

Verschlüsselte Dateisysteme

Vortrag zum Seminar „Dateisysteme“

26. Juni 2007
Ruprecht-Karls-Universität Heidelberg

Jochen Gärtner und Ralf Seeliger

- Warum verschlüsselte Dateisysteme?
- Ansätze
- Beispiel einer Blockchiffre: AES
- DM-Crypt
- Truecrypt
- Enc-FS
- verschlüsseltes Brennen auf CD / DVD
- Zusammenfassung

Dieser Vortrag soll eine Einführung in verschlüsselte Dateisysteme unter Linux und Windows geben. Zunächst werden verschiedene Ansätze verschlüsselter Dateisysteme allgemein besprochen, bevor auf einige Vertreter speziell eingegangen wird. Der letzte Teil zeigt zwei Möglichkeiten, Daten verschlüsselt auf CD und DVD zu brennen.

- Warum verschlüsselte Dateisysteme?
- Ansätze
- Beispiel einer Blockchiffre: AES
- DM-Crypt
- Truecrypt
- Enc-FS
- verschlüsseltes Brennen auf CD / DVD
- Zusammenfassung

- Warum ist der Einsatz verschlüsselter Dateisysteme sinnvoll?
- Welchen Vorteil bilden sie gegenüber Tools wie GPG, die Dateien einzeln verschlüsseln?
- Welche Anforderungen stellen wir an die Verschlüsselung?

verschiedene Bedrohungsmodelle

- Mediendiebstahl: Diebstahl der Daten einschließlich des Mediums, auf dem sie liegen (Laptop, Memorysticks, PDA, CDs)
- Bedrohungsmodell MIB („Men In Black“): Ein Angreifer hat regelmäßigen Zugriff auf den PC und kann Änderungen an Daten beobachten

Notwendigkeit einer Verschlüsselung sensibler Daten

Wir unterscheiden im Wesentlichen zwischen zwei Bedrohungsmodellen.

Das „MIB“-Projekt setzt voraus, dass ein Angreifer regelmäßig einbrechen und Daten einsehen bzw. manipulieren kann. Gegen einen solchen Angriff kann auch ein Crypto-Filesystem keinen ausreichenden Schutz mehr bieten, da der Angreifer Daten abfangen kann, bevor sie verschlüsselt auf Platte geschrieben werden, die Crypto-Software manipulieren kann oder mit einem Keylogger das Passwort stehlen kann. Dieses Modell ist daher nur von theoretischem Interesse und wir werden uns auf das Modell des Mediendiebstahls beschränken.

Probleme

- Integration von Verschlüsselungstechnologien in bestehende Abläufe ist aufwendig
- einfache Anwendbarkeit muss garantiert sein
- in Anwendungen integrierte Verschlüsselungstechnologien verlieren ihre Wirksamkeit, wenn verschiedene Anwendungen auf die Daten zugreifen müssen
- sensitive Daten müssen transportiert werden

Diese Probleme können beispielsweise entstehen, wenn ein Unternehmen Verschlüsselungstechnologien verwenden möchte, die nicht schon in die bereits existierende Software integriert sind.

Ein weiteres Problem ist, dass Anwender oft sorglos sind und Sicherheitsrichtlinien ignorieren.

Verschlüsselung sollte...

- transparent und flexibel sein
- universell und leicht anwendbar sein
- unabhängig von den Anwendungen sein, die auf die Daten zugreifen
- sicher genug sein, um ernste Angriffe abzuwehren sicheren Datenaustausch erlauben

Lösungsansatz: verschlüsselte Dateisysteme

Crypto-Dateisysteme nehmen die Ver- und Entschlüsselung im Hintergrund vor. Im Gegensatz zur expliziten Verschlüsselung wie beispielsweise bei GPG, kann dies im Hintergrund ablaufen, so dass der User auf seine Daten wie auf unverschlüsselte Daten zugreifen kann.

- Warum verschlüsselte Dateisysteme?
- **Ansätze**
- Beispiel einer Blockchiffre: AES
- DM-Crypt
- Truecrypt
- Enc-FS
- verschlüsseltes Brennen auf CD / DVD
- Zusammenfassung

Wir wollen zunächst eine grobe Klassifizierung der existierende Ansätze angeben.

