
ĐẠI HỌC QUỐC GIA TPHCM

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

---

# BÁO CÁO ĐỒ ÁN 1

**Đề tài: Ứng dụng mô phỏng hệ thống bảo mật theo mô hình thực tế**

---

**Môn học: An ninh máy tính**

*Sinh viên thực hiện:*

Bế Lã Anh Thư - 22127402

Võ Hữu Tuấn - 22127439

*Giáo viên hướng dẫn:*

Thầy Phan Quốc Kỳ

Thầy Lê Hà Minh

Thầy Lê Giang Thanh

# Mục lục

# Phân công nhiệm vụ

| Nhiệm vụ | | | Thành viên đảm nhận | |
|---|---|---|---|---|
| **STT** | **Hạng mục** | **Nội dung chi tiết** | **Bế Lã Anh Thư** | **Võ Hữu Tuấn** |
| 1 | Đăng ký tài khoản người dùng | Nhập thông tin, lưu passphrase SHA-256 + salt, lưu bằng file JSON | x | |
| 2 | Đăng nhập & xác thực đa yếu tố | OTP/TOTP sinh đúng, giới hạn thời gian, xác thực mới truy cập | x | |
| 3 | Quản lý khóa RSA | Tạo cặp khóa, mã hóa private key bằng AES, kiểm tra hạn sử dụng | | x |
| 4 | QR code | Tạo QR đúng format, quét được từ ảnh, tìm kiếm khóa | x | |
| 5 | Mã hóa & giải mã tập tin | Mã hóa file bằng AES, mã hóa khóa bằng RSA, metadata, giải mã chính xác | | x |
| 6 | Ký số & xác minh chữ ký số | Ký bằng private key, xác minh bằng public key, thông báo người ký chính xác | x | |
| 7 | Cập nhật tài khoản & đổi passphrase | Sửa thông tin, đổi passphrase đảm bảo private key vẫn dùng được | x | |
| 8 | Phân quyền tài khoản | Phân quyền chức năng Admin, khóa tài khoản, xem danh sách người dùng | x | |
| 9 | Ghi log hoạt động bảo mật | Ghi log sự kiện quan trọng: login, mã hóa, ký số, lỗi truy cập | | x |
| 10 | Chia nhỏ file lớn khi mã hóa | Mã hóa block riêng (AES-GCM), kiểm tra toàn vẹn | | x |
| 11 | Kiểm tra & gia hạn khóa | Hiển thị trạng thái khóa, cho phép gia hạn hoặc tạo lại | | x |

| STT | Hạng mục | Nội dung chi tiết | Bế Lã Anh Thư | Võ Hữu Tuấn |
|---|---|---|---|---|
| 12 | Tìm kiếm public key | Tìm theo email, ngày tạo, hiển thị QR nếu có | | x |
| 13 | Giới hạn đăng nhập sai | Khoá tài khoản 5 phút sau 5 lần sai, cảnh báo người dùng | x | |
| 14 | Tuỳ chọn định dạng lưu file | Lưu gộp hoặc tách file `.enc`, `.key`, nhận diện đúng khi giải mã | | x |
| 15 | Khôi phục tài khoản | Dùng Recovery Key đúng quy trình, bảo mật | x | |
| 16 | Báo cáo, README, phân công | Viết báo cáo rõ ràng, README, minh hoạ chức năng | x | x |
| 17 | Demo sản phẩm | Demo | | x |
| 18 | Thiết kế giao diện | Giao diện ứng dụng đơn giản, dễ thao tác chức năng | x | |

Bảng 1: Bảng phân công nhiệm vụ

# 1 Giới thiệu

Mục tiêu của đồ án là xây dựng một ứng dụng mô phỏng hệ thống bảo mật theo mô hình thực tế: người dùng đăng ký, tạo khoá, mã hoá & ký tập tin, xác thực đa yếu tố, và quản trị tài khoản.

# 2 Phân Tích và Thiết Kế Hệ Thống

## 2.1 Tổng quan

- **Ngôn ngữ lập trình:** Python
- **Thư viện giao diện:** Tkinter, Pillow
- **Mã hóa và ký số:** `pyca/cryptography`, `hashlib`
- **Xác thực OTP/TOTP:** `pyotp`
- **Mã hóa QR code:** `qrcode`
- **Quét QR:** `pyzbar`
- **Định dạng lưu trữ dữ liệu:** JSON
- **Ghi lại các sự kiện bảo mật quan trọng:** security.log

## 2.2 Kiến trúc hệ thống

Hệ thống được tổ chức theo kiến trúc 3 lớp:

- **Dữ liệu (Data):** thư mục `data/` lưu thông tin người dùng, khóa công khai, QR, chữ ký cùng các file mã hóa (encrypted/decryted) và
- **Giao diện (Frontend):** thư mục `gui/`, các giao diện được tổ chức theo từng nhóm chức năng.
- **Chức năng (Backend):** thư mục `modules/`, chứa các modules xử lý logic bảo mật được thiết kế độc lập theo nguyên tắc "chia để trị".

