Problem 1: Fun with PRFs

a. \( F^a_s = F^0_s(x) \circ F_s(x) \) is not a PRF, for any choice of \( F \).

Proof. Consider a distinguisher \( D^a \) given access to an oracle \( O^a \):
(a) Choose some \( x \leftarrow \{0,1\}^k \).
(b) Calculate \( z = F^0_s(x) \).
(c) Query \( y = y_1 \circ y_2 = O^a(x) \).
(d) If \( y_1 = z \), return 1 (pseudorandom); else, return 0 (random).

If run on \( F^a_s \), \( D^a \) will be correct with probability 1, by definition of the function. If run on a random function, \( D^a \) will only incorrectly guess pseudorandom with probability \( \frac{1}{2^k} \). Thus, \( F^a_s \) is never a PRF. \( \square \)

b. \( F^b_s(x) = F^1_s(x_1) \circ F^2_s(x_2) \) is not a PRF, for any choice of \( F \).

Proof. Consider a distinguisher \( D^b \) given access to an oracle \( O^b \):
(a) Choose some \( x_1 \leftarrow \{0,1\}^{\lceil k/2 \rceil} \).
(b) Choose two distinct \( x_2, x'_2 \leftarrow \{0,1\}^{\lceil k/2 \rceil} \).
(c) Query \( y = y_1 \circ y_2 = O^b(x_1 \circ x_2) \).
(d) Query \( y' = y'_1 \circ y'_2 = O^b(x_1 \circ x'_2) \).
(e) If \( y_1 = y'_1 \), return 1 (pseudorandom); else, return 0 (random).

If run on \( F^b_s \), \( D^b \) will be correct with probability 1, by definition of the function. If run on a random function, \( D^b \) will only incorrectly guess pseudorandom with negligible probability. Thus, \( F^b_s \) is never a PRF. \( \square \)

c. \( F^c_s(x) = F^1_s(x) \oplus s_2 \) is a family of PRFs.

Proof. For the sake of contradiction, assume that \( F^c \) were not a PRF, and that we had a PPT adversary \( A \) which could distinguish \( F^c_s(x) \) from random with non-negligible advantage. We could use \( A \) to construct a ppt \( B \) that distinguishes \( F_s \) from random.

\( A \) expects to interact with an oracle \( O_A \) that responds to its queries either in accordance with \( F^c_s \) or in accordance with a truly random function \( R_1 \). \( B \) interacts with an oracle \( O_B \) that responds to its queries either in accordance with \( F^1_s \), or in accordance with a truly random function \( R_2 \).

\( B \) simulates \( O_A \) as follows: First \( B \) picks a random \( s_2 \leftarrow \{0,1\}^k \). Whenever \( A \) makes a query \( x \), \( B \) queries \( O_B \) on \( x \) to get a response \( y \), and then returns \( z = y \oplus s_2 \). When \( A \) returns a response, \( B \) returns the same response.

If \( O_B \) acts in accordance with \( F^1_s \), then \( B \) will be providing \( A \) with an oracle that acts exactly like \( F^c_s(x) \). If \( O_B \) outputs the results of a truly random function, then \( y \oplus s_2 \) is also truly random, because \( s_2 \) is fixed and a truly random value XORed with a fixed value is truly random.
Thus, $B$'s advantage for distinguishing $F_s(x)$ from the output of a random function is the same as $A$'s advantage at distinguishing $F'_{s}(x)$ from random. Meaning, if $A$ has non-negligible advantage, so does $B$, which is a contradiction. \(\square\)

d. $F^d_s(x) = \begin{cases} F_s(x) & \text{when } x \neq 0^k, \\ a \circ b & \text{when } x = 0^k, \end{cases}$ where we define $a$ to be the first $\lfloor k/2 \rfloor$ bits of $s$, and $b$ to be the last $\lceil k/2 \rceil$ bits of $F_s(x)$, is not necessarily a family of PRFs.

Proof. Let $G : \{0,1\}^{k/2} \rightarrow \{0,1\}^k$ be a PRG, and let $F'$ be a PRF. Then if we define

$$F_s(x) = F'_{G(s_1)}(x),$$

where $s_1$ is defined as the first $\lfloor k/2 \rfloor$ bits of $s$, then $F^d$ would not be a PRF.

First, we sketch why the $F$ defined is a PRF. Suppose it were not, and we had an adversary $A$ which could distinguish $F'_{G(s_1)}$ from a truly random function with non-negligible advantage. If $A$ distinguishes $F'_s$ from a truly random function with non-negligible advantage, then we can use $A$ to show that $F'$ is not a PRF. Otherwise, we would be able to use $A$ to show that $G$ is not a PRG.

But if $F^d$ is instantiated with $F$, then we can query the oracle on $0^k$ to obtain $s_1$. We can then compute $F^d_s(x)$ for any $x$, so we can distinguish $F^d$ from a truly random function. \(\square\)

e. $F^e_s(x) = F_s(0 \circ x) \circ F_s(1 \circ x)$ is a PRF.

