Answer to Question 1.

Suppose that $F'$ is not pseudo-random (against unrestricted adversaries). We will show how to break the pseudo-randomness of $F$ with a restricted adversary; we will either break the inner application of $F$ or the outer application.

Assuming the nonuniform model, let $D = \{D_n\}$ be an adversary for distinguishing $F'$ and define $p_D(n)$ and $r_D(n)$ in the usual way. Say (without loss of generality) that $p_D(n) - r_D(n) > 1/n^c$ for infinitely many $n$. Consider such an $n$.

Now consider the following additional experiment. Choose a random function $f$ that maps $\{0, 1\}^n$ to $\{0, 1\}^n$; then define function $f' : \{0, 1\}^* \rightarrow \{0, 1\}^n$ as follows: for $x \in \{0, 1\}^*$, $f'(x) = F_{f(\bar{\ell})}(x)$, where $\bar{\ell}$ is the $n$-bit representation of the length of $x$; run $D_n$ with its function gates interpreted as $f'$.

Define $q_D(n)$ to be the probability that $D_n$ accepts.

Then at least one of the following must hold:

- $p_D(n) - q_D(n) > \frac{1}{2n^c}$
- $q_D(n) - r_D(n) > \frac{1}{2n^c}$

This gives rise to the following two cases:

Case 1: $p_D(n) - q_D(n) > \frac{1}{2n^c}$.

We will describe a circuit $C_n$ for distinguishing $F$. $C_n$ will be restricted to only make queries of length $n$, so we will view $C_n$ as operating on a function $f : \{0, 1\}^n \rightarrow \{0, 1\}^n$. (That is, its function gates are interpreted as $f$.) $C_n$ will run $D_n$ with $D_n$'s function gates interpreted as $f'$ where $f'(x) = F_{f(\bar{\ell})}(x)$ where $\bar{\ell}$ is the $n$-bit representation of the length of $x$, and $C_n$ will accept iff $D_n$ accepts.

Let $p_C(n)$ be the probability that if $k$ is randomly chosen from $\{0, 1\}^n$ and $C_n$ is run with its function gates interpreted as $F_k$, then $C_n$ accepts. Clearly, $p_C(n) = p_D(n)$.

Let $r_C(n)$ be the probability that if $f$ is a randomly chosen function mapping $\{0, 1\}^n$ to $\{0, 1\}^n$ and $C_n$ is run with its function gates interpreted as $f$, then $C_n$ accepts. Clearly, $r_C(n) = q_D(n)$.

Then $p_C(n) - r_C(n) > \frac{1}{2n^c}$.

Case 2: $q_D(n) - r_D(n) > \frac{1}{2n^c}$.

Since $\{D_n\}$ is a polynomial-size family, there must exist a polynomial $l(n)$ such that for all $n$, each query made by $D_n$ to its function gates is no longer than $l(n)$ bits. Assume $n$ is sufficiently large so that $l(n) < 2^n$ (so that every length $\leq l(n)$ has a unique $n$ bit representation).

We begin by describing a sequence of experiments that are hybrids between the experiment that gives rise to $r_D(n)$ and the experiment that gives rise to $q_D(n)$. The idea for the $i$-th experiment...
is that if \(|x| \leq i\), we will use a truly random key \(s_{|x|}\) depending on the length of \(x\) and compute \(F_{s_{|x|}}(x)\); we will compute a random function of \(x\) otherwise.

So for \(-1 \leq i \leq l(n)\), consider the following experiment:

Select \(i + 1\) strings \(s_0, s_1, ..., s_i\) randomly from \(\{0, 1\}^n\); select a random function \(f\) that maps strings of length \(\leq l(n)\) to \(\{0, 1\}^n\); then define function \(f'\) that maps strings of length \(\leq l(n)\) to \(\{0, 1\}^n\) as follows:

\[
\begin{align*}
\text{for } x \in \{0, 1\}^*, \text{ if } |x| \leq i \text{ then } f'(x) &= F_{s_{|x|}}(x), \text{ and if } |x| > i \text{ then } f'(x) = f(x); \\
\text{run } D_n \text{ with its function gates interpreted as } f'; \\
\text{define } q_i(n) \text{ to be the probability that } D_n \text{ accepts.}
\end{align*}
\]

Clearly, \(q_{-1}(n) = r_D(n)\). Also, note that the experiment that gives rise to \(q_{l(n)}(n)\) is equivalent to the experiment that gives rise to \(q_D(n)\) (the main idea is that for each input length we randomly select a seed for \(F\)), and hence \(q_{l(n)}(n) = q_D(n)\). So there exists an \(i, -1 \leq i < l(n)\), such that \(q_{i+1} - q_i > \frac{1}{2(l(n)+1)w}\); fix such an \(i\).

