**Internship Project Report**

## **Title: Ransomware Simulation: A File Encryption-Decryption Demonstrator**

**Intern Name: Abinaya Balakrishnan  
Domain: Cybersecurity and Ethical Hacking  
Organization: Tamizhan Skills**

**I. Introduction**

This report presents the design and development of a **Ransomware Simulation by encrypting files** completed as part of my cybersecurity internship at Tamizhan Skills.The primary objective of this project was to simulate the core behavior of ransomware—specifically the encryption of user files and the presentation of a simulated ransom interface—using **Python** for backend logic and **Streamlit** for the graphical user interface.

What sets this project apart from other simulations is its **realistic user interaction**: when a file (ope is clicked in the interface, it appears to preview normally, but in the background, it is silently **encrypted in real-time**, mimicking the deceptive behavior of actual ransomware. This social engineering aspect makes the simulation both engaging and impactful, helping users understand how seemingly safe interactions can trigger harmful consequences.

The project was created **educational purposes**, with no malicious intent.

Throughout the development process, I gained **practical experience in secure coding practices**, cryptographic implementation (using the cryptography module), and frontend development. The simulation also served as a platform to explore and demonstrate key cybersecurity concepts such as **password-based encryption, file handling vulnerabilities, and the impact of social engineering**.

In addition to technical learning, the project emphasized **cybersecurity ethics**, responsible software development, and the importance of user awareness in preventing real-world cyber threats.

**II. Background and Significance**

Ransomware is a form of malware that encrypts user data and demands a ransom for its release. Over the past decade, ransomware attacks have caused billions of dollars in damage worldwide, affecting hospitals, schools, corporations, and individuals. Real-world examples include WannaCry, Petya, and REvil.

**III. Objectives**

* Simulate ransomware behavior in a safe, controlled environment.
* Implement file encryption and decryption using password-based keys.
* Design an interactive GUI using Streamlit.
* Demonstrate social engineering tactics through a mock file explorer and payment interface.
* Promote cybersecurity awareness and safe file handling practices.

**IV. Tools and Technologies Used**

| **Tool/Technology** | **Purpose** |
| --- | --- |
| Python | Programming and script automation |
| Cryptography | Secure encryption and decryption (Fernet, PBKDF2HMAC) |
| Streamlit | GUI development for web interface |
| Subprocess | Script invocation from GUI components |
| OS module | File and directory operations |
| Session State | Managing dynamic behavior in Streamlit |

**V. Project Modules**

1. **Key Generation (key.py)**  
   Generated a Fernet-compatible encryption key derived from a password using PBKDF2HMAC and a random salt stored in **salt.bin.**
2. **Encryption Script (encrypt.py)**  
   Read the target file in binary mode, encrypted the data, and overwrote the original file to simulate ransomware locking behavior.
3. **Decryption Script (decrypt.py)**  
   Reconstructed the original key using the saved salt and password, then decrypted the encrypted file to restore access.
4. **Main GUI (home.py)**  
   Presented a list of decoy files to the user. When a file like open.txt was clicked, the encryption process was triggered, followed by a simulated ransom message.
5. **Mock file explorer(files.py)**

