Full credit is given to the above companies including the OS that this PDF file was generated!

Red Hat Enterprise Linux Release 9.2 Manual Pages on 'cryptsetup-luksFormat.8' command

$ man cryptsetup-luksFormat.8

CRYPTSETUP-LUKSFORMAT(8) Maintenance Commands CRYPTSETUP-LUKSFORMAT(8)

NAME

cryptsetup-luksFormat - initialize a LUKS partition and set the initial passphrase

SYNOPSIS

cryptsetup luksFormat [<options>] <device> [<key file>]

DESCRIPTION

Initializes a LUKS partition and sets the initial passphrase (for key-slot 0), either via prompting or via <key file>. Note that if the second argument is present, then the passphrase is taken from the file given there, without the need to use the --key-file option. Also note that for both forms of reading the passphrase from a file you can give '-' as file name, which results in the passphrase being read from stdin and the safety-question being skipped.

You cannot call luksFormat on a device or filesystem that is mapped or in use, e.g., mounted filesystem, used in LVM, active RAID member, etc. The device or filesystem has to be un-mounted in order to call luksFormat.

To use specific version of LUKS format, use --type luks1 or type luks2.

<options> can be [--hash, --cipher, --verify-passphrase, --key-size, --key-slot, --key-file (takes precedence over optional second argument), --keyfile-offset, --keyfile-size, --use-random, --use-urandom, --uuid, --volume-key-file, --iter-time, --header,

--pbkdf-force-iterations, --force-password, --disable-locks, --timeout,

--type, --offset, --align-payload (deprecated)].

For LUKS2, additional <options> can be [--integrity,

--integrity-no-wipe, --sector-size, --label, --subsystem, --pbkdf,

--pbkdf-memory, --pbkdf-parallel, --disable-locks, --disable-keyring,

--luks2-metadata-size, --luks2-keyslots-size, --keyslot-cipher,

--keyslot-key-size, --integrity-legacy-padding].

WARNING: Doing a luksFormat on an existing LUKS container will make all data in the old container permanently irretrievable unless you have a header backup.

OPTIONS

--type <device-type>

Specifies required device type, for more info read BASIC ACTIONS section in cryptsetup(8).

--hash, -h <hash-spec>

Specifies the hash used in the LUKS key setup scheme and volume key digest. The specified hash is used for PBKDF2 and AF splitter.

The hash algorithm must provide at least 160 bits of output. Do not use a non-crypto hash like xxhash as this breaks security. Use cryptsetup --help to show the defaults.

--cipher, -c <cipher-spec>

Set the cipher specification string.

cryptsetup --help shows the compiled-in defaults.

If a hash is part of the cipher specification, then it is used as part of the IV generation. For example, ESSIV needs a hash function, while "plain64" does not and hence none is specified.

For XTS mode you can optionally set a key size of 512 bits with the -s option. Key size for XTS mode is twice that for other modes for the same security level.

--verify-passphrase, -y

When interactively asking for a passphrase, ask for it twice and complain if both inputs do not match. Ignored on input from file or stdin.

--key-file, -d name

Read the passphrase from file.

If the name given is "-", then the passphrase will be read from stdin. In this case, reading will not stop at newline characters.

See section NOTES ON PASSPHRASE PROCESSING in cryptsetup(8) for more information.

--keyfile-offset value

Skip value bytes at the beginning of the key file.

--keyfile-size, -l value

Read a maximum of value bytes from the key file. The default is to read the whole file up to the compiled-in maximum that can be queried with --help. Supplying more data than the compiled-in maximum aborts the operation.

This option is useful to cut trailing newlines, for example. If --keyfile-offset is also given, the size count starts after the offset.

--volume-key-file, --master-key-file (OBSOLETE alias)

Use a volume key stored in a file. WARNING: If you create your own volume key, you need to make sure to do it right. Otherwise, you can end up with a low-entropy or otherwise partially predictable volume key which will compromise security.

--use-random, --use-urandom

For luksFormat these options define which kernel random number generator will be used to create the volume key (which is a long-term key).

See NOTES ON RANDOM NUMBER GENERATORS in cryptsetup(8) for more information. Use cryptsetup --help to show the compiled-in default random number generator.

WARNING: In a low-entropy situation (e.g. in an embedded system) and older kernels, both selections are problematic. Using /dev/urandom can lead to weak keys. Using /dev/random can block a long time, potentially forever, if not enough entropy can be harvested by the kernel.

--key-slot, -S <0-N>

For LUKS operations that add key material, this option allows you to specify which key slot is selected for the new key.

The maximum number of key slots depends on the LUKS version. LUKS1 can have up to 8 key slots. LUKS2 can have up to 32 key slots based on key slot area size and key size, but a valid key slot ID can always be between 0 and 31 for LUKS2.

--key-size, -s bits

Sets key size in bits. The argument has to be a multiple of 8. The possible key-sizes are limited by the cipher and mode used. See /proc/crypto for more information. Note that key-size in /proc/crypto is stated in bytes.

This option can be used for open --type plain or luksFormat. All other LUKS actions will use the key-size specified in the LUKS header. Use cryptsetup --help to show the compiled-in defaults.

