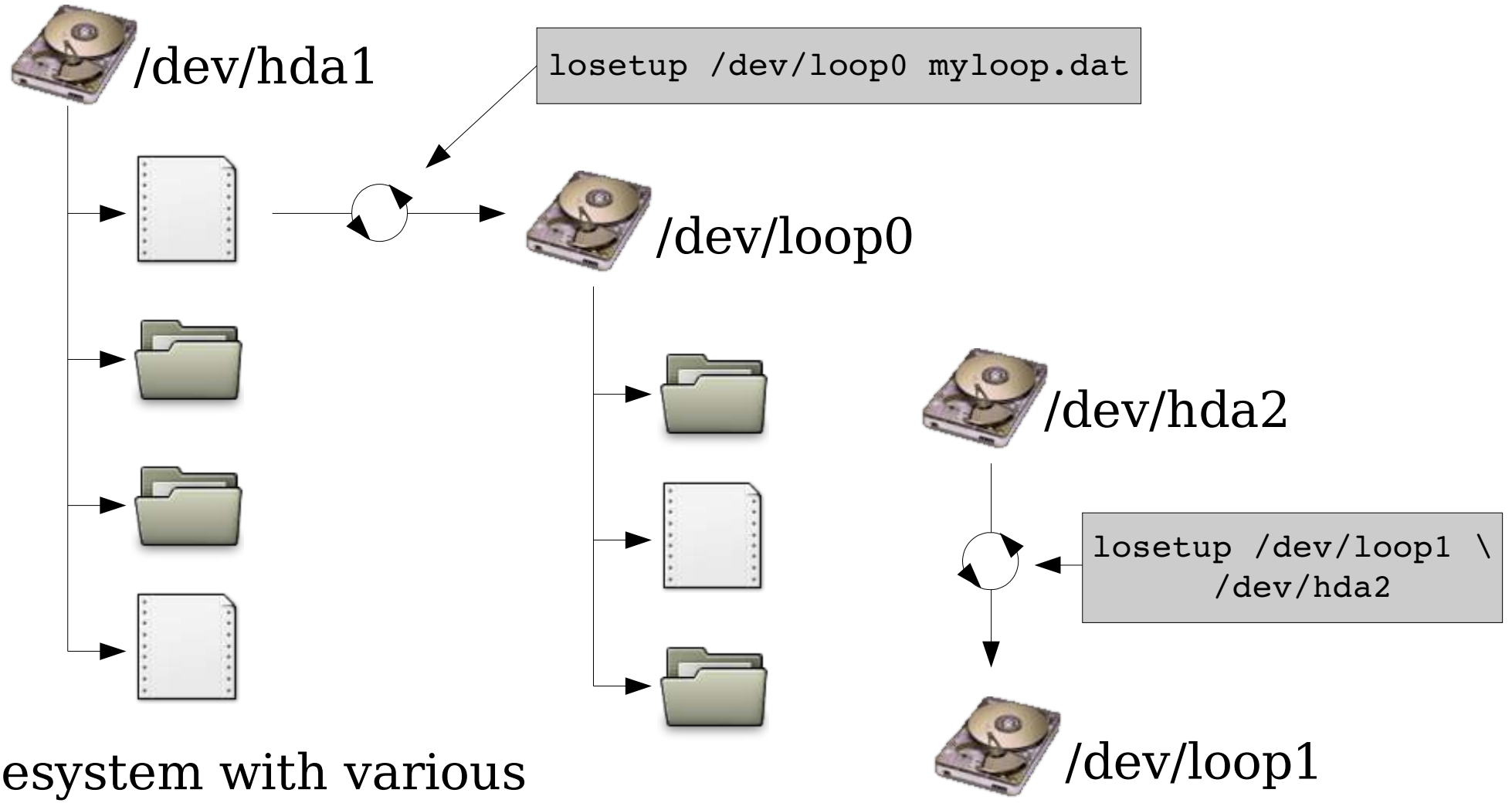
Cryptographic Filesystems

Background
and
Implementations
for Linux
and OpenBSD

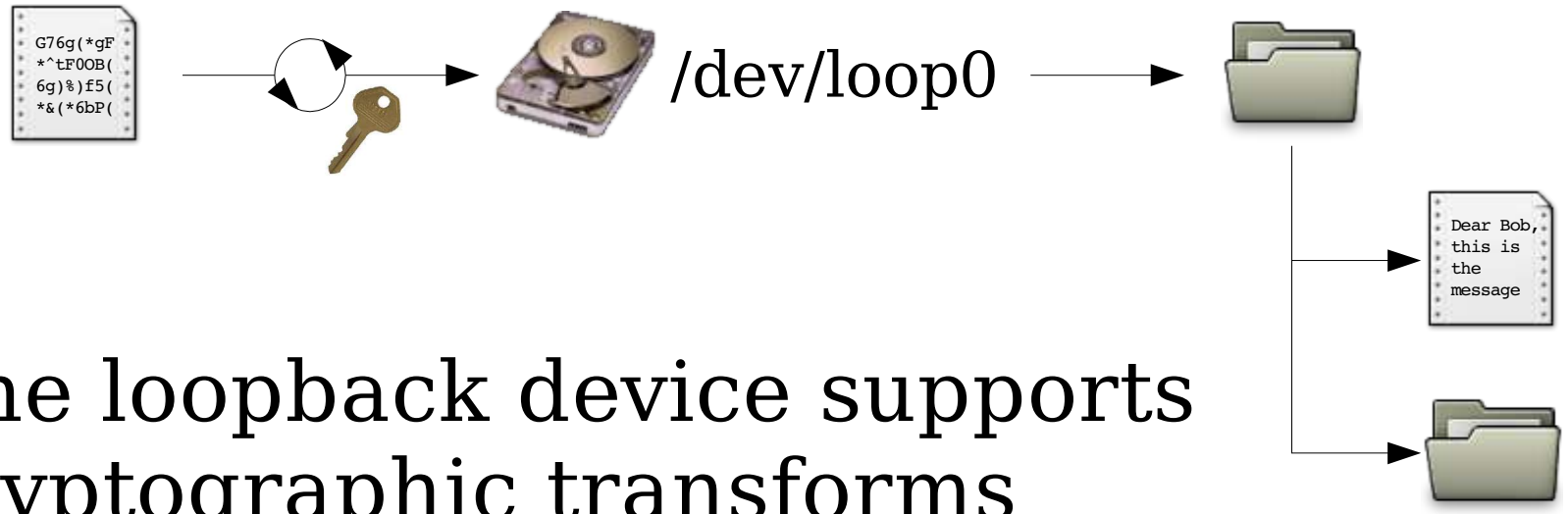
Background

- Loop device primer
- Keys, cipher modes, salting, key hashing
- Journaling file systems and encrypted swap
- Offset, sizelimit, hard/soft block size paramters explained
- Issues with software suspend

Loopback Primer



Loopback Primer



- The loopback device supports cryptographic transforms
- Knowing the key reveals a filesystem within a normal file (or block device)

Keys

- Encryption with user supplied key
- Encryption with hashed key
  - rmd160, sha256, sha384, sha512, ...
- Using multiple keys (typically 64)
  - every sector (512b) uses different key

Basic block cipher modes

- Currently only ECB (Electronic Code Book) and CBC (Cipher Block Chaining) widely supported
- ECB directly encipher plaintext
  - Same plaintext block always encrypts to same ciphertext. Is considered weak.
- CBC uses the previous block as IV
  - Basically solves ECB's dictionary problem
  - Only provides very limited integrity

Key iterations and salt

- Key can be hashed 1000 times
  - significantly slows down brute force attack on password
- `Salt` is a string appended to the password
  - slows down brute force attack on password
  - poor man's two factor authentication

Journaling

- Fine on device backed loop device
- Avoid on file backed loop devices
  - The VM can write pages in any order it likes and therefore breaks write order expectation of journaling filesystem
  - Underlying FS must be journaled and guarantees `data=ordered` and `data=journal`

Encryped swap

1)System runs out of memory

2)Pages flushed into (encrypted) swap

3)This requires encryption

4)Encryption requires memory

- unless special care is taken (loop-jari)

5)System very busy with swapping the same pages in and out

Other issues

- Offsets and sizelimits can be used for compatibility or to make partition tables futile
- Software suspend writes your kernel RAM (keys !) to the disk
- CDROM/DVD: Filesystem soft sector size must be an integer multiple of device hard sector size.

