Problem 1

Two ensembles $X = \{X_k\}_{k \in \mathbb{N}}$ and $Y = \{Y_k\}_{k \in \mathbb{N}}$ are statistically indistinguishable, denoted $X \simeq Y$, if for all positive constants $c$ and sufficiently large $k$,

$$\frac{1}{2} \sum_{\alpha \in \{0, 1\}^k} \left| \Pr[X_k = \alpha] - \Pr[Y_k = \alpha] \right| < \frac{1}{k^c}.$$ 

1. Prove that if $X$ and $Y$ are statistically indistinguishable, then they are computationally indistinguishable.

2. Show that there exist two ensembles $X$ and $Y$ that are computationally indistinguishable but not statistically indistinguishable. (Do not use any computational assumption!)

Solution:

1) We proceed by contradiction. Suppose $X$ and $Y$ are computationally distinguishable, i.e., there exists a family $D = \{D_k\}_{k \in \mathbb{N}}$ of polynomial-size circuits and a positive constant $c$ such that, for infinitely many $k$,

$$\left| \Pr[D(X_k) = 0] - \Pr[D(Y_k) = 0] \right| \geq \frac{1}{k^c}.$$ 

For each $k \in \mathbb{N}$, consider the set

$$A_k = \left\{ \alpha \in \{0, 1\}^k : \Pr[X_k = \alpha] \geq \Pr[Y_k = \alpha] \right\}.$$
We can now compute the following:

$$\sum_{\alpha \in \{0,1\}^k} \left| \Pr[X_k = \alpha] - \Pr[Y_k = \alpha] \right|$$

$$= \sum_{\alpha \in A_k} \left| \Pr[X_k = \alpha] - \Pr[Y_k = \alpha] \right| + \sum_{\alpha \not\in A_k} \left| \Pr[Y_k = \alpha] - \Pr[X_k = \alpha] \right|$$

$$\geq \sum_{\alpha \in A_k} \left| \Pr[X_k = \alpha] - \Pr[Y_k = \alpha] \right| \cdot \Pr[D_k(\alpha) = 0] + \sum_{\alpha \not\in A_k} \left| \Pr[Y_k = \alpha] - \Pr[X_k = \alpha] \right| \cdot \Pr[D_k(\alpha) = 0]$$

$$= \sum_{\alpha \in A_k} \left( \Pr[X_k = \alpha] - \Pr[Y_k = \alpha] \right) \cdot \Pr[D_k(\alpha) = 0] + \sum_{\alpha \not\in A_k} \left( \Pr[Y_k = \alpha] - \Pr[X_k = \alpha] \right) \cdot \Pr[D_k(\alpha) = 0]$$

$$\geq \sum_{\alpha \in A_k} \left( \Pr[X_k = \alpha] - \Pr[Y_k = \alpha] \right) \cdot \Pr[D_k(\alpha) = 0] - \sum_{\alpha \not\in A_k} \left( \Pr[Y_k = \alpha] - \Pr[X_k = \alpha] \right) \cdot \Pr[D_k(\alpha) = 0]$$

$$= \sum_{\alpha \in \{0,1\}^k} \Pr[X_k = \alpha] \cdot \Pr[D_k(\alpha) = 0] - \sum_{\alpha \in \{0,1\}^k} \Pr[Y_k = \alpha] \cdot \Pr[D_k(\alpha) = 0]$$

$$= \Pr[D(X_k) = 0] - \Pr[D(Y_k) = 0]$$

$$\geq \frac{1}{k^{c/2}} ,$$

which is non-negligible, contradicting with the fact that $X$ and $Y$ are statistically indistinguishable. We conclude that $X$ and $Y$ must be computationally indistinguishable.

2) For each $k \in \mathbb{N}$, let $Y_k$ be the uniform distribution over $\{0,1\}^k$. We claim that for all sufficiently large $k$, there exists a random variable $X_k$, with support over a set $S_k$ of at most $2^{k/2}$ $k$-bit strings, such that for every circuit $C_k$ of size $2^{k/8}$, it holds that

$$\left| \Pr[C_k(X_k) = 0] - \Pr[C_k(Y_k) = 0] \right| < 2^{-k/8} \, .$$

(1)

Following this claim, we know that for any family of polynomial-size circuits, the distinguishing gap is at most $2^{-k/8}$, which is negligible. Thus, $X = \{X_k\}_{k \in \mathbb{N}}$ and $Y = \{Y_k\}_{k \in \mathbb{N}}$ are computationally indistinguishable. On the other hand, $X$ and $Y$ are not statistically indistinguishable, because

$$\sum_{\alpha \in \{0,1\}^k} \left| \Pr[X_k = \alpha] - \Pr[Y_k = \alpha] \right| \geq (2^k - 2^{k/2}) \cdot \left| 0 - \frac{1}{2^k} \right| = 1 - \frac{1}{2^{k/2}} > \frac{1}{2} ,$$

which is non-negligible.

Now we prove the above claim. Specifically, we show that if we select uniformly a multi-set of $2^{k/2}$ strings in $\{0,1\}^k$ and let $X_k$ be uniform over this multi-set, then (??) holds with overwhelmingly high probability (over the choices of the multi-set). In particular, there exists a random variable $X_k$ with the required properties.

