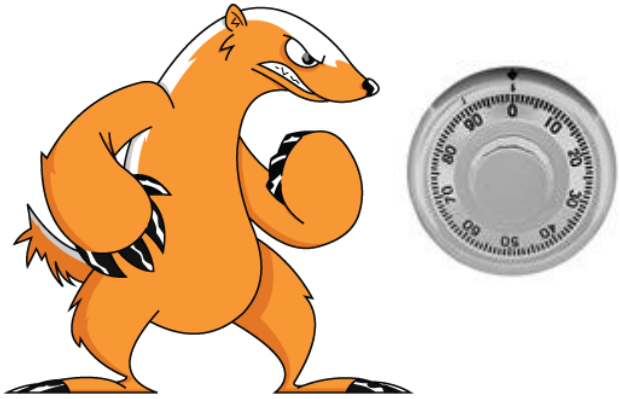
Linux File System Encryption



An Overview of (Some) Current Options

By Gilbert Detillieux, 6 Jan 2026 MUUG Meeting

University of Manitoba

About me…

- First exposed to UNIX at U of M in 1979, worked on it starting in 1980
- Began current job as a SysAdmin in October 1989 (36+ years ago)
- Introduced to Linux around 1992 (Slackware, Red Hat, and later Debian/Ubuntu & Raspberry Pi OS)
- By no means an expert on *crypto*
- Exploring viable options for *Multi-user setups*

Why encrypt files or file systems?

- Protect data from malicious/inquisitive users
- Protect removable media, in case of loss/theft
- Protect internal media in case of loss/theft of entire system! (E.g. laptops)
- Ensure compliance with privacy laws, corporate standards, insurance/audit requirements, etc.

https://www.ssh.com/academy/encryption/what-is-file-encryption
https://www.splunk.com/en_us/blog/learn/data-encryption-methods-types.html
https://ico.org.uk/for-organisations/uk-gdpr-guidance-and-resources/security/encryption/encryption-and-data-protection/

Encryption Types/Levels

- Block/device level:
  - E.g. dm-crypt, LUKS
  - “Full Disk Encryption” (FDE)
- File-system level:
  - E.g. eCryptfs, Fscrypt
  - Allows mix of encrypted/plain-text files on same FS
  - Allows multi-user setups with different keys
- Individual file level
  - E.g. PGP, GPG
- In-transit encryption
  - E.g. SSL/TLS
- Can be used in combination

https://en.wikipedia.org/wiki/Disk_encryption
https://docs.fedoraproject.org/en-US/quick-docs/encrypting-drives-using-LUKS/
https://en.wikipedia.org/wiki/Filesystem-level_encryption

Linux Unified Key Setup (LUKS)

- Works at the block device level
  - Can use any file system type, even swap partitions
- Superset/replacement of dm-crypt
  - Can be combined with LVM, MD
- Two versions:
  - LUKS1 allows up to 8 “user” encryption keys
  - LUKS2 allows 32 keys, JSON metadata format
- Full disk encryption
  - Can even have /boot encrypted; LUKS support in GRUB
  - Make sure installer supports it … *correctly!*

https://en.wikipedia.org/wiki/Linux_Unified_Key_Setup
https://docs.fedoraproject.org/en-US/quick-docs/encrypting-drives-using-LUKS/
https://docs.redhat.com/en/documentation/red_hat_enterprise_linux/9/html/security_hardening/encrypting-block-devices-using-luks_security-hardening

LUKS Setup *(redundant?)*

- **apt install cryptsetup** *or* **dnf install cryptsetup-luks**
- **cryptsetup luksFormat /dev/sdb1** *(or other partition)*
- **cryptsetup open /dev/sdb1** *my_fs*
  - *Enter passphrase for /dev/sdb1:*
- **mkfs.ext4 /dev/mapper/***my_fs*
- **mkdir /mnt/***my_mnt*
- **mount /dev/mapper/***my_fs* **/mnt/***my_mnt*
- *To automate, edit /etc/fstab and /etc/crypttab*
- *Note that /dev/mapper/my_fs is decrypted device*
- *Mounted file system is also decrypted*

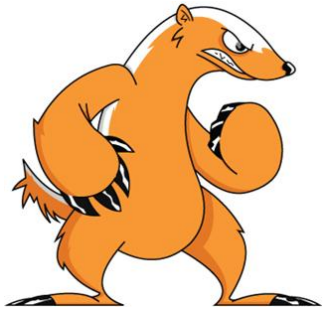
https://reintech.io/blog/configuring-luks-encrypted-disk-debian-12
https://www.cyberciti.biz/security/howto-linux-hard-disk-encryption-with-luks-cryptsetup-command/
https://gist.github.com/superjamie/d56d8bc3c9261ad603194726e3fef50f (LUKS on LVM)
https://www.itstorage.net/index.php/ldce/islme/572-instructions-for-auto-mounting-a-luks-encrypted-device-on-debian-12

LUKS Notes and Cautions:

- Once opened/mounted, device/files are…
  - accessible as unencrypted data
  - still protected like any other clear-text device/files
  - possibly visible to other users (and definitely root)
- Suspend/sleep leaves data accessible
  - Can still hibernate safely
  - Can use **cryptsetup luksSuspend**
- Unencrypted swap can still leak data!
- May want to overwrite entire device before **mkfs**
  - zeros on mapped dev look random on raw dev
- In theory, can **umount** and **cryptsetup close** *dev*
  - Can be locked by non-obvious processes or kernel itself

https://security.stackexchange.com/questions/49074/erasing-encryption-luks-keys-before-suspend-to-ram
https://superuser.com/questions/648333/how-to-make-suspend-to-ram-secure-on-ubuntu-with-full-disk-encryption-lvm-on-to
https://waaaaargh.github.io/gnu&linux/2013/08/06/lukssuspend-with-encrypted-root-on-archlinux/

eCryptfs



- Works at the file system level
  - “Stacked” cryptographic file system
  - Can use any underlying file system type
- Derived from Erez Zadok’s Cryptfs (1998)
- Now considered deprecated
  - Some unresolved (minor?) security issues
  - Stale codebase for kernel (2020) and utilities (2017)
- Encryption of individual directories/files
  - Per-user keys
  - Handy utility to migrate user’s home directory
  - New mount point for each encrypted top-level directory

https://en.wikipedia.org/wiki/ECryptfs
https://www.ecryptfs.org/
https://launchpad.net/ecryptfs/
https://wiki.archlinux.org/title/ECryptfs

eCryptfs Setup

- **apt install ecryptfs-utils** *or* **dnf install ecryptfs-utils**
  - *Kernel also needs to have driver support compiled in*
  - *Support being dropped in some distros*
- **modprobe ecryptfs** *(on initial setup only?)*
- **ecryptfs-migrate-home -u** *username*
  - *Enter passphrase, cipher, etc.*
- **Manual setup:**
  - **mkdir ~/***my_data*
  - **sudo mount –t ecryptfs ~/***my_data* **~/***my_data*

https://askubuntu.com/questions/1335006/what-is-the-recommended-method-to-encrypt-the-home-directory-in-ubuntu-21-04
https://sysphere.org/~anrxc/j/articles/ecryptfs/index.html
https://gist.github.com/drmalex07/097963cc89a0e9cf90e8

eCryptfs Notes and Cautions:

- Once mounted, directories/files are…
  - accessible as unencrypted data
  - still protected like any other clear-text device/files
  - possibly visible to other users (and definitely root)
- Should **umount** after use
  - Hopefully done automatically on logout
  - Can be locked by non-obvious processes or kernel itself
- File names & content encrypted, but ***not*** metadata
- Unencrypted swap can still leak data!

https://www.systutorials.com/setting-up-ecryptfs-in-linux/
https://wiki.gentoo.org/wiki/Encrypt_a_home_directory_with_ECryptfs
https://ostechnix.com/how-to-encrypt-directories-with-ecryptfs-in-linux/
https://dataswamp.org/~solene/2023-03-12-encrypt-with-ecryptfs.html

Fscrypt

- Works at the file system level
  - Integrated into underlying file system (more efficient)
  - Support of ext4, f2fs, UBIFS, CephFS & Lustre only
- Encryption of individual directories/files
  - Per-user keys
  - Migration of existing directories done manually
  - No new mount points for each directory

https://github.com/google/fscrypt
https://wiki.archlinux.org/title/Fscrypt
https://forums.linuxmint.com/viewtopic.php?t=378514

Fscrypt Setup

- **apt install fscrypt libpam-fscrypt**
  or **dnf install fscrypt pam_fscrypt**
  - *Kernel also needs to have fscrypt support compiled in*
- **fscrypt setup** *(initial setup only)*
- **mkfs.ext4 -O encrypt /dev/***device_part* *(for new FS)*
- **tune2fs -O encrypt /dev/***device_part* *(for existing FS)*
- **fscrypt setup /***mount_point* *(for each mounted file system)*
- **Manual setup (per user/top-level directory):**
  - **mv /home/***username***{,.old}**
  - **mkdir /home/***username*
  - **fscrypt encrypt /home/***username* **--user=***username*
  - **rsync -Hav --info={progress2,name0} /home/***username***{.old,}/.**

https://forums.linuxmint.com/viewtopic.php?t=378514
https://wiki.archlinux.org/title/Fscrypt
https://www.kernel.org/doc/html/v5.0/filesystems/fscrypt.html
https://gist.github.com/plembo/bf3343a6f387251c501b031f43c919a7

University of Manitoba

Fscrypt Notes and Cautions:

- Once decrypted, directories/files are…
  - accessible as unencrypted data
  - still protected like any other clear-text device/files
  - possibly visible to other users (and definitely root) if still in disk cache
- No specific mounts/unmounts per-user
  - pam_fscrypt.so automatically unlocks/locks directories
  - Can be manually un/locked by root
- File names & content encrypted, but ***not*** metadata
- Unencrypted swap can still leak data!

Which One Wins?

- LUKS is best for …
  - Full disk encryption
  - Single-user systems, especially laptops
  - Removable media
  - When performance matters
- Fscrypt is best for …
  - Multi-user home directory FS, if supported
  - Servers with multiple users/clients
- eCryptfs is fine for …
  - Use cases where above aren't viable options
  - Users want some privacy, but data not extremely sensitive (low risk)
  - Physical media is otherwise safe

https://www.phoronix.com/review/ext4-crypto-418
https://wiki.archlinux.org/title/Data-at-rest_encryption
https://www.sciencedirect.com/science/article/pii/S2666281723001816

Questions?

University of Manitoba