Disk Encryption HOWTO

David Braun <cruhmoz602 [at] sneakemale [period] com (rewrite 'male' as 'mail')>

Copyright © 2004 David Braun

2004-11-17

Revision History

Revision 1.5	2004-11-17	DB
	added warning about dm-crypt	
Revision 1.4	2004-08-17	DB
	bug fixes, hints toward plausible deniability and dm-crypt	
Revision 1.3	2003-12-18	DB
	added link to Diceware	
Revision 1.2	2003-10-09	DB
	added idle logout and Gentoo instructions	
Revision 1.1	2003-09-13	DB
	added step to zero out keychain	
Revision 1.0	2003-08-28	tmm
	Initial release, reviewed by LDP	
Revision 0.92	2003-08-27	DB
	first submission to LDP	

Abstract

A method is described for encrypting a hard disk, either in whole or in part, with the encryption key stored on an external medium for increased security.

Table of Contents

Introduction

I've got a laptop computer running Linux and I don't want to worry about someone reading the personal information it contains, in case it gets lost or stolen. My log on password may stop someone from booting it, but it won't prevent an attacker from removing the hard disk and reading its data. I need stronger protection.

Fortunately, it's relatively easy to use encryption so the hard disk data would be unreadable if it were to fall into the wrong hands. Encryption's not only useful for portable computers like laptops—it can be used to protect any computer with personal information. I protect my computer's files with encryption for the same reason I lock my filing cabinet at home. For further motivation, you may be interested in reading Michael Crawford's Why You Should Use Encryption [http://www.goingware.com/encryption/].

I could encrypt only certain files, such as those in my home directory. This would protect the files but then I'd have to worry about information leaking out of them to other, unencrypted places on the disk. Instead I encrypt the whole disk so I don't have to manage this problem.

There are many encryption algorithms to choose from. I chose AES [http://csrc.nist.gov/CryptoToolkit/aes/] because it has been approved by the US government's National Institute of Standards and Technology [http://www.nist.gov/] and is well regarded by the cryptography community. I want my use of it to be resistant to dictionary attacks, so I use a long, randomly generated key. There's no way I'm going to memorize such a key so I keep it in a form I can carry with me easily: on a USB flash drive on my keychain. I encrypt the key with a passphrase so my data is protected in two ways: by a) what I have (the USB flash drive) and b) what I know (the passphrase). I can even give a friend access to my computer without giving away my passphrase—she uses her own USB flash drive and her own passphrase.

The operating system keeps the data encrypted on the disk at all times and decrypts it in RAM only as it's used. This way if the computer loses power suddenly the data will remain protected. The decryption key is loaded into RAM at boot time and kept there while the computer is on, so I don't need to keep the USB flash drive plugged in after starting the computer.

The procedure outlined in this HOWTO is written for version 2.4 of the Linux kernel. It will become less complicated with the release of Linux 2.6, which will have built-in support for encryption and do a better job of managing partitions within loopback devices.

This document assumes the reader has a moderate level of experience with Linux (you should be comfortable patching and compiling kernels [http://www.tldp.org/HOWTO/Kernel-HOWTO/index.html] as well as partitioning, mounting, and unmounting disks [http://www.tldp.org/HOWTO/Multi-Disk-HOWTO.html]).

Technical Summary

The encryption is implemented through a special kind of *loopback device*. A loopback device doesn't store any data itself; instead it takes all the data storage and retrieval requests it receives and passes them along to a real storage device, such as a disk or a file. As the data passes through, it can be filtered, and in our case the filter used is encryption.

When the system is deployed, a removable medium (USB flash drive) boots using GRUB, a kernel, and an initrd. Both the key and the kernel are selected from the GRUB menu, allowing a single removable medium to be used with multiple computers. The initrd contains just enough tools to ask for a passphrase, set up an encrypted loopback device, and mount it. After mounting, `pivot_root` is used to resume the boot process from the encrypted device. Loopback device offsets are used, instead of partitions, to access separate swap and root file system spaces within the encrypted loopback device because the 2.4 kernel doesn't provide access to partitions within loopback devices. The offset method does not generalize to multiple partitions (unfortunately) because the maximum offset understood by `losetup` is 2GB.

Copyright and License

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in Appendix A, *GNU Free Documentation License*.

Linux is a registered trademark of Linus Torvalds.®

Disclaimer

No liability for the contents of this document can be accepted. Use the concepts, examples and information at your own risk. There may be errors and inaccuracies that could be damaging to your system and you may lose important data. Proceed with caution, and although this is highly unlikely, the author does not take any responsibility.

