RED HAT
DEVELOPER CONFERENCE 2012
February 17–18th

devconf.cz

FACULTY of INFORMATICS
MASARYK UNIVERSITY, BRNO

Disk encryption ...
(not only) in Linux

Milan Brož
mbroz@redhat.com

redhat

FDE - Full Disk Encryption

- **FDE** (Full Disk Encryption) – whole disk

- **FVE** (Full Volume Encryption) – just some volumes

- (dis)advantages?

  + for notebook, external drives (**offline protection**)
  + **transparent for filesystem**
  + **no user decision later** what to (not) encrypt
  + hibernation, swap
  + key removal = **easy data disposal**

  - more users – whole disk accessible
  - key disclosure – complete data leak
  - for sw sometimes performance problems

FDE - Full Disk Encryption

		Examples (illustrative)
sw	• **userspace**	• Truecrypt (FUSE handled)
	• **sw driver** (encryption on CPU)	• dm-crypt, Truecrypt (native), loop-AES, ...
	• **driver + hw** (hw acceleration)	AES-NI, Via Padlock, special chips (mobile)
hw	• **disk controller**	**Chipset FDE** External disk drives with USB interface with "full hw encryption"
	• **on-disk**	**HDD FDE** special SSD/USB drives

Block device, sector

- **Sector – disk atomic IO unit**
  - 512 bytes, 4096 bytes

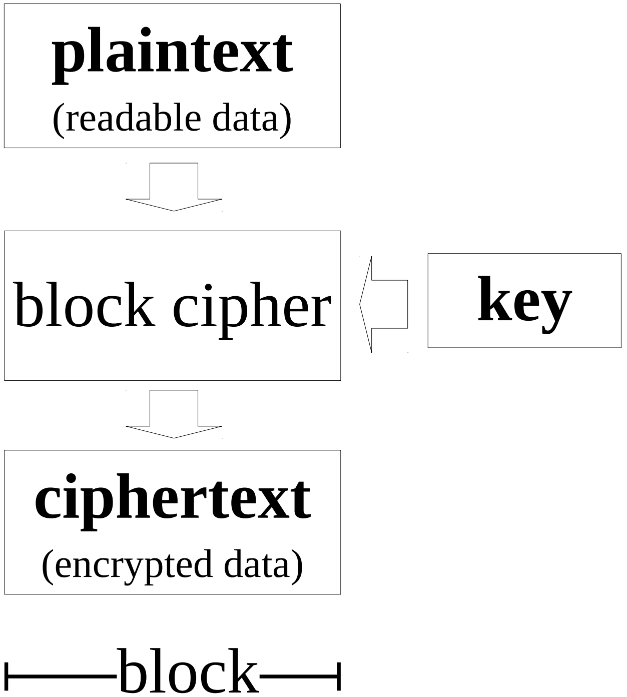
- **Block device**
  - disk, partition, virtual block devices (MD, device-mapper, loop)
  - block device stacking

    To avoid block/sector confusion here
    sector = disk unit (typically 512 bytes)
    block = encryption block (typically 16 bytes)

  - **FDE – encrypted block device -> decrypted block device**

Plaintext & ciphertext

- **plaintext – original (open) data**
  - virtual device
- **ciphertext – encrypted data**
  - hw disk
- **symmetric algorithms** (secret key)
  - speed, throughput (~disk)
  - **block** as atomic unit (~16 bytes)



plaintext
(readable data)

block cipher

key

ciphertext
(encrypted data)

block

Cipher block mode, Initialization vector

- **BLOCK** (of cipher) **< SECTOR** (of disk)

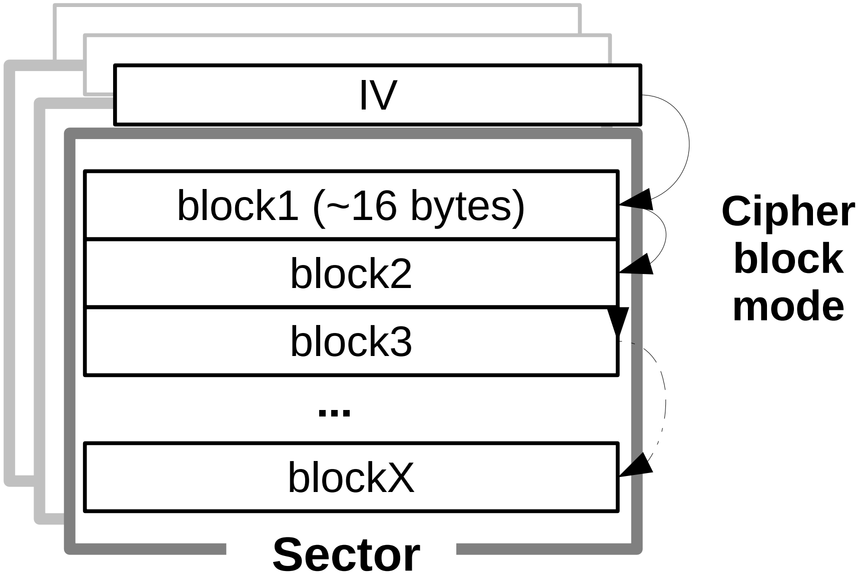
- split sector to blocks

- chained/parallel processing

  block mode

- Problem:
  same data in different sectors
  – different ciphertext

  initialization vector IV (tweak)
  (different for every sector)



