JOHANNES KEPLER
UNIVERSITÄT LINZ
Netzwerk für Forschung, Lehre und Praxis

Forensische Auswertung von Festplattenimages

MASTERARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Masterstudium

INFORMATIK

Eingereicht von:
Thomas Weisshaar BSc, 0156317

Angefertigt am:
Institut für Informationsverarbeitung und Mikroprozessortechnik (FIM)

Betreuung:
o. Univ.-Prof. Dr. Jörg R. Mühlbacher

Mitbetreuung:
Priv.-Doz. Mag. iur. Dipl.Ing. Dr. Michael Sonntag

Linz, September 2008

Zusammenfassung

Der Fokus dieser Arbeit liegt auf der Beschreibung mehrerer Übungsszenarien, welche im Rahmen einer Lehrveranstaltung von den Teilnehmern forensisch untersucht werden sollen. Diese Übungsszenarien enthalten, neben der Angabe, eine detaillierte Anleitung für die Durchführung, mit anschließender Auflistung der zu generierenden Resultate. Diese Szenarien sind so aufgebaut, dass sich die Studenten zuerst in die Thematik und die Programme einarbeiten können, und anschließend Aufgaben mit steigendem Schwierigkeitsgrad zu lösen haben. Damit bei der Ausarbeitung nicht auf der grünen Wiese begonnen werden muss, werden in einem eigenen Kapitel zuerst hilfreiche Programme und Tools aus dem Bereich Computer-Forensik vorgestellt. Anschließend folgt die Beschreibung eines allgemeinen Dateisystemmodels und der Dateisysteme FAT, NTFS und Ext3. Neben den Konzepten wird ebenfalls darauf eingegangen, wie gelöschte Daten in den einzelnen Systemen gefunden und wiederhergestellt werden können. Die Unterschiede zwischen den einzelnen Computer-Forensik-Programmen sollen anhand der Ausarbeitung identischer Aufgaben mit verschiedenen Programmen verdeutlicht werden. Es ist noch darauf hinzuweisen, dass die Datenstrukturen der Dateisysteme nicht detailliert, sondern nur soweit es für das Verständnis der Studenten in den Szenarien erforderlich ist, beschrieben werden.

Abstract

The main part of this work is to describe scenarios about forensic investigations to be solved by students of a special computer forensic course. In addition to the description of these scenarios, this work will give a detailed tutorial of how to solve them. The intended results of these investigations are listed in an extra paragraph. For a better start, a separate chapter lists and describes useful tools and programs for forensic investigation. Describing a generic file system and three popular file systems, FAT, NTFS and Ext3 including the possibilities of the recovering deleted files complete this work. It is important to know that the description does not include every file system datastructure. There are only those datastructures included wich are important for the students to know for solving the scenarios.

Inhaltsverzeichnis

1 Motivation und Einleitung

1.1 Motivation

Das Interesse an der Computer-Forensik entstand auf der einen Seite aus der Beschäftigung mit Computersicherheit im Rahmen der projektorientierten Wahlfachgruppe, auf der anderen Seite aus persönlichen Erfahrungen eines Praktikums und von Vorfällen in meinem Verwandten- und Bekanntenkreis. Dabei beschränkt sich letztere Erfahrung auf die Wiederherstellung von gelöschten Daten und auf das Suchen von Systemfehlern.
Ein weiterer wichtiger Punkt für die Beschäftigung mit Computer-Forensik sind diverse Berichte in den Medien über Diebstähle von Daten und Einbrüche in Computersysteme.

1.2 Verwendung

Im Rahmen eines geplanten Masterstudiums für Computer-Forensik soll zu einer Einführungsvorlesung eine ergänzende Übung abgehalten werden, in welcher die teilnehmenden Studenten anhand von Übungsszenarien eine Einführung in Computer-Forensik-Werkzeuge und in das selbstständige Lösen von Aufgaben erhalten. Die dazu benötigten Szenarien ergänzt um eine Musterlösung bilden den Hauptteil dieser Arbeit.

1.3 Einleitung

Ausgehend von der Motivation für das Thema Computer-Forensik beschreibt dieses Einleitungskapitel, worüber es in dieser Arbeit geht und worüber nicht.
Nachdem der Bereich der Computer-Forensik sehr umfangreich ist, beschränkt sich diese Arbeit auf die Analyse von Festplattenimages, sozusagen auf die Post-mortem-Analyse, die Beschreibung von Dateisystemen sowie auf die Möglichkeiten der Datenwiederherstellung. Neben dem theoretischen Teil, der sich auf die gängigsten Dateisysteme FAT, NTFS und Ext3 konzentriert, werden in einem eigenen Kapitel Übungsszenarien, welche im Rahmen einer eigenen Lehrveranstaltung über Computer-Forensik von den teilnehmenden Studenten ausgearbeitet werden sollen, beschrieben.

1.3.1 Computer-Forensik

Computer-Forensik kann als die Untersuchung von digitalen Beweisen, die nach einem Sicherheitsvorfall gesammelt wurden, gesehen werden. Dazu zählt das Sammeln, Analysieren und Rekonstruieren möglichst vieler digitaler Daten im Zuge eines Ermittlungsprozesses. Des Weiteren zählen das Vorbereiten und Aufspüren von Zwischenfällen sowie das Wiederher-

stellen wichtiger, vermeintlich gelöschter Daten nach einem Zwischenfall zu diesem Bereich. In den meisten Fällen dienen diese Untersuchungen der Beweise dem Nachweis von Verbrechen und der Strafverfolgung. Vergleiche dazu die Definition im [IT-Lexikon].
Nach dem Eintreten eines solchen Zwischenfalls gibt es drei wichtige Punkte die durchzuführen sind. Diese Punkte werden auf Englisch auch die 3 A's genannt.

1.3.1.1 Die 3 A's der Computer-Forensik

1. Aquire: Die Beweise sichern, ohne dabei das Original zu verändern (*acquire the evidence*)
2. Authenticate: Überprüfung der Beweise auf ihre Echtheit (*authenticate the recorded evidence*)
3. Analyze: Analysieren der gesammelten Daten ohne diese zu zerstören (*analyze the data*)

[Haa01]

Um den Studenten vorzuführen, wie wichtig eine korrekte Erstellung der Images ist, wird an einem Image gezeigt, welche Daten sich ändern, wenn das betroffene System „nur" gebootet wird.
Nicht in dieser Arbeit behandelt wird die gezielte Suche nach der Aufklärung von Angriffen und das Aufspüren von *Rootkits*, da für diese Fälle weitere Informationen über die betroffenen Betriebssysteme und Informationen von externen Quellen, wie die Loggfiles von Firewalls und Intrusion Detection Systemen, benötigt werden.

2 Übersicht über Computer-Forensik-Programme

Mittlerweile gibt es im Bereich Computer-Forensik eine Menge an Programmen, die allgemein für die Untersuchung von digitalen Beweisen und im Speziellen für die Wiederherstellung von Daten verwendet werden können. Aus diesem Grund umfasst die folgende Auflistung eine Übersicht über die wichtigsten Programme und Toolsammlungen, jedoch ohne einen Anspruch auf Vollständigkeit erheben zu wollen.

2.1 Kommerzielle Tools

In diesem Kapitel werden die wichtigsten und am weitest verbreiteten kommerziellen Tools angeführt.

2.1.1 X-Ways Forensics

Hersteller: X-Ways Software Technology AG, www.x-ways.net
Plattform: Windows 2000/XP/2003, eingeschränkt auch unter Windows Vista/2008. Die Unterstützung für Windows 98/Me wird seit v12/v13 nach und nach aufgegeben.

X-Ways Forensics ist eine Erweiterung des bekannten Hexeditors WinHex um forensische Funktionen, wie eine komplette Fallverwaltung, automatische Protokollierung der Arbeitschritte und automatischer Erstellung von Berichten im HTML-Format. Eine weitere wichtige Funktion, die für die Ermittlung von Kinderpornografie-Delikten nützlich sein kann, ist das Berechnen des Hautanteils bei Bildern.

Funktionsübersicht (Auszug aus den Herstellerangaben):

- Klonen von Datenträgern und Erstellen von Diskimages
- Extrahieren von Daten aus Diskimages
- Unterstützung folgender Dateisysteme: FAT, NTFS, Ext2/3/4, CDFS, UDF, HFS, HFS+, ReiserFS, Reiser4, UFS, UFS2
- Unterstützung folgender Partitionierungsarten: MBR, Windows dynamische Platten, GUID (GPT), Apple, unpartitioniert (Floppy/ Superfloppy)
- Native Interpretation von RAID-Systemen (Level 0 und 5) und dynamischen Platten
- Forensisch sicheres Löschen von Festplatten
- Extraktion von Schlupfspeicher (slack space), freiem Laufwerksspeicher, Partitionslückenspeicher und Text auf Datenträgern und Image-Dateien

- Erkennung und Zugriff auf alternative Datenströme (ADS) von NTFS
- Automatisches Auflisten von Archivinhalten, auch in rekursiven Ansichten

Hinweis: Testversion auf Anfrage erhältlich, wobei diese nur mit dem lokalen Laufwerk C: arbeiten kann.

2.1.2 EnCase Forensic

Hersteller: Guidance Software, www.guidancesoftware.com
Plattform: Windows 2000/XP/2003

EnCase Fornesic ist aufgrund seiner großen Verbreitung gewissermaßen das Standardprogramm für Ermittler im Bereich Computer-Forensik. Eine Spezialität ist die integrierte Scriptsprache EnScript, mit welcher es möglich ist, Arbeitsabläufe zu automatisieren.

Funktionsübersicht (Auszug aus den Herstellerangaben):

- Folgende Dateisysteme werden unterstützt: FAT12/16/32, NTFS, Ext2/3, ReiserFS, UFS, AIX Journaling File System (JFS) LVM8, FFS, Palm, HFS, HFS+, CDFS, ISO 9669, UDF, DVD, TiVo® 1 und TiVo® 2
- Erstellen von Diskimages
- Analyse verschlüsselter Partitionen und Festplatten
- Forensische Analyse der Active Directory Datenbank
- Wiederherstellung von Ordnern auf NTFS und FAT Laufwerken
- Registry und Hardware Analyse aus der Registry
- Analyse von Dateisignaturen
- Einbindung externer Programme z. B. für das Öffnen von MP3 oder Video Dateien
- Analyse von virtuellen Laufwerken, die von VMware verwendet werden
- Analyse von E-Mails, Browser Caches
- Instant Messenger Toolkit
- Kazaa Toolkit

Hinweis: Demo-DVD auf Anfrage erhältlich (ca. 2-4 Wochen). Auf dieser DVD befinden sich zwei Image-Dateien, eine dieser Dateien ist ein komplettes Image einer Windows XP Installation, die andere Datei ist eine Sicherung einer Mail-Datenbank.
Die Demo-Version von EnCase Forensic ist voll funktionsfähig, jedoch auf die beiden Images der Demo-DVD beschränkt.

2.1.3 Forensic Toolkit

Hersteller: AccessData Corp., www.accessdata.com

Plattform: Windows 2000/XP/2003 und Vista je nach Version

Eine Besonderheit dieses Programms ist, dass der Benutzer mit einem Wizard begrüßt wird, mit welchem ein neuer Fall angelegt, die Datenquelle angegeben und Einstellungen für die Analyse getroffen werden. Nach dem Abschluss des Wizards werden die Eingabedaten anhand der vorgegebenen Einstellungen analysiert und ein Index der enthaltenen Dateien erstellt. Nach dieser zeitintensiven Arbeit kann mit der gezielten Analyse begonnen werden. Wie es sich für ein professionelles Forensik-Werkzeug gehört, werden alle Tätigkeiten des Ermittlers automatisch protokolliert und können als Bericht exportiert werden.

Funktionsübersicht (Auszug aus den Herstellerangaben):

- Erstellen von Festplattenimages
- Unterstützte Dateisysteme: NTFS, NTFS compressed, FAT 12/16/32, und Linux Ext2 & Ext3
- Verarbeiten von Images, welche in folgenden Formaten vorliegen: Encase, SMART, Snapback, Safeback (bis Exklusive v.3) und Linux DD
- Automatisches Wiederherstellen von Dateien und Partitionen
- Datenextrahierung aus Dateien, welche mit folgenden Programmen komprimiert wurden: PKZIP, WinZip, WinRAR, GZIP und TAR
- Wiederherstellung von E-Mails
- Erstellen von benutzerdefinierten Filtern

Hinweis: Demoversion kann von der Hersteller-Homepage heruntergeladen werden. Diese Version ist voll funktionsfähig, jedoch auf 5000 Dateien beschränkt.

2.2 Nicht kommerzielle Tools

Nach den kommerziellen Tools folgen hier nun einige nicht kommerzielle Tools für forensische Analysen.

2.2.1 ILookv8

Hersteller: IRS-CI Electronic Crimes Program and Perlustro, LP, www.ilook-forensics.org

Plattform: Windows 2000/XP 32bit und Windows Server 2003 64bit

Funktionsübersicht (Auszug aus den Herstellerangaben):

- Erstellen von Festplattenimages
- Verarbeiten von Images, welche mit anderen Tools erstellt wurden, falls es sich hierbei um „raw bit stream“ Images handelt.
- Unterstützte Dateisysteme: FAT12, FAT16, FAT32, FAT32x, VFAT, NTFS, HFS, HFS+, Ext2FS, Ext3FS, SysV AFS, SysV EAFS, SysV HTFS, CDFS, Netware NWFS, Reiser FS, ISO9660
- Extrahieren von Daten aus dem Image
- Überprüfung von Dateisignaturen
- Wiederherstellung gelöschter FAT-Verzeichnisse
- Multiformat Datei-Viewer
- Hash Analysefunktion
- Datenbank für Suchergebnisse
- Datei Filter- und Eliminierungsfunktion

Hinweis: Dieses Tool ist nur für Geheimdienste und staatliche Organisationen, welche sich mit der forensischen Aufklärung von Verbrechen beschäftigen, erhältlich. Wobei ausdrücklich darauf hingewiesen wird, dass Universitäten keinen Zugriff auf dieses Tool erhalten.

2.2.2 The Sleuth Kit / Autopsy Forensic Browser

Hersteller: Brian Carrier, http://www.sleuthkit.org/

Plattform: Linux, Mac OS X, Open & FreeBSD und Solaris

The Sleuth Kit ist eine Sammlung von Command Line Tools, mit welchen umfangreiche forensische Untersuchungen an Festplatten und Festplattenimages durchgeführt werden können. Im Folgenden eine kurze Funktionsübersicht (Auszug aus den Herstellerangaben):

- Unterstützte Dateisysteme: NTFS, FAT, UFS1/2, Ext2/3 und ISO 9660
- Analyse von verschiedenen Diskimages: raw (z. B. dd), Expert Witness (z. B. EnCase), AFF Datei System und Disk Images
- Suchen von zugeteilten (*allokierten*) und nicht zugeteilten (*nicht allokierten*) Dateien
- Anzeige aller NTFS-Attributdetails und Daten inklusive aller Alternate Data Streams
- Anzeige von Dateisystem und Metadatendetails
- Sortierung nach Dateitypen

Mit dem Autopsy Forensic Browser ist eine graphische webbasierte Benutzeroberfläche für The Sleuth Kit vorhanden. Ergänzend zu den Funktionen aus dem Sleuth Kit bietet der Autopsy Forensic Browser die Möglichkeit der Verwaltung einzelner Fälle.
Ein spezielles Feature des Autopsy Forensic Browser ist die Live CD, die nach dem erfolgreichen Compilieren, mittels eines Perl Skripts, erzeugt werden kann. Diese CD beinhaltet alle benötigten Programme für die Verwendung des Autopsy Forensic Browsers. Anschließend kann mit dieser CD an einem laufenden System eine forensische Untersuchung durchgeführt werden.

Hinweis: The Sleuth Kit und der Autopsy Forensic Browser sind als Open Source auf der Seite des Herstellers erhältlich.

2.2.3 F.I.R.E.

Das *Forensic and Incident Response Environment* ist eine Toolsammlung, welche aus statisch vorkompilierten Systemdateien für verschiedene Betriebssysteme und eine Menge von leistungsfähigen Duplizier- und Forensiktools besteht. Weiters beinhaltet die CD, welche aus der vom Hersteller zur Verfügung gestellten Image Datei erzeugt werden kann, eine kleine bootfähige RedHat-Linux-Version. Wird ein Computer mit dieser CD gebootet, werden alle vorhandenen Partitionen read-only gemountet und sind dann für eine umfassende Analyse bereit. Unterstützt werden FAT-, NTFS-, Ext2-, Ext3-, und ReiserFS-Partitionen. Für die Analyse stehen bekannte Open Source Tools wie The Sleuth Kit, Autopsy Forensic Browser, macrobber, lsof usw. zur Verfügung.
Neben den Programmen für die forensische Analyse finden sich auch Programme für das Sammeln flüchtiger Daten von Live Systemen, Tools für das Durchführen von Penetrations-Tests und eine Menge von statisch kompilierten Systemprogrammen für Windows, Linux und Solaris auf dieser CD.

