Problem sheet 5 for Course 02231, 2025

These practice problems have the purpose of helping you understand the material better and learning the skills that are necessary to analyze cryptographic constructions, and sometimes to prepare you for the next class. All answers should be supported by a written justification. To gauge whether a justification is sufficient, ask yourself if your peers would be convinced by it without additional explanations.

As in the lecture, we write $k \leftarrow K$ if k is sampled from the set K such that it can be each element from K with equal probability $1/|K|$. For two strings x, y we use $x|y$ to denote the string obtained from concatenating x with y.

Exercise 1. (**Mix and Match for ECB-MAC**)

Assume that $\Theta = (Keygen, MAC)$ is a secure MAC scheme where $MAC : \{0,1\}^\lambda \times \{0,1\}^\lambda \to \{0,1\}^\lambda$.
In the class, we mentioned why ECB-MAC is a terrible idea: one can simply swap blocks in the message (and the MAC) and the result is again a valid message! A quick fix is to encode the block number as part of the message. For example, consider the following algorithm where $m_1, m_2 \in \{0,1\}^{\lambda-1}$:

$Keygen()$**:**

1. Output $k \leftarrow \Theta.Keygen()$

$ECB - MAC'(k, m_1|m_2)$**:**

1. $t_1 \leftarrow \Theta.MAC(k, 0|m_1)$
2. $t_2 \leftarrow \Theta.MAC(k, 1|m_2)$
3. Output (t_1, t_2)

1. Assume that you obtain two outputs $(t_1, t_2) \leftarrow ECB - MAC'(k, 0^{2\lambda-2})$ and $(t'_1, t'_2) \leftarrow ECB - MAC'(k, 1^{2\lambda-2})$. Can you find a message m such that $(t_1, t'_2) == ECB - MAC'(k, m)$?
2. Using your previous observation, construct an attacker which can distinguish $L_{MAC-real}$ from $L_{MAC-fake}$ for $ECB - MAC'$ with probability 1. Your attacker should make two queries to *Gettag* and one to *Checktag*.

Exercise 2.

As in the previous exercise, assume that $\Theta = (Keygen, MAC)$ is a secure MAC scheme where $MAC : \{0,1\}^\lambda \times \{0,1\}^\lambda \to \{0,1\}^\lambda$.
In the lecture, it was mentioned that CBC-MAC is a secure MAC for message space $M = \{0,1\}^{\lambda\ell}$. It is only secure *if all messages have the same length*. We will now see why this is the case. First, let us remember how $CBC - MAC$ works:

$Keygen()$**:**

1. Output $k \leftarrow \Theta.Keygen()$

$CBC - MAC(k, m_1|\ldots|m_\ell)$**:**

1. $t \leftarrow 0^\lambda$
2. For $i = 1, \ldots, \ell$
   (a) $t \leftarrow \Theta.MAC(k, t \oplus m_i)$
3. Output t

1. Assume that you set $\ell = 2$ and generate a MAC t for input $m_1|m_2$. What is the MAC of the message $m_1|m_2|(m_1 \oplus t)|m_2$?
2. Using the previous observation, construct an attacker which can distinguish $L_{MAC-real}$ from $L_{MAC-fake}$ for CBC-MAC with probability 1. Your attacker should make one query to *Gettag* and one to *Checktag*.

Exercise 3.

In this exercise, we will prove that *Encrypt-then-MAC* is a CCA-secure encryption scheme. For this, we assume that $\Omega = (Keygen, Enc, Dec)$ is a CPA-secure encryption scheme (for the left-right definition), that $\Theta = (Keygen, MAC)$ is a secure MAC scheme and that $\Omega.C \subseteq \Theta.M$ (meaning that the ciphertexts generated by Ω are valid messages which can be MACed by Θ). To recap, the resulting MAC-then-encrypt scheme Σ looks as follows:

$\Sigma.Keygen()$**:**

1. $k_E \leftarrow \Omega.Keygen()$
2. $k_M \leftarrow \Theta.Keygen()$
3. Output (k_E, k_M)

$\Sigma.Enc((k_E, k_M), m \in \Omega.M)$**:**

1. $c \leftarrow \Omega.Enc(k_E, m)$
2. $t \leftarrow \Theta.MAC(k_M, c)$
3. Output (c, t)

$\Sigma.Dec((k_E, k_M), (c, t))$**:**

1. If $t \neq \Theta.MAC(k_M, c)$ then output $\perp$
2. Output $\Omega.Dec(k_E, c)$

Our goal in the proof is to show that the libraries L^{Σ}_{CCA-0} and L^{Σ}_{CCA-1} are indistinguishable for any polynomial-time attacker. Here, we define L^{Σ}_{CCA-i} as

L^{Σ}_{CCA-i}**:**

$k \leftarrow \Sigma.Keygen()$

$S \leftarrow \varnothing$

$CTXT(m_0, m_1 \in \Sigma.M)$**:**

1. If $|m_0| \neq |m_1|$ then output $\perp$
2. $c \leftarrow \Sigma.Enc(k, m_i)$
3. $S \leftarrow S \cup \{c\}$
4. Ouptut c

$Decrypt(c \in \Sigma.C)$**:**

1. If $c \in S$ then output $\perp$
2. Output $\Sigma.Dec(k, c)$

1. Check that the encryption scheme Σ is correct. For this, you can assume that Ω is correct.

2. Write down the library L^{Σ}_{CCA-0} where you replace $\Sigma.Keygen, \Sigma.Enc, \Sigma.Dec$ with the algorithms that implement Σ, i.e. the algorithms from Θ and Ω. For example, you replace the line $k \leftarrow \Sigma.Keygen()$ with the two lines $k_E \leftarrow \Omega.Keygen()$ and $k_M \leftarrow \Theta.Keygen()$...

3. Write down the library L_{Step-1}. L_{Step-1} should be the same as L^{Σ}_{CCA-0}, but use a library $L^{\Theta}_{MAC-real}$ to perform the MAC computations instead of having the MAC algorithms being run inside L_{Step-1}. Argue why $L^{\Sigma}_{CCA-0} \equiv L_{Step-1} \circ L^{\Theta}_{MAC-real}$.

4. Argue why $L_{Step-1} \circ L^{\Theta}_{MAC-real} \approx L_{Step-1} \circ L^{\Theta}_{MAC-fake}$.

5. You can now rewrite $L_{Step-1} \circ L^{\Theta}_{MAC-fake}$ into a new library L_{Step-2} which does not use $L^{\Theta}_{MAC-fake}$ anymore (so the code of $L^{\Theta}_{MAC-fake}$ now resides inside L_{Step-2}). Argue why $L_{Step-1} \circ L^{\Theta}_{MAC-fake} \equiv L_{Step-2}$.

6. In L_{Step-2} you still have $\Omega.Dec(k_E, c)$ inside $Decrypt$. Consider the library L_{Step-3} which is the same as L_{Step-2}, but without computing $\Omega.Dec(k_E, c)$. Why is $L_{Step-2} \equiv L_{Step-3}$? For this, you can check under which conditions $\Omega.Dec(k_E, c)$ will be reached.

7. You can now construct a library L_{Step-4} which itself uses L^{Ω}_{CPA-0} to do the encryption. Argue that $L_{Step-3} \equiv L_{Step-4} \circ L^{\Omega}_{CPA-0}$.

8. Finally, show that $L_{Step-4} \circ L^{\Omega}_{CPA-0} \approx L_{Step-4} \circ L^{\Omega}_{CPA-1}$. Describe how to finish the proof from here.