Notes for Lecture 14

Scribed by Madhur Tulsiani, posted March 20, 2009

Summary

Today we show how to construct a pseudorandom function from a pseudorandom generator.

1 Pseudorandom generators evaluated on independent seeds

We first prove a simple lemma which we will need. This lemma simply says that if G is a pseudorandom generator with output length m, then if we evaluate G on k independent seeds the resulting function is still a pseudorandom generator with output length km.

Lemma 1 (Generator Evaluated on Independent Seeds) *Let $G : \{0,1\}^n \to \{0,1\}^m$ be a (t, ϵ) pseudorandom generator running in time t_g. Fix a parameter k, and define $G^k : \{0,1\}^{kn} \to \{0,1\}^{km}$ as*

$$G^k(x_1, \ldots, x_k) := G(x_1), G(x_2), \ldots, G(x_k)$$

Then G^k is a $(t - O(km + kt_g), k\epsilon)$ pseudorandom generator.

Proof: We will show that if for some (t, ϵ), G^k is not a (t, ϵ) psedorandom generator, then G cannot be a $(t + O(km + kt_g), \epsilon/k)$ pseudorandom generator.

The proof is by a hybrid argument. If G^k is not a (t, ϵ) pseudorandom generator, then there exists an algorithm D of complexity at most t, which distinguishes the output of G^k on a random seed, from a truly random string of km bits i.e.

$$\left| \mathop{\mathbb{P}}_{x_1,\ldots,x_k}[D(G(x_1), \ldots, G(x_k)) = 1] - \mathop{\mathbb{P}}_{r_1,\ldots,r_k}[D(r_1, \ldots, r_k) = 1] \right| \; > \; \epsilon$$

We can then define the hybrid distributions $H_0, \ldots, H_k$, where in H_i we relplace the first i outputs of the pseudorandom generator G by truly random strings.

$$H_i = (r_1, \ldots, r_i, G(x_{i+1}), \ldots, G(x_n))$$

As before, the above statement which says $|\mathbb{P}_{z\sim H_0}[D(z)=1]-\mathbb{P}_{z\sim H_k}[D(z)=1]|>\epsilon$ would imply that there exists an i between 0 and $k-1$ such that

$$\left|\mathop{\mathbb{P}}_{z\sim H_i}[D(z)=1]-\mathop{\mathbb{P}}_{z\sim H_{i+1}}[D(z)=1]\right|>\epsilon/k$$

We can now define an algorithm D' which violates the pseudorandom property of the generator G. Given an input $y\in\{0,1\}^m$, D' generates random strings $r_1,\ldots,r_i\in\{0,1\}^m$, $x_{i+2},\ldots,x_k\in\{0,1\}^n$, and outputs $D(r_1,\ldots,r_i,y,G(x_{i+2}),\ldots,G(x_k))$. It then follows that

$$\mathop{\mathbb{P}}_{x\sim\{0,1\}^n}[D'(G(x))=1] \;=\; \mathop{\mathbb{P}}_{z\sim H_i}[D(z)=1] \;\text{ and }\; \mathop{\mathbb{P}}_{r\sim\{0,1\}^m}[D'(r)=1] \;=\; \mathop{\mathbb{P}}_{z\sim H_{i+1}}[D(z)=1]$$

Hence, D' distinguishes the output of G on a random seed x from a truly random string r, with probability at least ϵ/k. Also, the complexity of D' is at most $t+O(km)+O(kt_g)$, where the $O(km)$ term corresponds to generating the random strings and the $O(kt_g)$ terms corresponds to evaluating G on at most k random seeds. □

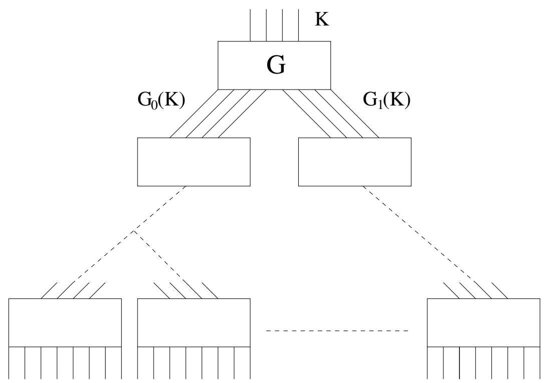
2 Construction of Pseudorandom Functions