2.2.4 Knoppix STD

Durch die sehr gute Hardwareunterstützung von Knoppix erweitert um Programme für die forensische Untersuchung wie The Sleuth Kit / Autopsy Forensic Browser, siehe Kapitel 2.2.2, ist die Knoppix STD CD ein praktisches Tool. Ein wichtiger Punkt bei dieser Knoppix-Version ist, dass die vorhandenen Festplatten nur im read only Modus gemountet werden. Die erhobenen Daten werden in einer RAM-Disk zwischengespeichert und können später etwa über das Netzwerk gesichert werden.

Neben den Forensik Tools sind noch weitere Programme, zum Beispiel für die Verschlüsselung von Daten, Firewalls, Intrusion Detection Systeme, Passwort Utilities und mehr, vorhanden. Eine genaue Auflistung der Programme findet sich auf der Homepage des Herstellers. Wobei hier beachtet werden muss, dass es sich bei dieser Knoppix Version nicht um ein spezielles Forensik-Tool handelt, was jedoch die Verwendung für diesen Zweck nicht ausschließt.

2.3 Weitere nützliche Programme

2.3.1 Live View

Mit diesem Java-Programm können aus forensischen Festplattenimages oder Laufwerken virtuelle Maschinen für den VMware Server erstellt werden. Wobei ein großer Vorteil darin besteht, dass alle Änderungen an den Images in separaten Dateien gespeichert werden und so keine Sicherheitskopien für die virtuellen Maschinen benötigt werden.
Unterstützt werden folgende Betriebsysteme

- Windows 98 / Me / NT / 2000 / XP / 2003
- Eingeschränkt Linux

Damit dieses Programm verwendet werden kann, wird eine lauffähige Java-Installation und der VMware Server benötigt. Mit dem für Windows verfügbaren Installationsprogramm werden beide Programme, wenn nötig, von den Hersteller-Homepages herunter geladen und installiert.
Hersteller dieses als Open Source[1] verfügbaren Programms ist CERT[2] und das Software Engineering Institut[3] der Carnegie Mellon Universität[4].

2.3.2 dd

Mit diesem ursprünglich für das Kopieren unter Unix entwickelten Programm können einfach raw Kopien einzelner Partitionen oder ganzer Festplatten erstellt werden. Da dieses Programm für die forensischen Untersuchungen immer wichtiger geworden ist, gibt es einige nützliche Erweiterungen. So zum Beispiel die Version `dcfldd`[5], welche vom amerikanischen

[1] http://liveview.sourceforge.net/
[2] http://www.cert.org
[3] http://www.sei.cmu.edu
[4] http://www.cmu.edu
[5] http://sourceforge.net/projects/dcfldd/

Verteidigungsministerium für die zusätzliche simultane Berechnung und Überprüfung von MD5 Prüfsummen während der Imageerstellung entwickelt wurde.

2.3.3 netcat (nc)

Dieses universell einsetzbare Programm für die Übertragung von Daten über ein Netzwerk kann, in Kombination mit Unix-Pipes, für das Übertragen von Programm Ein- bzw. Ausgaben, etwa für die Sicherung von Laufwerken über Netzwerkverbindungen, verwendet werden. Steht hierfür nur ein unsicheres Netzwerk zur Verfügung, so gibt es das Programm `cryptcat`[6] anstelle von `netcat`.

2.3.4 foremost

Mit diesem, von den Special Agents Kris Kendall und Jesse Kornblum vom United States Air Force Office of Special Investigation entwickelten Programm ist es möglich, aus einer Menge zusammenhängender Raw-Daten einzelne Dateien anhand ihrer Signaturen (Magic Numbers) zu extrahieren. Das ursprünglich nur der amerikanischen Regierung zur Verfügung stehende Programm wurde mittlerweile als Open Source Programm[7] der Öffentlichkeit für die Verwendung und Weiterentwicklung zur Verfügung gestellt.

[6] http://sourceforge.net/projects/cryptcat
[7] http://www.sourceforge.net/projects/foremost

3 Beschreibung eines allgemeinen Dateisystemmodels

In den vorherigen Kapiteln wurde bereits viel über Dateisysteme geschrieben, doch bevor es im nächsten Kapitel mit der Beschreibung von Dateisystemen weitergeht, muss zuerst geklärt werden, was unter einem Dateisystem zu verstehen ist bzw. wozu es verwendet wird.
Für das Verwenden von Dateisystemen gibt es einen einfachen Grund: Computer benötigen eine Methode für die Langzeitspeicherung von Daten. Dateisysteme bieten hierfür Mechanismen zum Speichern, hierarchischem Ordnen in Dateien und Ordner sowie für das Wiederfinden der gespeicherten Daten. Ein wichtiger Punkt hierbei ist, dass das Dateisystem in den meisten Fällen unabhängig vom verwendeten Computer bzw. Betriebssystem ist.
Vor der Beschreibung konkreter Dateisysteme stellt dieses Kapitel als Einführung das Referenzmodel für Dateisysteme von Brian Carier aus [Car05] vor. Bei dieser Quelle handelt es sich um die Primärquelle dieses und des nächsten Kapitels. Wenn nicht direkt angegeben, dann handelt es sich um freie Übersetzungen aus [Car05] ergänzt um eigene Teile und Teilen aus weiteren Quellen.

3.1 Allgemeine Datenstrukturen eines Dateisystems

Für ein leichteres Verständnis werden die Daten nach [Car05 Seite 174] in fünf Kategorien unterteilt: *file system*, *content*, *metadata*, *file name* and *application*.
In der *file system* Kategorie sind alle allgemeinen Dateisystemdaten enthalten, wie Informationen über den Speicherort wichtiger Datenstrukturen und über die Größe der einzelnen Dateneinheiten, oder einfach die Verwaltung des Dateisystems selbst. Für eine bessere Vorstellung kann man sich auch eine Karte über den Inhalt des Dateisystems vorstellen.
In der *content* Kategorie befinden sich die aktuellen Inhaltsdaten der einzelnen Dateien, welche typischerweise als eine Sammlung von Dateneinheiten gesehen werden.
Die *metadata* Kategorie beinhaltet jene Daten, die für die Beschreibung einer Datei benötigt werden, wie der Speicherort der Datei, die Dateigröße, die Zugriffsdaten, wie Datum und Zeit des letzten Schreib- und Lese-Zugriffs. Zu beachten ist bei dieser Kategorie, dass keine Dateiinhalte gespeichert sind. Beispiele für Daten dieser Kategorie sind die FAT-Ordnereinträge, die NTFS Master File Table (MFT) Einträge und die UFS und Ext3 Inode-Strukturen.
Die *file name* Kategorie beinhaltet die Namen der gespeicherten Dateien. Diese sind in den meisten Dateisystemen im Content der Ordner gespeichert und können als Liste von Tupeln von Dateinamen mit den dazugehörigen Metadaten-Adressen gesehen werden. Die Funktionsweise dieser Kategorie kann auch mit der von DNS verglichen werden. Über die *file name*

Kategorie erhält man zu einem Dateinamen die Speicheradresse einer Datei, von einem DNS-Server die IP-Adresse zu einer URL.
In der *application* Kategorie befinden sich spezielle Funktionen, welche nicht unmittelbar zu den Funktionen eines Dateisystems gehören, die aber aus Gründen der Effizienz in die Spezifikation aufgenommen werden. Beispiele für Daten dieser Kategorie sind Dateisystemjournals und Statistiken über User-Quotas.
Den Zusammenhang der verschiedenen Kategorien zeigt die folgende Abbildung.

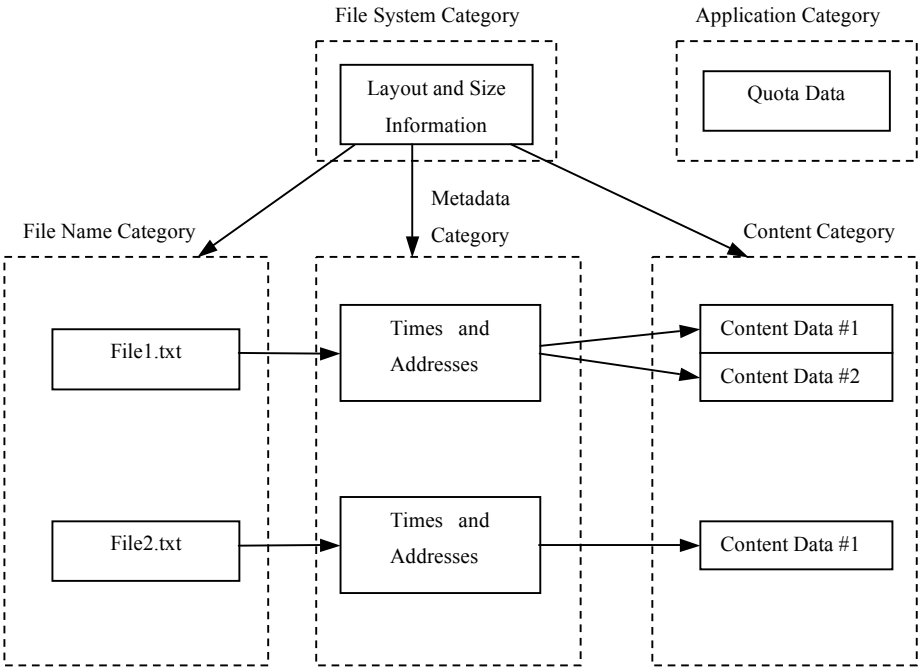


Abb. 3.1 Zusammenhang der 5 Kategorien [Car05 Figure 8.1 Seite 175]

3.2 Beschreibung der verschiedenen Kategorien

Bevor in Kapitel 3.3 die Möglichkeiten der Dateiwiederherstellung behandelt werden, muss zuerst geklärt werden, welche Daten in den einzelnen Kategorien gespeichert sind und wie diese zusammenhängen. Da sich diese Arbeit auf die Wiederherstellung von Dateien beschränkt, werden nur die verschiedenen Kategorien, nicht aber die verschiedenen Analysemethoden zu diesen Kategorien beschrieben. Weitere Informationen zu diesen Analysemethoden finden sich im Kapitel 8 „File System Analysis“ [Car05 Seite 173], wobei zu beachten ist, dass in den meisten Analysetools die verwendeten Methoden nicht gesondert nach ihren zugehörigen Kategorien angegeben werden, sondern integrierte Bestandteile der angebotenen Analysemethoden sind.

3.2.1 File System Kategorie

Obwohl es sich bei den Daten dieser Kategorie streng genommen ebenfalls um Metadaten handelt, wird diese Kategorie, da es sich um die Daten des Dateisystems handelt, als File System Kategorie bezeichnet. Dazu zählen alle wesentlichen Daten, welche das Dateisystem beschreiben und angeben, wo andere wichtige Daten gespeichert sind. In den meisten Fällen werden diese Daten in Standarddatenstrukturen in den ersten Sektoren des Dateisystems gespeichert.

Die Kontent-Daten eines Dateisystems werden üblicherweise in Einheiten derselben Größe, welche allgemein als Dateneinheiten bezeichnet werden, gespeichert. Diese Dateneinheiten werden in einem konkreten Dateisystem etwa als *Cluster* oder *Block* bezeichnet. Die Daten dieser Kategorie können, aber müssen nicht, in diesen Dateneinheiten gespeichert sein. So kann ein Teil der Partition für das Speichern der Dateisystemdaten und der Rest für die Unterteilung in Dateneinheiten für alle anderen Daten verwendet werden. Zudem besitzt jedes Dateisystem eine eigene Datenstruktur, in welcher es den Zuteilungsstatus, *allocation status*, jeder Dateneinheit speichert.

Die Analyse der Daten dieser Kategorie ist von großer Bedeutung, da die Datenstrukturen der anderen Kategorien nur über die Analyse dieser Daten erreichbar sind. Werden Teile oder die gesamten Daten beschädigt oder gelöscht, wird die Analyse bedeutend schwieriger, da diese zuerst über mögliche Backups oder Standardwerte rekonstruiert werden müssen. Zu den allgemeinen Layout Informationen kann über die Analyse dieser Kategorie die Version des Dateisystems, die erzeugende Applikation, das Erstellungsdatum und das Dateisystem-Label herausgefunden werden.

3.2.2 Content Kategorie

Hier befinden sich die Speicherorte, welche den vorhandenen Dateien und Ordner zugewiesen sind. Die Daten in dieser Kategorie sind üblicherweise in den Dateneinheiten des Dateisystems gespeichert.

Wird eine neue Datei angelegt oder eine bestehende vergrößert, so sucht das Betriebssystem nach nicht zugeteilten Dateneinheiten und teilt diese der Datei zu. Die verschiedenen Möglichkeiten für das Suchen dieser Dateneinheiten beschreibt ein eigenes Unterkapitel.

Werden Daten gelöscht, so wird der Zuteilungsstatus der dazugehörigen Dateneinheiten vom Betriebssystem auf ‚*nicht zugeteilt*' gesetzt, anschließend können diese für andere Daten verwendet werden. In der Art und Weise wie das Betriebsystem das Setzen des Status auf ‚*nicht zugeteilt*' durchführt, liegt das Potential für das Suchen nach gelöschten Daten bzw. Bewei-

sen. In einfachen Implementierungen wird nur der Status, nicht jedoch die Daten selbst, verändert.
Aufwendigere Implementierungen ändern den Status und überschreiben die Daten entweder mit Nullen oder mit zufälligen Werten. Für das Auffinden und Wiederherstellen gelöschter Daten ist die erste Art der Implementierung natürlich von Vorteil.
Die Analyse in dieser Kategorie besteht aus dem Versuch, gelöschte Daten wiederherzustellen und der Suche nach Schlüsselwörtern. Wegen der großen Datenmenge ist es so gut wie unmöglich, innerhalb dieser Kategorie eine Analyse ohne Unterstützung von Werkzeugen durchzuführen.

Logische Dateisystemadressen

Ein Sektor hat, je nach Sichtweise, unterschiedliche Adressen. Daher hat ein Sektor eine physikalische Adresse, welche vom relativen Abstand des Sektors zum Beginn des Speichermediums abhängt. Des Weiteren hat dieser Sektor, falls er sich in einem Volume befindet, eine logische Adresse, welche den relativen Abstand zum Beginn des Volumes angibt.
Zu diesen Adressen kann noch eine logische Dateisystemadresse hinzukommen, wenn das Dateisystem mehrere Sektoren zu einer Speichereinheit verbindet. Die folgende Abbildung zeigt, wie Sektoren zu Dateneinheiten verbunden werden. Dabei beginnt das Dateisystem erst bei Sektor 4 und endet bei Sektor 15. Der letzte nicht mehr verwendete Sektor 16 wird als *volume slack* bezeichnet.

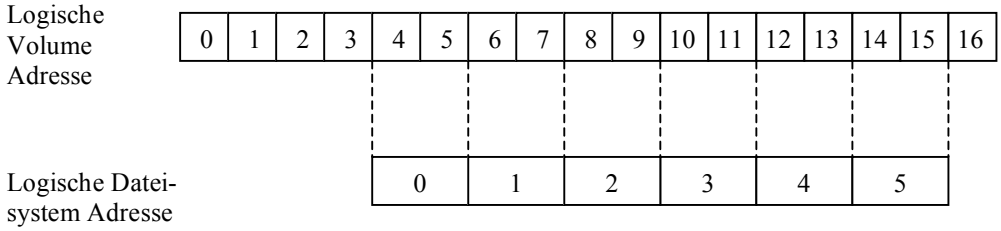


Abb. 3.2 Beispiel für die Adressierung von Speichereinheiten in einem Dateisystem vgl. [Car05 Figure 8.2 Seite 179]

Zuteilungsstrategien

Für das Zuteilen von Speichereinheiten gibt es für das Betriebssystem verschiedene Möglichkeiten, wobei in erster Linie versucht wird, zusammenhängende Dateneinheiten mit der passenden Größe zu finden. Falls dies nicht gelingt, entsteht eine fragmentierte Datei.

