**Final Report**

**ON**

**Encrypted USB File Backup Tool**

By

SANJANA GOUD NALLA (001097128)

GOWTHAM SAI BODDULURI(001120517)

Instructor

Dr. Manohar Raavi

Department of Computer Science

**Final Report**

**Abstract**

The Encrypted USB File Backup Tool solves the most urgent problem of protecting data on portable USB devices from unauthorized persons who may steal, or access its content with ill intentions. As for the encryption methods, this tool uses both AES to encrypt files at a higher speed and essential security and RSA to protect the keys. An intuitive and easy-to-use graphical user interface (GUI) makes encryption and decryption more accessible for regular users while allowing easy processing of all standard file formats and directory structures. The significant accomplishments include optimized performance, secure data handling, and smooth end-user functionality. This tool strengthens data safety for people and companies. Additional features that are expected to enhance it in the future include multi-factor authentication and cloud key management.

**Introduction**

Increased breaches, theft, and unauthorized access require secure solutions, especially for portable storage like USB drives. Flash drives are widely accepted for transferring and storing personal and corporate data and are very susceptible to loss or theft because of their small size and portability (Singh et al., 2022). Especially when unencrypted data on USB drives is compromised, individuals and organizations stand to lose their data and incur substantial financial losses. Encryption is commonly used to protect the identity of data, which is converted into codes that cannot be easily understood by anyone other than authorized users in a situation where the device on which they were stored is lost or stolen (Sassani et al., 2020).

This project addresses these challenges through the use of the Advanced Encryption Standard (AES) for efficient symmetric encryption and Rivest Shamir Adleman (RSA) for the asymmetric encryption of keys, combining their strengths to create a secure, user‐friendly tool for USB file encryption and decryption. File contents are AES encrypted with speed and reliability, and RSA supplies the safety of the AES keys, providing a double layer of protection (Pattanavichai, 2022). Encrypted USB File Backup Tool gives users an intuitive graphical user interface (GUI) from which even beginners can protect their data. The project aims to increase data security, usability, and portability by offering a solution that protects essential information and addresses hazards related to USB storage devices.

**Methodology**

As depicted in Figure 1 below, the Encrypted USB File Backup Tool aims to provide secure and user-friendly encryption and decryption for any data stored on a USB, as seen in the pinned icon and home screen. The tool is developed in Python with robust libraries, such as cryptography libraries for encryption, tkinter for the user interface (GUI), and psutil for USB drive detection. Finally, our symmetric encryption uses AES (Advanced Encryption Standard), based on Galois/Counter Mode (GCM), to protect the content of a file and ensure its confidentiality and integrity. RSA is used to secure AES encryption keys through RSA asymmetric encryption, encrypting the key with a public RSA key, requiring the corresponding private key to decrypt, and all the functionality is in separate Python files for code modularity, as shown in Figure 2 below. This approach is layered to prevent lost USB from being accessed by unauthorized persons so that sensitive files stay protected.

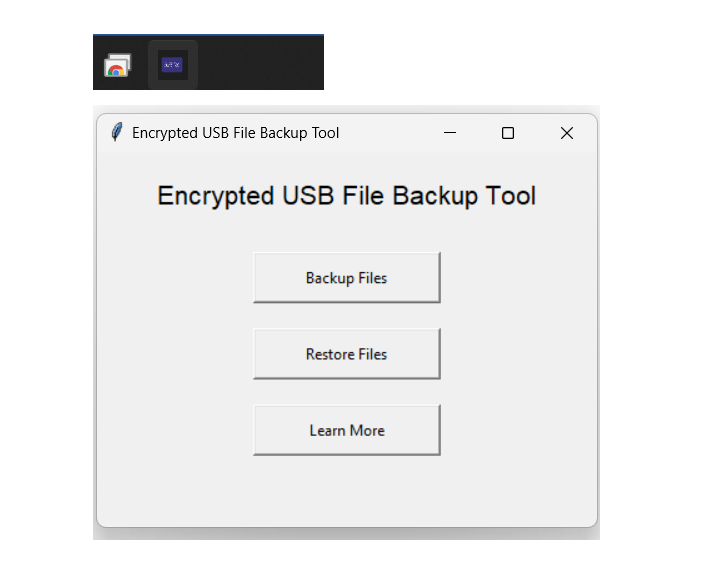


Figure 1 shows the Encrypted USB Backup Tool Icon and the product landing screen in session.

