HOWTO: EncFS Encrypted Filesystem in Ubuntu and Fedora GNU/Linux

I mentioned recently that I planned to keep using TrueCrypt in GNU/Linux since I had used it profitably in Windows, and that I also intended to keep using the container approach where you create a single file of a certain size and then mount it to get your virtual file system.

I'm reevaluating my plan. I still like TrueCrypt and will likely keep using it, maybe by alternating DVD backups between it and my new intended: the EncFS Encrypted Filesystem. (And of course GPG is always good for many crypto jobs, and will also be part of my security framework.)

The drawback with my TrueCrypt method is having to create files of fixed sizes. You either have a lot of empty space tied up or you're bursting at the seams and can't expand a volume. And you're dealing with some large files; perhaps with additional risk of an entire volume being more easily corrupted? I had managed this ok in the past, but I always agonized over the size of volumes to create for backups, since it's hard to predict future requirements there. I know TrueCrypt has ways of encrypting entire devices, but I've been hesitant to go down that road.

After adding another 200GB drive to the slug the other day, I didn't want to make more decisions about container sizes. I started thinking about alternatives. Maybe it was time to figure out how to encrypt the entire drive with a non-container method, and I wanted to look away from TrueCrypt since I'm interested in crypto diversity.

Summary of my vague crypto knowledge

Let me be clear that I'm no expert at this stuff. I'm just a dabbler. It is challenging to write about it because I know I don't understand things well enough to explain as well as I'd like. At the same time, I think I have something to offer that may be helpful to others, so I'm pressing ahead anyway. That, and it helps my own understanding to try explaining things. Those, and maybe you'll help correct my misunderstandings.

I have some experience using public key encryption in my day job, and know a little bit about using file encryption whether on single files or with a single file virtual drive system as in TrueCrypt.

Here are a few of the options I'm aware of for file system encryption, along with my dim understanding of them. For those readers who know their stuff, this may help clarify how utterly ignorant I am, and for those further down the clue ladder like myself… I hope I'm not just spreading misinformation.

- **Individual File Encryption:** For example, using GPG to encrypt individual or tarred or zipped files. This has its uses, but would be cumbersome for managing sets of files you want to keep encrypted. It's a pain to encrypt and decrypt the files every time you want to work with them, and a very slow and crude method for files you routinely work on. Much better is to have some system that lets you enter your password and then work with the files normally.

- **Virtual File System:** One example would be using TrueCrypt with the single file method, where you can mount the file and have it behave as a filesystem. Another is how EncFS does it, encrypting files separately. These methods will cause a performance hit, with all the crypto operations being handled on the fly by the CPU, but it's barely noticeable in my usage.

- **Block Methods?:** Other encryption methods that I haven't gotten in to very far seem to involve encrypting whole partitions or blocks of disk. There's also talk of loopbacks and things that I'm currently resisting learning more about. Similar to the large container file method in TrueCrypt, I think you have to commit to a certain size volume. From my brief readings, I gather that these are usually lower level techniques, tightly integrated with certain kernel features. Probably still some overhead and performance considerations, but I think these would be faster than other methods.

There's this whole realm of security considerations out there. I know in addition to securing your files, you also want to encrypt your swap space, `/tmp` directory, and probably many other things.

For me, I just have some financial and personal files I want to protect in case a computer or backup disc is stolen, so my security goals are pretty simple. (Although I also think about if I want to encrypt family pictures and videos. The thought of someone stealing your data and having all that is kind of creepy.)

For now, I just want to get certain files and backup files encrypted. Later I'll worry about swap and so on.

What to do?

I’ve been seeing a lot of references to dm-crypt/cryptsetup/LUKS out there, but it’s more than I want to wade in to at the moment. I think the numerous ingredients need to cook for a while longer, and probably be made in to a bland microwaveable package that I can handle preparing.

Also, I read a rant against a lower-level method of encryption that made me consider how much I wanted to trust more sophisticated techniques. With kernel upgrades and the way things are still emerging that could cause compatibility problems and make it hard to retrieve my old encrypted backups, I wanted something that was fairly simple, even if not the most advanced or highest performing technique in the world. I like the idea of using something that is more insulated from the coal-fired engines below deck.