So let $C_k$ be a circuit with $k$ inputs and let $p_k \equiv \Pr[C_k(Y_k) = 0]$. Independently and uniformly select $2^{k/2}$ strings $s_1, \ldots, s_{2^{k/2}}$ in $\{0,1\}^k$. Define the random variables $z_i$ by the rule $z_i \equiv C_n(s_i)$,
that is, the $z_i$ depend on the random choices of the corresponding $s_i$. Hence, $\Pr[z_i = 0] = p_k$. Using the Chernoff bound, we get that

$$\Pr \left[ \left| p_k - \frac{1}{2^{k/2}} \cdot \sum_{i=1}^{2^{k/2}} z_i \right| \geq 2^{-k/8} \right] \leq 2 \cdot e^{-2 \cdot 2^{k/2} \cdot (2^{-k/8})^2} < 2^{-2^{k/4}}.$$ 

Because there are at most $2^{2^{k/4}}$ different circuits of size $2^{k/8}$, it follows that there exists at least one sequence $s_1, \ldots, s_{2^{k/2}} \in \{0, 1\}^k$ such that for every circuit $C_k$ of size $2^{k/8}$ it holds that

$$\left| p_k - \frac{\sum_{i=1}^{2^{k/2}} C_k(s_i)}{2^{k/2}} \right| < 2^{-k/8},$$

because the fraction of all sequences of $s_i$ such that (3) does not hold for some $C_k$ is less than $2^{-2^{k/4}} \cdot 2^{2^{k/4}} = 1$.

Fixing such a sequence of $s_i$ and letting $X_k$ be distributed uniformly over the elements in that sequence, the claim follows.

Remark. Above, we have used one of the formulations of the Chernoff bound, i.e.,

$$\Pr \left[ \left| \frac{\sum_{i=1}^{n} X_i}{n} - p \right| > \epsilon \right] < 2 \cdot e^{-\epsilon^2 n/(2p(1-p))},$$

where the $X_i$ are independent indicator random variables with $\Pr[X_i = 1] = p$. Other formulations of the Chernoff bound would have also worked in the analysis.

Problem 2

Let $G$ be a pseudorandom generator with expansion factor $\ell$ and let $h$ be any (not necessarily polynomial-time computable) length-preserving permutation over $\{0, 1\}^*$. (The expansion factor of a pseudorandom generator $G$ is a positive polynomial $\ell$ such that $|G(x)| = \ell(k)$ for all $x \in \{0, 1\}^k$ and $k \in \mathbb{N}$.)

1) Is it always the case that $\{s \leftarrow \{0, 1\}^k : h(G(s))\}$ and the uniform distribution over $\{0, 1\}^{\ell(k)}$ are computationally indistinguishable? Is $G'(s) \equiv h(G(s))$ a pseudorandom generator?

2) Is it always the case that $\{s \leftarrow \{0, 1\}^k : G(h(s))\}$ and the uniform distribution over $\{0, 1\}^{\ell(k)}$ are computationally indistinguishable? Is $G'(s) \equiv G(h(s))$ a pseudorandom generator?

3) If you know that $h$ is polynomial-time computable, do your answers to (1) and (2) change?

Solution:

1) It may not be the case that $\{s \leftarrow \{0, 1\}^k : h(G(s))\}$ and the uniform distribution over $\{0, 1\}^{\ell(k)}$ are computationally indistinguishable. To see that, let $\ell(k)$ be injective (e.g., $\ell(k) = k + 1$) and let $h$ be defined as follows:

$$h(x) = \begin{cases} 0^{\ell(|x|) - |x|} s & \text{if } s \text{ is the lexicographically smallest value s.t. } G(s) = x \\ h'_{|x|}(x) & \text{if such an } s \text{ does not exist} \end{cases}$$

where $h'_{|x|}$ is any length-preserving bijection from the remaining domain in $\{0, 1\}^{\ell(|x|)}$ to the remaining range in $\{0, 1\}^{\ell(|x|)}$. 

2-3
For the above choices of $\ell$ and $h$, it is easy to distinguish $\{s \leftarrow \{0,1\}^k : h(G(s))\}$ and the uniform distribution over $\{0,1\}^{\ell(k)}$. Indeed, consider the algorithm $D$ that, on input $(1^k, y)$, does the following:

1. If $y$ starts with $0^{\ell(k)-k}$, output pseudorandom.
2. Otherwise, output random.

On input a string $\sigma$ drawn from $\{s \leftarrow \{0,1\}^k : h(G(s))\}$, $D$ outputs pseudorandom with probability one; on input a string $\sigma$ drawn from the uniform distribution over $\{0,1\}^{\ell(k)}$, $D$ outputs random with probability $1 - 2^{k-\ell(k)}$.

We conclude that $h(G(s))$ is not pseudo-random; in particular, $h(G(s))$ is not a pseudo-random generator.

2) Since $h$ is a length-preserving permutation, $\{s \leftarrow \{0,1\}^k : h(s)\}$ and the uniform distribution over $\{0,1\}^k$ are identical probability distributions. Therefore, $\{s \leftarrow \{0,1\}^k : G(s)\}$ and $\{s \leftarrow \{0,1\}^k : G(h(s))\}$ are identical probability distributions. Moreover, since the first is computationally indistinguishable from the uniform distribution over $\{0,1\}^{\ell(k)}$, so is the second.

Note, however, that $G'(s) \equiv G(h(s))$ is not necessarily a pseudo-random generator: a pseudo-random generator is required to be a deterministic polynomial-time algorithm, but $h$ may not be polynomial-time computable.

3) If $h$ is polynomial-time computable, then both of the previous constructions are pseudo-random generators.