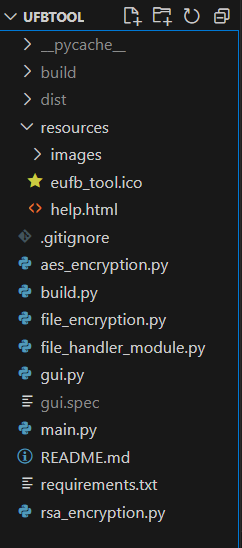


Figure 2 shows the USBTool code structure, showing modularity.

First, the AES key and an Initialization Vector (IV) are randomly generated with a secure random function, each 32 bytes. This technique encrypts the file content with AES in GCM mode, resulting in the ciphertext and an authentication tag for integrity verification. Then, the AES key is encrypted using the RSA public key so that the key is safe and cannot be retrieved without the private key. The tool places new encrypted files, keys, and metadata into specific directories on the USB drive for easy storage and retrieval, as shown in Figure 3 below. Decryption reverses the process: The user decrypts the AES key using the provided private RSA key. The file is decrypted using the AES key to regain the original content.

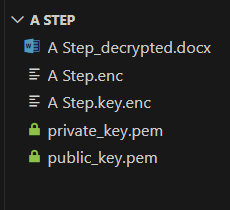


Figure 3 below shows an example of the encrypted Word document and the file after decryption.

The GUI provides options to encrypt files, batch process a directory, or decrypt data to facilitate user interaction with the tool, as shown in Figure 4 below. Users will select drives and files through intuitive prompts, and the drives are automatically detected. Moreover, the app offers a simple webpage that provides detailed instructions on how the app can be used for new users. Real-time updates are given with progress bars, and confirmations are given as complete operations. It works with several file formats for broad compatibility. The project strives for portability and ease of deployment across devices by packaging the application into a standalone executable using pyinstaller. The tool takes a balanced approach — it allows users to protect their data securely without knowing how to while remaining compliant with strict encryption standards.

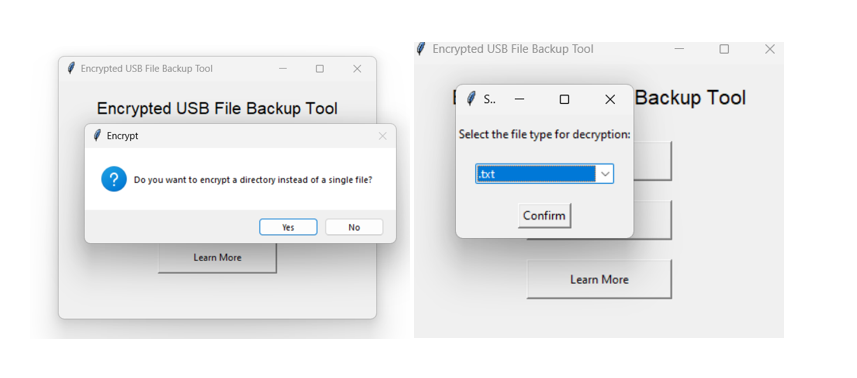


Figure 4 shows batch processing and various file types selection.

**Results**

During testing, the Encrypted USB File Backup Tool performed outstandingly. With AES encryption being known for being very fast and efficient, we could successfully process any big file — up to 285 MB without noticeable delays. The AES keys were RSA encrypted, adding negligible overhead, but AES's encryption and decryption workflows were still fast and seamless. The graphical user interface (GUI) was friendly even for nontechnical users, allowing them to interact with the tool effortlessly as far as the selection of files, detection of USB devices, and progress monitoring. The tool also gained functionality for batch processing of directories and supporting multiple file formats to cover different user needs.

The tool was robust, given the security checking results. When USB drives were purposely lost or accessed from unauthorized systems, the files were still secure. As a result, the AES keys—encrypted with RSA—were unlikely to be decrypted without authorization, providing extra security. This tool was extensively tested to prove its reliability under different file formats and directory sizes and to provide adequate error handling when private keys for decryption don't exist or when the files to decrypt aren't supported. These findings highlight the tool's ability to deliver on its core objectives to give users fast, secure, yet user-friendly data protection with helped data prevention from possible threats and vulnerabilities.