The EncFS Encrypted Filesystem

My search brought me to EncFS, which I think will work really well for me. It uses the regular file system and grows or shrinks as needed. From what I can tell, it is reasonably secure and robust. It also is licensed under the GPL, which is a big plus for me. Thanks to Valient Gough for creating, maintaining, and sharing this great work of **free** software.

Rather than recapping even more things from the hazy outland borders of my comprehension, I’ll point to a few pages that were helpful to me:

- EncFS Home Page
- EncFS Home Page Extended Introduction
- EncFS Wikipedia Article
- How to get started with EncFS in three easy steps
- And especially: Ubuntu Forums HOWTO: Encrypted directory with EncFS, which was one of the first things I found in relation to EncFS that showed me how easy it was to start using. I had seen references popping up here and there and finally glommed on to this howto to give it a chance. Then I started reading more and liked what I was seeing functionality-wise and “hearing” about it.

The **disadvantages** listed in those pages are not a problem for me. As long as the data is secure, I don’t care if meta data is available. For my current purposes, it doesn’t matter if someone can see how big the file is, permissions, or when it was last modified. I also doubt that performance will be an issue. (Again, not for me, anyway.)

On the other hand, the **advantages** are nice. There is the flexibility already mentioned in not having to specify the size of the encrypted space. It just dynamically grows and plays nicely with other non-encrypted files on the disk. You can backup the encrypted files using something like rsync, without having to decrypt them first. It works for a regular user and there is no need for SUID as in TrueCrypt, with resulting risks. The clear files are only accessible by your user account while mounted.

Lighting the FUSE

EncFS is enabled by FUSE: “Filesystem in Userspace.” It’s a kernel module that allows creation of virtual file systems without needing to write kernel code. We’ll see below that it’s readily available for Ubuntu and Fedora.

I’m interested in learning about a couple of other cool FUSE projects: SSHFS, which allows access to remote file systems through SSH, and GmailFS, which stores data in Gmail. You can use EncFS on top of GmailFS for secure storage. That means almost 3GB worth of free (as in free beer), secure, offsite backup storage, if I can convince myself it’s safe to use that way.

The HOWTO from Ubuntu Forums explains things very well, but I’ll recap/copy it below for my own records, consolidate some of the responses, and add some of my own commentary. (And also show what to do in Fedora.)

HOWTO: EncFS for Ubuntu 6.10 (Edgy Eft) / 7.04 (Feisty Fawn)

Credit where credit is due. Remmelt’s HOWTO: Encrypted directory with EncFS:

http://www.ubuntuforums.org/showthread.php?t=148600

1. Install the software

6.10

```
sudo apt-get install encfs fuse-utils
sudo modprobe fuse
```

Says to add "fuse" to `/etc/modules` so that you don't have to run modprobe every time you reboot, but when I looked it was already there. If not, use sudo or gksudo and your preferred editor to modify `modules`. "fuse" goes on a line of its own.

7.04

Comes with fuse-utils already installed, so just:

```
sudo apt-get install encfs
```

I'm not sure if `sudo modprobe fuse` is necessary. I don't think it is, although I ended up running it when I had problems described below. (It looks like modprobe creates the necessary entry in the device directory `/dev`. I didn't think to check for `/dev/fuse` before running modprobe fuse, but I suspect it was there.)

I'm guessing this stuff comes from the "universe," so you'll need to have that repository enabled. If that doesn't mean anything to you, it may or may not help to read about my own initial bafflement at the world of Ubuntu software repositories.

2. Add yourself to the fuse group.

The installer creates a fuse group and to use fusermount (for mounting/unmounting FUSE filesystems) you need to be in this group.

```
sudo adduser your_username fuse
```

In 7.04, I tried this from the Fedora instructions below:

```
sudo usermod -a -G fuse username
```

And that worked just as well.

You can also use the GUI to do this. One commenter pointed out that you need to reboot to be recognized as a member of the fuse group. I first tried just logging out and back in, but that didn't do the job. Rebooting took care of it.

Another commenter suggested: "After `adduser` you can say `newgrp - fuse` to join the newly-made group in your current session (saves having to log out)."

This works just fine. It also causes the fuse group to become the default group for that session, so that new directories and files will be created with fuse as the group instead of the default group for that user. I haven’t found a way to stay in the fuse group and get my default group back without rebooting.

