dirlock: a tool to manage encrypted filesystems

Alberto Garcia

Open Source Summit Europe 2025

Amsterdam

About me

- Software engineer at Igalia.
  - GNOME
  - Maemo / MeeGo
  - QEMU
- Debian developer.
- Currently working on SteamOS.

dirlock

A tool to manage disk encryption

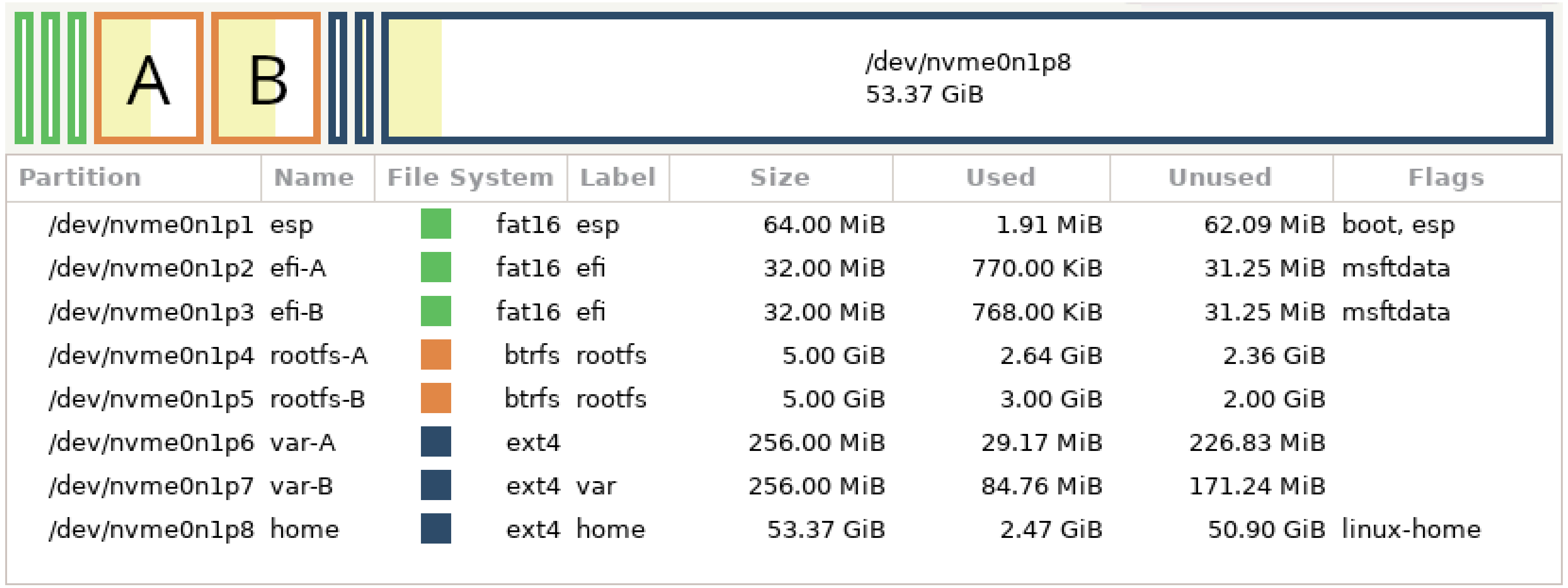
Not a new encryption system

Built on top of existing technologies

Use case: encryption for SteamOS

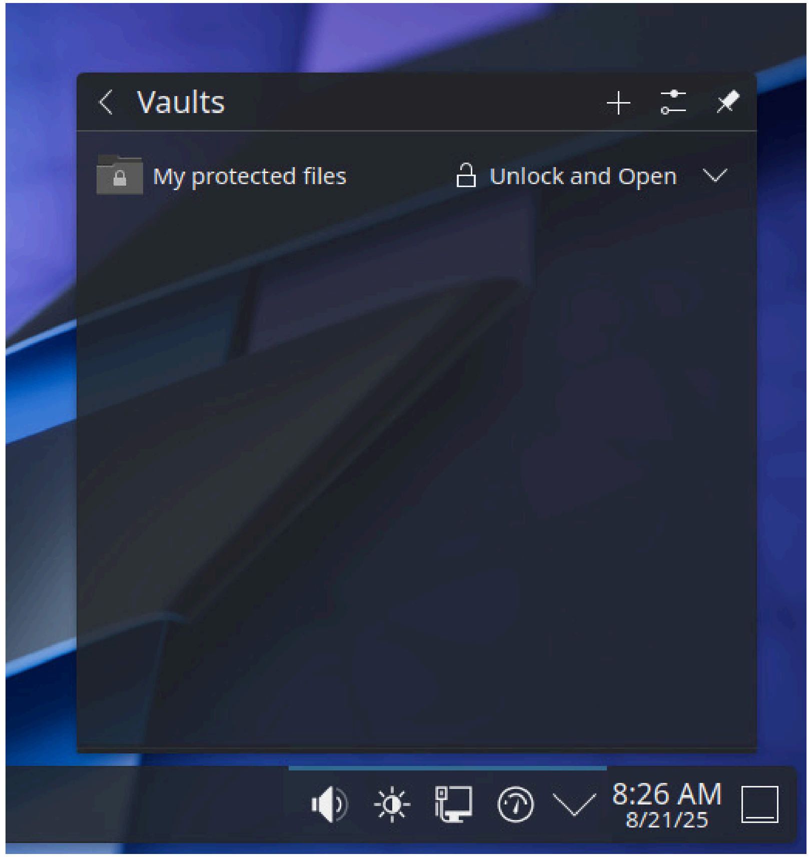
- Steam Deck and others: portable devices, easy to lose.
- No encryption at the moment.
- Not only usable for gaming.
- Anyone can read the contents of the hard drive.

Disk layout



Partition	Name	File System	Label	Size	Used	Unused	Flags
/dev/nvme0n1p1	esp	🟩 fat16	esp	64.00 MiB	1.91 MiB	62.09 MiB	boot, esp
/dev/nvme0n1p2	efi-A	🟩 fat16	efi	32.00 MiB	770.00 KiB	31.25 MiB	msftdata
/dev/nvme0n1p3	efi-B	🟩 fat16	efi	32.00 MiB	768.00 KiB	31.25 MiB	msftdata
/dev/nvme0n1p4	rootfs-A	🟧 btrfs	rootfs	5.00 GiB	2.64 GiB	2.36 GiB	
/dev/nvme0n1p5	rootfs-B	🟧 btrfs	rootfs	5.00 GiB	3.00 GiB	2.00 GiB	
/dev/nvme0n1p6	var-A	🟦 ext4		256.00 MiB	29.17 MiB	226.83 MiB	
/dev/nvme0n1p7	var-B	🟦 ext4	var	256.00 MiB	84.76 MiB	171.24 MiB	
/dev/nvme0n1p8	home	🟦 ext4	home	53.37 GiB	2.47 GiB	50.90 GiB	linux-home

Current alternative: Plasma Vaults



Our goals

- **Personal files must be unreadable if the computer is stolen.**
- `$HOME` should be be encrypted.
  - Possibility to encrypt other directories.
