

FileVault 2 Decoded

Rich Trouton
Howard Hughes Medical Institute,
Janelia Farm Research Campus

HHMI
HOWARD HUGHES MEDICAL INSTITUTE

Before we get started, there's two things I'd like to mention. The first is that, all of the slides, speakers' notes and the demos are available for download and I'll be providing a link at the end of the talk. I tend to be one of those folks who can't keep up with the speaker and take notes at the same time, so for those folks in the same boat, no need to take notes. Everything I'm covering is going to be available for download.

The second is to please hold all questions until the end. If you've got questions, make a note of them and hit me at the end of the talk. With luck, I'll be able to answer most of your questions during the talk itself.

Similar Names, Different Beasts

- Apple has completely revamped FileVault in Lion
- Grown from an encryption solution that protected only home folders to one that can protect entire drives.
- For simplicity, the older FileVault encryption will be referred to as “FileVault 1” during this talk.

One of the changes that Apple introduced with Lion is that its FileVault encryption solution has been completely revamped, changing it from encryption that primarily protected your account's home folder to encryption that protects your whole boot volume. Despite the common name, the two solutions are very different beasts. If you've used the older FileVault, be aware that everything you know about it is changed in Lion. Since Apple's dubbed the new encryption as FileVault 2, I'm going to refer to the older FileVault as FileVault 1.

Where FileVault Has Been

- Uses encrypted disk images to protect your home folder.
- The contents of the FileVault 1-protected home folder are encrypted and decrypted on the fly.
- The disk image grows and shrinks as needed.

To understand where FileVault 2 is going, it's a good idea to look back at how FileVault 1 works. At its heart are encrypted disk images. Encrypted disk images are just like any other disk image you can create with Disk Utility, with the exception that they are password protected and encrypted with AES encryption. As long as you know the passphrase, you can use them as you would any other disk image file.

FileVault 1 took the same encrypted disk image technology and used it to protect one particular folder: your account's home folder. How FileVault 1 did this was by creating an encrypted disk image that's able to grow or shrink with the amount of data stored in your home folder, mounting that encrypted disk image when you log in and then un-mounting it when you log out. The contents of the home folder are automatically encrypted and decrypted on the fly.

FileVault 1 Upsides

- Strong encryption
- Came with the OS, no extra charge to use it.
- Designed to work like an decrypted account wherever possible.



FileVault 1 had some upsides and downsides. The biggest upsides were that it used strong encryption, was low cost, and it was designed for ease of use. It used AES encryption (started with AES 128 in 10.3 and 10.4 and later used AES 256 in 10.5 and 10.6.) It came with Mac OS X, so you're getting it for the same price that you paid for your operating system. Apple went to a considerable amount of trouble to make sure that you hardly noticed anything different about working in an account that's encrypted from one that wasn't encrypted.

FileVault 1 Downsides

- Backups
- Network accounts
- Whole disk encryption not available



The biggest downsides had to do with backups, using FileVault 1 with network accounts, and that FileVault 1 was not able to provide whole disk encryption.

Backing up FileVault 1

- Most backup software not able to handle backing up data reliably from a FileVault-protected home folder.
- User either needed to be logged in (data not encrypted in the backup) or logged out (not able to back up just the files that had changed in the home folder.)

With regards to backups, the problem was that FileVault 1, at its heart, uses a password-protected encrypted disk image. Most backup software was not able to handle backing up the data in the account when the machine was logged out, so the user needed to be logged in for the backups to run. Alternatively, if you were backing up the encrypted disk image file itself, the situation could be even more problematic because you would now be backing up the encrypted disk image and not the actual files stored inside the disk image. Also, if you change the encrypted disk image file while it's being backed up (for example, by logging in to the account) you can corrupt the backup, making it hard or impossible to restore your files. That's one of the reasons why Time Machine on 10.5 and 10.6 only backed up a FileVault 1-encrypted home when the user logged out.

FileVault 1 & Network Accounts

- FileVault disk image did not know about password changes made outside of the Mac.
- User could be locked out of their account by a routine password change by the help desk.
- Using the Master Password to help recover FileVault-encrypted network users usually required some command-line work.

The problem with network accounts combined with FileVault 1 is again that FileVault 1 was using a password-protected disk image. The disk image only knew the password that's able to unlock it and didn't check with any other sources, like the external directory service that actually manages your account's password. So the FileVault disk image didn't know when you forgot your password and had to call your help desk to get it reset. It also didn't pick up the new password when the help desk reset your account's password on their end. All it knew was that the password that you put in to the login screen didn't match the one that it needed to unlock the disk image. Before Apple put in a way to update your FileVault password as part of the 10.5 login process, password changes with FileVaulted network accounts either needed to be done from the encrypted machine, or you had to call IT for help every time there was a password change.

Apple did provide a way to reset the encrypted disk image's password via the Master Password, but this was a solution primarily built for dealing with OS X's own local accounts instead of network accounts. Leveraging the Master Password to help you recover network accounts that have FileVaulted local homes usually required some work on the command line.

No Whole Disk Encryption

- FileVault 1 was unable to encrypt the whole boot drive.
- For environments that required the use of whole disk encryption, FileVault 1 flunked.

On top of all of that, the biggest issue for FileVault 1 in a number of corporate and government environments was that those entities were requiring full-disk encryption, where everything on the machine (not just in the home folders) needed to be encrypted. FileVault 1's ability to only protect the home folders meant that it automatically failed that requirement, regardless of its virtues and blemishes.

Back to the drawing board

- Complete rebuild of FileVault for Lion
 - Uses new virtual volume storage (Core Storage), whose primary purpose is to provide encrypted volume storage.
- Core Storage encrypted volumes are built on a per-partition basis.
 - Allows both encrypted and decrypted partitions on the same physical hard drive.

Apple had known about its customers' problems with FileVault 1 as it was designed and went back to the drawing board to provide a full disk encryption solution. Ultimately, the effort would require the complete rebuild of FileVault from the ground up, basing the new FileVault 2 on an entirely new type of virtual volume storage, which Apple named Core Storage. While you can make unencrypted Core Storage volumes, its primary use in 10.7 presently is to provide FileVault's encrypted volume storage. Core Storage works on a per-partition basis, which means that you can have both encrypted and unencrypted partitions on the same physical hard drive.

How FileVault 2 works

- On startup, the Mac initially boots to a small decrypted partition that only provides access to the tools to unlock the larger encrypted storage.
- When the right authentication is provided, the encryption unlocks and the Mac boots from the Mac's regular OS.
- By unlocking the encryption before the OS boots, the issues with network accounts and backups are solved.

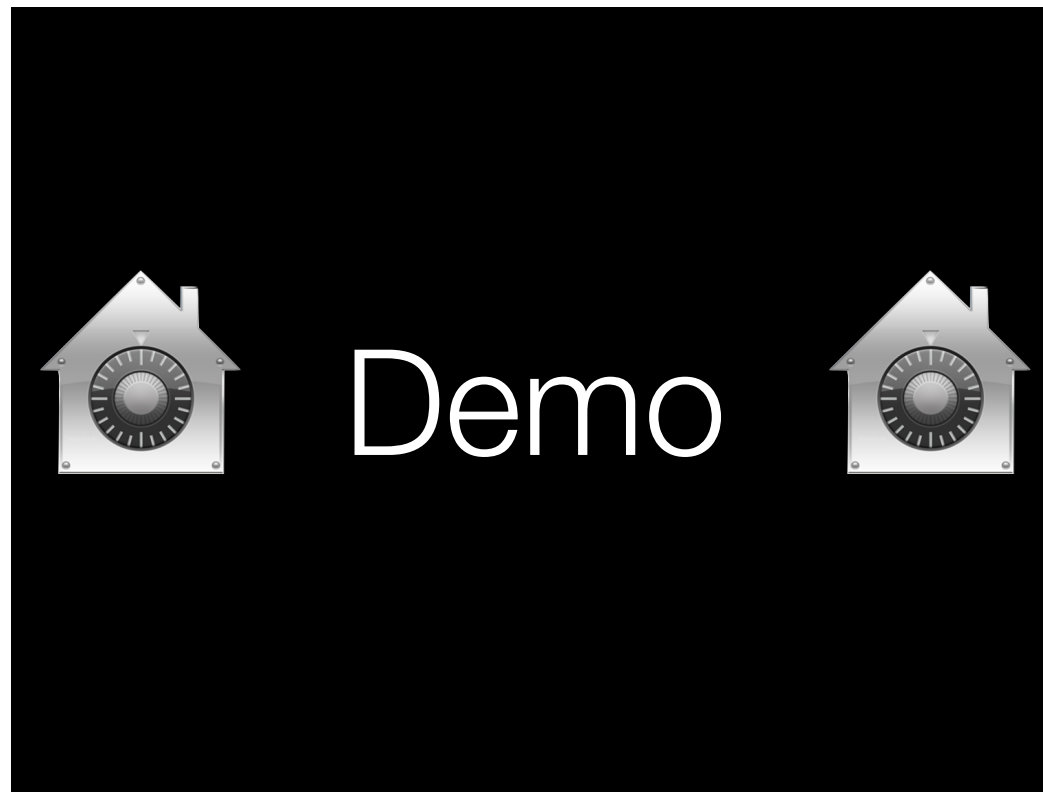
FileVault 2, in broad terms, works like PGP or other full disk encryption solutions available for the Mac. When you start up your system, a small unencrypted partition provides just enough access to provide you with the tools to unlock the much larger encrypted storage, usually through what's called a pre-boot login screen. If you provide the right authentication to the pre-boot login, the encrypted storage then unlocks and the OS on the machine (previously inaccessible on the encrypted storage until it was unlocked) boots the Mac. Once the Mac is unlocked and booted, the user doesn't have the issues with backups and network accounts that FileVault 1 users had because, from the OS's perspective, everything on the hard drive it needs to access is accessible.

