Wikipedia
The Free Encyclopedia
Wikipedia

Linux Unified Key Setup

The **Linux Unified Key Setup** (**LUKS**) is a disk encryption specification created by Clemens Fruhwirth in 2004 and originally intended for Linux.

LUKS implements a platform-independent standard on-disk format for use in various tools. This facilitates compatibility and interoperability among different programs and operating systems, and assures that they all implement password management in a secure and documented manner.[1]

Description

LUKS is used to encrypt a block device. The contents of the encrypted device are arbitrary, and therefore any filesystem can be encrypted, including swap partitions.[2] There is an unencrypted header at the beginning of an encrypted volume, which allows up to 8 (LUKS1) or 32 (LUKS2) encryption keys to be stored along with encryption parameters such as cipher type and key size.[3][4]

The presence of this header is a major difference between LUKS and dm-crypt, since the header allows multiple different passphrases to be used, with the ability to change and remove them. If the header is lost or corrupted, the device will no longer be decryptable.[5]

Encryption is done with a multi-layer approach. First, the block device is encrypted using a *master key*. This master key is encrypted with each active *user key*.[6] User keys are derived from passphrases, FIDO2 security keys, TPMs or smart cards.[7][8] The multi-layer approach allows users to change their passphrase without re-encrypting the whole block device. Key slots can contain information to verify user passphrases or other types of keys.

There are two versions of LUKS, with LUKS2 featuring resilience to header corruption, and using the Argon2 key derivation function by default, whereas LUKS1 uses PBKDF2.[9] Conversion between both versions of LUKS is possible in certain situations, but some features may not be available with LUKS1 such as Argon2.[3] LUKS2 uses JSON as a metadata format.[3][10]

Available cryptographic algorithms depend on individual kernel support of the host. Libgcrypt can be used as a backend for hashing, which supports all of its algorithms.[11] It is up to the operating system vendor to choose the default algorithm.[12] LUKS1 makes use of an anti-forensics technique called AFsplitter, allowing for secure data erasure and protection.[13]

LUKS with LVM

Logical Volume Management can be used alongside LUKS.[14]

LVM on LUKS

When LVM is used on an unlocked LUKS container, all underlying partitions (which are LVM logical volumes) can be encrypted with a single key. This is akin to splitting a LUKS container into multiple partitions. The LVM structure is not visible until the disk is decrypted.[15]

LUKS on LVM

When LUKS is used to encrypt LVM logical volumes, an encrypted volume can span multiple devices. The underlying LVM volume group is visible without decrypting the encrypted volumes.[16]

Full disk encryption

Debian-Installer showing an option for automated partitioning with LVM on LUKS

A common usage of LUKS is to provide full disk encryption, which involves encrypting the root partition of an operating system installation, which protects the operating system files from being tampered with or read by unauthorized parties.[14]

On a Linux system, the boot partition (`/boot`) may be encrypted if the bootloader itself supports LUKS (e.g. GRUB). This is undertaken to prevent tampering with the Linux kernel. However, the first stage bootloader or an EFI system partition cannot be encrypted (see Full disk encryption#The boot key problem).[14]

On mobile Linux systems, postmarketOS has developed osk-sdl (https://wiki.postmarketos.org/wiki/Osk-sdl) to allow a full disk encrypted system to be unlocked using a touch screen.

For systems running systemd, the `systemd-homed` component can be used to encrypt individual home directories.[17]

Operating system support

The reference implementation for LUKS operates on Linux and is based on an enhanced version of cryptsetup, using dm-crypt as the disk encryption backend. Under Microsoft Windows, LUKS-encrypted disks can be used via the Windows Subsystem for Linux.[18] (Formerly, this was possible with LibreCrypt,[19] which currently has fundamental security holes,[20][21] and which succeeded FreeOTFE, formerly DoxBox.)

DragonFly BSD supports LUKS.[22]

Installer support

Several Linux distributions allow the root device to be encrypted upon OS installation. These installers include Calamares,[23] Ubiquity,[24] Debian-Installer,[25] and more.

