Cryptography and Machine Learning: Foundations and Frontiers

Lecture 5: Crypto Basics 2

Feb 19, 2026

Plan for Today

- **Review** last lecture key concepts: Computational Indistinguishability; OWF AND Hard-Core Predicates, Strong PRG from OWF
- **Extenders**
- **Pseudo Random Functions**(PRF)
- **Pseudo Random Permutations**(PRP)
- **Applications** of PRF and PRP
  - Stateless encryption schemes satisfying computational indistinguishability
  - Identify Friend or Foe
- **From Crypto to ML:** PRF in class C imply C is impossible to PAC learn independent of hypothesis class H representation
- **From ML to Crypto**: Learning Parity with Noise (LPN)

Review

Computational Indistinguishability

$k \leftarrow K$

World 0:

$c = E(k, m_0)$

World 1:

$c = E(k, m_1)$

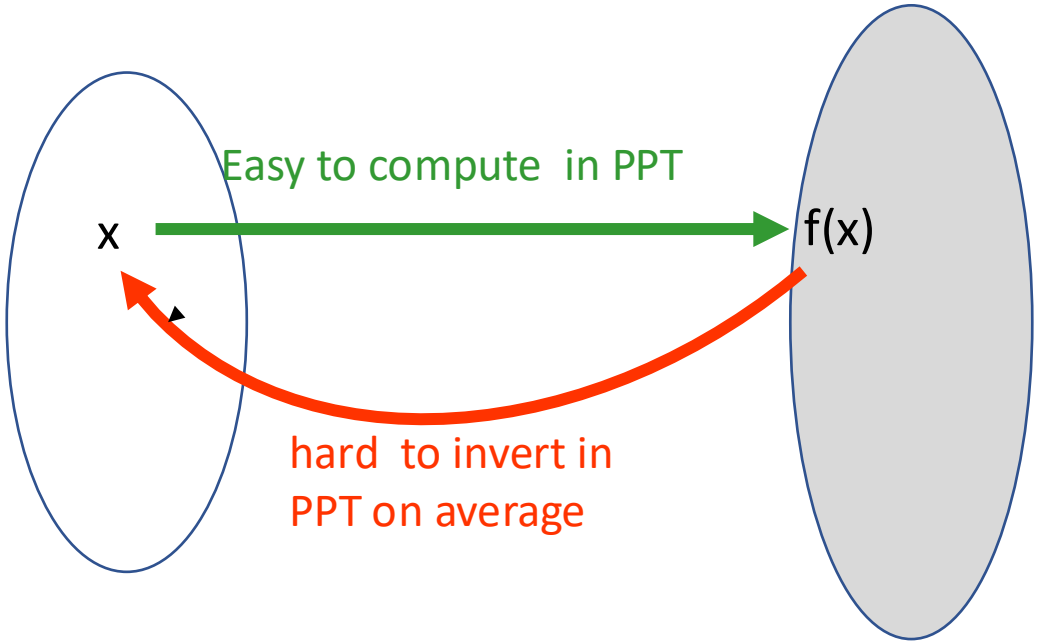
Chosen Plaintext attack: Adv EVE can press a button and get samples from world 0 and world 1

For all PPT EVE, and message spaces M, for all m_0, m_1 *in* $M[1^n]$

$$\Pr[\, EVE(c) = b] \leq \frac{1}{2} + negl(n)$$

Over $k \leftarrow K; b \leftarrow \{0,1\}; c = E(k, m_b)$:

One Way Functions (OWF)



f: $\{0,1\}^* \Rightarrow \{0,1\}^*$ is a one-way function(OWF) if:

- Easy to Evaluate: $\exists$ PPT algorithm A s.t. $A(x)=f(x)$ $\forall$ x.
- Hard to Invert: $\forall$ PPT algorithms Inv, $\forall$ n sufficiently large,

  prob (Inv(y) =x' s.t. y=f(x')) < neg(n)

  $x \in \{0,1\}^n$, y=f(x), coins of Inv

Length preserving OWF: functions s.t. $|f(x|) = |x|$.

One-way Permutations: One-to-one length preserving OWF

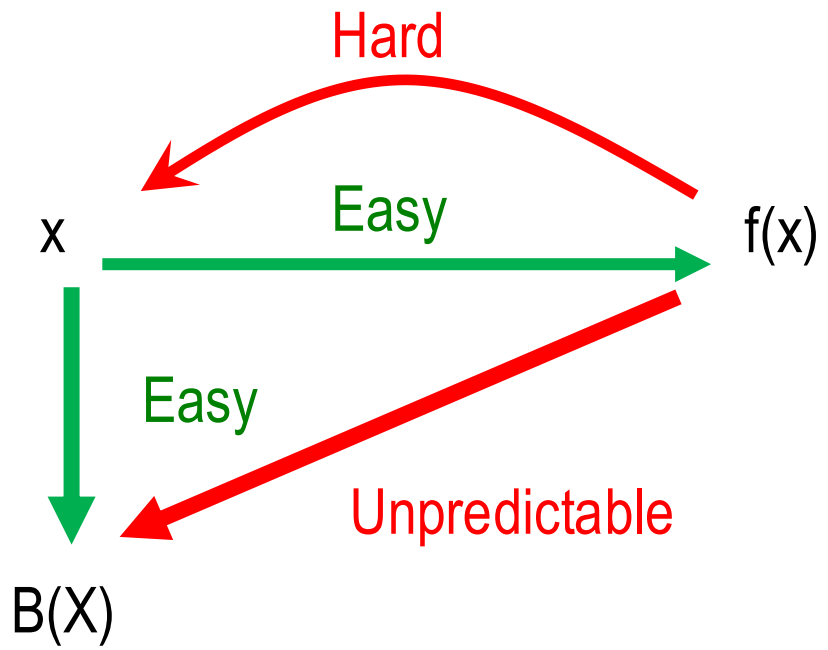
Hard Core Predicates for OWF (HCP)

A hard-core predicate for one-way function f:{0,1}* →{0,1}*
is a Boolean predicate B: {0,1}* → {0,1} such that
B(x) is easy to compute **given x** but hard to predict **given f(x)**

∀ PPT algorithm **P** ("predictor"),

Prob [**P**(y) = B(x)]= ½ + negl(n)

(over x in $\{0,1\}^n$, y=f(x)
and Pred's coins)



