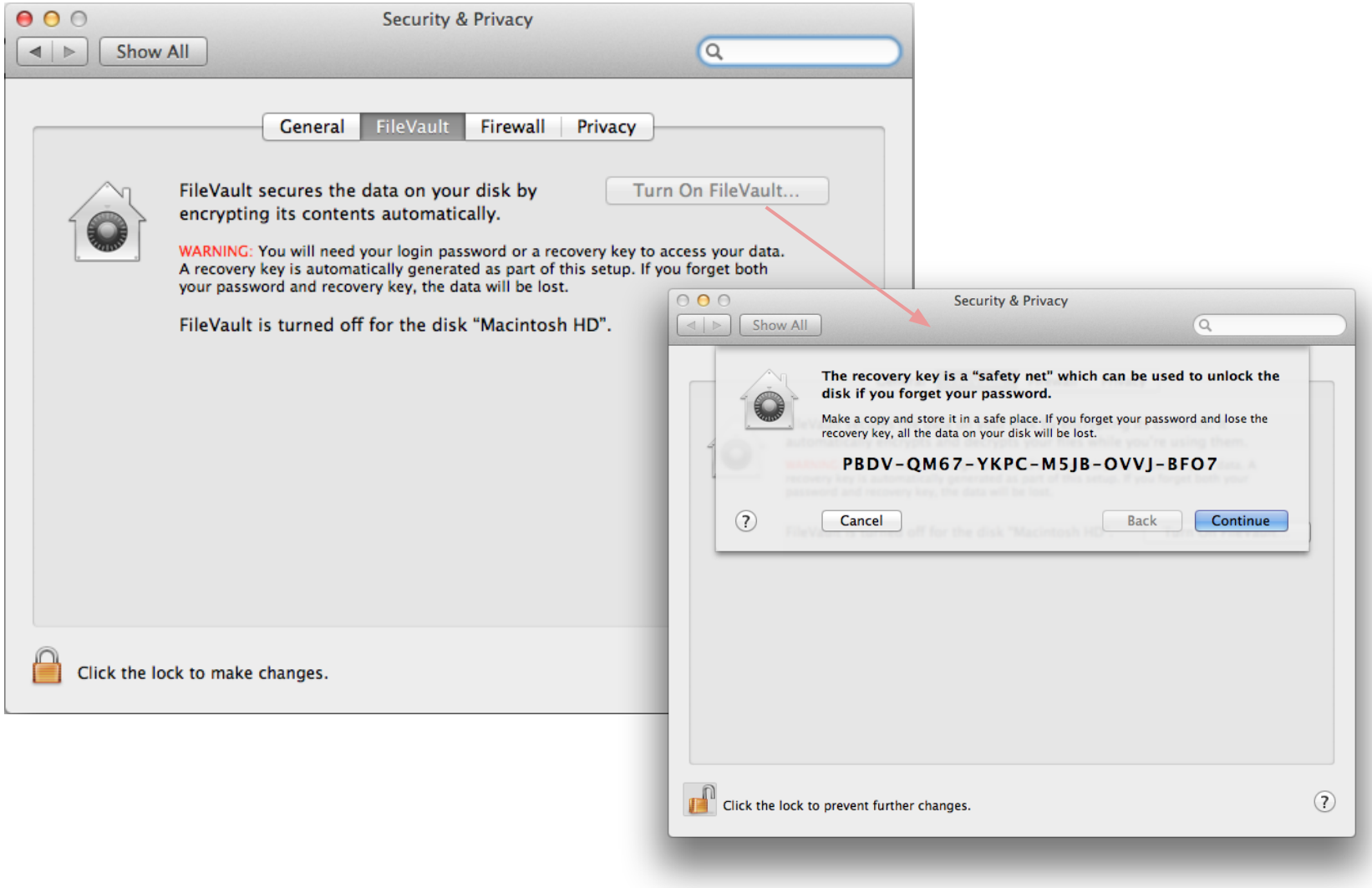
Analysis of FileVault 2: Apple's full disk encryption

Omar Choudary
Felix Grobert
Joachim Metz

FileVault 2


Security & Privacy
Show All
General
FileVault
Firewall
Privacy
FileVault secures the data on your disk by encrypting its contents automatically.
Turn On FileVault...
WARNING: You will need your login password or a recovery key to access your data. A recovery key is automatically generated as part of this setup. If you forget both your password and recovery key, the data will be lost.
FileVault is turned off for the disk "Macintosh HD".
Click the lock to make changes.
Security & Privacy
Show All
The recovery key is a "safety net" which can be used to unlock the disk if you forget your password.
Make a copy and store it in a safe place. If you forget your password and lose the recovery key, all the data on your disk will be lost.
PBDV-QM67-YKPC-M5JB-OVVJ-BFO7
Cancel
Back
Continue
Click the lock to prevent further changes.

Project Overview

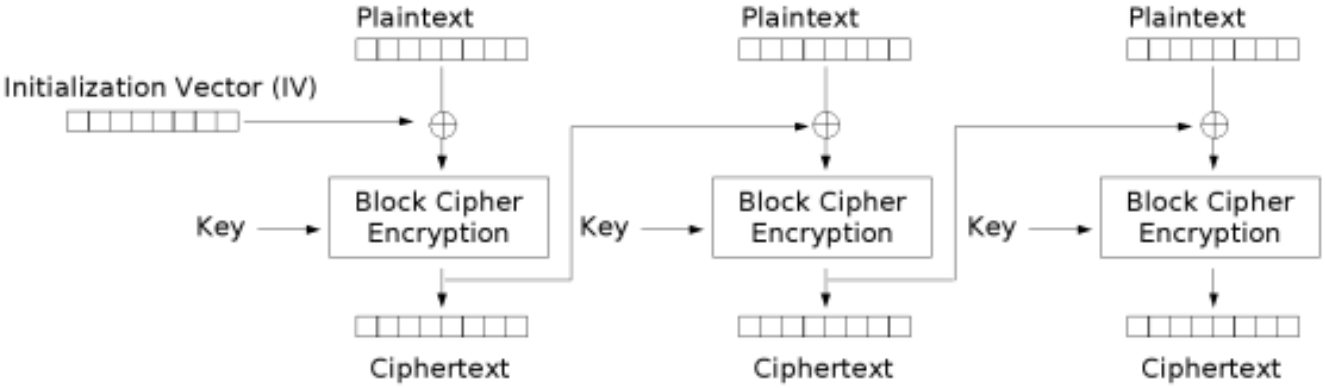
- Goal
  - reverse engineer and analyse Apple's full disk encryption (aka File Vault)
    - introduced in OS X 10.7 (Lion)
  - develop a cross-platform tool to read File Vault encrypted disks
    - also known as CoreStorage volumes
- Why
  - Need to know if secure
  - Use in forensic investigation
  - No trust in the operating system
  - Interoperability
  - Need for access of remote files on encrypted drives