- Block Device Layer Encryption (z.B. Loop-AES, DM-Crypt)
- transparente Verschlüsselung einzelner Dateien (z.B. Microsoft EFS, eCryptfs)
- explizite Verschlüsselung einzelner Dateien (z.B. GPG, Mcrypt)

- Bei der Block Device Layer Encryption werden ganze Blockgeräte (Partitionen, Volumes) verschlüsselt.
- Dem steht die transparente Verschlüsselung einzelner Dateien entgegen.
- Die explizite Verschlüsselung einzelner Dateien wollen wir hier nicht betrachten.

Während zunächst die Verschlüsselung einzelner Dateien (z.B. Microsoft EFS) im Vordergrund stand, war ein gewisser Trend hin zur Verschlüsselung kompletter Blockgeräte zu verzeichnen (z.B. DM-Crypt). Das seit der Version 2.6.19 im Kernel enthaltene eCryptfs stellt sich diesem Trend entgegen und setzt wieder auf die Verschlüsselung einzelner Dateien.

Block Device Layer Encryption: Verschlüsselung erfolgt auf Ebene von Blockgeräten (Partitionen, Volumes)

Vorteile:

- einfache Implementierung
- Angreifer hat keine Informationen über das Dateisystem (Typ, Verzeichnisbaum...)
- einfache Unterstützung von Sparse Files

Mit der Block Device Layer Encryption kann die Festplatte mit Ausnahme des MBR und der Boot-Partition vollständig verschlüsselt werden. Insbesondere ist auch ein verschlüsseltes Root-Dateisystem denkbar, hierfür ist allerdings eine Modifikation der initialen RAM-Disk nötig.

Block Device Layer Encryption

Nachteile:

- allokiertes Speicher ist fix
- alle Daten werden mit den gleichen Keys verschlüsselt
- ineffizient (nicht nur sensitive Daten werden verschlüsselt)
- beim Transfer der Daten in ein anderes Medium ist eine Neuverschlüsselung notwendig

Um Daten, die auf einer verschlüsselten Partition liegen, sicher weiterzugeben (z.B. auf CD), müssen diese erst mit einer User-Space-Anwendung neu verschlüsselt werden. Dies bedeutet auch einen Verlust der Transparenz.

transparente Verschlüsselung einzelner Dateien

Vorteile:

- keine unnötige Verschlüsselung von Daten
- verschiedene Keys für verschiedene Dateien möglich
- hohe Flexibilität (die verschlüsselten Dateien können sicher übertragen werden)

Das Verschlüsseln verschiedener Dateien bzw. Ordner auf einer Partition mit verschiedenen Schlüsseln bietet auch die Möglichkeit einer Benutzerverwaltung.

transparente Verschlüsselung einzelner Dateien

Nachteile:

- Log- und Konfigurationsdateien können Informationen über die verschlüsselten Dateien enthalten
- Metadaten (Größe, Name, Berechtigungen) bleiben sichtbar

Lösungsansatz: Kombination beider Ansätze

Eine mit eCryptfs verschlüsselte Datei besitzt einen Header mit 26 Bytes Metadaten. Diese enthalten u.a. die Größe der unverschlüsselten Datei, die für den Angreifer somit sichtbar bleibt.

- Warum verschlüsselte Dateisysteme?
- Ansätze
- **Beispiel einer Blockchiffre: AES**
- DM-Crypt
- Truecrypt
- Enc-FS
- verschlüsseltes Brennen auf CD / DVD
- Zusammenfassung

Jedem Crypto-Filesystem liegt (mindestens) ein Verschlüsselungsalgorithmus zu Grunde. Wir wollen hier exemplarisch die Blockchiffre AES genauer betrachten.

AES (Advanced Encryption Standard)

- AES („Rijndael-Algorithmus“) ist eine symmetrische Blockchiffre
- Blockgröße 128 Bit und Schlüssellänge von 128, 192 oder 256 Bit
- die Verschlüsselung erfolgt in mehreren Runden (z.B. 10 Runden bei 128 Bit-Schlüssel) mit unterschiedlichen Schlüsseln
- wird u.a. auch bei SSH und WPA2 eingesetzt

Der Advanced Encryption Standard wurde vom US-amerikanischen National Institute of Standards and Technologie (NIST) als Verschlüsselungsstandard gewählt. Er ist Nachfolger des DES (Data Encryption Standard). Dies geschah im Jahr 2001 nach einem 5-jährigen Auswahlverfahren. Er basiert auf dem „Rijndael-Algorithmus“ der beiden Belgier John Daemen und Vincent Rijmen.

