# 1 Lecture 4: Pseudo-Random Generators

## 1.1 Motivation: One-Time Pad Encryption

**One-Time Pad** is a method of symmetric encryption that involves two parties $A$ and $B$ sharing information in the following way:

0. $A$ plans to share a message $m \in \{0,1\}^n$ with $B$
1. Both parties have a randomly-generated key $k \in \{0,1\}^n$
2. $A$ computes ciphertext $c = m \oplus k$ and sends $c$ to $B$
3. $B$ computes the message $m = c \oplus k$

Now, this system is favorable for a few reasons. First of all, it's clearly very easy for the desired parties to encrypt and decrypt the information they need to share.

One important point that makes one-time pad encryption so favorable is the following observation. Since the key $k$ is randomly generated, there really is no way for an adversary who does not have $k$ to figure out the message from the ciphertext. Especially for longer messages, a single ciphertext could correspond to many different messages, with completely different meanings, depending on the key.

This makes one-time pad encryption "**information-theoretic secure**". Some protocol is information-theoretic secure if, no matter how powerful an adversary is, there is nothing that they can possibly do to break it. And, for a one-time pad, as long as the adversary does not know $k$, there's no way they can possibly figure out which of the $2^n$ possible values of $k$ will give them the proper value of $m$ from $c$.

But we run into a couple problems with implementing this. One issue is that the key has to be the same size as the message, and generating long random strings can be very computationally intensive, which defeats the purpose of the efficiency of the rest of the protocol. The main problem, though, is that every time the parties want to send some message in either direction, they would somehow have to share the key without anyone intercepting it. If we could do this reliably and efficiently, we wouldn't even need to use one-time pads in the first place.

**Pseudo-random Generators (PRGs)** come in as a solution to both of these issues. The idea of a pseudo-random generator is that we want some function that can turn a short random-looking string called a seed into an arbitrarily long, "random-looking" string $k$.

This solves both of our problems because

1. We no longer have to generate a random string that is as long as our message– we need only generate a seed, and then use the PRG to get a key of the proper length.
2. Only one exchange– the sharing of the PRG– needs to be non-intercepted. Then, each time the parties want to communicate, they can send a seed, independently generate the keys, and then share the ciphertext.

## 1.2 Statistical Tests and "Random-Looking"

If we want both parties to end up with the same key, we can't add any more randomness to the mix, which means that if $k = PRG(s)$, then $k$ cannot have any more randomness to it than $s$ does. This is especially important when $k$ is much longer than $s$, since this means it can't technically be exactly the same as a fully randomly generated string of the same length. So, the best we can try to do is something that we would consider to be "random-looking", or **pseudo-random**.

So what does it mean for something to be pseudo-random? We'll say that a set is pseudo-random if no **PPT** algorithm can distinguish it from a sample of a truly random (uniform) distribution, except with negligible probability. (cf. the Turing Test for AI)

Remark: It is important to note that an implication of this definition is that both the pseudo-random and truly random distributions are "sampleable"– that is, for each distribution, there exists a **PPT** algorithm $S$ which randomly outputs values according to the distributions.

### 1.2.1 Computational Indistinguishability

First, some definitions:

A *probability ensemble* $X = \{X_n\}_{n \in \mathbb{N}}$ is a sequence of random variables (distributions) $X_n$, indexed by a security parameter $n \in \mathbb{N}$, where $n$ is usually the length of the strings in the distribution.

Additionally, a **distinguisher**, or **statistical test**, is some algorithm which samples from a random variable and decides which probability ensemble it is from, given two ensembles $X$ and $Y$.

Then, we have the following formal definition of **computational indistinguishability**:

Probability ensembles $X = \{X_n\}_{n \in \mathbb{N}}$ and $Y = \{Y_n\}_{n \in \mathbb{N}}$ are computationally indistinguishable if for all **distinguishers** $D \in$ **PPT** and constants $c \in \mathbb{N}$, there exists an integer $N$ such that $n \geq N$ gives:

$$|Pr[D(X_n) = 1] - Pr[D(Y_n) = 1]| < \frac{1}{n^c}.$$

### 1.2.2 Indistinguishability by Repeated Experiments

But note that above, we have the distinguisher only taking one sample from each random variable. This might raise some concerns– just how much information could you get from only one sample? It seems reasonable to believe that you could make a (perhaps significantly) more accurate evaluation given more samples. And, in reality, an adversary might be able to see many outputs from a pseudo-random number generator before they try to determine whether it is random or only pseudo-random.

Of course, it wouldn't be reasonable to take any arbitrary amount of samples, but just in case taking multiple samples is more effective than taking just one, we now consider distinguishers (statistical tests) which take polynomially many samples from each random variable.

As it turns out, $X$ and $Y$ are computationally indistinguishable for 1 sample if and only if they are computationally indistinguishable for polynomially many samples.

