Umans Complexity Theory Lectures

Pseudo-Random Generators (PRGs) for Cryptography & BPP

- Blum-Micali-Yao (BMY) PRG: using:
  - Goldreich-Levin hard bit
  - Yao's Lemma

Randomized complexity classes

- Model: probabilistic Turing Machine

  Deterministic TM with additional read-only tape containing "coin flips"

  "p.p.t" = probabilistic polynomial time

- **BPP** **(Bounded-error Probabilistic Poly-time)**

  $L \in$ **BPP** if there is a **p.p.t. TM M** taking input vector x and using **random bit vector r**:

$$x \in L \Rightarrow \Pr_r[M(x,r) \text{ accepts}] \geq 2/3$$

$$x \notin L \Rightarrow \Pr_r[M(x,r) \text{ rejects}] \geq 2/3$$

Randomized complexity classes

- Model: probabilistic Turing Machine

  Deterministic TM with additional read-only tape containing "coin flips"

  "p.p.t" = probabilistic polynomial time

- **BPP** **(Bounded-error Probabilistic Poly-time)**

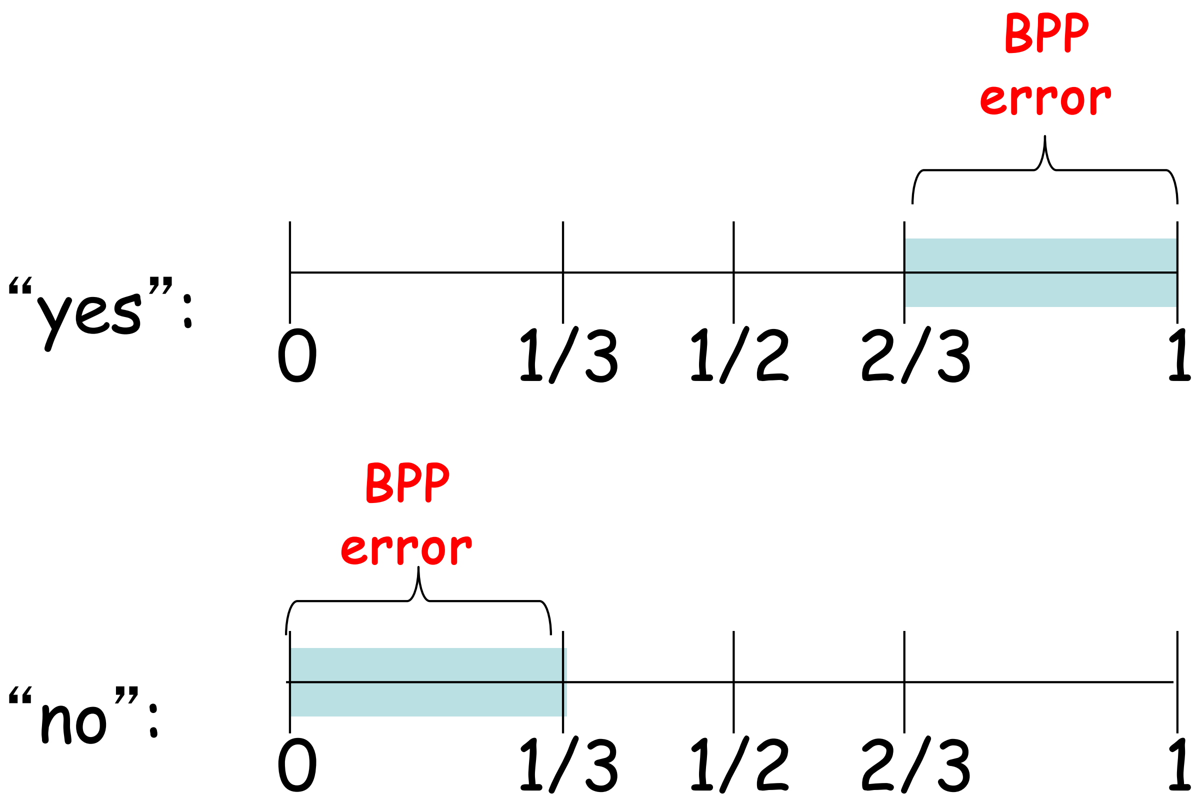
  $L \in$ **BPP** if there is a **p.p.t. TM M** taking input vector x and using **random bit vector r**:

$$x \in L \Rightarrow \Pr_r[M(x,r) \text{ accepts}] \geq 2/3$$

$$x \notin L \Rightarrow \Pr_r[M(x,r) \text{ rejects}] \geq 2/3$$

BPP Errors

- Error 1/3 on ”yes” or “no” answers:



BPP

- Next:

  Further explore the relationship between

  randomness

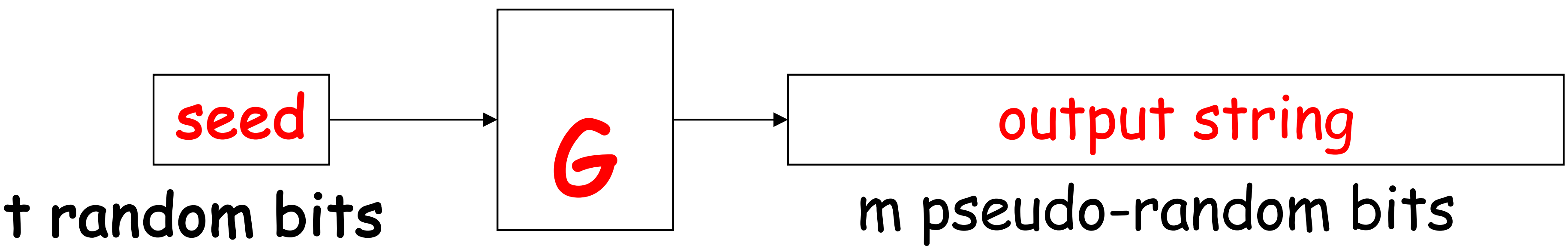
  and

  nonuniformity

- Main tool: **pseudo-random generators**

Derandomization

- Goal: try to simulate BPP in subexponential time (or better)
- Use **Pseudo-Random Generator (PRG) G**:



- PRG "good" if it passes (ad-hoc) statistical tests

Derandomization

- Ad-hoc statistical tests not good enough to prove BPP has non-trivial simulations
- Our requirements:

  - G is **efficiently computable (polynomial time)**

  - "**stretches**" **t** random bits into **m** pseudo random bits

  - "**fools**" small circuits: for all circuits C of size at most $\mathbf{s=m^c}$:

$$|\Pr_r[C(r) = 1] - \Pr_z[C(G(z)) = 1]| \leq \varepsilon$$

where r is random input, z is random seed

Simulating **BPP** using PRGs

- Recall: $L \in$ **BPP** implies exists p.p.t.TM M

  $x \in L \Rightarrow Pr_r[M(x,r) \text{ accepts}] \geq 2/3$

  $x \notin L \Rightarrow Pr_r[M(x,r) \text{ rejects}] \geq 2/3$

- Given an input x:

  convert **p.p.t. TM M** into **circuit C(x, r) where r is the random number input to circuit**

- **Hardwire input x to get circuit C_x**

  $x \in L \Rightarrow Pr_r[C_x(r) = 1] \geq 2/3$

  $x \notin L \Rightarrow Pr_r[C_x(r) = 1] \leq 1/3$

Simulating **BPP** using PRGs

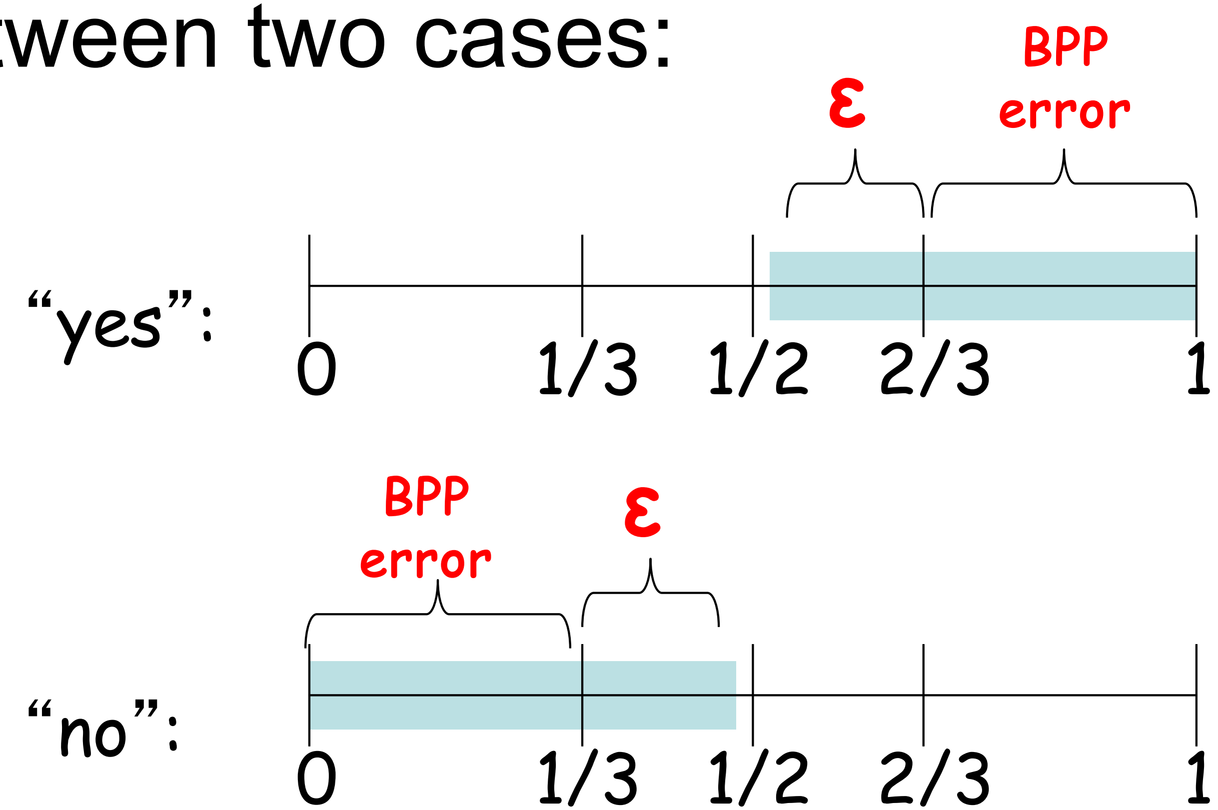
- Use a Pseudo Random-Number Generator (PRG) G with
  - **output length m**
  - **seed length t** $\ll$ m
  - **error ε** < 1/6
  - **fooling size s** = m
- Compute **$\Pr_z[C_x(G(z)) = 1]$** exactly
  - evaluate **$C_x(G(z))$** on **every seed z** $\in$ **$\{0,1\}^t$**

    (note: there are **2^t** possible seeds **z** $\in$ **$\{0,1\}^t$**)

- Total running time (O(m)+(time for G))**2^t**

Simulating **BPP** using PRGs

- Knowing **$\Pr_z[C_x(G(z)) = 1]$**, can distinguish between two cases:



- PRG error **ε < 1/6 added to BPP error < 1/3**

Blum-Micali-Yao (BMY) PRG

- **Initial goal:** for all $1 > \delta > 0$, we will build a family of PRGs $\{G_m\}$ with:

  output length m

  fooling size s = m

  seed length t = m^δ

  running time m^c

  error ε < 1/6
- Implies: **BPP** $\subseteq \cap_{\delta>0}$ **TIME(2^{n^δ})** $\underset{}{\overset{\text{proper subset}}{\subset}}$ **EXP**
- **Proof:** Need to enumerate all seeds $\mathbf{2^t}$
- of length **t =** m^δ. So simulation runs in time