Goldreich-Levin (GL) Theorem

Every OWF Has a (Randomized) Hard-Core Bit

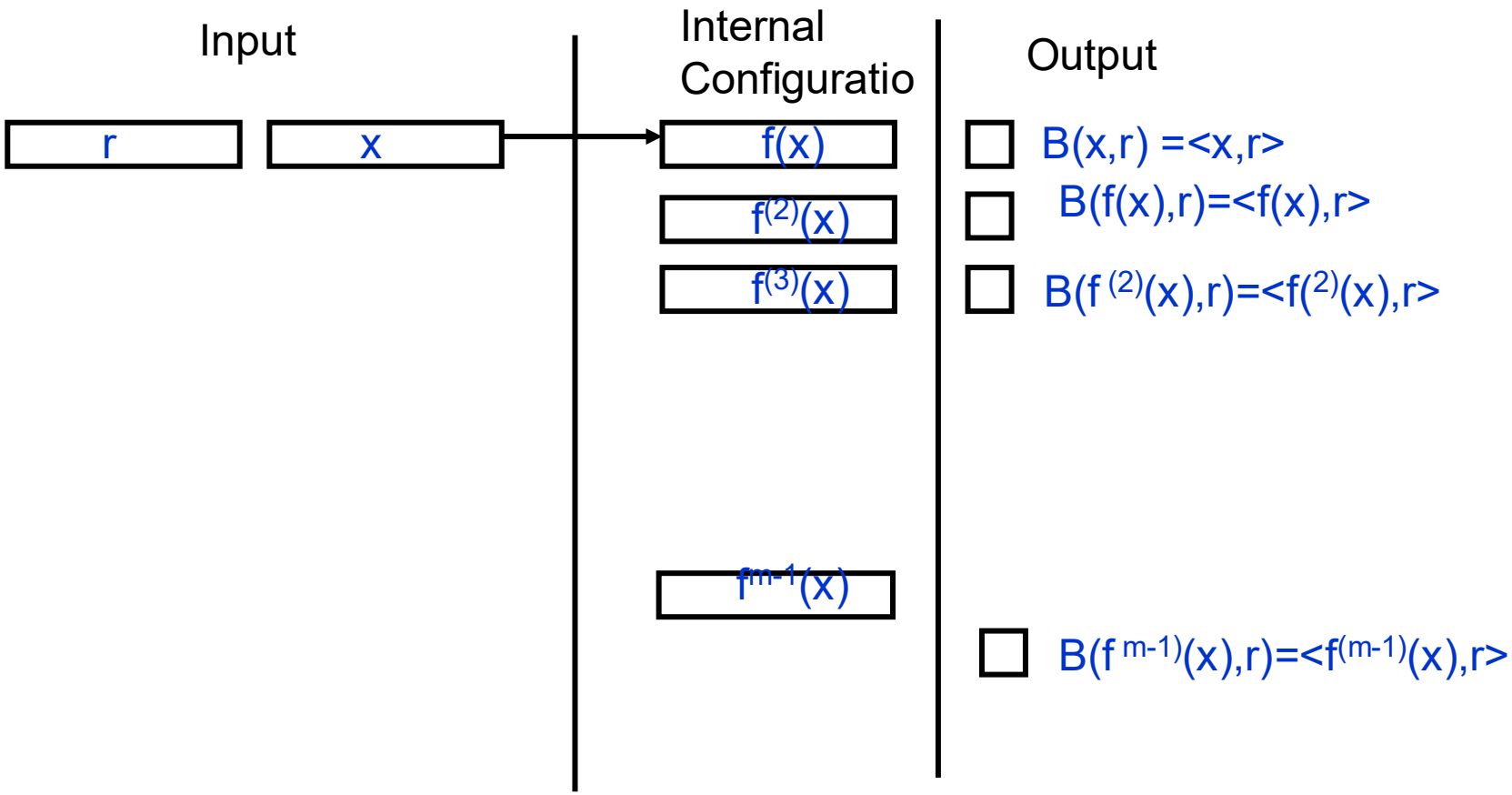
Let $\{B_r : \{0,1\}^n \to \{0,1\}\}$ where $B_r(x) = \langle r, x \rangle = \sum_{i=1}^{n} r_i x_i \bmod 2$

be a collection of predicates (one for each r).
For every one-way function F, every PPT A,
For all n sufficiently large

$$\Pr_{x \leftarrow \{0,1\}^n; r \leftarrow \{0,1\}^n}[Inv(f(x), r) = B_r(x)] \leq \frac{1}{2} + negl(n)$$

Alternative Interpretation : For every one-way function f, there is a related one-way function $f'(x, r) = (f(x), r)$ which has a *deterministic* hardcore predicate which is B(x,r)= <r,x>.

Strong Pseudo Random Generator from any one-way permutation based on Goldreich-Levin Hard Core Bit



- Use the same r and even can make r public

Extenders

Extending by 1 bit is all it takes.

Theorem 1 :

If there exists a strong -PRG that extends n bits to n+1 bits (called an extender), then for every polynomial P, there exists a strong PRG that extends n bits to P(n) bits.

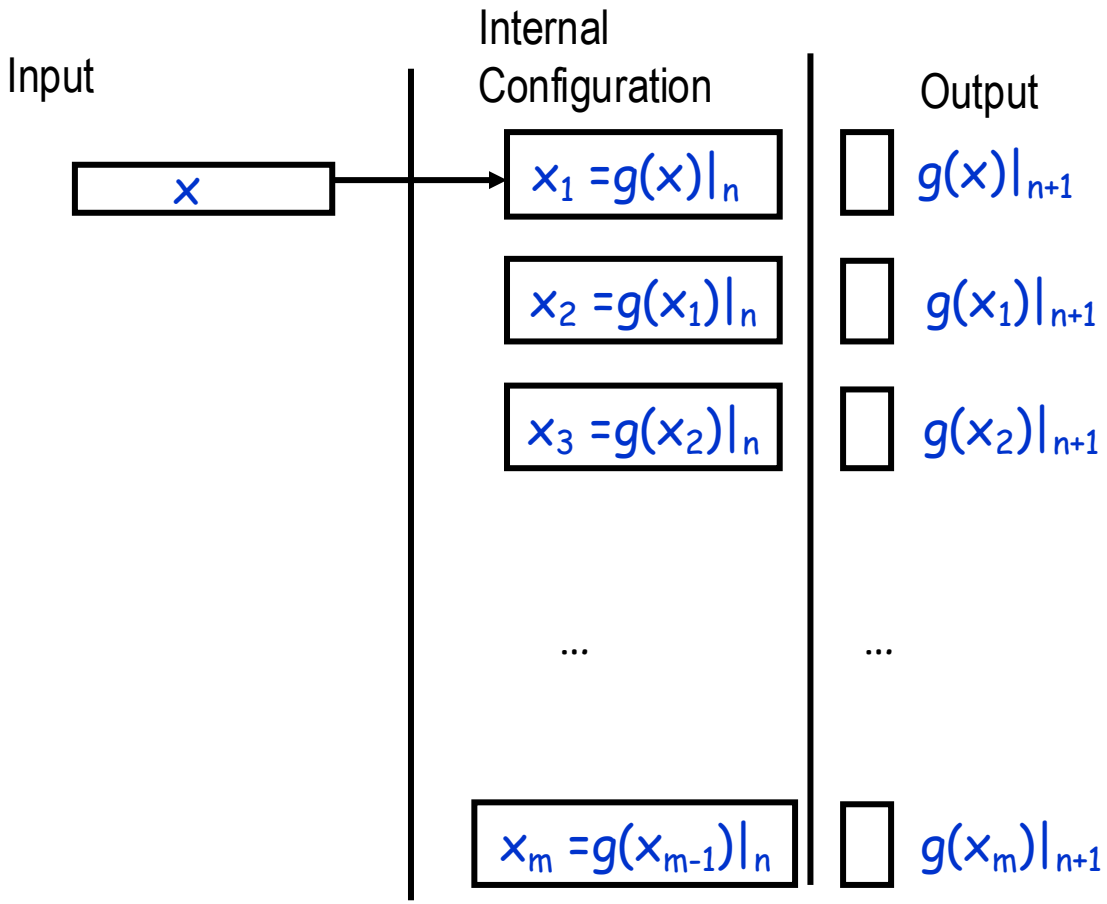
Theorem 2:

Let f be a one-way permutation and B a hard-core predicate for f.

Then G(x)=f(x)||B(x) is a strong PRG from n bits to n+1 bits where |x|=n.

AKA Extender.

Strong PRG with a Single bit extension can be converted to many bit extension (same proof idea)



Building Block:

Single Bit Expanding

strong-PSRG

$g:\{0,1\}^n \mapsto \{0,1\}^{n+1}$

Distinguishers and Predictors

Theorem 1 :

If there exists a strong -PRG that extends n bits to n+1 bits (called an extender), then for every polynomial P, there exists a strong PRG that extends n bits to P(n) bits.

