PV204: Disk encryption

Laboratory of Security and Applied Cryptograph (LaBAK)

May 5, 2011

1 Introduction

Confidentiality and authenticity of user files is typically protected with PGP or other encryption tools (ZIP with password, etc). The biggest disadvantage of this approach is the need for explicit encryption and decryption of the file. It must be decrypted before it can be read or modified and it must be re-encrypted afterwards.

Disk encryption counters those disadvantages. It allows transparent protection of whole disk volumes (or directories) without any user intervention (other than entering the password). This seminar will focus on those methods implemented in 3 different tools.

EncFS

Easy-to-use pass-thru cryptographic filesystem, that uses 2 directories: encrypted source directory and virtual destination directory, which allows an user (when mounted) to transparently access the files in the source directory. Encrypted directory has the same structure as the decrypted one, but file names and their content is always encrypted. Disadvantage of this approach is that an adversary is then able to see meta data, i.e. how many files are in the source directory, what are their permissions, approximate size, times of last access or modification, etc. We will configure and use own encrypted folder via *Cryptkeeper* GUI in one of the exercises.

Advantages of pass-thru system vs an encrypted block device[8]

- **Size**: an empty EncFS filesystem consists of a couple dozen bytes and can grow to any size without needing to be reformatted. With a loopback encrypted filesystem, you allocate a filesystem ahead of time with the size you want. Depending on the filesystem, there may be ways of resizing it later, but that requires user intervention.
- **Automated Backups**: An EncFS filesystem can be backed-up on a file-by-file basis. A backup program can detect which files have changed, even though it won't be able to decipher the files. This way backups can be made without needing to mount the encrypted filesystem.
- **Layering / Separation of Trust**: EncFS can be layered on top of other filesystems in order to add encryption to unencrypted filesystems. This also allows you to store data on filesystems you trust for storage but not for security. For example, EncFS could be used on top of a CD, or a remote NFS filesystem, Samba share, etc.

TrueCrypt

Pretty popular tool for transparent disk encryption. It allows an user to create a virtual encrypted block device within a partition or a regular file. This virtual disk can be formatted as if it was a physical device and mounted. Encrypted virtual volume starts always with the volume header (128 KB of data), which is encrypted with key derived from entered user password. When decrypted, it contains master keys, which are used for encrypting/decrypting actual volume. Therefore there is no need to reencrypt the volume when changing the password.

In addition, *TrueCrypt* support steganography and allows to create a hidden virtual volume within another volume, which existence can't be proved (if all precautions are followed). We will use this method to create hidden virtual volume, that will be embedded into an innocent-looking video file.

Cryptoloop

Disk encryption kernel module for Linux, that can create encrypted block device, similarly to *TrueCrypt*. *Cryptoloop* encrypts/decrypts the volume directly with key derived from password. When used with weak modes of operations, it is vulnerable to multiple attacks. We will demonstrate watermark attack on AES-CBC as described in [5]. *Cryptoloop* has been marked deprecated and it's successor is `dm-crypt`, which has basically the same functionality.

1.1 Modes of operation

Block-cipher modes of operation play an important role in the security of transparent disk encryption. Unlike in regular file encryption, in disk encryption it is crucial to have a random access into the encrypted volume. Therefore it is unacceptable to use such a chaining-mode, in which change in the first sector will affect all the following ones.

EBC mode will obviously meet this random access requirement, but will provide only weak security, as shown on Fig. 1. Similarly, CTR mode and CBC with predictable IVs (usually derived from sector number) are weak and there are multiple known attacks against them.

Some of the examples of attacks on CBC are:[7]

- **Corrupt**. Corruption of chosen data blocks is difficult to detect. As CBC decryption has little error propagation, modifying a ciphertext block within sector will only corrupt the corresponding plaintext block (8 or 16 bytes) and cause chosen bit changes the one block immediately following it.
- **Move**. It is easy to shift multiple ciphertext blocks anywhere within the hard disk. Only the first plaintext block will be corrupted. This allows "cut & paste" attacks.
- **Revert**. It is possible to revert chosen sectors to their previous values without detection. An attacker can from two ciphertext images detect where the changes lie and choose the sectors to be reverted accordingly.

In CBC mode, the IV must depend at least on both the encryption key and sector number (tweak value). Nowadays it is recommended to use modes of operations designed specifically for disk encryption, like XTS[1].

