexity-theoretic* assumptions → PRG
  - Look up Nisan-Wigderson PRG!

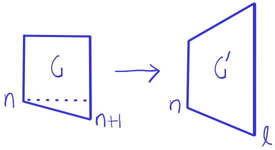
Hardness vs Randomness*

NOAM NISAN† AND AVI WIGDERSON‡

Institute of Computer Science, Hebrew University of Jerusalem, Israel

Plan for this Lecture

1. Length-Extension of PRG



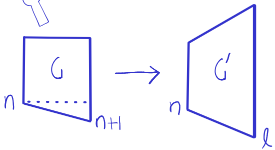
Plan for this Lecture

a) Get a feel for pseudorandomness

b) We'll get to see another reduction

c) Introduces "hybrid argument"

1 Length-Extension of PRG



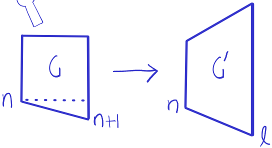
Plan for this Lecture

a) Get a feel for pseudorandomness

b) We'll get to see another reduction

c) Introduces "hybrid argument"

1 Length-Extension of PRG



2 Unpredictability

- Unpredictability is Equivalent to Pseudorandomness
- Unpredictable Sequence from Integer Factoring

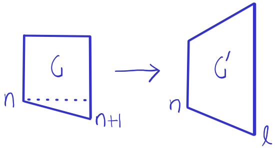
PRG

Unpredictable sequence

Factoring

Plan for this Lecture

1. Length-Extension of PRG



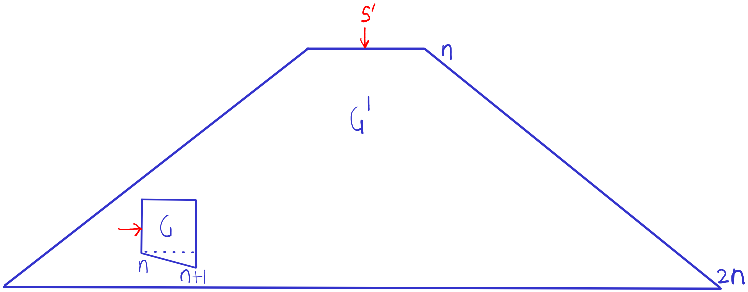
2. Unpredictability
   - Unpredictability is Equivalent to Pseudorandomness
   - Unpredictable Sequence from Integer Factoring

Let's Start by Stretching

- Recall: PRG is an expanding function whose output (on uniformly random input) "seems random" to PPT *distinguishers*.

Let's Start by Stretching

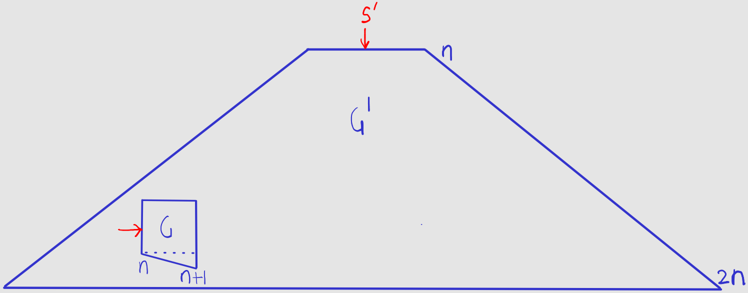
- Recall: PRG is an expanding function whose output (on uniformly random input) "seems random" to PPT *distinguishers*.
- Goal: PRG G with stretch $n+1 \to$ PRG G' with stretch $2n$



Let's Start by Stretching

- Recall: PRG is an expanding function whose output (on uniformly random input) "seems random" to PPT *distinguishers*.
- Goal: PRG G with stretch $n+1$ → PRG G' with stretch $2n$

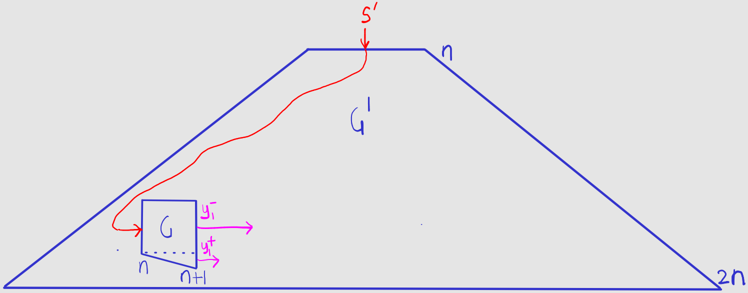
Construction 1



Let's Start by Stretching

- Recall: PRG is an expanding function whose output (on uniformly random input) "seems random" to PPT *distinguishers*.
- Goal: PRG G with stretch $n+1 \rightarrow$ PRG G' with stretch $2n$

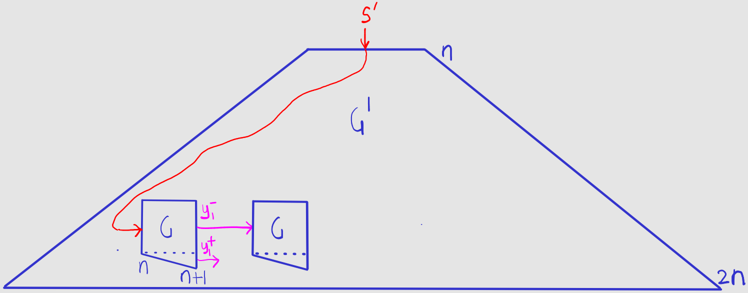
Construction 1



Let's Start by Stretching

- Recall: PRG is an expanding function whose output (on uniformly random input) “seems random” to PPT *distinguishers*.
- Goal: PRG G with stretch $n+1 \rightarrow$ PRG G' with stretch $2n$

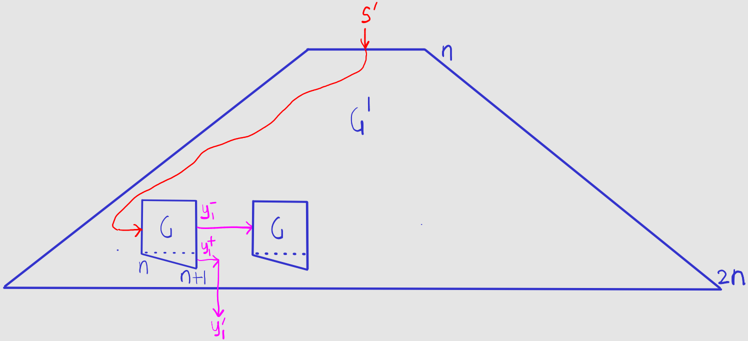
Construction 1



Let's Start by Stretching

- Recall: PRG is an expanding function whose output (on uniformly random input) "seems random" to PPT *distinguishers*.
- Goal: PRG G with stretch $n+1 \rightarrow$ PRG G' with stretch $2n$

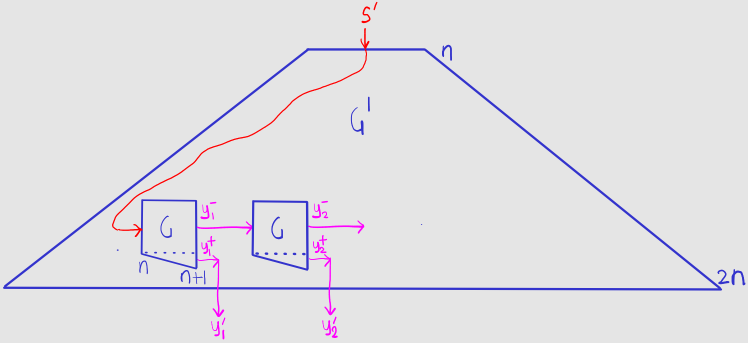
Construction 1



Let's Start by Stretching

- Recall: PRG is an expanding function whose output (on uniformly random input) "seems random" to PPT *distinguishers*.
- Goal: PRG G with stretch $n+1 \rightarrow$ PRG G' with stretch $2n$

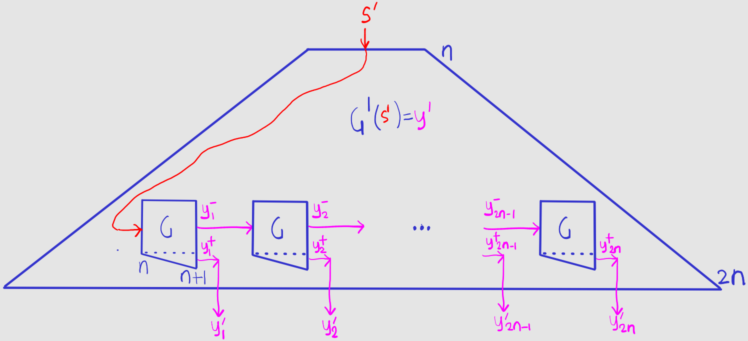
Construction 1



Let's Start by Stretching

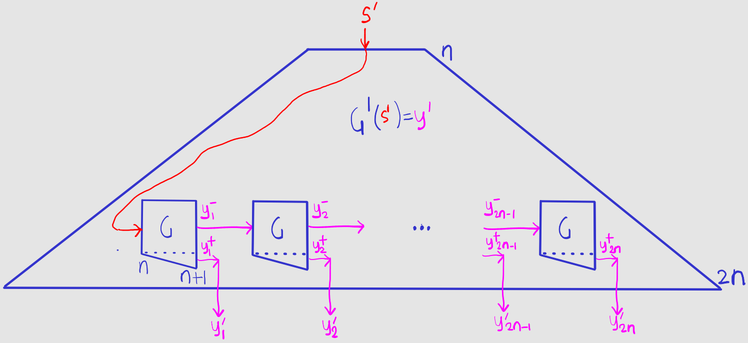
- Recall: PRG is an expanding function whose output (on uniformly random input) "seems random" to PPT *distinguishers*.
- Goal: PRG G with stretch $n+1 \to$ PRG G' with stretch $2n$

Construction 1



- Recall: PRG is an expanding function whose output (on uniformly random input) "seems random" to PPT *distinguishers*.
- Goal: PRG G with stretch $n+1 \to$ PRG G′ with stretch $2n$

Construction 1



Exercise 1

Formally write down the construction of G′.

