Instructions: You must typeset your solution in LaTeX using the provided template. Please submit your problem set via Gradescope. Include your name and the names of any collaborators at the top of your submission.

Problem 1: Double One-Way Functions [15 points]. 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) The iteration of one-way functions need not be one-way. Let \( f : X \rightarrow X \) be a one-way function, where \( 0 \in X \). Let \( \hat{f} : X^2 \rightarrow X^2 \) 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.

Note: while it’s possible that iterating a one-way function breaks security, that is not thought to be the case with the one-way functions used to instantiate S/Key.

(b) Show that the iteration of a one-way permutation is also one-way. In particular, suppose \( f : X \rightarrow X \) is a one-way permutation, and 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?

Problem 2: Conceptual Questions [16 points]. For each of the following statements, say whether it is TRUE or FALSE. Write at most one sentence to justify your answer.

(a) Which of the following are true in a world where \( P = NP \).

i Secure PRFs exist in the standard model.

ii Secure PRFs exist in the random oracle model.

iii The one-time-pad cipher is secure.

(b) If there exists a PRG with 1-bit stretch, there exists a PRG with \( n^{600} \)-bit stretch (where \( n \) is the length of the PRG seed).

(c) Let \( P : K \times X \rightarrow X \) be a pseudorandom permutation. Then:

i \( f_{k=0}(x) := F(0, x) \) is (always) a one way function.

ii \( f_{k=0}(x) := F(0, x) \) is (always) a one way permutation.

iii \( f_{x=0}(k) := F(k, 0) \) is (always) a one way function.

iv \( f_{x=0}(k) := F(k, 0) \) is (always) a one way permutation.
**Problem 3: Our Favorite PRF [15 points].** In class we saw the PRF $F(k, x) = H(x)^k$ and were told that it has many useful properties. In this problem, we will explore two applications of this PRF.

(a) Let $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ be a PRF defined over groups $(\mathcal{K}, +)$ and $(\mathcal{Y}, \otimes)$, where $+$ and $\otimes$ are the respective group operations in those groups. We say $F$ is key-homomorphic if it holds that

$$F(k_1 + k_2, x) = F(k_1, x) \otimes F(k_2, x).$$

Is the PRF $F(k, x) = H(x)^k$ defined with a random oracle $H : \mathcal{X} \to G$ (where $G$ is a group of prime order $p$) a key-homomorphic PRF? Please prove your answer one way or the other (only a few sentences needed).

(b) Key rotation is a common problem encountered in cloud storage: how to change the key under which data is encrypted without sending the keys to the storage provider? A naive solution is to download the encrypted data, decrypt it, re-encrypt it under a new key, and re-upload the new ciphertext. We will now see how this process can be made more efficient with a key-homomorphic PRF.

Suppose you have a ciphertext $c$ made up of blocks $c_1, ..., c_N$ that corresponds to a message $m = (m_1, ..., m_N)$ encrypted under a key $k_1$ using a key-homomorphic PRF $F$ in counter mode, i.e., $c_i = m_i \otimes F(k_1, i)$. Now you want to rotate to a key $k_2$. It turns out you can send the storage provider a single element $k_{\text{update}} \in \mathcal{K}$ which it can then use to generate $c'$, an encryption of $m$ under $k_2$. Please tell us how you can compute $k_{\text{update}}$ and how the storage provider can use $k_{\text{update}}$ and $c$ to compute $c'$.

(c) An oblivious PRF is an interactive protocol between a client who holds a message $x$ and a server who holds a key $k$. The protocol allows the client to learn the PRF evaluation $F(k, x)$ without the server learning anything about $x$. Oblivious PRFs are used in many advanced crypto protocols.

It turns out that there is an oblivious PRF protocol for the PRF $F(k, x) = H(x)^k$. Please show us how a client holding $x$ and a server holding $k$ can interact so that the client learns $H(x)^k$ while the server learns nothing about $x$. You don't need to prove security, we would just like to see the protocol.

**Problem 4: Multi-Commitments [10 points].** Let $G$ be a group of prime order $q$ in which the discrete logarithm problem is hard. Let $g$ and $h$ be generators of $G$. As we saw in class, the Pedersen commitment scheme commits to a message $m \in \mathbb{Z}_q$ using randomness $r \in \mathbb{Z}_q$ as $\text{Commit}(m; r) := g^m h^r \in G$. Moreover, we saw that Pedersen commitments are additively homomorphic, meaning that given commitments to $m_1$ and $m_2$, one can compute a commitment to $m_1 + m_2$. The “public parameters” associated with the Pedersen commitment scheme are the description of the prime-order group $G$ and the group elements $g$ and $h$.

(a) Use $G$ to construct an additively homomorphic commitment scheme $\text{Commit}_n(m_1, \ldots, m_n; r)$ that commits to a length-$n$ vector of messages $(m_1, \ldots, m_n) \in \mathbb{Z}_q^n$ using randomness $r \in \mathbb{Z}_q$. The output of the commitment should be short – only a single group element. You should specify both the public parameters of your scheme (which may be different from that of the basic Pedersen commitment scheme) as well as the description of the $\text{Commit}_n$ function.

You do not need to prove (or even argue for) the security of your scheme, but the resulting scheme should be perfectly hiding and computationally binding assuming hardness of discrete log in $G$. 
(b) Show that if you are given a hash function \( H: \mathbb{Z}_q \rightarrow G \) (modeled as a random oracle), the public parameters for your construction from Part (a) only needs to consist of the description of the group \( G \) and the description of \( H \). Argue informally why your modification to the construction is secure. You do not need to provide a formal proof. This problem shows that getting rid of public parameters is another reason why random oracles are useful in practice!

Optional Feedback [5 points]. Please answer the following questions to help design future problem sets. You are not required to answer these questions (the points are free), and if you would prefer to answer anonymously, please use the anonymous feedback form. However, we do encourage you to provide feedback on how to improve the course experience.

(a) Roughly how long did you spend on this problem set?

(b) What was your favorite problem on this problem set?

(c) What was your least favorite problem on this problem set?

(d) Any other feedback for this problem set? Was it too easy/difficult?

(e) Any other feedback on the course so far?