CMPUT 675: Computational Complexity Theory **Winter 2019**

Lecture 17 (March 12, 2019): Cryptography

Lecturer: Zachary Friggstad *Scribe: Joseph Meleshko*

17.1 Chebyshev's Inequality

Recall we are proving Yao's lemma, that a function is a pseudorandom generator if and only if it is unpredictable. Before getting to the details, we start by introducing another concentration bound. It is not as strong as Chernoff bounds but it applies in more general settings as will see.

Lemma 1 *For a random variable X with* $\mathrm{E}[X] = \mu$ *and* $\mathrm{Var}[X] = \mathrm{E}[(X-\mu)^2] = \sigma^2$

$$\Pr[|X-\mu| \geq k] \leq \frac{\sigma^2}{k^2}, \quad \textit{for any } k > 0 \tag{17.1}$$

Proof. We apply Markov's Inequality applied to the random variable $(X-\mu)^2$

$$\Pr[(X-\mu)^2 \geq k^2] \leq \frac{\mathrm{E}[(X-\mu)^2]}{k^2} \tag{17.2}$$

Taking the square root of both sides to get the inequality we want (as the bound amounts to the same event):

$$\Pr[|X-\mu| \geq k \cdot \sigma] \leq \frac{1}{k^2} \tag{17.3}$$

■

Recall a family of variables is said to be pairwise independent if any two of them are independent. We encountered such random variables when we discussed hashing. The following simple claim shows we can apply Chebyshev's inequality to sums of pairwise independent random variables and use the sum of the individual variances instead of the variance of the sum.

Lemma 2 *Let $X_1, X_2, ..., X_n$ be pairwise independent random variables over $\{0,1\}$. Let $X = \Sigma_{i=1}^n X_i$. Then* $\mathrm{Var}[X] = \sum_{i=1}^n \mathrm{Var}[X_i]$.

Proof. Say each X_i has $\mathrm{E}[X_i] = \mu_i$ and say $\mathrm{E}[X] = \mu$. Then $\mu = \sum_i \mu_i$. Also, by pairwise independence we have $\mathrm{E}[X_i X_j] = \mu_i \cdot \mu_j$ for $i \neq j$ and because each X_i is boolean we always have $X_i^2 = X_i$ so $\mathrm{E}[X_i^2] = \mu_i$.

Note $\mathrm{Var}[X_i] = \mathrm{E}[X_i^2] - \mathrm{E}[X_i]^2 = \mu_i - \mu_i^2$. So,

$$\begin{aligned}
\mathrm{Var}[X] &= \mathrm{E}[X^2] - \mathrm{E}[X]^2 \\
&= \sum_{i,j} \mathrm{E}[X_i X_j] - \mathrm{E}[\sum_i \mu_i]^2 \\
&= \sum_i \mu_i + \sum_{i \neq j} \mu_i \cdot \mu_j - \left(\sum_i \mu_i\right)^2 \\
&= \sum_i \mu_i - \sum_i \mu_i^2 \\
&= \sum_i \mathrm{Var}[X_i]
\end{aligned}$$

■

Though, this observation will not be used until next lecture.

17.2 Cryptography

Theorem 1 (Yao 82') *Given a polynomial-time computable function $G : \{0,1\}^* \to \{0,1\}^*$, if G is unpredictable, then G is pseudorandom.*

Proof.

Assume that we have an unpredictable function G with stretch $\ell(n)$ that is not pseudorandom. This means that there exists a polynomial time, probabilistic Turing machine A and a constant $c \geq 0$ such that:

$$\underbrace{\Pr_{r \sim \{0,1\}^n}[A(G(r)) = 1]}_{①} - \underbrace{\Pr_{r' \sim \{0,1\}^{\ell(n)}}[A(r') = 1]}_{②} \geq \frac{1}{n^c}, \quad \text{for } \infty\text{-ly many n. (never becomes negligible).} \tag{17.4}$$

The original definition of pseudorandom had absolute values on the left side of the above expression, but we may assume they are not there by negating the output of A, if necessary.

