

# Verschlüsselung von Dateisystemen

*Transparente Verschlüsselung von Daten in verschiedenen  
Betriebssystemen  
Version 1.0 – Mai 2009*

**Stefan Hinker  
Sun Microsystems GmbH**

Verlust von Datenträgern mit sensiblen Daten kann einen erheblichen Schaden bedeuten. Verschlüsselung der Daten auf diesem Datenträger wird als wirksamer Schutz gegen den Missbrauch der so verlorenen Daten angesehen. Dieses Papier stellt einige verbreitete Lösungen für aktuelle Desktop-Betriebssysteme vor. Mögliche Angriffsziele, Schwachstellen und ihre Vermeidung werden ebenfalls betrachtet.

# Inhaltsverzeichnis

1	Einführung.....	3
2	Bedrohungsszenarien & Schutzmaßnahmen.....	5
2.1	Verschlüsselung als Schutz bei nicht laufendem System.....	6
2.2	Angriffsmöglichkeiten.....	7
3	Verschlüsselungslösungen.....	9
3.1	Allgemeines.....	9
3.2	Betriebssystem-Eigene Verschlüsselungslösungen.....	9
3.2.1	Windows.....	9
3.2.2	Mac OS X.....	12
3.2.3	Linux .....	12
3.2.4	Solaris.....	15
3.3	Systeme von Drittanbietern.....	17
3.3.1	PGP Whole Disk Encryption.....	17
3.3.2	TrueCrypt.....	17
3.3.3	FreeOTFE.....	19
3.3.4	BestCrypt.....	19
4	Fazit.....	21

# 1 Einführung

„Britisches Ministerium verliert Daten von 84.000 Häftlingen - USB-Stick enthält Namen und Geburtsdaten der Straftäter“<sup>1</sup>, „US-Bank verliert Daten von 4,5 Millionen Kunden“<sup>2</sup>, „Britisches Verteidigungsministerium verliert geheime Daten - Zwischen 2004 und 2008 sollen 121 USB-Speicher verschwunden sein“<sup>3</sup> - Immer wieder lesen wir derartige Meldungen in den Nachrichten, und immer häufiger. Problematisch ist dabei natürlich weniger der Laptop, der schlimmstenfalls einige tausend Euro gekostet hat und leicht wieder beschafft werden kann. Problematisch sind die Daten, die auf der Festplatte des Laptops gespeichert sind. Nicht immer müssen es persönliche Daten von Versicherten einer Krankenkasse sein, auch (regelwidrig) dort gespeicherte Passwörter oder sonstige Zugangscodes zu Bankkonten, Web-Accounts oder Firmennetzen sind willkommene Beute für Datendiebe, ob nun geplant kriminell oder aus Gelegenheit. Datenspeicher werden immer kleiner, und nicht nur Laptops sondern auch USB-Sticks sind beliebte Datenträger, die jedoch Dank ihrer geringen Größe auch leicht verloren gehen.

Der Schaden sollte hierbei nicht unterschätzt werden, wie die obigen Beispiele bereits deutlich machen. Die Kosten eines Datenlecks sind enorm. Im Jahr 2008 entstand deutschen Firmen im Durchschnitt Kosten in Höhe von 2,4 Millionen € pro Datenleck. Bei 28% der Vorfälle war ein gestohlener oder verlorener Laptop die Ursache<sup>4</sup>.

Darüber hinaus gibt es inzwischen immer mehr gesetzliche Regelungen, die zum Schutz von Daten verpflichten<sup>5</sup>:

- In Deutschland das Bundesdatenschutzgesetz<sup>6</sup>
- In der EU die „Data Protection Directive“
- In Japan den „Japanese Personal Information Protection Act“
- In den USA eine ganze Reihe von Gesetzen, z.B. der Federal Agency Data Breach Protection Act

1 Meldung vom 22.8.2008 auf [http://www.zdnet.de/news/wirtschaft\\_sicherheit\\_security\\_britisches\\_ministerium\\_verliert\\_daten\\_von\\_84\\_000\\_haeftlingen\\_story-39001024-39195209-1.htm](http://www.zdnet.de/news/wirtschaft_sicherheit_security_britisches_ministerium_verliert_daten_von_84_000_haeftlingen_story-39001024-39195209-1.htm)

2 Meldung vom 2.6.2008 auf [http://www.zdnet.de/news/wirtschaft\\_sicherheit\\_security\\_us\\_bank\\_verliert\\_daten\\_von\\_4\\_5\\_millionen\\_kunden\\_story-39001024-39191617-1.htm](http://www.zdnet.de/news/wirtschaft_sicherheit_security_us_bank_verliert_daten_von_4_5_millionen_kunden_story-39001024-39191617-1.htm)

3 Meldung vom 18.7.2008 auf [http://www.zdnet.de/news/wirtschaft\\_sicherheit\\_security\\_britisches\\_verteidigungsministerium\\_verliert\\_geheime\\_daten\\_story-39001024-39193670-1.htm](http://www.zdnet.de/news/wirtschaft_sicherheit_security_britisches_verteidigungsministerium_verliert_geheime_daten_story-39001024-39193670-1.htm)

4 „2008 Annual Study: Cost of a Data Breach“, Gefunden auf [http://download.pgp.com/pdfs/whitepapers/Ponemon\\_COB\\_2008\\_DE-Eng\\_090205.pdf](http://download.pgp.com/pdfs/whitepapers/Ponemon_COB_2008_DE-Eng_090205.pdf)

5 Quelle: IDC Document #210396, gefunden auf [http://www.pgp.com/insight/research\\_reports/idc\\_full\\_disk\\_encryption\\_white\\_paper.html](http://www.pgp.com/insight/research_reports/idc_full_disk_encryption_white_paper.html)

6 Zu beachten ist, dass es in Deutschland derzeit keine gesetzliche Bestimmung gibt, die die Berichtspflichten eines Unternehmens bei einem Datenleck regelt. In anderen Ländern ist dies teilweise geregelt, was zu deutlich höheren Kosten bei einem Datenleck führt.

Um diesen Gesetzen zu entsprechen und als Maßnahme zur Minimierung des Schadens eines Datenverlustes, sei er nun versehentlich oder durch Diebstahl verursacht, finden verschiedene Verschlüsselungsmethoden seit einiger Zeit immer größere Beachtung. Denn sind die Daten verschlüsselt, sind sie für einen Finder, Dieb oder Spion wertlos. Natürlich soll eine solche Verschlüsselung aber den Komfort der Geräte und den gewohnten Umgang mit ihnen nicht beeinträchtigen, da Benutzer diese sonst erfahrungsgemäß ausschalten oder umgehen. Auch der Zugriff von Anwendungen auf die Daten muss transparent erfolgen können, da natürlich nicht sämtliche bestehenden Anwendungen um eine kryptographische Komponente erweitert werden können. Die Anforderungen an eine solche Verschlüsselung lassen sich daher wie folgt knapp zusammen fassen:

- Der Datenträger muss gegen unberechtigte Zugriffe geschützt sein
- Die tägliche Arbeit mit den Daten soll möglichst unverändert möglich sein
- Die Verschlüsselung muss für Anwendungen transparent sein

## 2 Bedrohungsszenarien & Schutzmaßnahmen

Beim Schutz von Daten gegen unberechtigten Zugriff muss man zwischen verschiedenen Szenarien bzw. Ebenen unterscheiden:

- **Authentifizierung**  
Das System fordert den Benutzer auf, sich zu authentifizieren und gewährt dann auf der Grundlage von Zugriffsrechten Zugang zu bestimmten Daten. Hier ist natürlich vorausgesetzt, dass das System in Betrieb ist und der Angreifer bereits in der Lage ist, regulär mit dem System zu kommunizieren. Der Angreifer muss nun versuchen den Zugangsschutz zu umgehen, entweder unter Ausnutzung bekannter Schwächen oder durch die Verwendung entworfener Zugangscodes. Einmal auf dem System, hat er vollen Zugriff auf die Daten der gekaperten Identität. Verschlüsselung der Daten kann hier nur dann einen zusätzlichen Schutz bieten, wenn der Benutzer nach erfolgreicher Authentifizierung weitere Angaben machen muss, entweder ein Kennwort oder auch das Einstecken eines Crypto-Token, um die Daten zu entschlüsseln.
- **Schutz der Daten bei laufendem System gegen Angriffen von außen, typischerweise über das Netzwerk**  
In diesem Szenario hat der Angreifer noch keinen regulären Zugriff auf das System, kann sich also auch nicht authentifizieren. Vielmehr versucht er hier, ohne Authentifizierung Zugang zu dem System zu erlangen, im Erfolgsfall natürlich mit möglichst hohen Rechten. Auch hier ist ein in Betrieb befindliches System notwendig. Hat der Angreifer erst einmal Zugang, kann er gemäß der gekaperten System-Identität auf Daten zugreifen, ganz so als hätte er sich regelgerecht authentifiziert. Auch hier wird Verschlüsselung erst dann wirksam, wenn zusätzliche Kennworte etc. notwendig sind.
- **Schutz der Daten bei abgeschaltetem System**  
Ein abgeschaltetes System ist natürlich erst dann gefährdet, wenn ein Angreifer physischen Zugriff darauf hat. Der beste Schutz ist entsprechend, das System sorgsam zu verwahren. Hat ein Angreifer erst einmal Zugang zum System, kann er entweder versuchen, das System zu starten und es in laufendem Zustand anzugreifen, oder er kann die enthaltenen Datenträger ausbauen und in ein anderes System einbringen, über das er dann auf die Daten zugreifen kann. Hier versagen alle bisherigen Schutzmechanismen, da der Angreifer die volle Kontrolle über das System erlangt, mit dem auf die Daten zugegriffen wird. Lediglich die Verschlüsselung der Daten bietet hier Schutz. Der Angreifer muss nun versuchen, die Daten zu entschlüsseln oder, die erfolgsversprechendere Variante, die Schlüssel zu finden. Dies kann ihm immer dann besonders einfach gelingen, wenn das System erst seit wenigen Sekunden abgeschaltet war, da dann typischerweise unverschlüsselte Volume-Keys im Hauptspeicher des Systems zu finden sind. Diese bleiben auch nach dem Abschalten eines Systems für kurze Zeit erhalten und können mit der sogenannten „Cool-Boot Attacke“<sup>7</sup> gefunden werden. Gelingt diese nicht, bspw. weil das System bereits zu lange stromlos war, bleibt nur eine Cryptoanalyse

---

7 [http://en.wikipedia.org/wiki/Cold\\_boot\\_attack](http://en.wikipedia.org/wiki/Cold_boot_attack)

der Daten, die mit hohem Aufwand und geringen Erfolgsaussichten verbunden ist.

- Schutz der Daten bei „schlafendem“ System Systeme, idR. Laptops die lediglich in den Suspend-Modus gefahren wurden bevor sie in den physischen Zugriff des Angreifers gelangten, sind leicht durch die Cold-Boot Attacke anzugreifen, da der Hauptspeicherinhalt entweder noch voll von der Systembatterie betrieben wird oder durch ein kurzes „aufwecken“ des Systems wieder mit Strom versorgt werden kann. Danach sind die Bedingungen für eine solche Attacke ideal, die Erfolgsaussichten groß.
- Schutz der Daten auf „verlorenen“ Datenträgern Daten auf Datenträgern wie z.B. USB-Sticks oder gestohlene Festplatten sind ausschließlich durch Verschlüsselung zu schützen, da sie leicht an ein beliebiges, unter der Kontrolle eines Angreifers stehendes System angeschlossen werden können. Anders als vollständige Systeme besteht hier nicht das Problem, dass ein ungeschützter Masterkey im Hauptspeicher eines Systems gefunden werden könnte, da ein solches System nicht existiert. Daher ist hier die Verschlüsselung ein wirksamer Schutz, ähnlich wie bei einem System, das bereits lange Zeit stromlos war.

Insgesamt kann Verschlüsselung also in zweierlei Hinsicht Schutz für sensible Daten geben. Sie kann zusätzlich zur normalen Authentifizierung diese verstärken. Allerdings wird hierbei der Benutzer mehr oder weniger häufig mit den Schutzmaßnahmen konfrontiert, was er nur bei ausreichendem Risikobewusstsein über längere Zeit hinweg tolerieren wird. Des weiteren schützt Verschlüsselung Daten auf beweglichen Datenträgern und Systemen immer dann, wenn diese nicht in Betrieb sind.

## 2.1 Verschlüsselung als Schutz bei nicht laufendem System

Prinzipiell können Daten auf mehreren verschiedenen Ebenen verschlüsselt werden:

- Die Anwendung selbst kann die Daten verschlüsseln. Typische Anwendungen sind z.B. PGP oder moderne Email-Clients, die mittels S/MIME die Möglichkeit zur Verschlüsselung von Emails bieten. Problematisch an dieser Lösung ist jedoch die mangelnde Transparenz und Anwendungs-Unabhängigkeit – hier verschlüsselt jede Anwendung selbst, und der Benutzer wird immer wieder mit dieser Tatsache konfrontiert. Vorteil ist, dass die Daten auch dann noch verschlüsselt bleiben, wenn sie z.B. durch kopieren oder Backup den Datenträger selbst verlassen.
- Das Dateisystem verschlüsselt die Daten. Hier ist die Transparenz für die Anwendung gegeben, auch der Benutzer muss sich nur minimal umstellen indem er, wie bei den anderen Varianten auch, einmalig eine Kennung zur Freigabe der Daten eingeben oder sich sonst wie authentifizieren muss. Die Daten sind jedoch nur solange verschlüsselt, wie sie sich auf diesem Filesystem befinden. Werden einzelne Dateien kopiert, z.B. auf einen USB-Stick dessen Dateisystem nicht verschlüsselt ist, werden sie dort im Klartext gespeichert.

chert. Gleiches gilt für ein Backup, das die Dateien ebenfalls im Klartext übernimmt.

- Treiber oder Volumemanager verschlüsseln die Daten in einer Schicht zwischen Hardware und Dateisystem. Diese Variante ist genauso transparent wie die Verschlüsselung im Dateisystem. Es gibt jedoch darüber hinaus die Möglichkeit, z.B. ISO-Dateien zu verschlüsseln oder Dateisysteme in verschlüsselten logischen Volumes oder Dateien anzulegen. Ähnlich wie bei Dateisystemen gibt es die Möglichkeit, von solchen Volumes zu booten, falls die Plattform vor dem Booten bereits die Schlüsseingabe ermöglicht.
- Die Daten werden auf dem Festplatten-Controller oder im Speichersubsystem verschlüsselt. Hierbei ist die größtmögliche Transparenz gegeben, selbst Bootplatten können hier problemlos verschlüsselt werden. Schwierigkeiten gibt es jedoch bei der Eingabe des Schlüssels zur Freigabe der Daten, hier sind je nach Gerät (USB Stick, Laptop Platte, Speicher-Array) unterschiedliche Schnittstellen denkbar und notwendig. Wie auch beim Dateisystem oder verschlüsselten Volumes werden die Daten jedoch entschlüsselt, sobald sie per Kopie oder Backup das Medium verlassen.

Während die meisten Lösungen mit Volume-Verschlüsselung arbeiten, wird in der Anwendungs-Dokumentation in der Praxis häufig nicht streng zwischen Dateisystem und Datenvolume unterschieden, da diese Unterscheidung vielen Benutzern nicht klar ist. Es ist häufig einfacher und flexibler, ein verschlüsseltes Volume bereitzustellen, da darauf verschiedene Dateisysteme ohne weitere Änderung installiert werden können.

