**🔐 1. Caesar Cipher**

def caesar\_encrypt(text, key):

result = ""

for char in text:

if char.isalpha():

shifted = (ord(char.lower()) - 97 + key) % 26

result += chr(shifted + 97)

else:

result += char

return result

def caesar\_decrypt(cipher, key):

return caesar\_encrypt(cipher, -key)

# Example

text = "hello"

key = 3

cipher = caesar\_encrypt(text, key)

original = caesar\_decrypt(cipher, key)

print("Encrypted:", cipher)

print("Decrypted:", original)

**🔠 2. Monoalphabetic Cipher**

def mono\_encrypt(text, key\_map):

return ''.join(key\_map.get(c, c) for c in text.lower())

def mono\_decrypt(cipher, key\_map):

reverse\_map = {v: k for k, v in key\_map.items()}

return ''.join(reverse\_map.get(c, c) for c in cipher.lower())

# Example

import string

import random

plain\_alphabet = string.ascii\_lowercase

shuffled = list(plain\_alphabet)

random.shuffle(shuffled)

key\_map = dict(zip(plain\_alphabet, shuffled))

text = "hello"

cipher = mono\_encrypt(text, key\_map)

original = mono\_decrypt(cipher, key\_map)

print("Encrypted:", cipher)

print("Decrypted:", original)

**🔲 3. Playfair Cipher**

def create\_matrix(key):

key = key.lower().replace('j', 'i')

seen = ""

for char in key:

if char not in seen and char.isalpha():

seen += char

for char in "abcdefghiklmnopqrstuvwxyz":

if char not in seen:

seen += char

matrix = [list(seen[i:i+5]) for i in range(0, 25, 5)]

return matrix

def find\_position(matrix, char):

for i in range(5):

for j in range(5):

if matrix[i][j] == char:

return i, j

def prepare\_text(text):

text = text.lower().replace('j', 'i')

i = 0

result = ""

while i < len(text):

a = text[i]

b = text[i+1] if i+1 < len(text) else 'x'

if a == b:

result += a + 'x'

i += 1

else:

result += a + b

i += 2

if len(result) % 2 != 0:

result += 'x'

return result

def playfair(text, key, mode='encrypt'):

matrix = create\_matrix(key)

text = prepare\_text(text)

result = ""

shift = 1 if mode == 'encrypt' else -1

for i in range(0, len(text), 2):

a, b = text[i], text[i+1]

r1, c1 = find\_position(matrix, a)

r2, c2 = find\_position(matrix, b)

if r1 == r2:

result += matrix[r1][(c1 + shift) % 5]

result += matrix[r2][(c2 + shift) % 5]

elif c1 == c2:

result += matrix[(r1 + shift) % 5][c1]

result += matrix[(r2 + shift) % 5][c2]

else:

result += matrix[r1][c2]

result += matrix[r2][c1]

return result

# Example

text = "hello"

key = "monarchy"

cipher = playfair(text, key, 'encrypt')

original = playfair(cipher, key, 'decrypt')

print("Encrypted:", cipher)

print("Decrypted:", original)

**🔤 4. Polyalphabetic Cipher (Vigenère)**

def vigenere\_encrypt(text, key):

key = key.lower()

result = ""

for i, char in enumerate(text.lower()):

if char.isalpha():

shift = (ord(char) - 97 + ord(key[i % len(key)]) - 97) % 26

result += chr(shift + 97)

else:

result += char

return result

def vigenere\_decrypt(cipher, key):

key = key.lower()

result = ""

for i, char in enumerate(cipher.lower()):

if char.isalpha():

shift = (ord(char) - ord(key[i % len(key)])) % 26

result += chr((shift + 26) % 26 + 97)

else:

result += char

return result

# Example

text = "hello"

key = "key"

cipher = vigenere\_encrypt(text, key)

original = vigenere\_decrypt(cipher, key)

print("Encrypted:", cipher)

print("Decrypted:", original)