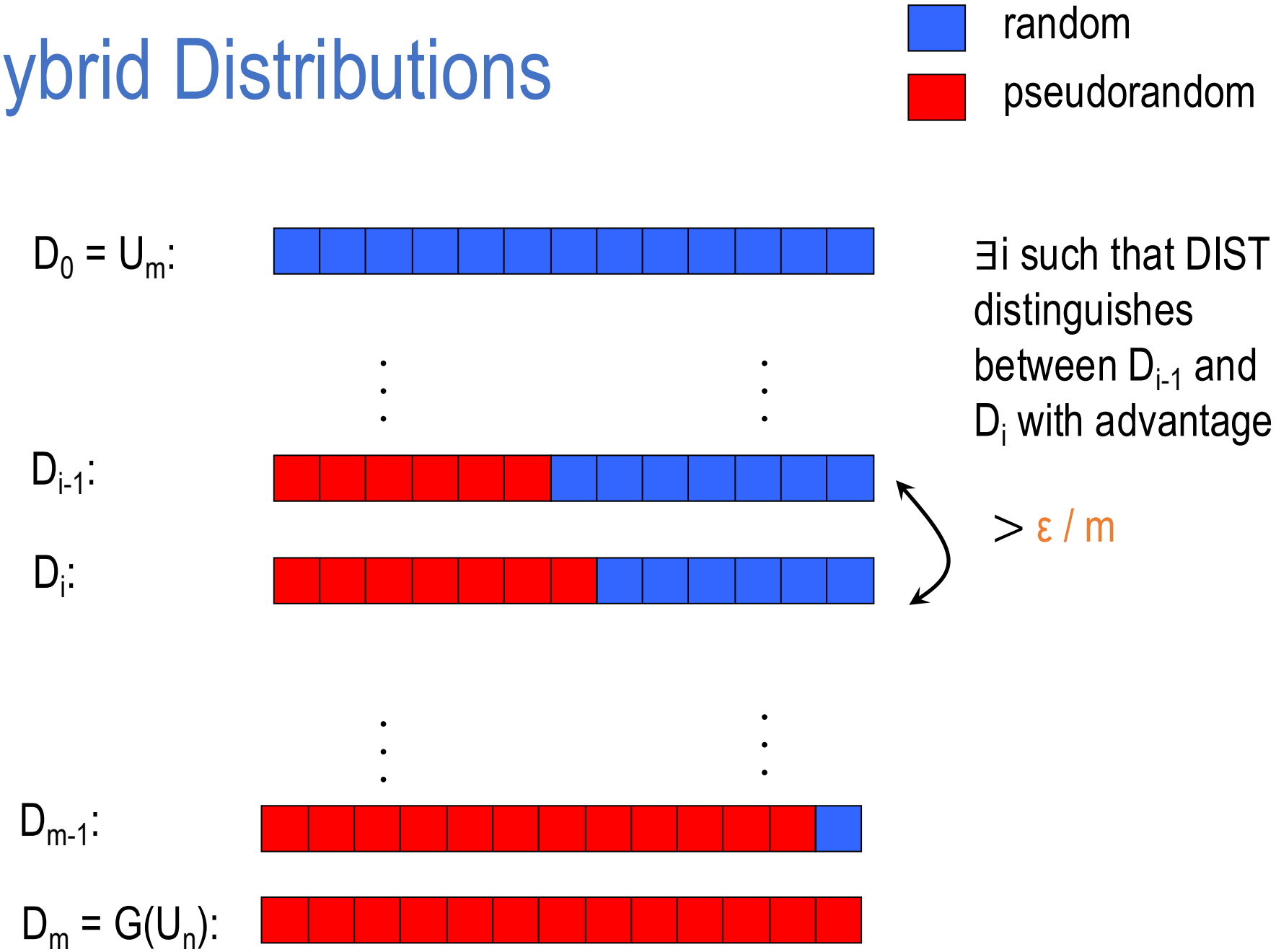
Proof

- Given a distinguisher algorithm DIST with advantage ε, we have:

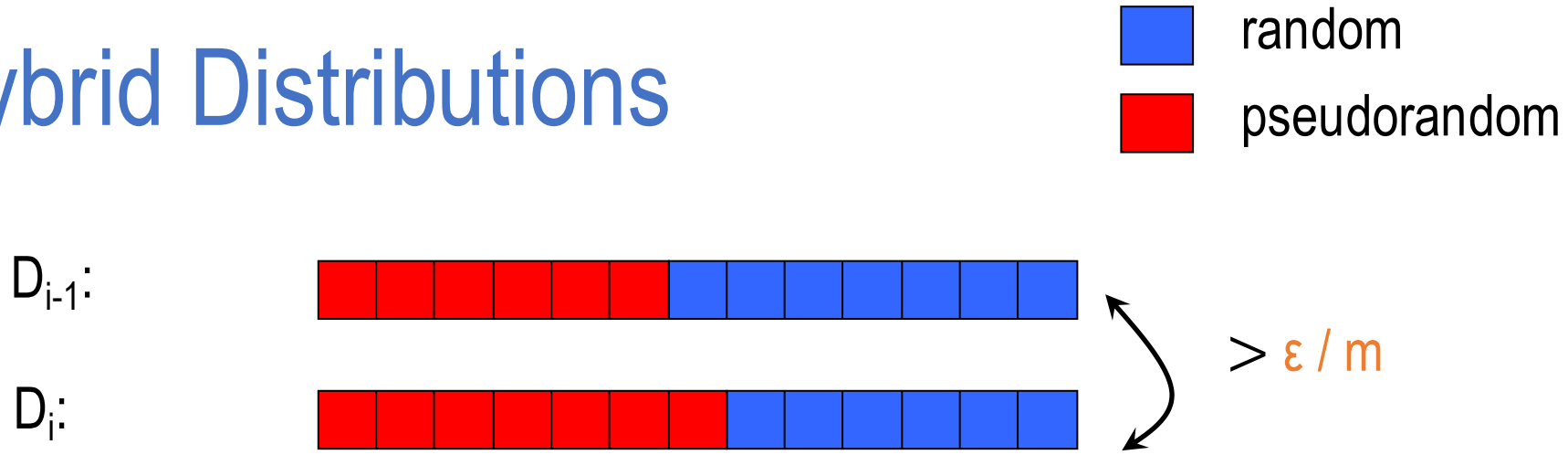
$$\left| \Pr[\, DIST(G(U_n)) = 1] - \Pr[DIST(U_m) = 1] \right| > \varepsilon$$

- Define m+1 **hybrid** distributions.

Hybrid Distributions



Hybrid Distributions



- Define: $p_i = Pr[DIST(y) = 1]$ where $y \leftarrow D_i$
  - Then: $p_0 = Pr[DIST(y) = 1]$ where $y \leftarrow U_m$: and

    $p_m = Pr[DIST(y)=1]$ where $y \leftarrow G(U_n)$
- WLOH this. implies $p_i - p_{i-1} > \varepsilon/m$. [exercise: deal with absolute values]
- THEN: Can design a predictor (next-bit test) PRED for i-th bit of pseudo-random sequences given the (i-1)-bit prefix.

Distinguishing to Prediction: Analysis

- Recall: $p_i - p_{i-1} > \varepsilon/m$

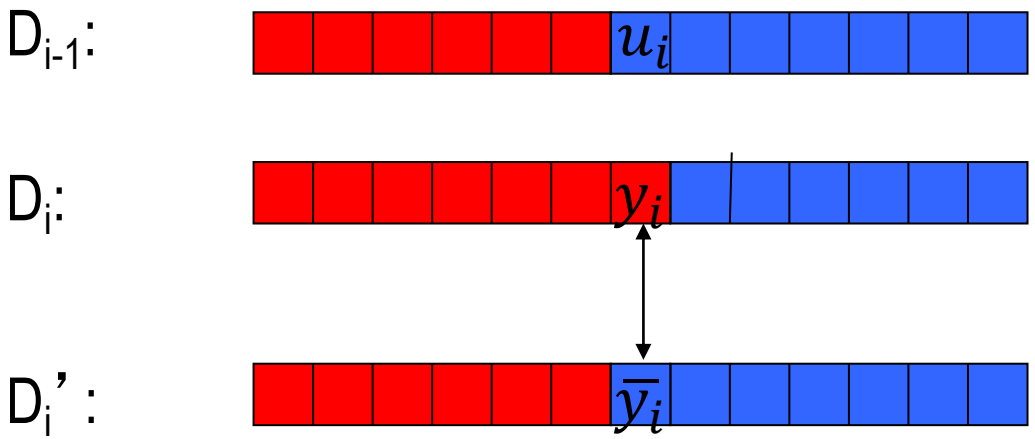
  (i.e prob D outputs 1 higher when i-th bit is from the output of the PRG as opposed to random)

- Let distribution D_i' be D_i with i-th bit flipped and

  $p_i' = \Pr[DIST(y) = 1]$ where $y \leftarrow D_i'$

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

Proof: Exercise.



