Problem 1: Recall that in most suggested use of property-preserving encryption applications one first encrypts using the PPE scheme and then wraps that in conventional encryption. In CryptDB, for example, columns are encrypted using conventional symmetric encryption until a client makes a search query that requires some PPE ciphertext to be exposed. Explain how this impacts attacks, and what you would need to do to measure the impact of this setup on attack efficacy in practice.

Explain how ORAM differs from the PPE-based systems. What information is leaked by ORAM accesses to a malicious outsourced storage?

Problem 2: Consider a synchronous mixnet $X$ that takes in ciphertexts on a set of messages $M = \{m_1, m_2, \ldots, m_n\}$ and outputs a set of messages $M' = \{m'_1, m'_2, \ldots, m'_n\}$, where messages are in $\{1, \ldots, q - 1\}$ for some prime $q$ (as for El Gamal plaintexts).

(a) A verifiable mixnet $X$ is supposed to prove that $M = M'$. As discussed in class, a naïve way one might try to do this is to show that every $m_i \in M$ has a corresponding $m'_j \in M'$ such that $m_i = m'_j$. Why doesn’t this approach work? (Give an example of an $(M, M')$ such that $M \neq M'$ but a valid proof can be constructed.)

(b) Suppose that $M$ consists of “yes” / “no” votes encoded respectively as messages $m = 1$ and $m = 2$. For how big a gap in “yes” / “no” votes can an adversarial mix $X$ in principle change the outcome of an election if it uses the proof in (a)?
(c) Consider an alternative “proof” that shows that for any pair \((m_i, m_j) \in M \times M\), there exists a pair \((m'_u, m'_v) \in M' \times M'\) such that \(m_i m_j = m'_u m'_v\). Does this fix the problem in (a)? If so, explain why. If not, show a counterexample.

**Problem 3:** Many censorship circumvention tools tunnel communications over TLS. A censor might try to develop tests that seek to identify this tool by passively inspecting application layer headers or actively probing a destination IP/port. Discuss how you would go about determining both passive and active fingerprinting attacks against a given TLS-using tool (for which you have the ability to run the code). Discuss how you would measure its efficacy.

**Problem 4:** The “Logjam” paper (“Imperfect Forward Secrecy: How Diffie-Hellman Fails in Practice”) describes how precomputation can help compute discrete logarithms over 512-bit primes in a matter of minutes. What implications does this have for the public keys surveyed in the “Ron was wrong, Whit is right” paper?