Pseudorandomness

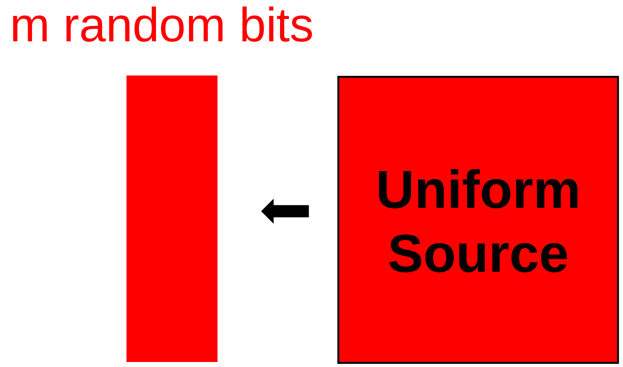
Benny Applebaum

Bar-Ilan Winter School, 2014

Randomness as a resource

Pure Randomness is

- Valuable, in fact, necessary for crypto
- But typically expensive

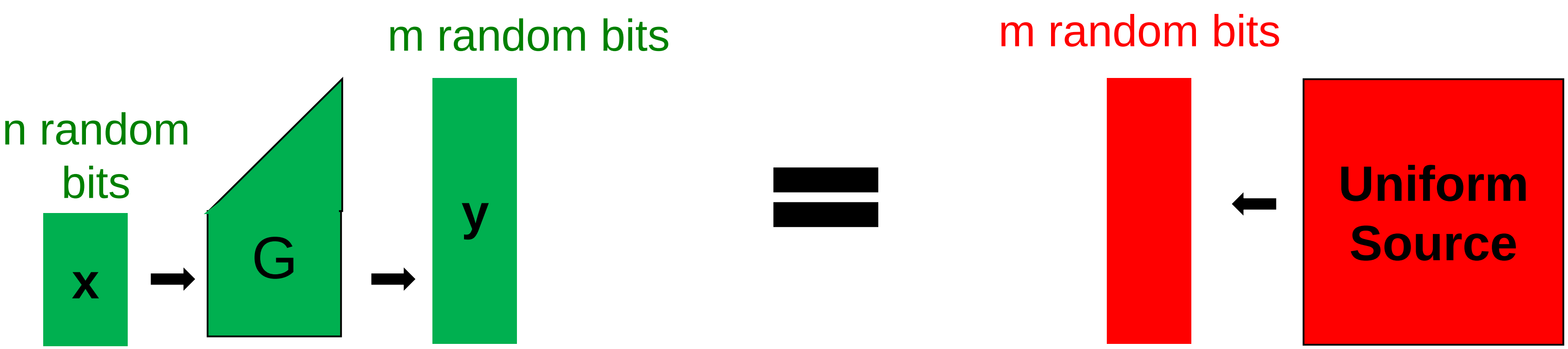


Goal: Given a short random string generate a long sequence of random bits?

Generating randomness

G is a deterministic efficient function

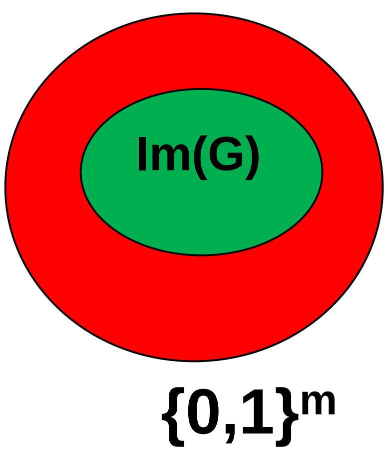
short random seed $\rightarrow$ **long** random string



Impossible !

The image of G consists of 2^n strings

$\Rightarrow$ doesn't cover all possible 2^m strings

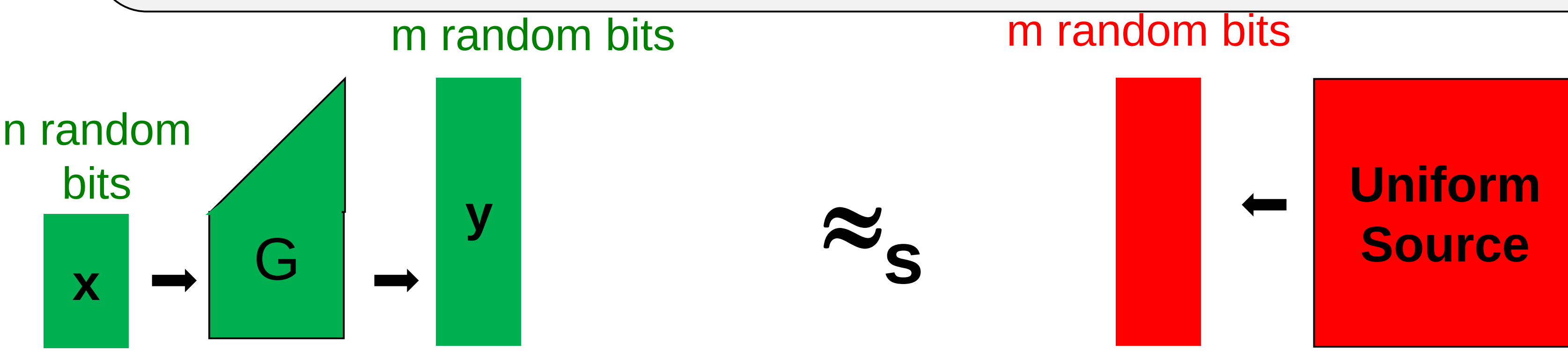


Generating randomness (relaxation I)

Output is **Statistically-Close** to uniform:

For every event **A,**

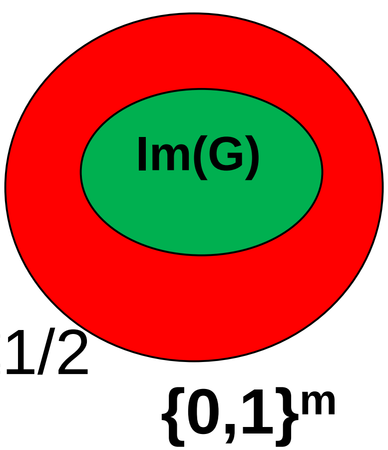
$$\mathbf{Pr_x[A(G(x))] = Pr[A(Uniform)] \pm negligible(n)}$$



Still Impossible !

Let A(y) be the event $y \in Im(G)$

Then $Pr[A(G(x))]=1$ but $Pr[A(uniform)] \leq 2^n/2^m \leq 1/2$

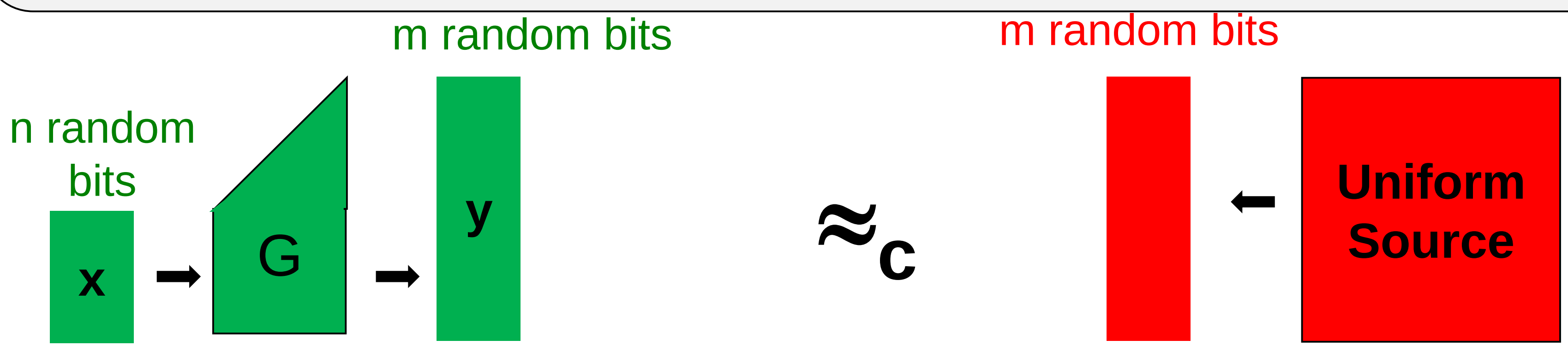


Generating randomness (relaxation II)

Output is **Computationally-Close** to uniform (**pseudorandom**):

