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

$ man cryptsetup-reencrypt.8

CRYPTSETUP-REENCRYPT(8) Maintenance Commands CRYPTSETUP-REENCRYPT(8)

NAME

cryptsetup-reencrypt - reencrypt LUKS encrypted volumes in-place

SYNOPSIS

cryptsetup reencrypt [<options>] <device> or --active-name <name>

[<new_name>]

DESCRIPTION

Run LUKS device reencryption.

There are 3 basic modes of operation:

? device reencryption (reencrypt)

? device encryption (reencrypt --encrypt/--new/-N)

? device decryption (reencrypt --decrypt)

<device> or --active-name <name> (LUKS2 only) is mandatory parameter.

Cryptsetup reencrypt action can be used to change reencryption

parameters which otherwise require full on-disk data change

(re-encryption). The reencrypt action reencrypts data on LUKS device

in-place.

You can regenerate volume key (the real key used in on-disk encryption

unclocked by passphrase), cipher, cipher mode or encryption sector size (LUKS2 only).

Reencryption process may be safely interrupted by a user via SIGINT signal (ctrl+c). Same applies to SIGTERM signal (i.e. issued by systemd during system shutdown).

For in-place encryption mode, the reencrypt action additionally takes all options available for luksFormat action for respective LUKS version (see cryptsetup-luksFormat man page for more details). See cryptsetup-luksFormat(8).

NOTE that for encrypt and decrypt mode, the whole device must be treated as unencrypted ? there are no quarantees of confidentiality as part of the device contains plaintext.

ALWAYS BE SURE YOU HAVE RELIABLE BACKUP BEFORE USING THIS ACTION ON LUKS DEVICE.

<options> can be [--batch-mode, --block-size, --cipher, --debug, --debug-json, --decrypt, --device-size, --disable-locks, --encrypt, --force-offline-reencrypt, --hash, --header, --hotzone-size, --iter-time, --init-only, --keep-key, --key-file, --key-size, --key-slot, --keyfile-offset, --keyfile-size, --tries, --timeout, --pbkdf, --pbkdf-force-iterations, --pbkdf-memory, --pbkdf-parallel, --progress-frequency, --progress-json, --reduce-device-size, --resilience, --resilience-hash, --resume-only, --sector-size, --use-directio, --use-random, --use-urandom, --use-fsync, --uuid, --verbose, --volume-key-file, --write-log].

LUKS2 REENCRYPTION

With <device> parameter cryptsetup looks up active <device> dm mapping. If no active mapping is detected, it starts offline LUKS2 reencryption otherwise online reencryption takes place.

To resume already initialized or interrupted reencryption, just run the cryptsetup reencrypt command again to continue the reencryption operation. Reencryption may be resumed with different --resilience or --hotzone-size unless implicit datashift resilience mode is used: either encrypt mode with --reduce-device-size option or decrypt mode

with original LUKS2 header exported in --header file.

If the reencryption process was interrupted abruptly (reencryption process crash, system crash, poweroff) it may require recovery. The recovery is currently run automatically on next activation (action open) when needed or explicitly by user (action repair).

Optional parameter <new_name> takes effect only with encrypt option and it activates device <new_name> immediately after encryption initialization gets finished. That?s useful when device needs to be ready as soon as possible and mounted (used) before full data area encryption is completed.

LUKS1 REENCRYPTION

Current working directory must be writable and temporary files created during reencryption must be present. During reencryption process the LUKS1 device is marked unavailable and must be offline (no dm-crypt mapping or mounted filesystem).

WARNING: The LUKS1 reencryption code is not resistant to hardware or kernel failures during reencryption (you can lose your data in this case).

OPTIONS

--block-size value (LUKS1 only)

Use re-encryption block size of value in MiB.

Values can be between 1 and 64 MiB.

--use-directio (LUKS1 only)

Use direct-io (O_DIRECT) for all read/write data operations related to block device undergoing reencryption.

Useful if direct-io operations perform better than normal buffered operations (e.g. in virtual environments).

--use-fsync (LUKS1 only)

Use fsync call after every written block. This applies for reencryption log files as well.

