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 the LUKS device in−place.

You can regenerate **volume key** (the real key used in on−disk encryption unlocked by passphrase), **cipher**, **cipher mode** or **encryption sector size** (LUKS2 only).

If you need to use both luksChangeKey and reencrypt (e.g., to recover from a leak), you need to use them in that order to avoid leaking the new volume key.

The reencryption process may be safely interrupted by a user via SIGINT signal (ctrl+c). The same applies to the SIGTERM signal (i.e., issued by systemd during system shutdown).

For in−place encryption mode, the *reencrypt* action additionally takes all options available for the *luksFormat* action for the 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 guarantees 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 the <device> parameter, cryptsetup looks up the active <device> dm mapping. If no active mapping is detected, it starts offline LUKS2 reencryption; otherwise, online reencryption occurs.

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, or power off), it may require recovery. The recovery is run automatically on next activation (action *open*) when needed or explicitly by the user (action *repair*).

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

LUKS1 REENCRYPTION

The current working directory must be writable, and temporary files created during reencryption must be present. During reencryption, 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

−−align−payload <*number of 512 byte sectors*> (DEPRECATED, use −−offset)
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.

This option is DEPRECATED and has an unexpected impact on the data offset and keyslot area size (for LUKS2) due to the complex rounding. For fixed data device offset, use −−offset option instead.

−−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.

−−block−size *value* (LUKS1 only)
Use re−encryption block size of *value* in MiB.

Values can be between 1 and 64 MiB.

−−cipher, **−c** <*cipher−spec*>
LUKS2: Set the cipher specification string for the data segment only.

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

The default cipher is applied if the cipher specification is omitted in encrypt mode.

In reencrypt mode, if no new cipher specification is requested, the existing cipher will remain. The only exception is if the cipher is "cipher_null", then the default cipher is used.

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.

−−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.

−−decrypt
Initialize (and run) device decryption mode.

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

LUKS2: When used together with −−reduce−device−size, only the initial *size* value (−−device−size parameter) of data is shifted backwards while being encrypted.

The sum of −−device−size and −−reduce−device−size values must not exceed the real device size.

WARNING: This is a destructive operation. Data beyond −−device−size limit may be lost after the operation is 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).

−−disable−blkid
Disable use of the blkid library for checking and wiping on−disk signatures.

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

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

With locking disabled, LUKS2 images in files can be fully (re)encrypted offline without the need for superuser privileges provided that the used block ciphers are available in the 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).

−−encrypt, **−−new**, **−N**
Initialize (and run) the device in−place encryption mode.

−−force−no−keyslots (LUKS2 only)
Enforce initialization of reencryption operation with additional −−volume−key−file, −−new−volume−key−file, −−volume−key−keyring or −−new−volume−key−keyring parameters. It would result in the deletion of all remaining LUKS2 keyslots containing the volume key.

LUKS2 keyslot with the new volume key may be added after the reencryption operation is finished.

See **cryptsetup−luksAddKey**(8) command.

WARNING: Use with extreme caution! If you lose the volume key stored in a file or in a kernel keyring before adding the LUKS2 keyslot containing the new volume key, the device will become unusable, and all data will be lost.

−−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 a 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).

−−hash, **−h** <*hash−spec*>
LUKS1: Specifies the hash used in the LUKS1 key setup scheme and volume key digest.

If this parameter is not specified, the default hash algorithm is always used for a new LUKS1 device header.

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

−−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 the 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 the 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 the head of the data device so that data can be moved at the original LUKS2 header location. Later on, the decryption operation continues as if the ordinary detached header was passed.

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

−−help, **−?**
Show help text and default parameters.

−−hotzone−size *size* (LUKS2 only)
This option can be used to set an upper limit on the size of the reencryption area (hotzone). The *size* can be specified with a unit suffix (for example, 50M). Note that the 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 the head of the data device, the option specifies how large is the first data segment moved from the original data offset pointer.

−−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.

−−iter−time, **−i** <*number of milliseconds*>
The number of milliseconds to spend with PBKDF passphrase processing for the new LUKS header.

−−keep−key
LUKS2: Do not change the effective volume key, and change other parameters if requested.

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

−−key−file, **−d** *file*
Read the passphrase from the 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 −−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 the 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.

−−key−size, **−s** *bits*
LUKS2: Provide current key size in *bits*. The argument has to be a multiple of 8. Useful when specifying the size of the current volume key when no keyslot is active.

LUKS1: See −−new−key−size.

−−key−slot, **−S** <*0−N*>
For LUKS operations that add key material, this option allows you to specify which keyslot is selected for the new key.