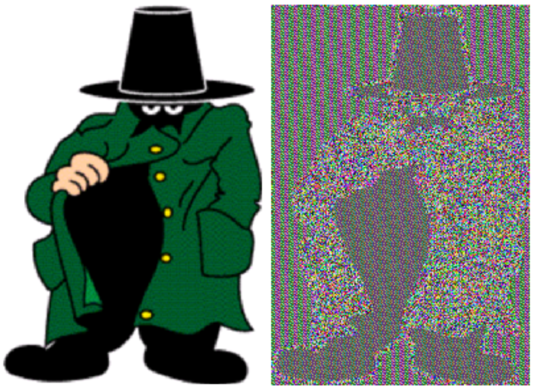

Figure 1: Result of statistical analysis on image encrypted in ECB mode [6]

2 Exercises

During the exercises, we will use prepared Virtual Machine PV204 (Ubuntu 11.04). VM can be run in Oracle VirtualBox.

```
User name: pv204
Password: pv204
```

Note: User `pv204` is a member of group `admin` and can run superuser commands via `sudo`, e.g. `sudo ls /root`.

2.1 Exercise: EncFS Disk Encryption

During this exercise, we will try out how to work with *EncFS* via *Cryptkeeper* GUI.

1. Open *Cryptkeeper* by clicking on icon on main panel or by command-line (`cryptkeeper`).
2. Create new encrypted folder (right-click on icon in tray).
   - Choose name of your new encrypted folder (e.g. `Encrypted`) and open directory, where it will be placed (e.g. `Documents`).
   - Click *Forward*, enter you password and *Cryptkeeper* should automatically create and mount encrypted folder (via *EncFS*).
3. Test your new encrypted folder.
   - Use mounted folder (create some directories and files) and observe how is the underlying hidden folder beeing modified.
   - Note: You can view hidden files in *Nautilus* via View → Show Hidden Files (CTRL+H).
   - For mounting/dismounting use again *Cryptkeeper*, there should be already bookmark for you directory.

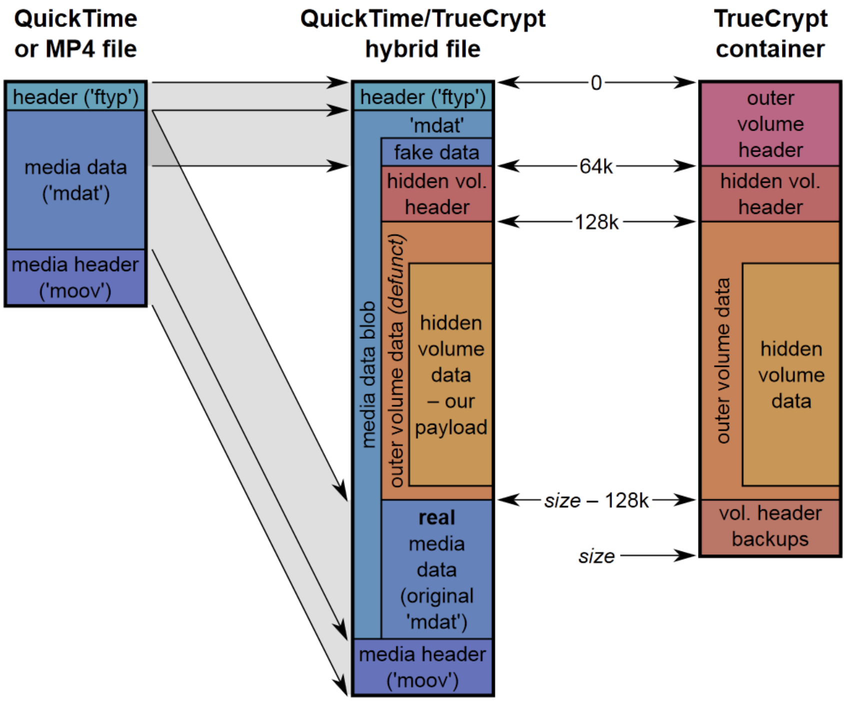

Figure 2: QuickTime/TrueCrypt hybrid file

2.2 Exercise: Steganography with TrueCrypt

During this exercise, we will try out practical steganography with *TrueCrypt*, as described in [4]. Our goal will be to hide *TrueCrypt* volume in *QuickTime / MP4* video file. When opened in media player, the file will appear as a regular media file. However, when opened via *TrueCrypt*, the file will act as a virtual volume and can be mounted. See Fig. 2 and [4] for more details.