If you run encfs without being in the fuse group, you’ll probably see something like this:

```
fuse: failed to exec fusermount: Permission denied
fuse failed.  Common problems:
 - fuse kernel module not installed (modprobe fuse)
 - invalid options -- see usage message
```

In 7.04, I got that error even though I was in the fuse group. I noticed that `/dev/fuse` had root for the owner and the group. I changed it to have fuse for the group and that took care of it. (`sudo chown root:fuse /dev/fuse`)

3. Create a directory where your encrypted files and directories will be stored

(Can be in the home dir as below, or anywhere.)

```
mkdir ~/.crypt-raw
```

4. Create a mount point

The directory where you will mount the encrypted dir and be able to access the decrypted files.

```
mkdir ~/crypt
```

5. Create the encrypted system and mount it

The first time you mount the directory, encfs will create the encrypted filesystem.

```
encfs folder_to_mount mount_point
```

e.g.

```
encfs ~/.crypt-raw ~/crypt
```

Encfs requires absolute paths, either starting from / or using the ~ to represent your home dir. I didn’t do that the first time and was rebuffed with a note on proper usage.

What does it mean to create the encrypted filesystem? Essentially it means the creation of a `.encfs5` control file in the encrypted folder. You interactively pick the options you want and they go in that file. The way I understand it is that a random key is picked to encrypt the files and that is also encrypted with your password in that file, so that if you later change the password (using `encfsctl`), nothing needs to change other than in .encfs5.

So it looks like the .encfs5 file is the mechanism that tells the program whether to mount an already created filesystem or to generate a new one. In my testing, it was easy to copy files around and mount on other machines and on remote drives.

From the EncFS home page: “The control file contains the filesystem parameters, in addition to encrypted key data which is different for every filesystem.. You need both the password and this control file in order to access the data. If you loose either one, there isn’t anything I can do to help.” So **be careful**! I plan on keeping my control files backed up to CDs/DVDs/USB drives. Maybe encrypt them with GPG and send them to my Gmail account.

EncFS Options

When choosing “expert configuration mode,” you’re presented with several options. A little bit of explanation is given with each by the program. The EncFS Wikipedia article provides additional information. I’ve copied from both sources and added some of my own comments below.

* * * It is also easy to pick the standard configuration on your first try or speed through the expert choices, if you want to **skip this box to reach the exciting conclusion** of trying out your encrypted filesystem…

- **Cipher algorithm:** Lists what is available. My system offered **AES** with 16 byte block cipher or **Blowfish** with 8 byte block cipher. Both support key lengths of 128 to 256 bits and block sizes of 64 to 4096 bytes. There was also blowfish-compat which is compatible with older versions of encfs. (Standard configuration used Blowfish.)

- **Key size in bits:** For AES, 128 to 256 in 64 bit increments, so three choices: 128, 192, 256. More is better, with the trade-off of slower performance for increased security. (Standard config with Blowfish used 160 bits.)

- **Block size in bytes:** 64 to 4096 in 16 byte increments. According to Wikipedia, each time a single byte is read, the entire block that contains it must be decrypted, and for a write operation the block must be decrypted, altered, and re-encrypted.

I'd like to know more about the trade-offs here and why you'd use one size over another. Wikipedia suggests the default block size of 512 bytes is sufficient for most purposes. Bigger block sizes are better for large files, but don't mean increased security?

- **Filename encoding algorithms: Block** hides file name sizes "somewhat." **Null** doesn't encrypt filenames at all. **Stream** keeps filenames as short as possible.

  Wikipedia says that keeping names shorter might save space on the source directory's filesystem depending on how that filesystem manages the directory tree. I don't imagine this being an issue. I wonder what the difference is between block and stream for obscuring the filename length and/or filename itself? I'm thinking I'll just go with block encoding.

- **Enable filename initialization vector chaining:** Makes filename encoding dependent on the complete path, rather than just encoding each path element individually. With this method, identically named files in different folders will have different encoded names.

  Default is **Yes**, and I have no reason to use otherwise. Wikipedia says that with IV chaining, if a directory is renamed, all files and directories within it must have their encrypted filenames re-encrypted, and that this method shouldn't be used if heavily populated directories will be renamed often. So maybe I should say "No" here, because I don't think it would matter to me if identical names are encoded differently.

