Homework 3

Task 1 - Key Recovery (5 + 5 points)

(a)

oracle $O^{\mathsf{LR}_{b[\Pi]}}$:
private **procedure** Init():
$b \xleftarrow{\$} \{0,1\}$

public **procedure** Eval(M_0, M_1):
$C \leftarrow \mathsf{LR}_{b[\Pi]}.\mathsf{Encrypt}(M_0, M_1)$
return C

private **procedure** Fin(K'):
$M' \xleftarrow{\$} \mathcal{M}$
$C' \leftarrow \mathsf{LR}_{b[\Pi]}.\mathsf{Encrypt}(M', M')$
$M'' \leftarrow \mathsf{Dec}(K', M')$
if $M'' = M'$ **return** 1 **else return** 0

Given the oracle $O^{\mathsf{LR}_{b[\Pi]}}$ described, we can construct a distinguisher D to distinguish whether the oracle's behavior corresponds to left or right oracle with the help of A.
Distinguisher D does the following:

1. D chooses two messages $M_0 = 0^l$ and $M_1 = 1^l$.
2. D queries $O^{\mathsf{LR}_{b[\Pi]}}$'s Eval procedure with M_0 and M_1 to receive ciphertexts.
3. D then queries A to get a key K' based on the received ciphertexts.
4. Using the key K', D call $O^{\mathsf{LR}_{b[\Pi]}}$'s Fin procedure and receives a response, which is either 1 or 0.
5. If the output is 1, D return 0, D otherwise return 1.

   If $b = 0$, then the $\mathsf{LR}_{b[\Pi]}$ oracle will always encrypt the first message. So, if A has a high key recovery advantage, the key K' it produces be the correct one. When D queries the Fin procedure, the decryption of C' using K' result in M', leading the oracle to return 1.

   If $b = 1$, then the $\mathsf{LR}_{b[\Pi]}$ oracle will always encrypt the second message. Given A's advantage, the key K' will not be the correct one for the $b = 0$ case. When D queries the Fin procedure, the decryption of C' using K' will not result in M', leading the oracle to return 0.

   Thus, $\mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\Pi}(D) = 1$

Task 2 - When IVs Collide (8 points)

(a)

By the encryption procedure of CTR mode, we know:

$$C_0[i] \leftarrow M_0[i] \oplus \mathsf{E}(K, \mathsf{IV}_0 + i)$$

$$C_1[i] \leftarrow M_1[i] \oplus \mathsf{E}(K, \mathsf{IV}_1 + i)$$

Since $C_0[0] = C_1[0]$ (i.e. $\mathsf{IV}_0 = \mathsf{IV}_1$), we have:

$$C_0[i] \oplus C_1[i] = M_0[i] \oplus \mathsf{E}(K, \mathsf{IV}_0 + i) \oplus M_1[i] \oplus \mathsf{E}(K, \mathsf{IV}_1 + i) = M_0[i] \oplus M_1[i]$$

If we apply $C_0[i] \oplus C_1[i] = M_0[i] \oplus M_1[i]$ to each block, we have:

$$M_0 \oplus M_1 = C_0 \oplus C_1$$

Therefore, if the initialization vector for CTR mode collides, we can learn the value of the XOR of two plaintexts by XORing their ciphertexts.

(b)

By the encryption procedure of CBC mode, we know:

$$C_0[i] \leftarrow \mathsf{E}(K, M_0[i] \oplus C_0[i-1])$$

$$C_1[i] \leftarrow \mathsf{E}(K, M_1[i] \oplus C_1[i-1])$$

Since $C_0[0] = C_1[0]$ (i.e. $\mathsf{IV}_0 = \mathsf{IV}_1$), we have:

$$C_0[1] \leftarrow \mathsf{E}(K, M_0[1] \oplus C_0[0])$$

$$C_1[1] \leftarrow \mathsf{E}(K, M_1[1] \oplus C_1[0])$$

As we can see, if $M_0 = M_0[1]M_0[2]...M_0[q] = M_1[1]M_1[2]M_1[q] = M_1$ we have:

$$C_0[1] = E(K, M_0[1] \oplus C_0[0]) = E(K, M_1[1] \oplus C_1[0]) = C_1[1]$$

$$C_0[2] = E(K, M_0[2] \oplus C_0[1]) = E(K, M_1[2] \oplus C_1[1]) = C_1[2]$$

$$...$$

$$C_0[q] = E(K, M_0[q] \oplus C_0[q-1]) = E(K, M_1[q] \oplus C_1[q-1]) = C_1[q]$$

Therefore, when the initialization vector for CBC mode collides, we know that if two ciphertexts are the same, the two corresponding plaintexts are also the same.

Task 3 - IND-CPA Security (10 points)

(a)

Since we use PKCS#7 to pad plaintext M into 16-byte blocks, for plaintexts that are less than 16 bytes in length, we will only have one block of ciphertexts:

$$C_{M<16 \text{ bytes}} \leftarrow R||\mathsf{AES}(K, M[1] + 1 + R)$$

Since every time we encrypt a new plaintext, we generate a random 16-byte mask R, the probability of having different masks for two different plaintexts is very high (i.e. $1 - 2^{-128}$).
So for two plaintexts M_0 and M_1, the probability of $M_0[1] + 1 + R_0 = M_1[1] + 1 + R_1$ is very low (less than 2^{-128}).
Because we assume AES is a good PRF, given $M_0[1] + 1 + R_0 \neq M_1[1] + 1 + R_1$, AES should return two blockciphers C_0 and C_1 which is as good as directly sampling two uniformly random 16-byte strings (effectively a one-time pad).
Therefore, we can't tell which ciphertext is the encryption of which plaintext, so the scheme is IND-CPA secure.

