CS 346 Class Notes

Mark Lindberg

Mar 23, 2016

Last Time:
Theoretical results in chapter 7!
This Time:
More chapter 7?! :)
Chapter 7: One-way functions.
There are two definitions given in the text, which we will refer to as the "simple" definition and the "complex" definition.
A one-way function is a function $f : \{0,1\}^* \to \{0,1\}^*$ that is

1. Polynomial-time computable.
2. "Hard to invert". We will define this in a moment with an experiment.

Experiment: $\mathsf{Invert}_{\mathcal{A},f}(n)$. Choose a uniform random n-bit random string x. We will compute $y = f(x)$. Then we give $1^n, y$ to $\mathcal{A}$, $\mathcal{A}$ runs in polynomial time in the size of $\max(n, |y|)$, and $\mathcal{A}$ computes x', and succeeds if $f(x') = y$.

We say that f is hard to invert if $\forall$ PPT $\mathcal{A}$, $\Pr[\mathsf{Invert}_{\mathcal{A},f}(n) = 1] \leq \mathtt{negl}(n)$.

A one-way permutation is a one-way function that is length-preserving and one-to-one. (bijective, is a permutation.)

Have we seen any one-way functions? What are some likely one-way functions?

Integer factoring. If we are given a number which is the product of 2 large primes $n = p \cdot q$, it is not known how to factor n into p and q in polynomial time in the size of p and q. The associated one-way function, f, maps $2n$-bits to $2n$-bits.

Other candidates:

- One in the text based on the subset-sum problem, which is known to be np-complete. We want to know if there is a subset of a given set of integers which add up to a specified target. It takes a set of elements, and a set of indices, and maps them to the set of elements, and the sum of the specified elements. It's easy to compute, but finding the subset that gives the specified sum is hard.
- Something based on the discrete log problem. Mod arithmetic!

PRGs, PRFs, PRPs, Strong PRPs. $\rightarrow$ Existence of 1-way functions established the existence of all structures studied in chapters 3 and 4. (Because we have the stuff listed here.)

Notice that chapter 5 has been left out. Poor little chapter 5. One-way functions are known to be necessary for collision resistant hash functions, however, they are NOT KNOWN to be sufficient.

New concept: Hard-core predicate. Wat.

We are trying to get PRGs from one-way functions. The key step in this is the Goldreich-Levin Theorem.

First, we will show the expansion factor $\ell(n) = n + 1$. The book uses one-way permutations. The output of the one-way permutation, concatenated with hard-core predicate value. But what is that?

Hard-core predicate: A function $hc : \{0,1\}^* \rightarrow \{0,1\}$ (It goes to a single bit!) is a hard-core predicate for $f : \{0,1\}^* \rightarrow \{0,1\}^*$, if hc is polynomial-time computable and $\forall$ PPT adversaries $\mathcal{A}$, $\Pr[\mathcal{A}(1^n, f(x)) = hc(x)] \leq \frac{1}{2} + \mathtt{negl}(n)$, where the probability is over the random n-bit value x.

Goldreich-Levin Theorem: Let f be a 1-way function (respectively, permutation). Then g, defined by $g(x,r) = (f(x), r)$, is a 1-way function (respectively, permutation) and $hc(x,r) \neq \bigoplus_{\text{bit } i \text{ of } r=1} x_i$ is a hard-core predicate for g.

Then we want PRGs with arbitrary polynomial expansion factors. In the text, we use 1-way permutations, just for simplification.

Let's see how we get a second bit, $n \rightarrow n+2$. We take $f(f(x))||hc(f(x))||hc(x)$. Whee.

For a 3rd bit, $f(f(f(x)))||hc(f(f(x)))||hc(f(x))||hc(x)$. Etc.

PRFs from PRGs.

Use a PRG G with expansion factor $2n$.

We take our n-bit key, and construct $F_k(x)$, with $|k| = |x| = n$-bits.

Let $G(x) = G_0(x)||G_1(x)$, basically splitting into 2x n-bit outputs.

For each bit of the key, take either G_0 or G_1 of the output of the previous step, where the first step is to take the key.. So if $n = 3$, and $x = 011$, $F_k(x) = G_1(G_1(G_0(k)))$ (Should that be of x? I think so, but he wrote k.)

How to get a PRP? Feistel Network.

Yeah, the picture gets weird, again. Check the textbook. 3 rounds are enough for a PRP, 4 for a strong PRP. Anything less, you fail!