NAME

`geli` — control utility for cryptographic GEOM class

SYNOPSIS

To compile GEOM_ELI into your kernel, place the following lines in your kernel configuration file:

```
device crypto
options GEOM_ELI
```

Alternately, to load the GEOM_ELI module at boot time, place the following line in your `loader.conf`(5):

```
geom_eli_load="YES"
```

Usage of the `geli`(8) utility:

```
geli init [-bPv] [-a algo] [-i iterations] [-K newkeyfile] [-l keylen] [-s sectorsize]
      prov
geli label - an alias for init
geli attach [-dpv] [-k keyfile] prov
geli detach [-fl] prov ...
geli stop - an alias for detach
geli onetime [-d] [-a algo] [-l keylen] [-s sectorsize] prov ...
geli setkey [-pPv] [-i iterations] [-k keyfile] [-K newkeyfile] [-n keyno] prov
geli delkey [-afv] [-n keyno] prov
geli kill [-av] [prov ...]
geli backup [-v] prov file
geli restore [-v] file prov
geli clear [-v] prov ...
geli dump [-v] prov ...
geli list
geli status
geli load
geli unload
```

DESCRIPTION

The `geli` utility is used to configure encryption on GEOM providers.

The following is a list of the most important features:

- Utilizes the `crypto`(9) framework, so when there is crypto hardware available, `geli` will make use of it automatically.
- Supports many cryptographic algorithms (currently `AES`, `Blowfish` and `3DES`).
- Can create a key from a couple of components (user entered passphrase, random bits from a file, etc.).
- Allows to encrypt the root partition - the user will be asked for the passphrase before the root file system is mounted.
- The passphrase of the user is strengthened with: B. Kaliski, *PKCS #5: Password-Based Cryptography Specification, Version 2.0.* , RFC, 2898.
- Allows to use two independent keys (e.g. "user key" and "company key").
- It is fast - `geli` performs simple sector-to-sector encryption.
- Allows to backup/restore Master Keys, so when a user has to quickly destroy his keys, it is possible to get the data back by restoring keys from the backup.

- Providers can be configured to automatically detach on last close (so users don't have to remember to detach providers after unmounting the file systems).
- Allows to attach a provider with a random, one-time key - useful for swap partitions and temporary file systems.

The first argument to `geli` indicates an action to be performed:

`init` Initialize provider which needs to be encrypted. Here you can set up the cryptographic algorithm to use, key length, etc. The last provider's sector is used to store metadata.

Additional options include:

`-a` *`algo`* Encryption algorithm to use. Currently supported algorithms are: `AES`, `Blowfish` and `3DES`. The default is `AES`.

`-b` Ask for the passphrase on boot, before the root partition is mounted. This makes it possible to use an encrypted root partition. One will still need bootable unencrypted storage with a `/boot/` directory, which can be a CD-ROM disc or USB pen-drive, that can be removed after boot.

`-i` *`iterations`*
Number of iterations to use with PKCS#5v2. If this option is not specified, `geli` will find the number of iterations which is equal to 2 seconds of crypto work. If 0 is given, PKCS#5v2 will not be used.

`-K` *`newkeyfile`*
Specifies a file which contains part of the key. If *`newkeyfile`* is given as -, standard input will be used. Here is how more than one file with a key component can be used:

```
# cat key1 key2 key3 | geli init -K - /dev/da0
```

`-l` *`keylen`*
Key length to use with the given cryptographic algorithm. If not given, the default key length for the given algorithm is used, which is: 128 for `AES`, 128 for `Blowfish` and 192 for `3DES`.

`-s` *`sectorsize`*
Change decrypted provider's sector size. Increasing sector size allows to increase performance, because we need to generate an IV and do encrypt/decrypt for every single sector - less number of sectors means less work to do.

`-P` Do not use passphrase as the key component.

`attach` Attach the given provider. The master key will be decrypted using the given passphrase/keyfile and a new GEOM provider will be created using the given provider's name with an `".eli"` suffix.

Additional options include:

`-d` If specified, a decrypted provider will be detached automatically on last close. This can help with short memory - user doesn't have to remember to detach the provider after unmounting the file system. It only works when the provider was opened for writing, so it will not work if the file system on the provider is mounted read-only. Probably a better choice is the `-l` option for the `detach` subcommand.

`-k` *keyfile*
Specifies a file which contains part of the key. For more information see the description of the `-K` option for the `init` subcommand.

`-p` Do not use passphrase as the key component.

`detach` Detach the given providers, which means remove the devfs entry and clear the keys from memory.

Additional options include:

`-f` Force detach - detach even if the provider is open.

`-l` Mark provider to detach on last close. If this option is specified, the provider will not be detached until it is open, but when it will be closed last time, it will be automatically detached (even if it was only opened for reading).

`onetime` Attach the given providers with random, one-time keys. The command can be used to encrypt swap partitions or temporary file systems.

Additional options include:

`-a` *algo* Encryption algorithm to use. For more information, see the description of the `init` subcommand.

`-d` Detach on last close. Note, the option is not usable for temporary file systems as the provider will be detached after creating the file system on it. It still can (and should be) used for swap partitions. For more information, see the description of the `attach` subcommand.

`-l` *keylen*
Key length to use with the given cryptographic algorithm. For more information, see the description of the `init` subcommand.

`-s` *sectorsize*
Change decrypted provider's sector size. For more information, see the description of the `init` subcommand.

`setkey` Change or setup (if not yet initialized) selected key. There is one master key, which can be encrypted with two independent user keys. With the `init` subcommand, only key number 0 is initialized. The key can always be changed: for an attached provider, for a detached provider or on the backup file. When a provider is attached, the user does not have to provide an old passphrase/keyfile.

