Problem 2: Implementing a Meet-in-the-Middle Attack on a Mini Block Cipher

Task 1: Implementing Mini Block Cipher with key size 16 bit and block size 16 bit

Our implementation of the mini block cipher starts with importing a popular Python library for the get_random_bytes function to generate a cryptographically secure pseudorandom 16 bit initial key value. We then created a key expansion function that takes the two bytes from this initial value and break them into four bit 'nibbles'. To generate a new key value, we left shift one byte, and substitute its nibbles from the s-box provided in lecture, XORing with a round constant based on the polynomial $x^4 + x + 1$, and XORing with the other byte of the key value to get one half of the next round's key value. The resulting new byte is then XORed with the unmodified other byte of the current round's key value to generate the second half of the next round's key value. This is done twice.

For the encryption, we first pad the plaintext values to a length divisible by 16, break them into 16 bit blocks, and break those blocks into 4 bit nibbles. We then pass these nibbles to the encrypt_round1 function, which uses four separate functions based on 2x2 nibble tables: substituting nibbles, shifting rows, mixing columns, and XORing with the first round key value. The resulting intermediate value list of lists of nibbles is passed to encrypt_round2, which substitutes nibbles, shifts rows, and XORs with the second round key value, but does not mix columns. We then reassemble the nibbles back into strings of 1s and 0s the same length as the original plaintext binaries, and convert those strings into latin-1 encoded cipher texts.

For the decryption function, we take the cipher texts and break them into nibbles again, and then pass those nibbles into decrypt_round2. We XOR with the same second round key as the encryption function, then use an inverse shift rows function, followed by a substitution with an an inverse Sbox of the encryption function. These nibbles are passed to the decrypt round 1 function, where they are XORed with the round 1 key value, columns are mixed inverse to the matrix of the encryption function, rows are inverse shifted again, and nibbles are substituted from the inverse Sbox table. The decrypted nibbles are then concatenated back into their original lengths, converted back into latin-1 encoded characters, and padding is removed.

We provide two test cases of 10 plaintexts each, as well as the option for the user to input 10 plaintexts.