## 2.2 Angriffsmöglichkeiten

Um ohne Kenntnis der verwendeten Schlüssel Zugriff auf verschlüsselte Daten zu erlangen ist immer ein gewisser Aufwand nötig. Der Versuch, mit purer Rechengewalt den passenden Schlüssel zu finden ist selten erfolgreich und nur unter Einsatz von erheblichen Rechenressourcen überhaupt möglich. Falls die verschlüsselten Volumekeys bekannt sind ist auch hier ein entsprechender Entschlüsselungsversuch denkbar, allerdings mit nochmals erheblich geringeren Erfolgsaussichten, da hier typischerweise Einwege-Hashes oder noch stärkere Chiffren als für die eigentlichen Daten verwendet werden. Wirklich erfolgversprechend ist ein Angriff daher nur, wenn die unverschlüsselten Volumekeys auf andere Weise erlangt werden können.

Neben den natürlich ebenfalls bestehenden Möglichkeiten des Social Engineering ist hierzu in erster Linie die Cold Boot Attacke ein erfolgversprechender Angriff auf Systeme mit verschlüsselten Datenträgern. Hierbei wird die Tatsache ausgenutzt, dass der Inhalt von DRAM Speichermodulen auch einige Sekunden nach dem Verlust der Stromversorgung seinen Inhalt noch nahezu unverändert behält, bevor er langsam zerfällt. Kühlt man die Speichermodule, kann man diese Zeit auf einige Minuten ausdehnen. Der eigentliche Angriff besteht nun darin, das Zielsystem mit einem speziellen, sehr kleinen Bootimage neu zu booten, welches den Hauptspeicherinhalt auf ein Sicherungsmedium kopiert. Hier kann nun mit Analysewerkzeugen das Hauptspeicher-Abbild nach den Volumekeys durchsucht werden. Diese stehen idR. unverschlüsselt im Hauptspeicher, da sie dort ja vor dem Angriff ständig gebraucht wurden. Eine Alternative zum Booten des angegriffenen Systems ist es, die eigentlichen DRAM-Bausteine aus dem System zu entfernen, in ein zweites System einzubauen, dieses dann

entsprechend zu booten und ebenfalls die Volumekeys zu finden. Sind diese erst einmal bekannt, ist die Entschlüsselung der Daten des Zielsystems einfach möglich.

Die Cold Boot Attacke wurde von einer Forschergruppe an der Princeton University detailliert untersucht und beschrieben<sup>8</sup>. Dabei wurden viele gängige (und hier beschriebenen) Verschlüsselungslösungen angegriffen – keine einzige konnte widerstehen. Als mögliche Abhilfe wurde neben der sicheren Verwahrung der zu schützenden Systeme in vollständig abgeschaltetem Zustand auch technische Änderungen an den Systemen diskutiert, bspw. DRAM das seinen Inhalt schneller verliert oder spezielle, manipulations-sichere Hardware zur Speicherung der Schlüssel. Diese Vorkehrungen setzen jedoch Anpassungen in den verwendeten Betriebssystemen voraus und erhöhen die Hardwarekosten.

Datenträger, die nicht gemeinsam mit dem zugehörigen System in die Hände eines Angreifers fallen sind dagegen mittels Verschlüsselung gut geschützt – hier ist die Cold Boot Attacke naturgemäß nicht anwendbar.

---

8 <http://citp.princeton.edu.nyud.net/pub/coldboot.pdf>

## 3 Verschlüsselungslösungen

### 3.1 Allgemeines

Allen hier vorgestellten Lösungen gemein ist ein zweistufiges Schlüsselssystem. Die eigentlichen Daten werden idR. mit einer symmetrischen Chiffre verschlüsselt. Der dazugehörigen Schlüssel wird meist gemeinsam mit den Metadaten auf dem Datenträger gespeichert. Da dies natürlich nicht im Klartext geschehen kann, wird der Volumekey seinerseits mit einem zweiten Schlüsselverfahren verschlüsselt. Der hierfür notwendige Masterkey wird je nach Verfahren durch die Eingabe eines Benutzerpassworts erzeugt, durch ein Passwort geschützt auf einem Crypto-Token oder anderweitig getrennt von den Daten und dem Volumekey gespeichert. Während die Verwendung und Speicherung der Volumekeys sich bei den diversen Verfahren kaum unterscheiden, ist die Behandlung, Speicherung und Sicherung des Masterkeys sehr unterschiedlich.

Die grundlegende Anforderung an eine Chiffre zur Verschlüsselung von Daten auf einem Datenträger ist, dass auch relativ kleine Datenmengen, idR. ein Datenblock, unabhängig von den vorausgehenden und nachfolgenden Daten verschlüsselt, entschlüsselt, geändert und erneut verschlüsselt werden müssen. Je kleiner dabei die jeweilige Datenmenge ist, desto einfacher wird es einem Angreifer, diese Daten zu dechiffrieren. Um dem entgegenzuwirken, wird mit der Verkettung einzelner Block-Chiffren für jeden Datenblock ein neuer Schlüssel verwendet, der sich aus dem vorhergehenden, verschlüsselten Datenblock und dem Volumekey ergibt. Die erste, und immer noch weit verbreitete Implementierung hierzu ist AES-CBC. Da hierzu jedoch inzwischen einige Schwachstellen bekannt sind, wurden mit LRW und später XTS neuere Modi entwickelt, die eine höhere Sicherheit bieten<sup>9</sup>. XTS ist auch Grundlage für den Standard für Datenträgerverschlüsselung IEEE P1619<sup>10</sup>.

Im Folgenden werden neben den bei den jeweiligen Betriebssystemen mitgelieferten Verschlüsselungs-Systemen auch einige weiter verbreitete Systeme vorgestellt, die teilweise für mehrere Betriebssysteme verfügbar sind. Da die in erster Linie bedrohten Systeme neben Einzel-Datenträgern wie USB-Platten hauptsächlich Laptops sind, werden hier lediglich die Desktop-Betriebssysteme Windows, Linux, Mac OS X sowie Solaris betrachtet.

### 3.2 Betriebssystem-Eigene Verschlüsselungslösungen

#### 3.2.1 Windows

Entsprechend der großen Verbreitung von Windows in seinen diversen Versionen gibt es eine Vielzahl von Zusatzsoftware, die die unterschiedlichsten Lösungen zur Verschlüsselung anbieten. Das Betriebssystem selbst bietet seit Windows XP bzw. Windows Server 2000 eine eigene Dateisystem-Verschlüsselung unter dem Namen „Encrypted Filesystem“ (EFS) an. Zusätzlich bietet Windows Vista mit „Bit Locker“ die

<sup>9</sup> [http://en.wikipedia.org/wiki/Disk\\_encryption\\_theory](http://en.wikipedia.org/wiki/Disk_encryption_theory)

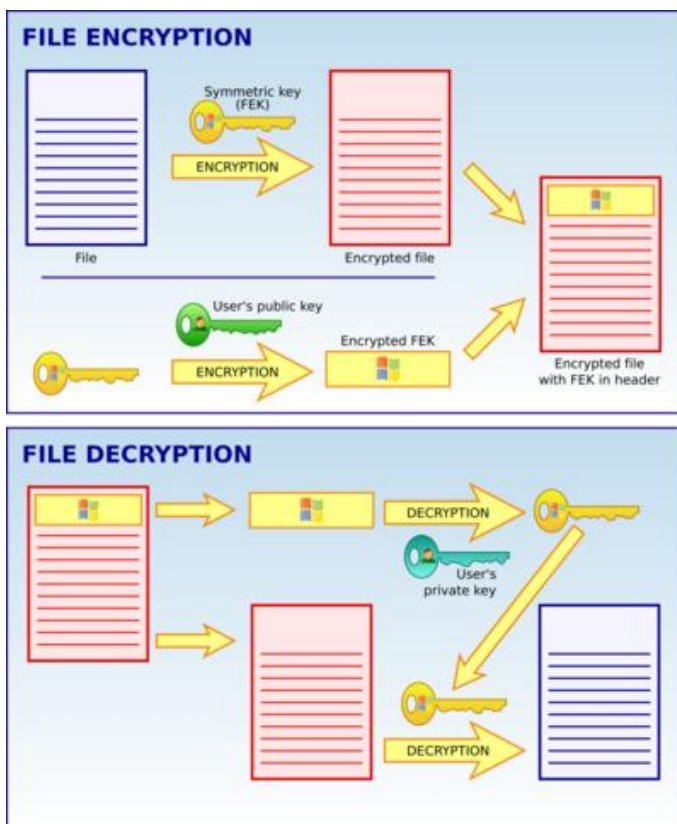
<sup>10</sup> [http://en.wikipedia.org/wiki/IEEE\\_P1619](http://en.wikipedia.org/wiki/IEEE_P1619)

