NEXT SLIDE

Keep Your Secrets Secret in the Cloud with eCryptfs

Scott Randby

The University of Akron

srandby@gmail.com

http://srandby.org

Table of Contents

Why Keep Secrets Secret?

- National Security Agency
- Necessity
  - Confidential files
- Privacy
- Dropbox hacked

How eCryptfs Works

- eCryptfs is part of the Linux kernel.
- eCryptfs is a layered file system.
  - It mounts a top file system over a bottom file system.
- The bottom file system contains encrypted files.
  - File and directory names may also be encrypted.
- Data may read from and written into the top file system after it is mounted by eCryptfs.
- Encryption is transparent and occurs in the bottom file system when data in the top file system is altered.
- The directory structures of the two file systems are identical.

- The bottom file system may be 'safely' stored in the cloud.
- Each file is encrypted or decrypted using a randomly generated File Encryption Key (FEK)
  - We will use the Advanced Encryption Standard (AES) for encryption/decryption with a key length of 32 bytes (256 bites).
- Each FEK is encrypted with a File Encryption Key Encryption Key (FEKEK) resulting in the Encrypted File Encryption Key (EFEK)
  - The FEKEK is generated from the passphrase given by the user.
  - The FEKEK may also be generated by `ecryptfsd` to support public keys.
  - The EFEK is stored in the header of each encrypted file.
- See http://ecryptfs.org/ for more information about eCryptfs.

Install eCryptfs User Space Utilities

- Open a terminal.
- Install the eCryptfs user space utilities.
  - `sudo apt-get install ecryptfs-utils`
  - This will install `ecryptfs-utils` and `libcryptfs0`.

Create a Layered File System

- Open a terminal.
- Create two empty directories.
  - `mkdir ~/Encrypted ~/Decrypted`
  - Encrypted files will be stored in `~/Encrypted`.
  - Data will be read from and written to `~/Decrypted` by the user.
  - eCryptfs will mount `~/Decrypted` on top of `~/Encrypted`.
- You may name these directories whatever you wish, or you may choose to create one empty directory and mount it over itself.

- Mount ~/Decrypted over ~/Encrypted and encrypt ~/Encrypted.
  - `sudo mount -t ecryptfs ~/Encrypted ~/Decrypted`
- Enter the required information.
  - `[sudo] password for user:`
  - `Passphrase:`
    - For this example we will use `secret` as the passphrase.
  - `Select cipher:`

```
1) aes: blocksize = 16; min keysize = 16;
   max keysize = 32 (loaded)
2) des3_ede: blocksize = 8; min keysize = 24;
   max keysize = 24 (not loaded)
3) cast6: blocksize = 16; min keysize = 16;
   max keysize = 32 (not loaded)
4) cast5: blocksize = 8; min keysize = 5;
   max keysize = 16 (not loaded)
Selection [aes]:
```

- Enter the required information (cont.).
  - `Select key bytes:`
    `1) 16`
    `2) 32`
    `3) 24`
    `Selection [16]: 32`
  - `Enable plaintext passthrough (y/n) [n]:`
    - Plaintext passthrough permits the reading of any unecrypted files stored in `~/Encrypted`.
  - `Enable filename encryption (y/n) [n]: y`
  - `Filename Encryption Key (FNEK) Signature [7a1719eb53966dd1]:`
    - The digital signature of FNEK is used to verify its validity.
    - There is also a digital signature of FEKEK. In our case we FEKEK and FNEK will be identical so they have the same signature.

- The following will appear in the terminal.

```
Attempting to mount with the following options:
ecryptfs_unlink_sigs
ecryptfs_fnek_sig=7a1719eb53966dd1
ecryptfs_key_bytes=32
ecryptfs_cipher=aes
ecryptfs_sig=7a1719eb53966dd1
WARNING: Based on the contents of
[/root/.ecryptfs/sig-cache.txt],
it looks like you have never mounted with this key
before. This could mean that you have typed your
passphrase wrong.

Would you like to proceed with the mount (yes/no)? :
```

- Answer `yes`.

