Computational Complexity December 3, 2018

Lecture 15 (Notes)

Lecturer: Ola Svensson *Scribes: Ola Svensson*

Disclaimer: These notes were written for the lecturer only and may contain inconsistent notation, typos, and they do not cite relevant works. They also contain extracts from the two main inspirations of this course:

1. The book *Computational Complexity: A Modern Approach* by Sanjeev Arora and Boaz Barak;
2. The course *http://theory.stanford.edu/ trevisan/cs254-14/index.html* by Luca Trevisan.

Recall last lecture:

- Introduction to crypto.
- Perfect secrecy requires large keys, instead computational security.
- One-way functions and pseudorandom generators (PRGs).
- PRGs imply computational security with "small" keys using the idea of one-time padding with the pseudorandom string.
- Other cool implications of PRGs are derandomization, commitment schemes, lower bounds on machine learning.

Today:

- We are going to prove that one-way permutations imply PRGs. We remark that to simplify the proof our assumption is a little stronger than in last lecture's theorem statement: instead of assuming the existence of one-way functions we assume the existence of one-way permutations $f : N \to N$, i.e., a one-way function that is a permutation.

1 Simplifying security assumption on PRGs: unpredictability implies pseudorandomness

Recall the definition of *secure* PRGs:

Definition 1 *Let $G : \{0,1\}^* \to \{0,1\}^*$ be a polynomial-time function. We say that G is a* secure pseudorandom generator of stretch $\ell(n) > n$ *if $|G(x)| = \ell(|x|)$ for every $x \in \{0,1\}^*$ and for every probabilistic polynomial-time A, there exists a negligible function ϵ such that*

$$\left|\Pr[A(G(U_n)) = 1] - \Pr[A(U_{\ell(n)}) = 1]\right| \leq \epsilon(n)$$

for every $n \in \mathbb{N}$.

Yao proved that the above seemingly stronger notion of security is equivalent to that of predicting "the next bit in the sequence".

Theorem 2 *Let $G : \{0,1\}^* \to \{0,1\}^*$ be a polynomial-time function such that $|G(x)| = \ell(|x|)$ for every $x \in \{0,1\}^*$. For every probabilistic polynomial-time algorithm A, there exists a probabilistic polynomial-time algorithm B such that for every $n \in \mathbb{N}$ and $\epsilon > 0$, if $\Pr[A(G(U_n)) = 1] - \Pr[A(U_{\ell(n)}) = 1] \geq \epsilon$, then*

$$\Pr_{x \in \{0,1\}^n, y = G(x), i \in [\ell(n)]}[B(1^n, y_1, \ldots, y_{i-1}) = y_i] \geq 1/2 + \epsilon/\ell(n)\,.$$

Before giving the proof, we note that this shows that unpredictability implies pseudorandomness. Indeed, the theorem states that if G is not a PRGs (there is a distinguisher A[1]), then there is a predictor B. In other words, the non-existence of a predictor B implies that G is a PRG. We now proceed with the proof.

Proof

- Let A be a probabilistic polynomial-time algorithm that is more likely to output 1 on inputs from the distribution $G(U_n)$ than on inputs from $U_{\ell(n)}$ as in the theorem statement.
- Our predictor algorithm B now works as follows: On input $1^n, i \in [\ell(n)]$ and $y_1, \ldots, y_{i-1}$,
  - Algorithm B chooses $z_i, z_{i+1}, \ldots, z_{\ell(n)}$ independently at random.
  - It then computes $a = A(y_1, \ldots, y_{i-1}, z_i, \ldots, z_{\ell(n)})$.
  - If $a = 1$ then B outputs z_i (its guess was correct), otherwise it outputs $1 - z_i$.
- To analyze B's performance, we use the *hybrid argument*. Define distributions $D_0, \ldots, D_{\ell(n)}$ over $\{0,1\}^{\ell(n)}$, where the distribution D_i is obtained as follows:
  - Choose $x_i \in \{0,1\}^n$ at random and let $y = G(x)$.
  - Output $y_1, \ldots, y_i, z_{i+1}, \ldots, z_{\ell(n)}$, where $z_{i+1}, \ldots, z_{\ell(n)}$ are chosen independently at random in $\{0,1\}$.