(b)

Assume we pick two 32-byte long plaintexts M_0 and M_1 where after padding:

$$M_0[1] + 1 = M_0[2] + 2$$

$$M_1[1] + 1 \neq M_1[2] + 2$$

By the encryption scheme, we will have the following ciphertexts:

$$C_0 \leftarrow R||\mathsf{AES}(K, M_0[1] + 1 + R_0)||\mathsf{AES}(K, M_0[2] + 2 + R_0)$$

$$C_1 \leftarrow R||\mathsf{AES}(K, M_1[1] + 1 + R_1)||\mathsf{AES}(K, M_1[2] + 2 + R_1)$$

Since AES is deterministic, and we picked $M_0[1] + 1 = M_0[2] + 2$ and $M_1[1] + 1 \neq M_1[2] + 2$, we know:

$$\mathsf{AES}(K, M_0[1] + 1 + R_0) = \mathsf{AES}(K, M_0[2] + 2 + R_0)$$

$$\mathsf{AES}(K, M_1[1] + 1 + R_1) \neq \mathsf{AES}(K, M_1[2] + 2 + R_1)$$

Base on this, we can construct a distinguisher which can distinguish C_0 and C_1 with probability 1 by checking if the second and third block of the ciphertext are the same.
Therefore, the scheme is not IND-CPA secure for plaintexts that are longer than 16 bytes.

Task 4 - Padding-Oracle Attack (22 points)

(a)

Case 1 (if the last byte of the last plaintext block is validly padded):
For all possible guesses X:

1. Change the last byte Y of the second-last ciphertext block to $X \oplus Y \oplus 01$
2. Submit resulting ciphertext to padding oracle. If padding oracle says ciphertext has valid padding (we should only have one X that satisfy this condition), take X as the value of the last byte.

Case 2 (if the last byte of the last plaintext block is not validly padded):
For all possible guesses X :

1. Change the last byte Y of the second-last ciphertext block to $X \oplus Y \oplus 01$.
2. Submit the resulting ciphertext to the padding oracle. If the padding oracle says the ciphertext has valid padding, make a note of this value of X .

   After searching through all $X \in \{0,1\}^8$, we'll likely have identified two values X_1 and X_2 that satisfy the condition.

   [To show this, we can use the example in class:
   If the original plaintext is not padded correctly, let's say the last byte was $0B$, then after searching through all possible X's, we will get two values that will achieve the two valid paddings: (i.) $0A$ (ii) 01

   To find correct value between X_1 and X_2, we do the following:
3. Change the second-last byte Z of the second-last ciphertext block to a different random value (or we can simply flip the last bit of Z).
4. Change the last byte Y of the second-last ciphertext block to $X_1 \oplus Y \oplus 01$.
5. Submit this resulting ciphertext to the padding oracle.
6. If the padding oracle says the ciphertext has valid padding, then X_2 is the correct value for the last byte (because if X_1 was the correct value, changing the second-last byte Z would have invalidated the padding). Otherwise, X_1 is the correct value for the last byte (because it suggests the correct padding is just 01, changing the second-last byte didn't break anything).

(b)

Strategy:

1. Split the ciphertext C into blocks: $C_0, C_1, ...C_n$ (target is to recover $P_1, P_2, ..., P_n$).
2. For each block C_i starting from C_n and moving to C_1: recover each byte in the block, starting from the last byte and moving to the first.
3. For each byte position in block C_i:
   Assume we have $C_i = [c_1, c_2, ...c_{16}]$ and we want to recover $P_i = [p_1, p_2, ...p_{16}]$.
   (a) Using the strategy described in part a) to recover the last byte of the last plaintext block (p_{16}).
   (b) Recovering the second-last byte (p_{15}):
   The target is to change C_i to C_i' such that when decrypted, its last two bytes become 0202.
      i. Change the already recovered last byte: $c'_{16} = c_{16} \oplus p_{16} \oplus 02$ (so the last byte will be decrypted to 02).
      ii. For all possible guesses X:
         A. Change the second-last byte: $c'_{15} = X \oplus c_{15} \oplus 02$.
         B. Submit resulting ciphertext $C_i' = [c_1, c_2, ..., c'_{15}, c'_{16}]$ to padding oracle.
         C. If padding oracle says ciphertext has valid padding, then $p_{15} = X$.
   (c) Continue this process for each preceding byte:
      i. For the 3rd last byte, target decryption of the last three bytes becomes 030303.
      ii. For the 4th last byte, target decryption of the last four bytes becomes 04040404.
      iii. ... and so on.
   (d) Once all bytes in P_i are recovered, we move to the next block $P_{i-1}$and use C_{i-1} to recover it.
4. Repeat step 3 for all blocks.
5. Once recovered the last block (P_1), we remove the PKCS#7 padding from the final plaintext block to get the original message.

   Number of padding validity checks:
   In the worst case scenario, for each byte position, we have to check $2^8 = 256$ possibilities. Given a block size of b bytes and n blocks we need $b \times 256 \times n$ checks in the worst case.