Background - full disk encryption

- Problem:
  - need to encrypt all data
  - user should not memorize or enter a large encryption key
    - e.g. 128 or 256 bits
    - => key is stored in the disk somehow
  - we would like to independently encrypt sectors (normally 512 bytes)

Background - full disk encryption

- AES-CBC alone is not really suitable
  - random IV in metadata and just go on? (quite bad)
  - zero/constant IV? (even worse)
  - sector-based IV? (better, but still not good)



Cipher Block Chaining (CBC) mode encryption

(Wikipedia)

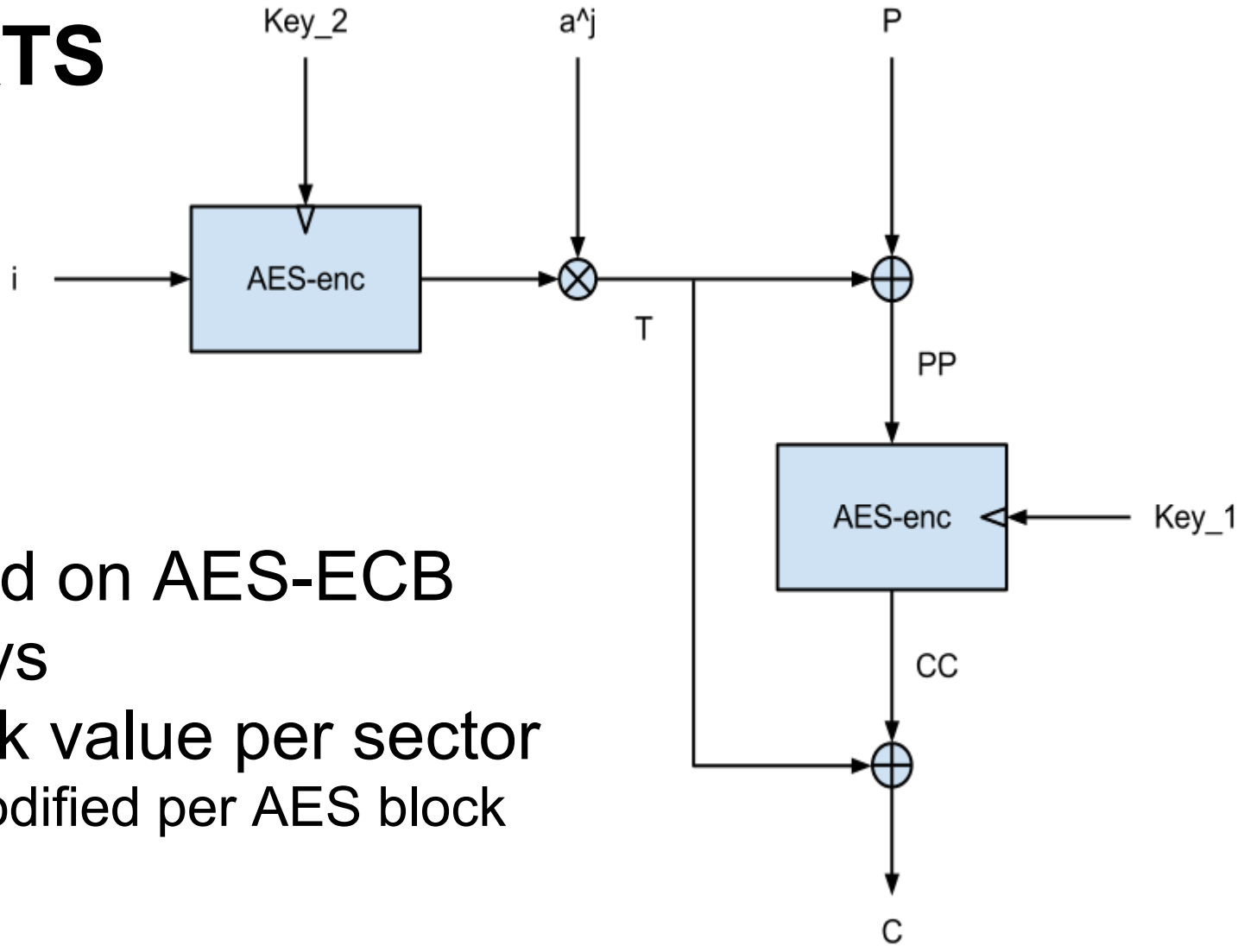
Background - popular systems

- PGP
- BitLocker (used with MS Windows)
  - uses AES-CBC with a sector-based tweak

  AES-CBC + Elephant diffuser. A Disk Encryption Algorithm for Windows Vista. Niels Ferguson.
- LUKS (Linux Unified Key Setup)

