Stack-basierte Festplattenverschlüsselung

eCryptFS

Robby Zippel
robby.zippel@informatik.stud.uni-erlangen.de

26. Mai 2010

Inhaltsübersicht

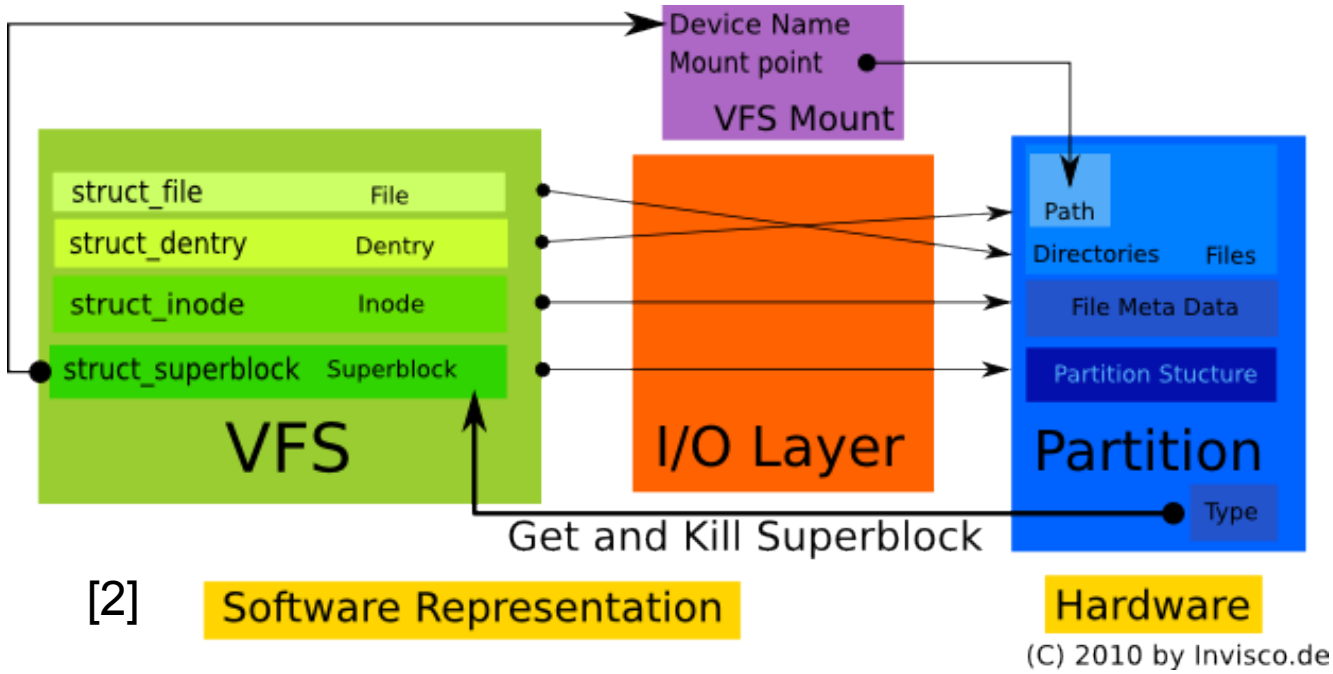
- Grundlagen
  - Virtual File System
  - FiST – Framework
- eCryptFS
- Fazit

Stack-basierte Verschlüsselung

- Verschlüsselungsdateisystem ist eine Schicht über dem logischen Dateisystem
- Framework zur Entwicklung solcher Dateisysteme: FiST

Virtual File System - VFS

- Abstraktionsschicht oberhalb konkreter Dateisysteme
- Verarbeitet alle Systemaufrufe in Bezug auf ein Linux Dateisystem und leitet diese an konkrete Dateisysteme weiter
- Interoperabilität zwischen verschiedenen Dateisystemen



