Exercise 7-1  Suppose $F$ is a pseudo-random function.

Define a fixed-length message-authentication code $(\text{Gen}, \text{MAC})$ as follows: The key generation function $\text{Gen}$ takes as argument the security parameter $n$ and returns a random key of length $n$. The function $\text{MAC}$ takes as input the key of length $n$ and a message $m$ of length $2n - 2$. It splits the message $m$ into two halves $m_0$ and $m_1$ and outputs $F_k(0m_0) \parallel F_k(1m_1)$.

Is this scheme secure? Prove your answer.

Solution (Sketch):

Recall the definition of “secure” from the lecture notes.

The message authentication experiment $\text{Mac-forge}_{A,\Pi}(n)$:

$\Pi = (\text{Gen}, \text{Mac})$ a MAC, $A$ an adversary, $n$ the security parameter

1. A random key $k$ is generated with $\text{Gen}(n)$.

2. The adversary is given $n$ and oracle access to $\text{Mac}_k()$, i.e. it can request arbitrary tags w.r.t $k$. The adversary eventually outputs a pair $(m, t)$.

3. The output of the experiment is defined to be 1 if
   (1) $\text{Mac}_k(m) = t$ and (2) $m$ has not previously been queried by $A$.

We define a MAC to be secure if for all adversaries
   the probability that $\text{Mac-forge}$ yields 1 is negligible.

We can define an adversary $A$ that succeeds in the experiment $\text{Mac-forge}_{A,\Pi}(n)$ with non-negligible probability.

The attacker requests the Mac for the two messages $0^{n-1}1^{n-1}$ and $1^{n-1}0^{n-1}$. It will therefore be given $F_k(00^{n-1}) \parallel F_k(11^{n-1})$ and $F_k(01^{n-1}) \parallel F_k(10^{n-1})$.

This means, in particular, that the attacker has learned $F_k(00^{n-1})$ and $F_k(10^{n-1})$.

The attacker then outputs the pair $(m, t)$ consisting of $m = 0^{n-1}0^{n-1}$ and $t = F_k(00^{n-1}) \parallel F_k(10^{n-1})$.

The attacker thus wins with probability 1, since the message $m$ had not been used previously, and the Mac is correct.
**Exercise 7-2** Recall from the lecture that CBC-MAC computes a message-authentication code from a message consisting of \( L \) equal-sized blocks \( m = m_1m_2 \ldots m_L \) using a pseudo-random function \( F \) as follows:

\[
\begin{align*}
t_0 &= F_k(L) \\
t_{i+1} &= F_k(t_i \oplus m_i) \quad \text{for } i = 0, \ldots, L - 1.
\end{align*}
\]

The message-authentication code for \( m \) is \( t_L \).

Show that this scheme becomes insecure if the code is taken to be \( t_0 \parallel t_1 \parallel \ldots \parallel t_L \) instead.

**Solution (Sketch):** Construct an attacker as follows.

Choose two words \( x \) and \( y \) of length \( n \) at random.

Request the CBC-MAC for \( xy \) from the oracle. This gives us \( t_0 = F_k(2) \), \( t_1 = F_k(t_0 \oplus x) \), \( t_2 = F_k(t_1 \oplus y) \).

Request the CBC-MAC for \( t_0y \) from the oracle. This gives us \( t'_0 = F_k(2) \), \( t'_1 = F_k(t'_0 \oplus t_0) = F_k(0^n) \), \( t_2 = F_k(t'_1 \oplus y) \).

We now know: \( t_0t'_1 \) will produce \( t''_0 = F_k(2) \), \( t''_1 = F_k(t''_0 \oplus t_0) = F_k(0^n) \), \( t_2 = F_k(t''_1 \oplus t'_1) = F_k(F_k(0^n) \oplus F_k(0^n)) = F_k(0^n) \).

Hence we output message \( t_0t'_1 \) and hash \((t_0 \parallel F_k(0^n) \parallel F_k(0^n))\).

This works if \( x \neq t_0 \) and \( y \neq t'_1 \). But the probability of \( x \) being \( F_k(2) \) is \( 1/2^n \) and the probability of \( y \) being \( F_k(0^n) \) is also \( 1/2^n \). But the adversary needs to win only with non-negligible probability, so this is still more than enough.