  Note that the distributions interpolate between $D_0 = U_{\ell(n)}$ and $D_{\ell(n)} = G(U_n)$.
- Define $p_i = \Pr[A(D_i) = 1]$ and so
$$p_{\ell(n)} - p_0 = \Pr[A(G(U_n)) = 1] - \Pr[A(U_{\ell(n)}) = 1] \geq \epsilon\,.$$
- By telescoping $p_{\ell(n)} - p_0 = (p_{\ell(n)} - p_{\ell(n)-1}) + (p_{\ell(n)-1} + p_{\ell(n)-2}) + \ldots + (p_1 - p_0)$ and so
$$\mathbb{E}_{i \in [\ell(n)]}[p_i - p_{i-1}] \geq \epsilon/\ell(n)\,.$$
  We will analyze B and prove the theorem by showing
$$\Pr_{x \in \{0,1\}^n, y = G(x)}[B(1^n, y_1, \ldots, y_{i-1}) = y_i] \geq 1/2 + (p_i - p_{i-1})\,.$$
- B predicts y_i correctly if either $a = 1$ and $y_i = z_i$ or $a = 0$ and $y_i = 1 - z_i$. The probability that this happens is
$$\frac{1}{2}\Pr[a = 1 | z_i = y_i] + \frac{1}{2}(1 - \Pr[a = 1 | z_i = 1 - y_i])\,. \tag{1}$$
  One can verify that condition on $z_i = y_i$, B invokes A with the distribution D_i, meaning that $\Pr[a = 1 | z_i = y_i] = p_i$. On the other hand, if we don't condition on z_i, then the distribution B invokes A is equal to D_{i-1}. Hence,
$$p_{i-1} = \Pr[a = 1] = \frac{1}{2}\Pr[a = 1 | z_i = y_i] + \frac{1}{2}\Pr[a = 1 | z_i = 1 = y_i] = \frac{1}{2}p_i + \frac{1}{2}\Pr[a = 1 | z_i = 1 = y_i]\,.$$
  We thus have
$$(1) = \frac{1}{2}p_i + \frac{1}{2} - \left(p_{i-1} - \frac{1}{2}p_i\right) = \frac{1}{2} + (p_i - p_{i-1})\,,$$
  as required.

■

[1] We remark that the absence of absolute value can be dealt with by either considering A or $1 - A$.

2 The Goldreich-Levin Theorem

Recall the overall goal of this lecture: from a one-way permutation we wish to construct a PRG with arbitrary long stretch. It turns out that the crucial step is to obtain a stretch of one more bit!

Theorem 3 *Suppose that $f : \{0,1\}^* \to \{0,1\}^*$ is a one-way permutation. Then for every probabilistic polynomial-time algorithm A there is a negligible function $\epsilon : \mathbb{N} \to [0,1]$ such that*

$$\Pr_{x,r\in\{0,1\}^n}[A(f(x),r) = x \odot r] \leq 1/2 + \epsilon(n)$$

were $x \odot r$ is defined to be $\sum_{i=1}^{n} x_i r_i \mod 2$.

The above theorem implies that $G(x,r) = f(x), r, x \odot r$ is a PRG that extendes the output with one-bit: because f is a permutation, the first $2n$ bits are completely random and the bit $x \odot r$ is hard to predict by the above theorem.

Proof

- Suppose there is some probabilistic polynomial-time algorithm A that violates the statement. Specifically assume that for some n:

$$\Pr_{x,r\in\{0,1\}^n}[A(f(x),r) = x \odot r] \geq 1/2 + \epsilon\,. \tag{2}$$

- We will then use A to design an algorithm B that runs in time $O(n^2/\epsilon^2$ and inverts the one-way permutation f on inputs of length n with probability at least $\Omega(\epsilon)$.

- This means that if A's success probability is more than $1/2+n^{-c}$ for some constant c and infinitely many n's, then B runs in polynomial-time and inverts the one-way permutation with probability at least $\Omega(n^{-c})$ for infinitely many n's. (This would contradict that f is a one-way function.)

- Assuming (2), a simple averaging argument implies that for at least $\epsilon/2$ fraction of the x's satisfy:

$$\Pr_{r\in\{0,1\}^n}[A(f(x),r) = x \odot r] \geq 1/2 + \epsilon/2\,.$$

  We call such x's *good* and show an algorithm B that with high probability inverts $f(x)$ for every good x.

- The setting is thus as follows. We are given a black box A that for a random r computes $x \odot r$ correctly with probability at least $1/2 + \epsilon/2$. Our goal is to recover x. We will do this in three steps every time weakening the correctness probability of A (starting with 1, then 0.9, and finally $1/2 + \epsilon/2$).

