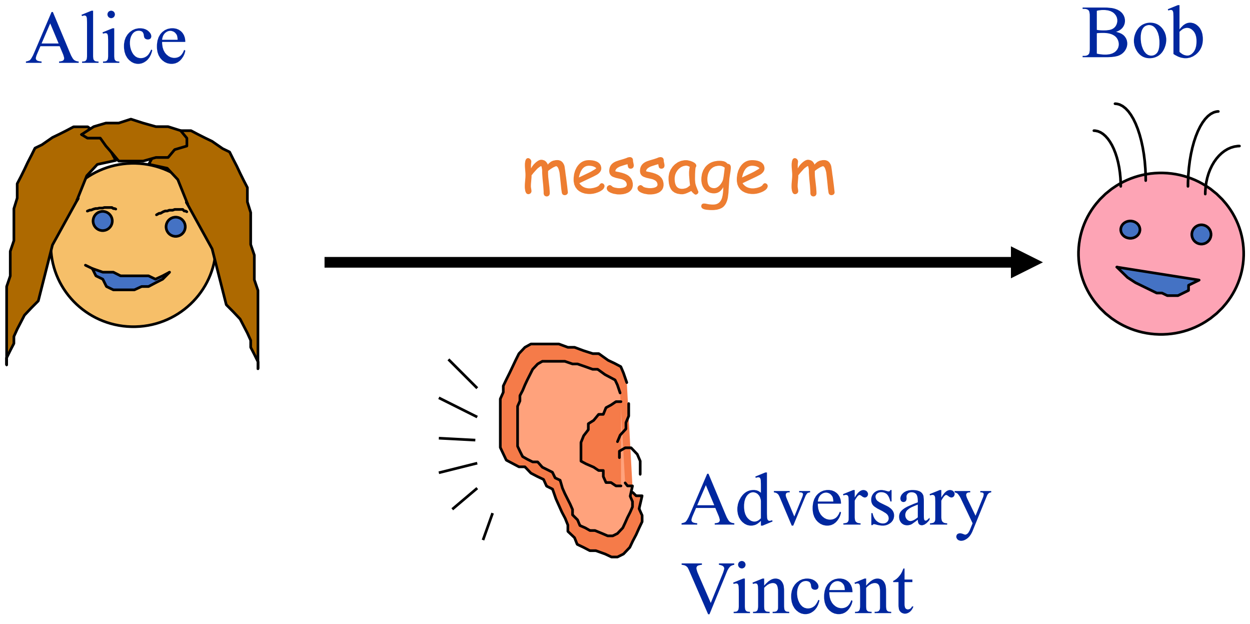
Cryptography and Machine Learning: Foundations and Frontiers

Lecture 4: Crypto Basics 1

Feb 12, 2026

Secret Communication



Alice

Bob

message m

Adversary
Vincent

Encryption scheme

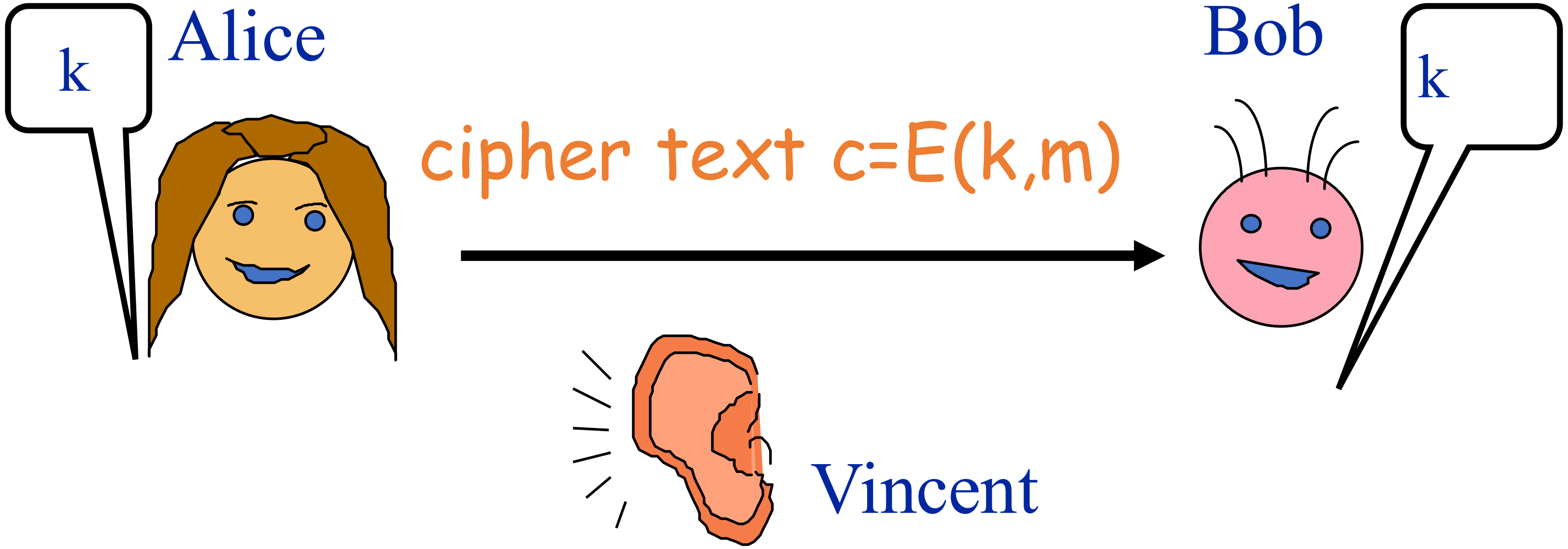
- An encryption scheme (G,E,D) is 3 probabilistic algorithms:
  - key generation G(1^n) outputs secret key k of length n

    [n - called the security parameter]
  - Encryption E(k,m) outputs ciphertext c
  - Decryption D(k,c) outputs plaintext m
- Requirements:
  - Correctness: D(k,E(k,m)) = m ∀m in message space M
  - **Security Definition**…with respect to adversaries

- K = key probability space Prob[K=k]
- *M* = message probability space, Prob[M=m]
- C = ciphertext probability space. Prob[C=c] =Prob[E(K,M)=c]

  (over K and M and coins of E)

Secret Communication



Alice and Bob met to agree on a secret key k in G(1^n)

Shannon ‘49:
Perfect Secrecy Theory

Adversary: unbounded computationally,
security analysis is **information theoretic**

What Does the Adversary Know?

