CS 171: Problem Set 5

Due Date: March 7th, 2024 at 8:59pm via Gradescope

1 Insecure Candidates for MACs

Two candidate constructions of MACs are given below. The schemes use a pseudrandom function function F that maps $\{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$.

Show that each of the following MAC schemes is insecure.

1. Scheme 1:

   (a) $\mathsf{Gen}(1^n)$: Output $k \leftarrow \{0,1\}^n$.

   (b) $\mathsf{Mac}(k, m)$: Let $m = m_0||m_1$, where $m_0, m_1 \in \{0,1\}^n$. Then Mac outputs

   $$t := F(k, m_0)||F(k, m_0 \oplus m_1)$$

   (c) $\mathsf{Verify}(k, m, t)$: Output 1 if $t = \mathsf{Mac}(k, m)$, and output 0 otherwise.

2. Scheme 2:

   (a) $\mathsf{Gen}(1^n)$: Output $k \leftarrow \{0,1\}^n$.

   (b) $\mathsf{Mac}(k, m)$: Let $m = m_0||m_1$, where $m_0, m_1 \in \{0,1\}^{n-1}$. Then Mac samples $r \leftarrow \{0,1\}^n$, and outputs

   $$t := r||\big[F(k, r) \oplus F(k, 0||m_0) \oplus F(k, 1||m_1)\big]$$

   (c) $\mathsf{Verify}(k, m, t)$: Let $m = m_0||m_1$, where $m_0, m_1 \in \{0,1\}^{n-1}$, and let $t = r||t'$, where $r, t' \in \{0,1\}^n$. Output 1 if

   $$t' = F(k, r) \oplus F(k, 0||m_0) \oplus F(k, 1||m_1)$$

   and output 0 otherwise.

2 Encrypt-Then-Authenticate

The encrypt-then-authenticate approach constructs a CCA-secure encryption scheme using any CPA-secure encryption scheme and any *strongly* secure MAC.[1] You will show that a MAC with regular security will not suffice.

1. Describe a MAC $\mathsf{MAC}' := (\mathsf{Gen}', \mathsf{Mac}', \mathsf{Verify}')$ that is secure but not strongly secure. In your construction, you may start with a secure MAC, $\mathsf{MAC} := (\mathsf{Gen}, \mathsf{Mac}, \mathsf{Verify})$.
2. Prove that MAC' is secure or cite a security proof given in discussion or lecture.[2]
3. Prove that when MAC' is combined with any CPA-secure encryption scheme using encrypt-then-authenticate, it results in an encryption scheme that is not CCA-secure.

[1] The encrypt-then-authenticate approach is described in Katz & Lindell, 3rd edition, construction 5.6 and also in lecture 10, slide 21.

[2] You don't need to prove that MAC' is not strongly secure.

3 Randomized MACs

Previously we've dealt mainly with deterministic MACs, and here we will examine one reason why: we will show that any randomized MAC can be converted into a deterministic MAC using a PRF.

A **randomized MAC** is a scheme where $\mathsf{Mac}(k, m)$ is allowed to sample a uniformly random string r each time it runs.[3] A **deterministic MAC** is a scheme where $\mathsf{Mac}(k, m)$ is a deterministic function of the inputs (k, m).

Question: Given a randomized MAC $\Pi_R = (\mathsf{Gen}_R, \mathsf{Mac}_R, \mathsf{Verify}_R)$, construct a deterministic MAC $\Pi_D = (\mathsf{Gen}_D, \mathsf{Mac}_D, \mathsf{Verify}_D)$, and prove that Π_D is secure.[4]

You may assume that Π_R takes n-bit messages and l-bit random strings, and Π_D takes n-bit messages. In your construction, you may also use a PRF F that maps $\{0,1\}^n \times \{0,1\}^n \to \{0,1\}^l$.

You may find it useful to follow the template below and fill in the blanks.[5]

1. Construction of Π_D:
   (a) $\mathsf{Gen}_D(1^n)$: Sample $k = (k_1, k_2) \leftarrow \{0,1\}^n \times \{0,1\}^n$ and output it.
   (b) $\mathsf{Mac}_D(k, m)$:
   (c) $\mathsf{Verify}_D(k, m, t)$: Output 1 if $\mathsf{Mac}_D(k, m) = t$, and output 0 otherwise.
2. **Claim 3.1** Π_D *is a secure MAC.*

   Proof

   (a) We will define two hybrids and show that they are indistinguishable[6]:

   i. Hyb_0 is the MAC security game $\mathsf{MAC\text{-}forge}_{\mathcal{A},\Pi_D}(n)$:

   A. Sample $k \leftarrow \mathsf{Gen}_D(1^n)$.

   B. *Query Phase*: The adversary $\mathcal{A}$ gets oracle access to $\mathsf{Mac}_D(k, \cdot)$. Let $\mathcal{Q}$ be the set of all the message queries that the adversary submits to the oracle.

   C. $\mathcal{A}$ outputs (m^*, t^*). The challenger checks that $\mathsf{Verify}_D(k, m^*, t^*) = 1$ and $m^* \notin \mathcal{Q}$. The output of the game is 1 if both checks passed and 0 otherwise.

[3] We sometimes sharpen our notation from $\mathsf{Mac}(k, m)$ to $\mathsf{Mac}(r; k, m)$ to make the algorithm's random input explicit.

[4] The definition of security is given in Katz & Lindell, 3rd edition, definition 4.2 and in lecture 9, slide 5.

[5] The size of each blank box below doesn't indicate how long your answer should be. The box just marks an incomplete section of the proof, and your answers will sometimes be larger than the boxes.

[6] We've given more detail for the hybrids than necessary. The extra detail is shown in gray.

ii. Hyb_1 is the same as Hyb_0 except that any calls to F are replaced with calls to a uniformly random function R:

A. Sample $k \leftarrow \mathsf{Gen}_D(1^n)$, and sample a function R uniformly at random from the set of functions that map $\{0,1\}^n \to \{0,1\}^l$.

B. *Query Phase*: Let $\mathsf{Mac}'(k,m)$ be the same as $\mathsf{Mac}_D(k,m)$, except any calls to F are replaced with calls to R. Next, the adversary $\mathcal{A}$ gets oracle access to $\mathsf{Mac}'(k,\cdot)$. Finally, let $\mathcal{Q}$ be the set of all the message queries that the adversary submits to the oracle.

C. $\mathcal{A}$ outputs (m^*, t^*). The challenger checks that $\mathsf{Verify}_D(k, m^*, t^*) = 1$ and $m^* \notin \mathcal{Q}$. The output of the game is 1 if both checks passed and 0 otherwise.

(b) **Claim 3.2** *For any probabilistic polynomial-time adversary $\mathcal{A}$, there exists a negligible function* negl *such that*

$$\Big| \Pr[\mathsf{Hyb}_0 \to 1] - \Pr[\mathsf{Hyb}_1 \to 1] \Big| \leq \mathsf{negl}(n)$$

Proof

■

(c) **Claim 3.3** *For any probabilistic polynomial-time adversary $\mathcal{A}$, there exists a negligible function* negl *such that*

$$\Pr[\mathsf{Hyb}_1 \to 1] \leq \mathsf{negl}(n)$$

Proof

■

(d) Finish the proof:

■