Before the Proof, Recall Definition of PRG

Defintion 1 (PRG, via Imitation Game)

Let G *be an efficient deterministic algorithm that for any* $n \in \mathbb{N}$ *and input* $s \in \{0,1\}^n$*, outputs a string of length* $\ell(n) > n$. ← "stretch/ expansion factor"
G *is PRG if for every PPT distinguisher* D

$$\delta(n) := \left| \Pr_{s \leftarrow \{0,1\}^n}[\mathsf{D}(G(s)) = 0] - \Pr_{r \leftarrow \{0,1\}^{\ell(n)}}[\mathsf{D}(r) = 0] \right|$$

is negligible.

Before the Proof, Recall Definition of PRG

Defintion 1 (PRG, via Imitation Game)

Let G *be an efficient deterministic algorithm that for any* $n \in \mathbb{N}$ *and input* $s \in \{0,1\}^n$*, outputs a string of length* $\ell(n) > n$. ← "stretch/ expansion factor"

G *is PRG if for every PPT distinguisher* D

$$\delta(n) := \left| \Pr_{s \leftarrow \{0,1\}^n}[D(G(s)) = 0] - \Pr_{r \leftarrow \{0,1\}^{\ell(n)}}[D(r) = 0] \right|$$

pseudorandom world   random world

is negligible.

Before the Proof, Recall Definition of PRG

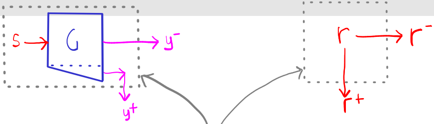
Defintion 1 (PRG, via Imitation Game)

Let G *be an efficient deterministic algorithm that for any* $n \in \mathbb{N}$ *and input* $s \in \{0,1\}^n$*, outputs a string of length* $\ell(n) > n$. ← "stretch/ expansion factor"
G *is PRG if for every PPT distinguisher* D

$$\delta(n) := \left| \Pr_{s \leftarrow \{0,1\}^n}[D(G(s)) = 0] - \Pr_{r \leftarrow \{0,1\}^{\ell(n)}}[D(r) = 0] \right|$$

pseudorandom world　　random world

is negligible.



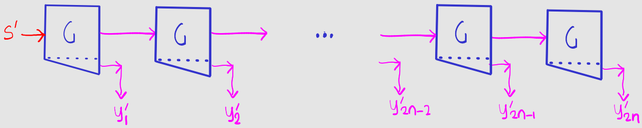
For G stretching by one bit

Proving Pseudorandomness: a Hybrid (Security) Argument

Theorem 1

If G *is a PRG, then so is* G′.

Proof. Intuition



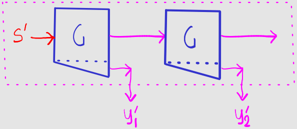
Proving Pseudorandomness: a Hybrid (Security) Argument

Theorem 1

If G *is a PRG, then so is* G′.

Proof. Intuition .

Let's focus on just two iterations



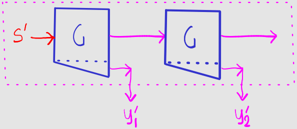
Proving Pseudorandomness: a Hybrid (Security) Argument

Theorem 1

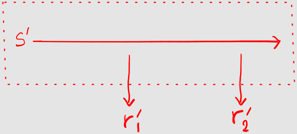
If G *is a PRG, then so is* G′.

Proof. Intuition

Let's focus on just two iterations



H0 pseudorandom world



H2 random world

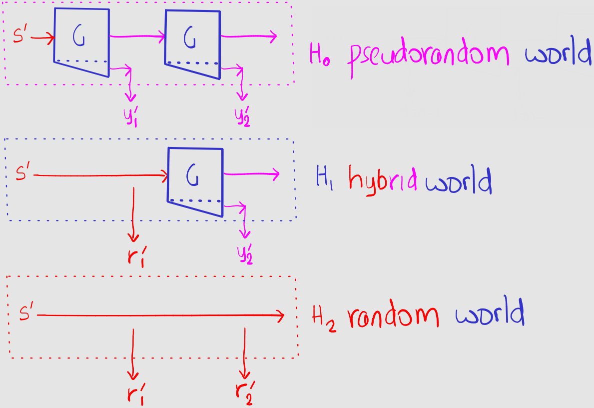
Proving Pseudorandomness: a Hybrid (Security) Argument

Theorem 1

If G *is a PRG, then so is* G′.

Proof. Intuition: consider hybrid worlds .

Let's focus on just two iterations



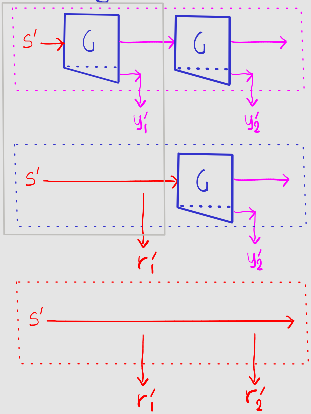
Proving Pseudorandomness: a Hybrid (Security) Argument

Theorem 1

If G *is a PRG, then so is* G′.

Proof. Intuition: consider hybrid worlds .

Let's focus on just two iterations



H_0 pseudorandom world

H_1 hybrid world

H_2 random world

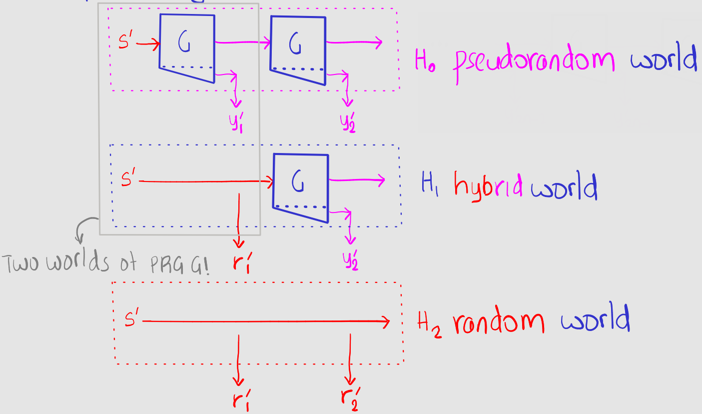
Proving Pseudorandomness: a Hybrid (Security) Argument

Theorem 1

If G *is a PRG, then so is* G′.

Proof. Intuition: consider hybrid worlds .

Let's focus on just two iterations

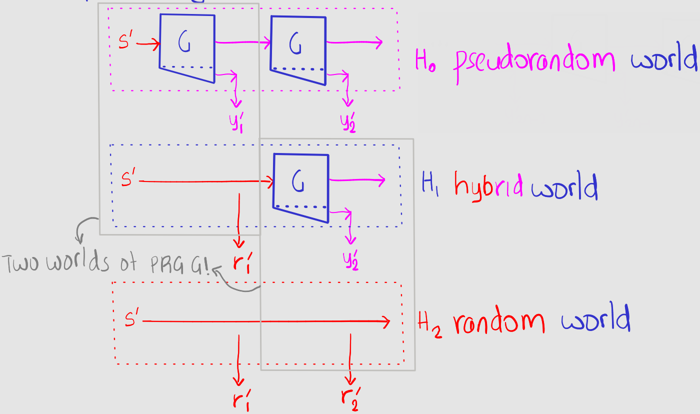


Theorem 1

If G is a PRG, then so is G'.

Proof. Intuition: consider hybrid worlds .

Let's focus on just two iterations



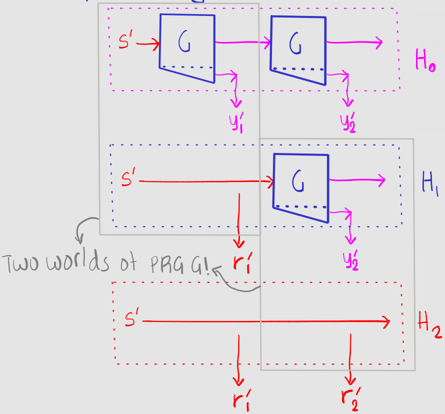
Proving Pseudorandomness: a Hybrid (Security) Argument

Theorem 1

If G *is a PRG, then so is* G′.

Proof. Intuition: consider hybrid worlds .

Let's focus on just two iterations



Claim 1

Distinguisher for G′

$$\Pr[D'(H_0)=0]-\Pr[D'(H_2)=0]\geq\delta$$

$$\Downarrow$$

$\exists i\in[0,1]$ such that

$$\Pr[D'(H_i)=0]-\Pr[D'(H_{i+1})=0]\geq\delta/2$$

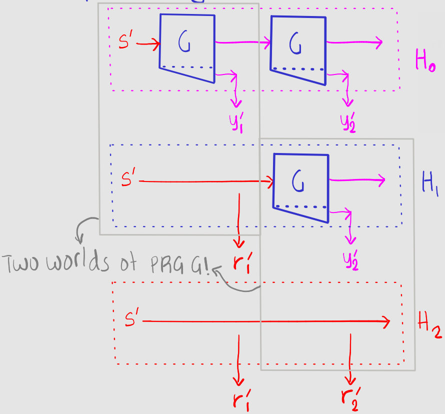
Proving Pseudorandomness: a Hybrid (Security) Argument

Theorem 1

If G *is a PRG, then so is* G'.

Proof. Intuition: consider hybrid worlds .

Let's focus on just two iterations



Claim 1

Distinguisher for G'

$$\Pr[D'(H_0)=0] - \Pr[D'(H_2)=0] \geq \delta$$

$$\Downarrow$$

$\exists i \in [0,1]$ such that ?

$$\Pr[D'(H_i)=0] - \Pr[D'(H_{i+1})=0] \geq \delta/2$$

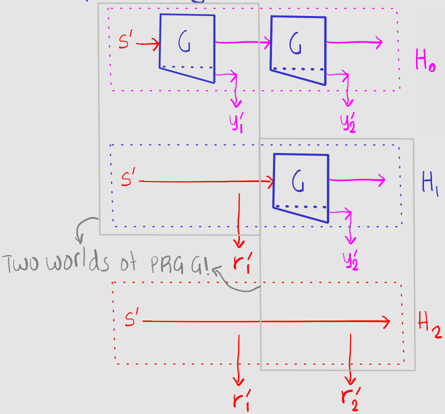
Proving Pseudorandomness: a Hybrid (Security) Argument

Theorem 1

If G *is a PRG, then so is* G′.