Clearly, both $G(h(s))$ and $h(G(s))$ are polynomial-time computable. Moreover, we have already argued that $\{s \leftarrow \{0,1\}^k : G(h(s))\}$ is pseudo-random (regardless of whether $h$ is polynomial-time computable or not). Hence, we are left to show that $\{s \leftarrow \{0,1\}^{\ell(k)} : h(G(s))\}$ is pseudo-random.

Suppose not, and let $A$ be a successful distinguisher. Then, consider the algorithm $B$ that, on input $(1^k, y)$, outputs $A(1^k, h(y))$. Note that $B$ is polynomial-time, since $h$ is polynomial-time computable. If $y$ is drawn from the uniform distribution over $\{0,1\}^{\ell(k)}$, so is $h(y)$, and $A$ is given a random $\ell(k)$-bit string. If $y$ is drawn from $\{s \leftarrow \{0,1\}^{\ell(k)} : G(s)\}$, then $h(y)$ follows $\{s \leftarrow \{0,1\}^{\ell(k)} : h(G(s))\}$. By assumption, $A$ distinguishes between the uniform distribution over $\{0,1\}^{\ell(k)}$ and $\{s \leftarrow \{0,1\}^{\ell(k)} : h(G(s))\}$, therefore $B$ distinguishes between the uniform distribution over $\{0,1\}^{\ell(k)}$ and $\{s \leftarrow \{0,1\}^{\ell(k)} : G(s)\}$, contradicting the fact that $G$ is a pseudo-random generator.

**Problem 3**

Let $G_1$ and $G_2$ be pseudorandom generators with respective expansion factors $\ell_1$ and $\ell_2$. For each of the candidates below, justify whether the function is a pseudorandom generator or not.

A: $G_A(x) = \text{reverse}(G_1(x))$, where reverse($\cdot$) reverses the bits of its argument.

B: $G_B(x) = G_1(x) \| G_2(x)$.

C: $G_C(x|y) = G_1(x) \| G_2(y)$, where $|x| = |y|$ or $|x| = |y| + 1$.

D: $G_D(x) = G_2(G_1(x))$.

E: $G_E(x) = G_1(x) \oplus (x|0^{\ell_1(|x|)-|x|})$. 

2-4
Solution:

**A) PRG.** The reducibility argument is straightforward. Suppose $G_A$ is not a pseudo-random generator, i.e., there is a PPT algorithm $T$ that successfully distinguishes random outputs of $G_A$ from truly random strings. Then, consider the PPT algorithm $S$ that, on input $(1^k, y)$, outputs $T(1^k, \text{reverse}(y))$. The algorithm $S$ succeeds in distinguishing exactly when $T$ does.

**B) NOT A PRG.** Consider the case where $G_1$ and $G_2$ are the same generator, i.e., $G_1 = G_2$. Then, it is easy to distinguish random outputs of $G_B$ from truly random strings: simply test if the two halves of the input string are equal; this occurs to outputs of $G_B$ with probability one, and to random strings with negligible probability $2^{-\ell_1(k)}$.

**C) PRG.** The reducibility argument involves a simple hybrid argument. Suppose $G_C$ is not a pseudo-random generator, i.e., there is a PPT algorithm $T$ that successfully distinguishes random outputs of $G_C$ from truly random strings for infinitely many string lengths $k$. (Assume without loss of generality that infinitely many of these $k$ are even; otherwise, in the hybrid below we increase $G_i$’s seed length by one.)

Consider the following hybrids:

\[
H_0^k \equiv \{ x \leftarrow \{0,1\}^{\ell_1(k)} \ ; \ y \leftarrow \{0,1\}^{\ell_2(k)} \ ; \ x||y \} ,
\]

\[
H_1^k \equiv \{ s \leftarrow \{0,1\}^k \ ; \ y \leftarrow \{0,1\}^{\ell_2(k)} : G_1(s)||y \} ,
\]

\[
H_2^k \equiv \{ s \leftarrow \{0,1\}^k \ ; \ t \leftarrow \{0,1\}^k : G_1(s)||G_2(t) \} .
\]

Then, define the following probabilities:

\[
p_k^0 = \Pr [ \sigma \leftarrow H_0^k : T(1^k, \sigma) = 0] ,
\]

\[
p_k^1 = \Pr [ \sigma \leftarrow H_1^k : T(1^k, \sigma) = 0] ,
\]

\[
p_k^2 = \Pr [ \sigma \leftarrow H_2^k : T(1^k, \sigma) = 0] .
\]

By assumption, $|p_k^0 - p_k^1|$ is non-negligible. Therefore, by the triangle inequality, at least one of $|p_k^0 - p_k^1|$ or $|p_k^1 - p_k^2|$ is non-negligible.

If $|p_k^0 - p_k^1|$ is non-negligible, then $T$ can be used to break the pseudo-randomness of $G_1$ by using the algorithm $S_1$ that, on input $(1^k, z)$, draws $y$ from $\{0,1\}^{\ell_2(k)}$ and returns $T(1^k, z||y)$. Indeed, if $z$ is truly random, then $T$ is called with a string $\sigma = z||y$ drawn from $H_0^k$; if $z$ is a random output of $G_1$, then $T$ is called with a string $\sigma = z||y$ drawn from $H_1^k$.

If $|p_k^1 - p_k^2|$ is non-negligible, then $T$ can be used to break the pseudo-randomness of $G_2$ by using the algorithm $S_2$ that, on input $(1^k, z)$, draws $s$ from $\{0,1\}^k$ and returns $T(1^k, G_1(s)||z)$. Indeed, if $z$ is truly random, then $T$ is called with a string $\sigma = G_1(s)||z$ drawn from $H_1^k$; if $z$ is a random output of $G_2$, then $T$ is called with a string $\sigma = G_1(s)||z$ drawn from $H_2^k$.