--offset, -o <number of 512 byte sectors>

Start offset in the backend device in 512-byte sectors.

The --offset option sets the data offset (payload) of data device and must be aligned to 4096-byte sectors (must be multiple of 8).

This option cannot be combined with --align-payload option.

--pbkdf <PBKDF spec>

Set Password-Based Key Derivation Function (PBKDF) algorithm for LUKS keyslot. The PBKDF can be: pbkdf2 (for PBKDF2 according to RFC2898), argon2i for Argon2i or argon2id for Argon2id (see Argon2 <https://www.cryptolux.org/index.php/Argon2> for more info).

For LUKS1, only PBKDF2 is accepted (no need to use this option).

The default PBKDF for LUKS2 is set during compilation time and is available in cryptsetup --help output.

A PBKDF is used for increasing dictionary and brute-force attack cost for keyslot passwords. The parameters can be time, memory and parallel cost.

For PBKDF2, only time cost (number of iterations) applies. For Argon2i/id, there is also memory cost (memory required during the

process of key derivation) and parallel cost (number of threads that run in parallel during the key derivation.

Note that increasing memory cost also increases time, so the final parameter values are measured by a benchmark. The benchmark tries to find iteration time (--iter-time) with required memory cost --pbkdf-memory. If it is not possible, the memory cost is decreased as well. The parallel cost --pbkdf-parallel is constant and is checked against available CPU cores.

You can see all PBKDF parameters for particular LUKS2 keyslot with cryptsetup-luksDump(8) command.

NOTE: If you do not want to use benchmark and want to specify all parameters directly, use --pbkdf-force-iterations with --pbkdf-memory and --pbkdf-parallel. This will override the values without benchmarking. Note it can cause extremely long unlocking time. Use only in specific cases, for example, if you know that the formatted device will be used on some small embedded system.

MINIMAL AND MAXIMAL PBKDF COSTS: For PBKDF2, the minimum iteration count is 1000 and maximum is 4294967295 (maximum for 32bit unsigned integer). Memory and parallel costs are unused for PBKDF2. For Argon2i and Argon2id, minimum iteration count (CPU cost) is 4 and maximum is 4294967295 (maximum for 32bit unsigned integer). Minimum memory cost is 32 KiB and maximum is 4 GiB. (Limited by addressable memory on some CPU platforms.) If the memory cost parameter is benchmarked (not specified by a parameter) it is always in range from 64 MiB to 1 GiB. The parallel cost minimum is 1 and maximum 4 (if enough CPUs cores are available, otherwise it is decreased).

--iter-time, -i <number of milliseconds>

The number of milliseconds to spend with PBKDF passphrase processing. Specifying 0 as parameter selects the compiled-in default.

--pbkdf-memory <number>

Set the memory cost for PBKDF (for Argon2i/id the number represents kilobytes). Note that it is maximal value, PBKDF benchmark or

available physical memory can decrease it. This option is not available for PBKDF2.

--pbkdf-parallel <number>

Set the parallel cost for PBKDF (number of threads, up to 4). Note that it is maximal value, it is decreased automatically if CPU online count is lower. This option is not available for PBKDF2.

--pbkdf-force-iterations <num>

Avoid PBKDF benchmark and set time cost (iterations) directly. It can be used for LUKS/LUKS2 device only. See --pbkdf option for more info.

--progress-frequency seconds

Print separate line every seconds with wipe progress.

--progress-json

Prints progress data in JSON format suitable mostly for machine processing. It prints separate line every half second (or based on --progress-frequency value). The JSON output looks as follows during progress (except it?s compact single line):

```
{
  "device":"/dev/sda"       // backing device or file
  "device_bytes":"8192",    // bytes of I/O so far
  "device_size":"44040192", // total bytes of I/O to go
  "speed":"126877696",      // calculated speed in bytes per second (based on progress so far)
  "eta_ms":"2520012"        // estimated time to finish an operation in milliseconds
  "time_ms":"5561235"       // total time spent in IO operation in milliseconds
}
```

Note on numbers in JSON output: Due to JSON parsers limitations all numbers are represented in a string format due to need of full 64bit unsigned integers.

--timeout, -t <number of seconds>

The number of seconds to wait before timeout on passphrase input via terminal. It is relevant every time a passphrase is asked. It has no effect if used in conjunction with --key-file.

This option is useful when the system should not stall if the user

does not input a passphrase, e.g. during boot. The default is a value of 0 seconds, which means to wait forever.

--align-payload <number of 512 byte sectors>

Align payload at a boundary of value 512-byte sectors.

If not specified, cryptsetup tries to use the topology info provided by the kernel for the underlying device to get the optimal alignment. If not available (or the calculated value is a multiple of the default) data is by default aligned to a 1MiB boundary (i.e. 2048 512-byte sectors).

For a detached LUKS header, this option specifies the offset on the data device. See also the --header option.

WARNING: This option is DEPRECATED and has often unexpected impact to the data offset and keyslot area size (for LUKS2) due to the complex rounding. For fixed data device offset use --offset option instead.

--uuid <UUID>

Use the provided UUID for the luksFormat command instead of generating a new one. Changes the existing UUID when used with the luksUUID command.

