LUKS1 vs. LUKS2

or how do I encrypt my whole disk...

Cyril Brulebois <cyril@debamax.com>

9 June 2019

Mini-DebConf Hamburg

Introduction to LUKS

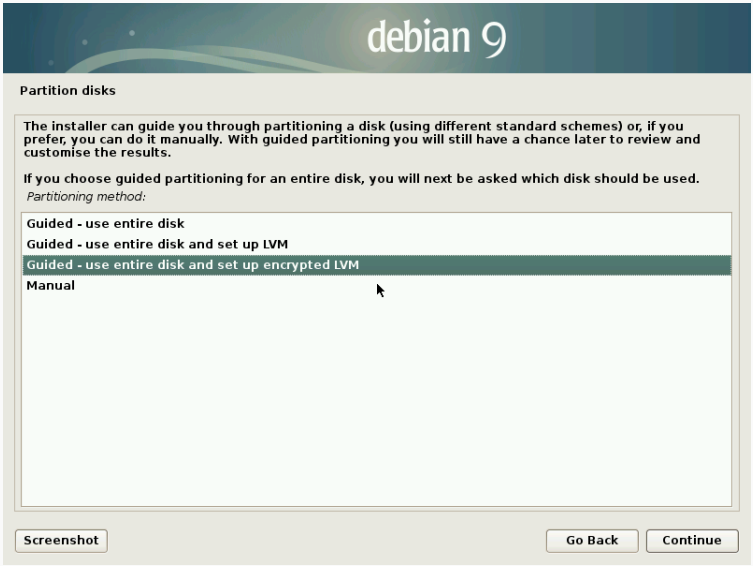
LUKS = Linux Unified Key Setup

Provides disk encryption

Works on a block device, rather than on a file system

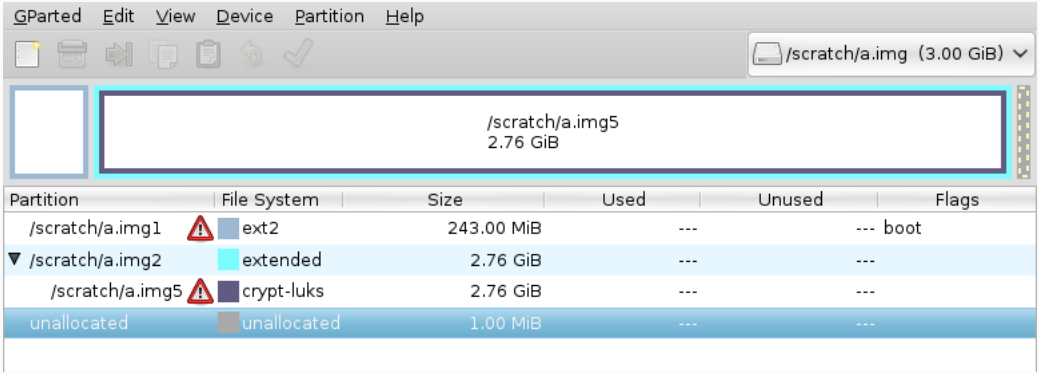
Makes full disk encryption possible

Full disk encryption seems possible



Really full?

(in)famous separate /boot



Cryptodisk to the rescue!

GRUB's cryptodisk feature:

- GRUB asks for a passphrase
- then unlocks the device on its own
- then looks for config, kernel, initramfs, etc.

Supported backends:

- GELI (FreeBSD): since 2011
- LUKS (Linux): since 2014

Unfortunately: no support in the Debian Installer

Hackish solutions

User-submitted, untested instructions to convince Debian Installer not to perform its usual checks → `https://lists.debian.org/debian-boot/2019/01/msg00035.html`

Otherwise, possible workaround:

- follow guided partitioning (use entire disk and set up encrypted LVM)
- move `/boot` to the root filesystem
- enable cryptodisk support in GRUB
- (pretend the wasted partition/space are not an issue)

Hackish solution: instructions for Stretch

Copy-move `/boot` into the root FS:

```
cp -r /boot /boot.copy
umount /boot
rmdir /boot
mv /boot.copy /boot
```

Comment out the `UUID=` line for `/boot` in `/etc/fstab`

Enable cryptodisk support in GRUB and re-install it on target device:

```
echo 'GRUB_ENABLE_CRYPTODISK=y' >> /etc/default/grub
update-grub
grub-install /dev/sda
```