For every **efficiently computable** event **A,**

$$\mathbf{Pr_x[A(G(x))] = Pr[A(Uniform)] \pm negligible}$$



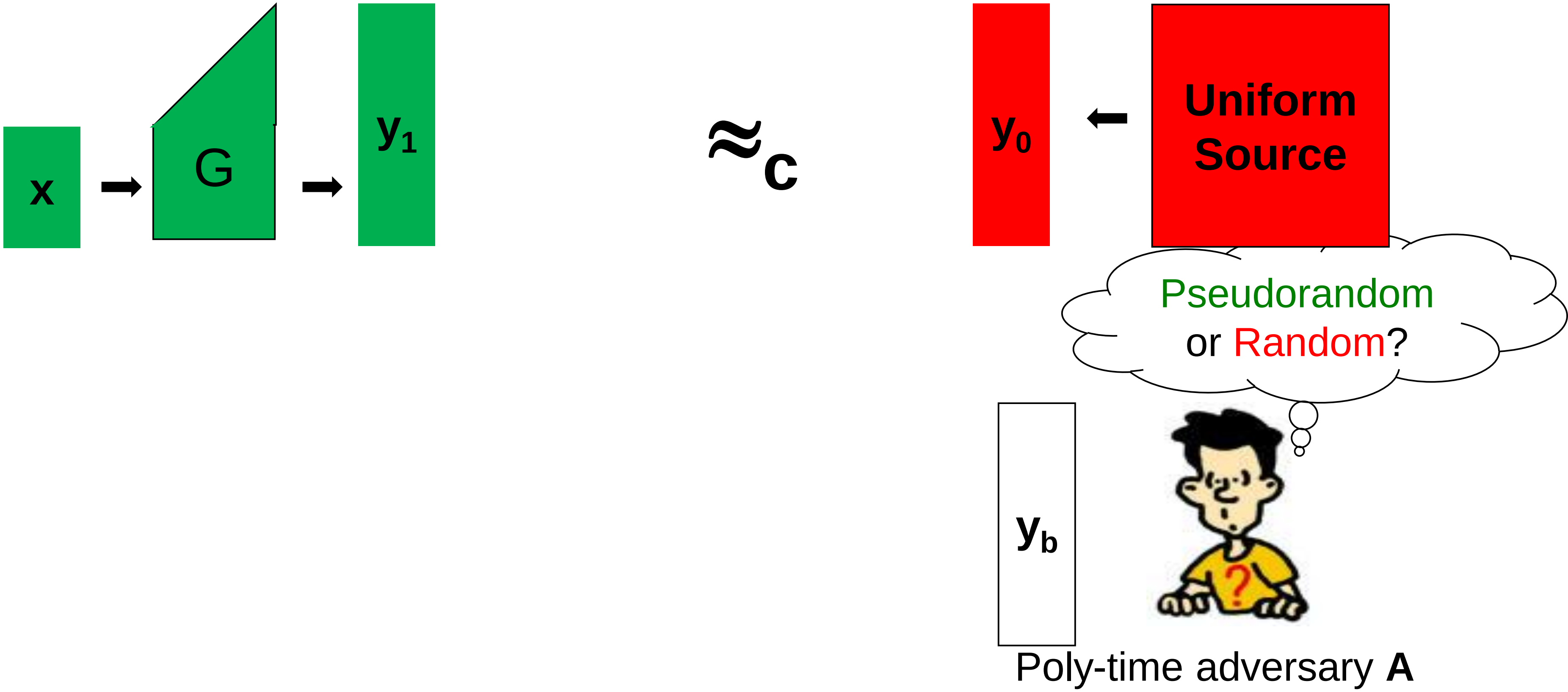
Observations:

- Strict relaxation of statistical closeness
- Must be computationally hard to decide if $y \in$ Image(G)
- In fact, G must be one-way (Exercise)
- WLOG, require $\mathbf{Pr_x[A(G(x))]-Pr[A(Uniform)] < neg}$

Alternative view: Indistinguishability

- The adversary **A** is given $\mathbf{y_b}$ where $\mathbf{b \leftarrow \{0,1\}}$
- **A** outputs a guess bit **b'** and **wins** if **b'=b**

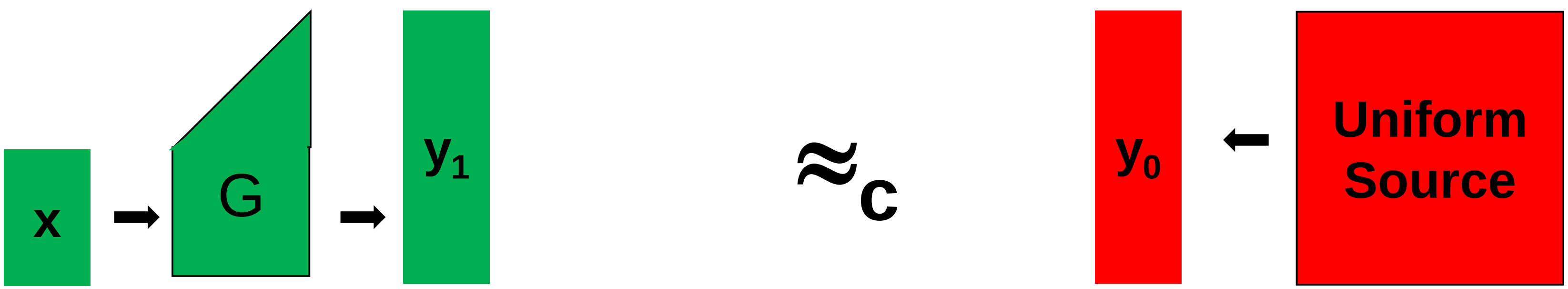
Claim: G is **pseudorandom** iff **Pr[win]<1/2+ neg**



Alternative view: Indistinguishability

- The adversary **A** is given $\mathbf{y_b}$ where $\mathbf{b \leftarrow \{0,1\}}$
- **A** outputs a guess bit **b'** and **wins** if **b'=b'**

Claim: G is **pseudorandom** iff **Pr[win]<1/2+ neg**



$$\Pr[\text{win}] = \Pr[A(\mathbf{y_1})=1]*\Pr[b=1] \quad + \quad \Pr[A(\mathbf{y_0})=0]*\Pr[b=0]$$

$$= \tfrac{1}{2}\big(\Pr[(A(\mathbf{y_1})=1]+\Pr[A(\mathbf{y_0})=0]\big)$$

$$= \tfrac{1}{2}\big(\Pr_x[A(\mathrm{PRG}(x))]+1-\Pr[A(\mathbf{U_m})]\big)$$

$$= \tfrac{1}{2}+\tfrac{1}{2}\big(\Pr_x[A(\mathrm{PRG}(x))]-\Pr[A(\mathbf{U_m})]\big) < \tfrac{1}{2}+\text{neg}$$

Properties

Pseudorandomness is preserved under multiple samples

Proof by reduction to a single instance.

