Homework 3

Please submit your homeworks through CS Dropbox:
https://dropbox.cs.princeton.edu/COS433_S2018/HW3

1 Problem 1 (25 points)

Consider the following notions of security for encryption schemes.

(i) **Left-or-Right (LoR) Indisitnguishability.** This is the notion of security we saw in class

(ii) **Real-or-Random Plaintext (RoRP) Indistinguishability.** This security notion is defined by the following experiment. The adversary makes polynomially-many queries to the challenger on messages m in the message space. The challenger responds to the queries as follows. If its input bit is $b = 0$, then the challenger encrypts m to get a ciphertext c, which it returns to the adversary. If the challengers input bit is $b = 1$, then the challenger chooses a new random message m' and encrypts m' to get a ciphertext c, which it then returns to the adversary. Security is defined in the usual way: a scheme has (t, q, ϵ)-RoRP indisitnguishability if, for all adversary's A running in time t and making at most q queries, the advantage of A in the experiment above is at most ϵ.

(iii) **Real-or-Random Ciphertext (RoRC) Indistinguishability.** This security notion is defined by the following experiment. The adversary makes polynomially-many queries to the challenger on messages m in the message space. The challenger responds to the queries as follows. If its input bit is $b = 0$, then the challenger encrypts m to get a ciphertext c, which it returns to the adversary. If the challengers input bit is $b = 1$, then the challenger chooses a random string c in the ciphertext space C, which it then returns to the adversary. (t, q, ϵ)-RoRC security is defined in the natural way.

(iv) **Real-or-Zero (RoZ) Indistinguishability.** This security notion is defined by the following experiment. The adversary makes polynomially-many queries to the challenger on messages m in the message space. The challenger responds to the queries as follows. If its input bit is $b = 0$, then the challenger encrypts m to get a ciphertext c, which it returns to the adversary. If the challengers input bit is $b = 1$, then the challenger encrypts $m' = 0$ to get a ciphertext c, which it then returns to the adversary.

(v) **Real-or-Complement (RoC) Indistinguishability.** This security notion is defined by the following experiment. The adversary makes polynomially-many queries to the challenger on messages m in the message space. Assume the message space is $0,1^n$ for some n. The challenger responds to the queries as follows. If its input bit is $b = 0$, then the challenger encrypts m to get a ciphertext c, which it returns to the adversary. If the challengers input bit is $b = 1$, then the challenger encrypts $m' = m \oplus 1^n$ the bitwise complement of m to get a ciphertext c, which it then returns to the adversary. Security is defined in the usual way.

Some of these notions are equivalent (in the sense that if $(\mathsf{Enc}, \mathsf{Dec})$ satisfies one notion, then it must also satisfy the other, up to small losses in t, q, ϵ), and some are stronger than others (in the sense that if $(\mathsf{Enc}, \mathsf{Dec})$ satisfies notion (a), it must satisfy notion (b), but there are examples of schemes that satisfy (b) but not (a)). Your goal is to figure out the relationships between each of these security notions.

Your solution will contain several proofs of statements of the form: "if $(\mathsf{Enc}, \mathsf{Dec})$ satisfies notion (a), then it also satisfies notion (b)" (this can succinctly be stated as "notion (a) implies notion (b)"). Note that you do not necessarily need to prove all implications: if notion (a) implies notion (b) and notion (b) implies notion (c), then you can conclude without proof that notion (a) also implies notion (c).

Your solution will also contain some proofs of statements of the form: "There exist $(\mathsf{Enc}, \mathsf{Dec})$ satisfying notion (a) but that does not satisfy notion (b)" (this can be succinctly stated as notion (a) does not imply notion (b)). For these kind of statements, you may assume as a starting point a secure PRG, a secure PRF, or an encryption scheme satisfying any of the notions above (LoR, RoRP, RoRC,RoZ), which you then use to build your $(\mathsf{Enc}, \mathsf{Dec})$ counter example.

Again, note that you do not necessarily need to prove all implications. For example, if (a) does not imply (b), but (c) does imply (b), then you can conclude without proof that (a) does not imply (c).

There are a total of 20 statements to decide on (for every pair of notions (a) and (b), you must decide whether or not (a) implies (b) and whether or not (b) implies (a)). As a hint, it is possible to select 7 statements, prove those, and then derive the remaining 13 from these 7. You will not be penalized or rewarded based on the number of statements you prove; if you prove all 12 directly, that is fine (though it will be more work on your part).

2 Problem 2 (10 points)

Often when using block ciphers for encryption, messages need to be padded to a multiple of the block size. 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:** Let N be the number of bits 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. Append 10^{N-1} — that is, a 1 followed by $N - 1$ 0's — to the message.

(c) **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.

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

3 Problem 3 (25 Points)

Consider CBC mode encryption, using PKCS7 padding from Problem 2c. 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.