Problem 1. Let’s explore why in the RSA trapdoor permutation every party has to be assigned a different modulus $n = pq$. Suppose we try to use the same modulus $n = pq$ for everyone. Every party is assigned a public exponent $e_i \in \mathbb{Z}$ and a private exponent $d_i \in \mathbb{Z}$ such that $e_i \cdot d_i = 1 \mod \varphi(n)$. At first this appears to work fine: to sign a message $m \in \mathcal{M}$, Alice would publish the signature $\sigma_a \leftarrow H(m)^{d_a} \in \mathbb{Z}_n$ where $H : \mathcal{M} \to \mathbb{Z}_n^*$ is a hash function. Similarly, Bob would publish the signature $\sigma_b \leftarrow H(m)^{d_b} \in \mathbb{Z}_n$. Since Alice is the only one who knows $d_a \in \mathbb{Z}$ and Bob is the only one who knows $d_b \in \mathbb{Z}$, this seems fine.

Let’s show that this is completely insecure: Bob can use his secret key $d_b$ to sign messages on behalf of Alice.

a. Show that Bob can use his public-private key pair $(e_b, d_b)$ to obtain a multiple of $\varphi(n)$. Let us denote that integer by $V$.

b. Now, suppose Bob knows Alice’s public key $e_a$. Show that for any message $m \in \mathcal{M}$, Bob can compute $\sigma \leftarrow H(m)^{1/e_a} \in \mathbb{Z}_n$. In other words, Bob can invert Alice’s trapdoor permutation and obtain her signature on $m$.

Hint: First, suppose $e_a$ is relatively prime to $V$. Then Bob can find an integer $d$ such that $d \cdot e_a = 1 \mod V$. Show that $d$ can be used to efficiently compute $\sigma$. Next, show how to make your algorithm work even if $e_a$ is not relatively prime to $V$.

Note: In fact, one can show that Bob can completely factor the global modulus $n$.

Problem 2. Consider again the RSA-FDH signature scheme. The public key is a pair $(N, e)$ where $N$ is an RSA modulus, and a signature on a message $m \in \mathcal{M}$ is defined as $\sigma := H(m)^{1/e_a} \in \mathbb{Z}_N$, where $H : \mathcal{M} \to \mathbb{Z}_N^*$ is a hash function. Suppose the adversary could find three messages $m_1, m_2, m_3 \in \mathcal{M}$ such that $H(m_1) \cdot H(m_2) = H(m_3)$ in $\mathbb{Z}_N$. Show that the resulting RSA-FDH signature scheme is no longer existentially unforgeable under a chosen message attack.

Problem 3. A commitment scheme enables Alice to commit a value $x$ to Bob. The scheme is hiding if the commitment does not reveal to Bob any information about the committed value $x$. At a later time Alice may open the commitment and convince Bob that the committed value is $x$. The commitment is binding if Alice cannot convince Bob that the committed value is some $x' \neq x$. Here is an example commitment scheme:

Public values: A group $\mathbb{G}$ of prime order $q$ and two generators $g, h \in \mathbb{G}$. 

Commitment: To commit to an integer \( x \in \mathbb{Z}_q \) Alice does the following: (1) she chooses a random \( r \in \mathbb{Z}_q \), (2) she computes \( b = g^x \cdot h^r \in \mathbb{G} \), and (3) she sends \( b \) to Bob as her commitment to \( x \).

Open: To open the commitment Alice sends \( (x, r) \) to Bob. Bob verifies that \( b = g^x \cdot h^r \).

Show that this scheme is hiding and binding.

a. To prove the hiding property show that \( b \) reveals no information about \( x \). In other words, show that given \( b \), the committed value can be any element \( x' \in \mathbb{Z}_q \).

   Hint: show that for any \( x' \in \mathbb{Z}_q \) there exists a unique \( r' \in \mathbb{Z}_q \) so that \( b = g^{x'} h^{r'} \).

b. To prove the binding property show that if Alice can open the commitment as \( (x', r') \), where \( x \neq x' \), then Alice can compute the discrete log of \( h \) base \( g \). In other words, show that if Alice can find an \( (x', r') \) such that \( b = g^{x'} h^{r'} \) and \( x \neq x' \) then she can find the discrete log of \( h \) base \( g \). Recall that Alice also knows the \( (x, r) \) used to create \( b \).

c. Show that the commitment is additively homomorphic: given a commitment to \( x \in \mathbb{Z}_q \) and a commitment to \( y \in \mathbb{Z}_q \), Bob can construct a commitment to \( z = ax + by \), for any \( a, b \in \mathbb{Z}_q \) of his choice.

Problem 4. Time-space tradeoff. Let \( f : X \rightarrow X \) be a one-way permutation (i.e., a one-to-one function on \( X \)). Show that one can build a table \( T \) of size \( 2^B \) elements of \( X (B \ll |X|) \) that enables an attacker to invert \( f \) in time \( O(|X|/B) \). More precisely, construct an \( O(|X|/B) \)-time deterministic algorithm \( A \) that takes as input the table \( T \) and a \( y \in X \), and outputs an \( x \in X \) satisfying \( f(x) = y \). This result suggests that the more memory the attacker has, the easier it becomes to invert functions.

   Hint: choose a random point \( z \in X \) and compute the sequence
   \[ z_0 := z, \quad z_1 := f(z), \quad z_2 := f(f(z)), \quad z_3 := f(f(f(z))), \ldots \]

Since \( f \) is a permutation, this sequence must come back to \( z \) at some point (i.e. there exists some \( j > 0 \) such that \( z_j = z \)). We call the resulting sequence \( (z_0, z_1, \ldots, z_j) \) an \( f \)-cycle. Let \( t := ⌈|X|/B⌉ \). Try storing \( (z_0, z_1, z_2t, z_3t, \ldots) \) in memory. Use this table (or perhaps, several such tables) to invert an input \( y \in X \) in time \( O(t) \).

Discussion: Time-space tradeoffs of this nature can be used to attack unsalted hashed passwords, as discussed in class. Time-space tradeoffs also exist for general one-way functions (not just permutations), but their performance is not as good as your time-space tradeoff above. These algorithms are called Hellman tables and discussed in Section 18.7 in the [book].

Problem 5. In the lecture on identification protocols we saw a protocol called S/key that uses an iterated one-way function. In this question we explore the security of iterated one-way functions.
a. Let’s show that the iteration of a one-way function need not be one-way. To do so, let \( f : \mathcal{X} \to \mathcal{X} \) be a one-way function, where \( 0 \in \mathcal{X} \). Let \( \hat{f} : \mathcal{X}^2 \to \mathcal{X}^2 \) be defined as:

\[
\hat{f}(x, y) = \begin{cases} 
(0, 0) & \text{if } y = 0 \\
(f(x), 0) & \text{otherwise}
\end{cases}
\]

Show that \( \hat{f} \) is one-way, but \( \hat{f}^{(2)}(x, y) := \hat{f}(\hat{f}(x, y)) \) is not.

b. Let’s show that the iteration of a one-way permutation is also one-way (recall that a permutation is a one-to-one function). Suppose \( f : \mathcal{X} \to \mathcal{X} \) is a one-way permutation. Show that \( f^{(2)}(x) := f(f(x)) \) is also one-way. As usual, prove the contrapositive.

c. Explain why your proof from part (b) does not apply to a one-way function. Where does the proof fail?