All copyrights are held by their by their respective owners, unless specifically noted otherwise. Use of a term in this document should not be regarded as affecting the validity of any trademark or service mark. Naming of particular products or brands should not be seen as endorsements.

I know you hate reading directions and want to skip to the meaty bit right away, but I advise you to read the whole document first before touching anything. I know all the HOWTOs say that, but I really mean it for this one. It's worth it; trust me. You may also want to run through the procedure first on a test system before tackling a production system.

Acknowledgments

Thanks to Linus Torvalds, Jari Ruusu, and all the developers who contributed to their software, without which this HOWTO would have been impossible.

Thanks to the National Institute of Standards and Technology [http://www.nist.gov/] for carefully selecting a strong, open encryption algorithm.

Thanks to Mark Garboden and others on the linux-crypto [http://mail.nl.linux.org/linux-crypto/] mailing list and The Linux Documentation Project mailing lists [http://www.tldp.org/mailinfo.html#maillists] who took the time to critique my writing and offer suggestions.

Thanks to alert readers Ladislao Bastetti and Norris Pouhovitch for struggling through unusual hardware configurations, finding mistakes in the HOWTO, and suggesting good ideas.

Feedback

Feedback is solicited for this document. Please send additions, comments, and criticisms to the author.

Approaches

There are three different approaches we can take to encrypt the disk: encrypt the whole thing, a single partition, or a single file. I strongly recommend the first approach for best security. The first two approaches assume you'll be booting from removable media, such as a USB flash drive or a business card size CD-ROM. If you don't want to do this, you may modify the method to boot from the disk instead by making a small, unencrypted boot partition. If you want to use a USB flash drive to boot your computer, be sure your motherboard can do it first. At the time of this writing many cannot.

To avoid having to enumerate all three approaches everywhere I'm going to refer to what you're protecting as the *asset*. I will refer to the removable medium used to store the key as the *keychain*. I call it the keychain instead of the key because we can store lots of keys, each for different computers, on the same medium.

Whole Disk

A problem with keeping data secret with encryption is that the data likes to move around. Imagine the encryption is like a fence around your data. While the data's inside the fence, it's safe. To be most useful, however, data likes to be transmitted on networks, put on removable disks like CD-ROMs, and shared with friends. Any time your data leaves the fenced area it's unprotected. We can't put an encryption fence around all possible locations where our data might play but we do want to make the fence as large as practical. By putting the encryption fence around your whole hard disk, you won't have to worry about data becoming unprotected if it jumps to another part of the disk.

Warning

In this approach, we create one swap space and one root file system. Some people want more than a single encrypted partition for the root file system. Unfortunately, the method detailed here relies on the offset parameter of `losetup` to create "subpartitions" within the asset. The offset parameter is limited to a maximum value of 2GB, limiting the size of all but the last partition to 2GB. This works nicely for swap, which is already limited to 2GB on the i386 architecture, but I'm guessing it won't be practical for other uses. Using it to create multiple partitions smaller than 2GB is left as an exercise for the reader.

Another way to handle multiple partitions is to encrypt each partition separately (using the same key) to avoid the technical limitation above. This isn't secure as encrypting the whole disk because the partition table is exposed. When an attacker attempts to break encryption, the first thing he does is try to figure out what it's encrypting. A partition table listing Linux partitions is a big hint. For this reason I discourage encrypting multiple partitions separately, but arguably it's a good compromise for getting around the current `losetup` limitation. Another option is simply to wait for the release of Linux 2.6 because it is expected to make the offset parameter unnecessary.

Partition (for multiboot systems)

Encrypting the whole disk is fine if Linux is the only operating system on it, but this won't work for people who have set up their computer to boot multiple operating systems, e.g., Linux [http://www.kernel.org], NetBSD [http://www.netbsd.org/], and Darwin [http://developer.apple.com/darwin/]. In this case we can encrypt just the Linux partition and leave the others alone. Since we're booting from a removable medium, we won't even need to include the Linux partition in the multiboot menu with the others. To see why this isn't as secure as encrypting the whole disk, read Table 1, “Attack Tree”.

File (for home directories)

You may want to encrypt only a file on a file system. Once you've encrypted it you can put into it whatever you want, including other file systems. You might want to use this approach to encrypt only your home directory, for example. This is the least secure of the three approaches and not recommended. If you choose this approach you will notice instructions below to skip whole sections. This is because I'm assuming you've already booted an operating system and have your swap issues handled, so those sections don't apply to you. This HOWTO may be overkill for your needs and you can probably get away with just reading the fine `README [http://loop-aes.sourceforge.net/loop-AES.README]` that comes with loop-AES [http://sourceforge.net/projects/loop-aes/]. If you do, be sure to read the section called “Threat Model” before you finish here.