Hackish solution: instructions for Stretch, consequences

Some differences in /boot/grub/grub.cfg:

```
 insmod part_msdos
+insmod cryptodisk
+insmod luks
+insmod gcry_rijndael
+insmod gcry_rijndael
+insmod gcry_sha256
+insmod lvm
 insmod ext2

-set root='hd0,msdos1'
+cryptomount -u 2072b09dcb25447386121d0103ad7db5
+set root='lvmid/dwLFt5-njhz-t2iF-kfEH-5Fwv-df3U-OdZDYR/gKjDoO-2itn-YbMk-nrT1-dkI9-faQv-kGPaaf'

-linux  /vmlinuz-4.9.0-9-amd64 root=/dev/mapper/autotest--vg-root ro  quiet
+linux  /boot/vmlinuz-4.9.0-9-amd64 root=/dev/mapper/autotest--vg-root ro  quiet
 echo   'Chargement du disque mémoire initial…'
-initrd /initrd.img-4.9.0-9-amd64
+initrd /boot/initrd.img-4.9.0-9-amd64
```

Hackish solution: boot process with Stretch (1/4)

```
SeaBIOS (version 1.10.2-1)


iPXE (http://ipxe.org) 00:03.0 CA00 PCI2.10 PnP PMM+3FF90DC0+3FED0DC0 CA00



Booting from Hard Disk...
Attempting to decrypt master key...
Enter passphrase for hd0,msdos5 (2072b09dcb25447386121d0103ad7db5):
```

Hackish solution: boot process with Stretch (2/4)

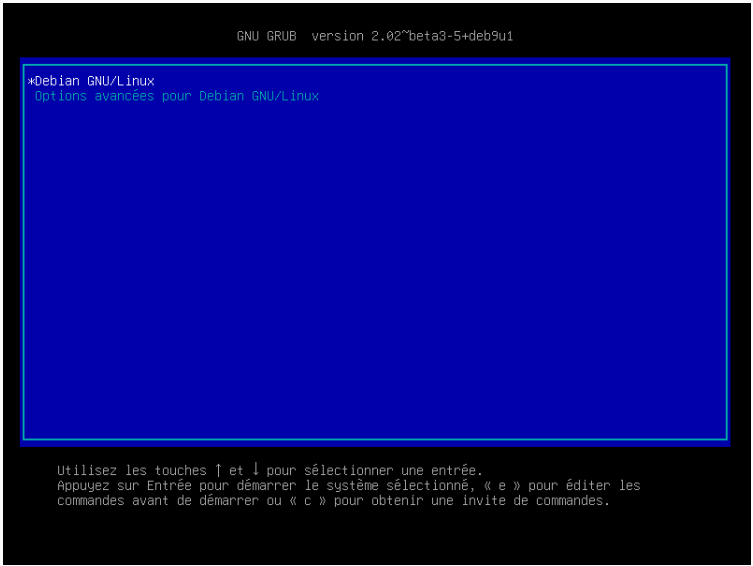
```
SeaBIOS (version 1.10.2-1)


iPXE (http://ipxe.org) 00:03.0 CA00 PCI2.10 PnP PMM+3FF90DC0+3FED0DC0 CA00



Booting from Hard Disk...
Attempting to decrypt master key...
Enter passphrase for hd0,msdos5 (2072b09dcb25447386121d0103ad7db5):
Slot 0 opened
```

Hackish solution: boot process with Stretch (3/4)



Hackish solution: boot process with Stretch (4/4)

```
  WARNING: Failed to connect to lvmetad. Falling back to device scanning.
  Volume group "autotest-vg" not found
  Cannot process volume group autotest-vg
  WARNING: Failed to connect to lvmetad. Falling back to device scanning.0
  Volume group "autotest-vg" not found
  Cannot process volume group autotest-vg
Please unlock disk sda5_crypt:
```

All good then?

Reportedly working since Wheezy...

All good then?

Let's adapt Debian Installer finally?

Not so quick...

Hackish solution: boot process with Buster

```
SeaBIOS (version 1.10.2-1)


iPXE (http://ipxe.org) 00:03.0 CA00 PCI2.10 PnP PMM+3FF90DC0+3FED0DC0 CA00



Booting from Hard Disk...
error: failure reading sector 0x0 from `fd0'.
error: disk `lvmid/2lwb8R-pGQ6-zkqq-1us5-mwg1-2sce-jcoyMU/yRZykf-CI42-IAEG-U9i3-
VasH-fThS-6mQcSj' not found.
Entering rescue mode...
grub rescue>
```