**D) PRG.** The reducibility argument involves a simple hybrid argument. Suppose $G_D$ is not a pseudo-random generator, i.e., there is a PPT algorithm $T$ that successfully distinguishes random outputs of $G_D$ from truly random strings for infinitely many string lengths $k$.

Consider the following hybrids:

\[
H_0^k \equiv \{ x \leftarrow \{0,1\}^{\ell_2(\ell_1(k))} : x \} ,
\]

\[
H_1^k \equiv \{ t \leftarrow \{0,1\}^{\ell_1(k)} : G_2(t) \} ,
\]

\[
H_2^k \equiv \{ s \leftarrow \{0,1\}^k : G_2(G_1(s)) \} .
\]
Then, define the following probabilities:

\[ p_k^0 \equiv \Pr[\sigma \leftarrow H^0_k: T(1^k, \sigma) = 0] , \]
\[ p_k^1 \equiv \Pr[\sigma \leftarrow H^1_k: T(1^k, \sigma) = 0] , \]
\[ p_k^2 \equiv \Pr[\sigma \leftarrow H^2_k: T(1^k, \sigma) = 0] . \]

By assumption, \( |p_k^0 - p_k^2| \) is non-negligible. Therefore, by the triangle inequality, at least one of \( |p_k^0 - p_k^1| \) or \( |p_k^1 - p_k^2| \) is non-negligible.

If \( |p_k^0 - p_k^2| \) is non-negligible, then \( T \) distinguishes between truly random strings of length \( \ell_2(\ell_1(k)) \) and random outputs of \( G_2 \) with seed length \( \ell_1(k) \), for infinitely many \( k \) (and hence, for infinitely many \( \ell_1(k) \)). This contradicts the pseudo-randomness of \( G_2 \).

If \( |p_k^1 - p_k^2| \) is non-negligible, then \( T \) can be used to break the pseudo-randomness of \( G_1 \) by using the algorithm \( S \) that, on input \( (1^k, z) \), outputs \( T(1^k, G_2(z)) \). Indeed, if \( z \) is truly random, then \( T \) is called with a string \( \sigma = G_2(z) \) drawn from \( H^1_k \); if \( z \) is a random output of \( G_1 \), then \( T \) is called with a string \( \sigma = G_2(z) \) drawn from \( H^2_k \).

**E) NOT A PRG.** Consider the case where \( \ell_2(k) = k + 1 \) and \( G_1 \) is defined as follows:

\[ G_1(x||r) = \begin{cases} x||G_2(r) & \text{if } |x| = |r| , \\ G_2(x||r) & \text{otherwise} . \end{cases} \]

Note that \( G_1 \) has expansion factor \( \ell_1(k) = k + 1 \).

We claim that, if \( G_2 \) is pseudo-random, so is \( G_1 \). Suppose not, i.e., there is a PPT algorithm \( T \) that successfully distinguishes random outputs of \( G_1 \) from truly random strings for infinitely many string lengths \( k \). Either \( T \) still works for infinitely many even \( k \) or for infinitely many odd \( k \) (or both).

In the first case, consider the PPT algorithm \( S_{\text{even}} \) that, on input \( (1^k, z) \) with \( |z| = k + 1 \), draws \( x \) from \( \{0, 1\}^k \) and outputs \( T(1^{2k}, x||z) \). If \( z \) is a random \( (k + 1) \)-bit string, then \( x||z \) is a random \( (2k + 1) \)-bit string; if \( z \) is a random \( (k + 1) \)-bit output of \( G_2 \), then \( x||z \) is a random \( (2k + 1) \)-bit output of \( G_1 \). Therefore, \( S_{\text{even}} \) distinguishes exactly when \( T \) distinguishes.

In the second case, consider the PPT algorithm \( S_{\text{odd}} \) that, on input \( (1^k, z) \) with \( |z| = k + 1 \), outputs \( T(1^k, z) \). Since for odd inputs \( G_1 = G_2 \), it is clear that \( S_{\text{odd}} \) distinguishes exactly when \( T \) distinguishes.

In either case, the pseudo-randomness of \( G_2 \) is contradicted, so \( G_1 \) must be pseudo-random.

On the other hand, we now argue that \( G_E \) is not pseudo-random. To see that, note that, for even \( k \)'s and \( |x| = |r| \),

\[ G_E(x||r) = (x||G_2(r)) \oplus (x||r||0) = 0^{k/2}||G_2(r) \oplus (r||0) . \]

Truly random strings and random outputs of \( G_E \) can easily be distinguished by the PPT algorithm \( S \) that, on input \( (1^k, z) \) with \( |z| = k + 1 \) and even \( k \), outputs pseudorandom if \( y \) starts with \( 0^{k/2} \) and random otherwise. Indeed, on input a random output of \( G_E \), \( S \) outputs pseudorandom with probability one; on input a truly random string, \( S \) outputs random with probability \( 1 - 2^{-k/2} \).