Warm-up (recovery for success probability 1): If $\Pr_r[A(f(x),r) = x \odot r] = 1$, then run $A(f(x), e^1)$, $\ldots, A(f(x), e^n)$, where e^i is the string whose ith coordinate is equal to one and all the other coordinates are zero. Clearly, $x \odot e^i$ equals the ith bit and so we can recover x from the n calls.

Recovery for success probability 0.9:

- If $\Pr_r[A(f(x),r) = x \odot r] \geq 0.9$, we cannot trust that $A(f(x), e^i) = x \odot e^i$ since it may be that $e^1, \ldots, e^n$ are among the $2^n/10$ strings r on which A answers incorrectly.

- Still, similar to the local decoding of the Walsh-Hadamard code, there is a simple way to bypass this problem: if we choose $r \in \{0,1\}^n$ uniformly at random then the string $x \odot r$ is also uniformly distributed. Hence

$$\Pr_r[A(f(x),r) \neq x \odot r \text{ or } A(f(x), r \oplus e^i) \neq x \odot \left(x \oplus e^i\right)] \leq 0.2\,. \tag{3}$$

  But $x \odot \left(r \oplus e^i\right) = (x \odot r) \oplus (x \odot e^i)$ and so if we choose r at random then $A(f(x,r) \oplus A(f(x), r \oplus e^i)$ equals the ith bit of x with probability at least 0.8.

- To obtain every bit of x we amplify this probability to $1-1/(10n)$ by taking majorities. Specifically, we use the following Algorithm B:
  - Choose $r^1, \ldots, r^m$ independently at random from $\{0,1\}^n$ (we'll specify m shortly).
  - For every $i \in [n]$:
    - Guess that x_i is the majority value among $\left(A(f(x), r^j) \oplus A(f(x), r^j \oplus e^i)\right)_{j \in [m]}$.

- We claim that if $m = 200n$, then for every $i \in [n]$, the majority value will be correct with probability at least $1 - 1/(10n)$ and hence B will recover every bit of x with probability at least 0.9.

- Define Z_j be the random indicator variable that takes value 1 if $A(f(x), r^j) = x \odot r^j$ and $A(f(x), r^j \oplus e^i) = x \odot (r^j \oplus e^i)$; otherwise $Z_j = 0$.

- The variables $Z_1, \ldots, Z_m$ are independent and $\mathbb{E}[Z_j] \geq 0.8$ by (3).

- To prove that B outputs the correct guess of i with probability at least $1 - 1/(10n)$ it is sufficient to show

$$\Pr[Z \leq m/2] \leq \frac{1}{10n}\,, \text{ where } Z = \textstyle\sum_j Z_j.$$

  Since $\mathbb{E}[\sum_j Z_j] \geq 0.8m$ it is sufficient to bound $\Pr[|Z - \mathbb{E}[Z]| \geq 0.3m$.

- For this, we use Chebychev's inequality:

$$\Pr\left[|Z - \mathbb{E}[Z]| \geq k\sqrt{\mathrm{Var}[Z]}\right] \leq \frac{1}{k^2}\,.$$

- In our case $\mathrm{Var}[Z] = \sum_j \mathrm{Var}[Z_j]$ by independence and $\mathrm{Var}[Z_j] \leq 1$ since $Z_j \in \{0,1\}$. Hence $\mathrm{Var}[Z] \leq m$ and

$$\Pr\left[|Z - \mathbb{E}[Z]| \geq 0.3\sqrt{m}\sqrt{\mathrm{Var}[Z]}\right] \leq \frac{1}{0.09m} \leq 1/(10n)\,.$$

Recovery for success probability $1/2 + \epsilon/2$:

- The previous strategy can be seen to fail even with a success probability 3/4 so how can we deal with a success probability as low as $1/2 + \epsilon/2$?

- A key insight is that the previous analysis would have worked if the strings $r^1, \ldots, r^m$ would have only been pairwise independent and not completely independent. Indeed, the only place where we used independence was in $\mathrm{Var}[Z] = \sum_j \mathrm{Var}[Z_j]$ and this condition also holds for pairwise independent variables (exercise).

- We now show how to pick $r^1 \ldots r^m$ in a pairwise independent fashion in such a way that we "know" each $x \odot r^i$ already. This seems weird since we don't know x but we will achieve this by exhaustive guessing.