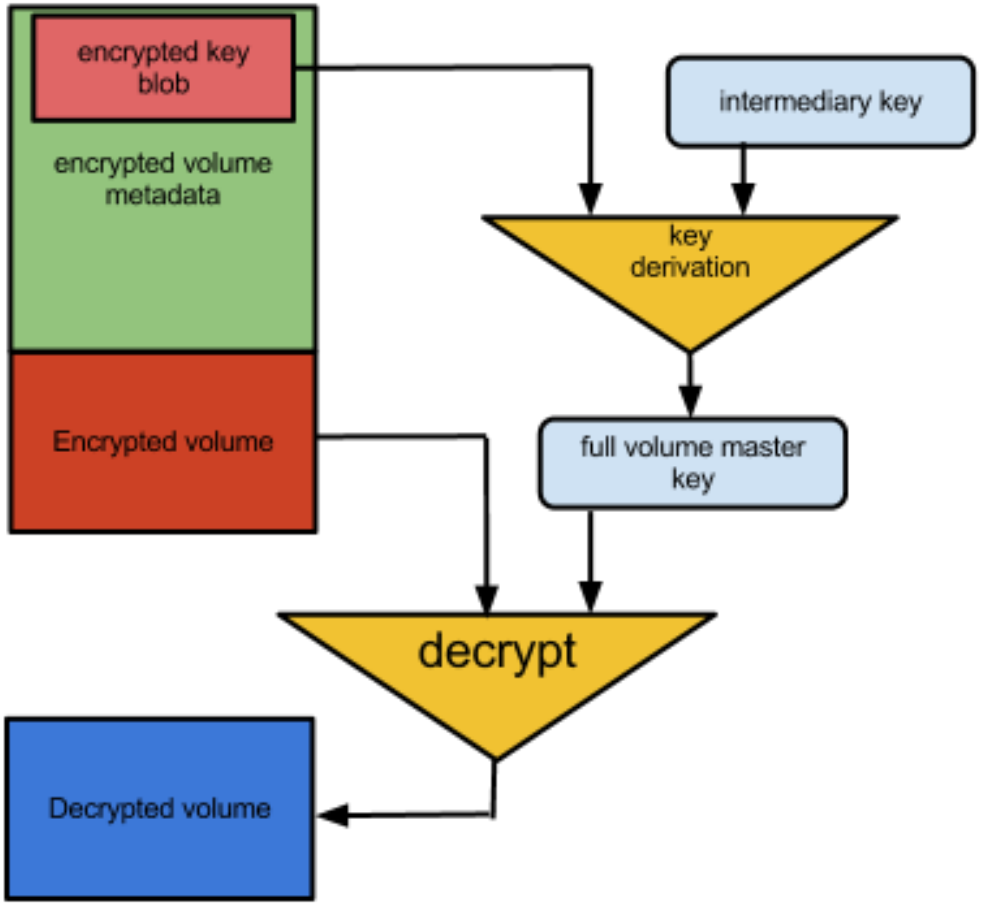
  New methods in hard disk encryption. Clemens Fruhwirth.
- ... others

AES-XTS



- based on AES-ECB
- 2 keys
- tweak value per sector
  - modified per AES block

General full disk encryption architecture



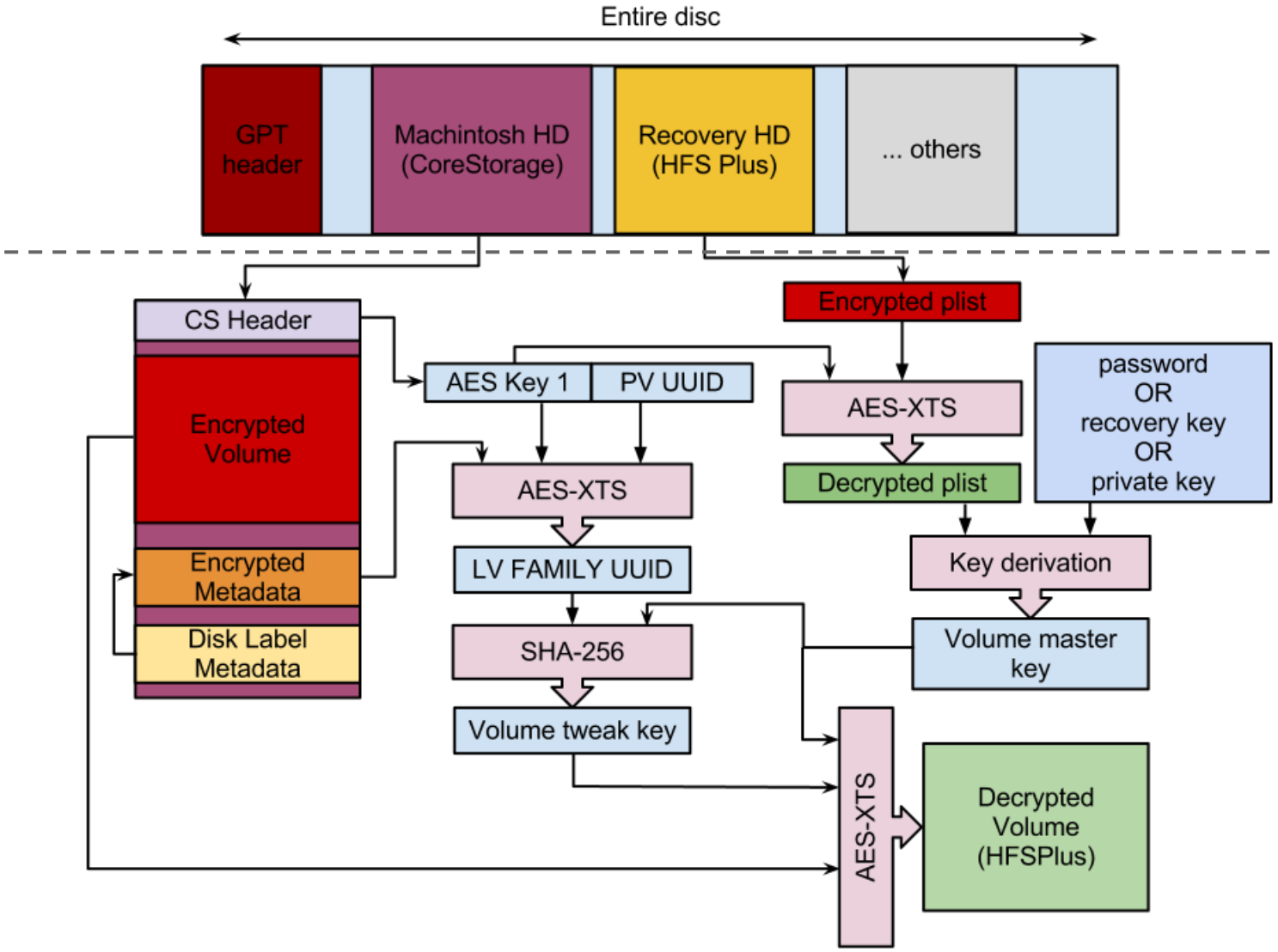
The quest for Apple's File Vault FDE

- what are the key derivation mechanisms?
- what are the encryption mechanisms?
- how is the data encrypted?

