CS23334 -Fundamental Of Data Science

240701463

Data Encryption and Decryption

Exp :1D

Date: 29-07-2025

Aim:

To **encrypt and decrypt sensitive data** using the **Fernet** symmetric encryption scheme from the cryptography library.

Algorithm:

1. Generate a **Fernet key** for encryption and decryption.
2. Define the **sensitive data** (plaintext) to be encrypted.
3. Initialize the Fernet cipher object using the generated key.
4. **Encrypt** the plaintext data using the cipher object.
5. **Decrypt** the resulting ciphertext back into the original plaintext.
6. Print the key, ciphertext, and decrypted plaintext to verify the process.

Code:

```
from cryptography.fernet import Fernet

key = Fernet.generate_key()

# Initialize the Fernet cipher object

f = Fernet(key)

# 2. Define the sensitive data (must be a byte string)

sensitive_data = b"This is the secret message for Experiment 1.d."

print(f"Original Data (Plaintext): {sensitive_data.decode()}")

print(f"Encryption Key: {key.decode()}")

# 3. Encrypt the data

ciphertext = f.encrypt(sensitive_data)
```

```
# 4. Decrypt the data
decrypted_data = f.decrypt(ciphertext)
print("-" * 50)
print(f"Encrypted Data (Ciphertext): {ciphertext.decode()}")
print(f"Decrypted Data (Plaintext): {decrypted_data.decode()}")
print("-" * 50)
```

Output:

When the Python code above is executed, the output will resemble:

Original Data (Plaintext): This is the secret message for Experiment 1.d.

Encryption Key: [A unique base64-encoded key string will appear here]

Encrypted Data (Ciphertext): [A long, garbled ciphertext string will appear here]

Decrypted Data (Plaintext): This is the secret message for Experiment 1.d.

Result:

The experiment successfully used the **Fernet key** to convert the **plaintext** sensitive data into an **unreadable ciphertext**. The same key was then used to **decrypt** the ciphertext, perfectly restoring the original sensitive message:

- **Encryption** converts sensitive_data to ciphertext.
- **Decryption** converts ciphertext back to decrypted_data, which matches sensitive_data.

This demonstrates the core functionality of **symmetric cryptography** where the security of the data relies entirely on the secrecy of the shared key. Thus the python program was executed successfully, and the output is verified.