Proof. Intuition: consider hybrid worlds .

Let's focus on just two iterations



Claim 1 Distinguisher for G′

$\Pr[D'(H_0)=0] - \Pr[D'(H_2)=0] \geq \delta$

$\Downarrow$

$\exists i \in [0,1]$ such that ?

$\Pr[D'(H_i)=0] - \Pr[D'(H_{i+1})=0] \geq \delta/2$ (∗)

Claim 2

(∗) $\Rightarrow$ Distinguisher D for G !

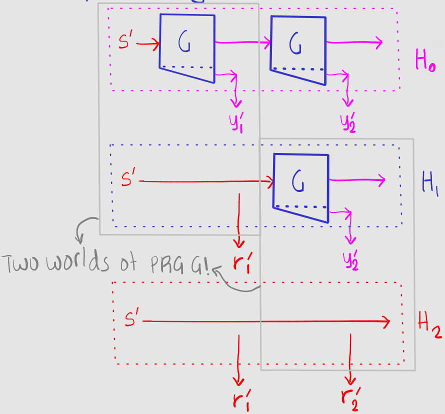
Proving Pseudorandomness: a Hybrid (Security) Argument

Theorem 1

If G *is a PRG, then so is* G′.

Proof. Intuition: consider hybrid worlds .

Let's focus on just two iterations



Claim 1

Distinguisher for G′

$$\Pr[D'(H_0)=0] - \Pr[D'(H_2)=0] \geq \delta$$

$$\Downarrow$$

$\exists i \in [0,1]$ such that

$$\Pr[D'(H_i)=0] - \Pr[D'(H_{i+1})=0] \geq \delta/2 \quad (*)$$

Claim 2

$(*) \Rightarrow$ Distinguisher D for G !

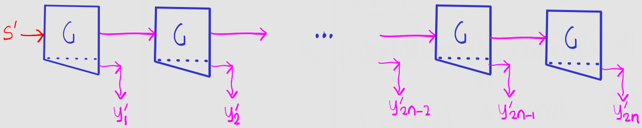
Claims 1 & 2 $\Rightarrow$ Theorem 1

Proving Pseudorandomness: a Hybrid (Security) Argument

Theorem 1

If G *is a PRG, then so is* G′.

Proof. Intuition: consider hybrid worlds .

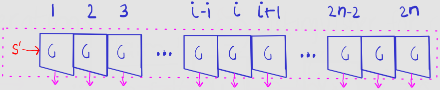


Proving Pseudorandomness: a Hybrid (Security) Argument

Theorem 1

If G *is a PRG, then so is* G′.

Proof. Intuition: consider hybrid worlds .

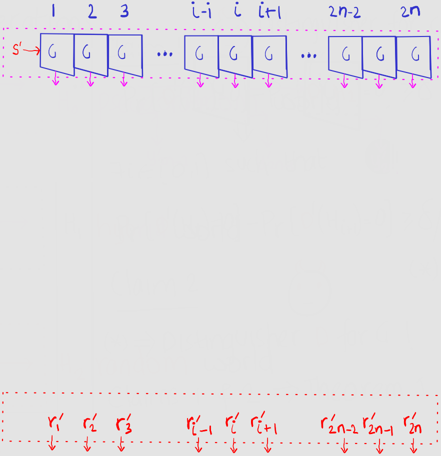


Proving Pseudorandomness: a Hybrid (Security) Argument

Theorem 1

If G *is a PRG, then so is* G′.

Proof. Intuition: consider hybrid worlds .

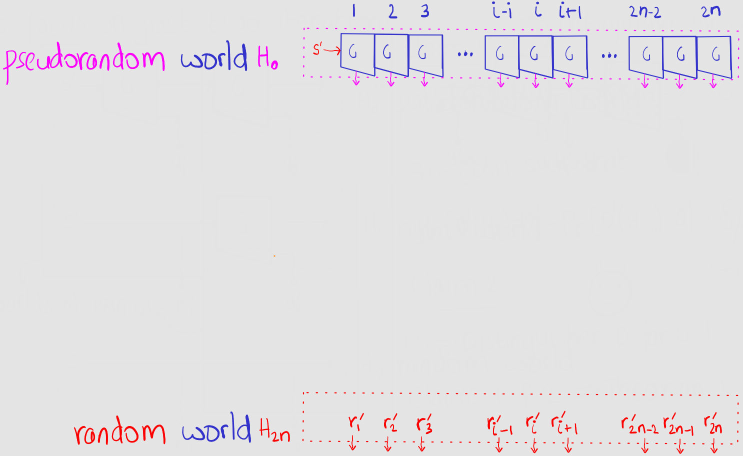


Proving Pseudorandomness: a Hybrid (Security) Argument

Theorem 1

If G *is a PRG, then so is* G'.

Proof. Intuition: consider hybrid worlds .

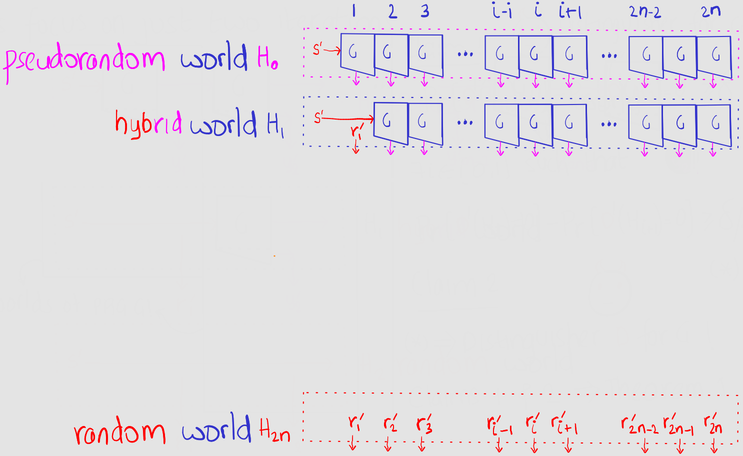


Proving Pseudorandomness: a Hybrid (Security) Argument

Theorem 1

If G *is a PRG, then so is* G′.

Proof. Intuition: consider hybrid worlds .

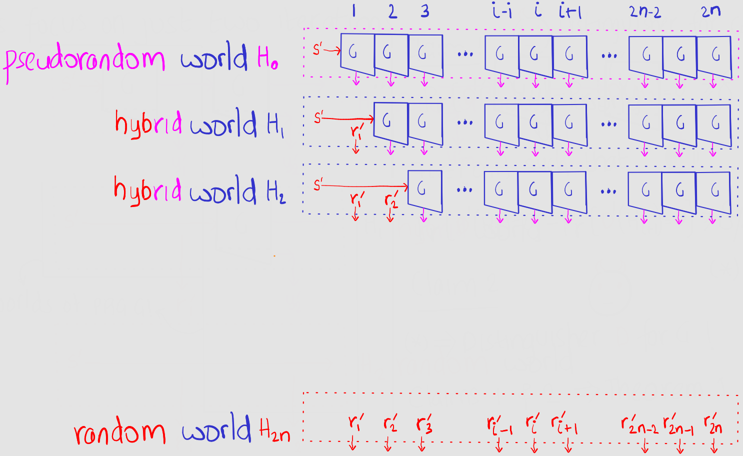


Proving Pseudorandomness: a Hybrid (Security) Argument

Theorem 1

If G *is a PRG, then so is* G′.

Proof. Intuition: consider hybrid worlds .

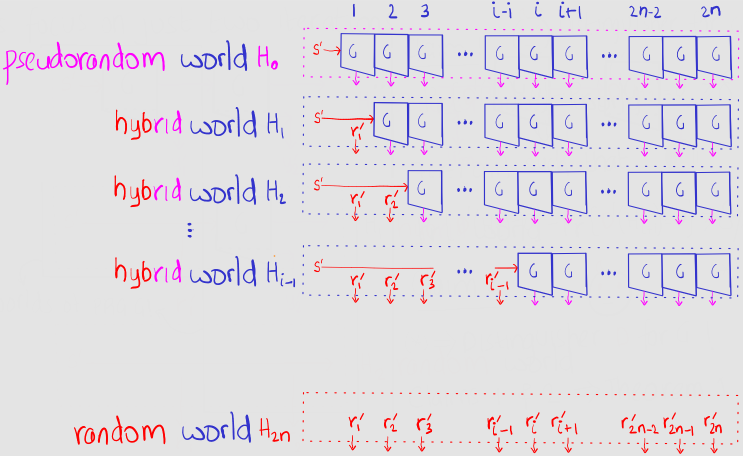


Proving Pseudorandomness: a Hybrid (Security) Argument

Theorem 1

If G *is a PRG, then so is* G′.

Proof. Intuition: consider hybrid worlds .

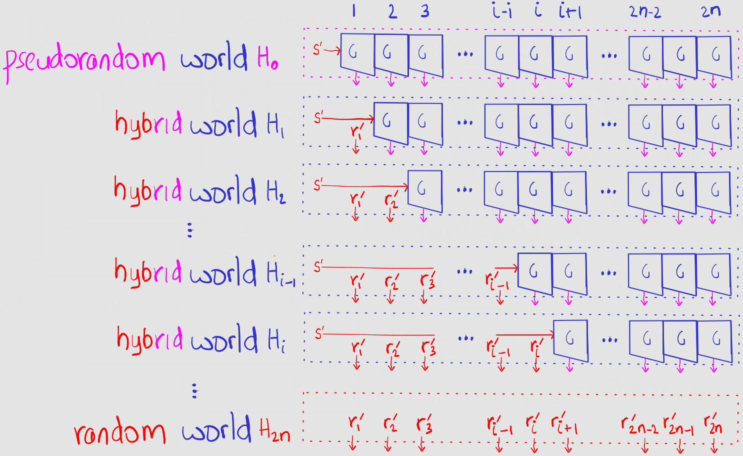


Proving Pseudorandomness: a Hybrid (Security) Argument

Theorem 1

If G *is a PRG, then so is* G′.

Proof. Intuition: consider hybrid worlds .