Kernel: where it starts to be confusing

loop-AES:

- Ciphers identified by numbers (1-20)
- Comes with it's own set of ciphers compiled as individual modules.

CryptoAPI:

- Registers itself as cipher 18.
- Ciphers added to the kernel and can be included in the kernel or compiled as modules

Userland: What is the right variant of losetup ?

- util-linux (losetup, mount)
  - Jari Ruusu's patches (gpg, aespipe)
  - Ben Slusky's patches (hashalot)
  - Debian patches (hybrid ?)
  - busybox's losetup
  - etc.
- All have different sets of functionality
- Default parameters change sometimes

Good news: In reality it's easy

- Creating and mounting a loopback device

```
losetup -e ${AGLO} /dev/loop0 /dev/hda1
mkfs.ext2 /dev/loop0      #format the loopback device
mount /dev/loop0 /usr     #mount like a normal device
umount -d /dev/loop0      # -d frees the loopback
```

- or

```
mount testfile.dat /mnt -o loop,encryption=${ALGO}
```

- or put it into your fstab file

What the previous slide left out

- `${ALGO}` can be something like `twofish-cbc-256` (kerneli) or `AES192` (jari)
- Make sure you know how your variant of `losetup` hashes the password.

OpenBSD

- One clear interface though vnodes:

```
vnconfig svnd0 -k myfile.dat
```

- No choice of cipher (blowfish), hash, ...
- Encryption of swap simply by setting:

```
sysctl -w vm.swapencrypt.enable=1
```

Installation of the international patch

- With 2.6.0 integrated into mainstream kernel
- Kernels <=2.4 need kernel patching
  - patch-int-2.4.21.0.gz
- Apply one loop patch of your choice:
  - patch-cryptoloop-hvr-2.4.22.0
  - patch-cryptoloop-jari-2.4.22.0

Installation of the international patch

- Plain util-linux-2.12 useless in most cases (no --pass-fd parameter)
- Debian util-linux does the job
- Or apply Ben Slusky's patch to util-linux
- Or have a look at the scatterlist cryptoapi (tm) inside backported 2.6.0 patches

Installation of loop-AES

- Very well documented
- Build loop-AES from sources (does not touch kernel)
- Patch util-linux-2.12
- load modules and you are ready
- uses initrd to allow encrypted root FS
- Jari's initrd also works with CryptoAPI

Other implementations

- Deniable encryption
  - http://www.rubberhose.org
- CFS
  - http://www.crypt.com/papers/cfs.anounce
- Cryptfs (stackable vnode fs)
  - http://www1.cs.columbia.edu/~ezk/research/cryptfs/index.html

Future ?

- Cryptography done in hardware
  - OpenBSD's Crypto(9) API
  - VIA Eden's AES engine
  - Todays CPUs and caching minimise performance impact
- replace (broken) loop.c by device mapper

Conclusion

- Choose kerneli, jari or dm
- For a workstation performance is not an issue
- Integrity has to be established though other measures
  - tripwire, integrit, ...
  - integrity aware block cipher modes

URLs

- International Kernel
  - http://www.kerneli.org/
- loop-AES
  - http://loop-aes.sourceforge.net/
- util-linux
  - http://ftp.cwi.nl/aeb/util-linux/
- Ben Slusky's patch to util-linux
  - http://www.stwing.org/~sluskyb/util-linux/

URLs

- linux-2.6.0-test3 scatterlist backport
  - http://www.kernel.org/pub/linux/kernel/people/hvr/testing/
    - patch-cryptoloop-jari-2.4.22.0
    - patch-cryptoloop-hvr-2.4.22.0