**Discussion**

This Encrypted USB File Backup Tool is a solid and valuable tool for several reasons because of its benefits. The dual layer methodology provides high levels of security against unauthorized access and gives AES to protect encrypted file content and RSA to contain essential protection. It has a simplified GUI, which nontechnical users can use to encrypt and decrypt. Additionally, the tool can be packaged into a standalone executable for additional portability distribution and deployment on various systems without requiring complicated installation procedures. All these features make the tool most effective for protecting sensitive data stored in USB drives.

It is a powerful tool but with its limitations. For it to work, the user must securely store the RSA private key, and we cannot decrypt any file if the RSA private key is lost. The tool proved efficient for operating files of up to 285MB but not for files larger than this size, as it may require assistance processing files in environments with fewer system resources. On the other hand, the GUI is straightforward, but technical users may need help to have much more customized ability. The tool's dual encryption approach is far more secure than password-based encryption. However, future improvements could make it more secure and usable, such as multi-factor authentication and cloud backup integration for RSA private keys. Moreover, the GUI design also can be refined to appeal to a more extensive user base.

**Conclusion**

In conclusion, the Encrypted USB File Backup Tool can solve the increasing problem of protecting private data on removable USB devices. The tool utilizes formidable AES encryption for the file content. It bundles it up with the comforting RSA encryption for crucial savings to deliver a dual safety layer for maintaining the confidentiality of files and preventing unauthorized access. Significant achievements include the user-friendly implementation of a GUI, fast encryption and decryption workflows, and support for various file formats and directory processing. As such, this tool is helpful because it offers a practical and portable data security solution to individuals and organizations wanting to do so. The tool promises broader applications such as multi-factor authentication and cloud integration, offering the potential for further enhancements to enable it to address subsequent security demands better.

**References**

Pattanavichai, S. (2022). Program for Simulation and Testing of Apply Cryptography of Advance Encryption Standard (AES) Algorithm with Rivest-Shamir-Adleman (RSA) Algorithm for Good Performance. *International Journal of Electronics and Telecommunications*, 475-481. <https://journals.pan.pl/Content/124255/PDF/3-3548-12075-1-PB.pdf>

Sassani, B. A., Alkorbi, M., Jamil, N., Naeem, M. A., & Mirza, F. (2020). Evaluating encryption algorithms for sensitive data using different storage devices. *Scientific Programming*, *2020*(1), 6132312. <https://doi.org/10.1155/2020/6132312>

Singh, D., Biswal, A. K., Samanta, D., Singh, D., & Lee, H. N. (2022). Juice jacking: security issues and improvements in USB technology. *Sustainability*, *14*(2), 939. <https://doi.org/10.3390/su14020939>

**Appendix**

**Appendix 1: How to execute the app**

To execute the app through the terminal, we use *the Python main.py* command and follow the subsequent prompts, as shown in Figure 5 below. However, before that, we need to ensure that all the required dependencies are installed on our local machine by running the command *pip install -r requirements.txt* on our project root folder.

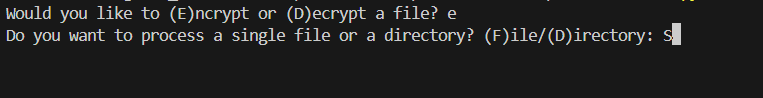


Figure 5 shows our application executing through the terminal.

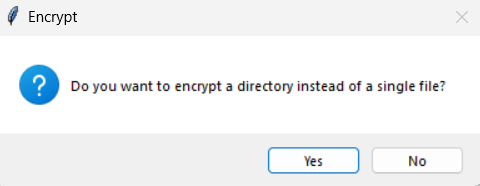
To execute the application through the GUI. There are two methods: one. Calling the GUI directly through the command *python gui.py* is for people who do not want to install the app on their machines for various reasons. Second, calling the build command *python build.py* creates an executable installer for Windows machines, which we can then use to execute the app and pin on our taskbar, as shown in Figure 1.

**Appendix 2: Encryption Flow**

Step 1: Calling the backup file button.



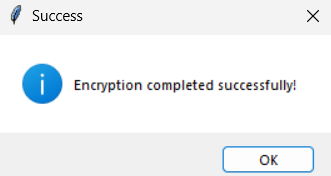
Step 2: Choosing whether to batch process or execute a single file



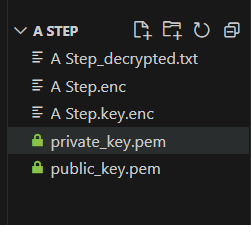
Step 3: Go with a single file and select a preferred one.

