Notes for Lecture 13

Scribed by Siu-On Chan, posted March 12, 2009

Summary

Today we complete the proof that it is possible to construct a pseudorandom generator from a one-way permutation.

1 Pseudorandom Generators from One-Way Permutations

Last time we proved the Goldreich-Levin theorem.

Theorem 1 (Goldreich and Levin) Let $f : \{0,1\}^n \rightarrow \{0,1\}^n$ be a $(t,\epsilon)$-one way permutation computable in time $r \leq t$. Then the predicate $x, r \mapsto \langle x, r \rangle$ is $(\Omega(t \cdot \epsilon^2 \cdot n^{-O(1)}), 3\epsilon)$ hard core for the permutation $x, r \mapsto f(x), r$.

A way to look at this result is the following: suppose $f$ is $(2^{\Omega(n)}, 2^{-\Omega(n)})$ one way and computable in $n^{O(1)}$ time. Then $\langle x, r \rangle$ is a $(2^{\Omega(n)}, 2^{-\Omega(n)})$ hard-core predicate for the permutation $x, r \mapsto f(x), r$.

From now on, we shall assume that we have a one-way permutation $f : \{0,1\}^n \rightarrow \{0,1\}^n$ and a predicate $P : \{0,1\}^n \rightarrow \{0,1\}$ that is $(t, \epsilon)$ hard core for $f$.

This already gives us a pseudorandom generator with one-bit expansion.

Theorem 2 (Yao) Let $f : \{0,1\}^n \rightarrow \{0,1\}^n$ be a permutation, and suppose $P : \{0,1\}^n \rightarrow \{0,1\}$ is $(t, \epsilon)$-hard core for $f$. Then the mapping

$$x \mapsto P(x), f(x)$$

is $(t - O(1), \epsilon)$-pseudorandom generator mapping $n$ bits into $n + 1$ bits.

Note that $f$ is required to be a permutation rather than just a function. If $f$ is merely a function, it may always begin with 0 and the overall mapping would not be pseudorandom.
For the special case where the predicate $P$ is given by Goldreich-Levin, the mapping would be

$$x \mapsto (x, r, f(x), r)$$

**Proof:** Suppose the mapping is not $(t - 2, \epsilon)$-pseudorandom. There is an algorithm $D$ of complexity $\leq t - 2$ such that

$$\left| \Pr_{x \sim \{0,1\}^n} [D(P(x)f(x)) = 1] - \Pr_{b \sim \{0,1\}, \ x \sim \{0,1\}^n} [D(bf(x)) = 1] \right| > \epsilon$$

(1)

where we have used the fact that since $f$ is permutation, $f(x)$ would be a uniformly random element in $\{0,1\}^n$ when $x$ is such.

We will first remove the absolute sign in (1). The new inequality holds for either $D$ or $1 - D$ (i.e. the complement of $D$), and they both have complexity at most $t - 1$.

Now define an algorithm $A$ as follows.

On input $y = f(x)$, pick a random bit $r \sim \{0,1\}$. If $D(r, y) = 1$, then output $r$, otherwise output $1 - r$.

Algorithm $A$ has complexity at most $t$. We claim that

$$\Pr_{x \sim \{0,1\}^n} [A(f(x)) = P(x)] > \frac{1}{2} + \epsilon$$

so $P(\cdot)$ is not $(t, \epsilon)$-hard core.

To make explicit the dependence of $A$ on $r$, we will denote by $A_r(f(x))$ the fact that $A$ picks $r$ as its random bit.

To prove the claim, we expand

$$\Pr_{x, r} [A_r(f(x)) = P(x)] = \Pr_{x, r} [A_r(f(x)) = P(x) | r = P(x)] \Pr[r = P(x)] + \Pr_{x, r} [A_r(f(x)) = P(x) | r \neq P(x)] \Pr[r \neq P(x)]$$
Note that \( \Pr[r = P(x)] = \Pr[r \neq P(x)] = 1/2 \) no matter what \( P(x) \) is. The above probability thus becomes

\[
\frac{1}{2} \Pr_{x,r}[D(rf(x)) = 1 | r = P(x)] + \frac{1}{2} \Pr_{x,r}[D(rf(x)) = 0 | r \neq P(x)]
\]

(2)

The second term is just \( \frac{1}{2} - \frac{1}{2} \Pr_{x,r}[D(rf(x)) = 1 | r \neq P(x)] \). Now we add to and subtract from (2) the quantity \( \frac{1}{2} \Pr_{x,r}[D(rf(x)) = 1 | r = P(x)] \), getting

\[
\frac{1}{2} + \Pr_{x,r}[D(rf(x)) = 1 | r = P(x)] -
\left( \frac{1}{2} \Pr[D(rf(x)) = 1 | r = P(x)] + \right.
\]

\[
\left. \frac{1}{2} \Pr[D(rf(x)) = 1 | r \neq P(x)] \right)
\]

The expression in the bracket is \( \Pr[D(rf(x)) = 1] \), and by our assumption on \( D \), the whole expression is more than \( \frac{1}{2} + \epsilon \), as claimed.

\[\square\]

The main idea of the proof is to convert something that distinguishes (i.e. \( D \)) to something that outputs (i.e. \( A \)). \( D \) helps us distinguish good answers and bad answers.