Tools at hand

- GDB
- IDA Pro
- 3 MacBook's for kernel debugging
  - 2 of them connected via FireWire => disk access
  - 3rd one connected via Ethernet => remote gdb
- The Sleuth Kit
  - disk forensic tool

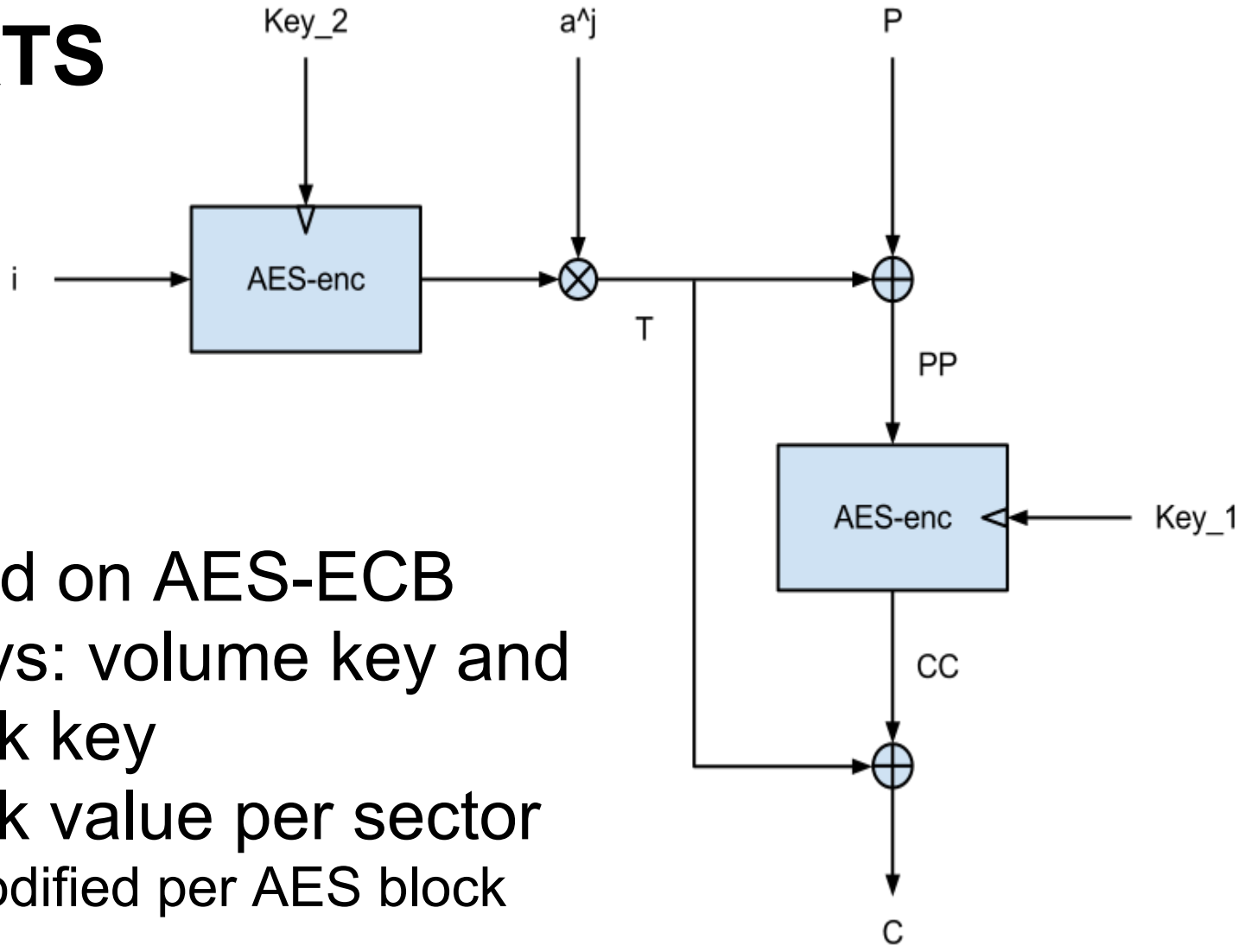
FileVault overview



EncryptedRoot.plist file

- Introduced after FileVault activation
- Contains wrapped volume key
- Available on Recovery HD partition
- Encrypted with key in volume header
- Hints from Apple:
  - AES-XTS as encryption
  - Keys wrapped
- From IDA Pro we get pointers also for
  - AES Wrap
  - PBKDF2

AES-XTS

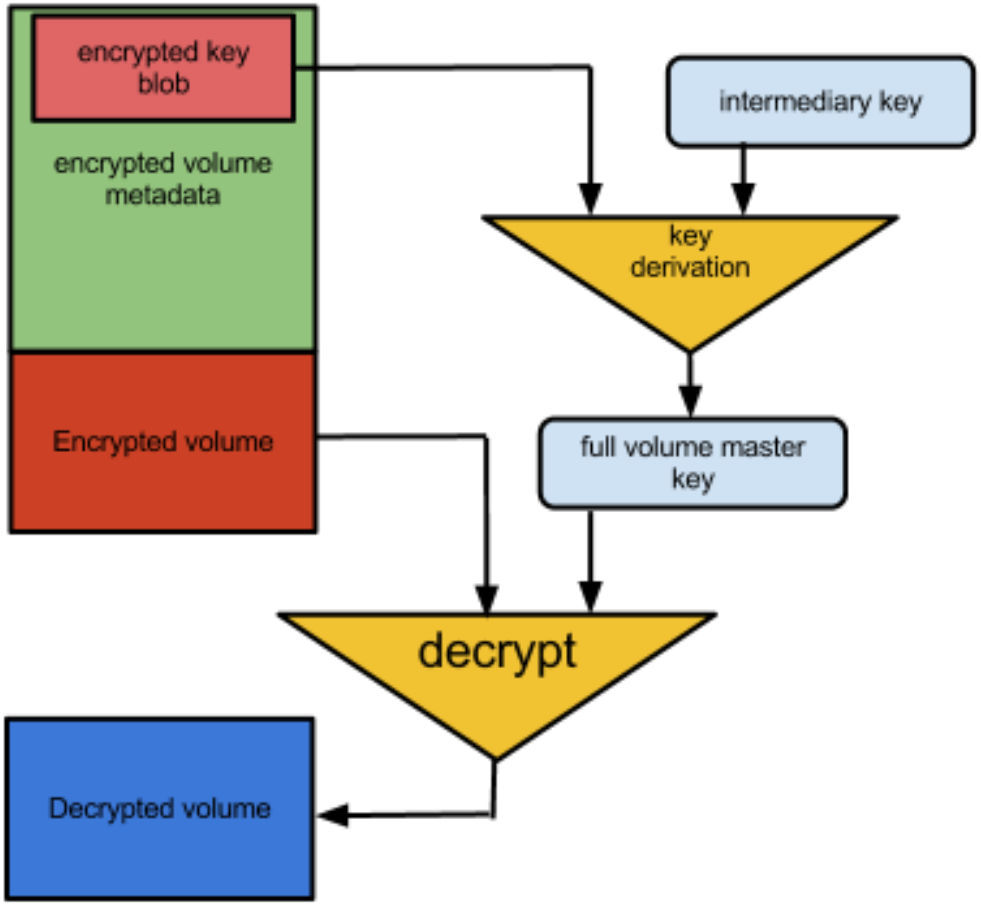


- based on AES-ECB
- 2 keys: volume key and tweak key
- tweak value per sector
  - modified per AES block