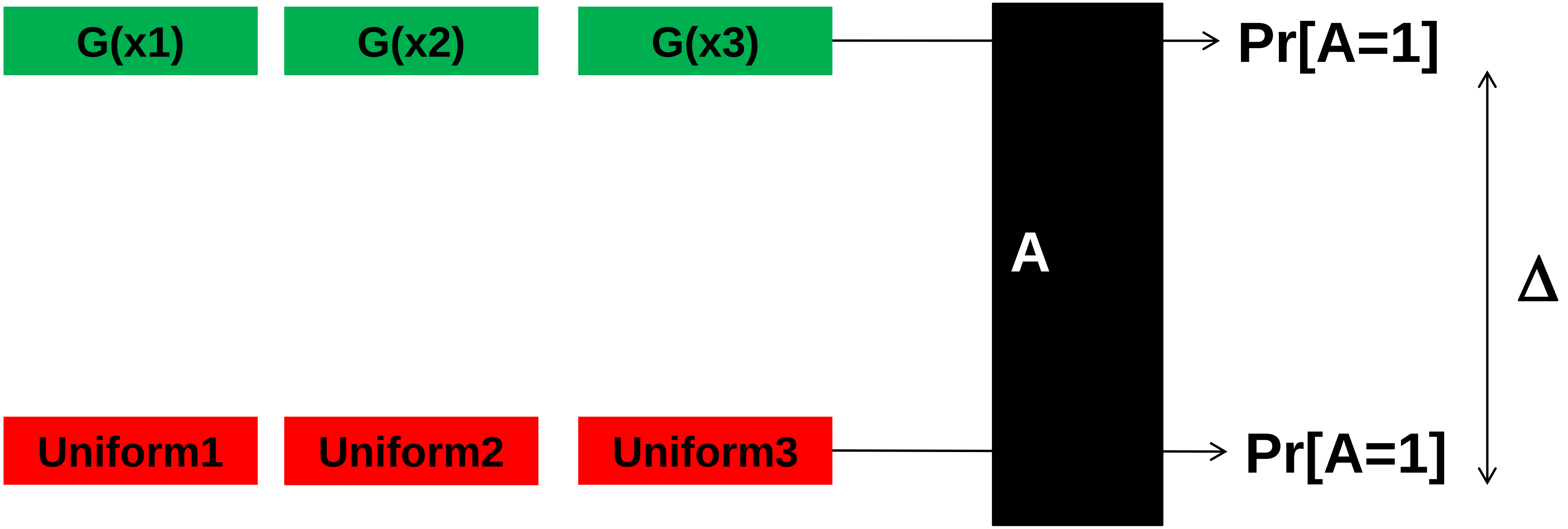
G(x1) **G(x2)** **G(x3)**

Uniform1 **Uniform2** **Uniform3**

Pseudorandomness is preserved under multiple samples

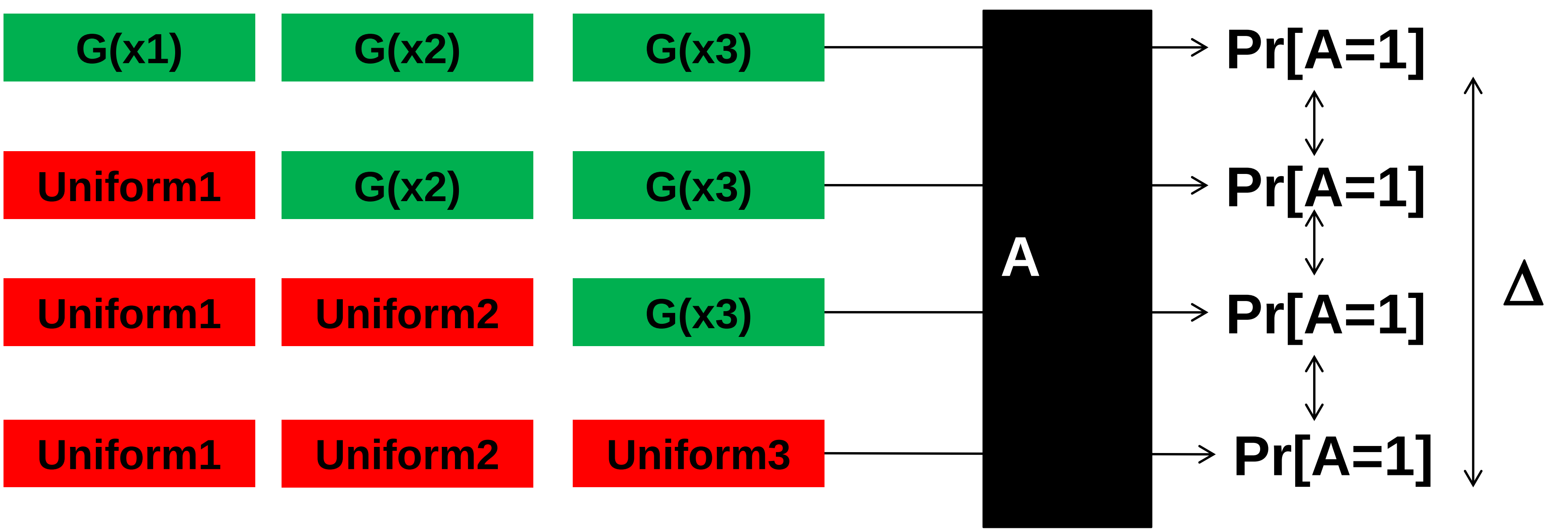
Assume a multiple-samples adversary **A**

Goal: Construct a single-instance adversary **B**



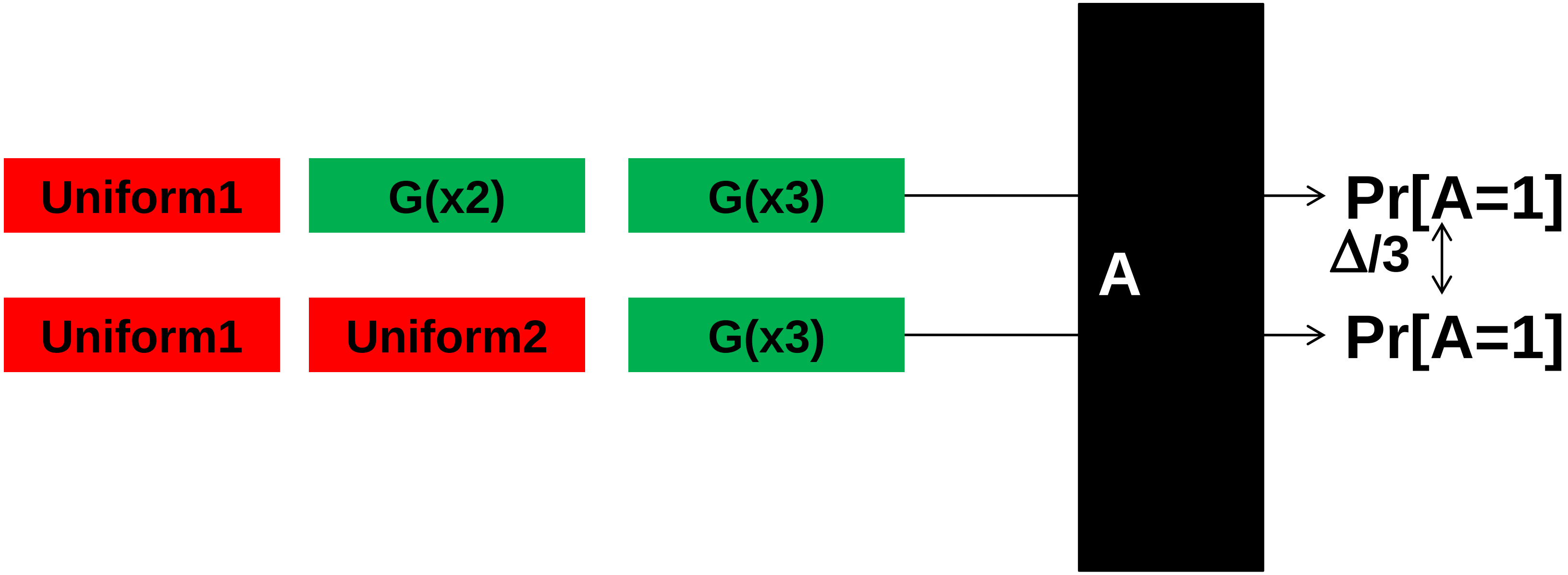
Pseudorandomness is preserved under multiple samples

