Homework 4

1 Problem 1 (20 points)

Explain how to efficiently find collisions in the following hash functions:

(a) $H_a : \{0,1\}^{512} \to \{0,1\}^{256}$ is defined as follows. Let F, F^{-1} be a secure block cipher with block size and key length 256. $H_a(x, y) = F(y, x \oplus y) \oplus y$ That is, y is interpreted as a key for F, and to compute H_a, we first XOR y with x, apply the block cipher to the result, and then XOR with y one more time

(b) $H_b(x, y) = F(y \oplus x, x)$, where F, F^{-1} is as in part (a)

(c) H_c is a sponge function where the message block size is equal to the internal state size, so that when we XOR in a block of the message, it affects the entire state (so in other words, in the lecture slides the blue boxes represent the entire state and the orange boxes are non-existant). Here, the round function f is an un-keyed SPN network.

(d) $H_d : \{0,1\}^{257} \to \{0,1\}^{256}$ is defined as follows. Let $H : \{0,1\}^* \to \{0,1\}^{256}$ be a collision-resistant hash function for arbitrary-length messages, and let $H_d(x, b) = H(x)$ if $b = 0$ and $H(H(x))$ if $b = 1$.

2 Problem 2 (35 points)

In this problem, we will see how to combine cryptosystems, so that the resulting scheme is secure, as long as *either* component is secure.

(a) Let $G_0, G_1 : \{0,1\}^\lambda \to \{0,1\}^{4\lambda}$ be two efficiently computable functions. Suppose you know that either G_0 or G_1 is a secure pseudorandom generator, but you do not know which one. Construct a new function G that is guaranteed to be a secure pseudorandom generator, and prove security regardless of which of G_0, G_1 are secure.

(b) Let (F_0, F_0^{-1}) and (F_1, F_1^{-1}) be two block ciphers, both with n-bit blocks and n-bit keys. Suppose you know that one of the two schemes is a secure block cipher, but you do not know which one. Both schemes are guaranteed to be

correct. Construct a new block cipher F, F^{-1} for n-bit blocks that is guaranteed to be secure. Prove security regardless of which of (F_0, F_0^{-1}) or (F_1, F_1^{-1}) are secure (but assuming both are correct). The key length for F may be different than n.

(c) Suppose in part b that the insecure scheme was not guaranteed to even be correct. Show that your solution in part (b) is then not correct, and also that it is not secure.

(d) Let $(\mathsf{MAC}_0, \mathsf{Ver}_0)$ and $(\mathsf{MAC}_1, \mathsf{Ver}_1)$ be two message authentication codes with the same message space. Suppose you know that one of the two schemes are (weakly) CMA secure, but you do not know which one. Construct a new message authentication code $(\mathsf{MAC}, \mathsf{Ver})$ that is guaranteed to be (weakly) CMA secure. Proof security regardless of which scheme is secure.

(e) Explain why your solution to part (d) does *not* work, in general, for *strong* CMA security.

(f) Suppose Com_0, Com_1 are two commitment schemes that are perfectly hiding, but only one of them is (computationally) binding; you do not know which. Construct a commitment scheme Com that is perfectly hiding and computationally binding; prove security.

(g) Suppose Com_0, Com_1 are two commitment schemes that are perfectly *binding*, but only one of them is (computationally) *hiding*; you do not know which. Construct a commitment scheme Com that is perfectly binding and computationally hiding; prove security.

3 Problem 3 (25 points)

Consider the following scenario. Alice wants to send Bob a commitment to a large list $L = (x_1, \ldots, x_n)$. Here the x_i are small, constant-sized values, and n is very large (say, in the billions). Suppose L contains sensitive data: perhaps the x_i are medical records or financial transactions. At a later point, Alice wants to reveal x_i to Bob. Alice wants to ensure hiding holds, before giving any openings[1]. Bob wants to make sure that x_i was indeed in the original list L. In particular, binding should hold separately for each x_i.

A trivial solution to this problem is to have Alice simple commit to each x_i separately. The problem is that the commitment is now very large, and grows with n. Perhaps Bob only has limited storage, in which case he would want a tiny commitment.

[1]It is also possible to ask for hiding to hold for all un-opened x_i. For simplicity, we will ignore such considerations

(a) Prove that achieving perfect binding is impossible, for large enough n and a tiny constant-sized commitment.

(b) Explain how to combine any standard commitment scheme with collision resistant hashing to achieve the desired security properties, all with a commitment that is constant-sized (independent of n). How do you open x_i? Prove that your resulting commitment scheme is binding with respect to each x_i, and hiding when no openings have been given. You may assume either a perfectly binding or a perfectly hiding standard commitment. It is fine if both hiding and binding are proved in the computational setting. Note that there are many possible solutions here.

Unfortunately, your solution in part (b) still requires large *openings*. That is, while the commitment is small, the opening sent during the reveal phase still grows with n. Next, you will explore a possible solution. Assume you have a standard commitment such that the commitments produced are half the size of the messages being committed (such a commitment can be built similar to your solution in part (b)). We will built a commitment scheme with a similar structure as in HW3, problem 5.

Assume n is a power of 2. To commit to L, Alice creates a binary tree with n leaves, and places the values x_i in the leaves of the tree. Now she proceeds to fill in the internal nodes of the tree with various commitments. Let c_L, c_R be the values found in two neighboring nodes (which are either two x_i values if the nodes are leaves, or two commitments if the nodes are internal). Then c_P, the value at the parent node, is set to be a commitment to the pair (c_L, c_R), using the standard commitment scheme. Recall that we assumed the commitment scheme produced commitments that were half the size of the inputs. Thus, c_P, c_L, c_R all have the same size.

The final commitment that is sent to Bob is then the value in the root node.

(c) Explain how Alice can open x_i, such that her opening has size $O(\log n)$. Prove hiding and binding.