Threat Model

In order to protect our asset well, we must first understand what we're protecting it against. The general idea is that you've got a laptop which is vulnerable to being stolen or lost, and have a USB flash drive on your keychain that isn't, so this system is designed to handle the case that your laptop is stolen. I'm guessing your keychain won't be as easily stolen because it's in your pocket, and because an attacker won't know that it's important. If you pull your USB flash drive out of your pocket and someone non-technical exclaims, "What's that?", tell them it's a Pez dispenser [http://www.pez.com/].

Note

This system falls short when it comes to *plausible deniability*, which means there's no way to hide the fact that your personal data is encrypted. This is like locking your jewels in a safe and keeping the safe in plain sight in the middle of your living room. Only you can open the safe, but a man with a gun can tell you to open the safe for him. So if you're worried about your computer being subpoenaed [http://zdnet.com.com/2100-11-502433.html?legacy=zdnn] and being told to hand over your laptop, keychain, and passphrase, you'd better look at other solutions such as StegFS [http://stegfs.sourceforge.net/].

The following solution to the deniability problem has been suggested by Norris Pouhovitch. It should be possible to install a minimal Windows partition at the front of the disk and to encrypt the remainder. When the computer is turned on without the keychain, it boots Windows normally. When the keychain is booted, it skips the Windows partition, decrypts the remainder of the disk, and boots Linux.

The advantage of this scheme is that if the laptop is stolen and turned on, it will look like what a casual attacker is expecting to see (a Windows computer). On the other hand, a serious attacker could notice the unusually small partition and become suspicious. I will flesh out this idea further in a future version of the HOWTO.

Table 1. Attack Tree

Attack		Reaction	Notes
attacker steals laptop	while it is on	*SOL*	The asset is unprotected while the computer is running because the encryption key is in RAM. You can lower the risk by us-

Attack				Reaction	Notes
					ing an idle logout (the section called “Idle Logout”), but if you think your laptop is about to be stolen, turn off the power immediately and quickly read the Aikido HOWTO [http://www.aikido-faq.com/].
	while it is off	attacker doesn't steal keychain		*new key*	
		attacker steals keychain	attacker knows your passphrase	*SOL*	
			attacker doesn't know your passphrase	*new key*	
attacker steals keychain but doesn't have laptop	attacker knows passphrase			*new key*	Your asset is at risk because the attacker can decrypt it.
	attacker doesn't know passphrase	you're feeling lazy or you're convinced the keychain was lost, not stolen		*new passphrase*	You're probably OK without changing the asset key because the attacker can't decrypt the asset without the passphrase.
		you're feeling paranoid		*new key*	
attacker convinces you to send data over network				*SOL*	
attacker convinces you to copy data to removable medium				*SOL*	
you encrypt only a partition and a process writes data to a different partition				*SOL*	
you encrypt only a file and a process copies data from RAM to the unencrypted swap, or to a file in `/tmp`, or elsewhere on the unencrypted disk				*SOL*	
attacker demands you hand over laptop, keychain, and passphrase while waving a rubber hose menacingly				*SOL*	There is no plausible deniability built into the system.

new passphrase	Restore the keychain backup and choose a new passphrase.

new key	Generate a new random key to re-encrypt the asset, choose a new passphrase, and restore the asset backup.
SOL	Sorry Over your Loss

Caveats

- This method won't work (yet) with Software Suspend for Linux [http://swsusp.sourceforge.net/].
- Encrypting the disk will undoubtedly slow it down. I don't know by how much. If anyone has done some benchmarks, please send them to me.
- There is nothing in this method to support *plausible deniability* (see the section called “Threat Model”).
- It won't prevent information leaks via networks and removable disks.
- Encrypting backups is beyond the scope of this HOWTO.

Requirements

- a computer with an easily accessible removable medium reader (such as a USB port or a CD-ROM drive)
- a motherboard which supports booting from removable media (check carefully for USB, not all do)
- removable medium (such as a *USB flash drive*) to be used as the *keychain*
- Linux [http://www.kernel.org/] 2.4
- loop-AES [http://sourceforge.net/projects/loop-aes/]

A Digression about USB Flash Drives

There are many choices on the market. When I bought mine, I found one which fit the following requirements:

- physically small (I carry it on my physical keychain)
- supports USB 2.0 at full speed
- has a write-protect switch, so I don't clobber my encryption keys by accident

You might be tempted to get one with a fingerprint reader. I strongly encourage you not to. It might initially seem like a good idea, because by adding the biometric, your security protection expands to:

- something you have (the USB flash drive)
- something you know (the passphrase)
- something you are (your fingerprint, or whatever)

However, suppose something goes wrong. If you are now asking yourself, "What could go wrong?", then why are you reading this HOWTO? If something goes wrong, you make a change (see Corrective Reactions):

- Change what you have by using a different USB flash drive.
- Change what you know by learning a new passphrase.
- *You can't change what you are.*

Stop and ponder that last line for a while.