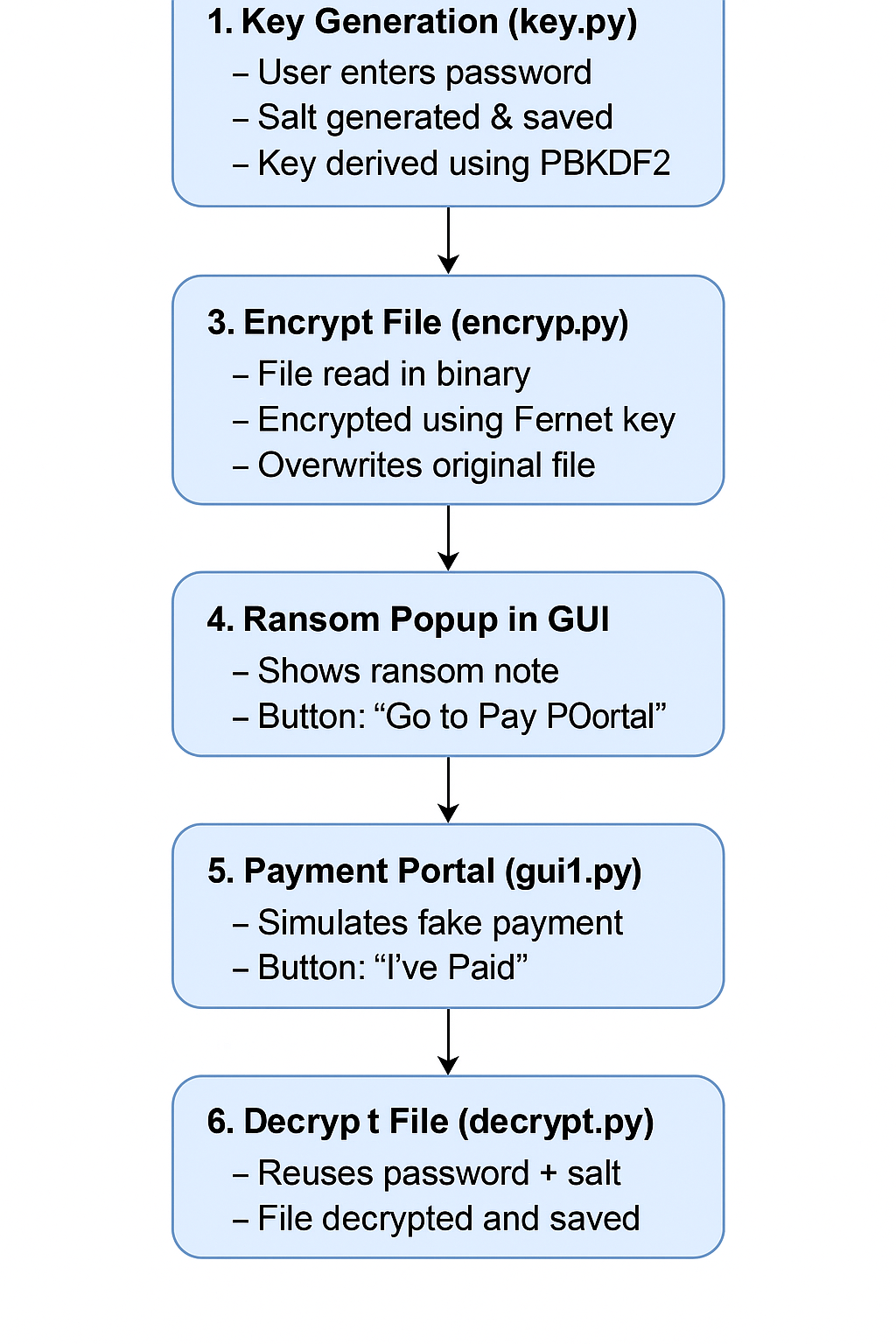
Displayed a mock file explorer which lists out all the files in the folder (my files),while trying to access any of the files in the folder the encryption script is triggered.

1. **Payment Portal (pay.py)**  
   Displayed a fake payment interface. Once the user clicked “I’ve Paid,” it invoked the decryption script to restore file access.

**Why Bitcoin (BTC) Was Used in the Simulation?**

Bitcoin is commonly used in real-world ransomware attacks due to its anonymity, decentralization, and irreversible transactions. These features make it harder to trace or block payments, giving attackers more control. Including a fake BTC address in the simulation adds realism, helping users understand how actual ransom demands typically appear. This approach strengthens the educational value of the project while ensuring no real transactions are involved.

**VI. System Architecture / Workflow**



**VII. Security Considerations**

* **PBKDF2HMAC**: A widely recommended key derivation function that applies a hash algorithm (SHA-256) 390000 times to a password. This slows down brute-force attacks.
* **Salt**: A random value added to the password before hashing. Prevents attackers from using precomputed hash tables (rainbow tables).
* **Key Storage**: Instead of storing raw keys, the key is derived during runtime using the password + saved salt. If the salt or password is lost, decryption becomes impossible.
* **Safe File Handling**: Binary modes (rb/wb) were used to prevent corruption during encryption.

**VIII. Challenges, Errors & Solutions**

| **Issue** | **Cause** | **Solution** | **What I Learnt** |
| --- | --- | --- | --- |
| Encryption not triggering | Incorrect script name in subprocess | Corrected to the exact script filename | Double-check subprocess syntax and paths |
| File not encrypted | Script ran asynchronously | Used delay and checked file state manually | subprocess requires manual verification |
| Path issues | Used backslashes in file paths | Replaced with os.path.join() or forward slashes | Ensure OS-independent path compatibility |
| Encoding errors | Used text mode (r/w) instead of binary | Switched to binary mode (rb/wb) | Encryption should always handle binary data |
| Unusable key | Saved key directly without salt | Used PBKDF2HMAC with salt to regenerate the key | Learned secure key derivation best practices |
| Password input failure | Used getpass in Streamlit | Replaced with text input box | GUI input must use compatible functions |
| State resetting on rerun | Used st.rerun() without state handling | Stored values in st.session\_state | Proper session management prevents state loss |

**IX. Testing and Validation**

* **Encryption Validation**: Tested encryption on .txt and dummy files. Verified that original content was replaced by encrypted data.
* **Decryption Validation**: Checked if the same password and salt reliably restored files.
* **Edge Cases**:
  + Tried decrypting with a wrong password → Result: failed decryption, file unreadable (expected).
  + Deleted salt.bin and tested → Result: decryption impossible.
* **GUI Testing**: Verified that buttons appeared conditionally, file actions triggered subprocess correctly, and file content changed as expected.