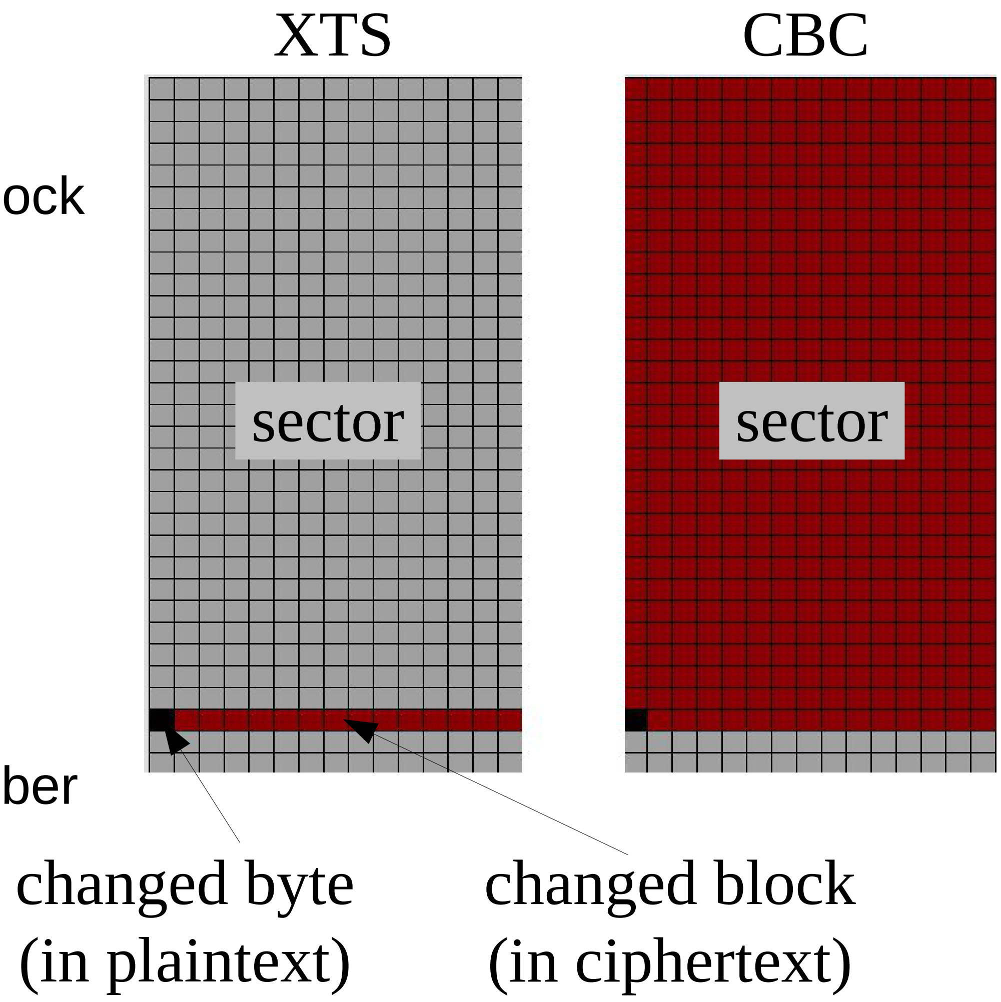
  - usually derived from seq. sector number (and key, if needed)

  - e.g. ESSIV – Encrypted Salt-Sector IV

Cipher block mode – examples

How a single change in plaintext changes ciphertext?

- **CBC** – cipher block chaining
  - ciphertext XOR with next block
- **XTS / XEX** (XOR encrypt XOR)
  - internally 2 keys
    - key for tweak
    - encryption key
  - IV can be directly sector number (known to attacker)



Block mode vs sector

Goal:
arbitrary change (plaintext) – **change of the whole sector** (ciphertext)

Solution:

- **wide mode** (encryption block size = sector size)
  - requires at least 2x encryption loop
  - modes are patent encumbered (~ but free standard EME-2)
  - not used in implementations
- **additional operations**

  example – **Elephant diffuser** in Bitlocker

  - special operation before CBC (mix/rotate input)
  - tweak key (independent of encryption key)

Disk encryption + data channel encryption

Example*: iSCSI exported encrypted disk, decryption on client side.*
Is there plain data on data channel? No.
So it is secure? ***No.***

- **FDE is offline protection** (stolen disk)
  - attacker cannot access snapshots in time
    (repeated access to hw, much worse attack vectors)
  - mode designed for transparent disk access
    (**IV is always constant for sector**)
- **Encrypted data channel**
  - Attacker can listen the whole communication
    but he **cannot replay data** – reply attack.