- Kerckohoff Law: A cryptographic system should be secure even if everything about the system (the algorithms G,E and D) is known to the adversary except for the key and the randomness used by legal users in the course of running E & D
- Good for ML too…
- Ciphertext Only: Can see c transmitted over an insecure channel

Perfect Secrecy (aka Shannon secrecy)

(G,E,D) satisfies

Shannon-secrecy if:

∀ probability distribution over M,

∀ c ∈C, ∀ m ∈ M

$\Pr[M\text{=m}] = \Pr[M\text{=m} \mid E(K,M)\text{=}c]$

A-priori = A-posteriori

:When a r.v. (random variable) Appears in a context of prob statement., the prob is taken over the choices of the r.v.

Slight Notational Abuse: All capital letters denote r.v's and prob distribution at the same time

Note : Adversary is not explicitly used In the definition but Is implicitly there computing probabilities…

Perfect Indistinguishability (Alternative Definition)

(G,E,D) satisfies

Perfect indistinguishability if :

∀Probability distribution over *M*

∀ m_0, m_1 ∈ *M*,

∀c ∈ C

Prob [$E(K,m_0)=c$] = Prob [$E(K,m_1)$=c]

Let EVE be an unbounded adversary,
EVE can't distinguish worlds apart

```
World 0:
```

k ← $G(1^n)$

$c = E(k, m_0)$

```
World 1:
```

k ← $G(1^n)$

$c = E(k, m_1)$

$\forall m_0, m_1$

$\text{Prob}_k(\text{EVE } (c)=1 \text{ on } c \in E(K,m_1)) =$

$\text{Prob}_k(\text{EVE } (c)=1 \text{ on } c \in E(K,m_0))$

The Definitions are Equivalent

Theorem:

(G,E,D) satisfies Perfect indistinguishability

if and only if

(G,E,D) satisfies Perfect secrecy.

Proof: Simple use of Bayes Theorem

In notes

Shannon Secrecy is Achievable via One Time Pad

One Time Pad: G chooses k at random in $\{0,1\}^n$

$E(k,m)=k\oplus m$,

$D(k,c)=k\oplus c$

How about using one-time pad to send more than one message?

Q: Would it preserve Shannon Secrecy?
A: No, Consider the case of two messages each of length n, each encrypted by "xoring" the message with the same sk.

Shannon Theorem: For any perfect secrecy schemes, $|K| \geq |M|$
Namely, the key is as long as the message.

One Time Pads

Disadvantage

- The size of the key is huge: as many key bits as message bits and need to know in advance how many message bits

- Receiver needs to know which key goes with which ciphertext (some synchronization or state)

Advantage

- By Shannon's Theorem, it is **IMPOSSIBLE** TO DO ANY BETTER

How to get around Impossibilities (first lecture)

- ✓Weaken the adversary model
- Weaken your definition of security
- Find new class of assumptions

Prepare for Worst Case Adversary Strategy

AI systems are VERY attractive targets

- **Adversarial modeling:**
  - ➢ **Prepare** for **worst case** adversary
  - ➢ **Probabilistic Polynomial Time**

Probabilistic Polynomial Time algorithms (PPT)

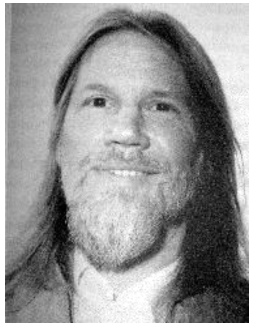
- A runs in polynomial time $O(n^c)$ for some $c>0$.
  where n is its input length (security parameter)

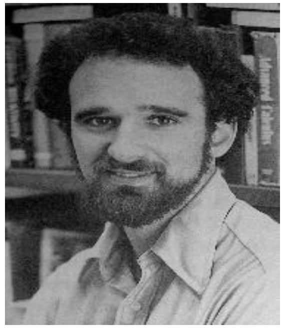
- A is randomized: can flip fair coins
  - Las Vegas: $\forall$input, A is correct or
    with negligible probability A outputs $\perp$

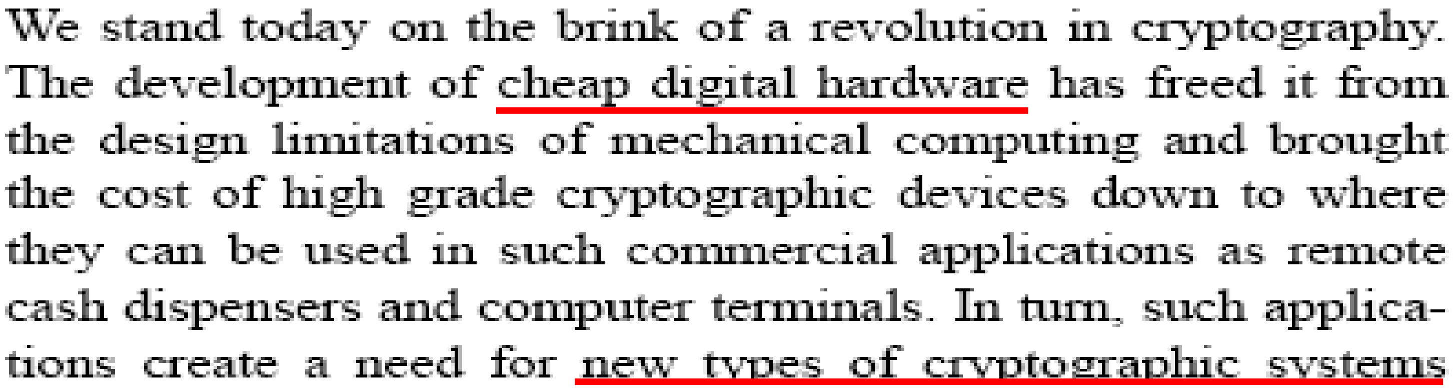
  - Monte Carlo: $\forall$input, A is correct
    With all but negligible probability

Modern Cryptography

1976, New Directions in Cryptography







“ We stand today on the brink of a revolution in cryptography. The development of cheap digital hardware has freed it from the design limitations of mechanical computing and brought the cost of high grade cryptographic devices down to where they can be used in such commercial applications as remote cash dispensers and computer terminals. In turn, such applications create a need for new types of cryptographic systems ”

W. Diffie, M. Hellman, “New Directions in Cryptography”, 1976.

The Adversary

Polynomial time

Lessons from Impossibilities (first lectre)

✓Weaken the adversary model

- Weaken your definition of security

- Find new class of assumptions

Encryption scheme (Revisit Def)

An encryption scheme (G,E,D) is a triplet of PPT algorithms s.t.