- The following will appear in the terminal.

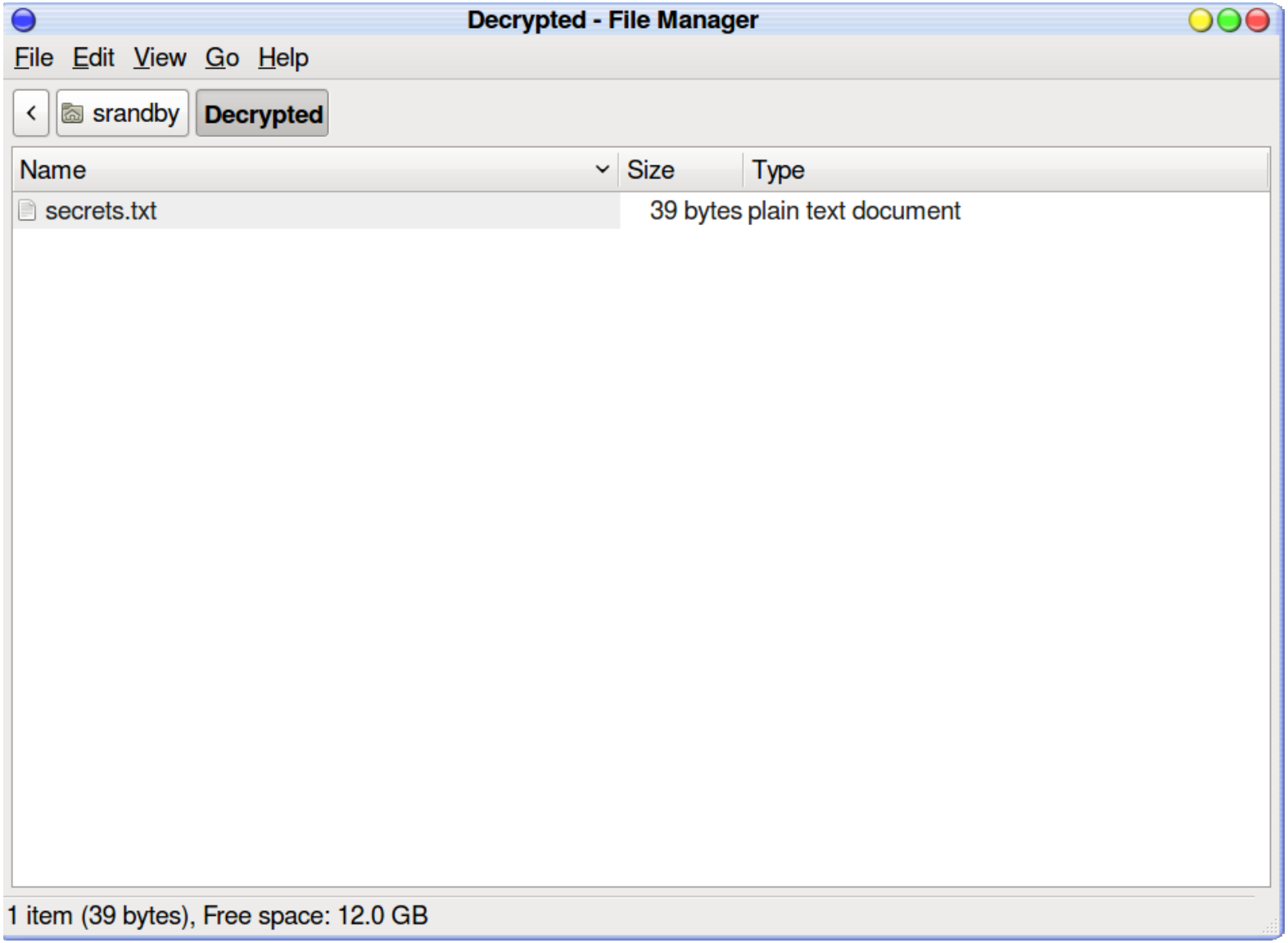
```
Would you like to append sig [7a1719eb53966dd1]
to [/root/.ecryptfs/sig-cache.txt]
in order to avoid this warning
in the future (yes/no)? :
```

- Answer `yes` if you plan to use this setup in the future. For this example, the answer will be `no`.
- The following will now appear.