Predictor PRED for i^{th} bit:

On input: $y = y_1 y_2 \ldots y_{i-1}$

PRED:

- flip a coin: $\mathbf{c} \in \{0,1\}$
- $u = u_{i+1} u_{i+2} \ldots u_m \leftarrow U_{m-i}$
- Run DIST($y\mathbf{c}u$)
- if D outputs 1, output c;
- if D outputs 0, output $\neg c$

(intuition: 1 is a vote for psr bit since $p_i > p_{i-1}$)

Claim:

$$\Pr[\text{PRED}(y_1 \ldots_{i-1}) = y_i] > \tfrac{1}{2} + \varepsilon/m.$$

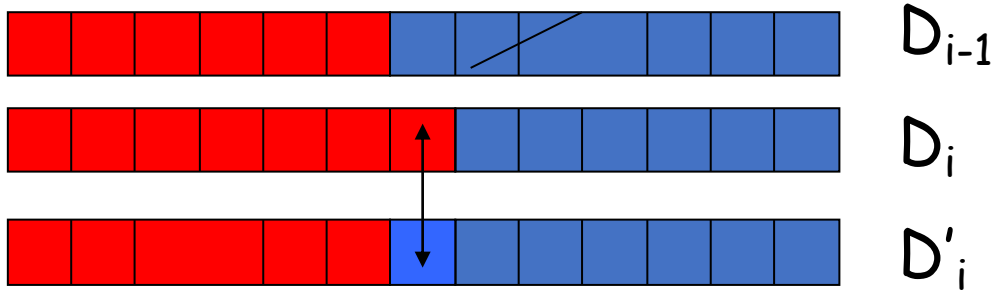
Proof of Claim

$y = y_1 y_2 \cdots y_{j-1}$

$\Pr[y \leftarrow D_i: \mathrm{PRED}(y_{1 \ldots i-1}) = y_j] =$
$\Pr_c[y_i = c \text{ and } \mathrm{DIST}(ycu) = 1] +$
$\Pr_c[y_j = \neg c \text{ and } \mathrm{DIST}(ycu) = 0] =$
$\Pr_c[c=y_i] \Pr[\mathrm{DIST}(ycu) = 1 | y_i = c\ ] +$
$\Pr[|\ \neg c = y_i) \Pr[\mathrm{DIST}(ycu) = 0 | y_i = \neg c\ |) =$
$\frac{1}{2}(p_i + (1 - p_i')) = 1/2 + 1/2(p_i - p_i') =$
$\frac{1}{2} + \frac{1}{2}(p_i - (2p_{i-1} - p_i)) =$
$\frac{1}{2} + (p_i - p_{i-1}) = 1/2 + \varepsilon/m$

We used that

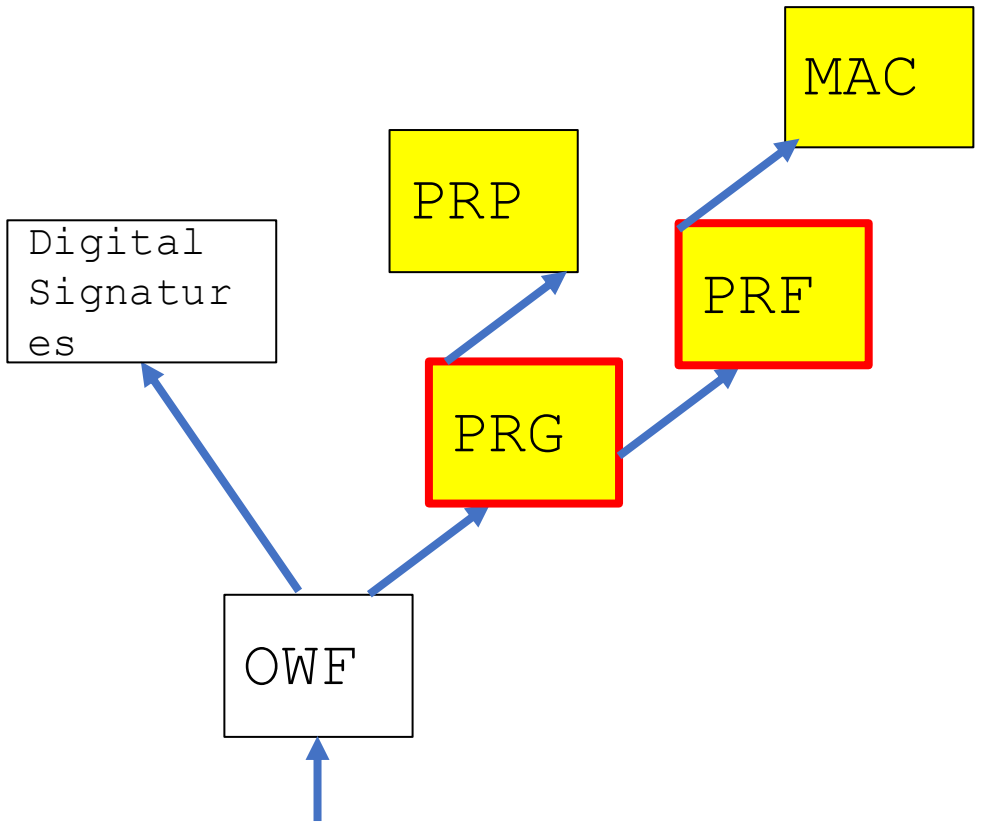
- $p_{i-1} = (p_i + p_i')/2$ and thus $p_i' = 2p_{i-1} - p_i$
- $p_i - p_{i-1} > \varepsilon/m$



Examples

- RSA Based Generators
- Rabin Based Generators
- Discrete Log Based Generators

Some Implications of OWF



Pseudo Random Functions

Recall, **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 i =0

Enc(m): compute and send c=(i,"ith block of G(s)" $\oplus$m); i=i+1

Dec(i, c): set m= "i-th block of G(s)" $\oplus$c

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

Need to maintain state. Is that inherent?

Weak Pseudo Random Functions(PRF): Informal

Collection of indexed functions $\{f_s:\{0,1\}^n \Rightarrow \{0,1\}^n\}_s$

is **weak** pseudo-random if

- **Efficient to compute** $f_s(x)$ given s
- **No PPT adversary can distinguish** between random samples of $(x, f_s(x))$ or (x, U_n) (truly random function values).