Looking to the Future

I wrote this document while using the 2.4 kernel. Linux 2.6 introduces the Device-mapper [http://sources.redhat.com/dm/] which we will be able to use to avoid playing games with losetup offsets. Linux 2.6 also introduces dm-crypt [http://www.saout.de/misc/dm-crypt/], an encryption layer for the Device-mapper which looks quite elegant. Unfortunately, it's not safe! [http://mareichelt.de/pub/texts.cryptoloop.php] Hopefully someday it will be fixed, but in the mean time the best course is to stick with loop-AES.

A future version of this HOWTO will explain how to use the Device-mapper with Linux 2.6.

Procedure

This method is designed to erase the contents of the asset before encrypting it. If you already have data on the disk you intend to encrypt, you should copy it somewhere else temporarily and then move it back once the encryption is set up. It is possible to encrypt data in place, but for now I consider such magic too advanced for this HOWTO. See loop-AES [http://sourceforge.net/projects/loop-aes/]'s README [http://loop-aes.sourceforge.net/loop-AES.README] for more details if you're interested in that method.

To do the following operations you will need to be running a system which has a loop-AES [http://sourceforge.net/projects/loop-aes/] capable kernel. If you don't have one already, I recommend using KNOPPIX [http://www.knoppix.com/]. It boots off a CD-ROM and doesn't need to be installed, so it's very little hassle.

For simplicity these instructions assume you'll be preparing the keychain and the asset on the same computer, but this needn't be the case. Adapt the instructions to whatever's convenient for you.

Prepare the Keychain

If you're taking the approach of encrypting only a file instead of a disk or a partition, you may skip this section and proceed directly to the section called “Prepare the Asset”.

In the ideal setup you will use a bootable keychain device, such as a *USB flash drive* or a business card size CD-ROM. This is because we want to expose as little of your disk as possible, but we're going to have to expose a minimal boot process or the computer will never start. Since the boot process will be necessarily unencrypted, it's better to have it away from your computer (on your keychain). If you can't or don't want to use a bootable keychain for some reason, then follow these instructions anyway but instead apply them to a small boot partition on your disk instead of the keychain.

In the following example the keychain shows up as the first SCSI drive `/dev/sda`. Replace `/dev/sda` with the device for your drive as appropriate.

The first step—zeroing out the keychain—is technically unnecessary, but it will make the keychain backup smaller if you back it up as an image as I suggest in the section called “Testing and Backup”.

```
bash# dd if=/dev/zero of=/dev/sda
```

Next, partition the keychain as you would any bootable disk. See the Linux Partition HOWTO [http://www.tldp.org/HOWTO/Partition/index.html] if you need help with partitioning.

```
bash# cfdisk /dev/sda
```

Put a file system on the first partition.

```
bash# mkfs /dev/sda1
```

Mount the keychain.

```
bash# mkdir /tmp/keychain
bash# mount /dev/sda1 /tmp/keychain
bash# cd /tmp/keychain
```

Build the Kernel

If you use the keychain with multiple computers you may want to build a different kernel for each one.

You probably need to build a custom kernel for your keychain so you can ensure two things:

- It has been patched correctly with loop-AES [http://sourceforge.net/projects/loop-aes/] and encryption support is turned on.
- All the device drivers necessary to boot your computer and make the asset accessible have been compiled in instead of loaded as modules.

You can load device drivers as modules, since we're using an initrd, but I chose to compile them into the kernel in order to keep the boot disk as simple as possible. Feel free to do differently.

For help building a custom kernel read The Linux Kernel HOWTO [http://www.tldp.org/HOWTO/Kernel-HOWTO/index.html]. Be sure to set `CONFIG_BLK_DEV_RAM` in the kernel configuration so it can boot using an initrd.

Follow the directions that come with loop-AES [http://sourceforge.net/projects/loop-aes/] to build the new loop driver. Also follow the directions to rebuild the util-linux [http://www.kernel.org/pub/linux/utils/util-linux/] tools, some of which we'll copy to the keychain later. Your distribution may have already built them for you (e.g., see the `loop-aes-utils` and `loop-aes-source` packages in Debian).