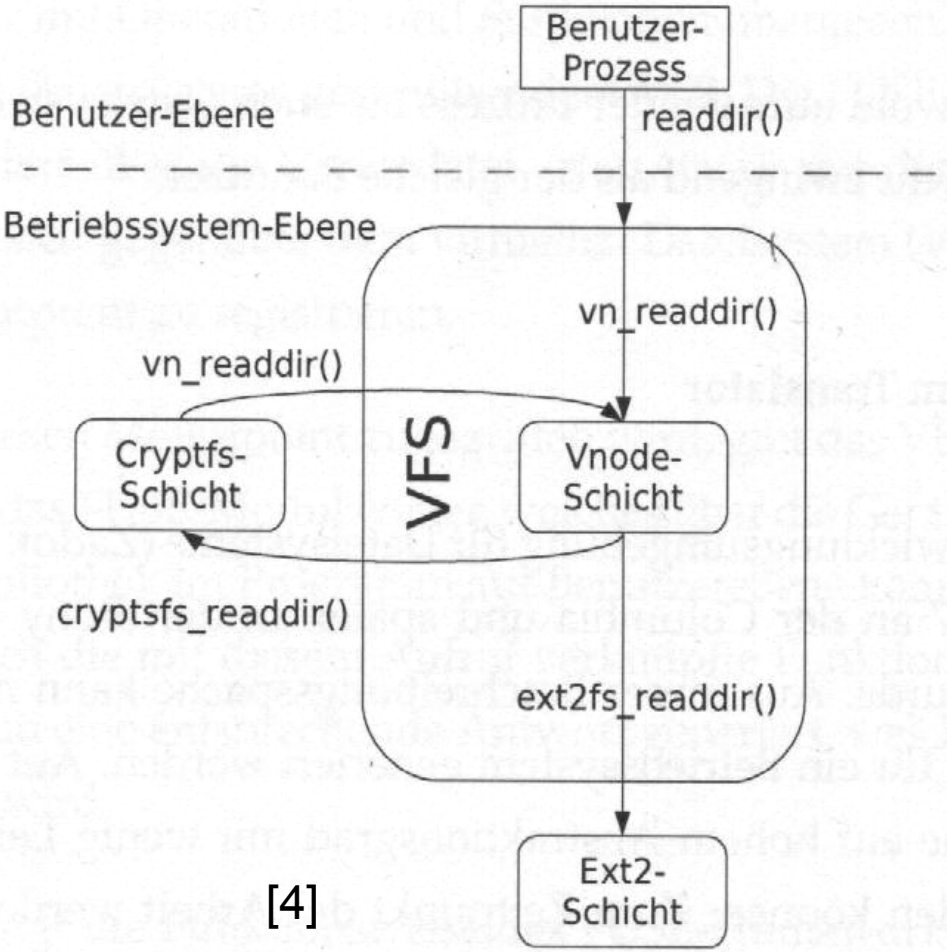
[2]

FiST – File System Translator

- Dateisystembeschreibungssprache
  - Dateisysteme auf hohem Abstraktionsgrad mit wenig Leistungseinbußen entwickelbar
  - Nur geschichtete Dateisysteme implementierbar
- Ziel: Plattformunabhängiges Framework

FiST – Vnode-stacking

- Einhängen mehrerer Ebenen übereinander im VFS
- Vnode repräsentiert Inode bzw. Dateideskriptor im Hauptspeicher



[4]

FiST - Aufbau

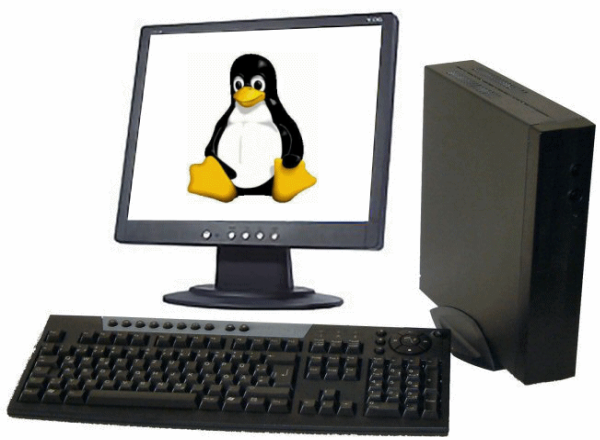
```
%{
1 C Declarations
%}
2 FiST Declarations
%%
3 FiST Rules
%%
4 Additional C Code
```