We now describe a probabilistic circuit \(C_n\) for distinguishing \(F\), such that \(C_n\) is restricted to having all of its function queries the same length, namely \(i + 1\). The function gates of \(C_n\) are interpreted as some function \(g : \{0, 1\}^{i+1} \rightarrow \{0, 1\}^n\). \(C_n\) will choose \(i + 1\) strings \(s_0, s_1, ..., s_i\) randomly from \(\{0, 1\}^n\). \(C_n\) will then simulate \(D_n\). However, \(C_n\) will handle \(D_n\)'s function gate queries in a special way, depending on the length of the query. When the query \(x\) is of length \(\leq i\), \(C_n\) will use \(F_{s_{|x|}}(x)\) as the result of the query. When the query \(x\) is of length \(i + 1\), \(C_n\) will use one of its own function gates to answer the query, that is, the result of the query will be \(g(x)\). Finally, for each new query of length \(> i + 1\), \(C_n\) will select a string randomly from \(\{0, 1\}^n\) and use this string as the result of the query. \(C_n\) will accept iff \(D_n\) accepts. It is important to note that only queries of length \(i + 1\) are made to \(C_n\)'s function gates, so all queries to \(g\) are of this same length.

Let \(p_C(n)\) be the probability that if \(k\) is randomly chosen from \(\{0, 1\}^n\) and \(C_n\) is run with its function gates interpreted as \(F_k\), then \(C_n\) accepts. Clearly, \(p_C(n) = q_{i+1}(n)\).

Let \(r_C(n)\) be the probability that if \(g\) is a randomly chosen function mapping \(\{0, 1\}^{i+1}\) to \(\{0, 1\}^n\) and \(C_n\) is run with its function gates interpreted as \(g\), then \(C_n\) accepts. Clearly, \(r_C(n) = q_i(n)\).

Then, \(p_C(n) - r_C(n) > \frac{1}{2(l(n)+1)w}\). By appropriately fixing the random bits of \(C_n\) we can make a normal, deterministic circuit \(C'_n\) that does at least as well as \(C_n\) as an adversary against \(F\).

**Answer to Question 2.** The main idea we will use is that for all \(k\), we have \(F'_k(\overline{0}F'_k(\lambda)) = F'_k(2F'_k(0F'_k(\lambda)))\). However, for a random function \(f\), it is extremely unlikely that we will have \(f(\overline{0}f(\lambda)) = f(2f(\overline{0}f(\lambda)))\).

We begin by showing that for all \(n\), if \(k\) is an \(n\) bit string then \(F'_k(\overline{0}F'_k(\lambda)) = F'_k(2F'_k(0F'_k(\lambda)))\). So let \(k \in \{0, 1\}^n\).

We have:

\[
\begin{align*}
F'_k(\overline{0} F'_k(\lambda)) &= F'_k(\overline{0} \text{ CBC}(\lambda)) \\
&= F'_k(\overline{0} F_k(F_k(\overline{0}))) \\
&= \text{CBC}(\overline{0} F_k(F_k(\overline{0}))) 2 \\
&= F_k(2 \oplus F_k(F_k(F_k(\overline{0}))) \oplus F_k(\overline{0} \oplus F_k(\overline{0}))) \\
&= F_k(2 \oplus F_k(F_k(F_k(\overline{0}))) \oplus F_k(F_k(\overline{0}))) \\
&= F_k(2 \oplus F_k(\overline{0}))
\end{align*}
\]
We also have:
\[
F'_k(2F'_k(0F'_k(\lambda))) = F'_k(2F_k(2 \oplus F_k(0)))
\]
\[
= CBC(2F_k(2 \oplus F_k(0)) \oplus 2)
\]
\[
= F_k(2 \oplus F_k(F_k(2 \oplus F_k(0)) \oplus F_k(2 \oplus F_k(0))))
\]
\[
= F_k(2 \oplus F_k(0))
\]

Therefore, we have \(F'_k(0F'_k(\lambda)) = F'_k(2F'_k(0F'_k(\lambda)))\).

