Lecture Notes 13

32 Review of Lecture 12

Lecture 12 covered several technical proofs fairly quickly. In this lecture, we reviewed the three theorems from last time and focused on the key ideas that made the proofs work. (The complete proofs are contained in the written notes for Lecture 12.)

Building G from G_1: The big idea that makes the hybrid proof work is that the distinguisher D' is given as argument either $\alpha = G_1(U_n)$ or $\alpha = U_{n+1}$. In either case, it splits α into two parts $\tau \cdot \alpha'$ and constructs the string $w = \tau \cdot G(\alpha')$. If $\alpha = G_1(U_n)$, then $w = \tau \cdot G(\alpha') = G(U_n)$. If $\alpha = U_n$, then $w = U_1 \cdot G(U_n)$. Hence, the difference is simply whether w starts directly with the bits from $G(U_n)$ or whether it starts with a random bit τ followed by $G(U_n)$. This fits in with the construction of the hybrids, so when w is prefixed with a random β of length-k and truncated to total length $p(n)$, the resulting hybrid is either H_n^k or H_n^{k+1}.

Pseudorandom $\Leftarrow$ Unpredictable: Here, the hybrids are strings with k bits from the pseudorandom generator followed by $n - k$ random bits. Assume D is a distinguisher for the pseudorandom strings. The next-bit predictor A that we construct generates a hybrid for a randomly-chosen k and then calls the distinguisher D on it. A outputs the first of the $n - k$ random bits if the distinguisher answers 1, and it outputs the complement otherwise. This works since D was assumed to be more likely to output 1 on pseudorandom strings than on truly random ones.

Pseudorandom implies strongly one-way Given a pseudorandom function $G(s)$ with expansion factor $2n$, the proof constructs the function $f(x, y) = G(x)$ and proceeds to derive a contradiction to the assumption that f is not strongly one-way. While f is attractive because it is length-preserving, it seems that the proof goes through just fine for G itself. Assume G were not strongly one-way, and let A be an algorithm that inverts it with non-negligible success probability. Then here's how to distinguish $G(s)$ from U_{2n}. On input α, use A to find s' such that $G(s') = \alpha$. If such an s' is found, output 1; else output 0. If the input is $\alpha = G(s)$, the distinguisher will output 1 with probability the same as A's success probability, which is at least $\frac{1}{p(n)}$ for some polynomial $p(\cdot)$. If the input is $\alpha = U_{2n}$, then the distinguisher will output 0 whenever U_{2n} is not in the range of $G(s)$ (as s ranges over length-n strings). The size of the range is 2^n but the number of choices for α is 2^{2n}, so the probability of giving output 1 in this case is at most $2^n/2^{2n} = 1/2^n$. Hence, the difference in probabilities is at least $\frac{1}{p(n)} - \frac{1}{2^n} > \frac{1}{2p(n)}$, contradicting the assumption that G is pseudorandom.

33 Pseudorandom Generator from Strongly One-Way Permutation

The construction of a pseudorandom generator from an arbitrary strongly one-way function is complicated and not included in the textbook. However, if strongly one-way permutations exist, then it is straightforward to construct a pseudorandom generator.

Theorem 1 *Let f be a length-preserving 1-1 strongly one-way function and b a hard core for f. Then*

$$G(s) = f(s) \cdot b(s)$$

is a pseudorandom generator.

Proof: Assume G is not pseudorandom. Then there exists a bit predictor A with significant advantage. Let $\alpha = G(s)$, where $s = U_n$. Because f is 1-1, $f(U_n)$ is uniformly distributed on length-n strings, so A can't predict any of the first n bits of α with any advantage. Since A does have an advantage overall, then it must have an advantage at predicting bit $n+1$. But this means that A is able to get an advantage at predicting $b(s)$ after having read $f(s)$, contradicting the assumption that $b(s)$ is a hard core for f. ■

Theorem 1 gives an easy and efficient method for constructing a polynomial-expansion pseudorandom number generator given a strongly one-way permutation f with a hard core predicate b. Namely, on initial seed s, do the following:

1. $s_0 = s$
2. $n = |s|$
3. for $j = 1$ to $p(n)$ {
4. $\quad \sigma_j = b(s_{j-1})$
5. $\quad s_j = f(s_{j-1})$
6. }
7. Output $\sigma_1 \ldots \sigma_{p(n)}$.

34 Pseudorandom Functions

An *ℓ-bit function ensemble* is a sequence $F = \{F_n\}_{n \in \mathbb{N}}$ of random variables such that F_n assumes values in the set of functions mapping $\ell(n)$-bit-long strings to $\ell(n)$-bit-long strings. The *uniform ℓ-bit function ensemble*, $H = \{H_n\}_{n \in \mathbb{N}}$, has H_n uniformly distributed over the set of all functions mapping $\{0,1\}^{\ell(n)} \to \{0,1\}^{\ell(n)}$.

Definition: An *ℓ-bit function ensemble* F is *pseudorandom* if, for every probabilistic polynomial-time oracle machine M, every positive polynomial $p(\cdot)$, and all sufficiently large n,

$$|\Pr[M^{F_n}(1^n) = 1] - \Pr[M^{H_n}(1^n) = 1]| < \frac{1}{p(n)}.$$

Definition: An *ℓ-bit function ensemble* F is *efficiently computable* iff there exist polynomial-time algorithms I, V such that

1. $\phi(I(1^n))$ and F_n are identically distributed, where ϕ maps strings to functions. (Think of the strings as somehow describing functions, and $\phi(i)$ is the function that the string i describes.)
2. $V(i, x) = f_i(x)$ for every i in the range of $I(1^n)$ and $x \in \{0,1\}^{\ell(n)}$, where $f_i = \phi(i)$ is the function described by i.

In the next lecture, we will show how to construct an efficiently computable ℓ-bit pseudorandom function ensemble, starting from a pseudorandom generator G with expansion factor $2n$.