Example EncryptedRoot.plist

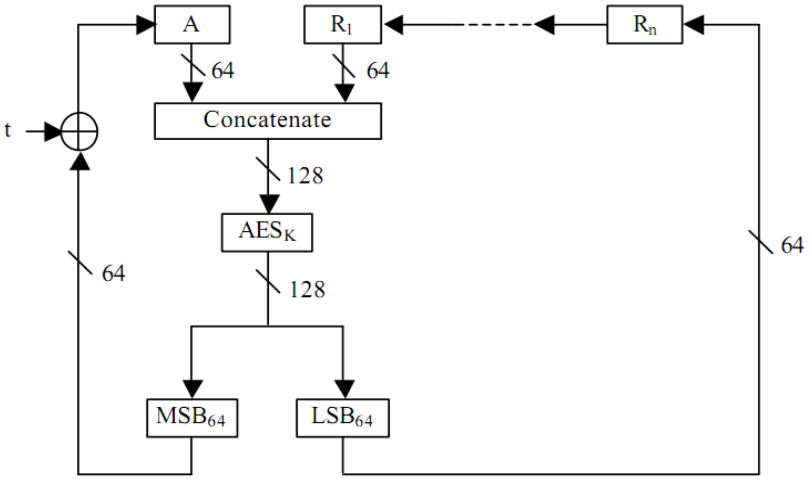

```
<plist version="1.0">
 <dict>
   <key>ConversionInfo</key>
  <dict>
    <key>ConversionStatus</key>
    <string>Complete</string>
    <key>TargetContext</key>
    <integer>1</integer>
  </dict>
  <key>CryptoUsers</key>
 <array>
  <dict>
    <key>EFILoginGraphics</key>
    <data></data>
    <key>KeyEncryptingKeyIdent</key>
    <string>BDE446E1-4925-4586-9143-8A2F84D4F0D0</string>
    <key>PassphraseHint</key>
    <string/>
    <key>PassphraseWrappedKEKStruct</key>
   <data>
     AwAABAAAACzwpAKQRjEW+WLFIaH9CXzEAAAABgAAAB2NG/IV1Be DgnnORdPQK8TWc43H2nWgYMIAAAAAAAAAAAAAAABAAAAAAAAAAA AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAQAAAAMAAAAKAAAA
     ZMeKPMz0HBlcsv1JKKAAAAEAAAABAAAAAwAAAAoAAAAbjWJJ8O/P OUVX+BP2pwkTlo92d77KTxx0oDJsshIToFqpQQX5JAeFhZjzZbku dAfEIC4Vg6iyVa5SvDe0zJd8G6Xxm3rP58uyAHd1B/iwLkZqAbuf ajYWH6s7gB0+m6E=
    </data>
    <key>UserFullName</key>
    <string/>
    <key>UserIcon</key>
    <data></data>
    <key>UserIdent</key>
    <string>187A631A-4586-4CD2-91F0-712C161DC046</string>
    <key>UserNamesData</key>
    <string/>
    <key>UserType</key>
    <integer>268500997</integer>
    <key>WrapVersion</key>
    <integer>1</integer>
   </dict>
  </array>
  <key>LastUpdateTime</key>
  <integer>1320355930</integer>
  <key>WrappedVolumeKeys</key>
 <array>
  ▶<dict>...</dict>
  <dict>
    <key>BlockAlgorithm</key>
    <string>AES-XTS</string>
    <key>KEKWrappedVolumeKeyStruct</key>
   <data>
     AgAAABgAAAD4K867ZJuXw1qMBr5FX/TOg7oy2xydPa4IAAAAAAAA AAAAAABAAAAAAAAAAEAAAAIAAAAAAAAAAAAAAABAAAAAAAAAAA AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAQAAAAMAAAAKAAAA
     ZMeKPMz0HBlcsv1J2VNkXNdnI7JKi5t73TMjPgNyLPmVeR7FyOoG I6EY+oFpe/1cw0zakt8VszS01FVWutSSS5+Msq7f/9u7CzhGue7n Fjecc/AcTkGHedfc1+cAdVpkAQAAAA==
    </data>
    <key>KeyEncryptingKeyIdent</key>
    <string>BDE446E1-4925-4586-9143-8A2F84D4F0D0</string>
    <key>VolumeKeyIdent</key>
    <string>5F4C0D71-A27F-45DF-A2AA-21E22FF0E4AB</string>
    <key>VolumeKeyIndex</key>
    <integer>1</integer>
    <key>WrapVersion</key>
    <integer>1</integer>
   </dict>
  </array>
 </dict>
</plist>
```