  $(O(m)+(\text{time for } G_m))\mathbf{2^t}$

  $= O(m+m^c)(2^{m^\delta}) \leq O(2^{m^{2\delta}})$

Blum-Micali-Yao (BMY) PRG

- PRGs of this type imply existence of **one-way-functions**
- We will use widely believed **cryptographic assumptions**:

Definition: **One Way Function (OWF)** is a function family $f = \{f_n\}$ where:

(a) Each $f_n : \{0,1\}^n \rightarrow \{0,1\}^n$

(b) Easy to Compute: Each f_n is computable in poly(n) time,

(c) **Hard to Invert:** For every family of **poly-size circuits $\{C_n\}$**

$$\Pr_x[C_n(f_n(x)) \in f_n^{-1}(f_n(x))] \leq \varepsilon(n)$$

where $\varepsilon(n) = o(n^{-c})$ for any constant c.

Blum-Micali-Yao (BMY) PRG

- **Assume one-way functions exist**
  - e.g. integer multiplication, discrete log, RSA.

Definition: **One Way Permutation (OWP):**

OWF in which f_n is 1-1

In this case can simplify:

"$Pr_x[C_n(f_n(x)) \in f_n^{-1}(f_n(x))] \leq \varepsilon(n)$" to

$$\mathbf{Pr_y[C_n(y) = f_n^{-1}(y)] \leq \varepsilon(n)}$$

So each f_n is a Hard to Invert Permutation:

For every family of **poly-size circuits $\{C_n\}$,**

$$\mathbf{Pr_y[C_n(y) = f_n^{-1}(y)] \leq \varepsilon(n)}$$

where $\varepsilon(n) = o(n^{-c})$ for any constant c

First attempt at PRG

- **Attempt constructing a PRG from one way function f:**

  Seed: $y_0 \in \{0,1\}^t$

  For $i = 1,\ldots,k-1$

  $y_i = f_t(y_{i-1})$ (recursive application of f_t)

  $G(y_0) = y_{k-1}y_{k-2}y_{k-3}\ldots y_0$ has $m=kt$ bits so $k = m/t$

- **Output $G(y_0)$** is computable in **polynomial time** at most

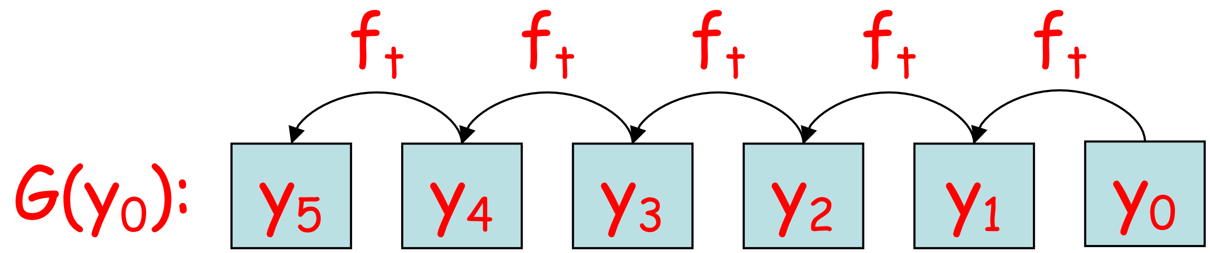
  $$kt^{c'} < mt^{c'-1} \leq m^{c'} \text{ for a constant } c'$$

First attempt at PRG

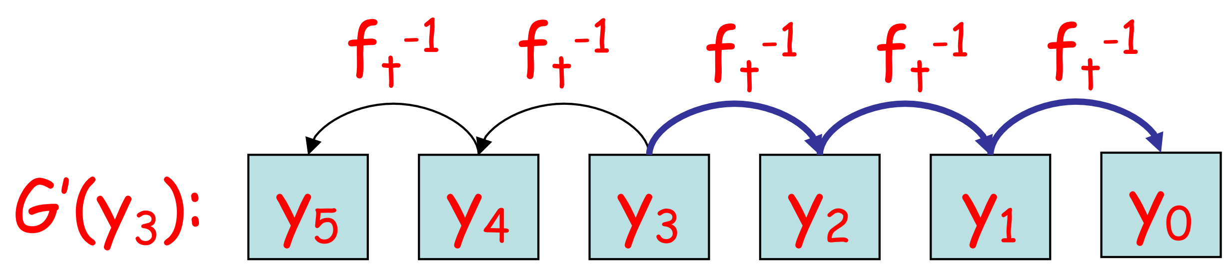
attempt:

- $y_0 \in \{0,1\}^t$
- $y_i = f_t(y_{i-1})$

- $G(y_0) =$
  $y_{k-1} y_{k-2} y_{k-3} \cdots y_0$



same distribution!



First attempt at PRG

Theorem: Output $G(y_0)$ is "**unpredictable**":

Claim: no poly-size circuit C can output $\mathbf{y_{i-1}}$ given $\mathbf{y_{k-1} y_{k-2} y_{k-3} \ldots y_i}$ with non-negligible success prob.

Proof: if C could do this, then using assumption that f_t is 1-1, result is **poly-size circuit to compute:**

$$\mathbf{y_{i-1} = f_t^{-1}(y_i) \text{ from } y_i}$$

contradicting assumption that f is a **one-way function. QED**

First attempt at PRG

- **One key issue:**
  - We assumed **one-way function f,** so it is **hard** to compute y_{i-1} from y_i
  - but **might be easy to compute single bit** (or several bits) of y_{i-1} from y_i
  - Perhaps could use to build small circuit C that distinguishes generator G's output from uniform distribution on $\{0,1\}^m$

First attempt at PRG

- **Second issue:**
  - Also, we haven't proved "**unpredictability**" of generator G given a prefix is sufficient to meet **fooling** requirement (against a truly random sequence r) for each polynomial size circuit C:

$$|\Pr_r[C(r) = 1] - \Pr_z[C(G(z)) = 1]| \le \varepsilon$$

Hard bits

- Again, let $\varepsilon(n) = o(n^{-c})$ for any constant c.

