Computer Science 2426F — Fall, 2018
St. George Campus — University of Toronto

Notes #1 (for Lectures 2 - 3)

The material from the early part of this course relates to Chapters 1 and 3 of *Goldreich*.

Pseudo-Random Number Generators

We will begin the course proper by discussing "pseudo-random number generators". Recall the motivation from one-time pads. We have a n-bit random key K but we wish we had a long random key. So we "stretch" K to $K' = G(K)$ using a pseudo-random number generator G to obtain K' of length $l(n) > n$. To an efficient adversary, it should "look" as if K' were randomly chosen.

Informally, a pseudo-random number generator is an efficiently computable function that on an n-bit input, outputs a longer string, and such that the probability distribution induced on the longer strings is indistinguishable from the truly random distribution, from the point of view of any efficient algorithm. That is, the induced distribution passes every (efficient) statistical test.

Definitions: (See Definition 3.3.1 of *Goldreich*.)
A *number generator* is a polynomial time computable function $G : \{0,1\}^* \to \{0,1\}^*$, such that $|G(s)| = l(|s|) > |s|$ for some function l and every string s. For convenience we also insist that $|s|$ is determined by $l(|s|)$. (That is, l is one-one. Of course, G need not be one-one.) We also assume for convenience that l is *monotone*: $n < m \Rightarrow l(n) < l(m)$.

The number generator G is *pseudo-random* if the following holds for every D:
Let D (for *distinguisher*) be a probabilistic, polynomial time algorithm with inputs of the form $\alpha \in \{0,1\}^*$; D has a 1-bit output indicating whether or not the input is accepted (say output 1 means "yes" and output 0 means "no").
For each $n \in \mathbb{N}$, define
$p_D(n)$ = the probability that if s is randomly chosen from $\{0,1\}^n$ and D is run on $G(s)$, then D accepts;
$r_D(n)$ = the probability that if α is randomly chosen from $\{0,1\}^{l(n)}$, and D is run on α, then D accepts.
THEN for every c and sufficiently large n, $|p_D(n) - r_D(n)| \leq \frac{1}{n^c}$.
(We may omit the subscript D if it is understood.)

Note that D, given α of length $l(n)$, is able (if he wants) to determine n. Since l is one-one, all he has to do is compute G on strings length $0, 1, 2, \ldots$ until he computes a string whose length is the same as that of α.

The above definition of pseudo-random is actually what we call "pseudo-random against uniform adversaries". An alternative definition would be as above, except that D is a polynomial-size family $\{D_1, D_2, \ldots\}$ of (deterministic) circuits, D_n having $l(n)$ input bits and one output bit; we call this definition "pseudo-random against nonuniform adversaries". It is easy to see that this latter definition would not change if we permitted the circuits to be probabilistic. This is because for each such circuit there would be a way to fix its random input to an optimum value. Such a value might be hard to find, but it would exist (Exercise: Prove this). It is therefore easy to see that "pseudo-random against nonuniform adversaries" implies "pseudo-random against uniform adversaries" (Exercise: Why is this?), but the converse is believed not to be true. Every definition of security in this course will have a "uniform adversary" version and a "nonuniform adversary" version; even if we state just one version, the other version will be self-evident. All the theorems we will state in this course of the form "if this thingy is secure then that thingy is secure" will be true, assuming we *consistently* use one of the two versions. In practice, it is usually easier to define security and to prove theorems in the *nonuniform adversary* setting.

The intuition is that a generator being pseudo-random means that an efficient algorithm cannot tell the difference between a randomly generated string and a pseudo-randomly generated string. Here is another way of expressing this concept (in the *nonuniform adversary* setting).
We say G is *alternatively-pseudo-random* if the following holds for every C:

Let $C = \{C_n\}$ be a polynomial size family of circuits where C_n has $l(n)$ input bits and one output bit. Consider the following experiment:
A random bit $b \in \{0, 1\}$ is chosen;
if $b = 0$, then C_n is run on a randomly chosen string $\alpha \in \{0, 1\}^{l(n)}$;
if $b = 1$, then a random string $s \in \{0, 1\}^n$ is chosen and C_n is run on $G(s)$;
let $q_C(n)$ be the probability that C_n outputs b.
THEN $q_C(n) \leq \frac{1}{2} + \frac{1}{n^e}$ for each e and sufficiently large n.
Exercise: Prove that G is pseudo-random $\iff$ G is alternatively-pseudo-random.

