First attempt

- attempt at PRG from OWP $f$:
  - $t = m^\delta$
  - $y_0 \in \{0,1\}^t$
  - $y_i = f_i(y_{i-1})$
  - $G(y_0) = y_{k-1}y_{k-2}y_{k-3}...y_0$
  - $k = m/t$
- computable in time at most $kt^c < mt^{c-1} = m^c$

Blum-Micali-Yao 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 $t = m^\delta$
  - running time $m^c$
  - error $\epsilon < 1/6$
- implies: $BPP \subseteq \cap_{\delta>0} \text{TIME}(2^{m^\delta}) \subseteq \text{EXP}$
- Why? simulation runs in time $O(m+m^\delta) = O(2^{m^{2\delta}}) = O(2^{n^{2k^\delta}})$

First attempt

- output is "unpredictable":
  - no poly-size circuit $C$ can output $y_{i-1}$ given $y_{k-1}y_{k-2}y_{k-3}...y_i$ with non-negl. success prob.
  - if $C$ could, then given $y_i$ can compute $y_{k-1}, y_{k-2}, ... y_{i+1}, y_{i+1}$ and feed to $C$
  - result is poly-size circuit to compute $y_{i+1} = f_{i+1}(y_i)$ from $y_i$
  - note: we’re using that $f_i$ is 1-1

First attempt

- one problem:
  - 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 $y_i$
  - could use to build small circuit $C$ that distinguishes $G$’s output from uniform distribution on $\{0,1\}^m$
First attempt

- second problem
  - we don’t know if “unpredictability” given a prefix is sufficient to meet fooling requirement:
    \[ |\Pr_y[C(y) = 1] - \Pr_z[C(G(z)) = 1]| \leq \varepsilon \]

Hard bits

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

- We want to identify a single bit position \( j \) for which:
  - no poly-size circuit can compute \( (f_n^{-1}(x))^j \) from \( x \) with non-negligible advantage over a coin flip
    \[ \Pr_y[C_n(y) = (f_n^{-1}(y))^j] \leq \frac{1}{2} + \varepsilon(n) \]

Hard bits

- For some specific functions \( f \) we know of such a bit position \( j \)
- More general:
  - function \( h_n : \{0,1\}^n \rightarrow \{0,1\} \)
    rather than just a bit position \( j \).

Definition:

- hard bit for \( g = \{g_n\} \) is family \( h = \{h_n\} \), \( h_n : \{0,1\}^n \rightarrow \{0,1\}^2 \) such that if circuit family \( \{C_n\} \) of size \( s(n) \) achieves:
  \[ \Pr_{x,y}[C_n(x,y) = h_n(g_n(x))] \geq \frac{1}{2} + \varepsilon(n) \]
  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:
  - \( \varepsilon'(n) = (\varepsilon(n)/n) \cdot (\varepsilon(n)/n) \cdot (\varepsilon(n)/n) \cdot \ldots \cdot (\varepsilon(n)/n) \cdot (\varepsilon(n)/n) \cdot (\varepsilon(n)/n) \)
  - \( s'(n) = (s(n)n/n) \cdot (s(n)n/n) \cdot (s(n)n/n) \cdot \ldots \cdot (s(n)n/n) \cdot (s(n)n/n) \cdot (s(n)n/n) \cdot \ldots \cdot (s(n)n/n) \cdot (s(n)n/n) \)

Goldreich-Levin

- To get a generic hard bit, first need to modify our one-way permutation
- 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

• The Goldreich-Levin function:
  \( GL_{2^n} : \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\} \)
  is defined by:
  \( GL_{2^n}(x,y) = \oplus_{i:y_i = 1} x_i \)
  – parity of subset of bits of \( x \) selected by 1’s of \( y \)
  – inner-product of \( n \)-vectors \( x \) and \( y \) in GF(2)

**Theorem** (GL): for every function \( f \), \( GL \) is a hard bit for \( f' \).
(proof: problem set)

Distinguishers and predictors

• Distribution \( D \) on \( \{0,1\}^n \)
  \( D \epsilon \)-passes statistical tests of size \( s \) if for all circuits of size \( s \):
  \( |Pr_{y \leftarrow U_n}[C(y) = 1] - Pr_{y \leftarrow D}[C(y) = 1]| \leq \epsilon \)
  – circuit violating this is sometimes called an efficient “distinguisher”

Distinguishers and predictors

**Theorem** (Yao): if a distribution \( D \) on \( \{0,1\}^n \)
  \( \epsilon/n \)-passes all prediction tests of size \( s \),
  then \( \epsilon \)-passes all statistical tests of size \( s' = s - O(n) \).

