CS 229r - LECTURE 23 4/25/2017

Codes in Complexity:

1. PSEUDORANDOM GENERATION
2. HARDNESS AMPLIFICATION

P.R.G (Defn): $G: \{0,1\}^n \to \{0,1\}^m$ is a ϵ-PRG if

1. $m > n$
2. G efficient to compute
3. ~~Given~~ for every polynomial time $0/1$ algorithm D (for distinguisher)

$$\left| \Pr_{x \sim U_m}\left[D(x)=1 \right] - \Pr_{x\sim U_n}\left[D(G(x))=1 \right] \right| \le \epsilon.$$

One-way Permutation (function)

$f: \{0,1\}^n \to \{0,1\}^n$ is a ϵ One-way Permutation (function)

if
0. f is 1-1 (omitted for O.W.F.)
1. f is polytime computable.
2. f is hard to invert on average. i.e.,

$\forall$ polytime A $\Pr_x\left[f(A(f(x)))=f(x) \right] \le \epsilon.$

[Blum-Micali], [Yao], [Goldreich-Levin] ...

Theorem: O.W.P $\Longrightarrow$ PRG [More strong theorem due to [Håstad, Impagliazzo, Levin, Luby]]
OWF $\Leftrightarrow$ PRG.

Prop.: P=NP $\Rightarrow$ OWP, OWF, PRG doesn't exist.

Proof of Theorem: Claim: Suppose f is a o.w.p.

~~G is PRG ⇒~~

Let $C : \{0,1\}^n \to \{0,1\}^N$ be a $(\frac{1}{2}-\epsilon)$-error polytime list decodable code.

Then $G : \{0,1\}^n \times [N] \to \{0,1\}^{n+1} \times [N]$

given by $G(x,i) = \big(f(x), C(x)_i, i\big)$

is a PRG.

Proof of Claim: ~~let D be~~ Suppose G is not PRG.

then $\exists D$ polytime, s.t.

$$\Pr_{x,i}\left[D(f(x), i, C(x)_i) = 1\right] - \Pr_{x,i,b}\left[D(f(x), i, b) = 1\right] > \epsilon$$

(i) Distinguisher $\Rightarrow$ Predictor

$$P(f(x), i) = 1 \quad \text{if} \quad D(f(x), i, 1) = 1 \;\&\; D(f(x), i, 0) = 0$$
$$= 0 \quad \text{if} \quad \quad '' = 0 \quad \quad '' = 1$$
$$= \text{random} \quad \text{o.w.}$$

<u>Sub Claim</u>: $\Pr_{x,i}\left[P(f(x), i) = C(x)_i \right] \geq \frac{1}{2} + \frac{\epsilon}{2}$

<u>Proof</u>: Omitted / Trivial / ~~Tea~~ Can't do it in real-time.

(ii) Predictor + List-decoder $\Rightarrow$ Guesser.

$A(f(x))$: ① for $i = 1 \ldots N$ do $y_i = P(f(x), i)$

② List decode $y = y_1 \ldots y_N$ to get $x^{(1)} \ldots x^{(\ell)} \in \{0,1\}^n$

if $f(x^{(j)}) = f(x)$ output $x^{(j)}$.

<u>Sub Claims</u>: ① $\Pr_x\left[\Delta(y, C(x)) \leq \frac{1}{2} - \frac{\epsilon}{4} \right] \geq \frac{\epsilon}{4}$

② if '' then list-decoder's output includes x & so inverts successfully.

$\boxtimes$

Hardness Amplification

General goal: Suppose some one proves $NP \neq P$.
Are we good to launch cryptography from this point on?

- Say they prove $\exists\ f: \{0,1\}^n \to \{0,1\}^m$ s.t.
$\forall x$ $f(x)$ is easy to compute. But for every alg.
A $\exists x$ s.t. $A(f(x)) \cancel{\neq} \cancel{f} \notin f^{-1}(f(x))$

- Can we use f for encrypting passwords?

Big Issue: $\exists X$ s.t. X not recoverable from $f(x)$ is security for one person, not security for all!

Needed: f s.t. f easy to compute & very hard to invert

$$\Pr_x \left[A(f(x)) \in f^{-1}(f(x)) \right] \leq \epsilon = \mathrm{negl}(n).$$

HARDNESS AMPLIFICATION

Given $f \in C_{BIG}$ s.t. $f \notin C_{Small}$ find F

s.t. $F \in C_{BIG}$ s.t. $\forall$ ~~any~~ function G close to F

$$G \notin C_{Small}$$

Would love it with $\left.\begin{array}{l} C_{BIG} = NP \\ C_{Small} = P \end{array}\right\}$ But still open

---

[Impagliazzo - Wigderson / S. Trevisan - Vadhan] : $C_{BIG} = \text{Exptime}$
$C_{Small} \cong P$

Key Ingredient :

$E : \{0,1\}^K \to \{0,1\}^N$ that is polytime encodable.

& polylog time locally list-decodable from $(\frac{1}{2} - \epsilon)$-errors

---

Proof of IW/STV Theorem

$f: \{0,1\}^n \to \{0,1\}$ be in Exptime, but not in P.

let $K = 2^n$ so $f \in \{0,1\}^K$ (truth table)

Let $E(f) = F \in \{0,1\}^N$ be truth table of

$F: \{0,1\}^m \to \{0,1\}$.

(1) $f \in$ Exptime $\Rightarrow$ $F \in$ Exptime;

$f(x)$ computable in time 2^{cn} $\Rightarrow$ $\langle f(x) \rangle_{x \in \{0,1\}^n}$

Computable in $2^{(c+1)n}$.

$\Rightarrow \langle F(y) \rangle_{y \in \{0,1\}^m}$ computable in $2^{(c+1)n} \cdot \text{poly}(2^n)$

$= 2^{O(n)}$.

(2) $F \approx G \in P \Rightarrow f \in P$?

Rough Idea: G gives oracle access to noisy F.

Local-list decoder corrects errors in time $\text{poly} \log K = \text{poly}(n)$

& gives oracle access to $\langle f(x) \rangle_x$

$\Rightarrow$ computes f on every input whp. $\boxtimes$