Chapter 3

Indistinguishability and Pseudo-Randomness

Recall that one main drawback of the One-time pad encryption scheme—and its simple encryption operation $Enc_k(m) = m \oplus k$—is that the key k needs to be as long as the message m. A natural approach for making the scheme more efficient would be to start off with a short random key k and then try to use some *pseudo-random generator* g to expand it into a longer "random-looking" key $k' = g(k)$, and finally use k' as the key in the One-time pad.

Can this be done? We start by noting that there can not exist pseudo-random generators g that on input k generate a *perfectly random* string k', as this would contradict Shannon's theorem (show this). However, remember that Shannon's lower bound relied on the premise that the adversary Eve is computationally unbounded. Thus, if we restrict our attention to efficient adversaries, it might be possible to devise pseudo-random generators that output strings which are "sufficiently" random-looking for our encryption application.

To approach this problem, we must first understand what it means for a string to be "sufficiently random-looking" to a polynomial time adversary. Possible answers include:

- Roughly as many 0 as 1.
- Roughly as many 00 as 11
- Each particular bit is "roughly" unbiased.
- Each sequence of bits occurs with "roughly" the same probability.

- Given any prefix, it is hard to guess the next bit.
- Given any prefix, it is hard to guess the next sequence.

All of the above answers are examples of specific *statistical tests*—and many many more such test exist in the litterature. For specific simulations, it may be enough to use strings that pass some specific statistical tests. However, for cryptography, we require the use of string that passes *all* (efficient) statistical tests. At first, it seems quite overwhelming to test a candidate pseudo-random generator against *all* efficient tests. To do so requires some more abstract concepts which we now introduce.

3.1 Computational Indistinguihability

We introduce the notion of *computational indistinguishability* to formalize what it means for two probability distributions to "look" the same in the eyes of a computationally bounded adversary. This notion is one of the corner stones of modern cryptography. As our treatment is asymptotic, the actual formalization of this notion considers sequences—called *ensembles*—of probability distributions (or growing output length).

Definition 48.1 (Ensembles of Probability Distributions)**.** ...add

Definition 48.1. (Computational Indistinguishability). Let $\{X_n\}_{n\in N}$ and $\{Y_n\}_{n\in N}$ be ensembles of probability distributions where X_n, Y_n are probability distributions over $\{0,1\}^{l(n)}$ for some polynomial $l(\cdot)$. We say that $\{X_n\}_{n\in N}$ and $\{Y_n\}_{n\in N}$ are *computationally indistinguishable* (abbr. $\{X_n\}_{n\in N} \approx \{Y_n\}_{n\in N}$) if for all non-uniform PPT D (called the "distinguisher"), there exists a negligible function $\epsilon(n)$ such that $\forall n \in N$

$$|\Pr[t \leftarrow X_n, D(t) = 1] - \Pr[t \leftarrow Y_n, D(t) = 1]| < \epsilon(n).$$

In other words, two (ensembles of) probability distributions are computationally indistinguishable if no efficient distinguisher D can tell them apart better than with a negligible advantage.

To simplify notation, we say that *D distinguishes the distributions X_n* and Y_n with probability ϵ if

$$|\Pr[t \leftarrow X_n, D(t) = 1] - \Pr[t \leftarrow Y_n, D(t) = 1]| > \epsilon.$$

Additionally, we say *D distinguishes the probability ensembles $\{X_n\}_{n\in N}$* and $\{Y_n\}_{n\in N}$ with probability $\mu(\cdot)$ if $\forall n \in N$, D distinguishes X_n and Y_n with probability $\mu(n)$.

Properties of Computational Indistinguishability

We highlight some important (and natural) properties of the notion of indistinguishability. This properties will be used over and over again in the remainder of the course.

Closure under efficient opertations

The first property formalizes the statement "If two distributions look the same, then they look the same no matter how you process them" (as long as the processing is efficient). More formally, if two distributions are indistinguishiable, then they remain indistinguishable also if one applies a p.p.t. computable operation to them.

