


## TrueCrypt, cross-platform OTFE

On the fly encryption, or  [OTFE](#), is a method to have your data safely tucked away in a filesystem hidden inside an encrypted container (either file, or disk partition, or another block device) and be able to mount this encrypted filesystem in such a way that access to the data is transparent - the computer user does not even have to know the data in the container is de/encrypted on the fly when it is accessed.

You will need a kernel device driver for this kind of transparent manipulation of filesystem data.

One implementation of OTFE for Linux is [TrueCrypt](#).

TrueCrypt originated as a Windows-only program which knew both supporters and adversaries. As development progressed, a Linux equivalent was written and with the 4.2 release of TrueCrypt, finally all Windows functionality (most notably creating new containers) is also available in the Linux version.

The Linux version of TrueCrypt uses the 2.6 kernel's device mapper infrastructure and needs the [device-mapper](#) userland tools to interface with the kernel. Kernels older than 2.6.5 are not supported by TrueCrypt, which means that the Slackware 2.6 */testing* kernels are suitable candidates. The Slackware 2.6 kernels contain the device mapper (dm) as a module, and the corresponding userland tools can be found in */testing* as well, under *lvm2*.

### Slackware packages

I have made Slackware packages (with SlackBuild and sources available) for both TrueCrypt and device-mapper (a slightly more recent version than the one that is available in Slackware "testing"):

- [TrueCrypt](#)
- [device-mapper](#)

Note that the TrueCrypt package contains a kernel module, and I'll try to make packages available for Slackware 10.2 and "current". If a version for your 2.6 kernel is not present, you can easily create your own package by downloading the entire *build* directory and running the build script:

```
cd /tmp
mkdir truecrypt && cd truecrypt
lftp -c "open http://www.slackware.com/~alien/slackbuilds/truecrypt/; mirror
build"
cd build
./truecrypt.SlackBuild
```

The resulting truecrypt package will be written to the */tmp* directory.



Note that the SlackBuild script for TrueCrypt, and the resulting package, installs the truecrypt program setuid root. This allows non-root users to mount their own encrypted containers and use these. If you're not happy with that, edit this part of the SlackBuild before you build a package:

```
# If you do *not* want non-admin users to be able to use
truecrypt, set
# BINPERM=0750
```

```
BINPERM=4755
```



or if you have already installed the package, change the program's attributes:

```
chmod 0750 /usr/bin/truecrypt
```

After installing device-mapper and truecrypt packages, you're ready to start experimenting.

## Using OTFE

This is a quick walk-through to create and mount your own truecrypt container. For now, I will limit myself to using a file-based container. You can read the manual if you want to find out how to encrypt entire disk partitions.

- Create a container file, make it 50MB in size - let us call the file *overpocket.tc* (some SciFi readers will recognize that name)

```
truecrypt -c overpocket.tc
```

The program will ask a couple of questions, my answers are listed after the ': '.

Volume type:

- 1) Normal
- 2) Hidden

Select [1]: 1

Filesystem:

- 1) FAT
- 2) None

Select [1]: 1

Enter volume size (bytes - size/sizeK/sizeM/sizeG): 50M

Hash algorithm:

- 1) RIPEMD-160
- 2) SHA-1
- 3) Whirlpool

Select [1]: 1

Encryption algorithm:

- 1) AES
- 2) Blowfish
- 3) CAST5
- 4) Serpent
- 5) Triple DES
- 6) Twofish
- 7) AES-Twofish
- 8) AES-Twofish-Serpent

```

9) Serpent-AES
10) Serpent-Twofish-AES
11) Twofish-Serpent
Select [1]: 10

Enter password for new volume 'overpocket.tc':
Re-enter password:
Passwords do not match.

Enter password for new volume 'overpocket.tc':

```

The last question is to enter a passphrase that will be used to seal off the container. Choose a long and secure one! Actually, this passphrase is used to encrypt a second passphrase (that will remain for ever unknown to you). The second “internal” passphrase is used to encrypt the data inside the container. The encrypted passphrase is stored in the first part of the container, the *header*. This dual-layered approach makes it possible to “swap” the header of a container after you created the container, and assign a new password to it, without having to re-encrypt the whole container with the new passphrase. Remember, a container may be a partition, and that can be gigabytes in size.

You do of course have to know the original “outer” password to be able to re-encrypt the “inner” passphrase with the new passphrase you've chosen.

I think the password question has a bug that it uses your previous Return as the first password entry... that's why you can leave it empty at first, and afterwards:

- Change the password on the container

```
truecrypt -C overpocket.tc
```

- Map the container file to a device file and mount the mapped device to a directory in our filesystem. Being an ordinary user, I will create a mount point in my own homedirectory.

```
mkdir -p ~/mnt/tc
truecrypt overpocket.tc ~/mnt/tc
```

- Show the mapped device and the mount information

```

truecrypt -l
/dev/mapper/truecrypt0 /home/alien/overpocket.tc

df -T /home/alien/mnt/tc

```

| Filesystem             | Type | 1K-blocks | Used | Available | Use% | Mounted on |
|------------------------|------|-----------|------|-----------|------|------------|
| /dev/mapper/truecrypt0 |      |           |      |           |      |            |
|                        | vfat | 50979     | 0    | 50979     | 0%   |            |
| /home/alien/mnt/tc     |      |           |      |           |      |            |

Now that the encrypted container's filesystem is mounted somewhere under the computer's root filesystem, we can start using it by copying files and directories to and from it. The truecrypt kernel module takes care of all the data encryption and decryption that occurs when you are writing data to the container or reading out of there. All this is transparent to you, the user, as long as the container

remains mounted. Once unmounted, the container's data is indistinguishable from random bytes. You can then safely take the file with you on the road for instance, stored on a USB stick.

- Unmount the container's filesystem:

```
truecrypt -d overpocket.tc
```

Note this command used to be “truecrypt -d ~/mnt/tc” but with truecrypt 4.2a that does not (no longer?) work. I have no system to check if I had just made a typo here so I will leave this comment for the interested.

That's all for now folks! If I find the time, I will write more on this topic, since the concept of OTFE will appeal to many.

Remains to be said that TrueCrypt containers are interchangeable between Linux and Windows hosts. Create once, use everywhere!

From:

<https://wiki.alienbase.nl/> - **Alien's Wiki**

Permanent link:

<https://wiki.alienbase.nl/doku.php?id=linux:truecrypt>

Last update: **2006/10/03 21:26**