--write-log (LUKS1 only)

Update log file after every block write. This can slow down reencryption but will minimize data loss in the case of system

crash.

--type <device-type>

Specifies required (encryption mode) or expected (other modes) LUKS format. Accepts only luks1 or luks2.

--hash, -h <hash-spec>

LUKS1: Specifies the hash used in the LUKS1 key setup scheme and volume key digest.

NOTE: if this parameter is not specified, default hash algorithm is always used for new LUKS1 device header.

LUKS2: Ignored unless new keyslot pbkdf algorithm is set to PBKDF2 (see --pbkdf).

--cipher, -c <cipher-spec>

LUKS2: Set the cipher specification string for data segment only.

LUKS1: Set the cipher specification string for data segment and keyslots.

NOTE: In encrypt mode, if cipher specification is omitted the default cipher is applied. In reencrypt mode, if no new cipher specification is requested, the existing cipher will remain in use. Unless the existing cipher was "cipher_null". In that case default cipher would be applied as in encrypt mode.

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.

WARNING: --key-file option can be used only if there is only one active keyslot, or alternatively, also if --key-slot option is specified (then all other keyslots will be disabled in new LUKS device).

If this option is not used, cryptsetup will ask for all active keyslot passphrases.

--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 (set) new 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

Define which kernel random number generator will be used to create the volume key.

--keep-key

LUKS2: Do not change effective volume key and change other parameters provided it is requested.

LUKS1: Reencrypt only the LUKS1 header and keyslots. Skips data in-place reencryption.

--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.

For reencryption mode it selects specific keyslot (and passphrase) that can be used to unlock new volume key. If used all other keyslots get removed after reencryption operation is finished.

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.

LUKS1: If you are increasing key size, there must be enough space in the LUKS header for enlarged keyslots (data offset must be large enough) or reencryption cannot be performed.

If there is not enough space for keyslots with new key size, you can destructively shrink device with --reduce-device-size option.

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

Start offset in the backend device in 512-byte sectors. This option is only relevant for the encrypt mode.

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.

--device-size size[units]

Instead of real device size, use specified value. It means that only specified area (from the start of the device to the specified size) will be reencrypted.

WARNING: This is destructive operation. Data beyond --device-size limit may be lost after operation gets finished.

If no unit suffix is specified, the size is in bytes.

Unit suffix can be S for 512 byte sectors, K/M/G/T (or

KiB,MiB,GiB,TiB) for units with 1024 base or KB/MB/GB/TB for 1000 base (SI scale).

--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 for the new LUKS header.

--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 reencryption 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.

--tries, -T

How often the input of the passphrase shall be retried. The default is 3 tries.

--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>

When used in encryption mode use the provided UUID for the new LUKS header instead of generating a new one.

LUKS1 (only in decryption mode): To find out what UUID to pass look for temporary files LUKS-UUID.[|log|org|new] of the interrupted decryption process.

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.

If used with --encrypt/--new option, the header file will be created (or overwritten). Use with care.

LUKS2: For decryption mode the option may be used to export original LUKS2 header to a detached file. The passed future file must not exist at the time of initializing the decryption operation. This frees space in head of data device so that data can be moved at original LUKS2 header location. Later on decryption operation continues as if the ordinary detached header was passed.

WARNING: Never put exported header file in a filesystem on top of device you are about to decrypt! It would cause a deadlock.

--force-offline-reencrypt (LUKS2 only)

Bypass active device auto-detection and enforce offline reencryption.

This option is useful especially for reencryption of LUKS2 images put in files (auto-detection is not reliable in this scenario).

It may also help in case active device auto-detection on particular data device does not work or report errors.

WARNING: Use with extreme caution! This may destroy data if the device is activated and/or actively used.

--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.

NOTE: With locking disabled LUKS2 images in files can be fully (re)encrypted offline without need for super user privileges provided used block ciphers are available in crypto backend.

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 (LUKS2 only)

Reencrypt device with new encryption sector size enforced.

WARNING: Increasing encryption sector size may break hosted filesystem. Do not run reencryption with --force-offline-reencrypt if unsure what block size was filesystem formatted with.

