ECS 227: Modern Cryptography
Lecture 4

Pseudorandom Generators and Pseudorandom Functions

Phillip Rogaway
23 January 1996

5 Pseudorandom Generators

Class began by reviewing the definition of a block cipher and the success of an adversary in attacking one. We gave some plausible numeric conjectures and absurdities, based on DES. We then went on to define pseudorandom generators—a primitive we talked described —informally— in Lecture 1.

5.1 Definitions

Intuitively, a pseudorandom generator takes a short, truly-random string and stretches it into a long, pseudorandom one. The pseudorandom string should look "random enough" to use it in place of a truly random string. As with block ciphers, let's split up the syntax and the notion of security.

Definition 5.1.1 *A* pseudorandom generator *(PRG) with seed length ℓ and output length L is a (deterministic) function $g : \{0,1\}^\ell \to \{0,1\}^L$, where $\ell < L$.*

The argument $s \in \{0,1\}^\ell$ to g is often called the *seed.*

The "game" which defines the worth of a PRG compares the behavior of an adversary D in distinguishing truly-random L-bit strings from pseudorandomly generated ones.

Definition 5.1.2 *Let $g : \{0,1\}^\ell \to \{0,1\}^L$ be a PRG. A PRG adversary, D, has a distinguished oracle tape, from which D reads the bits left-to-right. We* *define the* advantage *of D in distinguishing the output of g from a truly random string by*

$$\mathrm{Adv}_D^{g,\$} \quad = \quad \Pr[s \leftarrow \{0,1\}^\ell;\ y \leftarrow g(s) :\ D^y = 1] - \Pr[y \leftarrow \{0,1\}^L :\ D^y = 1]$$

As before, "good" schemes are those for which no adversary which uses a "reasonable" amount of computational resources manages to achieve a "substantial" advantage. What are the resources of relevance? The most important ones are the amount of time the adversary runs in, t, and the number of bits, μ, which the adversary inspects of her oracle tape.

Definition 5.1.3 *Let $g : \{0,1\}^\ell \to \{0,1\}^L$ be a PRG, and let D be a PRG adversary. Then D is said to (t, μ, ϵ)-*break *the PRG g if D runs in at most t computational steps and inspects at most μ bits of its oracle tape and $\mathrm{Adv}_D^{g,\$} \geq \epsilon$.*

The earlier remark about including the size of D along with its running time applies once again. We will return to this issue shortly, when we discuss uniform *vs.* non-uniform adversaries.

5.2 Construction from a block cipher

How do we make a PRG? From what primitive should do we start? In practice, one of the easiest and most common things is to start with is a block cipher. Let $F : \{0,1\}^\kappa \times \{0,1\}^\ell \to \{0,1\}^\ell$ be one. In class we described two constructions of a PRG from a block cipher. Let $\langle n \rangle_b$ be the binary representation of n, encoded in the usual way into b-bits.

1. **The PRP(CFB) Generator.** This generator $g : \{0,1\}^\kappa \to \{0,1\}^L$ takes a κ-bit seed, s, and produces, for any L, the L-bit prefix of

$$F_s(\langle 0 \rangle_\ell) \ \| \ F_s^{(2)}(\langle 0 \rangle_\ell) \ \| \ F_s^{(3)}(\langle 0 \rangle_\ell) \ \| \ F_s^{(4)}(\langle 0 \rangle_\ell) \ \| \ \cdots .$$

   Here $F_s^{(i)}$ denotes the i-th iterate of F_s—that is, $F_s^{(0)}$ is the identity map on ℓ-bit strings and, for all $i \geq 1$, $F_s^{(i)} = F_s^{(i-1)} \circ F_s$.

2. **The PRP(Counter) Generator.** This generator $g : \{0,1\}^\kappa \to \{0,1\}^{\ell 2^\ell}$ takes a κ-bit seed, s, and produces the following long string:

$$F_s(\langle 0 \rangle_\ell) \ \| \ F_s(\langle 1 \rangle_\ell) \ \| \ F_s(\langle 2 \rangle_\ell) \ \| \ \cdots \ \| \ F_s(\langle 2^\ell - 1 \rangle_\ell) .$$

The style of security analysis which we give allows one to make exact comparison between such schemes. We shall restrict ourselves to looking at the second of these two methods. In the following section, the variable $g = g_F$ refers to the PRG given by the PRP(Counter) construction.

5.3 Security of the Counter/PRG generator

Suppose you're the adversary D and you're faced with the problem of distinguishing $g(s)$-values, for a random s, from a truly random sequence of bits. What might be your strategy?

