

**NAME**

veritysetup - manage dm-verity (block level verification) volumes

**SYNOPSIS**

**veritysetup** <options> <action> <action args>

**DESCRIPTION**

Veritysetup is used to configure dm-verity managed device-mapper mappings.

Device-mapper verity target provides read-only transparent integrity checking of block devices using kernel crypto API.

The dm-verity devices are always read-only.

Veritysetup supports these operations:

*format* <data\_device> <hash\_device>

Calculates and permanently stores hash verification data for data\_device. Hash area can be located on the same device after data if specified by `--hash-offset` option.

Note you need to provide root hash string for device verification or activation. Root hash must be trusted.

The data or hash device argument can be block device or file image. If hash device path doesn't exist, it will be created as file.

<options> can be [`--hash`, `--no-superblock`, `--format`, `--data-block-size`, `--hash-block-size`, `--data-blocks`, `--hash-offset`, `--salt`, `--uuid`]

*open* <data\_device> <name> <hash\_device> <root\_hash> *create* <name> <data\_device> <hash\_device> <root\_hash>

Creates a mapping with <name> backed by device <data\_device> and using <hash\_device> for in-kernel verification.

The <root\_hash> is a hexadecimal string.

<options> can be [`--hash-offset`, `--no-superblock`, `--ignore-corruption` or `--restart-on-corruption`, `--ignore-zero-blocks`, `--check-at-most-once`]

If option `--no-superblock` is used, you have to use as the same options as in initial format operation.

*verify* <data\_device> <hash\_device> <root\_hash>

Verifies data on data\_device with use of hash blocks stored on hash\_device.

This command performs userspace verification, no kernel device is created.

The <root\_hash> is a hexadecimal string.

<options> can be [`--hash-offset`, `--no-superblock`]

If option `--no-superblock` is used, you have to use as the same options as in initial format operation.

*close* <name>

Removes existing mapping <name>.

For backward compatibility there is **remove** command alias for **close** command.

*status* <name>

Reports status for the active verity mapping <name>.

*dump* <hash\_device>

Reports parameters of verity device from on-disk stored superblock.

<options> can be [--no-superblock]

## OPTIONS

**--verbose, -v**

Print more information on command execution.

**--debug**

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

**--no-superblock**

Create or use dm-verity without permanent on-disk superblock.

**--format=number**

Specifies the hash version type. Format type 0 is original Chrome OS version. Format type 1 is current version.

**--data-block-size=bytes**

Used block size for the data device. (Note kernel supports only page-size as maximum here.)

**--hash-block-size=bytes**

Used block size for the hash device. (Note kernel supports only page-size as maximum here.)

**--data-blocks=blocks**

Size of data device used in verification. If not specified, the whole device is used.

**--hash-offset=bytes**

Offset of hash area/superblock on hash\_device. Value must be aligned to disk sector offset.

**--salt=hex string**

Salt used for format or verification. Format is a hexadecimal string.

**--uuid=UUID**

Use the provided UUID for format command instead of generating new one.

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

**--ignore-corruption, --restart-on-corruption**

Defines what to do if data integrity problem is detected (data corruption).

Without these options kernel fails the IO operation with I/O error. With **--ignore-corruption** option the corruption is only logged. With **--restart-on-corruption** the kernel is restarted immediately. (You have to provide way how to avoid restart loops.)

**WARNING:** Use these options only for very specific cases. These options are available since Linux kernel version 4.1.

**--ignore-zero-blocks**

Instruct kernel to not verify blocks that are expected to contain zeroes and always directly return zeroes instead.

**WARNING:** Use this option only in very specific cases. This option is available since Linux

kernel version 4.5.

**--check-at-most-once**

Instruct kernel to verify blocks only the first time they are read from the data device, rather than every time.

**WARNING:** It provides a reduced level of security because only offline tampering of the data device's content will be detected, not online tampering. This option is available since Linux kernel version 4.17.

**--hash=hash**

Hash algorithm for dm-verity. For default see --help option.

**--version**

Show the program version.

**--fec-device=fec\_device**

Use forward error correction (FEC) to recover from corruption if hash verification fails. Use encoding data from the specified device.

The fec device argument can be block device or file image. For format, if fec device path doesn't exist, it will be created as file.

Note: block sizes for data and hash devices must match. Also, if the verity data\_device is encrypted the fec\_device should be too.

**--fec-offset=bytes**

This is the offset, in bytes, from the start of the FEC device to the beginning of the encoding data.

**--fec-roots=num**

Number of generator roots. This equals to the number of parity bytes in the encoding data. In RS(M, N) encoding, the number of roots is M-N. M is 255 and M-N is between 2 and 24 (including).

**RETURN CODES**

Veritysetup returns 0 on success and a non-zero value on error.

Error codes are:

- 1 wrong parameters
- 2 no permission
- 3 out of memory
- 4 wrong device specified
- 5 device already exists or device is busy.

**EXAMPLES**

**veritysetup --data-blocks=256 format <data\_device> <hash\_device>**

Calculates and stores verification data on hash\_device for the first 256 blocks (of block-size). If hash\_device does not exist, it is created (as file image).

**veritysetup format <data\_device> <hash\_device>**

Calculates and stores verification data on hash\_device for the whole data\_device.

**veritysetup --data-blocks=256 --hash-offset=1052672 format <device> <device>**

Verification data (hashes) is stored on the same device as data (starting at hash-offset). Hash-offset must be greater than number of blocks in data-area.

```
veritysetup --data-blocks=256 --hash-offset=1052672 create test-device <device> <device>  
<root_hash>
```

Activates the verity device named test-device. Options `--data-blocks` and `--hash-offset` are the same as in the format command. The `<root_hash>` was calculated in format command.

```
veritysetup --data-blocks=256 --hash-offset=1052672 verify <data_device> <hash_device>  
<root_hash>
```

Verifies device without activation (in userspace).

```
veritysetup --fec-device=<fec_device> --fec-roots=10 format <data_device> <hash_device>
```

Calculates and stores verification and encoding data for `data_device`.

## REPORTING BUGS

Report bugs, including ones in the documentation, on the cryptsetup mailing list at `<dm-crypt@saout.de>` or in the 'Issues' section on LUKS website. Please attach the output of the failed command with the `--debug` option added.

## AUTHORS

The first implementation of `veritysetup` was written by Chrome OS authors.

This version is based on verification code written by Mikulas Patocka `<mpatocka@redhat.com>` and rewritten for `libcryptsetup` by Milan Broz `<gmazyland@gmail.com>`.

## COPYRIGHT

Copyright © 2012-2019 Red Hat, Inc.

Copyright © 2012-2019 Milan Broz

This is free software; see the source for copying conditions. There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

## SEE ALSO

The project website at <https://gitlab.com/cryptsetup/cryptsetup>

The verity on-disk format specification available at <https://gitlab.com/cryptsetup/cryptsetup/wikis/DMVerity>