Homework 4

1 Problem 1 (10 points)

For many block cipher encryption modes such as CBC mode, messages need to be a multiple of the block size. Messages that are not a multiple of the block size can still be encrypted, but need to be padded to a multiple of the block size. The padding moreover needs to be reversible so that the receiver can recover the original (unpadded) message when decrypting. For each of the following padding schemes, decide if the padding is reversible: that is, for any message, after padding to a multiple of the block length, it is possible to recover the message again. If the padding is reversible, explain how to recover the message and why recovery is guaranteed to work. If not, explain how it fails.

(a) **Null Padding**: Append 0's to the message until it is a multiple of the block length

(b) **Bit Padding, version 1:** Let N be the number of bits necessary to pad to a multiple of the block length. If $N > 0$, append 10^{N-1} — that is, a 1 followed by $N-1$ 0's — to the message. If $N = 0$ (the message is already a multiple of the block length), do nothing.

(c) **Bit Padding, version 2:** This is the same as part (b), except that in the case $N = 0$, we append an entire block, set to 10^{B-1}, where B is the block length in bits.

(d) **PKCS7 Padding:** Assume the message is an integer number of bytes, but not an integer number of blocks. Let N be the number of *bytes* necessary to pad to a multiple of the block length. To ensure that $N > 0$, if the message is already a multiple of the block length, let N be the block length (in bytes). Now pad with N bytes, each byte set to the value N. For example, if $N = 3$, append 3 bytes to the message, each byte set to 00000011.

(e) PKCS7 padding, except that if the message is already a multiple of the block length, do not add any padding.

2 Problem 2 (20 points)

Consider CBC mode encryption, using PKCS7 padding from Problem 1d. We now will see a potential attack on CBC mode encryption using this padding. Suppose Alice is sending messages to a server, encrypted with CBC mode using PKCS7 padding. The server decrypts the CBC encryption, and then checks that the padding is correct: it verifies that there is some $N > 0$ such that the last N bytes of the plaintext (after decrypting, but before stripping the padding) are set to N. If correct, the server does nothing. If not, the server sends a "reject" message back to Alice indicated that the message was not well-formed.

Suppose an adversary Eve sits between Alice and the server. Eve can intercept messages from Alice, as well as send messages to the server, and read the response from the server. However, Eve does not know the secret key Alice and the server are using to communicate. Here is a sketch of Eve's attack:

- Eve intercepts a ciphertext c from Alice.
- Eve has a guess g for the last byte in the last full block of plaintext. In other words, if the message length is $\ell s + t$ bytes, where ℓ is the number of bytes per block and $t < \ell$, then Eve has a guess g for byte ℓs.
- Eve first strips off the last block of c, obtaining c'
- Next, Eve changes some portion of c', obtaining c''.
- Eve sends c'' to the server, and based on the server's response, learns whether or not g was a correct guess.

Answer the following:

(a) Fill in the details of the attack, namely decide how Eve computes c''. [Hint: you only need to modify one byte of c']

(b) Extend the attack to actually learn byte ℓs given no other information about the message, by making multiple queries to the server. [Hint: there are only 256 possible values for that byte]

(c) Under what circumstances will the attack in (a),(b) not be guaranteed to give the right answer? Can Eve tell if this situation arises?

(d) Explain how to still recover the byte, even if the attack in (a),(b) was inconclusive.

(e) Explain how to extend the attack to recover the entire message. This includes learning the first ℓs bytes, as well as the last t bytes.

(f) But wait! We said that CBC encryption was CPA secure, and yet we just showed how to recover the entire message. Explain why this is not a contradiction.

3 Problem 3 (20 Points)

Let $(\mathsf{MAC}, \mathsf{Ver})$ be a *strongly* CMA-secure message authentication code. For each of the following, decide whether the scheme is (1) strongly CMA-secure, (2) weakly CMA-secure, or (3) neither. For case (1), prove strong CMA security, assuming the strong CMA security of $(\mathsf{MAC}, \mathsf{Ver})$. For (2), prove weak CMA security, and demonstrate an attack against strong CMA security. For (3), demonstrate an attack against weak CMA security.

(a) $\mathsf{MAC}_a(k, m) = \sigma'$, where σ' is obtained by computing $\sigma \leftarrow \mathsf{MAC}(k, m)$ and deleting the last bit of σ. $\mathsf{Ver}_a(k, m, \sigma')$ does the following: run $b_0 \leftarrow \mathsf{Ver}(k, m, \sigma'||0)$ and $b_1 \leftarrow \mathsf{Ver}(k, m, \sigma'||1)$. Output $b_0 \vee b_1$ (that is, output 1 if and only if at least one of b_0, b_1 are 1).

(b) $\mathsf{MAC}_b(k, m) = (0, \mathsf{MAC}(k, m))$, where 0 is a single bit. $\mathsf{Ver}_b(k, m, (b, \sigma)) = \mathsf{Ver}(k, m, \sigma)$

(c) $\mathsf{MAC}_c(k, m) = (0, \mathsf{MAC}(k, m))$, where 0 is a single bit. $\mathsf{Ver}_c(k, m, (b, \sigma)) = (b == 0) \wedge \mathsf{Ver}(k, m, \sigma)$. That is, accept a tag (b, σ) if an only if $b = 0$ and σ is valid according to $(\mathsf{MAC}, \mathsf{Ver})$.

(d) $\mathsf{MAC}_d(k, m) = \mathsf{MAC}(k, m')$, where m' is m with the last bit removed. $\mathsf{Ver}_d(k, m, \sigma) = \mathsf{Ver}(k, m', \sigma)$, where again, m' is m with the last bit removed.

4 BONUS Problem 4 (5 points)

Bonus problem: Show that CCA security does not imply authenticated encryption. That is, give an example of an encryption scheme that is CCA secure, but which is not authenticated encryption. You may assume a secure block cipher, or any objected that is derivable from a secure block cipher (a CMA-secure MAC, an authenticated encryption scheme, etc).