- Multiple users with independent encryption keys.
- Access should be authenticated.
  - PIN, password, or similar to log in.
  - Not all computers have a keyboard!
  - Support hardware-backed mechanisms.

- Enable encryption without reinstalling the OS from scratch.
  - Ideally: a simple *"Encrypt data"* button or command.
- D-Bus API.
- Reasonable performace.

Available encryption technologies

- Stacked filesystem encryption
- Block device encryption
- Native filesytem encryption

Stacked filesytem encryption

- Data is stored as (encrypted) regular files.
- Mount the encrypted directory to see the data.
- Examples: gocryptfs, EncFS
  - Implemented in user space (FUSE).
  - Used by tools like Plasma Vaults.

Block device encryption

- Encrypts blocks on disk, does not care about what's inside. (normally a filesystem but it can be anything).
- Uses raw partitions or loopback files.
- The contents are completely hidden.
- Most popular technology: LUKS.
    - The header contains the encryption keys.

Native filesystem encryption

- Files are encrypted directly at the filesystem level.
- A filesystem can contain a mix of encrypted and unencrypted directories.
- Only partial confidentiality:
  - Data is safe, but metadata, file sizes, … are not protected.
- The Linux kernel provides the *fscrypt* API:
  - Implemented by ext4, f2fs and others.
- User space is responsible for the encryption keys.

