Data Privacy and Security and 6CS1022

REPORT

On

Face Lock

Submitted in Partial Fulfillment of Award of

BACHELOR OF TECHNOLOGY
In

Computer Science and Engineering

By

P. Rohini-2022BCSE07AED733

P. Bharath Naik-2022BCSEAED745

J. Anusha-2022BCSE07AED796

Under the Supervision of

<Dr. K. Sathesh Kumar>

<Associate professor>

ALLIANCE SCHOOL OF ADVANCED COMPUTING

ALLIANCE UNIVERSITY

BENGALURU

APRIL 2025

Problem Statement:

The general recommendation is for people to manage and control sensitive files through basic alphanumeric password systems that are brute-force vulnerable, as well as phishing and social engineering attacks. Passwords becoming harder to remember, forgotten changing, along with reuse leads to reduced security and reliability of such systems.

Alongside these fatigue concerns, the growing issue of privacy violations, and digital data theft raises the demand for an innovative, reliable, and simply easier solution tailored to ease the burden of users. The task creates a paradox requiring to solve completely providing effective file security, minimal user workload and high accessibility. So, the goal of this project is to create an intuitive, hands-free lock file system based on facial biometrics. The system leverages the user's face as the key, enabling effortless and highly secure access.

Abstract:

In this project, we introduce the system FaceLock – A File Locker System Using Facial Recognition, which aims to secure data by employing biometric methods of

authentication. Passwords are no longer relevant since files can be unlocked and locked by facial recognition based on identity verification.

The solution is powered by remarkable Python libraries such as OpenCV which handles image processing, Deep Face which takes care of deep learning-based face verification, and the Cryptography library which handles encryption and decryption of files securely. The project takes a picture from the webcam in real-time and compares it with a stored reference image to decide whether the user should be granted access.
The system automatically unlocks files by facial recognition and if the recognition is successful the files are decrypted and accessible. Access is denied and no action is taken on the file if the recognition fails. Users can start the application manually; however, the process is fully hands-free and automated during file access. The system functions on the predetermined illumination making it easy to use.

Introduction:

In the last few decades, biometric authentication has proven to be an effective and convenient means of securing information. Out of all biometric methods on offer and widely in use today, such as fingerprints, iris and voice biometrics, facial recognition is the most appealing as it requires no manual intervention and can be done at a distance.

The rationale behind the FaceLock project derives from the need to have smart security systems that do not need too much effort from the user. With this system, file access is restricted until a user's identity is authenticated with facial recognition. Security is ensured by two powerful technologies: DeepFace, which is a face recognition framework based on deep learning, and Fernet encryption from the Python Cryptography package.

FaceLock is fully automated; it does not require any interaction in the form of password entry or button clicking. It accesses a live webcam stream, takes a snapshot, and compares it with a stored facial image. If they match, the file unlocks; if they don't, the file locks. This project serves to demonstrate the possibilities of Artificial Intelligence in advancing digital security while also making personal data handling effortless and sophisticated.

Project Implementation:

a. Write up(Proposed system module detail):

1. Webcam Capture Module
   - activates the webcam, taking a real-time image snapshot of the user.
   - Saves the image temporarily until it is processed.

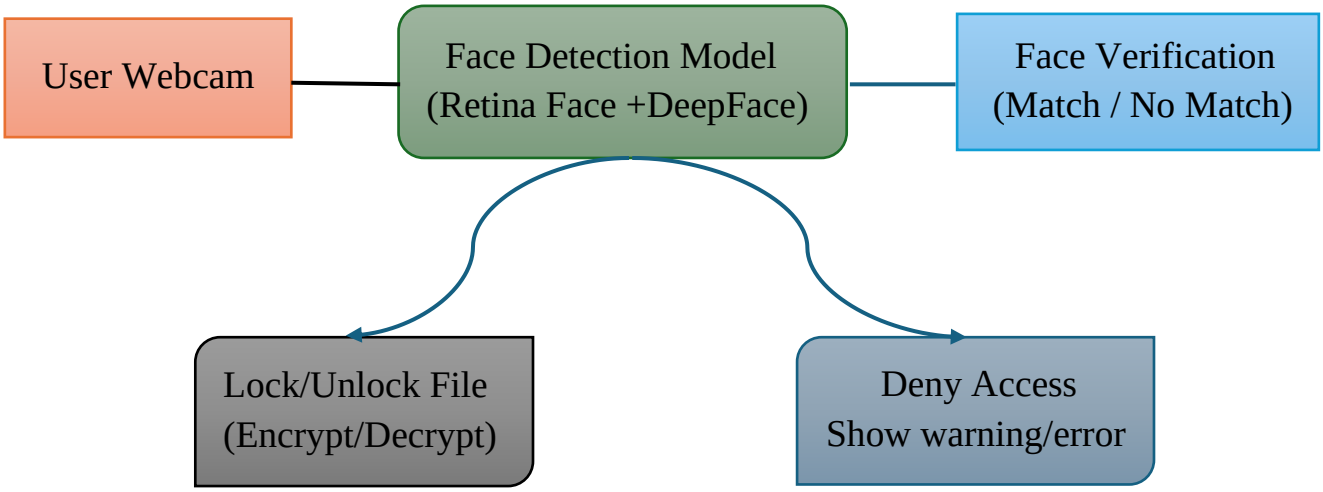
2. Face Detection and Recognition Module
   - Acts as a face detection and recognition system utilising DeepFace with RetinaFace backend for feature detection and comparison.
   - Verification is done by comparing the image to a reference image to validate identity.