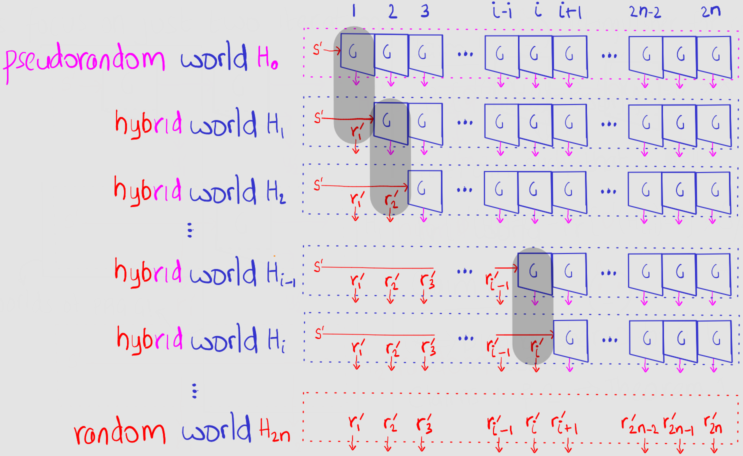


Proving Pseudorandomness: a Hybrid (Security) Argument

Theorem 1

If G *is a PRG, then so is* G′.

Proof. Intuition: consider hybrid worlds .

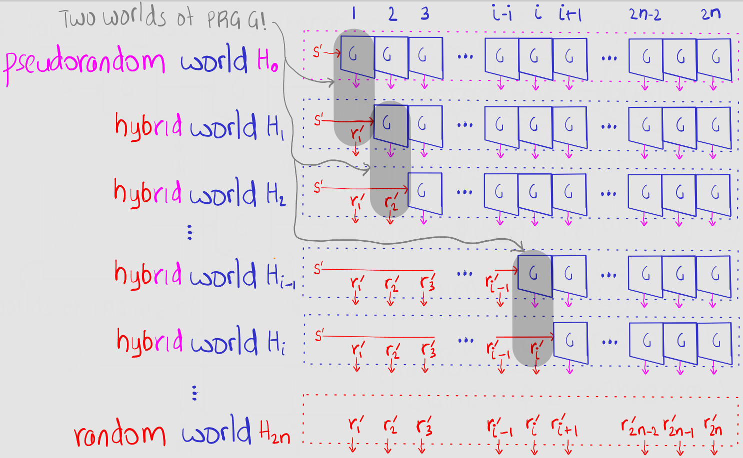


Proving Pseudorandomness: a Hybrid (Security) Argument

Theorem 1

If G *is a PRG, then so is* G′.

Proof. Intuition: consider hybrid worlds .



Proving Pseudorandomness: a Hybrid (Security) Argument

Theorem 1

If G *is a PRG, then so is* G′.

Proof. ∃ distinguisher D for G ⇐ ∃ distinguisher D′ for G′.

PRG G′

Proving Pseudorandomness: a Hybrid (Security) Argument

Theorem 1

If G *is a PRG, then so is* G′.

Proof. ∃ distinguisher D for G ⇐ ∃ distinguisher D′ for G′.

PRG G

PRG G′

Distinguisher D
"Reduction"

Distinguisher D′

Proving Pseudorandomness: a Hybrid (Security) Argument

Theorem 1

If G *is a PRG, then so is* G'.

Proof. $\exists$ distinguisher D for $\mathsf{G} \Leftarrow \exists$ distinguisher D′ for G'.

PRG G

PRG G′

$\underbrace{w = w^- \| w^+}$

G(s) or random r $\longrightarrow$

Distinguisher D

"Reduction"

Distinguisher D′

Proving Pseudorandomness: a Hybrid (Security) Argument

Theorem 1

If G *is a PRG, then so is* G′.

Proof. ∃ distinguisher D for G ⇐ ∃ distinguisher D′ for G′.

PRG G

PRG G′

$G(s)$ or $\overbrace{\text{random } r}^{w = w^- \| w^+}$ $\rightarrow$

Distinguisher D

"Reduction"

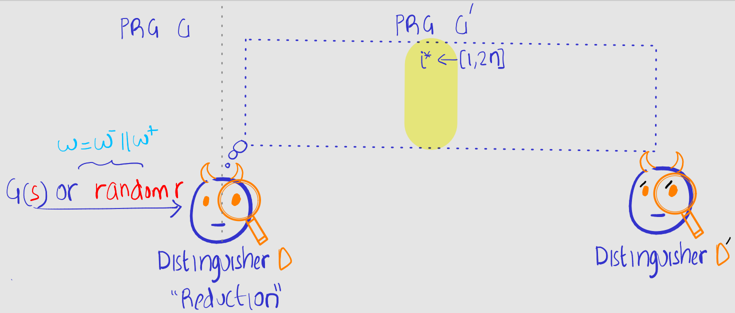
Distinguisher D′

Proving Pseudorandomness: a Hybrid (Security) Argument

Theorem 1

If G *is a PRG, then so is* G′.

Proof. ∃ distinguisher D for G ⇐ ∃ distinguisher D′ for G′.

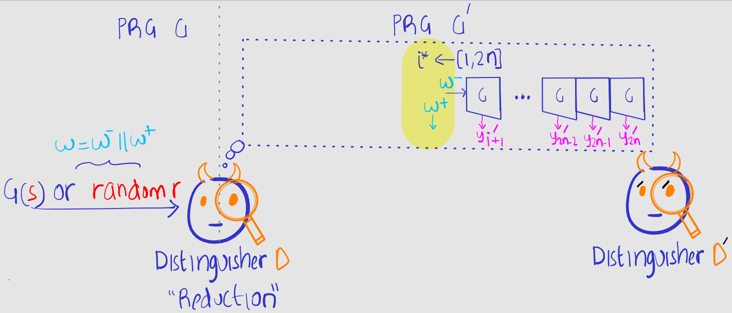


Proving Pseudorandomness: a Hybrid (Security) Argument

Theorem 1

If G *is a PRG, then so is* G′.

Proof. ∃ distinguisher D for G ⇐ ∃ distinguisher D′ for G′.

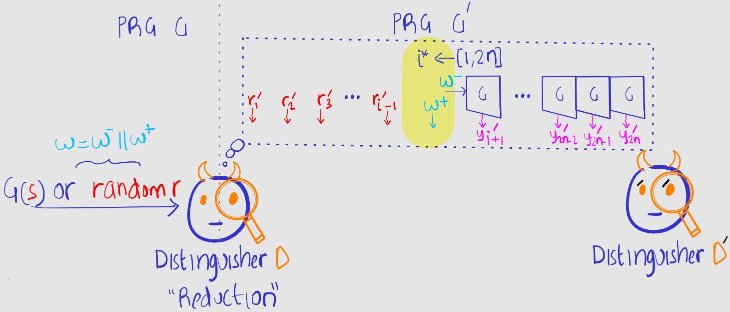


Proving Pseudorandomness: a Hybrid (Security) Argument

Theorem 1

If G *is a PRG, then so is* G′.

Proof. ∃ distinguisher D for G ⇐ ∃ distinguisher D′ for G′.

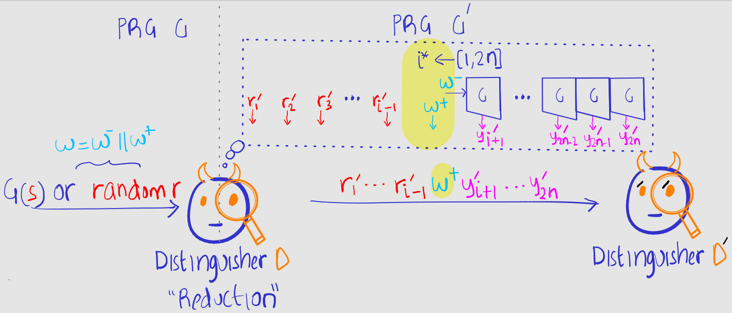


Proving Pseudorandomness: a Hybrid (Security) Argument

Theorem 1

If G *is a PRG, then so is* G′.

Proof. ∃ distinguisher D for G ⇐ ∃ distinguisher D′ for G′.

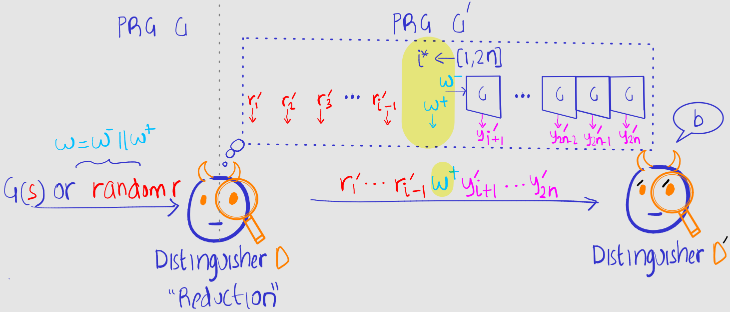


Proving Pseudorandomness: a Hybrid (Security) Argument

Theorem 1

If G *is a PRG, then so is* G′.

Proof. ∃ distinguisher D for G ⇐ ∃ distinguisher D′ for G′.

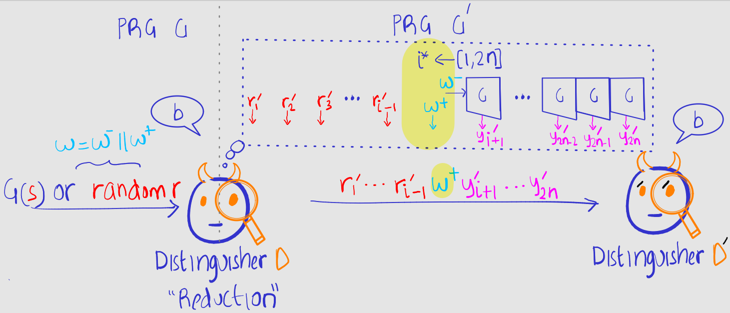


Proving Pseudorandomness: a Hybrid (Security) Argument

Theorem 1

If G *is a PRG, then so is* G'.

Proof. $\exists$ distinguisher D for $\mathsf{G} \Leftarrow \exists$ distinguisher D′ for G'.