Inhaltsübersicht

- Grundlagen
- eCryptFS
  - Funktionsweise
  - Spezifisches
- Fazit

eCryptFS - Allgemein

- Unternehmenstaugliches natives Verschlüsselungsdateisystem für Linux
  - Seit Version 2.6.19 im Betriebssystemkern
  - POSIX-konform



- *Stand-alone* Kernel Modul
  - Keine Modifikationen am Linux-Kernel nötig
  - Arbeitet im Kernelspace

- Weiterentwicklung von CryptFS (1998 von Zadok)

Funktionsweise(1)

- Beim *Mounten* Authentifizierung durch:
  - Passphrase, Private/Public-Key-Paar, OpenSSL, Open-PGP, TPM-PKI, PAM

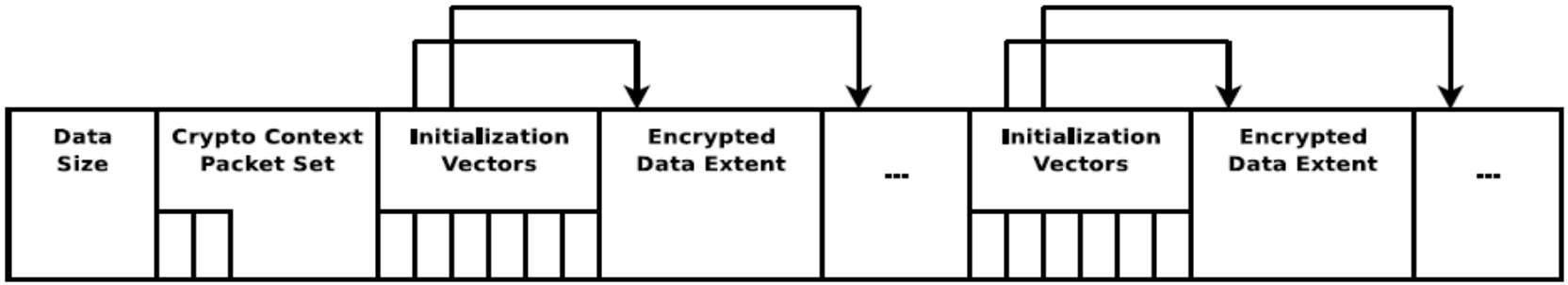
- *Mount-Helper* erstellt *Authentication Token* für Passphrase des Users (→ *Linux kernel keyring service*)

- eCryptFS nutzt Token-Inhalt für Erstellung neuer und Öffnen existierender Daten

Funktionsweise(2)

- Ablage der Verschlüsselungsinformationen im Header der Zieldatei (**bleibt unverschlüsselt**)
  - Leicht abgewandelter OpenPGP-Standard