We now describe the construction of a pseudorandom function from a pseudorandom generator. Let $G:\{0,1\}^n\to\{0,1\}^{2n}$ be a length-doubling pseudorandom generator. Define $G_0:\{0,1\}^n\to\{0,1\}^n$ such that $G_0(x)$ equals the first n bits of $G(x)$, and define $G_1:\{0,1\}^n\to\{0,1\}^n$ such that $G_1(x)$ equals the last n bits of $G(x)$.

The the GGM pseudorandom function based on G is defined as follows: for key $K\in\{0,1\}^n$ and input $x\in\{0,1\}^n$:

$$F_K(x) := G_{x_n}(G_{x_{n-1}}(\cdots G_{x_2}(G_{x_1}(K))\cdots)) \tag{1}$$

The evaluation of the function F can be visualized by considering a binary tree of depth n, with a copy of the generator G at each node. The root receives the input K and passes the outputs $G_0(K)$ and $G_1(K)$ to its two children. Each node of the tree, receiving an input z, produces the outputs $G_0(z)$ and $G_1(z)$ which are passed to its children if its not a leaf. The input x to the function F_K, then selects a path in this tree from the root to a leaf, and produces the output given by the leaf.



We will prove that if $G : \{0,1\}^n \to \{0,1\}^{2n}$ is a (t, ϵ) pseudorandom generator running in time t_g, then F is a $(t/O(n \cdot t_g), \epsilon \cdot nt)$ secure pseudorandom function.

2.1 Considering a tree of small depth

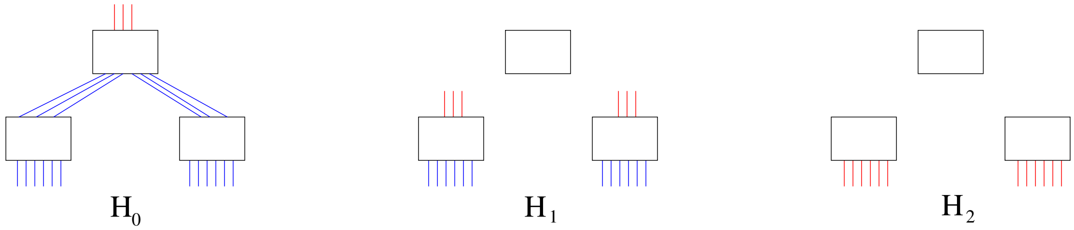
We will first consider a slightly simpler situation which illustrates the main idea. We prove that if G is (t, ϵ) pseudorandom and runs in time t_g, then the concatenated output of all the leaves in a tree with l levels, is $(t - O(2^l \cdot t_g), l2^l \cdot \epsilon)$ pseudorandom. The result is only meaninful when l is much smaller than n.

Theorem 2 *Suppose $G : \{0,1\}^n \to \{0,1\}^{2n}$ is a (t, ϵ) pseudorandom generator and G is computable in time t_g. Fix a constant l and define $F_K : \{0,1\}^l \to \{0,1\}^n$ as $F_K(y) := G_{y_l}(G_{y_{l-1}}(\cdots G_{y_2}(G_{y_1}(K)) \cdots))$ Then $\overline{G} : \{0,1\}^n \to \{0,1\}^{2^l \cdot n}$ defined as*

$$\overline{G}(K) := (F_K(0^l), F_K(0^{l-1}1), \ldots, F_K(1^l))$$

is a $(t - O(2^l \cdot t_g), l \cdot 2^l \cdot \epsilon)$ pseudorandom generator.