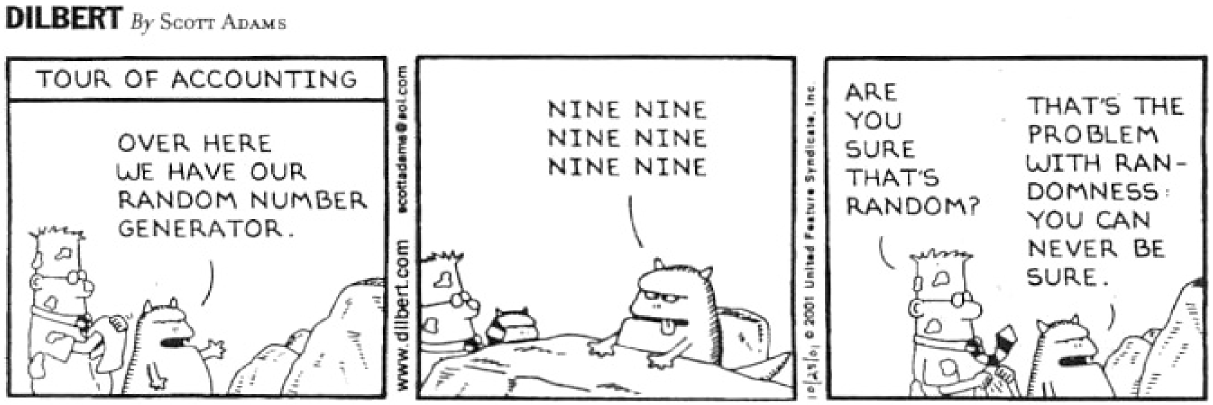
NEVER use encryption designed for exact use for something else,
solve problems separately (FDE + ipsec).

Key Management

Key generator

- **very important for the encryption system security**
  Note difference: encryption key / unlocking passphrase

- **encryption key**
  random, unique
  generated by RNG (Random Number Generator)


DILBERT By Scott Adams
TOUR OF ACCOUNTING
OVER HERE WE HAVE OUR RANDOM NUMBER GENERATOR.
NINE NINE NINE NINE NINE NINE
ARE YOU SURE THAT'S RANDOM?
THAT'S THE PROBLEM WITH RANDOMNESS: YOU CAN NEVER BE SURE.
scottadams@aol.com
www.dilbert.com
10/25/01 © 2001 United Feature Syndicate, Inc.

or derived from passphrase
- i.e. PBKDF2 (Password Based Key Derivation)
- usually not desirable (~restricted in security policy)

Key storage

- **outside of encrypted device**
  - token, SmartCard, TPM, EEPROM
  - file (protected by another encryption system)
  - (encrypted) on another disk (separation of metadata)

- **on the same disk** (with encrypted data)
  - **metadata** (header)
  - unlocking using passphrase of different key
  - brute force and dictionary attack contermeasures (slow down attack)
  - hw problems (e.g. firmware sector reallocation)

- **integration with key management tools**
  - enterprise use (LDAP, Active Directory, ...)

Key removal

- **key removal** (wipe of key storage area) **= data disposal**
  - **intended** (secure disk disposal)
  - **unintended (error)**
    - the most common problem
    - metadata overwrite – operator error
    - hw error, bad sector, controller, TPM, ...

Key recovery

Trade-off between security and user-friendly approach.

- disk copy (metadata)
- **Key Escrow** (key backup to diferent system)
- **duplicated metadata** on disk
- **recovery key** to regenerate encryption key

- wrongly designed user-friendly "extensions" destroys security

  Examples (3rd party Linux based NAS ...)

  - CVE 2009-3200 - undocumented recovery key in flash memory,

    allows local users decrypt the hard drive.

