Encrypted disks in Linux

Milan Brož
mbroz@redhat.com

VOLUME ENCRYPTION (layer selection)

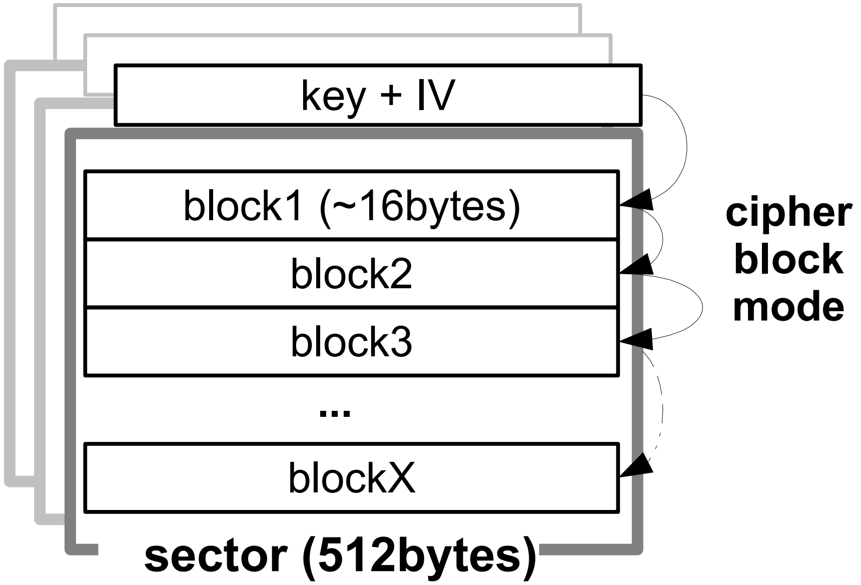
- **HW**
- **Application data**

- **File system: (EncFS, eCryptfs, ...)**
  - file level
  - metadata of algorithm in file, copy/move with data
  - parts of FS metadata unencrypted (attributes, file names, ...)
  - selection what is encrypted (which files, directories)

- **Block device: (dm-crypt, truecrypt, loop-aes, ...)**
  - sector level encryption
  - independent of FS (layer below the FS)
  - mostly used with volume management (LVM)
  - swap partition

disc encryption

- **block device – atomic unit is sector**
  - in Linux sector always 512 bytes, random access
  - **sectors encrypted independently**
  - (sectors contains random data before write)
  - encryption algorithm uses **block** <= sector
    - block is ususually 128bits (16 bytes)
    - last block has the same size as others



- IV – initialization vector
  - different for every sector
  - derived from block number

- granularity – block vs sector

Block cipher algorithms (examples)

aes-cbc-essiv:sha256, aes-xts-plain, ...

- **Algorithm** (define key length)
  AES, twofish, serpent, ...

- **Cipher block mode**
  CBC (cipher block chaining)
  LRW (Liskov,Rivest,Wagner), since kernel 2.6.20
  XTS (XEX-TCB-CTS), since kernel 2.6.24, suitable for <1 TB data

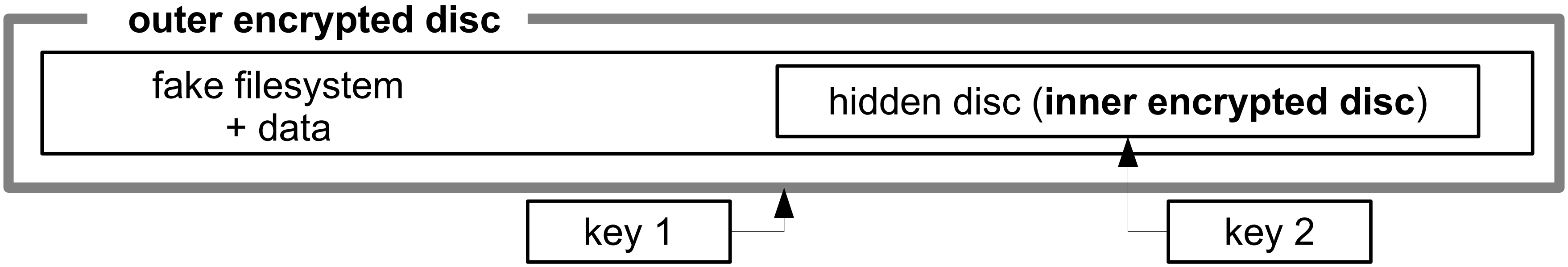
- Wide modes (block=sector) need two pass processing, patented, mostly not used.

(Maybe this change with prepared EME-2 which is not patent encumbered.)