We define an **extended statistical test** to be a statistical test (distinguisher) which takes polynomially many samples from a distribution and decides which probability ensemble it is from, given two ensembles $X$ and $Y$.

Then, probability ensembles $X = \{X_n\}_{n\in\mathbb{N}}$ and $Y = \{Y_n\}_{n\in\mathbb{N}}$ are *computationally indistinguishable by repeated experiments* if for all extended statistical tests $T \in \mathbf{PPT}$ and polynomials $p, q$, there exists an integer $N$ such that $n \geq N$ gives:

$$|Pr[T(X_n^1, \ldots, X_n^{p(n)}) = 1] - Pr[T(Y_n^1, \ldots, Y_n^{p(n)}) = 1]| < \frac{1}{q(n)}.$$

Note that in the definition given above, the extended statistical test takes its $p(n)$ samples either all from $X_n$ or all from $Y_n$, but never from both at the same time. This will become important for what we are about to do.

The following note will also come in handy for proofs in this and the next lecture. If $X$ and $Y$ are distinguishable by repeated experiments, then there exists $T \in \mathbf{PPT}$ and polynomials $p, q$ such that for infinitely many $n \in \mathbb{N}$:

$$A = Pr[T(X_n^1, \ldots, X_n^{p(n)}) = 1], B = Pr[T(Y_n^1, \ldots, Y_n^{p(n)}) = 1] \implies |A - B| \geq \frac{1}{q(n)}.$$

This means that either $A - B \geq \frac{1}{q(n)}$ or $\leq -\frac{1}{q(n)}$ occurs infinitely many times. If it is the first, then we are done. If it is the second, we may consider the extended statistical test $T'(\cdot) = \overline{T(\cdot)}$. Then, we define $A'$ and $B'$ accordingly to see that $A' - B' \geq \frac{1}{q(n)}$ happens infinitely many times.

So, this means is that we can remove the absolute value signs: if $X$ and $Y$ are distinguishable, then there exists $T \in \mathbf{PPT}$ and polynomials $p, q$ such that there are infinitely many $n \in \mathbb{N}$ for which

$$Pr[T(X_n^1, \ldots, X_n^{p(n)}) = 1] - Pr[T(Y_n^1, \ldots, Y_n^{p(n)}) = 1] \geq \frac{1}{q(n)}.$$

### 1.2.3 The Hybrid Argument

Given probability ensembles $X, Y$, a parameter $n$ and a sampling polynomial $p$, we define a set of hybrids $\mathcal{H}_n^k$, $0 \le k \le p(n)$ as follows:

$$\mathcal{H}_n^k = \left(X_n^1, \ldots, X_n^k, Y_n^{k+1}, \ldots, Y_n^{p(n)}\right).$$

Note that $\mathcal{H}_n^0$ and $\mathcal{H}_n^{p(n)}$ consist only of samples from $Y_n$ and $X_n$, respectively.

Now, we suppose that $X$ and $Y$ are distinguishable by repeated experiments, by $T \in \mathbf{PPT}$ with polynomials $p, q$, in accordance with the definition. Then, we consider what happens when we feed each of the hybrids as input to $T$.

For each hybrid $\mathcal{H}_n^k$, we have some probability $Pr_k = T(\mathcal{H}_n^k) = 1$. And, by our assumption, we have $Pr_{(p(n))} - Pr_0 \ge \frac{1}{q(n)}$.

For any value of $k$, consider the sequence of samples given by

$$v = \left(x_1, \ldots, x_k, z, y_{k+2}, \ldots, y_{p(n)}\right),$$

where $z$ is a sample from an arbitrary distribution. Note that if $z$ is taken from $X_n$, then $v$ corresponds to a sequence of samples from $\mathcal{H}_n^k$. And if $z$ is taken from $Y_n$, then $v$ corresponds to a sequence of samples from $\mathcal{H}_n^{k+1}$.

Then, define a distinguisher $D$ that will take a random variable $Z$ and does the following to distinguish whether it belongs to $X$ or $Y$:

1. select a random value of $k \in [0, p(n))$.
2. sample from $X, Y, Z$ to make $v = \left(x_1, x_2, \ldots, x_k, z, y_{k+2}, \ldots, y_{p(n)}\right)$.
3. output $T(v)$

Now, we have

$$Pr[D(X_n) = 1] = \sum_{k=0}^{p(n)-1} \frac{1}{p(n)} Pr_{k+1}, \quad Pr[D(Y_n) = 1] = \sum_{k=0}^{p(n)-1} \frac{1}{p(n)} Pr_k$$

$$Pr[D(X_n) = 1] - Pr[D(Y_n) = 1] = \frac{1}{p(n)}(Pr_{p(n)} - Pr_0) \ge \frac{1}{p(n)q(n)}.$$

Which means that $D$ distinguishes $X$ and $Y$, so these two ensembles are also distinguishable by a single sample.