- **Enable per-file initialization vectors:** At the cost of only 8 bytes per file, you can make the filesystem more secure. Wikipedia says with this option disabled, each file is encrypted with the same initialization vector, which can make the volume key easier to break. Default is **Yes**.

- **Enable filename to IV header chaining:** Makes filename data encoding dependent on the complete file path. If a file is renamed, it will not decode successfully unless it was renamed by encfs with the proper key. If enabled, hard links will not be supported. Default is **No**.

  Not sure what this is about. Is it saying if you rename or move an encrypted file and then try to decrypt it you won't be able to? I'd like to know why you'd pick one over the other.

- **Enable block authentication code headers:** Significantly affects performance but means (almost) any modifications or errors within a block will be caught and will cause an error. Default is **No**.

  Again, not sure what this is about. If disabled and an encrypted file is modified, wouldn’t that still cause some problem with decryption? Or might it be hard to detect in a text file (for example)? I think the default is fine for me.

- **Finally! The Password:** If you’re interested in encryption, you probably realize that long passwords are better. Password strength is not emphasized or nagged about here, but I recommend using a long phrase of at least 20 characters, with 32 or more being ideal. Use upper and lowercase letters and numbers and punctuation.

  I don’t fault Mr. Gough for the approach taken here. It’s not the job of the program to lecture and teach everything. But I do like how TrueCrypt raises much more of a fuss about this. TrueCrypt and GPG also have you generating random data from the keyboard or mouse when creating encrypted volumes and keys, versus EncFS which does not, which makes me wonder if we’ve sacrificed some security here.

6. Use the encrypted file system!

Create some files and directories in your mount folder. In our example: `~/crypt`. You can look in `~/.crypt-raw` to see how they are created and encoded there.

For example:

Decrypted

```
scarpent@prometheus:~/crypt$ ls -l
total 56
-rw-r--r-- 1 scarpent scarpent 31971 2007-02-19 10:53 abc.txt
-rw-r--r-- 1 scarpent scarpent    12 2007-02-19 10:50 another_file.txt
-rw-r--r-- 1 scarpent scarpent     2 2007-02-19 10:51 b
-rw-r--r-- 1 scarpent scarpent  1725 2007-02-17 13:47 dir_list.txt
drwxr-xr-x 2 scarpent scarpent  4096 2007-02-17 16:26 test2
drwxr-xr-x 2 scarpent scarpent  4096 2007-02-17 13:48 test_dir
-rw-r--r-- 1 scarpent scarpent     5 2007-02-17 13:46 test.txt
```

Encrypted

```
scarpent@prometheus:~/.crypt-raw$ ls -l
total 56
drwxr-xr-x 2 scarpent scarpent  4096 2007-02-17 13:48 dxQXJsmGIOsMRGqrzrE5b-kN
```

```
-rw-r--r-- 1 scarpent scarpent    10 2007-02-19 10:51 lbpnkIn4nP9eKv0Eiyx46NtO
drwxr-xr-x 2 scarpent scarpent  4096 2007-02-17 16:26 Mrs-1Dukjix9fMRsEImrK33m
-rw-r--r-- 1 scarpent scarpent  1733 2007-02-17 13:47 O8qmtJ,dBqBlVeCfzem2k0zM
-rw-r--r-- 1 scarpent scarpent    20 2007-02-19 10:50 OmO5QPE-2-rDGKxi4Ife3mzM-Qds3wfCSRnuy5
-rw-r--r-- 1 scarpent scarpent    13 2007-02-17 13:46 SNhhXNn5F7-GMHS4YJOWCjfb
-rw-r--r-- 1 scarpent scarpent 31979 2007-02-19 10:53 WwounHd5LnhtSbSZR3qcXniT
```

7. Unmount the encrypted filesystem

```
fusermount -u ~/crypt
```

The next time you mount the filesystem with encfs, it will prompt you for your password and mount your directory.

That's All for Ubuntu

I'd recommend playing around with this for a while with test directories and files until you feel comfortable with how it works. Make sure that you only work on files in the mounted folder, ~/crypt. Verify that after you unmount, you don't see the plain files and directories. You also might want to see for yourself is that when you mount directories with this method, the decrypted files are only visible to your username.