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

The maximum number of keyslots depends on the LUKS version. LUKS1 can have up to 8 keyslots. LUKS2 can have up to 32 keyslots based on keyslot area size and key size, but a valid keyslot ID can

always be between 0 and 31 for LUKS2.

−−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 a specific key size for the LUKS2 keyslot area.

−−label <*label*>, **−−subsystem** <*subsystem*>
Set label and subsystem description for LUKS2 device. These are similar to filesystem labels. The label and subsystem are optional fields and can be later used in udev scripts to trigger user actions once the device marked by these labels is detected.

−−luks2−keyslots−size *size*
This option can be used to set a specific size of the LUKS2 binary keyslot area (key material is encrypted there). The value must be aligned to a multiple of 4096 bytes with a maximum size 128MB. The <size> can be specified with a unit suffix (for example, 128k).

−−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 the LUKS2 specification, only these values are valid: 16, 32, 64, 128, 256, 512, 1024, 2048 and 4096 kB. The <size> can be specified with a unit suffix (for example, 128k).

−−new−key−size *bits*
Sets new key size in *bits*. The argument has to be a multiple of 8. The possible key sizes are limited by the new cipher and mode used in reencryption.

See /proc/crypto for more information. Note that the 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 the new key size, you can destructively shrink the device with −−reduce−device−size option.

−−new−volume−key−file *file*
Use (set) the new volume key stored in a file. The option must be paired with −−new−key−size parameter when initializing the reencryption operation.

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.

−−new−volume−key−keyring <*key description*>
Use (set) the new volume key stored in a keyring.

The size of the key stored in a keyring must be compatible with the new cipher used in the reencryption operation. See /proc/crypto for more information. Note that the key size in /proc/crypto is stated in bytes.

The <*key description*> uses keyctl−compatible syntax. This can either be a numeric key ID or a string name in the format *%<key type>:<key name>*. See also the **KEY IDENTIFIERS** section of **keyctl**(1). When no *%<key type>:* prefix is specified, we assume the key type is *user* (default type).

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.

−−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 the data device and must be aligned to 4096−byte sectors (must be a 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 the *cryptsetup −−help* output.

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

For PBKDF2, only the 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 a particular LUKS2 keyslot with the **cryptsetup−luksDump**(8) command.

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 or cause out−of−memory conditions with unconditional process termination. 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 the maximum is 4294967295 (maximum for 32−bit unsigned integer). Memory and parallel costs are not supported for PBKDF2. For **Argon2i** and **Argon2id**, the minimum iteration count (CPU cost) is 4, and the maximum is 4294967295 (maximum for a 32−bit unsigned integer). Minimum memory cost is 32 KiB and maximum is 4 GiB. If the memory cost parameter is benchmarked (not specified by a parameter), it is always in the range from 64 MiB to 1 GiB. Memory cost above 1GiB (up to the 4GiB maximum) can be setup only by the −−pbkdf−memory parameter. The parallel cost minimum is 1 and maximum 4 (if enough CPU cores are available, otherwise it is decreased by the available CPU cores).

WARNING: Increasing PBKDF computational costs above the mentioned limits provides negligible additional security improvement. While elevated costs significantly increase brute−force overhead, they offer negligible protection against dictionary attacks. The marginal cost increase for processing an entire dictionary remains fundamentally insufficient.

The hardcoded PBKDF limits represent engineered trade−offs between cryptographic security and operational usability. LUKS maintains portability and must be used within a reasonable time on resource−constrained systems.

Cryptsetup deliberately restricts maximum memory cost (4 GiB) and parallel cost (4) parameters due to architectural limitations (like embedded and legacy systems).

PBKDF memory cost mandates actual physical RAM allocation with intensive write operations that must remain in physical RAM. Any swap usage results in unacceptable performance degradation. Memory management often overcommits allocations beyond available physical memory, expecting most allocated memory to remain unused. In such situations, as PBKDF always uses all allocated memory, it frequently causes out−of−memory failures that abort cryptsetup operations.

−−pbkdf−force−iterations *number*
Avoid the PBKDF benchmark and set the time cost (iterations) directly. It can be used only for a LUKS/LUKS2 device. See −−pbkdf option for more info.

−−pbkdf−memory *number*
Set the memory cost for PBKDF (for Argon2i/id, the number represents kilobytes). Note that it is the 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 the maximal value; it is decreased automatically if the CPU online count is lower. This option is not available for PBKDF2.

−−progress−frequency *seconds*
Print a separate line every *seconds* with reencryption progress.

