losetupcryptoloop（传统块设备加密）

0 losetup/cryptoloop / linux

losetup/cryptoloop 加密磁盘的原理与逻辑分析

---

1. 核心原理

（1）循环设备（/dev/loop）的作用

- `/dev/loopX` 是 Linux 内核提供的**虚拟块设备**，允许将普通文件（如 `bigf1`）模拟成块设备（类似物理磁盘）。
- 通过 `losetup` 命令绑定文件后，可以像操作真实磁盘一样对其进行分区、格式化、挂载。

（2）加密流程（cryptoloop）

- **加密层**：`cryptoloop` 是内核模块，在循环设备之上添加加密功能。
  - 使用 `-e aes` 指定加密算法（如 AES），数据写入文件前会先加密，读取时再解密。
  - 密钥由用户输入的密码生成。
- **透明加解密**：所有通过挂载点（如 `d1`）的读写操作，均自动经过加密/解密，用户无感知。

---

2. 实验步骤与逻辑分解

步骤	命令	作用	关键点
1. 创建空白文件	`dd if=/dev/zero of=bigf1 bs=1M count=77`	生成 77MB 的空白文件	文件内容全为 `0`，用于模拟磁盘
2. 加载加密模块	`modprobe cryptoloop aes`	激活内核的 `cryptoloop` 和 AES 算法	需内核支持，现代系统可能已移除
3. 绑定加密设备	`losetup -e aes /dev/loop1 bigf1`	将文件关联到 `/dev/loop1` 并启用 AES 加密	需输入密码，密钥用于加解密
4. 分区与格式化	`mkfs /dev/loop1`	格式化为文件系统（如 ext4）	加密发生在文件系统层之下
5. 挂载与使用	`mount /dev/loop1 d1`	通过挂载点 `d1` 访问加密存储	所有写入 `d1` 的数据自动加密

---

3. 实际使用与技术评价

（1）优点

- **轻量级**：仅依赖内核模块和 `losetup` 工具，无需额外软件。
- **历史意义**：早期 Linux 加密磁盘的简易方案，适合学习底层原理。

（2）缺点

- **过时技术**：
  - 现代内核（≥2.6.37）默认禁用 `cryptoloop`，推荐 `dm-crypt`（LUKS）。
  - 加密算法受限（如默认 AES-CBC 易受攻击）。
- **功能简陋**：
  - 无密钥管理（密码直接生成密钥，无盐值或迭代）。
  - 无法支持现代加密模式（如 AES-XTS）。

（3）与现代方案（dm-crypt/LUKS）对比

特性	cryptoloop	dm-crypt/LUKS
加密强度	弱（AES-CBC）	强（支持 AES-XTS 等）
密钥管理	无（直接密码派生）	支持密钥文件、恢复头等
兼容性	仅旧内核	所有现代 Linux 发行版
易用性	手动操作多	工具链完善（cryptsetup）

注意事项：

经过我的测试表明，在现代 Linux 系统（如 Kali Linux 2024 使用内核 6.8.11）中，传统的 `losetup -e` 加密方式已经完全被移除了。这是 Linux 内核发展的必然结果，因为 `cryptoloop` 机制存在安全性不足的问题，已被更先进的 `dm-crypt` 技术取代。

只有较为远古的版本可以使用，如Ubuntu14的32位版本，检查步骤如下：**检查 `losetup` 是否支持加密（`-e` 选项）**

```
losetup --help | grep -- -e
```

```
yang@ubuntu:~/桌面$ losetup --help | grep -- -e
 -e, --encryption <type> enable data encryption with specified <name/num>
```

losetup中有-e选择说明可以加密，新版本这个命令会报错

进一步验证是否真正可用

运行以下命令确认模块是否已加载：

```
lsmod | grep cryptoloop  # 检查模块是否在内存中
```

```
yang@ubuntu:~/桌面$ lsmod | grep cryptoloop
cryptoloop             16384  0
```

确实有这个模块，那么开始实验吧

1. 测试步骤

以下是在 **Ubuntu14（32位）** 上测试 `losetup + cryptoloop` 的完整流程：

(1) 创建测试文件

```
mkdir -p ~/tmp/cryptoloop_test && cd ~/tmp/cryptoloop_test
dd if=/dev/zero of=encrypted_disk.img bs=1M count=100  # 创建100MB空白文件
```

(2) 加载 cryptoloop 模块

```
sudo modprobe cryptoloop  # 加载 cryptoloop 模块（现代内核可能已移除）
sudo modprobe aes         # 加载 AES 加密算法支持
```

> **注意**：
>
> - 在较新内核（如 5.x）中，`cryptoloop` 可能已被移除，需使用 `dm-crypt`（见4.2节）。
> - 若报错 `modprobe: FATAL: Module cryptoloop not found`，则说明该功能已废弃。
> - 实验过程中发现 `cryptoloop` 模块可以加载，但 `aes` 模块加载失败（`padlock_aes` 报错），这说明你的系统可能缺少某些加密硬件支持或内核配置问题。
> - 检查当前可用的加密算法：cat /proc/crypto | grep -A 5 "aes" | grep "name|driver"

```
yang@ubuntu:~/tmp/cryptoloop_test$ cat /proc/crypto | grep -A 5 "aes" | grep "na
me\|driver"
name         : xts(aes)
driver       : xts-aes-aesni
name         : lrw(aes)
driver       : lrw-aes-aesni
name         : __xts-aes-aesni
driver       : __driver-xts-aes-aesni
name         : __lrw-aes-aesni
driver       : __driver-lrw-aes-aesni
name         : pcbc(aes)
driver       : pcbc-aes-aesni
name         : cbc(aes)
driver       : cbc-aes-aesni
name         : ecb(aes)
driver       : ecb-aes-aesni
name         : __cbc-aes-aesni
driver       : __driver-cbc-aes-aesni
name         : __ecb-aes-aesni
driver       : __driver-ecb-aes-aesni
name         : __aes-aesni
driver       : __driver-aes-aesni
name         : aes
driver       : aes-aesni
name         : aes
driver       : aes-asm
name         : aes
driver       : aes-generic
```