LUKS vs fscrypt

LUKS: pros and cons

- Pros:
  - Maximum confidentiality and protection.
  - Supports TPM, FIDO tokens (via systemd).
- Cons:
  - Usually unlocked early on boot.
  - No fine-grained control about what to encrypt.
  - Hard to encrypt an existing installation, it needs a new filesystem.

fscrypt: pros and cons

- Pros:
  - Easy to encrypt an existing installation, no preallocation needed.
  - Multiple directories and user accounts with different keys.
  - Easy integration with PAM.
  - Can be unlocked after booting, also remotely (ssh).
- Cons:
  - Metadata not encrypted, some information can be seen or guessed.
    - Approximate directory structure, sizes, permissions, timestamps, extended attributes.
  - Attackers can delete files.

Our choice is fscrypt

- Good confidentiality guarantees for the main use case.
- Flexible.
- It can be enabled in existing installations.
- Good performance.

But fscrypt is just a kernel API

- We need to handle the encryption keys in user space.
- Existing tools:
    - The *fscrypt* command-line application.
        - Related to, but different from the fscrypt kernel API.
    - systemd-homed

/usr/bin/fscrypt

- Reference tool to manage encrypted directories.
- Written in Go by Joe Richey and Eric Biggers.
- Simple to use, covers all essential functionalities.
- PAM support.
- Only allows passwords and raw binary keys.
  - No hardware-backed mechanisms.
- No D-Bus API.

systemd-homed

- A tool to manage *human* user accounts.
- Various storage backends, two of them encrypted:
  - LUKS (homedir inside a LUKS loopback file).
  - fscrypt (only the deprecated v1 API).
- D-Bus API, PAM and FIDO support (but no TPM).
- However: it's primarily *not* an encryption tool.
  - It encrypts `$HOME` and nothing else.
  - Own user database (no `/etc/passwd`).
  - Uses idmapped mounts, issues with podman.

dirlock

A new high-level tool that uses the fscrypt API

Overview

- Does encryption, authentication and nothing else.
- Heavily inspired by /usr/bin/fscrypt.
- Still under development.
  - PAM support working.
  - FIDO support working.
  - (basic) TPM support working.
  - D-Bus API in prototype stage.

Where to find it

- **https://gitlab.steamos.cloud/holo/dirlock/**
- Free software, BSD license.
- Written in Rust.
- Works in any Linux system.
- Available in SteamOS 3.8 as an *experimental* feature.

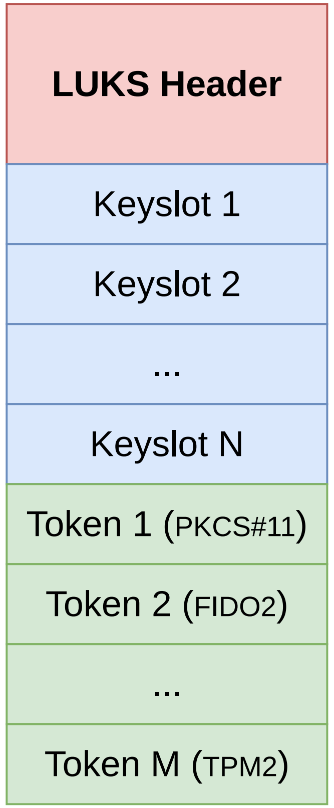
Basic architecture

Encryption policies and master keys

- An encrypted directory has an **encryption policy** (master key and various parameters).
- The master key is loaded into the kernel to *unlock* a directory and removed from the kernel to *lock* it.
- User space (e.g. `dirlock`) must manage the master key and keep it safe.

Protectors

- The master key is not used directly.
- Wrapped with intermediate keys called *protectors*.
- Different types of protectors (password, FIDO2, ...).
- Compromised protectors can be deleted without exposing the master key.
- Design taken from /usr/bin/fscrypt. Similar idea used in LUKS or BitLocker.



LUKS Header

Keyslot 1

Keyslot 2

...