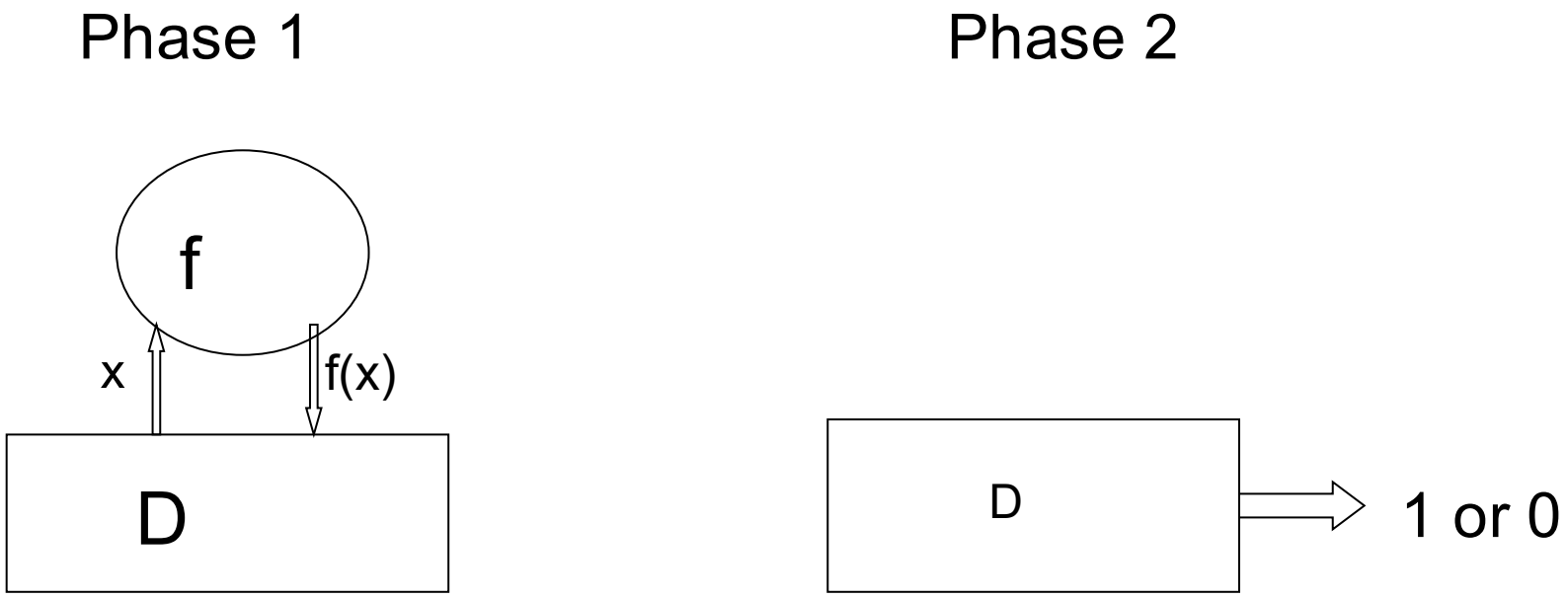
Strong Pseudo Random Functions(PRF): Informal

Collection of indexed functions $f_s:\{0,1\}^n \Rightarrow \{0,1\}^n$

is weak pseudo-random if

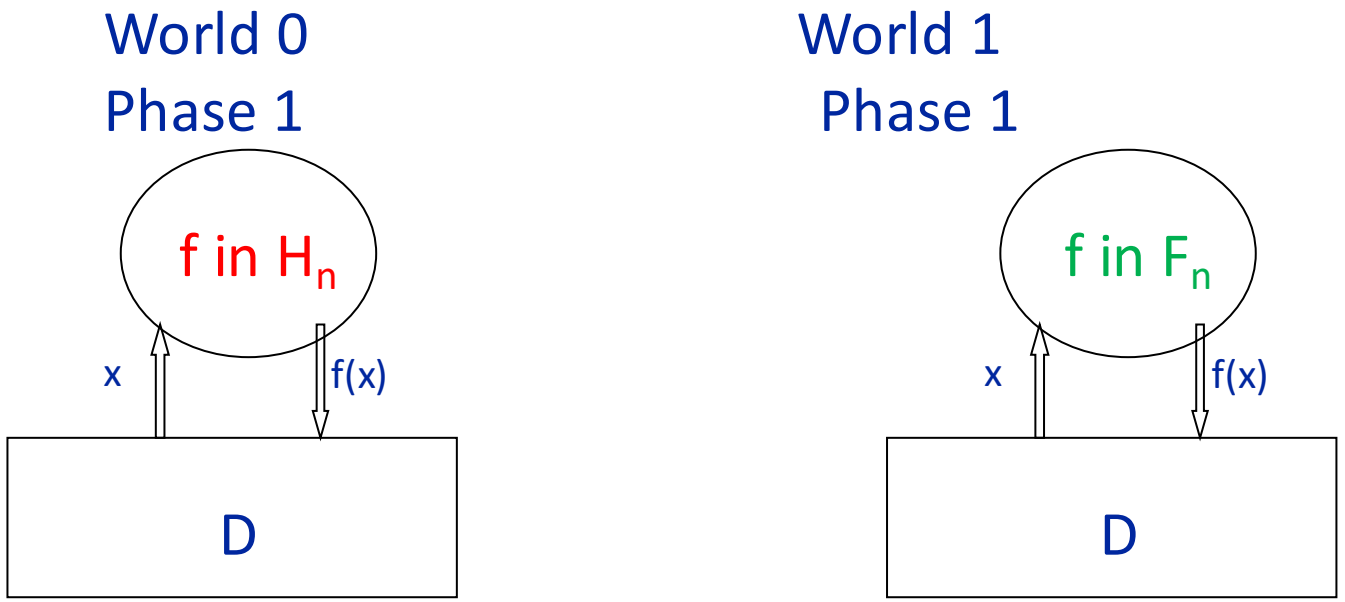
- **Efficient to compute** $f_s(x)$ given s
- **No PPT adversary can distinguish** between $(x, f_s(x))$ for x of adversary choice, or (x, U_n) (truly random function values).

Define: “statistical test” D for 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 computationally indistinguishable from Random F: for all statistical tests D



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

Pseudo Random Functions: Formal

Let $H_n = \{f: \{0,1\}^n \rightarrow \{0,1\}^n\}$ 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

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

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

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

Existence of PSRF's

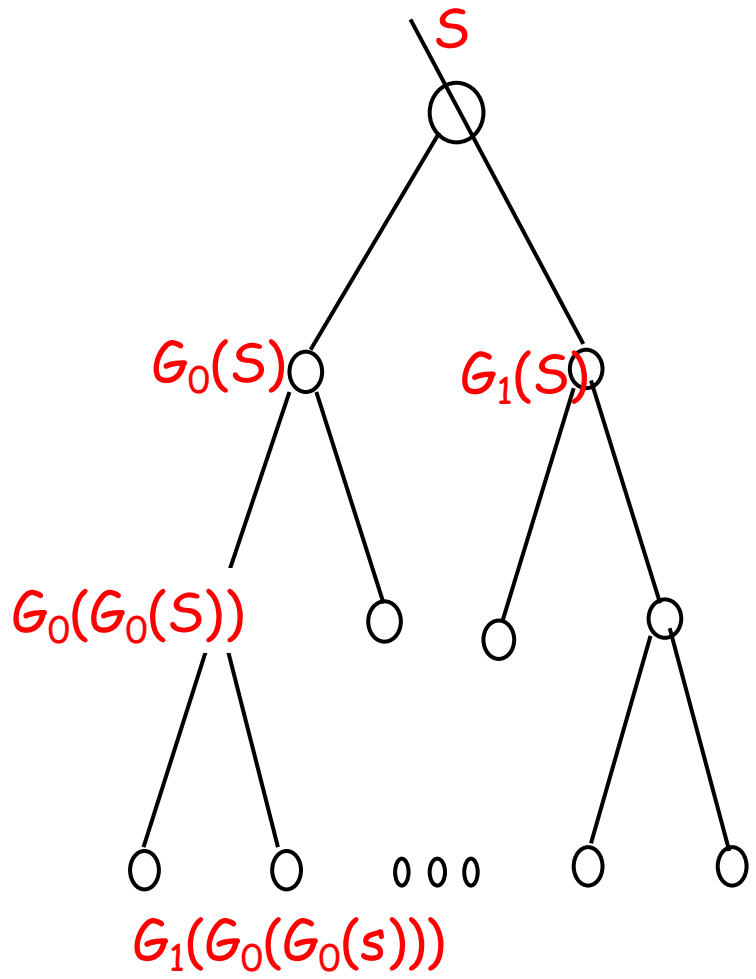
Theorem: If one-way functions exist, then collections of pseudo random functions exist

Proof:

Construction starts from Strong-PRG G s.t.