FileVault 2 and Recovery HD

- One of the other new features in Lion is the Recovery HD partition.
- Small hidden partition that provides tools to fix or reinstall Lion.
- To use FileVault 2, you need to have the Recovery HD partition present.
- Why? Because Recovery HD provides the unencrypted space needed to unlock and boot your encrypted Mac.

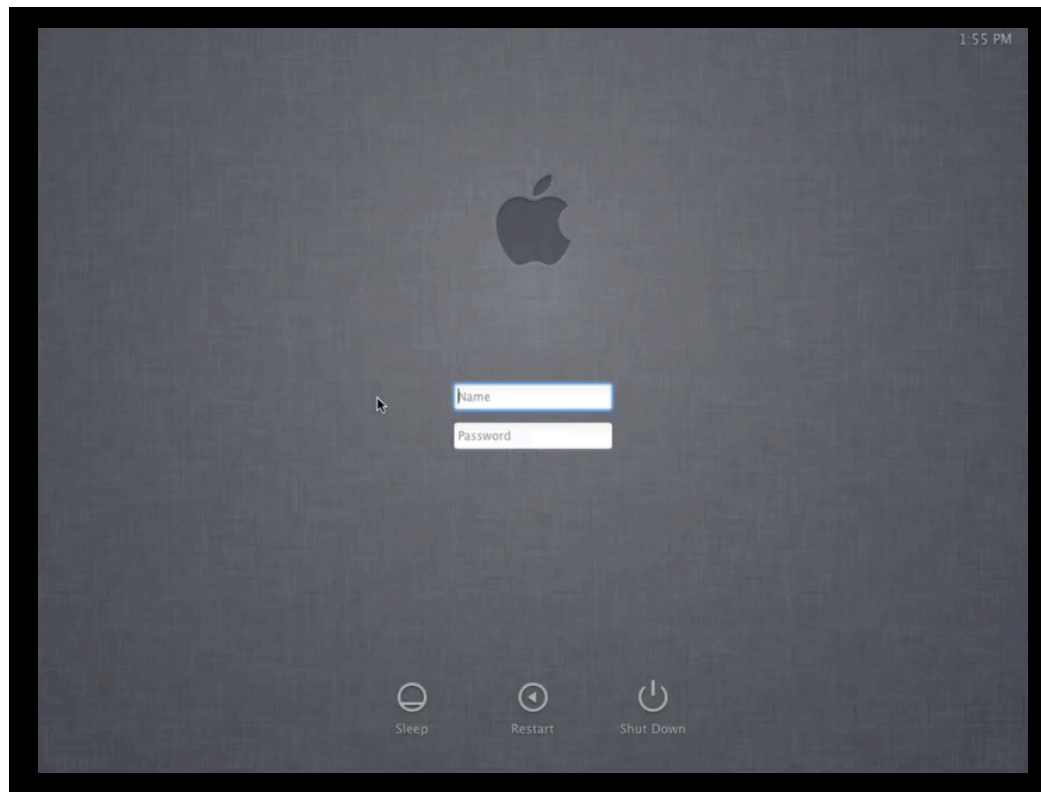
Now, because not everything about FileVault 2 is obvious at first glance, I want to make sure to cover one of the new less-known features in 10.7, the Lion Recovery feature. The general idea is that you can boot from the hidden Recovery HD partition on your hard drive. Once booted to it, you'll have access to all of the tools you need to reinstall Lion, repair your disk, and even restore from a Time Machine backup.

So, what does Recovery HD have to do with FileVault 2? FileVault 2 encrypts your boot partition, but your Mac still needs an unencrypted space to boot to and allow access to the unlock tools. The recovery partition serves as the needed space. The FileVault encryption process will check before beginning the encryption to see if the recovery partition is there and will not start the encryption process if it's not there.



During the demo, the following points will be covered:

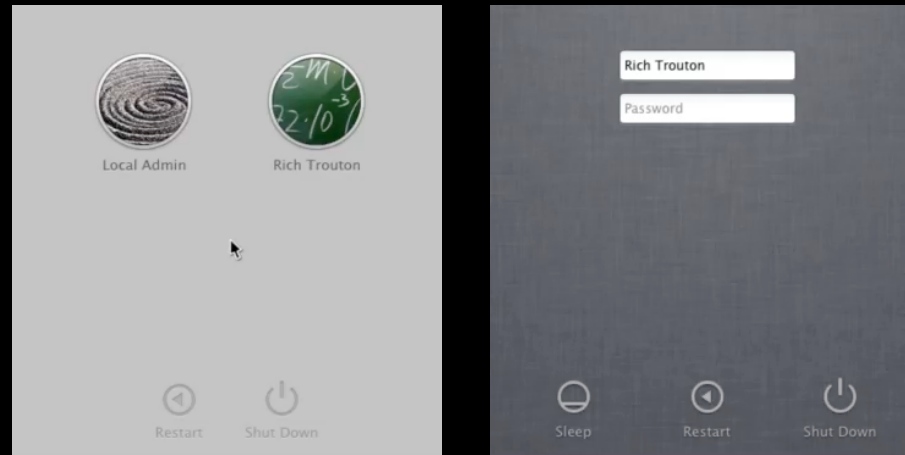
1. Going through the procedure of setting up FileVault 2 with an individual recovery key.
2. Rebooting and showing the pre-boot login screen.



During the demo, the following points will be covered:

1. Going through the procedure of setting up FileVault 2 with an individual recovery key.
2. Rebooting and showing the pre-boot login screen.

Using the Recovery Key To Reset Password



The chief use of the FileVault 2 individual recovery key is to help you reset your account password in the event that you forget what your password is. To use the recovery key at the pre-boot login screen, enter your password incorrectly three times. After that, the login screen should prompt you for the recovery key.

An important thing to note here is that this functionality was built to reset the password of a local account. If you have a network account, where your password is being managed by a directory service like Active Directory or Open Directory, resetting your password using this method will likely break the password sync between your account on the Mac and the directory service. In that event, you will need to delete and re-setup the account on the Mac in order to fix this.

When you enter the recovery key, FileVault 2 will take you to the regular login window and prompt you to reset your password. Once the password is reset, the rest of the login process will complete.

Managed Deployment

- With the exception of how the recovery key is generated and handled, setting up FileVault for home use and setting it up for a managed deployment is exactly the same.
- To avoid the administrative headache of tracking multiple individual recovery keys, Apple brought one part of FileVault 1 into FileVault 2.

For those of you who want to roll out a managed deployment of FileVault 2, deploying and supporting FileVault 2 in a larger Mac environment is almost exactly the same as setting it up for an individual. Apple recognized that it would be a administrative headache to track multiple individual recovery keys, so the sole difference between managed and unmanaged is how the recovery key is generated. That recovery key system is the one part of FileVault 1 that has survived in FileVault 2.

Sole Survivor - FileVaultMaster.keychain

- What does FileVaultMaster.keychain do?
 - It sets a backdoor to encrypted Macs and is an alternate way to unlock the encryption when the account passwords don't work.
 - Apple calls this the Master Password.
- What role does the password you set as the Master Password actually play?
 - It's the password used to unlock the FileVaultMaster.keychain.



What does FileVaultMaster.keychain do?

FileVaultMaster.keychain serves as a centrally-set backdoor. In the event that you weren't able to get into an encrypted Mac using the usual passwords, you could use FileVaultMaster.keychain to unlock the encryption and recover the data stored inside. In the FileVault context, Apple historically called this a "master password". This master password is in fact the password to a FileVault-specific keychain.

What role does the password that you set as the Master Password play?

It is the password used to secure the FileVaultMaster.keychain itself. Because this password is only used to unlock the keychain, this password can be rotated without affecting the contents of that keychain (much like how changing your account password doesn't affect the contents of your login keychain.)

Sole Survivor - FileVaultMaster.keychain

- Wait, what? If all the Master Password does is unlock a keychain, how does it do recovery?
 - FileVaultMaster.keychain's contents are what actually do the recovery.
 - Inside the keychain is a public key (shows up as a SSL certificate) and an accompanying private key.
 - When you have both keys available in the FileVaultMaster keychain, you can use them to unlock or decrypt FileVault 2's encryption.



FileVaultMaster.keychain is a FileVault-specific keychain, whose only contents are an SSL certificate (referred to in this talk as the public key) and an accompanying private key. These two keys are specifically used for FileVault certificate-based authentication.

When both are available, you can use them to unlock or decrypt FileVault 2's encryption without knowing any of the passwords of the accounts authorized to log in at the pre-boot login screen.