Proof: The proof is again by a hybrid argument. The hybrids we consider are easier to describe in terms of the tree with nodes as copies of G. We take H_i to be the distribution of outputs at the leaves, when the input to the nodes at depth i is replaced by truly random bits, ignoring the nodes at depth $i-1$. Hence, H_0 is simply distributed as $\overline{G}(K)$ for a random K i.e. only the input to the root is random. Also, in H_l we replace the outputs at depth $l-1$ by truly random strings. Hence, H_l is simply distributed as a random string of length $n \cdot 2^l$. The figure below shows the hybrids for the case $l = 2$, with red color indicating true randomness.



We will prove that $\overline{G}$ is not a (t, ϵ) secure pseudorandom generator, then G is not $(t+O(2^l \cdot t_g), \epsilon/(l \cdot 2^l))$ secure. If we assume that there is an algorithm D of complexity t such that

$$\left| \mathop{\mathbb{P}}_{x \sim \{0,1\}^n}[D(\overline{G}(x)) = 1] - \mathop{\mathbb{P}}_{r \sim \{0,1\}^{2^l \cdot n}}[D(r) = 1] \right| \; > \; \epsilon$$

then we get that there is an i such that $\left|\mathbb{P}_{z \sim H_i}[D(z) = 1] - \mathbb{P}_{z \sim H_{i+1}}[D(z) = 1]\right| \; > \; \epsilon/l$.

We now consider again the difference between H_i and H_{i+1}. In H_i the $2^i \cdot n$ bits which are the inputs to the nodes at depth i are replaced by random bits. These are then used to generate $2^{i+1} \cdot n$ bits which serve as inputs to nodes at depth $i+1$. In H_{i+1}, the inputs to nodes at depth $i+1$ are random.

Let $\overline{G}_{i+1} : \{0,1\}^{2^{i+1} \cdot n} \to \{0,1\}^{2^l \cdot n}$ denote the function which given $2^{i+1} \cdot n$ bits, treats them as inputs to the nodes at depth $i+1$ and evaluates the output at the leaves in the tree for $\overline{G}$. If $r_1, \ldots, r_{2^i} \sim \{0,1\}^{2n}$, then $\overline{G}_{i+1}(r_1, \ldots, r_{2^i})$ is distributed as H_{i+1}. Also, if $x_1, \ldots, x_{2^i} \sim \{0,1\}^n$, then $\overline{G}_{i+1}(G(x_1), \ldots, G(x_{2^i}))$ is distributed as H_i.

Hence, D can be used to create a distinguisher D' which distinguishes G evaluated on 2^i independent seeds, from 2^i random strings of length $2n$. In particular, for $z \in \{0,1\}^{2^{i+1} \cdot n}$, we take $D'(z) = D(G_{i+1}(z))$. This gives

$$\left| \mathop{\mathbb{P}}_{x_1, \ldots, x_{2^i}}[D'(G(x_1), \ldots, G(x_{2^i})) = 1] - \mathop{\mathbb{P}}_{r_1, \ldots, r_{2^i}}[D'(r_1, \ldots, r_{2^i}) = 1] \right| \; > \epsilon/l$$

Hence,D' distinguishes $G^{2^i}(x_1, \ldots, x_{2^i}) = (G(x_1), \ldots, G(x_{2^i}))$ from a random string. Also, G' has complexity $t + O(2^l \cdot t_g)$. However, by Lemma 1, if G^{2^i} is not $(t + O(2^l \cdot t_g), \epsilon/l)$ secure then G is not $(t + O(2^l \cdot t_g + 2^i \cdot n), \epsilon/(l \cdot 2^i))$ secure. Since $i \leq l$, this completes the proof. $\square$

2.2 Proving the security of the GGM construction

Recall that the GGM function is defined as

$$F_K(x) := G_{x_n}(G_{x_{n-1}}(\cdots G_{x_2}(G_{x_1}(K)) \cdots))$$

We will prove that

Theorem 3 *If $G : \{0,1\}^n \to \{0,1\}^{2n}$ is a (t, ϵ) pseudorandom generator and G is computable in time t_g, then F is a $(t/O(nt_g), \epsilon \cdot n \cdot t)$ secure pseudorandom function.*