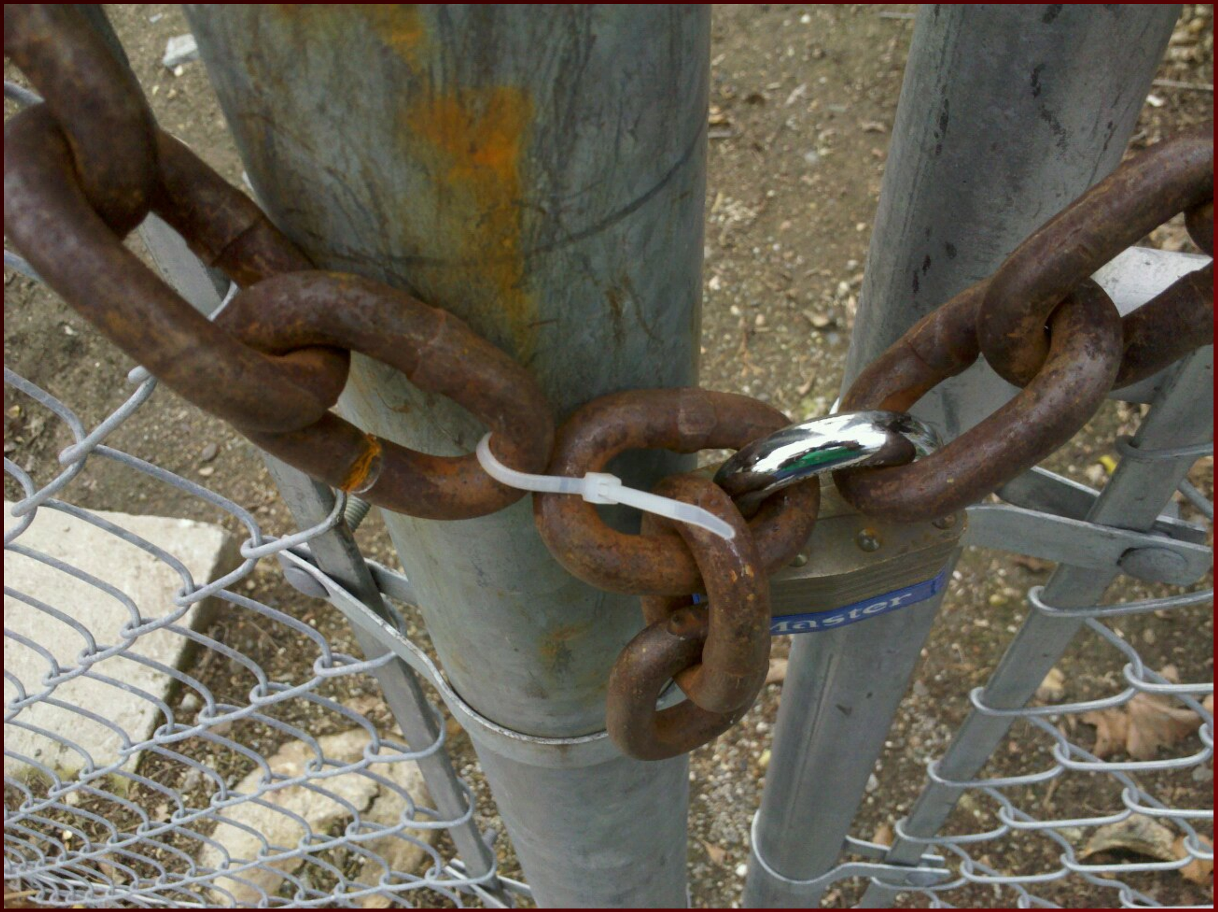
    CVE 2009-3278 - use the rand library to generate recovery key,

    brute-force attack possible.

    CVE 2008-1431 - firmware stores a partition encryption key

    in an unencrypted file with base64 encoding.

  ...



Attacks ...

Attacks always get better, they never get worse.

- **Attacks to algorithm**
- **Attacks to implementation**
  - e.g. side channels
- **Obtaining key or passphrase in open form**
  - hw attack (keylogger, Cold Boot)
  - malware – boot / OS / hypervisor modification
  - social engineering

If you let your machine out of your sight,
it’s no longer your machine.

Examples of FDE implementations

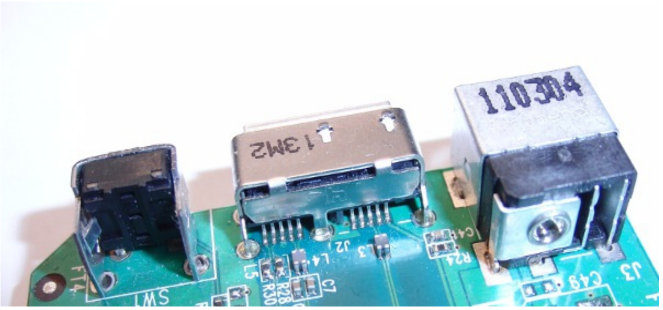
Chipset FDE

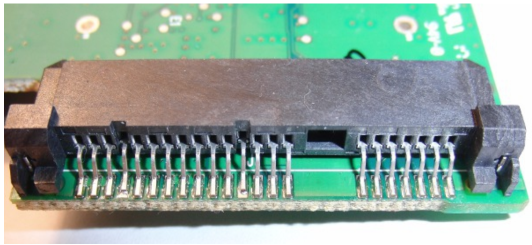
- **Encryption on disk controller**
  - example: USB3 external disk enclosure
  - standard SATA disk
  - AES-256 encryption on chipset

Chipset FDE

- **Encryption on disk controller**
  - which mode is used?
  - where and how is the encryption key stored?
  - HW board failure – what happens?