Once you've built the kernel, copy it to the keychain.

```
bash# mkdir boot
bash# cp arch/i386/boot/bzImage boot/vmlinuz-laptop
```

Install GRUB [http://www.gnu.org/software/grub/grub.html] or your favorite boot loader.

```
bash# grub-install --root-directory=. /dev/sda
```

Here is a sample `menu.lst` for GRUB. It has entries for two computers named `laptop` and `desktop`.

Important

It is required to pass the name of the key (I suggest you name it after the computer) as the first parameter to linuxrc.

Example 1. `/tmp/keychain/boot/grub/menu.lst`

```
title  laptop
root (hd0,0)
kernel /boot/vmlinuz-laptop root=/dev/ram0 init=/linuxrc laptop
initrd /boot/initrd

title  desktop
root (hd0,0)
```

```
kernel /boot/vmlinuz-desktop root=/dev/ram0 init=/linuxrc desktop
initrd /boot/initrd.old
```

Make the initrd

We boot the keychain using an initrd so we can remove it after the boot process starts (who wants a USB flash drive hanging out of their laptop while trying to look cool in a café?). To gain access to the asset we create a loopback device attached to the initrd's `/dev/loop0`. Putting the device file on the initrd means the initrd will have to stay mounted while the asset is mounted (not a big deal).

To learn all about making initial RAM disks you're welcome to read The Linux Bootdisk HOWTO [http://www.tldp.org/HOWTO/Bootdisk-HOWTO/index.html] and Linux's `Documentation/initrd.txt`, [http://linux.bkbits.net:8080/linux-2.4/anno/Documentation/initrd.txt@1.2?nav=index.html%7Csrc/%7Csrc/Documentation] or don't bother and just follow along.

We start by choosing 4MB for the size of the initial RAM disk, all of which we won't need, but it's the conventional maximum size (and it won't hurt) so that's one less decision to make.

```
bash# head -c 4m /dev/zero > boot/initrd
bash# mke2fs -F -m0 -b 1024 boot/initrd
```

Mount the initrd so we can work on it.

```
bash# mkdir /tmp/initrd
bash# mount -o loop=/dev/loop3 boot/initrd /tmp/initrd
bash# cd /tmp/initrd
```

Create the minimal directory structure we'll need.

```
bash# mkdir -p {bin,dev,lib,mnt/{keys,new-root},usr/sbin,sbin}
```

Create the minimal set of devices we'll need. Note that `tty` is necessary for the password prompt. This command assumes your asset is the drive `/dev/hda`. Change it as appropriate.

```
bash# cp -a /dev/{console,hda,loop0,loop1,tty} dev
```

We'll copy the six programs we'll need.

Tip

You can use `which` to find a program's full pathname, e.g.:

```
bash# which mount
/bin/mount
```

Copy the programs:

```
bash# cp /bin/{mount,sh,umount} bin
bash# cp /sbin/{losetup,pivot_root} sbin
bash# cp /usr/sbin/chroot usr/sbin
```

Use `ldd` to find out which shared libraries are used by each program:

```
bash# ldd /bin/{mount,sh,umount} /sbin/{losetup,pivot_root} /usr/sbin/chroot
/bin/mount:
        libc.so.6 => /lib/libc.so.6 (0x40023000)
        /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
```

```
/bin/sh:
        libncurses.so.5 => /lib/libncurses.so.5 (0x40020000)
        libdl.so.2 => /lib/libdl.so.2 (0x4005c000)
        libc.so.6 => /lib/libc.so.6 (0x4005f000)
        /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
/bin/umount:
        libc.so.6 => /lib/libc.so.6 (0x40023000)
        /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
/sbin/losetup:
        libc.so.6 => /lib/libc.so.6 (0x40023000)
        /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
/sbin/pivot_root:
        libc.so.6 => /lib/libc.so.6 (0x40023000)
        /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
/usr/sbin/chroot:
        libc.so.6 => /lib/libc.so.6 (0x40023000)
        /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
```

Copy the libraries. On my system I copied these libraries (yours may be different):

```
bash# cp /lib/{ld-linux.so.2,libc.so.6,libdl.so.2,libncurses.so.5} lib
```

Prepare the Asset

It's possible to repeat these steps as many times as you want to handle multiple computers using the same keychain. Each computer will have its own key and probably its own kernel. The instructions here assume the computer's name is `laptop`; substitute the name of the computer you're working with each time you repeat the steps.

First, back up your data. See the Linux Complete Backup and Recovery HOWTO [http://www.tldp.org/HOWTO/Linux-Complete-Backup-and-Recovery-HOWTO/index.html].