We define a polynomial time probabilistic Turing machine B such that $B(1^n, i, y_1, y_2, ..., y_{i-1})$:

- Samples $z_i, z_{i+1}, ..., z_n \sim \{0,1\}$ independently.
- Outputs z_i if $A(y_1, y_2, ..., y_{i-1}, z_i, z_{i+1}, ..., z_{\ell(n)}) = 1$, otherwise outputs $1 - z_i$.

We will use this machine B that invokes A with a mix of pseudorandom and truly random bits to show that:

$$\Pr_{\substack{x \sim \{0,1\}^n \\ i \sim \{1,2,...,\ell(n)\}}}[B(1^n, i, G(x)_1, G(x)_2, ..., G(x)_{i-1}) = G(x)_i] \geq \frac{1}{2} + \frac{1}{n^c \cdot \ell(n)} \tag{17.5}$$

Which demonstrates that G is not unpredictable. So, for $0 \leq i \leq \ell(n)$, let D_i be a distribution over $\{0,1\}^{\ell(n)}$ sampled by:

- $x \sim \{0,1\}^n$

- $z \sim \{0,1\}^{\ell(n)}$
- output the vector $(G(x)_1, G(x)_2, ..., G(x)_i, z_{i+1}, z_{i+2}, ..., z_{\ell(n)})$

Let $p_i = \Pr_{r \sim D_i}[A(r) = 1]$. By definition, $p_{\ell(n)} = \textcircled{1}$ and $p_0 = \textcircled{2}$ as defined above. This means:

$$p_{\ell(n)} - p_0 \geq \frac{1}{n^c} \tag{17.6}$$

$$p_{\ell(n)} - p_0 = \sum_{i=1}^{\ell(n)} p_i - p_{i-1} \tag{17.7}$$

$$\mathop{\mathrm{E}}_{i \sim \{1,2,...,\ell(n)\}}[p_i - p_{i-1}] \geq \frac{1}{n^c \cdot \ell(n)} \tag{17.8}$$

We will show

$$\forall i, \Pr_{x \sim \{0,1\}}[B(1^n, i, G(x)_1, G(x)_2, ..., G(x)_{i-1}) = G(x)_i] \geq \frac{1}{2} + (p_i - p_{i-1}). \tag{17.9}$$

For a given x and z, B predicted $G(x)_i$ correctly if:

- $\textcircled{\bar{1}} := A(G(x)_1, G(x)_2, ..., G(x)_{i-1}, z_i, z_{i+1}, ..., z_{\ell(n)}) = 1$ and $G(x)_i = z_i$

or

- $\textcircled{\bar{2}} := A(G(x)_1, G(x)_2, ..., G(x)_{i-1}, z_i, z_{i+1}, ..., z_{\ell(n)}) = 0$ and $G(x)_i \neq z_i$

Observe that conditioning D_{i-1} on $G(x)_i = z_i$ yields the same distribution as D_i. To see this, observe that an alternative way to sample from D_i is to first have it sample from D_{i-1} (but remember the whole vector x) and then replace z_i with $G(x)_i$. Since z_i is totally independent of all other choices then this is just the same as conditioning D_{i-1} on $z_i = G(x)_i$.

So,

$$\Pr_{x,z}\left[\textcircled{\bar{1}}\right] = \Pr[G(x)_i = z_i] \cdot \underbrace{\Pr[A(G(x)_1, G(x)_2, ..., G(x)_{i-1}, z_i, z_{i+1}, ..., z_{\ell(n)}) = 1 \mid G(x)_i = z_i]}_{p_i} \tag{17.10}$$

$$= \frac{1}{2} \cdot p_i \tag{17.11}$$

Similarly,

$$\Pr_{x,z}\left[\textcircled{\bar{2}}\right] = \frac{1}{2} \cdot \underbrace{(1 - \Pr[A(G(x)_1, G(x)_2, ..., G(x)_{i-1}, z_i, z_{i+1}, ..., z_{\ell(n)}) = 1 \mid G(x)_i \neq z_i])}_{\textcircled{\star}} \tag{17.12}$$

but we do not have a nice simplification for $\textcircled{\star}$ just yet.