**Problem 4**

Let \( \mathcal{F} = \{ F_x \colon \{0, 1\}^k \to \{0, 1\}^k \}_{x \in \{0, 1\}^k} \) be a pseudorandom function. For each of the candidates below, justify whether the function is a pseudorandom function or not.
Solution:

1) NOT PSEUDO-RANDOM. Consider the test $T$ that, on input $1^k$ and with oracle access to $O$ that is equal to $G$ or a truly random function, does the following:

1. Query $O$ at point $0^k$, and obtain answer $a||b$ with $|a| = |b| = k$.
2. Query $O$ at point $0^k = 1^k$, and obtain answer $c||d$ with $|c| = |d| = k$.
3. If $c = b$ and $d = a$, then output pseudorandom.
4. Otherwise, output random.

If $O$ is equal to $G$, then $T$ outputs pseudorandom with probability one. If $O$ equals a truly random function, then $T$ outputs pseudorandom with probability $2^{-2k}$, because $O(1^k)$ would be independent of $O(0^k)$. We conclude that the test $T$ distinguishes between $G$ and a truly random function with non-negligible probability (that is in fact almost 1).

2) NOT PSEUDO-RANDOM. Consider the test $T$ that, on input $1^k$ and with oracle access to $O$ that is equal to $G$ or a truly random function, does the following:

1. Query $O$ at some arbitrary point $x^k$, and obtain answer $a||b$ with $|a| = |b| = k$.
2. Compute $F_{0^k}(x)$ locally. (Note that this can be done by simply using the evaluator for $F$!)
3. If $a = F_{0^k}(x)$, then output pseudorandom.
4. Otherwise, output random.

If $O$ is equal to $G$, then $T$ outputs pseudorandom with probability one. If $O$ equals a truly random function, then $T$ outputs pseudorandom with probability $2^{-k}$. We conclude that the test $T$ distinguishes between $G$ and a truly random function with non-negligible probability (that is in fact almost 1).

3) PSEUDO-RANDOM. The proof is by contradiction, via a hybrid argument. Consider the following hybrid function families:

$$G_0^0(x) = F_{F_s(0^k)}(x) || F_{F_s(1^k)}(x)$$
$$G_1^1(x) = F_u(x) || F_v(x)$$
$$G_2^2(x) = S_1(x) || F_s(x)$$
$$G_3^3(x) = S_1(x) || S_2(x)$$
In the above, $u$ and $v$ are random seeds that are independent of each other, and the $\delta_i$ represent truly-random functions. Note that $G^1$ is equal to $G$, and $G^3$ is equal to a truly random function. So suppose by contradiction that $G$ is not pseudo-random, so that there exists some test $T$ that distinguishes between $G$ and a truly random function with non-negligible probability. By the triangle inequality, there must exist some $i \in \{0, 1, 2\}$ such that $T$ distinguishes between $G^i$ and $G^{i+1}$ with non-negligible probability. We show that for any choice of $i$, this leads to a contradiction. (Actually, we show that only for $i = 0$, because that is the challenging case; the cases $i = 1$ and $i = 2$ are straightforward.)

Suppose $T$ distinguishes between $G^0$ and $G^1$ with non-negligible probability. Consider the test $S$ that, on input $1^k$ and with oracle access to $O$ that is equal to $F$ or a truly random function, does the following:

1. Query $O$ at point $0^k$, and obtain answer $u$.
2. Query $O$ at point $1^k$, and obtain answer $v$.
3. Run $T$, and when $T$ makes a query $x$, reply with $F_u(x)||F_v(x)$.
4. Output the output of $T$.

If $O$ is equal to $F$, then $S$ gives answers to $T$ exactly as if $T$ were interacting with $G^0$. If $O$ is equal to a truly random function, then $u$ and $v$ are random and independent, and $S$ gives answers to $T$ exactly as if $T$ were interacting with $G^1$. Therefore, $S$ distinguishes between $F$ and a truly random function with non-negligible probability, which is a contradiction.

4) NOT PSEUDO-RANDOM. Let $H = \{H_s: \{0, 1\}^k \rightarrow \{0, 1\}^k\}_{s \in \{0, 1\}^k}$ be a pseudo-random function family. Define the function family $F$ as follows:

$$F_s(x) = \begin{cases} 0^k & \text{if } s = 0^k \\ H_s(x) & \text{otherwise} \end{cases}$$

We claim that the function family $F$ is pseudo-random. Indeed, the event $s = 0^k$ occurs with probability $2^{-k}$, and whenever $s \neq 2^{-k}$ then $F_s$ is equal to $H_s$. Hence, for any test $T$, the view of $T^F$ is statistically close to the view of $T^H$. It follows that $F$ and $H$ are indistinguishable by any test $T$, and we conclude that $F$ is pseudo-random.

On the other hand, note that $G$ is not pseudo-random. Consider the test $T$ that, on input $1^k$ and with oracle access to $O$ that is equal to $G$ or a truly random function, does the following:

1. Query $O$ at point $0^k$, and obtain answer $z$.
2. If $z = 0^k$, then output pseudorandom.
3. Otherwise, output random.

If $O$ is equal to $G$, then $T$ outputs pseudorandom with probability one, because $z = G_s(0^k) = F_{0^k}(s) = 0^k$. If $O$ equals a truly random function, then $T$ outputs pseudorandom with probability $2^{-k}$. We conclude that the test $T$ distinguishes between $G$ and a truly random function with non-negligible probability (that is in fact almost 1).