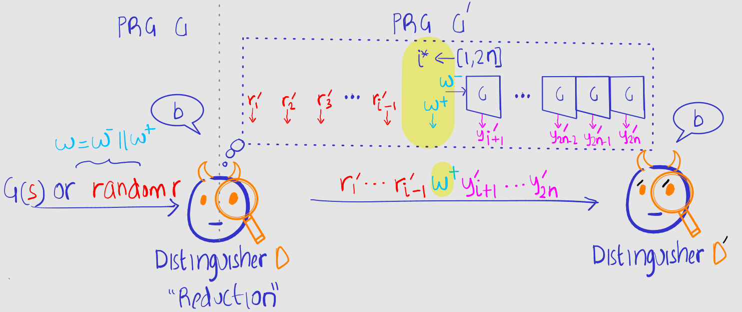


Proving Pseudorandomness: a Hybrid (Security) Argument

Theorem 1

If G *is a PRG, then so is* G′.

Proof. ∃ distinguisher D for G ⇐ ∃ distinguisher D′ for G′.



Analysis: similar to claims 1 and 2.

$\Pr[D'(H_0)=0] - \Pr[D'(H_{2n})=0] \geq \delta \Rightarrow$ $\qquad \Pr[i^*=i] = 1/2n$

$\exists i \in [0,2n-1]$ such that $\Pr[D'(H_i)=0] - \Pr[D'(H_{i+1})=0] \geq \delta/2n$ ∎

Let's Take Stock of Theorem 1

- Construction 1 and Theorem 1 work for any polynomial stretch
  - ? What happens if we stretch it exponentially?

Let's Take Stock of Theorem 1

- Construction 1 and Theorem 1 work for any polynomial stretch
  - ? What happens if we stretch it exponentially?

- There is also a "loss in pseudorandomness"
  - D′ distinguishes with some probability $1/p(n) \Rightarrow$
    D distinguishes with probability only $\approx 1/p(n) \cdot 2n$

Let's Take Stock of Theorem 1

- Construction 1 and Theorem 1 work for any polynomial stretch
  - ? What happens if we stretch it exponentially?

- There is also a "loss in pseudorandomness"
  - D′ distinguishes with some probability $1/p(n) \Rightarrow$ D distinguishes with probability only $\approx 1/p(n) \cdot 2n$
  - More the stretch, greater the loss

Let's Take Stock of Theorem 1

- Construction 1 and Theorem 1 work for any polynomial stretch
  - What happens if we stretch it exponentially?

- There is also a "loss in pseudorandomness"
  - D′ distinguishes with some probability $1/p(n) \Rightarrow$
    D distinguishes with probability only $\approx 1/p(n) \cdot 2n$
  - More the stretch, greater the loss

- More generally: "loss in security" of a security reduction
  - One way to measure how "wasteful" the reduction is

Let's Take Stock of Theorem 1

- Construction 1 and Theorem 1 work for any polynomial stretch

  ? What happens if we stretch it exponentially?

- There is also a "loss in pseudorandomness"
  - D′ distinguishes with some probability $1/p(n) \Rightarrow$
    D distinguishes with probability only $\approx 1/p(n) \cdot 2n$
  - More the stretch, greater the loss

- More generally: "loss in security" of a security reduction
  - One way to measure how "wasteful" the reduction is

Exercise 2

- *Think of a less wasteful reduction strategy for Theorem 1. Do you feel it is possible?*

Let's Take Stock of Theorem 1

- Construction 1 and Theorem 1 work for any polynomial stretch
  - What happens if we stretch it exponentially?

- There is also a "loss in pseudorandomness"
  - D′ distinguishes with some probability $1/p(n) \Rightarrow$ D distinguishes with probability only $\approx 1/p(n) \cdot 2n$
  - More the stretch, greater the loss

- More generally: "loss in security" of a security reduction
  - One way to measure how "wasteful" the reduction is

Exercise 2

- *Think of a less wasteful reduction strategy for Theorem 1. Do you feel it is possible?*
- *Maybe need a different construction?*

Plan for this Lecture

1 Length-Extension of PRG

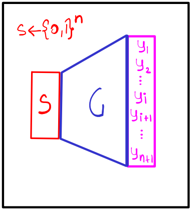
2 Unpredictability

- Unpredictability is Equivalent to Pseudorandomness
- Unpredictable Sequence from Integer Factoring

PRG
↑
Unpredictable sequence
↑
Factoring

Let's Define Unpredictability...

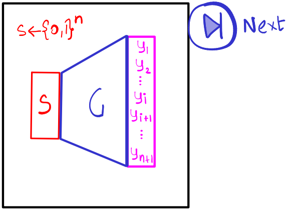
- Intuition: no PPT *predictor* can, given first i output bits, predict $i+1$-th output bit



Predictor P

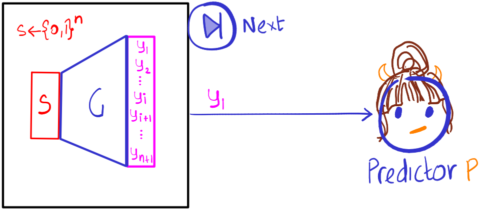
Let's Define Unpredictability...

- Intuition: no PPT *predictor* can, given first i output bits, predict $i+1$-th output bit



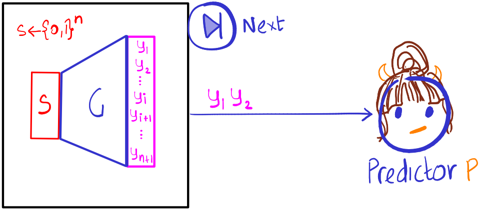
Let's Define Unpredictability...

- Intuition: no PPT *predictor* can, given first i output bits, predict $i+1$-th output bit

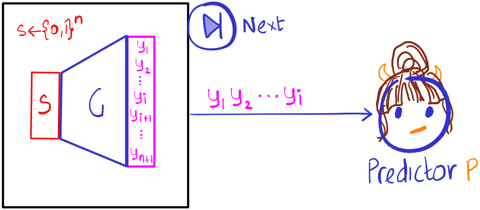


Let's Define Unpredictability...

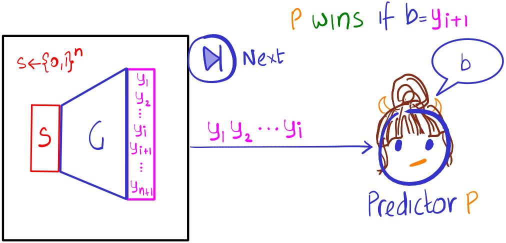
Intuition: no PPT *predictor* can, given first i output bits, predict $i+1$–th output bit



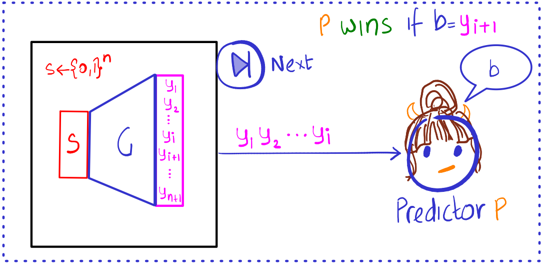
- Intuition: no PPT *predictor* can, given first i output bits, predict $i+1$-th output bit



- Intuition: no PPT *predictor* can, given first i output bits, predict $i+1$-th output bit



- Intuition: no PPT *predictor* can, given first i output bits, predict $i+1$-th output bit

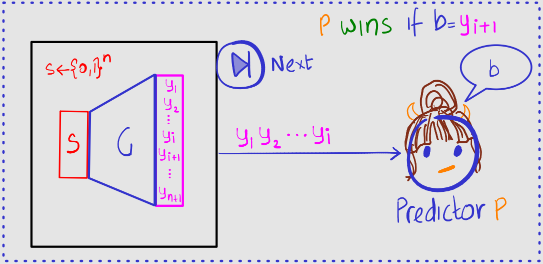


Let's Define Unpredictability...

Intuition: no PPT *predictor* can, given first i output bits, predict $i+1$-th output bit

Defintion 2 (Tailored for expanding functions of stretch $n+1$)

Let G *be an efficient deterministic algorithm that for any $n \in \mathbb{N}$ and input $s \in \{0,1\}^n$, outputs a string of length $n+1$.*



Let's Define Unpredictability...

Intuition: no PPT *predictor* can, given first i output bits, predict $i+1$-th output bit

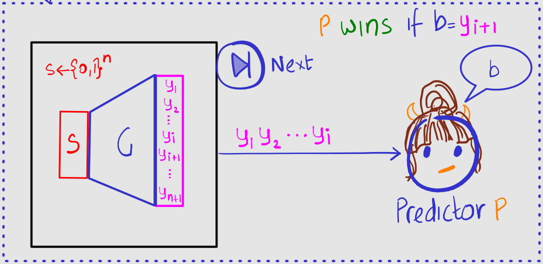
Defintion 2 (Tailored for expanding functions of stretch $n+1$)

Let G *be an efficient deterministic algorithm that for any* $n \in \mathbb{N}$ *and input* $s \in \{0,1\}^n$*, outputs a string of length* $n+1$.
G *is next-bit unpredictable if for all PPT predictors* P*:*

"on the right"

$$\Pr[\mathsf{P}\ \mathit{wins}] - \frac{1}{2}$$

is negligible.



Let's Define Unpredictability...

- Intuition: no PPT *predictor* can, given first i output bits, predict $i+1$-th output bit

Defintion 2 (Tailored for expanding functions of stretch $n+1$)

Let G *be an efficient deterministic algorithm that for any* $n \in \mathbb{N}$ *and input* $s \in \{0,1\}^n$, *outputs a string of length* $n+1$.

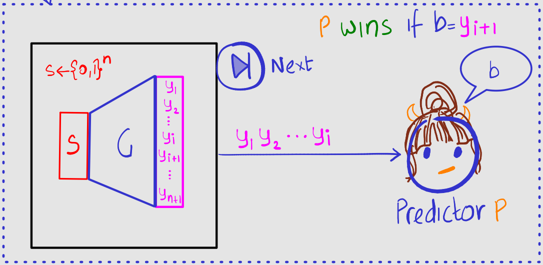
G *is* ~~*next-bit*~~ *unpredictable if for all PPT predictors* P*:*

~~"on the right"~~

"on the left"

$$\Pr[\mathsf{P}\ \textit{wins}] - \frac{1}{2}$$

is negligible.



