**Introduction to Cryptography** 02/06/2018

# Lecture 7: Pseudo Random Generators

*Instructor: Vipul Goyal* *Scribe: Eipe Koshy*

## 1 Introduction

Randomness is very important in modern computational systems. For example, randomness is needed for encrypting a session key in an SSL connection or for encrypting a hard drive. There are multiple challenges associated with generating randomness for modern systems:

- True randomness is difficult to get
- Large amounts of randomness are needed in practice

Given these challenges, several different input sources are used for randomness on modern systems. For example, key strokes or mouse movements can be used as a source of randomness. However, these sources are limited in the amount of randomness they can generate.

In addition, it is critical that the sources of randomness be truly random or close to being truly random. Several crypto-systems have been defeated[1] because their randomness generators were broken (i.e. not truly random.)

As a solution to both the challenges detailed above for generating randomness, an approach to build a kind of randomness generator that creates random strings of a desired length would be to first, start off with building a short random string (of length, say, $n$ bits), and then, expand it to a longer random looking string $k$ (of length, say, $l(n)$ bits.) We can represent this "randomness" generator as $f : \{0,1\}^n \rightarrow \{0,1\}^{l(n)}$

A key question is: Can we can really expand a short string with a few random bits into a longer string with many random bits? The answer lies in the definition of the term "random looking" that we we used in our definition above. The long string that we have generated is not truly random (since with an input of length $n$ bits, we can only generate $2^n$ outputs, as compared to the $2^{l(n)}$ outputs needed if the string was truly random.) However, as detailed in the next section, we can generate a "pseudo-random" string such that the output appears "random looking" to a PPT (probabilistic polynomial time) adversary.

## 2 Pseudorandomness

Let us suppose there are n uniformly random bits. Pseudorandomness can be defined as finding a deterministic (polynomial-time) algorithm $f : \{0,1\}^n \rightarrow \{0,1\}^{l(n)}$ such that:

- the output of $f(x)$ contains $l(n)$ bits

[1] https://en.wikipedia.org/wiki/Random_number_generator_attack#Prominent_examples

- $\{0,1\}^{l(n)}$ appears to be as "random looking" as a truly random string

$f : \{0,1\}^n \rightarrow \{0,1\}^{l(n)}$ is called a pseudorandom generator (PRG) that takes $n$ random bits with no additional randomness of its own and outputs $l(n)$ random bits.

What do we mean by "random looking"? The key idea is that the the pseudorandom generator (PRG) produces output that is indistinguishable from that of a true random number generator for a PPT adversary. "Random looking" indicates that the bits should not follow any pattern. The "random looking" string should also pass all necessary statistical tests like:

- There should be as many $0s$ as there are $1s$
- Each particular bit is roughly unbiased
- Each sequence of bits occur roughly with same probability

The main idea is that no efficient computer should be able to tell apart the output of $f(x)$ (i.e. the pseudorandom generator) from that of a truly random generator. Before knowing more about pseudorandomness, let us look at some of the underlying definitions that will help us understand the concept better.

**Definition 1 (Support)** *The support or sample space can be defined as the set of all possible values of outcomes of an experiment.*

**Definition 2 (Distribution)** *$X_n$ is a distribution over support $\xi$ if it assigns probability $p_s$ to each $s \in \xi$ such that $\sum\limits_{s \in \xi} p_s = 1$.*

**Definition 3 (Distribution Ensembles)** *A sequence $\{X_n\}_{n \in N}$ is called an ensemble if for each $n \in N$, $X_n$ is a distribution over $\{0,1\}^*$*

*Generally, Xn will be a distribution over the sample space $\{0,1\}^{l(n)}$*

**Definition 4 (Identical Distributions)** *$X_n$ and $X'_n$ are said to be identical distributions if $\forall$ s $\in \{supp^2(X_n) \cup supp(X'_n)\}$:*

$$\begin{aligned} Pr[x \leftarrow X_n : x = s] = Pr[x \leftarrow X'_n : x = s] \\ \implies supp(X_n) = supp(X'_n) \end{aligned} \tag{1}$$

## 3 Computational Indistinguishability

The term computationally indistinguishable is used to formalize a way to capture what it means for two distributions $X$ and $Y$ to look alike to any efficient test.
In short,

$$\text{Efficient test} = \text{Efficient computation} = \text{Non-uniform PPT}$$

[2]Note: supp refers to the support of the distribution

Intuition: No non-uniform PPT distinguisher algorithm $D$ can tell the two distributions $X$ and $Y$ apart i.e. behavior of $D$ is same for both of them.

