Birzeit University
Faculty of Engineering & Technology
Computer Science Department
Encryption Theory
Assignment Two

Instructions:

- **Each student is required to develop their own approach and techniques to solve this assignment. Originality and independent thought are expected.**
- **The assignment will be individually discussed with each student to ensure understanding and verify the originality of their work. Any indication of unauthorized collaboration or copying will result in disciplinary action.**
- **The use of AI assistant tools or any other unauthorized resources is strictly prohibited for this assignment. Any student found violating this rule will receive a score of zero.**
- **Adherence to these guidelines is mandatory.**

Question One:

An encryption method known as the "one-time pad" makes use of a randomly generated key in which the length of the key is at least equal to the plaintext. Every bit or character in the plaintext is XORed with the matching bit or character in the key throughout the encryption process.

- Implement the functions one_time_pad_encr and one_time_pad_decr. These functions should accept the plaintext or ciphertext, its length, and the randomly generated key as arguments, and return the result accordingly.
- Use a pseudorandom number generator to generate the key. Provides a new random value upon each read, it is essential to generate a random secret key and store it in memory to successfully decrypt the encrypted message.
- Validate that the key and plaintext lengths match.
- Ensure your implementation works with alphanumeric characters and whitespace.
- Modify the program to handle binary data as input (e.g., files).
- Explain, in your own words, why reusing a key weakens security in the one-time pad encryption method. Demonstrate this insecurity by encrypting two different plaintext messages using the same key, and analyzing the results.

Birzeit University
Faculty of Engineering & Technology
Computer Science Department
Encryption Theory
Assignment Two

Question Two:

The Rail Fence cipher rearranges the letters in the plaintext by writing them in a zigzag pattern across a predetermined number of rails and then read them out to form the ciphertext. After each letter of a rail is read, a space is added to the ciphertext. The process is repeated for each rail.

- Implement the functions rail_fence_encr and rail_fence_decr. These functions should accept the plaintext or ciphertext, as well as the number of rails in the encryption process, and return the result accordingly.
- During encryption, plaintext is written diagonally across the rails. The ciphertext consists of the letters of the fence in each row-rail with a space added after reading each rail.
- In the decryption process, the number of rails are found from the ciphertext and it is written once again diagonally to produce the plaintext.
- It is assumed that the plaintext consists of letters, punctuation, and/or spaces, and the program should handle letters in both upper and lower cases. During encryption, spaces and punctuation should be omitted, and they should be added back to the generated plaintext after decryption.