Keyslot N

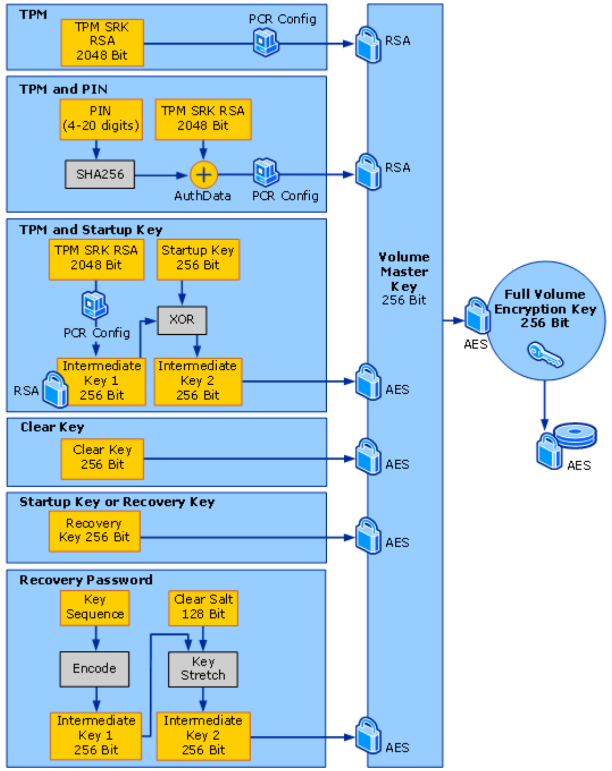
Token 1 (PKCS#11)

Token 2 (FIDO2)

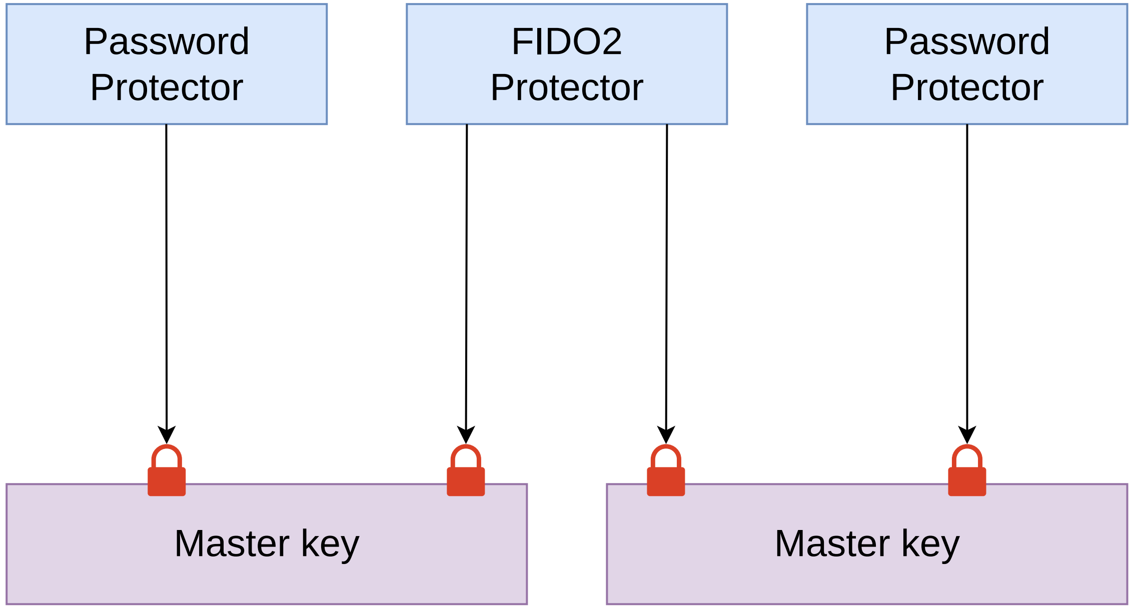
...

Token M (TPM2)