No, stop, listen to me. Back up your data. Really. It's no fun to have an encrypted hard disk if you can't decrypt it because of some mistake you made. These tools are powerful magic; if you blow it you can't just call up *Computer Gurus Are Us* and expect them to get your data back for you. That's the whole point of this exercise.

If you are encrypting your whole disk (recommended), replace `/dev/hda` with the device for your disk.

```
bash# ln -s /dev/hda /tmp/asset
```

If you are encrypting a partition (multiboot case), replace `/dev/hda3` with the device for your partition.

```
bash# ln -s /dev/hda3 /tmp/asset
```

If you are encrypting a file only, replace `~/encrypted` with the name of the file and create a link named `/tmp/keychain` that points to where you decide to store your key file (an already prepared removable medium, e.g., `/mnt/cf`).

```
bash# ln -s ~/encrypted /tmp/asset
bash# ln -s /mnt/cf /tmp/keychain
```

Initialize the asset with random data. This will make it less obvious to the attacker which parts are free space.

```
bash# shred -n 1 -v /tmp/asset
```

Here we create an encrypted file system to hold the keys. More encryption, you say? Yes, in case your keychain is stolen (see Table 1, “Attack Tree”), you don't want your keys to be exposed. I chose one megabyte as the size of the file system because it's a round number. There's no way we're going to need that much space for keys so feel free to chose a smaller size if you like (each key file will be 61 bytes long).

Again, initialize with random data.

```
bash# cd /tmp/initrd
bash# head -c 1m /dev/urandom > keys
```

To make the passphrase resistant to dictionary attacks we'll generate a seed. Whenever you see the symbol `<seed>` be sure to replace it with the one you generated. The following command will display a random seed on the screen.

```
bash# head -c 15 /dev/random | uuencode -m - | head -2 | tail -1
```

Set up the loopback device using the seed. This is where you choose your passphrase, which must be at least 20 characters in length. Choose one with care that you know you won't forget. You may want to use the Diceware method [http://world.std.com/~reinhold/diceware.html] for choosing a secure passphrase.

```
bash# losetup -e AES128 -C 100 -S <seed> -T /dev/loop1 keys
```

Format and mount the keys file system (the decrypt.sh script assumes you use the `ext2` file system here).

```
bash# mke2fs /dev/loop1
bash# mkdir /tmp/keys
bash# mount /dev/loop1 /tmp/keys
```

Now for the actual asset key, 45 bytes as random as your computer can make them. Try a dictionary attack against that, attacker! Ha! We name the key after the computer with which it will be used (`laptop`). Substitute the name of your computer instead.

```
bash# head -c 45 /dev/random | uuencode -m - | head -2 | tail -1 > /tmp/keys
```

Set up a loopback device with the key for encrypted access to the asset.

```
bash# losetup -e AES128 -p 0 /dev/loop0 /tmp/asset < /tmp/keys/laptop
```

Unmount the keys file system.

```
bash# umount /tmp/keys
bash# losetup -d /dev/loop1
```

Swap Partition

Skip this section if you're encrypting only a file.

It's critical to give `mkswap` a size parameter here because we're not handing it a dedicated partition. Choose whatever size you want; I chose 2GB.

```
bash# mkswap /dev/loop0 $((2*1024*1024))
mkswap: warning: truncating swap area to 2097144kB
Setting up swapspace version 1, size = 2147471360 bytes
```

Root File System

If you're encrypting only a file, format it with a file system like this and skip to the section called “Scripts”.

```
bash# mkfs /dev/loop0
```

We'll create the root "partition" after the swap space. I put the word 'partition' in quotes because it's not a real partition. We're faking it using the offset argument of `losetup`.

Notice how `mkswap` told us the actual size of the swapspace, which is not necessarily the size requested. Use the actual size (which was 2147471360 in the above example) when specifying the offset to begin the root file system.

```
bash# losetup -o <root offset> /dev/loop1 /dev/loop0
```

If the asset is the whole disk or the last partition on the disk, then we needn't worry about specifying a size for the file system. If this applies to you, do the following and skip to the section called “initrd Mount Point”.

```
bash# mkfs /dev/loop1
```

Since the asset isn't the last partition on the disk, we must give `mkfs` a size limitation or it will write all over whatever partitions are between this one and the end of the disk. I repeat, *if you don't give `mkfs` the correct size parameter here, you may lose data*. `mkfs` is actually just a front end, so to be as careful as possible we'll choose an actual file system maker, in this case `mke2fs`.