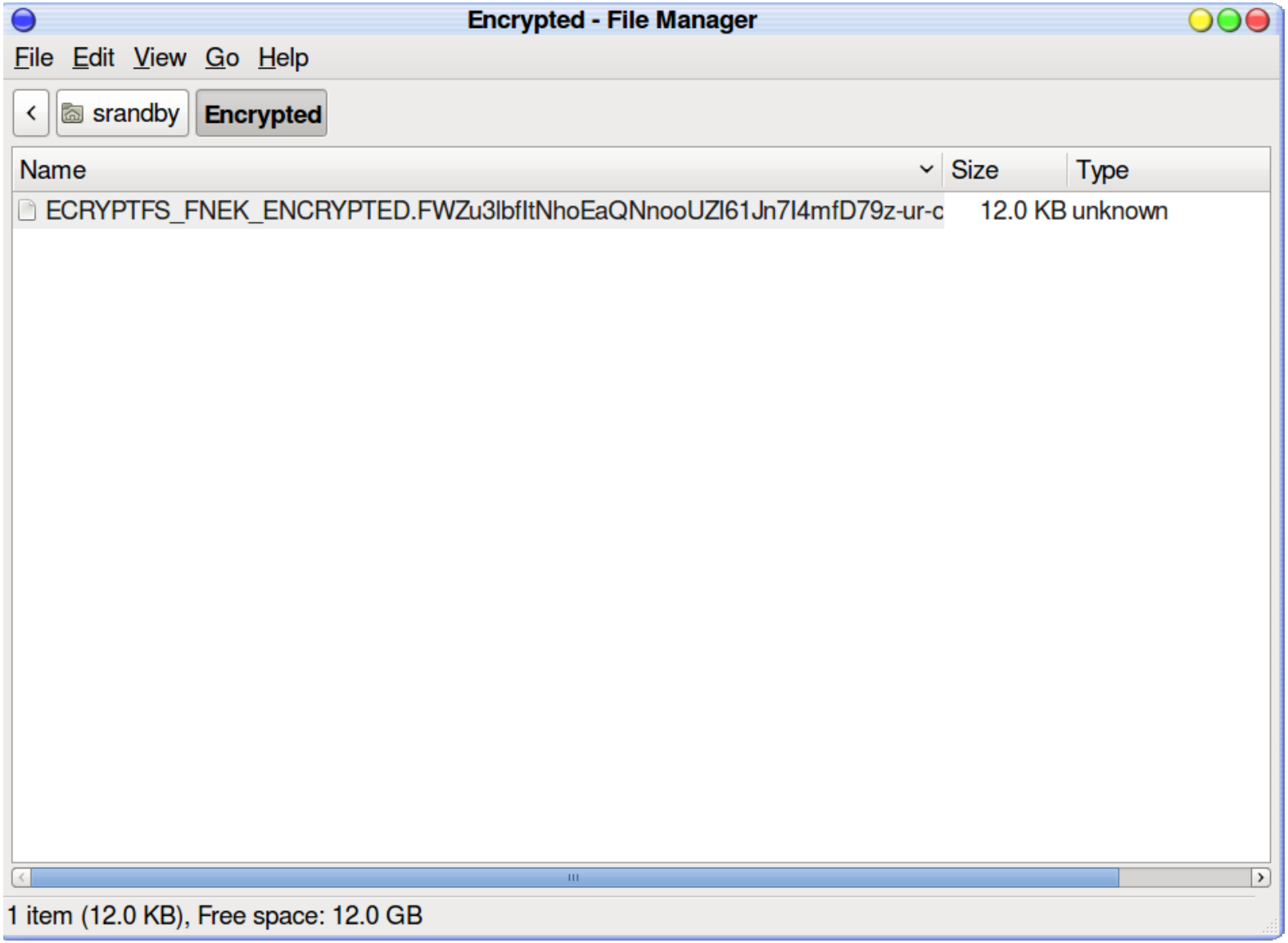
```
Not adding sig to user sig cache file;
continuing with mount.
Mounted eCryptfs
```