The UUID must be provided in the standard UUID format, e.g. 12345678-1234-1234-1234-123456789abc.

--header <device or file storing the LUKS header>

Use a detached (separated) metadata device or file where the LUKS header is stored. This option allows one to store ciphertext and LUKS header on different devices.

With a file name as the argument to --header, the file will be automatically created if it does not exist. See the cryptsetup FAQ for header size calculation.

The --align-payload option is taken as absolute sector alignment on ciphertext device and can be zero.

--force-password

Do not use password quality checking for new LUKS passwords.

This option is ignored if cryptsetup is built without password

quality checking support.

For more info about password quality check, see the manual page for pwquality.conf(5) and passwdqc.conf(5).

--disable-locks

Disable lock protection for metadata on disk. This option is valid only for LUKS2 and ignored for other formats.

WARNING: Do not use this option unless you run cryptsetup in a restricted environment where locking is impossible to perform (where /run directory cannot be used).

--disable-keyring

Do not load volume key in kernel keyring and store it directly in the dm-crypt target instead. This option is supported only for the LUKS2 type.

--sector-size bytes

Set sector size for use with disk encryption. It must be power of two and in range 512 - 4096 bytes. This option is available only with LUKS2 format.

For LUKS2 devices it?s established based on parameters provided by underlying data device. For native 4K block devices it?s 4096 bytes. For 4K/512e (4K physical sector size with 512 bytes emulation) it?s 4096 bytes. For drives reporting only 512 bytes block size it remains 512 bytes. If data device is regular file put in filesystem it?s 4096 bytes.

Note that if sector size is higher than underlying device hardware sector and there is not integrity protection that uses data journal, using this option can increase risk on incomplete sector writes during a power fail.

If used together with --integrity option and dm-integrity journal, the atomicity of writes is guaranteed in all cases (but it cost write performance - data has to be written twice).

Increasing sector size from 512 bytes to 4096 bytes can provide better performance on most of the modern storage devices and also with some hw encryption accelerators.

--label <LABEL> --subsystem <SUBSYSTEM>

Set label and subsystem description for LUKS2 device. The label and subsystem are optional fields and can be later used in udev scripts for triggering user actions once the device marked by these labels is detected.

--integrity <integrity algorithm>

Specify integrity algorithm to be used for authenticated disk encryption in LUKS2.

WARNING: This extension is EXPERIMENTAL and requires dm-integrity kernel target (available since kernel version 4.12). For native AEAD modes, also enable "User-space interface for AEAD cipher algorithms" in "Cryptographic API" section (CONFIG_CRYPTO_USER_API_AEAD .config option).

For more info, see AUTHENTICATED DISK ENCRYPTION section in cryptsetup(8).

--integrity-legacy-padding

Use inefficient legacy padding.

WARNING: Do not use this option until you need compatibility with specific old kernel.

--luks2-metadata-size <size>

This option can be used to enlarge the LUKS2 metadata (JSON) area. The size includes 4096 bytes for binary metadata (usable JSON area is smaller of the binary area). According to LUKS2 specification, only these values are valid: 16, 32, 64, 128, 256, 512, 1024, 2048 and 4096 kB The <size> can be specified with unit suffix (for example 128k).

--luks2-keyslots-size <size>

This option can be used to set specific size of the LUKS2 binary keyslot area (key material is encrypted there). The value must be aligned to multiple of 4096 bytes with maximum size 128MB. The <size> can be specified with unit suffix (for example 128k).

--keyslot-cipher <cipher-spec>

This option can be used to set specific cipher encryption for the

LUKS2 keyslot area.

--keyslot-key-size <bits>

This option can be used to set specific key size for the LUKS2 keyslot area.

--integrity-no-wipe

Skip wiping of device authentication (integrity) tags. If you skip this step, sectors will report invalid integrity tag until an application write to the sector.

NOTE: Even some writes to the device can fail if the write is not aligned to page size and page-cache initiates read of a sector with invalid integrity tag.

--batch-mode, -q

Suppresses all confirmation questions. Use with care!

If the --verify-passphrase option is not specified, this option also switches off the passphrase verification.

--debug or --debug-json

Run in debug mode with full diagnostic logs. Debug output lines are always prefixed by #.

If --debug-json is used, additional LUKS2 JSON data structures are printed.

--version, -V

Show the program version.

--usage

Show short option help.

--help, -?

Show help text and default parameters. == REPORTING BUGS

Report bugs at cryptsetup mailing list <cryptsetup@lists.linux.dev> or in Issues project section <https://gitlab.com/cryptsetup/cryptsetup/-/issues/new>.

Please attach output of the failed command with --debug option added.

SEE ALSO

Cryptsetup FAQ <https://gitlab.com/cryptsetup/cryptsetup/wikis/FrequentlyAskedQuestions>

cryptsetup(8), integritysetup(8) and veritysetup(8)

CRYPTSETUP

Part of cryptsetup project <https://gitlab.com/cryptsetup/cryptsetup/>.

cryptsetup 2.6.0 2022-12-14 CRYPTSETUP-LUKSFORMAT(8)