Wird vom Betriebsystem die *first available* Strategie verwendet, so beginnt die Suche nach einer Speichereinheit immer am Beginn des Dateisystems. Da bei dieser Strategie die Dateien nicht als Ganzes erzeugt werden, entstehen sehr leicht fragmentierte Dateien. Für das Wiederherstellen von Daten ist zu beachten, dass durch diese Strategie die Daten am Beginn des Dateisystems viel wahrscheinlicher überschrieben werden als solche, die am Ende des Dateisystems gespeichert waren.
Eine ähnliche Strategie ist die *next available* Strategie, bei welcher ausgehend von der zuletzt zugeteilten Speichereinheit die Suche nach freien Speichereinheiten begonnen wird. Diese Strategie ist für die Datenwiederherstellung viel günstiger, da die Speichereinheiten am Beginn des Dateisystems erst dann überschrieben werden, wenn bei der Suche das Ende des Dateisystems erreicht wurde.
Eine etwas andere Strategie ist die *best fit* Strategie. Bei dieser wird versucht, zusammenhängende Dateneinheiten in der benötigten Größe zu finden. Zu beachten ist jedoch, dass dies nur funktioniert, solange die benötigte Größe bekannt ist. Tritt der Fall auf, dass eine Datei z.B. nachträglich mehr Speicher benötigt, so wird diese ebenfalls leicht fragmentiert.
Findet diese Strategie keine zusammenhängenden Speichereinheiten in der benötigten Größe, so wird entweder auf die *first* oder *next available* Strategie zurückgegriffen.
Obwohl in einigen Dateisystemspezifikationen eine zu verwendende Strategie angegeben ist, wird diese jedoch von den meisten Betriebssystemen selbst gewählt.

Beschädigte Dateneinheiten

Ältere Festplatten hatten noch keine Möglichkeiten, fehlerhafte Sektoren aufzuspüren und als solche zu markieren. Daher gibt es in vielen Dateisystemen die Möglichkeit, Sektoren als beschädigt zu markieren. Diese Funktion ist bei neuen Festplatten jedoch nicht mehr notwendig, da diese Funktion bereits in die Hardware integriert ist. Das bedeutet jedoch nicht, dass diese Funktion nicht mehr verwendet wird. So wird sie etwa für das Verstecken von Daten auf einem Volume verwendet.

3.2.3 Metadata Kategorie

In dieser Kategorie befinden sich die beschreibenden Daten der Dateien, wie zum Beispiel die Zeit des letzten Zugriffs oder die Speicheradressen der Dateneinheiten die eine Datei bilden. Gespeichert werden diese Daten in den meisten Fällen in Tabellen mit fixer oder dynamischer Größe.

Logische Dateiadresse

Das vorherige Kapitel beschrieb, wie einer Dateneinheit eine logische Dateisystemadresse zugeteilt wird. Diese Adresse ist jedoch nicht die einzige logische Adresse, welche eine Dateneinheit erhält. Nach dem Anlegen einer Datei bekommen die zu dieser Datei gehörigen Dateneinheiten eine weitere logische Adresse, und zwar die logische Dateiadresse, welche relativ zum Beginn der Datei ist. Die folgende Abbildung zeigt den Zusammenhang der einzelnen Adressen einer Dateneinheit.

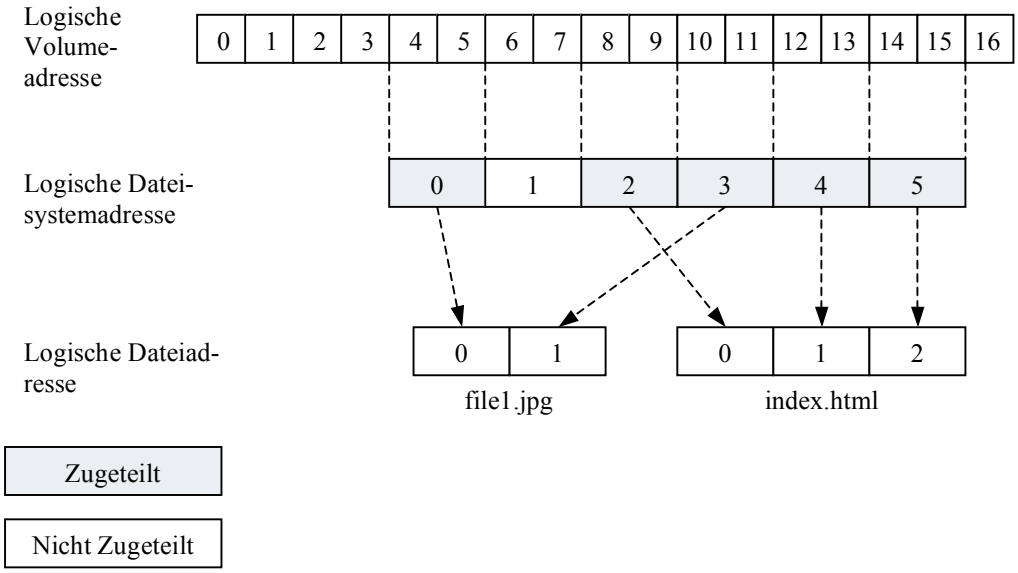


Abb. 3.3 Logische Dateiadressen zweier Dateien in einem kleinen Dateisystem vgl. [Car05 Figure 8.8 Seite 187]

Slack Space

Slack space ist eines der Schlagworte in der Computer-Forensik, welches die meisten Menschen, die in diesem Bereich arbeiten, zumindest schon einmal gehört haben.

Doch was ist *slack space*? Eigentlich ganz einfach zu erklären: Wenn für eine Datei Dateneinheiten reserviert werden und diese Datei nicht alle Bytes benötigt, werden die restlichen Bytes der letzten Dateneinheit als *slack space* bezeichnet. Zur Veranschaulichung ein kleines Beispiel: Für das Speichern einer 200 Byte großen Datei muss eine Dateneinheit mit 2048 Bytes zugeteilt werden. Die 1848 ungenützten Bytes werden als *slack space* bezeichnet.

Ausgehend von der Architektur der meisten Computer gibt es zwei interessante Bereiche von *slack spaces*. Der erste Bereich ist zwischen dem Ende der Dateidaten und dem Ende der Dateneinheit, der Zweite ist jener Speicher in oder außerhalb von Partitionen, der nicht zugeteilt ist.

Das Interessante an diesem Speicher ist, wie er vom Betriebssystem behandelt wird. Im ersten Bereich kommt es darauf an, mit welchen Daten das Betriebssystem die restlichen Bytes auffüllt. Hierfür gibt es mehrere Möglichkeiten: Mit Null-Werten, mit zufälligen Werten oder, wie bei älteren Betriebssystemen wie DOS oder den ersten Windows-Versionen üblich, mit Daten aus dem Hauptspeicher, dieser *slack space* wird auch als *RAM slack* bezeichnet.
Für den zweiten Bereich ist es von Interesse, wie das Betriebssystem nicht mehr benötigte Daten löscht. Wird lediglich der Eintrag in der Metadaten Kategorie gelöscht, oder werden die gespeicherten Daten zum Beispiel mit Null-Werten überschrieben. Im ersten Fall ist es möglich, dass in dem nicht benötigten Speicher der Dateneinheiten noch Daten bereits gelöschter Dateien vorhanden sind. Zu beachten ist hier jedoch, dass Festplatten blockorientierte Speichermedien sind und deswegen jeweils nur ganze Sektoren lesen oder schreiben können. Aus diesem Grund können etwa keine 100 Byte geschrieben werden, wenn die Sektorgröße 512 Byte beträgt. Sollen dennoch diese 100 Byte gespeichert werden, muss das Betriebssystem die restlichen 412 Bytes mit Daten auffüllen, im Normalfall werden hierfür Null-Werte verwendet.

Komprimierte Dateien

Für das Komprimieren von Daten gibt es drei verschiedene Möglichkeiten:

1. Die gesamte Datei wird mit einem Komprimierungsprogramm wie zum Beispiel WinZip®[8] komprimiert.
2. Der Inhalt einer Datei wird komprimiert gespeichert, zum Beispiel mp3 Dateien oder jpg Bilder.
3. Die Daten werden transparent auf der Dateisystemebene komprimiert.

Für die Komprimierung auf der Dateisystemebene gibt es zwei unterschiedliche Möglichkeiten, entweder die Daten werden nach demselben Verfahren wie in Punkt 1 komprimiert, oder es wird versucht, Teile der Daten gar nicht physikalisch zu speichern. Bei dieser Methode wird nur die Existenz von zusammenhängenden Blöcken mit Null-Werten gespeichert. Dateien, die nach dieser Methode komprimiert wurden, werden *sparse files* genannt.
Komprimierte Dateien stellen eine besondere Herausforderung an die forensische Untersuchung, da der Komprimierungsalgorithmus vom verwendeten Forensik-Programm unterstützt werden muss und daher das Suchen nach Schlüsselwörtern und das Wiederherstellen sehr erschwert wird.

[8] Erhältlich unter http://www.winzip.com bzw. für die deutsche Version http://www.winzip.de

Verschlüsselte Dateien

Neben den Möglichkeiten, Daten vor dem Speichern zu verschlüsseln oder ein externes Programm für die Verschlüsselung, analog zur Komprimierung, zu verwenden, kann dies auch auf Dateisystemebene durchgeführt werden. Hierfür gibt es wieder zwei unterschiedliche Möglichkeiten. Die erste Möglichkeit verschlüsselt die Nutzdaten (Daten ohne Metadaten) einer Datei und schreibt das Chiffrat auf die Festplatte. Die zweite Möglichkeit, bei welcher alle Daten, auch die Metadaten, verschlüsselt werden, ist die Verwendung von verschlüsselten Volumes.

Verschlüsselte Dateien stellen, wie die komprimierten Dateien, die forensische Untersuchung vor eine große Herausforderung. Da in den meisten Fällen der Schlüssel für das Entschlüsseln nicht vorhanden ist, können diese Dateien nur durch eine *Brute Force Attacke* entschlüsselt werden, bevor eine inhaltliche Analyse möglich wird.

3.2.4 File Name Kategorie

Hier werden in erster Linie die Namen der Dateien mit den dazugehörigen Metadatenadressen gespeichert. Über diese Dateinamen ist es möglich, auf Dateien ohne Kenntnis der Metadatenadresse zuzugreifen. In einigen Dateisystemen werden zum Dateinamen auch Daten über den Dateityp gespeichert.

Die wichtigste Aufgabe, für welche Daten dieser Kategorie benötigt werden, ist das Auffinden des Root-Directories, da dieses für die weitere Suche nach Dateien von großer Bedeutung ist.

3.2.5 Application Kategorie

In einigen Dateisystemen werden Daten, die eigentlich in Dateien gespeichert werden könnten, aus Gründen der Effizienz ins Dateisystem integriert. Ein Beispiel hierfür sind Journals.

Dateisystemjournals

Jeder Computerbenutzer kennt aus eigener Erfahrung das Problem des Abstürzens oder sich Aufhängens seines Computers. Nach so einem Ereignis ist es nicht ungewöhnlich, wenn das Dateisystem durch nicht abgeschlossene Schreibvorgänge in einem inkonsistenten Zustand ist. Um diese Inkonsistenzen zu beseitigen, führt das Betriebssystem eine Überprüfung der Festplatte durch und versucht, Fehler zu finden und zu beheben. Solche Überprüfungen können bei größeren Dateisystemen mitunter sehr zeitaufwendig sein. Aus diesem Grund führen

viele Dateisysteme ein Journal, in welchem zuerst gespeichert wird, welche Metadaten gespeichert werden sollen, sowie nach dem erfolgreichen Speichern, dass das Speichern erfolgreich war. Unter Verwendung dieser Daten kann ein konsistenter Zustand schneller wiederhergestellt werden.
Da diese Funktionalität nicht essenziell für ein Dateisystem ist, befindet sie sich in der Application Kategorie.

3.3 Methoden für die Datenwiederherstellung

Ist es für eine Untersuchung notwendig, nach Beweisen in gelöschten Daten zu suchen, so gibt es zwei Möglichkeiten für die Wiederherstellung von Dateien: die Metadaten-basierte und die Applikations-basierte Methode.

3.3.1 Metadaten-basierte Dateiwiederherstellung

Diese Methode funktioniert nur, wenn die Metadaten der gelöschten Dateien noch existieren. Ist dies nicht mehr der Fall, ist eine Wiederherstellung nur mehr mit der Methode der Applikations-basierten Wiederherstellung möglich.
Ist die Metadatenstruktur einer Datei gefunden, so unterscheidet sich das Wiederherstellen in keinster Weise vom Lesen einer Datei.

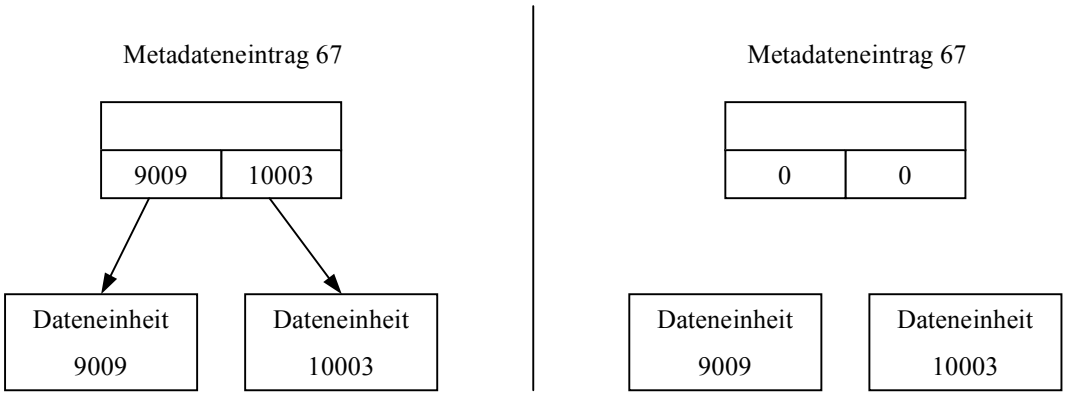


Abb. 3.4 Gelöschte Datei mit Metadateneintrag (links) und gelöschte Datei mit gelöschtem Metadateneintrag (rechts) vgl. [Car05 Figure 8.10 Seite 189]

Bei dieser Art der Wiederherstellung ist zu beachten, dass die Metadatenstrukturen und die Dateneinheiten im Laufe der Zeit nicht mehr zusammenpassen müssen, da etwa die Dateneinheiten für eine neue Datei verwendet werden können, oder der Metadateneintrag inzwischen für eine neue Datei verwendet wird und Adressen anderer Dateneinheiten beinhaltet. So kann es vorkommen, dass zwei oder mehrere, als gelöscht markierte, Metadateneinträge auf diesel-

be(n) Dateneinheit(en) verweisen. In diesem Fall besteht die Schwierigkeit darin, herauszufinden welcher von diesen Einträgen der Richtige ist. Um das herauszufinden, kann, wenn vorhanden, die Zeit der letzten Änderung, oder der in den Metadaten gespeicherte Dateityp mit dem Dateityp aus den Daten verglichen werden, um so zumindest Dateien von Ordnern zu unterschieden.

Ein weiteres Problem mit wiederhergestellten Daten ist, dass eine oder mehrere Dateneinheiten einer Datei einer anderen Datei zugeteilt und von dieser überschrieben wurden, und so den Inhalt der alten Datei in einen inkonsistenten Zustand brachte. Um das herauszufinden, ist es am einfachsten, wenn man diese Datei mit einem Programm öffnet, von dem man annimmt, dass es diese Datei erzeugt hat.

3.3.2 Dateinamen-basierte Dateiwiederherstellung

Diese Dateiwiederherstellung beruht darauf, über den Namen einer gelöschten Datei deren Metadaten zu suchen, um mit diesen eine Wiederherstellung dieser Datei zu erreichen. Wurden für einen Dateinamen die dazugehörigen Metadaten gefunden, so erfolgt die Wiederherstellung wie bei der Metadaten basierten Wiederherstellung.

Jedoch gibt es hier genauso das Problem, dass Dateinamen und Metadateneinträge im Laufe der Zeit nicht mehr zusammenpassen müssen. Das folgende Beispielszenario aus [Car05 Seite 200] zeigt, wie sich die Zusammenhänge von Dateinamen und Metadaten verändern können.