- key Generation $G(1^n)$ outputs pairs of keys (sk,pk) of length n

  [n is also called the security parameter]
- Encryption E(pk,m) outputs ciphertext c
- Decryption D(sk,c) outputs plaintext m'

Requirements:

- Correctness: D(sk,E(pk,m)) = m ∀m with high probability over (implicit) coins of G, E and D

- Security **Computational Indistinguishability**

Computational Indistinguishability

(G,E,D) satisfies computationally indistinguishability if

$\forall$PPT EVE,

$\forall$ PPT sampleable M,

$\forall$ m_0, m_1 *in M(1ⁿ),*
1

|Prob$_k$(EVE (c)=1 on c $\in$ E(K,m_1)) - Prob$_k$(EVE (c)=1 on c $\in$ E(K,m_0))|< negl(n)

World 0:

k ← G (1^n)

m_0 ← M (1^n)

$c = E(k, m_0)$

World 1:

k ← G(1^n)

m_1 ← M(1^n)

$c = E(k, m_1)$

PPT EVE can't distinguish t worlds apart

Important Notion: Negligible Functions

Functions that grow slower than 1/p(n) for any polynomial p.

Definition: A function $\mu: \mathbb{N} \rightarrow \mathbb{R}$ is **negligible** if

for every polynomial function p,

there exists an n_0

s.t.

for all $n > n_0$:

μ(n) < 1/p(n)

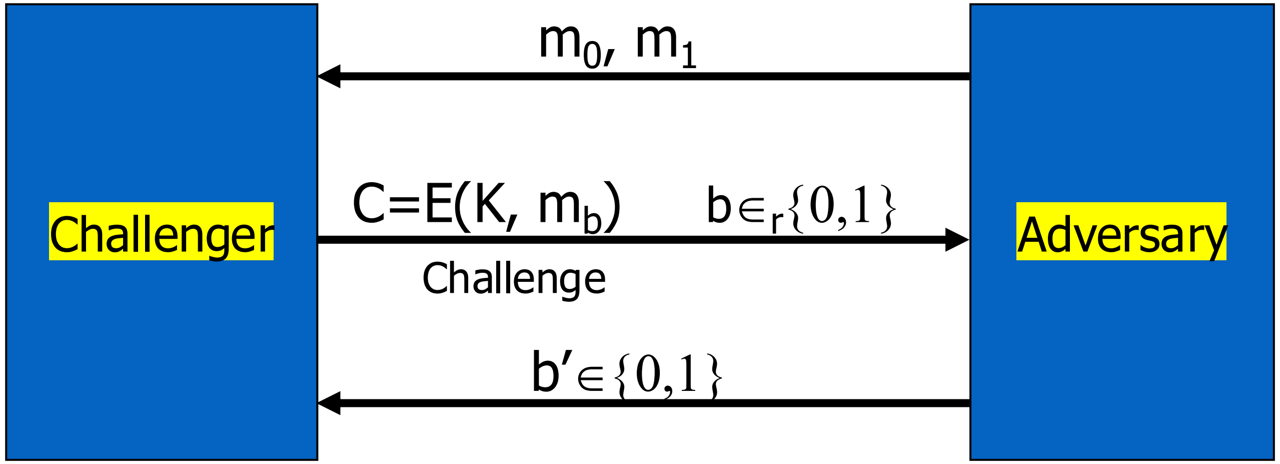
Events that occur with negligible probability look ***to poly-time algorithms*** like they ***never*** occur.

Notation: negl(n) denotes a function which is negligible in n

Game Formulation: Computational Indistinguishability

No PPT adversary can **win** the following game:

(Public-Key Encryption Scheme : even if Adv is given access to pk)



Adversary **wins**: if b=b’

better then ½+ 1/P(n) for some polynomial P (i.e non-negligible (security parameter)

Now Can Ask New Questions

Can A and B meet to agree on key and subsequently exchange P(n) messages securely, where P is any polynomial?

Yes

Can A and B exchange messages without meeting

Yes

How?

Lessons from Impossibilities (first lectre)

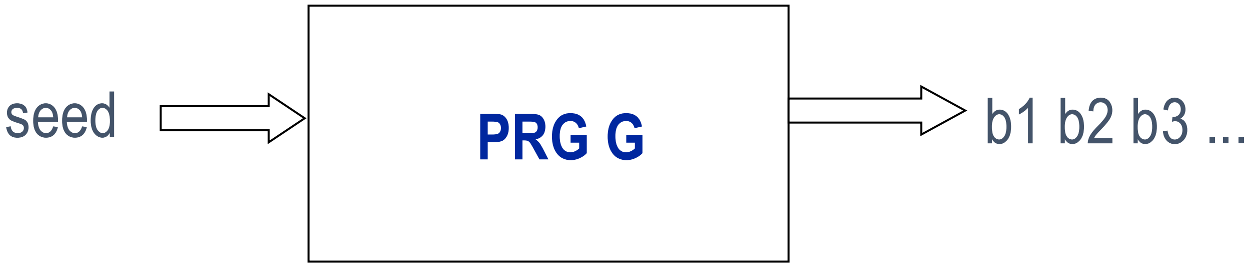
- ✓Weaken the adversary model
- ✓Weaken your definition of security
- Make assumptions

Ambitious Plan for next 2 lectures

- New Primitive: Secure Pseudo Random Generators (PRG)
- PRG imply Encryption schemes that satisfy Computational Indistinguishability with |keys| << |message bits|
- Build PRG from One Way Functions
- Candidate One Way Functions
- New Primitive: Secure Pseudo Random Functions (PRF)
- Applications of PRF
  - ∃ concepts which are **not** PAC learnable

Pseudo-random Generators

Informally: **Deterministic** Programs that stretch a "truly random" seed into a (much) longer sequence of **"seemingly random"** bits.