Step 4: Choosing the USB device through a drop-down selector.

Step 5: We will be shown a progress bar, and then the last part will be a confirmation message, as shown here.



Here is our folder structure after encryption.



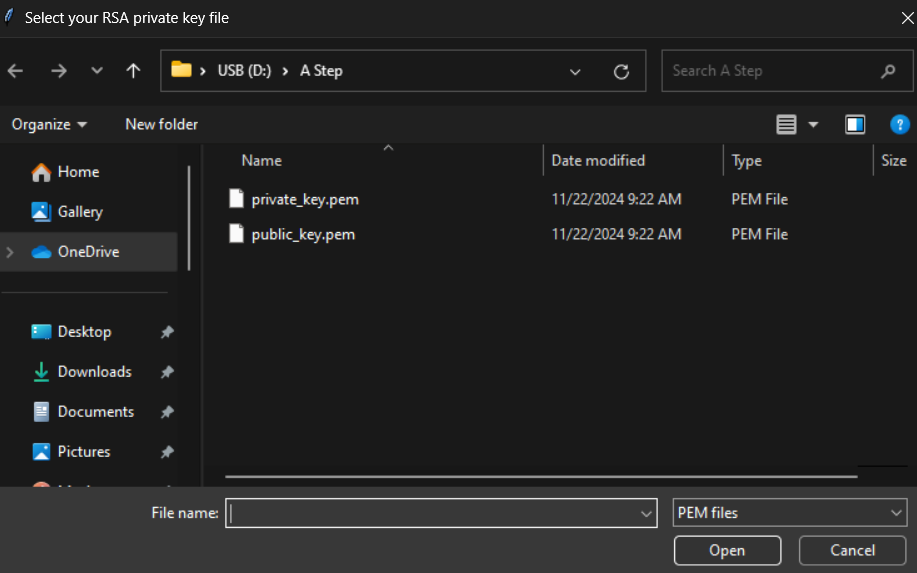
**Appendix 2: Decryption Flow**

Step 1: The decryption process begins by selecting the Restore files button.



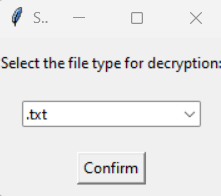
Step 2: In step 2, we select the encrypted file.

Step 3: We select the private key to decrypt the file.

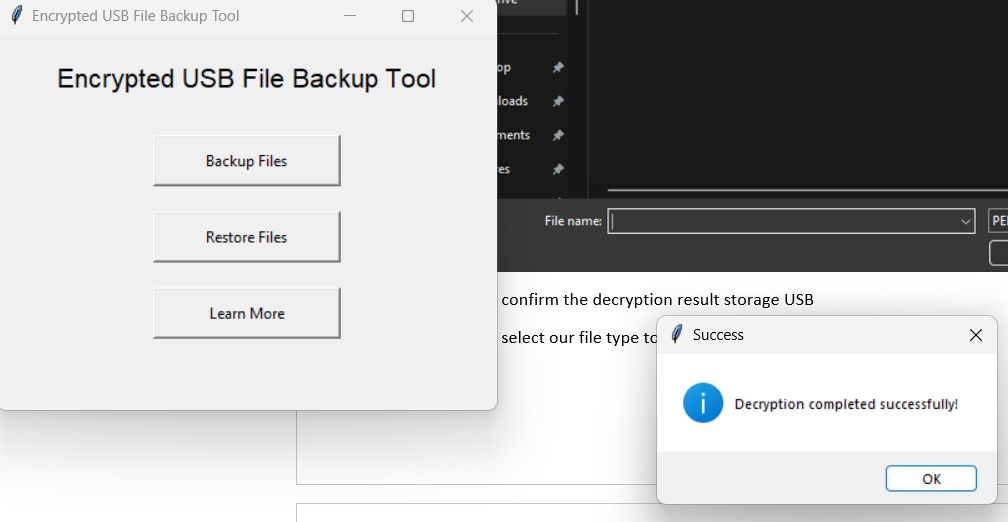


Step 4: We confirm the decryption results in storage USB

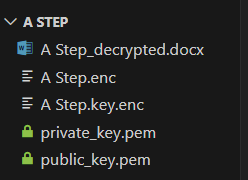
Step 5: We select our file type to restore to the original structure



Step 6: Final decryption confirmation message



Now, here is the Final folder structure.



**Learning webpage**