# 3 Triển Khai Chi Tiết Các Chức Năng Bảo Mật

## 3.1 Đăng ký & Quản lý Định danh

Chức năng này được triển khai trong module `register.py`, đảm bảo an toàn ngay từ bước đầu tiên:

- **Bảo vệ mật khẩu:** Thay vì lưu mật khẩu dạng thô, hệ thống sử dụng hàm băm **SHA-256**. Để chống lại các cuộc tấn công từ điển và bảng cầu vồng (rainbow table), một chuỗi **Salt** ngẫu nhiên 16 bytes (`os.urandom(16)`) được tạo ra cho mỗi người dùng và kết hợp với passphrase trước khi băm. Công thức băm là:

$$\text{pass\_hash} = \text{SHA-256}(\text{salt} + \text{passphrase})$$

- **Khôi phục tài khoản:** Khi đăng ký, một mã khôi phục (`Recovery Key`) duy nhất được tạo ra. Mã này là phương án cuối cùng để người dùng lấy lại quyền truy cập nếu quên passphrase.

## 3.2 Đăng nhập & Xác thực Đa yếu tố (MFA)

Module `login.py` triển khai một quy trình xác thực hai bước nghiêm ngặt:

- **Chống tấn công Brute-force:** Hệ thống đếm số lần đăng nhập sai. Nếu người dùng nhập sai passphrase **5 lần** liên tiếp, tài khoản sẽ bị khóa tạm thời trong **5 phút**.
- **Xác thực đa yếu tố (MFA):** Sau khi xác thực passphrase thành công, hệ thống yêu cầu một yếu tố thứ hai là mã **TOTP**.
  - **Thiết lập:** Trong lần đăng nhập đầu tiên, hệ thống dùng `pyotp` để tạo một khóa bí mật (secret) cho người dùng. Khóa này được hiển thị dưới dạng mã QR để người dùng quét bằng các ứng dụng như Google Authenticator.
  - **Xác thực:** Ở các lần đăng nhập sau, người dùng phải cung cấp mã gồm 6 chữ số được tạo ra bởi ứng dụng xác thực. Hệ thống sẽ dùng hàm `totp.verify()` để kiểm tra tính hợp lệ của mã này.

## 3.3 Quản lý thông tin người dùng

Chức năng này được triển khai trong module `account_manager.py`, hỗ trợ các thao tác cơ bản liên quan đến thông tin cá nhân của người dùng như:

1. **Hiển thị thông tin cá nhân:** Hiển thị thông tin cơ bản của người dùng đang đăng nhập như:

- Email
- Họ tên
- Ngày sinh
- Số điện thoại
- Địa chỉ
- Vai trò (admin/user)
- Trạng thái tài khoản (bị khóa / đang hoạt động)

2. **Cập nhật thông tin cá nhân:** Người dùng có thể cập nhật các trường:
   - Họ tên
   - Ngày sinh (theo định dạng YYYY-MM-DD)
   - Số điện thoại
   - Địa chỉ Nếu thông tin không hợp lệ (ví dụ sai định dạng ngày tháng hoặc số điện thoại không hợp lệ), hệ thống sẽ cảnh báo và không cho lưu.
3. **Đổi Passphrase:** Người dùng cần nhập đúng passphrase hiện tại để đổi sang passphrase mới. Passphrase mới phải đủ mạnh (ít nhất 8 ký tự, có chữ hoa, số, ký tự đặc biệt)

## 3.4 Quản lý Khóa RSA

Đây là chức năng cốt lõi, được xử lý bởi `rsa_key_manager.py`:

- **Tạo khóa:** Hệ thống tạo ra cặp khóa **RSA 2048-bit**, một độ dài an toàn cho các ứng dụng hiện tại.
- **Bảo vệ Khóa riêng tư (Private Key):** Khóa riêng tư là tài sản tối mật, do đó nó không bao giờ được lưu ở dạng văn bản gốc. Thay vào đó, nó được mã hóa bằng thuật toán **AES ở chế độ EAX**. Chế độ EAX được chọn vì nó cung cấp đồng thời cả tính bí mật (encryption) và tính toàn vẹn (authenticity).
- **Dẫn xuất khóa (Key Derivation):** Khóa AES dùng để mã hóa khóa riêng tư không được sinh trực tiếp từ passphrase. Để tăng cường bảo mật, passphrase của người dùng được đưa qua hàm dẫn xuất khóa **PBKDF2**. PBKDF2 thực hiện hàng nghìn vòng lặp băm, khiến cho việc dò tìm khóa từ passphrase trở nên cực kỳ tốn kém về mặt tính toán.