5) NOT PSEUDO-RANDOM. Let $H = \{H_s: \{0, 1\}^{2k} \rightarrow \{0, 1\}^{2k}\}_{s \in \{0, 1\}^k}$ be a pseudo-random function family. (Assuming pseudo-random function families exist, a pseudo-random function family
such as \( \mathcal{H} \) is guaranteed to exist by an argument similar to Part (3) above.) Define the function family \( \mathcal{F} \) as follows: for even-length seeds \( s \) with \( s = s_1|s_2 \) and \( |s_1| = |s_2| \),

\[
F_s(x) = \begin{cases} 
  s_1 | \text{half}(H_{s_2}(0^k)) & \text{if } x = 0^k \\
  H_{s_2}(x) & \text{otherwise},
\end{cases}
\]

and \( \text{half}(y) \) denotes the second half of string \( y \); for odd-length seeds \( s \), simply define \( F_s \) to be equal to some pseudo-random function family with \( k \)-bit seeds, inputs, and outputs.

We claim that the function family \( \mathcal{F} \) is pseudo-random. Suppose not, i.e., there is a test \( T \) that distinguishes between \( \mathcal{F} \) and a truly random function with non-negligible probability. Either \( T \) distinguishes between the two for even-length seeds or for odd-length seeds (or both). In the latter case, we get a contradiction because \( \mathcal{F} \) is exactly equal to a pseudo-random function family on odd-length seeds. So let us consider the even-length seeds case. Consider the test \( S \) that, on input \( 1^{2k} \) and with oracle access to \( O \) that is equal to \( \mathcal{H} \) or a truly random function, does the following:

1. \( s_1 \leftarrow \{0,1\}^k \).
2. Run \( T \), and when \( T \) makes a query \( x \), check if \( x = 0^{2k} \); if so, reply \( s_1 | \text{half}(O(0^{2k})) \), otherwise, reply \( O(x) \).
3. Output the output of \( T \).

If \( O \) is equal to \( \mathcal{H} \), then \( S \) gives answers to \( T \) exactly as if \( T \) were interacting with \( \mathcal{F} \). If \( O \) is equal to a truly random function, then \( S \) is just replacing the first half of \( O(0^k) \) with fresh random bits, and \( S \) gives answers to \( T \) exactly as if \( T \) were interacting with a truly random function. Therefore, the distinguishing advantage of \( S \) on input \( 1^{2k} \) is the distinguishing advantage of \( T \) on input \( 1^k \), which is non-negligible in \( k \); thus, \( S \) distinguishes between \( \mathcal{H} \) and a truly random function with non-negligible probability, which is a contradiction.

On the other hand, note that \( \mathcal{G} \) is not pseudo-random. For all even-length seeds \( s = s_1|s_2 \) with \( |s_1| = |s_2| \),

\[
G_s(0^{2k}) = F_s(0^{2k}) \oplus (s_1|s_2) = \left( s_1 | \text{half}(H_{s_2}(0^{2k})) \right) \oplus (s_1|s_2) = 0^k | z ,
\]

for some \( k \)-bit string \( z \). The distinguishing test is then obvious: query the oracle on point \( 0^{2k} \), and see if the result starts with \( 0^k \).

6) PSEUDO-RANDOM. The proof is by contradiction, via a hybrid argument. Consider the following hybrid function families:

\[
\begin{align*}
G_{s_1,s_2}(x) &= (F_{s_1}(x) \oplus s_2)||F_{s_2}(x) \\
H_{s_2}(x) &= s_1(x)||F_{s_2}(x) \\
R(x) &= s_1(x)||s_2(x)
\end{align*}
\]

In the above, \( s_1 \) and \( s_2 \) are random seeds that are independent of each other, and the ‘\( s_1 \)’ represent truly-random functions.

So suppose by contradiction that \( \mathcal{G} \) is not pseudo-random, so that there exists some test \( T \) that distinguishes between \( \mathcal{G} \) and a truly random function with non-negligible probability. By the triangle inequality, either \( T \) distinguishes between \( \mathcal{G} \) (the first hybrid) and the middle hybrid, or between the middle hybrid and a truly random function (the third hybrid), or both. We show that either case leads to a contradiction.
Suppose that $T$ distinguishes between the first hybrid and the middle hybrid. Consider the statistical test $S$ that, on input $1^k$ and using $T$ as a subroutine, does the following:

1. $s_2 \leftarrow \{0,1\}^k$.
2. Run $T$, and when $T$ makes a query $x$, query $O$ on input $x$ to obtain answer $z$, and reply $(z \oplus s_2)||F_{s_2}(x)$ to $T$.
3. Output the output of $T$.

If $O$ is equal to $F$, then $S$ gives answers to $T$ exactly as if $T$ were interacting with $G$ (the first hybrid). If $O$ is equal to a truly random function, then $z \oplus s_2$ is a truly random value for each query, and so $S$ gives answers to $T$ exactly as if $T$ were interacting with the middle hybrid. Therefore, the distinguishing advantage of $S$ on input $1^k$ is the distinguishing advantage of $T$ on input $1^k$, which is non-negligible in $k$; thus, $S$ distinguishes between $F$ and a truly random function with non-negligible probability, which is a contradiction.

Suppose that $T$ distinguishes between the middle hybrid and the third hybrid. The reduction in this case is straightforward, and leads to a contradiction of the pseudo-randomness of $F$.

**Remark.** The other obvious choice of hybrid does not work. If we define the middle hybrid as:

$$H_{s_1,s_2}(x) = (F_{s_1}(x) \oplus s_2)||S(x),$$

the reduction between the first and middle hybrid does not go through. The reason is that, given an oracle that is equal to either $F_{s_2}$ or a truly random function, we cannot prepend the correct value of $(F_{s_1}(x) \oplus s_2)$ to the oracle’s responses, because we do not know what $s_2$ is.