- If **$\{f_n\}$ is one-way permutation** we know:
  - **no poly-size circuit can compute $f_n^{-1}(y)$ from y with non-negligible success probability:**

$$\Pr_r[C_n(r) = f_n^{-1}(r)] \leq \varepsilon(n)$$

- We want to identify **a single bit position j** for which:
  - **no poly-size circuit can compute the jth bit position $(f_n^{-1}(x))_j$ from x with non-negligible advantage ½ + $\varepsilon(n)$ over a coin flip**

$$\Pr_r[C_n(r) = (f_n^{-1}(r))_j) \leq ½ + \varepsilon(n)$$

Generalized Hard Bits

- For some specific functions f we know of such a hard bit position j

- More general:

  function $h_n:\{0,1\}^n \rightarrow \{0,1\}$

  rather than just a bit position j.

Defining Generalized Hard Bits

Definition: **Hard Bit for $g = \{g_n\}$** is family $h = \{h_n\}$, **$h_n:\{0,1\}^n \rightarrow \{0,1\}$** such that **if circuit family $\{C_n\}$ of size $s(n)$** achieves:

$$\Pr_x[C_n(x) = h_n(g_n(x))] \geq ½ + \varepsilon(n)$$

then **there is a circuit family $\{C'_n\}$ of size $s'(n)$** that achieves:

$$\Pr_x[C'_n(x) = g_n(x)] \geq \varepsilon'(n)$$

with:

$\varepsilon'(n) = (\varepsilon(n)/n)^{O(1)}$

$s'(n) = (s(n)n/\varepsilon(n))^{O(1)}$

Generalized Hard bits

<u>**Equivalent Definition**</u>: **Hard Bit for $g = \{g_n\}$** is family $h = \{h_n\}$, $\mathbf{h_n : \{0,1\}^n \rightarrow \{0,1\}}$ such that if for all **circuit families $\{C'_n\}$ of size $s'(n)$:**

$$\Pr_x[C'_n(x) = g_n(x)] < \varepsilon'(n)$$

Then for **all circuit families $\{C_n\}$ of size $s(n)$:**

$$\Pr_x[C_n(x) = h_n(g_n(x))] < ½ + \varepsilon(n)$$

with:

$\varepsilon'(n) = (\varepsilon(n)/n)^{O(1)}$
$s'(n) = (s(n)n/\varepsilon(n))^{O(1)}$

Goldreich-Levin Hard Bit

- To get a generic hard bit, we will modify our one-way permutation $f_n(x)$, to allow **two inputs in a new function** $f'_n(x,y)$:

- Define $f'_n : \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}^{2n}$ as:

$$f'_n(x,y) = (f_n(x), y)$$

Goldreich-Levin Hard Bit

$f'_n(x,y) = (f_n(x), y)$

- Two observations:

  (a) **f' is a permutation if f is**

  (b) if circuit C_n achieves

  $$\Pr_{x,y}[C_n(x,y) = f'^{-1}_n(x,y)] \geq \varepsilon(n)$$

  then for some y^*

  $$\Pr_x[C_n(x,y^*)=f'^{-1}_n(x,y^*)=(f_n^{-1}(x), y^*)] \geq \varepsilon(n)$$

  and so:

  f' is a one-way permutation if f is.

Goldreich-Levin Hard Bit

- The **Goldreich-Levin (GL) function**:

$$\mathbf{GL_{2n} : \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}}$$

is defined by:

$$\mathbf{GL_{2n}(x,y) = \oplus_{i:y_i = 1} x_i}$$

where $\oplus$ is the XOR operation.

- This gives the **parity of subset of bits of x selected by 1's of y** (since we XOR only the x_i where $x_i = 1$)
- Same as inner-product of n-vectors x and y in GF(2)

<u>Theorem</u> (G-L): For every one-way function f, GL is a hard bit for f' (we will not prove this)

Yao's Lemma: Distinguishers & Predictors

- $\{0,1\}^n$ = set of binary strings of length n
- **U_n is uniform random distribution** on $\{0,1\}^n$
- **D is a given distribution** on $\{0,1\}^n$
- **D ε-passes statistical tests of size s** if for all circuits of size s:

$$|Pr_{x \in U_n}[C(x) = 1] - Pr_{x \in D}[C(x) = 1]| \leq \varepsilon$$

  – In other words, **no circuit C of size s can distinguish the uniform distribution U_n from distribution D**

  – Circuit violating this is sometimes called an efficient "**distinguisher**"

Yao's Lemma: Distinguishers & Predictors

- **Distribution D ε-passes prediction tests of size s** if for all circuits of size s:

$$\Pr_{x\in D}[C(x_1,x_2,\ldots,x_{i-1}) = x_i] \leq ½ + \varepsilon$$

  - Circuit violating this is sometimes called an efficient "**predictor**"
- Predictor seems stronger than Distinguisher
- Yao showed they are essentially the same!
  - important result and proof using Yao's "**hybrid argument**"

Yao's Lemma: Distinguishers & Predictors

Distribution D ε-passes prediction tests of size s if for all circuits of size s:

$$\Pr_{y \in D}[C(x_1, x_2, \ldots, x_{i-1}) = x_i] \le \tfrac{1}{2} + \varepsilon$$

<u>**Lemma**</u> (Yao): If a **distribution D on $\{0,1\}^n$ (ε/n)-passes all prediction tests of size s**, then:

distribution D ε-passes all statistical tests of size s' = s – O(n).