Möglichkeit, die Bootplatte des Systems einschl. aller Betriebssystem- und Benutzerdaten zu verschlüsseln. Bit Locker ist damit keine Dateisystemverschlüsselung im eigentlichen Sinne. Im folgenden werden neben EFS und Bit Locker auch einige andere, weit verbreitete Lösungen beschrieben.

## Windows EFS

EFS ist ein Feature von NTFS. Eingeführt in Windows 2000, bietet es die Möglichkeit, Dateien oder Ordner-Hierarchien gezielt zu verschlüsseln. Durch die Verwendung mehrerer öffentlicher Schlüssel wird auch ein Zugriff mehrerer Benutzer auf eine so verschlüsselte Datei möglich.

Zur Verschlüsselung der Dateien wird, je nach Betriebssystem-Version verschieden, ein symmetrischer Algorithmus verwendet – seit Windows XP SP1 AES. Der hierfür verwendete symmetrische Schlüssel wird seinerseits mit einem asymmetrischen Verfahren (RSA) verschlüsselt und im Header der jeweiligen Datei gespeichert. Durch dieses zweistufige Verfahren ist – durch Verwendung mehrerer öffentlicher Schlüssel – die gemeinsame Nutzung durch verschiedene Benutzer möglich. Zuletzt wird der private Schlüssel durch das Passwort des jeweiligen Benutzers geschützt<sup>11</sup>. Somit ist das gesamte Verfahren genauso sicher wie das Passwort des Benutzers. Ist dieses kompromittiert, z.B. mit Hilfe eines der vielen frei verfügbaren Tools geknackt, sind die Daten nicht weiter geschützt. Wird das Passwort verloren und durch einen Administrator zurückgesetzt, ist ein Zugriff auf die Daten nicht mehr möglich, da der private RSA Schlüssel des Benutzers nicht mehr verfügbar ist. Insbesondere ist es auch notwendig, kein automatisches Login des Benutzers zu verwenden, da sonst das Benutzerpasswort im Klartext in der Registry gespeichert wird – und darüber hinaus natürlich jeder, auch unbefugte Benutzer den Computer verwenden kann und die Daten lesen.



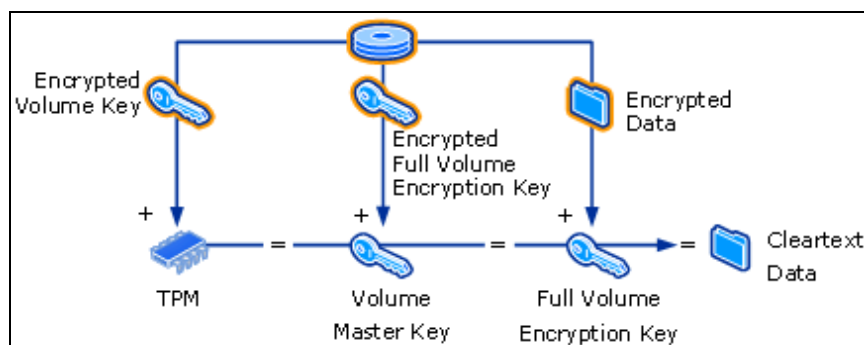
<sup>11</sup> Illustration: [http://en.wikipedia.org/wiki/Encrypting\\_File\\_System](http://en.wikipedia.org/wiki/Encrypting_File_System)

Der private Schlüssel des Benutzers wird mit dem Benutzerpasswort verschlüsselt auf dem lokalen System abgelegt. Eine Kopie wird, je nach Umgebung, auf der „Password Reset Disk“ oder im Active Directory gespeichert. Weiteres Keymanagement existiert nicht.

EFS ist sehr einfach zu benutzen. Es schützt Dateien effektiv gegen Zugriffe unberechtigter Benutzer eines laufenden Systems, dessen prinzipielle Authentifizierungsmechanismen nicht kompromittiert wurden und ist daher als Erweiterung der normalen Dateisystem-Zugriffsrechte wirksam. Der Schutz der Daten vor unberechtigtem Zugriff im Falle von Hardware-Verlust oder -Diebstahl ist jedoch gering, da das System selbst ungeschützt bleibt und die verwendeten Schlüssel relativ leicht über das Knacken der Benutzerpasswörter zu beschaffen sind.

## Windows Bit Locker

Bit Locker<sup>12</sup> wurde mit Windows Vista bzw. Windows 2008 eingeführt. Bit Locker verschlüsselt vollständige Volumes und schützt damit sowohl die Daten des Volumes als auch das auf dem Volume gespeicherte Betriebssystem selbst vor unbefugter Benutzung. Damit ist Bit Locker kein verschlüsselndes Dateisystem im eigentlichen Sinne, erweitert jedoch den Schutz durch EFS effektiv, indem nun zusätzlich auch das gesamte System gegen unbefugtes Benutzen gesichert wird.



Bit Locker arbeitet ebenfalls mit einem symmetrischen Schlüssel, mit dem das Volume verschlüsselt wird. Dieser „Volume Key“ wird mittels RSA verschlüsselt und idR. im TPM-Modul<sup>13</sup> (oder alternativ auf einem USB-Stick) gespeichert. Das System verwendet eine nicht-verschlüsselte Boot-Umgebung um das Betriebssystem zu starten, und eine zweite, verschlüsselte für das eigentliche Betriebssystem („C:“). Die Integrität der Bootumgebung wird dabei beim Start des Systems geprüft. Seit Vista SP1 ist es auch möglich, weitere Volumes zu verschlüsseln.

Während es bei EFS sehr schnell und einfach ist, einzelne Dateien oder Ordner zu verschlüsseln, muss bei Bit Locker ein gewisser Aufwand getrieben werden. Die zusätzliche Boot-Umgebung, ca. 1.5GB, muss erzeugt werden und das gesamte Volume, idR. „C:“, muss verschlüsselt werden. Dies kann je nach Volumengröße erhebliche Zeit in Anspruch nehmen und verlangt vom Benutzer deutlich mehr als bei EFS, wo das ein-

<sup>12</sup> Quelle: <http://technet.microsoft.com/en-us/library/cc732774.aspx>

<sup>13</sup> TPM: Trusted Platform Module – eine in vielen neueren Laptops und PCs verbaute Hardware zur sicheren Speicherung von kryptographischen Schlüsseln. Siehe auch [http://en.wikipedia.org/wiki/Trusted\\_Platform\\_Module](http://en.wikipedia.org/wiki/Trusted_Platform_Module)

fach Auswählen einer Option in den Eigenschaften des Ordners ausreicht. Einmal initialisiert, arbeitet Bit Locker jedoch für den Benutzer vollkommen transparent.