**Exercise 7-3** Consider the following changes to the Merkle-Damgård construction. In which of these cases does the construction still produce a collision-resistant hash function?

a) The message length \( L \) is not appended in the last step, i.e. the output is \( z_B \) instead of \( h_s(z_B \parallel L) \).

**Solution (Sketch):** This is not collision-resistant in general.

Consider the message \( 1^k \) of length \( k < l(n) \) (where \( l(n) \) is the length of a single block). In the construction, the last block is padded with zeros. In this case, \( x_1 = 1^k0^{l(n)-k} \).

But the message \( 1^k0^{l(n)-k} \) of length \( l(n) \) (this is a different message!) would have the same hash.

For messages that need not be padded, the scheme would be collision-resistant.

b) Instead of letting \( z_0 \) be a word of all zeros, one chooses some random word \( IV \) and sets \( z_0 := IV \). Then one computes \( z_B \) as before, i.e. \( z_i = h_s(z_{i-1} \parallel x_i) \) for \( i = 1 \ldots, B \), and returns \( IV \parallel h_s(z_B \parallel L) \) as the final output.
Solution (Sketch): We show that a collision for \( H \) (the function defined in this exercise) leads to a collision for \( h_s \). So if an attacker could find a collision for \( H \), then he could also find one for \( h_s \). If \( h_s \) is collision resistant, then the probability of this must be low. So the probability of finding a collision for \( H \) must also have been low.

Suppose \( H(x) = H(x') \) with \( x \neq x' \). By definition, this means \( IV \parallel h_s(z_B \parallel L) = IV' \parallel h_s(z'_B \parallel L') \). In particular, \( IV = IV' \) and \( h_s(z_B \parallel L) = h_s(z'_B \parallel L') \). So we are essentially in the same situation as in the original definition of MD (only with \( IV \) instead of 0\(^n\) in the initial step).

We can now construct a collision just like in the lecture.

- If \( L \neq L' \), then we have found a collision.
- Otherwise \( L = L' \).
  - If \( z_B \neq z'_B \), then we have found a collision.
  - If \( z_B = z'_B \), then we unfold the definitions \( z_B = h_s(z_{B-1} \parallel x_B) \) and \( z'_B = h_s(z'_{B-1} \parallel x'_B) \) to get \( h_s(z_{B-1} \parallel x_B) = h_s(z'_{B-1} \parallel x'_B) \). Now, if \( x_B \neq x'_B \) or \( z_{B-1} \neq z'_{B-1} \), then we have found a collision. Otherwise \( z_{B-1} = z'_{B-1} \), and we repeat this last step again. Since \( x \neq x' \), we will eventually arrive at a collision.

c) One completely omits the initial value \( z_0 \) and starts computation with \( z_1 := x_1 \). This means that one computes \( z_i = h_s(z_{i-1} \parallel x_i) \) for \( i = 2 \ldots , B \), and then returns \( h_s(z_B \parallel L) \) as the output.

Solution (Sketch): This scheme is also collision-resistant.

Consider the proof from b). If the messages \( x \) and \( x' \) differ in a block after the last, we will find a collision just like there. The last step in the proof there amounts to showing that if \( z_i = z'_i \) then one either has a collision or \( z_{i-1} = z'_{i-1} \), first for \( i = B \), then \( i = B - 1 \), and so on until \( i = 1 \). Since in b) we have \( z_0 = z'_0 = 0^n \), we must find a collision at some \( i \geq 1 \).

Here, we can proceed in exactly the same way, first for \( i = B \), then \( i = B - 1 \), and so on until \( i = 2 \). If we arrive at \( i = 2 \), then we must find a collision there, for otherwise we would have \( z_1 = z'_1 \). This is not possible: If we have not found a collision up until this point, then would mean that \( x_i = x'_i \) for all \( i = 2 \ldots , B \). But since \( x \neq x' \), then we must have \( x_1 \neq x'_1 \). But we also have \( z_1 = x_1 \) and \( z'_1 = x'_1 \) by definition, so we cannot have \( z_1 = z'_1 \).