What about Sources of True Randomness for the seed?

1) Specialized Hardware: e.g. Transistor noise

2) User Input: Every time random number used,
user is queried

Usually biased, but can “extract” unbiased bits assuming the source has “some structure and enough entropy” [von Neumann, Elias, Blum]

True randomness is an expensive
non-replicable commodity.

Strong Pseudo Random Number Generator

Def 1 [Indistinguishability]

“No polynomial-time algorithm can distinguis between the output of a PRG on a random seed vs. a andom string”

= “as good as” a truly random string f practical purposes.

Def 2 [Next-bit Unpredict y]

“No polynomial-time a n can predict the (i+1)th bit of the output of a PRG gi e first i bits”

Def 3 [Incon ssibility]

“No polynomial-time algorithm can compress the output of the PRG into a shorter string”

ALL THREE DEFS EQUIVALENT!

PRG Def 1: Indistinguishability

Definition [Indistinguishability]:

A deterministic polynomial-time computable function G: $\{0,1\}^n \to \{0,1\}^m$ is a PRG which "passes all poly time statistical tests" if

(a) m > n and

(b) for every PPT algorithm D, there is a negligible function negl such that:

$$\left| \Pr[\, \mathbf{D(G(U_n))} = 1 \,] - \Pr[\, \mathbf{D(U_m)} = 1 \,] \right| = \mathrm{negl}(n)$$

Notation: U_n (resp. U_m) denotes the uniform distribution on n-bit (resp. m-bit) strings; m is shorthand for m(n).

PRG Def 1: Indistinguishability

Definition [Indistinguishability]:

A deterministic polynomial-time computable function G: $\{0,1\}^n \rightarrow \{0,1\}^m$ is a PRG which "passes all poly time statistical tests" if

(a) m > n and

(b) for every PPT algorithm D, there is a negligible function negl such that:

$$\left|\ \Pr[\ \mathbf{D(G(U_n))} = 1\ ] - \Pr[\ \mathbf{D(U_m)} = 1\ ]\ \right| = \mathrm{negl}(n)$$

We call D that takes a sequence and outputs 0 or 1 a *statistical test.*

PRG Def 1: Indistinguishability

Def: A deterministic function $G: \{0,1\}^n \rightarrow \{0,1\}^m$ is a strong PRG if $m > n$ and for every PPT algorithm **D**, there is a negligible function negl such that:

$$\left| \Pr[\, D(G(U_n)) = 1 \,] - \Pr[\, D(U_m) = 1 \,] \right| = \mathrm{negl}(n)$$

WORLD 1:

Pseudorandom World

$y \leftarrow G(U_n)$

WORLD 2:

Truly Random World

$y \leftarrow U_m$

PPT Distinguisher gets y but cannot tell which world she is in

Why is this a good definition

Good for all Applications:

As long as we can find truly random seeds, can replace true randomness by the output of PRG(seed) in ANY “computational” setting.

If it behaves differently,
can convert “application”=statistical test

Enables to overcome Shannon Theorem
|key|≥|message|

PRG $\Longrightarrow$ (How to Encrypt n+1 bits using an n-bit key)

$Gen(1^n)$: Generate a random n-bit key k.

$E(k, m)$ where m is an $\boldsymbol{(n+1)}$**-bit** message:

Expand k into a (n+1)-bit pseudorandom string $\mathrm{k}' = G(\mathrm{k})$

One-time pad with k': ciphertext is $k' \oplus m$

$D(k, c)$ outputs $\mathrm{G}(k) \oplus c$

For longer messages
Expand G(k) for more bits

(Length extension:
If there is a PRG that stretches
by one bit, there is one that stretches b

Correctness: $D(k, c)$ outputs $\mathrm{G}(k) \oplus c = \mathrm{G}(k) \oplus G(k) \oplus \mathrm{m} = \mathrm{m}$

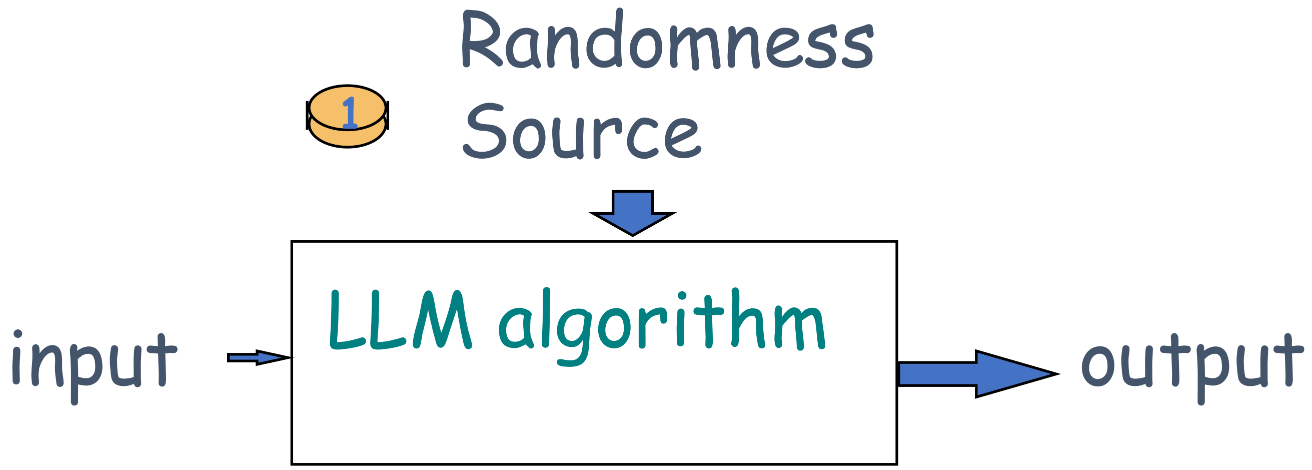
Security: EVE can't distinguish between $\mathrm{U}_{n+1} \oplus m_0$ *and* $\mathrm{U}_{n+1} \oplus m_1$ Since they are both uniform. If EVE can distinguish between
$\mathrm{G}(\mathrm{U}_n) \oplus m_0$ *and* $\mathrm{G(Un)} \oplus m_1$
then, EVE can distinguish $\mathrm{G}(\mathrm{U}_n)$ from $\mathrm{U}_{n+1,}$ contradiction.

