Class 15 Exercises

CS250/EE387, Winter 2025

In the lecture videos/notes, we saw *local list decoding,* and an algorithm to locally list-decode the Hadamard code. We saw one example (to learning Fourier-sparse functions) in the lecture videos, and today we'll see another application: *hardcore predicates from one-way-functions.*

Our goal will be to make *pseudorandom generators* from *one-way permutations.* Here are some intuitive definitions[1]:

Definition 1. *A* pseudorandom generator *(PRG) $\mathcal{G}$ takes a short seed $x \in \mathbb{F}_2^k$ and outputs a (much longer) string of bits $\mathcal{G}(x) \in \mathbb{F}_2^N$ so that it is computationally difficult to tell if a string $y \in \mathbb{F}_2^N$ was generated uniformly at random or if it was generated as the output of $\mathcal{G}$.*

Definition 2. *A* one-way permutation *(OWP) is a permutation $f : \mathbb{F}_2^k \to \mathbb{F}_2^k$ so that:*

- *Given $x \in \mathbb{F}_2^k$, it is computationally easy to compute $f(x)$*
- *Given $y \in \mathbb{F}_2^k$, it is computationally hard to find x so that $f(x) = y$, with any non-negligable probability.*

Group Work: Here are some ways we might try to make a PRG from a OWP.

1. Let $f : \mathbb{F}_2^k \to \mathbb{F}_2^k$ be a OWP. Consider the generator

$$\mathcal{G}(x) = f(x) \bullet f(f(x)) \bullet f(f(f(x))) \bullet \cdots \bullet f^{(\circ t)}(x) \bullet \cdots ,$$

   where $\bullet$ denotes concatenation and $f^{(\circ t)}$ denotes f composed with itself t times. Explain why $\mathcal{G}$ is *not* a good PRG.

2. How about this attempt?

$$\mathcal{G}(x) = [f(x)]_1 \bullet [f(f(x))]_1 \bullet [f(f(f(x)))]_1 \bullet \cdots \bullet [f^{(\circ t)}]_1 \bullet \cdots ,$$

   where $[y]_1$ denotes the first element of $y \in \mathbb{F}_2^k$. Is this a good PRG? If not, give an example that proves it (assuming one-way-permutations exist).

It turns out that there's something like the second attempt that will work, using the following definition:

Definition 3 (Hard-core bit)**.** *Let $f : \mathbb{F}_2^k \to \mathbb{F}_2^k$ be a function. We say that $b : \mathbb{F}_2^k \to \mathbb{F}_2$ is a* hard-core bit *for f if:*

- *It is computationally efficient to compute b.*

[1] For more formal versions of everything we'll do today, see `https://www.wisdom.weizmann.ac.il/~oded/prg-primer.html`

- *Given $f(x)$, it is hard to determine $b(x)$ with any probability non-negligably larger than $1/2$. Formally, for any randomized algorithm $\mathcal{A}$ that runs in time polynomial in k, and for any function $\varepsilon(k)$ that tends to zero polynomially fast in k,*

$$\Pr_{x \sim \mathbb{F}_2^k}[\mathcal{A}(f(x)) = b(x)] \leq \frac{1}{2} + \varepsilon(k).$$

(The probability is over both the choice of x and any randomness in $\mathcal{A}$).

Group Work:

3. Show that if b is a hard-core bit for a OWP f. Show that $\mathcal{G}$ given below is a PRG:

$$\mathcal{G}(x) = b(x) \bullet b(f(x)) \bullet b(f(f(x)) \bullet \cdots \bullet b(f^{(\circ t)}(x)) \bullet \cdots$$

More precisely, show that it's hard to predict $b(x)$ given $(b(f(x)), b(f(f(x))), \ldots, b(f^{(\circ t)}(x)), \ldots)$. It turns out that if you can't predict any one bit given the others, then you can't distinguish the whole string from uniformly random.

Hint: Try a proof by contradiction. What could you do if you *could* predict $b(x)$ from the other bits? It's okay to be very hand-wavey in your answer, since we haven't given a precise definition of a PRG.

It turns out that in fact, *any* one-way-permutation has a hard-core predicate!

Theorem 1 (Goldreich-Levin Theorem)**.** *Suppose that $f : \mathbb{F}_2^k \to \mathbb{F}_2^k$ is a OWP. Consider the function $g : \mathbb{F}_2^{2k} \to \mathbb{F}_2^{2k}$ given by*

$$g(x, r) = f(x) \bullet r,$$

where $\bullet$ denotes concatenation. Then $g(x, r)$ is a one-way permutation, and

$$b(x, r) = \langle x, r \rangle$$

is a hard-core bit for g.

We'll prove this theorem[2] by contradiction: suppose that b is *not* hard-core. Then there is some efficient algorithm $\mathcal{A}$ that can predict $b(x, r)$ given $f(x, r)$. We will use $\mathcal{A}$ as a black box to build an efficient algorithm $\mathcal{B}$ that inverts f. But since f was supposed to be a one-way permutation, this will be a contradiction!

Group Work:

4. Suppose that $\mathcal{A}$ is an efficient algorithm so that $\Pr_{x,r \sim \mathbb{F}_2^k}[\mathcal{A}(g(x, r)) = \langle x, r \rangle] = 1$. Give an efficient algorithm $\mathcal{B}$ so that $\Pr_{x \sim \mathbb{F}_2^k}[\mathcal{B}(f(x)) = x] = 1$.

   (Above and throughout, the probabilities are also over the randomness of $\mathcal{A}, \mathcal{B}$).

5. Suppose that $\mathcal{A}$ is an efficient algorithm so that $\Pr_{x,r \sim \mathbb{F}_2^k}[\mathcal{A}(g(x, r)) = \langle x, r \rangle] \geq 3/4 + \varepsilon$. Give an efficient algorithm $\mathcal{B}$ so that $\Pr_{x \sim \mathbb{F}_2^k}[\mathcal{B}(f(x)) = x] \geq \varepsilon'$ for some constant ε' (that can depend on ε).

6. Suppose that $\mathcal{A}$ is an efficient algorithm so that $\Pr_{x,r \sim \mathbb{F}_2^k}[\mathcal{A}(g(x, r)) = \langle x, r \rangle] \geq 1/2 + \varepsilon$. Give an efficient algorithm $\mathcal{B}$ so that $\Pr_{x \sim \mathbb{F}_2^k}[\mathcal{B}(f(x)) = x] \geq \varepsilon'$ for some ε' that may depend on ε.

7. Conclude that $\langle x, r \rangle$ is a hard-core bit for $g(x, r)$.

[2] We'll prove that $\langle x, r \rangle$ is hard-core for g. You can check for yourself that $g(x, r)$ is a OWP if $f(x)$ is.