Question 1 (5 points)

A block cipher with an 8-bit block size is very easy to break with a known-plaintext attack (assuming each block is just encrypted independently with the same key). Describe how you would do so.

Answer:

an 8-bit block size will only have 256 possible entries (plaintext blocks and ciphertext blocks). Known-plaintext attacks use a bunch of plaintext-ciphertext pairs to build a table out of the plaintext and map it to a corresponding ciphertext. Similarly to the cryptogram example given, this would be like determining that every “A” will give you a “D” when run through the cipher. With enough pairs, you can break the cipher and determine what the plaintext was when given a ciphertext. This is possible because block ciphers are deterministic–meaning that the same plaintext block and key will always generate the same ciphertext output.

Question 2 (10 points)

Assume you're sending a long message using a block cipher (like AES) with the following scheme: split the message into block-sized chunks, then encrypt each with the same key. Basically Alice sends Bob AES(m1, k), AES(m2, k), AES(m3, k), etc.

- Part A (3 points): Even if they can't decrypt blocks, what information can an eavesdropper discern from this scheme? Hint: Imagine that Alice is sending a table of data where each cell is exactly one block of data.
    - The attacker can discover the length of the message by looking at the number of blocks being transmitted
    - Since the blocks are fixed-size they could determine the block size
    - If there are any repeated-blocks or data that would be picked up on
    - Could maybe detect something about the structure of the message (is there a header? Something like that)
- Part B (4 points): Things are actually even worse! A malicious attacker can actually CHANGE the message that Bob receives from Alice (slightly). How? This is particularly bad if the attacker knows the structure of the data being sent (like in part A).
    - Most likely, this will be done using a bit-flipping attack. A bit-flipping attack will change the content of the message by strategically changing a few bits. If the interceptor knows the structure of the message this will be

particularly bad because they can modify it in a way that still makes sense to the receiver (bob)

- Part C (3 points): How could you modify the scheme to mitigate/prevent these types of attack?
    - A different key should definitely be used for each block to help with this. Maybe one where the key is dependent on the previous block, so that any flipped bits are easier to detect. (a mode of operation)
    - Using some kind of pseudo-randomness may also be useful. Maybe you can use the pseu-random numbers to change the size of the blocks with each message so it is less predictable and the message is harder to attack?

Programming Part 1: A (bad) block cipher (25 points)

- Try modifying 1 bit of the ciphertext and then decrypting with the correct passwords. What do you see?
    - A portion of the correct message comes through with the decryption, but not all of it.

Programming Part 2: RC4 Cipher (20 points)

Verify that encrypting 2 messages using the same keystream is insecure. What do you expect to see if you xor the two encrypted messages?

- You see the same thing as if you had xor'd the two plaintext messages together. This can reveal some information about the messages and I think if you did this enough time you could break the cipher by determining the keystream.

Modify part of a message using a bit-flipping attack. For example, try sending the message "Your salary is $1000" encrypted with RC4. Modify the ciphertext so that when decrypted is says that your salary is $9999, instead. Hint: This should just require using xor.