### **4.2.1 Installing Cryptsetup and Developing an Encryption Script**

To enable encryption, Cryptsetup was installed using the following command:

|  |
| --- |
| sudo apt-get install cryptsetup |

To simplify the encryption process for users, a script was developed to guide them through setting up encryption on their USB devices. The script included the following steps:

* Checking user privileges and ensuring the script was run with sudo
* Listing available storage devices for user selection.
* Initializing LUKS encryption and prompting the user to create a secure passphrase
* Formatting the encrypted device and providing instructions for mounting and usage

Making the script executable:

|  |
| --- |
| chmod +x encrypt\_flash.sh |

Checking for Mounted Devices: Before running the script, ensure the USB device is unmounted. To check for mount points:

|  |
| --- |
| lsblk -o NAME,SIZE,TYPE,MOUNTPOINT |

If your device is mounted (e.g., /dev/sda1), unmount it:

|  |
| --- |
| sudo umount /dev/sda1 |

Verify that no mount point exists:

|  |
| --- |
| lsblk -o NAME,SIZE,TYPE,MOUNTPOINT |

Running the Script: The script will guide the user through the encryption process, including selecting the device, confirming the choice, and creating a passphrase:

|  |
| --- |
| sudo ./encrypt\_flash.sh |

During execution, the user is prompted to provide a new passphrase, ensuring only authorized users can access the encrypted device.

### **4.2.2 Security and Usability Considerations**

This approach leverages the capabilities of Cryptsetup and the dm-crypt kernel module to provide robust encryption and password-based authentication for USB flash drives. By operating at the block device level, this solution integrates seamlessly with the Linux storage stack and avoids the risks and complexities of directly modifying kernel drivers.

Advantages:

* Compatibility with existing kernel modules.
* Minimal performance overhead.
* User-friendly automation through scripting.

Limitations:

* Reliance on user-space tools rather than kernel-level enforcement.
* Potential for password vulnerabilities, which could be mitigated with stronger authentication methods.

By using Cryptsetup, USB devices can be secured effectively without altering the underlying Linux USB driver, maintaining stability and compatibility while addressing key security concerns.

**4.3 Testing the Modified Driver**

To ensure the functionality and reliability of the encryption setup, rigorous testing was conducted in a controlled virtual environment. The process involved verifying encryption, password authentication, and data integrity using various test scenarios.

### **4.3.1 Test Environment:**

* A virtual machine (VirtualBox) was used for safe testing to avoid potential issues on production systems.
* The test system was equipped with Cryptsetup and dm-crypt modules for encryption and decryption.

### **4.3.2 Test Scenarios and Steps:**

Ensure the Flash Drive Is Not Mounted: Before starting, the encrypted USB flash drive was unmounted and closed to test its behavior when locked.

|  |
| --- |
| sudo umount /mnt sudo cryptsetup close encrypted\_flash |

Attempt to Access the Encrypted Device: An attempt was made to directly list the contents of the encrypted device without unlocking it. This step verified that the encryption was working correctly:

|  |
| --- |
| sudo ls /dev/mapper/encrypted\_flash |

**Expected result:**

|  |
| --- |
| ls: cannot access '/dev/mapper/encrypted\_flash': No such file or directory |

This error message confirmed that the device was securely locked and inaccessible.

**4.3.3 Unlock and Open the Encrypted Device:**

The encrypted flash drive was reopened using the correct password:

|  |
| --- |
| sudo cryptsetup open /dev/sda encrypted\_flash |

**4.3.4 Verify the Device Mapper:**

The mapped encrypted device was verified using the lsblk command:

|  |
| --- |
| lsblk -o NAME,SIZE,TYPE,MOUNTPOINT |

The output displayed the device name, confirming successful unlocking.

**4.3.5 Test Data Integrity:**

To ensure that data could be written and read correctly, test files were created and listed:

|  |
| --- |
| sudo touch /mnt/testfile sudo ls /mnt |

**4.3.6 Expected result:** The file testfile appeared in the directory, demonstrating correct read/write functionality.

**4.3.7 Unmount and Close the Encrypted Device:** After completing the tests, the device was securely unmounted and closed:

|  |
| --- |
| sudo umount /mnt sudo cryptsetup close encrypted\_flash |

# **5. Results**

* Encryption: The encryption functionality worked as intended. Without the correct password, the encrypted device was inaccessible.
* Password Authentication: The system accepted only the correct password, providing secure access to the encrypted device. Incorrect password attempts failed as expected.
* Data Integrity: Files written to the encrypted device were accessible upon unlocking, with no corruption or data loss observed.
* Device Stability: The device remained stable during testing, with no crashes or unexpected behavior during multiple lock/unlock cycles.

By following these test procedures, the functionality and security of the encryption setup were verified. The system demonstrated robust password authentication, seamless data encryption/decryption, and reliable performance in a virtualized testing environment.