Midterm II

Name:

SID:

- Not all parts of a problem are weighted equally. Before you answer any question, read the problem carefully. Be precise and concise in your answers.
- You may consult at most *10 sheets of notes.* Apart from that, you may not look at books, notes, etc. Calculators, phones, computers, and other electronic devices are NOT permitted for looking up content. However, you may use an electronic device such as a tablet for writing your answers.
- You have 80 minutes: there are 3 questions on this exam worth a total of 100 points.
- You are allocated 90 minutes and the extra 10 minutes are provided for the submission of the exam on Gradescope. You must submit/upload the exam on time. Note that late submissions will not be accepted.
- **DSP Students** must submit the exam in time as per your accommodation. Thus, if you are allowed 1.5× (resp., 2×) the exam time then you must submit it within $80 * 1.5 + 10$ (resp., $80 * 2 + 10$) mins, i.e. 130 (resp., 170) mins. **Your allotted time has already been adjusted on Gradescope, and you will not need to email the midterm to the instructors like last time. Please submit on Gradescope.**
- The exam must be submitted before 8 PM PT on March 31st, 2021. No exams later than this time will be accepted regardless of when you start the test.
- We will not be answering questions during the exam. If you feel that something is unclear please write a note in your answer.

1 One-Way Functions and Collision-Resistant Hash Functions (30 points)

(a) Let $f : \{0,1\}^n \to \{0,1\}^n$ be an efficiently computable (not necessarily one-to-one) function with a hard-code predicate $B : \{0,1\}^n \to \{0,1\}$. Show that f is not necessarily a one-way function. That is, describe a function f and a predicate B, and prove that i) B is a hard-code predicate for f and ii) f is not one-way (**10 points**).

Name:

Solution: Let $f(x) = 0$ for all $x \in \{0,1\}^n$ and $B(x) = \bigoplus_i x_i$. Then B is a hard-core predicate because $f(x)$ reveals no information about x and parity is a balanced predicate, and f is not one-way because any x is a pre-image of 0.

(b) Let $f_1 : \{0,1\}^n \to \{0,1\}^n, f_2 : \{0,1\}^n \to \{0,1\}^n$ be one-way functions. Show that $g : \{0,1\}^{2n} \to \{0,1\}^n$ defined as $g(x,y) = f_1(x) \oplus f_2(y)$ is not necessarily one-way. That is, describe two functions f_1, f_2, and prove that i) each is one-way, and ii) g is not one-way. You may **either** follow the template below, **or** come up with your own construction.

Template: Let $h' : \{0,1\}^n \to \{0,1\}^{n-2}$ be a one-way function, and let $h(x) = (1, h'(x), 1)$. Argue that h is one-way (**2 points**).

$\boxed{}$.

Now, define

$$f_1(x) = \begin{cases} \boxed{} & \text{if } x = (z, 0^{n/2}) \\ \boxed{} & \text{otherwise} \end{cases}, f_2(x) = \begin{cases} \boxed{} & \text{if } x = (z, 0^{n/2}) \\ \boxed{} & \text{otherwise} \end{cases}$$

and argue that each is one-way (**5 points**).

$\boxed{}$.

Finally, show that $g(x, y) = f_1(x) \oplus f_2(y)$ is not one-way (**3 points**).

.

Name:

Free response: Alternatively, present your own construction (**10 points**).

Solution: Let $h : \{0,1\}^n \to \{0,1\}^n$ be a one-way function with the property that both the first bit and the last bit of its output are always 1 (it is easy to see that pre-pending and appending a 1 to the output of any one-way function will produce a function that is also one-way). Define $f_1(x) = h(x)$ except if $x = (z, 0^{n/2})$, in which case the output is $(z, 0^{n/2})$. Define $f_2(x) = h(x)$ except if $x = (z, 0^{n/2})$, in which case the output is $(0^{n/2}, z)$. Each of f_1, f_2 is one-way since a random input x will only be such that $x_{n/2+1,\ldots,n} = 0^{n/2}$ with negligible probability. And if x is not of this form, the output of $f_1(x)$ will not end with $n/2$ zeros (since the last bit will be 1), and the output of $f_2(x)$ will not start with $n/2$ zeros (since the first bit will be 1). Thus, inverting either of f_1 or f_2 requires inverting h, except with negligible probability, showing that f_1, f_2 are one-way. Next, note that for any $z = (z_1, z_2)$, $f_1(z_1, 0^{n/2}) \oplus f_2(z_2, 0^{n/2}) = z$, so any image of g is easy to invert.

