Lecture 13: Pseudorandomness: Hardness vs Randomness

Randomized algo : uses n bits

How do we supply this? Can we derandomize the algorithm?

Cryptography: Randomness crucial

- One-time pad
- Semantic security

$Enc_K(m) = m \oplus K$ → K secret key

$|K| = |m|$ lot of randomness

Can you get by with fewer random bits?

Shannon: No → Bypass this lower bound?

Computational vs Information-theoretic security

Information leaks in a way that a computationally bounded adversary can't detect it.

Key concept: "Pseudorandomness"

"A random looking string"

Computational Indistinguishability

D_1 & D_2 on $\{0,1\}^n$

D_1 & D_2 are statistically close:

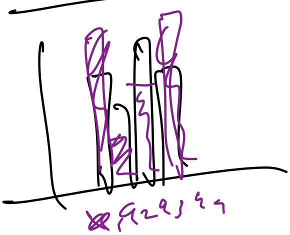
D_1, D_2 are ε-close if $\forall$ tests $T : \{0,1\}^n \to \{0,1\}$

$$\left| \Pr_{x \in D_1}[T(x)=1] - \Pr_{x \in D_2}[T(x)=1] \right| \le \varepsilon$$

$\Delta_{TV}(D_1, D_2) :=$ max over all T of above defn

Exercise: $\Delta_{TV}(D_1, D_2) = \frac{1}{2} \| D_1 - D_2 \|_1$

$$= \frac{1}{2} \sum_{x \in \{0,1\}^n} | D_1(x) - D_2(x) |$$



$$\Delta_{TV}(D_1, D_2) \in [0, 1]$$

Computational Indistinguishability:

Restrict test T to complexity $C(n)$
- $\to$ time complexity
- $\to$ circuit size

$D_1 \underset{(C,\varepsilon)}{\sim} D_2$ if $\forall$ Tests T of complexity $C(n)$

$$\left| \Pr_{x \in D_1}[T(x)=1] - \Pr_{x \in D_2}[T(x)=1] \right| \le \varepsilon$$

~~Defn~~ Dist D : said to be (C, ε) pseudorandom

if $D \underset{(C,\varepsilon)}{\sim}$ Uniform Dist

Claim ~~If~~ (randomized) algorithm A with "complexity" $C(n)$ using n random bits is fed a ~~pseudorandom~~ pseudorandom string from a (C, ε)-pseudorandom dist., then its acc. prob deviates by at most ε.

$$\left| \Pr_{x \in \mathrm{Unif}} [A(x)=1] - \Pr_{x \in D} [A(x)=1] \right| \leq \varepsilon$$

How to produce a pseudorandom sample.

PSEUDORANDOM GENERATOR :

$$G : \{0,1\}^s \longrightarrow \{0,1\}^n$$

$s = s(n)$ = seed length

$s \ll n$

is an (C, ε)-PRG if

$G(\mathrm{Unif}(\{0,1\}^s))$ is (C, ε)-pseudorandom

$\hookrightarrow \underset{(C,\varepsilon)}{\sim} \mathrm{Unif}(\{0,1\}^n)$

G stretches a random string of ~~size~~ length s into a "random looking" string of length n

Even for $s = n-1$ this is impossible statistically.

Computationally seek $s \ll n$, (like $n^{0.01}$, or even $O(\log n)$)

Food for Thought, If G is computable in $2^{O(s)}$ time and it is

$G: \{0,1\}^{s(n)} \to \{0,1\}^n$

$(C(n), 1/10)$ - pseudorandom

then a randomized algo A with complexity $C(n)$ & success prob $2/3$, can be derandomized into a det. algo with run time $\boxed{2^{O(s(n))} \cdot C(n)}$.

Run A on $G(a)$ ($\forall a \in \{0,1\}^s$) as its "random" string, take majority.

So if $s(n) = O(\log n)$
then randomized polytime algo

$\xrightarrow[\text{into}]{\text{converted}}$ det polytime algo

---

Hardness vs. Randomness :

$n = s+1$

$G : \{0,1\}^s \longrightarrow \{0,1\}^{s+1}$

b = random bit

$G(x) = \boxed{(x, \cancel{g(x)})}$ $\quad g : \{0,1\}^s \to \{0,1\}$

g – very hard to compute

$\forall$ CCT complexity T, $\Pr_{x \in \{0,1\}^s}[T(x) = g(x)] \leq \frac{1}{2} + \varepsilon$

Intuitively, output of G is (T, ε) – pseudorandom

1. First s bits are random (x is random)
2. $(x, g(x)) \longrightarrow (x, b)$

Link between hardness & pseudorandomness

Great, but there's a small problem.

$$G(x) = (x, g(x))$$

hard, so now can G compute it.