Lemma 49.1. *Let $\{X_n\}_{n \in N}$,$\{Y_n\}_{n \in N}$ be ensembles of probability distributions where X_n, Y_n are probability distributions over $\{0,1\}^{l(n)}$ for some polynomial $l(\cdot)$, and let M be a p.p.t. machine. If $\{X_n\}_{n \in N} \approx \{Y_n\}_{n \in N}$, then $\{M(X_n)\}_{n \in N} \approx \{M(Y_n)\}_{n \in N}$.*

Proof. Suppose there exists a non-uniform p.p.t. D and non-negligible function $\mu(n)$ s.t D distinguishes $\{M(X_n)\}_{n \in N}$ and $\{M(Y_n)\}_{n \in N}$ with probability $\mu(n)$. That is,

$$\begin{aligned} &|\Pr[t \leftarrow M(X_n) : D(t) = 1] - \Pr[t \leftarrow M(Y_n) : D(t) = 1]| &>& \mu(n) \\ \Rightarrow &|\Pr[t \leftarrow X_n : D(M(t)) = 1] - \Pr[t \leftarrow Y_n : D(M(t)) = 1]| &>& \mu(n). \end{aligned}$$

In that case, the non-uniform p.p.t machine $D'(\cdot) = D(M(\cdot))$ also distinguishes $\{X_n\}_{n \in N}$, $\{Y_n\}_{n \in N}$ with probability $\mu(n)$, which contradicts that the assumption that $\{X_n\}_{n \in N} \approx \{Y_n\}_{n \in N}$. □

Transitivity - The Hybrid Lemma

We next show that the notion of computational indistinguishability is transitive; namely, if $\{A_n\}_{n \in N} \approx \{B_n\}_{n \in N}$ and $\{B_n\}_{n \in N} \approx \{C_n\}_{n \in N}$, then $\{A_n\}_{n \in N} \approx \{C_n\}_{n \in N}$. In fact, we prove a generalization of this statement which considers $m = poly(n)$ distributions.

Lemma 49.2 (The Hybrid Lemma)**.** *Let $X^1, X^2, \cdots, X^m$ be a sequence of probability distributions. Assume that the machine D distinguishes X^1 and X^m with probability ϵ. Then there exists some $i \in [1, \cdots, m-1]$ s.t. D distinguishes X^i and X^{i+1} with probability $\frac{\epsilon}{m}$.*

Proof. Assume D distinguishes X^1, X^m with probability ϵ. That is,

$$|\Pr\left[t \leftarrow X^1 : D(t) = 1\right] - \Pr\left[t \leftarrow X^m : D(t) = 1\right]| > \epsilon$$

Let $g_i = \Pr\left[t \leftarrow X^i : D(t) = 1\right]$. Thus, $|g_1 - g_m| > \epsilon$. This implies,

$$\begin{aligned} &|g_1 - g_2| + |g_2 - g_3| + \cdots + |g_{m-1} - g_m| \\ &\quad \geq |g_1 - g_2 + g_2 - g_3 + \cdots + g_{m-1} - g_m| \\ &\quad = |g_1 - g_m| > \epsilon. \end{aligned}$$

Therefore, there must exist i such that $|g_i - g_{i+1}| > \frac{\epsilon}{m}$. □

Remark 50.1 (A geometric interpretation). Note that the probability with which D outputs 1 induces a metric space over probability distributions over strings t. Given this view the hybrid lemma is just a restatement of the traingle inequality over this metric spaces; in other words, if the distance between two consequetive points—representing probability distributions—-is small, then the distance between the extremal points is small too.

Note that because we lose a factor of m when we have a sequence of m distributions, the hybrid lemma can only be used to deduce transitivity when m is polynomially related to the security parameter n. (In fact, it is easy to construct a "long" sequence of probability distributions which are all indistinguishable, but where the extremal distributions are distinguishable.)

3.2 Pseudo-randomness

Using the notion of computational indistinguishability, we next turn to defining pseudo-random distributions.

Definition of Pseudo-random Distributions

Let U_n denote the uniform distribution over $\{0,1\}^n$, i.e, $U_n = \{t \leftarrow \{0,1\}^n : t\}$. We say that a distribution is pseudo-random if it is indistinguishable from the uniform distribution.

Definition 50.1. (Pseudo-random Ensembles). The probability ensemble $\{X_n\}_{n\in N}$, where X_n is a probability distribution over $\{0,1\}^{l(n)}$ for some polynomial $l(\cdot)$, is said to be *pseudorandom* if $\{X_n\}_{n\in N} \approx \{U_{l(n)}\}_{n\in N}$.