- Now add a file (or whatever) to `~/Decrypted`.
  - `emacs ~/Decrypted/secrets.txt`
  - Add content to the file and save it.
    - `This file contains my darkest secrets.`
    - `C-x C-s`

A view of ~/Decrypted.

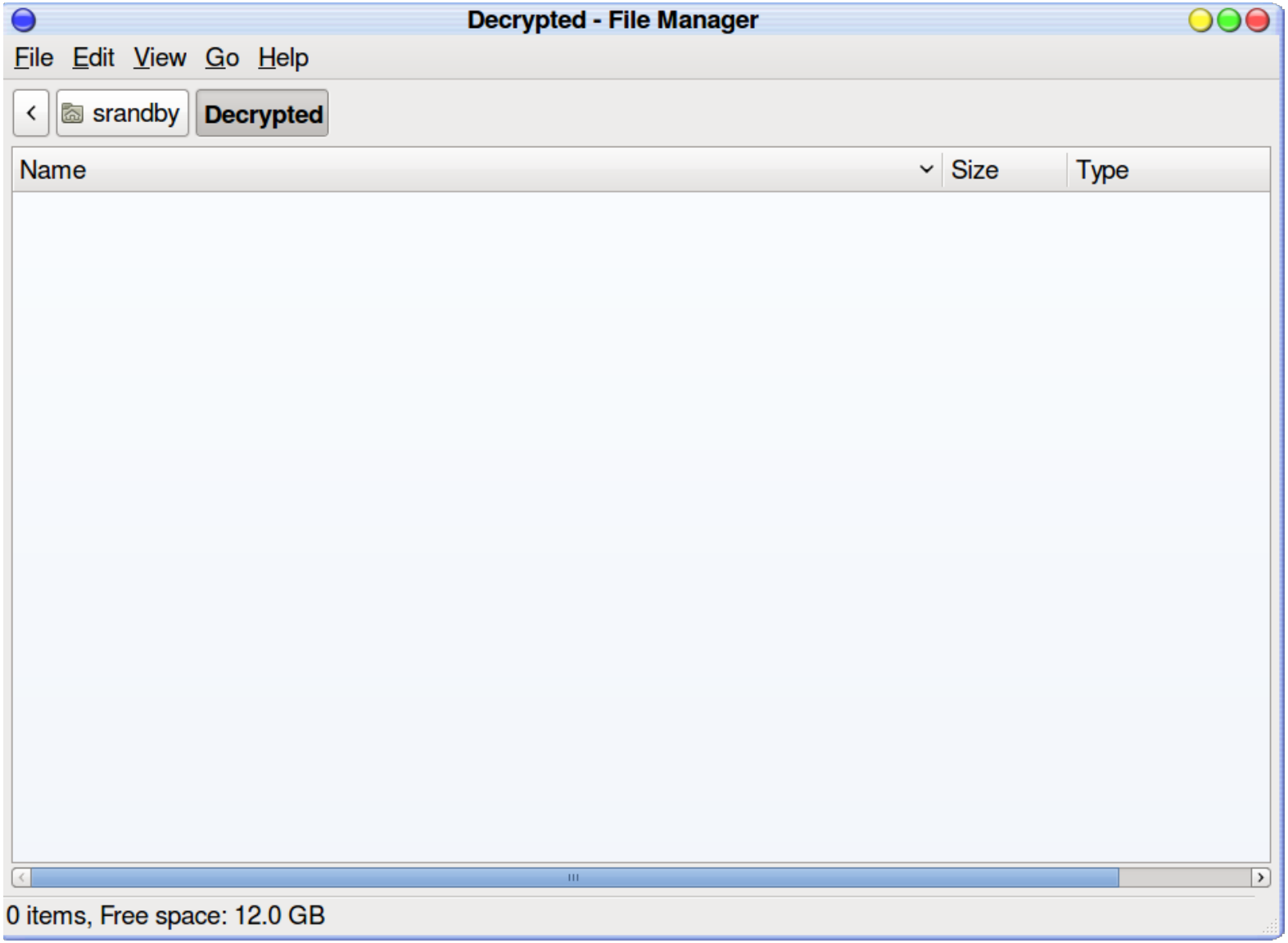


A view of ~/Encrypted.



- Unmount `~/Decrypted` once you are done adding content.
- `sudo umount ~/Decrypted`

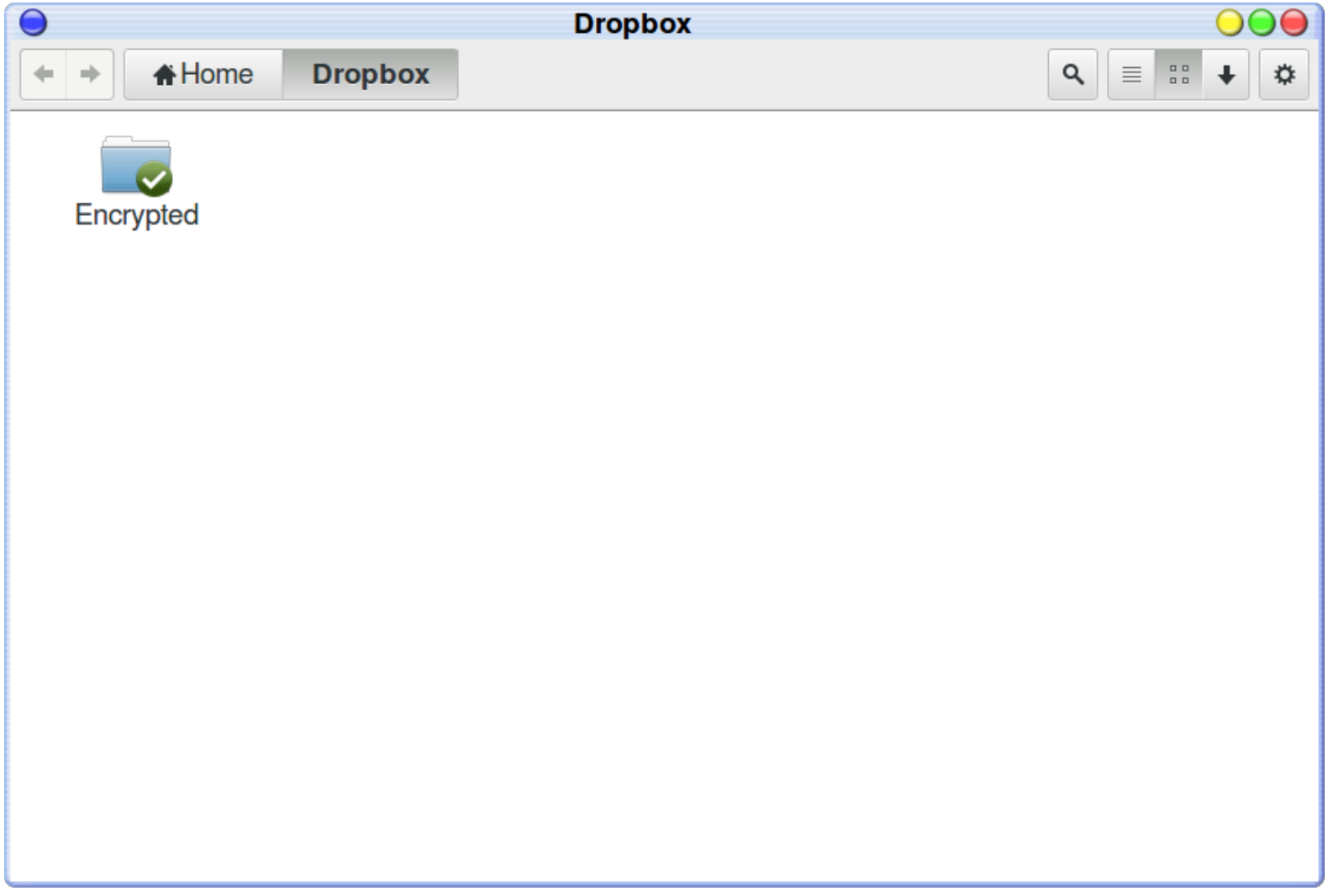
A view of ~/Decrypted after it is unmounted.



