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

Rocky Enterprise Linux 9.2 Manual Pages on command 'cryptsetup-luksAddKey.8'

$ man cryptsetup-luksAddKey.8

CRYPTSETUP-LUKSADDKEY(8)   Maintenance Commands   CRYPTSETUP-LUKSADDKEY(8)

NAME

cryptsetup-luksAddKey - add a new passphrase

SYNOPSIS

cryptsetup luksAddKey [<options>] <device> [<key file with new key>]

DESCRIPTION

Adds a keyslot protected by a new passphrase. An existing passphrase
must be supplied interactively, via --key-file or LUKS2 token (plugin).
Alternatively to existing passphrase user may pass directly volume key
(via --volume-key-file). The new passphrase to be added can be
specified interactively, read from the file given as the positional
argument (also via --new-keyfile parameter) or via LUKS2 token.

NOTE: with --unbound option the action creates new unbound LUKS2
keyslot. The keyslot cannot be used for device activation. If you don?t
pass new key via --volume-key-file option, new random key is generated.
Existing passphrase for any active keyslot is not required.

NOTE: some parameters are effective only if used with LUKS2 format that
supports per-keyslot parameters. For LUKS1, PBKDF type and hash

algorithm is always the same for all keyslots.

<options> can be [--key-file, --keyfile-offset, --keyfile-size,

--new-keyfile, --new-keyfile-offset, --new-keyfile-size, --key-slot,

--new-key-slot, --volume-key-file, --force-password, --hash, --header,

--disable-locks, --iter-time, --pbkdf, --pbkdf-force-iterations,

--pbkdf-memory, --pbkdf-parallel, --unbound, --type, --keyslot-cipher,

--keyslot-key-size, --key-size, --timeout, --token-id, --token-type,

--token-only, --new-token-id, --verify-passphrase].

OPTIONS

--type <device-type>

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

--hash, -h <hash-spec>

The specified hash is used for PBKDF2 and AF splitter.

--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.

The passphrase supplied via --key-file is always the passphrase for existing keyslot requested by the command.

If you want to set a new passphrase via key file, you have to use a positional argument or parameter --new-keyfile.

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.

--new-keyfile name

Read the passphrase for a new keyslot 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.

This is alternative method to positional argument when adding new
passphrase via kefile.

--new-keyfile-offset value

Skip value bytes at the start when adding a new passphrase from key
file.

--new-keyfile-size value

Read a maximum of value bytes when adding a new passphrase from key
file. The default is to read the whole file up to the compiled-in
maximum length that can be queried with --help. Supplying more than
the compiled in maximum aborts the operation. When
--new-keyfile-offset is also given, reading starts after the
offset.

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

Use a volume key stored in a file. This allows adding a new keyslot
without having to know passphrase to existing one. It may be also
used when no keyslot is active.

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.

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

When used together with parameter --new-key-slot this option allows
you to specify which key slot is selected for unlocking volume key.

NOTE: This option is ignored if existing volume key gets unlocked via LUKS2 token (--token-id, --token-type or --token-only

parameters) or when volume key is provided directly via

--volume-key-file parameter.

NOTE: To maintain backward compatibility, without --new-key-slot

parameter, 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.

--new-key-slot <0-N>

This option allows you to specify which key slot is selected for

the new key.

NOTE: When used this option affects --key-slot option.

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

Provide volume key size in bits. The argument has to be a multiple

of 8.

This option is required when parameter --volume-key-file is used to

provide current volume key. Also, it is used when new unbound

keyslot is created by specifying --unbound parameter.

--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.

--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.

--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.

For commands that change the LUKS header (e.g. luksAddKey), specify the device or file with the LUKS header directly as the LUKS device.

--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).

--token-id

Specify what token to use when unlocking existing keyslot to get volume key.

--new-token-id

Specify what token to use to get the passphrase for a new keyslot.

--token-only

Use only LUKS2 tokens to unlock existing volume key.

NOTE: To create a new keyslot using passphrase provided by a token use --new-token-id parameter.

--token-type type

Specify what token type (all type tokens) to use when unlocking existing keyslot to get volume key.

--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.

--unbound

Creates new LUKS2 unbound keyslot.

--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.

EXAMPLES

NOTE: When not specified otherwise interactive passphrase prompt is always default method.

Add new keyslot using interactive passphrase prompt for both existing and new passphrase:

```
cryptsetup luksAddKey /dev/device
```

Add new keyslot using LUKS2 tokens to unlock existing keyslot with interactive passphrase prompt for new passphrase:

```
cryptsetup luksAddKey --token-only /dev/device
```

Add new keyslot using LUKS2 systemd-tpm2 tokens to unlock existing keyslot with interactive passphrase prompt for new passphrase (systemd-tpm2 token plugin must be available):

```
cryptsetup luksAddKey --token-type systemd-tpm2 /dev/device
```

Add new keyslot using interactive passphrase prompt for existing keyslot, reading new passphrase from key_file:

cryptsetup luksAddKey --new-keyfile key_file /dev/device or cryptsetup luksAddKey /dev/device key_file

Add new keyslot using volume stored in volume_key_file and LUKS2 token in slot 5 to get new keyslot passphrase (token in slot 5 must exist and respective token plugin must be available):

```
cryptsetup luksAddKey --volume-key-file volume_key_file --new-token-id 5 /dev/device
```

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/>.