We now observe that,

$$
\begin{aligned}
p_{i-1} &= \Pr_{r \sim D_i}[A(r) = 1] && (17.13)\\
&= \Pr_{x,z}[\underbrace{A(G(x)_1, G(x)_2, ..., G(x)_{i-1}, z_i, z_{i+1}, ..., z_{\ell(n)}) = 1}_{\textcircled{$\dagger$}}] && (17.14)\\
&= \frac{1}{2} \cdot \Pr[\textcircled{$\dagger$} \mid z_i = G(x)_i] + \frac{1}{2} \cdot \Pr[\textcircled{$\dagger$} \mid z_i \neq G(x)_i] && (17.15)\\
&= \frac{1}{2} \cdot p_i + \frac{1}{2} \cdot \Pr[\textcircled{$\dagger$} \mid z_i \neq G(x)_i] && (17.16)\\
&\frac{1}{2} \cdot p_i + \frac{1}{2} \cdot (1 - \textcircled{$\star$}). && (17.17)
\end{aligned}
$$

$\therefore$ For each given i,

$$
\begin{aligned}
\Pr_x[B \text{ predicts } G(x)_i \text{ given } G(x)_1, G(x)_2, ..., G(x)_{i-1}] &= \frac{1}{2} \cdot p_i + \frac{1}{2} \cdot \textcircled{$\star$} && (17.18)\\
&= \frac{1}{2} + (p_i - p_{i-1}) \text{ by the above observation} && (17.19)
\end{aligned}
$$

Therefore, G is not unpredictable which is a contradiction. So if G is an unpredictable function, then it is pseudorandom. ■

As mentioned in the lectures, we cannot cover why the existence of one-way functions in general imply the existence of a pseudorandom generator. But we can prove it under the slightly stronger assumption that one-way *permutations* exist: functions $f : \{0,1\}^* \to \{0,1\}^*$ that are one-to-one and satisfy $|f(x)| = |x|$ for all x.

To do this, we start with the following result which will allow us to extend a random pair (x, r) by a single bit and remain unpredictable.

Theorem 2 (Goldreich-Levin '89) *If f is a one-way permutation, for any polynomial time, probabilistic Turing machine A, there exists a negligible function $\epsilon(n)$ such that:*

$$\Pr_{x,r \sim \{0,1\}^n}[A(f(x), r) = x \circ r] \leq \frac{1}{2} + \epsilon(n), \forall n \tag{17.20}$$

Proof. Suppose not, then there exists a polynomial time, probabilistic Turing machine A such that,

$$\Pr_{x,r \sim \{0,1\}^n}[A(f(x), r) = x \circ r] \geq \frac{1}{2} + \frac{1}{n^c}, \quad \text{for } \infty\text{-ly many n. (never becomes negligible)} \tag{17.21}$$

Fix such an n. We will call an x "good" if,

$$\Pr_{r \sim \{0,1\}^n}[A(f(x), r) = x \circ r] \geq \frac{1}{2} + \frac{1}{2 \cdot n^c} \tag{17.22}$$

We only need to argue about our "good" x since the fraction of x that are good is $\frac{1}{2 \cdot n^c}$. To see this, let β be the fraction of strings in $\{0,1\}^n$ that are good. For every bad x, the probability (over r) the algorithm computes $x \circ r$ is at most $\frac{1}{2} + \frac{1}{2 \cdot n^c}$. For every good x, the probability is trivially bounded by 1. Therefore,

$$\frac{1}{2} + \frac{1}{n^c} \leq \Pr_{x,r \sim \{0,1\}^n}[A(f(x), r) = x \circ r] \leq (1 - \beta) \cdot \frac{1}{2 \cdot n^c} + \beta \cdot 1 \leq \beta + \frac{1}{2 \cdot n^c}.$$

Thus, $\beta \geq \frac{1}{2 \cdot n^c}$.

Our intuition is that for a given x, if,

$$\Pr_{r \sim \{0,1\}^n}[A(f(x), r) = x \circ r] = 1 \tag{17.23}$$

then we can just set r to e_i (standard basis vectors) and compute $A(f(x), e_i)$ which is always $x \circ e_i = x_i$ (ith bit of x) so we can find x exactly by going through each e_i. Of course, this is too strong of an assumption. We first relax it to see what happens if we have a really high probability of computing $r \circ x$.