(c) Let (Gen, H) be a collision resistant hash function, where $H^s : \{0,1\}^{2n} \to \{0,1\}^n$. Let (Gen', H') be defined as follows. $\mathsf{Gen}'(1^n)$ runs $\mathsf{Gen}(1^n)$ twice to obtain independently sampled hash keys s_1, s_2. Then $H'^{s_1,s_2} : \{0,1\}^{2n} \to \{0,1\}^n$ is defined as $H'^{s_1,s_2}(x) = H^{s_1}(x) \oplus H^{s_2}(x)$. Show that (Gen', H') is not necessarily collision-resistant. That is, define a hash function (Gen, H) and prove that i) (Gen, H) is collision-resistant (assuming the existence of some other collision-resistant hash function (Gen^*, H^*)), and ii) (Gen', H') is not collision-resistant (**10 points**).

Solution: Let $H^* : \{0,1\}^{2n-1} \to \{0,1\}^{n-1}$ be collision-resistant, and define $H(x) = x_1, H^*(x_2, \ldots, x_{2n})$. Then for any collision (x, y) in H, it must be that $x_1 = y_1$, so (x_{-1}, y_{-1}) is a collision in H^*, showing that H is collision-resistant. Next, note that for any $x' \in \{0,1\}^{2n-1}$, $(0, x'), (1, x')$ is a collision for H'.

Name:

2 CCA-secure MAC (45 points)

Consider a "CCA-style" extension to the definition of secure message authentication codes, where the adversary is provided with both a Mac and a Vrfy oracle. Our starting point will be the "standard" notion of MAC security, called "existential unforgeability under adaptive chosen-message attacks", and we will consider a variant of this definition that allows for Vrfy oracle queries.

(a) Provide a formal definition of CCA-secure MACs. That is, describe an experiment called $\mathsf{CCA}-\mathsf{Mac}-\mathsf{Forge}_{\mathcal{A},\Pi}(n)$, and provide a security requirement stating that no adversary can win your game except with negligible probability (**10 points**).

Solution: The CCA-MAC experiment $\mathsf{CCA}-\mathsf{Mac}-\mathsf{Forge}_{\mathcal{A},\Pi}(n)$ is defined as follows:

(i) The challenger samples $k \leftarrow \mathsf{Gen}(1^n)$.

(ii) The adversary $\mathcal{A}$ is given input 1^n and oracle access to $\mathsf{Mac}_k(\cdot)$ and $\mathsf{Vrfy}_k(\cdot,\cdot)$. The adversary eventually outputs a pair (m,t). Let Q denote the set of all queries that $\mathcal{A}$ asked to its $\mathsf{Mac}_k(\cdot)$ oracle.

(iii) The output of the experiment is defined to be 1 if and only if (1) $\mathsf{Vrfy}_k(m,t) = 1$ and (2) $m \notin Q$.

Π is a CCA-secure MAC if for all adversaries $\mathcal{A}$,

$$\Pr[\mathsf{CCA}-\mathsf{Mac}-\mathsf{Forge}_{\mathcal{A},\Pi}(n) = 1] = \mathsf{negl}(n).$$

(b) Assume that Π is a standard secure *deterministic* MAC that has *canonical verification*, meaning that i) the Mac algorithm is deterministic and ii) the Vrfy algorithm, on input (m, t), recomputes $t' := \mathsf{Mac}_k(m)$ and accepts if $t' = t$. Prove that Π also satisfies your definition from part (a) (**15 points**).

Solution: When Π is deterministic and has canonical verification, each message has only a single valid tag. Thus, if the scheme is secure, then access to a Vrfy oracle does not help (and so Π is secure in the sense of the definition given in part (a)). To see this, note that for any query (m, t) to the Vrfy oracle there are 3 possibilities:

(i) m was previously queried to the Mac oracle, and response t was received. Here the adversary already knows that $\mathsf{Vrfy}_k(m, t) = 1$.

(ii) m was previously queried to the Mac oracle, and response $t' \neq t$ was received. Since Π is deterministic, here the adversary already knows that $\mathsf{Vrfy}_k(m, t) = 0$.

(iii) m was not previously queried to the Mac oracle. By security of Π, we can argue that $\mathsf{Vrfy}_k(m, t) = 0$ with all but negligible probability. This is because otherwise, m, t is a valid forgery.

Name:

(c) Assume that $\Pi = (\mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrfy})$ is a standard secure MAC. Construct another MAC $\Pi' = (\mathsf{Gen}', \mathsf{Mac}', \mathsf{Vrfy}')$ that is standard secure, but is *not* secure under your definition from part (a). You may **either** follow the template below, **or** come up with your own construction.