- Recovery: you need the same board / firmware
- Encryption always present (even if password is empty!)
- weak part: connectors on board





- proprietary key storage format
- proprietary key handling protocols

Truecrypt

Truecrypt, www.truecrypt.org

- the most known "opensource" disk encryption system

- AES, Twofish, Serpent
- chained ciphers (e.g. AES-Twofish)
- XTS mode

- hidden disk (including disk with OS), bootloader
- not using TPM

- on-disk metadata encrypted (no detectable header)
- duplicated metadata
- recovery CD

- on Linux with dm-crypt backend
- tc-play reimplementation with BSD license

loop-AES

loop-AES, loop-aes.sourceforge.org

- project outside of the main kernel tree
- loop device extension

- AES, (Twofish, Serpent)
- modified CBC mode (IV derived from sector, key and plaintext)
- multikey – 64 keys (modulo sector) + key for IV

- external store for key in file (GPG encrypted)

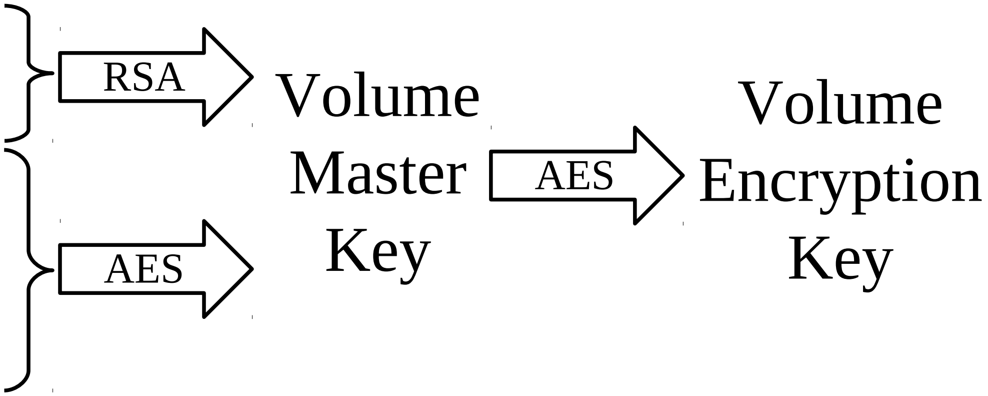
- dm-crypt / crypsetup loop-AES compatible mode

BitLocker (Windows proprietary)

Native FDE in Windows Ultimate edition

- in future combined with "secure boot" (Windows 8)
- many options (system policy)

- TPM
- TPM + PIN
- TPM + Startup Key
- Clear Key
- Startup/Recovery Key
- Recovery Password



RSA

AES

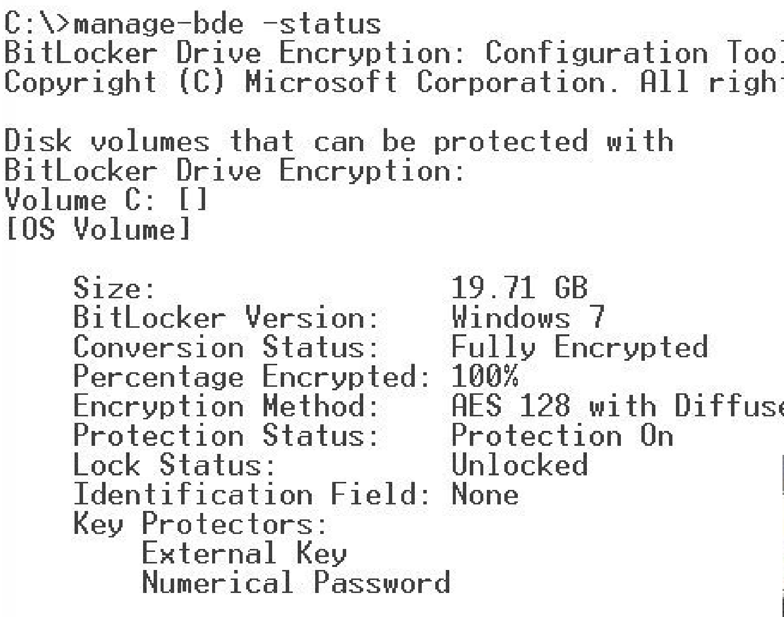
Volume
Master
Key

AES

Volume
Encryption
Key

- AES 128 CBC
- AES 128 CBC + Elephant diffuser
- AES 256 CBC
- AES 256 CBC + Elephant diffuser