Pseudo-Random Against Multiple Sampling

Essentially, a generator being pseudo-random means that an efficient algorithm cannot tell the difference between a single randomly generated string and a single pseudo-randomly generated string. But what if the distinguisher was given *two* sample strings, or even a polynomial number of sample strings; could he then distinguish between the two distributions? The answer turns out to be no! For convenience, we will state and prove this theorem for exactly n samples, but the same proof works for any (fixed) polynomial number of samples.

Definition: (See Definition 3.2.4 of *Goldreich*) (*uniform adversary* setting)
Let G be a number generator. We say G is *pseudo-random against multiple sampling* if the following holds for every D:

Let D be a probabilistic, polynomial time, algorithm with inputs of the form $\alpha \in \{0, 1\}^{n \cdot l(n)}$ for some n; D has a 1-bit output indicating whether or not the input is accepted.
For each n, define
$p_D(n) =$ the probability that if $s_1, s_2, \cdots, s_n$ are randomly (and independently) chosen from $\{0, 1\}^n$ and D is run on $[G(s_1)G(s_2)\cdots G(s_n)]$, then D accepts;
$r_D(n) =$ the probability that if α is randomly chosen from $\{0, 1\}^{n \cdot l(n)}$ and D is run on α, then D accepts.
THEN for every c and sufficiently large n, $|p_D(n) - r_D(n)| \leq \frac{1}{n^c}$.

The definition for the *nonuniform adversary* setting is as follows.

Definition: Let G be a number generator. We say G is *pseudo-random against multiple sampling* if the following holds for every D:

Let $D = \{D_n\}$ be a polynomial-size family of circuits where D_n has $n \cdot l(n)$ input bits and one output bit indicating whether or not the input is accepted.

For each n, define
$p_D(n)$ = the probability that if $s_1, s_2, \cdots, s_n$ are randomly (and independently) chosen from $\{0,1\}^n$, then D_n accepts $[G(s_1)G(s_2)\cdots G(s_n)]$;
$r_D(n)$ = the probability that if α is randomly chosen from $\{0,1\}^{n\cdot l(n)}$, then D_n accepts α.
THEN for every c and sufficiently large n, $|p_D(n) - r_D(n)| \leq \frac{1}{n^c}$.

The following theorem (as well as the others we shall give) holds as long as one either consistently uses the *uniform adversary* setting or the *nonuniform adversary* setting.

Theorem: Let G be a number generator.
G is pseudo-random $\iff$ G is pseudo-random against multiple sampling.

Proof of $\Longleftarrow$: Exercise.

Proof of $\Longrightarrow$: We will prove this in the "nonuniform adversary" setting. Assume that G is *not* pseudo-random against multiple sampling. We will show that G is not pseudo-random.

Let $\{D_n\}$ be an adversary for distinguishing G using multiple sampling; D_n has $n \cdot l(n)$ input bits and one output bit. For each n define $p_D(n)$ and $r_D(n)$ as in the definition of "pseudo-random against multiple sampling". Fix n and say (without loss of generality) that $p_D(n) - r_D(n) > \frac{1}{n^c}$. We will describe a circuit D'_n for distinguishing G.

