CS 171: Discussion Section 6 (2/26)

1 Insecure Candidates for MACs

Two candidate constructions of MACs are given below. The schemes use a pseudorandom function F that maps $\{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$. The differences between schemes 1 and 2 are shown in red.

Show that each of the following MAC schemes is insecure.

Scheme 1:

1. $\mathsf{Gen}(1^n)$: Output $k \leftarrow \{0,1\}^n$.
2. $\mathsf{Mac}(k,m)$: Let $m = m_0||m_1$, where $m_0, m_1 \in \{0,1\}^n$. Then Mac outputs
$$t = F(k,m_0) \oplus F(k,m_1)$$
3. $\mathsf{Verify}(k,m,t)$: Output 1 if $t = \mathsf{Mac}(k,m)$, and output 0 otherwise.

Scheme 2:

1. $\mathsf{Gen}(1^n)$: Output $k \leftarrow \{0,1\}^n$.
2. $\mathsf{Mac}(k,m)$: Let $m = m_0||m_1$, where $m_0, m_1 \in \{0,1\}^n$. Then Mac outputs
$$t = F(k,m_0)||F(k,m_1)$$
3. $\mathsf{Verify}(k,m,t)$: Output 1 if $t = \mathsf{Mac}(k,m)$, and output 0 otherwise.

Solution

1. For scheme 1: the adversary $\mathcal{A}$ does not have to make any queries. It just outputs the message $m = m_0||m_0$ for an arbitrary $m_0 \in \{0,1\}^n$, together with a tag $t = 0^n$. $\mathcal{A}$ succeeds with probability 1 because for any key k, $\mathsf{Mac}(k,m) = F(k,m_0) \oplus F(k,m_0) = 0^n$.
2. For scheme 2: let adversary $\mathcal{A}$ do the following:
   (a) Pick a message $m = m_0||m_1$ where $m_0, m_1 \in \{0,1\}^n$, $m_0 \neq m_1$.
   (b) Query $\mathsf{Mac}(k,\cdot)$ on m to obtain
   $$\mathsf{Mac}(k,m) = \underbrace{F(k,m_0)}_{=:t_0}||\underbrace{F(k,m_1)}_{=:t_1}$$
   (c) Output message $m^* = m_1||m_0$ and tag $t^* = t_1||t_0$.

   We will argue that $\mathcal{A}$ succeeds with probability 1. Note that m^* has not yet been submitted as a query to $\mathsf{Mac}(k,\cdot)$ because $m_0 \neq m_1$. Furthermore, $\mathsf{Verify}(k,m^*,t^*) = 1$ because $\mathsf{Mac}(k,m^*) = F(k,m_1)||F(k,m_0) = t_1||t_0 = t^*$.

$\square$

2 Difference Between Regular and Strong Security for MACs

Construct 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, $\mathsf{MAC} := (\mathsf{Gen}, \mathsf{Mac}, \mathsf{Verify})$.

Solution

Construction of MAC':

- $\mathsf{Gen}'(1^n)$: Run $\mathsf{Gen}(1^n)$.
- $\mathsf{Mac}'(k, m)$:
  1. Compute $t = \mathsf{Mac}(k, m)$.
  2. Sample $b \leftarrow \{0, 1\}$.
  3. Output $t' := t||b$.
- $\mathsf{Verify}'(k, m, t)$: Let $t_{\mathsf{truncated}}$ be t with the final bit removed. Run $\mathsf{Verify}(k, m, t_{\mathsf{truncated}})$, and output the result.

Claim 2.1. MAC' *is a secure message authentication code.*

Proof.