Sole Survivor - FileVaultMaster.keychain

- You can set a recovery key for a FileVault 2 managed deployment in exactly the same way that you set the recovery key in FileVault 1.
- Crucial difference
 - In FileVault 1, you could have both the private and public key stored in the FileVaultMaster.keychain when you encrypted.
 - In FileVault 2, only the public key can be stored in FileVaultMaster.keychain when you encrypt your Mac.

For those who are experienced with a managed FileVault 1 deployment, setting a FileVault 2 recovery key with FileVaultMaster.keychain is exactly the same as setting it with FileVault 1. In fact, for those following Apple's recommended best practice, you don't need to change anything.

The reason I mentioned Apple's recommended best practice is that Apple has always recommended escrowing the FileVault private key somewhere outside of the encrypted Mac. However, in FileVault 1, you could have both the private and public key stored in FileVaultMaster.keychain and it worked fine.

In FileVault 2, this recommendation is now a requirement. If FileVault 2 detects both the private and public keys in FileVaultMaster.keychain, it makes the programmatic assumption that this is leftover from an older OS and ignores it. In this case, it'll default to setting up an individual recovery key.

Preparing FileVaultMaster.keychain

- Make your FileVaultMaster.keychain by setting the Master Password on a specific machine. (You can skip this step if you've already got a set FileVaultMaster.keychain)
 - This FileVaultMaster.keychain will contain both private and public keys.
- Next, make several copies of the FileVaultMaster.keychain file and store the copies in a secure place.
 - A locked safe would be a good place, or in an encrypted disk image on a secured file share.

If you do not already have a pre-built FileVaultMaster.keychain, you'll need to create one and add it to your machines before encrypting them. Build your keychain by setting the Master Password on a particular machine and copying the FileVaultMaster.keychain file off of it. At this point, the keychain will contain both the private and public keys needed for FileVault recovery.

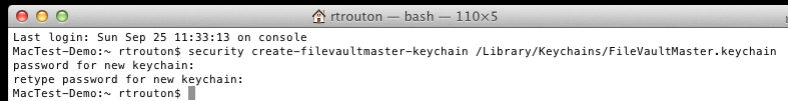
Once you have your keychain, make several copies and store them in a safe place. Everyone has a different conception of "safe", but I usually recommend storing them in a safe, or inside an encrypted disk image that is itself stored somewhere secure.

Preparing FileVaultMaster.keychain

In 10.7.2 and higher, you can create FileVaultMaster.keychain from the command line.

To create a FileVaultMaster.keychain:

security create-filevaultmaster-keychain /path/to/FileVaultMaster.keychain



```
rtROUTON — bash — 110x5
Last login: Sun Sep 25 11:33:13 on console
MacTest-Demo:~ rtROUTON$ security create-filevaultmaster-keychain /Library/Keychains/FileVaultMaster.keychain
password for new keychain:
retype password for new keychain:
MacTest-Demo:~ rtROUTON$
```

You'll be prompted to set a password. Please enter the Master Password you want to use at this point.

A new feature as of 10.7.2 is that you can use the security command to create a FileVaultMaster.keychain from the command line. This method creates a 2048-bit private key by default.

Preparing FileVaultMaster.keychain

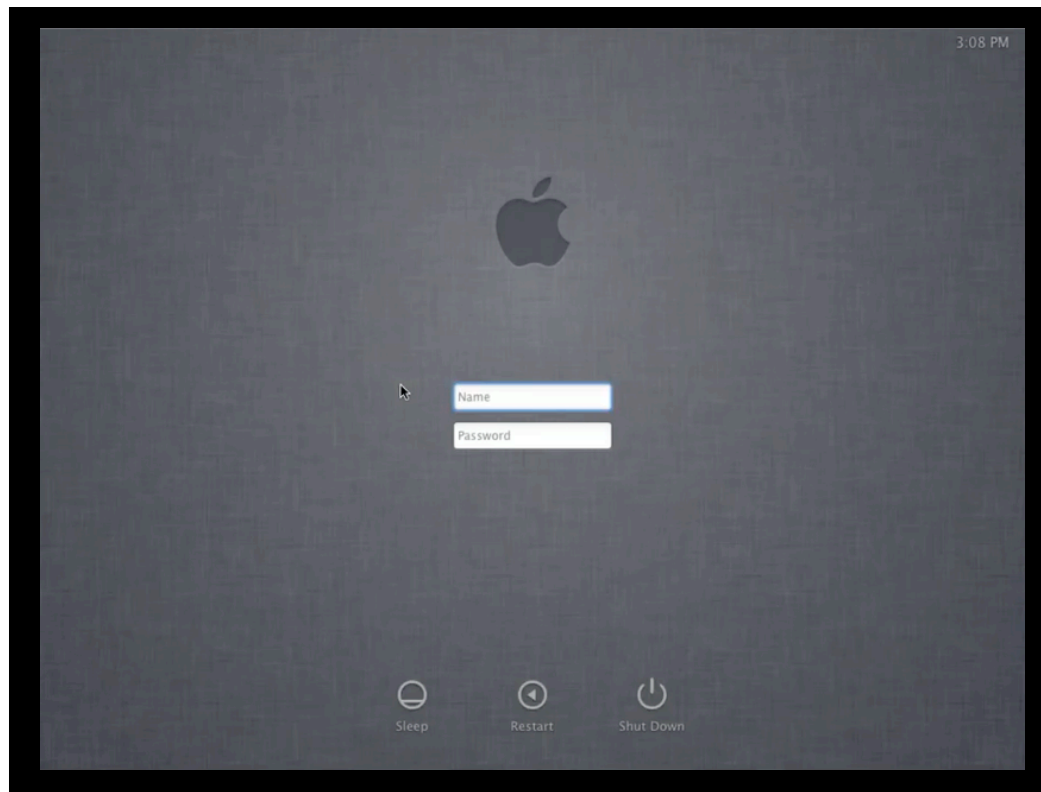
- Once you've got copies, unlock your FileVaultMaster.keychain by running the command below and entering the Master Password when prompted for the password:
security unlock-keychain /path/to/FileVaultMaster.keychain
- After FileVaultMaster.keychain unlocks, go into Keychain Access and access FileVaultMaster.keychain
 - Remove the private key. It will be called **FileVault Master Password Key** and its kind is listed as **private key**.

Once you've got your copies safely stowed, edit yet another copy of your keychain and remove the private key to prepare for use as your FileVault 2 recovery key.

Confused?



No problem.
Here's how to do this.

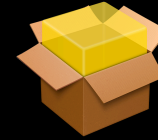


During the demo, the following points will be covered:

1. Going through the procedure of creating a managed recovery key for FileVault 2.
2. Showing the encryption process when using a managed recovery key.

Deploying FileVaultMaster.keychain

- You can deploy the prepared FileVaultMaster.keychain using a variety of methods.
 - Install it via an installer package
 - Include it with your image
 - Copy it to your Macs using your system management tool(s).
- A FileVaultMaster.keychain can be built once and then deployed to as many Macs as needed.



Once you have your FileVaultMaster.keychain ready, you can take that keychain and deploy it to as many machines as needed. If your worksite mandates password rotation, you can rotate the Master Password by updating the keychain with the new password and deploying the updated FileVaultMaster.keychain to your Macs. As long as the public key on your Macs and the private and public keys in your escrowed FileVaultMaster.keychain stay the same, the password to unlock FileVaultMaster.keychain can be updated as often as needed.

Disaster Recovery

Sometimes bad things happen
to good Macs

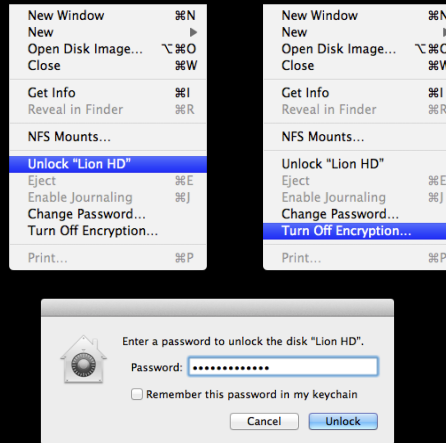
- If you have the password to an authorized account available, you can unlock and/or decrypt from Disk Utility or the command line.
- You can also use your recovery key to unlock and/or decrypt from the command line.

Disaster recovery is always something you should plan for when dealing with encrypted machines. After all, these are designed to protect your data against external threats unless properly authenticated. If your OS takes a dive, your normal way of unlocking may no longer work.

