



Secure Documents

Mathias Jeschke

Sven Wittig

Seminar: Security Engineering

15.02.2005



Motivation

- Many people:
 - think operating systems can handle all access from users to their data
 - don't care about the sense of their data
 - trust on security and integrity of their data



Motivation

- Assume somebody steal your laptop with sensitive company data stored on it (e.g. project specs, private keys, ...)
- Access security is not given if someone obtain physical access to your device
 - KNOPPIX CD-ROM, ...
 - Only solution: hard cryptography



Overview

- Motivation
- File based security
(OpenSSL, GPG, PDF)
- File system based security
(EFS, cryptoloop, EncFS/FUSE,
dm_mod)
- Some thoughts on attacks



File based Security

- UNIX way: container for any filetypes
 - Symmetric encryption: AES

```
$ openssl aes-256-cbc -e  
-in document.txt \  
-out document.enc
```

```
$ openssl aes-256-cbc -d \  
-in document.enc \  
-out document.dec
```



File based Security

- UNIX way: container for any filetypes
 - Asymmetric encryption: RSA

```
$ openssl genrsa -out bob.priv 1024
$ openssl rsa -in bob.priv -pubout -out bob.pub

$ openssl rsautl -encrypt \
-in document.txt \
-out document.enc \
-inkey bob.pub -pubin

$ openssl rsautl -decrypt \
-in document.enc \
-out document.dec \
-inkey bob.priv
```



File based Security

- UNIX way: container for any filetypes
 - GnuPG (GPG)
 - Ciphers
 - RSA (sign only)
 - ElGamal/DSA (crypt & sign)
 - Symmetric cipher available, but unusual:
AES, CAST5, BLOWFISH, TWOFISH, 3DES
 - Often used to encrypt asymmetrically
email contents



File based Security

- UNIX way: container for any filetypes
 - GnuPG (GPG)

```
$ gpg --search-keys Mathias Jeschke
```

```
...
```

```
(1) Mathias Jeschke <jeschke@info...>
```

```
1024 bit DSA ...
```

```
Enter number(s), N)ext, or Q)uit > 1
```

```
....
```

```
$ gpg --list-keys
```

```
/home/mj/.gnupg/pubring.gpg
```

```
-----
```

```
pub 1024D/EAE54729 2002-06-12 Mathias Jeschke ...
```

```
sub 1024g/C1FD3060 2002-06-12
```

```
$ gpg --gen-key
```



File based Security

- UNIX way: container for any filetypes
 - GnuPG (GPG)

```
$ gpg -er "Mathias Jeschke" passwd
```

```
$ ls passwd*
```

```
passwd passwd.gpg
```

```
$ gpg -d passwd.gpg
```

```
...
```

```
root:x:0:0:root:/root:/bin/bash
```

```
...
```



File based Security

- Windows way: specific filetypes
 - Adobe Acrobat
 - Symmetric and Asymmetric encryption
 - Microsoft Word
 - Only password based (symmetric) encryption
 - Password recovery tools exist!



File based Security

- Windows way: specific filetypes
 - Adobe Acrobat
 - Encryption
 - Symmetric
 - Asymmetric
 - Signatures
 - Visible signatures
 - Hidden signatures
 - Demonstration...



File system based Security

- Windows
 - **E**ncrypted **F**ile **S**ystem (EFS)
- Linux
 - EncFS/FUSE (userspace file system)
 - Cryptoloop/CryptoAPI (kernel modules)
 - dm_mod (next generation of kernel crypto layer since Linux 2.6.4)
 - TCFS (NFS based encryption)



Windows FS based Security



File system based Security

- Windows: **E**ncrypted **F**ile **S**ystem (EFS)
 - What is EFS?
 - Why EFS?
 - Encryption
 - Use
 - Security



Encrypted File System

- What is EFS?
 - makes encryption of files and/or directories possible
 - requires NTFS
 - provided by Windows 2000, XP and 2003 Server
 - is part of the OS



Encrypted File System

- Why EFS?
 - file access authorization
 - security problem: other user have physical access too
- => file encryption