Let's Define Unpredictability...

Intuition: no PPT *predictor* can, given first i output bits, predict $i+1$-th output bit

Defintion 2 (Tailored for expanding functions of stretch $n+1$)

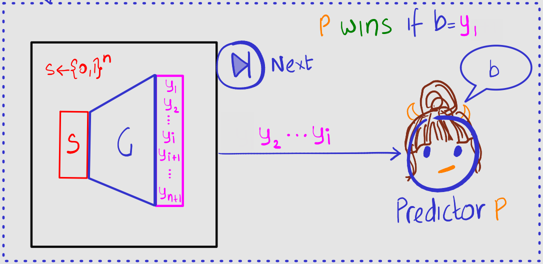
Let G *be an efficient deterministic algorithm that for any* $n \in \mathbb{N}$ *and input* $s \in \{0,1\}^n$*, outputs a string of length* $n+1$*.*
G *is* ~~*next-bit*~~ *unpredictable if for all PPT predictors* P*:*

~~"on the right"~~

"on the left"

$$\Pr[\text{P } \textit{wins}] - \frac{1}{2}$$

is negligible.

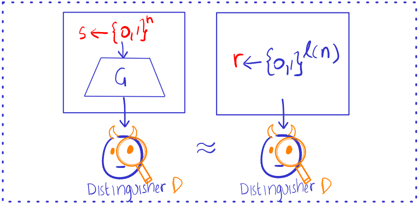


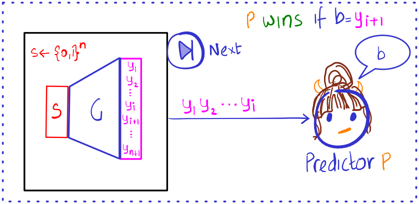
Plan for this Lecture

1 Length-Extension of PRG

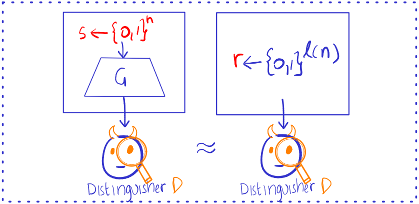
2 Unpredictability
- Unpredictability is Equivalent to Pseudorandomness
- Unpredictable Sequence from Integer Factoring

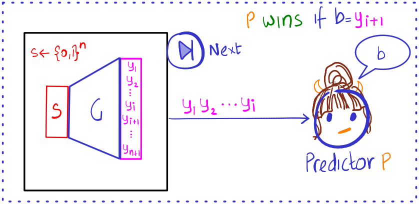
Unpredictability is Equivalent to Pseudorandomness





Which definition do you feel is easier to achieve?





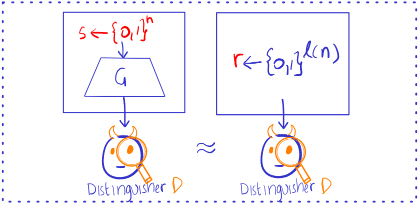
Which definition do you feel is easier to achieve?

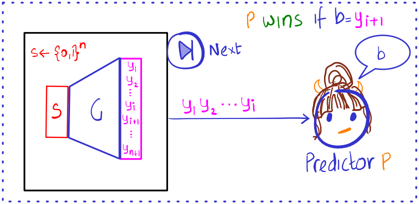
- Easier direction:

Exercise 3

Show that pseudorandomness (Defintion 1) implies next-bit unpredictability (Defintion 2).

Unpredictability is Equivalent to Pseudorandomness





Which definition do you feel is easier to achieve?

- Easier direction:

Exercise 3

Show that pseudorandomness (Defintion 1) implies next-bit unpredictability (Defintion 2). Hint:

- *Goal:* $\exists$ *distinguisher* D *for* G $\Leftarrow$ $\exists$ *predictor* P *for* G
- *Feed* P *with prefix of challenge* w *(*r *or* $\mathsf{G}(s)$*) of random length.*
- *If* P *predicts the next bit of* w *correctly, then we're likely in the pseudorandom world*

Unpredictability Implies Pseudorandomness

Theorem 2

If G *is next-bit unpredictable, then it is a pseudorandom.*

Proof Sketch. Intuition: 1) hybrid argument .

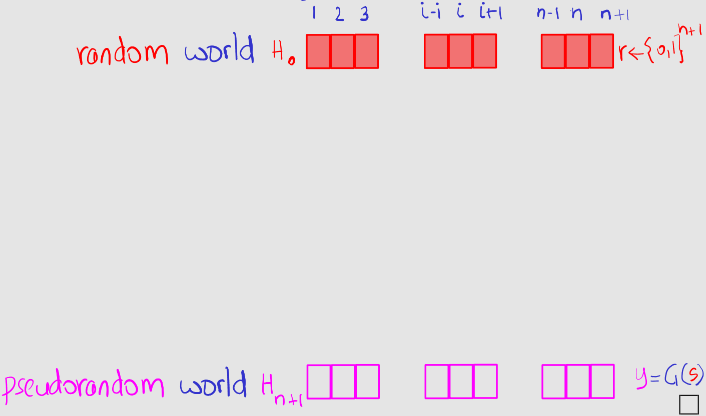
$\square$

Unpredictability Implies Pseudorandomness

Theorem 2

If G *is next-bit unpredictable, then it is a pseudorandom.*

Proof Sketch. Intuition: 1) hybrid argument



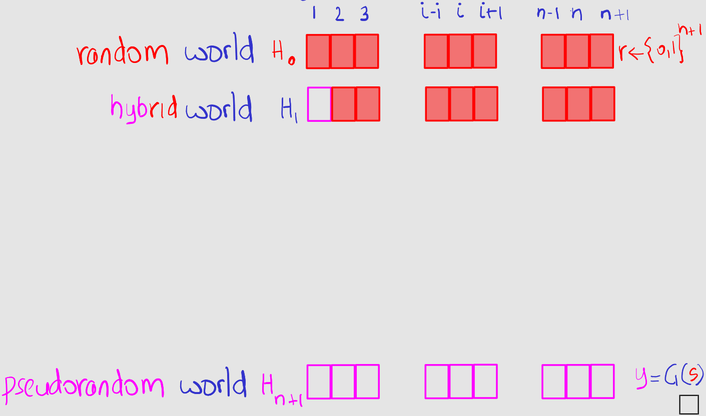
□

Unpredictability Implies Pseudorandomness

Theorem 2

If G is next-bit unpredictable, then it is a pseudorandom.

Proof Sketch. Intuition: 1) hybrid argument .



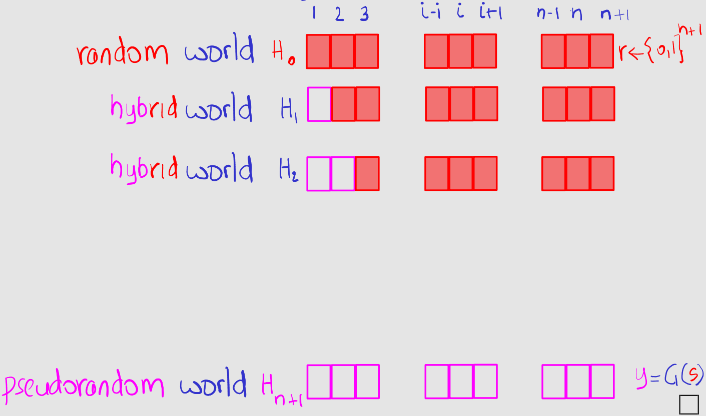
□

Unpredictability Implies Pseudorandomness

Theorem 2

If G *is next-bit unpredictable, then it is a pseudorandom.*

Proof Sketch. Intuition: 1) hybrid argument



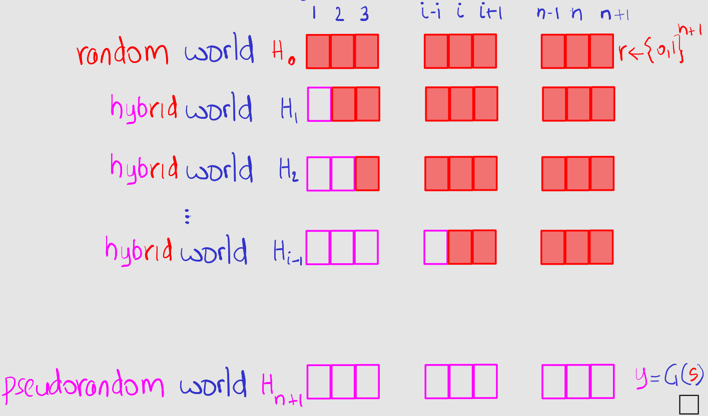
$\square$

Unpredictability Implies Pseudorandomness

Theorem 2

If G *is next-bit unpredictable, then it is a pseudorandom.*

Proof Sketch. Intuition: 1) hybrid argument



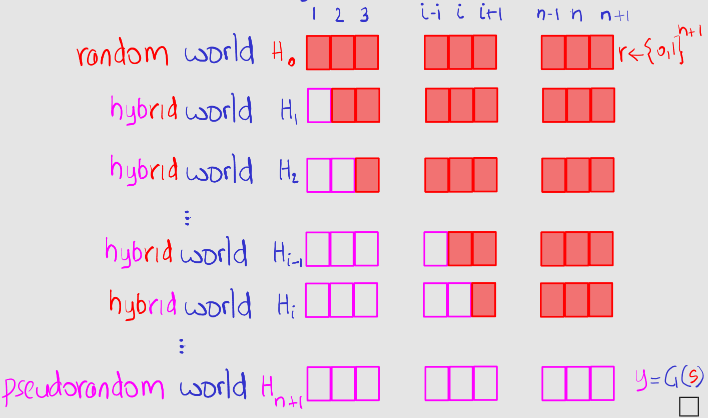
$\square$

Unpredictability Implies Pseudorandomness

Theorem 2

If G *is next-bit unpredictable, then it is a pseudorandom.*

Proof Sketch. Intuition: 1) hybrid argument



□

Unpredictability Implies Pseudorandomness

Theorem 2

If G *is next-bit unpredictable, then it is a pseudorandom.*

Proof Sketch. Intuition: 1) hybrid argument

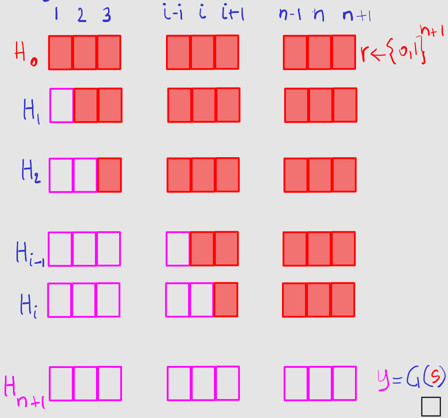
Claim 1 Distinguisher for G

$\Pr[D(H_{n+1})=0] - \Pr[D(H_0)=0] \geq \delta$

$\Downarrow$

$\exists i \in [1, n+1]$ such that ?