Der ursprüngliche Rijndael-Algorithmus unterstützt auch Blockgrößen von 192 und 256 Bit.

CBC - Modus (Cipher Block Chaining)

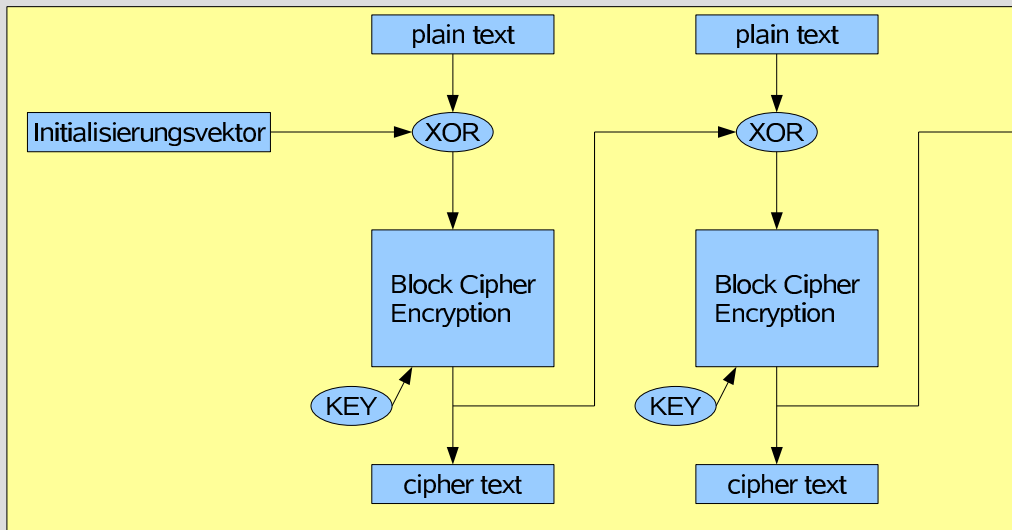
- vor dem Verschlüsseln eines Klartextblocks wird dieser mit dem vorigen Block XOR-verknüpft
- für den ersten Block wird ein Initialisierungsvektor benötigt (Zeitstempel, Pseudo-Zufallszahlenfolge)
- Non-Malleability: Crypto-Prüfsummen (MAC) können eingesetzt werden, um Manipulationen an verschlüsselten Daten zu erkennen

Im CBC-Modus kann ein Block nur mit Kenntnis des vorherigen Blocks dechiffriert werden. Haben mehrere Klartextblöcke identischen Inhalt, entstehen so dennoch verschiedene Chiffre-Texte.

Ein Problem sind Copy & Paste-Angriffe, bei denen ein Angreifer im MIB-Modell Blöcke verschiebt. An den Blockgrenzen entsteht zwar Datenmüll, der Rest wird jedoch problemlos entschlüsselt. Zur Integrationssicherung können MACs eingesetzt werden. Diese müssen allerdings bei jeder Änderung im Sektor neu berechnet werden und führen zu einem nicht unerheblichen Performance-Verlust.

Ein weiteres Problem des CBC-Algorithmus ist seine Nicht-Parallelisierbarkeit.

CBC - Modus (Cipher Block Chaining)



- Vor dem Verschlüsseln eines Klartextblocks wird dieser mit dem vorherigen XOR-verknüpft.
- Der erste Block wird mit einem Initialisierungsvektor XOR-verknüpft. Dieser sollte möglichst zufällig, also für einen Angreifer schlecht zu erraten sein.

Vorteile von CBC:

- jeder Geheimtextblock hängt von den vorherigen Klartextblöcken ab
- Klartextmuster in der Verschlüsselung nicht mehr zu erkennen
- gleiche Klartextblöcke ergeben unterschiedliche verschlüsselte Texte

CBC erhöht die Sicherheit, da Muster im Klartext verborgen werden und vom Angreifer nicht mehr erkannt werden können.

Die Wirksamkeit gegenüber Wasserzeichenangriffe hängt entscheidend von der Wahl des Initialisierungsvektors ab.

Eine unsichere (aber dennoch gebräuchliche) Variante verwendet beispielsweise die Sektornummern als Initialisierungsvektoren.