Now, fix \(n\). We will describe a circuit \(D_n\) for distinguishing \(F'\). The only inputs to \(D_n\) will be the constants 0 and 1. The “function” gates of \(D_n\) are interpreted as some function \(f : \{0, 1\}^n \to \{0, 1\}^n\). \(D_n\) will accept if \(f(0f(\lambda)) = f(2f(0f(\lambda)))\), and \(D_n\) will reject otherwise.

Let \(p_D(n)\) be the probability that if \(k\) is randomly chosen from \(\{0, 1\}^n\) and \(D_n\) is run with its “function” gates interpreted as \(F'_k\), then \(D_n\) accepts. Clearly, \(p_D(n) = 1\).

Let \(r_D(n)\) be the probability that if \(f\) is a randomly chosen function mapping \(\{0, 1\}^n\) to \(\{0, 1\}^n\) and \(D_n\) is run with its “function” gates interpreted as \(f\), then \(D_n\) accepts. Note that if \(f\) is a randomly chosen function mapping \(\{0, 1\}^n\) to \(\{0, 1\}^n\), the probability that \(f(0f(\lambda)) = f(2f(0f(\lambda)))\) is 1/2\(^n\). This means that \(r_D(n) = 1/2^n\).

**Answer to Question 3.** The main idea is that if we have an adversary that breaks the pseudo-randomness of \(F'\), then either

- this adversary can distinguish between \(F_{k_2} \circ H_{k_1}\) and \(f \circ H_{k_1}\) (where \(k_1\) and \(k_2\) are randomly chosen \(n\)-bit strings, and \(f\) is a randomly chosen function mapping \(\{0, 1\}^n\) to \(\{0, 1\}^n\)),
- OR this adversary can distinguish between \(f \circ H_{k_1}\) and \(f'\) (where \(f'\) is a randomly chosen function mapping \(\{0, 1\}^n\) to \(\{0, 1\}^n\)).

If the adversary can distinguish between \(F_{k_2} \circ H_{k_1}\) and \(f \circ H_{k_1}\), it is straightforward to show how to break the pseudo-randomness of \(F\).

If the adversary can distinguish between \(f \circ H_{k_1}\) and \(f'\), we show that when the adversary is run with its “function” gates interpreted as \(f \circ H_{k_1}\) (for randomly chosen \(f\) and \(k_1\)), it must, with non-negligible probability, make at least one pair of queries (to its “function” gates) that collide under \(H_{k_1}\). We then show that even if the adversary’s queries are answered with randomly chosen values, it still must, with non-negligible probability, make at least one pair of queries that collide under \(H_{k_1}\) for randomly chosen \(k_1\). Extending this idea, we show that there exists a polynomial-size fixed set of strings \(Q\) such that for randomly chosen \(k_1\) we will have, with non-negligible probability, that some pair of strings in \(Q\) collide under \(H_{k_1}\). Finally, we show that some fixed pair of strings in \(Q\) collide with non-negligible probability under \(H_{k_1}\) for randomly chosen \(k_1\), and this is a violation of the security property of \(H\).

We now do the proof more formally. Suppose that \(F'\) is not pseudo-random. We will show that either \(F\) is not pseudo-random or \(H\) is not a privately collision resistant hash family.

Let \(\{D_n\}\) be an adversary for distinguishing \(F'\); For each \(n\), define
\(p_D(n) = \) the probability that if \(k_1\) and \(k_2\) are each randomly chosen from \(\{0, 1\}^n\) and \(D_n\) is run with its “function” gates interpreted as \(F'_{k_1k_2}\), then \(D_n\) accepts;
\(r_D(n) = \) the probability that if \(f\) is a randomly chosen function mapping \(\{0, 1\}^*\) to \(\{0, 1\}^n\) (what
we mean here is that the result of each new query to \( f \) is randomly chosen from \( \{0,1\}^n \) and \( D_n \) is run with its “function” gates interpreted as \( f \), then \( D_n \) accepts.

Furthermore, for each \( n \), consider the following experiment, which we will refer to as Experiment 1: Choose a random function \( f \) that maps \( \{0,1\}^n \) to \( \{0,1\}^n \). Choose a string \( k_1 \) randomly from \( \{0,1\}^n \). Then, define function \( f' : \{0,1\}^* \to \{0,1\}^n \) as follows: for \( x \in \{0,1\}^* \), \( f'(x) = f(H_{k_1}(x)) \). Run \( D_n \) with its “function” gates interpreted as \( f' \). Define \( q_D(n) \) to be the probability that \( D_n \) accepts.

Then \( |p_D(n) - r_D(n)| > 1/n^c \) for some \( c \) and infinitely many \( n \). Say without loss of generality that \( p_D(n) - r_D(n) > 1/n^c \) for some \( c \) and infinitely many \( n \).