Yao's Lemma: Distinguishers & Predictors

- **Proof of Yao's Lemma:**
  - idea: proof by **contradiction**
  - Assume a **size s' distinguisher C**:

$$\Pr_{y \in U_n}[C(x) = 1] - \Pr_{y \in D}[C(x) = 1]| > \varepsilon$$

  (so C can distinguish uniform distribution U_n from distribution D with likelihood > ε)

  - We will produce a **size s predictor P**:

$$\Pr_{y \in D}[P(x_1, x_2, \ldots, x_{i-1}) = x_i] > ½ + \varepsilon/n$$

  - Proof uses distributions that are "**hybrids**" of the **uniform random distribution U_n** and **distribution D**

Yao's Lemma: Distinguishers & Predictors

– Assume a size s' distinguisher circuit C:

$$|Pr_{x \in U_n}[C(x) = 1] - Pr_{x \in D}[C(x) = 1]| > \varepsilon$$

– Define n+1 hybrid distributions å

– Form **hybrid distribution D_i**:

- **Uniformly randomly sample $x = x_1 x_2 \ldots x_n$ from D**
- **Uniformly randomly sample $r = r_1 r_2 \ldots r_n$ from U_n**
- **Output:**

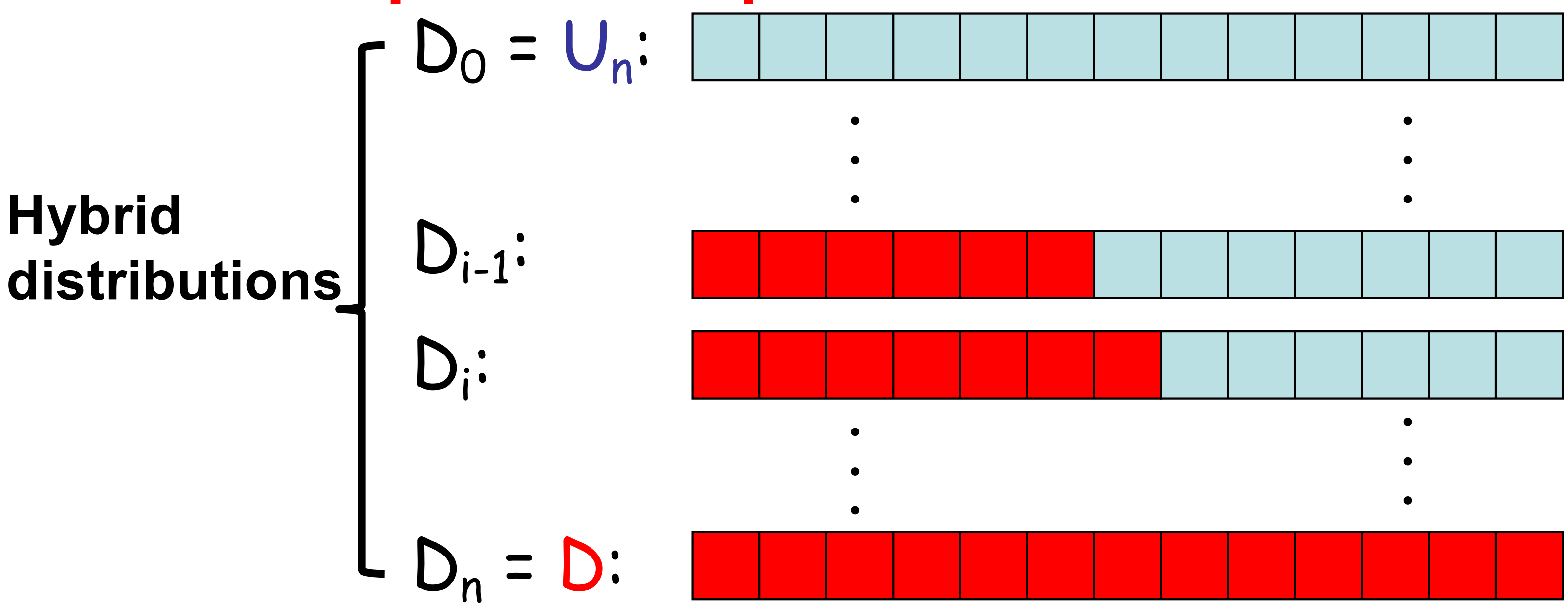
$$x_1 x_2 \ldots x_i \, r_{i+1} r_{i+2} \ldots r_n$$

Yao's Lemma: Distinguishers & Predictors

For sake of contradiction, we assume:

A size s' circuit C can distinguish uniform distribution U_n from distribution D with likelihood > ε

U_n is a uniform random distribution so passes all prediction tests



At some point get a predictor **with likelihood > ε/n** ***for a bit of a hybrid distribution***

Assume distribution D can be distinguished from uniform distribution U_n with likelihood > ε

Yao's Lemma: Distinguishers & Predictors

For sake of contradiction, we assume:

A size s' circuit C can distinguish uniform distribution U_n from distribution D with likelihood > ε

- Define: $p_i = Pr_{x \in D_i}[C(x) = 1]$
- Note: $p_0 = Pr_{x \in U_n}[C(x)=1]$; $p_n = Pr_{x \in D}[C(x)=1]$
- By assumption: $\varepsilon < |p_n - p_0|$
- Use triangle inequality: $|p_n - p_0| \leq \Sigma_{1 \leq i \leq n} |p_i - p_{i-1}|$
- Hence $\Sigma_{1 \leq i \leq n} |p_i - p_{i-1}| > \varepsilon$
- So there must be **some i** for which $|p_i - p_{i-1}| > \varepsilon/n$
- WLOG assume $p_i - p_{i-1} > \varepsilon/n$
  - can invert output of C if necessary

