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).

Solution

1. This is not a good PRG because we can easily check that, say, the second chunk is f([the first chunk]).
2. This might not be a good PRG... Let $g : \mathbb{F}_2^{k-1} \to \mathbb{F}_2^{k-1}$ be a one-way permutation. Let $f : \mathbb{F}_2^k \to \mathbb{F}_2^k$ be defined by

$$f(x) = x_1 \bullet g(x_2, \ldots, x_k).$$

Then f is a permutation, and it's one-way (otherwise we could invert g). But then the

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

PRG defined in the problem is just $x \mapsto x_1 \bullet x_1 \bullet x_1 \bullet \cdots$, which is not very random looking...

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.*
- *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.

Solution

Suppose that we could predict $b(x)$ from the rest of the bits. We will show that we can predict $b(x)$ from $f(x)$, which would be a contradiction. But indeed we can, since given $f(x)$, we can compute the rest of those bits $b(f(x)), b(f(f(x)), \ldots$, and use those to predict $b(x)$.

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 $\mathcal{A}, \mathcal{B}$).

   Solution

   The algorithm $\mathcal{B}$ is:

   - For $i = 1, \ldots, k$, set $x_i \leftarrow \mathcal{A}(f(x) \bullet e_i)$.
   - Return $(x_1, \ldots, x_k)$.

   This works since $\mathcal{A}(f(x) \bullet e_i) = \mathcal{A}(g(x, e_i)) = \langle x, e_i\rangle = x_i$.

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).

   Solution

   Essentially, the algorithm $\mathcal{A}$ is giving us access to a noisy Hadamard codeword. In more detail, the hypothesis tells us that

   $$\mathbb{E}_x \Pr_r[\mathcal{A}(g(x,r)) \neq \langle x, r\rangle] \leq \frac{1}{4} - \varepsilon.$$

   Say that x is "good" if

   $$\Pr_r[\mathcal{A}(g(x,r)) \neq \langle x, r\rangle] \leq \frac{1}{4} - \frac{\varepsilon}{2}.$$

   For any "good" x, then $\mathcal{A}$ gives us query access to some y so that $y_r = \langle x, r\rangle$ for at least a $3/4 + \varepsilon$ fraction of the positions r. Then, we can use $\mathcal{A}$ as a query "oracle" in the *unique* local decoding algorithm that we saw for the Hadamard code to recover x.

   In more detail, let $T = O(\log(k)/\varepsilon^2)$, and suppose that x is good. The algorithm $\mathcal{B}$ is:

   - **Input**: $f(x)$
   - For $i = 1, \ldots, k$:
     - For $t = 1, \ldots, T$:
       * Draw a random $r_i^{(t)} \sim \mathbb{F}_2^k$.
       * $x_i^{(t)} \leftarrow \mathcal{A}(f(x) \bullet r_i^{(t)}) \oplus \mathcal{A}(f(x) \bullet (r_i^{(t)} + e_i))$
     - Let x_i be the majority vote of $\{x_i^{(t)}\}_{t\in[T]}$.
   - Return $(x_1, \ldots, x_k)$

[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.

To analyze this algorithm, notice that the probability that $\mathcal{A}(f(x) \bullet r_i^{(t)}) = \left\langle x, r_i^{(t)} \right\rangle$ *and* $\mathcal{A}(f(x) \bullet r_i^{(t)} + e_i) = \left\langle x, r_i^{(t)} + e_i \right\rangle$ is at least $1/2 + \varepsilon$, by the union bound. If that happens, the $x_i^{(t)} = x_i$. Thus, the probability that the majority-vote fails is the probability that the sum of T Bernoulli-$1/2 + \varepsilon$ random variables is less than $T/2$. By a Chernoff bound, this is at most $\exp(-\Omega(\varepsilon^2 T)) = 1/\mathrm{poly}(k)$, so we can choose the constants so that we can take a union bound over all values of $i \in [k]$, and this succeeds with very high probability.

It remains to figure out the probability that x is good; we want this to be at least ε', for some $\varepsilon' > 0$. We know that

$$\mathbb{E}_x[\Pr_r[\mathcal{A}(g(x,r)) \neq \langle x, r \rangle]] \neq \frac{1}{4} - \varepsilon.$$

Thus, by Markov's inequality, the probability that x is bad is at most

$$\Pr_x[\Pr_r[\mathcal{A}(g(x,r)) \neq \langle x, r \rangle] \geq \frac{1}{4} - \varepsilon/2] \leq \frac{1/4 - \varepsilon}{1/4 - \varepsilon/2} = 1 - O(\varepsilon).$$

Above, we have used the fact that

$$\frac{1/4 - \varepsilon}{1/4 - \varepsilon/2} = \frac{1 - 4\varepsilon}{1 - 2\varepsilon} = (1 - 4\varepsilon)(1 + 2\varepsilon + 4\varepsilon^2 + \cdots) = 1 - 4\varepsilon + O(\varepsilon^2) = 1 - O(\varepsilon).$$

Thus, the probability that we succeed over *both* the randomness of x and of r is at least $\Omega(\varepsilon)$ (the probability that x is good), minus the probability that the Hadamard local list-decoder fails, which can be as small as we like; let's make it $O(1/\varepsilon^2)$ to be very conservative. Thus, we can take $\varepsilon' = \Omega(\varepsilon)$ and guarantee that we win with probability at least ε', as desired.

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.

Solution

Now we can use the local-list-decoding algorithm for Hadamard codes! Let List-Decode be that algorithm (or rather, what we get when we run that algorithm k times, once for each of the message bits). Our new algorithm is:

- **Input:** $f(x)$.
- Run List-Decode on $z \in \mathbb{F}_2^{2^k}$ with query access given by

  $$z_r = \mathcal{A}(f(x) \bullet r).$$

  Get a list $\mathcal{S} = \{x^{(1)}, \ldots, x^{(L)}\} \subseteq \mathbb{F}_2^k$ of possible messages.
- For each $i = 1, \ldots, L$, compute $f(x^{(i)})$. If it is equal to $f(x)$, return $x^{(i)}$.

Then we can do exactly the same argument as above about why $\varepsilon' = \Omega(\varepsilon)$ fraction of the x's are good.

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

Solution

As outlined above, suppose otherwise. Then we can recover $\langle x, r \rangle$ from $g(x, r)$ with probability at least $1/2 + \varepsilon$ for some non-negligable ε. But then by the above we can recover x from $f(x)$ with some non-negligable probability ε'. This contradicts the one-way-ness of f.