$\Pr[D(H_i)=0] - \Pr[D(H_{i-1})=0] \geq \delta/n$



□

Unpredictability Implies Pseudorandomness

Theorem 2

If G is next-bit unpredictable, then it is a pseudorandom.

Proof Sketch. Intuition: 1) hybrid argument 2) distinguisher → Predictor

Claim 1

Distinguisher for G

$\Pr[D(H_{n+1})=0] - \Pr[D(H_0)=0] \geq \delta$

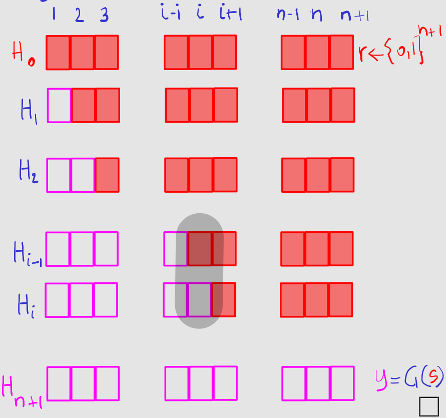
$\Downarrow$

$\exists i \in [1, n+1]$ such that ?

$\Pr[D(H_i)=0] - \Pr[D(H_{i-1})=0] \geq \delta/n$

Claim 2 :

D outputs 0 more often when given correct bit y_i than wrong bit $\bar{y}_i$!



□

Unpredictability Implies Pseudorandomness

Theorem 2

If G *is next-bit unpredictable, then it is a pseudorandom.*

Proof Sketch. $\exists$ predictor P for G $\Leftarrow$ $\exists$ distinguisher D for G.

Pseudorandomness

Predictor P

Distinguisher D

$\square$

Unpredictability Implies Pseudorandomness

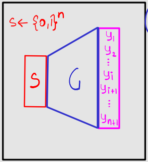
Theorem 2

If G is next-bit unpredictable, then it is a pseudorandom.

Proof Sketch. ∃ predictor P for G ⇐ ∃ distinguisher D for G.

Unpredictability

Pseudorandomness



Next

Predictor P
"Reduction"

Distinguisher D

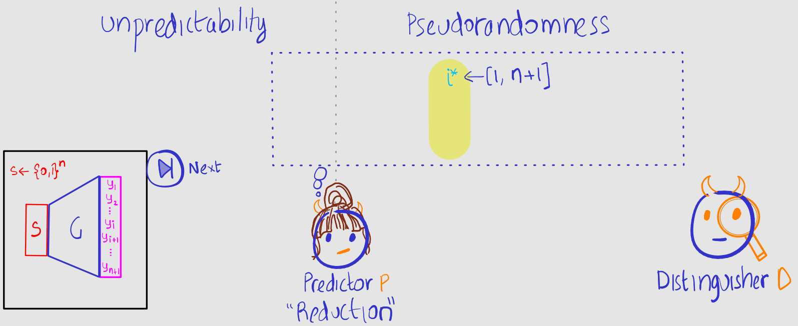
□

Unpredictability Implies Pseudorandomness

Theorem 2

If G *is next-bit unpredictable, then it is a pseudorandom.*

Proof Sketch. $\exists$ predictor P for G $\Leftarrow$ $\exists$ distinguisher D for G.



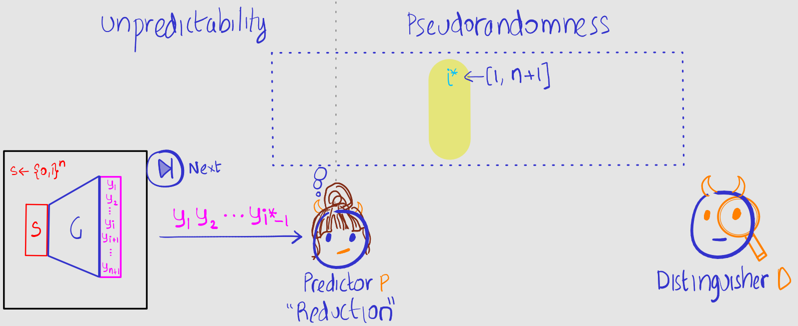
□

Unpredictability Implies Pseudorandomness

Theorem 2

If G is next-bit unpredictable, then it is a pseudorandom.

Proof Sketch. $\exists$ predictor P for G $\Leftarrow$ $\exists$ distinguisher D for G.



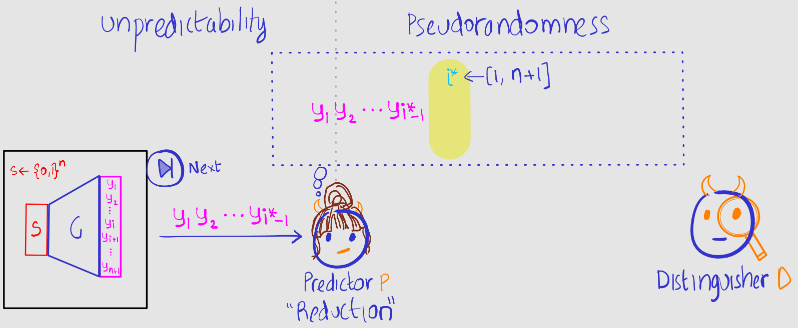
□

Unpredictability Implies Pseudorandomness

Theorem 2

If G *is next-bit unpredictable, then it is a pseudorandom.*

Proof Sketch. $\exists$ predictor P for G $\Leftarrow$ $\exists$ distinguisher D for G.



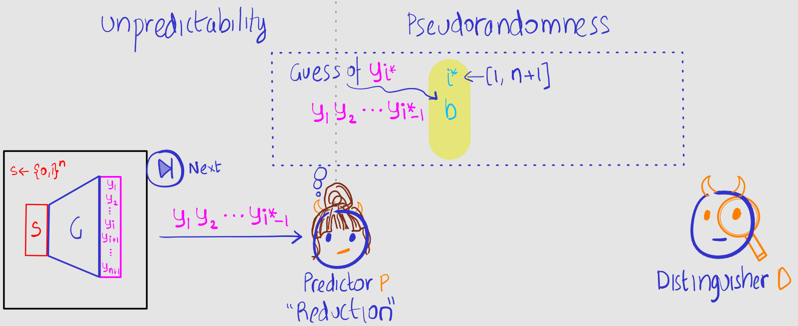
□

Unpredictability Implies Pseudorandomness

Theorem 2

If G *is next-bit unpredictable, then it is a pseudorandom.*

Proof Sketch. ∃ predictor P for G ⇐ ∃ distinguisher D for G.



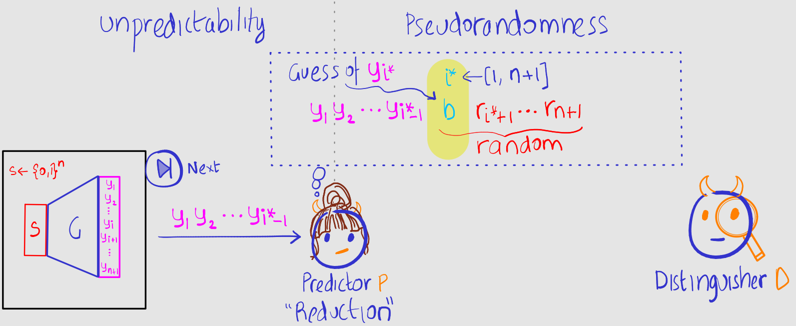
□

Unpredictability Implies Pseudorandomness

Theorem 2

If G is next-bit unpredictable, then it is a pseudorandom.

Proof Sketch. ∃ predictor P for G ⇐ ∃ distinguisher D for G.



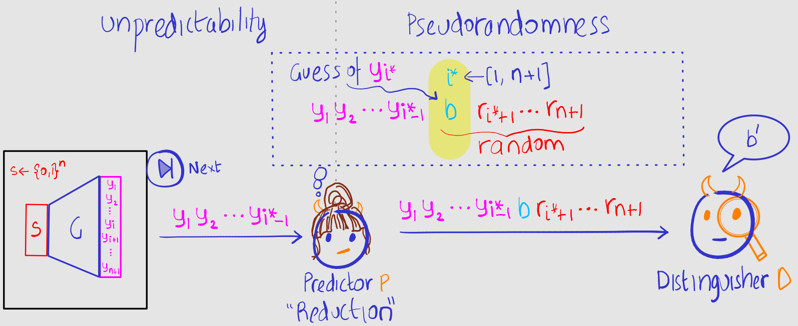
□

Unpredictability Implies Pseudorandomness

Theorem 2

If G is next-bit unpredictable, then it is a pseudorandom.

Proof Sketch. $\exists$ predictor P for G $\Leftarrow$ $\exists$ distinguisher D for G.