Der Volume Master Key von Bit Locker wird bei abgeschaltetem System idR. Im TPM-Modul gespeichert und idealer Weise durch ein Benutzerpasswort und eventuell zusätzlich einen USB-Token beim Einschalten des Systems freigegeben. Er ist damit recht effektiv geschützt, solange das System abgeschaltet ist. Bei laufendem System liegt dieser Schlüssel jedoch im Hauptspeicher, wo er auch im Standby- oder Sleep-Modus verbleibt. Ein erfolgreich ausgeführter Angriff, um diesen Schlüssel auszulesen ist daher die bereits beschriebene „Cold Boot Attacke“, bei dem bspw. ein laufender oder „schlafender“ Laptop hart resettet und sofort von einer entsprechend präparierten CD neu gebootet wird.

### 3.2.2 Mac OS X

Apple liefert mit Mac OS X seit der Version 10.3 das Feature „FileVault“ zur Verschlüsselung von Benutzerdaten aus. FileVault erzeugt aus dem Homedirectory des Benutzers ein logisches, verschlüsseltes Volume und mountet dieses statt des Originals. Das Volume wird mit AES-128 verschlüsselt, der Schlüssel mit dem Passwort des Benutzers geschützt. Im Mehrbenutzerbetrieb ist ein Zugriff anderer Benutzer auf diese Daten nun unmöglich. Ordner, die für den Netzwerkzugriff freigegeben sind, sind nur solange verfügbar wie der Benutzer angemeldet ist.

Das von FileVault erzeugte logische Volume liegt dabei als normale Datei im Dateisystem des Rechners, wovon der Benutzer allerdings bei seiner täglichen Arbeit nichts merkt. Beeinträchtigt sind jedoch Tools wie Apples TimeMachine sowie Backup Tools, da diese im Dateisystem arbeiten, und nicht mit den Dateien innerhalb des FileVault Volumes.

Der zur Verschlüsselung verwendete Schlüssel wird auf dem lokalen System abgelegt und mit dem Benutzerpasswort geschützt. Zusätzlich wird ein Master Passwort angelegt, das als zweiter Zugang zum FileVault-Schlüssel dient. Eine Schlüsselverwaltung außerhalb des lokalen Systems gibt es nicht. Die Einrichtung von FileVault setzt Administrationsrechte sowie ausreichend temporären Plattenplatz (100% der zu verschlüsselnden Daten) voraus.

Zusätzlich zu FileVault gibt es die Möglichkeit, mit Hilfe des „Disk Utility“ weitere verschlüsselte Volumes beliebiger Größe anzulegen. Diese haben die gleichen Eigenschaften wie das FileVault-Volume, lassen sich jedoch ähnlich wie externe Laufwerke beliebig an- und abhängen und bspw. auch auf CD/DVD brennen.

### 3.2.3 Linux

Unter Linux gab es in den vergangenen Jahren eine Vielzahl von Ansätzen zur Verschlüsselung von Volumes. Verschlüsselnde Dateisysteme wie bspw. EFS gibt es keine. Durchgesetzt hat sich das Kernel-Modul dm-crypt in Verbindung mit LUKS, dem Linux Unified Key Setup. Dieses wird inzwischen auch von einigen Distributionen zur Verschlüsselung der Bootpartitionen eingesetzt, so dass hier ebenfalls ein Schutz des gesamten Systems möglich ist.

## dm-crypt & LUKS

dm-crypt<sup>14</sup> ist ein Modul des Linux Device Mappers und kann als solches eingesetzt werden, um Block Devices aller Art zu verschlüsseln. Als Anwendung ist damit bspw. ein Ext3-Dateisystem in einer durch dm-crypt verschlüsselten Datei genauso denkbar wie ein FAT32-Dateisystem auf einem LVM-RAID, dessen Volumes durch dm-crypt verschlüsselt werden. dm-crypt entstand als Nachfolger von cryptoloop, das verschlüsselte Loopback-Mounts ermöglichte. Zusätzlich zur deutlich erweiterten Funktionalität bei dm-crypt wurden auch Schwächen in der Verschlüsselung selbst behoben. Damit schützt es zwar besser gegen sogenannte „Watermarking“-Angriffe<sup>15</sup>, bleibt jedoch wie alle anderen Verfahren auch durch Cold Boot Attacks angreifbar. Als Kryptographie-Backend verwendet dm-crypt die CryptoAPI des Linux Kernel. Damit stehen eine ganze Reihe der gebräuchlichsten Algorithmen zur Verfügung, ohne weitere Einstellung wird jedoch weiterhin ein CBC-Algorithmus verwendet. Damit sind solche Volumes zwar rückwärts-kompatibel mit cryptoloop, aber ebenso angreifbar.

dm-crypt ist ein reines Kernel-Modul ohne eigene Benutzerschnittstelle. Erst die Kombination mit weiteren Tools macht es für den Anwender benutzbar. Aktuelle Distributionen wie z.B. Suse Linux liefern dm-crypt in Kombination mit Kommandozeilen- oder graphischen Tools aus, um einzelne Festplatten, Volumes oder auch Bootlaufwerke zu verschlüsseln. Das Softwarepaket selbst wird idR. mit den Tools „cryptosetup“ und „cryptomount“ verwendet. Beides sind Kommandozeilen-Tools, die selbst noch verhältnismäßig viel Expertenwissen voraussetzen. Einmal konfiguriert, ist jedoch ein auch für unerfahrene Benutzer transparenter Betrieb der verschlüsselten Volumes möglich. Für die Verschlüsselung der Boot-Volumes wird das unter Linux gebräuchliche Bootverfahren der initrd verwendet. Dabei wird die gesamte Bootplatte verschlüsselt, mit Ausnahme der kleinen Boot-Partition und eben der initrd, die den Inhalt der RAM-Disk enthält, die initial gebootet wird. Dort wird nach dem Master-Passwort gefragt, mit welchem sodann die eigentliche Bootplatte entschlüsselt und gemountet wird.

LUKS übernimmt dabei die Rolle der Schlüsselverwaltung. Anders als bei anderen Systemen speichert dm-crypt keine Metadaten wie z.B. symmetrische Schlüssel gemeinsam mit den Daten in den verschlüsselten Volumes. LUKS erweitert daher die Volumes um ein standardisiertes On-Disk-Format, in dem dann auch diese Schlüssel abgelegt sind. Es ermöglicht damit u.A. auch die Änderung von Master-Passworten und das Zurückziehen bzw. Ändern der Schlüssel bei vermuteter Kompromittierung des Master-Passworts. LUKS kann angewiesen werden, die Schlüssel bspw. auf USB-Sticks oder anderen externen Medien zu speichern. Dort werden sie dann wiederum verschlüsselt abgelegt. Eine über diese Möglichkeit hinausgehende Verwaltung der Schlüssel findet jedoch in diesem System nicht statt. Damit ist zwar ein umfassender Schutz eines einzelnen Systems möglich, ein unternehmensweiter Einsatz erfordert jedoch weitergehende Infrastruktur, die zusätzlich entwickelt werden müsste.

---

14 <http://www.saout.de/misc/dm-crypt/>

15 [http://en.wikipedia.org/wiki/Watermarking\\_attack](http://en.wikipedia.org/wiki/Watermarking_attack)

## EncFS

Anders als dm-crypt arbeitet EncFS<sup>16</sup> völlig im Userspace und benötigt keine Root-Rechte. Es verwendet dabei das Framework von FUSE<sup>17</sup> – dem Userspace Filesystem von Linux. Es ermöglicht damit einem einfachen Benutzer, durch das Anlegen zweier Verzeichnisse und dem Aufruf eines Kommandos, in diesem so definierten verschlüsselten Verzeichnis Daten sicher zu speichern. Die einzelnen Dateien werden je nach Bedarf angelegt, das gesamte Dateisystem benötigt genau soviel Platz wie die Summe der dort gespeicherten Dateien. Auch Backup-Tools können diese Dateien ganz normal sichern, die Dateien und auch die Dateinamen bleiben dabei verschlüsselt. Der Nachteil dieser Methode ist, dass die Existenz der einzelnen Dateien sowie deren Meta-Daten wie Zugriffsrechte und -zeiten sichtbar sind. Der Vorteil ist, dass mit EncFS verschlüsselte Verzeichnisse auf Systemen angelegt werden können, auf denen Root-Zugriff nicht möglich ist, und auf denen man den Storage-Administratoren keinen Zugriff auf die Daten ermöglichen will. Ein Beispiel für solch einen Fall sind Root-Server eines Hosting-Anbieters oder ein nicht vollständig Vertrauenswürdiger NFS-Server.