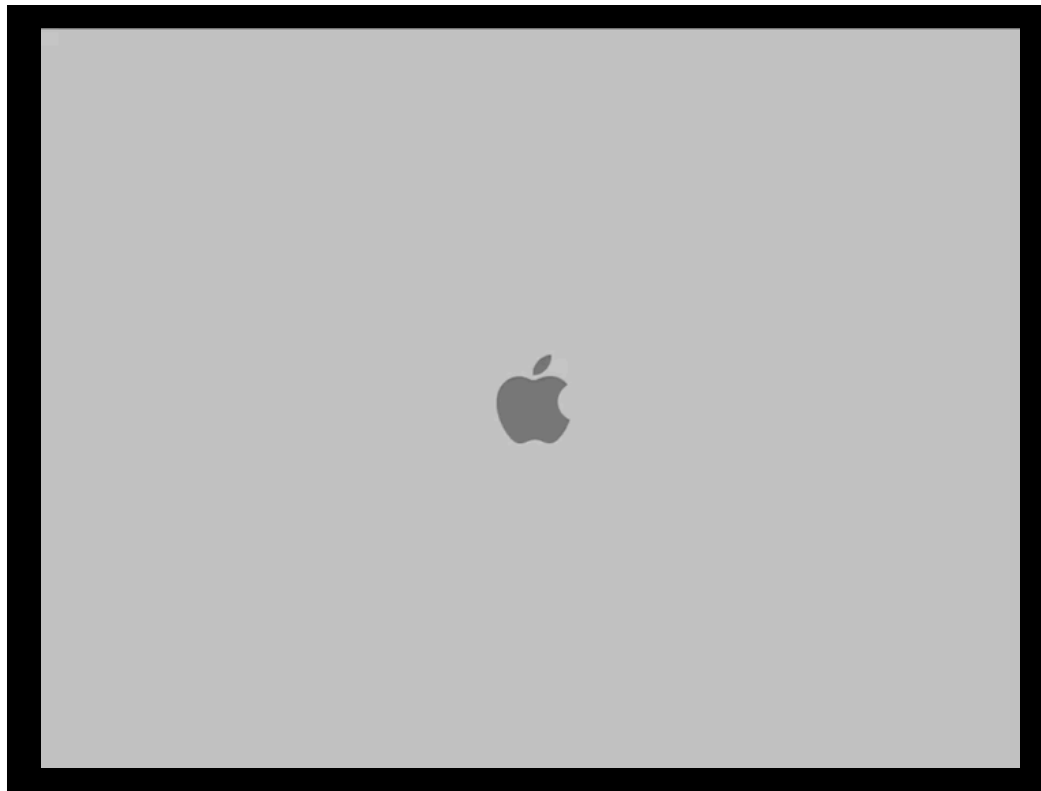
The first thing you should generally try is the password to one of the authorized accounts. This applies to both managed and unmanaged FileVault 2 encrypted Macs.

Recovering using your password

- Boot your Mac and hold down ⌘-R (Command – R) to boot from the Mac's Recovery HD partition.
- Use Disk Utility and the password of an authorized user to either unlock or turn off the encryption.



If you have the password to an authorized account, unlocking or even turning off the encryption is pretty straightforward. Boot to Recovery HD or another boot drive running 10.7, open Disk Utility and either select “Unlock” or “Turn off encryption”. You’ll be prompted for a password of an authorized account, you provide it, and you’re golden.



During the demo, the following points will be covered:

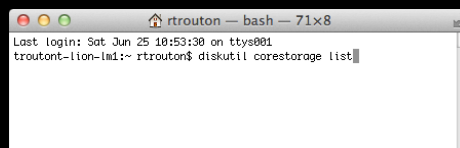
1. Booting to Recovery HD
2. Opening Disk Utility and using the password of an authorized user to unlock an encrypted drive.

Recovering from Terminal

Boot your Mac and hold down ⌘-R (Command -R) to boot from the Mac's Recovery HD partition.

Open Terminal and use the following to you need to identify the Logical Volume UUID of the encrypted drive.

diskutil corestorage list



```
rttrouton ~ bash — 71x8
Last login: Sat Jun 25 18:53:38 on ttys001
rttrouton-110n-lm1:~ rttrouton$ diskutil corestorage list
```

You can also recover from the Terminal, but you'll need a few more pieces of information. The first is to get the Logical Volume UUID of the drive you want to work on. You can do this by running “diskutil corestorage list”. You can also substitute “cs” in place of “corestorage” in commands.

Recovering from Terminal

Once you have the UUID, you'll use it to identify the disk to be unlocked or decrypted.

```
trouton@l1on-lal:~$ sudo fdisk -l
=====
Name:               L1on HD
Sequence:           1
Free Space:         0 B (0 B)
|
|<- Physical Volume 8C54C06F-4412-4138-BA26-2F226E65A811
|-----
| Index:            0
| Disk:             disk3s3
| Status:           online
| Size:             69477036832 B (69.5 GB)
|
|>>- Logical Volume Family 148BCFCC-21D8-4191-93A5-265AAE1EB578
|-----
| Sequence:         10
| Encryption Status: Locked
| Encryption Type:   AES-XTS
| Encryption Context: Present
| Conversion Status: Complete
| Has Encrypted Extents: Yes
| Conversion Direction: -none-
|
|>>- Logical Volume 0464C2D3-746F-4C63-B469-23C6E80034FF
|-----
| Disk:             -none-
| Status:           Locked
| Sequence:         4
| Size (Total):     69158264832 B (69.2 GB)
| Size (Converted): -none-
| Revertible:       Yes (unlock and decryption required)
| LV Names:         L1on HD
| Content Hint:     Apple_HFS
|
trouton@l1on-lal:~$
```

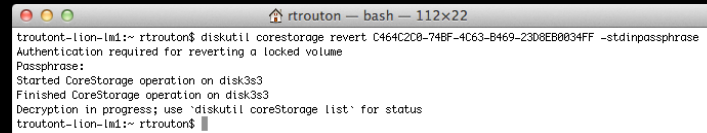
Once you have the UUID, you can then use it to specifically ID the drives you want to either unlock or decrypt. In the slide, I've highlighted what you should be looking for when checking for the UUID.

Recovering from Terminal

Unlocking or encrypting using your password

To unlock: *diskutil corestorage unlockVolume UUID -stdinpassphrase*

To decrypt: *diskutil corestorage revert UUID -stdinpassphrase*



```
trouton-lion-lal1:~ rtrouton$ diskutil corestorage revert C464C2C8-74BF-4063-B469-2308EB0834FF -stdinpassphrase
Authentication required for reverting a locked volume
Passphrase:
Started CoreStorage operation on disk3s3
Finished CoreStorage operation on disk3s3
Decryption in progress; use 'diskutil coreStorage list' for status
trouton-lion-lal1:~ rtrouton$
```

The *-stdinpassphrase* flag will cause the command to prompt you for the password/passphrase of an account that's authorized to unlock the encryption.

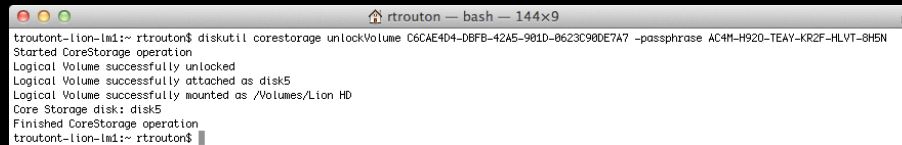
You can unlock or decrypt using the password of an authorized user using the commands shown on the screen.

Recovering from Terminal

Unlocking or decrypting with an individually-set recovery key

To unlock: *diskutil corestorage unlockVolume UUID -passphrase recoverykey*

To decrypt: *diskutil corestorage revert UUID -passphrase recoverykey*



```
trouton-lion-lm1:~ rtrouton$ diskutil corestorage unlockVolume C6CAE4D4-DBFB-42A5-991D-8623C980E7A7 -passphrase AC4M-H920-TEAY-KR2F-HLVT-8H5N
Started CoreStorage operation
Logical Volume successfully unlocked
Logical Volume successfully attached as disk5
Logical Volume successfully mounted as /Volumes/Lion HD
Core Storage disk: disk5
Finished CoreStorage operation
trouton-lion-lm1:~ rtrouton$
```

This command would be used only if FileVault 2 generated the recovery key. This would not apply if you're using a managed recovery key using FileVaultMaster.keychain for your recovery key.

You can also unlock or decrypt using the individually-set recovery key using the commands shown. This would apply if you're not managing the recovery key and FileVault 2 has generated it for you as part of the initial encryption process.

Recovering from Terminal

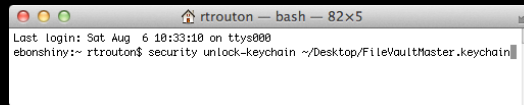
Unlocking or decrypting with a managed recovery key

First, copy the FileVaultMaster.keychain with the private key in the keychain from its secured place to a convenient place on the Mac.

Next, to allow the Mac to get access to both the keys inside, unlock the FileVaultMaster.keychain.

To unlock FileVaultMaster.keychain:

security unlock-keychain /path/to/FileVaultMaster.keychain

A screenshot of a macOS terminal window. The title bar reads "rtrouton -- bash -- 82x5". The terminal content shows "Last login: Sat Aug 6 10:33:10 on ttys000" followed by the prompt "ebonshiny:- rtrouton\$ security unlock-keychain ~/Desktop/FileVaultMaster.keychain" with a cursor at the end of the line.

```
rtrouton -- bash -- 82x5
Last login: Sat Aug 6 10:33:10 on ttys000
ebonshiny:- rtrouton$ security unlock-keychain ~/Desktop/FileVaultMaster.keychain
```

You'll be prompted a password. Please enter the Master Password at this point.

With the managed recovery key, it's a two step process. First, you'll to retrieve your keychain that has both your private and public keys from the safe place that you put it in. Second, you'll need to unlock that keychain so that the encrypted Mac has access to both keys.