There must be two neighboring hybrids with gap >Δ/3



Pseudorandomness is preserved under multiple samples

There must be two neighboring hybrids with gap >Δ/3

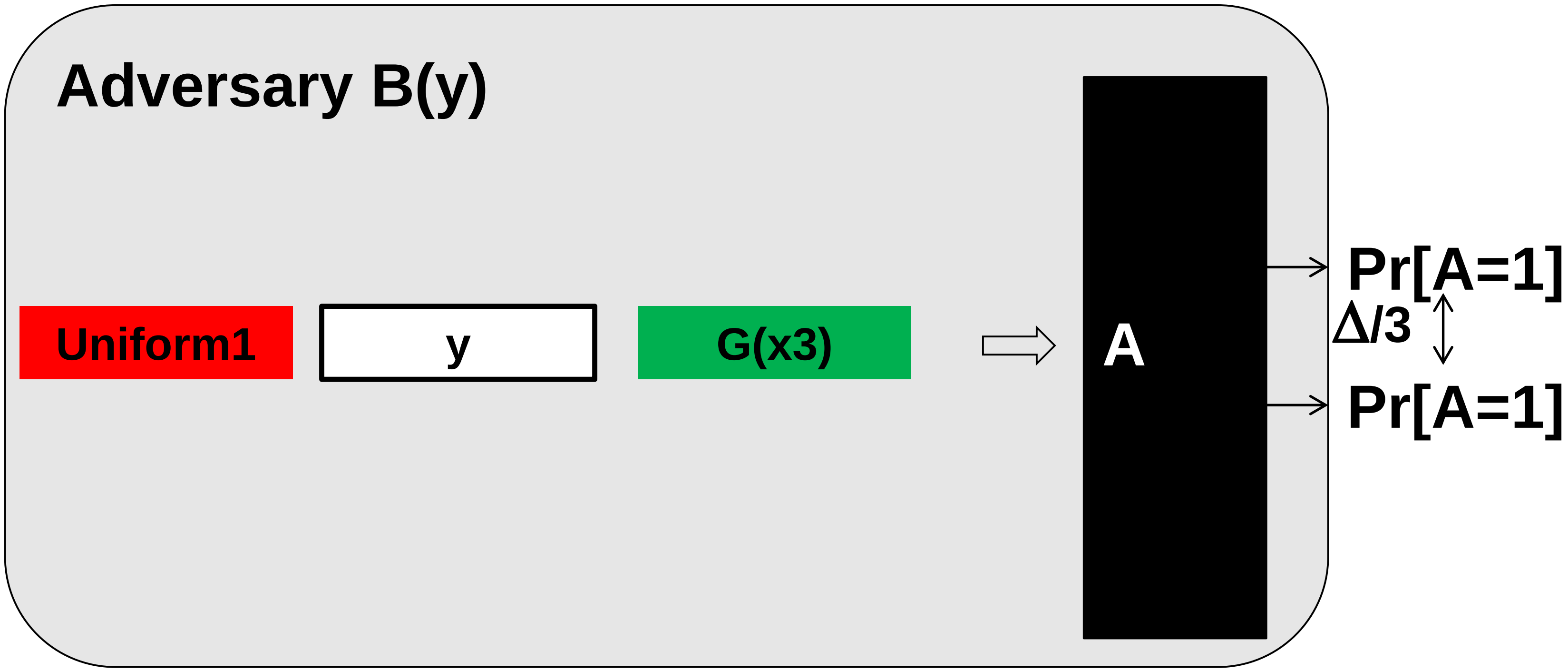


Pseudorandomness is preserved under multiple samples

B(y): Plant y in the changing point and call A.

$\Rightarrow \Pr_x[B(PRG(x))=1]-\Pr[B(Random)=1]>\Delta/3$

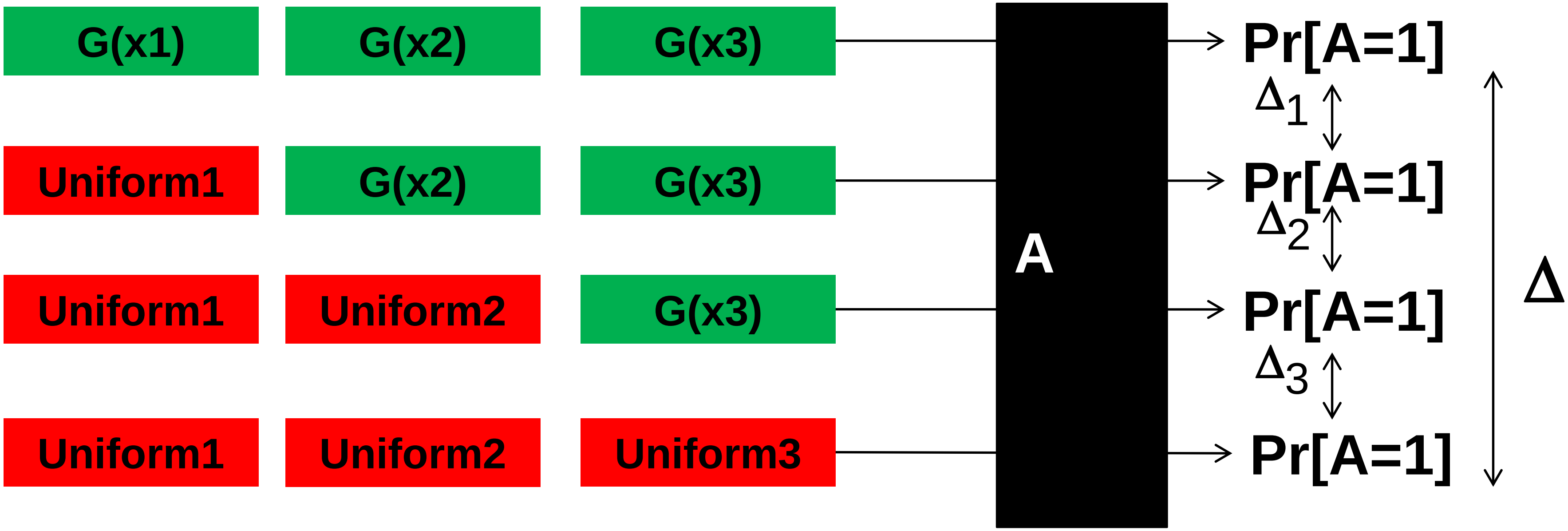
$\Rightarrow$ Contradicting the security of the PRG!



How to find a good pair of hybrids ?

Observation: the **average gap** $\sum\Delta_i/t \geq \Delta/t$

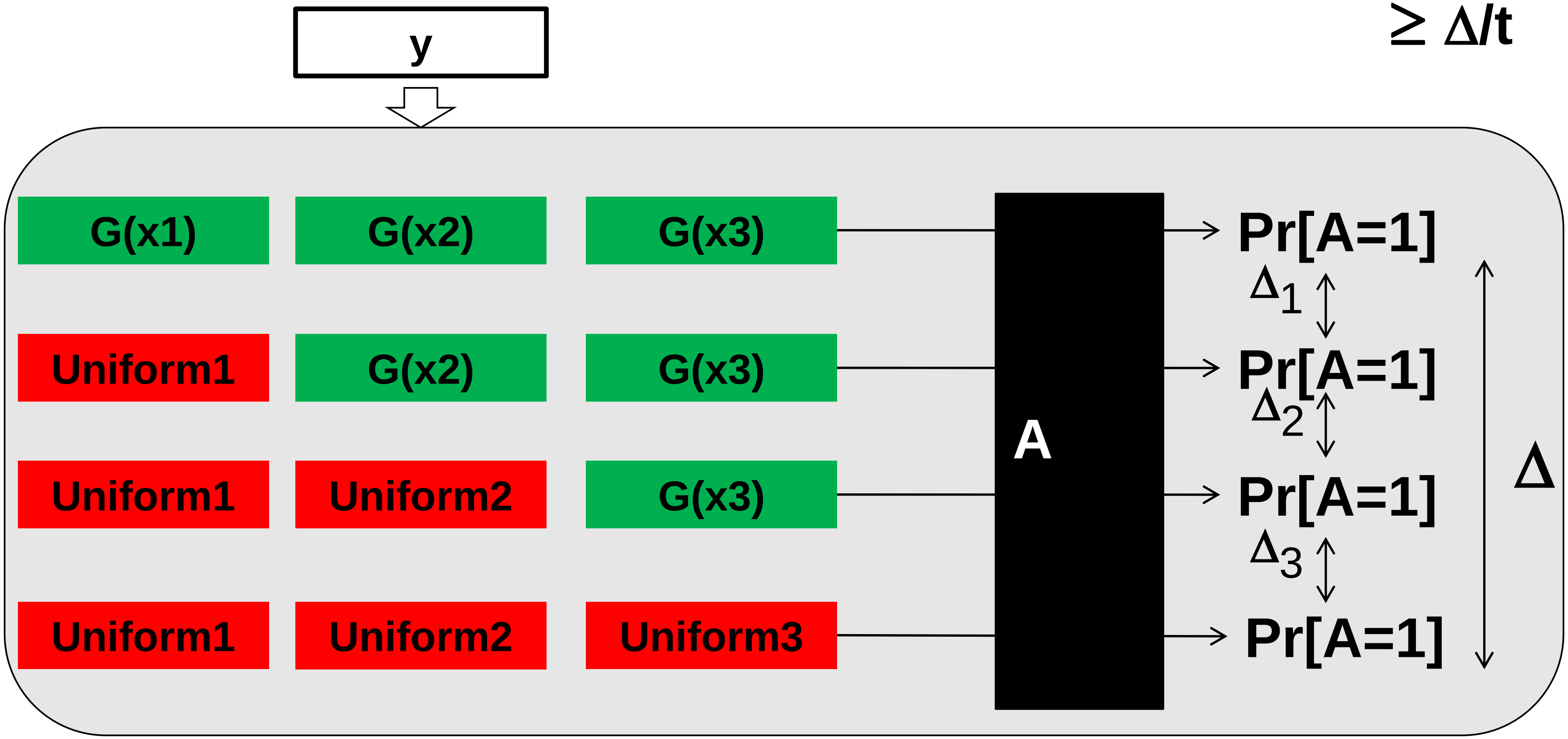
Idea: Let **B** Choose a **random** pair



How to find good pair of hybrids ?