Try changing your password using `encfsctl passwd crypt_dir`. With this command you don't have to give a full path, so that if you're in the home dir in our example, you could type `encfsctl passwd crypt`. It will change the password whether the encrypted filesystem is mounted or not.

There are things you can do with automount and fstab and etc., and I plan on learning about these real soon. For now I'm working on getting the basics down. I'm trying to confirm that the things I want to use will work, and then I'll figure out how to make them more convenient and automated with scripts and so on.

HOWTO: EncFS for Fedora Core 5 and 6

Now, would it be as easy to get working in Fedora?

Searching for [fedora encfs] with Google brings up a good howto as the first result, "HOWTO: encrypt a filesystem with Fuse-EncFS on Fedora Core 6":

http://www.dma.org/~tw/howtos/howto-fuse-encfs-the-web.html

Worked fine for me in Fedora Core 5 also. I think it may have been FC5 where "easy" support of this was added.

1. Log in as or become root

2. Install the software

```
yum install encfs
```

YUM is the Yellow dog Updater, Modified and apparently the counterpart of the Advanced Packaging Tool (apt-get) on Debian systems.

I've been very impressed with how both tools work. They've been like magic in my latest foray in to GNU/Linux. (I noticed that Fedora's yum got me a more recent version of EncFS: 1.3.1. In Ubuntu, I have 1.2.5.)

3. Add username to the fuse Unix group and preserve original group memberships

```
/usr/sbin/usermod -a -G fuse username
```

(Eric suggested the -a and better wording in the comments. It seemed to work ok with just -Gfuse, but looking at the `man` page it appears that -a should be used so as not to wipe out other groups. I haven't tried it this way, but trust that it will work.)

4. Start the service

Todd Warner, the author of these instructions, is unsure if this is necessary. I went ahead and ran it with the resulting message:

```
/sbin/service fuse start
```

```
Starting FUSE: done.
```

5. Log out of X and back in again

And now we can continue as in the Ubuntu HowTo.

That's All for Fedora

Todd has some more detailed tests to try out, and also has some good suggestions for scripting and for using symbolic links in conjunction with Firefox and Thunderbird to protect your .mozilla and .thunderbird folders. (Thanks, Todd, for taking the time to put stuff like this out there.)

That’s All for Now

Please let me know if there are gross (or even just yucky) errors in my write-up or if you found this post helpful. It takes a lot of time to put these things together and your satisfaction/feedback are my only rewards. **:-)**

Related: TrueCrypt in Ubuntu and Fedora GNU/Linux

by **Scott Carpenter**
21 February 2007 at 4:30 am
category: tech | tags: crypto, encfs, fuse
3,740 words

43 Comments

1. Hi,
   you can use libpam-encfs to automagically mount the encfs volume when you log in.

   Have fun, Simon

   Simon
   27 February 2007 at 12:11 am

2. Thanks for the tip, Simon.

   Scott Carpenter
   27 February 2007 at 5:04 am

3. Great howto, nice and simple enough to get it to work.

   etTea
   6 March 2007 at 10:31 pm

4. Thanks, etTea — I was pleasantly surprised at how easy it is to use EncFS.

   Scott Carpenter
   7 March 2007 at 5:34 am

5. Dear Scott,

   Thanks for your HOWTO.

I'm happy to report that the Ubuntu 6.10 story works for 7.04 (Feisty Fawn beta) too.

Onno Zweers
4 April 2007 at 10:09 am

6. Thanks, Onno. That's good to know.

Scott Carpenter
4 April 2007 at 3:22 pm

7. Hello Scott,

Thanks for a great howto and sharing you know's (and don't knows! very recognizable ;-) with us.
A small error, you write (in the Fedora section):

> 3. Create a protected mount point for the user
> /usr/sbin/usermod -Gfuse username

I'd re-phrase (and correct) this as:

3. Add (-a) 'username' to the fuse Unix group, while keeping his or her original group memberships intact (-G) :

```
# /usr/sbin/usermod -a -G fuse username
#_
```

Regards, Eric

Eric Maryniak
1 May 2007 at 8:59 am

8. Thanks, Eric! I've updated that step.

Scott Carpenter
1 May 2007 at 7:53 pm

9. Nice guide!

I'll have to check the wikipedia entry, as it sounds like a lot of detail has appeared since the last time I looked.

