C2107 Tutorial 2 Answer(Encryption)

E.-C. Chang, School of Computing, NUS

August 19, 2025

1. This is an example of the ciphertext xor-ing attack described in Lecture 1:
   $C_1 \oplus C_1$ (by omitting the IV) $= P_2 \oplus P_3 = 11110000$.
   Notice that the attack is applicable because: (a) a stream cipher is employed; (b) the same secret key and IV are used for generating the two ciphertexts.

2. For each pair of k_1 and k_2, we need 2 encryptions and 2 decryption. There is a total of 2^{112} pairs. Using a straightforward method of applying 2 encryptions and 2 decryptions for each pairs, the overall is 2^{114} cryptographic operations.

   We can do slightly better than that. One can exhaustively enumerate k_1, and for each k_1, exhaustively search all k_2. So, the number of encryptions will be (number of encryptions using k_1) + (number of encryptions using k_2) $= 2^{56} + 2^{112} \approx 2^{112}$. We need the same number of operations for k_3 and k_4. So total is approximately $2^{112} \times 2 = 2^{113}$.

   Remarks:

   (a) If applied 3 times, meet-in-the-middle needs approximately 2^{112} operations. So, increase from 3 to 4 times only increase the "difficulty" from 2^{112} to 2^{113} (or "bit-strength" from 112 to 113).

   (b) What about applying $2t - 1$ times vs applying it $2t$ times, when $t = 3, 4, \ldots$.

3. (Remark: The question doesn't state precisely what type of info can be sniffed from the communication channel. From the context, it should be clear that the sniffed data are the ciphertext.)

   - The block size for
     - "buy" message is 1.
     - "sell" message is 1.
     - "sell_everything" message is 2.
     - "hold_and_see" message is 2.

     So, including IV, the ciphertext size corresponds to "buy" is 2 blocks and so on. Now simply from ciphertext block size, the attacker can know whether the message is in {sell, buy} or in {sell_everything, hold_and_see }. Using ciphertext's size as extra information is an example of side-channel attack.

     This leakage present in both CBC and CTR setting.

(a) (**CBC** mode). Due to the re-start and the way IV is generated, attacker can get a few ciphertexts with the same IV. Note that IV is sent in clear and so attacker will know whether the IVs of two ciphertexts are the same.

Let's consider many ciphertexts (including IV) with the same block-size 3 that are captured over past few days with the same IV. The first block is the IV, and the 2nd, 3rd are output from AES.

Let's consider a particular IV, and the ciphertexts with that IV. The attacker groups those ciphertexts whose 2nd block are exactly the same in a group. Next, the attacker declares that ciphertext in the group corresponds to "sell_everything" ; and ciphertext not in the group as "hold_and_see".

Remark:

- The above works because due to same IV, the ciphertext is the same iff the plaintext is the same. "sell_everything:" is 16 bytes and fit into a block. Furthermore, it appears as first block in the plaintext. By CBC, they will be encrypted to the same ciphertext. However, "hold_and_see:" is less than 16 bytes and will be padded with the days that likely to be different. So the ciphertext likely to be different. Note that the 3rd block would be different.
- Unfortunately, the above can't apply to "buy" and "sell".
- Whether attacker know the date or not would not affect the outcome.
- (minor remark) There could be some cases that due to the same "day" in the date, the first block of "hold_and_see:" are the same for two different capture. In such cases, attacker would observe two groups of ciphertexts having the same first block. Likely that the larger on is "sell_everything:".

(b) **(CTR mode).** Drawing a figure would be much clearer for this question. Suppose two instructions are consecutively encrypted, and the first ciphertext consists of 3 blocks (including the IV). Let the first ciphertext be (v, c_1, c_2) and the corresponding plaintext is $p_1 \| p_2$. Let the third ciphertext be $(v+1, c_3)$ or $(v+1, c_3, c_4)$ depending its size, and the corresponding plaintext of c_3 is p_3.

Note that by CTR mode, $c_2 = p_2 \oplus \text{AES}(v+2)$. Furthermore, $c_3 = p_3 \oplus \text{AES}(v+2)$. That is, the "iv" is the same. So, similar to the "zebra" example in lecture note, the attacker can obtain $p_2 \oplus p_3$.

4. $v = IV \oplus \langle 0,0,0,0,0,0,0,0,t,08,08,\texttt{F7},\texttt{0C},\texttt{0C},\texttt{0C},\texttt{0C}\rangle$ and output $t \oplus 08$. That is,

   (a) For t= 0 to `FF`

(b) let $v = IV \oplus \langle 0,0,0,0,0,0,0,0,t,08,08,\texttt{F7},\texttt{0C},\texttt{0C},\texttt{0C},\texttt{0C}\rangle$.

(c) send $v \| c$ to oracle.

(d) If `yes`, output $t \oplus 08$.

To understand the algo, you might want to draw the decryption process (i.e. the flow chart of $D_k(c) \oplus v$), assume b_9 is some value (e.g. 04) and then trace the loop.

5. Assume the ciphertext (including the IV) is three blocks. Unlike the lecture note, here, we modify the 2nd block of the ciphertext instead of the IV.

   Method 1: Assume that the length is i, and test it. Test one by one starting from 1,...,16.

   (a) for i in 1,2,3...,16.

   (b) Change the (i)-th byte of the 2nd block and query the oracle. If the reply is "invalid", declare the number of padded bytes to be $17 - i$ and break from the loop.

   Method 2: Improved version. Use binary search to find the i that goes from valid to invalid.
   (optional remark: can't do better than binary search. There are 16 possible outcomes, and every query would eliminate at most halve.)

6. There are many. E.g. CVE-2023-41097. "*An Observable Timing Discrepancy, Covert Timing Channel vulnerability in Silabs GSDK on ARM potentially allows Padding Oracle Crypto Attack on CBC PKCS7.This issue affects GSDK: through 4.4.0.*"