dm-crypt

Device-Mapper's "crypt" target provides transparent encryption of block devices using the kernel crypto API.

For a more detailed description of supported parameters see: https://gitlab.com/cryptsetup/cryptsetup/wikis/DMCrypt

Parameters:

```
<cipher> <key> <iv_offset> <device path> \
<offset> [<#opt_params> <opt_params>]
```

<cipher>

Encryption cipher, encryption mode and Initial Vector (IV) generator.

The cipher specifications format is:

```
cipher[:keycount]-chainmode-ivmode[:ivopts]
```

Examples:

```
aes-cbc-essiv:sha256
aes-xts-plain64
serpent-xts-plain64
```

Cipher format also supports direct specification with kernel crypt API format (selected by capi: prefix). The IV specification is the same as for the first format type. This format is mainly used for specification of authenticated modes.

The crypto API cipher specifications format is:

```
capi:cipher_api_spec-ivmode[:ivopts]
```

Examples:

```
capi:cbc(aes)-essiv:sha256
capi:xts(aes)-plain64
```

Examples of authenticated modes:

```
capi:gcm(aes)-random
capi:authenc(hmac(sha256),xts(aes))-random
capi:rfc7539(chacha20,poly1305)-random
```

The /proc/crypto contains a list of currently loaded crypto modes.

<key>

Key used for encryption. It is encoded either as a hexadecimal number or it can be passed as <key_string> prefixed with single colon character (':') for keys residing in kernel keyring service. You can only use key sizes that are valid for the selected cipher in combination with the selected iv mode. Note that for some iv modes the key string can contain additional keys (for example IV seed) so the key contains more parts concatenated into a single string.

<key_string>

The kernel keyring key is identified by string in following format: <key_size>:<key_type>:<key_description>.

<key_size>

The encryption key size in bytes. The kernel key payload size must match the value passed in <key_size>.

<key_type>

Either 'logon', 'user', 'encrypted' or 'trusted' kernel key type.

<key_description>

The kernel keyring key description crypt target should look for when loading key of <key_type>.

<keycount>

Multi-key compatibility mode. You can define <keycount> keys and then sectors are encrypted according to their offsets (sector 0 uses key0; sector 1 uses key1 etc.). <keycount> must be a power of two.

<iv_offset>

The IV offset is a sector count that is added to the sector number before creating the IV.

<device path>

This is the device that is going to be used as backend and contains the encrypted data. You can specify it as a path like /dev/xxx or a device number <major>:<minor>.

<offset>

Starting sector within the device where the encrypted data begins.

<#opt_params>

Number of optional parameters. If there are no optional parameters, the optional parameters section can be skipped or #opt_params can be zero. Otherwise #opt_params is the number of following arguments.

Example of optional parameters section:

3 allow_discards same_cpu_crypt submit_from_crypt_cpus

allow_discards

Block discard requests (a.k.a. TRIM) are passed through the crypt device. The default is to ignore discard requests.

WARNING: Assess the specific security risks carefully before enabling this option. For example, allowing discards on encrypted devices may lead to the leak of information about the ciphertext device (filesystem type, used space etc.) if the discarded blocks can be located easily on the device later.

same_cpu_crypt

Perform encryption using the same cpu that IO was submitted on. The default is to use an unbound workqueue so that encryption work is automatically balanced between available CPUs.

submit_from_crypt_cpus

Disable offloading writes to a separate thread after encryption. There are some situations where offloading write bios from the encryption threads to a single thread degrades performance significantly. The default is to offload write bios to the same thread because it benefits CFQ to have writes submitted using the same context.

no_read_workqueue

Bypass dm-crypt internal workqueue and process read requests synchronously.

no_write_workqueue

Bypass dm-crypt internal workqueue and process write requests synchronously. This option is automatically enabled for host-managed zoned block devices (e.g. host-managed SMR hard-disks).

integrity:<bytes>:<type>

The device requires additional <bytes> metadata per-sector stored in per-bio integrity structure. This metadata must by provided by underlying dm-integrity target.

The <type> can be "none" if metadata is used only for persistent IV.

For Authenticated Encryption with Additional Data (AEAD) the <type> is "aead". An AEAD mode additionally calculates and verifies integrity for the encrypted device. The additional space is then used for storing authentication tag (and persistent IV if needed).

sector_size:<bytes>

Use <bytes> as the encryption unit instead of 512 bytes sectors. This option can be in range 512 - 4096 bytes and must be power of two. Virtual device will announce this size as a minimal IO and logical sector.

iv_large_sectors

IV generators will use sector number counted in <sector_size> units instead of default 512 bytes sectors.

For example, if <sector_size> is 4096 bytes, plain64 IV for the second sector will be 8 (without flag) and 1 if iv_large_sectors is present. The <iv_offset> must be multiple of <sector_size> (in 512 bytes units) if this flag is specified.

Example scripts

LUKS (Linux Unified Key Setup) is now the preferred way to set up disk encryption with dm-crypt using the 'cryptsetup' utility, see https://gitlab.com/cryptsetup/cryptsetup

```
#!/bin/sh
# Create a crypt device using dmsetup
dmsetup create crypt1 --table "0 `blockdev --getsz $1` crypt aes-cbc-essiv:sha256 babebabebabebabebabebabebabebab
```

```
#!/bin/sh
# Create a crypt device using dmsetup when encryption key is stored in keyring service
dmsetup create crypt2 --table "0 `blockdev --getsize $1` crypt aes-cbc-essiv:sha256 :32:logon:my_prefix:my_key 0
```

```
#!/bin/sh
# Create a crypt device using cryptsetup and LUKS header with default cipher
cryptsetup luksFormat $1
cryptsetup luksOpen $1 crypt1
```