## 3.5 Mã hóa & Giải mã File

Để mã hóa các file lớn một cách hiệu quả và an toàn, đồ án áp dụng mô hình mã hóa lai (Hybrid Encryption) trong module `file_encryptor.py`:

1. Hệ thống tạo một khóa đối xứng ngẫu nhiên, gọi là **khóa phiên (session key)**, sử dụng thuật toán **AES-256**.
2. Khóa phiên này sau đó được mã hóa bằng **khóa công khai RSA** của người nhận. Quá trình này sử dụng padding **PKCS1_OAEP**, một chuẩn an toàn giúp chống lại các cuộc tấn công vào RSA.
3. Nội dung file gốc được mã hóa bằng khóa phiên với thuật toán **AES ở chế độ GCM**. Chế độ GCM (Galois/Counter Mode) được ưu tiên lựa chọn vì nó không chỉ mã hóa dữ liệu mà còn tạo ra một thẻ xác thực (authentication tag), đảm bảo dữ liệu không bị thay đổi trên đường truyền.
4. Để xử lý các file lớn hơn 5MB, file được chia thành các khối 1MB và được mã hóa riêng rẽ.

## 3.6 Ký số & Xác minh Chữ ký

Module `signature.py` cho phép người dùng xác thực nguồn gốc và tính toàn vẹn của file:

- **Quy trình Ký số:**
  1. File cần ký được đưa qua hàm băm **SHA-256** để tạo ra một chuỗi đại diện duy nhất (digest).
  2. Chuỗi digest này được ký bằng **khóa riêng tư RSA** của người gửi. Quá trình ký sử dụng lược đồ `pkcs1_15`.
  3. Kết quả là một file chữ ký số (`.sig`) chứa thông tin người ký và chữ ký đã mã hóa.
- **Quy trình Xác minh:**
  1. Hệ thống băm file gốc bằng SHA-256 để có digest.
  2. Hệ thống dùng **khóa công khai** của người ký để giải mã file chữ ký, thu được digest ban đầu.
  3. Nếu hai chuỗi digest trùng khớp, chữ ký được xác nhận là hợp lệ, chứng tỏ file không bị thay đổi và đúng là do người đó ký.

## 3.7 Mã hoá thông tin qua QR Code

Chức năng này được triển khai trong module `qr_code_manager.py`. Hệ thống sử dụng QR Code để mã hóa và chia sẻ thông tin công khai của người dùng (public key) dưới dạng dễ quét và truyền tải. Cụ thể:

- Khi người dùng sử dụng chức năng này, hệ thống sẽ tạo QR chứa:
  - Email người dùng
  - Ngày tạo khóa
  - Public Key dạng PEM (được cắt gọn và mã hóa base64)
- Sử dụng thư viện `qrcode` để tạo QR từ dữ liệu JSON trên.
- Thư viện `pyzbar` dùng để quét QR từ ảnh và trích xuất lại thông tin.
- Người dùng khác có thể quét QR để lưu public key và xác minh chữ ký.

Việc sử dụng QR Code giúp đơn giản hóa quá trình chia sẻ khoá công khai, đặc biệt trong môi trường thiếu tiện ích sao chép truyền thống.

## 3.8 Phân quyền tài khoản

Hệ thống phân chia tài khoản thành hai vai trò chính:

- **User (Người dùng):** chỉ được sử dụng các chức năng cá nhân như đăng nhập, mã hóa, ký số, quản lý khóa riêng.
- **Admin (Quản trị viên):** ngoài các quyền như User, còn có các quyền nâng cao như:
  - Xem danh sách tất cả tài khoản.
  - Khóa hoặc mở khóa tài khoản bất kỳ (chức năng `lock/unlock`).
  - Xem nhật ký bảo mật toàn hệ thống (`security.log`).
- Các quyền nâng cao của quản trị viên trên được triển khai trong module `admin_tools.py` và gọi qua giao diện `AccountAdminFrame`.
- Quyền truy cập được kiểm tra sau khi đăng nhập dựa trên trường vai trò `is_admin` được lưu trong file thông tin người dùng `users/json`.
- Cơ chế phân quyền đảm bảo rằng các chức năng nhạy cảm chỉ được thao tác bởi quản trị viên của hệ thống.

## 3.9 Khôi phục tài khoản

Hệ thống hỗ trợ người dùng khôi phục tài khoản trong trường hợp quên passphrase bằng cách sử dụng **recovery key** đã cấp phát sau khi đăng ký. Cụ thể:

- Người dùng nhập **email**, **recovery key** và **passphrase mới**.
- Hệ thống xác thực **recovery key**, sau đó:
- Cập nhật **salt** và **passphrase** mới.
- Giải mã và mã hóa lại **private key RSA** bằng passphrase mới (nếu có).
- Ghi log quá trình khôi phục vào **security.log**