7) **PSEUDO-RANDOM.** Define the function family $\mathcal{H} = \{H_S\}_{S \in \mathcal{U}}$ by the rule:

$$H(x) = F_{S(x)}(x),$$

where $S$ denotes a truly-random function. It is easy to show that no statistical test can distinguish between $\mathcal{H}$ and $\mathcal{G}$. Hence, we are left to show that $\mathcal{H}$ is pseudo-random.

Suppose not, i.e., there is a statistical test $T$ that distinguishes between $\mathcal{H}$ and truly random functions. Without loss of generality, assume that $T$ never queries the same input more than once (as $T$ knows that it will get the same answer). Let $p$ be a polynomial such that the number of queries made by $T(1^k)$ is at most $p(k)$. By a hybrid argument, for each $k$, there exists a $j$ (depending on $k$) in $\{0, \ldots, p(k) - 1\}$ such that $T$ distinguishes two experiments: in the first experiment, the first $j$ queries of $T$ are answered according to $F_{S_1}(x)$, and the remaining queries are answered according to a truly random function $S_2$; in the second experiment, the first $j + 1$ queries of $T$ are answered according to $F_{S_1}(x)$ and the remaining queries are answered according to a truly random function $S_2$. (Note that the experiments corresponding to $j = 0$ and $j = p(k)$ correspond to the original experiments, i.e., answering all queries according to $\mathcal{H}$ or a truly random function, respectively.)

Consider the statistical test $S$ that, on input $1^k$ and using $T$ as a subroutine, does the following:

1. $j \leftarrow \{0, \ldots, p(k) - 1\}$, where $p(k)$.
2. Run $T$, and when $T$ asks its $i$-th query $x_i$:
   
   (a) $i < j + 1$: compute $s_i = S_1(x_i)$ and reply with $F_{s_i}(x_i)$.
   
   (b) $i = j + 1$: query the oracle $O$ on input $x_i$, and reply with $O(x_i)$. 

2-10
In the above, “computing” $s_1$ (resp., $s_2$) means toss $k$ coins to get a random $k$-bit string.

With probability $1/p(k)$, $S$ guesses a “good” $j$ for $k$. Now condition on $S$ guessing a good $j$. If $O$ is equal to $\mathcal{H}$, then $S$ gives answers to $T$ exactly as if $T$ were interacting with the $j$-th hybrid. If $O$ is equal to a truly random function, then $S$ gives answers to $T$ exactly as if $T$ were interacting with the $(j+1)$-th hybrid. Therefore $S$ distinguishes between $\mathcal{H}$ and a truly random function, as desired.

Problem 5

In this problem we consider two other ways of modeling what it means to be a pseudorandom function family, and investigate how these new definitions compare to the one we discussed.

1. In the definition of a PRF, we allow for an adversary to adaptively query its oracle in order to distinguish whether the oracle is truly random or pseudorandom. Suppose we now consider non-adaptive tests: an adversary provides a list of testing points, then receives the values of the oracle at each of those testing points, and finally makes a decision (without consulting the oracle again).

**Definition 1 (Non-Adaptive Pseudo-Random Function Families)** A function family is non-adaptively pseudorandom if a random member of the family is indistinguishable from a random function, under all polynomial-time non-adaptive tests.

Is the above definition of PRF strictly stronger, strictly weaker, equivalent, or incomparable to our original adaptive notion? Prove your answer.

2. Now we consider a different kind of test in which we see if not being able to predict an output of a function is equivalent to the function seeming random. A predictor is allowed to adaptively query the oracle on several points, and then outputs a pair $(x, y)$. The predictor succeeds in the test if: (1) it has not already queried the oracle on point $x$, and (2) the value of the oracle at $x$ is equal to $y$.

**Definition 2 (Unpredictable Function Family)** A function family is unpredictable if no polynomial-time oracle machine can succeed in the prediction experiment with non-negligible advantage over random guessing.

Is the above definition of PRF strictly stronger, strictly weaker, equivalent, or incomparable to our original adaptive notion? Prove your answer.

**Solution:**

1) Non-adaptive pseudo-randomness is strictly weaker than adaptive pseudo-randomness. (Of course, assuming non-adaptive pseudorandom function families exist at all.)

First, note that any non-adaptive test can be written as an adaptive test, just by getting the list of points to query and then querying them individually. Therefore non-adaptive pseudo-randomness is no stronger than adaptive pseudo-randomness.
Next, we construct a function family that is non-adaptively pseudo-random, but not adaptively pseudo-random. Suppose \( F = \{F_s : \{0,1\}^k \rightarrow \{0,1\}^k\}_{s \in \{0,1\}^k} \) is a non-adaptively pseudorandom function family. Define \( G = \{G_s : \{0,1\}^k \rightarrow \{0,1\}^k\}_{s \in \{0,1\}^k} \) as follows: for all strings \( s \),

\[
G_s(x) := \begin{cases} 
0^k & \text{if } x = F_s(0^k) \\
F_s(x) & \text{otherwise}
\end{cases}
\]

The function \( G_s \) is well-defined for all strings \( s \): if it happens that \( F_s(0^k) = 0^k \), then \( G_s \) is exactly \( F_s \); otherwise it differs at only one point. Moreover, given \( s \) and \( x \), \( G_s \) is poly-time computable, because \( F_s \) is polynomial-time computable and the check on \( x \) is efficient.

