Encryption_Decryption Tool

Muhammad Gulraiz Khan
Air University, Islamabad

Abstract
This report presents the design and implementation of two Bash-based tools: a Manual Encryption-Decryption Tool and an Automatic Encryption-Decryption Tool. Developed in a Linux environment, these tools leverage various classical and modern cryptographic ciphers to perform secure data manipulation. The manual tool provides an interactive interface for users to encrypt, decrypt, or gain information about ciphers directly via terminal inputs. In contrast, the automatic tool automates the processing of text files, applying the chosen cipher to produce encrypted or decrypted outputs in a new file. This report details the ciphers used, implementation methodologies, key features, and workflows of both tools, highlighting their significance in practical cybersecurity applications.

Contents

1 Introduction

This technical report outlines the development of two tools—a Manual Encryption-Decryption Tool and an Automatic Encryption-Decryption Tool. Both tools were implemented using Bash scripting in a Linux environment, specifically Kali Linux. These tools demonstrate practical implementations of various cryptographic ciphers and provide functionalities for secure data manipulation.

2 Overview of Ciphers Used

- **Caesar Cipher:** A substitution cipher that shifts characters by a fixed number of places.
- **ROT13:** A variant of the Caesar cipher that rotates characters by 13 positions.

- **Base64:** An encoding method that represents binary data in ASCII text format.
- **Substitution Cipher:** A cipher that replaces plaintext characters with predefined substitutes.
- **Atbash Cipher:** A substitution cipher that reverses the alphabet.
- **Vigenère Cipher:** A polyalphabetic cipher using a key to determine the shift for each character.
- **Hexadecimal Encoding:** Converts binary data to hexadecimal representation.
- **Base32:** Similar to Base64 but uses a 32-character set.
- **Beaufort Cipher:** A cipher based on polyalphabetic substitution.
- **ROT47:** An extension of ROT13 that includes more ASCII characters.

---

3 Manual Encryption-Decryption Tool

3.1 Purpose

The manual tool allows users to interactively encrypt, decrypt, or retrieve information about the supported ciphers. Users input plaintext directly into the terminal and specify the cipher operation.

3.2 Implementation

The manual tool was implemented as a Bash script that:

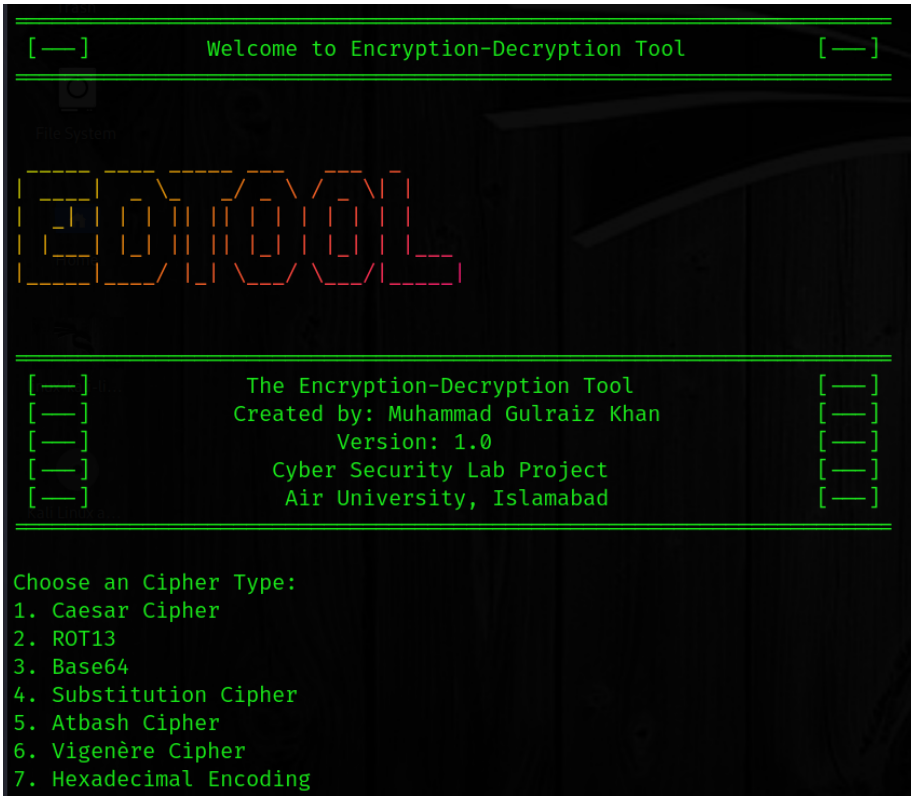
- Displays a menu of options for the user to select:
  - Encrypt text
  - Decrypt text
  - Get information about a cipher
- Accepts user input for the plaintext, cipher type, and operation mode.
- Performs the selected operation using a switch-case structure.
- Outputs the result to the terminal.