## 3.10 Ghi log hoạt động bảo mật

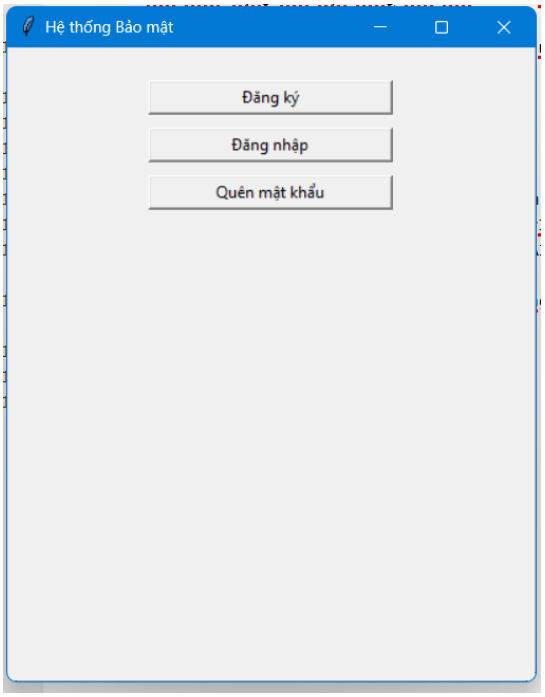
Toàn bộ các hoạt động quan trọng trong hệ thống đều được ghi lại vào một tệp log có tên là `security.log` được lưu ở thư mục `data/`. Chức năng ghi log hoạt động bảo mật này được triển khai trong module `logger.py`.
Cụ thể, các hành động được ghi lại bao gồm:

- Đăng ký, Đăng nhập tài khoản thành công/thất bại, .
- Các thao tác sử dụng các chức năng bảo mật như tạo khóa, gia hạn khóa, mã hóa/giải mã file, ký số và xác minh chữ ký.
- Các hành động bảo mật nâng cao của admin như khóa/mở tài khoản, xem danh sách tài khoản.
- Các lỗi truy cập hoặc hành vi sai (ví dụ như nhập sai passphrase nhiều lần).

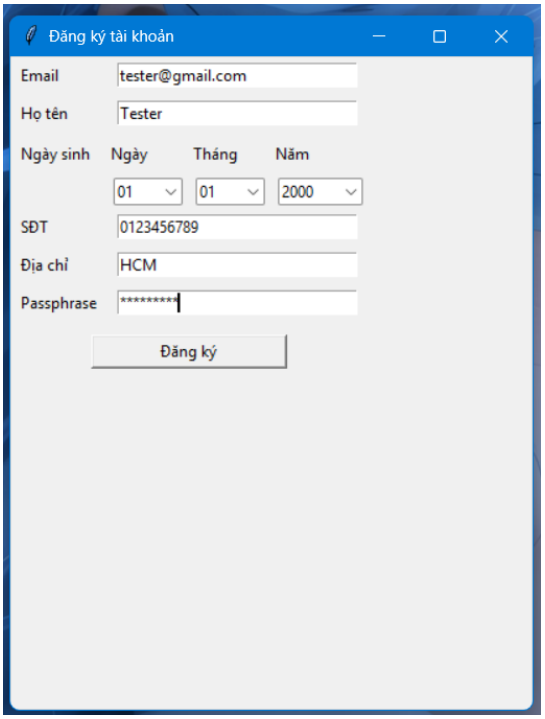
Mỗi dòng log bao gồm:

- **Thời gian (timestamp)**: thời điểm ghi sự kiện.
- **Người thực hiện**: email hoặc userID.
- **Loại hành động**: ví dụ "LOGIN_FAILED", "ENCRYPT_SUCCESS", "ADMIN_LOCKED_USER".
- **Mô tả chi tiết**: thông tin mở rộng để kiểm tra hoặc phân tích.

