Homework 2

Problem 1: FHE to Collision Resistance. A collision-resistant hash function consists of two algorithms Gen and Eval such that $\mathsf{Gen}(\lambda)$ is a randomized procedure that outputs a hash key hk, and $\mathsf{Eval}(\mathsf{hk}, x)$ is deterministic and outputs a hash y. Here, x is a sting of length $p(\lambda)$ for some polynomial p, and y is a string of length $q(\lambda)$ for some polynomial q, where we require that $q(\lambda) \ll p(\lambda)$. This means there are very many collisions in the function $\mathsf{Eval}(\mathsf{hk}, \cdot)$. For security, we require that no such collision can be found. For all PPT adversaries $\mathcal{A}$, there exists a negligible function ϵ such that

$$\Pr[x_0 \neq x_1, \mathsf{Eval}(\mathsf{hk}, x_0) = \mathsf{Eval}(\mathsf{hk}, x_1) : \mathsf{hk} \leftarrow \mathsf{Gen}(\lambda), (x_0, x_1) \leftarrow \mathcal{A}(\mathsf{hk})] < \mathsf{negl}(\lambda)$$

In this problem, we will show how to use FHE to build collision resistant hash functions. Actually, we will show that a weaker object called *additively homomorphic encryption* (which allows additions, but not multiplications) implies collision resistance. We will assume a public key homomorphic scheme, though it is straightforward to adapt to a secret key scheme. Given a public key homomorphic scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, we construct the hash function $(\mathsf{Gen}', \mathsf{Eval})$ as follows:

- $\mathsf{Gen}'(\lambda)$: First fun $(\mathsf{ek}, \mathsf{dk}, \oplus) \leftarrow \mathsf{Gen}(\lambda)$. Then, for $i = 1, \ldots, p(\lambda)$, do the following. Sample a random $b_i \in \{0, 1\}$, and compute $c_i = \mathsf{Enc}(\mathsf{ek}, b_i)$. Output the hash key $\mathsf{hk} = (\oplus, \{c_i\}_{i \in [p(\lambda)]})$.
- $\mathsf{Eval}(\mathsf{hk}, x \in \{0, 1\}^{p(\lambda)})$: Let $c_x = \oplus_{i:x_i=1} c_i$. That is, homomorphically add the c_i for i with $x_i = 1$. Homomorphic addition is not necessarily associative, so pick some fixed order (independent of x) of parenthesis for the computation. Output c_x.

Prove that this scheme is a secure collision resistant hash function.

Problem 2: Merkle Hash Trees. Let $\mathsf{Gen}, \mathsf{Eval}$ be a collision resistant hash function with $p(\lambda) = 2\lambda$ and $q(\lambda) = \lambda$. That is, the hash function shrinks the input by a factor of 2. Merkle Hash Trees are a way to get collision resistant hash functions for much larger inputs. Let $\mathsf{Eval}_0 = \mathsf{Eval}$. Then for $i > 0$, let $\mathsf{Eval}_i(\mathsf{hk}, (x \in \{0,1\}^\lambda)^{2^i})$ do the following. Write $x = (x_0, x_1)$ where $x_b \in (\{0,1\}^\lambda)^{2^{i-1}}$ and output $\mathsf{Eval}(\mathsf{hk},\ (\mathsf{Eval}_{i-1}(\mathsf{hk}, x_0), \mathsf{Eval}_{i-1}(\mathsf{hk}, x_1))\ )$. Intuitively, Eval_i corresponds to a tree of depth i, where the leaves consist of the 2^i blocks of the input, and the value of each internal node is obtained by hashing the values of the two children. The output is the root node.

Part (a): Prove that $\mathsf{Gen}, \mathsf{Eval}_i$ is a secure collision resistant hash function for any integer i.

