**Problem 1.**  (a) Suppose we can find two message/hash pairs \langle M_1, h(M_1) \rangle \ and \ \langle M_2, h(M_2) \rangle \) such that \( M_1 \neq M_2 \) and \( h(M_1) = h(M_2) \). Then, there exists two distinct Merkle hash trees \( T_1 \) and \( T_2 \) whose outputs are identical.

We can find a collision for the compression function using a top-down side-by-side comparison of our two trees, looking across trees for a case where the outputs of \( f \) are the same, but the inputs differ. Starting at the root, suppose that either the first block or msg-len (thus height) of our trees differ, in this case, clearly we have found a collision in \( f \), so we can stop our search. Otherwise, note that msg-len is the same, so the trees have exactly same height and structure.

We can continue down examining the \( i \)th node of \( T_1 \) and \( T_2 \) and compare the inputs. If they differ, then we have found a collision for the compression function, and the procedure terminates. If not, we proceed to level \( i + 1 \) and repeat the same procedure. Note that since \( M_1 \neq M_2 \), we cannot go through the entire tree without finding a collision. It follows that one can find a collision in the compression function.

(b) A simple example of two messages that hash to the same value is \( M_1 = x \| y \) and \( M_2 = f(x, y) \), for any blocks \( x, y \).

To construct a more general collision without the msg-len block, proceed as follows: Choose any \( M_1 \) and construct the Merkle tree. Without loss of generality, we assume that \(|M_1| = 2^\ell\), and therefore, the depth of the Merkle tree is \( \ell + 1 \). Next, consider the outputs of the compression function \( f(\cdot) \) applied to the leaves to get \( 2^{\ell-1} \) first-level nodes. Let the outputs be \( f_1, \ldots, f_{2^{\ell-1}} \). It is easy to see that by setting \( M_2 = f_1 \| f_2 \| \cdots \| f_{2^{\ell-1}} \), the Merkle tree remains the same (except it has depth \( \ell \), rather than \( \ell + 1 \)). Thus, \( \langle M_1, M_2 \rangle \) is a collision.

This attack does not work if msg-len block is used because, as seen above, the two messages are of different lengths.

**Note:** It is important to note that unlike in the definition of semantic security, there are no restrictions on the lengths of the two colliding messages for breaking collision resistance of a hash function.

**Problem 2.**  \( f_1 \): For any given value of \( y \), is is actually possible to find a value \( x \) to force a given output \( c \):

\[
f_1(x, y) = E(y, x) \oplus y = c
\]

\[
\Leftrightarrow E(y, x) = c \oplus y
\]

\[
\Leftrightarrow x = D(y, c \oplus y)
\]

Thus, we can pick an arbitrary values \( c, y, y' \) such that \( y \neq y' \), and exhibit the following colliding pairs:

- \( f_1(D(y, c \oplus y), y) \)
- \( f_1(D(y', c \oplus y'), y') \)

Since \( y \neq y' \), this is a non-trivial collision.

\( f_2 \): Similarly, we can pick an arbitrary value of \( x \) and solve for \( y \) to produce a desired output \( c \):

\[
f_2(x, y) = E(x, x \oplus y) = c
\]

\[
\Leftrightarrow x \oplus y = D(x, c)
\]

\[
\Leftrightarrow y = x \oplus D(x, c)
\]

Thus, we can pick an arbitrary values \( c, x, x' \) such that \( x \neq x' \), and exhibit the following colliding pairs:
• \( f_2(x, x \oplus D(x, c)) \)
• \( f_2(x', x' \oplus D(x', c)) \)

Since \( x \neq x' \), this is a non-trivial collision.

**Problem 3.** (a) Since \( A \) and all \( B_i \in B \) share the secret key \( k \), any \( B_i \) can send to the parties \( B \setminus \{B_i\} \) a message \( M \) appended with the MAC under \( k \) of \( M \). To the recipients this looks exactly like a message sent by \( A \), and hence, the recipient is not sure the message came from \( A \).

(b) \( B_i \) can successfully fool \( B_j \) (\( i \neq j \)) if and only if \( B_i \) has every key that \( B_j \) has. This is easy to see since \( B_j \) verifies a message by verifying the MACs corresponding to the keys he has. Thus, for the scheme to work, it must be the case that for every \( i \neq j \), \( S_j \not\subseteq S_i \) where \( S_i, S_j \) are the sets of keys held by \( B_i \) and \( B_j \), respectively.