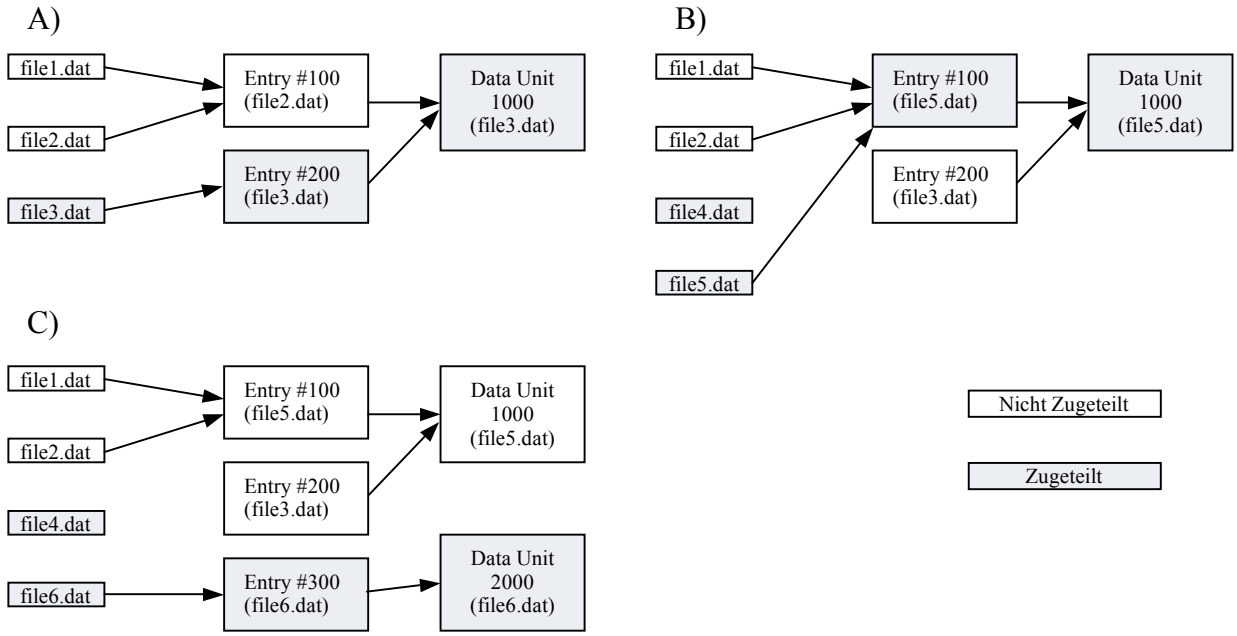


Abb. 3.5 Veränderungen an einem Dateisystem aus der Sicht der Dateinamen und der Metadaten im Laufe der Zeit vgl. [Car05 Figure 8.19 Seite 201]

Abb. 3.5 A) zeigt einen Metadateneintrag mit der Nummer 100, auf welchen zwei Dateinamen gelöschter Dateien verweisen, wobei die Datei file2.dat nach der Datei file1.dat angelegt und gelöscht wurde. Dieser Metadateneintrag verweist wiederum auf die Dateneinheit 1000. Auf diese Dateneinheit verweist aber auch der Metadateneintrag 200 welcher zur Datei file3.dat gehört. In Teil B dieser Abbildung wurde der Dateiname der Datei file3.dat gelöscht und es wurden zwei neue Dateien, file4.dat und file5.dat, angelegt, wobei von file4.dat nur der Name existiert und file5.dat der Metadateneintrag 100 und die Dateneinheit 1000 zugewiesen wurde.
Für die Abrundung dieses Szenarios wurde im Teil C die Datei file5.dat wieder gelöscht, genauer gesagt der Dateiname dieser Datei, und eine neue Datei file6.dat mit einem neuen Metadateneintrag 300 und einer neuen Dateneinheit 2000 angelegt.
Möchten wir dieses System nun im Zustand C untersuchen, so ergeben sich einige Probleme:

- Auf die Dateneinheit 1000 zeigen zwei Metadateneinträge, wobei wir nicht wissen, welcher dieser beiden Einträge zuletzt auf die Dateneinheit gezeigt hat und wir wissen nicht, ob nicht noch mehrere Metadateneinträge, welche bereits gelöscht wurden, auf diese Dateneinheit verwiesen haben.
- Der Metadateneintrag 100 wird von zwei Dateinamen referenziert, wobei wir hier wiederum nicht wissen, welcher Dateiname der Letzte war, und ob es eventuell ein anderer bereits gelöschter Dateiname war. In diesem Fall war es der bereits gelöschte Dateiname file5.dat.
- Es existiert kein Name mehr für den Metadateneintrag 200, was wiederum bedeutet, dass wir keinen Dateinamen zu diesem Eintrag haben.

Aus diesen Problemen ergibt sich, dass bei einer Wiederherstellung die Zeitinformationen und der Inhalt der Dateien file1.dat und file2.dat nicht stimmen, da diese zu der gelöschten Datei file5.dat gehören, von welcher jedoch der Name nicht mehr existiert. Nachdem auf den Metadateneintrag 200 kein Dateiname mehr zeigt, werden die darin enthaltenen Daten beim Auflisten der Dateien nicht angezeigt. Solche Situationen sind auch der Grund, wieso es in diesem Referenzmodel eine Aufteilung in Metadata und File Name Kategorie gibt.

3.3.3 Applikations-basierte Dateiwiederherstellung

Die Applikations-basierte Dateiwiederherstellung wird auch als *Data Carving* bezeichnet. Dabei macht man sich zunutze, dass viele Dateien mit speziellen Anfangs- und End-Signaturen versehen sind. Als Beispiel hierfür eignen sich sehr gut JPEG-Dateien, bei welchen der Beginn durch den Wert 0xffd8 und das Ende durch 0xffd9 angegeben wird. Werden Programme wie das in Kapitel 2.3.4 vorgestellte Programm `foremost` so konfiguriert, dass

sie nach diesen Signaturen suchen, können aus einer Menge von Rohdaten Dateien mit passender Signatur extrahiert werden.

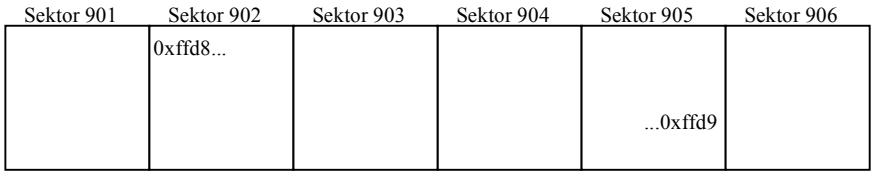


Abb. 3.6 Anfangs- und End-Signatur einer JPEG-Datei in einem Rohdatenstrom [Car05 Figure 8.21 Seite 207]

Diese Methode birgt jedoch auch einige Probleme. Es ist nicht immer sichergestellt, dass alle Daten zwischen zwei Signaturen zusammengehören. So können etwa Lücken mit Daten anderer Dateien zwischen den Datenblöcken der gefundenen Datei liegen. Im schlimmsten Fall kann es sogar vorkommen, dass die beiden gefundenen Signaturen von zwei verschiedenen Dateien stammen und so wie in den anderen Fällen das Ergebnis der Suche nicht fehlerfrei ist.

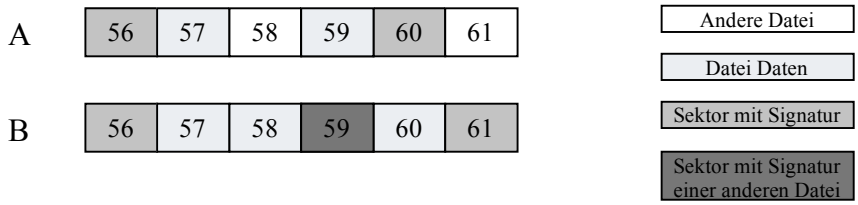


Abb. 3.7 Zwei mögliche Fälle einer fehlerhaften Wiederherstellung

4 Beschreibung ausgewählter Dateisysteme

Aufbauend auf die allgemeine Beschreibung stellt dieses Kapitel Dateisysteme, welche nach ihrer Verbreitung und Relevanz ausgewählt wurden, vor. Den Beginn macht dabei das FAT-Dateisystem, welches nicht nur das Älteste sondern auch das Einfachste der vorgestellten Dateisysteme ist.

4.1 FAT 12/16/32

Das FAT bzw. *File Allocation Table* Dateisystem wurde von Microsoft mit dem Betriebssystem DOS eingeführt und wird seither von jedem Windows und von den meisten Unix Betriebssystemen unterstützt. Einer der wichtigsten Gründe für die Beschäftigung mit FAT ist der, dass dieses Dateisystem auf beinahe jedem mobilen Datenträger, wie USB-Sticks, Speicherkarten für Digitalkameras oder dem sehr beliebten MP3-Player zu finden ist.

4.1.1 Allgemeine Beschreibung

Obwohl es im Laufe der Jahre an neue technische Gegebenheiten angepasst wurde, handelt es sich bei FAT um ein relativ einfaches Dateisystem. Diese Einfachheit kommt von der Verwendung einer nur kleinen Anzahl verschiedener Datenstrukturen. Aufgrund dieser einfachen Strukturen passt es auch nicht ganz in das von Brian Carrier in [Car05 ab Seite 173] beschriebene Model für Dateisysteme, da es zum Beispiel keine Daten für die Application Kategorie gibt.

Wie in der Kapitelüberschrift schon angedeutet, gibt es mehrere Versionen des FAT-Dateisystems. Genauer gesagt sind es drei Versionen, FAT12, FAT16 und FAT32, welche sich, wie in [MS00] beschrieben, in der Anzahl der Bits für die Adressierung, in der Größe der Cluster, so werden in FAT die Dateneinheiten bezeichnet, und der daraus resultierenden Unterstützung von Festplattengrößen unterscheiden. Ein Windows XP System beispielsweise unterstützt bei einer maximalen Clustergröße von 32KB Festplatten bis zu einer Größe von ca. 8 Terabyte [MS03-3] [MS00].

Das Layout des FAT-Dateisystems hat, je nach Version, drei oder vier physikalische Abschnitte. Diese Abschnitte befinden sich in folgender Reihenfolge auf der Festplatte:

0 – Reserved Region

1 – FAT Region

2 – Root Directory Region (ist in FAT32 Laufwerken nicht vorhanden)

3 – File and Directory Data Region

[MS00]

Das konkrete Layout für ein FAT32-System zeigt die folgende Abbildung, wobei hier, wie auch im restlichen Dokument, die Bezeichnung *Area* anstelle von *Region* verwendet wird.

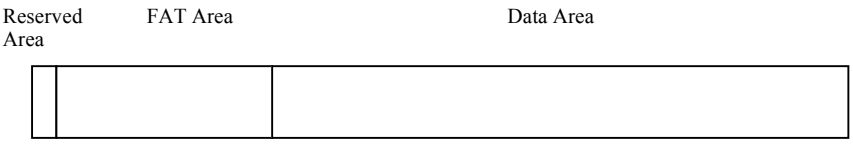


Abb. 4.1 Layout eines FAT32-Systems vgl. [Car05 Figure 9.2 Seite 213]

Die Größe der *Reserved Area* ist im Bootsektor festgelegt, wobei sie in FAT12/16 genau ein Sektor groß ist. In FAT32 werden hingegen mehrere Sektoren dafür verwendet. Die *FAT Area* beginnt unmittelbar nach der *Reserved Area* und beinhaltet die Primäre und die Backup-FAT-Strukturen, wobei sich die Größe aus der Anzahl und der Größe der FAT-Strukturen ergibt. In der *Data Area* befinden sich diejenigen Cluster, welche für das Speichern von Dateien und Ordner verwendet werden.

Das Basiskonzept besteht darin, dass jeder Datei und jedem Ordner eine sogenannte Ordnereintrag (*directory entry*) Datenstruktur zugeordnet wird. In dieser Datenstruktur wird der Name, die Größe, die Startadresse des Dateiinhalts sowie weitere Metadaten gespeichert. Benötigt eine Datei oder ein Ordner mehr als einen Cluster zum Speichern der Daten, so wird für das Auffinden der weiteren Cluster eine Datenstruktur mit dem Namen FAT (*File Allocation Table*) benötigt. Die FAT wird neben dem Auffinden der weiteren Cluster einer Datei auch zum Identifizieren des Zuordnungsstatus der einzelnen Cluster verwendet. Mit dem Model aus Kapitel 3 verglichen, beinhaltet die FAT Daten der Content und der Metadata Kategorie. Die folgende Abbildung zeigt die soeben beschriebenen Zusammenhänge.

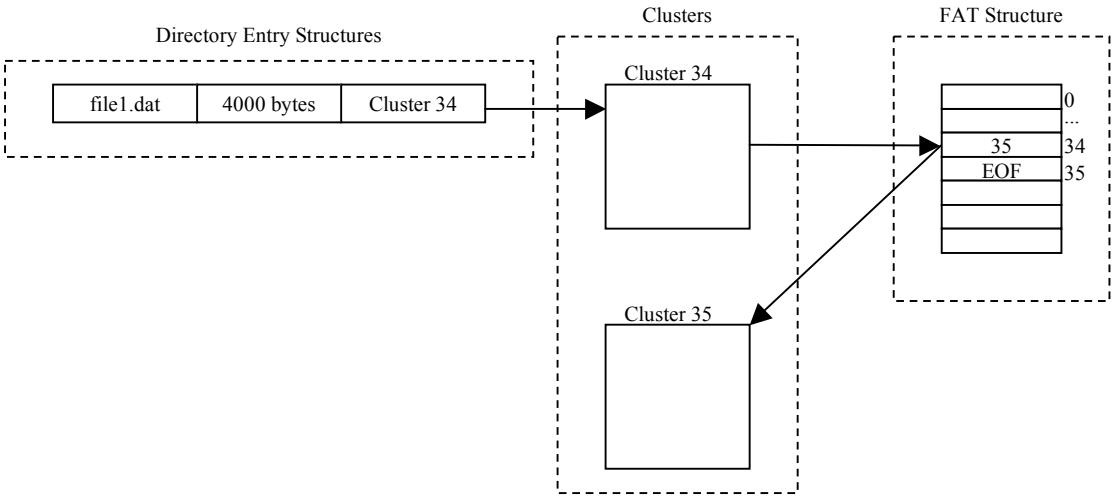


Abb. 4.2 Zusammenhang zwischen Ordnereintrag (directory entry), Cluster und FAT vgl. [Car05 Figure 9.1 Seite 212]

4.1.2 Kategorisierung der Daten

Aufbauend auf den allgemeinen Informationen über das Dateisystem, werden in diesem Kapitel die verschiedenen Daten den einzelnen Kategorien, welche im Kapitel 3.2 beschrieben wurden, zugeordnet und beschrieben.

4.1.2.1 File System Kategorie

Die Daten der File System Kategorie befinden sich in einem FAT-System im ersten Sektor eines Volumes. Dieser Sektor wird von Microsoft als *BIOS Parameter Block (BPB)* bezeichnet [MS00]. Da die Bezeichnung *Bootsektor* viel gebräuchlicher ist, wird sie auch in dieser Arbeit weiter verwendet.

Neben den Daten der File System Kategorie werden noch Daten anderer Kategorien gespeichert und erst später bei diesen Kategorien erläutert.

Unter all diesen Daten wird zum Beispiel die Adresse für die Backupkopie des Bootsektors gespeichert, welche sich laut Dokumentation [MS00] immer im Sektor 6 befindet, und Daten die für die Berechnung des Volumelayouts benötigt werden. Zu diesen Daten zählen die Größe des Bootsektors sowie die Anzahl und die Größe der FAT-Strukturen. Im Unterschied zu FAT12/16 kann in FAT32 die Position und die Größe des Root-Directories frei gewählt werden. Die Angaben darüber befinden sich ebenfalls im Bootsektor, wobei der Beginn des Root-Directories nur in ganz seltenen Fällen, etwa wenn Sektoren in diesem Bereich beschädigt sind, nicht der erste Sektor des Datenbereichs ist. Wichtig ist noch zu erwähnen, dass es in FAT kein Feld gibt, welches das Dateisystem eindeutig identifiziert.

Für eine einfache Veranschaulichung, wie sich die Struktur eines FAT-Volumes anhand der im Bootsektor gespeicherten Daten ergibt, eine Grafik für FAT12/16 und FAT32.

FAT12/16

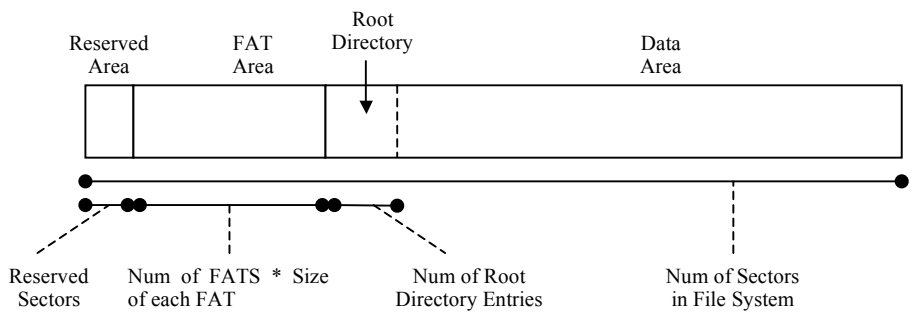


Abb. 4.3 Layout eines FAT12/16-Systems mit den Daten aus dem Bootsektor für die Berechnung der genauen Position. [Car05 Figure 9.3 Seite 215]

FAT32

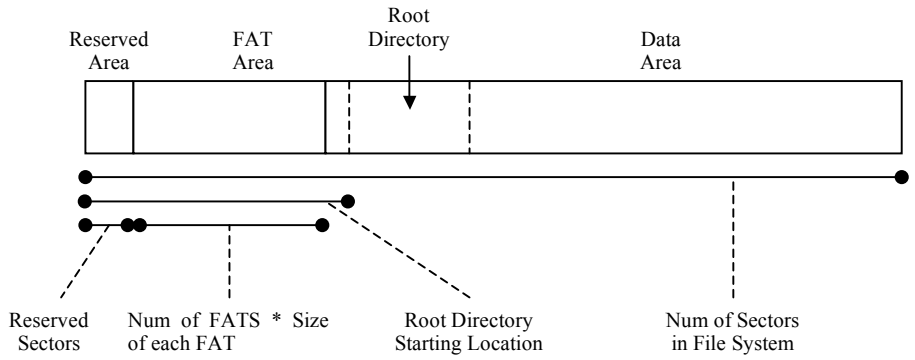


Abb. 4.4 Layout eines FAT32-Systems mit den Daten aus dem Bootsektor für die Berechnung der genauen Position. [Car05 Figure 9.3 Seite 215]

Neben diesen relevanten Daten gibt es noch weitere nicht relevante Daten wie zum Beispiel den OEM-String. Für genauere Informationen über diese Felder wird an dieser Stelle auf die Spezifikation in [MS00] verwiesen.