Encrypted File System

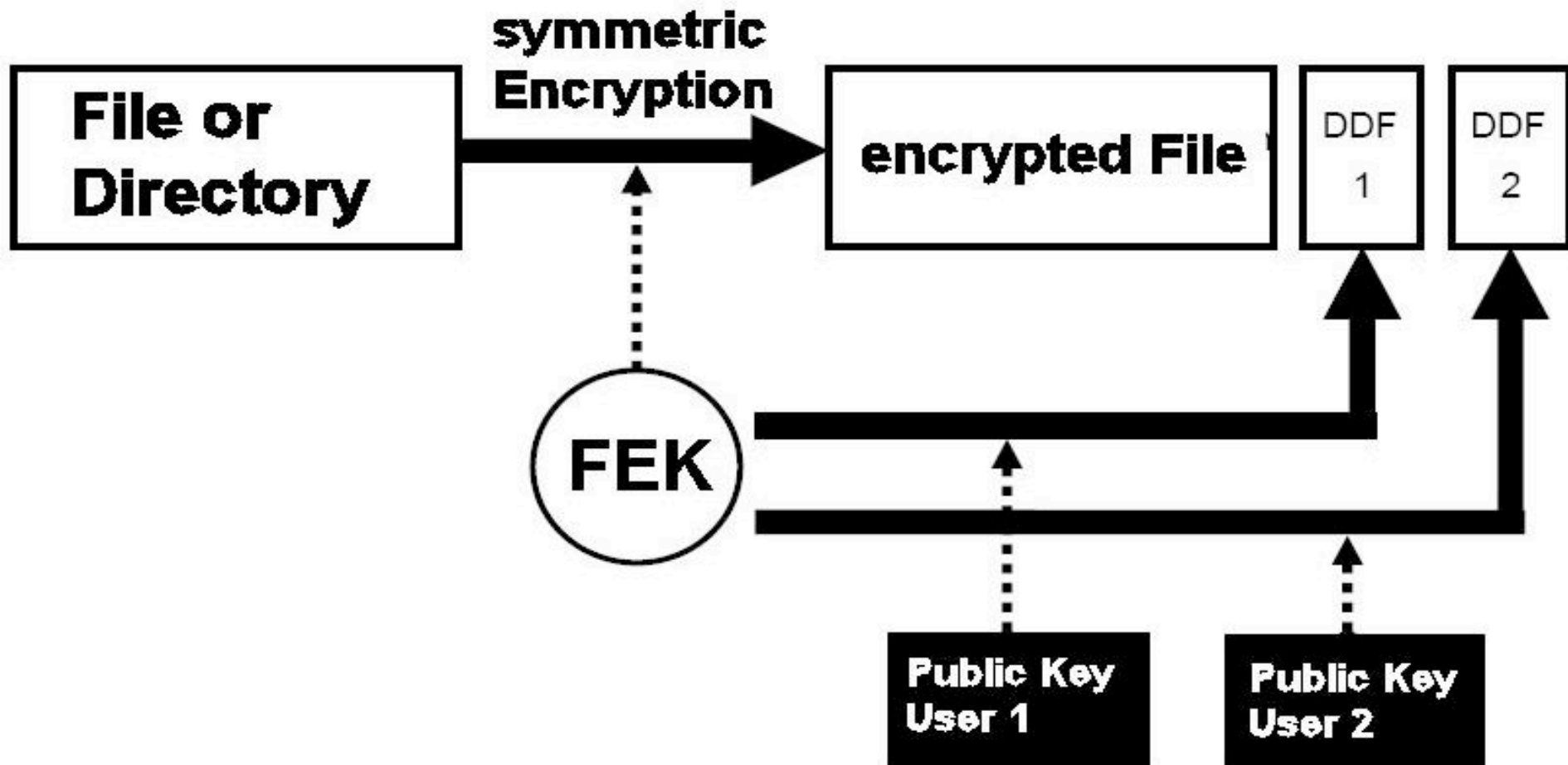
- Why EFS?
 - EFS works transparently
 - Data Recovery Agent
 - Encryption with DESX(128 bit) or 3DES(168bit) and RSA (1024bit)