- **IV – initialization vector**
  - **plain** - sector number (padded with zeroes to requested size)
  - **ESSIV** – Encrypted Salt-Sector, derived from hash of key

Hidden volume

- **plausible deniability**
  - **ability to "plausible" deny that there are data**
  - data stored in "unused" part of disc
    need another key, encrypted data looks like "random noise"
  - no visible header for encrypted data
    (you must decrypt the data and verify signature inside
    to prove that data are there)



- used in Truecrypt
- with device-mapper you can create similar design
- note possible "data leaks" to system from hidden disc

dm-crypt + cryptsetup

- kernel device-mapper **crypt target**
  used for transparent block device encryption
  (and use arbitrary filesystem above that)

- **Using kernel cryptoAPI**
  - **HW support** (VIA Padlock, Geode, AES-NI, ...)

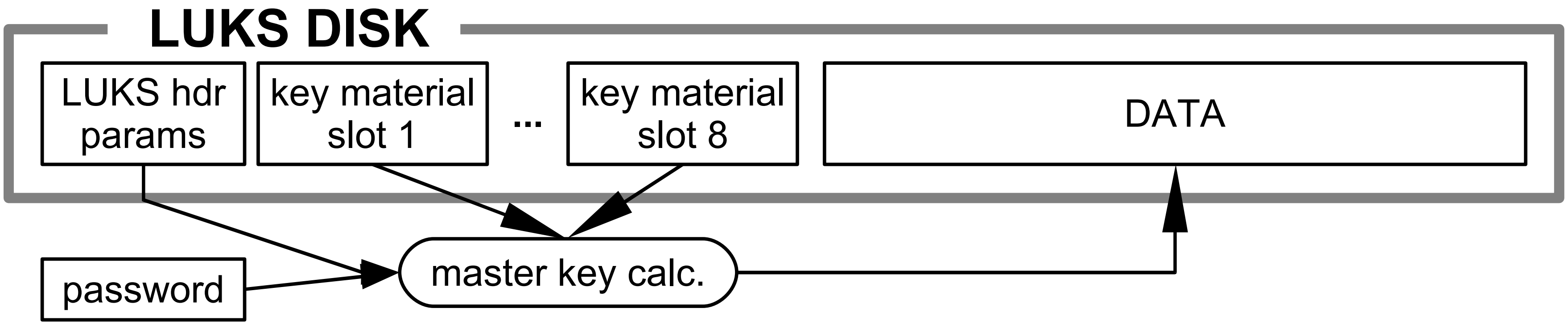
- **cryptsetup[-luks]** – tool for dm-crypt configuration
  - **defines LUKS key store**

- + GUI, udev, HAL integration

LUKS (Linux Unified Key Setup)

- de facto standard in Linux

- portable, supported by other OS (FreeOTFE.org)
- **several passphrases** unlocks strong volume **key,** PBKDF2
- **passphrase change** (invalidation)
  - without whole disc reencryption



AF-splitter – anti-forensic protection
(against master key revive from hw reallocated sectors)

cryptsetup

LUKS commands: Format, Open, Close, AddKey, KillSlot, Dump

(create, remove, status – direct dm-crypt setting without LUKS)

luksFormat – create LUKS header

```
cryptsetup [-c serpent-cbc-essiv:sha256 -s 256] luksFormat $DEV
```

luksOpen – make device available (map device)

```
cryptsetup luksOpen $DEV $CRYPT_DEV_NAME
```

luksClose – remove mapping

```
cryptsetup luksClose $CRYPT_DEV_NAME
```

luksAddKey, luksKillSlot, (luksRemoveKey) – keyslots manipulation

luksDump – show info about encrypted disc

```
Cipher name:    serpent
Cipher mode:    cbc-essiv:sha256
Payload offset: 2056
UUID: 09714b0c-9a70-4652-86d2-7300b755eb4f
```

cryptsetup

Configuration example – distro dependent

/etc/crypttab:

```
#<tgt.dev> <src.dev>  <key file>   <options>
```

- simple disc, LUKS (no paramaters needed)

```
$CDISK     /dev/sdX   none         retry=5
```

- swap over LVM volume, no LUKS, random volume key (differs every boot)

- note that init scripts must initializei RNG during bootu before – random seed

```
$CSWAP    /dev/VG/lv  /dev/urandom swap,cipher=aes-cbc-essiv:sha256
```

/etc/fstab:

```
/dev/mapper/$CDISK $MNT  auto    defaults     0 0
/dev/mapper/$CSWAP swap  swap    defaults     0 0
```

cryptsetup

- **Resize of encrypted disc**
  - No device size in LUKS header,
    just resize underlying device and activate
  - ... and change FS size above
  - ... and fill new space with random data