Recovering from Terminal

Unlocking or decrypting with an managed recovery key

Once you've unlocked FileVaultMaster.keychain, you can unlock or decrypt the disk.

To unlock:

```
diskutil corestorage unlockVolume UUID -recoveryKeychain /path/to/FileVaultMaster.keychain
```

To decrypt:

```
diskutil corestorage revert UUID -recoveryKeychain /path/to/FileVaultMaster.keychain
```

As long as FileVaultMaster.keychain is unlocked, you should not be prompted for a password.

Once you've unlocked the keychain with both the private and public keys inside, you'll run the commands shown to either unlock or decrypt the disk. Since FileVault 2 now has access to both keys, you shouldn't need to provide any other passwords.



During the demo, the following points will be covered:

1. Booting to Recovery HD
2. Opening Terminal and using the FileVaultMaster.keychain on a separate drive to unlock an encrypted drive.

The Guest User

- As of 10.7.2, some people started seeing an account named **Guest User** appear at the pre-boot login screen.
- When selected, there was no password needed.
- It took you into Safari. Nothing was available but Safari and the network.



One thing that's appeared as of 10.7.2 is that FileVault is sometimes enabling a guest user at the pre-boot login screen. When you log in as that guest user, you don't get access to your hard drive. In fact, the only thing you get access to is Safari and a network connection.

Why is it here? Also, why does it appear on some encrypted machines and not others?

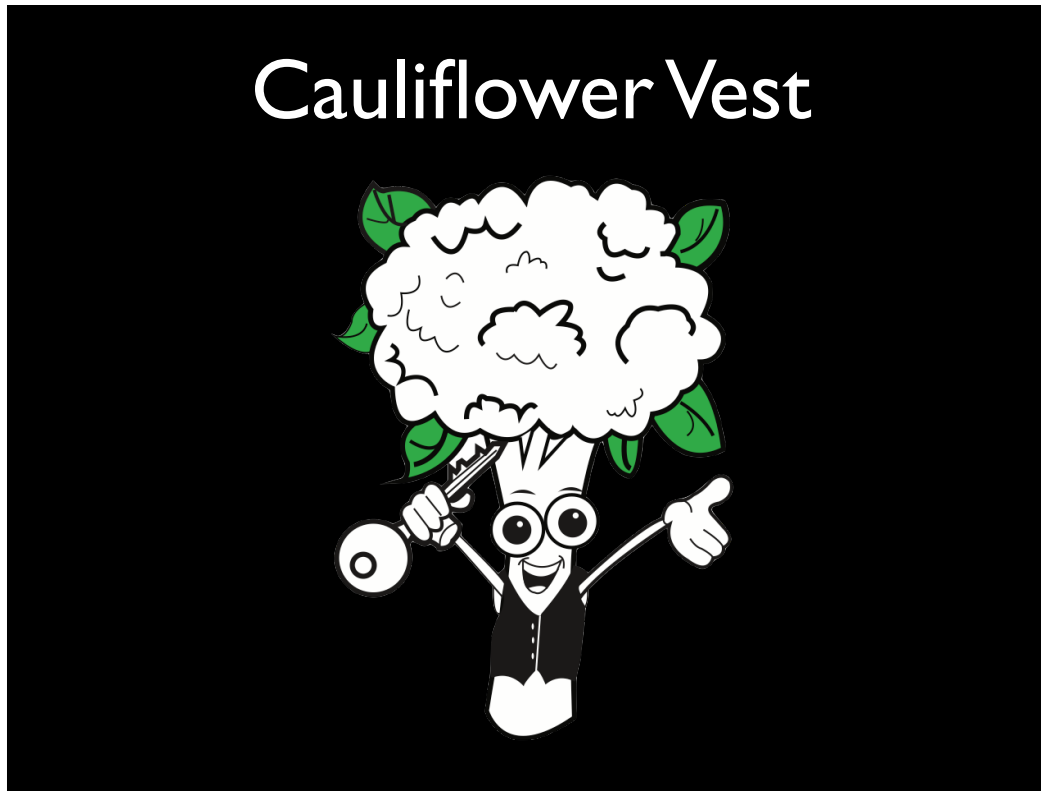
The Guest User

- The **Guest User** account appears when you do two things:
 - Sign into iCloud on the Mac.
 - Enable **Find My Mac**.
- When you log in with the **Guest User** account, **Find My Mac** phones home with the Mac's location.



Long story short, in my opinion, the guest user is a trap for anyone stealing your machine. The guest user at the pre-boot login screen is a feature tied to signing into iCloud and enabling Find My Mac. If you don't sign in with iCloud and then enable Find My Mac from that machine, you don't get that guest user login at the FileVault pre-boot login screen.

Cauliflower Vest



Let's talk about Cauliflower Vest. Aside from having a curious name and an unusual mascot, what is it?

Cauliflower Vest

- Google wanted to use FileVault 2 for its preferred Mac encryption solution
 - Strong whole disk encryption
 - Came with the OS
 - Low likelihood that future Apple updates would break Apple's own encryption solution.

Google wanted to use Apple's FileVault 2 encryption for the reasons shown on the screen. It was good, it was free with the OS and unlike a number of other third party encryption solutions, there was a very low probability that a future Apple OS or security update would render the encrypted Mac unbootable.

Cauliflower Vest

- Google needed to build additional features onto FileVault 2 for their enterprise users
 - No way to enable encryption of the boot volume from the command-line
 - Managed recovery key was a shared key
 - Encryption designed for the “voluntarily encrypted”

When Google took a look at FileVault 2's capabilities and compared them to their needs, they saw that they needed to build upon FileVault 2's current capabilities.

You can monitor, unlock or decrypt a FileVault 2-encrypted boot drive using command line tools, but you can't start the encryption process from the command line. Instead, the encryption needs to be enabled from System Preference's FileVault preference pane.

Another issue has to do with recovery keys. Apple has built in an excellent way to have each encrypted Mac generate its own recovery keys. Apple has even built a secure way for you to call Apple, answer some security questions and get your recovery key back in the event that you no longer have it. However, the only place where you can store your recovery keys in that fashion is with Apple itself. There is no Apple-provided functionality for setting up your own server for storing those keys. Instead, Apple allows their customers to set up a managed recovery key using Apple's FileVaultMaster keychain, which is a file created on one Mac and then copied to the other Macs you want to encrypt. While this is a fairly good solution, it does mean that multiple Macs are sharing one common recovery key instead of each having their own.

Lastly, Apple has built an encryption solution for the voluntarily encrypted. Unlike a number of other full disk encryption vendors, Apple makes it very easy for FileVault 2-enabled users to unlock or even decrypt their encrypted Macs. Apple's solution is focused on keeping the bad guys out, not keeping authorized users in. Once users figure out how to decrypt themselves, the IT staff who needs to maintain encryption policy compliance would need to chase those users down and get their machines re-encrypted.

Google's Macintosh Operations team needed to address those issues. In short, they succeeded and in turn have launched an open source project to bring their FileVault 2 solution for enterprise to the greater Mac community. That open source project is Cauliflower Vest. The name itself is an anagram for "FileVault Escrow".

Cauliflower Vest

- What problems does Cauliflower Vest solve?
 - Allows individual recovery keys to be automatically generated and escrowed for each Mac.
 - Allows FileVault 2 encryption to be force-enabled on a Mac
 - Provides secure access to recovery keys and delegating secure access as needed to those recovery keys

Cauliflower Vest is a solution for Mac admins who want to use FileVault 2 as a full disk encryption solution for their enterprise. As currently designed, Cauliflower Vest does the following:

- A. Allows individual recovery keys to be automatically generated and escrowed for each Mac.
- B. Allows FileVault 2 encryption to be force-enabled on a Mac
- C. Provides secure access to recovery keys for authorized admins.

Cauliflower Vest is a three-part solution with the following components:

1. **csfde**, a command line tool which interfaces with Apple's built-in functionality to activate FileVault 2 encryption on the client Mac.
2. A Google App Engine-based service which receives and securely escrows FileVault 2 recovery keys.
3. A GUI client running on the managed Macs, which works with csfde to enable FileVault 2 encryption, get the Mac's recovery key, and send it to the App Engine-based recovery key server.

All three parts were designed by Google to work together, but it is possible for the **csfde** command line tool to be used completely independently of any server infrastructure.

Google App Engine



- What is Google App Engine?
 - Google-hosted cloud platform for web applications.
 - Automatically scales to meet web application demand.
 - Languages supported include Python, Java, and Go.

Before we continue on to talk about Cauliflower Vest's component parts, I want to talk briefly about Google App Engine, as it's the server backend that Cauliflower Vest's server application runs on.

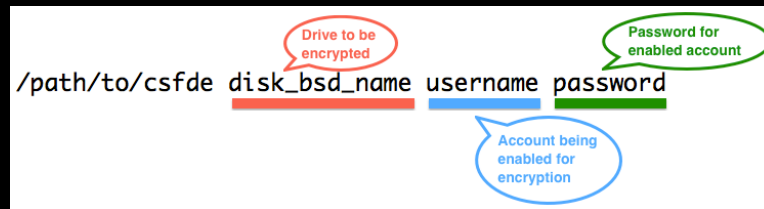
Google App Engine is a cloud web application hosting platform. It runs entirely on Google's infrastructure, in Google's data centers, and is monitored and maintained by Google engineers.

