Файловые системы и шифрование

1. Введение

В современном мире утечки данных происходят не только через сеть, но и физически — **украденный ноутбук, утерянный накопитель или скомпрометированный бэкап** могут привести к серьёзным последствиям. Именно поэтому **шифрование файловой системы** стало стандартной практикой в защищённых и пользовательских системах.

🔐 Почему важно шифрование?

- Защищает данные при краже физического диска или устройства.
- Предотвращает несанкционированный доступ к бэкапам и образам дисков.
- Даже root-пользователь на другой системе не сможет получить доступ без ключа.

⚙️ Уровни шифрования

Существуют два основных подхода к шифрованию данных в Linux:

- **Блочное шифрование** — шифруется весь раздел или устройство, независимо от файловой системы:
  - `LUKS` — стандарт блочного шифрования;
  - `dm-crypt` — механизм в ядре Linux для реализации LUKS.
- **Файловое шифрование** — шифруются отдельные каталоги и файлы на уровне файловой системы:
  - `fscrypt` — встроенное в ядро решение для ext4, F2FS, Btrfs.

💡 **Разница:** LUKS защищает весь диск (включая структуру ФС), а fscrypt — только содержимое конкретных каталогов.

🔑 Где используется шифрование

- 🖥️ **Десктопы и ноутбуки** — защита пользовательских данных и /home.
- 📱 **Мобильные устройства** — Android активно использует fscrypt.

- 🛡️ **Защищённые операционные системы** — Astra Linux, НАЙС.ОС и др.
- 🏢 **Серверы и облака** — защита бэкапов, образов, tenant-данных.

🔐 **Шифрование — это не опция, а необходимость** при работе с конфиденциальной информацией.

2. Обзор уровней шифрования

В Linux существует два принципиально разных подхода к шифрованию данных — **на уровне блочного устройства** и **на уровне файловой системы**. Каждый из них решает разные задачи и имеет свои плюсы и минусы.

🔒 Шифрование блочного устройства (LUKS / dm-crypt)

- Осуществляется **до загрузки файловой системы** — всё, включая метаданные, inode, имена файлов, скрыто от системы без ключа.
- Использует `dm-crypt` — подсистему ядра Linux для прозрачного шифрования блочных устройств.
- Наиболее распространённая реализация — `LUKS` (Linux Unified Key Setup).
- Позволяет хранить несколько ключей, защищает swap, tmp, boot (частично), root.

✅ **Плюсы:** надёжность, полная защита всех данных, прозрачность для ФС.
⚠️ **Минусы:** невозможность гибкого разграничения шифрования, сложнее интегрировать с пользователями и сессиями.

🔐 Шифрование на уровне файловой системы (fscrypt)

- Поддерживается в **ext4**, **F2FS**, **Btrfs** — начиная с ядра 4.1+.
- Позволяет шифровать **отдельные каталоги**, а не весь диск.
- Привязка к учётным записям, возможность автоматической расшифровки при входе в систему (через PAM).
- Каталоги выглядят незашифрованными только для владельца с корректным ключом.

✅ **Плюсы:** гибкость, простота настройки, совместимость с многопользовательскими системами.
⚠️ **Минусы:** не шифрует структуру ФС и имена файлов, не защищает swap.

📊 Сравнение подходов

Критерий	Блочное шифрование (LUKS/dm-crypt)	Файловое шифрование (fscrypt)
Что шифруется	Всё (данные, метаданные, имена)	Только содержимое каталогов
Гибкость	✕ (один ключ на всё)	✓ (разные ключи на каталоги)
Производительность	Высокая, стабильная	Зависит от реализации ФС
Совместимость	С любой ФС	Только ext4, F2FS, Btrfs
Уровень защиты	Максимальный	Средний (метаданные доступны)

Вывод: если нужно зашифровать всё — используйте LUKS.
Если достаточно защиты определённых каталогов — fscrypt даёт больше гибкости.

3. `LUKS` + `dm-crypt`

LUKS (Linux Unified Key Setup) — это стандарт шифрования блочных устройств в Linux. Он работает поверх `dm-crypt` — низкоуровневой подсистемы ядра для прозрачного шифрования блочных устройств.

🔐 Как это работает

- 📦 Шифруется весь блочный раздел — включая метаданные, inode, структуру директорий.
- 🔑 Доступ осуществляется по ключу (паролю) или ключевому файлу.
- Поддерживается до **8 ключей** на одном устройстве (можно добавлять и удалять).
- После расшифровки создаётся виртуальное устройство (`/dev/mapper/...`), с которым уже работает файловая система.