1. Overview: Assume toward contradiction that there is an adversary $\mathcal{A}$ that can break the security of MAC'. Then we will construct an adversary $\mathcal{B}$ that can break the security of MAC. This is a contradiction because MAC is known to be secure. Therefore, our assumption was false, and in fact, MAC' is secure.
2. Construction of $\mathcal{B}$:
   (a) $\mathcal{B}$ runs $\mathcal{A}$ and simulates the security game for MAC', which $\mathcal{A}$ is designed to play in.
   (b) When $\mathcal{A}$ outputs a query m_i for the $\mathsf{Mac}'(k, \cdot)$ oracle,
      i. $\mathcal{B}$ forwards the query m_i to its oracle for $\mathsf{Mac}(k, \cdot)$ to obtain $t_i := \mathsf{Mac}(k, m_i)$.
      ii. Then $\mathcal{B}$ samples a bit $b_i \leftarrow \{0, 1\}$,
      iii. and sends the tag $t'_i := (t_i||b_i)$ to $\mathcal{A}$.
   (c) In the end, when $\mathcal{A}$ outputs (m^*, t^*), $\mathcal{B}$ removes the last bit of t^*. Let $t^*_{\mathsf{truncated}}$ be t^* with the last bit removed. Finally, $\mathcal{B}$ outputs $(m^*, t^*_{\mathsf{truncated}})$.
3. Note that $\mathcal{B}$ correctly simulates the security game for MAC' with $\mathcal{A}$ as the adversary. In particular, $\mathcal{B}$ correctly simulates $\mathcal{A}$'s queries to the $\mathsf{Mac}'(k, \cdot)$ oracle.
4. We claim that if $\mathcal{A}$ outputs an (m^*, t^*) that would win in the simulation of the MAC' security game, then $\mathcal{B}$'s output $(m^*, t^*_{\mathsf{truncated}})$ will win in the security game for MAC. First, m^* was not previously output as a query by $\mathcal{A}$ or $\mathcal{B}$. Second, $\mathsf{Verify}'(k, m^*, t^*)$ would output 1, which implies that $\mathsf{Verify}(k, m^*, t^*_{\mathsf{truncated}})$ outputs 1 as well.
5. If $\mathcal{A}$ wins the security game for MAC' with non-negligible probability, then $\mathcal{B}$ wins the security game of MAC with non-negligible probability. Since MAC is secure, this is a contradiction. So our assumption was false, and in fact, MAC' is also secure.

□

Claim 2.2. MAC' *is not strongly secure.*

Proof.

1. The strong security game differs from the regular security game in that the adversary can win even if they output a valid tag on a message that was previously queried. More specifically, the adversary wins the strong security game if it outputs an (m^*, t^*) such that $\mathsf{Verify}'(k, m^*, t^*) = 1$, and the pair (m^*, t^*) was not previously computed by the oracle for $\mathsf{Mac}'(k, \cdot)$ during the query phase. For more detail, see Katz & Lindell, 3rd edition, definition 4.3.

2. We will construct an adversary $\mathcal{A}$ that wins the strong security game with non-negligible probability.

   Description of $\mathcal{A}$:

   (a) $\mathcal{A}$ outputs a query for an arbitrary message m and receives in response $t := \mathsf{Mac}'(k, m)$.

   (b) Let b be the last bit of t, and let $t_{\mathsf{truncated}}$ be t with the last bit removed. Then $\mathcal{A}$ chooses a new tag
   $$t' = t_{\mathsf{truncated}} || (b \oplus 1)$$
   and outputs (m, t').

3. $\mathcal{A}$ will win the strong security game with probability 1. First, $\mathsf{Verify}'(k, m, t') = 1$ because Verify' just computes $\mathsf{Verify}(k, m, t_{\mathsf{truncated}})$, which outputs 1. Second, even though m was previously queried to the $\mathsf{Mac}'(k, \cdot)$ oracle, t' was not the tag that the oracle outputted. Therefore, (m, t') is a valid output for the strong security game.

□ □

3 MACs and Pseudorandom Functions

In the construction of a fixed-length MAC that we saw in lecture (and in construction 4.5 in the textbook), Mac is a pseudorandom function. However we will show that this feature is not necessary.

Construct a secure deterministic MAC for n-bit messages such that Mac is not a pseudorandom function. Note: you may use a pseudorandom function in your construction.

Solution

Construction:

Let $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be a pseudorandom function.

1. $\mathsf{Gen}(1^n)$: Sample $k \leftarrow \{0,1\}^n$.
2. $\mathsf{Mac}(k, m)$: Output
$$t = F(k, m)||m$$
(see footnote[1])
3. $\mathsf{Verify}(k, m, t)$: Output 1 if $\mathsf{Mac}(k, m) = t$, and output 0 otherwise.

Claim 3.1. $(\mathsf{Gen}, \mathsf{Mac}, \mathsf{Verify})$ *is a secure MAC.*

Proof.

1. Overview: Assume toward contradiction that there is an adversary $\mathcal{A}$ that breaks the MAC security of $(\mathsf{Gen}, \mathsf{Mac}, \mathsf{Verify})$ (i.e. $\mathcal{A}$'s success probability in the MAC security game is a non-negligible function of n). Then we will construct an adversary $\mathcal{B}$ that can break the PRF security of F. This is a contradiction because F is known to be secure. Therefore, our assumption was false, and in fact, $(\mathsf{Gen}, \mathsf{Mac}, \mathsf{Verify})$ is secure.
2. Construction of $\mathcal{B}$:
   (a) $\mathcal{B}$ runs $\mathcal{A}$ and simulates the MAC security game, which $\mathcal{A}$ is designed to play in.
   (b) When $\mathcal{A}$ outputs a query m_i for the $\mathsf{Mac}(k, \cdot)$ oracle,
      i. $\mathcal{B}$ forwards the query m_i to its oracle to obtain either $s_i = F(k, m_i)$ or $s_i = R(m_i)$, where R is a truly random function.
      ii. Then $\mathcal{B}$ sends the tag $t_i := (s_i||m_i)$ to $\mathcal{A}$.
   (c) In the end, when $\mathcal{A}$ outputs (m^*, t^*):
      i. $\mathcal{B}$ queries its oracle on m^* to obtain either $s^* = F(k, m^*)$ or $s^* = R(m^*)$.
      ii. $\mathcal{B}$ checks that $(s^*||m^*) = t^*$, and checks that m^* was not previously queried by $\mathcal{A}$. If both checks pass, then $\mathcal{B}$ outputs 1. Otherwise $\mathcal{B}$ outputs 0.