App Engine offers automatic scaling for web applications. As the number of requests increase, App Engine automatically allocates more resources for the web application to handle the additional demand.

One of the nicer parts of using App Engine is that it's free to get started. Anyone with a Google account can create an App Engine instance and start working with Cauliflower Vest. The level of resources you get with a free instance is likely sufficient for testing, but a large-scale enterprise rollout will require enough resources that Google will start charging for them.

csfde

Running **csfde** in normal standalone usage



The diagram shows the command `/path/to/csfde disk_bsd_name username password`. The path is in black, `disk_bsd_name` is underlined in red with a callout bubble saying "Drive to be encrypted", `username` is underlined in blue with a callout bubble saying "Account being enabled for encryption", and `password` is underlined in green with a callout bubble saying "Password for enabled account".

If password is set as "-", **csfde** will prompt you for the account's password.

To fully understand how the whole Cauliflower Vest package works together, it's best to look at `csfde` first and see how it works. In normal standalone usage, here's how to run `csfde` and have it enable FileVault 2 encryption on your boot drive:

The disk ID can be obtained by running the `diskutil list` command on your Mac and then getting the disk identifier from the list produced.

The username part of the command is to specify the account that you want to have enabled as part of the FileVault 2 encryption. The password is the password for the account being enabled.

If the password is supplied as a hyphen, `csfde` will prompt you for the account's password. One thing to be aware of is that the password will be displayed, so make sure you're comfortable with the password being shown on the screen after it's entered.

csfde

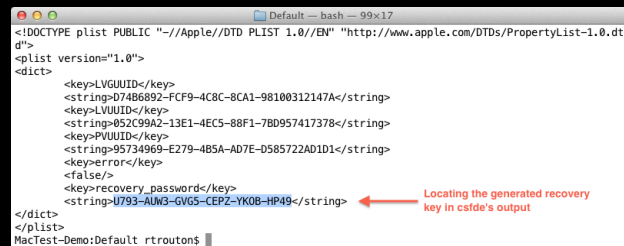
- After you hit enter, the following process occurs:
 - **csfde** resolves the disk identifier provided into an identifier that FileVault 2 can use.
 - **csfde** then uses undocumented functions in Apple's `/usr/lib/libcsfde.dylib` to create a full disk encryption recipe for FileVault 2.
 - **csfde** passes the full disk encryption recipe to the Apple CoreStorage **convertDisk** method.
 - Encryption is initialized, though the encryption itself begins after a reboot.

After you enter your `csfde` command and hit enter, the following process then takes place:

1. `csfde` resolves the disk ID into an identifier that FileVault 2 can use.
2. `csfde` then creates a full disk encryption recipe for FileVault 2 by supplying the disk ID, user credentials, and other static data to Apple's `libcsfde` dynamic library.
3. `csfde` passes the full disk encryption recipe to Apple's CoreStorage `convertDisk` method. This initializes the encryption process, though actual encryption does not begin until the next time the Mac restarts.

csfde

- CoreStorage calls back to **csfde** to supply it with the recovery key and volume ID for the volume to be encrypted.
- **csfde** checks return status and the information that CoreStorage has supplied.
- **csfde** exits and displays the recovery key information and other information.



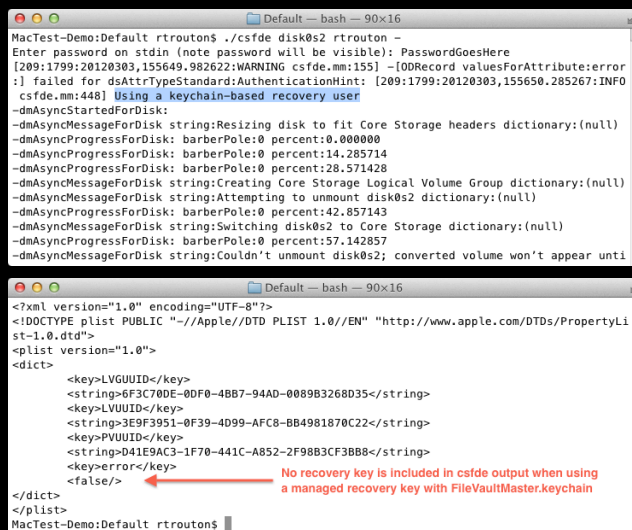
```
Default — bash — 99x17
<!DOCTYPE plist PUBLIC "-//Apple/DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
  <key>LVGUID</key>
  <string>074B6892-FCF9-4C8C-8CA1-98100312147A</string>
  <key>LVUUIID</key>
  <string>852C99A2-13E1-4EC5-88F1-7B0957417378</string>
  <key>PVUUIID</key>
  <string>95734969-E279-4B5A-AD7E-D585722AD1D1</string>
  <key>err</key>
  <false/>
  <key>recovery_password</key>
  <string>U793-AUW3-GV65-CEPZ-YK0B-HP49</string>
</dict>
</plist>
MacTest-Demo:Default rtrouton$
```

Locating the generated recovery key in csfde's output

4. CoreStorage calls back to csfde to supply it with information. This information includes the recovery key and the volume ID for the drive being encrypted.

5. csfde checks both the encryption status and the information that CoreStorage has supplied. It then exits and displays the recovery key and other relevant information to the user.

Using csfde with a managed recovery key

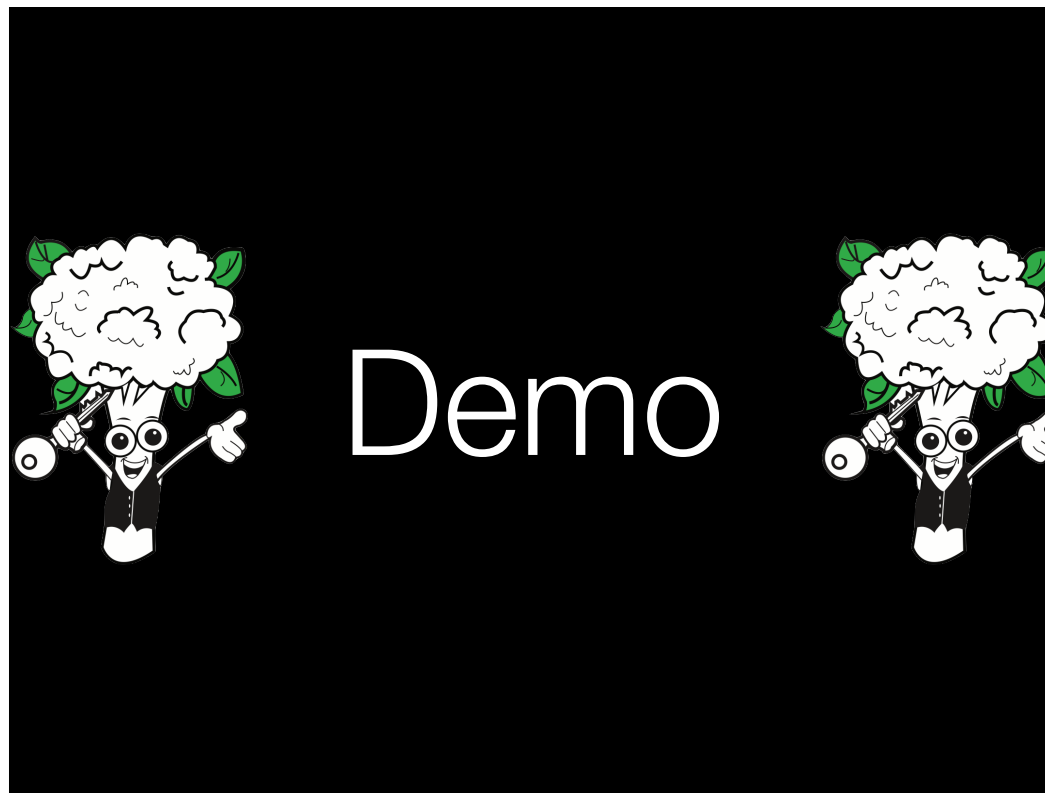


The image shows two terminal windows. The top window shows the execution of the command `./csfde disk0s2 rtrouton -`. It prompts for a password, displays a warning, and then shows progress messages for disk resizing and Core Storage operations. The bottom window shows the XML output of the command, which includes a dictionary of keys. A red arrow points to the `<key>error</key>` entry, with a red text annotation stating: "No recovery key is included in csfde output when using a managed recovery key with FileVaultMaster.keychain".

```
MacTest-Demo:Default rtrouton$ ./csfde disk0s2 rtrouton -
Enter password on stdin (note password will be visible): PasswordGoesHere
[209:1799:20120303,155649.982622:WARNING csfde.mm:155] -[ODRecord valuesForAttribute:error
:] failed for dsAttrTypeStandard:AuthenticationHint: [209:1799:20120303,155650.285267:INFO
csfde.mm:448] Using a keychain-based recovery user
-dmAsyncStartedForDisk:
-dmAsyncMessageForDisk string:Resizing disk to fit Core Storage headers dictionary:(null)
-dmAsyncProgressForDisk: barberPole:0 percent:0.000000
-dmAsyncProgressForDisk: barberPole:0 percent:14.285714
-dmAsyncProgressForDisk: barberPole:0 percent:28.571428
-dmAsyncMessageForDisk string:Creating Core Storage Logical Volume Group dictionary:(null)
-dmAsyncMessageForDisk string:Attempting to unmount disk0s2 dictionary:(null)
-dmAsyncProgressForDisk: barberPole:0 percent:42.857143
-dmAsyncMessageForDisk string:Switching disk0s2 to Core Storage dictionary:(null)
-dmAsyncProgressForDisk: barberPole:0 percent:57.142857
-dmAsyncMessageForDisk string:Couldn't unmount disk0s2; converted volume won't appear unti

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyLi
st-1.0.dtd">
<plist version="1.0">
<dict>
  <key>LVGUID</key>
  <string>6F3C70DE-0DF0-4BB7-94AD-0089B3268D35</string>
  <key>LVUID</key>
  <string>3E9F3951-0F39-4D99-AFC8-BB4981870C22</string>
  <key>PVUID</key>
  <string>D41E9AC3-1F70-441C-A852-2F98B3CF3BB8</string>
  <key>error</key>
  <false/>
</dict>
</plist>
MacTest-Demo:Default rtrouton$
```

One important thing to know is that if you are using a managed recovery key using FileVaultMaster.keychain, csfde will recognize this, use the managed recovery key, and report it in its output.