But to answer your questions about some of the options:
* block size: mostly a speed trade-off, which shouldn't have much (if any) impact on security. The trade-off is between sequential throughput and random io throughput (ie, how fast can you read/write huge files vs how many IO operations can be done per second). I just use the defaults

myself, but that can be tuned if desired. If anything, it should probably be expanded to allow much larger block sizes (like 256k).

* filename encoding: Null was by customer request – they only wanted data encrypted, not files. Stream existed first. Block is best, but if you have very long filenames (some people have 192+ char names), then stream may be useful to reduce the length a bit (on the order of half a dozen chars when using AES).

* filename -> IV header chaining and HMAC (block authentication) headers: some protection for some offline attacks. See encfs man page for a brief description. Lets say we worked together and our boss had an encrypted filesystem containing two directories, “scott” and “val”. And under each one of those, there was a file called “salary.txt”. Lets say that he just gave you a raise, so the timestamp has been updated on one of those files, giving me a way to guess which is which. Without knowing his password, I couldn’t change the files directly, but I could copy the *raw/encrypted* file from what I presume is your salary file to my raw file (assuming I got access to his raw/encrypted data). Then when he decrypts my salary file, he’ll see it has your salary data. Those sorts of attacks don’t compromise the data, but they do allow someone to modify it without detection. If IV header chaining was enabled, then the copied file would fail to decrypt (our boss would get an error message when trying to read the file).

Valient Gough
16 May 2007 at 2:07 am

10. Hi, Valient. Thank you for visiting and for your comment. And again thanks for EncFS! I’m still using and liking it very much.

Scott Carpenter
16 May 2007 at 5:37 am

11. To make your original group (e.g. mygroup) the default after using newgrp fuse, simply use newgrp mygroup. You will now be in both mygroup and fuse in the current session with mygroup as the default. It worked for me on Fedora 7.

Tim Watson
21 June 2007 at 3:27 am

12. Thanks, Tim — I’ll give that a try next time I set up a user with fuse.

Scott Carpenter
21 June 2007 at 4:17 am

13. Useful article. Would probably be useful to say whether the EncFS filing system is natively part of Fedora and Ubuntu.

David Legg
30 July 2007 at 9:20 am

14. Hi, David. I'm not sure what you mean by "natively part of Fedora or Ubuntu." It's not included out of the box with the distributions, but the package managers (apt-get in Ubuntu, yum in Fedora) will easily install it for you. EncFS uses the FUSE (Filesystem in Userspace) kernel module, which enables virtual file systems that don't require kernel code. In that sense, maybe it's not really "native," although it's transparent enough to me.

Scott Carpenter
30 July 2007 at 1:21 pm

15. dude… you rock

Anon
31 July 2007 at 2:06 pm

16. Our Linux group burlingtonlinux.org maintains a shared destop server that multiple users can connect to over the Internet using FreeNX with the NoMachine client. I hope to add further security by enlisting encfs-pam for user's individual home directories. I suppose /tmp and /var/tmp should be considered as well in a multi-user environment.

I can't wait to get this working!

-Joe Baker

Joe Baker
17 August 2007 at 8:53 am

17. Hi, Joe. Thanks for visiting and commenting.

Scott Carpenter
18 August 2007 at 4:09 pm

18. I have also been using Truecrypt on Windows and am moving to Linux. I've yet to find a way of mounting a Truecrypt volume from a normal user account on Linux, so I'm curious if you solved this problem. Or was it another reason you looked at encFS, as I am now doing?

Roger

Roger

3 December 2007 at 9:43 am

19. Hi, Roger. With the newer version explicitly preventing running as a normal user (as far as I can tell), I don’t think there is a way around it. (But I haven’t investigated at all.) I guess they just decided it was too much of a security risk.

    I still think Truecrypt is very nice even with that limitation, but EncFS does everything I need. It’s good to be able to run it as a normal user, and I like that it doesn’t require you to encrypt a whole drive or define a fixed-size volume/container. It also seems less risky to not have one large file that could potentially become corrupted.