Hackish solution: configuration changes with Buster

Let's check /boot/grub/grub.cfg differences in Buster:

```
-insmod part_msdos
+insmod lvm

-set root='hd0,msdos1'
+set root='lvmid/2lwb8R-pGQ6-zkqq-1us5-mwg1-2sce-jcoyMU/yRZykf-CI42-IAEG-U9i3-VasH-fThS-6mQcSj'

-linux  /vmlinuz-4.19.0-5-amd64 root=/dev/mapper/autotest--vg-root ro  quiet
+linux  /boot/vmlinuz-4.19.0-5-amd64 root=/dev/mapper/autotest--vg-root ro  quiet

-initrd /initrd.img-4.19.0-5-amd64
+initrd /boot/initrd.img-4.19.0-5-amd64
```

Regression in Buster

Problem:

- Recent change: `cryptsetup` defaults to LUKS2
- New, different on-disk format
- Not supported by GRUB2, yet

But maybe it's possible to add support for LUKS2?
→ `https://savannah.gnu.org/bugs/?55093`

LUKS implementation in GRUB

Seems rather small (hundreds of lines)

Entry points in `grub-core/disk/luks.c` (used by `grub-core/disk/cryptodisk.c`):

```
struct grub_cryptodisk_dev luks_crypto = {
  .scan = configure_ciphers,
  .recover_key = luks_recover_key
};
```

Everything happens with:

- `grub_luks_phdr` structure
- `configure_ciphers()`: parse LUKS headers, then configure ciphers
- `luks_recover_key()`: ask for passphrase, recover key

LUKS1's on-disk format (1/3): data for configure_ciphers()

start offset	field name	length	data type	description
0	magic	6	byte[]	magic for LUKS partition header, see LUKS_MAGIC
6	version	2	uint16_t	LUKS version
8	cipher-name	32	char[]	cipher name specification
40	cipher-mode	32	char[]	cipher mode specification
72	hash-spec	32	char[]	hash specification
104	payload-offset	4	uint32_t	start offset of the bulk data (in 512 bytes sectors)

LUKS1's on-disk format (2/3): data for configure_ciphers()

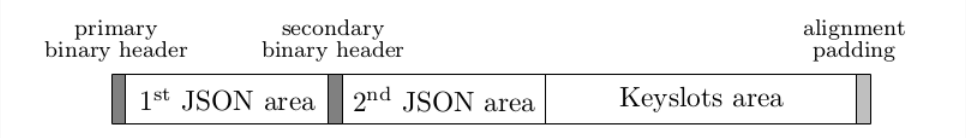
108	key-bytes	4	uint32_t	number of key bytes
112	mk-digest	20	byte[]	master key checksum from PBKDF2
132	mk-digest-salt	32	byte[]	salt parameter for master key PBKDF2
164	mk-digest-iter	4	uint32_t	iterations parameter for master key PBKDF2
168	uuid	40	char[]	UUID of the partition
208	key-slot-1	48	key slot	key slot 1
256	key-slot-2	48	key slot	key slot 2
...	...	...	...	...
544	key-slot-8	48	key slot	key slot 8
592	total phdr size			

LUKS1's on-disk format (3/3): data for luks_recover_key()

offset	field name	length	data type	description
0	active	4	unit32_t	state of keyslot, enabled/disabled
4	iterations	4	uint32_t	iteration parameter for PBKDF2
8	salt	32	byte[]	salt parameter for PBKDF2
40	key-material-offset	4	uint32_t	start sector of key material
44	stripes	4	uint32_t	number of anti-forensic stripes

PBKDF2 = Password-Based Key Derivation Function 2

LUKS2's on-disk format (1/2)



LUKS2's on-disk format (2/2)

```c
// All integers are stored as big-endian.
// Header structure must be exactly 4096 bytes.

struct luks2_hdr_disk {
  char      magic[MAGIC_L];        // MAGIC_1ST or MAGIC_2ND
  uint16_t  version;               // Version 2
  uint64_t  hdr_size;              // size including JSON area [bytes]
  uint64_t  seqid;                 // sequence ID, increased on update
  char      label[LABEL_L];        // ASCII label or empty
  char      csum_alg[CSUM_ALG_L]; // checksum algorithm, "sha256"
  uint8_t   salt[SALT_L];          // salt, unique for every header
  char      uuid[UUID_L];          // UUID of device
  char      subsystem[LABEL_L];    // owner subsystem label or empty
  uint64_t  hdr_offset;            // offset from device start [bytes]
  char     _padding[184];          // must be zeroed
  uint8_t   csum[CSUM_L];          // header checksum
  char     _padding4096[7*512];    // Padding, must be zeroed
} __attribute__ ((packed));
```