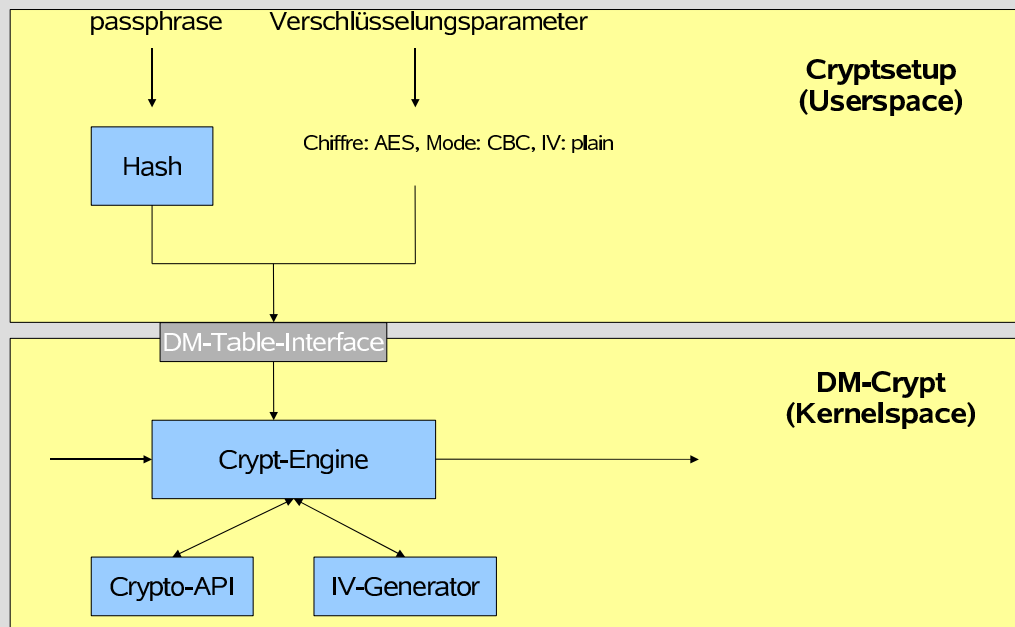
- Warum verschlüsselte Dateisysteme?
- Ansätze
- Beispiel einer Blockchiffre: AES
- **DM-Crypt**
- Truecrypt
- Enc-FS
- verschlüsseltes Brennen auf CD / DVD
- Zusammenfassung

DM-Crypt ist ein Vertreter der verschlüsselten Dateisysteme auf Block-Device-Ebene.

DM-Crypt

- seit Kernelversion 2.6.4 Bestandteil des Device-Mapper, Weiterentwicklung von Cryptoloop
- transparente Block-Device-Verschlüsselung; Daten können entweder in einer Containerdatei oder in einer eigenen Partition angelegt werden
- benötigt das Userspace-Tool Cryptsetup
- Benutzerverwaltung durch verschiedene Passwörter möglich
- verschlüsseltes Swap-Device

- Das Userspace-Tool Cryptsetup kommt heute meist in Verbindung mit dem Tool LUKS (Linux Unified Key Setup) zum Einsatz. Dieses speichert Metadaten im Header des Containers und kann mehrere Passwörter verwalten.
- Mit Hilfe einer angepassten initrd ist eine pre-boot-authentication möglich, so dass die komplette Platte mit Ausnahme des Bootloaders und der initrd verschlüsselt werden kann.
- DM-Crypt unterstützt eine Vielzahl von darunterliegenden Dateisystemen.



Auch für das Hashing des Passworts kann der User verschiedene Einstellungen vornehmen. Per Default verwendet DM-Crypt eine Iteration des RIPEMD-160-Hash ohne Salt.

Vorteile:

- gute Systemintegration
- Unterstützung verschiedener Chiffres
- verbesserte Variante von cryptsetup mit LUKS

Nachteile:

- unsichere Standardeinstellungen (kein Hash-Salt, 32-Bit Sektornummer als Initialisierungsvektor)

Im Default-Modus wird IV-plain verwendet, der die 32-Bit-Sektornummer als IV verwendet. Der sicherere ESSIV-Modus verwendet eine verschlüsselte Sektornummer, die ein Angreifer nicht mehr vorhersagen kann. Allerdings bleibt auch hier der IV für die Daten in einem Sektor konstant.

- Warum verschlüsselte Dateisysteme?
- Ansätze
- Beispiel einer Blockchiffre: AES
- DM-Crypt
- **Truecrypt**
- Enc-FS
- verschlüsseltes Brennen auf CD / DVD
- Zusammenfassung

Truecrypt ist ein Beispiel eines verschlüsselten Dateisystems, das Dateien und Partitionen in Containerdateien verschlüsselt. Ursprünglich für Windows, mittlerweile existiert eine Linux-Portierung.