3. Authentication Decision Module
   - With the outcome of the face recognition model, a decision is made whether to permit or deny access.

4. File Encryption / Decryption Module
   - Upon gaining access, it verifies whether the file is encrypted.
   - If yes → unlocks and decrypts the file.
   - If no → locks and encrypts the file.
   - File encryption uses symmetric encryption with Fernet

5. Security Measures and auto-delete feature
   - Removes pictures classified as temporary to ensure privacy.
   - Terminal notifications are set to indicate success or failure to the user.

b. Proposed system architecture:



System start, Webcam instantiation is achieved automatically to take a live picture. This picture is provided to the DeepFace model for identity verification. If the model confirms the match, it will proceed to decrypt (unlock) or encrypt (lock) a predetermined file with Fernet cryptography. If face does not match, the system locks face access and logs the attempt securely.

The architecture described here is optimal in usability and performance while enhancing security with the integration of biometric AI and cryptographic sentinels with deep defenses.

c. Proposed Algorithm:

1. Activate the application and set up the webcam.
2. Take a real time photograph of the participant.
3. Retrieve the existing reference photograph.
4. Run DeepFace with RetinaFace backend to check the images.
5. If face verification is successful:
    - If the photograph is encrypted, then decrypt it.
    - If the photograph is already decrypted, then encrypt it.
6. If face verification is not successful:
    - Deny permission and terminate the program.

7. Issue the corresponding notification.
8. Erase temporary data.

Tech Stack:

Component	Tool Library Used
Programming Language	Python
Face Detection	OpenCV, RetinaFace
Face Recognition	DeepFace
File Encryption	Python Cryptography (Fernet)
IDE	Visual Studio Code
OS	Windows 10 or later
Camera	Any standard webcam

Code Snaps:

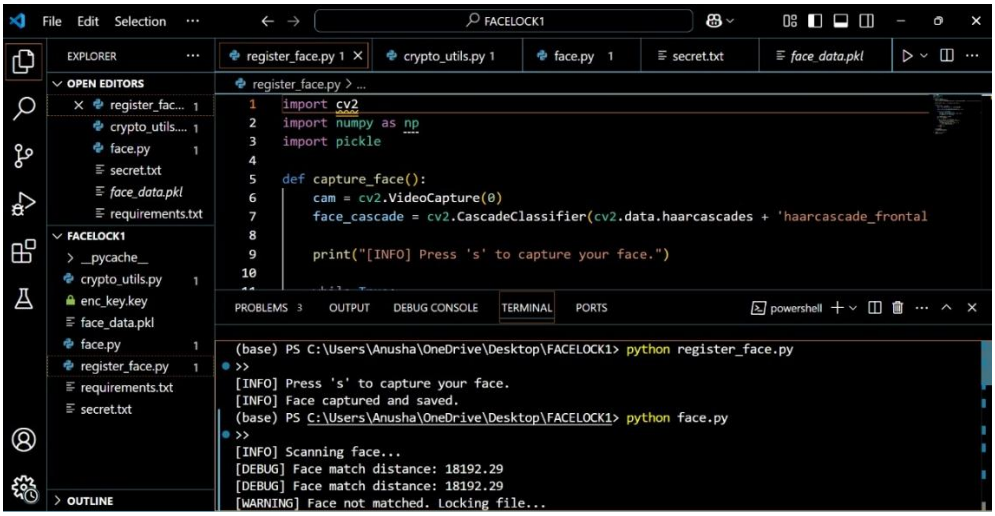
```
import cv2
import numpy as np
import pickle
def capture_face():
    cam = cv2.VideoCapture(0)
    face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades +
'haarcascade_frontalface_default.xml')
    print("[INFO] Press 's' to capture your face.")
    while True:
        ret, frame = cam.read()
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

        faces = face_cascade.detectMultiScale(gray, 1.3, 5)

        for (x, y, w, h) in faces:
            face_img = gray[y:y+h, x:x+w]
```

