6.875 Lecture 5

Spring 2020

Lecturer: Shafi Goldwasser

LAST TIME: Randomness I

NEW NOTION: Pseudo-random Generators
(Two different definitions; Equivalence)

CONSTRUCTION [Blum-Micali'82, Yao82]:
One-way Permutations + Hardcore Bits =
Pseudorandom Generator.

APPLICATIONS

TODAY: RANDOMNESS II

APPLICATIONS of CS-PRG

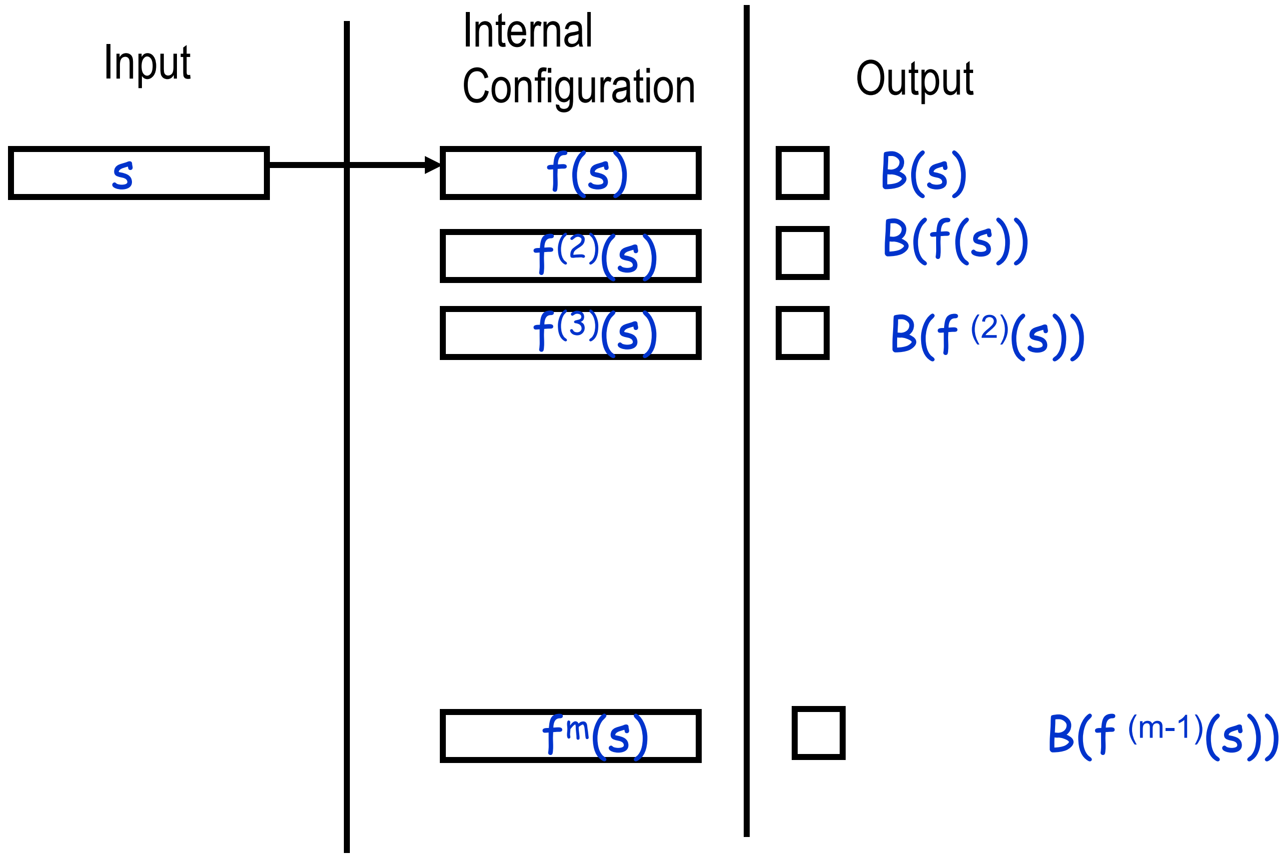
- **Complexity Theory**
- **Symmetric Encryption**

PSEUDO RANDOM FUNCTIONS[GGM85]

APPLICATIONS OF PSRF

WHERE DO WE FIND ONE-WAY FUNCTIONS?