🛠️ Основные команды `cryptsetup`

1. Инициализация раздела:

```
sudo cryptsetup luksFormat /dev/sdX1
```

2. Открытие раздела:

```
sudo cryptsetup luksOpen /dev/sdX1 secure_volume
```

После открытия доступ будет через `/dev/mapper/secure_volume`

3. Добавление нового ключа:

```
sudo cryptsetup luksAddKey /dev/sdX1
```

4. Просмотр информации о LUKS-контейнере:

```
sudo cryptsetup luksDump /dev/sdX1
```

💡 Интеграция с загрузкой

- Поддерживается в **initramfs** — система запрашивает пароль при загрузке.
- GRUB умеет работать с LUKS1 (но не всегда с LUKS2).
- Настройка через `/etc/crypttab` и `/etc/fstab`.

📦 Примеры использования с файловыми системами

После открытия LUKS-контейнера можно форматировать его как обычный раздел:

```
sudo mkfs.ext4 /dev/mapper/secure_volume
```

```
sudo mkfs.xfs /dev/mapper/secure_volume
```

```
sudo mkfs.btrfs /dev/mapper/secure_volume
```

Монтирование:

```
sudo mount /dev/mapper/secure_volume /mnt
```

✅ **Преимущество LUKS:** совместимость с любой ФС, полная защита данных, включая

swap, tmp, root.
⚠️ **Важно:** если вы потеряете ключ или заголовок LUKS, восстановление будет невозможно. Делайте резервную копию заголовка:

```
cryptsetup luksHeaderBackup /dev/sdX1 --header-backup-file luks-header.img
```

4. Файловое шифрование с `fscrypt`

fscrypt — это современный интерфейс ядра Linux для **встроенного шифрования на уровне файловой системы**. Он реализует шифрование на уровне каталогов и файлов, не требуя отдельного раздела или блочного устройства.

🔍 Что такое `fscrypt`

- ⛓️ Это API в ядре Linux (начиная с 4.1, активно развивается с 5.x);
- 🔐 Поддерживается в `ext4`, `F2FS`, `Btrfs` (экспериментально);
- 📂 Позволяет шифровать **отдельные каталоги** с помощью пользовательских ключей или паролей;
- 👥 Интегрируется с PAM и `loginctl` — автоматически расшифровывает каталог при входе пользователя.

💡 **Разница с LUKS:** fscrypt не шифрует весь диск — только указанные каталоги. Подходит для сценариев «многопользовательской» защиты или защиты /home.

✨ Особенности и преимущества

- ✅ Не требует переформатирования раздела — работает на существующей ФС;
- ✅ Можно шифровать только нужные каталоги (`~/Secrets`, `/srv/private` и т.д.);
- 🔑 Ключи хранятся в системном `keyring`, защищены паролем;
- 🎁 Поддерживает аппаратное ускорение (если доступно);
- 📁🔒 Возможность шифровать имена файлов и подкаталоги.

🛠️ Примеры использования

1. Инициализация поддержки шифрования:

```
sudo fscrypt setup /mnt
```

2. Шифрование каталога:

```
sudo fscrypt encrypt /mnt/secure-folder
```

Появится меню выбора: по паролю, пользовательскому ключу и т.д.

3. Проверка статуса:

```
fscrypt status /mnt/secure-folder
```

🔋 Где используется `fscrypt`

- **Android** — основное решение для защиты /data на современных устройствах;
- **Дистрибутивы Linux** — Ubuntu, Fedora, Debian с поддержкой PAM+fscrypt;
- **Встраиваемые устройства** — где невозможен LUKS, но нужна защита данных.

⚠️ **Важно:** имя каталога, владелец и часть структуры ФС остаются видимыми без ключа. Только содержимое файлов и подкаталогов — шифруется.
✅ **fscrypt** — это современное решение, позволяющее гибко шифровать данные с минимальными накладными расходами, не нарушая структуру системы.

5. Сравнение: `LUKS` vs `fscrypt`

Оба подхода — LUKS и fscrypt — применимы в зависимости от задач. Ниже — краткое сравнение по ключевым параметрам:

Критерий	**LUKS** **(dm-crypt)**	**fscrypt**
Уровень	Блочный	Файловый
Шифрует всё	✅	❌ (только каталоги)
Скрытие структуры ФС	✅	❌
Гибкость	🔶 (один ключ на раздел)	✅ (разные ключи на каталоги)
Совместимость	✅ со всеми файловыми системами	✅ только с поддержкой fscrypt (ext4, F2FS, Btrfs)

Критерий	LUKS (dm-crypt)	fscrypt
Производительность	✅ высокая, аппаратное ускорение	✅, зависит от реализации ФС
Подходит для `/` (root)	✅ шифрует весь диск	❌ обычно используется для `/home`, `/data`

💡 **Вывод:**

- **LUKS** подходит, если требуется полная защита всего диска — от swap до boot.
- **fscrypt** — оптимален для гибкой защиты данных отдельных пользователей или каталогов.