−−progress−json
Prints progress data in JSON format, which is suitable mostly for machine processing. It prints a separate line every half second (or based on −−progress−frequency value). The JSON output looks as follows during progress (except it’s a 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 pro
  "eta_ms":"2520012",       // estimated time to finish an operation in millisecc
  "time_ms":"5561235"       // total time spent in IO operation in milliseconds
}
```

Note on numbers in JSON output: Due to JSON parser limitations, all numbers are represented in a string format due to the need for full 64−bit unsigned integers.

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

It could be useful if you added some space to the underlying partition or logical volume (so the ast *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). The last *size* sectors on the original plaintext device is used for temporarily storing the original first data segment. The former first data segment is replaced with LUKS2 header (half the *size* value), and plaintext data is shifted backwards (again half the *size* value) while being encrypted.

The recommended minimum size is twice the default LUKS2 header size (−−reduce−device−size 32M) for encryption mode.

The sum of −−device−size and −−reduce−device−size values must not exceed the real device size.

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

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

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

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

−−sector−size *bytes* (LUKS2 only)
Reencrypt the device with a new encryption sector size enforced.

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

Note that using a sector size larger than the underlying storage device's physical sector size may result in data corruption during unexpected power failures. A power failure during write operations may result in only partial completion of the encryption sector write, leaving encrypted data in an inconsistent state that cannot be properly decrypted.

−−timeout, **−t** *seconds*

The number of seconds to wait before a 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.

−−token−id
LUKS2 reencryption initialization: Specify what keyslots (associated with the selected token) to use for LUKS2 reencryption. If the reencryption operation changes the effective volume key, only keyslots associated with the token and unlocked successfully will be available after the reencryption operation is finished.

LUKS2 reencryption resume: Specify what token to use and allow the token PIN prompt to take precedence over the interactive keyslot passphrase prompt. If omitted, all available tokens (not protected by PIN) will be checked before proceeding further with the passphrase prompt.

−−token−only
LUKS2 reencryption initialization: Specify that all keyslots associated with any token will be used for LUKS2 reencryption. If the reencryption operation changes the effective volume key, only keyslots associated with any token will be available after the reencryption operation is finished.

LUKS2 reencryption resume: Do not proceed further with the action if the token−based keyslot unlock failed. Without the option, the action asks for a passphrase to proceed further.

It allows LUKS2 tokens protected by PIN to take precedence over the interactive keyslot passphrase prompt.

−−token−type *type*
LUKS2 reencryption initialization: Specify what keyslots (associated with the selected token type) to use for LUKS2 reencryption. If the reencryption operation changes the effective volume key, only keyslots associated with the token type and unlocked successfully will be available after the reencryption operation is finished.

LUKS2 reencryption resume: Restrict tokens eligible for operation to a specific token *type*. Mostly useful when no −−token−id is specified.

It allows LUKS2 *type* tokens protected by PIN to take precedence over the interactive keyslot passphrase prompt.

−−tries, **−T**
How often the input of the passphrase shall be retried. The default is 3 tries.

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

−−usage
Show short option help.

−−use−directio (LUKS1 only)
Use direct−io (O_DIRECT) for all read/write data operations related to the 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 the fsync call after every written block. This applies to reencryption log files as well.

−−use−random, **−−use−urandom**
Define which kernel random number generator will be used to create the volume key.

−−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.

−−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.

−−version, **−V**
Show the program version.

−−volume−key−file *file*, −−master−key−file file (OBSOLETE alias)
LUKS2: Provides the current volume key stored in a file. It can be used to reencrypt the device with no active keyslot together with −−new−volume−key−file or −−new−volume−key−keyring options.

LUKS1: See −−new−volume−key−file.

−−write−log (LUKS1 only)
Update the log file after every block is written. This can slow down reencryption, but it will minimize data loss in the case of a system crash.

EXAMPLES

You may drop **−−type luks2** option as long as LUKS2 format is default.

LUKS2 ENCRYPTION EXAMPLES

Encrypt LUKS2 device (in−place). Make sure the 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). Only the initial 1 GiB of original */dev/plaintext* data is encrypted while being shifted backwards. Make sure the last 32 MiB (tail) on the data device is unused (e.g., does not contain any data):

cryptsetup reencrypt −−encrypt −−type luks2 −−device−size 1g −−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 the device initialized in the 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

Reencrypt LUKS2 device using keyslot(s) associated with the token 3. All other keyslots will be removed after the reencryption finishes.

cryptsetup reencrypt −−token−id 3 /dev/encrypted_device

Reencrypt LUKS2 device using keyslots associated with all 'systemd−tpm2' tokens. All other keyslots will be removed after the reencryption finishes.

cryptsetup reencrypt −−token−type systemd−tpm2 /dev/encrypted_device

LUKS2 DECRYPTION EXAMPLES

Decrypt LUKS2 device with header put in the head of the 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 the 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/>.