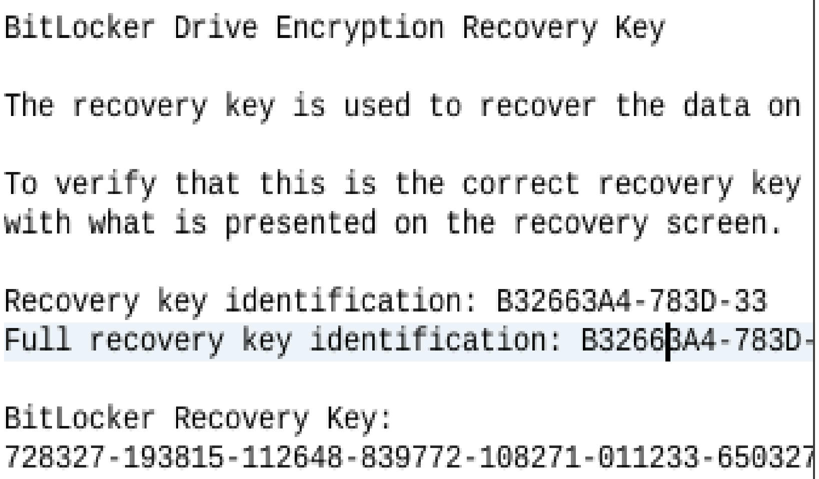
BitLocker (Windows)



```
C:\>manage-bde -status
BitLocker Drive Encryption: Configuration Tool version 6.1.7601
Copyright (C) Microsoft Corporation. All rights reserved.

Disk volumes that can be protected with
BitLocker Drive Encryption:
Volume C: []
[OS Volume]

    Size:                 19.71 GB
    BitLocker Version:    Windows 7
    Conversion Status:    Fully Encrypted
    Percentage Encrypted: 100%
    Encryption Method:    AES 128 with Diffuser
    Protection Status:    Protection On
    Lock Status:          Unlocked
    Identification Field: None
    Key Protectors:
        External Key
        Numerical Password
```



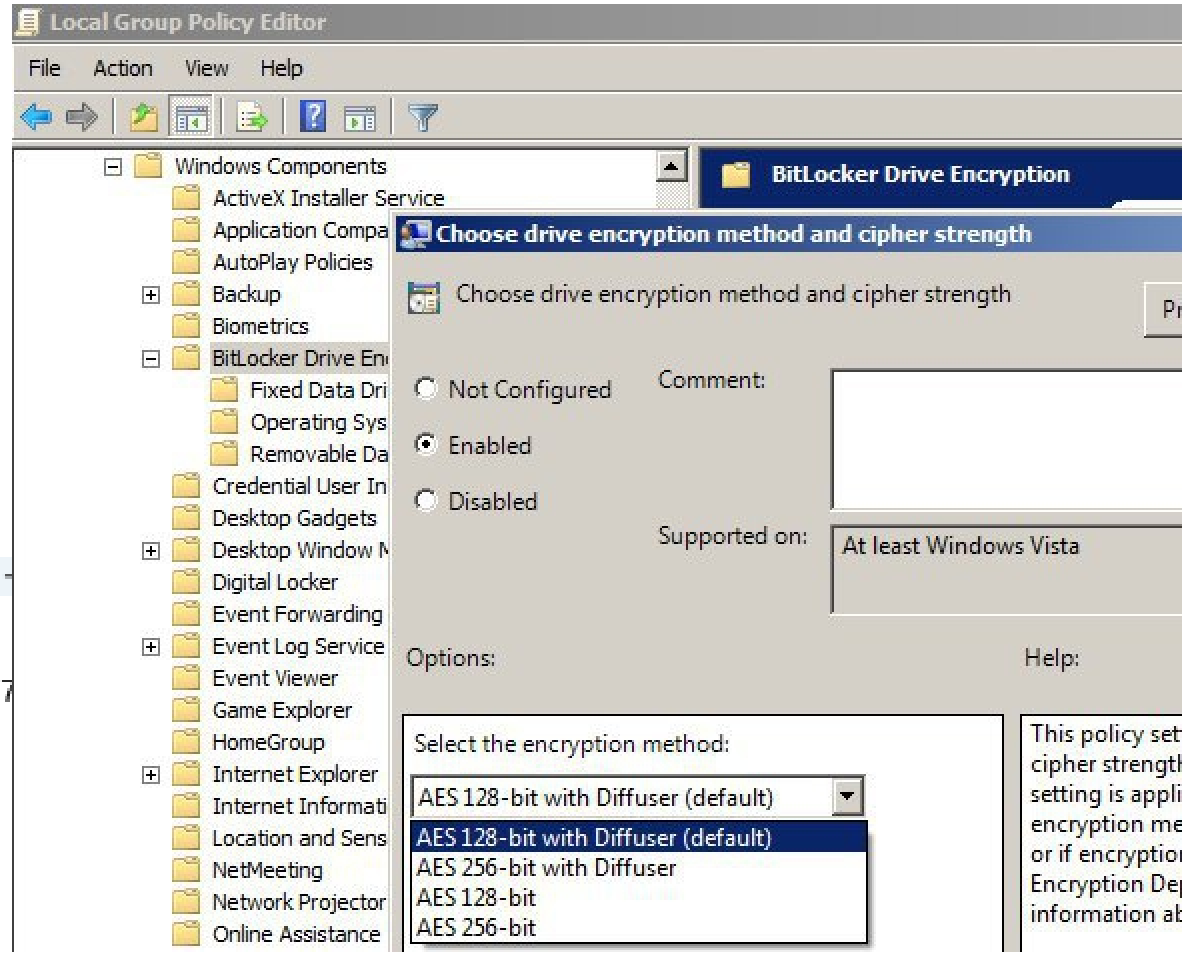
```
BitLocker Drive Encryption Recovery Key

The recovery key is used to recover the data on

To verify that this is the correct recovery key
with what is presented on the recovery screen.

Recovery key identification: B32663A4-783D-33
Full recovery key identification: B32663A4-783D-

BitLocker Recovery Key:
728327-193815-112648-839772-108271-011233-650327
```