On-disk format

LUKS headers are backward compatible; newer versions of LUKS are able to read headers of previous versions.[26]

LUKS1

LUKS1 Header[26]

Offset		Data type	Description
0	0_{hex}	char[6]	Magic number {'L', 'U', 'K', 'S', 0xBA, 0xBE }
6	6_{hex}	uint16_t	LUKS Version (0x0001 for LUKS1)
8	8_{hex}	char[32]	Cipher Algorithm (e.g. "twofish", "aes")
40	28_{hex}	char[32]	Cipher mode (e.g. "cbc-essiv:sha256")
72	48_{hex}	char[32]	Cryptographic hash function (e.g. "sha1", "ripemd160")
104	68_{hex}	uint32_t	Payload offset (position of encrypted data) in 512 byte offsets
108	$6C_{hex}$	uint32_t	Number of key bytes
112	70_{hex}	char[20]	PBKDF2 master key checksum
132	84_{hex}	char[32]	PBKDF2 master key salt parameter
164	$A4_{hex}$	uint32_t	PBKDF2 master key iterations (Default: 10)
168	$A8_{hex}$	char[40]	UUID of the partition (e.g. "504c9fa7-d080-4acf-a829-73227b48fb89")
208	$D0_{hex}$	(48 Bytes)	Keyslot 1
…			
544	220_{hex}	(48 Bytes)	Keyslot 8
592 Bytes total			

Format of each keyslot

Offset	Data type	Description
0	uint32_t	State of keyslot: Active=0x00AC71F3; Disabled=0x0000DEAD
4	uint32_t	PBKDF2 iteration parameter
8	char[32]	PBKDF2 salt parameter
40	uint32_t	Start sector of key
44	uint32_t	Number of anti-forensic stripes (Default: 4000)
48 Bytes total		

LUKS2

LUKS2 devices begin with a binary header intended to allow recognition and fast detection by blkid, which also contains information such as checksums. All strings used in a LUKS2 header are null-terminated strings. Directly after the binary header comes the JSON area, containing the objects `config` (configuration), `keyslots`, `digests`, `segments` (describes encrypted areas on the disk), and `tokens` containing extra metadata.[10]

The binary format for regular `luks2` keyslots are mostly similar to their predecessor, with the addition of different per-keyslot algorithms. Another type of key exists to allow redundancy in the case that a re-encryption process is interrupted.[10]

Examples

Cryptsetup is the reference implementation of the LUKS frontend.

To encrypt a device with the path `/dev/sda1`:

```
# cryptsetup luksFormat /dev/sda1
```

To unlock an encrypted device, where `name` is the mapped device name:

```
# cryptsetup open /dev/sda1 name
```

Re-encrypting

Re-encrypting a LUKS container can be done either with the `cryptsetup` tool itself, or with a legacy tool called `cryptsetup-reencrypt`. These tools can also be used to add encryption to an existing unencrypted filesystem, or remove encryption from a block device.[11][27]

Both methods have similar syntax:

```
# cryptsetup reencrypt /dev/sda1
```

```
# cryptsetup-reencrypt /dev/sda1
```

See also

- Comparison of disk encryption software

References

1. Fruhwirth, Clemens (2018-01-20). "LUKS On-Disk Format Specification Version 1.2.3" (https://gitlab.com/cryptsetup/cryptsetup/-/wikis/LUKS-standard/on-disk-format.pdf) (PDF). Retrieved 2021-09-23.
2. "Encrypting drives using LUKS" (https://docs.fedoraproject.org/en-US/quick-docs/encrypting-drives-using-LUKS/). *Fedora Docs*. Retrieved 6 May 2022.
3. "Chapter 12. Encrypting block devices using LUKS" (https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html/security_hardening/encrypting-block-devices-using-luks_security-hardening). *Red Hat Customer Portal*.
4. "How to Encrypt Hard Disk (partition) using LUKS in Linux" (https://www.golinuxcloud.com/how-to-encrypt-hard-disk-partition-luks-linux/). 27 February 2019.
5. "How to Encrypt Your Data with dm-crypt" (https://www.linode.com/docs/guides/encrypt-data-disk-with-dm-crypt/). *Linode*. 22 November 2022.
6. Bossi, Simone; Visconti, Andrea (2015). "What Users Should Know About Full Disk Encryption Based on LUKS" (https://eprint.iacr.org/2016/274.pdf) (PDF). `{{cite journal}}`: Cite journal requires `|journal=` (help)