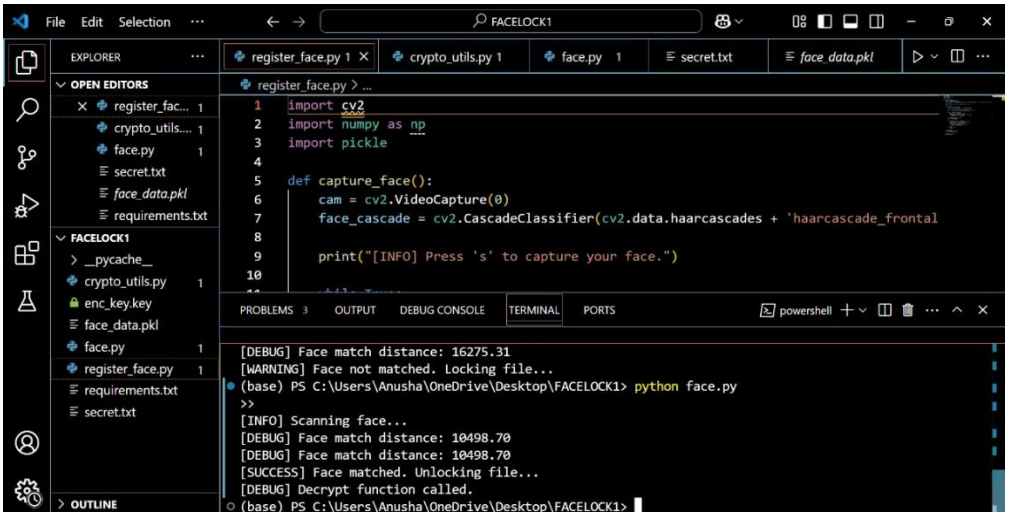
```
            face_img = cv2.resize(face_img, (100, 100))
            cv2.imshow("Face", face_img)
        cv2.imshow("Camera", frame)
        key = cv2.waitKey(1)
        if key == ord('s') and len(faces) > 0:
            print("[INFO] Face captured and saved.")
            encoding = face_img.flatten()
            with open("face_data.pkl", "wb") as f:
                pickle.dump(encoding, f)
            break
        elif key == 27:
            break
    cam.release()
    cv2.destroyAllWindows()
if _name_ == "_main_":
    capture_face()
import cv2
import numpy as np
import pickle
from crypto_utils import encrypt_file, decrypt_file, write_key
import os
def load_registered_face():
    with open("face_data.pkl", "rb") as f:
        return pickle.load(f)
def capture_current_face():
    cam = cv2.VideoCapture(0)
    face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades +
'haarcascade_frontalface_default.xml')
    print("[INFO] Scanning face...")
    while True:
        ret, frame = cam.read()
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        faces = face_cascade.detectMultiScale(gray, 1.3, 5)
        for (x, y, w, h) in faces:
            face_img = gray[y:y+h, x:x+w]
            face_img = cv2.resize(face_img, (100, 100))
```

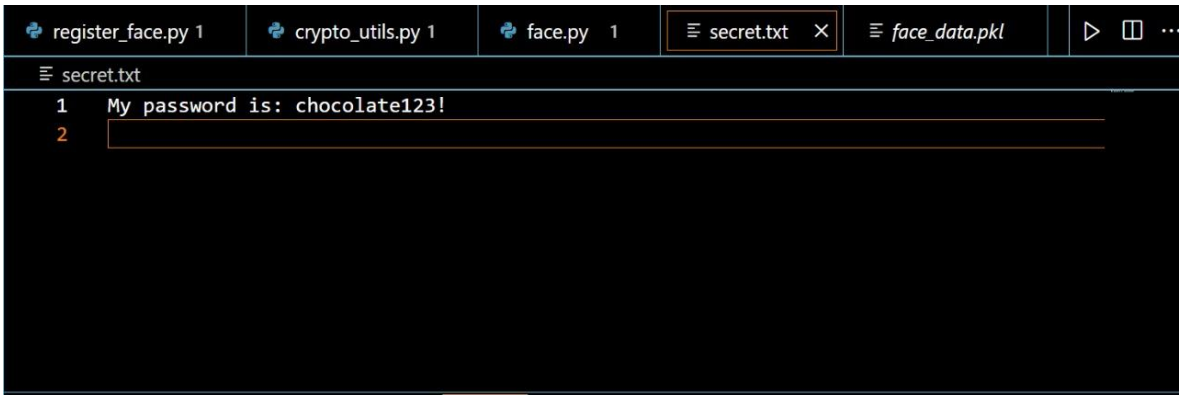
```
            encoding = face_img.flatten()
            cam.release()
            cv2.destroyAllWindows()
            return encoding
        cv2.imshow("Camera", frame)
        if cv2.waitKey(1) == 27:
            break
    cam.release()
    cv2.destroyAllWindows()
    return None
def is_face_match(stored, current, threshold=13000):
    distance = np.linalg.norm(stored - current)
    print(f"[DEBUG] Face match distance: {distance:.2f}")
    print(f"[DEBUG] Face match distance: {distance:.2f}")
    return distance < threshold
def main():
    if not os.path.exists("enc_key.key"):
        write_key()
        print("[INFO] Encryption key created.")
    stored_encoding = load_registered_face()
    current_encoding = capture_current_face()
    if current_encoding is not None and is_face_match(stored_encoding,
current_encoding):
        print("[SUCCESS] Face matched. Unlocking file...")
        decrypt_file("secret.txt")
    else:
        print("[WARNING] Face not matched. Locking file...")
        encrypt_file("secret.txt")
if _name_ == "_main_":
    main()
opencv-python
cryptography
numpy
from cryptography.fernet import Fernet
def write_key():
    key = Fernet.generate_key()
```