RECALL: CONSTRUCTION of CS-PRG



- f is one-way permutation
- B is hard-core predicate for F

Recall: Every OWF Has an Associated Hard Core Bit

Theorem [GoldreichLevin]:

Let f be a One-way Function.

Define $f'(x,r) = f(x) \,||\, r$ where $|r|=|x|=n$.

Then **$B(x,r) = \sum x_i r_i \bmod 2 = \langle x,r \rangle$** is a hard-core predicate for f'.

(Alternatively, $\{B_r(x) = \langle x,r \rangle \bmod 2\}_r$ is a collection of hardcore predicates for f_i.)

BPP

- Class of problems L:{0,1}*->{0,1}

- L $\in$ **BPP** implies ∃PPT algorithm M_L

  $x \in L \Rightarrow Pr_{\text{coins } y}$[M(x,y) accepts x] > 2/3

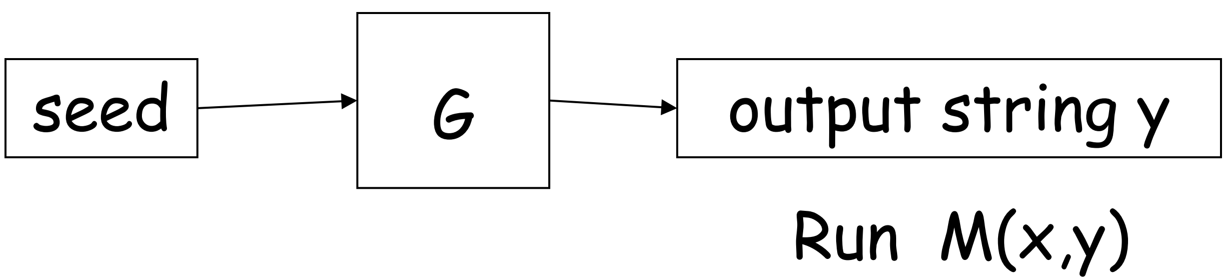
  $x \notin L \Rightarrow Pr_{\text{coins } y}$[M(x,y) with coins y, rejects. x] > 2/3

  Notation: M(x,y) = "M(x) with coins y"

Application: De-randomization

- Goal: simulate BPP in sub-exponential time

- Use Pseudo-Random Generator (PRG) to generate required randomness y:



Theorem: if one-way functions exist, then $BPP \subseteq \bigcap_{\varepsilon>0} DTIME(2^{n^{\varepsilon}})$

Proof[Yao] Given L in BPP

Convert BPP algorithm M into algorithm M ‘:

- On n-bit input x, say M uses n^c bits of randomness
- Let $m = n^{\varepsilon}$. Then $n^c=(m^{1/\varepsilon})^c = m^{c/\varepsilon}$
- Take CS-PRG $G:\{0,1\}^m \longrightarrow \{0,1\}$
- Output $\text{majority}_s \{M(x, G(s))\}$

Observation 1:

M’ is deterministic

Runtime of M’ = $O(2^{n^{\varepsilon}})$*runtime of M =

Theorem: if f one-way function, then $BPP \subseteq \cap_{\varepsilon>0} DTIME(2^{n^{\varepsilon}})$

Proof: Suppose not. ∃L & ε s.t. for inf. many n

Case 1: ∃x in L which M'(x) (incorrectly) rejects,

This implies that

- when using M(x,y) with pseudo-random y, M(x,y) will accept for <1/2 of the y's,

whereas

- when using M(x,y) with true randomness y, M(x,y) will accept >2/3 of the y's

⇒ M(x,) can be used as ia distinguisher between $U_{m^{c/\varepsilon}}$ and outputs of $G(U_m)$. See next page.

But G was CS-PRG, contradiction!

Case 2: ∃x not in L but M' (x) accepts, argue similarly….