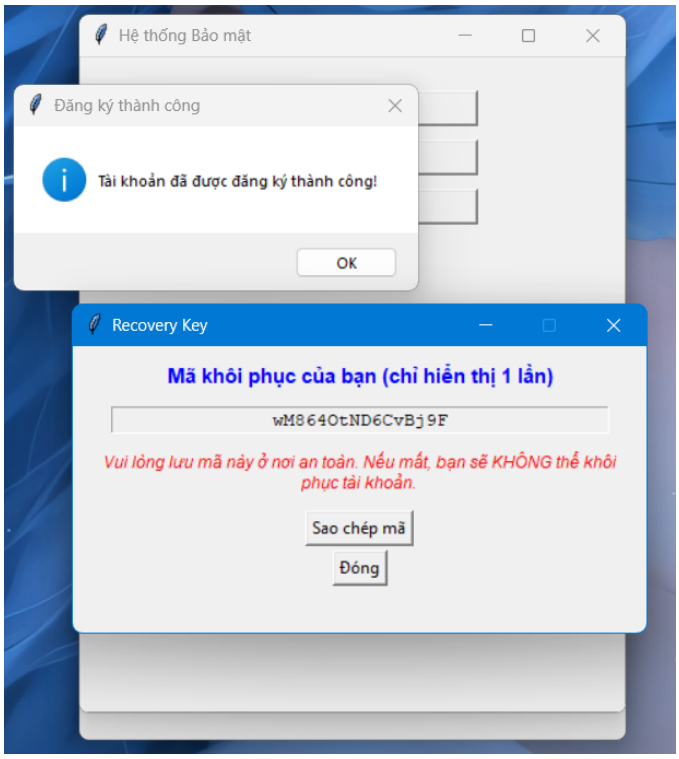
# 4 Giao diện và kết quả thực thi



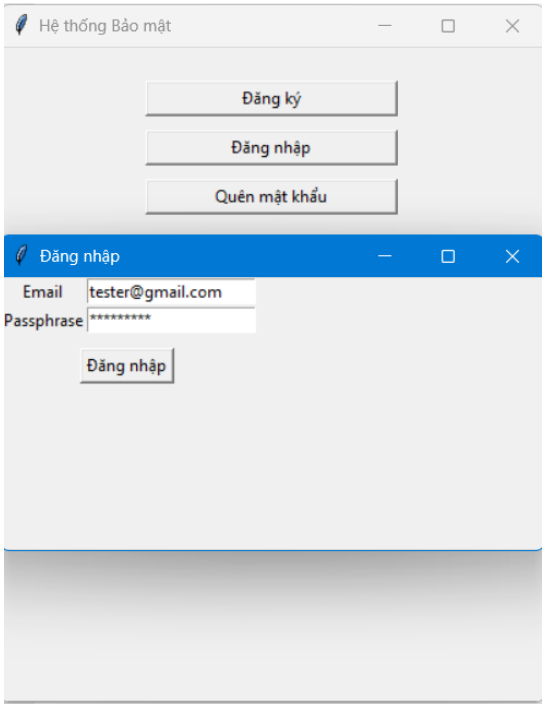

Hình 1: Giao diện đăng nhập



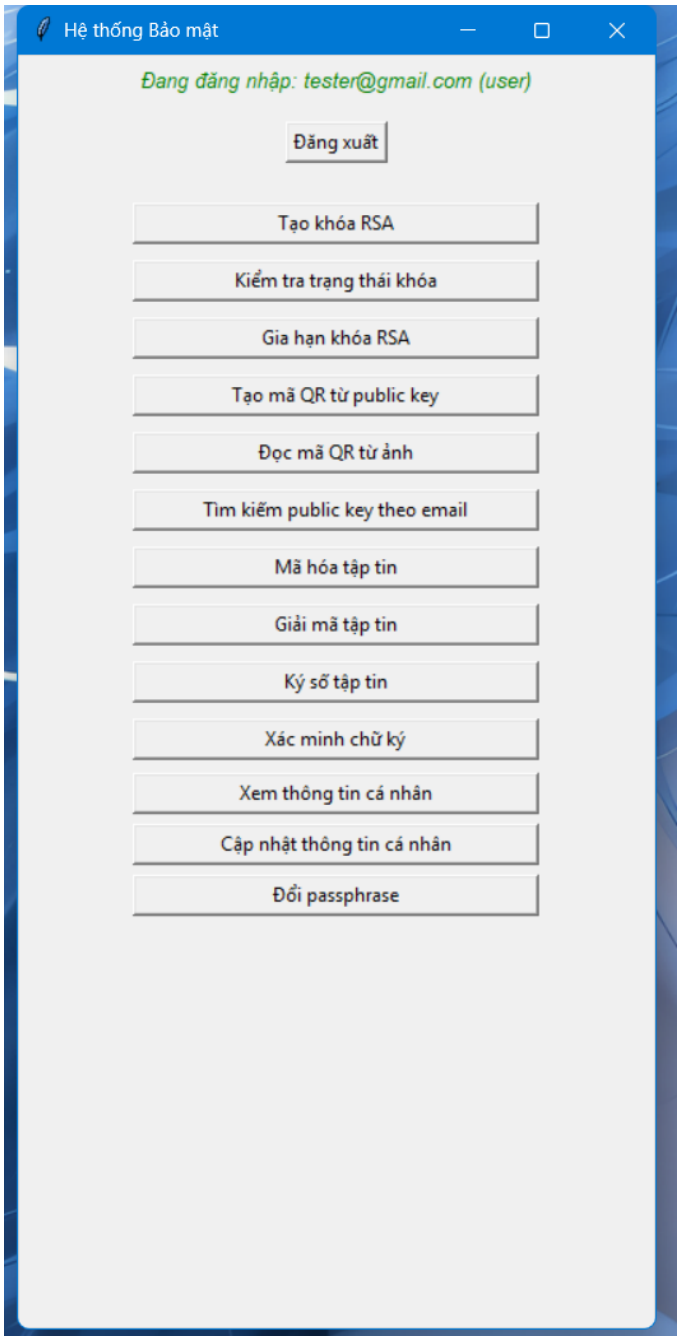

Hình 2: Đăng ký tài khoản



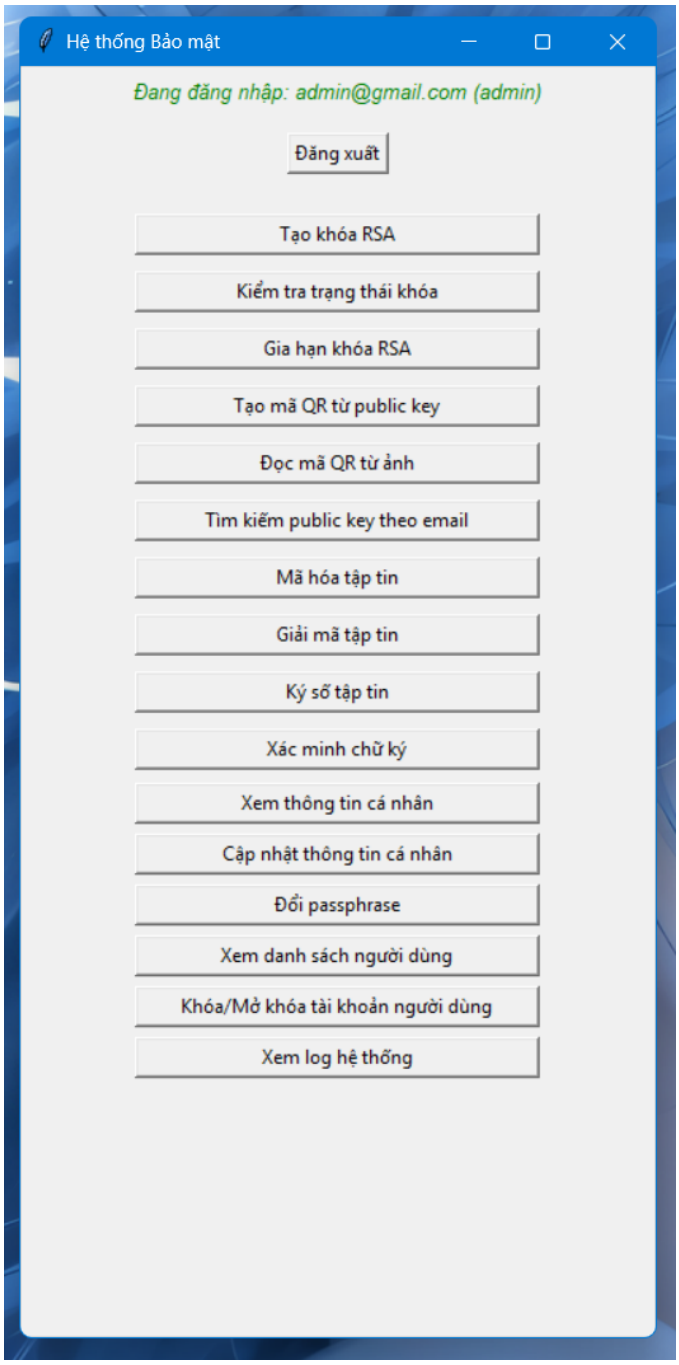