Randomness is the foundation of cryptography

Cryptographic keys have to be unpredictable to the adversary

- Cryptographic algorithms use additional randomness (beyond the key)
- If the random bits are revealed (or are predictable) the entire structure can collapse

Randomness in Machine Learning



Crucial for training (SGD, ERM use sampling)
and for generating

Raises interesting questions

- Can bad use of randomness lead to bias outcomes
- Can use of special PSRG enforce ML-desirable properties

Pass All Statistical Test is a great definition

But: it’s hard to work with. How could you show that generator G passes ALL statistical tests?

Do PRG that pass all statistical tests even Exist?

PRG Def 2: (Next-bit) Unpredictability

Definition [Next-bit Unpredictability]:

A deterministic polynomial-time computable function $G: \{0,1\}^n \to \{0,1\}^m$ is a PRG if

(a) $m > n$ and

(b) for every PPT algorithm PRED and every $i \in [1..m]$, there is a negligible function negl such that:

$$\Pr[\mathrm{PRED}(y_1 y_2 \ldots y_{i-1}) = y_i] < \tfrac{1}{2} + \mathrm{negl}(n)$$

Over $y \leftarrow G(U_n)$:

Notation: y_i denotes the i-th bit of y.

$y_{1\ldots i}$ denotes the first i bits of y.

PRG Def 2: (Next-bit) Unpredictability

Definition [Next-bit Unpredictability]:

A deterministic polynomial-time computable function G: $\{0,1\}^n \rightarrow \{0,1\}^m$ is a PRG if

(a) m > n and

(b) for every PPT algorithm PRED and every $i \in [1..m]$, there is a negligible function negl such that:

$$\Pr[\mathrm{PRED}(y_1 y_2 \ldots y_{i-1}) = y_i] < \tfrac{1}{2} + \mathrm{negl}(n)$$

Over $y \leftarrow G(U_n)$:

Notation: Call PRED a "next-bit test" and if (b) holds, we say that G "passes all next bit tests "

Def 1 and Def 2 are Equivalent

Theorem: A PRG G passes all polynomial time statistical tests if and only if it passes all polynomial time next-bit tests

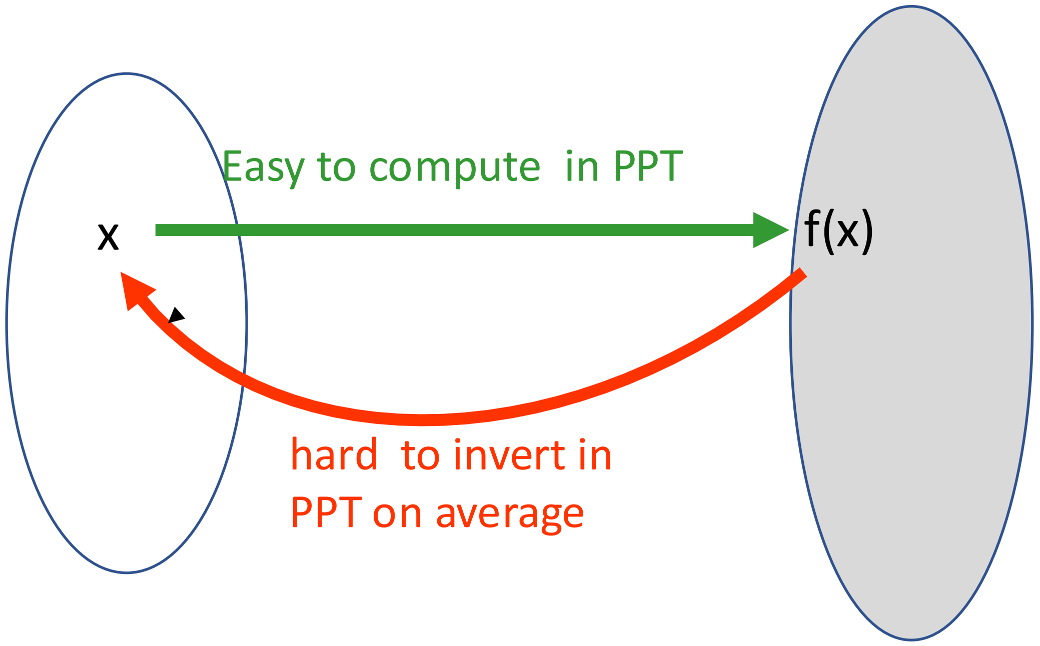
Proof: $\rightarrow$ $Easy,$

$\leftarrow$

$Harder.$ Show that any statistical test can be turned into a predictior for a next bit, via a Hybrid Argument

Next: Using **One Way Functions** will Build a PRG and prove that it passes all next-bit tests, hence CSPRG

One Way Functions



f: {0,1}* ⇒ {0,1}* is a one-way function(OWF) if:

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

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

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

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

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

Collection of One-way Functions

A function (collection) $\{f_n\}_{n\in\mathbb{N}}$ where f_n: $\{0,1\}^n \rightarrow \{0,1\}^{m(n)}$ is one-way if easy to evaluate f$_n$(x) for all x, and

hard to invert: ∀PPT Inv, ∀n sufficiently large

$$\Pr[Inv(y) = x' \; s.t. \, y = f_n(x')] \leq negl(n)$$

over $x \leftarrow \{0,1\}^n; y = f_n(x);$

- Can always find *an* inverse with unbounded time
- … but should be hard with probabilistic polynomial time

One-way Functions: Candidates

Subset sum:
f$(a_1, \ldots, a_n, x_1, \ldots, x_n)$ = $(a_1, \ldots, a_n, \sum_{i=1}^{n} x_i a_i$ mod $2^{n+1})$
where a_i random n-bit numbers, and x_i are random bits.

Rabin: $f_N(x) = x^2$ mod N
Is as hard to invert as factoring N

RSA. $f_N(x) = x^e$ mod N

DH$_{g,p}(x)$ = g^x mod p

One-way functions candidates are abundant in nature.