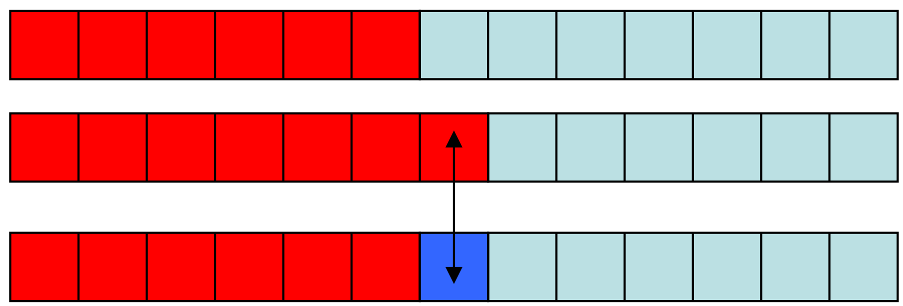
Yao's Lemma: Distinguishers & Predictors

- **Define** **"flipped" distribution D_i'** to be $\mathbf{D_i}$ with **i-th bit flipped**
- $\mathbf{p_i' = Pr_{x \in D_i'}[C(x) = 1]}$

D_{i-1}:

D_i:

D_i' :



- notice:

$$D_{i-1} = (D_i + D_i')/2 \qquad p_{i-1} = (p_i + p_i')/2$$

Yao's Lemma: Distinguishers & Predictors

- **Randomized predictor P' for i^{th} bit:**
  - **input:** $z = x_1 x_2 \ldots x_{i-1}$ (which comes from **D**)
  - **flip a coin:** $r \in \{0,1\}$ **(random bit)**
  - **Uniformly choose** $w = w_{i+1} w_{i+2} \ldots w_n$ **from** U_{n-i}
  - evaluate $C(z\ r\ w)$
  - **if 1, output r; if 0, output $\neg r$**

We will later prove:

Claim:

$$\Pr_{x \in D, r, w \in U_{n-i}}[P'(x_1, x_2 \ldots x_{i-1}) = x_i] > ½ + \varepsilon/n$$

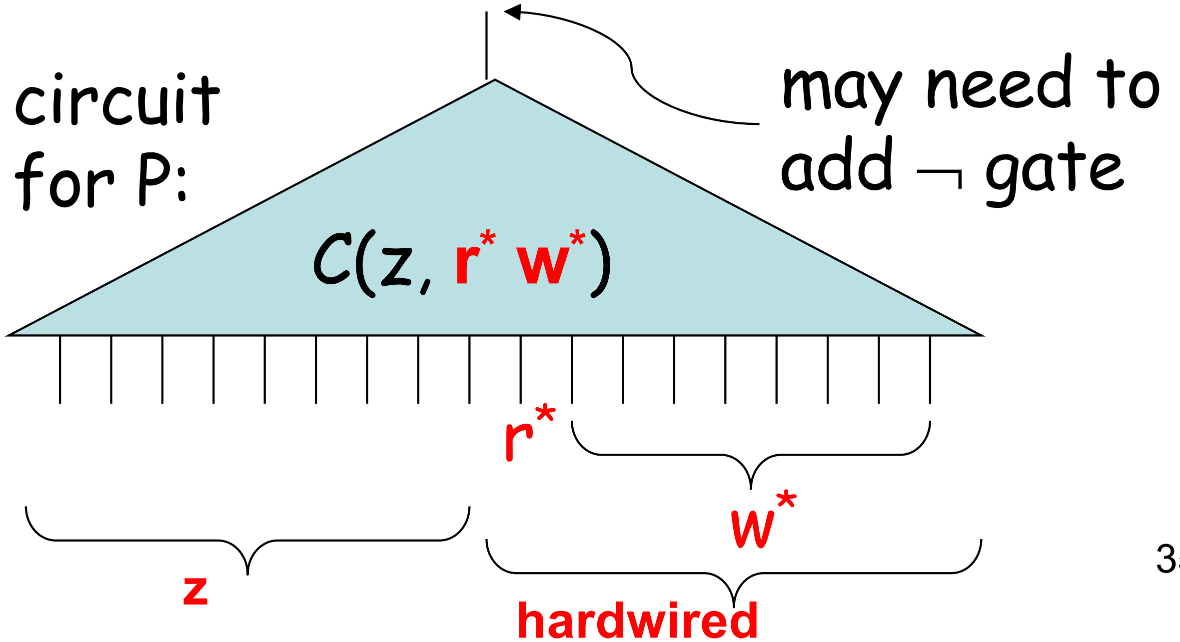
Yao's Lemma: Distinguishers & Predictors

- P' is randomized procedure
- Hence there exist some fixed values $\mathbf{r}^*$, $\mathbf{w}^*$ of its random bits **r, w** that preserves the success probability (this is nonconstructive)
- Final predictor P has $\mathbf{r}^*$ and $\mathbf{w}^*$ **hardwired**:

Size is

s' + O(n) = s

as promised



Yao's Lemma: Distinguishers & Predictors

$u = y_1 y_2 \ldots y_{i-1}$

Can show Claim:

$$\Pr_{x \in D, r, w \in U_{n-i}}[P'(x_1 \ldots x_{i-1}) = x_i] > ½ + \varepsilon/n.$$

This **requires a 3 page proof (can skip)**, using our prior proof that $p_i - p_{i-1} > \varepsilon/n$

Yao's Lemma: Distinguishers & Predictors

Claim:

$$\Pr_{x \in D, r, w \in U_{n-i}}[P'(x_1 \ldots x_{i-1}) = x_i] > \tfrac{1}{2} + \varepsilon/n.$$