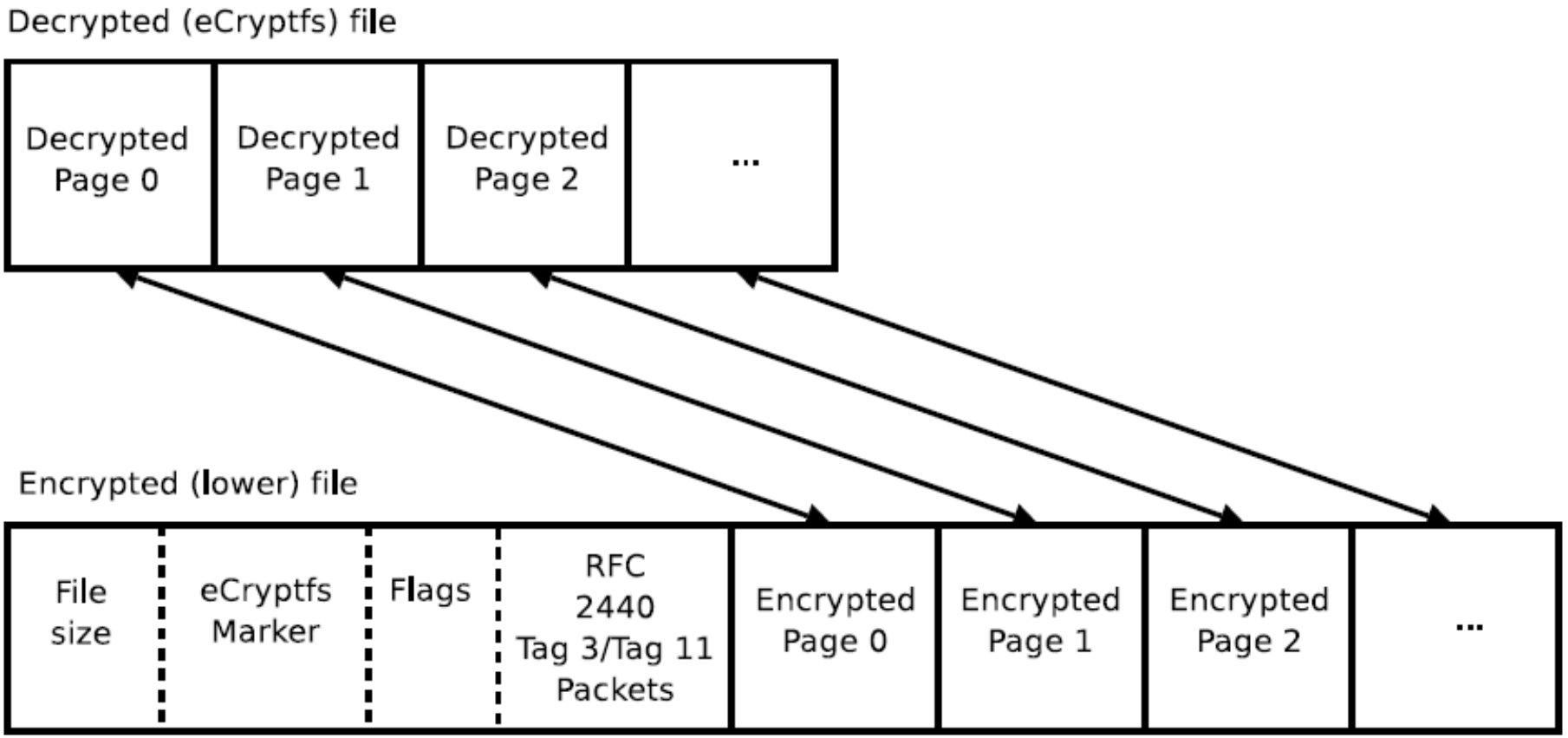
- Nutzdaten werden aufgespaltet (sog. *Extends*) und verschlüsselt (Standard: AES-128)



[5]