7. "systemd-cryptenroll - ArchWiki" (https://wiki.archlinux.org/title/Systemd-cryptenroll). *wiki.archlinux.org*. Retrieved 2023-11-22.
8. "How to encrypt a LUKS container using a smart card or token" (https://blog.g3rt.nl/luks-smartcard-or-token.html). 20 April 2014.
9. "How LUKS works with Full Disk Encryption in Linux" (https://infosecwriteups.com/how-luks-works-with-full-disk-encryption-in-linux-6452ad1a42e8). 25 September 2021.
10. "on-disk-format-luks2.pdf" (https://gitlab.com/cryptsetup/cryptsetup/blob/master/docs/on-disk-format-luks2.pdf) (PDF). 7 March 2024.
11. `cryptsetup(8) (https://linux.die.net/man/8/cryptsetup)` – Linux Administration and Privileged Commands Manual
12. "Breaking LUKS Encryption" (https://eforensicsmag.com/breaking-luks-encryption-by-oleg-afonin/). *eForensics*. 21 August 2020.
13. "AFsplitter" (https://clemens.endorphin.org/AFsplitter).
14. "dm-crypt/Encrypting an entire system" (https://wiki.archlinux.org/title/Dm-crypt/Encrypting_an_entire_system). Retrieved 6 May 2022.
15. "Arch with LVM on LUKS" (https://tonisagrista.com/blog/2020/arch-encryption/).
16. "LUKS on LVM: encrypted logical volumes and secure backups" (https://amkeisntall.wordpress.com/2014/09/12/luks-on-lvm-encrypted-logical-volumes-and-secure-backups/). 12 September 2014.
17. "Home Directories" (https://systemd.io/HOME_DIRECTORY/). *systemd*.
18. "Servicing the Windows Subsystem for Linux (WSL) 2 Linux Kernel" (https://devblogs.microsoft.com/commandline/servicing-the-windows-subsystem-for-linux-wsl-2-linux-kernel/). *Microsoft Developer Blogs*. 16 April 2021.
19. "LibreCrypt" (https://github.com/t-d-k/LibreCrypt). *GitHub*. 27 July 2022.
20. "Flaw in driver allows privilege escalation. Feedback wanted · Issue #38 · t-d-k/LibreCrypt" (https://github.com/t-d-k/LibreCrypt/issues/38). *GitHub*. 30 September 2015.
21. "Driver allows writing to arbitrary devices · Issue #39 · t-d-k/LibreCrypt" (https://github.com/t-d-k/LibreCrypt/issues/39). *GitHub*. 7 October 2015.
22. "DragonFly's Major Features List" (https://www.dragonflybsd.org/features/#index6h2). Retrieved 6 May 2022.
23. Michael Larabel (8 May 2016). "Calamares Installer Adds LUKS Encryption Support" (https://www.phoronix.com/scan.php?page=news_item&px=Calamares-LUKS-Support). *Phoronix*.
24. "How to Encrypt Your Hard Disk in Ubuntu" (https://www.maketecheasier.com/encrypt-hard-disk-in-ubuntu/). *Make Tech Easier*. 13 January 2017.
25. "PartmanCrypto" (https://wiki.debian.org/DebianInstaller/PartmanCrypto). *Debian Wiki*. Retrieved 6 May 2022.
26. "LUKS On-Disk Format Specification" (https://gitlab.com/cryptsetup/cryptsetup/wikis/LUKS-standard/on-disk-format.pdf) (PDF).
27. "CRYPTSETUP-REENCRYPT(8) Man page" (https://man7.org/linux/man-pages/man8/cryptsetup-reencrypt.8.html). *man7.org*.

External links

- Official website (https://gitlab.com/cryptsetup/cryptsetup/blob/master/README.md)
- Frequently Asked Questions (FAQ) (https://gitlab.com/cryptsetup/cryptsetup/wikis/FrequentlyAskedQuestions)
- LibreCrypt: Implementation for Windows (https://github.com/t-d-k/librecrypt)
- LUKS1 Specification (https://gitlab.com/cryptsetup/cryptsetup/wikis/Specification)
- LUKS2 Specification (https://gitlab.com/cryptsetup/cryptsetup/blob/master/docs/on-disk-format-luks2.pdf)