We first describe a sequence of experiments that are "hybrids" between the experiment that gives rise to $p_D(n)$ and the experiment that gives rise to $r_D(n)$. For $0 \leq i \leq n$ let q_i be the probability, IF $s_1, s_2, \cdots, s_i$ are i randomly chosen strings from $\{0,1\}^n$, and $t_{i+1}, t_{i+2}, \cdots, t_n$ are $n-i$ randomly chosen strings from $\{0,1\}^{l(n)}$, and D_n is given as input $[G(s_1)G(s_2)\cdots G(s_i)t_{i+1}t_{i+2}\cdots t_n]$, THEN D_n accepts. Clearly $q_n = p_D(n)$ and $q_0 = r_D(n)$. So there exists an i, $0 \leq i < n$, such that $q_{i+1} - q_i > \frac{1}{n^{c+1}}$; fix such an i.

We now describe a *probabilistic* circuit D'_n for distinguishing G. The input will be an $l(n)$ bit string α. D'_n will choose i strings $s_1, s_2, \cdots, s_i$ randomly from $\{0,1\}^n$ and $n-(i+1)$ strings $t_{i+2}, t_{i+3}, \cdots, t_n$ randomly from $\{0,1\}^{l(n)}$, and run D_n on $[G(s_1)G(s_2)\cdots G(s_i)\,\alpha\, t_{i+2}t_{i+3}\cdots t_n]$.

Let $p_{D'}(n)$ be the probability D'_n accepts $\alpha = G(s)$ for s randomly chosen from $\{0,1\}^n$; clearly $p_{D'}(n) = q_{i+1}$. Let $r_{D'}(n)$ be the probability D'_n accepts a random string α from $\{0,1\}^{l(n)}$; clearly $r_{D'}(n) = q_i$. (Note that these probabilities are over the random choices of D'_n, as well as over the random choices of s or α.) So $p_{D'}(n) - r_{D'}(n) > \frac{1}{n^{c+1}}$. As discussed earlier, by appropriately fixing the random bits of D'_n, we can make a normal, deterministic circuit D''_n that does at least as well as D'_n as an adversary against G. Note that the size of D''_n is polynomial in the size of D_n (and n), and hence that $\{D''_n\}$ is a polynomial-size family.

So $\{D''_n\}$ is an adversary that breaks the pseudo-randomness of G. $\square$

How would the proof of the $\Longrightarrow$ part of this theorem go in the "uniform adversary" setting? We are given a probabilistic polynomial time adversary D for breaking G using multiple sampling, and we wish to find a probabilistic polynomial time adversary D' for breaking G (on a single sample). D', on input α, will behave as D'_n as described in the proof above. (Note that because l is one-one and monotonic, n is determined – and easy to find – from $|\alpha|$.) The tricky part is that it is not clear what value of i to choose. It turns out that things work fine if we merely choose i *randomly* in the range $0 \leq i < n$. (Exercise: Why is this?)

There will be other theorems, however, where a careful choice of some parameter must be made, and just making a random choice will not do. Choices will have to be made by an adversary we construct that will depend on the values of certain probabilities. Fortunately, however, it will usually not be necessary to know these probabilities exactly, but only approximately. Usually we will be able to approximate these probabilities by performing an experiment sufficiently often. For example, say that we want to approximate the probability q_i as defined (with respect to a particular value of n) in the above proof. It is sufficient to approximate it to within $\epsilon = \frac{1}{n^{c+2}}$. q_i is the probability D_n accepts when run according to a certain experiment. We can run this experiment many times, and take the fraction q of acceptances to be a good approximation to q_i. More exactly, for some constant d (and it is sufficient to let $d = 4$), if we repeat the experiment $d(1/\epsilon)^2 m$ times, then q will be within ϵ of q_i with probability $> 1 - \frac{1}{2^m}$; this is a consequence of well known Chernoff bounds.

So say we given the uniform adversary D as above that breaks the pseudo-randomness of G with multiple sampling where, say, $p_D(n) - r_D(n) > \frac{1}{n^c}$ for infinitely many n. Define the uniform adversary D' for breaking the pseudo-randomness of G as follows, on input α (where $p_D(n) - r_D(n) > \frac{1}{n^c}$): By repeatedly running experiments, D' computes $q'_0, q'_1, \ldots, q'_n$ such that for each i, the probability is $< \frac{1}{2^n}$ that $|q'_i - q_i| > \frac{1}{n^{c+2}}$. So with probability $> 1 - \frac{n+1}{2^n}$, *every* q'_i is within $\frac{1}{n^{c+2}}$ of q_i. D' chooses the (first) value of i that maximizes $q'_{i+1} - q'_i$, and then proceeds as in the probabilistic nonuniform setting. Note that the i chosen by D' is a random variable.