Note that we use the assumption of non-collusion to ensure that if some proper subset of parties \( B^* \subset B \) have between them every key, they cannot work together to fool the parties in \( B \setminus B^* \).

(c) Let the keys be \( k_1, k_2, k_3, k_4, k_5 \). We note that \( 10 = (\binom{5}{2}) \). Hence each \( S_i \) need only contain two keys. The subsets are:

\[
\begin{align*}
S_1: & \{k_1, k_2\} \quad S_2: \{k_1, k_3\} \quad S_3: \{k_1, k_4\} \quad S_4: \{k_1, k_5\} \\
S_5: & \{k_2, k_3\} \quad S_6: \{k_2, k_4\} \quad S_7: \{k_2, k_5\} \quad S_8: \{k_3, k_4\} \\
S_9: & \{k_3, k_5\} \quad S_{10}: \{k_4, k_5\}
\end{align*}
\]

Note that for any two \( S_i, S_j \) (\( i \neq j \)), \( S_i \) and \( S_j \) differ in at least one element.

**Problem 4.** First, recall how a valid padding works. If \( d \) bytes of padding are needed at the end of the message, add \( d \) bytes each with the integer value \( d-1 \). After truncating the padding block, (for the sake of notational convenience) let the last block be denoted by \( c \) and the second last block be denoted by \( c_2 \). By changing \( c_2 \) to \( c'_2 = c_2 \oplus x \), upon decryption, the last message block \( m'_i = m_1 \oplus x \).

Now, given a target byte \( g \), to check if the last byte of \( m_1 \) is \( g \), first pick random bytes \( r_1, \ldots, r_{18} \) and set \( c'_2 = c_2 \oplus r_1r_2\ldots r_{18}g \). Thus, \( m'_i = m_1 \oplus r_1r_2\ldots r_{18}g \). If the last byte of \( m_1 \) were \( g \), then upon decryption, \( m'_i \) ends in a 0 byte. Note that this is a valid padding.

However, if \( m_1 \) ends in some other byte \( g' \neq g \), then the last byte of \( m'_i \) = \( g \oplus g' \). This is a valid padding if and only if the last \( g \oplus g' + 1 \) blocks each decrypt to \( g \oplus g' \). When \( g \oplus g' = 1 \), this occurs with probability \( 2^{-16} \) byte size = \( 2^{-16} \). When \( g \oplus g' = 2 \), this occurs with a probability \( 2^{-24} \). Thus, over all possible padding blocks \((1, \ldots, 15)\), the probability that \( g \oplus g' \) leads to a valid padding is less than \( \sum_{i=2}^{\infty} 2^{-8i} \approx (\frac{1}{27})(\frac{1}{27}) \approx 0.0015\% \) which is a very low probability of error.

Therefore, by modifying \( c_2 \) as described above, we use the server’s response to the ciphertext to determine whether the last byte of the message is some target byte \( g \) or not.

**Problem 5.** (a) This does not provide authenticated encryption. We construct an adversary for the ciphertext integrity game.

1. The adversary submits an arbitrary message \( m \) and receives the ciphertext \( c_1 = (c'_1, c'_2) \).
2. The adversary then queries for the encryption of another arbitrary message \( m' \neq m \) and receives the ciphertext \( c_2 = (c'_2, c'_2) \).
3. Finally, the adversary submits the ciphertext \( c_* = (c'_1, c'_2) \).

Since \( m \neq m' \), \( c'_1 \neq c'_2 \), the adversary wins the ciphertext integrity game with advantage 1.

(b) This scheme provides authenticated encryption. We show that the scheme provides both ciphertext integrity and is CPA-secure. First, we show that if \((E, D)\) provide ciphertext integrity, then \((E_2, D_2)\)
provide ciphertext integrity. As usual, we prove the contrapositive. Suppose we have an efficient adversary \( A \) that wins the ciphertext integrity game against \((E_2, D_2)\) with non-negligible advantage. We construct an adversary \( B \) that wins the ciphertext integrity game against \((E, D)\) with the same advantage:

1. Start running \( A \).
2. Whenever \( A \) requests the ciphertext on a message \( m \), forward the query to the challenger to obtain \( c ← E(k, m) \).
3. At some point, \( A \) outputs a new ciphertext \( c_∗ = (c'_∗, c''_∗) \). Output \( c'_∗ \).