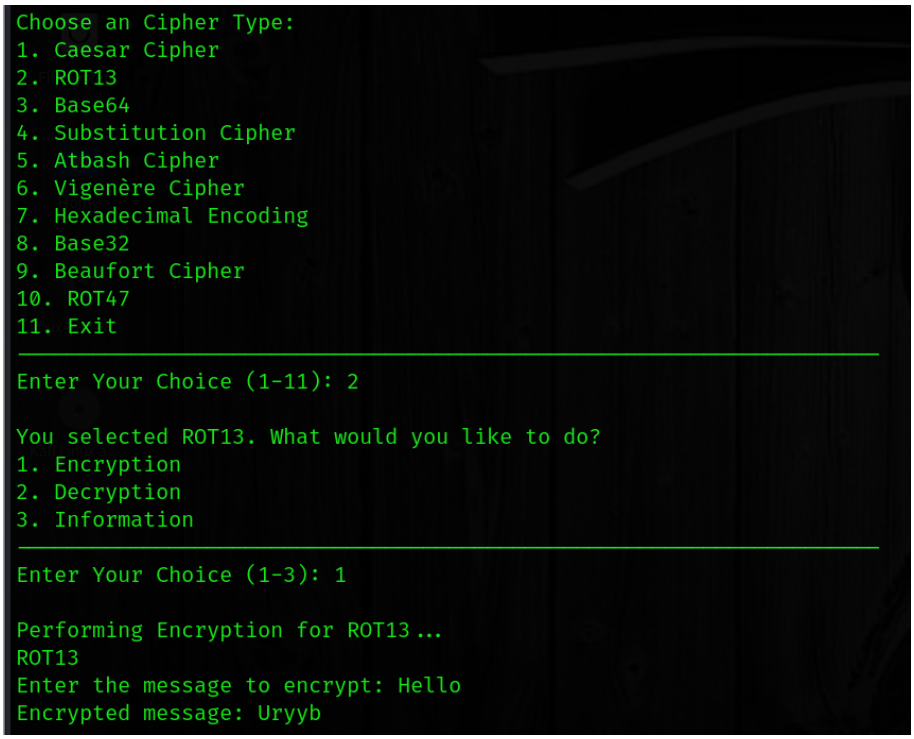
3.3 Key Features

- Interactive and user-friendly interface.
- Detailed feedback for invalid inputs or unsupported operations.
- Modular functions to handle encryption and decryption for each cipher.

3.4 Example Workflow

1. User executes the script.
2. A menu is displayed with available options.
3. User selects the desired operation (e.g., encrypt using Caesar Cipher).
4. The tool prompts for plaintext input and processes it based on the chosen cipher.
5. The output is displayed on the terminal.





4 Automatic Encryption-Decryption Tool

4.1 Purpose

The automatic tool automates the encryption or decryption of text files. Users provide the name of a file containing the data to be processed and specify the cipher type.

4.2 Implementation

The automatic tool was implemented as a Bash script that:

- Accepts the filename as a command-line argument.
- Validates the existence and readability of the file.
- Prompts the user to select a cipher and operation (encryption or decryption).
- Reads data from the specified file.
- Processes the data using the selected cipher.
- Creates a new file in the same directory with the processed output, naming it appropriately (e.g., `output.txt`).