True-Crypt

- ursprünglich Windows-Tool, ab Version 4.2 (04/06) auch als Linux-Portierung
- Dateien und Partitionen können in Containern verschlüsselt werden
- unterstützt verschiedene Verschlüsselungsalgorithmen (AES, Blowfish, Cast5...) und Hashalgorithmen
- unterstützt AES-Modus LWR („Tweakable Narrow-Block-Encryption“)
- Standard-Dateisystem: FAT

Truecrypt bietet auch die Möglichkeit einer Reihenschaltung der verschiedenen Verschlüsselungsalgorithmen.

Der LWR-Verschlüsselungsmodus gehört zu den justierbaren Blockchiffren. Er erwartet zusätzlich zum Schlüssel einen Wert „Justage“. Dieser wird ähnlich einem Salt nicht geheimgehalten, mit unterschiedlich Justage-Werten für jeden Block (z.B. Sektornummer und Position des Blocks innerhalb des Sektors) erhält so jeder Block einen eigenen Wert. Dies verbirgt wieder Klartextmuster und hat im Vergleich zum CBC-Modus Vorteile (z.B. Copy & Paste - Angriff).

Vorteile:

- sichere Default-Algorithmen
- gute Dokumentation unter Windows
- sorgfältige Implementierung (z.B. Überprüfung von Funktionsrückgabewerten)

Nachteil:

- schlechte Dokumentation und fehlendes GUI unter Linux

Die Überprüfung von Funktionsrückgabewerten verhindert beispielsweise, dass (versehentlich) mit einem Null-Key verschlüsselt wird und die Daten (vom Benutzer unbemerkt) unverschlüsselt geschrieben werden. Dies zeichnet Truecrypt gegenüber den meisten anderen Lösungen aus, bei denen eine solche Überprüfung fehlt.

- Warum verschlüsselte Dateisysteme?
- Ansätze
- Beispiel einer Blockchiffre: AES
- DM-Crypt
- Truecrypt
- **Enc-FS**
- verschlüsseltes Brennen auf CD / DVD
- Zusammenfassung

Wir geben Enc-FS als Beispiel eines verschlüsseltes Dateisystem im Userspace an. Ein weiteres Beispiel ist das als eher unsicher geltende Crypto-FS.

Enc-FS

- basiert auf Userspace-Dateisystem FUSE
- Verschlüsselung basiert auf Open-SSL
- Standard-Modus verwendet Blowfish-Algorithmus mit 512-Byte-Blöcken, Verschlüsselung von Dateinamen
- Paranoia-Modus: AES mit 256 Bit Blockgröße, CBC-Modus mit Prüfsummen, Dateiname fließt in den IV ein
- Windows-Portierung

Enc-FS verwendet die FUSE-Bibliotheken und Kernel-Module, die neueste Version benötigt mindestens FUSE 2.5. Während FUSE seit Version 2.6.14 zum Kernel-Standard gehört, muss Enc-FS separat erworben werden.

Vorteile:

- Verwendung von Hashed MACs zur Integrationssicherung
- Expertenmodus erlaubt genaue Konfiguration

Nachteile:

- wenig bzw. fehlerhafte Überprüfung von Funktionsrückgabewerten
- Passwortverschlüsselung ohne Salts

Enc-FS verschlüsselt auch die Dateinamen. Diese werden dann Base-64-kodiert unter Verwendung eines HMACs abgelegt.

- Warum verschlüsselte Dateisysteme?
- Ansätze
- Beispiel einer Blockchiffre: AES
- DM-Crypt
- Truecrypt
- Enc-FS
- **verschlüsseltes Brennen auf CD / DVD**
- Zusammenfassung

Ein Patch des Programms `cdrecord` erlaubt das Verschlüsseln von Daten beim Brennen.

on-the-fly encryption for cdrecord

<http://burbon04.gmxhome.de/linux/CDREncryption.html>

- Patch für das Programm cdrecord, geschrieben von Maximilian Decker
- verschlüsselt Daten direkt beim Brennen
- ermöglicht transparentes Entschlüsseln der Daten beim Lesen
- verwendet AES im CBC-Modus

