Problem 1. (25 points) In this problem, I want to show you an example of the challenges that come from implementing cryptography in real life. In this example, the adversary can do more than what we allowed it to do in mathematical model: it can tamper with the computation. (Cryptography has recently started modeling and addressing tampering adversaries, but we are still far from implementing protections against them.)

Suppose you have a device (smart card, computer, etc.) that is performing an RSA signature generation (computing \( \sigma = h^d \mod n \) for some \( h = H(m) \)) using CRT: separately computing \( \sigma_p = h^d \mod p \) and \( \sigma_q = h^d \mod q \) and then combining. Suppose you can hit the device with just enough radiation or power-glitch it to cause exactly one of the two modular exponentiations to compute an incorrect value. That is, you get \( \sigma_p' \neq \sigma_p \) but \( \sigma_q \) is correct. This causes the output to be some \( \sigma' \neq \sigma \). The signing device then outputs \( \sigma' \) as the signature, instead of the correct \( \sigma \). Show how to factor \( n \) given \( m \) and \( \sigma' \) (and, of course, the RSA public key \((n,e)\)). This attack can actually be carried out on certain smart cards that perform RSA signatures. (Hint: it may be easier to first figure out how to factor \( n \) given \( \sigma, \sigma' \), and the public key \((n,e)\).)

Problem 2. (25 points) Lamport’s signatures for a message space of size \( 2^\ell \) have \( 2^\ell \) values in the secret key (and, therefore, also in the public key). A signature consists of a particular message-dependent subset of those values. For simplicity, we will focus on \( \ell = 3 \): thus, the secret key consists of 6 values and the message space has size 8. Show how to generalize Lamport’s scheme to allow for signatures on the message space of size 20 (i.e., 20 different messages, not 20 bits of message) while keeping the key size at 6 values. Show, by reduction to security of the underlying one-way function, that the resulting scheme is secure. Hint: don’t think of the message as bit strings, just think of 20 distinct values. You need a different signatures for each of the 20 message values, and you need to ensure that forgery requires inverting a one-way function.

Problem 3. (25 points) Show how to compute the root of an \( n \)-leaf Merkle tree in \( \log n \) space. More precisely, given \( x_1, x_2, \ldots, x_n \) as values to be placed in the leaves of the tree, and a hash function \( H \), describe how to compute the root \( r \) of the tree while never storing more than \( \lceil \log n \rceil \) hash values.