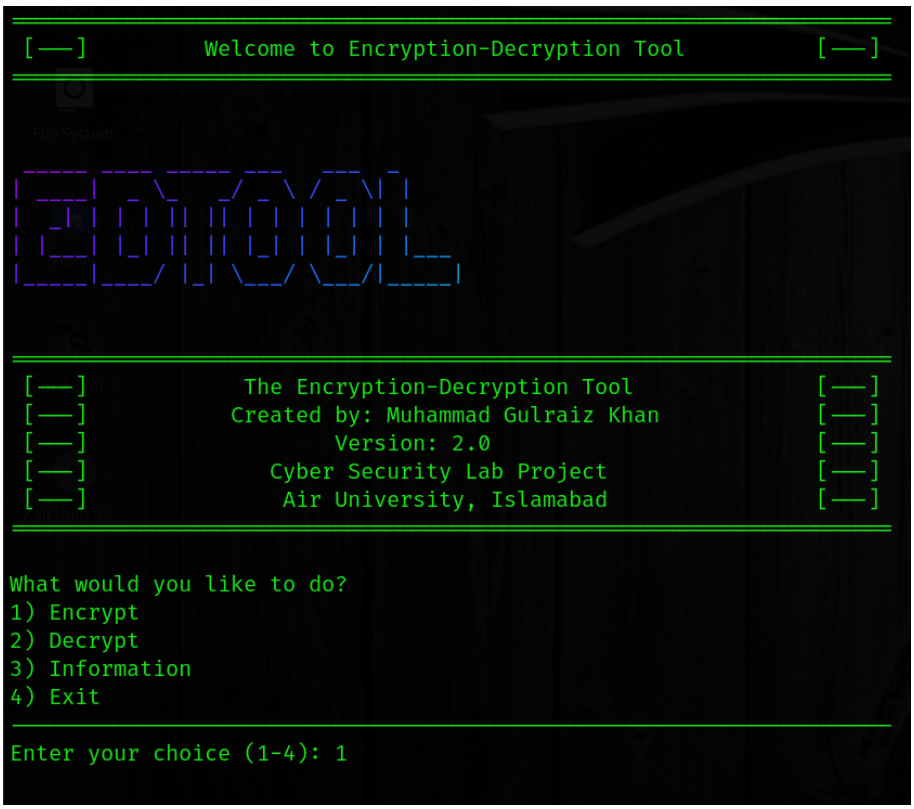
4.3 Key Features

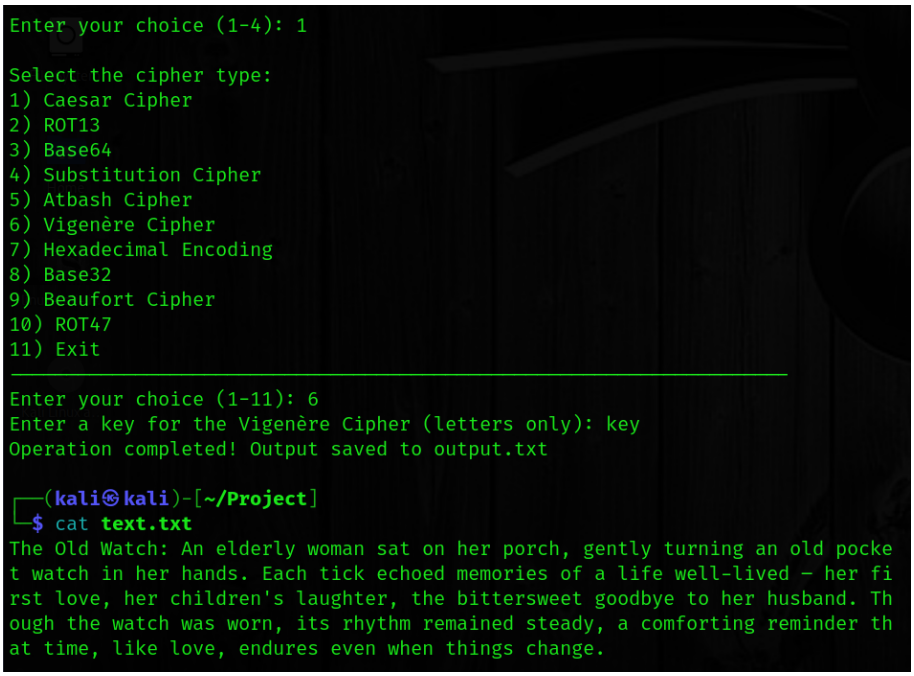
- File-based input and output for batch processing.
- Error handling for file operations (e.g., missing or unreadable files).

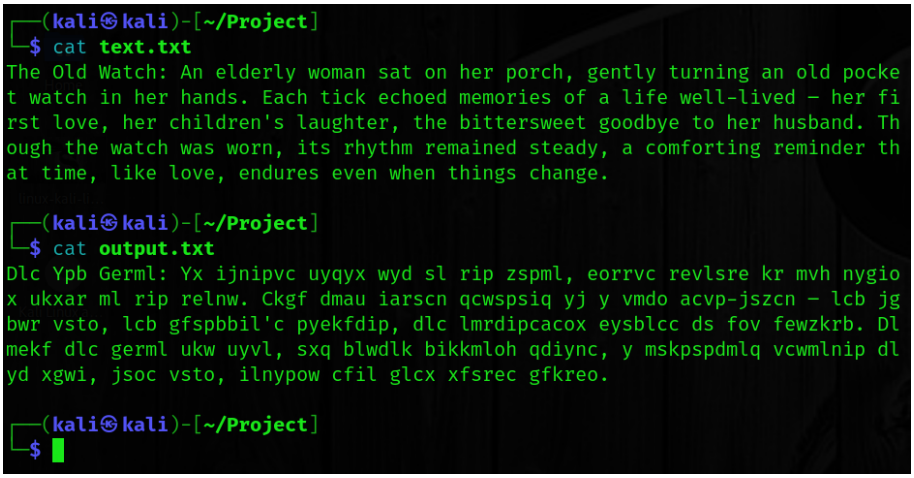
- Automated naming conventions for output files.

4.4 Example Workflow

1. User executes the script with the filename as an argument (e.g., `./EDTauto.sh text.txt`).
2. The script validates the file and displays a menu of cipher options.
3. User selects the cipher and operation (e.g., decrypt using Vigenère Cipher).
4. The tool processes the file content and saves the output to a new file.
5. The script notifies the user of successful operation and the output file location.







5 Challenges and Solutions

- **Input Validation:** Ensuring valid inputs for text and filenames was crucial. Implemented robust error-checking mechanisms.
- **Cipher Implementation:** Translating cryptographic algorithms into Bash commands required careful logic design.
- **File Handling:** Managing file read/write operations and preserving data integrity were addressed through conditional checks and temporary files.

6 Conclusions

The Manual and Automatic Encryption-Decryption Tools showcase the versatility of Bash scripting for cryptographic operations. While the manual tool is suitable for interactive tasks, the automatic tool excels in processing large datasets efficiently. These tools form a robust foundation for practical cybersecurity applications and demonstrate proficiency in implementing encryption techniques using Linux.