B(y): Choose a **random** hybrid, plant y in the changing point and call A

Ex: Prove $\Pr_x[B(\text{PRG}(x))=1]-\Pr[B(\text{Random})=1] = \sum \Delta_i / t \geq \Delta / t$



The Hybrid method

Goal: $\mathbf{X} \approx \mathbf{Y}$ for some complicated distributions

- Define a sequence of **poly**-many hybrids $\mathbf{H_0,\ldots,H_t}$
- $\mathbf{H_0=X}$ and $\mathbf{H_t=Y}$
- $\mathbf{H_i \approx_c H_{i+1}}$ typically by simple argument
- Conclude that $\mathbf{X}=\mathbf{H_0} \approx_c \mathbf{H_t=Y}$

An extremely powerful technique in crypto

Formal Definitions

- Let **X** and **Y** be a probability distributions over $\{0,1\}^n$
- Let **A**:$\{0,1\}^n \rightarrow \{0,1\}$ be an adversary (distinguisher)

The **distinguishing gap** is defined by

$$\Delta_A(X,Y) = |\Pr[A(X)=1]-\Pr[A(Y)=1]|$$

A pair of distribution ensembles $\mathbf{X=\{X_n\}}$ and $\mathbf{Y=\{Y_n\}}$ are **computationally indistinguishable**, $\mathbf{X \approx_c Y}$, if for every PPT **A,**

$$\Delta_A(X_n,Y_n) < \text{neg}(n).$$

A deterministic efficient function G is a **PRG** if:

1. G **expands** n-bits to m-bits where m(n)>n.
2. $\mathbf{\{G(U_n)\} \approx_c \{U_{m(n)}\}}$

Useful facts

Indistinguishability behaves like a distance

- (Transitive) If **X** $\approx_c$ **Y** and **Y** $\approx_c$ **Z** then **X** $\approx_c$ **Z**

Proof: $\Delta_A(X,Z) \leq \Delta_A(X,Y) + \Delta_A(Y,Z)$, for every **A**

Useful facts

Indistinguishability behaves like a distance

- (Transitive) If **X** $\approx_c$ **Y** and **Y** $\approx_c$ **Z** then **X** $\approx_c$ **Z**

- (Preserved under efficient computations):

  If **X** $\approx_c$ **Y** then **F(X)** $\approx_c$ **F(Y)** where **F is PPT**

Proof: (contra positive)

Assume $\Delta_A(F(X),F(Y))$ is non-negligible for some PPT **A**

Define a new PPT adversary **B=A°F** then

$\Delta_B(X,Y) = \Delta_A(F(X),F(Y))$ is non-negligible $\Rightarrow$ contradiction.

Useful facts

Indistinguishability behaves like a distance

- (Transitive) If $\mathbf{X} \approx_c \mathbf{Y}$ and $\mathbf{Y} \approx_c \mathbf{Z}$ then $\mathbf{X} \approx_c \mathbf{Z}$

- (Preserved under efficient computations):

  If $\mathbf{X} \approx_c \mathbf{Y}$ then $\mathbf{F(X)} \approx_c \mathbf{F(Y)}$ where **F is PPT**

- (Preserved under ind. samples)

  For efficiently samplable **X,X',Y,Y'** If $\mathbf{X} \approx_c \mathbf{X'}$ and $\mathbf{Y} \approx_c \mathbf{Y'}$ then $\mathbf{(X,Y)} \approx_c \mathbf{(X',Y')}$

Pf: Hybrid argument (as we saw)

Constructions

PRGs from One-Way Functions

Thm. [Hastad-Impagliazzo-Levin-Luby 1990]

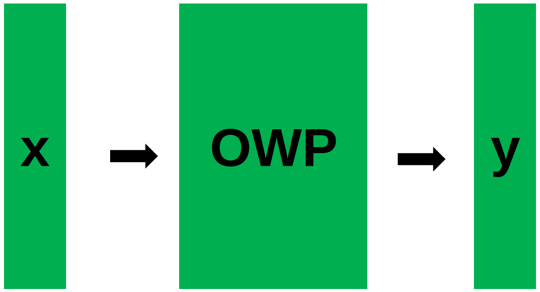
If one-way functions exist, then there are pseudorandom generators.

- Recall that the converse direction also holds.
- Fundamental theorem: “PRGs are feasible”
- Complicated and beautiful proof with many important concepts (randomness extractors, pseudoentropy,…).
- We will see a proof of a weaker theorem that builds PRGs from **one-way permutations**.

PRGs from One-Way Permutations

Recall that a **one way permutation** is a **bijection** over $\{0,1\}^n$

which is **easy-to-compute** but **hard-to-invert**



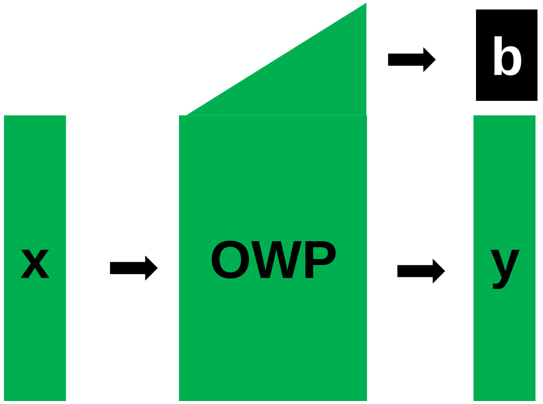
Good start: y is truly uniform

How to generate an extra pseudorandom bit?

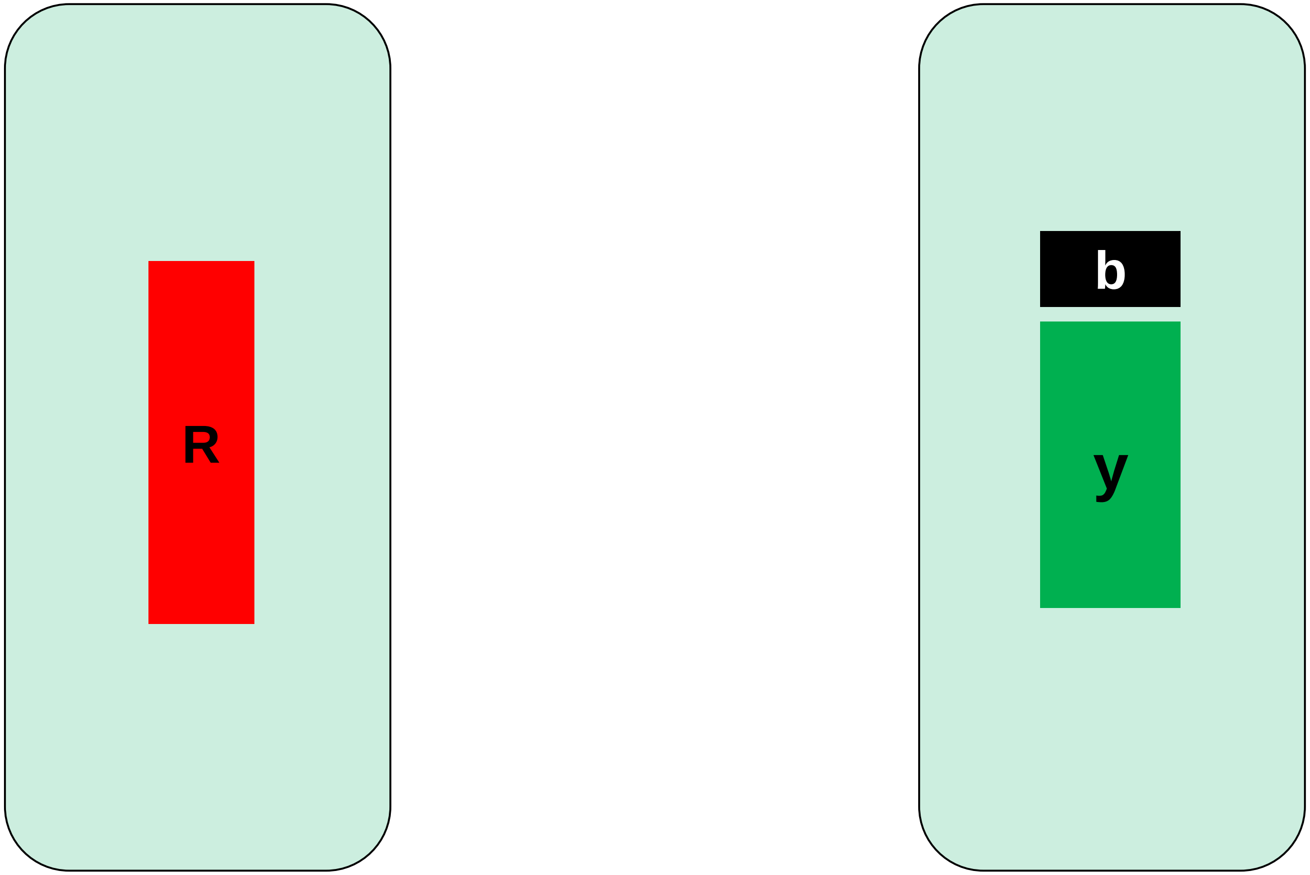
PRGs from One-Way Permutations

Thm. Let b(x) be a hard-core bit of the OWP.