EncFS speichert Verwaltungsdaten und die symmetrischen Schlüssel in einem versteckten Unterverzeichnis „.encfs5“ ab. Die Schlüssel werden dabei durch ein

Passwort geschützt (damit verschlüsselt), das beim ersten Zugriff auf die Daten anzugeben ist. Die Sicherung des Passworts bleibt dem Benutzer überlassen. Der Vorteil dieser Methode ist, dass eine Änderung des Passworts leicht möglich ist, ohne die eigentlichen Daten neu verschlüsseln zu müssen. Eine Änderung des Volume-Keys ist damit jedoch, wie in den meisten anderen Implementierungen auch, nicht möglich.

EncFS verwendet die Kryptographie-Module von OpenSSL. Als Chiffren kommen AES oder Blowfish mit Schlüssellängen von 128 Bit bis 256 Bit zum Einsatz.

## Loop-AES

Wie der Name schon sagt, ist Loop-AES ein weiterer Loopdevice-Treiber. Anders als dm-crypt ist Loop-AES ein vom Kernel-Release unabhängiges Kernel-Modul. Als solches muss es jedoch mit Root-Rechten installiert bzw. geladen werden. Die mit Loop-AES verwalteten Volumes müssen per „mount“-Befehl gemountet werden, was idR.

Ebenfalls Rootrechte voraussetzt.

Loop-AES verwendet die in üblichen Linux-Installationen bereits vorhandene Loopback-Infrastruktur. Ein Device „/dev/loop<n>“ wird dabei einem Speicher-Backend, ent-

weder eine Partition oder eine Datei, zugeordnet, der Datenverkehr über dieses Device dann verschlüsselt. Die Implementierung verwendet dabei zusätzlich zur Verschlüsselung keine Prüfsummen wie z.B. SHA1, so dass die Integrität der Daten nicht gewährleistet wird. Die Daten selbst werden mit AES im CBC-Modus verschlüsselt. Dabei kommen bis zu 65 verschiedene Schlüssel zum Einsatz. Diese Schlüssel werden mit Hilfe von GPG Keyfiles gesichert. Die Verwaltung dieser Keyfiles bleibt dem Benutzer überlassen.

---

<sup>16</sup> <http://www.argo.net/encfs>

<sup>17</sup> <http://fuse.sourceforge.net/>

Die Verwendung von GPG Keyfiles in Kombination mit der in Linux vorhandenen Möglichkeit, einem Nicht-Root User das Mounten von Dateisystemen zu erlauben macht es möglich, dass mehrere Benutzer ein verschlüsseltes Verzeichnis gemeinsam benutzen. Dazu wird der Master-Key des Volumes mit allen privaten Schlüsseln der diversen Benutzer verschlüsselt, so dass jeder mit seinem eigenen Passwort Zugriff auf den Master-Key hat. Mit diesem ist dann ein Zugriff auf die Daten möglich.

In der Dokumentation<sup>18</sup> zu Loop-AES wird auch ein Verfahren beschrieben, um das Root-Verzeichnis einer Linux-Installation zu verschlüsseln. Dazu wird eine kleine, unverschlüsselte Boot-Partition benötigt, das Verfahren ist dem von dm-crypt vergleichbar.

### 3.2.4 Solaris

Unter Solaris ist bis einschließlich Solaris 10 kein System zur Verschlüsselung von Logical Volumes oder Dateisystemen verfügbar. Erst für OpenSolaris wird an Modulen zur Verschlüsselung von ZFS Datasets bzw. Loopback Volumes (lofi) gearbeitet. Dieser vordergründige Mangel in bisherigen Versionen von Solaris relativiert sich ein wenig wenn man bedenkt, dass Dateisystem-Verschlüsselung in erster Linie gegen Dateneinsicht bei Verlust der Datenträger schützen soll, die meisten Solaris-Systeme jedoch in gut gesicherten Rechenzentren stehen. Erst OpenSolaris ist ein auch auf Laptops und ähnlichen Systemen weiter verbreitetes System, so dass hier auch der Bedarf für eine solche Verschlüsselung akuter ist.

### Verschlüsselung mit lofi

In SNV Build 105 wurde der lofi-Treiber um eine Komponente zur Verschlüsselung erweitert<sup>19</sup>. Bisher konnten mit diesem Treiber Dateien als Block Device gemountet und ein in diesen Dateien enthaltenes Dateisystem gemountet werden. Mit der Erweiterung ist es nun möglich, den Inhalt dieser Dateien mit einem CBC-Algorithmus zu verschlüsseln. Damit stellt der lofi-Treiber ähnliche Möglichkeiten zur Verfügung wie dm-crypt unter Linux, ohne allerdings das bereits recht ausgereifte Framework dazu ebenfalls anzubieten. In dieser ersten Implementierung wurde die etwas schwächere CBC-Variante von AES verwendet. Sobald in einer späteren Version des Solaris Cryptographic Framework die aktuelleren XTS-Varianten verfügbar sind, werden diese verwendet werden.

Das zur Verwendung der verschlüsselten Volumes notwendige Passwort kann entweder direkt an der Kommandozeile eingegeben werden oder aber durch das Solaris Cryptographic Framework über dessen PKCS#11-Schnittstelle bereitgestellt werden. Damit sind sämtliche Management-Funktionen, die dieses Framework zur Verfügung stellt auch für lofi-verschlüsselte Volumes nutzbar. Ein Schlüssel kann also bspw. auf einem Crypto-Token wie der FIPS-140-2-zertifizierten SCA6000-Karte und damit auch in einem LDAP-Verzeichnis gespeichert werden.

---

18 <http://loop-aes.sourceforge.net/loop-AES.README>

19 [http://blogs.sun.com/darren/entry/opensolaris\\_disk\\_encryption\\_in\\_snv](http://blogs.sun.com/darren/entry/opensolaris_disk_encryption_in_snv)

## ZFS

ZFS ist ein sehr junges Dateisystem. Daher ist sein Aufbau weniger monolithisch als der älterer Dateisysteme wie UFS oder FAT. Als einziges seiner Art vereinigt ZFS die Funktionen des Dateisystems mit denen des Volumemanagers. Entsprechend sind erweiterte Funktionen wie Datenkompression oder Verschlüsselung verhältnismäßig einfach integrierbar. Datenkompression ist als Modul bereits seit längerem vorhanden und kann für jedes Dateisystem eines Datenpools einzeln jederzeit an- bzw. abgeschaltet werden. Die Kompression ist dabei für den Anwender völlig transparent.