Say that the calculation of $q'_0, q'_1, \ldots, q'_n$ comes from choosing a random string U of bits and let the event E be the set of such strings that cause *every* q'_j to be within $\frac{1}{n^{c+2}}$ of q_j;
this event occurs with probability $> 1 - \frac{n+1}{2^n}$.
For each $u \in E$ we let i_u be the (first) value of i that maximizes $q'_{i+1} - q'_i$, and so $q'_{i_u+1} - q'_{i_u} > (1/n)(1/n^c - 2/n^{c+2}) > 1/n^{c+1} - 1/n^{c+2}$, and so $q_{i_u+1} - q_{i_u} > 1/n^{c+1} - 3/n^{c+2}$. Given that $U = u \in E$, the probability that D' accepts a pseudo-randomly generated α is q_{i_u+1}, and the probability that D' accepts a randomly generated α is q_{i_u}. The probability that D' accepts a pseudo-randomly generated α given that E occurs is the average over $u \in E$ of q_{i_u+1}, and the probability that D' accepts a randomly generated α given that E occurs is the average over $u \in E$ of q_{i_u}. so given that E occurs, the difference between these two probabilities is $> 1/n^{c+1} - 3/n^{c+2}$. So (exercise!) $p_{D'} - r_{D'} > 1/n^{c+1} - 3/n^{c+2} - 2(n+1)/2^n > 1/n^{c+2}$.

Unpredictability

Another notion of pseudo-randomness that often appears in the informal literature is that of "unpredictability". Informally, we say that G is *unpredictable* (from the left) if given a proper prefix of $G(s)$, one cannot guess the next bit with probability significantly above 1/2. It turns out that this condition is *equivalent* to pseudo-randomness. We will define "unpredictability" in the *nonuniform adversary* setting, since it is a little awkward to define in the *uniform adversary* setting.

Definition: (See Definition 3.3.6 of *Goldreich.*)
Let G be a number generator. We say G is *unpredictable* (sometimes called unpredictable from the left) if the following holds for every A:

Let $A = \{(A_n, i_n)\}$ where $1 \leq i_n \leq l(n)$ and $\{A_n\}$ is a polynomial-size family of circuits, where A_n has $i_n - 1$ input bits and one output bit.
For each n: define, letting $i = i_n$ (for notational convenience),
$pred_A(n)$ = the probability that if s is randomly chosen from $\{0,1\}^n$ and $G(s) = [b_1, b_2, \cdots, b_{l(n)}]$ and A_n is given $[b_1, b_2, \cdots, b_{i-1}]$, then A_n outputs b_i.
THEN for every c and sufficiently large n, $pred_A(n) \leq \frac{1}{2} + \frac{1}{n^c}$.

Theorem: (See Theorem 3.3.7 of *Goldreich.*)
Let G be a number generator.
G is pseudo-random $\iff$ G is unpredictable.

Proof of $\Longrightarrow$: The idea is that if G is not unpredictable, then G is predictable, which gives us a statistical test that breaks the pseudo-randomness of G.

Let $\{(A_n, i_n)\}$ be an adversary that breaks the unpredictability of G, and let $pred(n) = pred_A(n)$ be defined as above, so that for infinitely many n, $pred(n) > \frac{1}{2} + \frac{1}{n^c}$. Fix n, and say $pred(n) = \frac{1}{2} + \epsilon(n)$. For notational convenience, we will use i below instead of i_n.