Bootcode

Ebenfalls im Bootsektor enthalten ist der Bootcode für das Starten des Computers. Dieser ist in zwei Teile aufgeteilt, wobei der erste Teil ein Abschnitt des Bootsektors ist und einen Sprungbefehl in Maschinencode an das Ende der Bootsektordaten enthält, wo sich der zweite Teil des Bootcodes befindet.
Der Bootcode ist auch in nicht startbaren FAT-Volumes vorhanden und enthält Text, der darauf verweist, dass ein anderes Volume für das Starten benötigt wird.

4.1.2.2 Content Kategorie

In dieser Kategorie befinden sich alle Daten, die zu einer Datei oder einem Ordner gehören. Wie schon erwähnt werden in FAT Daten in Cluster gespeichert, wobei ein Cluster eine Menge zusammenhängender Sektoren ist, und die Anzahl der Sektoren pro Cluster eine 2er Potenz sein muss (1, 2, 4, 8, 16, ...). Die maximale Größe eines Clusters ist in [MS00] mit 32KB angegeben.
Im Unterschied zu anderen Dateisystemen werden diese Cluster nur in der *Data Area* des Dateisystems verwendet. In den beiden anderen Bereichen wird jeder Sektor einzeln adressiert. Oder anders formuliert, nicht jede logische Volumeadresse hat eine logische Dateisystemad-

resse. Das und die Tatsache, dass der erste Cluster die Adresse 2 hat, ist der Grund dafür, warum das Auffinden des ersten Clusters sehr kompliziert ist.
Wie in [Car05 Seite 223] ausgeführt, kann von einer Clusteradresse auf eine Sektoradresse und umgekehrt umgerechnet werden. Interessant ist in diesem Zusammenhang, dass Betriebssysteme und andere Programme wegen der einfacheren Handhabe nur die Sektoradressierung verwenden.

Zuteilungsstatus von Cluster

In der FAT wird zu jedem Cluster ein Eintrag gespeichert, wobei der Tabelleneintrag 371 mit dem Cluster 371 korrespondiert. Jeder Tabelleneintrag ist eine Nummer, deren Maximalwert von der Version des Dateisystems abhängt, FAT12 verwendet 12-Bit Werte, FAT16 16-Bit Werte und FAT32 32-Bit Werte, von denen jedoch nur 28 Bit verwendet werden.
Anhand des Wertes eines Eintrags können bestimmte Informationen herausgefunden werden. So bedeutet etwa der Wert 0, dass der Cluster nicht zugeteilt ist, der Wert 0xff7 in FAT12, der Wert 0xfff7 in FAT16 und der Wert 0x0fff fff7 in FAT32, dass dieser Cluster als beschädigt markiert ist und deswegen nicht mehr zugeteilt werden soll. Alle anderen Werte bedeuten, dass dieser Cluster einer Datei oder einem Ordner zugeteilt ist mit der Ausnahme des Wertes 0x0fff ffff für den *end of file* (EOF) Marker. Wobei die Bedeutung der Werte später behandelt wird.
Laut [Car05 Seite 224] wird in Windows 98 und XP für das Auffinden eines nicht zugeteilten Clusters der *next available* Algorithmus verwendet.

4.1.2.3 Metadata Kategorie

In einem FAT-Dateisystem zählen erweiterte Informationen von Dateien oder Informationen über suspekte Dateien, welche in den Ordnereinträgen gespeichert sind, zu dieser Kategorie. Des Weiteren wird die FAT-Struktur für das Speichern von Metadateninformationen über das Layout von Dateien und Ordner verwendet.

Ordnereinträge / Directory Entries

Ein Ordnereintrag ist eine 32-Byte Datenstruktur, welche für jede neu erstellte Datei und jeden neuen Ordner in dessen Überordner angelegt wird. In diesem Eintrag werden alle Attribute, die Größe, der Startcluster sowie Datum und Uhrzeit gespeichert. Da auch die Dateinamen in den Ordnereinträgen gespeichert werden, zählen diese ebenfalls zur *Filename* Kategorie.

Ordnereinträge werden in den Clustern, welche einem Ordner zugeteilt sind, gespeichert und befinden sich deswegen im gesamten Datenbereich, wobei Ordner in einem FAT-System wie spezielle Dateien behandelt werden. Für die Identifizierung der einzelnen Ordnereinträge gibt es keine eindeutige Nummer, sondern nur den vollen Namen und das Wissen über die Position der Daten aufgrund der fixen Größe.
Im Attributfeld werden alle sieben möglichen Attribute gespeichert, wobei von diesen die Attribute *directory*, *long file name* und *volume label* essenziell und die Attribute *read only*, *hidden*, *system* und *archive* nicht essenziell sind und deswegen nicht von jeder FAT-Implementierung beachtet werden.
In jedem Ordnereintrag werden drei verschiedene Zeiten gespeichert. Bei diesen Zeiten handelt es sich um die Erstellzeit, die Zeit des letzten Zugriffs und die Zeit des letzten Schreibzugriffs. Wobei hier zu beachten ist, dass jede dieser Zeiten eine unterschiedliche Genauigkeit besitzt, die für die Erstellzeit Zehntelsekunden, Tage für die letzte Zugriffszeit und zwei Sekunden für die Zeit des letzten Schreibzugriffs, welche auch die einzige Zeit ist die angegeben werden muss. Jedoch gibt es keine Vorschriften, wann diese Zeiten aktualisiert werden müssen, so hat jede Implementierung seine eigene Vorgehensweise. Eine genaue Beschreibung des Zeitformates findet sich in der FAT-Spezifikation [MS00] und in [Car05 Seite 263/264].
Für forensische Untersuchungen ist von Bedeutung, dass alle Zeiten in der lokalen Zeitzone gespeichert werden und deswegen kein Umrechnen erforderlich ist.
Der Zuteilungsstatus eines Ordnereintrags wird über das erste Byte, welches das erste Zeichen des Dateinamens enthält, bestimmt. Ist ein Ordnereintrag als nicht zugeteilt markiert, so hat das erste Byte den Wert 0xe5. Dies ist auch der Grund, warum bei einer Wiederherstellung gelöschter Dateien das erste Zeichen des Dateinamens manuell ausgewählt werden muss.

Große Dateien

Wie bereits erwähnt wird in der FAT zu jedem Cluster ein Wert gespeichert. Soll nun eine Datei gespeichert werden, deren Größe ein Vielfaches der Clustergröße ist, so wird ausgehend vom Startcluster, dessen Adresse im Ordnereintrag gespeichert ist, bei jedem Cluster in der FAT die Adresse des nachfolgenden Clusters gespeichert. Beendet wird diese Verkettung, welche auch Clusterkette (*cluster chain*) genannt wird, mit dem Wert für das *End of File* (EOF) Symbol. Nachdem, wie bereits erwähnt, in FAT32 nur 28 Bit für die Clusteradresse verwendet werden, ist die maximale Anzahl von Cluster in einer Kette 268.435.456, wobei hier die Werte für EOF, der Nullwert und der Wert für die Markierung als beschädigter Cluster noch enthalten sind. Da in jedem Ordnereintrag die Größe einer Datei gespeichert

wird, ist die maximale Größe einer Datei von der Anzahl der Bits dieses Feldes abhängig. Nachdem für dieses Feld 4 Byte im Ordnereintrag reserviert sind, sind Dateien auf eine Größe von 4 GB beschränkt.

Anlegen von Ordnern

Wird ein neuer Ordner angelegt, so wird diesem ein Cluster zugeordnet und dessen Inhalt mit 0en überschrieben. Da bei einem Ordner das Feld für die Größe im dazugehörigen Ordnereintrag nicht verwendet wird, sollte dessen Wert ebenfalls 0 sein. Mit dem Anlegen des Ordners werden ebenfalls die zwei ersten Einträge angelegt. Bei diesen Einträgen handelt es sich um die . und .. Ordner, welche für das Navigieren mit der Kommandozeile benötigt werden.
Diese Einträge werden auch für die Überprüfung des Erstelldatums verwendet, da es gleich mit dem des Ordners sein muss.
Für das Berechnen der Ordnergröße muss durch die FAT-Struktur, ausgehend vom Startcluster bis zum Erreichen der EOF-Markierung navigiert werden.

4.1.2.4 File Name Kategorie

In dieser Kategorie wird typischerweise einem Dateinamen eine Adresse seiner Metadatenstruktur zugewiesen. Nachdem in FAT der Dateiname aber ein Teil der Metadaten ist, gibt es in dieser Kategorie nur zu erwähnen, in welchem Format die Dateinamen gespeichert werden.
Das Standardformat für Dateinamen ist die 8.3 Benennungskonvention bei der 8 Zeichen für den Namen und 3 Zeichen für die Dateiendung verwendet werden können. Benötigt die Benennung einer Datei mehr Zeichen, so gibt es die Möglichkeit eines langen Dateinamens. Hat eine Datei einen Namen, der länger als die 8 Zeichen des Standardnamens ist, so wird ein eigener Ordnereintrag für diesen langen Namen im Ordner der Datei angelegt. Dieser Ordnereintrag beinhaltet neben den Daten für den Namen noch ein Attribut für Sequenznummer und eine Prüfsumme, welche für die Zusammengehörigkeit von langem und kurzem Dateinamen verwendet werden kann. Pro Ordnereintrag eines langen Namens können 13 UTF-16 Zeichen verwendet werden. Sollten diese Zeichen nicht ausreichen, so wird ein neuer Ordnereintrag für weitere 13 Zeichen angelegt.
Jede Datei mit einem langen Namen besitzt auch einen kurzen Namen, da in den Ordnereinträgen für den langen Namen keine weiteren Dateiattribute gespeichert sind.

4.1.2.5 Beschreibung der Datenstrukturen

Die Beschreibung der verwendeten Datenstrukturen eines FAT-Systems wie den Bootsektor, die FAT, die FSINFO unter FAT32, die Ordnereinträge und Ordnereinträge für die langen Dateinamen würde für den Umfang dieser Arbeit etwas zu weit gehen. Für interessierte Leser sei an dieser Stelle auf die FAT-Spezifikation [MS00] und auf die ausführliche Beschreibung von Brian Carrier in [Car05 Seite 253] Kapitel 10 „FAT Data Structures“ verwiesen.

4.1.3 Möglichkeiten der Datenwiederherstellung

Dieser Abschnitt befasst sich mit den Möglichkeiten, die es gibt, um gelöschte Dateien wiederherzustellen. Doch vor dem Wiederherstellen muss zuerst geklärt werden, wie in einem FAT-System Dateien und Ordner gelöscht werden, um zu wissen, wo bei der Wiederherstellung angesetzt werden muss.

4.1.3.1 Löschen von Dateien

Wie schon aus Kapitel 4.1.2.3 bekannt, werden die Metadaten einer Datei in einem Ordnereintrag gespeichert. Wird nun eine Datei gelöscht, so wird der Wert des ersten Bytes dieses Ordnereintrages auf den Hex-Wert 0xe5 gesetzt. Durch dieses setzten wird dieser Ordnereintrag auf ‚nicht zugeteilt’ gesetzt und der erste Buchstabe des kurzen Dateinamens gelöscht. Weiters werden ausgehend vom ersten Cluster, welcher in den Metadaten angegeben ist, alle Cluster dieser Datei in der FAT mit dem Wert 0 überschrieben und so als nicht zugeteilt markiert. Wie in [Car05 Kapitel 9] angegeben, werden die verwendeten Cluster selbst nach dem Löschen von Windows normalerweise nicht überschrieben und beinhalten so noch die Daten der gelöschten Datei.
Wurde für eine Datei ein langer Dateiname verwendet, so wird bei diesem Ordnereintrag ebenfalls das erste Byte mit 0xe5 und so die Sequenznummer dieses Eintrags überschreiben.

4.1.3.2 Wiederherstellung von Daten

Die Beschreibung, wie gelöschte Daten eines FAT-Laufwerks wiederhergestellt werden können, geht von der Annahme aus, dass für das Löschen ein Windows-System verwendet wurde. Ausgehend von dieser Annahme wird zuerst der als ‚nicht zugeteilt’ markierte Ordnereintrag der wiederherzustellenden Datei benötigt. In diesem Eintrag findet sich die Adresse des ersten Clusters dieser Datei. Nachdem beim Löschen der Zuteilungsstatus der Cluster in der FAT auf ‚nicht zugeteilt’ gesetzt wurde, gibt es zwei Möglichkeiten, die weiteren Cluster einer Datei zu finden. Die erste Möglichkeit beginnt beim ersten Cluster und nimmt solange den nächsten Cluster, ohne auf dessen Zuteilungsstatus zu achten, bis die im Ordnereintrag angegebene

Dateigröße erreicht ist. Die zweite Möglichkeit berücksichtigt den Zuteilungsstatus und verwendet nur ‚nicht zugeteilte' Cluster für die Wiederherstellung.
Damit die Unterschiede der beiden Möglichkeiten besser verständlich werden, ein kleines Beispiel aus [Car05 Seite 247/248] welches die Stärken und Schwächen beider Möglichkeiten zeigt.
In Abb. 4.5 werden drei Szenarien für die Positionierung und Wiederherstellung einer Datei angegeben.

Start-Cluster: 56
Dateigröße: 7094 Bytes
Clustergröße: 2048 Bytes

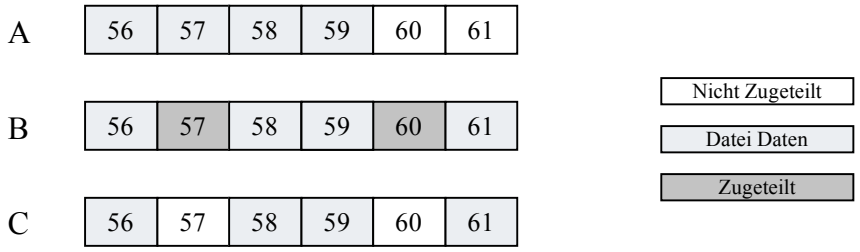


Abb. 4.5 Drei Szenarien für die Aufteilung einer Datei auf mehrere Cluster vgl. [Car05 Figure 9.19 Seite 248]

Die Aufteilung in Abb. 4.5 A führt, da alle Cluster aufeinander folgen, mit beiden Möglichkeiten zum Erfolg. In Abb. 4.5 B wird es hingegen schon schwieriger, da hier der Zuteilungsstatus der Cluster berücksichtigt werden muss, um die Datei richtig wiederherzustellen. Das in Abb. 4.5 C angegeben Szenario ist mit keiner der beiden Möglichkeiten zu lösen, da hier der nicht zugeteilte Cluster 57 immer zur wiederhergestellten Datei gegeben wird.
Sollen neben Dateien auch gelöschte Ordner gefunden werden, so muss der nicht zugeteilte Speicher nach Ordnereinträgen durchsucht werden. Hierfür ist es hilfreich zu wissen, dass jeder Ordner Ordnereinträge für den . und .. Ordner, wie bereits in Kapitel 4.1.2.3 erwähnt, beinhaltet. Mit diesem Wissen ergeben sich die Suchkriterien für das Auffinden der Ordnerstrukturen, welche durch die Struktur der Ordnereinträge erkannt werden können.
Schwierigkeiten bereiten hierbei nur die Möglichkeiten, dass der Cluster eines Ordners bereits von einem anderen Ordner oder einer Datei überschrieben wurde, oder, dass der Ordner im Laufe seiner Existenz weitere Cluster benötigte.
Erleichterungen für das Auffinden gelöschter Dateien bietet das regelmäßige Verwenden von Defragmentierungsprogrammen, da diese versuchen, die Daten einer Datei in zusammenhängende Cluster zu verschieben. Natürlich gelten diese Erleichterungen nur für Dateien, die erst

nach dem letzten Defragmentieren gelöscht wurden. Dateien, die vor dem Defragmentieren gelöscht wurden, können hingegen sehr schwer wiederhergestellt werden, da deren Cluster sehr wahrscheinlich bereits überschrieben wurden.

