

Network Security and Forensics

Lab Session 5

Submitted To:- Submitted By:-

Dr. Lokesh Chauhan Sir Saloni Rangari

M.Tech. AIDS

1. **Write a program to demonstrate the Vegenere Cipher  
   Input from file and Output into the file**

def vigenere\_encrypt(plaintext,keyword):  
 encrypted\_text = []  
 keyword\_repeated = (keyword \* (len(plaintext) // len(keyword) + 1))[:len(plaintext)]  
  
 for p,k in zip(plaintext,keyword\_repeated):  
 if p.isalpha(): # Check if the character is an alphabet  
 shift = ord(k.lower()) - ord('a')  
 base = ord('A') if p.isupper() else ord('a')  
 encrypted\_char = chr((ord(p) - base + shift) % 26 + base)  
 encrypted\_text.append(encrypted\_char)  
 else:  
 encrypted\_text.append(p) # Non-alphabet characters are not changed  
  
 return ''.join(encrypted\_text)  
  
  
def vigenere\_decrypt(ciphertext,keyword):  
 decrypted\_text = []  
 keyword\_repeated = (keyword \* (len(ciphertext) // len(keyword) + 1))[:len(ciphertext)]  
  
 for c,k in zip(ciphertext,keyword\_repeated):  
 if c.isalpha(): # Check if the character is an alphabet  
 shift = ord(k.lower()) - ord('a')  
 base = ord('A') if c.isupper() else ord('a')  
 decrypted\_char = chr((ord(c) - base - shift) % 26 + base)  
 decrypted\_text.append(decrypted\_char)  
 else:  
 decrypted\_text.append(c) # Non-alphabet characters are not changed  
  
 return ''.join(decrypted\_text)  
  
  
# Usage  
if \_\_name\_\_ == "\_\_main\_\_":  
  
 operation = int(input("Type 1 to encrypt or 2 to decrypt:"))  
  
 if operation == 1:  
  
 # Read plaintext and keyword from files  
 with open('e\_plain\_text.txt','r') as plaintext\_file:  
 plaintext = plaintext\_file.read()  
  
 with open('e\_key.txt','r') as keyword\_file:  
 key = keyword\_file.read().strip()  
  
 print("Plain Text: ",plaintext) # Print the plaintext  
 print("Key: ",key) # Print the key  
  
 # Encrypt the plaintext  
 encrypted = (vigenere\_encrypt(plaintext,key))  
  
 # Save the encrypted text to a file  
 with open('e\_cipher\_text.txt','w') as encrypted\_file:  
 encrypted\_file.write(encrypted)  
  
 # Decrypt the text back to verify  
 decrypted = vigenere\_decrypt(encrypted,key)  
  
 print(f"Encrypted: {encrypted}")  
 print(f"Decrypted: {decrypted}")  
  
 elif operation == 2:  
  
 # Read plaintext and keyword from files  
 with open('d\_cipher\_text.txt','r') as ciphertext\_file:  
 ciphertext = ciphertext\_file.read()  
  
 with open('d\_key.txt','r') as keyword\_file:  
 key = keyword\_file.read().strip()  
  
 print("Cipher Text: ",ciphertext) # Print the plaintext  
 print("Key: ",key) # Print the key  
  
 # Encrypt the plaintext  
 decrypted = vigenere\_decrypt(ciphertext,key)  
  
 # Save the encrypted text to a file  
 with open('d\_plain\_text.txt','w') as decrypted\_file:  
 decrypted\_file.write(decrypted)  
  
 # Decrypt the text back to verify  
 encrypted = vigenere\_encrypt(decrypted,key)  
  
 print(f"Encrypted: {encrypted}")  
 print(f"Decrypted: {decrypted}")

