Exercise 1.

**Definition 1** (Learning with Errors). Let \( \ell < k \in \mathbb{N}, n < m \in \mathbb{N}, q = 2^k, B = 2^\ell \), \( A \leftarrow \mathcal{U}((\mathbb{Z}_q)^{m \times n}) \). The Learning with Errors (LWE) distribution is defined as follows: \( D_{\text{LWE}, A} = (A, A \cdot s + e \mod q) \) for \( s \leftarrow \mathcal{U}(\mathbb{Z}_q^m) \) and \( e \leftarrow \mathcal{U}([-B, B]_m \cap \mathbb{Z}_m) \).

The LWE assumption states that, given suitable parameters \( k, \ell, m, n \), it is computationally hard to distinguish \( D_{\text{LWE}, A} \) from the distribution \( (A, \mathcal{U}(\mathbb{Z}_q^m)) \).

Let us consider the private-key encryption scheme below, which works under the following public parameters: \( k, \ell, m, n, A \), for which the LWE holds.

**Keygen**\((1^\lambda)\): from \( 1^\lambda \), this algorithm outputs a random vector \( s \leftarrow \mathcal{U}(\mathbb{Z}_q^m) \) as a secret key.

**Enc**\(_s\)(\(m\)): from the secret key \( s \) and a message \( m \in \{0, 1\}^m \), the algorithm Enc samples a random vector \( e \leftarrow \mathcal{U}([-B, B]_m \cap \mathbb{Z}_m) \) and outputs \( c = As + e + q^2 m \mod q \) as a ciphertext.

**Dec**\(_s\)(\(c\)): from the secret key \( s \) and a ciphertext \( c \), the decryption algorithm computes \( v = c - A \cdot s \).

Then Dec constructs the message \( m' \) from \( v \): for each component of \( v \), sets the corresponding component of \( m' \) as follows: 1 if it is closer to \( \pm \frac{q}{2} \) than to 0, and 0 otherwise.

1. Prove the correctness of this cipher.
2. Show that this cipher is computationally secure.

If you take a look at this cipher, you can view it as a one-time pad on \( \frac{q}{2} m \), which means that the message is hidden in the most significant bit of \( e + \frac{q}{2} m \).

Now, if one wants to hide the message in the least significant bit of the OTP, one solution is to encrypt a message as: \( c = 2 \cdot (A \cdot s + e) + m \mod q \).

3. Construct a “decryption” algorithm that does not use the secret key to compute \( m \).
4. Why is it also a bad idea to encrypt as \( c = A \cdot s + 2e + m \)?

Exercise 2.

Let us recall some definitions:

- A **known-ciphertext attack** is an attack where the adversary knows some ciphertexts.
- A **known-plaintext attack** is an attack where the adversary knows some ciphertexts and their corresponding plaintexts.
- A **chosen-plaintext attack** is an attack where the adversary can choose some plaintexts and obtain the corresponding ciphertexts.
- A **chosen-ciphertext attack** is an attack where the adversary can also choose some ciphertexts and obtain the corresponding plaintexts.

The goal of the exercise is to propose distinguishers between truly random permutations and Feistel scheme permutations, for different types of attackers.
1. Describe a way to distinguish a one-round Feistel scheme implemented with a secure PRF from a random permutation (using a known-plaintext attack).

2. Given a two round Feistel scheme implemented with a secure PRF using independent keys at each round, describe a way to distinguish it from a random permutation (using a chosen-plaintext attack).

Exercise 3.
We consider a block cipher $E$ operating on blocks of $n$ bits:

\[ E : K \times M \rightarrow C \]

\[ (k, m) \mapsto E_k(m) = c \]

The ECB (Electronic Code Book) mode is recalled in Figure 2. The message is divided into blocks and each block is encrypted separately. Another mode, the CBC$^*$ mode, is described in Figure 3.
1. Show that the ECB mode is not semantically secure.

2. Give a decryption algorithm for the CBC∗ mode.

3. Show that the CBC∗ mode is not semantically secure, even if the underlying \( E \) is a secure PRF.

**Exercise 4.**

A double encryption scheme consists in encrypting twice the plaintext \( m \in \{0,1\}^n \) with two independent keys \( k_1 \in \{0,1\}^\ell \) and \( k_2 \in \{0,1\}^\ell \). We have \( c = E_{k_2}(E_{k_1}(m)) \).

1. Consider the following attacker \( A \): Assume that it knows some pairs plaintext/ciphertext \( (m,c) \), it computes \( E_k(m) \) and \( D_k(c) \) for all the keys \( k \) and memorizes all the results in a table. Analyse the complexity in memory and time of this attack and explain how the adversary can find the pair of keys used for the double encryption (compared to the exhaustive search).

This attack explains why we use the Triple-DES, which consists in a triple encryption with the DES encryption using three different keys \( (K_1, K_2, K_3) \in \{0,1\}^{56} \times \{0,1\}^{56} \times \{0,1\}^{56} \):

\[
\text{Triple-DES}_{K_1,K_2,K_3}(X) = \text{DES}_{K_3}(\text{DES}^{-1}_{K_2}(\text{DES}_{K_1}(X))).
\]

2. Can the previous attack be adapted to Triple-DES? Is it practical? Explain how to recover DES from a triple-DES implementation. Why would that be interesting?

**Exercise 5.**

The CSS (Content Scrambling System) stream cipher was used to encrypt movies in DVDs. The principle of the stream cipher is to generate a pseudo-random stream which is added, byte by byte, to the plaintext to encrypt it.

The CSS secret key is a 5-byte key. Then, to generate the pseudo-random stream, CSS uses two LFSRs:

- One of 17-bit size,
- The second one of 25-bit size.

Each LFSR generates a stream of outputs, which are grouped together in pack of 8-bits, the two packs are added modulo 256 to give one byte of the pseudo-random stream.

To create the two initial states CSS concatenates 1 with the first two bytes of the secret key (and obtain a 17-bit initial state for the first LFSR), and it concatenates 1 with the last three bytes of the secret key (and obtain a 25-bit initial state for the second LFSR).

1. Assume we have an encrypted DVD at hand, and that we know the first 20-bytes of the underlying plaintext. How do you find the first 20-bytes of the keystream?

2. Propose an attack that takes the order of \( 2^{17} \) operations to recover the secret key used.