Problem 1: “Inverse” CBC

Consider the following mode of operation (which I call “inverse CBC”):

To encrypt a message $m$ consisting of blocks $m_1, \ldots, m_n$ with key $k$, pick a random initialization vector $iv$ and then compute $c_1 := E_0(k, m_1) \oplus iv$ and $c_i := E_0(k, m_i) \oplus m_{i-1}$ for $i = 2, \ldots, n$. Here $E_0$ is the block cipher. And $E(k, m) := iv \| c_1 \| \ldots \| c_n$.

The adversary has intercepted a ciphertext $c = E(k, m)$. He happens to know the last block $m_n$ of $m$ (e.g., because that one is prescribed by the protocol).

1. Explain how the adversary can completely decrypt $m$. He can make chosen plaintext queries (i.e., he can ask for encryptions of arbitrary message $m'$). He cannot make decryption queries.

   **Hint:** First think how you can, e.g., find out $E_0(k, m_n)$ by performing an encryption query $E(k, m_n)$.

   **1.1 Solution**

   By requesting the encryption of a one block message $m'$, the adversary can get $(iv', E_0(k, m') \oplus iv')$ for some $iv$. From this he can compute $E_0(k, m')$. Thus the adversary can get $E_0(k, m')$ for any $m'$.

   Given $c = iv \| c_1 \| \ldots \| c_n$, the adversary proceeds as follows: By assumption, the adversary already knows $m_n$. For $i = n-1, \ldots, 1$, he computes $m_i = E_0(k, m_{i+1}) \oplus c_{i+1}$. Then he knows $m = m_1 \ldots m_n$.

   Optionally, he can add a check whether things went right (i.e., whether his initial guess of $m_n$ was correct) by checking whether $c_1 = E_0(k, m_1) \oplus iv$.

2. Suggest how to fix the mode of operation so that it becomes secure at least against this attack (and simple modifications thereof). You do not need to prove security.

   **1.2 Solution**

   There could be many possibilities. We just give one as an example:

   One could encrypt each ciphertext block once more using $E_0(k, \cdot)$ (or possibly using a different key). Of course, this doubles the number of encryptions needed.

Problem 2: Breaking ECB

In the lecture we have seen that encrypting a file with ECB mode is not very secure. For example, if an uncompressed image file is encrypted, the result may still reveal much of the picture to the naked eye.

In this exercise, we consider the task of distinguishing the encryption of two given messages $m_0, m_1$ automatically. That is, assume that two messages $m_0, m_1$ (English texts)
of the same length are given and known to the adversary. Furthermore, the adversary
learns $c$, which is the ECB encryption of $m_0$ or $m_1$ (using a random and unknown key $k$). The adversary is now supposed to guess which message was encrypted. (I.e., we have
a known plaintext attack, not a chosen plaintext attack.)

1. Describe an algorithm that finds out (given $m_0$, $m_1$, $c$) whether $m_0$ or $m_1$ was
encrypted. It should work on “typical” text files. (That is, it should not require,
e.g., one of the text files to contain only spaces or similar.)

Example of “typical” text files are `ecb-distinguish-1.txt` and `ecb-distinguish-2.txt` from the lecture webpage.

2. (Bonus points) Implement the algorithm. That is, fill in the missing code for the
function `adv` in the code below (also available on the lecture webpage):

2.1 Solution

There could be many different solutions that work.

My solution (for which source code is given below) is the following:

- Given a given ciphertext $c$, we count which block occurs how often. (I.e., we make
  a histogram of the block occurrences.) That is, we maintain a map that maps block
  content to number of occurences.

  Notice that for different keys $k$, the histogram will be indexed by different block
  contents, but the number of occurrences are independent of the key $k$.

- We count which number of occurrences in the first histogram occurs how often.
  That is, we make a second histogram that is a map from number of occurrences to
  number of numbers of occurrences. (Uff.)

  Now for a given plaintext, no matter which key we use, we always get the same
  second histogram.

- We encrypt both plaintexts $m_0$, $m_1$ with some arbitrary key $k$, get the second
  histogram of each ciphertext.

- When given a ciphertext $c$, we compute its second histogram and check whether it
  is the same as that coming from $m_0$, or from $m_1$.

Other possible solutions that may have slightly lower success rate but should work
for long texts, including the examples provided (I did not test this):

- Count how many different blocks there are.

- Count how often the most common block occurs.

- Count how many blocks there are that occur only once.

- Many more, probably.