Choose drive encryption method and cipher strength

Choose drive encryption method and cipher strength

○ Not Configured

● Enabled

○ Disabled

Comment:

Supported on: At least Windows Vista

Options:

Help:

Select the encryption method:

AES 128-bit with Diffuser (default)

- AES 128-bit with Diffuser (default)
- AES 256-bit with Diffuser
- AES 128-bit
- AES 256-bit

This policy set
cipher strengtl
setting is appli
encryption me
or if encryptio
Encryption De
information ab

LUKS / dm-crypt

- **Native on Linux**

- strict separation of
  - **disk encryption engine**
    dm-crypt – device-mapper crypto target (kernel)
  - **key management (LUKS) and configuration**
    cryptsetup – userspace

- never implements crypto primitives itself
  - kernel cryptoAPI
  - userspace crypto libraries

- variability
- supports most of the other subsystem formats
  (with exception of diffuser and nonstandard encrypted sector size)

dm-crypt (kernel)

- maps virtual plaintext device
- no key management (ioctl uses key directly)

- device stacking (~ chained ciphers)

Cipher specification examples

- aes-cbc-essiv:sha256 (AES, CBC, ESSIV)
- aes-xts-plain64 (AES, XTS, IV is sector number)
- aes:64-cbc-lmk (loop-AES multikey compatible mode)

... and many other compatible modes (not secure)