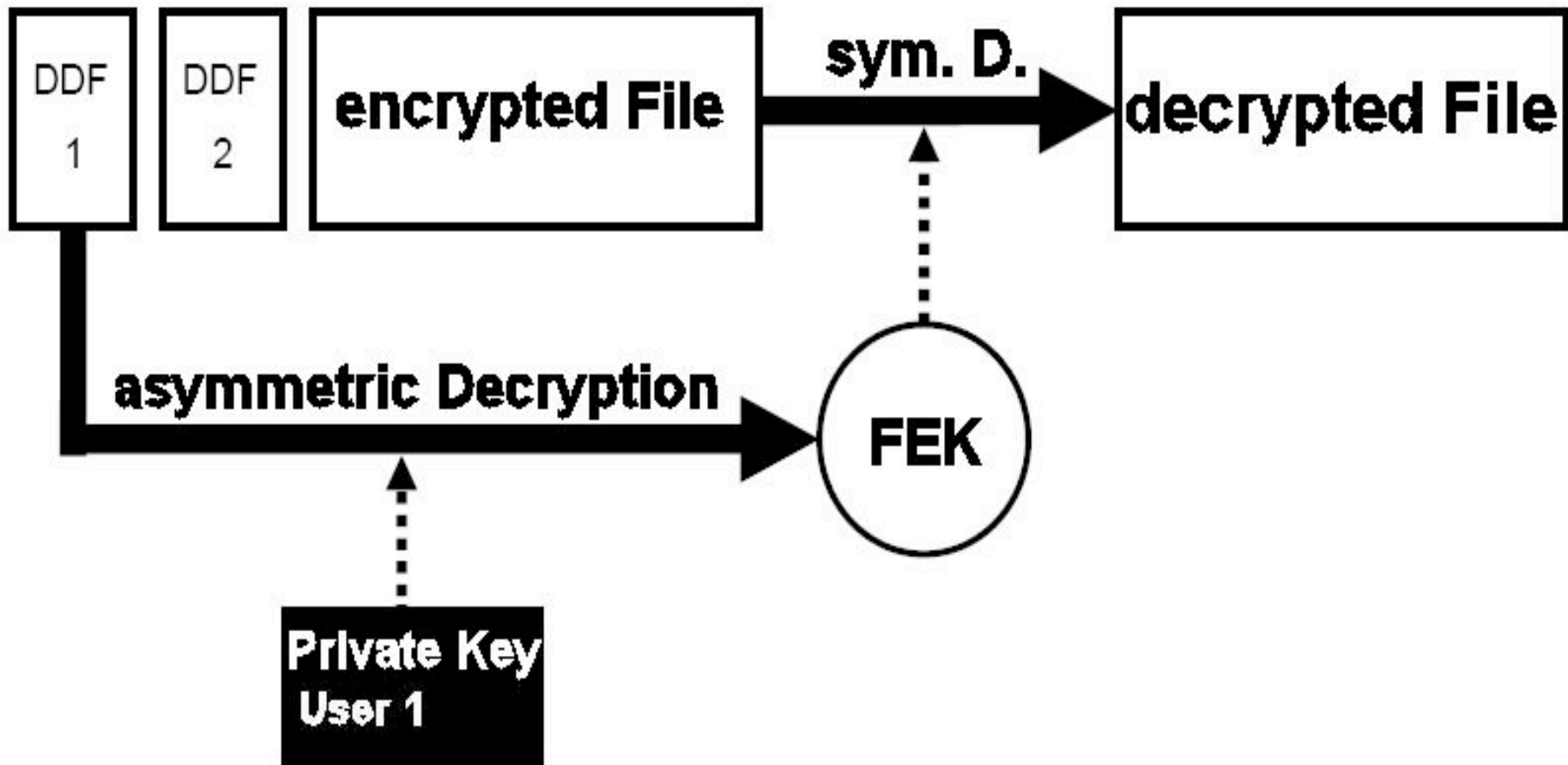
Encrypted File System

- Encryption:
 - EFS use a combination of symmetric and asymmetric encryption
 - symmetric encryption via DESX(128 bit) or 3DES(168bit)
 - asymmetric encryption via RSA (1024bit)

Encrypted File System



Encrypted File System





Encrypted File System

- Use:
 - possible encryptions:
 - files
 - directorys
 - impossible encryptions:
 - systemfiles
 - packed files



Encrypted File System

- Use:
 - lost of encryption after saving on a non NTFS-partition
 - only non encrypted transmission over network possible
 - tmp-files are not encrypted
 - ...



Encrypted File System

- Security:
 - 1.Recovery Agent:
 - Special user (admin) who has access to all encrypted files
 - this admin gets entry in DDF automaticly for all encrypted files
 - Access to this account means access to all encrypted files



Encrypted File System

- Security:
 - 2.Accounts:
 - Access to an account means access to the encrypted files => encryption only as good as the user master key.
 - Solution: key has to be saved on a disk or so and removed from the system.
 - => access to the account allows not access to the encrypted files.



Encrypted File System

- Security:
 - Brute-force on long DESX and 3DES keys are nearly not possible
 - always delete temp-files
 - physical access necessary?