1. Navigate to directory `/home/pv204/Documents/Steganography_with_TrueCrypt`.
2. Try to play `sample_iTunes.mov` in some media player (Totem Media Player). Double-click on `sample_iTunes.mov` or use command-line (`totem sample_iTunes.mov`).
3. Open *TrueCrypt* and create new hidden volume. Click on *TrueCrypt* icon on main panel or use command-line (`truecrypt`).
   - Use standard wizard: Create Volume → Encrypted file container → Hidden Volume.
   - Save virtual volume to `test_video.mov` in directory `Steganography_with_TrueCrypt`.
   - Continue with wizard, leave default options, choose outer volume size `5 MB`, inner volume size `4 MB` and make sure outer and inner passwords are different.
   - After the volume has been created, test mounting it with both outer and inner passwords (don't write anything in outer volume, you could damage the hidden inner one).
4. Run script `embed_video.sh` (make sure the volume `test_video.mov` is dismounted).
   - OR run script `tcsteg.py` with 2 arguments, video file `sample_iTunes.mov` and volume file `test_video.mov`.
5. Test your new video file `test_video.mov`, which can be both played and mounted (with inner password).

2.3 Exercise: Cryptoloop Watermark Exploit

```
pv204@pv204-VirtualBox:$ cd /home/pv204/Documents/Cryptoloop_Watermark_Exploit/
# create 5 MB random file disk.img
pv204@pv204-VirtualBox:$ dd if=/dev/urandom bs=1M count=5 of=disk.img
5+0 records in
5+0 records out
5242880 bytes (5.2 MB) copied, 0.872662 s, 6.0 MB/s
# associate virtual device /dev/loop0 with disk.img
# (first password is for sudo, second one is for encrypted device)
pv204@pv204-VirtualBox:$ sudo losetup -e aes-cbc /dev/loop0 disk.img
[sudo] password for pv204:
Password:
# format this virtual device
pv204@pv204-VirtualBox:$ sudo mkfs.ext3 /dev/loop0
mke2fs 1.41.14 (22-Dec-2010)
Filesystem label=
OS type: Linux
Block size=1024 (log=0)
...
# detach loop device /dev/loop0
pv204@pv204-VirtualBox:$ sudo losetup -d /dev/loop0
# mount disk.img to /media/cryptoloop
pv204@pv204-VirtualBox:$ sudo mount -o loop,encryption=aes-cbc disk.img /media/cryptoloop/
Password:
# check whether it was mounted successfully
pv204@pv204-VirtualBox:$ mount
...
/dev/loop0 on /media/cryptoloop type ext3 (rw,encryption=aes-cbc)
# set permission for user pv204
pv204@pv204-VirtualBox:$ sudo chown pv204:pv204 /media/cryptoloop/
```

The encrypted file system from `disk.img` should be mounted and accessible by now. It's time to try out the Watermark Exploit [5]. Cryptolooop with AES-CBC encryption uses predictable IVs and as mentioned before, it is not considered secure. With knowledge about structure of file system (block size 1024), the attacker is able to detect presence of special watermarked files. Watermarked files contain bit patterns that can be detected without knowing the password and decrypting the drive.

```
# we want to create watermarked file, that would encode ASCII text "LaBAK" (76 97 66 65 75)
# create watermarked file labak_watermarks and place it in mounted encrypted volume
pv204@pv204-VirtualBox:$ ./create-watermark-encodings 10:76 11:97 12:66 13:65 14:75 > \
> /media/cryptoloop/labak_watermarks
# unmount the volume
pv204@pv204-VirtualBox:$ sudo umount /media/cryptoloop/
# run the exploit - detect the presence of watermarks in ENCRYPTED file
pv204@pv204-VirtualBox:$ cat disk.img | ./detect-watermark-encodings
5242880 bytes scanned
watermark encoding 10, count 76
watermark encoding 11, count 97
watermark encoding 12, count 66
watermark encoding 13, count 65
watermark encoding 14, count 75
```

3 Assignment

There should be a file `assignment.zip` in study materials, which contains:

- `virtual_volume.tc` - TrueCrypt virtual volume
- `virtual_volume_header_master_keydata` - Dumped master keys

File `virtual_volume.tc` contains virtual volume created by *TrueCrypt 7.0a*, which is protected by 14-characters long password. When mounted successfully, volume contains single text file `password.txt`, which contains plain-text password to the volume. Your task is to **decrypt the volume and get the original password**. Of course, the task would be pretty hard without the master keys to the volume. File `virtual_volume_header_master_keydata` contains concatenated primary and secondary master key[1].

You are expected to implement XTS-AES decryption, according to IEEE 1619-2007[1]. Correct solution should contain decrypted volume, discovered password and source code of the tool, which you created and used for this decryption. For more details on *TrueCrypt* see [2] and [3].