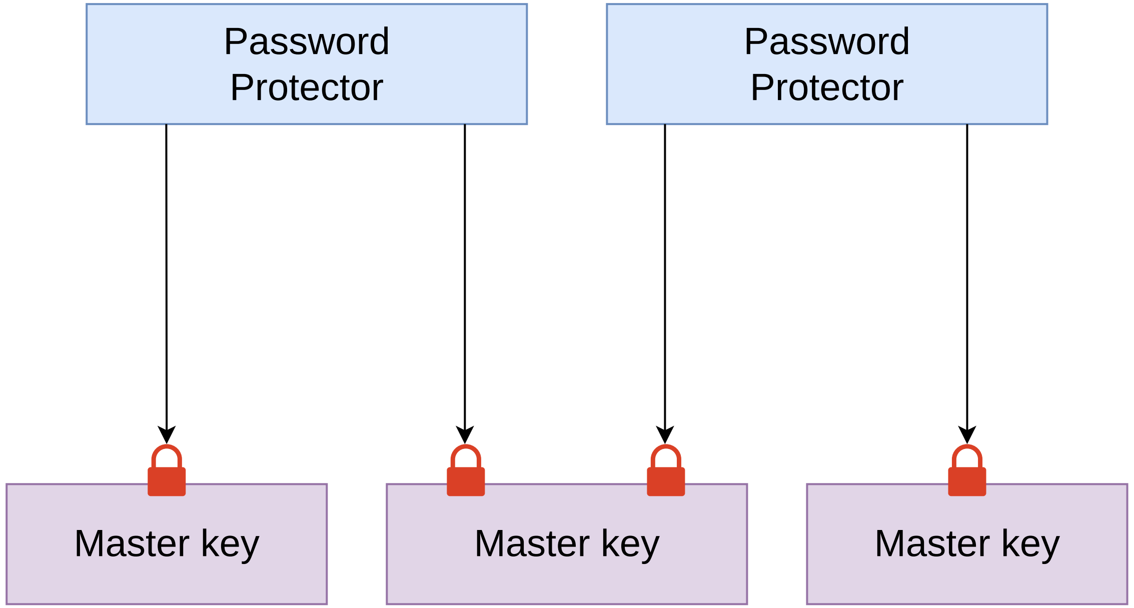
bitlocker (image source)