During the demo, the following points will be covered:

1. Running csfde from the command line.
2. Showing the encryption process.



Scripting with csfde

```
Last login: Wed Mar 14 20:28:07 on ttys000
/Users/rtrouton/Desktop/Interactive_Encrypt_With_FileVault_2/Interactive_Encrypt_With_File
Vault_2.command ; exit;
MacTest-Demo:~ rtrouton$ /Users/rtrouton/Desktop/Interactive_Encrypt_With_FileVault_2/Inte
ractive_Encrypt_With_FileVault_2.command ; exit;
***** Running Interactive_Encrypt_With_FileVault_2.command Version 1.0 *****

*** This application must be run as root. Please authenticate below. ***

Password:█
```

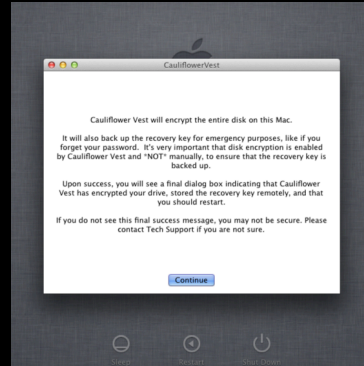
Since csfde is a command-line tool, you can use it in a script to automate the initialization of your boot volume's FileVault 2 encryption. Here's an example of an interactive script, where the prompts guide the user through the process of enabling a user for use with FileVault 2 and initializing the encryption.

This script is available for anyone to use. I'll be giving out a link to it at the end of the talk.

Ultimately, csfde has one function: it starts FileVault 2 encryption of the current boot volume and enables a single user account as part of the encryption initiation process. How Google has worked this functionality into an enterprise-grade encryption management tool has to do with how the Cauliflower Vest GUI application leverages csfde on the client end, so let's look at that next.

Cauliflower Vest GUI

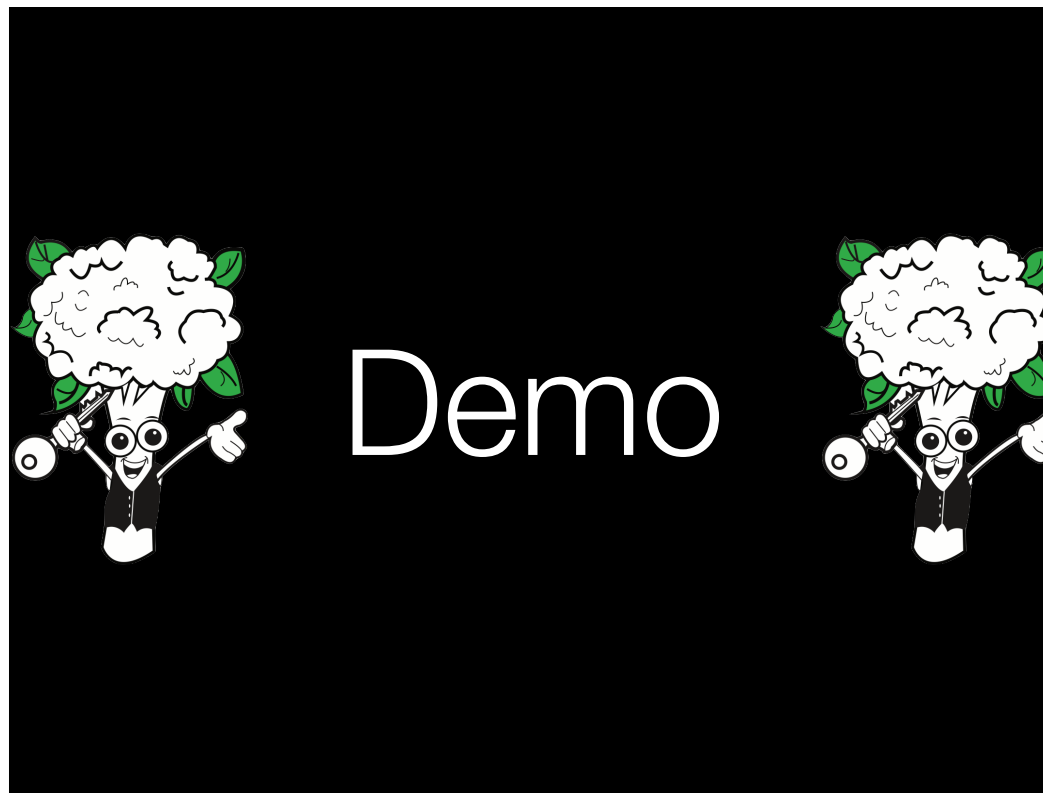
- Provides a GUI interface for end-user self-service.
- Allows FileVault 2 encryption to be force-enabled on unencrypted Macs
- Sends recovery key information to server



Cauliflower Vest GUI application

To make sure that their Macs were encrypted, Google built a GUI application that works with the csfde tool and the Cauliflower Vest App Engine service. This GUI application does the following:

1. Provides a user-friendly interface for people to encrypt themselves with FileVault 2
2. Allows FileVault 2 encryption to be force-enabled on a non-encrypted Mac
3. Gathers the recovery key from csfde and sends it up to the Cauliflower Vest server.



During the demo, the following points will be covered:

1. Going through the procedure of setting up FileVault 2 encryption with Cauliflower Vest's GUI application and csfde.
2. Rebooting and showing the pre-boot login screen.



Name

Password



Sleep



Restart

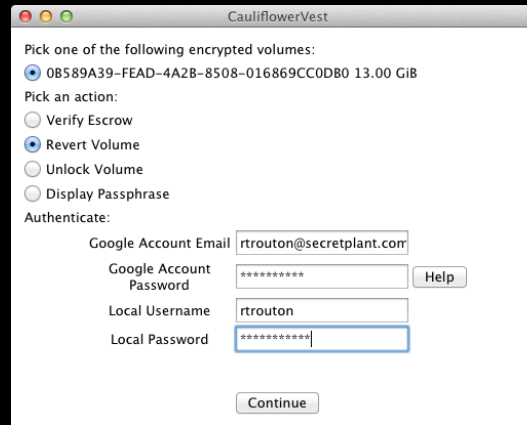


Shut Down

Cauliflower Vest GUI

Post-encryption, launch the GUI app with following command:

`/usr/local/bin/cauliflowervest`



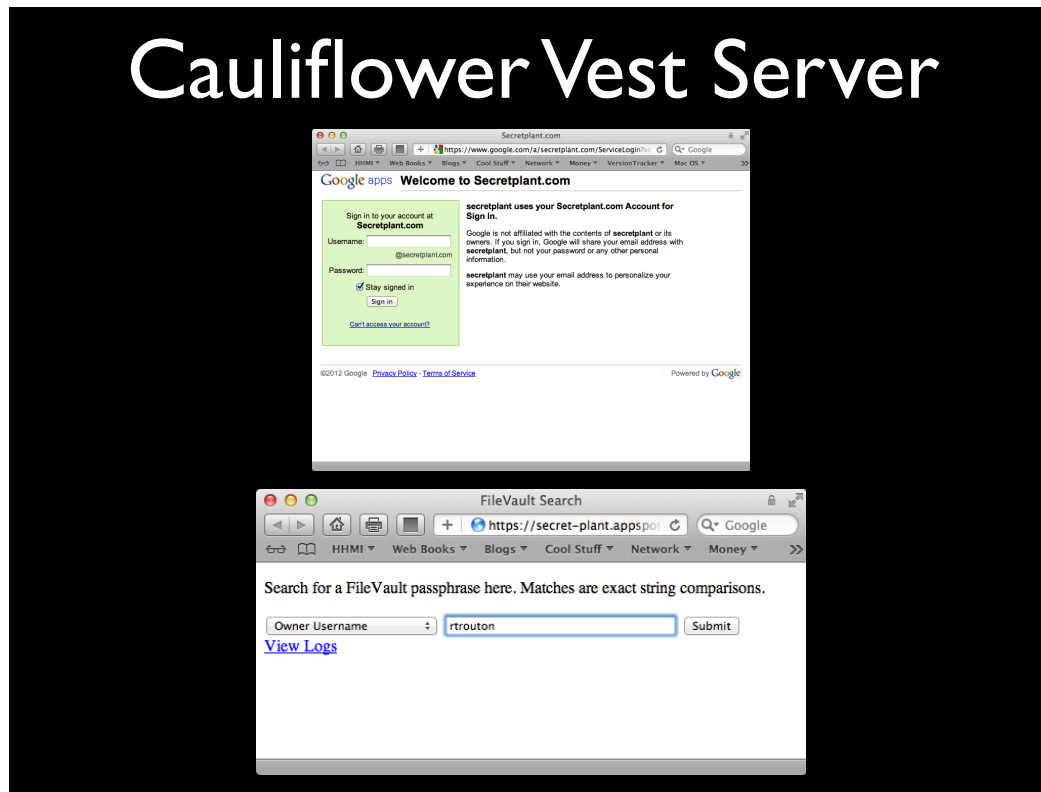
Once the Mac has been encrypted, you can also launch the Cauliflower Vest GUI application by using the following command:

`/usr/local/bin/cauliflowervest`

Once the application is launched, the user is prompted to log in. Once authenticated, the user can use the GUI application to access the Mac's encryption recovery key, unlock encrypted volumes or decrypt the encrypted boot drive.