Informally, we say that we have computational indistinguishability when the output of a PRG is indistinguishable from that of a true random number generator. We will begin by attempting to define computational indistinguishability.

**Definition 5 (Computational Indistinguishability - first attempt)** *Distribution ensembles $X_n$ and $X'_n$ are said to be computationally indistinguishable if $\forall n \in N$ and $s \in \{supp(X_n) \cup supp(X'_n)\}$,*

$$\left|Pr[x \leftarrow X_n : x = s] - Pr[x \leftarrow X'_n : x = s]\right| \leq \nu(n), \tag{2}$$

*where $\nu(n)$ represents a negligible function.*

Unfortunately, this definition does not work. To see this, let us consider the following distribution ensembles:

- $\{X_n\}$ represents all even numbers that can be represented using $n$ bits.
- $\{X'_n\}$ represents all odd numbers that can be represented using $n$ bits.

It is easy to see that the number of elements in each ensemble above is $2^{n-1}$.

Let us consider the case where "$s$" in our definition of computational indistinguishability above is an even element.
The LHS, i.e., $|Pr[x \leftarrow X_n : x = s] - Pr[x \leftarrow X'_n : x = s]|$ evaluates to:
$2^{n-1} - 0 = 2^{n-1} \nleq \nu(n)$
Given this definition does not work, we will attempt an alternate definition:

**Definition 6 (Computational Indistinguishability - second and correct attempt)** *Consider distribution ensembles $\{X_n\}$ and $\{X'_n\}$ and a PPT adversary $A$. We define:*

- *$A(x) = 0$ if $x$ is sampled from $\{X_n\}$*
- *$A(x) = 1$ if $x$ is sampled from $\{X'_n\}$*

*$\{X_n\}$ and $\{X'_n\}$ are said to be computationally indistinguishable if $\forall$ PPT adversaries $A$ and $n \in N$,*

$$\left|Pr[x \leftarrow X_n : A(x) = 0] - Pr[x \leftarrow X'_n : A(x) = 0]\right| \leq \nu(n) \tag{3}$$

*, where $\nu(n)$ represents a negligible function.*

Note: The LHS of this definition, i.e. $|Pr[x \leftarrow X_n : A(x) = 0] - Pr[x \leftarrow X'_n : A(x) = 0]|$ is also referred to as distinguishing advantage.

**Definition 7 (Prediction advantage)** *Denote distribution ensembles $X_n$ and $X'_n$ as $X_n^0$ and $X_n^1$ respectively. Then, an alternate representation for computational indistinguishability follows as below; the LHS of this expression is defined as prediction advantage.*
*The expression is $\forall$ PPT adversaries $A$,:*

$$\left|Pr[b \xleftarrow{\$} \{0,1\}, x \leftarrow X_n^b : A(x) = b] - (1/2)\right| \leq \nu(n) \tag{4}$$

*, where $\nu(n)$ represents a negligible function.*

An alternate definition of prediction advantage can be stated as below:

**Definition 8 (Prediction advantage - alternate definition)** *A non-uniform PPT A is said to guess the bit b from the sample x that is picked out of sequence of distribution $\{X_n^b\}$ (made with randomness , say $,) with prediction advantage* $\Big|Pr[b \overset{\$}{\leftarrow} \{0,1\}, x \leftarrow X_n^b : A(x) = b] - (1/2)\Big|$ *negligibly close to 0.*

**Lemma 1 (Informal)** *If distinguishing advantage is negligible, then prediction advantage is also negligible and vice versa. Alternately, we can restate the same lemma as that computational indistinguishability implies prediction advantage is negligible and vice versa.*

**Proof.** Let us start by considering prediction advantage, the expression for which is as follows:

$$\begin{aligned}
&\Big|Pr[b \overset{\$}{\leftarrow} \{0,1\}, x \leftarrow X_n^b : A(x) = b] - (1/2)\Big| \\
&= \big|Pr[x \leftarrow X_n^0 : A(x) = 0]Pr[b = 0] + Pr[x \leftarrow X_n^1 : A(x) = 1]Pr[b = 1] - (1/2)\big| \\
&= (1/2)\big|Pr[x \leftarrow X_n^0 : A(x) = 0] + Pr[x \leftarrow X_n^1 : A(x) = 1] - 1\big| ...(\because Pr[b = 0] = Pr[b = 1] = (1/2)) \\
&= (1/2)\big|Pr[x \leftarrow X_n^0 : A(x) = 0] + (1 - Pr[x \leftarrow X_n^1 : A(x) = 0]) - 1\big| \\
&= (1/2)\big|Pr[x \leftarrow X_n^0 : A(x) = 0] - Pr[x \leftarrow X_n^1 : A(x) = 0])\big| \\
&= (1/2) * \textit{the expression for distinguishing advantage} \\
&= (1/2) * \nu(n)
\end{aligned}$$