Template: The idea will be to set things up so that the Vrfy' oracle can be used by the adversary to learn the key k one bit at a time. In particular, Gen' will just run Gen, and Mac' will simply run Mac and then append 3 bits, all set to 0:

$$\mathsf{Mac}'_k(m) = (\mathsf{Mac}_k(m), 0, 0, 0).$$

Now, define your Vrfy' function (**10 points**):

.

Next, show that Π' is a secure MAC (**5 points**):

Finally, show that Π' is insecure when $\mathcal{A}$ is given a verify oracle (**5 points**).

Free response: Alternatively, present your own construction (**20 points**).

Solution: Let $\Pi = (\mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrfy})$ be any secure MAC. Define the scheme $\Pi' = (\mathsf{Gen}, \mathsf{Mac}', \mathsf{Vrfy}')$ as follows:

- $\mathsf{Mac}'_k(m)$ computes $t = \mathsf{Mac}_k(m)$ and outputs $\langle t, 0, 0, 0 \rangle$.
- $\mathsf{Vrfy}'_k(m, \langle t, f, i, b \rangle)$ works as follows: if $f = 0$ then output 1 if and only if $\mathsf{Vrfy}_k(m, t) = 1$; if $f = 1$ then output 1 if and only if $\mathsf{Vrfy}_k(m, t) = 1$ and the i-th bit of the key k is equal to b.

First observe that Π' is standard secure because the tags returned in scheme Π' can be generated from the tags returned from scheme Π. Furthermore, a forgery in Π' requires, in particular, a forgery in Π.

On the other hand, Π' is insecure when an adversary $\mathcal{A}$ is given access to both a Mac and a Vrfy oracle. The attack is to query an arbitrary message m to the Mac oracle to obtain the tag $\langle t, 0, 0, 0 \rangle$. Then query the Vrfy oracle with $(m, \langle t, 1, i, 0 \rangle)$ for $i = 1, 2, \ldots, n$. By doing so, $\mathcal{A}$ learns the entire key k and can then forge a valid tag for any message of its choice.

Name:

3 Feistel Network (25 points)

Consider a three-round Feistel network with block size 64 bits and mangler function $\widehat{f} : \{0,1\}^{48} \times \{0,1\}^{32} \to \{0,1\}^{32}$ that takes a key k of length 48 bits and maps 32-bit inputs to 32-bit outputs. Suppose that there was a flaw in the design of $\widehat{f}$ that resulted in the following behavior. With probability 1/2 over the sampling of the key, it holds that *for all inputs*, the first bit of the output of $\widehat{f}(k, \cdot)$ will be set to the first bit of its input. That is,

$$\Pr_{k \leftarrow \{0,1\}^{48}}[\forall x \in \{0,1\}^{32}, \widehat{f}(k,x)_1 = x_1] = 1/2.$$

Show that there exists some efficient adversary $\mathcal{A}$ and constant $\epsilon > 0$ such that $\mathcal{A}$ has advantage ϵ in distinguishing this Feistel network from a uniformly random permutation. Your $\mathcal{A}$ should only require a single input-output pair, and you are not required to optimize the distinguishing advantage ϵ (just show that it is greater than 0). You may make the following heuristic assumptions.

- Each round i of the network applies $\widehat{f}$ with an independent and uniformly sampled key k_i.
- For keys k that do not satisfy the condition that $\forall x \in \{0,1\}^{32}, \widehat{f}(k,x)_1 = x_1$, assume that $\widehat{f}(k, \cdot)$ is a uniformly random function.

Solution: First observe that if the mangler function for each round sends the first bit of its input to the first bit of its output, then the following will hold. Let x, y be the first bit of the first half and the first bit of the second half of the input to the Feistel network (i.e. the 1st and 33rd bit of the input). Then these bits will be updated as follows in the three rounds: $(x,y) \to (y, x \oplus y) \to (x \oplus y, x) \to (x, y)$. Now, given the heuristic assumptions stated above, we can calculate the probability that the 1st and 33rd bit of the output of the Feistel network is equal to the 1st and 33rd bit of the input: $(1/8)(1) + (7/8)(1/4) = 11/32$. However, a uniformly random input-output pair will have this property with probability 1/4. Thus, the distinguishing advantage of an adversary $\mathcal{A}$ that obtains a single input-output pair and checks whether the 1st and 33rd bits are equal is $11/32 - 1/4 > 0$.