We claim that \( G \) is non-adaptively pseudorandom. Suppose not, i.e., there exists a non-adaptive test \( T_G \) that distinguishes \( G \) from truly random functions. Consider the non-adaptive test \( T_F \) that, with oracle access to \( O \) and using \( T_G \) as a subroutine, does the following:

1. Run \( T_G \) and get the list \( x_1, \ldots, x_\ell \) of its queries.
2. Query the oracle \( O \) on the point \( 0^k \), and receive \( z \) as answer.
3. If \( z \) appears in the list \( x_1, \ldots, x_\ell \), output \texttt{pseudorandom}; otherwise output \texttt{random}.

The probability that the set of queries \( x_1, \ldots, x_\ell \) of \( T_G(1^k)^G \) contains \( z = F_s(0^k) \) is non-negligible (when taking the probability over \( s \in \{0,1\}^k \) and the coin tosses of \( T \) ), because if that were not the case we could use \( T_G \) to (non-adaptively) distinguish between \( F \) and truly random functions (without querying \( F_s(0^k) \), the views of \( T_G(1^k)^G \) and \( T_G(1^k)^F \) are equal), which is a contradiction.

Hence, if \( O \) is equal to \( F \), then \( T_F \) outputs \texttt{pseudorandom} with non-negligible probability. On the other hand, if \( O \) is equal to a truly random function, \( z \) is truly random and independent of all of \( T \)'s queries, and hence \( S \) outputs \texttt{pseudorandom} with negligible probability. We conclude that \( S \) (non-adaptively) distinguishes between \( F \) and truly random functions, which is a contradiction.

On the other hand, \( G \) is clearly \texttt{not} adaptively pseudo-random. Indeed, an adaptive test \( T \) may do the following:

1. Query the oracle \( O \) on the point \( 0^k \), and receive \( z \) as answer.
2. Query the oracle \( O \) on the point \( z \), and receive \( y \) as answer.
3. If \( y = 0^k \), output \texttt{pseudorandom}; otherwise output \texttt{random}.

Clearly, if \( O \) equals \( G_s \), then \( T^O \) always outputs \texttt{pseudorandom}. However, if \( O \) is a truly random function, then \( T^O \) outputs \texttt{pseudorandom} only if \( O(0^k) = 0^k \) or \( O(z \neq 0^k) = 0^k \); this occurs with probability at most \( 2 \cdot 2^{-k} \), which is negligible. We conclude that the adaptive test \( T \) distinguishes \( G \) from truly random functions.

2) Unpredictability is \textbf{strictly weaker} than adaptive pseudo-randomness. (Of course, assuming unpredictable function families exist at all.)

First, we argue that pseudo-randomness implies unpredictability, and we do so by contradiction. Suppose that \( F \) is a predictable function family and let \( P \) be a predictor for \( F \). Consider the adaptive test \( T \) that, with oracle access to \( O \) and using \( P \) as a subroutine, does the following:

1. Run \( P \) and, whenever \( P \) makes a query, pass the query to \( O \) and send back the result to \( P \).
2. When \( P \) outputs a pair \((x,y)\) such that \( x \) has not yet been queried, query \( O \) on \( x \) and test whether the result equals \( y \).

3. If so, output \textit{pseudorandom}; otherwise output \textit{random}.

If the oracle \( O \) equals \( F \), then \( T \) outputs \textit{pseudorandom} with non-negligible probability; this follows by the assumption on the predictor \( P \). If the oracle \( O \) is a truly random function, then \( P \) is correct only with probability \( 2^{-k} \), because \( x \) has not been queried. We conclude that the adaptive test \( T \) is a distinguisher, and hence \( F \) is not pseudo-random. (We remark that this argument works even if the function family \( F \) has a 1-bit output instead of a \( k \)-bit output, by replacing \( 2^{-k} \) above with \( 1/2 \).)

Next, we construct a function family that is unpredictable, but \textit{not} adaptively pseudo-random. Suppose \( \mathcal{F} = \{F_s : \{0,1\}^k \to \{0,1\}^k\}_{s \in \{0,1\}^k} \) is an unpredictable function family. Define \( \mathcal{G} = \{G_s : \{0,1\}^k \to \{0,1\}^k\}_{s \in \{0,1\}^k} \) as follows: for all strings \( s \), \( G_s(x) := 1 || F_s(x) \).

We claim that \( \mathcal{G} \) is unpredictable. Suppose not, i.e., there exists a predictor \( P_{\mathcal{G}} \) that succeeds in the prediction game for \( \mathcal{G} \) with non-negligible probability. Without loss of generality, we can assume that the predicted value of \( P \) always begins with a 1, because the actual value of \( \mathcal{G} \) does. Now consider the predictor \( P_{\mathcal{F}} \) that, on input \( 1^k \) and with oracle access to \( F_s \) for some \( s \in \{0,1\}^k \), does the following:

1. Run \( T \), and when \( T \) queries \( x \), query \( x \) to obtain \( F_s(x) \), and reply with \( 1 || F_s(x) \) (which is equal to \( G_s(x) \)).

2. When \( T \) outputs the prediction \((a,1||b)\), output \((a,b)\).

By the assumption on \( P_{\mathcal{G}} \), with non-negligible probability \((a,1||b)\) is a correct prediction for \( G_s \), and therefore with non-negligible probability \((a,b)\) is a correct prediction for \( F_s \). Thus, \( P_{\mathcal{F}} \) succeeds in the prediction game for \( \mathcal{F} \) with non-negligible probability, which is a contradiction.

On the other hand, \( \mathcal{G} \) is clearly \textit{not} adaptively pseudo-random: the first bit of its output on any \( x \) is always 1!