- Blockalgorithmus im CBC-Modus

Darstellung der Dateiverwaltung



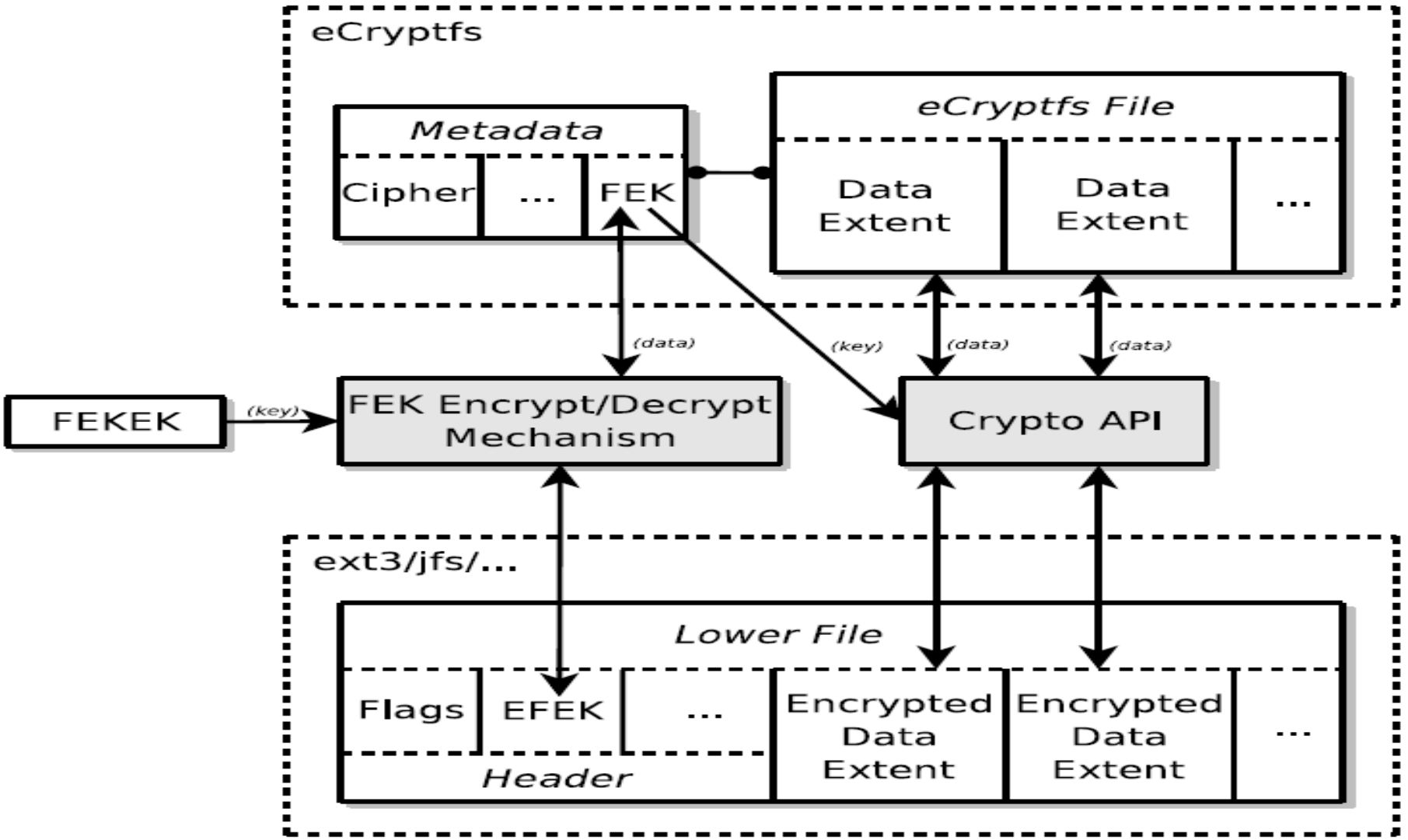
[5]

- Feingranulare Mehrbenutzerfähigkeit
  - Inode besitzt Array mit *Authentification Token* – Signaturen derjenigen Benutzer, die berechtigt sind diese Datei zu entschlüsseln

Datei lesen & schreiben

- eCryptFS öffnet *lower File* und liest *Header*
- *Authentication Token* im Benutzer Schlüsselring?
- Ver- und Entschlüsselung mit *File Encryption Key* via *Crypto-API* aus Kernel

Datei lesen & schreiben



[5]