We will amplify the expansion of the generator by the following idea: from an \( n \)-bit input, we run the generator to obtain \( n + 1 \) pseudorandom bits. We output one of those \( n + 1 \) bits and feed the other \( n \) back into the generator, and so on. Specialized to the above construction, and repeated \( k \) times the mapping becomes

\[
G_k(x) := P(x), P(f(x)), P(f(f(x))), \ldots, P(f^{(k-1)}(x)), f^{(k)}(x)
\]

(3)

This corresponds to the following diagram where all output bits lie at the bottom.

\[\text{Theorem 3 (Blum-Micali)} \ Let f : \{0,1\}^n \rightarrow \{0,1\}^n \ be a permutation, and suppose P : \{0,1\}^n \rightarrow \{0,1\} \ is (t, \epsilon)-hard core for f and that f, P are computable with complexity r.\]
Then $G_k : \{0, 1\}^n \to \{0, 1\}^{n+k}$ as defined in (3) is $(t - O(rk), \epsilon k)$-pseudorandom.

**Proof:** Suppose $G_k$ is not $(t - O(rk), \epsilon k)$-pseudorandom. Then there is an algorithm $D$ of complexity at most $t - O(rk)$ such that

$$\left| \Pr_{x \sim \{0,1\}^n} [D(G_k(x)) = 1] - \Pr_{z \sim \{0,1\}^{n+k}} [D(z) = 1] \right| > \epsilon k$$

We will then use the hybrid argument. We will define a sequence of distributions $H_0, \ldots, H_k$, the first is $G_k$’s output, the last is uniformly random bits, and every two adjacent ones differ only in one invocation of $G$.

More specifically, define $H_i$ to be the distribution where we intercept the output of the first $i$ copies of $G$’s, replace them with random bits, and run the rest of $G_k$ as usual (see the above figure in which blue lines represent intercepted outputs). Then $H_0$ is just the distribution of the output of $G_k$, and $H_k$ is the uniform distribution, as desired. Now

$$\epsilon k < \left| \Pr_{z \sim H_0} [D(z) = 1] - \Pr_{z \sim H_k} [D(z) = 1] \right| = \sum_{i=0}^{k-1} \left( \Pr_{z \sim H_i} [D(z) = 1] - \Pr_{z \sim H_{i+1}} [D(z) = 1] \right)$$

So there is an $i$ such that

$$\left| \Pr_{z \sim H_i} [D(z) = 1] - \Pr_{z \sim H_{i+1}} [D(z) = 1] \right| > \epsilon$$

In both $H_i$ and $H_{i+1}$, the first $i$ bits $r_1, \ldots, r_i$ are random.

We now define a new algorithm $D'$ that takes as input $b, y$ and has output distribution $H_i$ or $H_{i+1}$ in two special cases: if $b, y$ are drawn from $P(x), f(x)$, then $D'$ has output distribution $H_i$; if $b, y$ are drawn from (random bit), $f(x)$, then $D'$ has output distribution $H_{i+1}$. In other words, if $b, y$ are $P(x), f(x)$, $D'$ should output

$$r_1, \ldots, r_i, P(x), P(f(x)), \ldots, P(f^{(k-i-1)}(x)), f^{(k-i)}(x)$$
If $b, y$ are (random bit), $f(x)$, $D'$ should output

$$r_1, \ldots, r_i, P(f(x)), \ldots, P(f^{(k-i-1)}(x)), f^{(k-i)}(x)$$

This suggests that $D'$ on input $b, y$ should pick random bits $r_1, \ldots, r_i$ and output $r_1, \ldots, r_i, b, P(y), \ldots, P(f^{(k-i-2)}(y)), f^{(k-i-1)}(y)$.

We have

$$\left| \Pr_{x \sim \{0,1\}^n} [D'(P(x)f(x)) = 1] - \Pr_{z \sim \{0,1\}^{n+1}} [D'(z) = 1] \right|$$

$$\leq \left| \Pr_{x \sim H_i} [D'(x) = 1] - \Pr_{x \sim H_{i+1}} [D'(x) = 1] \right|$$

$$> \epsilon$$

and $P(\cdot)$ is not $(t, \epsilon)$-hard core. □

Thinking about the following problem is a good preparation for the proof the main result of the next lecture.

**Exercise 1 (Tree Composition of Generators)** Let $G : \{0,1\}^n \rightarrow \{0,1\}^{2n}$ be a $(t, \epsilon)$ pseudorandom generator computable in time $r$, let $G_0(x)$ be the first $n$ bits of the output of $G(x)$, and let $G_1(x)$ be the last $n$ bits of the output of $G(x)$.

Define $G' : \{0,1\}^n \rightarrow \{0,1\}^{4n}$ as

$$G'(x) = G(G_0(x)), G(G_1(x))$$

Prove that $G'$ is a $(t - O(r), 3\epsilon)$ pseudorandom generator.