Two possible avenues:

1. Allow G more time than the tests it "fools". (ok for derandomization. Not ok for cryptographic uses)

2. Twist the PRG construction to create some asymmetry that gives G an edge compared to tests it has to fool.

Few words about 2

$$G(x) = (x, h(x))$$

$h: \{0,1\}^s \to \{0,1\}$ hard to predict ~~a~~ from

$h(x)$ looks random

given $\pi(x)$ to a bounded adversary

$G(x) = (\cancel{x}, h(x))$ easy to compute

$G(x) = (\pi(x), h(x))$

ONE-WAY PERMUTATION

- permutation on $\{0,1\}^n$
- Easy to compute $x \mapsto \pi(x)$
- Hard to invert

$x \xrightarrow{\text{easy}} \pi(x)$

$x \xleftarrow{\text{hard}} \pi(x)$

h: hard core predicate for permutation π

$x \longmapsto h(x)$ easy

$\pi(x) \longmapsto h(x)$ hard

$\pi(x) \to x \to h(x)$

Examples of candidate $\left\{ \begin{array}{l} \text{one-way perm} \\ \text{\& hard-core predicate} \end{array} \right.$

- $\pi(x) = x^e \bmod N$ (RSA exponentiation)

  $h(x)$: least-significant-bit(x)

  $(x \in \mathbb{Z}_N)$

- $\pi(x) = g^x \bmod p$ (p prime)

  (inverting = discrete log)

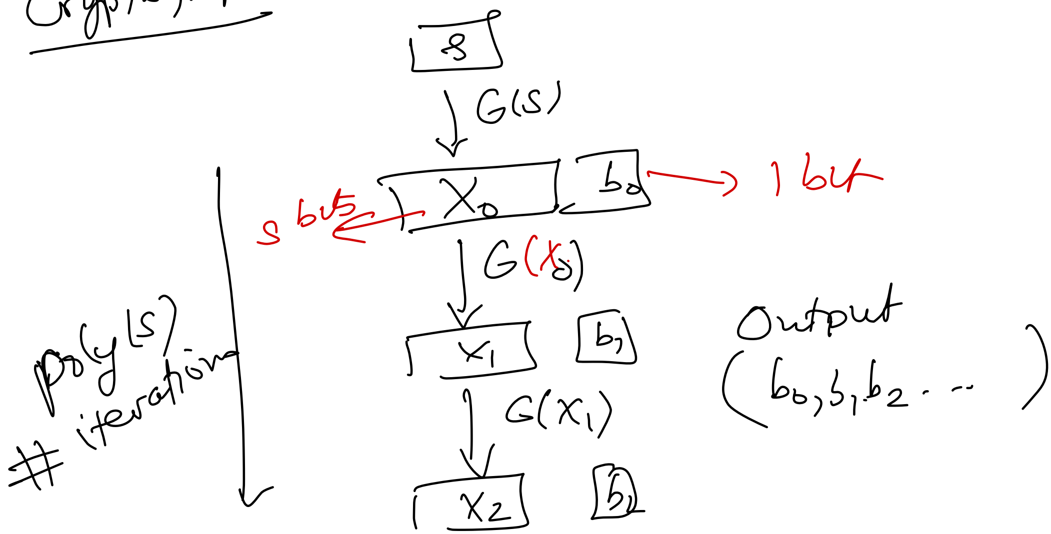
  $h(x)$: most-significant-bit(x)

  $x \in \{0, \ldots, p-2\}$ $\quad$ $\text{half}_{p-1}(x) = \begin{cases} 1 & \text{if } x \in [0, \frac{p-1}{2}] \\ 0 & x > \frac{p-1}{2} \end{cases}$

Comment: $\text{lsb}(x)$ is not hardcore for $(x \mapsto g^x \bmod p)$

Stretching by more bits

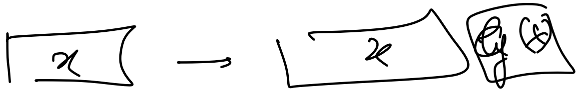
Cryptographic setting



$s \longmapsto s^{100}$ bits

One-way permutation $\Longrightarrow$ PRG with seed length $n^{0.01}$

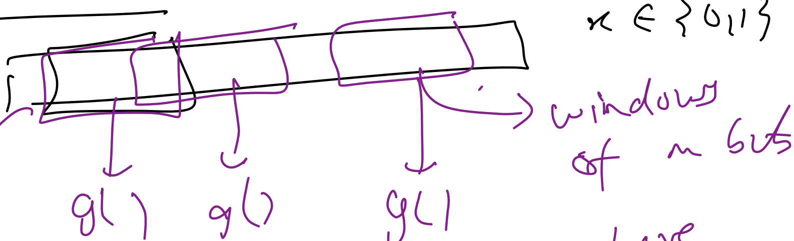
Approach (1) : G takes more time than tests it fools



$g : \{0,1\}^m \to \{0,1\}$

More bits?

$x \in \{0,1\}^s$



Choose windows so that they have small intersection : . Design
. Error-correcting code