Theorem: if f one-way function, then $BPP \subseteq \cap_{\varepsilon>0} DTIME(2^{n^{\varepsilon}})$

Proof (formalized)

Let $n' = m^{c/\varepsilon}$

$_u$se M as a distinguisher between $U_{n'}$ and $G(U_m)$ as follows

Hardwire x to M get polynomial time statistical test algorithm $D_x(y) := M(x,y)$:

On input y:

- (case 1) when $x \in L$,

$Pr[D_x(U_{n'})=1] \geq 2/3$ and $Pr[D_x(G(U_m)) = 1] < 1/2$

- (case 2) when $x \notin L$,

$Pr[D_x(U_{n'}) = 1] \leq 1/3$ and $Pr[D_x(U_m) = 1] > 1/2$

Simulating **BPP** in sub-exponential time

Remarks

D_x is a non-uniform algorithm (also called a circuit)

Sequence of algorithms, one for each length n for which there exists x of length n on which M and M' behave differently.

Contradicts the fact that f is a one-way function with respect to non-uniform algorithms

Application 2: Symmetric Encryption for long messages with short keys

Let G be CS-PRG which stretches n to m(n)-bits based on one-way function f.

- **Key Generation** Gen(1^n): randomly chose n-bit seed *s* in the domain of one-way function f
- **Encryption** Enc(m): for m(n)-bit message M
  compute G (*s*) , Send c=G(s) ⊕M (bit wise xor)
- **Decryption** D(c):
  compute G(s), let M=c⊕G(s)

Claim: Computational Secrecy

Proof: G(s) $\approx_{computationally}$ $U_{m(n)}$ implies

c=M⊕G(s) $\approx_{computationally}$ $U_{m(n)}$ (∀M adv can find)

Stateful encryption for many messages:

Let G be CS-PSRG which stretches n to m(n)-bits based on one-way function f.

Gen(1^n): randomly chose n-bit seed *s* in the domain of one-way function f . Initialize state i=0

Enc(m_i):

–compute and send c="ith block of G(s)" $\oplus m_i$

–set i=i+1

Dec(c_i):

–set m_i= "ith block of G(s)" $\oplus$c

–Set i=i+1

Need to maintain state. Is that inherent?

Questions:

Can you access directly the i-th block output of G?

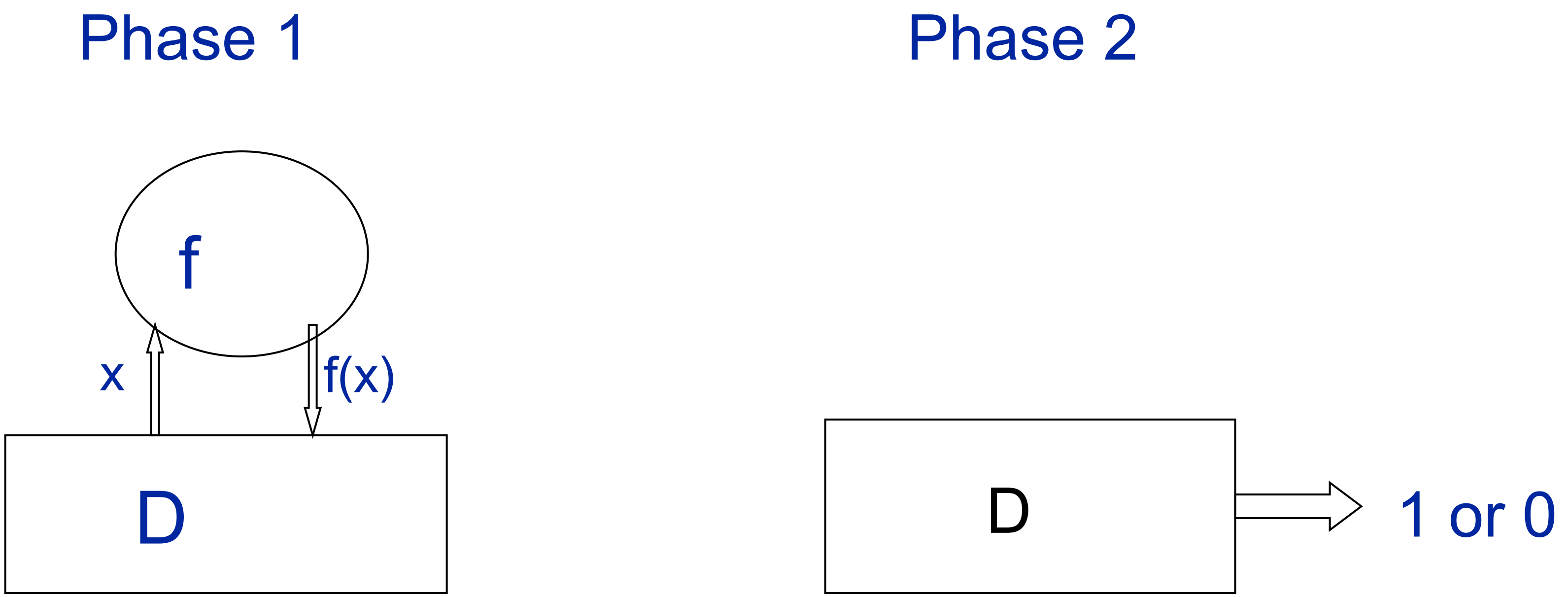
Can you do Stateless Encryption of many messages?