It's possible to limit the size of the file system by specifying its size in blocks, but `mke2fs` chooses the block size based on the size of the file system. A classic Catch-22! We can ask it to do a dry run on the rest of the disk (more than we want) to see what block size it would chose.

```
bash# mke2fs -n -j /dev/loop1
mke2fs 1.34-WIP (21-May-2003)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
9781248 inodes, 19544448 blocks
977222 blocks (5.00%) reserved for the super user
First data block=0
597 block groups
32768 blocks per group, 32768 fragments per group
16384 inodes per group
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 26542
        4096000, 7962624, 11239424
```

In this case it chose 4096. Whatever it chooses is probably close enough for our file system. Calculate the correct size in blocks.

file system size = (size of partition _ size of swap space) ÷ block size

Suppose the size of the partition is 10GB and the size of the swap is 2GB. The correct size for `mke2fs` is (10 _ 2) × 2^{30} ÷ 4096 = 2097152. Don't get this wrong! Make backups! Measure twice, cut once!

```
bash# mke2fs -j /dev/loop1 2097152
```

initrd Mount Point

Mount the new root file system and create the `initrd` mount point. This is necessary for the linuxrc script's call to `pivot_root`.

```
bash# mount /dev/loop1 mnt/new-root
bash# mkdir mnt/new-root/initrd
bash# umount mnt/new-root
```

Scripts

We have enough information to create the decryption script. Change the variables at the beginning to reflect your setup (including the seed you generated earlier).

If you're encrypting the whole disk or a partition, set `ROOT_OFFSET` to the size you got from `mkswap`. Put the script in `/tmp/initrd` and name it `decrypt.sh`.

If you're encrypting only a file then this script can live anywhere. In this case be sure to set `ROOT_OFFSET` to zero and set `MOUNT` to a convenient mount point (probably not `/mnt/new-root`).

Figure 1. `/tmp/initrd/decrypt.sh`

```
#!/bin/sh

SEED=<seed>
ASSET=/dev/hda
ROOT_OFFSET=<root offset>
ROOT_TYPE=ext3
MOUNT=/mnt/new-root
KEY="$1"

# Ask for a passphrase to open the keys (this prevents exposure of the keys in
# case the owner loses the keychain).  Give the user three tries to get the
# passphrase right.
for ((FAILED=1, TRY=1; ($FAILED != 0) && (TRY <= 3); TRY++))
do
        mount -n -t ext2 -o loop=/dev/loop1,encryption=AES128,itercountk=100,pseed
        FAILED=$?
done

if [ $FAILED -ne 0 ]; then
        echo "Sorry, you get only three attempts to guess the password."
        exit 1
fi

# Use the key to decrypt the asset.
losetup -e AES128 -p 0 /dev/loop0 $ASSET < "/mnt/keys/$KEY"

# Close the keys.
umount -n /mnt/keys
losetup -d /dev/loop1

# Set up the root "partition" device.
losetup -o $ROOT_OFFSET /dev/loop1 /dev/loop0

# Mount the root file system (read-only, so it can be checked with fsck).
mount -n -r -t $ROOT_TYPE /dev/loop1 $MOUNT
```

Make the script executable.

```
bash# chmod +x decrypt.sh
```

If you're encrypting only a file, skip to the section called “Testing and Backup”. Otherwise, save the following boot script as `linuxrc` and place it in `/tmp/initrd`.

Figure 2. `/tmp/initrd/linuxrc`

```
#!/bin/sh

# Decrypt the asset
source decrypt.sh "$1"

# Pivot to the asset's root file system.
cd $MOUNT
/sbin/pivot_root . initrd

# Pass control to init.
shift 1
exec chroot . /sbin/init $* <dev/console >dev/console 2>&1
```

Make the script executable.

```
bash# chmod +x linuxrc
```

Okay, the keychain and asset are now ready. Unmount everything.

```
bash# umount /tmp/{initrd,keychain}
```

You now have an empty, encrypted file system. Hurray!

Testing and Backup

Test your system by booting the keychain or executing the decrypt.sh script as appropriate (give it the name of the key you want to use as a parameter). After booting there may be a complaint about a nonexistent `/sbin/init` but that's okay for now.

Check to make sure your root file system mounted successfully. When you're confident everything is working, back up your keychain. In fact, make lots of backups. You might ask, "But isn't it insecure to have a copy of my keychain somewhere?" The answer is yes, it is, but not as insecure as losing your only keychain, if you define security as also meaning "securing access to my data".

Because my keychain is small I decided to back up the whole image so it's easy to restore:

```
bash# bzip2 -c /dev/sda > keychain.img.bz2
```

If you're encrypting only a file, you can pat yourself on your back at this point because you've finished.