4.2 NTFS

NTFS (*New Technologie File System*) wurde ebenfalls von Microsoft entwickelt, mit Windows NT eingeführt und ist seit Windows XP das Standard-Dateisystem von Windows. Bei der Entwicklung wurde speziell auf Zuverlässigkeit, Sicherheit und die Unterstützung von großen Laufwerken, sowie das schnelle Durchführen von Standarddateioperationen wie Lesen, Schreiben und Suchen, geachtet.
In diesem Zusammenhang ist jedoch zu beachten, dass es von Microsoft keine offizielle Dokumentation von NTFS gibt, die genau beschreibt, wie das Dateisystem aufgebaut ist und wie bzw. wo die Daten auf der Festplatte gespeichert werden. Interessierte Leser werden für weitere Informationen über das angenommene Layout der Daten auf der Festplatte auf die Dokumentation des Linux NTFS-Projektes [Linux-NTFS] verwiesen.
Das Fehlen dieser Dokumentation ist auch der Grund, weshalb es außer Windows kein anderes System gibt, welches sicher auf eine NTFS formatierte Festplatte schreiben kann. Erschwerend kommt noch hinzu, dass Microsoft mit beinahe jeder neuen Version von Windows auch Änderungen am Dateisystem vornimmt.
Trotz dieser Hürden verspricht die aktuelle Version der NTFS-Treiber des Linux NTFS-Projektes eine volle Lese und Schreibunterstützung [Linux-NTFS].

4.2.1 Allgemeine Beschreibung

Die allgemeinen Beschreibungen werden in zwei Hauptteile unterteilt, wobei sich der erste Teil mit dem Vorstellen und Beschreiben allgemeiner NTFS-Strukturen beschäftigt und der zweite Teil NTFS im Bezug auf das allgemeine Model für Dateisysteme von Brian Carrier aus [Car05 Kapitel 8] erklärt.

4.2.1.1 Allgemeine NTFS-Strukturen

NTFS ist ein sehr komplexes Dateisystem bei dem als wichtigstes Designkonzept alle Daten einzelnen Dateien zugeordnet sind. Diese Dateien können über die gesamte Partition verteilt gespeichert werden, weshalb ein NTFS formatiertes Laufwerk keine spezielle Struktur aufweist. Im Unterschied zu anderen Dateisystemen werden die Daten für die Dateisystemverwaltung nicht speziell versteckt. Das einzige was alle NTFS-Partitionen gemein haben, ist die Position des Bootsektors und des Bootcodes im ersten Sektor, sowie die Backupkopie des Bootsektors im letzten Sektor der Partition. Wie in FAT werden auch in NTFS Cluster, welche eine Menge zusammenhängender Sektoren sind, verwendet.

MFT

Das wichtigste Konzept in NTFS ist jenes der *Master File Table* (MFT). Diese Tabelle besteht aus einfachen Einträgen, deren Größe im Bootsektor angegeben wird. Es wird in Windows jedoch eine fixe Größe von 1024 Bytes verwendet. Die ersten 42 Bytes eines MFT-Eintrages bestehen aus 12 definierten Feldern, die restlichen 982 Bytes besitzen keine Struktur und können mit Attributen aufgefüllt werden. Diese Attribute sind kleine Datenstrukturen für spezielle Aufgaben. So wird zum Beispiel ein Attribut für das Speichern des Dateinamens oder ein anderes für das Speichern des Dateiinhaltes verwendet. Abb. 4.6 zeigt einen allgemeinen MFT-Eintrag. Wie ein konkretes Beispiel eines kleinen Ordners bzw. einer Datei aussehen kann, zeigt Abb. 4.7.

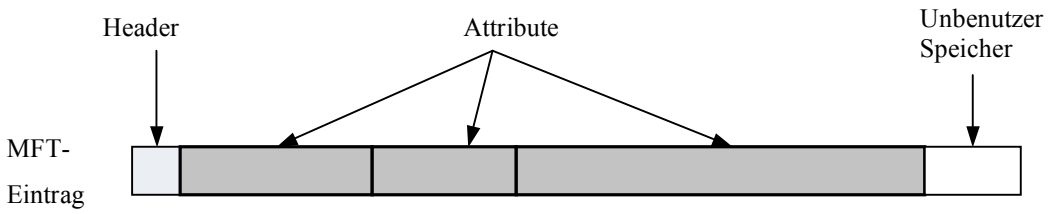


Abb. 4.6 Allgemeine Struktur eines MFT-Eintrags mit einem kleinen Header und mehreren Attributen. vgl. [Car05 Figure 11.1 Seite 275]

Standard Informationen	Datei- oder Ordner-Name	Sicherheits-beschreibung	Daten oder Index	

Abb. 4.7 Beispiel einer kleinen Datei oder eines kleinen Ordners [NTFS.com/ntfs-mft.htm]

Nachdem alle Daten in NTFS einer Datei zugeordnet sind, ist auch die MFT in einer Datei gespeichert. Damit diese Datei gefunden wird, muss im Bootsektor die Adresse des ersten MFT-Eintrags gesucht werden. In diesem ersten Eintrag, mit der Bezeichnung $MFT, sind die weiteren Speicherplätze der MFT gespeichert. Zu beachten ist hier, dass dieser Eintrag die einzige Stelle im Dateisystem ist, wo die Position der MFT gespeichert wird. Abb. 4.8 zeigt, wie die MFT über den Bootsektor gefunden wird und welche Struktur diese besitzt.

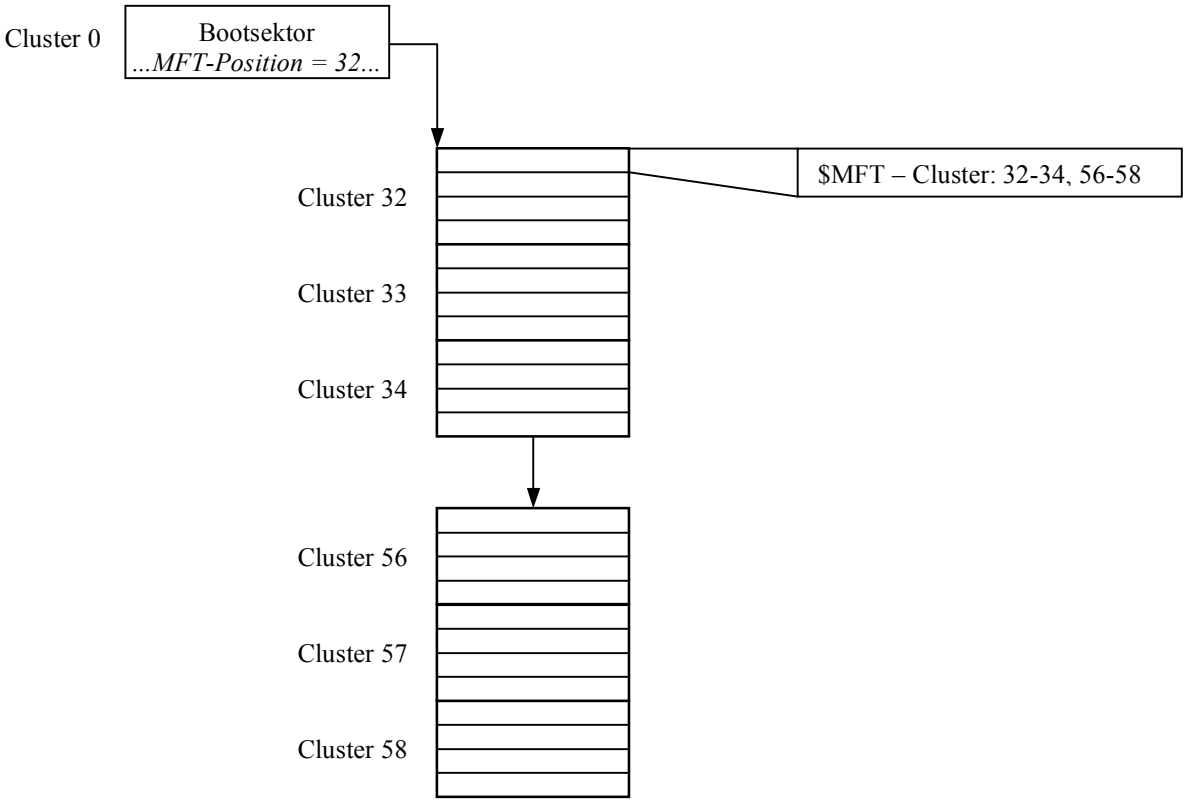


Abb. 4.8 Stilisierte Position und Struktur der MFT aus den Informationen des Bootsektors und der $MFT Eintrags. vgl. [Car05 Figure 11.2 Seite 275]

Die Größe des für die MFT reservierten Speichers könnte theoretisch fix angegeben werden, wird in Windows jedoch dynamisch verwaltet, damit das Dateisystem leichter durch eine Volumevergrößerung vergrößert werden kann. In diesem Zusammenhang ist zu beachten, dass die MFT-Einträge von Windows einmal angelegt und niemals gelöscht, sondern höchstens überschrieben werden. Daher kann die MFT nur vergrößert und nicht verkleinert werden.

Für die Adressierung der einzelnen MFT-Einträge wird eine sequenzielle 48-Bit Adresse, welche beim ersten Eintrag mit 0 beginnt und als Dateinummer (*file number*) bezeichnet wird, verwendet. Die maximale MFT-Adresse ist von der Größe der MFT abhängig und ist das Ergebnis aus MFT-Größe dividiert durch die Größe der MFT-Einträge. Zu dieser Adresse hat jeder MFT-Eintrag noch eine 16-Bit Sequenznummer. Wird ein MFT-Eintrag zum ersten Mal einer Datei oder einem Ordner zugeteilt, so wird diese Nummer mit 0 initialisiert. Diese Sequenznummer wird jedes Mal, wenn ein schon einmal verwendeter MFT-Eintrag wieder einer Datei oder einem Ordner zugeteilt wird, um 1 erhöht. Die Kombination der Sequenznummer mit der MFT-Adresse ergibt eine 64-Bit Dateireferenzadresse (*file reference address*), die für das Referenzieren der MFT-Einträge in NTFS verwendet wird und bei der die Sequenznummer für die obersten 16 Bit Verwendung findet.

Dateisystem-Metadaten

Wie bereits beschrieben, werden in NTFS alle Daten in Dateien gespeichert, was bewirkt, dass sich die Metadaten des Dateisystems ebenfalls in eigenen Dateien, welche von Microsoft als Metadatendateien (*metadata files*) bezeichnet werden, befinden. Für diese Dateien werden von Microsoft die ersten 16 MFT-Einträge reserviert [ntfs.com/ntfs-mft.htm] [ntfs.com/ntfs-system-files.htm]. Bei genauerer Betrachtung der MFT stellt sich jedoch heraus, dass der erste Eintrag eines Ordners oder einer allgemeinen Datei erst im MFT-Eintrag 24 gespeichert wird. Die nicht benötigten reservierten MFT-Einträge werden zwar zugeteilt, enthalten jedoch nur Standardattribute ohne Daten.

Die Metadatendateien befinden sich im Root-Directory und sind für normale Benutzer nicht sichtbar. Die Benennung dieser Dateien hat ein eigenes Schema: Das erste Zeichen ist ein $ und darauf muss ein Großbuchstabe folgen. Tab. 4.1 listet alle Metadatendateien mit ihrem dazugehörigen MFT-Eintrag auf.

MFT-Eintrag	Dateiname	Beschreibung
0	$MFT	Eintrag für die MFT
1	$MFTMirr	Enthält eine Backup Kopie der ersten MFT Einträge.
2	$LogFile	Enthält das Journal mit Logging-Informationen über Metadaten-Transaktionen.
3	$Volume	Enthält Informationen über das Volume, wie Bezeichnung, Version und darüber wie das Volume identifiziert werden kann.
4	$AttrDef	Enthält die Attributinformationen wie Werte für das Identifizieren, Name und Größen.
5	.	Eintrag für das Root-Directory des Dateisystems
6	$Bitmap	Enthält den Zuteilungsstatus aller Cluster des Dateisystems.
7	$Boot	Beinhaltet den Bootsektor und den Bootcode des Dateisystems.
8	$BadClus	Liste aller Cluster, welche fehlerhafte Sektoren besitzen.
9	$Secure	Beinhaltet Informationen über die Sicherheit und die Zugriffskontrolle der Dateien. Diese Datei wurde erst mit Windows 2000 eingeführt.
10	$Upcase	Enthält alle großgeschriebenen Unicodezeichen
11	$Extend	Ein Ordner welcher Dateien für optionale Erweiterungen, welche nicht in den reservierten MFT-Einträgen gespeichert werden, enthält.

Tab. 4.1: NTFS Metadatendateien mit ihrem MFT-Eintrag vgl. [Car05 Table 11.1 Seite 278]

4.2.1.2 Konzept der MFT-Attribute

Wie zuvor schon erwähnt, besitzt ein MFT-Eintrag nicht viel Struktur, sondern speichert lediglich eine Menge von Attributen. Von diesen Attributen existieren eine große Menge unterschiedlicher Attribute mit jeweils eigener Datenstruktur. Wie bereits in Abb. 4.7 gezeigt, gibt es beispielsweise Attribute für den Dateinamen, die Sicherheitsbeschreibungen oder den Dateiinhalt. Dies ist einer der wesentlichen Unterschiede zu anderen Dateisystemen, denn in NTFS werden nicht die Dateidaten sondern Attribute gelesen oder geschrieben, wobei eines dieser Attribute auch Dateidaten beinhalten kann.
Nachdem jeder Attributtyp unterschiedliche Arten von Daten speichert, werden die Daten in zwei Teile, den Header, welcher generisch und bei jedem Attribut gleich ist, und den Content, der die Daten des Attributes enthält, aufgeteilt.

Attribut-Header

Im Header wird der Typ, angegeben durch eine eindeutige Nummer, der Name und die Größe des Attributes gespeichert. Ebenfalls enthalten sind Informationen darüber, ob die Daten komprimiert oder verschlüsselt sind. In jedem MFT-Eintrag können mehrere Attribute desselben Typs, für die Unterscheidung mit einem eindeutigen Identifier belegt, enthalten sein. Details über die verwendete Datenstruktur des Headers können in [Car05 Kapitel 13, Seite 355] nachgelesen werden.

Attribut-Content

Für den Content gibt es keine vorgeschriebene Struktur oder Größe, so kann z. B. der Inhalt einer Datei von einigen KB bis zu mehreren GB reichen. Da jedoch ein MFT-Eintrag nur 1024 Bytes groß ist, werden in NTFS zwei mögliche Speicherorte angeboten. Befinden sich alle Daten im MFT-Eintrag, so werden diese *resident* Attribute genannt. *Non-resident* Attribute werden jene genannt, bei denen die Daten in externen Clustern im Dateisystem gespeichert werden. Um welches Attribut es sich handelt, wird im Header angegeben. Bei einem *resident* Attribut folgen nach dem Header die Daten, bei einem *non-resident* Attribut ist im Header die Clusteradresse gespeichert.
Benötigen die Daten mehrere Cluster, so werden sie in sogenannten *cluster runs* gespeichert. Ein *cluster run* ist eine Menge zusammenhängender Cluster, für welche die Startadresse und die Länge angegeben wird. Wie in Abb. 4.9 zu sehen ist, können mehrere *cluster runs* pro Attribut angegeben werden.

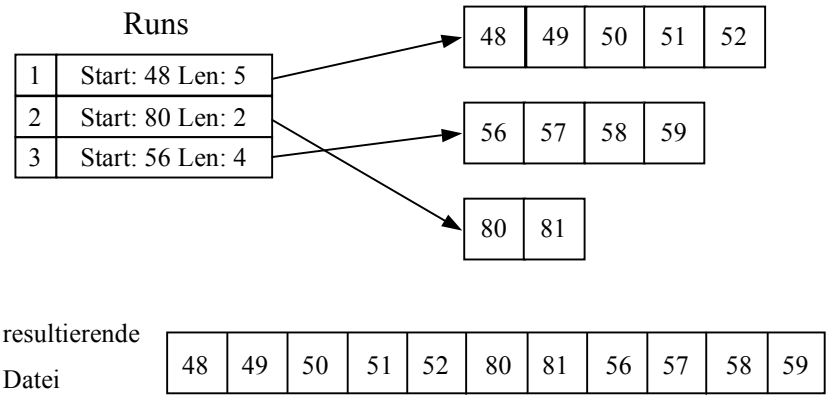


Abb. 4.9 Beispiel wie mehrere *cluster runs* einer Datei gespeichert sind und wie die daraus resultierende Datei zusammengesetzt wird vgl. [Car05 Figure 11.6 Seite 281]

In den Kapiteln 3.2.2 und 3.2.3 sowie in der Abb. 3.3 wurde bereits das Konzept der unterschiedlichen logischen Adressen von Clustern vorgestellt. In NTFS werden diese jedoch etwas anders benannt. So wird die logische Dateisystemadresse als *Logical Cluster Number* / logische Clusternummer (LCN) und die logische Dateiadresse als *Virtual Cluster Number* / virtuelle Clusternummer (VCN) bezeichnet.