We are perfectly simulating the view \( A \) expects when interacting with a challenger for \((E_2, D_2)\), so by assumption, with non-negligible probability \( ε \), \( A \) will produce a new ciphertext \( c_∗ \) that is valid for \((E_2, D_2)\). This means that \( c'_∗ = c''_∗ \), and moreover, is distinct from any ciphertext received from the challenger. Thus \( B \) achieves the same advantage \( ε \) in the ciphertext integrity game.

Next, we show that \((E_2, D_2)\) is CPA-secure if \((E, D)\) is CPA-secure. Again, we proceed by contrapositive. Suppose \( A \) is a CPA-adversary for \((E_2, D_2)\) that achieves distinguishing advantage \( ε \). Then, we can construct a CPA-adversary \( B \) for \((E, D)\) as follows:

1. Start running \( A \).
2. Whenever \( A \) makes a query \((m_ℓ, m_r)\), forward the query to the challenger to obtain a ciphertext \( c \).
   Forward \((c, c)\) to \( B \).
3. Output whatever \( A \) outputs.

It is easy to see that we have perfectly simulated the view of \( A \) interacting with a challenger for \((E_2, D_2)\), so with the same advantage \( ε \), adversary \( B \) will win the CPA game against \((E, D)\), thus concluding the proof. We conclude that since \((E_2, D_2)\) provides CPA security and ciphertext integrity, it provides authenticated encryption.

(c) This scheme does not provide authenticated encryption. We construct an adversary for the ciphertext integrity game.

1. The adversary queries for the ciphertext on an arbitrary message \( m \), and receives the cipher text \((c_1, c_2)\).
2. The adversary submits \((c_2, c_1)\) as his forged ciphertext.
3. With high probability (certainly non-negligible), the adversary wins the ciphertext integrity game.

With high probability, \( c_1 ≠ c_2 \), and the adversary wins the challenge with non-negligible advantage.

(d) Does not provide authenticated encryption. This scheme does not provide CPA-security. To play the CPA game, the adversary makes no queries and plays the semantic security game.

1. The adversary selects two messages \( m_0 ≠ m_1 \) of equal length and submits them to the challenger.
2. The challenger chooses a random \( m_b \) and gives the adversary \((E(k, m_b), H(m_b))\).
3. Since \( H \) is a hash function known to the adversary, the adversary computes \( H(m_0) \) and \( H(m_1) \).
4. If \( H(m_b) = H(m_1) \), the adversary outputs 1, and otherwise 0.

Since \( H \) is collision-resistant, \( H(m_0) ≠ H(m_1) \) (except perhaps with negligible probability). Thus, the adversary achieves advantage negligible close to 1.
Problem 6. (a) Since \( g = 1 \pmod{N} \), we have that \( g = 1 + aN \) for some integer \( a \). From the Binomial theorem,

\[
g^{x} = \sum_{i=0}^{x} \binom{x}{i} (aN)^{i} = 1 + axN \pmod{N^2},
\]

since for all \( i \geq 2, (aN)^{i} = 0 \pmod{N^2} \). Thus,

\[
g^{x} - 1 = axN \pmod{N^2}.
\]

Rewriting this as a relation over the integers, we have that \( g^{x} - 1 = axN + kN^2 \) for some integer \( k \), or equivalently,

\[
\frac{g^{x} - 1}{N} = ax + kN.
\]

Reducing modulo \( N \), we have the relation \( \frac{g^{x} - 1}{N} = ax \pmod{N} \). Since \( a \in \mathbb{Z}_{N}^{*} \), there exists \( a^{-1} \in \mathbb{Z}_{N}^{*} \), so we have

\[
x = a^{-1} \left( \frac{g^{x} - 1}{N} \right) \pmod{N}.
\]

Note that we can efficiently compute \( a^{-1} \in \mathbb{Z}_{N}^{*} \) using the extended Euclidean algorithm.

(b) From the factorization of \( N = pq \), we can compute \( \varphi(N) = (p-1)(q-1) \). Next, given the ciphertext \( c = g^{m} \cdot h^{N} \), we can compute \( c^{\varphi(N)} = g^{m\varphi(N)} \cdot h^{N \cdot \varphi(N)} \in \mathbb{Z}_{N^2} \) via repeated squaring. Since \( h \) is chosen uniformly over \( \mathbb{Z}_{N^2} \), with overwhelming probability, \( h \in \mathbb{Z}_{N^2}^{*} \), and since \( \varphi(N^2) = N \cdot \varphi(N) \), by Euler’s theorem, \( h^{N \cdot \varphi(N)} = 1 \in \mathbb{Z}_{N^2}^{*} \). Thus, with overwhelming probability, \( c^{\varphi(N)} = g^{m \cdot \varphi(N)} \in \mathbb{Z}_{N^2} \). From part (a), given \( g \) and \( g^{m \cdot \varphi(N)} \in \mathbb{Z}_{N^2} \), we can efficiently compute \( m \cdot \varphi(N) \in \mathbb{Z}_{N} \). Finally, assuming \( \varphi(N) \) is invertible modulo \( N \), we compute \( \varphi(N)^{-1} \in \mathbb{Z}_{N} \) and recover the message \( m \).

