Modern Cryptography

600.442

Lecture #11

Dr. Christopher Pappacena

Fall 2013

Last Time

- One-way functions
- Hardcore predicates
- Started to prove the Goldreich-Levin Theorem.

Goldreich-Levin Theorem

Let f be a one-way function and define g by $g(x, r) = (f(x), r)$ for $|r| = |x|$. Define gl$(x, r) = x \cdot r = \oplus_{i=1}^{n} x_i r_r$. Then gl$(x, r)$ is a hardcore predicate for g.

Proof

Given $\mathcal{A}$ which can guess $\mathsf{gl}(x, r)$ with advantage $\epsilon(n)$, run the following algorithm $\mathcal{A}'$:

- Set $\ell = \lceil \log(2n/\epsilon(n)^2 + 1) \rceil$.

- Choose $s_1, \ldots, s_\ell \in \{0, 1\}^n$ and guess the bits $\sigma_i = \mathsf{gl}(x, s_i)$.

- For every j and every nonempty subset I of $\{1, \ldots, \ell\}$, use $\mathcal{A}$ to produce a guess for $\mathsf{gl}(x, r_I \oplus e_j)$.

- Set x'_j equal to the majority vote of $\{\sigma_j \oplus \mathsf{gl}(x, r_I \oplus e_j)\}$ and return x'.

When Does $\mathcal{A}'$ Succeed?

The algorithm will succeed when the following events happen:

- The value x belongs to S_n.
- $\mathcal{A}'$ correctly guesses the ℓ values $\sigma_i = \mathsf{gl}(x, s_i)$.
- $\mathcal{A}$ computes the correct value of $\mathsf{gl}(x, r_I \oplus e_j)$ for a majority of the indices I.

Note that $\mathcal{A}'$ can also succeed if some of these events don't happen: Mod 2, two wrongs make a right!

The probability of each of the first two events is easy to estimate:

- x is in S_n with probability $\geq \epsilon(n)/2$.

- Since $\ell \leq \log(2n/\epsilon(n)^2 + 1) + 1$, $\mathcal{A}'$ guesses the ℓ values correctly with probability

$$2^{-\ell} \geq \frac{1}{2\left(2n/\epsilon(n)^2 + 1\right)} \geq \frac{\epsilon(n)^2}{5n}.$$

Note that these events are independent.

A Leap of Faith

To estimate the probability that $\mathcal{A}$ computes the correct value of x_j we need to use a result which we will not prove:

Lemma: Suppose that $\{X_1, \ldots, X_t\}$ are pairwise independent binary random variables such that $\Pr[X_i = 1] \geq \frac{1}{2} + \epsilon$ for some ϵ. Let X be the majority vote of $\{X_1, \ldots, X_t\}$. Then

$$\Pr[X = 0] \leq \frac{1}{4\epsilon^2 t}.$$

The proof can be found in Katz and Lindell, pp. 208–210.

We can apply this result to bound the probability that $\mathcal{A}$ correctly determines the bit x_j, conditioned on x being in S_n and $\mathcal{A}'$ guessing $\{\mathsf{gl}(x, s_i)\}$ correctly.

For each subset I we let X_I be the random variable which is 1 if the estimate for x_j is correct, 0 otherwise. Then $\Pr[X_I = 1] \geq \frac{1}{2} + \frac{\epsilon(n)}{2}$.

So the probability that the majority vote gets x_j wrong is

$$\Pr[X = 0] \leq \frac{1}{4(\epsilon(n)/2)^2(2^\ell - 1)} \leq \frac{1}{2n}.$$

Hence the conditional probability that $x' = x$ is $\geq \frac{1}{2}$.

Putting it all Together

In total, the probability that $\mathcal{A}'$ correctly inverts f is at least

$$\frac{\epsilon(n)}{2} \times \frac{\epsilon(n)^2}{5n} \times \frac{1}{2} = \frac{\epsilon(n)^3}{20n}.$$

So, if $\epsilon(n)$ is non-negligible, then we can invert f with non-negligble probability. This shows that gl is a hardcore predicate for g.

Remarks

- We have *not* shown that f itself has a hard-core predicate. In fact, it is *unknown* whether a general one-way function has a hardcore predicate.

- Many cryptography references (including the textbook and the original Goldreich-Levin paper) are sloppy on this point.

- If $f(x)$ is a permutation, then so is $g(x, r) = (f(x), r)$, a fact we will use later.

- We have taken full advantage of the asymptotic method by not being too careful with our probability estimates.

Whew!

With that hard work behind us we can use one-way functions and hard-core predicates to construct pseudorandom generators, pseudorandom functions, and strong pseudorandom permutations.

Actually, we'll base our constructions on one-way *permutations* with hardcore predicates.

Pseudorandom Generators

As a warm-up, we will construct a pseudorandom generator G with an expansion factor of $\ell(n) = n + 1$.

Theorem: Let f be a one-way permutation with hardcore predicate hc. Then the function $G(s) = (f(s), \mathsf{hc}(s))$ is a pseudorandom generator.

Proof: Intuitively, $f(s)$ is uniformly distributed and hc(s) looks random when all we can see is $f(s)$. So $G(s)$ looks random.

More detailed proof on board.

Bootstrapping

We can "bootstrap" a pseudorandom generator with $\ell(n) = n + 1$ to one with $\ell(n)$ any polynomial in n.

