CypherSafe

Encryption, Decryption & Hashing

Mohamed Achek
Mohamed Amine Saoudi

TUNIS BUSINESS SCHOOL
UNIVERSITY OF TUNIS

Agenda

1. Overview
2. Users
3. Architecture
4. Data Flow
5. Core Functions & steps
6. Tools & Libraries
7. Development Phases

Project Overview

CypherSafe is a security-focused tool that enables users to:

- Encrypt and decrypt files securely
- Generate file hashes (SHA256, SHA512, etc.)
- Use both CLI and GUI interfaces
- Ensure robustness, user-friendliness, and portability

Roles and Users

Project Roles	
Role	**Description**
End User	Uses the tool to hash, encrypt, or decrypt files. No technical expertise required.
Developer	Builds and maintains cryptographic core and interfaces.

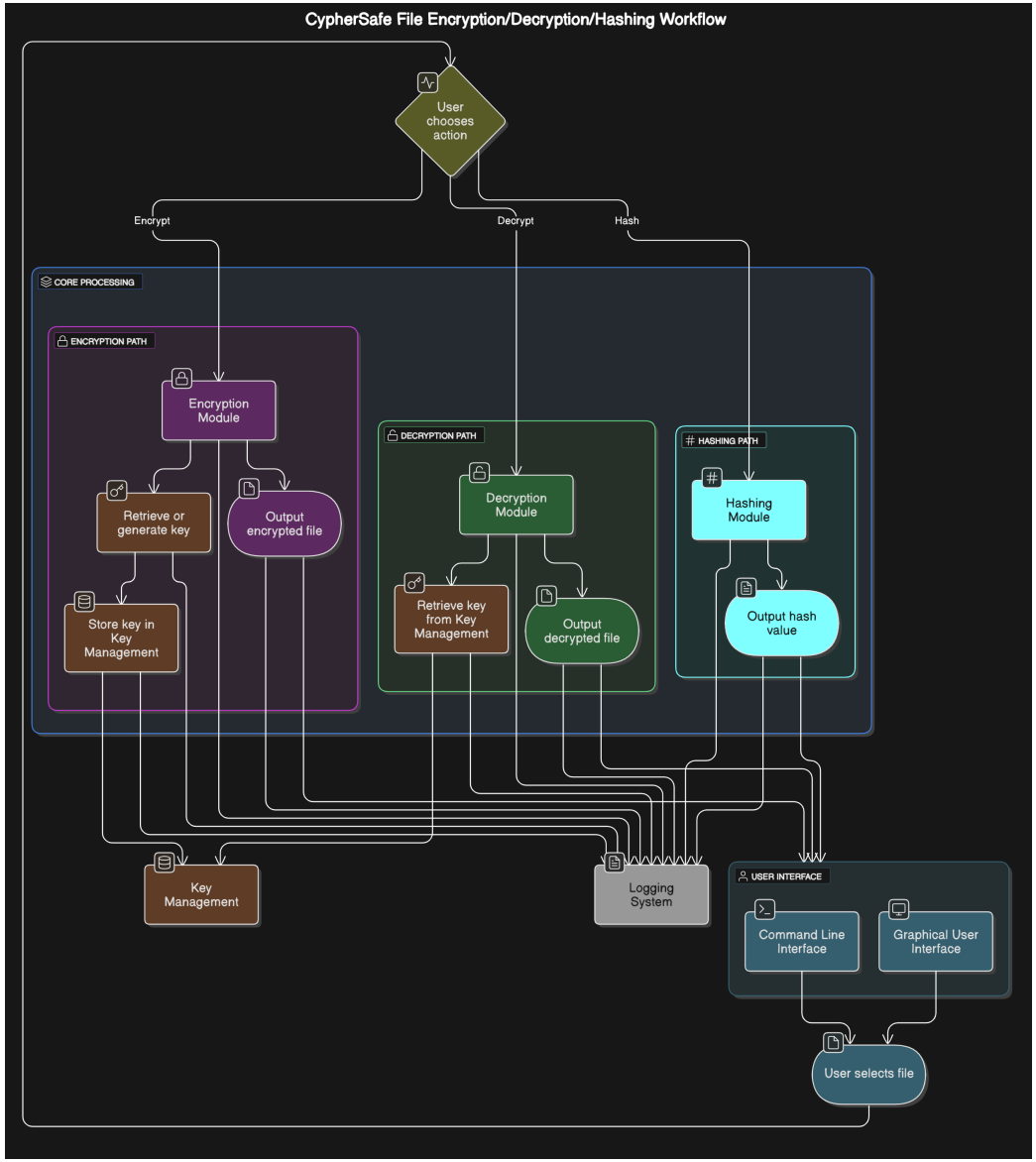
System Components

System Components	
Component	**Responsibility**
User Interface (CLI/GUI)	User input, messages, file selection
encryption.py	Encrypts/decrypts (Fernet (AES) /or RSA)
hashing.py	Generates file hashes (SHA256, etc.)
utils.py	File I/O, key generation helpers
keys/user_keys.json	Stores encryption keys securely
Logger	Logs major actions/errors
Backup System	Backups before encryption

Data Flow: General Workflow

1. User launches the tool (CLI or GUI)
2. Selects a file and desired operation
3. Tool processes file (encryption / decryption or hashing)
4. Output is displayed/saved securely

Data Flow: Diagram



Core Functions and & steps

Main Functionalities of CypherSafe

- **Encryption**
  - User inputs data and selects encryption algorithm .
  - Key is generated or selected.
  - Encrypted data is produced and optionally stored or transmitted.
- **Decryption**
  - Encrypted data and the correct decryption key are provided.
  - Decryption function reverses the encryption process.
  - Original file is recovered.
- **Hashing**
  - Input data is processed using a hashing algorithm (e.g., SHA-256).
  - A fixed-length hash is returned for integrity verification.

Tools & Libraries

Tools and Libraries Used	
Tool/Library	**Use**
`Python 3.12+`	Core language
`cryptography`	Secure encryption/decryption **(Fernet)**
`hashlib`	Hashing algorithms (SHA256, etc.)
`streamlit`	GUI (file picker, buttons, status text)
`os / base64 / json`	File ops, encoding, key storage
`PyInstaller`	Packaging the app into `.exe` or `.app`

Development Phases

Project Lifecycle

1. **Requirement Analysis**: Define security objectives and user needs.
2. **System Design**: Specify modules (Encryption, Decryption, Hashing, UI).
3. **Implementation**: Write and test each function in isolation.
4. **Testing**: Functional testing + security validation.
5. **Documentation and Deployment**: Write user manual, publish on GitHub or package.