Analog zur Kompression wird derzeit an einem Modul zur Verschlüsselung gearbeitet<sup>20</sup>. Im Unterschied zur Kompression werden dabei nicht einzelne Blöcke verschlüsselt, sondern alle Daten eines Dateisystems bzw. Datasets. Auch kann die Verschlüsselung nur einmalig beim Anlegen des Dateisystems eingeschaltet werden, ein Dateisystem ist also entweder immer und vollständig verschlüsselt, oder nicht. Die Schlüssel werden dabei gemeinsam mit dem Dateisystem angelegt und zusammen mit den Daten verschlüsselt gespeichert. Ein Masterkey zu diesem Schlüssel wird mit den Mitteln des Solaris Cryptographic Framework gesichert, also entweder in dessen Software Token oder in einem per PKCS#11 zugänglichen Hardware-Token wie der SCA6000. Beim Mounten des Dateisystems wird der Administrator jeweils nach einer Authentifizierung gefragt um diesen Schlüssel freizugeben. Danach kann das einzelnen Dateisystem wie gewohnt gemountet und verwendet werden. Dadurch, dass jedes Filesystem einen eigenen Schlüssel verwendet, sind z.B. je nach geforderter Sicherheit der Daten unterschiedliche Algorithmen pro Dateisystem möglich. Auch kann so ein sicheres Löschen der Daten eines Dateisystems einfach dadurch erreicht werden, dass der Schlüssel vernichtet wird. Schlüsseleigenschaften werden dabei wie sonstige ZFS-Eigenschaften bei der Erzeugung von Sub-Dateisystemen oder Clones vererbt. Dabei kann optional ein Clone einen neuen Schlüssel verwenden, um unabhängige Clone-Bäume zu erhalten.

Neben Dateisystemen können auf diese Weise natürlich auch ZVOLS verschlüsselt werden. Damit ist auch ein Schutz des Swap-Devices möglich, hierfür ist auch die Verwendung von Einmalschlüsseln vorgesehen. In einer ersten Phase ist lediglich der Schutz „normaler“ Datasets geplant. Eine Erweiterung auf Root-Dateisysteme ist in Phase 4 vorgesehen. Der erste Prototyp verwendet als Chiffre AES im CBC-Modus. Die endgültige Implementierung wird CCM, GCM oder eventuell XTS verwenden. Weitere geplante Features, die im ersten Release nicht enthalten sein werden sind:

- Remote Key Management
- Delegation eines verschlüsselten Dateisystems an eine Zone bzw. einen Benutzer, ohne den dazugehörigen Schlüssel preisgeben zu müssen. Diese Funktion setzt die allgemeine ZFS-Delegation an Benutzer und Zonen voraus, welche teilweise noch nicht verfügbar ist.
- Sicheres Löschen
- Hinterlegen bzw. Wiederherstellen von Schlüsseln
- Verschlüsseltes HA-ZFS

---

<sup>20</sup> <http://opensolaris.org/os/project/zfs-crypto/>

Die Entwicklung ist im Wesentlichen abgeschlossen, es bestehen jedoch noch Abhängigkeiten zu weiteren ZFS Projekten. Sobald diese abgeschlossen sind, wird Verschlüsselung mit ZFS, vermutlich Ende 2009 verfügbar sein.

## 3.3 Systeme von Drittanbietern

### 3.3.1 PGP Whole Disk Encryption

PGP bietet ein vollständiges Portfolio an Verschlüsselungslösungen für Desktops, Blackberries, Laptops und Server an, sowohl für Einzelgeräte als auch im Unternehmenseinsatz. PGP Whole Disk Encryption<sup>21</sup> ist das Produkt zur Verschlüsselung von Platten in Servern, Workstations und Laptops. Es ist in die „PGP Encryption Platform“<sup>22</sup> integriert, die einen unternehmensweiten Einsatz durch die Verwaltung von Schlüsseln und Policies erleichtert.

PGP WDE selbst arbeitet ähnlich wie Bit Locker und verschlüsselt Partitionen bzw. ganze Festplatten sowie USB-Sticks und ähnliche Geräte. Auch PGP WDE unterstützt ggf. ein TPM zur lokalen Speicherung von Schlüsseln. Ebenso ist auch eine Zwei-Faktor Authentifizierung mit diversen Smartcards oder USB-Tokens möglich. PGP WDE stellt eine Pre-Boot Umgebung zur Verfügung, auch hier wird daher die gesamte Arbeitsumgebung einschließlich aller Benutzerdaten geschützt. Andere, begleitende Produkte von PGP ermöglichen bspw. die Verschlüsselung von Email, Netzwerk-Laufwerken oder Dateien auf einem Server. PGP WDE ist für aktuelle Versionen von Windows sowie für die Intel Versionen von Mac OS X (10.4 und 10.5) verfügbar. Es richtet sich durch seine Integration in die PGP Encryption Platform und den dort vorgesehenen Policy Server in erster Linie an den Einsatz in Unternehmen, die eine größere Menge von Laptops etc. schützen müssen.

PGP WDE ist das einzige Produkt für Mac OS X, das die gesamte Systemplatte verschlüsselt und bereits vor dem Booten des Systems nach einer Authentifizierung fragt. Anders als bei Windows werden hier leider bisher keine SmartCards oder USB-Tokens unterstützt.

### 3.3.2 TrueCrypt

TrueCrypt<sup>23</sup> ist eine für Windows, Linux und Mac OS verfügbare Verschlüsselungs-Software, deren besonderes Augenmerk auf „unsichtbarer“ Verschlüsselung liegt. An einer Portierung für OpenSolaris wird derzeit in einem privaten Projekt gearbeitet<sup>24</sup>. TrueCrypt verschlüsselt ganze Partitionen oder auch USB-Devices, bietet aber auch die Möglichkeit, logische Volumes in hierfür angelegten Dateien anzulegen und deren Inhalt zu verschlüsseln. In solchen verschlüsselten Volumes können dann Dateisysteme nach Wahl (FAT, NTFS, Linux- und Mac OS Dateisysteme) angelegt werden. Auch Betriebssysteme können in solchen Volumes installiert werden, die dann durch den TrueCrypt Bootloader angesprochen und entsprechend geschützt werden. TrueCrypt ist da-

---

21 [http://www.pgp.com/insight/white\\_papers/howencryptionworks.html](http://www.pgp.com/insight/white_papers/howencryptionworks.html)

22 [http://www.pgp.com/insight/white\\_papers/encryptionplatform.html](http://www.pgp.com/insight/white_papers/encryptionplatform.html)

23 <http://www.truecrypt.org/>

24 [http://blogs.sun.com/mbp/entry/truecrypt\\_on\\_opensolaris](http://blogs.sun.com/mbp/entry/truecrypt_on_opensolaris)

mit, wie Bit Locker, kein verschlüsselndes Dateisystem sondern eine Volume-Verschlüsselung.

TrueCrypt bietet vielfältige Möglichkeiten zum Speichern von Schlüsseln. Neben Interaktiven Passwörtern und PKCS#11-Devices wie SmartCards sind dies vor allem sogenannte Keyfiles. Diese können beim Anlegen eines Volumes erzeugt werden. Eleganter ist es jedoch, bereits vorhandene, idR. unverdächtige Dateien, deren Inhalt sich nicht ändern wird, als Keyfile zu verwenden. TrueCrypt verwendet dann den Inhalt dieser Dateien als Schlüssel, ohne dass aus dem Vorhandensein der Dateien bereits auf eine Verschlüsselung geschlossen werden könnte. Schutz für das Volume entsteht durch das Entfernen der Keyfiles, z.B. indem diese auf einem vom Rechner separat aufbewahrten USB-Stick gespeichert werden. Zusätzlich kann die Eingabe einer PIN verlangt werden. Ein über die Verwendung von PKCS#11 hinausgehendes Keymanagement wird dagegen nicht unterstützt.

Die Verwendung von Keyfiles als Schlüssel ist ein Teil der Methode, mit der das Vorhandensein von verschlüsselten Daten versteckt werden soll. TrueCrypt Volumes werden beim Anlegen mit zufälligen Daten initialisiert, so dass insbesondere die Dateien, deren Inhalt ein TrueCrypt Volume ist, sich nicht nachweislich mit TrueCrypt in Verbindung bringen lassen. Als Volume mounten lassen sie sich erst in Verbindung mit den entsprechenden Keyfiles, somit fällt der Nachweis, dass eine Datei in Wirklichkeit ein TrueCrypt Volume ist, zumindest sehr schwer.