Für das Beschreiben der *cluster runs* wird in NTFS VCN zu LCN Mapping verwendet. Für das Beispiel in Abb. 4.9 bedeutet dies etwa, dass die VCN 0 bis 4 auf die LCN 48 bis 52 gemappt werden, usw.

Details über die verwendete Datenstruktur eines *cluster runs* sind in [Car05 Kapitel 13, Seite 357-359] zu finden.

4.2.1.3 Standardattributtypen

Nach der allgemeinen Beschreibung von Attributen geht dieses Kapitel auf die Standardattribute ein, und zeigt an einer Auswahl, wie diese verwendet werden. Wie bereits in Kapitel 4.2.1.2 dargestellt, besitzt jedes Attribut eine eindeutige Nummer. Diese Nummer wird für die Sortierung der Attribute in den MFT-Einträgen verwendet. Zu dieser Nummer kommt noch ein Name, der mit einem ‚$' Zeichen beginnt und nur Großbuchstaben verwendet. In Tab. 4.2 werden einige der Defaultattribute aufgelistet, wobei nicht alle dieser Attribute immer Verwendung finden.

Typ Identifier	Name	Beschreibung
16	$STANDARD_ INFORMATION	Allgemeine Informationen wie die Zugriffszeiten, Besitzer, Sicherheits-ID und Flags.
32	$ATTRIBUTE_LIST	Liste über die Speicherorte anderer Attribute dieser Datei.

48	$FILE_NAME	Dateiname in Unicode sowie die Zugriffszeiten
64	$VOLUME_VERSION	Informationen über das Volume. Existiert nur in Version 1.2 (Windows NT)
64	$OBJECT_ID	Ein eindeutiger 16-Byte Identifier für Dateien und Ordner, welchen es erst seit Version 3.0 (ab Windows 2000) gibt.
80	$SECURITY_ DESCRIPTOR	Zugriffs- und Sicherheitseigenschaften (access controll and security properties)
96	$VOLUME_NAME	Name des Volumes
112	$VOLUME_ INFORMATION	Version des Dateisystems und andere Flags
128	$DATA	Dateiinhalt
144	$INDEX_ROOT	Root-Knoten eines Index-Trees
160	$INDEX_ ALLOCATION	Knoten des Index-Trees
176	$BITMAP	Bitmap für die $MFT Datei und für Indizes
192	$SYMBOLIC_LINK	Softlink-Informationen; nur in Version 1.2 (Windows NT) vorhanden.
192	$REPARSE_POINT	Beinhaltet Daten über einen *reparse point*, welcher als Softlink ab Version 3.0 (ab Windows 2000) verwendet wird.
208	$EA_INFORAMTION	Wird für die Abwärtskompatibilität zu OS/2 verwendet (HPFS)
224	$EA	Wird für die Abwärtskompatibilität zu OS/2 verwendet (HPFS)
256	$LOGGED_UTILITY_STR EAM	Beinhaltet Schlüssel und Informationen über verschlüsselte Attribute ab Version 3.0 (Windows 2000)

Tab. 4.2 Defaultattribute von MFT Einträgen vgl [Car05 Table 11.2 Seite 282]

Nachdem das $STANDARD_INFORMATION Attribut Daten für die Durchsetzung der Datensicherheit und der Quotas beinhaltet, befindet es sich in jedem MFT-Eintrag. Es handelt sich bei diesen Daten jedoch um Daten eines Application-Level Features. So wie das $FILE_ NAME Attribut wird auch das $STANDARD_INFORMATION Attribut immer *resident* im MFT-Eintrag gespeichert.

Handelt es sich um eine Datei, so wird das Attribut $DATA für das Speichern des Dateiinhaltes verwendet und beim Erzeugen der Datei erstellt. Dieses Attribut besitzt normalerweise keinen Namen, außer es ist mehrmals vorhanden. In diesem Fall spricht man auch von *alternate data stream* (ADS). Diese ADS müssen jedoch einen Namen, welcher sich vom Attri-

butnamen unterscheiden kann, tragen. Zum Beispiel ist der Name des Attributtyps $DATA und der Name des Attributes ist „Test ADS“.
Alle Ordner enthalten das $INDEX_ROOT Attribut, in welchem Informationen über Dateien und Unterverzeichnisse dieses Ordners gespeichert sind. Große Ordner verwenden weiters die Attribute $INDEX_ALLOCATION und $BITMAP zum Speichern von Informationen. Typischerweise haben die Attribute $INDEX_ROOT und $INDEX_ALLOCATION den Namen „$I30“.
Da, wie bereits erwähnt, in NTFS alles eine Datei ist, kann auch ein Ordner das Attribut $DATA enthalten und darin beliebige Daten speichern.
Nach der Beschreibung können nun die Attribute der Abb. 4.6 benannt und deren Typ angegeben werden.

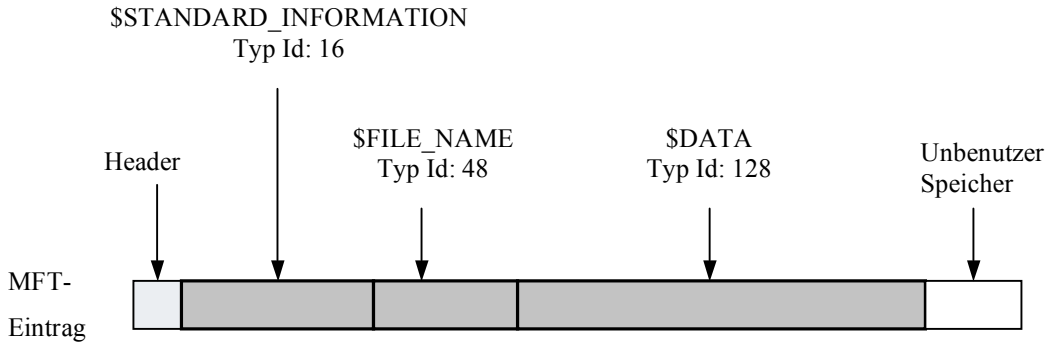


Abb. 4.10 MFT-Eintrag einer sehr kleinen Datei mit benannten Attributen vgl. [Car05 Figure 11.7 Seite 283]

4.2.1.4 Weitere Attributkonzepte

Dieses Kapitel widmet sich weiteren Arten von Attributen, welche nicht zu den Standardattributen zählen.

Basis MFT-Einträge

Wegen des 16-Bit Identifier für Attribute ist es möglich bis zu 65.536 Attribute pro Datei anzugeben. Dabei tritt jedoch das Problem auf, dass zu jedem Attribut zumindest der Header im MFT-Eintrag gespeichert werden muss. Doch die Größe eines MFT-Eintrages ist mit 1.024 Bytes beschränkt. Für die Lösung dieses Problems wird, falls benötigt, ein weiterer MFT-Eintrag zu der betreffenden Datei hinzugefügt und der erste Eintrag zu einem sogenannten *Basis* oder *base* MFT-Eintrag geändert. Wobei dessen Adresse in einem Feld der weiteren Einträge gespeichert wird.

Der Basiseintrag erhält ein $ATTRIBUTE_LIST Attribut, in welchem eine Liste der Attribute mit der dazugehörigen Adresse des MFT-Eintrags gespeichert ist. Ein weiterer Unterschied dieser MFT-Einträge besteht darin, dass die Attribute $FILE_NAME und $STANDARD_INFORMATION nur im Basiseintrag vorhanden sind.

Verschlüsselte Attribute

Eine erweiterte Funktion, die in NTFS auf Attribute angewendet werden kann, ist die Verschlüsselung. Theoretisch könnte diese Funktion auf jedes Attribut angewendet werden, wird von Windows jedoch auf das $DATA Attribut beschränkt.
Wird die Funktion der Verschlüsselung verwendet, so wird nicht das gesamte Attribut, sondern nur der Inhalt ohne dem Header verschlüsselt. Weiters wird für diese Dateien ein $LOGGED_UTILITY_STREAM Attribut, in welchem der Schlüssel für das Entschlüsseln der Daten gespeichert ist, angelegt.
Benutzer können in Windows zwischen der Verschlüsselung von Dateien oder Ordner wählen. Wobei bei der Verschlüsselung eines Ordners nicht der Ordner selbst, sondern nur die Dateien und Ordner, welche in diesem Ordner angelegt werden, verschlüsselt werden. Jeder dieser Ordner und Dateien hat ein spezielles Flag im $STANDARD_INFORMATION Attribut und jedes Attribut hat im Header ein Flag, welches die Verschlüsselung anzeigt.

Implementierung in NTFS

Damit Daten verschlüsselt werden können, ist logischerweise ein Schlüssel notwendig. Diese Anforderung wird in NTFS gelöst, indem für jeden Eintrag in der MFT ein eigener zufälliger Schlüssel, *file encryption key* (FEK), generiert wird. Dieser FEK wird anschließend für die Verschlüsselung mit dem symmetrischen DESX-Algorithmus[9] verwendet. Wobei hier seit Windows XP auch der 3DES und seit Windows XP SP1 der AES_256 verwendet werden kann [MS07-4]. Enthält eine Datei mehrere $DATA Attribute, so werden alle mit dem gleichen FEK verschlüsselt. Für jeden Benutzer, der Zugriffsrechte auf diese Datei hat, wird im $LOGGED_UTILITY_STREAM Attribut ein *data decryption field* (DDF) gespeichert. In diesem Feld wird die *Security ID* (SID) des Benutzers, Verschlüsselungsinformationen und der FEK, welcher selbst mit dem Public Key des Benutzers verschlüsselt wurde, gespeichert. In diesem Attribut werden ebenfalls sogenannte *data recovery fields* (DRF) gespeichert. Diese DRF werden für alle Wiederherstellungsmethoden angelegt und enthalten den, mit dem Public Key für die Datenwiederherstellung verschlüsselten, FEK.

[9] DESX = DES Erweitert um whithening vgl. [Rog96] [Kil97] sowie [Sch07] [Sch05] [Buc03]

Abb. 4.11 zeigt schematisch die Funktionsweise des Verschlüsselungsvorganges.

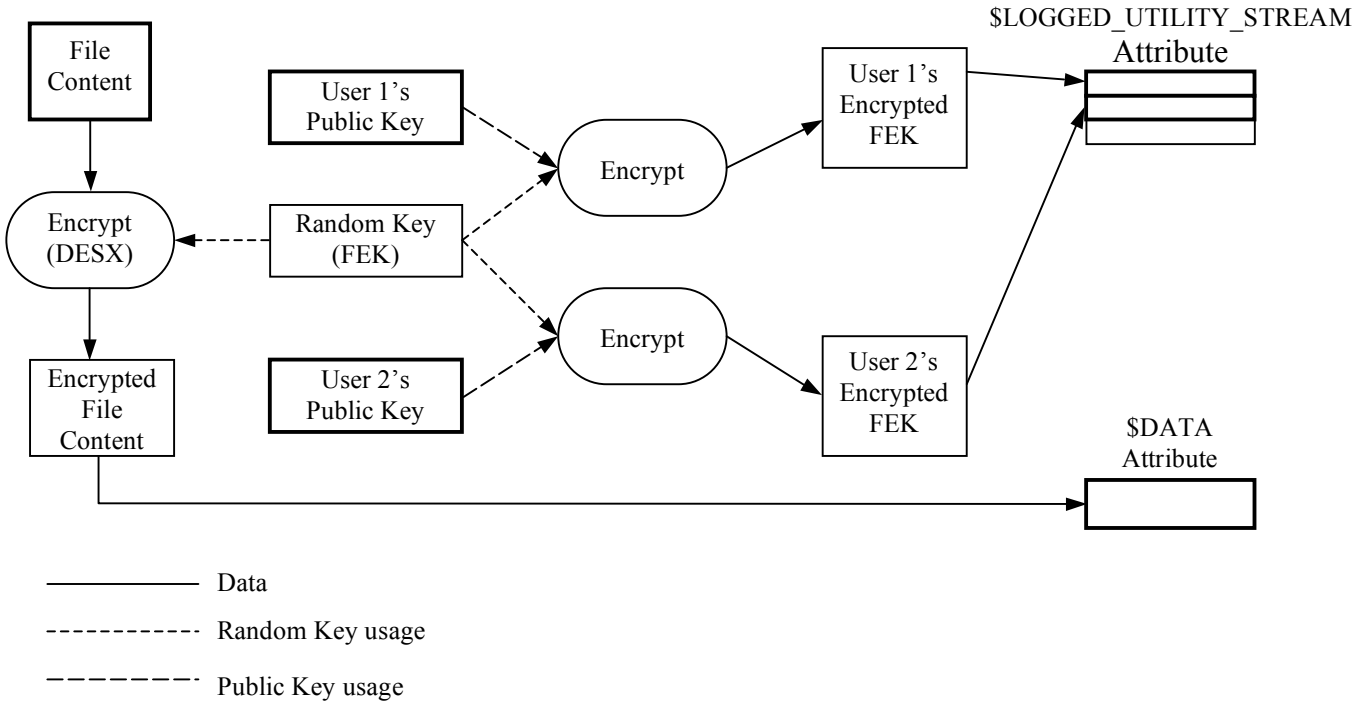


Abb. 4.11 Verschlüsselungsvorgang beginnend beim Dateiinhalt (File Content) vgl. [Car05 Figure 11.11 Seite 289]

Damit die Daten entschlüsselt werden können, wird der FEK aus dem $LOGGED_UTILITY_STREAM Attribut benötigt. Bevor dieser verwendet werden kann, muss er jedoch mit dem Private Key des Benutzers entschlüsselt werden. Dieser Private Key befindet sich, mit einem symmetrischen Verfahren verschlüsselt, in der Windows Registry und kann nur mit dem Passwort des Benutzers entschlüsselt werden. Nach dem Entschlüsseln der beiden Schlüssel sind alle Vorbereitungen für das Entschlüsseln des $DATA Attributes abgeschlossen und es kann nun mit dem FEK entschlüsselt werden. Den genauen Ablauf zeigt die folgende Abbildung.

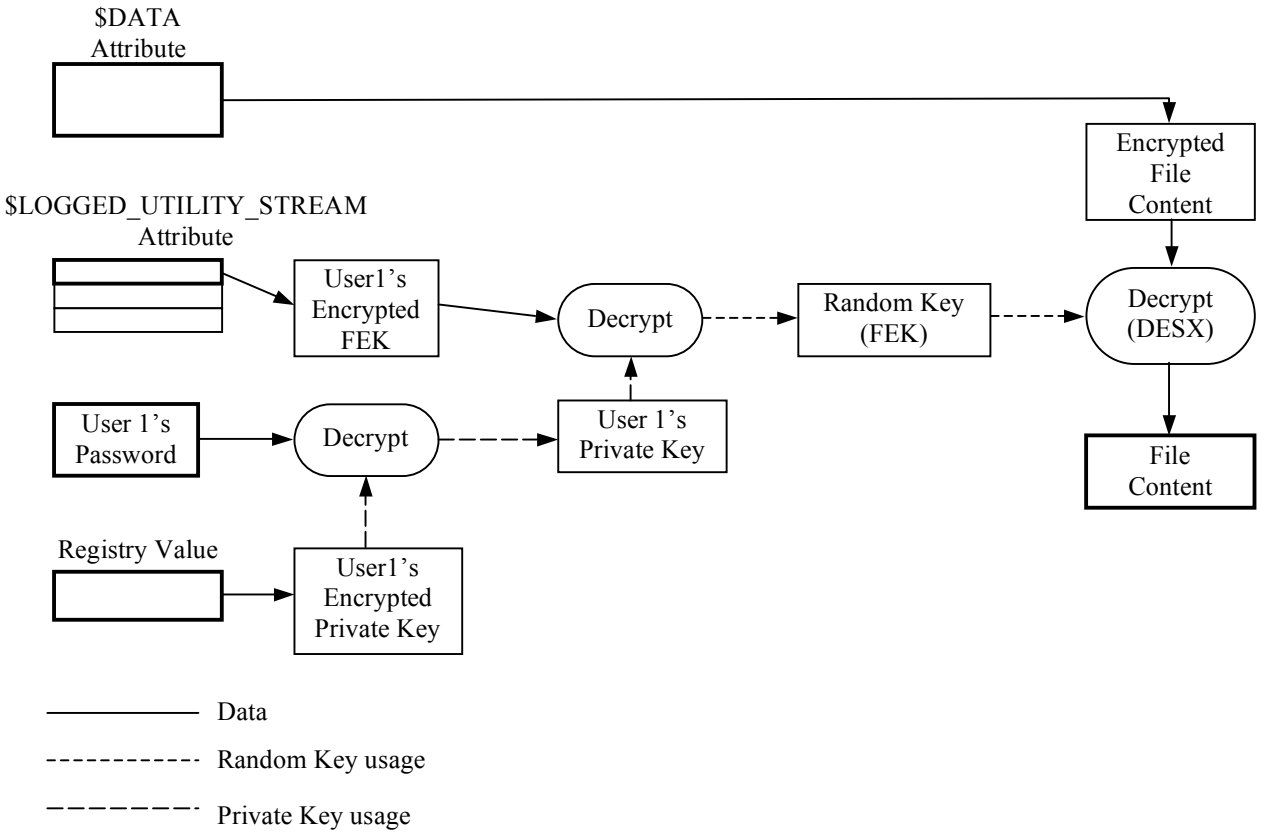


Abb. 4.12 Entschlüsselungsvorgang beginnend mit dem verschlüsselten Inhalt des $DATA Attributes vgl. [Car05 Figure 11.12 Seite 289]

Für die Analyse verschlüsselter Daten kann eine Brut Force Attacke auf das Benutzer-Passwort zielführend sein, wobei im NTFS-Design auch eine kleine Lücke existiert. So kann nach dem Verschlüsseln eine unverschlüsselte Kopie dieser Daten im freien Speicher der Festplatte existieren. Der Grund hierfür ist, dass für das Verschlüsseln die temporäre Datei `EFS0.TMP`, welche eine Kopie der nicht verschlüsselten Daten enthält, angelegt wird. Diese Datei wird zwar nach dem Verschlüsseln gelöscht, jedoch vom System nicht überschrieben.