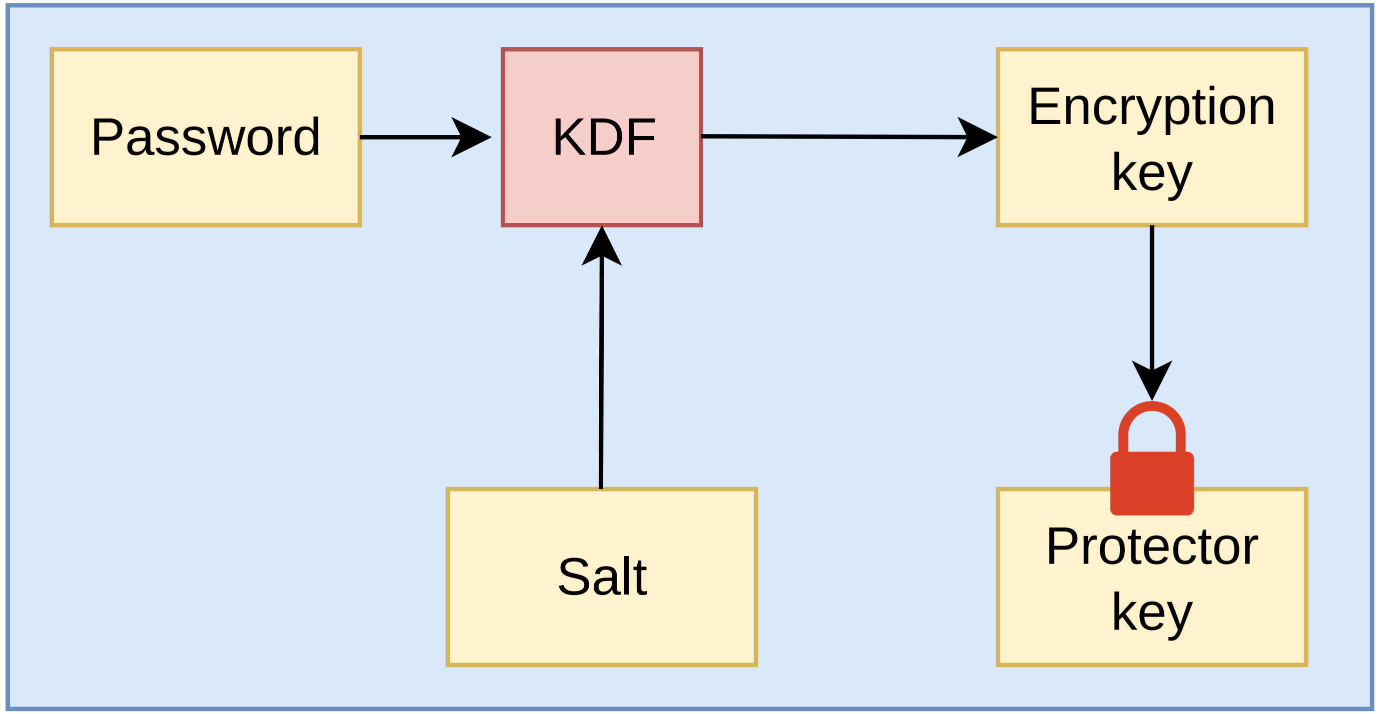
dirlock


Password Protector
FIDO2 Protector
Password Protector
Master key
Master key

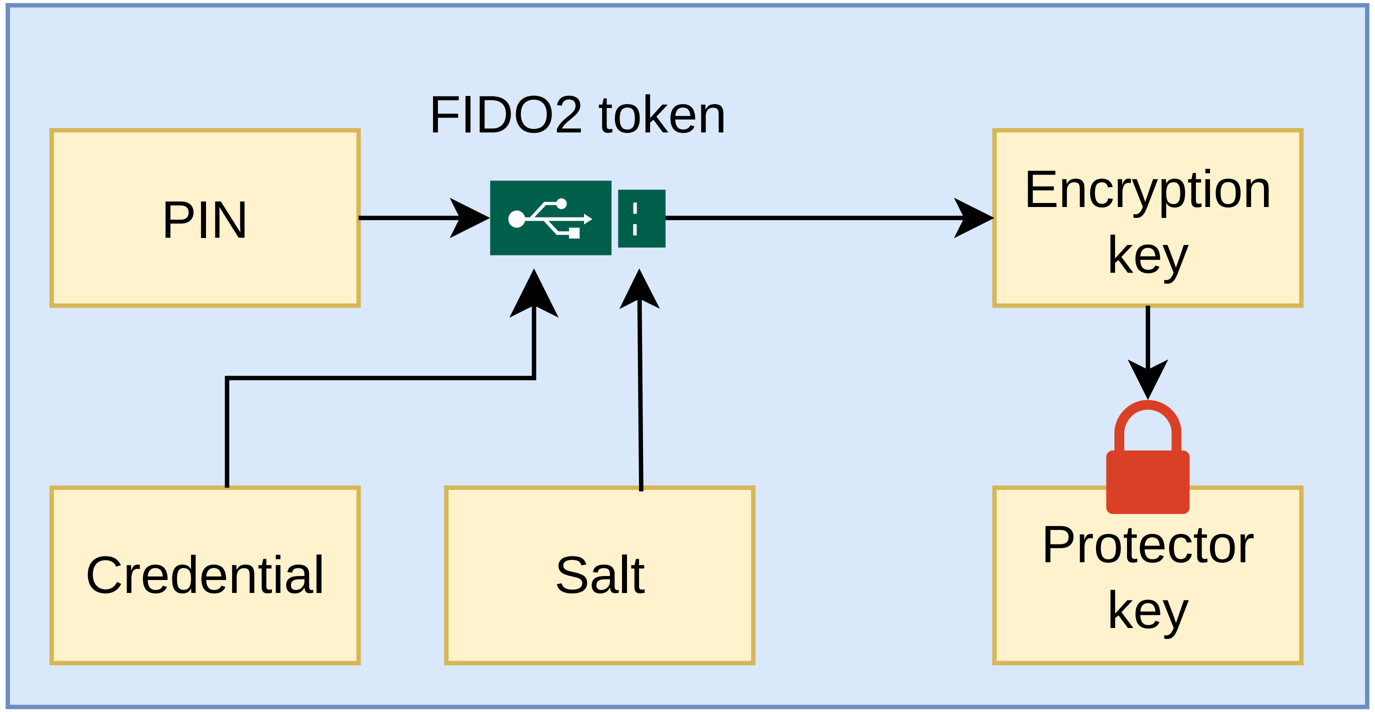
dirlock


Password Protector
Password Protector
Master key
Master key
Master key

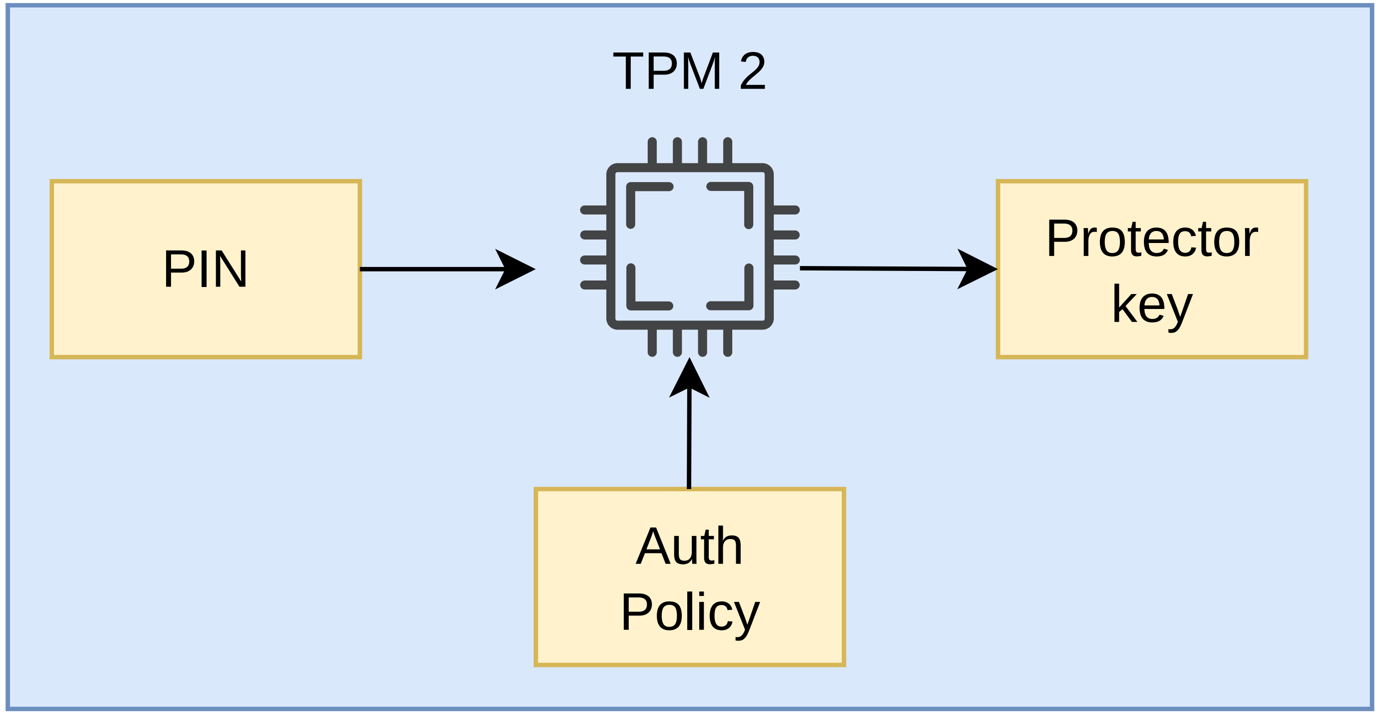
Password protector



FIDO2 protector



TPM2 protector



dirlock: basic commands

- `encrypt`: enable encryption on a directory.
  - This creates a new master key and encryption policy.
- `lock`: lock an encrypted directory.
- `unlock`: unlock an encrypted directory.
- `protector create`: create a new protector.
- `protector remove`: remove an existing protector.
- `protector change-password`: change a protector's password.
- `policy add-protector`: add a protector to an encryption policy.
- `policy remove-protector`: remove a protector from an encryption policy.

PAM integration

- PAM module available: `pam_dirlock.so`.
- No need to convert users:
  - Home directory encrypted? ⇒ handled by dirlock.
  - Otherwise ⇒ `PAM_USER_UNKNOWN` ⇒ next module.

PAM configuration

```
auth    [success=3 user_unknown=ignore default=die] pam_dirlock.so
auth    [success=2 default=ignore]  pam_systemd_home.so
auth    [success=1 default=ignore]  pam_unix.so nullok try_first_pass
auth    requisite           pam_deny.so
auth    required            pam_permit.so

session optional    pam_dirlock.so
session required    pam_unix.so

password    [success=3 user_unknown=ignore default=die] pam_dirlock.so
password    [success=2 default=ignore]  pam_systemd_home.so
password    [success=1 default=ignore]  pam_unix.so obscure yescrypt
password    requisite           pam_deny.so
password    required            pam_permit.so
```

Demo

Thanks!

igalia