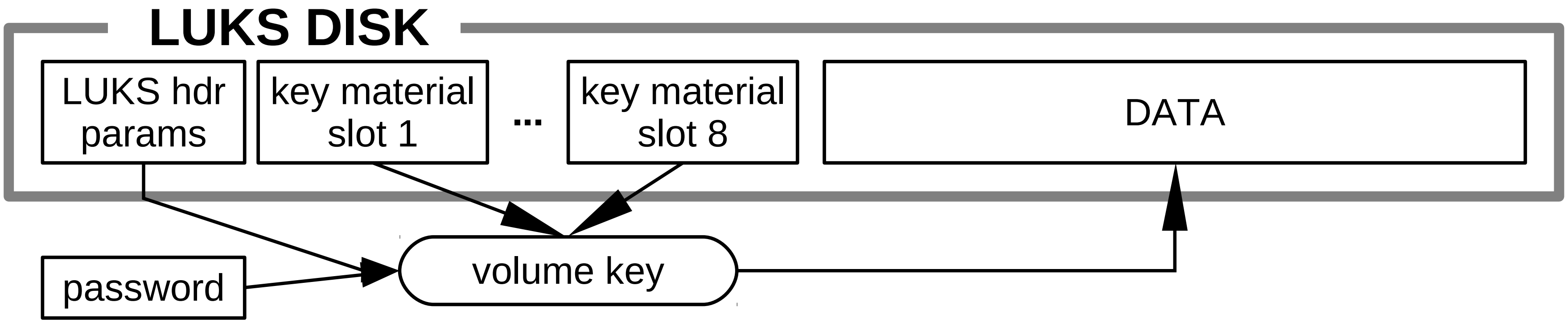
- twofish-ecb
- serpent-cbc-plain64

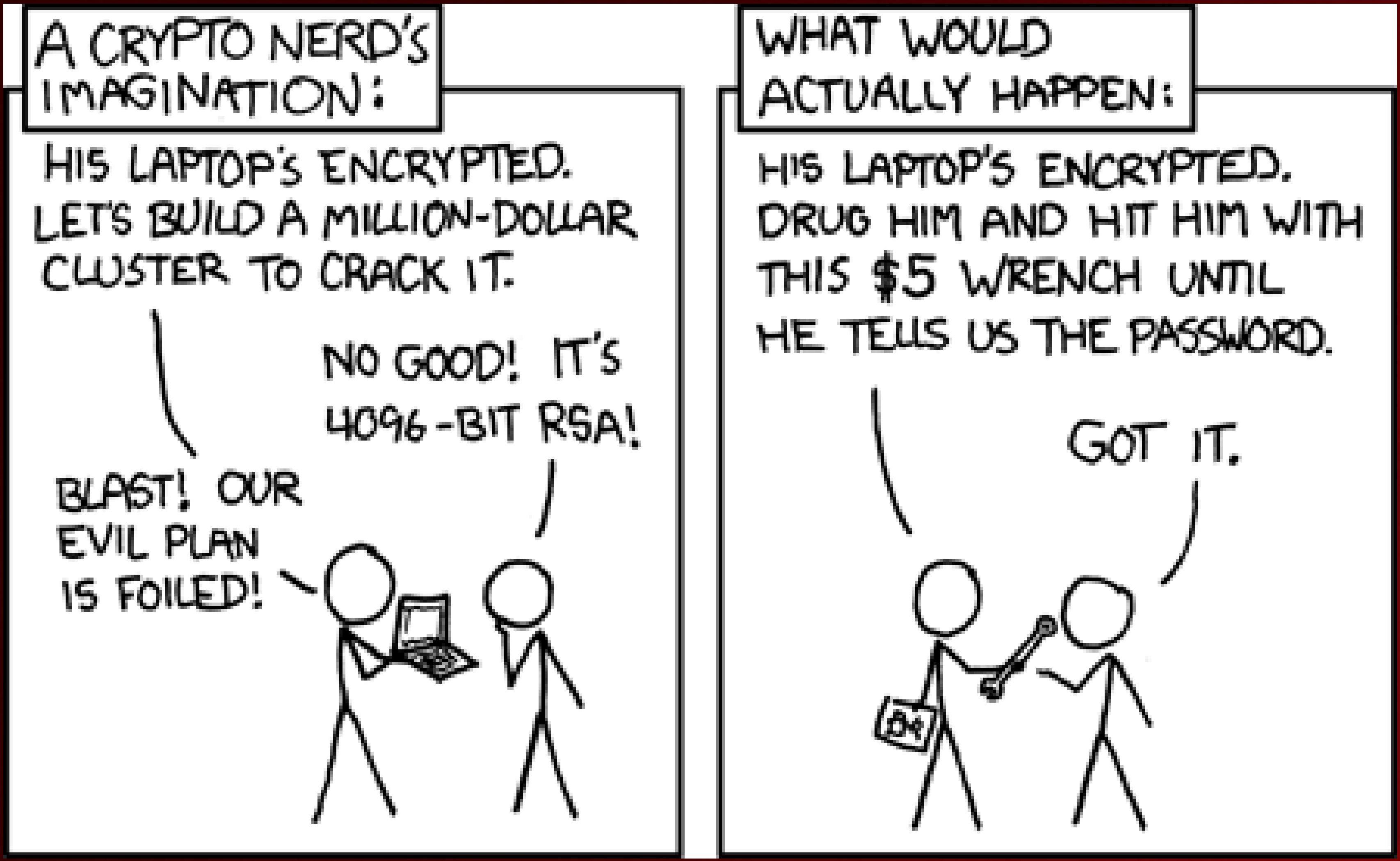
...

LUKS (Linux Unified Key Setup)

- Simple key / passphrase management system for dm-crypt

- de facto standard for disk encrytpion in Linux, portable
- **more passphrases (keyslots)**
- uses iterated PBKDF2 store key – slow down dictionary attacks
- **passphrase change** – no need to reencrypt disk
- **AF-splitter** – anti-forensic (fw sector reallocation issue)




A CRYPTO NERD'S IMAGINATION:
HIS LAPTOP'S ENCRYPTED. LET'S BUILD A MILLION-DOLLAR CLUSTER TO CRACK IT.
NO GOOD! IT'S 4096-BIT RSA!
BLAST! OUR EVIL PLAN IS FOILED!
WHAT WOULD ACTUALLY HAPPEN:
HIS LAPTOP'S ENCRYPTED. DRUG HIM AND HIT HIM WITH THIS $5 WRENCH UNTIL HE TELLS US THE PASSWORD.
GOT IT.

http://imgs.xkcd.com/comics/security.png

thanks for your attention