Hình 3: Thông báo đăng ký thành công (và mã khôi phục)



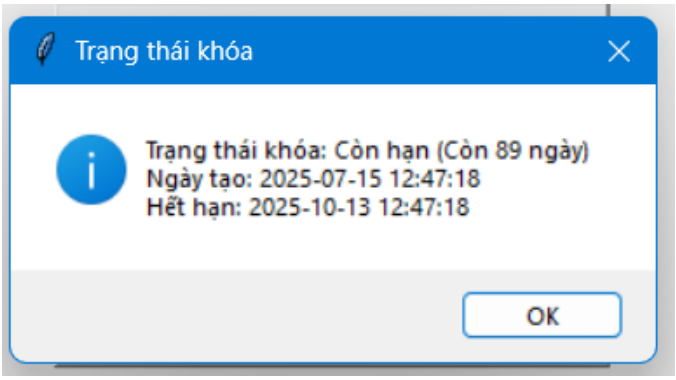

Hình 4: Đăng nhập tài khoản




Hình 5: Giao diện chính của người dùng (User)

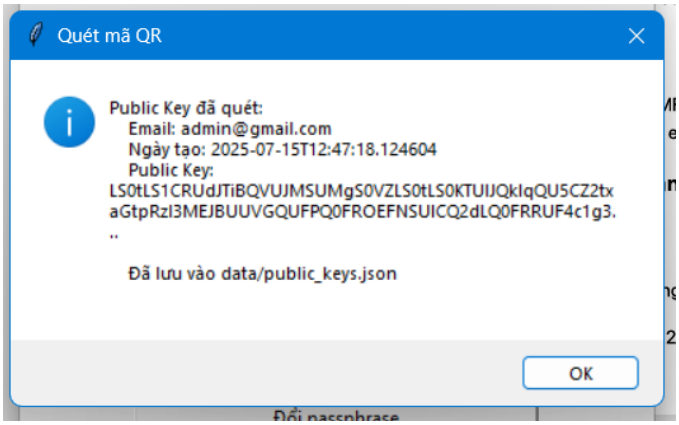



Hình 6: Giao diện chính của quản trị viên (admin)