General full disk encryption architecture

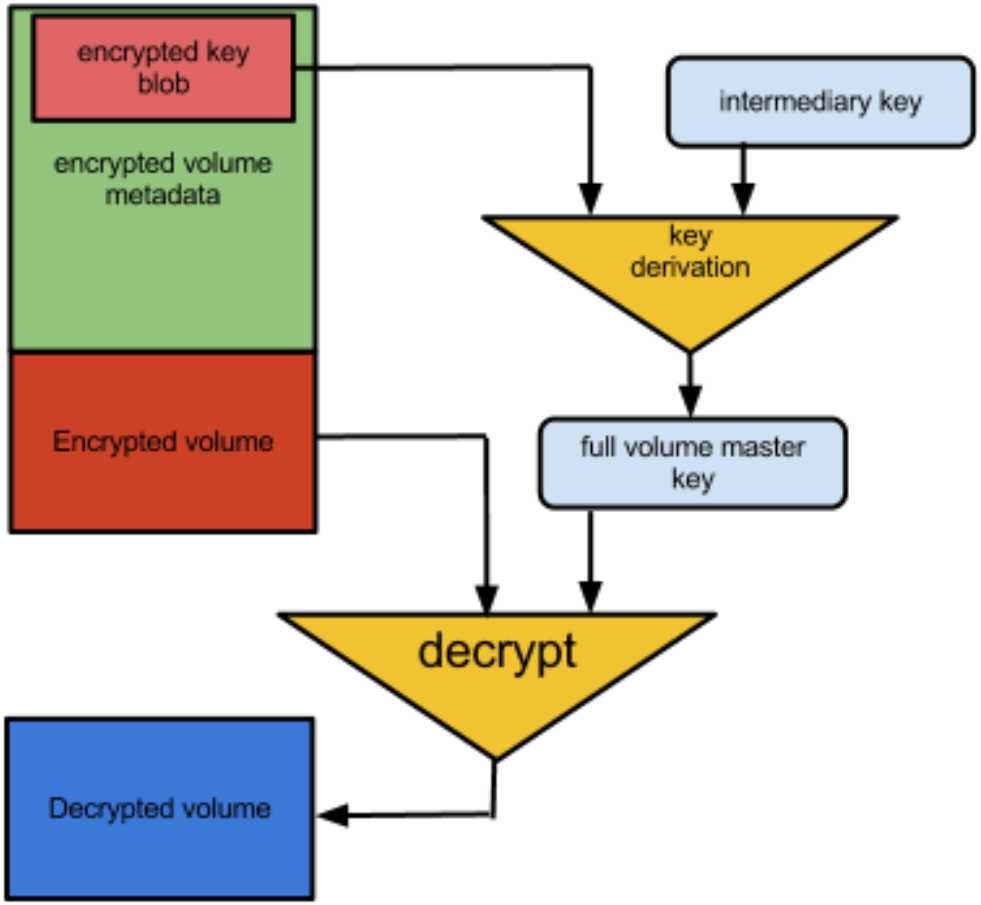


AES Wrap (RFC 3394)

- based on AES, like XTS
- needs a key for unwrapping
- used to protect volume master key
- can verify if unwrapping is successful



General full disk encryption architecture



PBKDF2

- output keys of arbitrary lengths from any text
- slow brute force attacks on passwords
- 3 parameters: iterations, salt, password
- option of PRF (e.g. HMAC-SHA256)
- brute force searching of iterations ... no luck
- salt given in EncryptedRoot.plist
- found iterations via IDA
  - existing code for time dependent value
  - turned out that a static value is used most of the time (41000)

Key derivation overview

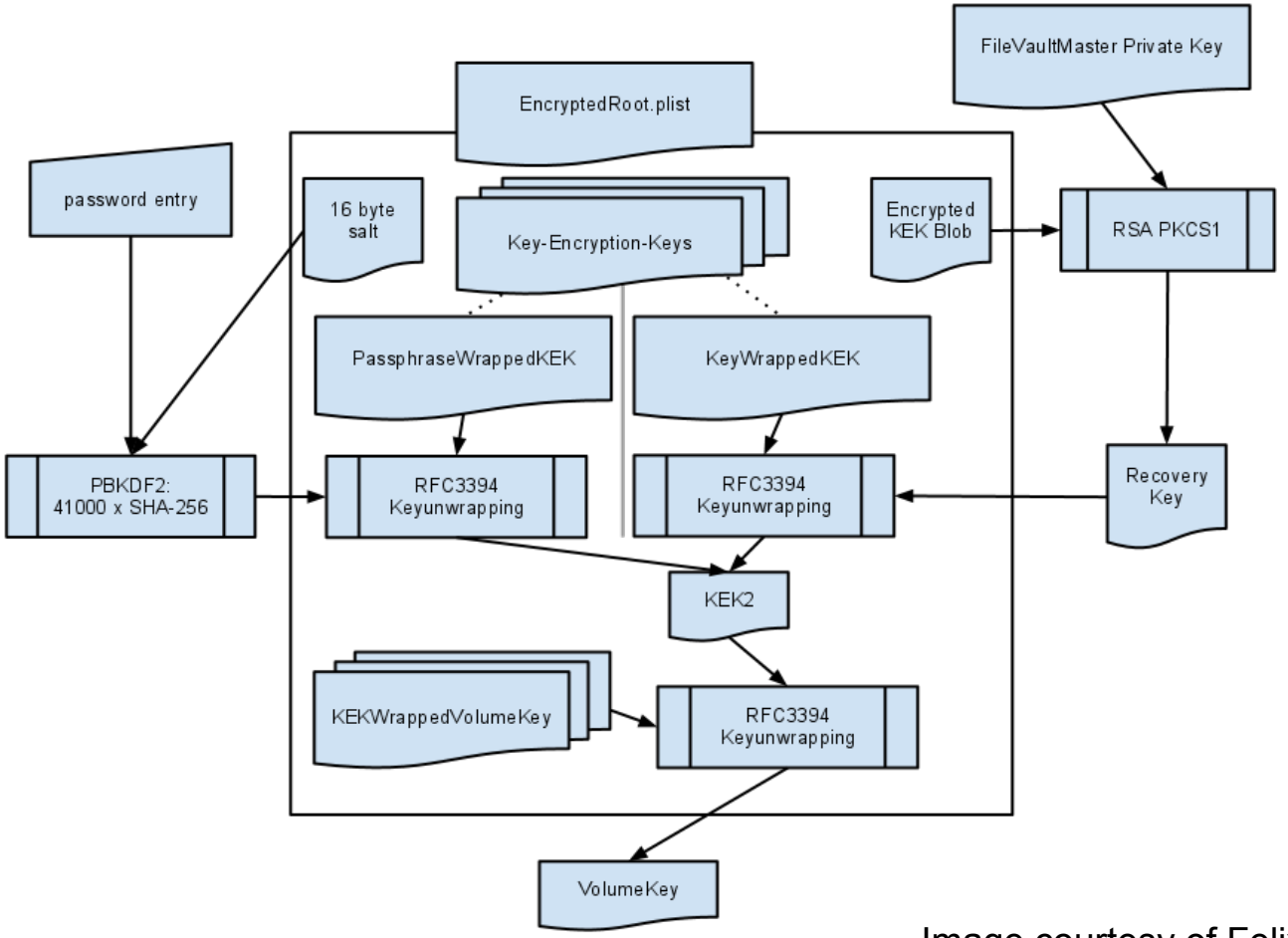
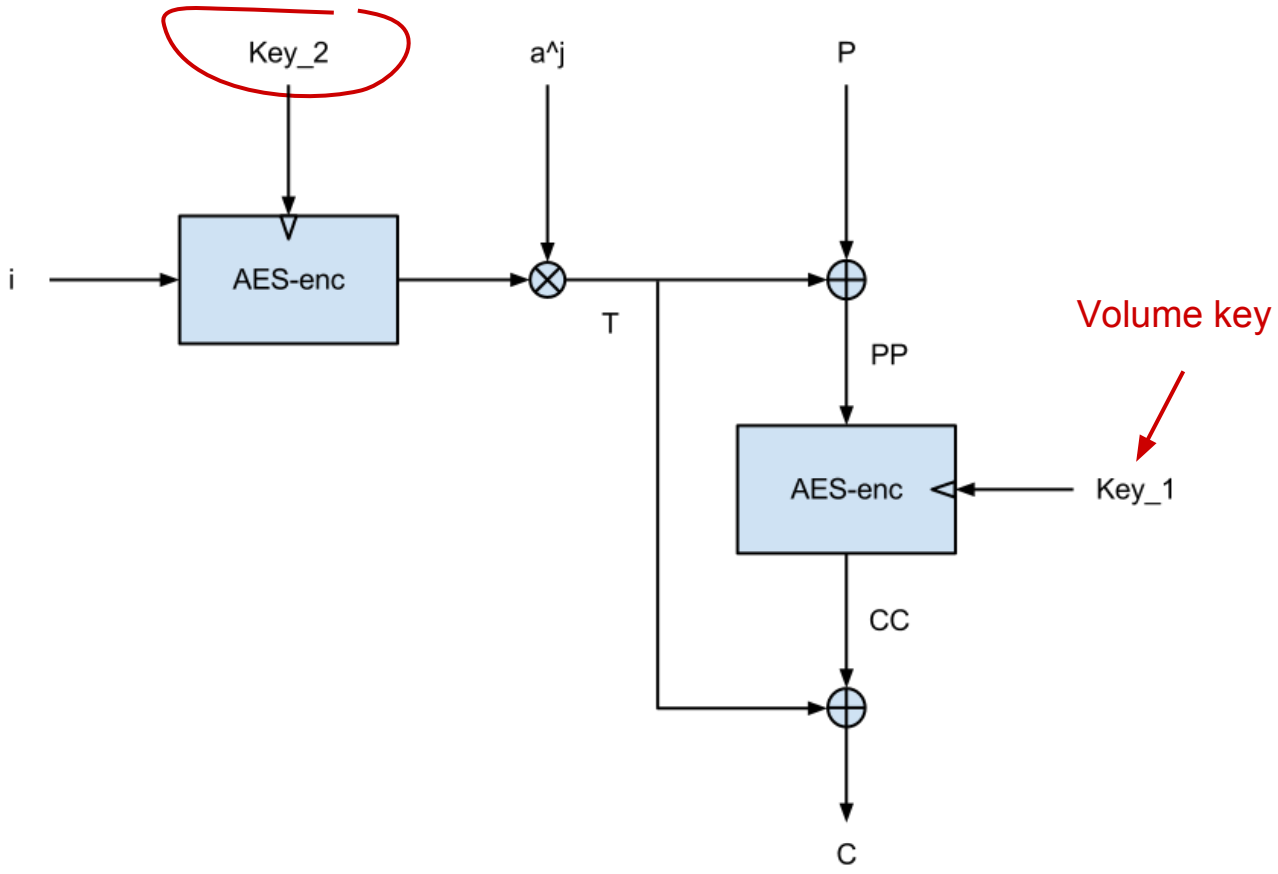