🔑 6. Как выбрать подход

Выбор между **LUKS** и **fscrypt** зависит от задач, уровня защищаемости и архитектуры системы. Ниже — краткие рекомендации по выбору.

✅ Когда использовать `LUKS`

- 🔐 Нужна **полная защита** всех данных, включая swap, tmp, root;
- Используется **ноутбук** или портативное устройство;
- 🖥️ Сервер хранит **чувствительные данные** и доступ имеют администраторы;
- ⚙️ Требуется защита GRUB/initramfs (возможно с ключом или TPM);
- 🎁 Один пользователь или централизованное управление ключами.

🎯 Когда лучше `fscrypt`

- 📂 Нужно шифровать **отдельные каталоги**, например `/home/user` или `/srv/private`;
- 👥 В системе есть **несколько пользователей**, каждому — своя защита;
- 📱 Используется **Android** или встраиваемое устройство;
- ☁️ **Cloud-сценарии** с ограниченным доступом к разделам;
- 🔧 Нельзя (или нежелательно) форматировать диск.

💡 **Комбинированный подход:**
Вы можете использовать `LUKS` для шифрования всего диска и `fscrypt` — для дополнительной защиты каталогов внутри. Это особенно полезно для многоуровневой безопасности и ограничения доступа по ролям.

7. Практические сценарии

Ниже приведены типичные примеры использования LUKS и fscrypt в реальных системах. Эти сценарии охватывают как персональные устройства, так и автоматизацию на серверной или корпоративной инфраструктуре.

💻 Защита ноутбука с LUKS

- Полное шифрование диска с LUKS — включая `/`, `/home`, `swap`.
- Запрос пароля при загрузке через initramfs или GRUB.
- Возможность резервного хранения ключа на USB или в TPM (при поддержке).
- Рекомендуется использовать `LUKS2` с современными методами защиты заголовка.

🔐 Шифрование каталога пользователя с fscrypt

- Шифруется только `/home/user` или отдельные директории (`~/Private`, `~/Projects/secure`).
- fscrypt интегрируется с PAM — ключ расшифровывается при входе в сессию.
- Возможно настроить `fscrypt.conf` на использование пользовательских политик шифрования.
- Подходит для многопользовательских рабочих станций или Linux-серверов с home-директориями.

⚙️ Интеграция с `systemd` и PAM

- Используйте `pam_fscrypt` для автоматической расшифровки каталога пользователя при входе.
- Совместимо с `systemd-homed` для централизованного управления учётными записями и зашифрованными домашними директориями.
- Позволяет полностью автоматизировать монтирование зашифрованных директорий.

🛡️ Автоматизация с ключевыми файлами и TPM

- LUKS можно разблокировать автоматически через:
  - `keyfile` в initramfs (например, с USB-флешки);
  - **TPM 2.0** с `clevis` и `tang` (сетевой разблокировщик);
  - Устройства с привязкой к плате (Trusted Boot).
- Это упрощает автоматические перезагрузки и развёртывания в датацентрах.

🎁 8. Особенности и подводные камни

Несмотря на надёжность современных решений для шифрования, есть нюансы, которые важно учитывать при проектировании и эксплуатации. Ниже приведены ключевые особенности и потенциальные риски.

ϟ Производительность и TRIM на SSD

- При использовании LUKS на SSD важно включить поддержку TRIM, чтобы не терять производительность и продлевать срок службы устройства.
- Используйте опцию `allow-discards` или `discard` в `/etc/crypttab`:

```
secure_volume UUID=xxxx none luks,discard
```

- Альтернативно — запускать `fstrim` через systemd-таймер:

```
sudo systemctl enable fstrim.timer
```

❗ Риски потери ключей и паролей

- В случае утери master-ключа LUKS — восстановление данных **невозможно**.
- Резервное копирование заголовка LUKS:

```
cryptsetup luksHeaderBackup /dev/sdX1 --header-backup-file luks-header.img
```

- fscrypt зависит от ключей в `keyring` — при их потере каталог становится недоступен.