Many other candidates from number theory, coding theory, combinatorics later in class.

Attempt: Strong- PSRG from one-way **permutations**

Idea: Let f be one-way permutation.

- Choose random seed s in $\{0,1\}^n$
- Compute $f(s)$ $f^2(s)$ $f^3(s)$ … $f^{\,m}(s)$
- Output in reverse order

- Why is this a good idea, Intuitively, ?
  - Somewhat Unpredictable: From $f^i(s)$ can't compute $f^{i-1}(s)$
- Why not good enough ?
  - Even though you cannot predict $f^{i-1}(s)$ some bits of it may be predictable.

Exercise: There are one-way functions for which it is easy to compute the first half of the bits of the inverse.

- Still, may be can be salvaged: there must be some bits of $f^{-1}(s)$ that are hard to compute for a random s.

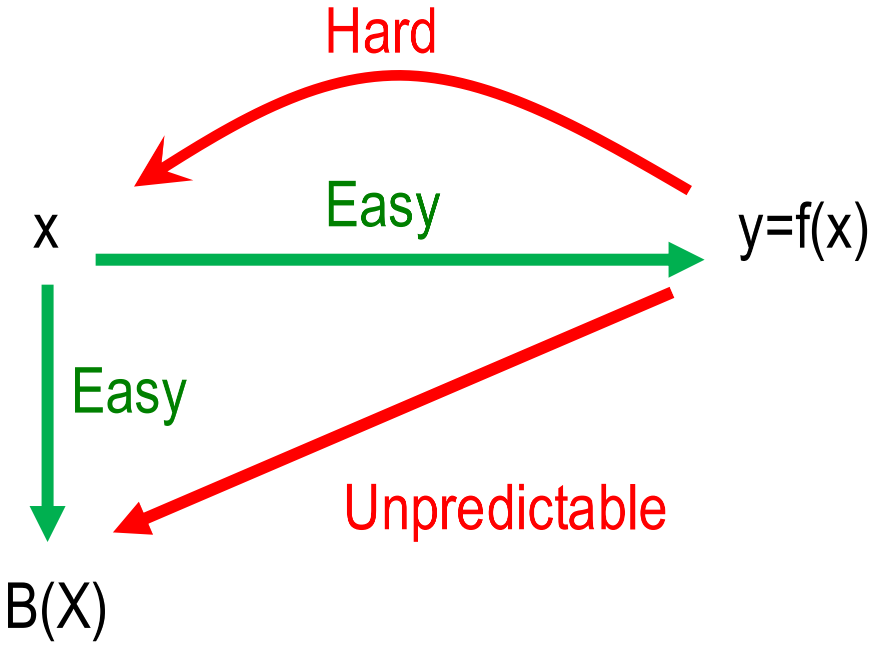
Hard Core Predicates for OWF

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)



Discussion on the Definition

1. Some functions can have information-theoretically hard to guess predicates (e.g., compressing functions)

2. We'll be interested in settings where x is uniquely determined given $\mathrm{f}(x)(i.e\ permutations)$ yet $\mathrm{B}(x)$is hard to predict given $\mathrm{F}(x)$.

3. Ex: LSB(x) is hard-core for RSA(x) but not for DH(x)

4. Is there a universal hard-core predicate?

Constructing PSRG

Theorem: If there exist one-way-**permutations** f with hard core bit B, then there exist strong PRG

$G:\{0,1\}^n \text{->} \{0,1\}^{P(n)}$ for any polynomial P.

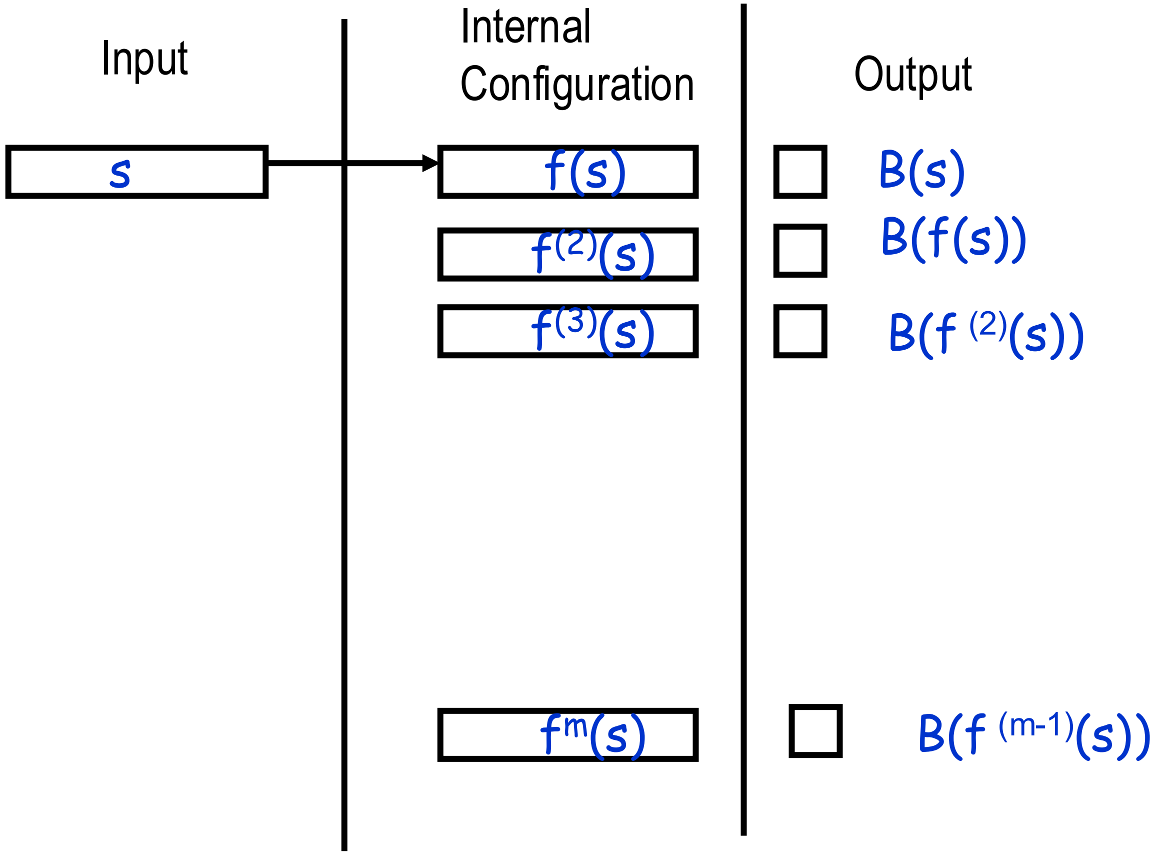
Proof: Let P be a polynomial function, set m=P(n)