Virtual volume details:

- Filesystem FAT (FAT12)
- Encrypted with AES in XTS mode, AES block length 128b
- Primary key length 256b, secondary key length 256b, data unit size 512B

3.1 Notes

You can find XTS-AES encryption pseudocode in [1], Annex C. The same pseudocode can be used for decryption with only small modifications.

The volume uses FAT filesystem, so when you decrypt the volume, the content of text file `password.txt` should be easily readable, see Fig. 3.

You can use TrueCrypt build-in test tool (Menu → Tools → Test Vectors) for testing your code, see Fig. 4.

[1] Decrypted part of volume header, bytes 256 - 512.

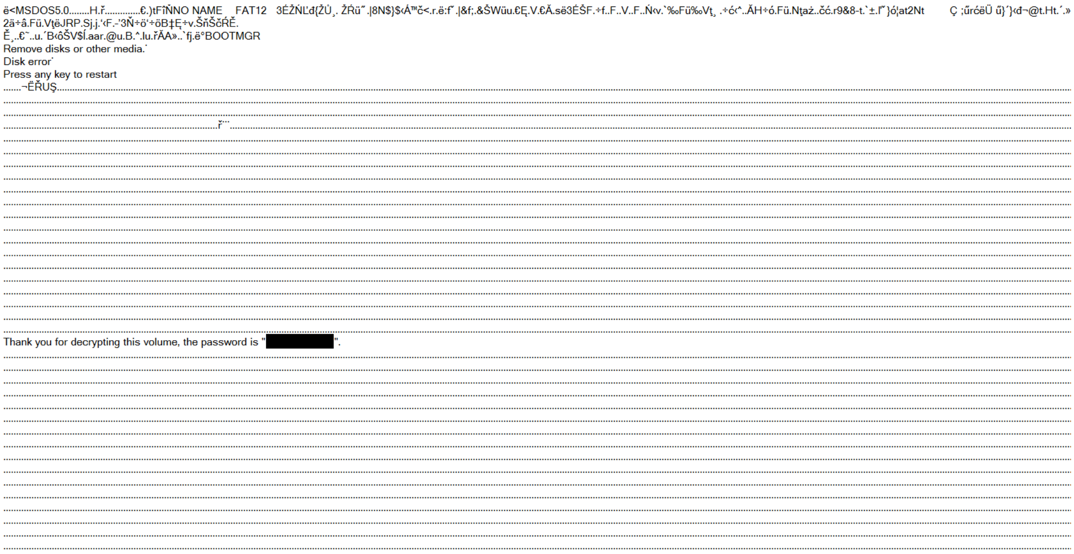

Figure 3: Volume decrypted using provided master keys

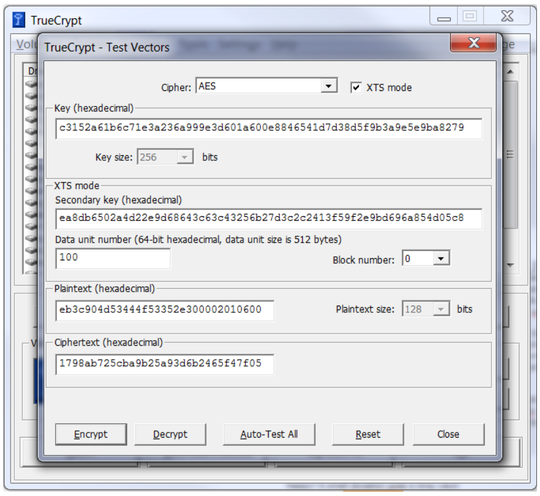

Figure 4: Decryption of first data block of volume

References

[1] *The XTS-AES Tweakable Block Cipher*, http://axelkenzo.ru/downloads/1619-2007-NIST-Submission.pdf

[2] *TrueCrypt Modes of Operation*, http://www.truecrypt.org/docs/?s=modes-of-operation

[3] *TrueCrypt Volume Format Specification*, http://www.truecrypt.org/docs/?s=volume-format-specification

[4] *Real Steganography with TrueCrypt*, http://keyj.s2000.at/?p=458

[5] *Linux Cryptoloop Watermark Exploit*, http://www.securiteam.com/exploits/5UP0P1PFPM.html

[6] Markku-Juhani O. Saarinen, *HERRING - Crafting a Cipher for Sector-Level Encryption*, http://mareichelt.de/pub/notmine/herring061103.pdf

[7] Markku-Juhani O. Saarinen, *Encrypted Watermarks and Linux Laptop Security*, http://mareichelt.de/pub/notmine/wisa2004.pdf

[8] *EncFS Introduction*, http://www.arg0.net/encfsintro