Note that if they are indistinguishable by repeated experiments, then by the definition we may simply choose $p(n) \equiv 1$, and it follows that they are computationally indistinguishable (by 1 sample). And, we have just shown that if they are not indistinguishable by repeated experiments, then they are not computationally indistinguishable. So, it follows that two ensembles are indistinguishable by repeated experiments if and only if they are computationally indistinguishable.

## 1.3 Constructing a PRG

Formally, a PRG is some deterministic algorithm $G$ which satisfies the following criteria:

1. $G(x, Q)$ runs in polynomial time with respect to $|x|$ and $Q(|x|)$, where $Q$ is some polynomial.

2. $G(x, Q)$ outputs strings of length $Q(|x|)$ for all $x$.

3. For all $Q$, the distribution $\{G(x, Q)\}$ is computationally indistinguishable from true randomness, which we define to be $\mathcal{U}_{Q(|x|)}$, the uniform distribution on strings of length $Q(|x|)$.

### 1.3.1 PRG from a 1WP

Let $f$ be a one-way permutation, and $n$ a security parameter.

Then, we define an algorithm $G$ which takes as input a random seed $X_0$ and a polynomial $Q(\cdot)$, and does the following:

0. choose a random string $P$ with $|P| = |X_0| = n$, for the security parameter $n$

1. for $i \in \{1, \ldots, Q(|X_0|)\}$, let $X_i = f(X_{i-1})$ and add the hard-core bit $\langle P, X_{i-1}\rangle$ to an output sequence $S$.

2. output $P$ and $X_{Q(|x|)}$

3. output the output sequence $S$ in reverse order.

We now wish to show that this $G$ is a PRG. It will follow from this that every 1WP can be used to create a PRG.

### 1.3.2 The Next-Bit Test

As a tool to prove that pseudo-random generators produce output that cannot be distinguished from true randomness, we define the "**Next Bit Test**":

We consider an ensemble $X = \{X_n\}_{\mathbb{N}}$, where $X_n$ consists of strings which are $P(n)$ bits long, for some predetermined polynomial $P$.

$X$ passes the *next bit test* if, for all $c \in \mathbb{N}$ and $A \in \mathbf{PPT}$, there exists an integer $N_c$ such that if $n > N_c$, we have

$$Pr_{x \in X_n}[A(x_1, \ldots, x_i) = x_{i+1}] < \frac{1}{2} + \frac{1}{n^c}.$$

That is, for any **PPT** adversary $A$, there are only finitely many $X_n \in X$ for which $A$, given a prefix of any $x \in X_n$, can predict the next bit of $x$, with probability $> \frac{1}{2}$ by a non-negligible amount. Note that this probability is an average over all $x \in X_n$.

We now show, by contradiction, that any PRG defined as previously specified will pass the next-bit test.

Consider a PRG $G(\cdot, P)$ which generates the ensemble $X = \{X_n\}_{\mathbb{N}}$ where $X_n$ is the distribution of pseudo-random strings of length $P(n)$, which are generated by $G$ operating on random seeds of length $n$.

Suppose then that there exists an adversary $A \in \mathbf{PPT}$ that, for infinitely many $n$, can predict the next bit of $x \in X_n$, given a prefix of some length $i$, with probability exceeding 1/2 by a non-negligible amount.

Then, consider some string $y = f(x)$ for some $x$, where $f$ is the one-way function used to create $G$. If $f$ is a 1WP, then $y$ may be any string of a given length.

Then, if we run $G(y)$, we may look at the last $i$ bits of the input: $\langle f^i(y), p\rangle, \ldots, \langle f(y), p\rangle, \langle y, p\rangle$. (recall that the bits are given in reverse order in the output.) Note that these are the same as the first $i$ bits of the output if we had some $z$ such that $f^{P(n)-i}(z) = y$, so we may feed these bits to our adversary $A$, which will predict the next bit $b = \langle f^{-1}(y), p\rangle$ with probability exceeding 1/2 by some non-negligible amount.

But what did we just do? We took $y = f(x)$, and we predicted the hard-core bit with some non-negligible advantage. As we showed before, this means that we can invert a one-way function, which by the assumption of one-way functions, is not possible. So, we have reached a contradiciton. Thus, our original assumption must have been false, which means that such an adversary must not exist. So, all PRG's defined as specified pass the next bit test.

### 1.3.3 Equivalence of Next-Bit Test and Pseudo-random Criteria

We show now that passing the next-bit test and being pseudo-random are equivalent.