Proof: As before, we assume that F is not a (t, ϵ) secure pseudorandom function, and will show that this implies G is not a $(t \cdot O(nt_g), \epsilon/(n \cdot t))$ pseudorandom generator. The assumption that F is not (t, ϵ) secure, gives that there is an algorithm A of complexity at most t which distinguishes F_K on a random seed K from a random function R, i.e.

$$\left| \mathbb{P}_K \left[A^{F_K(\cdot)} = 1 \right] - \mathbb{P}_R \left[A^{R(\cdot)} = 1 \right] \right| > \epsilon$$

We consider hybrids $H_0, \ldots, H_n$ as in the proof of Theorem 2. H_0 is the distribution of F_K for $K \sim \{0,1\}^n$ and H_n is the uniform distribution over all functions from $\{0,1\}^n$ to $\{0,1\}^n$. As before, there exists i such that

$$\left| \mathbb{P}_{h \sim H_i} \left[A^{h(\cdot)} = 1 \right] - \mathbb{P}_{h \sim H_{i+1}} \left[A^{h(\cdot)} = 1 \right] \right| > \epsilon / n$$

However, now we can no longer use A to construct a distinguisher for G^{2^i} as in Theorem 2 since i may now be as large as n. The important observation is that since A has complexity t, it can make at most t queries to the function it is given as an oracle. Since the (at most) t queries made by A will be paths in the tree from the root to the leaves, they can contain at most t nodes at depth $i+1$. Hence, to simulate the behavior of A, we only need to generate the value of a function distributed according to H_i or H_{i+1} at t inputs.

We will use this to contruct an algorithm D which distinguishes the output of G^t on t independent seeds from t random strings of length $2n$. D takes as input a string of length $2tn$, which we treat as t pairs $(z_{1,0}, z_{1,1}), \ldots, (z_{t,0}, z_{t,1})$ with each $z_{i,j}$ being of length n. When queried on an input $x \in \{0,1\}^n$, D will pick a pair $(z_{k,0}, z_{k,1})$ according to the first i bits of x (i.e. choose the randomness for the node at depth i which lies on the path), and then choose $z_{k,x_{i+1}}$. In particular, $D((z_{1,0}, z_{1,1}), \ldots, (z_{t,0}, z_{t,1}))$ works as below:

1. Start with counter $k = 0$.
2. Simulate A. When given a query x
   - Check if a pair $P(x_1, \ldots, x_i)$ has already been chosen from the first k pairs.
   - If not, set $P(x_1, \ldots, x_{i+1}) = k + 1$ and set $k = k + 1$.
   - Answer the query made by A as $G_{x_n}(\cdots G_{i+2}(z_{P(x_1,\ldots,x_{i+1}),x_{i+1}}) \cdots)$.
3. Return the final output given by A.

Then, if all pairs are random strings $r_1, \ldots, r_t$ of length $2n$, the answers received by A are as given by a oracle function distributed according to H_{i+1}. Hence,

$$\mathbb{P}_{r_1,\ldots,r_t} \left[D(r_1, \ldots, r_t) = 1 \right] = \mathbb{P}_{h \sim H_{i+1}} \left[A^{h(\cdot)} = 1 \right]$$

Similarly, if the t pairs are outputs of the pseudorandom generator G on independent seeds $x_1, \ldots, x_t \in \{0,1\}^n$, then the view of A is the same as in the case with an oracle function distributed according to H_i. This gives

$$\mathop{\mathbb{P}}_{x_1,\ldots,x_t}[D(G(x_1), \ldots, G(x_t)) = 1] = \mathop{\mathbb{P}}_{h \sim H_i}\left[A^{h(\cdot)} = 1\right]$$

Hence, D distinguishes the output of G^t from a random string with probability ϵ/n. Also, it runs in time $O(t \cdot n \cdot t_g)$. Then Lemma 1 gives that G is not $(O(t \cdot n \cdot t_g), \epsilon/(n \cdot t))$ secure. $\square$