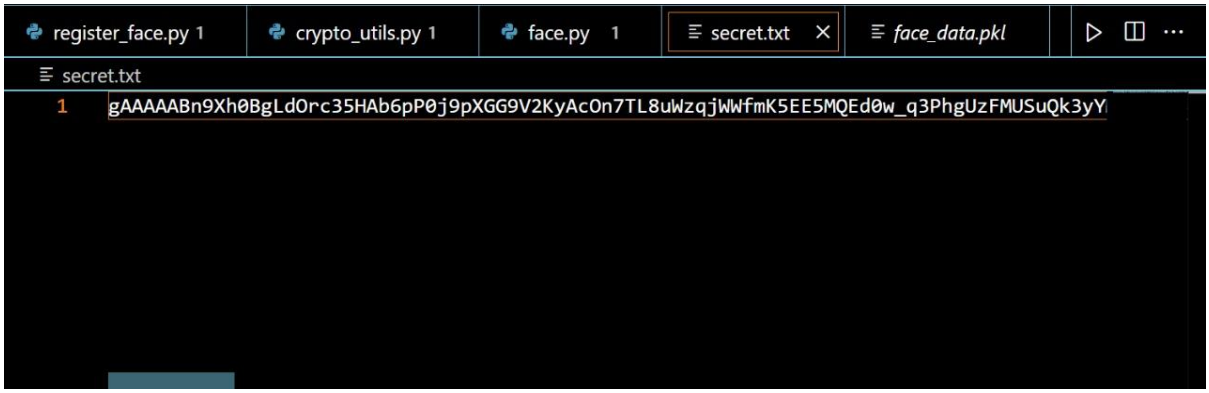
```
    with open("enc_key.key", "wb") as key_file:
        key_file.write(key)
def load_key():
    with open("enc_key.key", "rb") as key_file:
        return key_file.read()
def encrypt_file(filename):
    key = load_key()
    f = Fernet(key)
    with open(filename, "rb") as file:
        data = file.read()
    encrypted = f.encrypt(data)
    with open(filename, "wb") as file:
        file.write(encrypted)
def decrypt_file(filename):
    key = load_key()
    f = Fernet(key)
    print("[DEBUG] Decrypt function called.")
    with open(filename, "rb") as file:
        data = file.read()
    decrypted = f.decrypt(data)
    with open(filename, "wb") as file:
        file.write(decrypted)
```

Project View:(Output)









Future Improvements:

- Multi-user Authentication: Permit the registration and recognition of multiple users through facial recognition.
- Mobile Integration: Develop a mobile application for increased convenience and portability.

- Liveness Detection: Prevent video or photographic spoofing by checking for eye movements and blinking.
- GUI Development: Design a graphical user interface for improved accessibility to system functions.
- Cloud Integration: Securely store documents on the cloud, accessible by facial recognition granting users secure keys.
- Logging and Alerts: Maintain a record of failed attempts and notify the user via email or SMS.

Conclusion:

Face Lock demonstrates a groundbreaking method of AI-powered facial recognition applied to protect personal or confidential information and it scrubs the need for passwords, increasing ease of access and security simultaneously. Integrating DeepFace for face matching while employing Cryptography to secure the files creates a balanced system of privacy and convenience.
FaceLock's automated approach coupled with biometric identification facilitates smarter security systems which adapt to practical applications ranging from personal requirements to enterprise data security stowed behind layers of digital fortifications.

References:

- OpenCV Documentation
- Python Cryptography Docs
- RetinaFace Overview
- Python Official
- Face Recognition Research