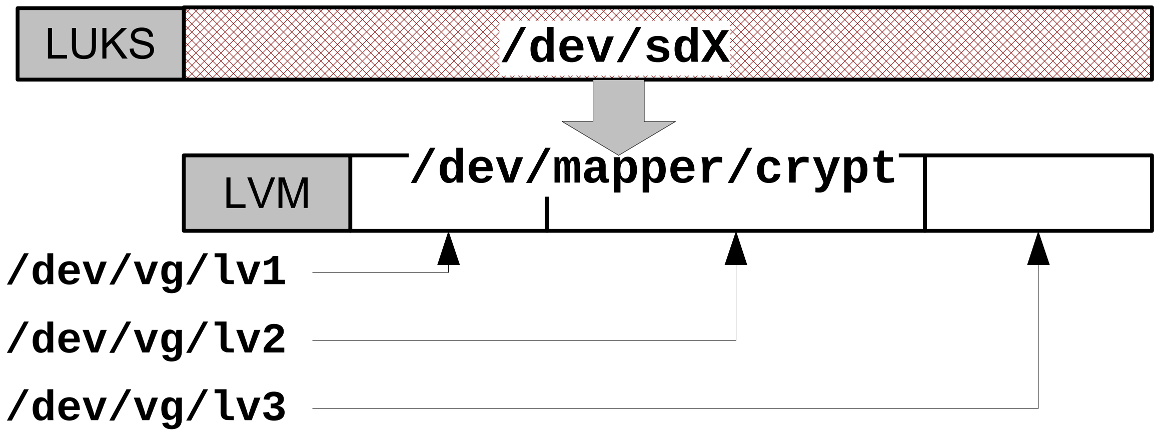
- **Change of encryption algorithm, volume key, ...**
  - The safest way: use copy to another disc

- In future integration into LVM and to key management system

Cryptsetup + LVM (Logical Volume Management)

- LVM metadata – redundance, history of changes
- LUKS metadata – undesirable to keep old header

- **LVM over encrypted disc**
  - PV is encrypted disc
  - LVM metadata are encrypted too



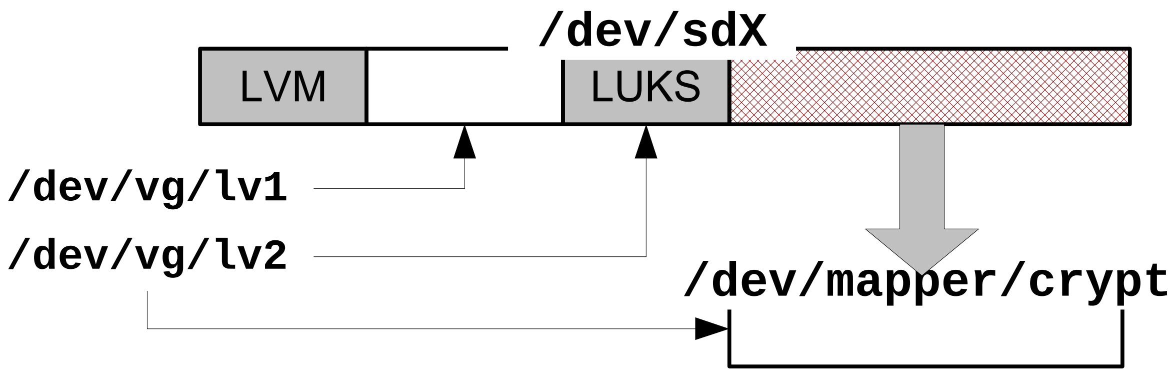
```
crypsetup luksOpen \
    /dev/sdX crypt

vgscan

vgchange -a y vg
```

Cryptsetup + LVM (Logical Volume Management)

- **encrypted LVM volume**
  - only some volumes can be encrypted



```
vgchange -a y vg
```

```
cryptsetup luksOpen \
/dev/vg/lv2 crypt
```

- both usable for system disc
- ... but you have to use initrd
- ... with all requested kernel modules (disc, dm-crypt, crypto)

Backup, recovery

- No LUKS header – all data gone
- Error diffusion
  - one bad bit in RAM – loss of one block of encrypted data (at least)
  - HW errors usually leads to worse situation then with plain disc

- **Data recovery**
  - Backup data inside the encrypted device :-)

  - LUKS
    - Backup of mapping table and volume key
      - `dmsetup table --showkeys`
      - If you know volume key and algorithm, no need for passphrase!
    - **LUKS header backup**
      - `dd if=/dev/<dev> of=backup.img bs=512 count=NUM`
      - NUM – sector count in Payload Offset (luksDump)
      - You need to know (one) passphrase

dm-crypt performance

- encryption speed...
  - CPU load, IO pattern type, various optimalisation

  - IOs are serialized (processed in sequence)
    - a file sync can wait for other data

  - Special encryption thread per volume, latency
  - multicore/SMP support?

  - **HW acceleration**
    - formerly used in IPsec
    - transparent, kernel drivers
    - asynchronnous mode