Now define a distinguishing circuit D_n for G as follows: On input $a_1, a_2, \cdots, a_{l(n)}$, D_n computes $A_n(a_1, a_2, \cdots, a_{i-1})$, and accepts if this equals a_i.
Then $p(n) =$ the probability that D_n accepts $G(s)$ for random $s = pred(n)$,
and $r(n) =$ the probability that D_n accepts a random string $= 1/2$.
So $p(n) - r(n) = \epsilon(n)$.

The size of D_n is polynomial in the size of A_n (and n), so $\{D_n\}$ is a polynomial-size family; also, since $\epsilon(n) > \frac{1}{n^c}$ for infinitely many n, then $|p(n) - r(n)| > \frac{1}{n^c}$ for infinitely many n. So $\{D_n\}$ is an adversary that breaks the pseudo-randomness of G.

Proof of $\Longleftarrow$: Let $\{D_n\}$ be an adversary for distinguishing G; D_n has $l(n)$ input bits and one output bit. For each n define $p(n)$ and $r(n)$ as in the definition of pseudo-random, and let $|p(n) - r(n)| = \epsilon(n)$; assume that $\epsilon(n) > \frac{1}{n^c}$ for some c and infinitely many n. Fix n and say (without loss of generality) that $p(n) - r(n) = \epsilon(n) > 0$. We will describe an adversary (A_n, i_n) for predicting G.

We first describe a sequence of experiments that are "hybrids" between the experiment that gives rise to $p(n)$ and the experiment that gives rise to $r(n)$. For $0 \le i \le l(n)$ let p_i be the probability, IF s is randomly chosen from $\{0,1\}^n$ and $G(s) = [b_1, b_2, \cdots, b_{l(n)}]$, and $a_{i+1}, a_{i+2}, \cdots, a_{l(n)}$ are $l(n) - i$ randomly chosen bits, and D_n is given as input $[b_1, b_2, \cdots, b_i, a_{i+1}, a_{i+2}, \cdots, a_{l(n)}]$, THEN D_n accepts. Clearly $p_{l(n)} = p(n)$ and $p_0 = r(n)$. So $p_{l(n)} - p_0 = \epsilon(n)$. So there exists an i, $0 < i \le l(n)$, such that $p_i - p_{i-1} \ge \epsilon(n)/l(n)$; let i_n be such an i. For notational convenience, we will use i below instead of i_n.

We now describe a *probabilistic* circuit A_n for predicting bit i of the output of G. The input to A_n will be an $i-1$ bit string α. A_n will choose a random bit a and $l(n) - i$ random bits $a_{i+1}, a_{i+2}, \ldots, a_{l(n)}$ and run D_n on $[\alpha, a, a_{i+1}, a_{i+2}, \ldots, a_{l(n)}]$; if D_n accepts then A_n outputs bit a, otherwise A_n outputs bit $\overline{a} = 1 - a$.

The experiment we are concerned with is as follows.
$s \leftarrow$ a random string from $\{0,1\}^n$; say that $G(s) = [b_1, b_2, \ldots, b_{l(n)}]$;
$a \leftarrow$ a random bit;
$a_{i+1}, a_{i+2}, \cdots, a_{l(n)} \leftarrow$ random bits;
D_n is run on $[b_1, b_2, \ldots, b_{i-1}, a, a_{i+1}, a_{i+2}, \ldots, a_{l(n)}]$.

We wish to compute $pred(n) = pred_A(n) =$ the probability A_n outputs b_i, the i-th bit of $G(s)$. This is the probability that either $a = b_i$ and D_n accepts, or $a = \overline{b_i}$ and D_n rejects. Since a is equally likely to be b_i as $\overline{b_i}$, we have

$$pred(n) =$$

$$\mathbf{prob}(a = b_i \text{ and } D_n \text{ accepts}) + \mathbf{prob}(a = \overline{b_i} \text{ and } D_n \text{ rejects}) =$$

$$\mathbf{prob}(a = b_i) \cdot \mathbf{prob}(D_n \text{ accepts} \,|\, a = b_i) + \mathbf{prob}(a = \overline{b_i}) \cdot \mathbf{prob}(D_n \text{ rejects} \,|\, a = \overline{b_i}) =$$

$$\frac{1}{2} \cdot \mathbf{prob}(D_n \text{ accepts} \,|\, a = b_i) + \frac{1}{2} \cdot \mathbf{prob}(D_n \text{ rejects} \,|\, a = \overline{b_i})$$