Part (b): Given h obtained by from $\mathsf{Eval}_i(\mathsf{hk}, x)$ for some x, and some chosen block $x_j \in \{0,1\}^\lambda$, we wish to give a short "proof" that the jth block of x is x_j. That is, we want a procedure $\mathsf{Open}(\mathsf{hk}, x, j)$ that outputs a "proof" π, and a procedure $\mathsf{Ver}(\mathsf{hk}, h, j, x_j, \pi)$ that verifies the proof. We make the following requirements:

- **Correctness**. For any $x \in (\{0,1\}^\lambda)^{2^i}$, any $j \in [2^i]$, if $\pi = \mathsf{Open}(\mathsf{hk}, x, j)$, $h = \mathsf{Eval}_i(\mathsf{hk}, x)$, then $\mathsf{Ver}(\mathsf{hk}, h, j, x_j, \pi)$ accepts.
- **Security**. Intuitively, it should be difficult for an adversary to compute $h = \mathsf{Eval}_i(\mathsf{hk}, x)$, and then later pick a block $x'_j \neq x_j$, and "prove" that the jth block of x is x'_j. We formalize this as follows. For any PPT adversary $\mathcal{A}$, there exists a negligible function ϵ such that

  $$\Pr[x'_j \neq x_j, \mathsf{Ver}(\mathsf{hk}, h, j, x'_j, \pi) = 1 : \mathsf{hk} \leftarrow \mathsf{Gen}(\lambda), \mathcal{A}(\mathsf{hk}) = (x, j, x'_j, \pi), h = \mathsf{Eval}_i(\mathsf{hk}, x)] < \mathsf{negl}(\lambda)$$

- **Compactness**. Up until this point, one could set $\pi = x$ as a valid proof. However, for applications such as multiparty NIKE, we would like π to be small. So the compactness requirement is that $|\pi| = \mathsf{poly}(i, \lambda)$. For $i \gg \lambda$, i will grow with $\log |x|$, and so $|\pi|$ is poly*logarithmic* in $|x|$, and is in particular much smaller than x.

Devise algorithms $\mathsf{Open}, \mathsf{Ver}$ for Merkle hash trees such that $|\pi| = O(i\lambda)$.

Problem 3: NIKE for unbounded users. Use $\mathsf{Open}, \mathsf{Ver}$ from above to construct multiparty NIKE where the number of users is unbounded. Use the abstract approach discussed in class, and use diO to prove security. Your NIKE may be in a trusted setup model. The trusted setup and public procedures must not depend on the number of users exchanging keys. This means that once the trusted setup is run, any polynomial number of users can establish a shared group key.

Hint: it suffices to construct NIKE for a bounded number of users, but where the bound is superpolynomial.

Problem 4: Multilinear Maps.

- (a) On a *symmetric* bilinear map, show that the decisional Diffie Hellman problem is easy. That is, show that the following two distributions are distinguishable: $([a]_1, [b]_1, [ab]_1)$ and $([a]_1, [b]_1, [c]_1)$ where a, b, c are random in $\mathbb{Z}_p$.
- (b) Explain why the decisional Diffie Hellman problem is plausibly still hard on an *asymmetric* bilinear map.
- (c) Given a matrix $A \in \mathbb{Z}_p^{n \times n}$, let $[A]_i$ denote the $n \times n$ matrix of encodings obtained by encoding A component-wise at level i. That is, $([A]_i)_{j,k} = [A_{j,k}]_i$. Explain how, given matrix encodings $[A]_1$ and $[B]_1$ on a symmetric multilinear map, it is possible to compute:
  - $[A + B]_1$
  - $[A \cdot B]_2$.

  (It is straightforward to extend this to asymmetric multilinear maps, though you do not have to show it.)
- (d) Show that, given a symmetric n-linear map, it is possible to distinguish the encoding $[A]_1$ of a full-rank $n \times n$ matrix from the encoding $[B]_1$ of a singular matrix (that is, B has rank less than n).
- (e) Explain why the same is not true on an asymmetric multilinear map.
- (f) For any integer n, devise two distributions on *level-1* encodings for a symmetric multilinear map such that (1) the encodings are plausibly indistinguishable on a n-linear map, but (2) the encodings are distinguishable on an $(n + 1)$-linear map. Justify why the distributions are indistinguishable on an n-linear map (though no formal proof is necessary).