--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.

--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.

--encrypt, --new, -N

Initialize (and run) device in-place encryption mode.

--decrypt

Initialize (and run) device decryption mode.

--init-only (LUKS2 only)

Initialize reencryption (any mode) operation in LUKS2 metadata only and exit. If any reencrypt operation is already initialized in metadata, the command with --init-only parameter fails.

--resume-only (LUKS2 only)

Resume reencryption (any mode) operation already described in LUKS2 metadata. If no reencrypt operation is initialized, the command with --resume-only parameter fails. Useful for resuming reencrypt operation without accidentally triggering new reencryption operation.

--resilience mode (LUKS2 only)

Reencryption resilience mode can be one of checksum, journal or none.

checksum: default mode, where individual checksums of ciphertext hotzone sectors are stored, so the recovery process can detect which sectors were already reencrypted. It requires that the device sector write is atomic.

journal: the hotzone is journaled in the binary area (so the data

are written twice).

none: performance mode. There is no protection and the only way it?s safe to interrupt the reencryption is similar to old offline reencryption utility.

Resilience modes can be changed unless datashift mode is used for operation initialization (encryption with --reduce-device-size option)

--resilience-hash hash (LUKS2 only)

The hash algorithm used with "--resilience checksum" only. The default hash is sha256. With other resilience modes, the hash parameter is ignored.

--hotzone-size size (LUKS2 only)

This option can be used to set an upper limit on the size of reencryption area (hotzone). The size can be specified with unit suffix (for example 50M). Note that actual hotzone size may be less than specified <size> due to other limitations (free space in keyslots area or available memory).

With decryption mode for devices with LUKS2 header placed in head of data device, the option specifies how large is the first data segment moved from original data offset pointer.

--reduce-device-size size

This means that last size sectors on the original device will be lost, data will be effectively shifted by specified number of sectors.

It could be useful if you added some space to underlying partition or logical volume (so last size sectors contains no data).

For units suffix see --device-size parameter description.

WARNING: This is a destructive operation and cannot be reverted. Use with extreme care - accidentally overwritten filesystems are usually unrecoverable.

LUKS2: Initialize LUKS2 reencryption with data device size reduction (currently only encryption mode is supported).

Recommended minimal size is twice the default LUKS2 header size

(--reduce-device-size 32M) for encryption mode.

LUKS1: Enlarge data offset to specified value by shrinking device size.

You cannot shrink device more than by 64 MiB (131072 sectors).

--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: You may drop --type luks2 option as long as LUKS2 format is default.

LUKS2 ENCRYPTION EXAMPLES

Encrypt LUKS2 device (in-place). Make sure last 32 MiB on /dev/plaintext is unused (e.g.: does not contain filesystem data):

cryptsetup reencrypt --encrypt --type luks2 --reduce-device-size 32m /dev/plaintext_device

Encrypt LUKS2 device (in-place) with detached header put in a file:

cryptsetup reencrypt --encrypt --type luks2 --header my_luks2_header /dev/plaintext_device

Initialize LUKS2 in-place encryption operation only and activate the device (not yet encrypted):

cryptsetup reencrypt --encrypt --type luks2 --init-only

--reduce-device-size 32m /dev/plaintext_device my_future_luks_device

Resume online encryption on device initialized in example above:

cryptsetup reencrypt --resume-only /dev/plaintext_device or cryptsetup reencrypt --active-name my_future_luks_device

LUKS2 REENCRYPTION EXAMPLES

Reencrypt LUKS2 device (refresh volume key only):

cryptsetup reencrypt /dev/encrypted_device

LUKS2 DECRYPTION EXAMPLES

Decrypt LUKS2 device with header put in head of data device (header file does not exist):

cryptsetup reencrypt --decrypt --header /export/header/to/file /dev/encrypted_device

Decrypt LUKS2 device with detached header (header file exists):

cryptsetup reencrypt --decrypt --header detached-luks2-header /dev/encrypted_device

Resume interrupted LUKS2 decryption:

cryptsetup reencrypt --resume-only --header luks2-hdr-file /dev/encrypted_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/>.

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