[1] We could have also chosen to let $\mathsf{Mac}(k, m)$ output $t = F(k, m)||0^n$ or $t = F(k, m)||0$. We claim (but won't prove) that with these other constructions, $(\mathsf{Gen}, \mathsf{Mac}, \mathsf{Verify})$ would be a secure MAC, but Mac would not be a PRF.

3. <u>Pseudorandom Case</u>: We will show that $\Pr[\mathcal{B}^{F(k,\cdot)} = 1] = \textsf{non-negl}(n)$.

   Note that if $\mathcal{B}$ is querying $F(k,\cdot)$, then $\mathcal{B}$ correctly simulates the MAC security game for $(\textsf{Gen}, \textsf{Mac}, \textsf{Verify})$. In step b, $\mathcal{B}$ correctly simulates $\mathcal{A}$'s queries to the $\textsf{Mac}(k,\cdot)$ oracle. In step c, $\mathcal{B}$ outputs 1 if and only if the MAC challenger would have accepted (m^*, t^*).

   This means that $\Pr[\mathcal{B}^{F(k,\cdot)} = 1]$ equals the probability that $\mathcal{A}$ wins the MAC security game, which is non-negligible.

4. <u>Truly Random Case</u>: We will show that $\Pr[\mathcal{B}^{R(\cdot)} = 1] = \textsf{negl}(n)$.

   If $\mathcal{B}$ outputs 1, that means m^* was not previously queried by $\mathcal{A}$. Since the function R was sampled uniformly at random, then the value of $R(m^*)$, given all of the queries and responses previously made by $\mathcal{A}$, is uniformly random. The probability that $\mathcal{A}$ outputs a t^* such that $t^*_{1,\dots,n} = R(m^*)$ is 2^{-n}. Therefore, $\Pr[\mathcal{B}^{R(\cdot)} = 1] \le 2^{-n}$, so $\Pr[\mathcal{B}^{R(\cdot)} = 1]$ is negligible.

5. In summary,

   $$\left|\Pr[\mathcal{B}^{F(k,\cdot)} = 1] - \Pr[\mathcal{B}^{R(\cdot)} = 1]\right| = \left|\textsf{non-negl}(n) - \textsf{negl}(n)\right|$$

   which is non-negligible. Then $\mathcal{B}$ would break the PRF security of F. However, this is a contradiction because F is secure. Therefore, our initial assumption was false, and in fact, $(\textsf{Gen}, \textsf{Mac}, \textsf{Verify})$ is a secure MAC.

$\square$

Claim 3.2. $\textsf{Mac}$ *is not a secure pseudorandom function.*

Proof.

1. <u>Construction</u>: Let's construct a distinguisher $\mathcal{D}$ that breaks the pseudorandomness of $\textsf{Mac}$.

   (a) $\mathcal{D}$ submits a query $m \in \{0,1\}^n$ and receives either $t = F(k,m)||m$ or $t = R(m)$, where R is sampled uniformly at random from the set of functions mapping $\{0,1\}^n \to \{0,1\}^{2n}$.

   (b) If the last n bits of t equal m, then $\mathcal{D}$ outputs 1. Otherwise, $\mathcal{D}$ outputs 0.

2. <u>Pseudorandom Case</u>: $\Pr[\mathcal{D}^{\textsf{Mac}(k,\cdot)} = 1] = 1$ because the last n bits of $\textsf{Mac}(k,m)$ are always equal to m.

3. <u>Truly Random Case</u>: If $\mathcal{D}$ is given query access to a truly random function R, then the probability that the last n bits of $R(m)$ equal m is 2^{-n}, where the probability is taken over the randomness of sampling R. This implies that $\Pr[\mathcal{D}^{R(\cdot)} = 1] = 2^{-n}$.

4. In summary:

   $$\left|\Pr[\mathcal{D}^{\textsf{Mac}(k,\cdot)} = 1] - \Pr[\mathcal{D}^{R(\cdot)} = 1]\right| = 1 - 2^{-n}$$

   which is non-negligible. Therefore, $\textsf{Mac}$ is not a secure pseudorandom function.

$\square$ $\square$