LUKS2's on-disk format, JSON (1/4)

```
"tokens": {},
"segments": {
  "0": {
    "type": "crypt",
    "offset": "16777216",
    "iv_tweak": "0",
    "size": "dynamic",
    "encryption": "aes-xts-plain64",
    "sector_size": 512
  }
},
"config": {
  "json_size": "12288",
  "keyslots_size": "16744448"
},
…
```

LUKS2's on-disk format, JSON (2/4)


```
  "digests": {
    "0": {
      "type": "pbkdf2",
      "keyslots": [
        "0"
      ],
      "segments": [
        "0"
      ],
      "hash": "sha256",
      "iterations": 87849,
      "salt": "Pn5s5EfvYrLN7zXr06mV+wK7odLESB+vY/V30eKH4SY=",
      "digest": "cBtlnzUXkqGlLKAUMIN8DkOF8SsUXX1rIHjFP2gayVo="
    }
  },
}
```

LUKS2's on-disk format, JSON (3/4)

```
"keyslots": {
  "0": {
    "type": "luks2",
    "key_size": 64,
    "af": {
      "type": "luks1",
      "stripes": 4000,
      "hash": "sha256"
    },
    "area": {
      "type": "raw",
      "offset": "32768",
      "size": "258048",
      "encryption": "aes-xts-plain64",
      "key_size": 64
    },
```

LUKS2's on-disk format, JSON (4/4)

```
    "kdf": {
      "type": "argon2i",
      "time": 4,
      "memory": 505358,
      "cpus": 1,
      "salt": "tXXj5Kb/uAjSJySNriF4pO16qmcEKBD2ai4Hkcabbgk="
    } …
```

Needed extra libraries

First clue:

- `cryptsetup` maintainers contacting the installer team (`debian-boot@lists.debian.org`)
- switch to cryptsetup 2.x: new udebs for `cryptsetup-udeb`/`libcryptsetup12-udeb`
  - `libargon2-1-udeb` (from `src:argon2`)
  - `libjson-c3-udeb` (from `src:json-c`)

Challenges with JSON

First attempt:

- try and link `json-c`'s static library into GRUB: failure...
- needs C headers that are no provided by GRUB: system headers are disabled
  possible work around: use the extra headers in `grub-core/lib/posix_wrap`
- linking extra libraries into `libgrubkern`: not trivial → looking for alternatives

Second attempt: `jsmn`

- single-file C header, no linking issue
- unfortunately, only a tokenizer: no data structure

Challenges with Argon2

Argon2:

- key derivation function (similar to PBKDF2), but much more recent (2017 vs. 2000)
- from the paper: "the memory-hard function for password hashing and other applications"
- from experiments on this laptop: `luksOpen` requires 600+ MiB
- thankfully it seems GRUB2 should be able to allocate up to 4 GiB

Integration challenges:

- namespace pollution in static library: `libargon2.a`
- would benefit from a linker script: no libtool yet (`ar rcs` …)
- needs C headers that are no provided by GRUB: system headers are disabled
- linking extra libraries into `libgrubkern`: not trivial
  possible work around: embedded needed `src:argon2` files into `src:grub2` (PoC-only!)

Game plans (1/2)

Current plan for GRUB:

- document my findings on the upstream bug report
- make sure it's possible to link against `argon2`
- leverage `jsmn` to get structured data for the JSON-based config
- use that in `configure_ciphers()` and `luks_recover_key()`
  → to allow LUKS2 with `pbkdf2` at least
- if that works, switch `luks_recover_key()` to using `argon2` calls
  → to allow LUKS2 with `argon2i` (default) and `argon2d`

Game plans (2/2)

Current plan for Debian/Buster:

- document the current LUKS2 vs. GRUB's cryptodisk no-go
  → current RC bug placeholder: `https://bugs.debian.org/927165`
- implement a new `partman-crypto` parameter
  → users can force LUKS1
- update installation guide accordingly
- also mention a LUKS2-to-LUKS1 conversion command
  → helping people who read the doc, but too late

Extra plan, thanks to Guilhem Moulin:

- avoid move-/`boot`-to-root-filesystem dance
- re-format `/boot` with LUKS1 instead
- then enable cryptodisk support

Thanks for your attention!

More Debian-related write-ups and news:

- `https://debamax.com/blog/`
- Twitter : `@debamax` et `@CyrilBrulebois`

Questions are welcome!

Many thanks to Guilhem Moulin for the fun, the challenges, and the help!