Rescue Disk

Rescue disks are useful when a system isn't behaving properly and/or refuses to boot. Check to make sure your rescue disk has loop-AES [http://sourceforge.net/projects/loop-aes/] support in its kernel and has the correctly patched util-linux [http://www.kernel.org/pub/linux/utils/util-linux/] tools such as `losetup` and `mount`, otherwise it will be worthless with your newly encrypted asset. In the future, all rescue disks will include this support because it will come standard with the 2.6 kernel. In the meantime, KNOPPIX [http://www.knoppix.com/] (for example) already has all the necessary support and can be used as a rescue disk.

After booting an appropriate rescue disk, mount your keychain and execute the decrypt.sh script.

```
bash# mkdir /tmp/{keychain,initrd}
bash# mount /dev/sda1 /tmp/keychain
bash# mount -o loop=/dev/loop3 /tmp/keychain/boot/initrd /tmp/initrd
bash# pushd /tmp/initrd
bash# ./decrypt.sh laptop
bash# popd
bash# umount /tmp/{initrd,keychain}
```

You can now access your asset through the mount point you specified in decrypt.sh.

Installing Linux

Your final task is to install Linux to your new encrypted file system. As you do this make sure the entries in your `/etc/fstab` for the root and swap look like those below:

```
# /etc/fstab: static file system information.
#
# <file system> <mount point>   <type>  <options>               <dump>  <pass>
/dev/loop0      none            swap    sw                      0       0
/dev/loop1      /               ext3    errors=remount-ro       0       1
```

If you already have an installation elsewhere, read the Hard Disk Upgrade Mini How-To [http://www.tldp.org/HOWTO/Hard-Disk-Upgrade/] to learn how to copy it over.

The procedure for a fresh installation of Linux is different for each distribution. Please send me instructions for distributions not listed below and I will include them here.

Debian [http://www.debian.org]

1. Boot from a rescue disk by following the instructions in the section called “Rescue Disk”.
2. Install using the method 3.7 Installing Debian GNU/Linux from a Unix/Linux System [http://www.debian.org/releases/stable/i386/ch-preparing.en.html#s-linux-upgrade].

Gentoo [http://www.gentoo.org/]

1. Boot from a rescue disk (Gentoo's Live CD 1.4 won't work) by following the instructions in the section called “Rescue Disk”.
2. Activate the swap partition if you created one.

   ```
   bash# swapon /dev/loop0
   ```

3. Point `/mnt/gentoo` to the root file system.

   ```
   bash# ln -s new-root /mnt/gentoo
   ```

4. Skip to Chapter 8. Stage tarballs and chroot [http://www.gentoo.org/doc/en/gentoo-x86-install.xml#doc_chap8] in the Gentoo Linux 1.4 Installation Instructions [http://www.gentoo.org/doc/en/gentoo-x86-install.xml].

Idle Logout

Once your system is up and running, consider configuring it to log out automatically after a period of inactivity. This will lessen (but not eliminate) the risk of exposing your asset if the laptop is stolen while on (see Table 1, "Attack Tree").

More Information

- The `README [http://loop-aes.sourceforge.net/loop-AES.README]` that comes with loop-AES [http://sourceforge.net/projects/loop-aes/] explains how to use it in multiple scenarios.
- Encrypted Root Filesystem HOWTO [http://www.tldp.org/HOWTO/Encrypted-Root-Filesystem-HOWTO/]
- The Hardened Gentoo Project [http://www.gentoo.org/proj/en/hardened/]'s A Structured Approach to Hard Disk Encryption [http://www.sdc.org/~leila/usb-dongle/readme.html] is more comprehensive and is targeted to Gentoo [http://www.gentoo.org/] users.

Glossary

AES [http://csrc.nist.gov/CryptoToolkit/aes/]	Advanced Encryption Standard, a strong, well-regarded *encryption* algorithm chosen by the United States National Institute of Standards and Technology [http://www.nist.gov/]
asset	the data being protected by *encryption*—either a disk, partition, or a file
encryption	a mathematical means of scrambling data so that it's unintelligible unless decrypted using a specific *key*
key	the small piece of data necessary to make encrypted data intelligible
keychain	the physical medium (such as a *USB flash drive)* used to hold the encryption *key* (and possibly the beginning of the boot process)
loopback device	a Linux block device which appears to store data (by using another device)
loop-AES [http://sourceforge.net/projects/loop-aes/]	software written by Jari Ruusu that implements the *AES* algorithm using a *loopback device*
plausible deniability	a means to avoid being coerced into decrypting one's own data for an attacker
USB flash drive	a small electronic device containing a memory chip and a USB interface

A. GNU Free Documentation License

Copyright (C) 2000,2001,2002 Free Software Foundation, Inc. 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent

modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/ or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright (c) YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with...Texts." line with this:

> with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.