Note that this definition effectively says that a pseudorandom distribution needs to pass *all* efficiently computable statistical tests that the uniform distribution would have passesd; otherwise the statistical test would distinguish the distributions.

Thus, at first sight it might seem very hard to check or prove that a distribution is pseudorandom. As it turns out, there are *complete* statistical test; such a test has the property that if a distribution passes only that test, it will also pass all other efficient tests. We proceed to present such a test.

A complete statistical test: The next-bit test

We say that a distribution passes the *next-bit test* if no efficient adversary can, given any prefix of a sequence sampled from the distribution, predict the next bit in the sequence with probability significantely better than $\frac{1}{2}$ (recall that this was one of the test originally suggested in the introduction of this chapter).

Definition 51.1. An ensemble of probability distributions $\{X_n\}_{n \in N}$ where X_n is a probability distribution over $\{0,1\}^{l(n)}$ for some polynomial $l(n)$ is said to pass the *Next-Bit Test* if for every non-uniform p.p.t. A, there exists a negligible function $\epsilon(n)$ s.t $\forall n \in N$ and $\forall i \in [0, \cdots, l(n)]$, it holds that

$$\mathsf{Pr}\left[t \leftarrow X_n : A(1^n, t_1 t_2 \ldots t_i) = t_{i+1}\right] < \frac{1}{2} + \epsilon(n).$$

Here, t_i denotes the i'th bit of t.

Remark 51.1. Note that we provide A with the additional input 1^n. This is simply allow A to have size and running-time that is polynomial in n and not simply in the (potentially) short prefix $t_0 \ldots t_i$.

Theorem 51.1 (Completeness of Next-Bit Test)**.** *If a probability ensemble* $\{X_n\}_{n \in N}$ *passes the next-bit test then* $\{X_n\}_{n \in N}$ *is pseudo-random.*

Proof. Assume for the sake of contradiction that there exists a nonuniform p.p.t. distinguisher D, and a polynomial $p(\cdot)$ such that for infinitely many $n \in \mathbb{N}$, D distinguishes X_n and $U_{l(n)}$ with probability $\frac{1}{p(n)}$. We contruct a machine A that predicts the next bit of X_n for every such n. Define a sequence of *hybrid distributions* as follows.

$$H_n^i = \{x \leftarrow X_n \,:\, u \leftarrow U_{l(n)} \,:\, x_0 x_1 \ldots x_i u_{i+1} \ldots u_{l(n)}\}$$

Note that $H_n^0 = U_{l(n)}$ and $H_n^{l(n)} = X_n$. Thus, D distinguishes between H_n^0 and $H_n^{l(n)}$ with probability $\frac{1}{p(n)}$. It follows from the hybrid lemma that there exists some $i \in [l(n)]$ such that D distinguishes between H_n^i and H_n^{i+1} with probability $\frac{1}{p(n)l(n)}$. That is,

$$\left|\Pr\left[t \leftarrow H_n^i \;:\; D(t) = 1\right] - \Pr\left[t \leftarrow H_n^{i+1} \;:\; D(t) = 1\right]\right| > \frac{1}{p(n)}$$

We assume without loss of generality that

$$\Pr\left[t \leftarrow H_n^{i+1} \;:\; D(t) = 1\right] - \Pr\left[t \leftarrow H_n^i \;:\; D(t) = 1\right] > \frac{1}{p(n)l(n)} \tag{3.1}$$

Note that this is without loss of generality since we could always replace D with $D'(\cdot) = 1 - D(\cdot)$; it then must be the case that there exists infinitely many n for which either D or D' works.