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

Tree Like Construction



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

$G_1(s)$ = Run a Strong-PRG $G:\{0,1\}^n \to \{0,1\}^{2n}$ on seed s and output the 2nd n output bits

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

$G_{01}(s) = G_1(G_0(s))$ …

$G_x(s) = G_{x_1 x_2 \ldots x_n}(s) = G_{x_n}(G_{x_{n-1}}(\ldots G_{x_1}(s)\ldots))$

A leaf at path x corresponds to a value of the function at x.

PRF's from PRG

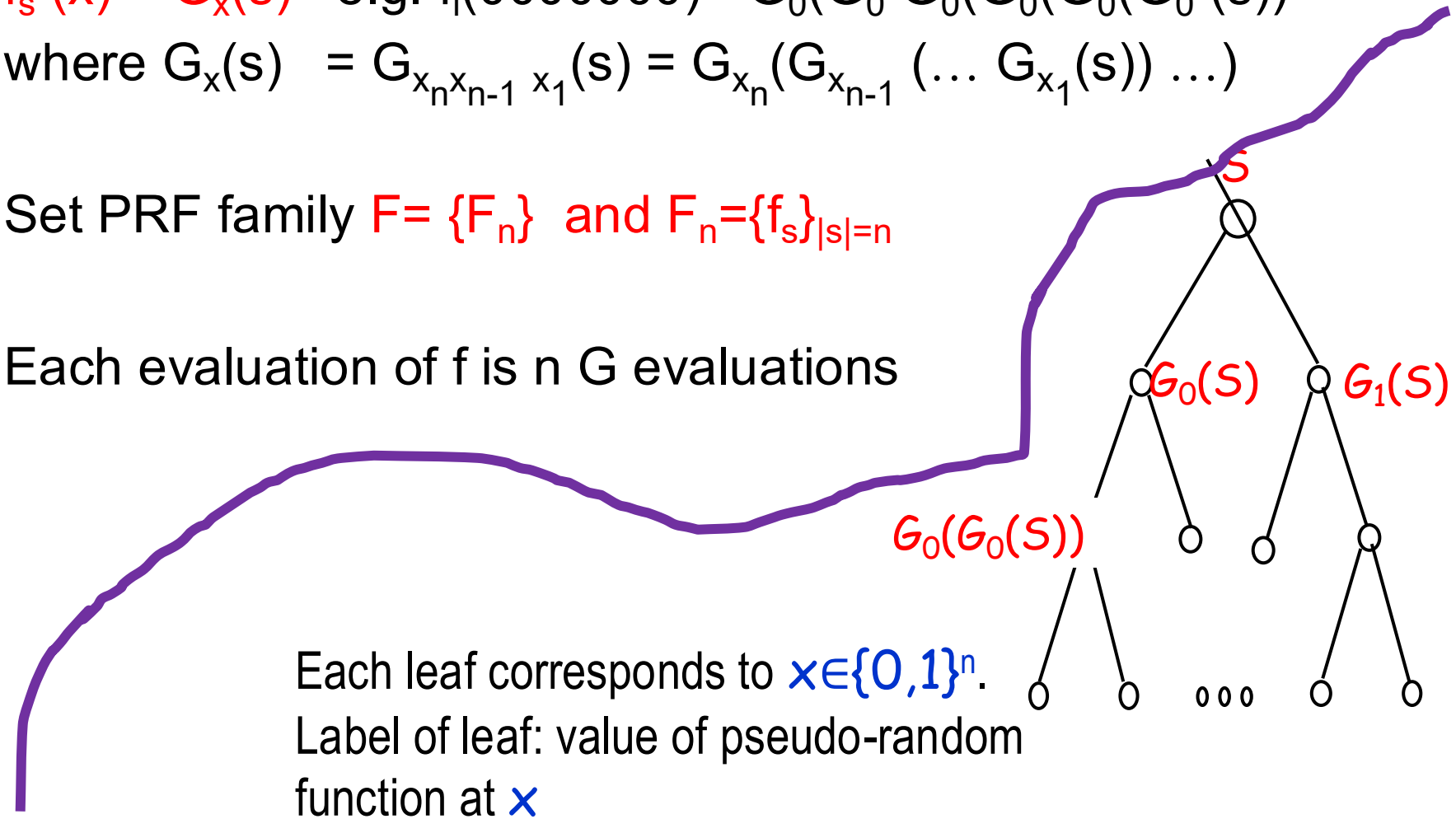
Define

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

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 PRF 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 strong-PRG, then F is strong PRF

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

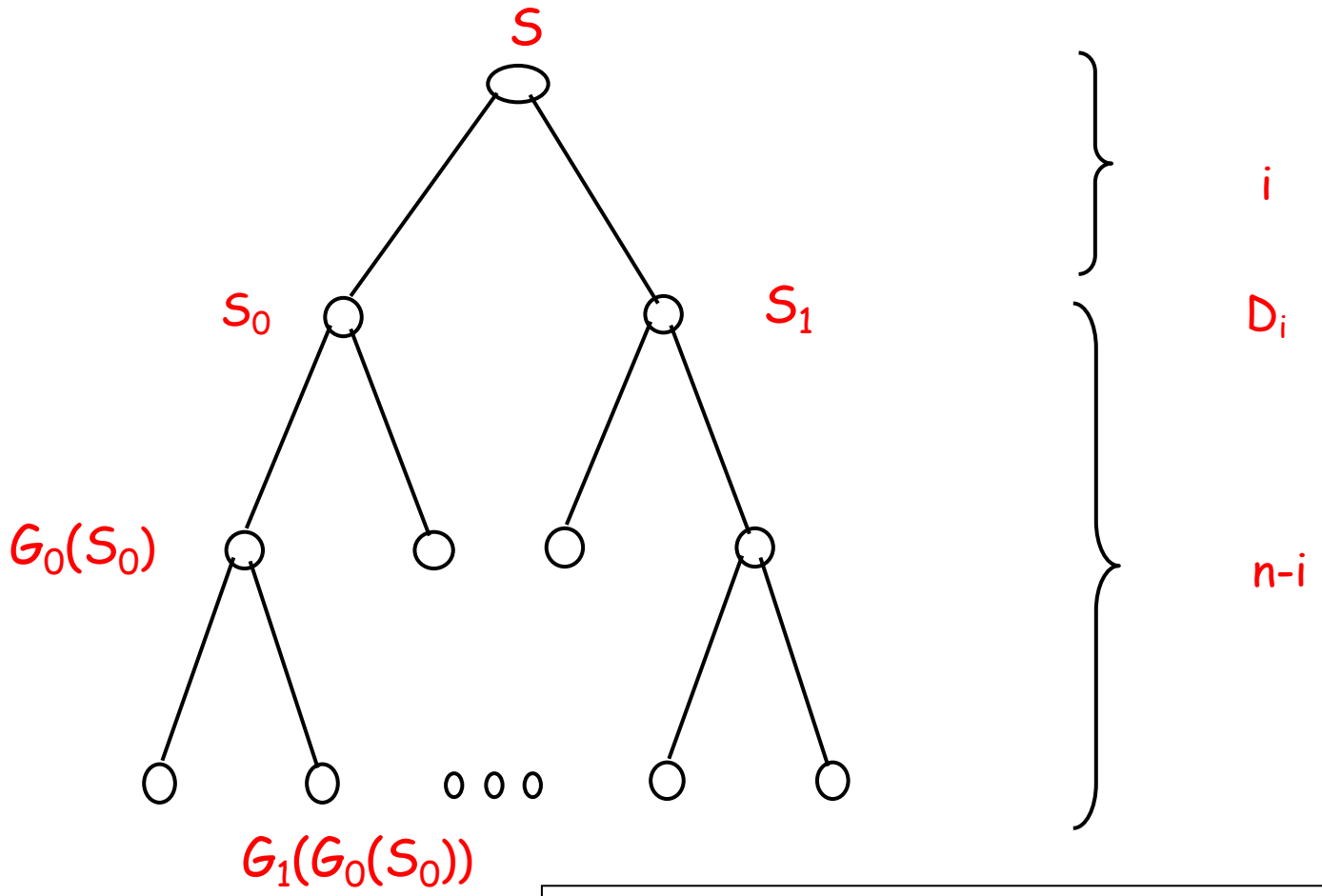
Let $p_i = \text{prob}\,(D^f(1^n) = 1 : f \in D_i)$.