Pseudo Random Functions(PSRF)

Collection of indexed functions $f_s:\{0,1\}^n \Rightarrow \{0,1\}^n$ is pseudo-random if

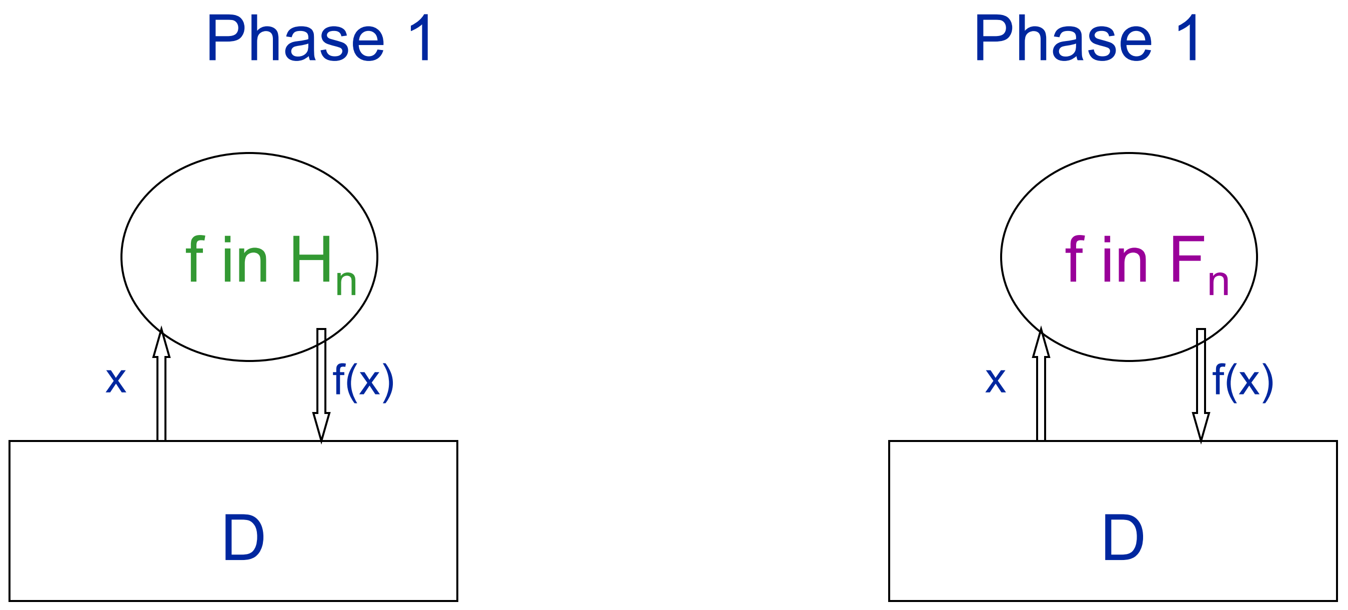
- Given s, can compute $f_s(x)$ is efficiently computable
- No adversary can distinguish between
  $(x, f_s(x))$ for x of its choice, and
  (x, U) (truly random function values).