Spezifisches

- Verwendung von CBC bringt dessen Schwachstellen mit
  - Viele bekannte Angriffe auf CBC
  - Keine Parallelisierung des CBC-Moduls möglich
- Einfach Erweiterbar
  - Verwendung der PKI-API

Inhaltsübersicht

- Grundlagen
- eCryptFS
- Fazit
  - Fallstricke
  - Performance
  - TrueCrypt vs. eCryptFS

Fallstricke

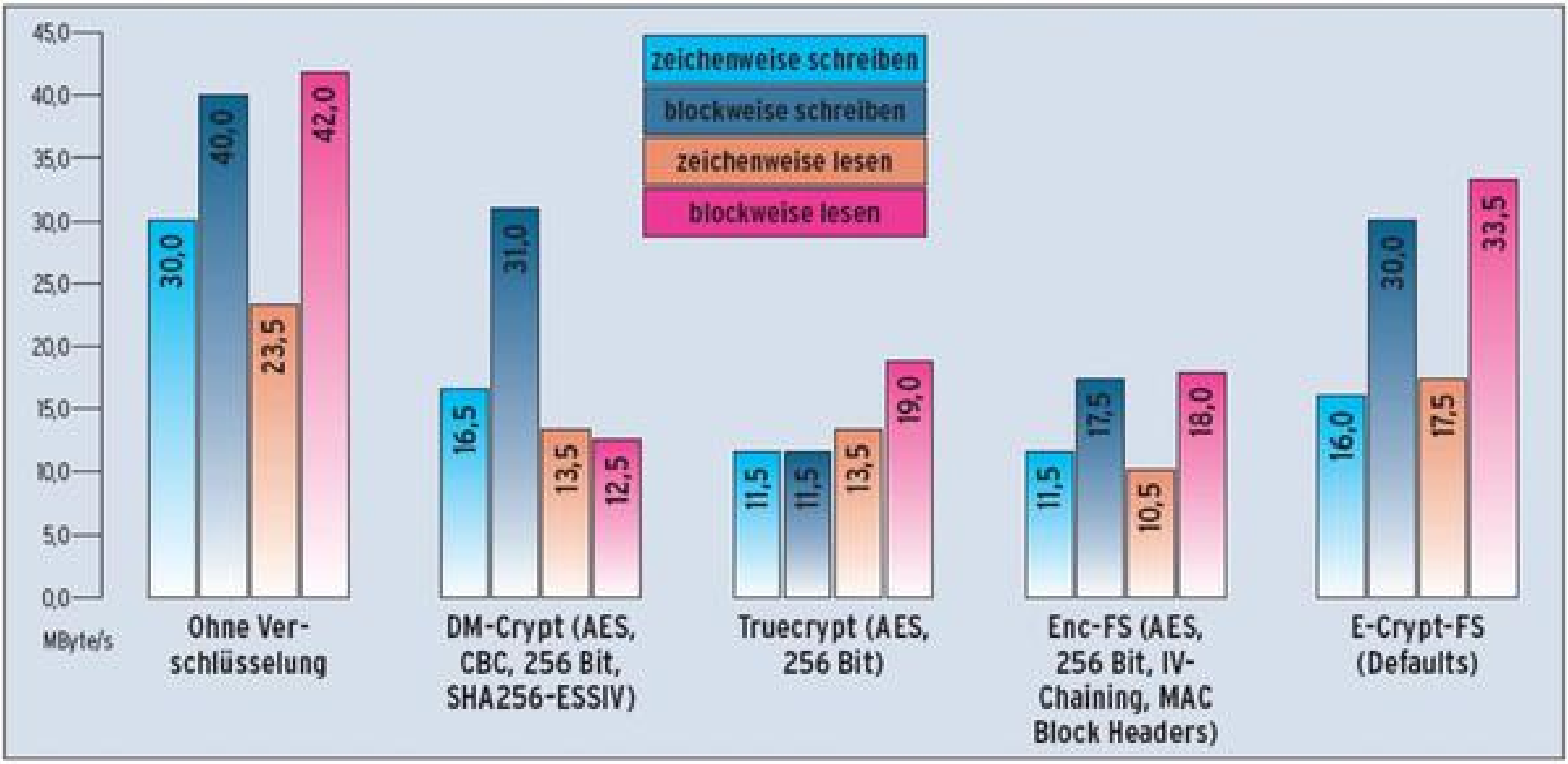
- Transparenz für Anwendungen durch einmalige Passworteingabe
  - Daten bis zum *unmount* unverschlüsselt
- Verlust des Passworts → Verlust der Daten
  - Beispiel:
    - Aufgesetzt und Liegengelassen
    - Passwortvergabe während durchzechter Nacht