... where, $\nu(n)$ , is a negligible function

**Lemma 2 (Stronger)** *Distinguishing advantage and predictive advantage are within a factor of 2 of each other.*

## 3.1 Properties of Computational Indistinguishability

Note: If distribution ensembles $\{X_n\}$ and $\{X'_n\}$ are computationally indistinguishable, then we can represent them as $\{X_n\} \underset{C}{\approx} \{X'_n\}$

1. *Closure*: For every PPT machine $M$, if $\{X_n\} \underset{C}{\approx} \{X'_n\}$, then $M\{X_n\} \underset{C}{\approx} M\{X'_n\}$
   Note: For proof, we can use contradiction, since if this were not true, we could use $M$ to distinguish the two ensembles.

2. *Transitivity*: Suppose $(X, Y)$ are computationally indistinguishable with advantage $\epsilon_1$ and $(Y, Z)$ are computationally indistinguishable with advantage $\epsilon_2$, then $(X, Z)$ are computationally indistinguishable with advantage $\epsilon \leq \epsilon_1 + \epsilon_2$.

   Note: We can also use the triangle inequality to prove this property.
   **Proof.** Given,

   $$\begin{aligned}
   &|Pr[x \leftarrow X : A(x) = 0] - Pr[x \leftarrow Y : A(x) = 0]| \leq \epsilon_1 \implies \\
   &|Pr[x \leftarrow X : A(x) = 0] - Pr[x \leftarrow Z : A(x) = 0])| \leq |\epsilon_1 + Pr[x \leftarrow Y : A(x) = 0] - Pr[x \leftarrow Z : A(x) = 0])| \\
   &\implies |Pr[x \leftarrow X : A(x) = 0] - Pr[x \leftarrow Z : A(x) = 0])| \leq |\epsilon_1 + \epsilon_2|
   \end{aligned}$$

**Lemma 3 (Hybrid lemma)** *Let $X_1$, $X_2$, $X_3$, ....., $X_n$ be distributional ensembles for $m =$ poly(n). Suppose $\exists$ PPT A which distinguishes $X_1$ from $X_n$ with advantage $\epsilon$. Then when $\exists$ i where 1 $\leq$ i $\leq$ m-1, a PPT algorithm B can distinguish $X_i$ and $X_{i+1}$ with advantage greater or equal to $\frac{\epsilon}{m}$*

Note: Note, this lemma is used in many proofs that contains hybrid arguments.

**Proof.**
Let $\epsilon_i$ denote distinguishing advantage between $X_i$ and $X_{i+1}$.
Let us assume that $\epsilon_i \leq \frac{\epsilon}{m}$ $\forall$ i.
Then, $\epsilon \leq \sum_{i=1}^{m-1} \epsilon_i \implies \epsilon \leq \epsilon * \frac{(m-1)}{m}$

This is, of course, a contradiction. Hence, we can say that there exists $i$ such that $\epsilon_i \geq \frac{\epsilon}{m}$.

## 4 Return to Pseudorandomness

Intuition: A distribution is pseudorandom if it looks like a uniform distribution for any efficient test.

**Definition 9 (Pseudorandom ensembles)** *An ensemble $\{X_n\}$, where $X_n$ is a distribution over $\{0,1\}^{l(n)}$, is said to be pseudorandom if:*

$$\{X_n\} = U_{l(n)}$$

*... where $U_{l(n)}$ represents a uniform distribution over $\{0,1\}^{l(n}$*

## 5 Pseudorandom Generators (PRG)

**Definition 10 (Pseudorandom Generators)** *A deterministic algorithm G: $\{0,1\}^n \rightarrow \{0,1\}^{l(n)}$ is called a pseudorandom generator (PRG) if:*

1. *G can be computed in polynomial time*
2. *l(n) > n*
3. *$\{x \leftarrow \{0,1\}^n : G(x)\} \underset{C}{\approx} \{U_{l(n)}\}$, where $\{U_{l(n)}\}$ = a uniform l(n) bit string*

Note: The stretch of $G$ is defined as $|G(x)| - |x|$