Image courtesy of Felix Grobert

are we done yet? ... tweak key?



Looking for disk encryption mechanism (1)

- Looking at HFS+ metadata
  - existing header at good location
  - apparently unencrypted HFS+ structure files (allocation, journal block)
  - but ... misleading => bug in OS (forgot to erase data)

Looking for disk encryption mechanism (2)

- chasing the encryption via GDB
  - found a tweak key and a tweak value for some data
  - no luck ... still no idea to what that corresponds (may be for virtual memory)

- comparing data with disk data we get
  - tweak value correspondence
  - start of encrypted value
  - block size

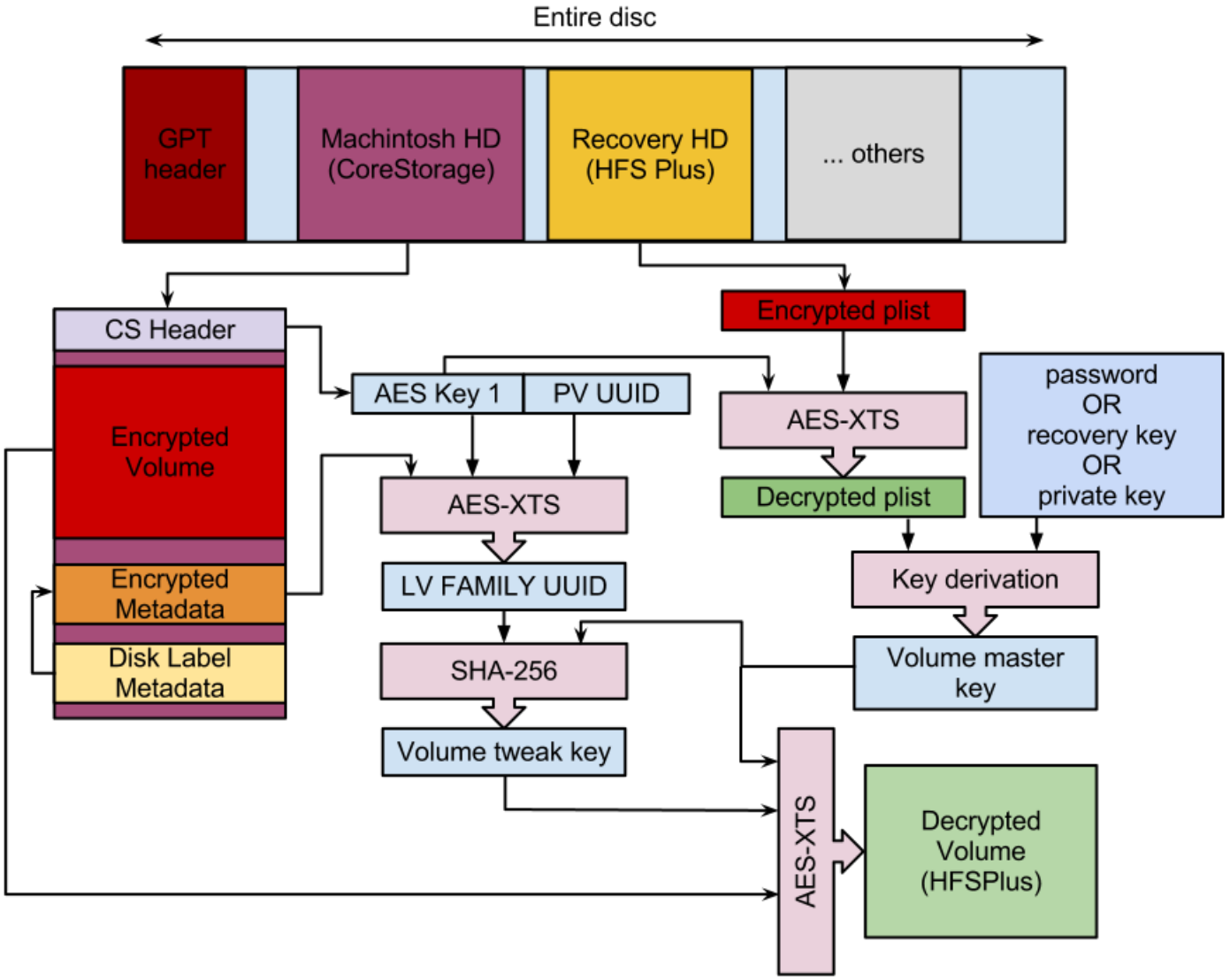
Looking for disk encryption mechanism (3)

