Group R

You have reached out to us to create an encryption method to transmit sensitive information while maintaining a relatively small size for encrypted text. Additionally, the text must also only use characters available on standard keyboards. We have devised a system that is both difficult for third parties to break and results in a very minimal increase in size from plain text to cipher text.

The basis of the system is matrix multiplication. The message is first transformed into a matrix, then (left) multiplied with an invertible matrix given by an encryption key. To decrypt the message, the matrix is (left) multiplied by the inverse of the matrix given by the encryption key.

We start by generating a square matrix based on the contents of the message [1]. We define the size as being the square root of the character length of the message rounded up to the nearest whole number. We extend the length of the message to match the length of the created matrix, if necessary, by appending tilde (~) characters to the end as padding. From here, we fill in the matrix right to left, top to bottom, with the ASCII value of each character in the message.

We then need to create an invertible matrix that we will use to encrypt the message matrix [2]. This matrix must be invertible so that we can undo its multiplication when decrypting the message.
We know by the Invertible Matrix theorem that a matrix is invertible if it is row equivalent to the identity matrix. This means that we can start with the identity matrix and modify it with multiple successive elementary row operations while preserving its invertibility.
In order to derive this matrix from the key, we first iterate over each row n of the identity matrix and multiply it by the nth digit in the key. We wrap around to the beginning of the key if we reach row numbers greater than the length of the key.
For further modification, we then perform a series of pivot operations, which involve adding a multiple of one row to another row. We grab pairs of digits from the key left to right, and for each pair, we generate three numbers: a starting row number, a destination row number, and a multiplier. We multiply the row referenced by the starting row number by the multiplier, and add the result to the destination row.

We then take the message matrix and multiply it with the key matrix to produce the encrypted matrix. The encrypted matrix is converted to a text format by placing a hex representation of each digit in the matrix followed by a space separator, with the entries written left to right, top to bottom [3].

We now have our ciphertext. Decryption is done by performing the above process in reverse: put the received message into a matrix, multiply by the inverse of the key matrix generated from the key, and read out the ASCII values of the resulting matrix right to left, top to bottom.

Code Appendix

[1] Generate matrix from text

```
def text_to_matrix(plain_text: str) -> np.ndarray:
    side_length = math.ceil(math.sqrt(len(plain_text)))
    text_matrix = np.zeros((side_length, side_length), dtype=np.uint32)
    # make the text have as many symbols as the array has elements
    padded_text = plain_text + ''.join(['~' for _ in range(side_length ** 2 - len(plain_text))])

    text_index = 0
    for i in range(side_length):
        for j in range(side_length):
            text_matrix[i][j] = ord(padded_text[text_index])
            text_index += 1

    return text_matrix
```

[2] Generate matrix from key

```
def key_to_matrix(key: str, n: int) -> np.ndarray:
    """
    Creates an invertible matrix from the provided string.
    """
    key_matrix = np.identity(n, dtype=np.uint32)

    new_key = ''
    for i in range(n):
        new_key += key[i % len(key)]

    for i, key_element in enumerate(new_key):
        key_matrix[i] *= ord(key_element) % 32

    for i in range(0, len(new_key)-1, 2):
        print(new_key[i:i+2])
        row1, row2, x = substring_to_numbers(new_key[i:i+2], n)
        key_matrix[row1] += (x % 32) * key_matrix[row2]

    return key_matrix


def substring_to_numbers(sub_str: str, n: int):
    """
    Substring should contain two characters
    :return: Two numbers in range(n)
    """
    num1 = ord(sub_str[0]) % n
    num2 = (5 * ord(sub_str[1])) % n
    num3 = (ord(sub_str[0]) + ord(sub_str[1])) % n
    print((num1, num2, num3))
    return num1, num2, num3
```

[3] Convert encrypted matrix to text

```
def matrix_to_text(matrix: np.ndarray) -> str:
    text = ''
    for row in matrix:
        for num in row:
            text += num_to_str(num)
    return text


def num_to_str(num: int) -> str:
    return hex(num)[2:] + ' '
```