While the end-user can decrypt their own Mac easily, Cauliflower Vest's design ensures that this state of affairs does not continue past the next logout or reboot. Once the Mac is back at the login window, the same functionality that triggered the initial encryption of the Mac will again detect that it's running on an unencrypted Mac. The Cauliflower Vest GUI application will again launch and force the FileVault 2 encryption of the Mac's boot volume.

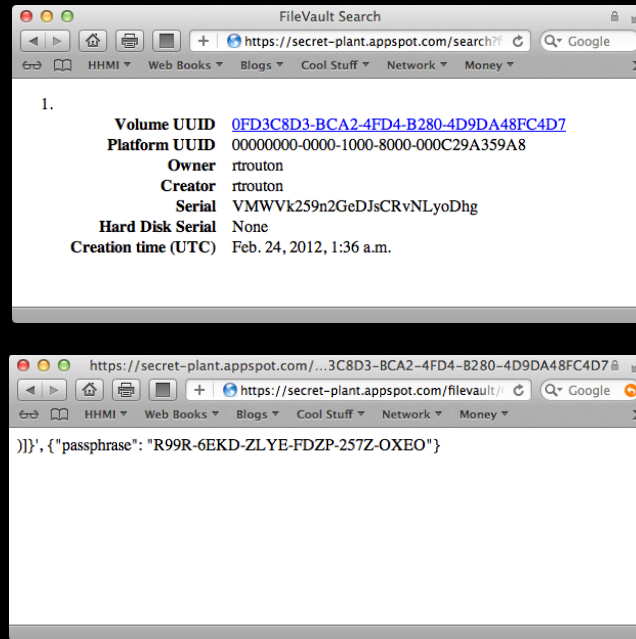
Cauliflower Vest Server



Since encryption is only half of what Cauliflower Vest provides, let's look next at the Cauliflower Vest App Engine service. This service acts as the secure repository for the escrowed recovery key sent from the client Macs and as a central web-based interface for authorized users to look up recovery keys.

To log in, first go to the address of your organization's App Engine service. After authenticating, you'll be taken to your organization's Cauliflower Vest site. At the site, you will have various search options to locate the machine in question. In this example, we'll be looking up the machine information by using the Owner Username selector.

Cauliflower Vest Server



Once the listing for the encrypted Mac comes up, you can click on the Volume UUID link shown in the top picture. Once clicked, you'll have access to the selected Mac's recovery key as shown in the picture below.

Using Cauliflower Vest in your environment

- Your enterprise will need to decide if using Google's App Engine meets your organization's security needs for an enterprise encryption management solution.
- Google has provided some guidance on how to port the Cauliflower Vest server to another server platform.

In my working with it, I've found that Cauliflower Vest solves the problems that it was built to solve. The main possible downside is that Cauliflower Vest's GUI client and server backend is closely tied to Google's App Engine and it will likely take some work for most enterprise environments to get it running entirely in-house.

That said, Google has recognized this and provided some guidance on how to port the Cauliflower Vest server on the project's wiki. As with all open source, if you don't like what you have now, you can take what's provided and roll out your own solution.

Tips for Improving FileVault 2 Performance

- FileVault 2 is able to encrypt and decrypt SSDs much faster than conventional hard drives.
- Processors that have built-in AES-NI support provide additional performance improvements for FileVault 2-encrypted Macs.

Here's some tips I've learned for improving FileVault 2's performance.

1. SSDs are awesome with FileVault 2 - To give a basis of comparison for how much faster you can encrypt an SSD, I ran a test in the same 15 inch Core 2 Duo laptop using a 256 GB SSD and a 256 GB hard drive. The SSD encrypted in about 30 minutes. The hard drive took four hours. Decryption showed the same speeds. That's anecdotal evidence, but it gives an idea of how much better performance you can get with SSDs specifically when used with FileVault 2.

2. Processors that have built-in AES-NI support provide additional performance improvements to FileVault 2 - AES-NI support in the processor improves the speed of applications performing AES encryption and decryption. FileVault 2 uses AES-XTS encryption, so it also benefits from having AES-NI support built into the Mac's processor.

Limitations of FileVault 2

- FileVault 2 is an overall better solution than FileVault 1 is, but it does not necessarily cover all workplaces' requirements for full disk encryption.
 - Can't use remote management tools at the pre-boot login screen
 - Can't use authentication methods other than password/passphrases
 - Pre-boot login screen can't display username / password blanks

FileVault 2 is an overall better solution than FileVault 1 is, but it does not necessarily cover all workplaces' requirements for full disk encryption.

Among the things you cannot currently do with FileVault 2 are the following:

1. Use remote management tools at the pre-boot login screen – All current remote management tools require the operating system to be running. The OS is not running at the pre-boot login screen, so there's no way to run these tools.
2. Use authentication methods other than password/passphrases – At this time, the EFI boot environment does not support the use of encryption tokens such as smart cards or USB encryption dongles to unlock FileVault 2's encryption.
3. You cannot set the FileVault pre-boot login screen to display username and password blanks. It only allows for the account icons.

FileVault 2 and the Law

- For folks who need to satisfy regulatory requirements for encryption:
 - FileVault 2 is not FIPS 140-2 validated.
 - FileVault 2's underlying low level encryption uses Apple's new Common Crypto implementation
 - Common Crypto is just starting to undergo FIPS evaluation

For those folks who want to use FileVault 2 in a government or other heavily-regulated environment, be aware that FileVault 2 is not currently certified by the US Government's FIPS 140-2 encryption standard.

For those who have not heard of it before, the FIPS 140-2 standard is a security accreditation program for cryptographic modules certified for use in US and Canadian government departments and regulated industries that collect, store, transfer, share and disseminate sensitive information. Apple is working on FIPS 140-2 certification for Lion's new Common Crypto cryptography foundation, which would also cover FileVault 2, but the certification process itself takes years to complete.

FileVault 2 and the Law

- If you're planning to roll out FileVault 2 in your regulated environment:
 - Make sure it meets your workplace's regulatory requirements.
 - Double-check with your data-protection folks.
 - Be prepared to do some justification writing.

If you're planning to deploy FileVault 2 in your regulated Mac environment, make sure that it meets your needs first. When in doubt, check with the folks who are responsible for your data-protection policies to make sure that FileVault 2's capabilities meets those requirements. However, if your workplace had previously certified FileVault 1 as an acceptable encryption solution, FileVault 2 should also be fine.

Useful Links

- OS X Lion: About FileVault 2 - <http://support.apple.com/kb/HT4790>
- Using Institutional Keys with FileVault 2 - <http://www.afp548.com/article.php?story=FileVault-2-Keys>
- Using a login banner with FileVault 2 - <http://tinyurl.com/mactechfv2-1>
- Displaying expiring password notifications when using FileVault 2 with Active Directory accounts - <http://tinyurl.com/mactechfv2-2>

Here's some useful links for FileVault 2, including links to some topics not discussed as part of today's talk.

Useful Links

- Cauliflower Vest project - <http://code.google.com/p/cauliflowervest/>
- Cauliflower Vest Introduction – <http://code.google.com/p/cauliflowervest/wiki/Introduction>
- Csfde - <http://code.google.com/p/cauliflowervest/wiki/Csfde>
- User Admin - <http://code.google.com/p/cauliflowervest/wiki/UserAdmin>

Here's some links for Cauliflower Vest, including links to some topics not discussed as part of today's talk.

CSFDE Script

<http://tinyurl.com/csfdescript>

Here's a link for the interactive script shown earlier in the talk, where I was answering questions to encrypt a Mac using Cauliflower Vest's csfde tool.

For More Information

See the July and August 2011 issues of MacTech



MACTECH



July 2011 - FileVault Decrypted
August 2011 - FileVault Decrypted for Enterprise

Downloads

PDF available from the following link:

<http://tinyurl.com/MacAdmin2012PDF>

Keynote slides available from the
following link:

<http://tinyurl.com/MacAdmin2012key>