Define: "statistical test" D or functions



Notation: D^f means "D has query access to f", i.e can ask for values of f(x) for x of its choice

Pseudo-Random F is indistinguishable from Random



Prob (D^f says 1 in Phase 2) ≈ Prob (D says 1 in phase 2)

Pseudo Random Functions: Formal

Let H_n = {f: $\{0,1\}^n$ -> $\{0,1\}^n$} all functions from n bits to n bits

Definition: F= $\{F_n\}_n$ where $F_n \subseteq H_n$ is a collection of pseudo random functions iff

1. *There exists* PPT algorithm G (1^n) to selects i s.t. $f_i \in F_n$
2. *There exists* PPT algorithm Eval s.t. Eval(x, i) =$f_i(x)$
3. For all PPT statistical tests for functions D^f, for all sufficiently large n

   | prob($f \in H_n$: $D^f(1^n)$ =1) -

   prob($f \in F_n$:$D^f(1^n)$ =1) | = negl(n)

NOTE: D^f makes polynomial number of calls to f

Existence of PSRF's

Theorem: If one-way functions exist, then collections of pseudo random functions exist

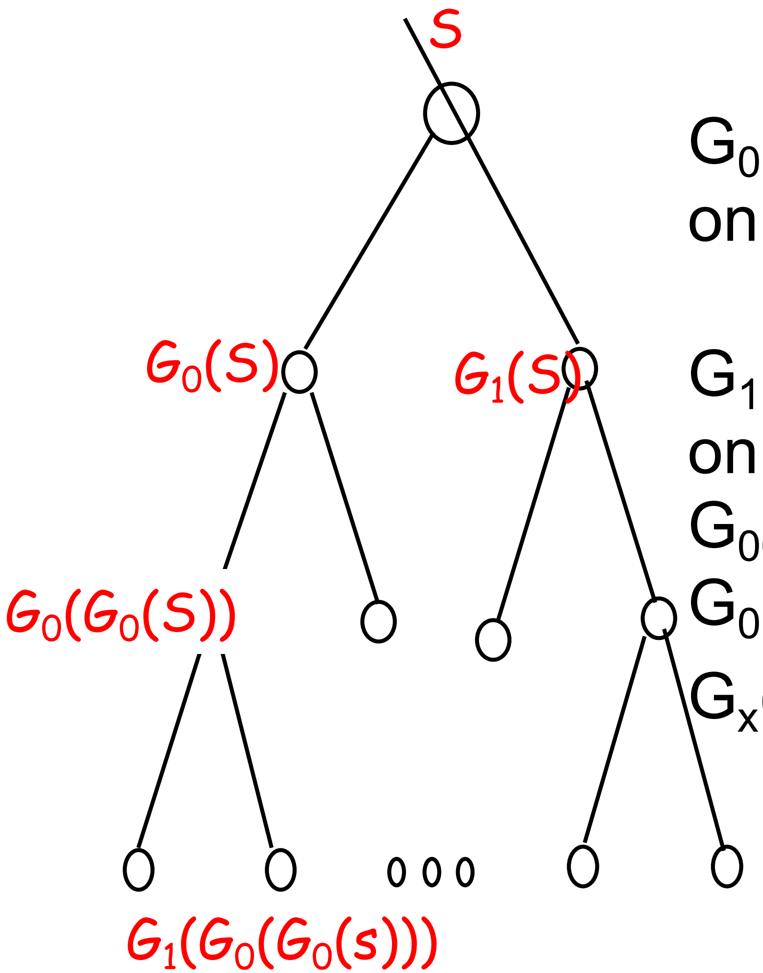
Proof:

Construction starts from CS-PRG G s.t.