It is easy to see that if an ensemble is pseudo-random, then it passes the next-bit test. This is because the next-bit test is a statistical test (distinguisher), as any random distribution must necessarily pass the next-bit test, since nothing can possibly predict hte next bit from truly random strings with probability $> 1/2$. And since any pseudo-random ensemble passes any statistical test, by definition, this means that any pseudo-random ensemble must pass the next-bit test.

To prove that there is implication in the other direciton, we use a hybrid argument and a proof by contrapositive.

That is, we show that if some ensemble is not pseudorandom, then it will fail the next bit test. In particular, if there is some distinguisher $D$ which can tell apart the ensemble from true randomness, then we can construct an adversary $A$ which predicts the next bit of strings from distributions in the ensemble with non-negligible advantage.

Before we begin, recall that for our distinguisher $D$ we have, for infinitely many $n$:

$$Pr[D(X_n) = 1] - Pr[D(\mathcal{U}_n) = 1] \geq \frac{1}{q(n)}$$

We now proceed by defining the hybrids:

$$\mathcal{H}_n^k = X_n[1], \ldots, X_n[k], \mathcal{U}_n[k+1], \ldots, \mathcal{U}_n[n]$$

Note that $\mathcal{H}_n^0$ is simply a sample from $\mathcal{U}_n$, and that $\mathcal{H}_n^n$ is simply a sample from $X_n$.

Then, define an adversary $A$ as follows, using our distinguisher $D$:

0. Take as input a string $x$ of length $n$, and a random $k$ from $\{1, \ldots, n\}$
1. read the first $k$ bits of $x$
2. append $n-k$ random bits, to make $v = x_1, \ldots, x_k, r_{k+1}, \ldots, r_n$
3. run $D(v)$.
   (a) if $D(v) = 1$, then output $r_{k+1}$
   (b) if $D(v) = 0$, then output $\overline{r_{k+1}}$

The idea here is that if $D(v) = 1$, $D$ thinks $v \in X_n$, and we know that $D$ is probably correct. Otherwise, $D$ thinks $v \notin X_n$, so we can go ahead and guess that $r_{k+1}$ is not the correct next bit.

<u>**Calculations**</u>

For each $k \in \{0, 1, \ldots, n\}$, define

$$p_k = Pr[D(x_1 \cdots x_k r_{k+1} \cdots r_n) = 1], \quad q_k = Pr[D(x_1 \cdots x_k \overline{x_{k+1}} r_{k+2} \cdots r_n) = 1].$$

Note that since the $r_i$'s are random, we have a 1/2 probability that $r_{k+1} = x_{k+1}$, and a 1/2 probability that $r_{k+1} = \overline{x_{k+1}}$. So:

$$p_k = \frac{p_{k+1}}{2} + \frac{q_k}{2} = \frac{1}{2}(p_{k+1} + q_k).$$

Then,

$$Pr[A(x,k) = x_{k+1}]$$
$$= Pr[x_{k+1} = r_{k+1}] \cdot Pr[D(v) = 1 \mid x_{k+1} = r_{k+1}]$$
$$+ Pr[x_{k+1} \neq r_{k+1}] \cdot Pr[D(v) = 0 \mid x_{k+1} \neq r_{k+1}].$$

Since $r_{k+1}$ is random, this is equivalent to

$$\frac{1}{2}(Pr[D(v) = 1 \mid x_{k+1} = r_{k+1}] + Pr[D(v) = 0 \mid x_{k+1} \neq r_{k+1}])$$

$$= \frac{1}{2}(Pr[D(x_1 \cdots x_k x_{k+1} r_{k+2} \cdots r_n) = 1] + Pr[D(x_1 \cdots x_k \overline{x_{k+1}} r_{k+2} \cdots r_n) = 0])$$

$$= \frac{1}{2}(p_{k+1} + (1 - q_k)) = \frac{1}{2}p_{k+1} + \frac{1}{2} - \frac{1}{2}q_k = \frac{1}{2} + (p_{k+1} - p_k).$$

Now, note that the hybrids defined here are precisely the hybrids we used above to show that computational indistinguishability is equivalent to indistinguishability by repeated experiments. That is, we have that $p_n - p_0 \geq \frac{1}{p(n)q(n)}$ for some polynomials $p, q$, and it follows then by the pigeonhole principle that there exists a $k$ for which $p_{k+1} - p_k \geq \frac{1}{np(n)q(n)}$, i.e., for which $p_{k+1} - p_k$ is some non-negligible $\varepsilon$.

Then, we have $Pr[A(x,k) = x_{k+1}] = \frac{1}{2} + (p_{k+1} - p_k) \geq \frac{1}{2} + \frac{\varepsilon}{n}$, so we have created a next-bit predictor with non-negligible advantage.

This shows that a non-pseudo-random ensemble is also next-bit predictable. So if something passes the next-bit test, then it is pseudo-random.

Thus, it follows that our generators, as constructed from 1WF's (1WP's) are indeed pseudo-random generators.