**X. User Experience and Interface Design**

* Designed GUI using Streamlit’s layout features for clean and centered display.
* Used dark mode CSS for visual clarity and realism.
* Applied conditional logic to show popup only after encryption.

**XI. What Makes This Project Unique**

**1. Realistic Attack Trigger** Unlike basic simulators that require manual file uploads, this tool automatically encrypts files when users attempt to access seemingly innocent files (like open.txt), closely mimicking real-world social engineering tactics and user curiosity exploitation.

**2. Complete Attack Simulation** Goes beyond simple encryption by presenting users with authentic ransom notes and simulated Bitcoin payment portals, providing comprehensive exposure to the full ransomware experience without actual risk.

* **3. Educational Impact Through Realism** Built specifically for cybersecurity awareness training, the tool demonstrates how easily attacks can be triggered by innocent-looking files, making abstract threats tangible and memorable for users.

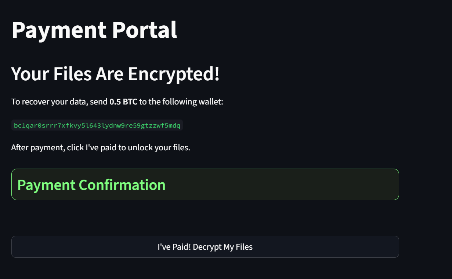
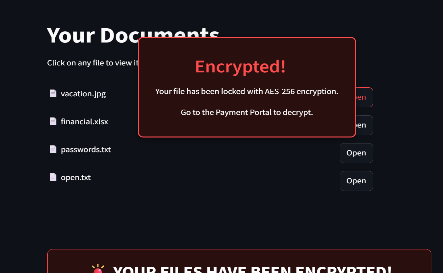
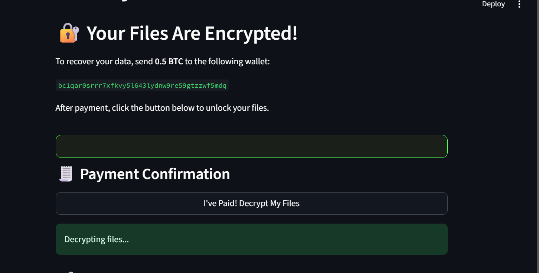
**XII. What I Learned**

* Acquired practical knowledge of Fernet encryption and PBKDF2HMAC-based key derivation.
* Gained experience building interactive Python-based GUI tools using Streamlit.
* Understood the psychology behind social engineering and how ransomware exploits user trust.
* Learned to debug errors systematically, improving coding and troubleshooting skills.
* Practiced modular programming and organized script-based architecture.
* Developed a better appreciation for the ethical responsibilities involved in cybersecurity simulation.

**XIII. Project Outcomes**

* Developed a functional ransomware simulation project.
* Created a secure encryption and decryption system.
* Designed an engaging and informative user interface using Streamlit.
* Learned to integrate multiple Python modules for a unified application.
* Gained insight into ransomware techniques and how to protect against them.
* Gained insight of real world Ransome problem and cyber ethics.

**XIV. Working model**



**XV. References**

* **Fernet (Symmetric Encryption) and PBKDF2HMAC Key Derivation**

-*Used for implementing secure file encryption using Fernet and deriving keys with PBKDF2HMAC.*

[Fernet (symmetric encryption) — Cryptography 46.0.0.dev1 documentation](https://cryptography.io/en/latest/fernet/#using-passwords-with-fernet)

* **NIST Special Publication 800-132**  
  *used for Password-Based Key Derivation Functions (PBKDF2)*  
  National Institute of Standards and Technology — <https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-132.pdf>
* **Cybersight Malware Sample Library – Ransomware Section**  
  *Overview and Behavior of Real-World Ransomware Samples*

[Ransomware - Cybersight Malware Samples](https://cybersight-security.github.io/Malware-Samples/ransomware.html)  
-Used to understand ransomware behavior and structure the simulation accordingly.

* Streamlit documentation guided the GUI design and interactive elements.

[Streamlit API cheat sheet - Streamlit Docs](https://docs.streamlit.io/develop/quick-reference/cheat-sheet)

**XVI. Conclusion**

This internship has significantly contributed to both my technical skills and ethical understanding within the cybersecurity field. By building a fully functional ransomware simulator, I was able to apply cryptographic concepts, GUI design principles, and secure programming techniques. The errors and challenges I encountered were valuable learning moments that helped me become a more careful and reflective developer. This project reinforced the importance of responsible programming, especially when simulating sensitive cybersecurity threats.