(3) 关联 loop 设备并加密

- `losetup` 是 Linux 的一个工具，用于将文件关联到 `/dev/loopX` 设备（虚拟块设备）。

```
sudo losetup -e aes /dev/loop0 encrypted_disk.img  # 使用 AES 加密
```

> **输入密码**：此处需设置加密密码（如 `test123`）。

(4) 格式化并挂载

```
sudo mkfs.ext4 /dev/loop0              # 格式化为 ext4
mkdir -p ./mnt
sudo mount /dev/loop0 ./mnt            # 挂载到 ./mnt
sudo chown $USER:$USER ./mnt           # 允许当前用户读写
```

```
yang@ubuntu:~/tmp/cryptoloop_test$ sudo mkfs.ext4 /dev/loop0
mke2fs 1.42.9 (4-Feb-2014)
文件系统标签=
OS type: Linux
块大小=1024 (log=0)
分块大小=1024 (log=0)
Stride=0 blocks, Stripe width=0 blocks
25688 inodes, 102400 blocks
5120 blocks (5.00%) reserved for the super user
第一个数据块=1
Maximum filesystem blocks=67371008
13 block groups
8192 blocks per group, 8192 fragments per group
1976 inodes per group
Superblock backups stored on blocks:
        8193, 24577, 40961, 57345, 73729

Allocating group tables: 完成
正在写入inode表: 完成
Creating journal (4096 blocks): 完成
Writing superblocks and filesystem accounting information: 完成
```

(5) 测试文件读写

```
echo "Hello, cryptoloop!" > ./mnt/test.txt  # 写入测试文件
cat ./mnt/test.txt                          # 读取验证
hexdump -C encrypted_disk.img | head -n 10  # 查看加密后的磁盘内容（应为乱码）
```

读取明文是"Hello, cryptoloop!"

密文为乱码，成功加密！

上面的流程解释：

加密流程

1. `sudo losetup -e aes /dev/loop0 encrypted_disk.img`
   - 把 `encrypted_disk.img` 文件关联到 `/dev/loop0` 设备。
   - `-e aes` 表示所有写入 `/dev/loop0` 的数据都会用 AES 加密后存储到 `encrypted_disk.img`。
   - **输入密码**（如 `test123`）：这个密码会用于生成 AES 加密密钥。
2. **加密后的数据存储**
   - 当你往 `/dev/loop0` 写数据时，数据会先被 AES 加密，再写入 `encrypted_disk.img`。
   - 当你从 `/dev/loop0` 读数据时，数据会先从 `encrypted_disk.img` 读取并解密，再返回给你。

(6) 卸载并清理

```
sudo umount ./mnt
sudo losetup -d /dev/loop0  # 解除 loop 设备关联
rm -rf encrypted_disk.img ./mnt
```

---

2. 测试结果

项目	结果
加密功能	成功用 AES 加密文件，`hexdump` 显示内容为密文。
读写性能	速度较慢（因旧版 cryptoloop 无硬件加速）。
内核支持	Debian 5（4.19内核）仍支持，但新内核（≥5.x）已移除。
与现代替代方案对比	远不如 `dm-crypt/LUKS`（见4.2节）安全、稳定。

---

3. 评价与体验

优点

1. **简单直接**：
   - 仅需 `losetup` + 文件即可模拟加密磁盘，适合快速测试。

2. **历史意义**：
    - 是 Linux 早期加密存储的解决方案，帮助理解块设备加密的基本原理。

缺点

1. **已被废弃**：
    - 现代内核（≥5.x）默认移除 `cryptoloop`，强制使用 `dm-crypt`。
    - 执行 `modprobe cryptoloop` 可能失败。
2. **安全性不足**：
    - 密钥管理简单（直接明文密码），无防暴力破解机制（如 LUKS 的迭代哈希）。
3. **性能低下**：
    - 无 AES-NI 硬件加速支持，加密/解密速度慢。

适用场景

- **学习用途**：理解 Linux 加密存储的历史演进。
- **老旧系统维护**：在古董级 Linux 系统（如 CentOS 5）中临时加密文件。

4. 体验总结

- **实验成功但实用性低**：
    - 在旧系统（如 Debian 5）中可复现，但现代环境需降级内核或改用 `dm-crypt`。
- **学习价值高**：
    - 理解 Linux 加密存储的演进：`cryptoloop` → `dm-crypt` → LUKS。
    - 掌握循环设备、内核模块、透明加密等概念。
- **生产环境警示**：
    - **切勿用于真实数据**！`cryptoloop` 缺乏安全审计，易受攻击。
- **作用**：通过 `loop设备` 将文件模拟为块设备，并用 `cryptoloop` 内核模块加密。
- **局限**：
    - 已被 `dm-crypt` 取代（安全性低，内核支持少）。
- **关联性**：
    - 是 `dm-crypt/LUKS` 的前身，用于理解块设备加密的基本原理。