Proof of claim: (can skip next 3 slides)

$$\Pr_{x \in D, r, w \in U_{n-i}}[P'(x_1 \ldots x_{i-1}) = x_i] =$$

$$\Pr[x_i = r \mid C(z,r,w) = 1]\Pr[C(z,r,w) = 1]$$

$$+ \Pr[x_i = \neg r \mid C(z,r,w) = 0]\Pr[C(z,r,w) = 0]$$

$z = x_{1}x_{2} \ldots x_{i-1}$

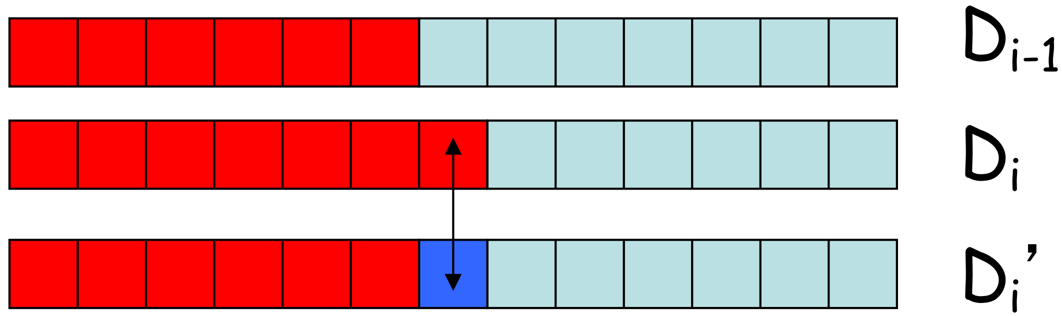
$$= \Pr[x_i = r \mid C(z,r,w) = 1](p_{i-1})$$

$$+ \Pr[x_i = \neg r \mid C(z,r,w) = 0](1 - p_{i-1})$$

Yao's Lemma: Distinguishers & Predictors

Proof of claim, Cont: (can skip)

$z = x_1 y_2 \ldots x_{i-1}$



– Can show:

$\Pr[x_i = r \mid C(z,r,w) = 1]$

$= \Pr[C(z,r,w) = 1 \mid x_i = r]\Pr[x_i=r] / \Pr[C(z,r,w) = 1]$

$= p_i/(2p_{i-1})$

$\Pr[x_i = \neg r \mid C(z,r,w) = 0]$

$= \Pr[C(z,r,w) = 0 \mid x_i= \neg r]\Pr[x_i=\neg r] / \Pr[C(z,r,w) = 0]$

$= (1 – p_i') / 2(1 - p_{i-1})$

Yao's Lemma: Distinguishers & Predictors

Proof of claim, Cont: (can skip)

- We know:

  $Pr[x_i = r \mid C(z,r,w) = 1] = p_i/(2p_{i-1})$

  $Pr[x_i = \neg r \mid C(z,r,w) = 0] = (1 - p_i')/2(1 - p_{i-1})$

  $p_{i-1} = (p_i + p_i')/2$

  $p_i - p_{i-1} > \varepsilon/n$

- Success probability: $Pr[P'(x_1 \ldots x_{i-1}) = x_i]$

  $= Pr[x_i=r|C(z,r,w)=1](p_{i-1}) + Pr[x_i=\neg r|C(z,r,w)=0](1-p_{i-1})$

  $= (p_i/(2p_{i-1}))(p_{i-1}) + ((1 - p_i')/2(1 - p_{i-1}))(1-p_{i-1})$

  $= \frac{1}{2} + (p_i - p_i')/2$

  $= \frac{1}{2} + p_i/2 - (p_{i-1} - p_i/2) = \frac{1}{2} + p_i - p_{i-1} > \frac{1}{2} + \varepsilon/n.$

$p_{i-1} = (p_i + p_i')/2$
So $p_i'/2 = p_{i-1} - p_i/2$

The BMY Generator

- Recall **Ultimate Goal:**

For all $1 > \delta > 0$, **family of PRGs $\{G_m\}$** with

output length m

fooling size $s = m$

seed length $t = m^\delta$

running time m^c

error $\varepsilon < 1/6$

- **If one way permutations exist then, by Goldreich-Levin Hard Bit Theorem,** there is a **one way permutation $f = \{f_n\}$ with hard bit $h = \{h_n\}$.**

The BMY Generator

- Let $1 > \delta > 0$.
- Generator $G^{\delta} = \{G^{\delta}_m\}$:

$t = m^{\delta}$

Seed: $y_0 \in \{0,1\}^t$

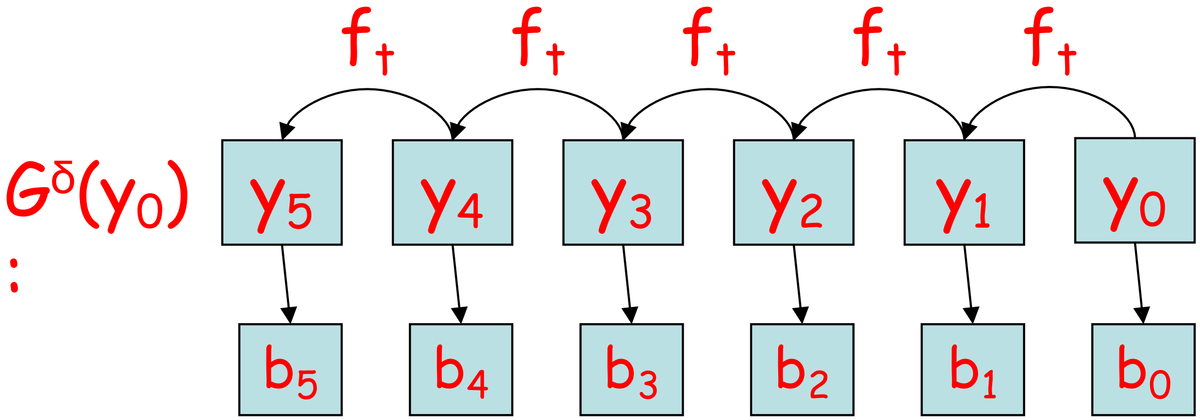
For $i = 1,\ldots,m$:

$y_i = f_t(y_{i-1})$ (recursive application of one-way permutation f_t)

$b_i = h_t(y_i)$ (generating hard bit b_i)

Generator outputs: $G^{\delta}(y_0) = b_{m-1}b_{m-2}b_{m-3}\ldots b_0$

The BMY Generator



Generator $G^\delta = \{G^\delta_m\}$:

$t = m^\delta$; $y_0 \in \{0,1\}^t$; $y_i = f_t(y_{i-1})$; $b_i = h_t(y_i)$

$G^\delta_m(y_0) = b_{m-1}b_{m-2}b_{m-3}\ldots b_0$

The BMY Generator

Finally ready to prove:

Theorem (BMY): for every $\delta > 0$, there is a constant c s.t. for all constants d, e,

The BMY PRG G^δ has:

error $\varepsilon < 1/m^d$

fooling size $s = m^e$

running time m^c

- Note: stronger than we need for BBP:

  sufficient to have $\varepsilon < 1/6$; $s = m$

The BMY Generator

Generator $G^\delta = \{G^\delta_m\}$:

$t = m^\delta$; $y_0 \in \{0,1\}^t$; $y_i = f_t(y_{i-1})$; $b_i = h_t(y_i)$

$G^\delta_m(y_0) = b_{m-1} b_{m-2} b_{m-3} \ldots b_0$

- **Proof:**

  G^δ is computable in time at most

  $$\mathbf{mt^c < m^{c+1}}$$

  For sake of **contradiction**:

  Assume G^δ does not $(1/m^d)$-pass statistical test $C = \{C_m\}$ of size m^e:

  $$|\Pr_{y \leftarrow U_m}[C(y) = 1] - \Pr_{z \leftarrow D}[C(z) = 1]| > 1/m^d$$

The BMY Generator

Generator $G^\delta = \{G^\delta{}_m\}$:

$t = m^\delta$; $y_0 \in \{0,1\}^t$; $y_i = f_t(y_{i-1})$; $b_i = h_t(y_i)$

$G^\delta{}_m(y_0) = b_{m-1}b_{m-2}b_{m-3}\ldots b_0$

Applying **Yao's Theorem**:

Transform this **distinguisher** into a **predictor** P of size $m^e + O(m)$:

$$\mathbf{Pr_y[P(b_{m-1}\ldots b_{m-i}) = b_{m-i-1}] > ½ + 1/m^{d+1}}$$

The BMY Generator

Generator $G^\delta = \{G^\delta{}_m\}$:

$t = m^\delta$; $y_0 \in \{0,1\}^t$; $y_i = f_t(y_{i-1})$; $b_i = h_t(y_i)$

$G^\delta{}_m(y_0) = b_{m-1}b_{m-2}b_{m-3}\ldots b_0$

Procedure to compute $h_t(f_t^{-1}(y))$:

- set $y_{m-i} = y$; $b_{m-i} = h_t(y_{m-i})$
- compute y_j, b_j for $j = m-i+1, m-i+2\ldots, m-1$ as above
- evaluate $P(b_{m-1}b_{m-2}\ldots b_{m-i})$
- f a permutation implies $b_{m-1}b_{m-2}\ldots b_{m-i}$ distributed as (prefix of) output of generator:

$$\Pr_y[P(b_{m-1}b_{m-2}\ldots b_{m-i}) = b_{m-i-1}] > ½ + 1/m^{d+1}$$

The BMY Generator

Generator $G^\delta = \{G^\delta_m\}$:

$t = m^\delta$; $y_0 \in \{0,1\}^t$; $y_i = f_t(y_{i-1})$; $b_i = h_t(y_i)$

$G^\delta_m(y_0) = b_{m-1}b_{m-2}b_{m-3}\ldots b_0$

We have: $\Pr_y[P(b_{m-1}b_{m-2}\ldots b_{m-i}) = b_{m-i-1}] > ½ + 1/m^{d+1}$

– What is $\mathbf{b_{m-i-1}}$?

$$b_{m-i-1} = h_t(y_{m-i-1}) = h_t(f_t^{-1}(y_{m-i})) = h_t(f_t^{-1}(y))$$

– **We have described a family of polynomial-size circuits that computes $h_t(f_t^{-1}(y))$ from y with success greater than:**

$$½ + 1/\mathrm{poly}(m)$$

– **Contradiction** with assumption, **proving BMY's Theorem.**

QED

Consequences to Computation Complexity

- **The BMY Generator result implies:**

If one-way permutations exist then:

$$\mathbf{BPP} \subseteq \bigcap_{\delta>0} \mathbf{TIME}(2^{n^{\delta}}) \overset{\text{proper subset}}{\subset} \mathbf{EXP\ time}$$

- This simulation is better than brute force, but just barely
- Even stronger assumptions on difficulty of inverting one-way functions leads to improved simulations.

Unbreakable Cryptography: One-Time Pad with purely random key bits

- Let $\oplus$ be the XOR operation.
- n Input bits to encode: $x_1,x_2,\ldots,x_n$ in {01,}
- n random key bits $k_1,k_2,\ldots,k_n$ in {0,1}
- n bit encoded bits $e_1,e_2,\ldots,e_n$ in {0,1}

  where $e_i = x_i \oplus k_i$ for I =1,…,n
- n bit decoded bits $d_1,d_2,\ldots,d_n$ in {0,1}

  where $d_i = e_i \oplus k_i$ for I =1,…,n

But $d_i = e_i \oplus k_i$

$= (x_i \oplus k_i) \oplus k_i$

$= x_i \oplus (k_i \oplus k_i)$

$= x_i$ since $k_i \oplus k_i = 0$ and $x_i \oplus 0 = x_i$

Cryptography using **Pseudorandom** key bits again use One-Time Pad

- n Input bits to encode: $x_1, x_2, \ldots, x_n$ in {01,}
- n **pseudorandom** key bits $k_1, k_2, \ldots, k_n$ in {0,1}

that can not be predicted in polynomial time

- n bit encoded bits $e_1, e_2, \ldots, e_n$ in {0,1}

  `where $e_i = x_i \oplus k_i$ for I =1,…,n
- n bit decoded bits $d_1, d_2, \ldots, d_n$ in {0,1}

  where $d_i = e_i \oplus k_i$ for I =1,…,n

But again $d_i = x_i$

Gives polynomial-Time secure Cryptography !