Recall, that the only difference between H^{i+1} and H^i is that in H^{i+1} the $(i+1)$'th bit is x_{i+1}, whereas in H^i it is u_{i+1}. Thus, intuitively, D—given only the prefix $x_1 \dots x_i$—can tell apart x_{i+1} from a uniformly chosen bit. We show how to constuct a predictor A that uses such a D to predict x_{i+1}: A on input $(1^n, t_1t_2 \dots t_i)$ picks $l(n) - i$ random bits $u_{i+1} \dots u_{l(n)} \leftarrow U^{l(n)-1}$, and lets $g \leftarrow D(t_1 \dots t_i u_{i+1} \dots u_{l(n)})$. If $g = 1$, it outputs u_{i+1}, otherwise it outputs $\bar{u}_{i+1} = 1 - u_{i+1}$. We show that

$$\Pr\left[t \leftarrow X_n : A(1^n, t_1t_2 \dots t_i) = t_{i+1}\right] > \frac{1}{2} + \frac{1}{p(n)l(n)}.$$

Towards this goal, consider the distribution $\tilde{H}(i)_n$ defined as follows:

$$\tilde{H}_n^i = \{x \leftarrow X_n \;:\; u \leftarrow U_{l(n)} \;:\; x_0x_1 \dots x_{i-1}\bar{x}_i u_{i+1} \dots u_{l(m)}\}$$

Note that,

$$\begin{aligned}
&\Pr\left[t \leftarrow X_n;\; A(1^n, t_1 \dots t_i) = t_{i+1}\right] \\
&\quad = 1/2 \Pr\left[t \leftarrow H_n^{i+1} \;:\; D(t) = 1\right] + 1/2 \Pr\left[t \leftarrow \tilde{H}_n^{i+1} \;:\; D(t) \neq 1\right] \\
&\quad = 1/2 \Pr\left[t \leftarrow H_n^{i+1} \;:\; D(t) = 1\right] - 1/2 (1 - \Pr\left[t \leftarrow \tilde{H}_n^{i+1} \;:\; D(t) = 1\right])
\end{aligned}$$

Also note that,

$$\Pr\left[t \leftarrow H_n^i \;:\; D(t) = 1\right] = 1/2 \Pr\left[t \leftarrow H_n^{i+1} \;:\; D(t) = 1\right] + 1/2 \Pr\left[t \leftarrow \tilde{H}_n^{i+1} \;:\; D(t) = 1\right]$$

By rearranding the above equation and substituting, we conclude

$$\begin{aligned}
&\Pr\left[t \leftarrow X_n \;:\; A(1^n, t_1 \ldots t_i) = t_{i+1}\right] \\
&\quad = {}^1\!/\!_2 \Pr\left[t \leftarrow H_n^{i+1} \;:\; D(t) = 1\right] \\
&\qquad - \left(\Pr\left[t \leftarrow H_n^{i} \;:\; D(t) = 1\right] - {}^1\!/\!_2 \Pr\left[t \leftarrow H_n^{i+1} \;:\; D(t) = 1\right] - {}^1\!/\!_2\right) \\
&\quad = {}^1\!/\!_2 + \Pr\left[t \leftarrow H_n^{i+1} \;:\; D(t) = 1\right] - \Pr\left[t \leftarrow H_n^{i} \;:\; D(t) = 1\right] \\
&\quad > {}^1\!/\!_2 + {}^1\!/\!_{p(n)l(n)}
\end{aligned}$$

where the last inequality follows from Equation 3.1. This concludes the proof the Theorem 51.1. □

3.3 Pseudo-random generators

We now turn to definitions and constructions of pseudo-random generators.

Definition of a Pseudo-random Generators

Definition 53.1. A function $G : \{0,1\}^* \to \{0,1\}^*$ is a *Pseudo-random Generator (PRG)* if the following holds.

1. (efficiency): G can be computed in PPT.
2. (expansion): $|G(x)| > |x|$
3. (pseudo-randomness): The ensemble $\{x \leftarrow U_n : G(x)\}_{n \in N}$ is pseudo-random.

Constructions of Pseudorandom generators.

First attempt The first attempt to provide a general construction of pseudo-random generators was by Adi Shamir. The construction is as follows:

Definition 53.2 (PRG-Shamir)**.** Start with a one-way permutation f, then define G_{Shamir} as follows: $G(s) = f^n(s) \parallel f^{n-1}(s) \parallel \ldots \parallel f(s) \parallel s$

(The $\parallel$ symbol stands for string concatentation.) The basic idea here to iterate a one-way function, then output, in reverse order, all the intermediary valuers. The insight behind the security of the scheme is that given some prefix of the random sequence, computing the next block is equivalent to inverting the one-way function f.