Additional options include:

`-i` *iterations*
Number of iterations to use with PKCS#5v2. If 0 is given, PKCS#5v2 will not be used. To be able to use this option with `setkey` subcommand, only one key have to be defined and this key has to be changed.

`-k` *keyfile*
Specifies a file which contains part of the old key.

`-K` *newkeyfile*
Specifies a file which contains part of the new key.

`-n` *keyno*
Specifies the number of the key to change (could be 0 or 1). If the provider is attached and no key number is given, the key used for attaching the provider

will be changed. If the provider is detached (or we are operating on a backup file) and no key number is given, the key decrypted with the passphrase/keyfile will be changed.

`-p` Do not use passphrase as the old key component.

`-P` Do not use passphrase as the new key component.

`delkey` Destroy (overwrite with random data) the selected key. If one is destroying keys for an attached provider, the provider will not be detached even if all keys will be destroyed. It can be even rescued with the `setkey` subcommand.

`-a` Destroy all keys (does not need `-f` option).

`-f` Force key destruction. This option is needed to destroy the last key.

`-n` *keyno*
Specifies the key number. If the provider is attached and no key number is given, the key used for attaching the provider will be destroyed. If provider is detached (or we are operating on a backup file) the key number has to be given.

`kill` This command should be used in emergency situations. It will destroy all keys on the given provider and will detach it forcibly (if it is attached). This is absolutely a one-way command - if you do not have a metadata backup, your data is gone for good.

`-a` If specified, all currently attached providers will be killed.

`backup` Backup metadata from the given provider to the given file.

`restore` Restore metadata from the given file to the given provider.

`clear` Clear metadata from the given providers.

`dump` Dump metadata stored on the given providers.

`list` See `geom`(8).

`status` See `geom`(8).

`load` See `geom`(8).

`unload` See `geom`(8).

Additional options include:

`-v` Be more verbose.

SYSCTL VARIABLES

The following `sysctl`(8) variables can be used to control the behavior of the `ELI` GEOM class. The default value is shown next to each variable.

kern.geom.eli.debug: 0
Debug level of the `ELI` GEOM class. This can be set to a number between 0 and 3 inclusive. If set to 0, minimal debug information is printed. If set to 3, the maximum amount of debug information is printed. This variable could be set in `/boot/loader.conf`.

kern.geom.eli.tries: 3
Number of times a user is asked for the passphrase. This is only used for providers which should be attached on boot (before the root file system is mounted). If set to 0, attaching providers on boot will be disabled. This variable should be set in `/boot/loader.conf`.

kern.geom.eli.overwrites: 5
Specifies how many times the Master-Key will be overwritten with random values when it is destroyed. After this operation it is filled with zeros.

kern.geom.eli.visible_passphrase: 0
If set to 1, the passphrase entered on boot (before the root file system is mounted) will be visible. This possibility should be used with caution as the entered passphrase can be logged and exposed via `dmesg`(8). This variable should be set in `/boot/loader.conf`.

kern.geom.eli.threads: 0
Specifies how many kernel threads should be used for doing software cryptography. Its purpose is to increase performance on SMP systems. If hardware acceleration is available, only one thread will be started. If set to 0, CPU-bound thread will be started for every active CPU. This variable could be set in `/boot/loader.conf`.

EXIT STATUS

Exit status is 0 on success, and 1 if the command fails.

EXAMPLES

Initialize a provider which is going to be encrypted with a passphrase and random data from a file on the user's pen drive. Use 4kB sector size. Attach the provider, create a file system and mount it. Do the work. Unmount the provider and detach it:

```
# dd if=/dev/random of=/mnt/pendrive/da2.key bs=64 count=1
# geli init -s 4096 -K /mnt/pendrive/da2.key /dev/da2
Enter new passphrase:
Reenter new passphrase:
# geli attach -k /mnt/pendrive/da2.key /dev/da2
Enter passphrase:
# dd if=/dev/random of=/dev/da2.eli bs=1m
# newfs /dev/da2.eli
# mount /dev/da2.eli /mnt/secret
...
# umount /mnt/secret
# geli detach da2.eli
```

Create an encrypted provider, but use two keys: one for your girlfriend and one for you (so there will be no tragedy if she forgets her passphrase):

```
# geli init /dev/da2
Enter new passphrase:     (enter your passphrase)
Reenter new passphrase:
# geli setkey -n 1 /dev/da2
Enter passphrase: (enter your passphrase)
Enter new passphrase:     (let your girlfriend enter her passphrase ...)
Reenter new passphrase:   (... twice)
```

You are the security-person in your company. Create an encrypted provider for use by the user, but remember that users forget their passphrases, so back Master Key up with your own random key:

```
# dd if=/dev/random of=/mnt/pendrive/keys/‘hostname‘ bs=64 count=1
# geli init -P -K /mnt/pendrive/keys/‘hostname‘ /dev/ad0s1e
# geli backup /dev/ad0s1e /mnt/pendrive/backups/‘hostname‘
(use key number 0, so the encrypted Master Key by you will be overwritten)
# geli setkey -n 0 -k /mnt/pendrive/keys/‘hostname‘ /dev/ad0s1e
```

```
(allow the user to enter his passphrase)
Enter new passphrase:
Reenter new passphrase:
```

Encrypted swap partition setup:

```
# dd if=/dev/random of=/dev/ad0s1b bs=1m
# geli onetime -d -a 3des ad0s1b
# swapon /dev/ad0s1b.eli
```

SEE ALSO

crypto(4), gbde(4), geom(4), gbde(8), geom(8), crypto(9)

HISTORY

The geli utility appeared in FreeBSD 5.5.

AUTHORS

Pawel Jakub Dawidek ⟨pjd@FreeBSD.org⟩