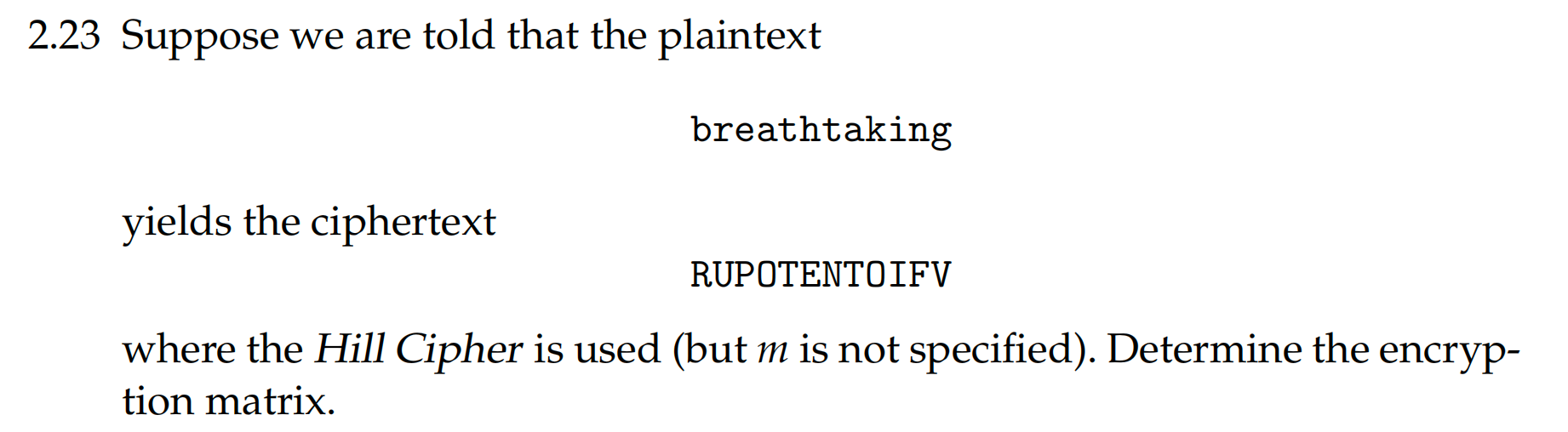


**Answer:**

|  |
| --- |
| from itertools import product  # 定义 LFSR 函数  def lfsr(initial\_state):  state = initial\_state[:] # 复制初始状态，避免修改原始输入  seen\_states = {tuple(state)} # 使用集合存储已见状态  count = 0  while True:  # 应用线性递归关系  new\_bit = (state[0] + state[1] + state[2] + state[3]) % 2  state = state[1:] + [new\_bit] # 移位并添加新状态  count += 1  # 如果新状态已经见过，结束循环  if tuple(state) in seen\_states:  break  seen\_states.add(tuple(state))  return count  # 遍历所有可能的初始向量（IVs），因为 Z2 的范围是 [0, 1]  ivs = list(product(range(2), repeat=4))  # 计算每个 IV 的周期  periods = {iv: lfsr(list(iv)) for iv in ivs}  # 打印周期  for iv, period in periods.items():  print(f"IV: {iv}, Period: {period}") |