Proof. Assume for the sake of contradiction that $F^e_s$ is not a PRF, and that there exists an adversary $A$ which can distinguish $F^e_s$ from a random function with non-negligible probability. Using $A$, we can construct an adversary $B$ that distinguishes $F_s$ from a random function with non-negligible advantage.

On input $1^k$ and with oracle access to $O^e$, $B$ runs $A(1^k)$. Every time $A$ makes a query $x$, $B$ makes two queries to $O^e$ getting $y_0 = O^e(0 \circ x)$ and $y_1 = O^e(1 \circ x)$, and returns to $A$ the value $y = y_0 \circ y_1$. When $A$ outputs a decision, $B$ outputs the same decision.

If $O^e$ was $F_s$, then $B$ is precisely mimicking the behavior of $F^e_s$ for $A$, so by assumption $B$ will be correct with the same probability as $A$. If $O^e$ was truly random, then $B$ is also providing $A$ with true randomness, so $B$ will still be correct with the same probability as $A$. Since $A$ had a non-negligible advantage, $B$ does too, contradicting our assumption that $F$ is a PRF. Thus, $F^e_s$ must be a PRF. \(\square\)

Problem 2: ElGamal Encryption

a. Let $PK = (p, q, g, h)$, $SK = s$, and $m \in QR_p$. Let $c = (c_1, c_2)$ be the encryption $Enc(PK, m) = (g^r, h^r m)$ for some $r \in \mathbb{Z}_q$. Then,

$$Dec(SK, PK, c) \equiv c_2 c_1^{-s} \pmod{p}$$
$$\equiv (h^r m)(g^r)^{-s} \pmod{p}$$
$$\equiv g^{r'} m q^{-rs} \pmod{p}$$
$$\equiv m \pmod{p}$$

HW 7-2
b. Our definition of security is that there exists some algorithm Simulator such that the following two distributions are computationally indistinguishable for all \( m \) in the message space:

\[
D_{\text{Enc}}(1^k, m) = \{(PK, SK) \leftarrow G(1^k); c \leftarrow \text{Enc}(PK, m) : (c, m, PK)\}
\]
\[
D_{\text{Sim}}(1^k, m) = \{(PK, SK) \leftarrow G(1^k); c' \leftarrow \text{Sim}(PK, |m|) : (c', m, PK)\}
\]

Our algorithm FakeCiphertext should first pick some random \( r \in \mathbb{Z}_p^\ast \) and compute \( r^2 = \mu \). This is the same as picking a random \( \mu \) from QR\(_p\), which is what we want for our message space (we should probably also check that \( \mu \) has length \(|m|\)). It also picks random \( z \) and \( \rho \) from \( \mathbb{Z}_q^\ast \) and computes \( \rho^2 \) and \( g^\rho \) (mod \( p \)). Put formally,

\[
D_{\text{Sim}}(1^k, m) = \{(PK, SK) \leftarrow G(1^k); r \leftarrow \mathbb{Z}_p^\ast; \rho, z \leftarrow \mathbb{Z}_q^\ast; \mu = r^2 : (c = (g^\rho, g^z \mu), m, PK)\}.
\]

c. First, we want to prove that, if the DDH assumption is true, then ElGamal is semantically secure; in other words, \( D_0 \approx D_1 \Rightarrow D_{\text{Enc}} \approx D_{\text{Sim}} \). The contrapositive of this is that, if there exists some adversary \( \mathcal{A} \) which can distinguish between \( D_{\text{Enc}} \) and \( D_{\text{Sim}} \) with non-negligible probability \( \epsilon \), we can build an adversary \( \mathcal{B} \) which can distinguish between \( D_0 \) and \( D_1 \) with some non-negligible probability \( \epsilon' \).

We will denote by \( p_{E,0} \) (respectively, \( p_{S,0} \)) as the probability that \( \mathcal{A} \) outputs 0 given an element of \( D_{\text{Enc}} \) (resp., an element of \( D_{\text{Sim}} \)). Assume, without loss of generality, that \( p_{E,0} > p_{S,0} \). We know that we can write \( |p_{E,0} - p_{S,0}| = \epsilon \), and we are trying to show that we can construct a \( \mathcal{B} \) such that

\[
\Pr[x \leftarrow D_0; b \leftarrow \mathcal{B}(x) : b = 0] - \Pr[x \leftarrow D_1; b \leftarrow \mathcal{B}(x) : b = 0]| = \epsilon'(k)
\]

for some non-negligible \( \epsilon'(\cdot) \).

d. To start, we see that \( \mathcal{A} \) takes input of the form \( ((g^\alpha, g^\beta \gamma), m, (p, q, g^\chi)) \), and that:

\begin{itemize}
  \item \textbf{Enc} If our input is from \( D_{\text{Enc}} \), then \( \gamma = m \), \( \alpha \) is random, and \( \beta = \alpha s \).
  \item \textbf{Sim} If our input is from \( D_{\text{Sim}} \), then \( \gamma = \mu \), \( \alpha \) is random, and \( \beta \) is random.
\end{itemize}