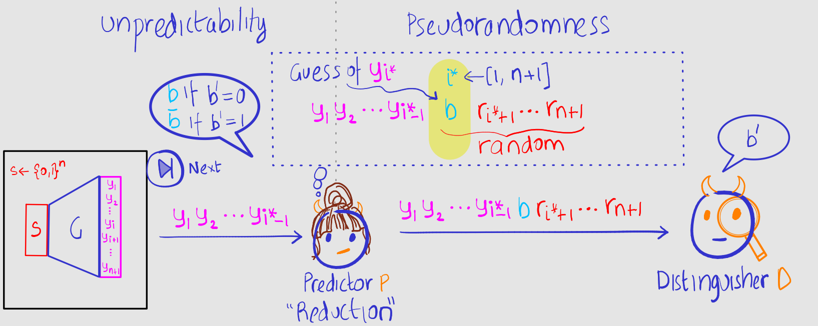
$\square$

Unpredictability Implies Pseudorandomness

Theorem 2

If G *is next-bit unpredictable, then it is a pseudorandom.*

Proof Sketch. $\exists$ predictor P for G $\Leftarrow$ $\exists$ distinguisher D for G.



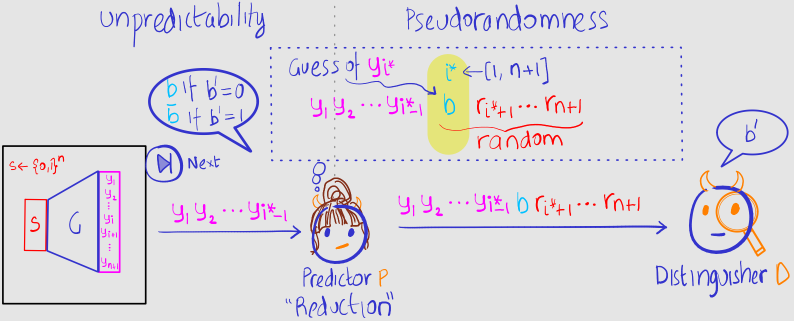
□

Unpredictability Implies Pseudorandomness

Theorem 2

If G is next-bit unpredictable, then it is a pseudorandom.

Proof Sketch. $\exists$ predictor P for G $\Leftarrow$ $\exists$ distinguisher D for G.



Analysis: If $b = y_{i^*}$ then D more likely to output 0 (claim 2).

$\square$

Plan for this Lecture

Integer Factoring...

- Given a integer N, find a factor p that divides N

Integer Factoring...

- Given a integer N, find a factor p that divides N
- Let's try to sample hard-to-factor integer N
  - Let's start with *random* integer N?

Integer Factoring...

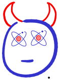
- Given a integer N, find a factor p that divides N
- Let's try to sample hard-to-factor integer N
  - Let's start with *random* integer N?
  - What about a random *odd* integer N?

Integer Factoring...

- Given a integer N, find a factor p that divides N
- Let's try to sample hard-to-factor integer N
  - Let's start with *random* integer N?
  - What about a random *odd* integer N?
  - $\cdots$
  - Pick two large random primes p and q and set $N = pq$
    - *Factoring assumption*: the probability with which any PPT adversary factors N sampled as above is negligible
    - Believed to be hardest instances to factor: best known factoring algorithms require *sub-exponential* time

Integer Factoring...

- Given a integer N, find a factor p that divides N
- Let's try to sample hard-to-factor integer N
  - Let's start with *random* integer N?
  - What about a random *odd* integer N?
  - ...
  - Pick two large random primes p and q and set $N = pq$
    - *Factoring assumption*: the probability with which any PPT adversary factors N sampled as above is negligible
    - Believed to be hardest instances to factor: best known factoring algorithms require *sub-exponential* time



  - Assumption does not hold against quantum adversaries! Shor's algorithm computes factors in quantum polynomial time

Exercise 4

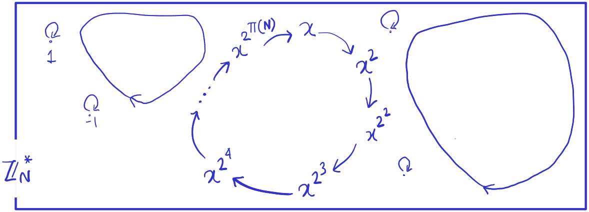
Show that taking square roots modulo N allows you to factor N

Integer Factoring...

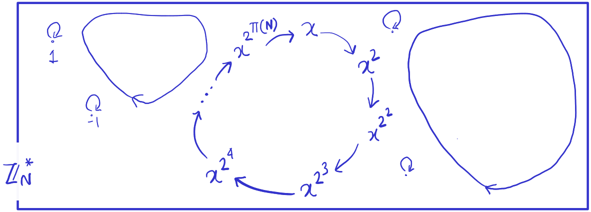
- We're interested in cycle structure of $\mathbb{Z}_N^*$, the multiplicative group of integers modulo N
  - $\mathbb{Z}_N^* := \{0 < x < N : GCD(x, N) = 1\}$

Integer Factoring...

- We're interested in cycle structure of $\mathbb{Z}_N^*$, the multiplicative group of integers modulo N
  - $\mathbb{Z}_N^* := \{0 < x < N : GCD(x, N) = 1\}$
- Let's consider the squaring map: $x \mapsto x^2 \bmod N$



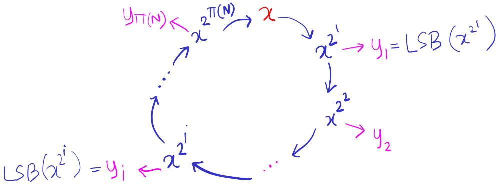
- We're interested in cycle structure of $\mathbb{Z}_N^*$, the multiplicative group of integers modulo N
  - $\mathbb{Z}_N^* := \{0 < x < N : GCD(x, N) = 1\}$
- Let's consider the squaring map: $x \mapsto x^2 \bmod N$



Exercise 5

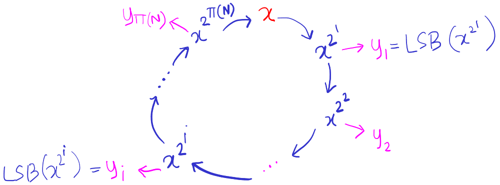
Show that the squaring map cycles, and has super-polynomially-long period π (with overwhelming probability)

- Given a random square (quadratic residue) as seed, square in each step and output the LSB (or parity bit)



The Squaring (Blum-Blum-Shub) Generator

- Given a random square (quadratic residue) as seed, square in each step and output the LSB (or parity bit)



Theorem 3 (Blum, Blum and Shub'84, Vazirani-Vazirani'82)

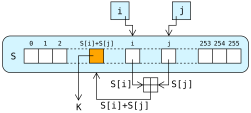
Assuming factoring (Blum) integers is hard, the squaring generator is unpredictable (on the left).

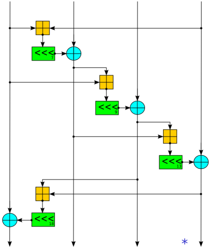
- Intuition: Why is the sequence unpredictable?
  - Non-linear operation in each step (linearity can be exploited)
  - Taking square root is hard (Exercise 4)
  - Period of the cycle hidden (which can be exploited: e.g., LFSR)

Unpredictability in the Wild

- The squaring generator is provably unpredictable, but is inefficient in practice

- The squaring generator is provably unpredictable, but is inefficient in practice
- In practice *stream ciphers* like ChaCha and Salsa20 are used
  - Non-linear Boolean operations
  - Cryptanalysis instead of security proof
  - Drawback: sometimes broken (e.g., RC4)





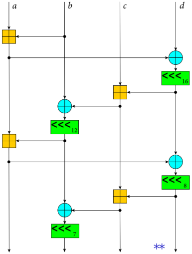
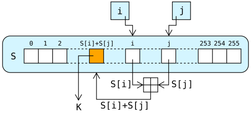
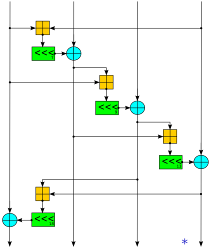


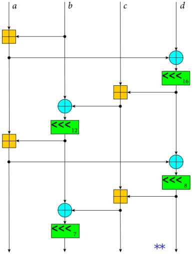
Image credit: Wikipedia (*Sissssou) (**Tony Arcieri)

Unpredictability in the Wild

- The squaring generator is provably unpredictable, but is inefficient in practice
- In practice *stream ciphers* like ChaCha and Salsa20 are used
  - Non-linear Boolean operations
  - Cryptanalysis instead of security proof
  - Drawback: sometimes broken (e.g., RC4)







- Salsa20 implemented in eStream, NACL, OpenSSL etc

Image credit: Wikipedia (*Sissssou) (**Tony Arcieri)

To Recap

- We saw an equivalent formulation of pseudorandomness via unpredictability
- Described construction of an unpredictable sequence under factoring assumption
  - Actually yields PRG of arbitrary stretch
- Saw how length-extension for PRG works
  - Reduces task to constructing PRG that stretches by single bit
  - Modular design always useful: will re-use theorem in Lecture 6
  - Proof technique: hybrid argument!

References

1. [Gol01, §3.3] for a formal proof of Theorems 1 and 2
2. Next-bit unpredictability was introduced in [BM84]. Yao introduced pseudorandomness [Yao82], and then proved its equivalence to unpredictability
3. The squaring pseudo-random generator was studied in [BBS86, VV84]
4. You can read about how PRGs are used for derandomisation in [AB09, Chapter 20]. This is also a great source for reading about complexity-theoretic (i.e., Nisan-Wigderson) PRG. Yao's result on derandomisation of BPP is from [Yao82].

Sanjeev Arora and Boaz Barak.
Computational Complexity - A Modern Approach.
Cambridge University Press, 2009.

L. Blum, M. Blum, and M. Shub.
A simple unpredictable pseudo-random number generator.
SIAM Journal on Computing, 15(2):364–383, 1986.

Manuel Blum and Silvio Micali.
How to generate cryptographically strong sequences of pseudo-random bits.
SIAM J. Comput., 13(4):850–864, 1984.

Oded Goldreich.
The Foundations of Cryptography - Volume 1: Basic Techniques.
Cambridge University Press, 2001.

Umesh V. Vazirani and Vijay V. Vazirani.
Efficient and secure pseudo-random number generation (extended abstract).
In *25th FOCS*, pages 458–463. IEEE Computer Society Press, October 1984.

Andrew Chi-Chih Yao.
Theory and applications of trapdoor functions (extended abstract).
In *23rd FOCS*, pages 80–91. IEEE Computer Society Press, November 1982.