This gives rise to the following two cases:

**Case 1**: \( p_D(n) - q_D(n) > \frac{1}{2n^c} \) for infinitely many \( n \).

Fix \( n \) and say that \( p_D(n) - q_D(n) > \frac{1}{2n^c} \). We will describe a circuit \( C_n \) for distinguishing \( F \). The “function” gates of \( C_n \) are interpreted as some function \( f : \{0,1\}^n \to \{0,1\}^n \). \( C_n \) will choose a string \( k_1 \) randomly from \( \{0,1\}^n \). Then, define function \( f' : \{0,1\}^* \to \{0,1\}^n \) as follows: for \( x \in \{0,1\}^* \), \( f'(x) = f(H_{k_1}(x)) \). \( C_n \) will run \( D_n \) with \( D_n \)'s “function” gates interpreted as \( f' \), and \( C_n \) will accept iff \( D_n \) accepts.

Let \( p_C(n) \) and \( r_C(n) \) be as in the definition of pseudo-random function generator. Then \( p_C(n) = p_D(n) \) and \( r_C(n) = q_D(n) \). Then \( p_C(n) - r_C(n) > \frac{1}{2n^c} \).

**Case 2**: \( q_D(n) - r_D(n) > \frac{1}{2n^c} \) for infinitely many \( n \).

Consider Experiment 1 again. Define \( s_D(n) \) to be the probability that there exist queries \( x \) and \( y \), made by \( D_n \) to its “function” gates, such that \( x \neq y \) but \( H_{k_1}(x) = H_{k_1}(y) \). That is, \( s_D(n) \) is the probability that some pair of distinct queries made by \( D_n \) collides under \( H_{k_1} \). Define \( a_D(n) \) to be the probability that \( D_n \) accepts given that some pair of distinct queries made by \( D_n \) collides under \( H_{k_1} \). Similarly, define \( b_D(n) \) to be the probability that \( D_n \) accepts given that no pair of distinct queries made by \( D_n \) collides under \( H_{k_1} \). Then, \( q_D(n) = a_D(n) \cdot s_D(n) + b_D(n) \cdot (1 - s_D(n)) \).

Now, note that when no pair of distinct queries made by \( D_n \) collides under \( H_{k_1} \), the responses to \( D_n \)'s distinct queries are distributed identically to values chosen independently and randomly from \( \{0,1\}^n \). It follows that the probability that no pair of distinct queries made by \( D_n \) collides under \( H_{k_1} \) and \( D_n \) accepts is at most \( r_D(n) \). That is, \( b_D(n) \cdot (1 - s_D(n)) \leq r_D(n) \). This means that we have \( q_D(n) \leq s_D(n) + r_D(n) \).

Then \( s_D(n) \geq q_D(n) - r_D(n) \). This means that \( s_D(n) > \frac{1}{2n^c} \) for infinitely many \( n \).

Fix \( n \) such that \( s_D(n) > \frac{1}{2n^c} \). We will show that the security property of \( H \) does not hold.

Consider Experiment 1 again. The key idea we will use is this: Until \( D_n \) sees the response to a query that collides under \( H_{k_1} \) with a previous query, the responses to \( D_n \)'s distinct queries are distributed identically whether they are produced using \( f \circ H_{k_1} \) or whether they are produced by randomly selecting values from \( \{0,1\}^n \). This means that the probability that some pair of queries made by \( D_n \) collides under \( H_{k_1} \) does not change if \( D_n \)'s queries are answered using randomly chosen values from \( \{0,1\}^n \) rather than using \( f \circ H_{k_1} \).

So consider a probabilistic circuit \( D'_n \) that works as follows. \( D'_n \) simulates \( D_n \). For each new query made by \( D_n \) to its “function” gates, \( D'_n \) randomly selects a string from \( \{0,1\}^n \) and uses this string as the response to the query.

Let \( s_{D'}(n) \) be the probability that if \( k_1 \) is randomly chosen from \( \{0,1\}^n \) and \( D'_n \) is run, then some pair of queries seen by \( D'_n \) collide under \( H_{k_1} \). Then, using the idea discussed above, we have \( s_{D'}(n) = s_D(n) \).
Now, by appropriately fixing the random bits of \( D'_n \), we can make a deterministic circuit \( D''_n \) that does at least as well as \( D'_n \) in finding collisions under \( H_{k_1} \) for random \( k_1 \). More precisely, we can make a deterministic circuit \( D''_n \) so that \( s_{D''}(n) \geq s_{D'}(n) \), where \( s_{D'} \) is defined analogously to \( s_{D''} \).