(c) Let \( c_0 \leftarrow \text{Enc}(pk, m_0) \) and \( c_1 \leftarrow \text{Enc}(pk, m_1) \) be encryptions of \( m_0 \) and \( m_1 \). Then, \( c_0c_1 \) is an encryption of \( m_0 + m_1 \). To see this, we have that \( c_0 = g^{m_0} \cdot h_0^{N} \), for some \( h_0 \in \mathbb{Z}_{N^2} \) and \( c_1 = g^{m_1} \cdot h_1^{N} \) for some \( h_1 \in \mathbb{Z}_{N^2} \). Then,

\[
c_0c_1 = g^{m_0 + m_1} (h_0h_1)^{N},
\]

which is a valid encryption of \( m_0 + m_1 \) with randomness \( h_0h_1 \). Clearly, this ciphertext is not independent of \( c_0, c_1 \). To re-randomize the ciphertext, we choose \( h \leftarrow \mathbb{Z}_{N^2} \) and instead compute

\[
c_0c_1h^{N} = g^{m_0 + m_1} (h_0h_1h)^{N},
\]

which is a properly distributed encryption of \( m_0 + m_1 \).

Problem 7. Let \( w \) be the world bit. At the beginning of the game, the challenger chooses exponents \( a, b \) uniformly from \( \mathbb{Z}_{2p} \). In World 0, the challenger sets \( t \leftarrow ab \). In World 1, the challenger sets \( t \leftarrow \mathbb{Z}_{2p} \). The challenger then sends the tuple \( (g, h, u, v) = (g, g^{t}, g^{b}, g^{f}) \) to the adversary. Our adversary for distinguishing World 0 from World 1 is described below:

1. The adversary computes \( (h^{q}, u^{q}, v^{q}) \).
2. The adversary outputs \( w' = 1 \) if and only if one of the following conditions hold:
   - \( v^{q} = 1 \) and at least one of \( h^{q} = 1 \) or \( u^{q} = 1 \), where 1 is the identity element in \( G \).
   - \( v^{q} \neq 1 \) and both \( h^{q} \neq 1 \) and \( u^{q} \neq 1 \).

   Otherwise, the adversary outputs \( w' = 0 \).
We show that this attack achieves distinguishing advantage $1/2$. The adversary’s advantage is given by

$$\text{Adv} = |\Pr[w' = 1 \mid w = 0] - \Pr[w' = 1 \mid w = 1]|.$$  

We use the fact that for all $x \in \mathbb{Z}_{2q}$, we have

$$g^{xq} = g^{(xq \mod 2q)} = g^{(x \mod 2)q} \in G.$$  

Thus, for all $x \in \mathbb{Z}_{2q}$, $g^{xq} = 1$ (when $x \mod 2 = 0$) or $g^{xq} = g^q \neq 1$ (when $x \mod 2 = 1$). Note that $g^q \neq 1$ since $\text{ord}(g) = 2q$. Using this fact, we can re-express the adversary’s criterion for outputting 1:

- $t = 0 \pmod{2}$ and at least one of $a = 0 \pmod{2}$ or $b = 0 \pmod{2}$.
- $t = 1 \pmod{2}$ and $a = b = 1 \pmod{2}$.

In other words, the adversary outputs 1 if and only if $t = ab \pmod{2}$. In World 0, $(g, h, u, v)$ is a DDH-tuple, so $t = ab \pmod{2q}$, so we also have $t = ab \pmod{2}$. Thus, the adversary will always outputs 1 in World 0. In World 1, $t$ is uniform over $\mathbb{Z}_{2q}$, and correspondingly, uniform over $\mathbb{Z}_2$. The probability that $t = ab \pmod{2}$ is thus $1/2$. The adversary’s advantage is thus $|1 - 1/2| = 1/2$.  