On input seed s from U_n,

G(s): (1) compute f(s) f(f(s)) … $f(f^{m-1}(s))$

(2) output B(s) B(f(s)) … $B(f^{m-1}(s))$

Picture Better than 1000 words



Input

Internal Configuration

Output

s

$f(s)$

$f^{(2)}(s)$

$f^{(3)}(s)$

$f^{m}(s)$

□ $B(s)$

□ $B(f(s))$

□ $B(f^{(2)}(s))$

□ $B(f^{(m-1)}(s))$

Proof : Show outputs of G pass all next-bit tests.

Suppose, for contradiction, ∃bit location j<m and next bit predictor **P** s.t. $\mathbf{Pr}_{y \leftarrow G(Un)}[\mathbf{P}(y_1y_2\ldots y_{j-1}) = y_i] > ½ + 1/P(n)$ **for poly P**

Then show a predictor P’ for Hard-Core B of f:

```
P’(f(x)):
    1. compute    $f(f(x))$ …                         $f(f^{j-1}(x))$
    2. compute    $\underset{\overset{\shortparallel}{y_{j-1}}}{B(f(x))}$ …     $\underset{\overset{\shortparallel}{y_1}}{B(f^{j-1}(x))}$
  3. Output $\mathbf{P}(y_1 \;\ldots\; y_{j-1})$
```

EUREKA: the next bit $y_{l.}$ in the sequence should be B(x)

And we assumed that P predicts next bit y_i with prob. ½+ 1/P(n)

Proof : Show outputs of G pass all next-bit tests.

Suppose, for contradiction, ∃bit location j<m and next bit predictor **P** s.t. $\Pr_{y \leftarrow G(U_n)}[P(y_1 y_2 \ldots y_{j-1}) = y_i] > ½ + 1/P(n)$ for poly P

Then show a predictor P' for Hard-Core B of f:

```
P'(f(x)):
    1. compute    $f(f(x))$ …    $f(f^{j-1}(x))$
    2. compute    $\underset{\substack{\shortparallel \\ y_{j-1}}}{B(f(x))}$ …    $\underset{\substack{\shortparallel \\ y_1}}{B(f^{j-1}(x))}$
    3. Output $P(y_1 \ \ldots \ y_{j-1})$
```

Claim: $\Pr[P'(f(x)=B(x)]=\text{Prob}[\ P(b_1 \ \ldots\ b_{j-1})=b_j\ ]>½ + 1/P(n)$ contradiction

Essential to Proof: f is a permutation ⇒$y_1 \ \ldots \ y_{j-1}$ is the same distribution as P is expecting and will perform well on.

We just went through
A sequence of reductions

- Since B is hard-core for one-way function f, P’ cannot exist

⇒ Next bit test P cannot exist

⇒ G passes all next bit tests

⇒G passes all polynomial time statistical tests

⇒G outputs are computationally indistinguishable from random

One Way **Functions vs.** One Way **Permutations**

Theorem: If $\exists$**one-way-functions** ,

then $\exists$CS-PSRG

$G:\{0,1\}^n \to \{0,1\}^{P(n)}$ for any polynomial P.

Proof: Much Harder

See web site [HILL]

Does every one-way function have a hardcore bit?

(Hard) Exercise: There are functions that are one-way, yet *every* bit is somewhat easy to predict (say, with probability $\frac{1}{2} + 1/n$).

So, we will generalize the notion of a hardcore "bit".

Goldreich-Levin (GL) Theorem

Every OWF Has an Associated Hard-Core Bit

Let $\{B_r: \{0,1\}^n \rightarrow \{0,1\}\}$ where

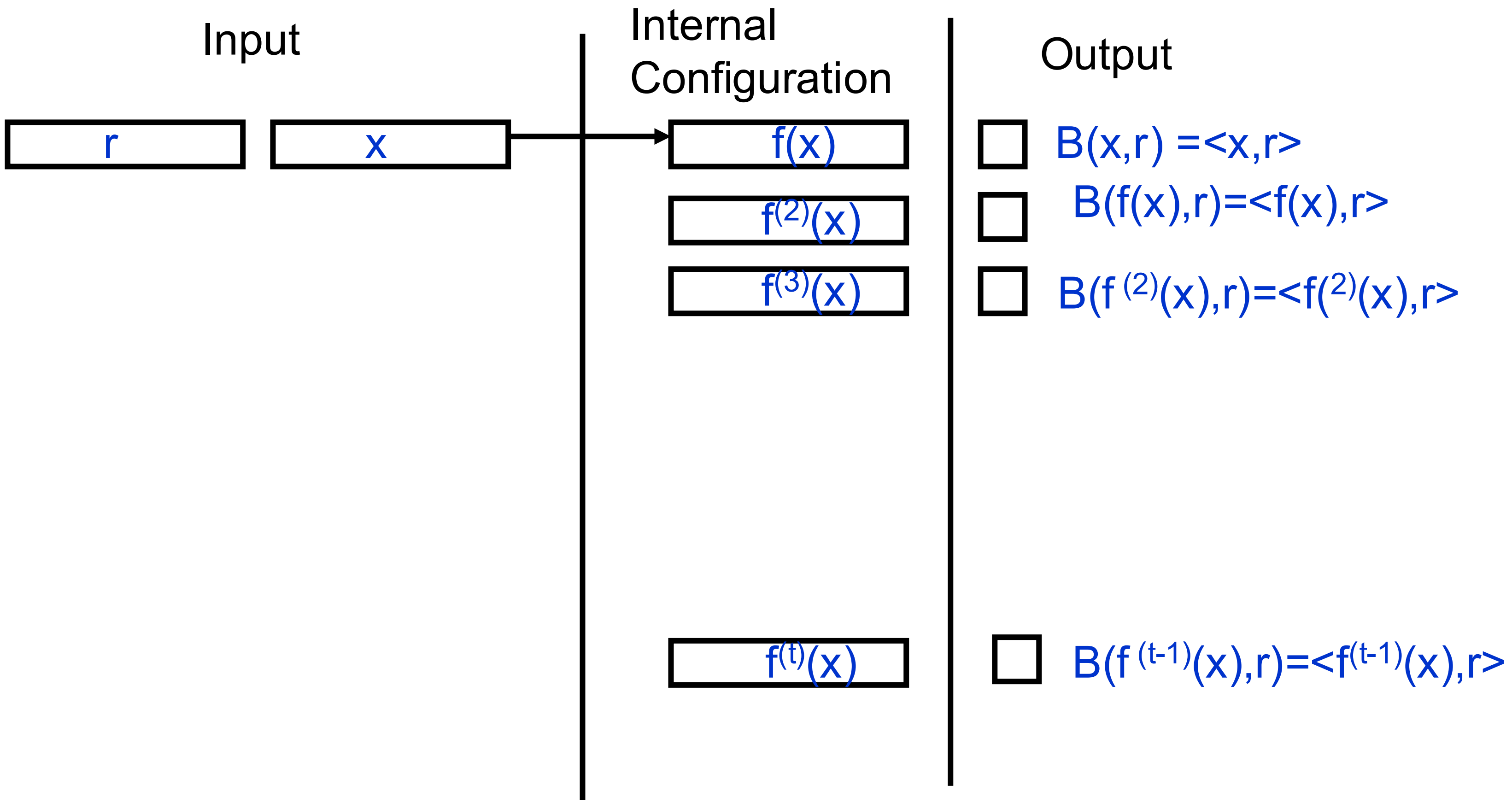
$$B_r(x) = \langle r, x \rangle = \sum_{i=1}^{n} r_i x_i \bmod 2$$

