Assignment

Transposition Cipher Implementation

Submitted By: M Zain UL Abideen Asghar (FA23-BCS-092)

Overview

Complete implementation meeting all 6 lab requirements: key handling, decode function, case preservation, space handling, random keys, and menu interface.

How It Works

The cipher rearranges text using a numeric key where each digit represents column order.
Example:

```
Key: 3214
Text: HELLO
Step 1 - Write in columns:

H(3) E(2) L(1) L(4)

O(3) X(2) X(1) X(4)  [X = padding]

Step 2 - Read columns in key order (1,2,3,4):
Column 1: LX
Column 2: EX
Column 3: HO
Column 4: LX
Result: LEXHOLX
```

Code Explanation

1. Text Splitting Function

```
def split_len(seq, length):
    return [seq[i:i + length] for i in range(0, len(seq), length)]
```

This breaks text into chunks. For "HELLO" with length 4: `["HELL", "O"]`

2. Padding Function

```
def add_padding(plaintext, key_length, padding_char='X'):
    remainder = len(plaintext) % key_length
    if remainder != 0:
        padding_needed = key_length - remainder
        plaintext += padding_char * padding_needed
    return plaintext
```

Adds 'X' characters so text length divides evenly by key length. "HELLO" (5 chars) + key length 4 = add 3 X's = "HELLOXXX"

3. Key Validation

```
def validate_key(key):
    try:
        key_digits = sorted([int(d) for d in key])
        expected = list(range(1, len(key) + 1))
        return key_digits == expected
    except ValueError:
        return False
```

Checks if key has consecutive digits 1,2,3... For "3214": sorted = [1,2,3,4], expected = [1,2,3,4]

4. Main Encode Function

```
def encode(key, plaintext, preserve_non_alpha=True):
    order = {int(val): num for num, val in enumerate(key)}
    # Creates mapping: {3:0, 2:1, 1:2, 4:3} for key "3214"
    if preserve_non_alpha:
        # Separate letters from spaces/punctuation
        alpha_chars = []
        non_alpha_positions = {}
        for i, char in enumerate(plaintext):
            if char.isalpha():
                alpha_chars.append(char)
            else:
                non_alpha_positions[len(alpha_chars)] = char
```

The function separates alphabetic characters from spaces/punctuation, encrypts only letters, then puts everything back together.

5. Decode Function

```
def decode(key, ciphertext, preserve_non_alpha=True):
    order = {int(val): num for num, val in enumerate(key)}
    reverse_order = {v: k for k, v in order.items()}
    # Reverses the encoding process
```

Creates reverse mapping to undo the encoding. If encode maps position 0→3, decode maps 3→0.

6. Random Key Generation

```

def generate_random_key(length):
    if length > 9:
        chars = list(string.ascii_lowercase[:length])
        random.shuffle(chars)
        return ''.join(str(ord(c) - ord('a') + 1) for c in chars)
    else:

```

```
        digits = list(range(1, length + 1))
        random.shuffle(digits)
        return ''.join(str(d) for d in digits)
```

For length 4: creates [1,2,3,4], shuffles to random order like [3,1,4,2], returns "3142"

7. Menu System

```

def main():
    while True:
        display_menu()
        choice = input("Enter your choice (1-4): ").strip()

        if choice == '1':
            # Encoding logic
        elif choice == '2':
            # Decoding logic
        elif choice == '3':
            # Test examples
        elif choice == '4':
            break
```

Simple loop that shows menu, gets user choice, and executes the selected function.

Usage Examples

Basic Usage

```
# Encode with key
encoded = encode('3214', 'HELLO', False)
print(encoded)  # Output: LXEXHOLX
# Decode back
decoded = decode('3214', encoded, False)
print(decoded)  # Output: HELLO
```

With Formatting

```
# Preserve spaces and punctuation
message = "Hello, World!"
encoded = encode('3214', message, True)
decoded = decode('3214', encoded, True)
print(decoded)  # Output: Hello, World!
```

Random Key

```
# Generate random key
key = generate_random_key(4)
print(key)  # Output: random like "2413"
```