$G:\{0,1\}^n \rightarrow \{0,1\}^{2n}$ on input seed of length n
output 2n bits

Easy-Lemma: ∀PPT A, ∀Poly P, ∀n suff. large,
$| \Pr[S \subseteq G(U_n) \text{ s.t } |S|=P(n): A(S) = 1] -$
$\Pr[S \subseteq U_{2n} \text{ s.t. } |S|=P(n): A(S) = 1] | = negl(n)$

Tree Like Construction



$G_0(s)$ = Run CS-PRG $G:\{0,1\}^n \to \{0,1\}^{2n}$ on seed s and output the <u>first</u> n output bits

$G_1(s)$ = Run a CS-PSRG $G:\{0,1\}^n \to \{0,1\}^{2n}$ on seed s and output the <u>2nd</u> n output bits

$G_{00}(s) = G_0(G_0(s))$

$G_{01}(s) = G_1(G_0(s))$ …

$G_x(s) = G_{x_n x_{n-1} \ldots x_1}(s) = G_{x_n}(G_{x_{n-1}}(\ldots G_{x_1}(s)\ldots))$

Each leaf corresponds to $x \in \{0,1\}^n$.

Construction of PSRF's

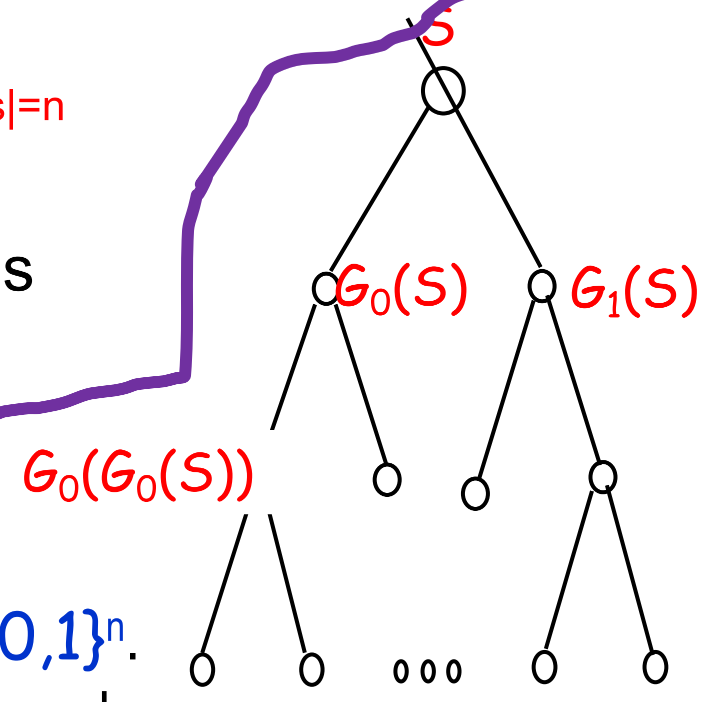
Define

$f_s(x) = G_x(s)$ e.g. $f_i(0000000) = G_0(G_0\ G_0(G_0(G_0(G_0\ (s))$

where $G_x(s) = G_{x_n x_{n-1}\ x_1}(s) = G_{x_n}(G_{x_{n-1}}(\ldots G_{x_1}(s))\ \ldots)$

Set PSRF family $F = \{F_n\}$ and $F_n = \{f_s\}_{|s|=n}$

Each evaluation of f is n G evaluations



Each leaf corresponds to $x \in \{0,1\}^n$.
Label of leaf: value of pseudo-random function at x

Theorem: If G is cs-prg, then F is psrf

Proof outline: By contradiction. Assume, algorithm D^f exists which "distinguishes" F_n from H_n with probability ε after poly many queries to f (f is either from F_n or all from H_n), then can construct algorithm A to "distinguish" outputs of $G(U_n)$ from U_{2n} with probability $\varepsilon' = \varepsilon/n$

Hybrid argument by levels of the tree

D_i : functions defined by filling *truly* random labels in nodes at level i and then filling lower levels with Pseudo-random values from i+1 down to n

