Reading Guide 1: Pseudorandom Functions

September 3, 2004

The course reading this week comes from a set of lecture notes by Mihir Bellare and Phil Rogaway, two of the leaders in the world of reductionist security. We're going to be looking at sections 3.1 through 3.4, you don't need to read the rest of the chapter.

Begin by reading section 3.1 which introduces the idea of a *function family.* The notation that's used here to indicate that a object x is chosen randomly from a set X, $x \overset{\$}{\leftarrow} X$, is non-standard: more often you'll see it denoted $x \overset{R}{\leftarrow} X$. A *bijection* is a function where each output in the range is produced by exactly one input in the domain. You can think of it as a complete pairing between elements of the domain and elements of the range. Note that in a bijection, the size of the domain must be the same as the size of the range, otherwise there's some element that's not paired up. A *permutation* is a bijection where the domain and the range are the same set. It's called a permutation because it can be thought of as a way of rearranging the domain.

Example 3.1 refers to two *block ciphers* called DES and AES. A block cipher is a construction that's designed to be a pseudorandom permutation. DES uses 56-bit keys and works on 64-bit inputs, thus it's keyspace is $\mathcal{K} = \{0,1\}^{56}$ and it's domain is $\{0,1\}^{64}$. AES uses 128-bit keys and works on 128-bit inputs.

In section 3.2 random functions and permutations are defined formally. We did the first four parts of example 3.3 in class; make sure that you feel comfortable with them. Part 5 says that the probability of getting a specific value for the first l bits of output of a pseudorandom function is just one out the total number of l-bit strings. In section 3.2.2, make sure you understand the new algorithmic definition. Most importantly, here the random choices are *not* independent (we can't repeat an answer we've already given) which ends up making things more complicated for us.

Work through example 3.5. Instead of f, π is often used to indicate a function that's a permutation because permutation and pi both start with p. Mathematicians can be so clever. In the computation of part four at the top of page 7, first $\pi(X_1)$ was exclusive-ored onto both sides of $\pi(X_1) \oplus \pi(X_2) = Y$ yeilding $\pi(X_2) = \pi(X_1) \oplus Y$ since $\pi(X_1) \oplus \pi(X_1) = 0$. Next, $\Pr[\,\pi(X_2) = \pi(X_1) \oplus Y\,]$ was broken down into cases where $\pi(X_1)$ takes on each of the possible values. The quantity $\Pr[\,\pi(X_2) = Y_1 \oplus Y \mid \pi(X_1) = Y_1\,] \cdot \Pr[\,\pi(X_1) = Y_1\,]$ is the probability that $\pi(X_2)$ will be some specific value, given that $\pi(X_1)$ is already Y_1, times the probability that $\pi(X_1)$ actually is Y_1. Note that we don't have any problems with $Y_1 \oplus Y$ being the same as Y_1 since Y is non-zero in this case.

Our lecture followed section 3.3 fairly closely, but be sure that you understand definition 3.6 which formalizes the notion of prf-advantage. The experiments in such definitions are also often referred to as *games.* Since we didn't read chapter 1, just imagine whatever model of computation you covered in your undergraduate algorithms or theory of computation class that had some notion of RAM.

Section 3.4 introduces pseudorandom permutations. Here there are two different definitions: PRP under chosen-plaintext attacks (CPA) and PRP under chosen-ciphertext attacks. These names come from the world of encryption: the input to an encryption function is called the plaintext and the output is called the ciphertext. In a chosen-plaintext attack, the attacker can only query the permutation on the inputs (the plaintexts), while in a chosen-ciphertext attack the attacker can also query the permutation's inverse, g^{-1}, on the outputs (the ciphertexts).

Note that in the PRP under CPA definition we don't require that F be a permutation: it just won't turn out to be a very good PRP since if the adversary ever sees the same output it knows right away that F isn't even a permutation, let alone a pseudorandom one. In class next time we'll talk about the opposite problem of passing off a PRP as a PRF. In the CCA definition we require that F be a permutation since otherwise we have no way of constructing the inverse function.

Section 3.4.3 just shows that if a PRP is secure under chosen-ciphertext attacks, it's also secure under chosen-plaintext attacks. Basically, if there's an adversary that can break a PRP without access to the inverse oracle, then giving it the inverse oracle won't make it any harder for it to break the PRP since it can just ignore the extra oracle.