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

$p_n = \text{prob}\,(D^f(1^n) = 1 : f \in H_n)$

and $|p_n - p_1| > \varepsilon \Rightarrow \exists\ 0 < i < n$ s.t. $|p_{i+1} - p_i| \geq \varepsilon/n = \varepsilon'$

Hybrid



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

Proof of Security

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

to distinguish S ⊆ outputs of generator from S ⊆ 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…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:

∀PPT A, ∀Poly P, n sufficiently large,

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

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

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

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

Sequence of Reductions

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 strong PRG
implies there exists strong PRFs

Each has a cost: start with security parameter n, end up with $n'=n^c$, deterioration of ε in the reduction.

But does the job!

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

Prediction Test 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

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

Claim: Is Prediction Test $\cong$ Statistical Tests for functions?

Prove it : Homework

Application: 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 $m=c \oplus f_s(r)$

Application: 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

Application: Passwords, Calling card id's

- Global secret seed – S

- To generate a password for user M –

  Let $PW_M = f_S(M)$

Pseudo Random Permutation (PRP): Formal

Let $H_n = \{f: \{0,1\}^n \rightarrow \{0,1\}^n\}$ all Permutations from n bits to n bits

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

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

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

Theorem: OWF imply PRP

Collision-Resistant Hash Functions(CRHF)

A compressing family of functions $\mathcal{H} = \{h: \{0,1\}^m \to \{0,1\}^n\}$ (where $m > n$) for which it is computationally hard to find collisions.

Def: $\mathcal{H}$ is collision-resistant if for every PPT algorithm A, there is a negligible function μ s.t.

$$\Pr_{h \leftarrow \mathcal{H}}[A(1^n, h) = (x, y): x \neq y, h(x) = h(y)] = \mu(n)$$

Constructions of CRHFs

From the hardness of factoring, discrete log, lattice problems etc.

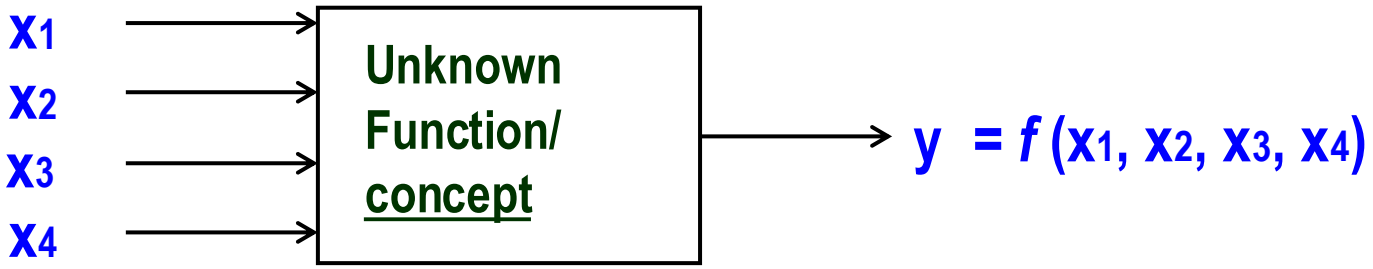
Not known to follow from the existence of one-way functions or even one-way permutations…

"Black-box separations": Certain ways of constructing CRHF from OWF/OWP cannot work.
"Finding collisions on a one-way street", Daniel Simon, Eurocrypt 1998.

Nevertheless, big open problem: OWF/OWP $\stackrel{?}{\Longrightarrow}$ CRHF?

Back to Learning

Target Function/Concept



Example	x_1	x_2	x_3	x_4	y
1	0	0	1	0	0
2	0	1	0	0	0
3	0	0	1	1	1
4	1	0	0	1	1
5	0	1	1	0	0
6	1	1	0	0	0
7	0	1	0	1	0

Can you learn this function in polynomial time?

How?

Must Restrict Concept Class to Learn

Complete Ignorance:
There are 2^{16} = 65536 possible functions over four input features.

We can't figure out which one is correct until we've seen every possible input-output pair.

After observing seven examples we still have 2^9 possibilities for f

Need to restrict the f we are attempting to le

Example	x_1	x_2	x_3	x_4	y
	0	0	0	0	?
	0	0	0	1	?
	0	0	1	0	0
	1	1	1	1	?

- There are $|\mathbf{Y}|^{|\mathbf{X}|}$ possible functions fc(**x**) from the instance space **X** to the label space **Y.**
- typically consider only a *subset* of the functions from **X** to **Y**, called the concept class space **C** .
- $\mathbf{C} \subseteq |\mathbf{Y}|^{|\mathbf{X}|}$

Which Concept Classes are Im(possible) to Learn

Computational Hardness of Learning

- What can and can't we learn efficiently?
  - Let C ={ $f:\{0,1\}^n \rightarrow \{0,1\}$} be a class of concepts
  - Let H={h: $f:\{0,1\}^n \rightarrow \{0,1\}$} be a class of hypotheses (e.g. decision trees, neural nets, linear functions)

Question:

- Can you learn hypothesis h in H, for any concepts/functions in C for all ε,δ s.t.
  - $Prob_{x \text{ in } D}[f(x)=h(x)\}>1-\varepsilon$ with probability $1-\delta$ given polynomial $(1/\varepsilon,1/\delta,n)$ number of random samples of (x,f(x)) for x drawn in distribution D and poly time.
  - How about with membership queries: request f(x) for x of your choice

Two versions to this question:

- **representation dependent** learning (depends on C and H)
- and **representation independent** learning (depends only on C. H contains **all functions** h which can be computed in polynomial time .
- Aside (Schapire: C which is not polynomial time to evaluate cannot be learned)

[Pitt-Valiant]: Its all about representation

Claim: k-term DNF is learnable by k-CNF

Claim [Valiant]:k-CNF is properly learnable by k-CNF.

Corollary: k-term DNF can be learnable by a k-CNF.

PRFs imply Representation Independent Learning is Hard for concepts in P

- Let concept class be $F=\{f_s:\{0,1\}^n \rightarrow \{0,1]\}$ a collection of pseudo random functions
  - There exists a small ckt (description of s) to evaluate $f_s(x)$

Assume F was learnable:

- Namely, there is a polynomial time algorithm which could ask for samples $\{x,f_s(x)\}$ for f_s in F, and output h s.t. h can be evaluated in polynomial time and $Prob_x[f_s(x)=h(x)) > 1-\varepsilon$ w.p $1-\delta$
- But h cannot agree with a truly random f more than ½ of the time.
- Thus, h can serve as distinguisher between f_s in F and a truly random f.

Thus F is impossible to learn regardless of representation of h.

