CS 435: Intro to Cryptography **Fall 2020**

Homework 4 Solutions

Professor Somesh Jha **Due:** Nov 5, 2020

1. Show the decryption logic for the four modes of operations ECB, CBC, OFB, and CTR.

   Solution:
   The formulas for decryption are as follows:

   (a) ECB $\rightarrow m_i = F_k^{-1}(c_i)$

   (b) CBC $\rightarrow m_i = F_k^{-1}(c_i) \oplus c_{i-1}$

   (c) OFB $\rightarrow m_i = c_i \oplus o_i$ where $\begin{cases} o_0 = IV \\ o_i = F_k(o_{i-1}) \end{cases}$

   (d) CTR $\rightarrow m_i = F_k(ctr + i) \oplus c_i$

   For a detailed diagram of the encryption/decryption logic check out this link:
   `https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation`

2. Consider a stateful variant of CBC-mode encryption where the sender simply increments the IV by 1 each time a message is encrypted (rather than choosing IV at random each time). Show that the resulting scheme is not CPA-secure.

   Solution:
   We construct an adversary $\mathcal{A}$ that wins the CPA-indistinguishability game with non-negligible advantage. By Kerckhoff's principle, we assume that $\mathcal{A}$ already knows the block length n, and $\mathcal{A}$ will use messages of length n.

   (a) $\mathcal{A}$ chooses an arbitrary message m and queries the oracle to get $(IV; c)$

   (b) $\mathcal{A}$ chooses two messages: $m_0 = m \oplus IV \oplus (IV + 1); m_1 =$ any arbitrary message distinct from m_0.

   (c) $\mathcal{A}$ sends them over to Π.

   (d) Π returns $(IV + 1; c')$

   (e) If $c' = c$, A outputs $b = 0$, else outputs b = 1

   Note that when we send m_0 and m_1, (IV + 1) is used as the initialisation vector, and we are only encrypting a single block. Therefore if m_0 had been encrypted,

$$\begin{aligned} c' &= F_k(m_0 \oplus (IV + 1)) \\ &= F_k(m \oplus IV \oplus (IV + 1) \oplus (IV + 1)) \\ &= F_k(m \oplus IV) \\ &= c \end{aligned}$$

If m_1 had been encrypted by Π, then $c' \neq c$ unless $F_k(m_1 \oplus (IV+1) = F_k(m \oplus IV))$, and for arbitrarily chosen m_1 and m, $Pr[F_k(m_1 \oplus (IV+1)) = F_k(m \oplus IV)]$ is negligible.

3. What is the effect of a single-bit error in the ciphertext when using the CBC, OFB, and CTR modes of operation?

   Solution: A single bit error in the cipher text basically corrupts a single cipher block. According to the decryption formulas (see the solution for question 1), CBC is the only mode that requires another ciphertext block, namely c_{i-1}, to decrypt c_i. Therefore, in CBC mode, the corrupted block will affect the decryption of the next block (all the other blocks can be decrypted normally). In other modes, the corrupted block has no effect on the decryption of other blocks. In conclusion, if a ciphertext block ck is corrupted, CBC will fail to decrypt 2 blocks (c_k and c_{k+1}), whereas OFB and CTR mode will fail to decrypt only one block (c_k).

4. Towards the end of LectureLet-16 we covered timing attacks for MACs. Let us say a message m has tag t which has m bytes, Let it take T milliseconds(ms) to compare equality of two bytes. In LectureLet-16 we only explained how to get two bytes of the tag t. Describe the full attack (i.e. recovering all the m bytes of the tag t). How much time in ms does the attack take?

   Solution:
   We will consider the worst case scenario. Since we compare starting from the rightmost byte, which has 256 possibilities, attacker will send 256 tags, one for each possibility of the last byte. Each of these tags will have the last byte different (to figure the last byte, the first $m-1$ bytes are irrelevant and can be anything). So there will be 256 one-byte comparisons for the last byte, each of which takes T ms, thus taking $256 \times T$ ms. The tag with the correct last byte will take longer to get rejected, to be precise it will take T ms more to get rejected, because the next byte will be also be compared if the previous one is correct. So it will take $(256 \times T) + T$ ms in total,to figure out the last byte of the correct tag.

   Now for the second last byte, we again have 256 possibilities. So the attacker sends 256 tags, the last byte of all those tags will be the correct byte found earlier, and the second last byte will run through all the 256 possibilities.There will be 256 two-byte comparisons for the last two bytes, because the attacker has to compare the last byte as well. Again for the correct byte, there will be one extra comparison. So this would take $(256 \times 2 \times T) + T$.

   Using the same strategy attacker will figure out all the m bytes of the tag, and it will take (in ms):

$$256 \times (1 + 2 + ... + m) \times T + (m-1)T$$

$$= 256 \times (\frac{m(m+1)}{2}) \times T + (m-1)T$$
$$= 128 \times (m(m+1)) \times T + (m-1)T$$