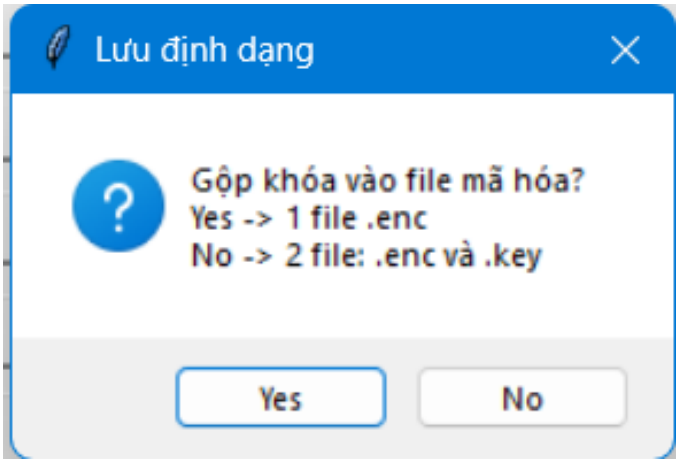



Hình 7: Kiểm tra trạng thái RSA

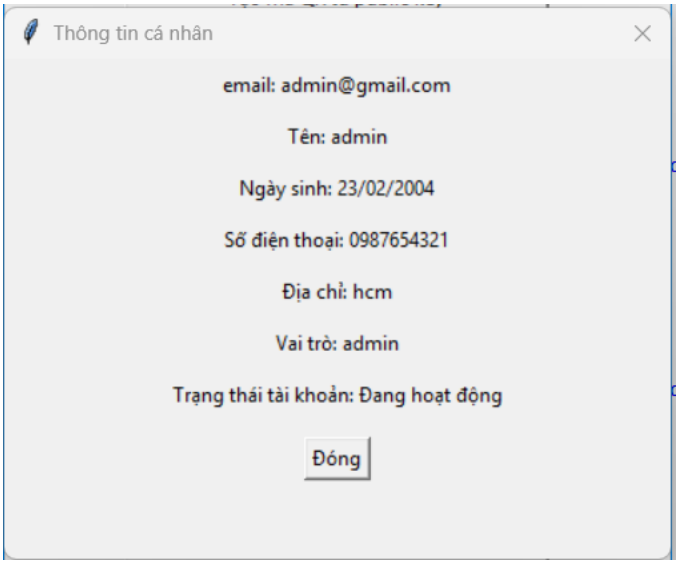

Hình 8: Tạo mã QR từ Public key



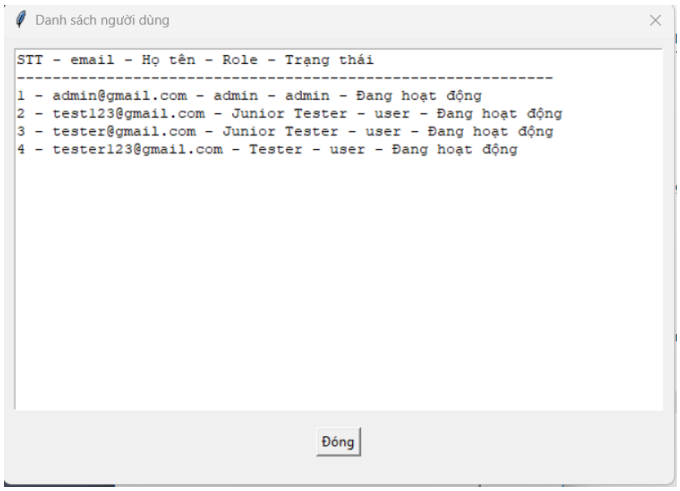

Hình 9: Kết quả trả về khi quét mã QR code Public key



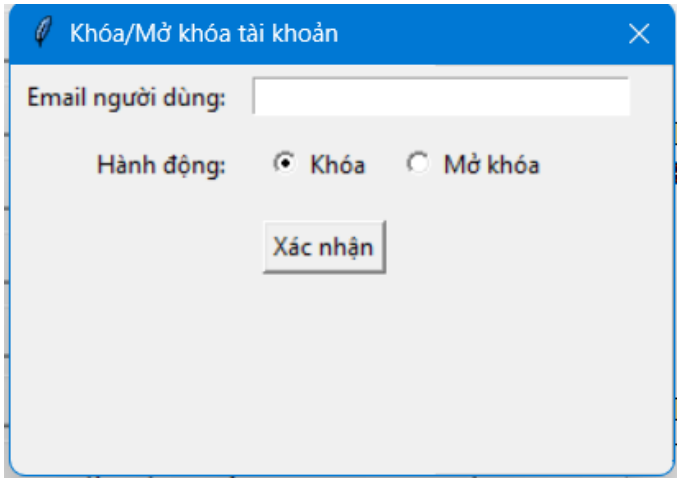

Hình 10: Tùy chọn lưu định dạng file



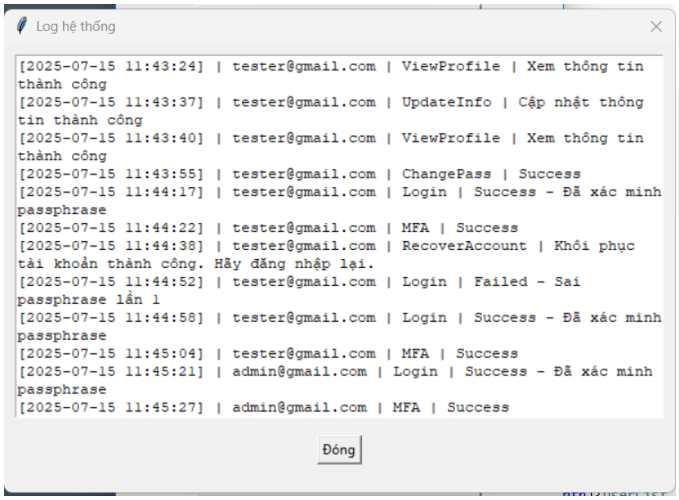

Hình 11: Giao diện xem thông tin cá nhân của người dùng đang đăng nhập



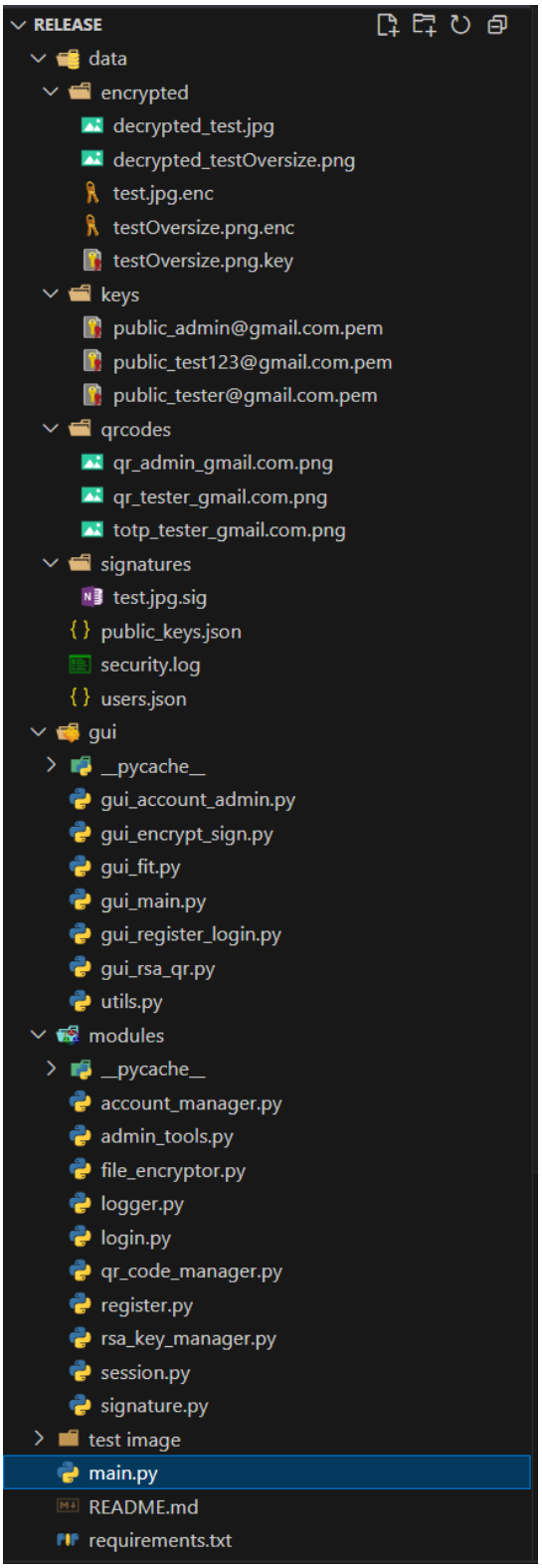

Hình 12: Giao diện danh sách người dùng (Quyền quản trị viên)




Hình 13: Giao diện khóa/mở khóa tài khoản người dùng (Quyền quản trị viên)




Hình 14: Giao diện xem log hệ thống (Quyền quản trị viên)




Hình 15: Cấu trúc thư mục của hệ thống

# 5 Hướng Dẫn Sử Dụng

Đọc hướng dẫn chi tiết tại README.md

# 6 Kết Luận

Đồ án đã đáp ứng tất cả các yêu cầu đặt ra và mô phỏng gần như đầy đủ các khía cạnh bảo mật thực tế: từ định danh, mã hóa, xác thực, đến phân quyền quản trị. Hệ thống lưu trữ thông tin dữ liệu bằng JSON giúp đơn giản hóa và dễ triển khai.

# Phụ Lục

## A. Liên kết mã nguồn

- Link GitHub Repository
- Link Test Data Files

## B. Video demo

- Link Google Drive

# Tài liệu Tham Khảo

1. Cryptography documentation
2. PyOTP documentation
3. tkinter documentation
4. qrcode documentation
5. pyzbar documentation

/