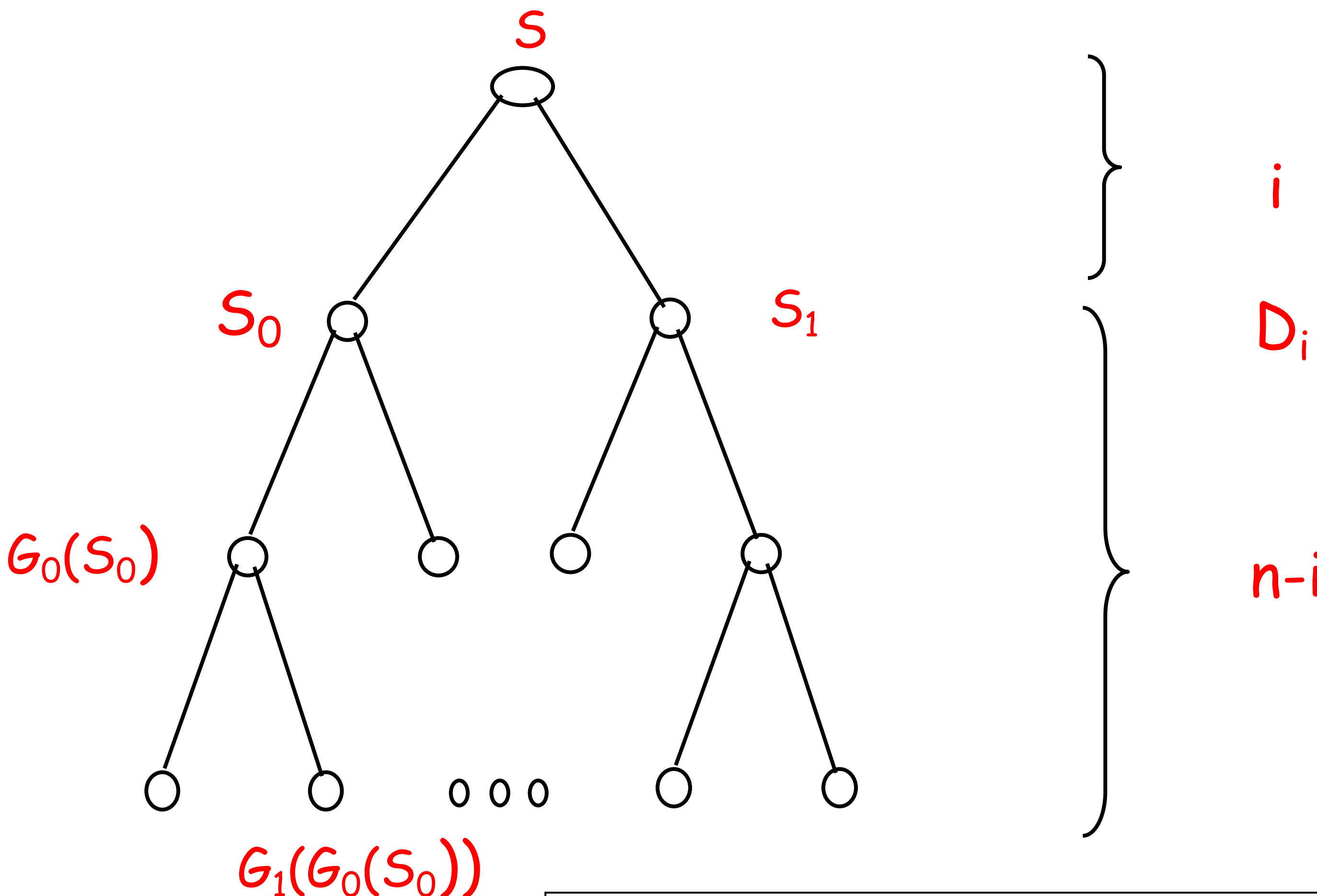
Let p_i = prob ($f \in D_i : D^f(1^n) = 1$).

Then p_1 = prob ($f \in F_n : D^f(1^n) = 1$) and

p_n = prob ($f \in H_n : D^f(1^n) = 1$)

and $|p_n - p_1| > \varepsilon \Rightarrow \exists\ 1 < i < n$ s.t. $|p_i - p_{i-1}| \geq \varepsilon/n = \varepsilon'$

Hybrid



p_i = prob ($g \in D_i$: $D^g(1^n) = 1$ |).