    I use EncFS every day for many things and so far have had no problems with it. Examples: Firefox (including the cache), Thunderbird, etc, etc.

    Scott Carpenter
    3 December 2007 at 5:41 pm

20. Oh, and good luck on the move! It’s *so* worth the effort.

    Scott Carpenter
    3 December 2007 at 5:42 pm

21. Dag/rpmforge rpms have broken dependences, so you should make this way:

    ```
    yum install fuse-encfs dkms-fuse
    ```

    dkms-fuse install the fuse kernel module.

    Without the dkms-fuse package, you’ll see
    # /etc/init.d/fuse start
    Loading fuse module failed!

    synapse
    10 January 2008 at 10:22 am

22. Hi, synapse — thanks for visiting.

    Scott Carpenter
    10 January 2008 at 8:58 pm

23. Scott, thanks for the great How-to (works like a charm on Ubuntu 7.10) and great site. Am looking forward to reading other articles here.

    Now that’s probably a dumb question but … do you know why encfs ~/.crypt-raw ~/crypt and fusermount -u ~/crypt don’t work from a launcher, including in Run in Terminal mode?

Leon
14 April 2008 at 10:55 pm

24. Hi, Leon. You’re welcome, and thank you! Have you tried writing a script and launching that instead? I just tried creating a launcher for the script I use — it works fine with type “Application in Terminal.” (I’m also using 7.10. I didn’t see a “Run in Terminal” option.)

Scott Carpenter
15 April 2008 at 4:46 am

25. A script, eh? I thought the Launcher was supposed to take care of that … oh, well I have much to learn :)

By the way, if you just type the command in the Run Button applet – it works.

And yes – it’s Application in Terminal. Run in Terminal is from another opera :)

leaves to read about scripts

Leon
15 April 2008 at 8:01 am

26. Thanks a lot for this complete howto! Now I’ve set up all the things in order to encrypt some data on an external memory used on my notebook! I’ve understood this howto better than the “Italian counterpart” (and I’m Italian….) :D

Hope to see some howto about the script….

Andrea
21 April 2008 at 8:16 am

27. Hi, Andrea. I’m glad to hear this post was useful for you. I suppose I could write something about the script, although there’s not much to it.

You might want to take a look at Cryptkeeper, which is a Gnome applet for managing encfs encrypted folders. (I haven’t used it and can’t vouch for it, but it looks nice.)

Scott Carpenter
21 April 2008 at 8:56 pm

28. A note re the generation of entropy (random data) from the

keyboard/mouse:

When creating a public/private key pair for a public key encryption like GPG (also SSL and ssh), the program basically needs to create two very large prime numbers. This is done by trying big random numbers and testing if they are prime, and repeating until 2 are found.

encfs uses symmetric key encryption, which means it generates the key from the password alone. No random numbers needed.

http://en.wikipedia.org/wiki/Public-key_cryptography may help.

Mick.

Michael
25 February 2009 at 6:22 am

29. Hi, Mick. That's a good point. I wonder why TrueCrypt does the mouse thing? Is it possible there is a still larger key being generated and the password just unlocks that key?

Scott Carpenter
25 February 2009 at 4:10 pm

30. This worked without a hitch using dapper on a Western Digital My Book 1Tb external Firewire drive. Thanks!

David Gillies
15 May 2009 at 12:22 am

31. thanks for your great guide

We posted a guide in Spanish based on your Fedora's encfs section: http://sietecoyote.com/2009/06/11/como-crear-directorios-personales-cifrados-en-blag/

btw, the distro used for the guide is BLAG, based on Fedora 9

thanks for moving-to

:)

Siete Coyote
18 August 2009 at 2:55 am

32. Hola, Senor Coyote, y gracias por sus amables palabras. (I don't really know Spanish very well — the Google Translator had to help me come up with that!)

I’m glad you found this useful and also happy to see it appearing in other languages.

Scott Carpenter
18 August 2009 at 6:58 am

33. thanks for the how to
after all is done and I have an encrypted folder with my files

my question is:
how can I remove this data and not be encrypted anymore ?

limitations:
1-encrypted folder is 200GB
2-free space is 10GB

thanks

Veronica
18 August 2009 at 1:40 pm

34. I’m unclear on what you mean.

You have a 200 GB drive (file system) that has 190 GB worth of encrypted data? (And it’s encrypted with encfs.)