**Output:** Encryption

**A screenshot of a computer program

Description automatically generated**

**Output:** Decryption

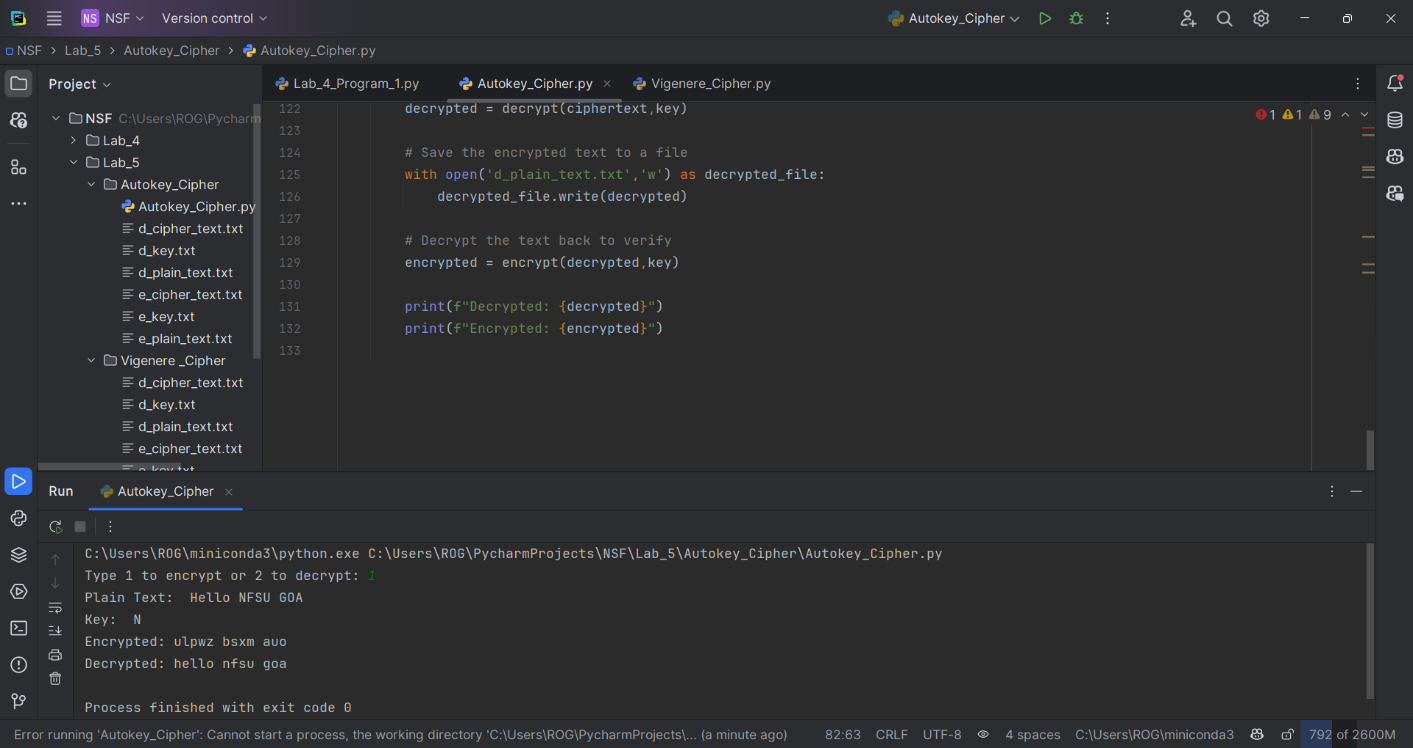
A screenshot of a computer program

Description automatically generated

1. **Write a Program to demonstrate the AutoKey Cipher.  
   Input from file and Output into the file**

from importlib.metadata import pass\_none  
  
  
def encrypt(plaintext: str, key: str) -> str:  
  
 if not isinstance(plaintext, str):  
 raise TypeError("plaintext must be a string")  
 if not isinstance(key, str):  
 raise TypeError("key must be a string")  
 if not plaintext:  
 raise ValueError("plaintext is empty")  
 if not key:  
 raise ValueError("key is empty")  
  