Since \( D''_n \) is a deterministic circuit, the queries that it sees will not change from one run to the next. So let \( Q \) be the set of queries that \( D''_n \) sees. Now, define \( s_Q(n) \) to be the probability that if \( k_1 \) is randomly chosen from \( \{0, 1\}^n \), then there exist distinct \( x, y \in Q \) such that \( H_{k_1}(x) = H_{k_1}(y) \). We have \( s_Q(n) = s_{D''}(n) \geq s_{D'}(n) = s_D(n) \).

Then \( s_Q(n) > \frac{1}{2^{2^n}} \). Now, there are fewer than \( |Q|^2 \) different pairs of distinct strings from \( Q \). Then there must exist a fixed such pair \( x, y \in Q \), such that if \( k_1 \) is randomly chosen from \( \{0, 1\}^n \) then the probability that \( H_{k_1}(x) = H_{k_1}(y) \) is > \( \frac{1}{2^{2^n} |Q|^2} \). (Do you see why?)

Since the size of \( D''_n \) is polynomial in the size of \( D_n \) (and hence polynomial in \( n \)), \( |Q| \) must be polynomial in \( n \). This means there exists \( e \) such that for sufficiently large \( n \) we have that if \( k_1 \) is randomly chosen from \( \{0, 1\}^n \) then the probability that \( H_{k_1}(x) = H_{k_1}(y) \) is > \( 1/n^e \). But this means that \( H \) is not a privately collision resistant hash family.

**Answer to Question 4.**

a) For a 2n bit key \( k_1 k_2 \), say that \( B \) receives encrypted pieces \( e'_0, e'_1, \ldots \), where each \( |e'_i| = 2n \). For each \( i \), \( B \) computes \( m'_i \beta'_i \leftarrow e'_i \oplus G_{k_2}(\bar{i}) \) where \( |m'_i| = |\beta'_i| = n \), and checks if \( \beta'_i = F_{k_1}(\bar{i} m'_i) \); if so, \( B \) outputs \( m'_i \) as the \( i \)-th decrypted message piece; if not, \( B \) outputs \( \text{FAIL} \) and then halts.

b) Say that an adversary \( \{C_n\} \) breaks the unchangeability security of the cryptosystem. We will show how to break the pseudo-randomness of \( F \). Fix \( n \) such that \( p_C(n) > 1/n^4 \).

Say we are given a function \( f : \{0, 1\}^{2n} \rightarrow \{0, 1\}^n \). We randomly select a key \( k_2 \in \{0, 1\}^n \) for \( G \). Using \( C_n \) we create \( m_0 \in \{0, 1\}^n \), see \( G_{k_2}(0) \oplus [m_0, f(0 m_0)] \), we create \( m_1 \in \{0, 1\}^n \), see \( G_{k_2}(1) \oplus [m_1, f(1 m_1)] \), ..., we create \( m_{\ell} \in \{0, 1\}^n \), see \( G_{k_2}(\bar{\ell}) \oplus [m_{\ell}, f(\ell m_{\ell})] \).

We now use \( C_n \) to compute a sequence of strings for a decryptor: \( e'_0, e'_1, \ldots, e'_{\ell'} \).

Then, for each \( i \), we compute \( m'_i \beta'_i = e'_i \oplus G_{k_2}(\bar{i}) \).

Our new adversary will accept iff there is an \( i \), \( 0 \leq i \leq \ell' \), such that \( \beta'_j = f(j, m'_j) \) for all \( j \leq i \) (that is, a decryptor would not abort up to and including piece \( i \)), and either \( i > \ell \), or \( i \leq \ell \) and \( m'_i \neq m_i \).

A pseudo-randomly generated \( f \) will be accepted with probability \( p_C(n) > 1/n^4 \).

If \( f \) is randomly generated, then it will be accepted only if the adversary was able to compute the value \( G_{k_2}(\bar{i}) \oplus [m'_i, f(\bar{i} m'_i)] \) (and hence the value of \( [m'_i, f(\bar{i} m'_i)] \)) for the first \( i \) for which either \( i > \ell \), or \( i \leq \ell \) and \( m'_i \neq m_i \); for such an \( i \), we have that \( (\bar{i} m'_i) \) is an input for which \( f \) has not been queried. So a randomly generated \( f \) will be accepted with probability at most \( 1/2^n \).