Then the mapping $x \to (OWP(x), b(x))$ is PRG

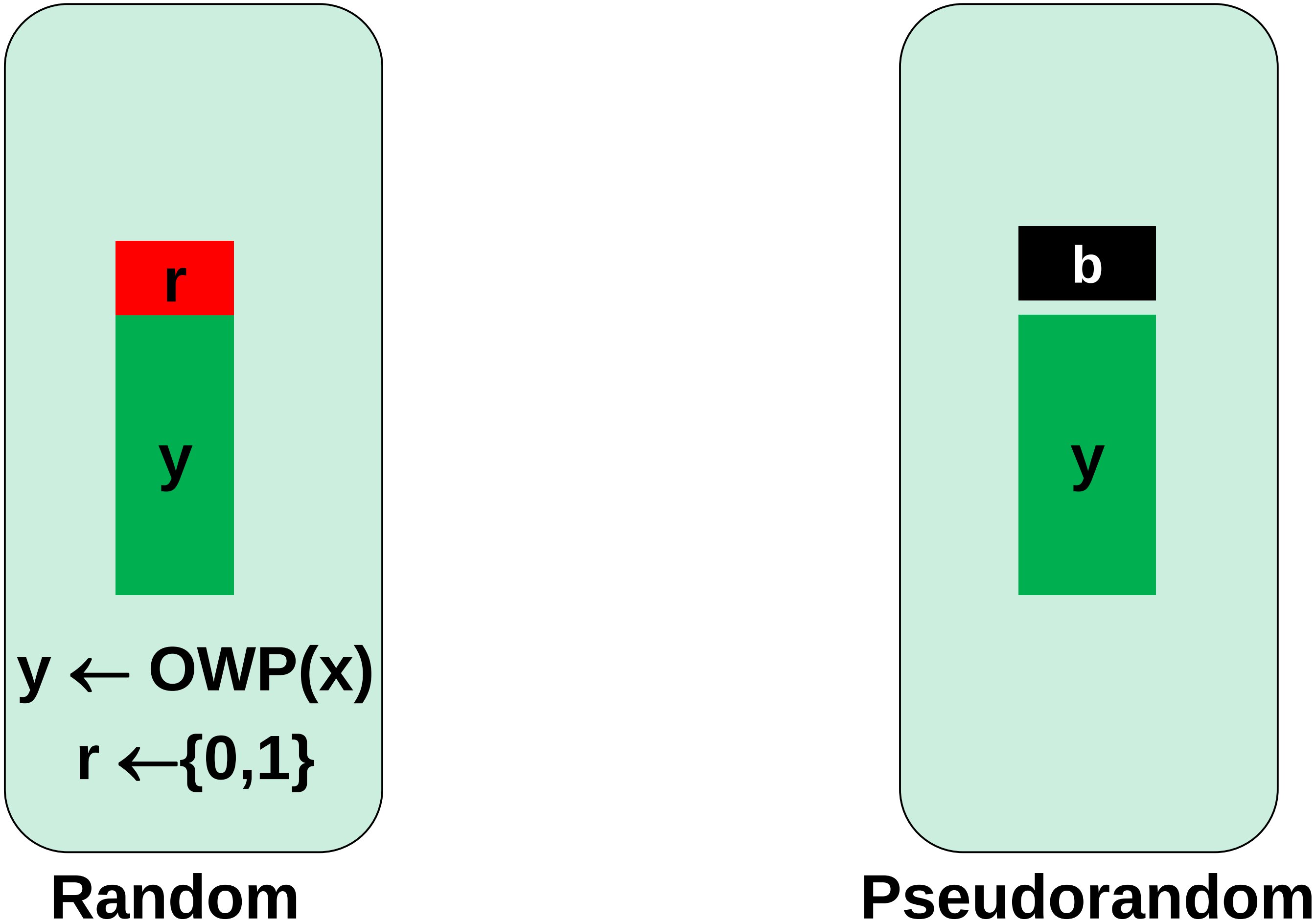


Goal: Prove indistinguishability

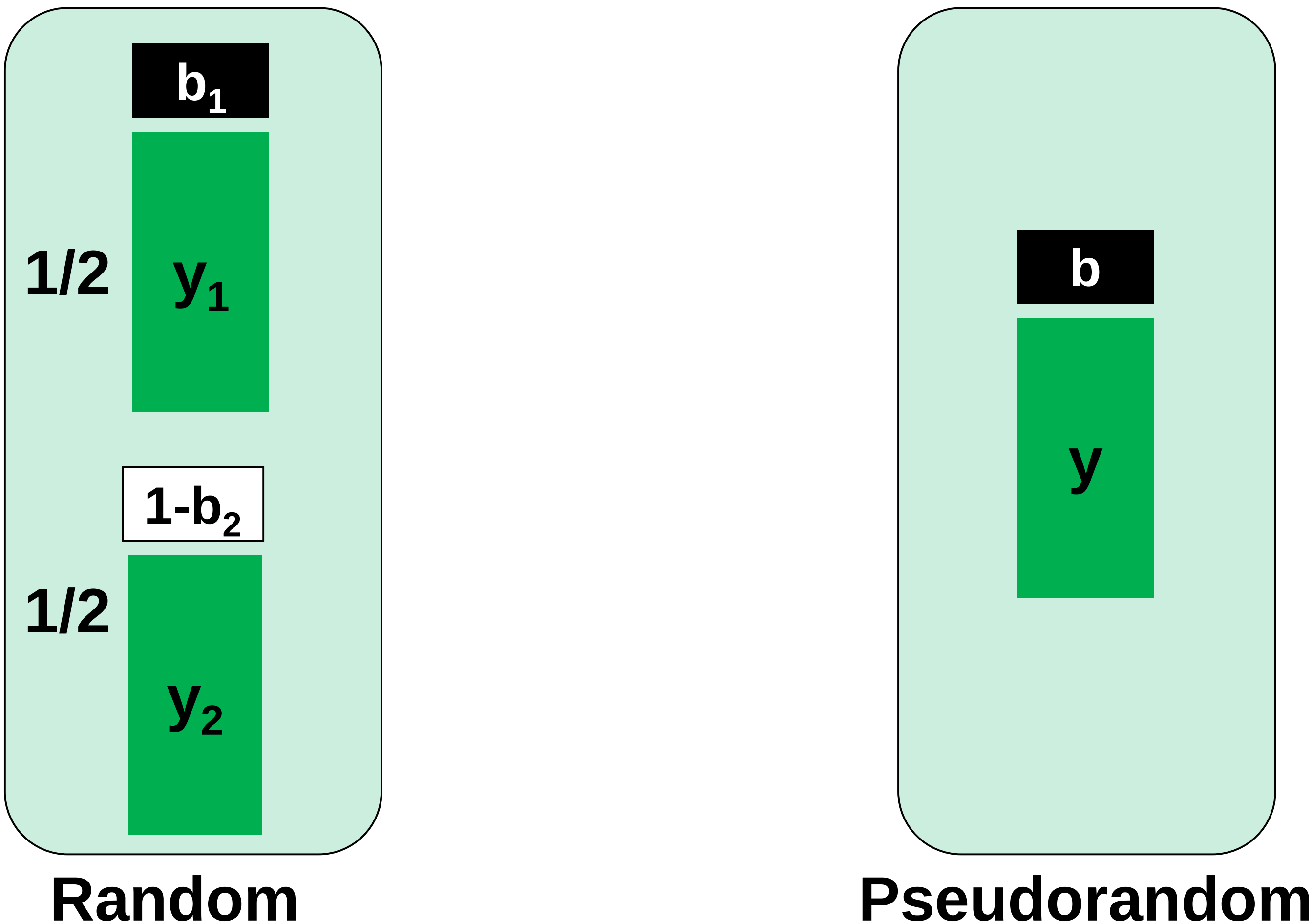


Random **Pseudorandom**

Goal: Prove indistinguishability

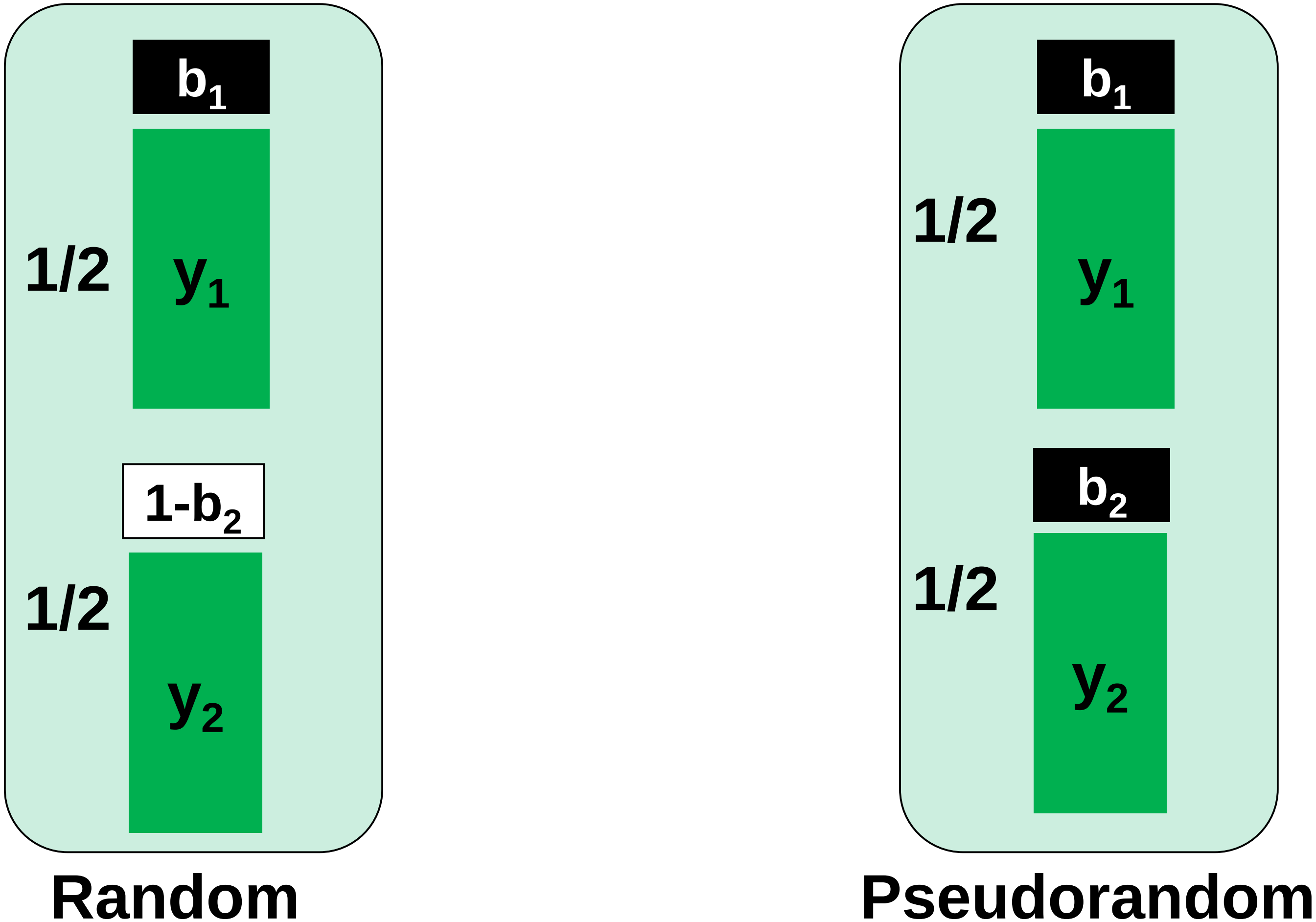


Goal: Prove indistinguishability



Goal: Prove indistinguishability

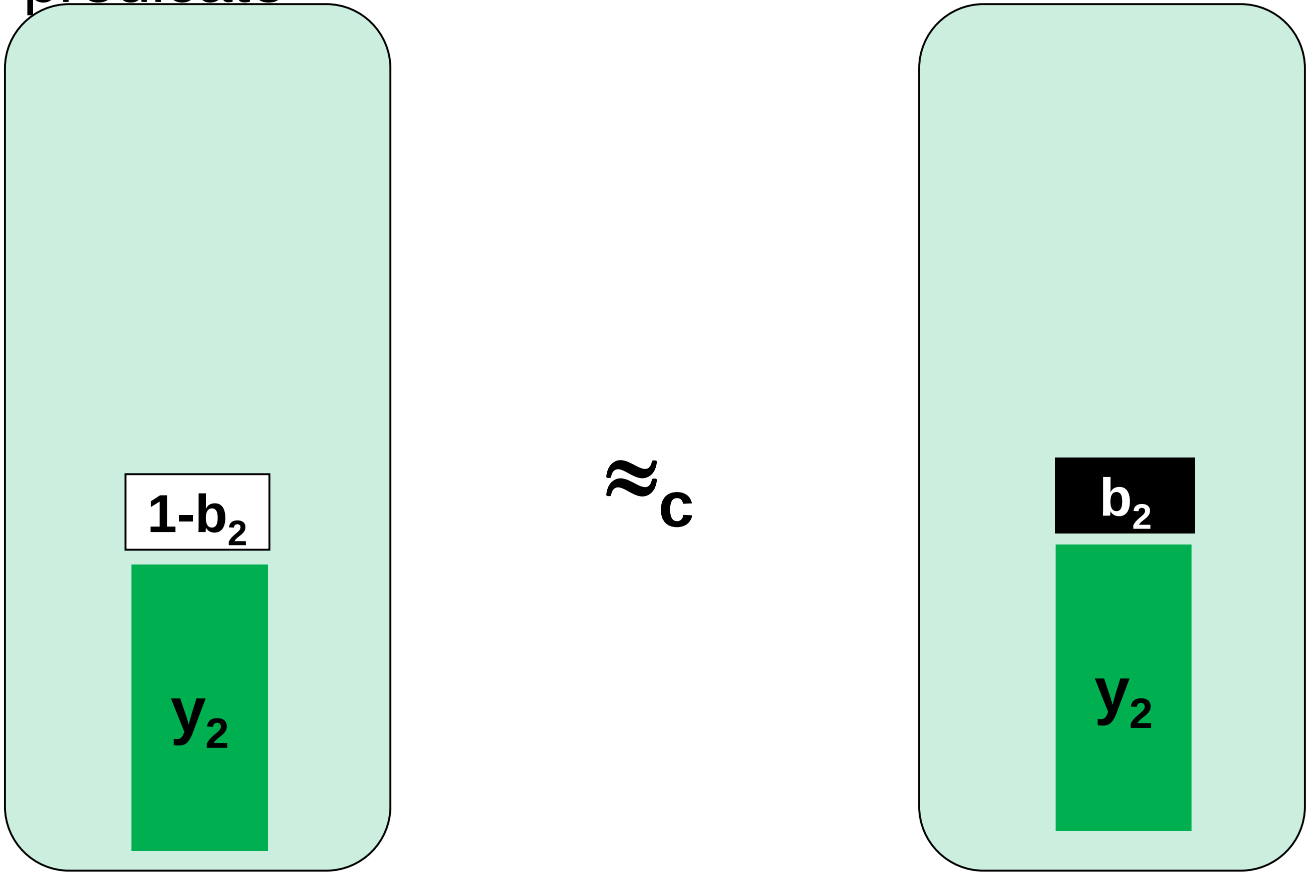
By **“useful fact”** it suffices to prove indistinguishability for



Goal: Prove indistinguishability

By **"useful fact"** it suffices to prove indistinguishability for

Indistinguishability follows immediately from the security of hardcore predicate

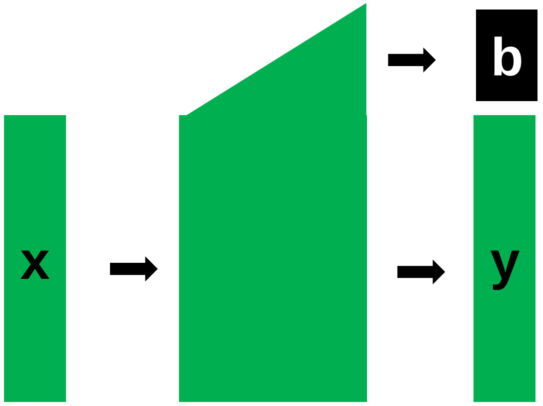


Expanding the Stretch

The length matters…

- PRG which stretches its input by a single-bit is not very useful…
- Can we expand the stretch?