 key += plaintext  
 plaintext = plaintext.lower()  
 key = key.lower()  
 plaintext\_iterator = 0  
 key\_iterator = 0  
 ciphertext = ""  
 while plaintext\_iterator < len(plaintext):  
 if (ord(plaintext[plaintext\_iterator]) < 97  
 or ord(plaintext[plaintext\_iterator]) > 122):  
 ciphertext += plaintext[plaintext\_iterator]  
 plaintext\_iterator += 1  
 elif ord(key[key\_iterator]) < 97 or ord(key[key\_iterator]) > 122:  
 key\_iterator += 1  
 else:  
 ciphertext += chr(((ord(plaintext[plaintext\_iterator]) - 97 + ord(key[key\_iterator]))- 97) % 26 + 97)  
 key\_iterator += 1  
 plaintext\_iterator += 1  
 return ciphertext  
  
  
def decrypt(ciphertext: str, key: str) -> str:  
 if not isinstance(ciphertext, str):  
 raise TypeError("ciphertext must be a string")  
 if not isinstance(key, str):  
 raise TypeError("key must be a string")  
 if not ciphertext:  
 raise ValueError("ciphertext is empty")  
 if not key:  
 raise ValueError("key is empty")  
  
 key = key.lower()  
 ciphertext\_iterator = 0  
 key\_iterator = 0  
 plaintext = ""  
 while ciphertext\_iterator < len(ciphertext):  
 if (  
 ord(ciphertext[ciphertext\_iterator]) < 97  
 or ord(ciphertext[ciphertext\_iterator]) > 122):  
 plaintext += ciphertext[ciphertext\_iterator]  
 else:  
 plaintext += chr(  
 (ord(ciphertext[ciphertext\_iterator]) - ord(key[key\_iterator])) % 26 + 97)  
 key += chr(  
 (ord(ciphertext[ciphertext\_iterator]) - ord(key[key\_iterator])) % 26 + 97)  
 key\_iterator += 1  
 ciphertext\_iterator += 1  
 return plaintext  
  
# Usage  
if \_\_name\_\_ == "\_\_main\_\_":  
  
 operation = int(input("Type 1 to encrypt or 2 to decrypt:"))  
  
 if operation == 1:  
  
 # Read plaintext and keyword from files  
 with open('e\_plain\_text.txt','r') as plaintext\_file:  
 plaintext = plaintext\_file.read()  
  
 with open('e\_key.txt','r') as keyword\_file:  
 key = keyword\_file.read().strip()  
  
 print("Plain Text: ",plaintext) # Print the plaintext  
 print("Key: ",key) # Print the key  
  
 # Encrypt the plaintext  
 encrypted = (encrypt(plaintext,key))  
  
 # Save the encrypted text to a file  
 with open('e\_cipher\_text.txt','w') as encrypted\_file:  
 encrypted\_file.write(encrypted)  
  
 # Decrypt the text back to verify  
 decrypted = decrypt(encrypted,key)  
  
 print(f"Encrypted: {encrypted}")  
 print(f"Decrypted: {decrypted}")  
  
 elif operation == 2:  
  
 # Read plaintext and keyword from files  
 with open('d\_cipher\_text.txt','r') as ciphertext\_file:  
 ciphertext = ciphertext\_file.read()  
  
 with open('d\_key.txt','r') as keyword\_file:  
 key = keyword\_file.read().strip()  
  
 print("Cipher Text: ",ciphertext) # Print the ciphertext  
 print("Key: ",key) # Print the key  
  
 # Encrypt the plaintext  
 decrypted = decrypt(ciphertext,key)  
  
 # Save the encrypted text to a file  
 with open('d\_plain\_text.txt','w') as decrypted\_file:  
 decrypted\_file.write(decrypted)  
  
 # Decrypt the text back to verify  
 encrypted = encrypt(decrypted,key)  
  
 print(f"Decrypted: {decrypted}")  
 print(f"Encrypted: {encrypted}")

**Output:** Encryption



**Output:** Decryption