So, what if

$$\Pr_{r \sim \{0,1\}^n}[A(f(x), r) = x \circ r] = \frac{9}{10}. \tag{17.24}$$

The idea is that we can still recover $x \circ e_i = x_i$ with very high probability using the "local decoding" trick we used in the PCP verifier with exponential proof size.

First, by the union bound on the probability that either run of A below does not give the correct answer we see

$$\Pr_{z \sim \{0,1\}^n}[A(f(x), z) = x \circ z \text{ and } A(f(x), z \oplus e^i) = x \circ (z \oplus e^i)] \geq \frac{8}{10} \tag{17.25}$$

If so, $x_i = (x \circ z) + (x \circ (z \oplus e^i)) = x \circ (z \oplus z \oplus e^i) = x \circ e^i = x_i$. Since the probability is a constant factor larger than $1/2$, we can repeat the protocol multiple times and take the majority to compute x_i correctly with very high probability using Chernoff bounds. High enough so that if we repeat this for each i then we recover *all* x_i with high probability.

This doesn't work for our original case since our probability is only slightly above $\frac{1}{2}$ so the union bound argument for the "local decoding" step fails. We now turn to the general case

$$\Pr_{r \sim \{0,1\}^n}[A(f(x), r) = x \circ r] \geq \frac{1}{2} + \frac{1}{2 \cdot n^c} \tag{17.26}$$

The key idea is that we can, in some sense, "guess" the values of $x \circ z$ and still treat $x \circ (z \oplus e^i)$ as being successfully read with probability $\frac{1}{2} + \frac{1}{2 \cdot n^c}$. The way we do this will not have the various z being truly independent, but they will be pairwise-independent. So querying multiple times with such z and taking the majority will still successfully determine x_i with high probability: we just have to use Chebyshev's inquality from the start of the lecture rather than the full power of Chernoff bounds (which do not hold in general if we only assume pairwise independence of the variables).

We now describe how to sample various z strings in a pairwise-independent fashion that also allows us to "know" the value of $x \circ z$. Let $m = 200 \cdot n^{2c+1}$ and k be the smallest integer such that $m \leq 2^k$. (k is the number of bits to write m).

- Sample $s_1, s_2, ..., s_k \sim \{0,1\}^n$
- $\forall 1 \leq j \leq m$, Let $T_j = \{i \mid \text{bit } i \text{ of } j \text{ is } 1\}$ (positions of 1 bits in j)
- $z_j = \sum\limits_{i \in T_j} s_i$ (the z_i's can be shown to be pairwise independent)
- Note: $\forall x \in \{0,1\}^n, x \circ z_j = \sum\limits_{i \in T_j} x \circ s_i$

Claim 1 *The vectors $z_j, 1 \leq j \leq m$ are pairwise-independent.*

Proof. Consider $j \neq j'$ and say bit k is 1 in j and 0 in j'. Consider first sampling all s_i except for $i = k$. Then $z_{j'}$ is already determined. But then s_k, being a completely random string, then means z_j will be completely independent of $z_{j'}$. ■

We will then "guess" each $x \circ s_i$ for each $1 \leq i \leq k$ at once. That is, try each of the $2^k = \text{poly}(m) = \text{poly}(n)$ guesses for all of these values and run the following for each guess.

- Let $y_j = \sum_{i \in T_j} x \circ s_i$ (we know y because we have guessed $x \circ s_i$)
- Let $y'_j = A(f(x), z_j \oplus e_i)$
- For each $1 \leq i \leq n$, let $\overline{x}_i$ be the majority of $y_i \oplus y'_i$.

We will output $\overline{x}$ if $f(\overline{x}) = f(x)$. We can verify if the $\overline{x}$ we computed works since we can evaluate $f(\overline{x})$ and compare, only halting if we found a vector $\overline{x}$ that works how we expect.

Next lecture, we will see that $\overline{x} = x$ with good probability for the iteration where we guessed correctly. ■