Upload Secrets to the Cloud

- Install the Dropbox client and set up an account if necessary.
  - https://www.dropbox.com/
  - Dropbox is not free software.
- Create a symbolic link from `~/Dropbox/Encrypted` to `~/Encrypted`.
  - `ln -s ~/Encrypted ~/Dropbox/Encrypted`
  - Any data put into `~/Decrypted` will be encrypted in `~/Encrypted`, put into `~/Dropbox/Encrypted`, and synched to Dropbox.

A view of Dropbox after ~/Encrypted has been added.



Synch with Another Computer

If you wish to use the encrypted file system that is stored in the cloud on another computer, you must set up a layered file system on that computer using eCryptfs.

- Install `ecryptfs-utils`.
  - `sudo apt-get install ecryptfs-utils`
- Install the Dropbox client, start the client, sign in, and synch.
- Stop Dropbox.
  - `dropbox stop`
- Move the encrypted files from Dropbox to their proper location.
  - `mv ~/Dropbox/Encrypted ~/`
  - The directory holding the encrypted files must have the same name and path as the directory on the first computer.

- Create a symbolic link from `~/Dropbox/Encrypted` to `~/Encrypted`.
  - `ln -s ~/Encrypted ~/Dropbox/Encrypted`
- Start Dropbox.
  - `dropbox start`

A directory must be mounted over `~/Encrypted` by eCryptfs before its data can be accessed and used.

- Create one empty directory.
  - `mkdir ~/Decrypted`

- Mount `~/Decrypted` over `~/Encrypted`.
  - `sudo mount -t ecryptfs ~/Encrypted ~/Decrypted`
  - Everything must be identical to the parameters used when the directory `~/Encrypted` was originally encrypted on the first computer.
    - Passphrase: `secret`
    - Cipher: `aes`
    - Select key bytes: 32
    - Enable plaintext passthrough: `n`
    - Enable filename encryption: `y`
    - Filename Encryption Key (FNEK) Signature: `7a1719eb53966dd1`
- Files in `~/Decrypted` may be edited, files may be added to and deleted from `~/Decrypted`, and everything will be synched to Dropbox.

Use a Script to Mount ~/Decrypted

The process of mounting ~/Decrypted may be automated by using the following shell script.

```
#!/bin/bash

sudo mount -t ecryptfs ~/Encrypted ~/Decrypted \
-o key=passphrase:passphrase_passwd=secret \
-o ecryptfs_cipher=aes \
-o ecryptfs_key_bytes=32 \
-o ecryptfs_passthrough=n \
-o ecryptfs_enable_filename_crypto=y \
-o ecryptfs_sig=7a1719eb53966dd1 \
-o ecryptfs_fnek_sig=7a1719eb53966dd1
```

The password may be stored in a file (on a USB stick for example) instead of being included in the script. In that case, the first option in the previous script should be replaced with something like:

```
key=passphrase:passphrase_passwd_file=/mnt/usb/passwd.txt
```

The process of unmounting ~/Decrypted may be automated by using the following shell script.

```
#!/bin/bash

sudo umount ~/Decrypted
```

Security Issues

- AES (Advanced Encryption Standard) flaws
- Password compromised
  - Brute force attack, etc.
- File permissions, file ownership, and file timestamps not encrypted
- No protection from `root` when file system is mounted
- eCryptfs bugs