Distinguishers and predictors

• Proof:
  – idea: proof by contradiction
  – given a size \( s' \) distinguisher \( C \):
    \( |Pr_{y \leftarrow U_n}[C(y) = 1] - Pr_{y \leftarrow D}[C(y) = 1]| > \epsilon \)
  – produce size \( s \) predictor \( P \):
    \( Pr_{y \leftarrow D}[P(y_1,2,...,n) = y] > \frac{1}{2} + \epsilon/n \)
  – work with distributions that are “hybrids” of the uniform distribution \( U_n \) and \( D \)
Distinguishers and predictors

• Hybrid distributions:

\[ D_0 = U_n \]

\[ D_1 : \]

\[ D_i : \]

\[ D_n = D_0 \]

April 22, 2015

Distinguishers and predictors

– Define: \( p_i = \Pr_{y \leftarrow D_i}[C(y) = 1] \)
– Note: \( p_0 = \Pr_{y \leftarrow U_n}[C(y) = 1] \); \( p_n = \Pr_{y \leftarrow D_n}[C(y) = 1] \)
– by assumption: \( \epsilon < |p_n - p_0| \)
– triangle inequality: \( |p_n - p_0| \leq \sum_{1 \leq i \leq n} |p_i - p_{i-1}| \)
– there must be some \( i \) for which \( |p_i - p_{i-1}| > \epsilon/n \)
– WLOG assume \( p_i - p_{i-1} > \epsilon/n \)

• can invert output of \( C \) if necessary

April 22, 2015

Distinguishers and predictors

– define distribution \( D_i' \) to be \( D_i \) with \( i \)-th bit flipped
– \( p_i' = \Pr_{y \leftarrow D_i'}[C(y) = 1] \)

\[ D_{i-1} = (D_i + D_i')/2 \]

\[ p_{i-1} = (p_i + p_i')/2 \]

April 22, 2015

Distinguishers and predictors

• randomized predictor \( P' \) for \( i \)-th bit:
  – input: \( u = y_1 y_2 \ldots y_{i-1} \) (which comes from \( D \))
  – flip a coin: \( d \in \{0,1\} \)
  – \( w = w_{i+1} w_{i+2} \ldots w_n \leftarrow U_n \)
  – evaluate \( C(u, d, w) \)
  – if 1, output \( d \); if 0, output \( \neg d \)

Claim:

\[ \Pr_{y \leftarrow D_i, w \leftarrow U_n}[P'(y_1 \ldots y_{i-1}) = y_i] > \frac{1}{2} + \epsilon/n. \]

April 22, 2015

Distinguishers and predictors

• \( P' \) is randomized procedure
• there must be some fixing of its random bits \( d, w \) that preserves the success prob.
• final predictor \( P \) has \( d^* \) and \( w^* \) hardwired:

Size is \( s' + O(n) = s \) as promised

April 22, 2015

Distinguishers and predictors

• Proof of claim:

\[ \Pr_{y \leftarrow D_i, w \leftarrow U_n}[P'(y_1 \ldots y_{i-1}) = y_i] = \Pr[y_i = d \mid C(u,d,w) = 1] \Pr[C(u,d,w) = 1] + \Pr[y_i = \neg d \mid C(u,d,w) = 0] \Pr[C(u,d,w) = 0] \]

\[ = \Pr[y_i = d \mid C(u,d,w) = 1] p_i + \Pr[y_i = \neg d \mid C(u,d,w) = 0] (1 - p_i) \]

April 22, 2015
Distinguishers and predictors

- Observe:

\[ \Pr[y_i = d | C(u,d,w) = 1] = \frac{\Pr[C(u,d,w) = 1 | y_i = d] \Pr[y_i = d]}{\Pr[C(u,d,w) = 1]} = \frac{p_i}{2p_i - 1} \]

\[ \Pr[y_i = \neg d | C(u,d,w) = 0] = \frac{\Pr[C(u,d,w) = 0 | y_i = \neg d] \Pr[y_i = \neg d]}{\Pr[C(u,d,w) = 0]} = \frac{(1 - p_i')}{2(1 - p_i - 1)} \]

\[ D_{i-1} = y_1 y_2 \ldots y_{i-1} \]