🌀 Совместимость со снапшотами

- LUKS отлично сочетается с LVM- и ZFS-снапшотами (вне области шифрования).
- fscrypt работает с Btrfs, но есть ограничения:
  - Снапшоты fscrypt-зашифрованных каталогов **не читаемы без ключа**.
  - Некоторые инструменты снапшотов (например, `rsnapshot`) не работают корректно без расшифровки.

Удаление ключей и остаточных данных

- Удаление зашифрованного каталога или LUKS-раздела **не гарантирует**

уничтожения ключа.

- Рекомендуется вручную удалить ключи, перезаписать заголовки, использовать `shred` или `blkdiscard`:

```
sudo cryptsetup luksErase /dev/sdX1
```

- Для SSD — используйте `blkdiscard` вместо `dd`:

```
sudo blkdiscard /dev/sdX1
```

⚠️ **Важно:** перед любыми операциями с ключами делайте резервные копии и проверяйте конфигурацию несколько раз. Потеря ключа — необратима.

🔐 9. Интеграция с защищёнными ОС

Шифрование в Linux активно используется в сертифицированных и защищённых операционных системах — от мобильных платформ до специализированных дистрибутивов с ИБ-функциями. Рассмотрим ключевые примеры.

🛡️ Поддержка в российских защищённых ОС

- **Astra Linux** (в т.ч. Special Edition):
  - Полноценная поддержка `LUKS` для защиты дисков и разделов;
  - `fscrypt` используется для защиты пользовательских и служебных каталогов;
  - Встроенная интеграция с SELinux/IMA/EVM (в зависимости от редакции).
- **НАЙС.ОС**, **РЕД ОС**, **ALT**:
  - Работают с LUKS по умолчанию;
  - Имеют модули для политики доверенной загрузки и контроля целостности;
  - Включают поддержку ключевых файлов, смарт-карт, TPM.

🔋 Android и fscrypt

- Система Android полностью перешла на `fscrypt` для шифрования пользовательских данных с Android 7.0+.
- Каталоги `/data` и `/sdcard` шифруются по пользовательскому ключу, с разблокировкой по паролю/FaceID/сканеру.
- Используется аппаратное шифрование, keymaster и TrustZone.

🔒 Интеграция с SELinux, IMA и AIDE

- **SELinux + fscrypt:** можно задавать политики доступа на уровне шифрованных каталогов.
- **IMA/EVM:** подписывают и проверяют целостность файлов, даже внутри зашифрованных разделов.
- **AIDE (Advanced Intrusion Detection Environment):**
  - Поддерживает контроль целостности зашифрованных директорий;
  - Важно правильно монтировать и расшифровывать тома до запуска AIDE.

💡 **Вывод:** современные защищённые ОС умеют использовать LUKS и fscrypt в сочетании с контролем целостности, политиками доступа и механизмами доверенной загрузки.

✔ 10. Заключение

Использование шифрования на Linux — это не только безопасность, но и осознанное проектирование архитектуры хранения. Правильный выбор инструмента зависит от задач, инфраструктуры и уровня требуемой защиты.

- 🔐 **Полная защита всего диска** — используйте `LUKS`.
- 📁🔑 **Гибкое шифрование каталогов** — применяйте `fscrypt`.
- Комбинация LUKS + fscrypt — лучший выбор в системах с разными зонами доступа.
- 💾 **Регулярно делайте резервные копии ключей**, заголовков LUKS и экспорт fscrypt-политик.
- 📊 Шифрование ≠ мониторинг — следите за целостностью, свободным местом и логами доступа.

Шифрование — это фундамент, но не единственный элемент безопасности. Сочетайте его с SELinux, IMA/EVM, бэкапами и контролем доступа.

📎 Бонус: примеры конфигурации

`/etc/crypttab` (автоматическое подключение LUKS при загрузке):

```
secure_volume UUID=xxxx-xxxx-xxxx-xxxx none luks,discard
```

`/etc/fstab` (монтаж расшифрованного раздела):

```
/dev/mapper/secure_volume / ext4 noatime,errors=remount-ro 0 1
```

`/etc/pam.d/common-session` (активация PAM для fscrypt):

```
session optional pam_fscrypt.so
```

Монтирование через `systemd` unit-файл:

```
[Unit]
Description=Mount encrypted home
After=cryptsetup.target

[Mount]
What=/dev/mapper/secure_volume
Where=/home
Type=ext4
Options=noatime

[Install]
WantedBy=multi-user.target
```

💡 **Совет:** храните ключи, backup-файлы и пароли отдельно от зашифрованных носителей — и, по возможности, защищайте их вторым фактором (например, смарт-картой или TPM).