- Im Ruhezustand landet der Festplattenschlüssel im *Hibernation-File*
- *Cold Boot Attack*

[6]

- Jede Datenbewegung verbraucht Rechenzeit
  - Mit aktueller Hardware allerdings kaum merklich
  - Ausnahme: Rechenintensive Anwendung und/oder hohe Datentransferraten
    - Videoschnitt, Server, ...

- Geschwindigkeit



[7]

TrueCrypt vs. eCryptFS – Zusammenfassender Vergleich

	TrueCrypt	eCryptFS
Konzept und Implementierung	Einfach	Sehr komplex
Speicherung verschlüsselter Daten	Allokieren eines Blocks aus dem vorhandenen Dateisystem; Komplette Partition verschlüsselt	Auf das gemountete Dateisystem; Ggf. Unverschlüsselte Daten auf gleicher Partition
Daten	Absolut nicht reproduzierbar; Vor Mounten keine Sichtbarkeit	MetaDaten unverschlüsselt; Sichtbar im Dateisystem
Portabilität	Ganzer Container, Partitionen nicht portabel	Einzelne Dateien, Verzeichnisse → sehr gut
Swap - Space	Verschlüsselbar	Nicht verschlüsselbar
Betriebssysteme	Windows, MacOS, Linux	Linux
Mehrbenutzerbetrieb	Dateiberechtigung des OS	User Access Control

Quellen

- [1] http://www.heise.de/ct/artikel/Datentresor-289846.html
- [2] http://www.invisco.de/files/VFS_Diagram.png
- [3] http://www.truecrypt.org
- [4] Max Lindner – Verschlüsselte Dateisysteme in Mehrbenutzer-Szenarien
- [5] http://pvs.informatik.uni-heidelberg.de/Teaching/DASY-07/seeliger.pdf
- [6] http://citp.princeton.edu/memory-content/memory_5.jpg
- [7] http://www.linux-magazin.de/Heft-Abo/Ausgaben/2007/03/Geheime-Geschaefte/(offset)/2
- http://ecryptfs.sourceforge.net/ecryptfs-faq.html
- http://sysphere.org/~anrxc/nsnd/nsnd-ecryptfs.html#sec-2.4

Quellen

- http://en.wikipedia.org/wiki/Comparison_of_disk_encryption_software
- http://pvs.informatik.uni-heidelberg.de/Teaching/DASY-07/nguyen.pdf
- Erez Zadok, Jason Nieh: FiST: A Language for Stackable File Systems
- Michael Austin Halcrow: eCryptfs: An Enterprise-class Cryptographic Filesystem for Linux
- http://berlin.ccc.de/~packet/cryptfs-eh02-workshop/cryptfs-eh02-workshop.pd
- http://koeln.ccc.de/archiv/drt//crypto/linux-disk.html#Cryptfs
- http://www.fsl.cs.sunysb.edu/docs/cryptfs/index.html
- c't Ausgabe 25/08: Lahmgesichert?

Any other questions?

I forgot my password/lost my key! What can I do to recover my data?

> *Nothing; you're screwed.*
>
> *http://ecryptfs.sourceforge.net/ecryptfs-faq.html*