Final Exam

1 Problem 1 (20 points)

(a) Your friend is confused why constructing CCA-secure public key encryption is so much harder than in the secret key setting. Inspired by the "Encrypt-then-MAC" scheme, your friend proposes the following "Encrypt-then-Sign" scheme, which simply replaces the components of "Encrypt-then-MAC" with public key objects: first encrypt the message using a CPA-secure public key encryption scheme, and then sign the scheme using a CMA-secure digital signature scheme.

Explain why this will not work.

(b) Show how to build a one-way function given any public key encryption scheme

(c) In class, we saw the following signature scheme: to sign a message m, compute $\sigma = F^{-1}(\mathsf{sk}, H(m))$ where (F, F^{-1}) is a trapdoor permutation, and H is a hash function. Show that collision resistance of H is not enough to prove security. To do so, devise a one-way permutation (F, F^{-1}) and collision resistant hash function H such that $F^{-1}(\mathsf{sk}, H(m))$ is not a CMA-secure signature scheme. To do so, start with an arbitrary collision resistant hash function H' and arbitrary trapdoor permutation (F', F'^{-1}).

(d) Suppose taking cube roots is hard mod N. Show that it is hard to solve the equation $27x^3 - 135x^2 + 225x = c \mod N$ for a random $c \in \mathbb{Z}_N$.

2 Problem 2 (20 points)

Recall that CCA1 security (aka lunch time security) is a weaker version of full CCA security where the CCA queries are limited to occurring before the challenge query.

(a) Devise a *public key* encryption scheme that is CPA secure, but not CCA1 secure. Prove both the security and insecurity. You may use as a starting point any reasonable assumption we've seen in class (CPA/CCA secure public key encryption, DDH, RSA, one-way functions, etc).

(b) Devise a *public key* encryption scheme that is CCA1 secure, but not full CCA secure (that is, CCA queries after the challenge query are devastating). Prove both security and insecurity. Again, you may use as a starting point any reasonable assumption we've seen in class.

(c) Determine if the following is true, and in either case prove it: if you encrypt a message $m = m_0 m_1 \dots$ as $\mathsf{Enc}(\mathsf{pk}, m_0), \mathsf{Enc}(\mathsf{pk}, m_1), \dots$ and Enc is CCA1 secure, then the resulting encryption scheme is CCA1 secure.

(d) Determine if the analogous statement is true for CCA2 security, and in either case prove it: If Enc is CCA2 secure, then the resulting scheme is CCA2 secure.

3 Problem 3 (20 points)

One disadvantage of the Merkle-Damgard hash function is that it cannot be parallelized. Here, we will see that it is possible hash in a way that can be parallelized.

Let $h : \{0,1\}^{2\lambda} \to \{0,1\}^{\lambda}$ be a collision resistant hash function. Let $T = 2^t$ be a power of 2. We will now construct a hash function H for λT-bit messages.

First, given a message $m \in \{0,1\}^{\lambda T}$, break m into T blocks $m_1, \dots, m_T$ of λ bits. Now, hash the message using h two blocks at a time $(m_1^{(1)} = h(m_1, m_2), m_2^{(1)} = h(m_3, m_4), \dots)$ to get $T/2$ blocks of λ bits, $m_1^{(1)}, \dots, m_{T/2}^{(1)}$. Now we hash these $T/2$ blocks using h again two blocks at a time $(m_1^{(2)} = h(m_1^{(1)}, m_2^{(1)}), m_2^{(2)} = h(m_3^{(1)}, m_4^{(1)}), \dots)$. Repeat until finally only one block remains. This one final block is the output of the hash function.

(a) Explain briefly how, given T processors, H can be computed in time $O(\log T)$.

(b) Show that H is collisions resistant. In particular, show how, given a collision $m \neq m'$ for H, it is possible to efficiently compute a collision $x \neq x'$ for h.

(c) Explain why this hash function will not be collision resistant for variable length messages. That is, show how to devise two messages m, m' with length $T\lambda$ and $T'\lambda$ respectively, such that $H(m) = H(m')$.

(d) Devise a way to block the attack from part (c).

4 Problem 4 (20 points)

Recall the tree-based PRF we saw in class using pseudorandom generators. Show how the secret key holder can do the following. Given an arbitrary point x, the secret key holder can devise a new key k_x, with the following properties:

- Given k_x, it is possible to compute $\mathsf{PRF}(k, y)$ for any $y \neq x$.
- Given k_x, it is *impossible* to compute $\mathsf{PRF}(k, x)$, or even learn anything about it. Formally, we say that the distributions $(k_x, \mathsf{PRF}(k, x))$ and (k_x, r) are computationally indistinguishable, where k is chosen at random, k_x is produced according to the procedure you describe, and r is uniformly random.

Note that k_x need not belong to the same space as the original key k. It may be a longer bit string or even a data structure.

Hint: A trivial (but incorrect) solution is for k_x to just contain $\mathsf{PRF}(k, y)$ for all $y \neq x$. This does not work, as the size of k_x is exponentially large. What other values can be given out as a part of k_x so that k_x has polynomial size, but still allows for computing all of the values except $\mathsf{PRF}(k, x)$.

5 Problem 5 (20 points)

(a) Construct a PRF F_a such that, if the first bit of the key is leaked, then it is insecure — that is, it is possible to distinguish the PRF from a random function with non-negligible advantage, given the first bit of the key. You may use as a starting point any secure PRF F'. Your F must be a secure PRF if the key is completely hidden.

(b) Construct a PRF F_b such that, if the XOR of all bits of the key is leaked, then it is insecure — that is, it is possible to distinguish the PRF from a random function with non-negligible advantage, given the XOR of the key bits. You may use as a starting point any secure PRF F'. Your F must be a secure PRF if the key is completely hidden.

(c) Construct a PRF F_c that is still secure if *any* single bit of the key is leaked. You may use as a starting point a secure PRF F', but you may not assume anything about F except its security: F' may be insecure if any bit of key is leaked. As a hint, if F' has key space $\mathcal{K}$, try having F_c have key space $\mathcal{K}^2$.

(d) Is your PRF from part (c) guaranteed to be secure if the adversary is allowed to learn the XOR of all of the key bits? (What if the starting PRF F' is actually F_b?).

(e) Let F be a PRF. Suppose the adversary is allowed to choose an efficiently computable function $f : \mathcal{K} \to \{0, 1\}$ that outputs a single bit, and the adversary learns $f(k)$ where k is the PRF key. Show how, no matter what F is — it could be any of the PRFs seen in class, the homeworks, this exam, or even a yet-undiscovered PRF — the adversary can devise an efficiently computable f such that he can distinguish $F(k, \cdot)$ from random given $f(k)$.