The following picture explains how we do this:

$$
\begin{array}{c}
\begin{array}{|c|c|c|c|}\hline \quad s_0 \quad & & & \\ \hline \end{array} \\
\downarrow G \\
\begin{array}{|c|c|c|c|}\hline \quad s_1 \quad & \sigma_1 & & \\ \hline \end{array} \\
\downarrow G \\
\begin{array}{|c|c|c|c|}\hline \quad s_2 \quad & \sigma_2 & \sigma_1 & \\ \hline \end{array} \\
\downarrow G \\
\vdots \\
\downarrow G \\
\begin{array}{|c|c|c|c|}\hline \quad s_t \quad & \sigma_t & \cdots & \sigma_1 \\ \hline \end{array}
\end{array}
$$

Theorem: The bootstrapping construction applied to G $p(n)$ times gives a pseudorandom generator $\tilde{G}$ with expansion factor $\ell(n) = n + p(n)$.

The proof of this theorem uses a technique called a *hybrid argument*:

- We define $p(n)+1$ different probability distributions $\{H_n^i\}$ on $\{0,1\}^{\ell(n)}$.

- H_n^0 is the distribution induced by $\tilde{G}$ and $H_n^{p(n)}$ is the uniform distribution.

- Distinguishing H_n^i from H_n^{i+1} amounts to distinguishing the output of G from random.

The Hybrid Construction

We define H_n^i as follows:

- Choose $s_i \leftarrow \{0,1\}^{n+i}$ uniformly at random.

- Run $\tilde{G}$ from iteration $i+1$ and output $s_{p(n)}$.

We see that H_n^0 is the distribution induced by $\tilde{G}$ and $H_n^{p(n)}$ is the uniform distribution on $\{0,1\}^{\ell(n)}$.

Given a distinguisher D, let $\epsilon(n)$ denote its advantage in telling the output of $\tilde{G}$ apart from random:

$$\begin{aligned}\epsilon(n) &= \left|\Pr_{s \leftarrow \{0,1\}^n}[D(\tilde{G}(s) = 1] - \Pr_{r \leftarrow \{0,1\}^{\ell(n)}}[D(r) = 1]\right| \\ &= \left|\Pr_{s_{p(n)} \leftarrow H_n^0}[D(s_{p(n)}) = 1] - \Pr_{s_{p(n)} \leftarrow H_n^{p(n)}}[D(s_{p(n)}) = 1]\right|\end{aligned}$$

Given D, we define a new distinguisher D' as follows. Given an input $w \in \{0,1\}^{n+1}$, do the following:

- Choose $i \leftarrow \{1, \ldots, p(n)\}$ uniformly.

- Choose $\sigma_i \leftarrow \{0,1\}^{i-1}$ uniformly.

- Set $s_i = (w, \sigma_i)$, run $\tilde{G}$ starting at iteration $i+1$ to compute $s_{p(n)}$, and output $D(s_{p(n)})$.

D' is a Distinguisher for G:

We have

$$\Pr_{w \leftarrow \{0,1\}^{n+1}}[D'(w) = 1] = \frac{1}{p(n)} \sum_{i=1}^{p(n)} \Pr_{s_{p(n)} \leftarrow H_n^i}[D(s_{p(n)}) = 1]$$

and

$$\Pr_{s \leftarrow \{0,1\}^n}[D'(G(s)) = 1] = \frac{1}{p(n)} \sum_{i=0}^{p(n)-1} \Pr_{s_{p(n)} \leftarrow H_n^i}[D(s_{p(n)}) = 1].$$

Details on board.

Putting these together gives:

$$|\Pr[D'(G(s)) = 1] - \Pr[D'(w) = 1]| = \frac{\epsilon(n)}{p(n)}.$$

Details on board.

Since G is pseudorandom, $\epsilon(n)$ must be negligible and $\tilde{G}$ is pseudorandom.

Pseudorandom Functions

With pseudorandom *generators* in hand, we can now construct pseudorandom *functions*.

Fix a pseudorandom generator $G : \{0,1\}^n \to \{0,1\}^{2n}$ and write $G(s) = (G_0(s), G_1(s))$ where G_0 and G_1 are each n bits long.

From G, we define a keyed function $F^{(1)} : \{0,1\}^n \times \{0,1\} \to \{0,1\}^n$ by $F_k^{(1)}(b) = G_b(k)$.

$F^{(1)}$ is pseudorandom, since a distinguisher which can tell the output of $F_k^{(1)}$ from random can tell the output of G from random.

Bootstrapping

We can bootstrap this basic construction to get a keyed function $F^{(m)}$: $\{0,1\}^n \times \{0,1\}^m \to \{0,1\}^n$ by setting

$$F_k^{(m)}(x_1 \dots x_m) = G_{x_m}(F_k^{(m-1)}(x_1 \dots x_{m-1}).$$

For example, with $m = 3$ we have

$$F_k^{(3)}(011) = G_1(G_1(G_0(k))).$$

The function $F_k^{(m)}$ can be viewed as a binary tree of depth m (picture on board).

Theorem: The keyed function $F^{(n)}$ is a pseudorandom function.

Proof: The proof uses a hybrid argument. Define a distribution H_n^i on binary trees of depth n as follows:

- For nodes at level $j \le i$, the values are chosen uniformly at random from $\{0, 1\}^n$.

- For nodes at level $j > i$, look at the value k' of the node's parent and assign $G_0(k')$ if it is a left child and $G_1(k')$ if it is a right child.

Note that H_n^n corresponds to a random function and H_n^0 corresponds to $F^{(n)}$ with a uniformly-chosen key.