One strategy is to inspect the first m ℓ-bit blocks of data, for some parameter m, and look to see if there's any repeated block. **Case 1.** Suppose that your input was pseudorandomly generated. Then, by the permutivity of F, there *can't* be any repeated block. **Case 2.** Alternatively, suppose the input was randomly generated. Then there *might* be some repeated block. Thus a reasonable strategy for D is the following

> Partition the input into m ℓ-bit blocks, $B_1, B_2, \ldots, B_m$. If these blocks are all distinct, output 1 (that is, guess "pseudorandom"). Otherwise (some $B_i = B_j$, for $i \neq j$) output 0 (that is, guess "truly random").

Of course D's "guess" when she answers 0 (truly random) is always correct, while D's guess when she answers 1 (pseudorandom) may well be wrong. It seems clear that D's guess is more and more accurate as m increases. We shall quantify this in the next lecture.

The above attack can be used to show an upper bound on security: we know that security is no better than what the specified adversary achieves. We use a reduction to show a corresponding lower bound on security.

Proposition 5.3.1 *Let $F : \{0,1\}^\kappa \times \{0,1\}^\ell \to \{0,1\}^\ell$ be a Finite PRPs, and let $g : \{0,1\}^\kappa \to \{0,1\}^{\ell 2^\ell}$ be the PRG constructed from F using the PRP(Counter) construction. Suppose there is an adversary D which (t, μ, ϵ)-breaks g. Then there is an adversary D' which (t', μ', ϵ')-distinguishes F from a random permutation, where*

$$\begin{aligned} t' &= t + c(\mu + \ell) \\ q' &= \left\lceil \frac{\mu}{\ell} \right\rceil \\ \epsilon' &= \epsilon + \left\lceil \frac{\mu}{\ell} \right\rceil^2 2^{-\ell-1} \end{aligned}$$

Here c is an absolute constants which depends only on details of the model of computation, and adversary D' is a fixed algorithm, apart from using D as a black box.

We will prove this proposition in two steps. First we'll show an analogous (but stronger) proposition in which we use a finite pseudorandom *function* in place of a Finite PRP. Then we'll show that a finite PRP *is* a finite PRF with security that is not *too* much off. To proceed, then, we need to define finite pseudorandom functions, and an adversary's success in attacking one.

5.4 Finite Pseudorandom Functions

Syntactically, a finite pseudorandom function is the natural relaxation of a finite PRP.

Definition 5.4.1 *A* Finite Pseudorandom Function *(Finite PRF), with* key length *κ,* input length *ℓ, and* output length *L is a map*

$$F : \{0,1\}^\kappa \times \{0,1\}^\ell \to \{0,1\}^L \ .$$

We'll denote $F(a,x)$ by $F_a(x)$.

Now we want to *quantify* the "goodness" of a Finite PRF. The intuition is that a "good" Finite PRF should "look like" a random function, in the following sense. Suppose we give an adversary D one of two types of boxes, as described below. Which type of box we give her is determined by flipping a fair coin.

- We might give the adversary a Finite PRF box. It is defined by choosing a random key $a \in \{0,1\}^\kappa$ and giving the adversary a box which computes $F_a(\cdot) : \{0,1\}^\ell \to \{0,1\}^L$.
- Alternatively, we might give the adversary a random function box. This is determined by selecting, at random, one of the 2^{L2^ℓ} possible functions from l-bits to L-bits.

The adversary's job is to guess which type of box she has. We want to quantify how well an adversary does in this game. We do this as follows:

Let $\mathcal{R}_{\ell\to L}$ be the set of functions from ℓ-bits to L-bits.

Definition 5.4.2 *Let D be a Finite PRF adversary — an adversary with an oracle, where the adversary outputs a bit. Let $F : \{0,1\}^\kappa \times \{0,1\}^\ell \to \{0,1\}^L$ be a Finite PRF. We define D's* advantage *at distinguishing F from a random function by*

$$\mathrm{Adv}_D^{F,\rho} = \Pr[a \leftarrow \{0,1\}^\kappa :\ D^{F_a(\cdot)} = 1] - \Pr[\rho \leftarrow \mathcal{R}_{\ell\to L} :\ D^{\rho(\cdot)} = 1]$$

As before, this definition doesn't exactly mimic the game we described: once again, a closer, but somewhat more complicated-looking translation, can easily be defined and shown to be equivalent to what's above.

Lecture ended here, before I got to cover the (simple) proofs for what we've claimed. We'll continue there on Thursday.