Thm. A **PRG:$\{0,1\}^n \rightarrow \{0,1\}^{n+1}$** can be transformed into **PRG:$\{0,1\}^n \rightarrow \{0,1\}^{m(n)}$** for an arbitrary polynomial m(n)



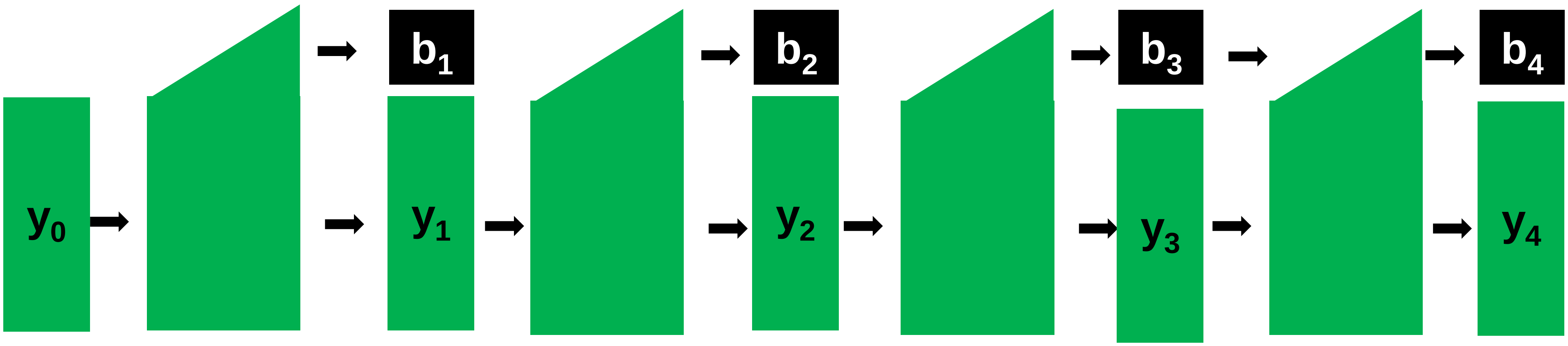
Expanding the stretch

NewPRG(y_0)

- For i=0 to m:

    - (y_{i+1}, b_{i+1})=PRG(y_i)

Output $\mathbf{b_1, \ldots, b_m}$



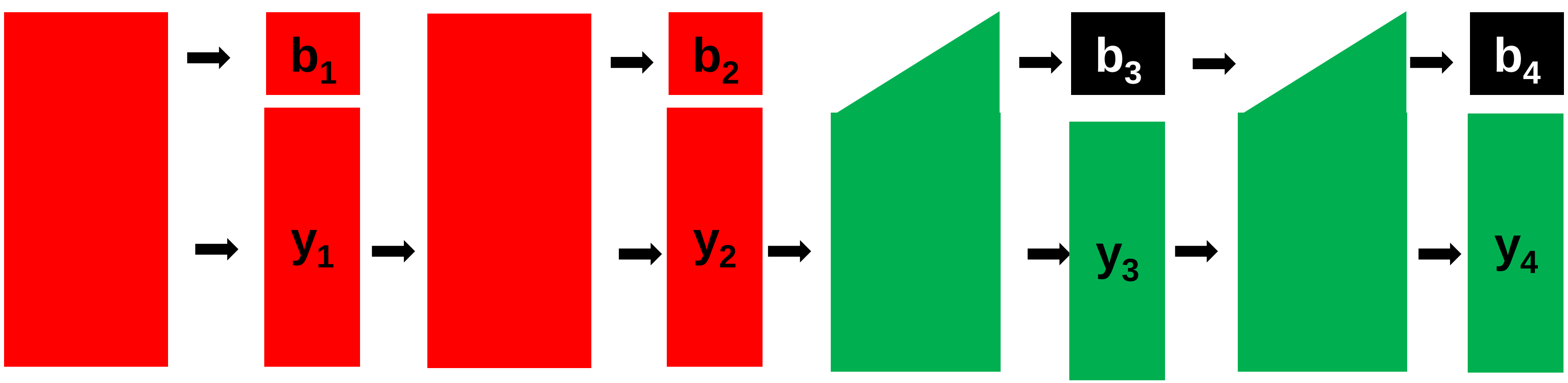
Proof via Hybrid Argument

Hybrid $\mathbf{H_k}$

- For i=0 to m:

  - $(y_{i+1}, b_{i+1}) = \begin{cases} \text{Random} & \text{if } i \le k \\ \text{PRG}(y_i) & \text{if } i > k \end{cases}$

Output $\mathbf{b_1, \ldots, b_m}$

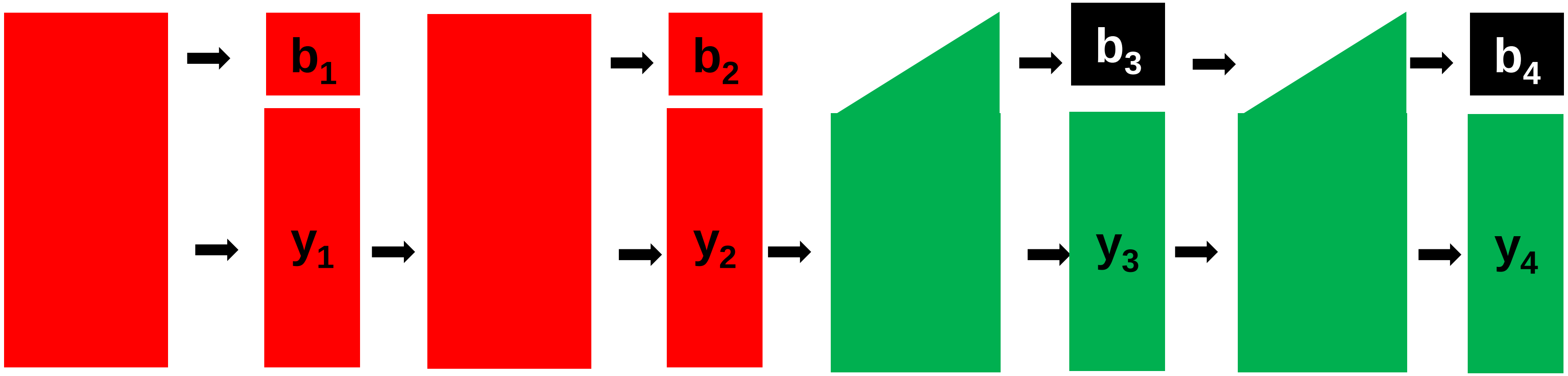


Proof via Hybrid Argument

H_0=NewPRG and **H_m=Random**

Assume **A**($b_1,\ldots,b_m$) distinguishes **H_0** from **H_m** with gap Δ

Transform **A** into a distinguisher **B**(y,b) for original PRG



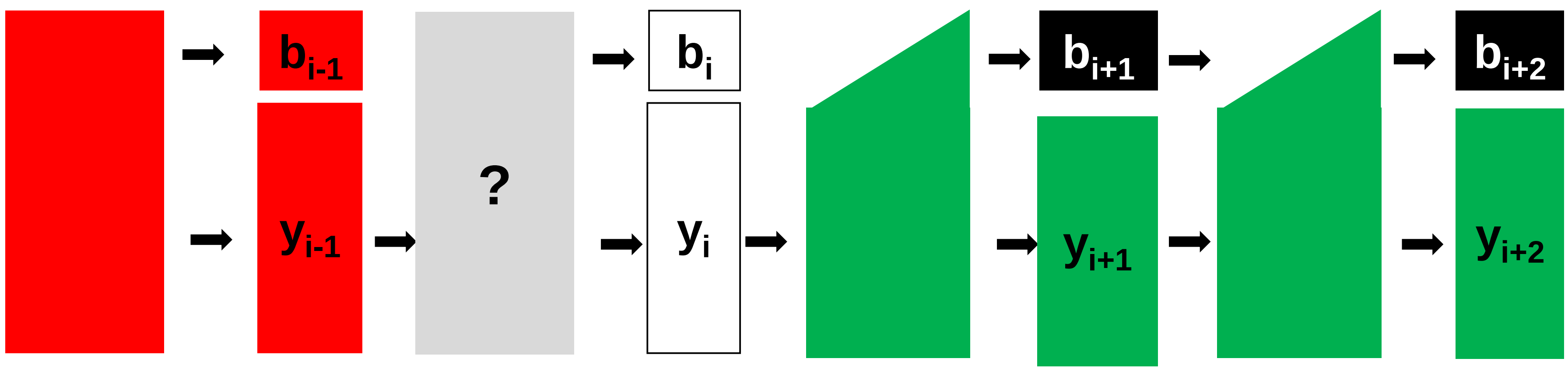
Breaking the original PRG

B puts challenge (y,b) in a random location i & calls **A**

Analysis: If (y,b) pseudorandom Pr[B=1]=Pr[A(H_{i-1})=1]

If (y,b) is random Pr[B=1]=Pr[A(H_i)=1]

$\Rightarrow$B's gap 1/m$\sum$(Pr[**A**(H_i)]-Pr[**A**(H_{i-1})])>Δ/m



Summary

PRGs generate long strings which are **indistinguishable** from **random** by **efficient** adversaries

- Extremely useful in crypto and complexity
- Can be constructed from any one-way function
- In practice, there are very efficient candidates with long stretch
- Computational Indistinguishability is a useful abstract notion with many friendly properties

Poly-time adversary **A**