- chasing tweak key derivation via IDA Pro
- problems encountered:
  - C++ obfcuscation
    - cdecl (int*) ... *(ebp+478)(ebp+x, ...) ... ???
    - many classes and pointers involved
    - IDA helps but not that much
  - encryption process goes through a Daemon
  - code is quite large

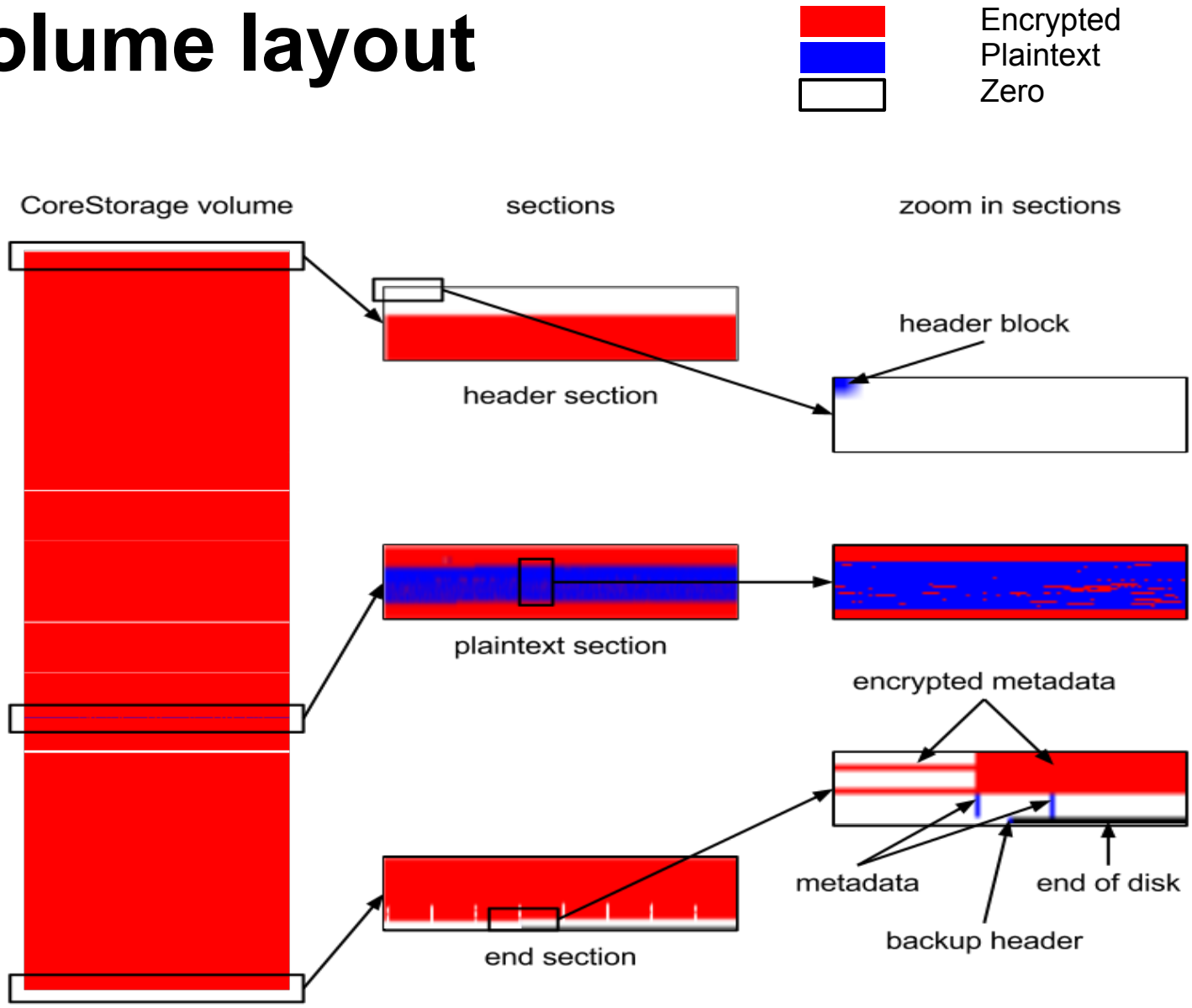
AES-XTS tweak key = trunc_{128}(SHA256(volume_key | lvf_uuid))

(lvf_uuid comes from encrypted (obfuscated) metadata)

FileVault overview



Volume layout



Random number generator

- used for derivation of recovery key
- randomness taken from /dev/random
- about 320 bits of randomness available after first boot of new OS installation
  - mostly from mach_absolute_time()
- seems ok
  - can be improved if needed

Memory extraction attacks

- possible
- keys easily available via gdb
- not much we can do ... open research issue
  - see "Lest We Remember: Cold Boot Attacks on Encryption Keys", USENIX Security 2008.

open source C library

- cross-platform tool to read and mount CoreStorage (FileVault 2 encrypted) volumes
- can mount a CoreStorage volume and read arbitrary files without first decrypting the entire volume
- available at: http://code.google.com/p/libfvde/

```
fvdemount -e EncryptedRoot.plist.wipekey -r 35AJ-AC98-TI1H-N4M3-HDUQ-UQFG /dev/sda2 /mnt/fvdevolume/
mount -o loop,ro /mnt/fvdevolume/fvde1 /mnt/hfs_file_system
```

That's all

Omar Choudary
www.cl.cam.ac.uk/~osc22