April 22, 2015

Distinguishers and predictors

- Success probability:

\[ \Pr[y_i = d | C(u,d,w) = 1] (p_i - 1) + \Pr[y_i = \neg d | C(u,d,w) = 0] (1 - p_i - 1) \]

- We know:

  \[ \Pr[y_i = d | C(u,d,w) = 1] = \frac{p_i}{2p_i - 1} \]

  \[ \Pr[y_i = \neg d | C(u,d,w) = 0] = \frac{1 - p_i'}{2(1 - p_i - 1)} \]

  \[ p_i - p_i' > \epsilon / n \]

- Conclude:

\[ \Pr[P' (y_1 \ldots y_{i-1}) = y_i] = \frac{1}{2} + \frac{p_i - p_i'}{2} = \frac{1}{2} + \frac{p_i}{2} > \frac{1}{2} + \frac{\epsilon}{n}. \]

April 22, 2015

The BMY Generator

- Recall 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 \( \epsilon < 1/6 \)

- If one way permutations exist then WLOG there is OWP \( f = \{f_n\} \) with hard bit \( h = \{h_n\} \)

April 22, 2015

The BMY Generator

- Generator \( G^\delta = \{G^\delta_m\} \):

  - \( t = m^\delta \)
  - \( y_0 \in \{0,1\}^t \)
  - \( y_1 = f_t(y_{i-1}) \)
  - \( b_i = h_t(y_i) \)
  - \( G^\delta(y_0) = b_{m-1} b_{m-2} \ldots b_0 \)

April 22, 2015

The BMY Generator

**Theorem (BMY):** for every \( \delta > 0 \), there is a constant \( c \) s.t. for all \( d, e \), \( G^\delta \) is a PRG with

- error \( \epsilon < 1/m^d \)
- fooling size \( s = m^\delta \)
- running time \( m^c \)

- Note: stronger than we needed

  - sufficient to have \( \epsilon < 1/6 \); \( s = m \)

April 22, 2015
The BMY Generator

Generator $G^b = \{G_m^b\}$:
- $t = m^2$: $y_0 \in \{0,1\}^t$: $y_i = f_i(y_{i-1})$: $b_i = h_i(y_i)$
- $G_m^b(y_0) = b_m, b_{m-1}, b_{m-3} - b_0$

- transform this **distinguisher** into a **predictor** $P$ of size $m^2 + O(m)$:
  
  $\Pr_{y}[P(b_{m-1} \ldots b_m) = b_{m+1}] > 1/2 + 1/m^{d+1}$

- What is $b_{m+1}$?
  
  $b_{m+1} = h_i(y_{m+1}) = h_i(f_i(y_m)) = h_i(f_i(y))$

- We have described a family of polynomial-size circuits that computes $h_i(f_i(y))$ from $y$ with success greater than $1/2 + 1/poly(m)$

- Contradiction.

April 22, 2015 30

The BMY Generator

Generator $G^b = \{G_m^b\}$:
- $t = m^2$: $y_0 \in \{0,1\}^t$: $y_i = f_i(y_{i-1})$: $b_i = h_i(y_i)$
- $G_m^b(y_0) = b_m, b_{m-1}, b_{m-3} - b_0$

- a procedure to compute $h_i(f_i^{-1}(y))$
  
  - set $y_{m+1} = y$; $b_{m+1} = h_i(f_i(y_{m+1}))$
  
  - compute $y_i, b_i$ for $i = m+1, m+2, \ldots, m-1$ as above
  
  - evaluate $P(b_m, b_{m-2}, \ldots, b_{m-1})$

- If a permutation implies $b_m, b_{m-2}, \ldots, b_{m-1}$ distributed as (prefix of) output of generator:
  
  $\Pr_{y}[P(b_{m}, b_{m-2} \ldots b_{m-1}) = b_{m+1}] > 1/2 + 1/m^{d+1}$

April 22, 2015 31

Hardness vs. randomness

- We have shown:
  
  If one-way permutations exist then
  
  $\text{BPP} \subset \cap_{b \geq 0} \text{TIME}(2^{b^{O(1)}}) \subset \text{EXP}$

- simulation is better than brute force, but just barely

- stronger assumptions on difficulty of inverting OWF lead to better simulations…

April 22, 2015 35

Hardness vs. randomness

- We will show:
  
  If $E$ requires exponential size circuits then
  
  $\text{BPP} = \text{P}$

  by building a different generator from different assumptions.