Likewise, \( \mathcal{B} \) will take input of the form \( (p, g, g^\mu, g^\ell) \), where \( \ell = x \gamma \) if we are in \( D_0 \) and random if we are in \( D_1 \).

e. Now, we hopefully see the correlation between these two tuples. On input \( (p, g, g^\mu, g^\ell) \), \( \mathcal{B} \) should calculate \( q = \frac{p-1}{\ell} \), then should act like \( \text{Simulator} \) by picking a random \( r \in \mathbb{Z}_p^\ast \) and computing \( \mu = r^2 \). Then, \( \mathcal{B} \) should run \( \mathcal{A} \) on the tuple

\[
(c' = (g^\mu, g^\ell \mu), \mu, PK = (p, q, g^\mu))
\]

If \( \mathcal{B} \)'s input was from \( D_0 \), we know that \( \ell = x \gamma y \) and therefore our tuple looks like

\[
((g^\gamma, g^{x \gamma y} \mu), (p, q, g^\chi))
\]

which (since \( y \) is drawn randomly from \( \mathbb{Z}_q^\ast \)) looks exactly like an element of \( D_{\text{Enc}} \). Thus if \( \mathcal{A} \) outputs 0, so should \( \mathcal{B} \).

Conversely, if \( \mathcal{B} \)'s input was from \( D_1 \), we know \( \ell \) looks like some random \( \beta \) and therefore the tuple looks like

\[
((g^\mu, g^\beta \mu), (p, q, g^\chi))
\]

which looks like it comes from \( D_{\text{Sim}} \). Therefore, if \( \mathcal{A} \) outputs 1, so should \( \mathcal{B} \).
f. We can now formally compute the advantage for \( B \). We refer back to the left-hand side of Equation 1 and see that

\[
\text{LHS of (1)} = \left| \Pr \left[ (p, g, g^x, g^y, g^{xy}) \leftarrow D_0; b \leftarrow B(p, g, g^x, g^y, g^{xy}) : b = 0 \right] \right| \\
- \left| \Pr \left[ (p, g, g^x, g^y, g^z) \leftarrow D_1; b \leftarrow B(p, g, g^x, g^y, g^z) : b = 0 \right] \right| \\
= \left| \Pr \left[ (\ldots) \leftarrow D_0; r \leftarrow \mathbb{Z}_p^*; \mu = r^2; b \leftarrow A((p, q, g, g^\mu), (g^y, g^{xy} \mu)) : b = 0 \right] \right| \\
- \left| \Pr \left[ (\ldots) \leftarrow D_1; r \leftarrow \mathbb{Z}_p^*; \mu = r^2; b \leftarrow A((p, q, g, g^\mu), (g^y, g^{xy} \mu)) : b = 0 \right] \right| \\
= \left| \Pr [d \leftarrow D_{\text{Enc}}; b \leftarrow A(d) : b = 0] - \Pr [d \leftarrow D_{\text{Sim}}; b \leftarrow A(d) : b = 0] \right| \\
= \epsilon(k).
\]

**Problem 3: Nested Encryption**

**Proof.** Recall that semantic security is equivalent to indistinguishability-security. Assume for the sake of contradiction that our nested encryption scheme \((\text{KeyGen}', \text{Enc}', \text{Dec}')\) is not ind-secure, and thus there exists some PPT adversary \( A' \) that can distinguish between encryptions of messages for \( m_0, m_1 \) with non-negligible advantage. That is, \( C'_b(1^k) \approx C'_b(1^k) \), where

\[
C'_b(1^k) = \left\{ (PK', SK') \leftarrow \text{KeyGen}'(1^k); c' \leftarrow \text{Enc}'(m_b, PK') : (1^k, PK', c', \ell(k)) \right\}.
\]

We will show via a reduction that we can use such a PPT to create an adversary \( A \) that breaks the ind-security of \((\text{KeyGen}, \text{Enc}, \text{Dec})\) with non-negligible advantage.

On input \((1^k, PK, c, \ell(k))\), \( A \) runs \( \text{KeyGen}(1^k) \) to get a keypair \((PK_2, SK_2)\). \( A \) then encrypts \( c \) again with this public key, finding \( z = \text{Enc}(c, PK_2) \). \( A \) runs \( A' (1^k, (PK, PK_2), z, \ell(k)) \) and returns whatever it returns.

If \( A \) was given an encryption \( c \) of \( m_b \), then \( z \) is the encryption \( \text{Enc}'(m_b, (PK, PK_2)) \); thus, \( A' \) is given something from the distribution \( C'_b \) (and, due to the correctness of the encryption scheme, assuredly \textit{not} from the distribution \( C_0(1^k) \)) and will be correct with nonnegligible advantage. But the original cryptosystem was ind-secure, and this is a contradiction; thus, \( A' \) cannot exist, and so \((\text{KeyGen}', \text{Enc}', \text{Dec}')\) must be ind-secure. By the equivalence of indistinguishability security and semantic security, the nested encryption scheme is also semantically secure. \( \Box \)