From Learning to Cryptography: Interesting Consequences

- PRFs cannot be implemented by linear threshold functions as can be learned
- PRF cannot be implemented by polynomial size formula in DNF form, as can be learned for uniform distribution
- etc

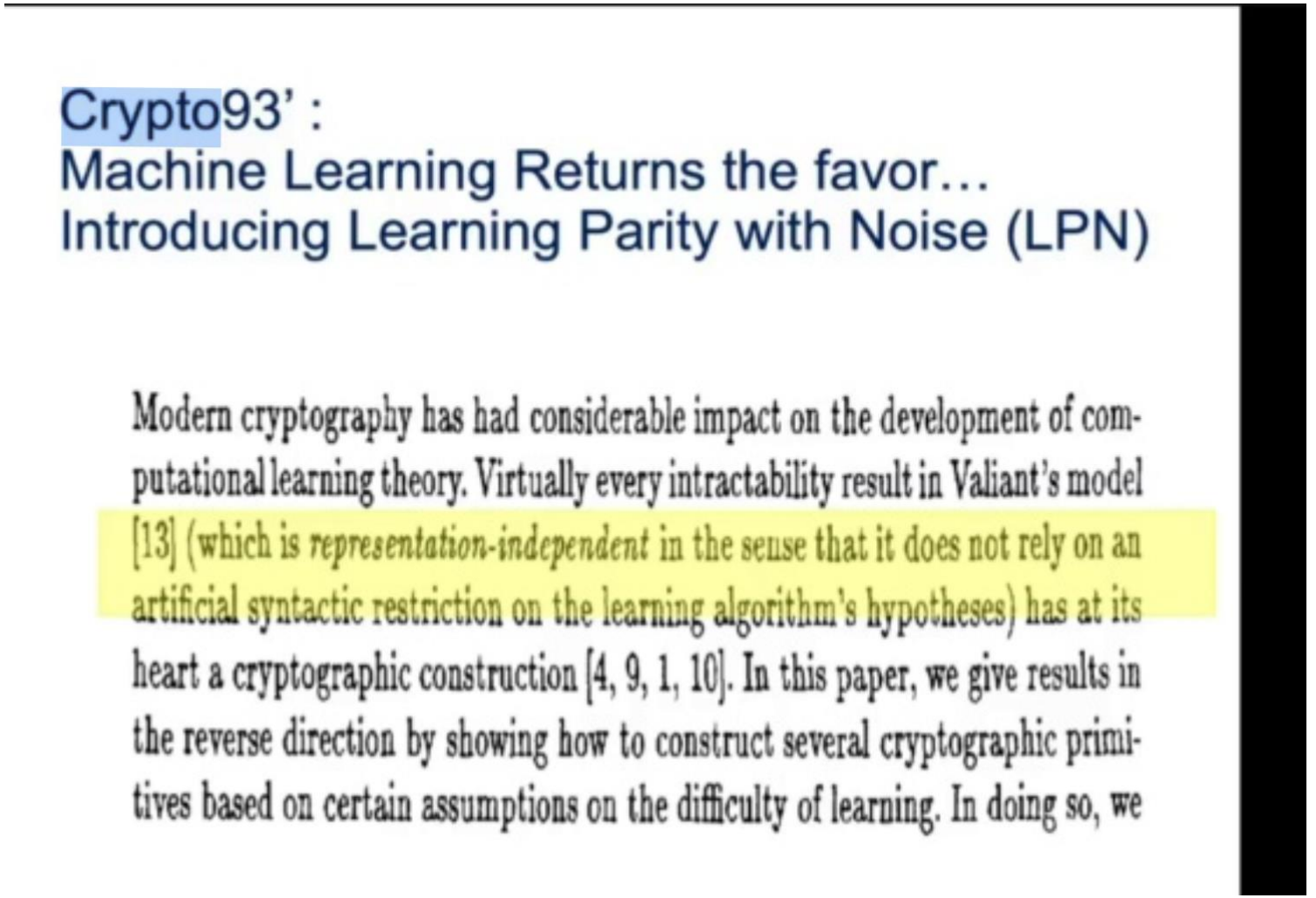
PRFs and Learning: Still hard to Learn even when

PRFs
Constrained
Aggregate
Key homomorphic
Related-key secure

Learning is HARD
Can label some of the examples of the concept yourself
Can get sums of labels in an interval you specify
Labels satisfy arithmetic
Can receive answers to Queries on related concepts

Learning Parity with Noise

Hard ML Problems in Use of Cryptography



Crypto93' :
Machine Learning Returns the favor…
Introducing Learning Parity with Noise (LPN)

Modern cryptography has had considerable impact on the development of computational learning theory. Virtually every intractability result in Valiant's model [13] (which is *representation-independent* in the sense that it does not rely on an artificial syntactic restriction on the learning algorithm's hypotheses) has at its heart a cryptographic construction [4, 9, 1, 10]. In this paper, we give results in the reverse direction by showing how to construct several cryptographic primitives based on certain assumptions on the difficulty of learning. In doing so, we

Learning Parity with Noise [BFKL93]

- Let s be a secret vector in Z_2^n
- $LPN_{n,\rho}$: Given an arbitrary number of "noisy" equations in s, find s?

$$
\begin{aligned}
0s_1+s_2+s_3+\ldots+sx_n &\approx 0 \bmod 2 \\
1s_1+0s_2+s_3+\ldots+1s_n &\approx 1 \bmod 2 \\
1s_1+1s_2+0s_3+\ldots+0s_n &\approx 0 \bmod 2 \\
1s_1+1s_2+0s_3+\ldots+0s_n &\approx 0 \bmod 2 \\
&\ldots \\
0s_1+1s_2+0s_3+\ldots+0s_n &\approx 1 \bmod 2
\end{aligned}
$$

Add noise vector ε:
Bernulli with ρ
$\Sigma|e_i|$ over Z is small

C is efficiently learnable in the presence of random classification noise under distribution D: if $\exists$an algorithm A s.t $\forall\varepsilon, \delta > 0, \eta < \frac{1}{2}$ & target $f \in C$, output h

1) Prob $_{x \text{ in } D}[h(x)<f(x)+\varepsilon]>1-\delta$ s.t. x labeled from c with noise η.
2) A must run in time polynomial in n, $1/\varepsilon$, $1/\delta$, $1/\eta$

Algorithms to Solve LPN

- Let s be a secret vector in Z_2^n
- $LPN_{n,\rho}$: Given an arbitrary number of "noisy" equations in s, find s?

$$0s_1+s_2+s_3+\ldots+sx_n \approx 0 \bmod 2$$
$$1s_1+0s_2+s_3+\ldots+1s_n \approx 1 \bmod 2$$
$$1s_1+1s_2+0s_3+\ldots+0s_n \approx 0 \bmod 2$$
$$1s_1+1s_2+0s_3+\ldots+0s_n \approx 0 \bmod 2$$
$$\ldots$$
$$0s_1+1s_2+0s_3+\ldots+0s_n \approx 1 \bmod 2$$

Add noise vector ε: Bernulli with ρ

$\Sigma|e_i|$ over Z is small

✓ Best-Algorithm[BKW03]: Best known time $2^{O(n/\log n)}$

✓ Worst case to average reductions [BLVW18] for noise: 1/2-1/poly(n)

Learning with Errors [Regev05]

- Let s be a secret vector in Z_q^n
- $LWE_{n,\alpha}$: Given arbitrary number of "noisy" equations in s, find s?

$$
\begin{aligned}
14s_1 + 15s_2 + 5s_3 + 2s_4 &\approx 8 \pmod{17}\\
13s_1 + 14s_2 + 14s_3 + 6s_4 &\approx 16 \pmod{17}\\
6s_1 + 10s_2 + 13s_3 + 1s_4 &\approx 3 \pmod{17}\\
10s_1 + 4s_2 + 12s_3 + 16s_4 &\approx 12 \pmod{17}\\
9s_1 + 5s_2 + 9s_3 + 6s_4 &\approx 9 \pmod{17}\\
3s_1 + 6s_2 + 4s_3 + 5s_4 &\approx 16 \pmod{17}\\
6s_1 + 7s_2 + 16s_3 + 2s_4 &\approx 3 \pmod{17}
\end{aligned}
$$

Add noise e: each $|e_i|$<small Gaussian in [q/2,-q/2], std dev

- ✓ Equivalent to approximating the size of the shortest vector in a worst-case integer lattice [Reg05, BLPRS13]
- ✓ Worst Case to Average [Ajtai98]
- ✓ Best known algorithm still $2^{O(n/\log n)}$ [BKW05]
- ✓ Revolutionary: Homomorphic Encryption, Functional/Attribute Encryption, and much more