Proof of Security

Now use the distinguisher D & i s.t. $|p_i - p_{i-1}| \geq \varepsilon/n = \varepsilon'$

to distinguish $S \subseteq$ outputs of generator from $S \subseteq U_{2n}$

Algorithm (S) for S set of 2n size strings:

start with empty tree

1. Run Distinguisher $D^f(1^n)$ Phase-1

   On **query** $x=x_1,\ldots,x_n$ to f:

   Pick pair (s_0,s_1) randomly from S

   ignore levels $1\ldots i-1$;

   fill pair of nodes $x_1,\ldots,x_{i-1}0$ and $x_1,\ldots,x_{i-1}1$ at level i

   with pair (s_0,s_1) [unless already filled]

   set $b=x_i$ and answer $G_{x_n x_{n-1} \ldots x_{i+1}}(s_b) = G_{x_n}(G_{x_{n-1}}(\ldots G_{x_{i+1}}(s_b))\ldots)$

2. Run $D^f(1^n)$ Phase-2. if it outputs 1, Output "S random"

   if it outputs 0, output "S pseudo-random"

Claim: $|\text{prob}(S \subseteq G(U_n): A(S)=1) - \text{prob}(S \subseteq U_{2n}: A(S)=1)| > \varepsilon/n$

Easy-Lemma:

$\forall$PPT A, $\forall$Poly P, n sufficiently large,

| Pr [A(S) = 1, $S \subseteq G(U_k)$ s.t |S|=P(n)] −

Pr [A(S) = 1 | $S \subseteq U_{2k}$ s.t. |S|=P(n)] | = neg(n)

Claim 1[|prob (A(S): $S \subseteq G(U_n)$) =1) - prob (A(S):$S \subseteq U_{2n}$)) =1)|>ε']

contradicts Easy-Lemma

Pf:

- if $S \subseteq G(U_k)$ then during the execution of A(S), we are answering the queries of D , in accordance with a function f drawn from D_{i-1} and the probability that D in phase 2 will output 1 is p_{i-1}
- However if $S \subseteq U_{2n}$ then during the execution of A(S) we are answering the queries of D , in accordance with a function f from D_i and the probability that D in phase 2 will output 1 is p_i

Since$|p_i-p_{i-1}| > \varepsilon'$, the response of D will distinguish between $S \subseteq G(U_n)$ and $S \subseteq U_{2n}$ contradicting the easy lemma. QED

Cost of PSRF

- Expensive - n invocations of G
- Sequential
- Deterioration of ε in the reduction: what does that mean?

But does the job!

Corollary

One-way functions (OWF) exist

if and only if

Pseudo-random functions (PRF) exist.

Proof:

⇒Sequence of. reductions.

F OWF Implies there exists hard core B

implies there exists CS PRG

implies there exists PRFs

Each reduction costs: starting with security parameter n, end with $n' = n^C$

⇐ exercise

Prediction Test for Functions? (analogue to Next-Bit Test)

Prediction Test P for functions:

- Requests $Y_i = f(X_i)$ for X_i, i=1..q
- Request Y for $X \notin \{X_1, X_2, \ldots, X_q\}$
- Decide whether given Y is

$$Y = F_S(X) \quad \text{or} \quad Y \in_R \{0,1\}^n$$

- Prediction Test is a

  Statistical Tests for functions.

  Is It Universal?

Prove it : Exercise

Applications of Pseudorandom Functions

- Learning Theory: lower bounds

  Can't learn any class containing (i.e evaluation time is within this class) pseudo-random function

- can replace randomness in. crypto applications

- **Caveat:** what happens when the seed is made public?
  - Can't trust the pseudo randomness any longer

Stateless Encryption Secure Against Chosen Cipher-text Attack

- Generation: Shared secret seed – S
- Encryption: On n-bit message m – -
  - choose n-bit r at random
  - Output ciphertext $(m\oplus f_S(r), r)$
- Decryption: On ciphertext (c,r)
  - Output $m=c\oplus f_s(r)$

Passwords, Calling card id's

- Global secret seed – S

- To generate a password for user M –

  Let $PW_M = f_S(M)$

Identify Friend of Foe

- Global secret seed of the reds is – S

- Challenge m, answer $f_S(M)$

- Security: Even though can obtain polynomial number of $(M, f_S(M))$, can't predict an additional one