And now you want it to not be encrypted?

You’d have to move it all to an regular unencrypted folder, and since you only have 10 GB free, you’d have to move it 10 GB or less at a time.

Scott Carpenter
18 August 2009 at 2:29 pm

35. what I mean is
total HD is 500GB
my encrypted folder have data worth 200GB
free space on HD is 10GB

I decrypt my folder and now I have both folders
secretfolder and .secretfolder_encfs

I dont want it to be encrypted anymore and I was hoping for an easier way than move files of lesser size than the free space.

is this is the only way ?

thanks

Veronica
18 August 2009 at 2:40 pm

36. You are going to have to move the files out of the encfs folder if you want them to be decrypted.

Scott Carpenter
18 August 2009 at 2:58 pm

37. Here’s the easy way for Debian users:

```
#get out of X
CTRL-ALT-F1
login as yourself
SUDO init 3

#install encfs:
sudo apt-get install encfs

#set up directories...
USER=`whoami`
export USER
sudo mkdir /home/{.$USER,old-$USER}
sudo chown $USER:$USER /home/{.$USER,old-$USER}
sudo mount --bind /home/{$USER,old-$USER}

#set up encfs:
encfs /home/.$USER /home/$USER

#copy all your home files there:
mv /home/old-$USER/{*,.*/} /home/$USER

#sudo umount old-$USER
sudo rmdir old-$USER

sudo sh -c "echo test -f /home/sid/.crypt_mounted || su $USER -c 'encfs /home/
```

This will prompt you for your encfs password at login. If you don’t provide a correct password you will still load up as usual but your home dir will be empty (and unencrypted)

I share my netbook with friends, I just reboot and let them do whatever, then reboot back as myself. All my data is secure.

Caveat: the /tmp, /var/tmp, and /var/cache folders may all have temporary data from your apps. You should ensure that these are deleted when you shut down, or you may leave some cached web data, or other app data there.

Wil
12 October 2009 at 2:22 am

38. A more simple way to make your original group the default after using newgrp fuse is to simply use newgrp without any parameters. This will restore the default group from /etc/passwd.

However, each newgroups opens a new shell, so you'll possibly have to 'exit' more than once.

Works fine with Ubuntu 9.04.

Andreas Heidemann
28 October 2009 at 12:41 pm

39. **Great guide!**

I have one question. Didn't find answer yet.

What if I want to move encoded folder to another location, other user account? is it enough to do "mv source dest"?

Thanks,
Predrag

Predrag
7 February 2010 at 10:03 am

40. So, for example, you have:

```
/home/account1/crypt
```

That you've been using with account1, and now you'd like to move to:

```
/home/account2/crypt
```

And start using with account2?

For that scenario, yes, it's just a simple move of the encrypted dir (plain dir should be unmounted, of course).

You could try a little experiment to see that it works as expected.

Scott Carpenter
7 February 2010 at 10:16 am

41. Thank you for fast response.

Ok, i did that and it works.

Just one more question. If I want to backup encrypted folder to DVD, is it enough to burn, for example,
`/home/account2/crypt`
folder?

Thank you

Predrag
7 February 2010 at 11:13 am

42. Yes, it's as easy as that. That will include the `.encfs5` or `encfs6.xml` file which is absolutely critical for your encrypted volume.

Good luck with encfs. I'm still using it as a core part of my data security and it hasn't let me down yet.

Scott Carpenter
7 February 2010 at 11:25 am

43. (Should clarify: make sure you're backing up that `.encfs*` file also. It will be in the root of your encrypted dir, e.g. `/home/account2/crypt/.encfs6.xml`.)

Scott Carpenter
7 February 2010 at 11:26 am

One Trackback

1. By HOWTO: EncFS Encrypted Filesystem in Ubuntu and Fedora GNU/Linux via YUM « This too was Dugg by ... on 16 March 2007 at 5:06 pm

   […] read more | digg story […]

---

http://www.movingtofreedom.org/2007/02/21/howto-encfs-encrypted-file-system-in-ubuntu-and-fedora-gnu-linux/