Clearly

$$\mathbf{prob}(D_n \text{ accepts} \,|\, a = b_i) \;=\; p_i$$

and

$$\mathbf{prob}(D_n \text{ rejects} \,|\, a = b_i) \;=\; 1 - p_i$$

and

$$\mathbf{prob}(D_n \text{ accepts}) = p_{i-1}$$

Since,

$$\frac{1}{2}\mathbf{prob}(D_n \text{ rejects} \,|\, a = \overline{b_i}) + \frac{1}{2}\mathbf{prob}(D_n \text{ rejects} \,|\, a = b_i) = \mathbf{prob}(D_n \text{ rejects}) = 1 - p_{i-1}$$

we have

$$\frac{1}{2}\mathbf{prob}(D_n \text{ rejects} \,|\, a = \overline{b_i}) \;=\; [(1 - p_{i-1}) - \frac{1}{2}(1 - p_i)]$$

So

$$pred(n) = \frac{1}{2}p_i + [(1 - p_{i-1}) - \frac{1}{2}(1 - p_i)] = \frac{1}{2} + (p_i - p_{i-1}) \geq \frac{1}{2} + \frac{\epsilon(n)}{l(n)}$$

Note that the size of A_n is polynomial in the size of D_n (and n), and A_n can be made deterministic, as described above. We have therefore broken the unpredictability of G. $\square$

We can also define a notion of G being "unpredictable from the right", meaning that from seeing a proper suffix of $G(s)$, one cannot predict the previous bit. Because our notion of pseudo-random is symmetric with respect to left and right, we easily get the theorem that G is pseudo-random if and only if G is unpredictable from the right. As a corollary, we therefore get that G is unpredictable from the left if and only if G is unpredictable from the right.

To define "unpredictable" in the uniform adversary setting, we would let A be a probabilistic algorithm that, on input 1^n, computes in time polynomial in n a number i_n, $1 \leq i_n \leq l(n)$; then A will be given an $i_n - 1$ bit string, and after computing for polynomial (in n) time, outputs a bit. We leave the rest of the details to the reader, as well as a proof of the uniform-adversary version of the above theorem. (HINT: As before, there are two ways of proving the hard direction of this theorem. One way is to probabilistically approximate the values of $p_0, p_1, \ldots, p_{l(n)}$ in order to find an appropriate value for i_n. Another way is to chose the value of i_n randomly from $\{1, 2, \ldots, l(n)\}$.)

The reason we talk about whether or not pseudo-random generators exist is because we think they do, but we are unable to prove it. We cannot prove it because it is a much stronger assertion than "$\mathbf{P} \neq \mathbf{NP}$", and we are unable to prove this.

Theorem: If $\mathbf{P} = \mathbf{NP}$, then there is no pseudo-random generator.

Proof: Assume $\mathbf{P} = \mathbf{NP}$. Let G be a number generator with length function $l(n) > n$. Since $\mathbf{P} = \mathbf{NP}$, there is a polynomial time algorithm D which on inputs 1^n and α, accepts if and only if there is an n-bit string s such that $G(s) = \alpha$. So $p(n)$, the probability D accepts $G(s)$ for random n-bit s, is 1. Since there are only 2^n strings of length n, we have that $r(n)$, the probability D accepts a random $l(n)$ bit string, is $\leq \frac{2^n}{2^{l(n)}} \leq \frac{1}{2}$. So $p(n) - r(n) \geq \frac{1}{2} > \frac{1}{n}$ for $n > 2$. $\square$

We now want to show how a pseudo-random number generator that only does a little bit of expansion, can be used to construct a pseudo-random generator that does a lot of expansion.