- Set k such that $m \leq 2^k - 1$ and do as follows:
  - Choose k strings $s^1 \ldots, s^k$ independently at random from $\{0,1\}^n$.
  - For every $j \in [m]$, we associate a unique nonempty set $T_j \subseteq [k]$ with j in some canonical fashion and define $r^j = \sum_{t \in T_j} s^t \mod 2$. That is, r^j is the bitwise XOR of all the strings among $s^1, \ldots, s^k$ that belong to the jth set.
- The strings $r^1, \ldots, r^m$ are pairwise independent (exercise). Moreover, for every $x \in \{0,1\}^n$, $x \odot r^j = \sum_{t \in T_j} x \odot s^t$. So if we know $s^1 \odot x, s^2 \odot x, \ldots, s^k \odot x$, then we can deduce the m values $r^1 \odot x, r^2 \odot x, \ldots, r^m \odot x$.
- This is where we do exhaustive guessing. Since $2^k = O(m)$, we can enumerate over all possible guesses for $x \odot s^1, \ldots, x \odot s^k$ in polynomial time.
- This gives us the following Algorithm B' for inverting f:
  - Let $m = 200n/\epsilon^2$ and k be the smallest such that $m \leq 2^k - 1$. Choose $s^1, \ldots s^k$ independently at random from $\{0,1\}^n$ and define $r^1, \ldots, r^m$ as previously. For every string (guess) $w \in \{0,1\}^k$ do the following:
  - For every $i \in [n]$ compute our guess z_j for $x \odot r^j$ by setting $z_j = \sum_{t \in T_j} w_t \mod 2$. Compute the guess z'_j for $x \odot (r^j \oplus e^i)$ by setting $z'_j = A(x, r^j \oplus e^i)$.
  - As before, our guess for x_i is the majority value among $\{z_j \oplus z'_j\}_{j \in [m]}$.
  - Test whether $f(x) = y$, if so halt and output x.
- The analysis of B' is very similar to B. Consider the iteration when the guess is correct, i.e., $w_t = s^t \odot x$ for $t \in [k]$ and so $z_j = x \odot r^j$ for every $j \in [m]$.
- Fix some $i \in [n]$ and define the random indicator variables $Z_1, \ldots, Z_m$ as in the analysis of B.
- With the correct guess, we have $\mathbb{E}[Z] \geq (1/2 + \epsilon/2)\, m$, where $Z = \sum_j Z_j$.
- Now pairwise independence and Chebychev's inequality implies that B' succeeds with good probability by the choice of $m = 200n/\epsilon^2$.

■

3 Getting Arbitrarily Long Stretch

Theorem 4 *If f is a one-way permutation and $c \in \mathbb{N}$, then the function G that maps $x, r \in \{0,1\}^n$ to $r, f^\ell(x) \odot r, f^{\ell-1} \odot r, \ldots, f^1(x) \odot r$, where $\ell = n^c$ is a secure pseudorandom generator of stretch $\ell(2n) = n + n^c$. Here f^i denotes the function obtained by applying the function f repeatedly to the input i times.*

Proof

- By Yao's Theorem 2, it suffice to show that individual bits of G are hard to predict.
- Suppose toward contradiction that there is a probabilistic polynomial-time algorithm A such that when $x, r \in \{0,1\}^n$ and $i \in \{1, \ldots, \ell\}$ are randomly chosen,

$$\Pr[A \text{ predicts } f^i(x) \odot r \text{ given } (r, f^\ell(x) \odot r, \ldots, f^{i+1}(x) \odot r)] \geq \frac{1}{2} + \epsilon\,.$$

- We will use A to devise an algorithm B that will predict $x \odot r$ from $f(x), r$ with probability $1/2+\epsilon$. Thus if A has nonnegligible success, the B violates Theorem 3.
- Algorithm B is given r and y such that $y = f(x)$ for some x. It then picks $i \in \{1, \ldots, \ell\}$ at random and compute the values $f^{\ell-i}(y), \ldots f(y)$ and output $a = A(r, f^{\ell-i}(y) \odot r, \ldots, f(y) \odot r, y \odot r)$.
- Because, f is a permutation, this is exactly the same distribution obtained where we choose $x' \in \{0,1\}^n$ and set $x = f^i(x')$ and hence A will predict $f^i(x') \odot r$ with probability $1/2 + \epsilon$, meaning that B predicts $x \odot r$ with the same probability.

■