**🧮 5. Hill Cipher (2x2 Matrix)**

def hill\_encrypt(text, key\_matrix):

text = text.lower().replace(" ", "")

while len(text) % 2 != 0:

text += 'x'

result = ""

for i in range(0, len(text), 2):

vector = [ord(text[i]) - 97, ord(text[i+1]) - 97]

encrypted = [

(key\_matrix[0][0]\*vector[0] + key\_matrix[0][1]\*vector[1]) % 26,

(key\_matrix[1][0]\*vector[0] + key\_matrix[1][1]\*vector[1]) % 26

]

result += chr(encrypted[0] + 97) + chr(encrypted[1] + 97)

return result

def hill\_decrypt(cipher, key\_matrix):

a, b, c, d = key\_matrix[0][0], key\_matrix[0][1], key\_matrix[1][0], key\_matrix[1][1]

det = (a \* d - b \* c) % 26

det\_inv = pow(det, -1, 26)

inv\_matrix = [

[(d \* det\_inv) % 26, (-b \* det\_inv) % 26],

[(-c \* det\_inv) % 26, (a \* det\_inv) % 26]

]

return hill\_encrypt(cipher, inv\_matrix)

# Example

text = "help"

key\_matrix = [[3, 3], [2, 5]]

cipher = hill\_encrypt(text, key\_matrix)

original = hill\_decrypt(cipher, key\_matrix)

print("Encrypted:", cipher)

print("Decrypted:", original)

**🔁 6. Rail Fence and Columnar Transposition**

# Rail Fence Cipher

def rail\_fence\_encrypt(text, rails):

fence = ['' for \_ in range(rails)]

rail = 0

var = 1

for char in text:

fence[rail] += char

rail += var

if rail == rails-1 or rail == 0:

var \*= -1

return ''.join(fence)

def rail\_fence\_decrypt(cipher, rails):

pattern = ['' for \_ in range(len(cipher))]

rail = 0

var = 1

for i in range(len(cipher)):

pattern[i] = rail

rail += var

if rail == rails-1 or rail == 0:

var \*= -1

sorted\_pos = sorted(range(len(cipher)), key=lambda x: pattern[x])

fence = ['' for \_ in range(len(cipher))]

for i, pos in enumerate(sorted\_pos):

fence[pos] = cipher[i]

return ''.join(fence)

# Columnar Transposition Cipher

def columnar\_encrypt(text, key):

n = len(key)

cols = ['' for \_ in range(n)]

for i, char in enumerate(text):

cols[i % n] += char

order = sorted(range(n), key=lambda x: key[x])

return ''.join([cols[i] for i in order])

def columnar\_decrypt(cipher, key):

n = len(key)

order = sorted(range(n), key=lambda x: key[x])

col\_len = len(cipher) // n

cols = ['' for \_ in range(n)]

i = 0

for index in order:

cols[index] = cipher[i:i+col\_len]

i += col\_len

result = ''

for i in range(col\_len):

for col in cols:

result += col[i]

return result

# Example

text = "attackatdawn"

rf\_cipher = rail\_fence\_encrypt(text, 3)

rf\_plain = rail\_fence\_decrypt(rf\_cipher, 3)

col\_cipher = columnar\_encrypt(text, "3214")

col\_plain = columnar\_decrypt(col\_cipher, "3214")

print("Rail Encrypted:", rf\_cipher)

print("Rail Decrypted:", rf\_plain)

print("Columnar Encrypted:", col\_cipher)

print("Columnar Decrypted:", col\_plain)

**🔐 7. Simplified DES (S-DES)**

def simplified\_des\_demo():

print("S-DES requires bit-level permutations, S-boxes, and XOR operations.")

print("Use CrypTool or logic gates simulation for full demo.")

simplified\_des\_demo()

**🔑 8. Diffie-Hellman Key Exchange**

def diffie\_hellman(p, g, a, b):

A = pow(g, a, p)

B = pow(g, b, p)

shared\_key1 = pow(B, a, p)

shared\_key2 = pow(A, b, p)

return shared\_key1, shared\_key2

# Example

p = 23

g = 5

a = 6

b = 15

key1, key2 = diffie\_hellman(p, g, a, b)

print("Shared Secret A:", key1)

print("Shared Secret B:", key2)

**🔐 9. RSA Algorithm**

def rsa\_encrypt\_decrypt(p, q, e, message):

n = p \* q

phi = (p - 1) \* (q - 1)

d = pow(e, -1, phi)

encrypted = pow(message, e, n)

decrypted = pow(encrypted, d, n)

return encrypted, decrypted

# Example

p = 61

q = 53

e = 17

msg = 65

cipher, plain = rsa\_encrypt\_decrypt(p, q, e, msg)

print("Encrypted:", cipher)

print("Decrypted:", plain)