Linux FS based Security



FS Security: Cryptoloop

- Kernel module for on-the-fly encryption/decryption (built-in since kernel version 2.6 and late 2.4.x)
- Many ciphers are supported
 - AES
 - Blowfish, Twofish
 - CAST5, CAST6
 - (3)DES ;-)
 - ...
- Can be used with loop devices (normal files as block devices)



FS Security: Cryptoloop

- Load the required kernel modules (e.g. for AES)
 - `$ modprobe cryptoloop`
 - `$ modprobe aes`
- You need an (empty) file or block device
 - Create it with: (for a 100MB example)
 - `$ dd if=/dev/urandom of=crypto-container \`
`bs=1M count=100`
 - Attach your container to a loop device:
 - `$ losetup -e aes-256 -T /dev/loop0 crypto-container`
 - Create a file system in your loop container:
 - for reiserfs: `$ mkreiserfs /dev/loop0`
 - or for ext3: `$ mkfs.ext3 /dev/loop0`



FS Security: Cryptoloop

- Mounting

- Explicit binding to loop device

```
$ losetup -e aes-256 /dev/loop0 crypto-container
```

```
$ mount /dev/loop0 /mnt
```

```
...
```

```
$ umount /mnt
```

```
$ losetup -d /dev/loop0
```

- or implicit binding to loop device

```
$ mount -o encryption=aes-256 crypto-container /mnt
```

```
$ umount /mnt
```



FS Security: EncFS/FUSE

- FUSE provides file systems in userspace
- EncFS encrypts directory trees based on FUSE library
- Kernel module: fuse (redirects syscalls to userspace, e.g. `open()`, `write()`, ...)



FS Security: EncFS/FUSE

- Install dependencies:
 - FUSE
 - RLog
 - OpenSSL
 - RPMs for SuSE 9.2 available on project homepage (fuse kernel module broken!)



FS Security: EncFS/FUSE

- Setup the EncFS environment

```
$ insmod fuse.o
```

```
$ encfs ~/.crypto ~/.secret
```

```
Creating new encrypted volume.
```

```
Please choose from one of the following options:
```

```
enter "x" for expert configuration mode,
```

```
enter "p" for pre-configured paranoia mode,
```

```
anything else, or an empty line will select  
standard mode.
```

```
?> p
```



FS Security: EncFS/FUSE

Paranoia configuration selected.

Configuration finished. The filesystem to be created has the following properties:

Filesystem cipher: "ssl/aes" , version 2:1:1

Filename encoding: "nameio/block", version 3:0:1

Key Size: 256 bits

Block Size: 512 bytes, including 8 byte Message Authentication Code header

Each file contains 8 byte header with unique IV data.

Filenames encoded using IV chaining mode.

File data IV is chained to filename IV.



FS Security: EncFS/FUSE

Now you will need to enter a password for your filesystem. You will need to remember this password, as there is absolutely no recovery mechanism. However, the password can be changed later using `encfsctl`.

New Password:

Verify Password:

```
$ cp /etc/passwd ~/.secret
```

```
$ ls ~/.secret
```

```
passwd
```

```
$ ls ~/.crypto
```

```
yMDxU2NMUsSUKn89NsKCWOMS
```

```
$ fusermount -u ~/.secret
```



File based Security: Summary (Linux)

- **OpenSSL/GnuPG**
 - **Pros**
 - OS independent (source code available)
 - Can run as non-privileged user
 - Stable
 - Performance similarly to kernel crypto-API
 - **Cons**
 - Every file of a set must be encrypted separately
possible solution (tar the set before encrypting/signing)
 - An decrypted version of the encrypted file always exists on storage



FS Security: Summary (Linux)

- Cryptoloop

- Pros

- Good performance
 - 30 MB/s with AES-256 @ P4-3GHz
- Easy handling (only mount/umount container)
- No decrypted files on any storage device

- Cons

- Impossible to change the size of container
- Root must run the setup (losetup && mount)
- Kernel module (could crash your system)
- May be not supported in next kernel releases (marked as deprecated)



FS Security: Summary (Linux)

- EncFS/FUSE

- Pros

- No allocation of needed disk space at setup necessary
- Easy backups of encrypted files possible (without decrypt them)
- Flexible (can be used as non-privileged user)
- Stable (based on OpenSSL)

- Cons

- Directory layout could tell more information, than the author want (e.g. number of files, owner, permissions)



Some thoughts on attacks

- Do not trust your system administrator!
 - umount userspace crypto container if not used
 - do not save unencrypted SSH keys
 - do not have a ssh-agent running, if you are logged out
- Do not forget to erase the unencrypted original files safely:

```
$ shred -z -u orig_file
```



References

- OpenSSL man page
- GPG man page
- TLDP Cryptoloop HOWTO
- <http://arg0.net/users/vgough/encfs.html>
- Merz, Drümmer: "Die PostScript- & PDF-Bibel"
- Linux-Magazin 01/2005 (Userspace FS)
- Hackin9 1/2005, ISSN-1731-7045
- Wikipedia (lots about cryptography)