  $E = \cup_k \text{DTIME}(2^{kn})$

April 22, 2015 36
Hardness vs. randomness

- BMY: for every $\delta > 0$, $G^\delta$ is a PRG with
  - seed length $t = m^\delta$
  - output length $m$
  - error $\epsilon < 1/m^d$ (all $d$)
  - fooling size $s = m^g$ (all $e$)
  - running time $m^c$
- running time of simulation dominated by $2^t$

- To get $\text{BPP} = \text{P}$, would need $t = O(\log m)$
- BMY building block is one-way permutation:
  - $f: \{0,1\}^t \rightarrow \{0,1\}^t$
- required to fool circuits of size $m^g$ (all $e$)
- with these settings a circuit has time to invert $f$ by brute force!
  - can't get $\text{BPP} = \text{P}$ with this type of PRG

NW PRG

- NW: for fixed constant $\delta$, $G = \{G_n\}$ with
  - seed length $t = O(\log n)$
  - running time $n^c$
  - output length $m = n^\delta$
  - error $\epsilon < 1/m$
  - fooling size $s = m$
- Using this PRG we obtain $\text{BPP} = \text{P}$
  - to fool size $n^g$ use $G_{n^{gb}}$
  - running time $O(n^g + n^{gb}2^t) = \text{poly}(n)$

Comparison

- BMY: $\forall \delta > 0$ PRG $G^\delta$
  - seed length $t = m^\delta$
  - running time $t^m$
  - output length $m$
  - error $\epsilon < 1/m^d$ (all $d$)
  - fooling size $s = m^g$ (all $e$)
  - running time $m^c$

- NW: PRG $G$
  - seed length $t = O(\log m)$
  - running time $t^m$
  - output length $m$
  - error $\epsilon < 1/m$
  - fooling size $s = m^g$ (all $e$)

NW PRG

- First attempt: build PRG assuming $E$ contains unapproximable functions

Definition: The function family

$$f = \{f_n\}, f_n: \{0,1\}^n \mapsto \{0,1\}$$

is $s(n)$-unapproximable if for every family of size $s(n)$ circuits $(C_n)$:

$$\Pr\{C_n(x) = f_n(x)\} \leq \frac{1}{2} + 1/s(n).$$
One bit

• Suppose \( f = \{ f_n \} \) is \( s(n) \)-unapproximable, for \( s(n) = 2^{\Omega(n)} \), and in \( \mathbb{E} \).

• A “1-bit” generator family \( G = \{ G_n \} \):
  \[
  G_n(y) = y^* f_{\log n}(y)
  \]

• Idea: if not a PRG then exists a predictor that computes \( f_{\log n} \) with better than \( \frac{1}{2} + \frac{1}{s(\log n)} \) agreement; contradiction.

---

Many bits

• Try outputting many evaluations of \( f \):
  \[
  G(y) = f(b_1(y)) \circ f(b_2(y)) \circ \ldots \circ f(b_m(y))
  \]

• Seems that a predictor must evaluate \( f(b_i(y)) \) to predict \( i \)-th bit

• Does this work?

---

Nearly-Disjoint Subsets

**Definition:** \( S_1, S_2, \ldots, S_m \subset \{ 1 \ldots t \} \) is an \(( h, a )\) design if
- for all \( i \), \( |S_i| = h \)
- for all \( i \neq j \), \( |S_i \cap S_j| \leq a \)

---

Almost-Disjoint Subsets

**Lemma:** for every \( \varepsilon > 0 \) and \( m < n \) can in poly\( (n) \) time construct an
\(( h = \log n, a = \varepsilon \log n )\) design
\( S_1, S_2, \ldots, S_m \subset \{ 1 \ldots t \} \) with \( t = O(\log n) \).
Nearly-Disjoint Subsets

- Proof sketch:
  - pick random \((\log n)\)-subset of \(\{1\ldots t\}\)
  - set \(t = O(\log n)\) so that expected overlap with a fixed \(S_i\) is \(\epsilon \log n/2\)
  - probability overlap with \(S_i\) is > \(\epsilon \log n\) is at most \(1/n\)
  - union bound: some subset has required small overlap with all \(S_i\) picked so far…
  - find it by exhaustive search; repeat \(n\) times.

The NW generator

**Theorem** (Nisan-Wigderson): \(G=\{G_n\}\) is a pseudo-random generator with:

- seed length \(t = O(\log n)\)
- output length \(m = n^{\delta_3}\)
- running time \(n^c\)
- fooling size \(s = m\)
- error \(\epsilon = 1/m\)