HOWTO: Disk encryption with dm-crypt / LUKS and Debian

A few weeks ago I published a small HOWTO for using loop-aes to encrypt your hard drive, usb thumb drive etc.

As I have bought a new 300 GB external USB disk drive on Friday, I have tried something new this time: disk encryption using dm-crypt / LUKS. It has been suggested to me multiple times that dm-crypt is superior to loop-aes, however I didn't get a real reason. Yes, it doesn't require any kernel patches and is easier to setup. But has any serious cryptographer looked at it sharply, yet? Did it withhold his eye contact?

Anyways, here's how I encrypted my 300 GB drive. I largely followed the guide at the EncryptedDeviceUsingLUKS wiki page...

1. Make sure you run Linux 2.6.16 or better. Previous versions suffer from an implementation problem which affects the security of dm-crypt, see Linux Kernel dm-crypt Local Cryptographic Key Disclosure.
2. Enable the following options in your kernel:
   - Code maturity level options
     - Prompt for development and/or incomplete code/drivers
   - Device Drivers -> Multi-device support (RAID and LVM)
     - Device mapper support
     - Crypt target support
   - Cryptographic options
     - AES cipher algorithms
3. Overwrite the whole drive with random data in order to slow down attacks on the encryption. At the same time perform a bad blocks scan to make sure the hard drive is not going to die too soon:
   badblocks -c 10240 -s -w -t random -v /dev/sdb
   Replace /dev/sdb with whatever is correct on your system. If you're really paranoid, and are willing to wait one or two days, do this:
   dd if=/dev/urandom of=/dev/sdb
4. Install the required packages:
   apt-get install cryptsetup hashalot
   The current cryptsetup in Debian unstable already supports LUKS, which was not the case a while ago, if I'm not mistaken. So Debian testing or stable will most probably *not* work!
5. Create one or more partitions on the drive:
   cfdisk /dev/sdb
   I created one big 300 GB partition, /dev/sdb1.
6. Setup LUKS:
   cryptsetup --verbose --verify-passphrase luksFormat /dev/sdb1
   Enter a good passphrase here. Don't spoil the whole endeavour by chosing a stupid or short passphrase.
7. Open the encrypted device and assign it to a virtual /dev/mapper/samsung300gb device:
   cryptsetup luksOpen /dev/sdb1 samsung300gb
8. Create a filesystem on the encrypted device:
   mkfs.ext3 -j -m 1 -O dir_index,filetype,sparse_super /dev/mapper/samsung300gb
   I used ext3 with some optimizations, see mke2fs(8).
9. Mount the encrypted partition:
   mkdir /mnt/samsung300gb
   mount /dev/mapper/samsung300gb /mnt/samsung300gb
   That's it. Everything you write to /mnt/samsung300gb will be encrypted transparently.
10. For unmounting use:
    umount /mnt/samsung300gb
    cryptsetup luksClose /dev/mapper/samsung300gb

After unmounting, nobody will be able to see your data without knowing the correct passphrase. Drive is stolen? No problem. Drive is broken, and you want to send it in for repair without the guys there poking in your data? No problem. You leave the USB drive at home and some jerk breaks into your house, steals your drive, rapes your wife, and kills your kids? No problem. Well, sort of, but you get the idea ;-)

There's more things you can do, thanks to LUKS: have multiple passphrases which unlock your data, change/add/remove passphrases as you see fit, etc.
