Blake
Johnson

Homework 2

Task 1:

a) The only difference between the two cipher texts is that one of the bits has been flipped. The 131st bit has been flipped in the second cipher text. From this we know that since the first 16 bytes are the same, the same iv was used for the two messages, and therefore the 1 bit difference means that the messages are the same except for that bit, due to the deterministic characteristics of ctr mode.

b) Knowing this information can be problematic because if you know that the iv doesn't change, you can figure out one of the plaintexts if you xor the cipher texts together and you know one of the messages. You also know that since the cipher texts only differ by one bit, the plaintext messages also only differ by the same bit.

c) Since the 3DES encryption is only using 64 bits, the amount of messages is relatively small, and therefore the same random counter will statistically be used after 2^(64/2) blocks, whereas 256 bit AES encryption will be able to send significantly more messages before it reaches a birthday bound.

Task 2:

a) Since AES iv is the first 16 bytes of the message, we know that the iv is:
FFBC1ADC607ACDDEAE7D837FA8123A3A
Since the plaintext is 8 bytes, and the iv is 16 bytes, there are still 8 (padding) bytes so we know the padding is 0808080808080808. Since there is only one plaintext block, the Ci-1 is equal to C0 which equals the iv. The corresponding byte of the iv is 3A. Therefore 3A xor 01 xor 08 = 33
Original:
FFBC1ADC607ACDDEAE7D837FA8123A**3A**9CFDD83A9CD55F15A8CD7F8CFA32A67B
Change:
FFBC1ADC607ACDDEAE7D837FA8123A**33**9CFDD83A9CD55F15A8CD7F8CFA32A67B

b) Original:
FFBC1ADC607ACDDEAE7D837FA8123A**3A**9CFDD83A9CD55F15A8CD7F8CFA32A67B
Change:
FFBC1ADC607ACDDEAE7D837FA8123A**44**9CFDD83A9CD55F15A8CD7F8CFA32A67B
This change will likely result will not likely decrypt into a valid plaintext because this padding will not end in 08 or 01

Task 3:

a) Since the message is known to you, and h is known to you, we can extend the Merkle-Damgard paradigm to add extra length to the message. Since we already know what H(K || M) is, we can use that value as the input for an arbitrary amount of extra blocks to compute a new tag H(K || M || L), that is valid regardless of which hash function is used. This will extend the message, and this can be sent with a valid tag because the tag is just the output H(K || M || L). In this case, knowing the message would have no extra benefit, because the key remains secure. We cannot change the original message other than to add to it because if we do, the previous hash H(K || M) would be incorrect, leading to an invalid tag. All we need to compute the new tag is the hash of the old message, which we use to hash new blocks to append to the message, with the new hash becoming the new tag.
b) HMAC fixes this vulnerability because it requires an extra key. The second key is derived from the secret key, and is used for a second hash of the message. The first hash would be similar to H(K || M) but when you try to hash the second time with the result of the first hash and the second key, you would not be able to hash a new appended tag. An extra input is required for the final hash that is reliant upon the first hash, so that you cannot make valid tags with a length extension attack.

Task 4:

a) 7eb4b13b8c4cd9001523ba1e55dc2cd5608e84c093cd21d1126ddac1e7b2a5e9 must be the tag because it is the same for both ciphers, and it is also the last 32 bytes. This means that the plaintexts for the ciphers must be the same as well because the tag is deterministic and although it is possible there is a hashing collision, it is highly, highly unlikely.

b) E&M can never be semantically secure. The definition of being semantically secure is that there is no extra information other than the length of the message that can be known from the cipher text. Encrypt & MAC is deterministic, and therefore we can know if two plaintexts are the same if their tags are the same. Encrypt THEN MAC can be semantically secure, because the tag will be computed based off of the pseudo random encryption of the plaintext instead of the plaintext itself

c) As long as the hash function is secure(don't use sha-256 or sha-512 or legacy hashes), Encrypt & Mac does satisfy integrity, however it does not satisfy confidentiality or semantic security. Even though you know something about the message based on the tag, you still will not be able to modify the original message without the key. Encrypt THEN Mac is much better though, because it satisfies integrity, confidentiality, and is semantically secure.

Task 5:

a) When xor'ing the last byte of the second-last block to 0x01 and X, the value of the last byte of the padded plaintext, there may be two values that lead to a valid decryption, but there doesn't need to be two. This is because 01 will grant a validly-padded plaintext, but if the plaintext is padded with more than 1 byte, there will also be another possible value for which the padded oracle will return true when xor'd with 0x01 and the padded plaintext.

b) If two valid paddings are found for X1 and X2, you can do a validity check with the second to last byte of the block. If it's the same as the X-value that produces a padding other than 01, you know that that X-value is the correct one, otherwise it's the X-value that produces the 01 padding.

c) This attack requires only 256 trials per byte, so in order to decrypt the entire message, there is a dependency on the length of the message. This process is linear with more cipher text blocks (256*number of bytes at worst case)