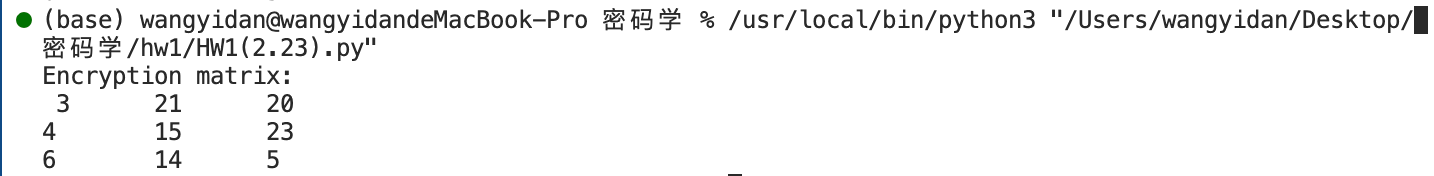


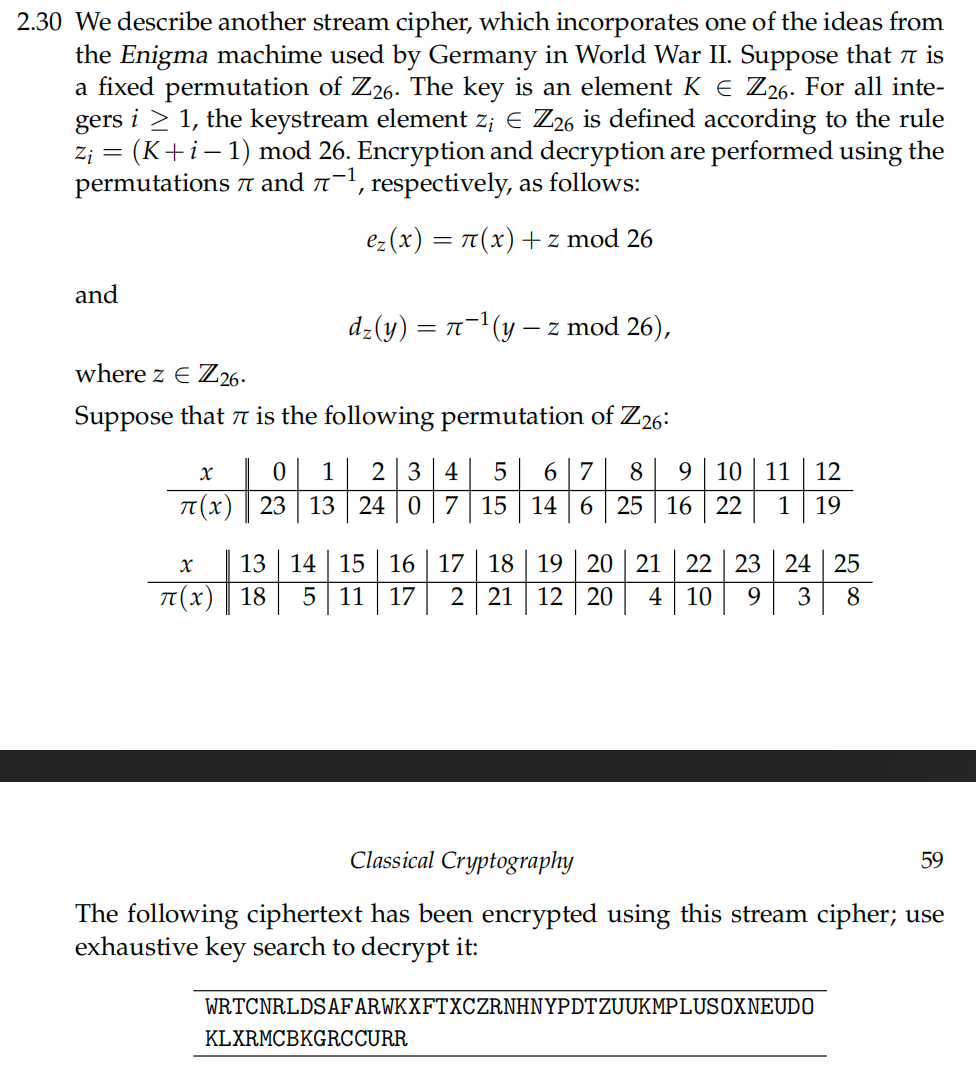


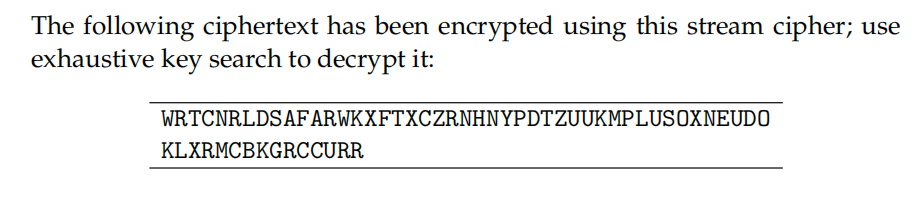
**Answer:**

|  |
| --- |
| from sympy import Matrix, mod\_inverse  # 将字母转换为数字  def letters\_to\_numbers(letters):  return [ord(letter) - ord('A') for letter in letters.upper()]  # 将文本转换为数字，假设它已经是大写且没有空格或非字母字符  def text\_to\_numeric(text):  return [letters\_to\_numbers(text[i:i+3]) for i in range(0, len(text), 3)]  # 使用明文和密文求解加密矩阵  def solve\_hill\_cipher(plaintext, ciphertext):  # 将明文和密文分割成大小为3的块，并转换为数字  plaintext\_blocks = text\_to\_numeric(plaintext)  ciphertext\_blocks = text\_to\_numeric(ciphertext)  # 使用第一个块来确定加密矩阵  P = Matrix(plaintext\_blocks[0:3])  C = Matrix(ciphertext\_blocks[0:3])  # 求加密矩阵A，使P \* A = C mod 26  # P求模26的倒数解A  try:  P\_inv = P.inv\_mod(26)  except ValueError as e:  return str(e), None # If the inverse doesn't exist, return the error message  A = P\_inv \* C % 26  return None, A  # 给定明文和密文  plaintext = "breathtaking"  ciphertext = "RUPOTENTOIFV"  # 假设n = 3，求解加密矩阵  error, encryption\_matrix = solve\_hill\_cipher(plaintext, ciphertext)  # 检查矩阵是否找到并打印出来  if encryption\_matrix:  # Format and print the matrix  matrix\_as\_list = encryption\_matrix.tolist()  formatted\_matrix = '\n'.join(['\t'.join(map(str, row)) for row in matrix\_as\_list])  print("Encryption matrix:\n", formatted\_matrix)  else:  # If there was an error, print it  print("Error:", error) |

因为明文有12个字符所以加密矩阵的类型可以是2\*2,3\*3,4\*4和6\*6在代码中我尝试使用3\*3的加密矩阵并输出了结果。







**Answer:**

|  |
| --- |
| def decrypt(ciphertext, K, pi):  # 反置换字典  pi\_inverse = {v: k for k, v in pi.items()}  # 解密函数  def dz(y, z):  return pi\_inverse[(y - z) % 26]  plaintext = ''  for i, c in enumerate(ciphertext):  z = (K + i) % 26  y = ord(c) - ord('A')  x = dz(y, z)  plaintext += chr(x + ord('A'))  return plaintext  # 置换π  pi = {0: 23, 1: 13, 2: 24, 3: 0, 4: 7, 5: 15, 6: 14, 7: 6, 8: 25, 9: 16, 10: 22, 11: 1, 12: 19,  13: 18, 14: 5, 15: 11, 16: 17, 17: 2, 18: 21, 19: 12, 20: 20, 21: 4, 22: 10, 23: 9, 24: 3, 25: 8}  ciphertext = "WRTCNRLDSAFARWKXFTXCZRNHNYPDTZUUKMPLUSOXNEUDOKLXRMCBKGRCCURR"  # 尝试每个可能的密钥K  for K in range(26):  plaintext = decrypt(ciphertext, K, pi)  print(f"K={K}: {plaintext}") |