Wenn sensible Daten auf CD oder DVD transportiert werden, darf die Sicherheitskette nicht durchbrochen werden. Es sind verschiedene Ansätze denkbar:

- Brennen von einzelnen beispielsweise mit GPG verschlüsselten Dateien auf CD / DVD. Nachteil: Der Empfänger muss diese zum Entschlüsseln auf einem beschreibbaren Medium speichern.
- Brennen kompletter verschlüsselter Blockdevices auf DVD. Nachteil: ineffizient aufgrund der Partitionsgrößen
- Verschlüsseln direkt beim Brennen und transparentes Entschlüsseln beim Lesen

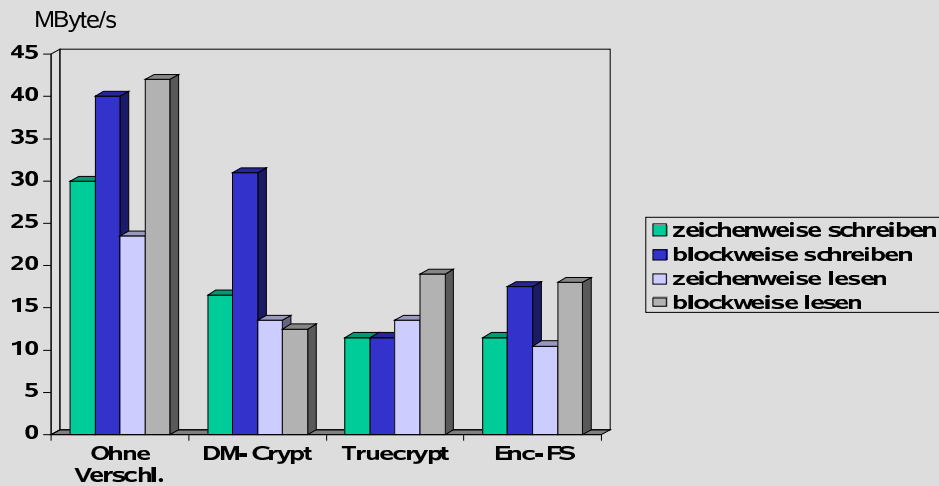
- erweitert die Funktion `write_track_data()`, die den internen Buffer auf CD / DVD brennt
- hierbei werden die Daten in 256 Bit große Pakete unterteilt und mit einem 256 Bit langen Key verschlüsselt
- diese werden zusammengefasst zu 2048-Byte-Blöcken, die auf CD geschrieben werden
- CD kann per DM-Crypt gemountet werden

Durch Angabe der entsprechenden Parameter für das gepatchte `cdrecord` kann jede beliebige Brennsoftware, die `cdrecord` verwendet, zum verschlüsselten Brennen eingesetzt werden.

Eine andere Möglichkeit zum verschlüsselten Brennen verwendet das Programm `AES-Pipe`, das beliebige IO-Puffer AES-verschlüsselt.

- Warum verschlüsselte Dateisysteme?
- Ansätze
- Beispiel einer Blockchiffre: AES
- DM-Crypt
- Truecrypt
- Enc-FS
- verschlüsseltes Brennen auf CD / DVD
- **Zusammenfassung**

Performance-Vergleich mit dem Benchmark Bonnie++



Quelle: Linux-Magazin 10/06

Der Test erfolgte auf einem IBM Thinkpad T40p mit 1,5 GByte RAM und einer Hitachi-Festplatte mit 7200 U/min unter Ubuntu 6.06 LTS.

- für Crypto-Dateisysteme existieren verschiedene Ansätze
- Sicherheit hängt entscheidend von der Wahl der Hashing- bzw. Verschlüsselungsalgorithmen und ihrer Parameter ab
- oftmals schlechte Implementierungen und Dokumentationen
- Truecrypt überzeugt unter Windows durch gute Dokumentation
- DM-Crypt bietet eine sichere Lösung mit LUKS-cryptsetup

- Leitner, A., Gutmann, P., Jansen, M. *Filesysteme verschlüsseln*, Linux-Magazin 10/06
- Wikipedia:
 - http://en.wikipedia.org/wiki/Advanced_Encryption_Standard
 - <http://en.wikipedia.org/wiki/CBC>
 - <http://www.saout.de/misc/dm-crypt>
 - <http://en.wikipedia.org/wiki/Dm-Crypt>
 - <http://www.truecrypt.org>
 - <http://arg0.net/wiki/encfs>