Weitere Features zur Verheimlichung der Verschlüsselung sind versteckte Volumes und versteckte Betriebssysteme. Versteckte Volumes entstehen dadurch, dass innerhalb eines TrueCrypt Volumes mittels eines zweiten Schlüssels ein zweites Volume angelegt wird. Je nachdem, welchen der beiden Schlüssel man beim Mounten des Volumes verwendet, wird das „offene“ oder das „versteckte“ Volume gemountet. Wird das offene Volume verwendet, ist die Existenz des versteckten Volumes nicht erkennbar. Installiert man nun ein Betriebssystem in diese beiden Volumes und verwendet den TrueCrypt Bootloader zum Booten des Systems, kann man durch Wahl des Passworts das jeweilige Betriebssystem auswählen. Es ist dann zwar bekannt, dass TrueCrypt auf dem System verwendet wird, die Existenz des versteckten Betriebssystems ist jedoch nicht nachzuweisen.

Die erreichbare Sicherheit der von TrueCrypt verwendeten Schlüssel ist dank der vielfältigen Möglichkeiten der Speicherung z.B. in SmartCards mit PIN-Eingabe im Smart-Card-Leser recht hoch. Auch TrueCrypt muss jedoch im Betrieb die entsprechenden Masterkeys im Hauptspeicher des Rechners halten. Dort sind sie daher kaum gegen Kompromittierung z.B. via Swap-Files, Dump-Files oder Cold Boot Angriffe zu schützen. Der zusätzliche Aspekt der Verschleierung kann die Sicherheit dieses Systems weiter erhöhen.

TrueCrypt schützt, ähnlich wie BitLocker, in erster Linie gegen unberechtigten Zugriff auf Daten bei Verlust der Datenträger. In einem Multi-User System haben alle angemeldeten Benutzer Zugriff auf ein gemountetes TrueCrypt-Volume, vorausgesetzt die Zugriffsrechte des Betriebssystems erlauben dies.

### 3.3.3 FreeOTFE

OTFE<sup>25</sup> steht für „On the fly Encryption“, das Softwarepaket für Windows verschlüsselt Partitionen oder Dateien und stellt diese als logische Laufwerk zur Verfügung. Im normalen Betrieb erfordert es Administrator-Rechte zur Installation der notwendigen Treiber. Als Besonderheit bietet es die Möglichkeit, alle zum Betrieb eines Volumes notwendigen Daten, also auch die geschützten Schlüssel, bspw. auf einem USB-Stick von System zu System zu transportieren. Des Weiteren ist es kompatibel mit den unter Linux verbreiteten On-Disk Formaten von dm-crypt bzw. LUKS und kann, ein kompatibles Dateisystem (idR. FAT32) vorausgesetzt, solche Daten unter Windows lesen und schreiben. Damit bietet es sich insbesondere dort zur Verwendung an, wo verschlüsselte Daten von beiden Betriebssystemen aus verwendet werden müssen. Für den vollständig treiberlosen Betrieb gibt es als Zusatzsoftware das Paket „FreeOTFE Explorer“, einen explorer-ähnlichen Dateimanager, der die verschlüsselten Volumes lesen kann. Mit Hilfe dieses Tools ist auch ein Zugriff ohne Administrationsrechte möglich.

FreeOTFE bietet eine Vielzahl von Chiffren an, insb. auch AES im modernen XTS-256 Modus. Zur Sicherung der Volume-Keys kann bei Bedarf auch Zwei-Faktor Authentisierung bspw. mittels Crypto-Tokens zum Einsatz kommen.

### 3.3.4 BestCrypt

BestCrypt<sup>26</sup> von Jetico<sup>27</sup> bietet eine kleine Suite von Anwendungen für Windows und Linux, um logische Volumes oder Partitionen zu verschlüsseln. Wie die meisten Produkte dieser Art werden die Volume-Schlüssel dabei mit einem Passwort verschlüsselt und gemeinsam mit den Dateien auf dem Volume gespeichert. Beim öffnen des Volumes wird das dafür notwendige Passwort vom Benutzer erfragt. BestCrypt bietet neben AES auch Twofisch, Blowfish und andere Chiffren an und verwendet dabei die jeweils größte verfügbare Schlüssellänge.

BestCrypt kann unter Windows neben logischen Volumes, die als Datei im Dateisystem liegen auch reguläre Partitionen sowie Swapfiles verschlüsseln. Es versucht den Benutzer bei der Eingabe des Passwortes vor Keyloggern zu schützen und bringt auch ein kleines Tool zum sicheren Löschen von Dateien mit, welches Datenreste in einem Dateisystem löscht.

Unter Linux wird BestCrypt als Source-RPM ausgeliefert und muss zur Installation übersetzt werden. Unter Linux werden lediglich Volumes in sogenannten „Container Files“ unterstützt, keine Partitionen oder Volumes. Die Container Files sind jedoch mit denen von BestCrypt unter Windows kompatibel solange ein kompatibler Key-Generator verwendet wurde<sup>28</sup>. Selbstverständlich muss der Container dabei ein für beide Betriebssysteme verfügbares Dateisystem verwenden, also idR. FAT32.

Zusätzlich zu den Betriebssystem-Modulen für Windows und Linux bietet Jetico als leicht gewichtige Anwendung den „BestCrypt Traveller“, um ohne Administrationsrechte und ohne Softwareinstallation den Inhalt eines Containerfiles bearbeiten zu können. Diese Funktionalität ist der des FreeOTFE-Explorers in etwa vergleichbar und

25 <http://www.freeotfe.org/index.html>

26 <http://www.jetico.com/bcrypt.htm>

27 <http://www.jetico.com/>

28 <http://www.jetico.com/linux.htm>

für Windows verfügbar. Des weiteren bietet Jetico mit „BestCrypt Corporate Edition“ eine Version zum Einsatz in Firmen oder anderen größeren Umgebungen an. Allerdings beschränkt sich die Unterstützung hier auf die Installation der Software auf einer Vielzahl von Clients. Firmenweite Schlüsselverwaltung wie bei PGP ist nicht enthalten. Auch ein Verschlüsseln der Systemplatte ist nicht vorgesehen.

## 4 Fazit

Die zunehmende Entwicklung und Verbreitung von Systemen zur Verschlüsselung von Dateisystemen und Datenträgern geht einher mit dem immer deutlicher werdenden Bedarf für solche Systeme. Auch jenseits von Unternehmen und Behörden wächst, wenn auch langsam, die Erkenntnis, dass Daten aller Art vor Missbrauch geschützt werden müssen. Die hier vorgestellten Lösungen eignen sich alle, um Daten nach einem Verlust der Datenträger vor unberechtigtem Zugriff zu schützen. Allerdings sind sie alle durch die Cold Boot Attacke angreifbar, ein wirklicher Schutz der Daten ist daher erst dann gewährleistet, wenn die Systeme entweder gegen physischen Verlust gut gesichert, oder aber vollständig abgeschaltet sind. Einzelne Datenträger wie bspw. USB-Sticks sind hiervon naturgemäß ausgenommen, Verschlüsselung ist hier ein wirksamer und geeigneter Schutz.

Linux, Windows und Mac OS X bringen jeweils eigene Verschlüsselungslösungen mit. ZFS von OpenSolaris wird in Kürze ebenfalls entsprechend ausgestattet sein. Des Weiteren gibt es eine Reihe von teilweise plattform-übergreifenden Systemen, die in Ihrer Funktionalität teilweise weit über die mitgelieferten Systeme hinaus reichen. Insbesondere in Bezug auf eine zentrale Schlüsselverwaltung unterscheiden sich die Systeme hier deutlich. Die Integration in Unternehmensweite Verzeichnisdienste wie LDAP oder ActiveDirectory sowie die zentrale Verwaltung der Sicherheits-Richtlinien wie sie bspw. PGP bereits anbietet, wird sicherlich der nächste Schritt bei der Weiterentwicklung und der größeren Verbreitung von Datenverschlüsselung sein.