be a collection of predicates (one for each r). Then, a ***random*** B_r is hardcore for ***every*** one-way function F. That is, for every one-way function F, every PPT A, there is a negligible function μ s.t.

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

Alternative Interpretation 1: 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.

Example of strong PRG: based on Goldreich-Levin Hard Core Bit

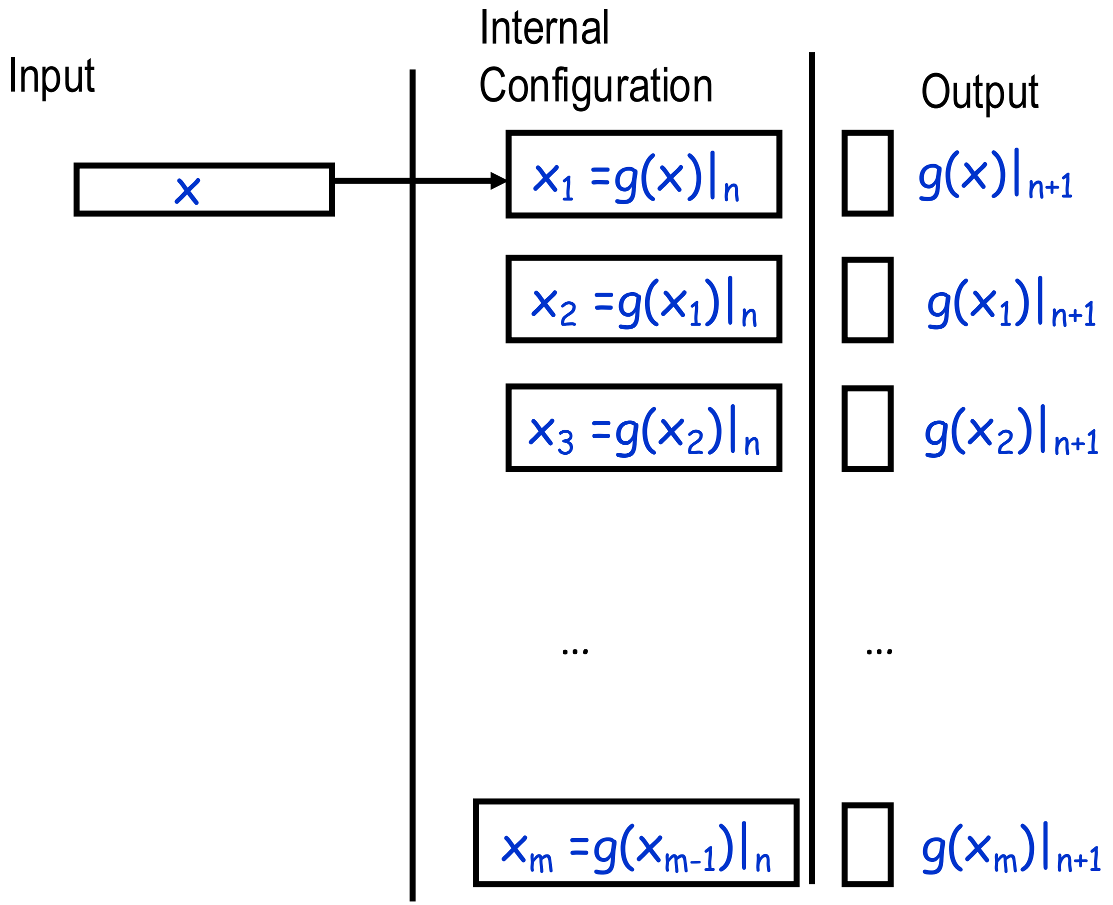


- Use the same r and even can make r public

Extenders

- Theorem: If there exists a CS-PSRG that extends n bits to n+1 bits, then there exists a CS-PSRG that extends n bits to any polynomial length.

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



Building Block:

Single Bit Expanding

CS-PSRG

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

- Excersize: what are the hybrids you would define to prove that this works?

What did we do today

1. Defined one-way functions (OWF).

2. Defined Hardcore bits (HCB).

3. Show that one-way *permutations* (OWP) ⇒ PRG

   (in fact, one-way functions ⇒ PRG, but that’s a much harder theorem)

4. Goldreich-Levin Theorem: every OWF has a HCB.

Extra slides

Notation, Missing Proofs