Sollte, aus welchem Grund auch immer, ein Account welcher als Recovery-Agent konfiguriert ist zugänglich sein, so kann, aufgrund der Tatsache, dass dieser Zugriff auf alle Dateien hat, über die Wiederherstellungsfunktion jede Datei entschlüsselt werden. Zu beachten ist hier weiters, dass dieser Recovery-Agent nicht immer Standardmäßig aktiviert ist, und wenn ja, es dafür je nach Betriebssystem verschiedene Voraussetzungen gibt. So wird beispielsweise unter Windows 2000 der erste lokale Administrator der sich Anmeldet auch der Standard-Recover-Agent, egal ob der Computer Mitglied in einer Arbeitsgruppe oder eine Windows NT 4.0 Domäne ist. Befindet sich ein Computer mit Windows XP oder Windows 2000 in einer Windows Server 2003 oder Windows 2000 Domäne, so wird standardmäßig das vordefinierte Administratorkonto auf dem ersten Domänencontroller der Domäne als Standard-

Recovery-Agent verwendet. Befindet sich ein Computer mit Windows XP in einer Arbeitsgruppe, so gibt es keinen Standard-Recovery-Agent. Wird hier dennoch ein Recovery-Agent benötigt, so muss dieser manuell eingerichtet werden. [MS06-5]
Nachdem eine tiefer gehende Erklärung der Verschlüsselungsmethoden und Algorithmen den Rahmen dieser Arbeit sprengen würde, wird an dieser Stelle auf weiterführende Literatur verwiesen, beispielsweise [Sch05], [Sch07] oder [Buc03].

4.2.1.5 Indizes

In NTFS werden in vielen Situationen Indizes zum Speichern von Daten verwendet. Genauer gesagt ist ein NTFS-Index eine Sammlung sortiert gespeicherter Attribute. Seit der Version 3.0 werden Indizes nicht nur für das Speichern der $FILE_NAME Attribute eines Ordners, sondern auch für einige andere Attribute verwendet. Beispielsweise werden jetzt Sicherheitsinformationen oder Informationen über Quotas ebenfalls in einem Index gespeichert.
Die verwendete Datenstruktur für einen Index ist das Konzept der B-Trees. Aus Gründen des Umfangs wird hier nicht weiter auf dieses Konzept eingegangen, sondern wie zuvor schon bei der Kryptographie auf weiterführende Literatur verweisen, zum Beispiel [Ott02].
Viel interessanter ist an dieser Stelle, wie das Konzept der B-Trees in NTFS für Indizes verwendet wird.
Jeder Eintrag in einem Tree verwendet eine Datenstruktur, Indexeintrag (*index entry*) genannt, für das Speichern der Werte eines Knotens. Es gibt zwar viele verschiedene Typen von Indizes, wobei diese aber alle dieselbe Headerstruktur verwenden. Detaillierte Auskunft zu diesen Datenstrukturen gibt das Kapitel 13 „Index Attributes and Data Structures“ in [Car05 ab Seite 369].
In einem Indexeintrag eines Ordners ist beispielsweise neben den Headerinformationen noch das $FILE_NAME Attribut dieses Ordners gespeichert. Diese Indexeinträge werden in Knoten der B-Trees organisiert und in einer Liste gespeichert, wobei das Ende der Liste durch einen leeren Indexeintrag markiert wird.
Die Knoten der Indizes können in zwei verschiedenen Attributen in den MFT-Einträgen gespeichert werden. Entweder, falls nur ein Knoten mit wenigen Einträgen gespeichert werden soll, im $INDEX_ROOT Attribut oder, wenn mehrere Knoten gespeichert werden sollen, im *non-resident* $INDEX_ALLOCATION Attribut, wobei in diesem Fall das $INDEX_ROOT Attribut für den Wurzelknoten verwendet wird.
Der Inhalt des $INDEX_ALLOCATION Attributes besteht aus einem großen Buffer, welcher einen oder mehrere Indexdatensätze (*index records*), von denen jeder bei 0 beginnend adres-

siert wird, enthält. Ein Indexdatensatz hat typischerweise eine fixe Größe von 4.096 Bytes, von denen jedoch nicht alle verwendet werden müssen, und enthält eine Liste von sortierten Indexeinträgen. Der Zuteilungsstatus (*allocation status*) der Indexdatensätze wird im $BITMAP Attribut, welches für die Verwaltung des Buffers verwendet wird, gespeichert. Vor dem Einfügen eines neuen Knotens wird im $BITMAP Attribut nach einem leeren Indexdatensatz gesucht. Ist keiner mehr vorhanden, so muss der Speicher des Buffers vergrößert werden.

Weiters erhält jeder Index einen Namen, welcher im Attribut-Header des $INDEX_ROOT, des $INDEX_ALLOCATION und des $BITMAP Attributen gespeichert wird.

Damit aus den Indexeinträgen eine Baumstruktur aufgebaut werden kann, hat jedes dieser Indexeinträge ein Flag welches anzeigt, ob es unter ihm weitere Kinderknoten gibt. Wenn ja, dann werden deren Indexdatensatz-Adressen im Indexeintrag gespeichert. Soll ein Eintrag gefunden werden, so muss dessen Wert aufgrund der Sortierung nur mit den Werten an der aktuellen Position verglichen werden. Ist der zu suchende Wert kleiner oder ist das Ende der Liste erreicht, so muss, falls vorhanden, in den Kinderknoten weitergesucht werden.

In Abb. 4.13 wird gezeigt, wo die Daten beispielsweise eines Ordner-Index gespeichert werden.

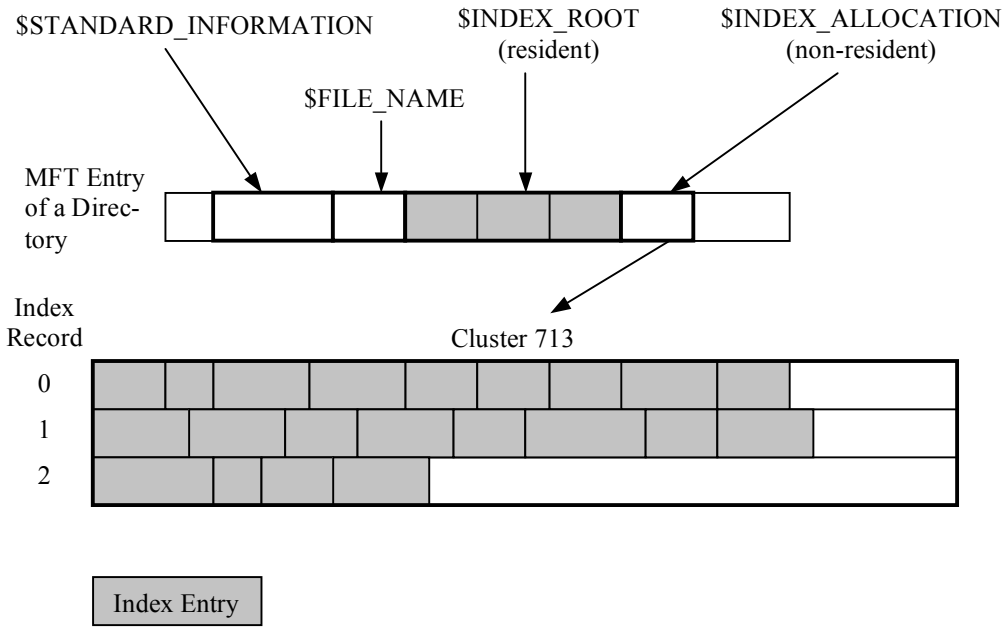


Abb. 4.13 Beispiel für die Verteilung der Daten eines Ordner-Index vgl. [Car05 Figure 11.19 Seite 295]

4.2.2 Kategorisierung der Daten

Wie schon zuvor im Abschnitt über das FAT-Dateisystem werden in diesem Kapitel die verschiedenen Daten den einzelnen Kategorien, welche im Kapitel 3.2 beschrieben wurden, zugeordnet und beschrieben.

4.2.2.1 File System Kategorie

In NTFS werden die Metadaten, welche das Dateisystem beschreiben, ebenfalls in Dateien, den Dateisystem-Metadatendateien, gespeichert. Logisch gesehen befinden sie sich im Root-Directory, physikalisch jedoch können sie, mit Ausnahme der Datei, die den Bootcode enthält, über das gesamte Volume verteilt gespeichert werden. Ein interessanter Nebeneffekt dieses Designs ist, dass diese Dateien Datums- und Zeitstempel besitzen, die für manche Untersuchungen hilfreich sein können.
Im weiteren Verlauf dieses Kapitels werden die Metadatendateien und die darin gespeicherten Daten kurz beschreiben.

$MFT Datei

Diese Datei ist eine der wichtigsten des gesamten Dateisystems, denn in ihr ist die *Master File Table* (MFT) gespeichert. Ergänzend zu der Beschreibung in Kapitel 4.2.1.1 ist noch zu erwähnen, dass im $DATA Attribut des ersten MFT-Eintrags, mit der Bezeichnung $MFT, die verwendeten Cluster der MFT gespeichert sind. Der Zuteilungsstatus der MFT-Einträge wird im $BITMAP Attribut gespeichert. Natürlich hat die $MFT wie jede andere Datei auch ein $FILE_NAME und ein $STANDARD_INFORMATION Attribut, aber dazu später.

$MFTMirr Datei

Nachdem die MFT für das Auffinden jeder Datei benötigt wird, stellt diese einen sogenannten *single point of failure* dar. Um dies zu verhindern, existiert eine Backupkopie der wichtigsten MFT-Einträge. Aus diesem Grund ist der MFT-Eintrag 1 für die $MFTMirr Datei, welche ein *non-resident* Attribut mit den Backupkopien der ersten MFT-Einträge enthält, reserviert.
Im $DATA Attribut welches in Cluster, die in der Mitte des Dateisystems liegen, gespeichert ist, werden Kopien der ersten vier MFT-Einträge, welche zu den Dateien $MFT, $MFTMirr, $LogFile und $Volume gehören, gespeichert.
Mit diesen vier Backupkopien kann ein Recoverytool das Layout, die Größe der MFT, die Position der $LogFile Datei, welches für die Wiederherstellung des Dateisystems benötigt wird, sowie die Version und die Statusinformationen des $Volume Attributes feststellen.

$Boot Datei

Diese Datei befindet sich im MFT-Eintrag 7 und beinhaltet den Bootsektor des Dateisystems. Wie schon erwähnt ist dies die einzige Datei mit einer fixen Speicherposition. Oder anders gesagt, das $DATA Attribut befindet sich immer im ersten Sektor des Dateisystems, da es für das Booten des Systems benötigt wird. Typischerweise werden die ersten 16 Sektoren für diese Datei reserviert, wobei jedoch nicht alle verwendet werden.

Zwischen dem NTFS und dem FAT-Bootsektor bestehen viele Ähnlichkeiten. So besitzen beide die 0xAA55 Signatur am Ende, was wiederum bei der Suche nach diesem dazu führt, dass beide gefunden werden. Weiters befinden sich im Bootsektor Informationen über die Größe der Cluster, die Anzahl der Sektoren des Dateisystems, die erste Clusteradresse der MFT sowie über die Größe der MFT-Einträge. Im Bootsektor findet sich ebenfalls eine Seriennummer für das Dateisystem.

Die Backupkopie des Bootsektors befindet sich laut Microsoft [MS04-3] entweder in der Mitte oder im letzten Sektor des Volumes. Wobei nach [Car05 Seite 304] in Windows NT 4.0, 2000 und XP immer der letzte Sektor des Volumes verwendet wird.

$Volume Datei

Diese Datei beinhaltet Metadaten des Dateisystems und befindet sich im MFT-Eintrag 3. Konkret werden in dieser Datei die Volumebezeichnung und Informationen über die Version des Dateisystems gespeichert. Für diese Information gibt es in dieser Datei zwei eigene Attribute, das $VOLUME_NAME für die Bezeichnung des Volumes und das $VOLUME_INFORMATION für die NTFS-Version und den ‚*dirty*' Status. Wie jede andere Datei besitzt die $Volume Datei ebenfalls ein $DATA Attribut, welches nach [Car05 Seite 305] jedoch keine Daten enthält und eine Größe von 0 Byte hat.

Wie schon einige Male erwähnt, gibt es zu den verschiedenen Windows-Versionen auch verschiedene Versionen der NTFS-Implementierung:

- Windows NT 4 verwendet NTFS-Version 1.2
- Windows 2000 verwendet NTFS-Version 3.0
- Windows XP, Windows Server 2003 und Windows Vista verwenden NTFS-Version 3.1

[Linux-NTFS]

$AttrDef Datei

Diese Datei ist ein weiterer wichtiger Teil der Dateisystem Kategorie, denn ihr $DATA Attribut speichert die Typ-Identifier und den Namen der verschiedenen Attributtypen. Damit das $DATA Attribut der $AttrDef Datei ohne Kenntnis über den Inhalt gelesen werden kann, gibt es Default-Werte für die Attribute.
Das Konzept dieser Datei ermöglicht jeder Instanz eines NTFS-Dateisystems die Standardattribute neu zu definieren und eigene Attribute für seine Dateien anzugeben.

4.2.2.2 Content Kategorie

Wie schon mehrfach erwähnt, werden in NTFS Attribute von Dateien, welche im MFT-Eintrag keinen Platz mehr haben, in Cluster ausgelagert. Unter einem Cluster versteht man eine zusammenhängende Gruppe von Sektoren, wobei die Anzahl der Sektoren pro Cluster eine 2er Potenz ist (1, 2, 4, 8, 16).
Jeder Cluster bekommt eine eigene Adresse, die beim ersten Cluster mit 0 beginnt. Anders als in FAT beginnen in NTFS die Cluster am Beginn des Dateisystems. Daraus ergibt sich die einfache Umrechnung von der Clusteradresse auf die Adresse des ersten Sektors dieses Clusters.

```
Sektor = Cluster x Sektoren_pro_Cluster
```

Mit Ausnahme des $DATA Attributes der $Boot Datei gibt es in NTFS keine strikten Layoutregeln. Daher kann jeder Cluster einer Datei bzw. einem Attribut zugeteilt werden. Sollte die Größe eines Volumes nicht dem Vielfachen der Clustergröße entsprechen, so werden die übrigen Sektoren vom Dateisystem nicht verwendet.
Abhängig vom verwendeten Betriebssystem kann es durchaus zu unterschiedlichen Layouts des Dateisystems kommen. Ein Konzept, das von jeder Windows-Version verwendet wird, ist das der *MFT Zone*. Da Windows die MFT am Beginn so klein wie möglich hält, könnte diese, durch das Benötigen weiterer Cluster, schnell fragmentiert werden. Um dem entgegen zu wirken, reserviert Microsoft einen Teil des Dateisystems für die MFT. Dieser Teil aus zusammenhängen Clustern wird *MFT Zone* genannt. Für diesen Bereich werden standardmäßig 12,5% des Dateisystems reserviert. Der Beginn dieser Zone ist jedoch wieder vom jeweiligen Betriebssystem abhängig. Die Cluster in diesem Bereich können erst nach dem Verwenden aller anderen Cluster für Dateidaten, außer die der MFT, verwendet werden.
Die folgende Abbildung zeigt die Unterschiede in der Verteilung der Dateisystemdaten bei der Verwendung von Windows 2000 und XP.

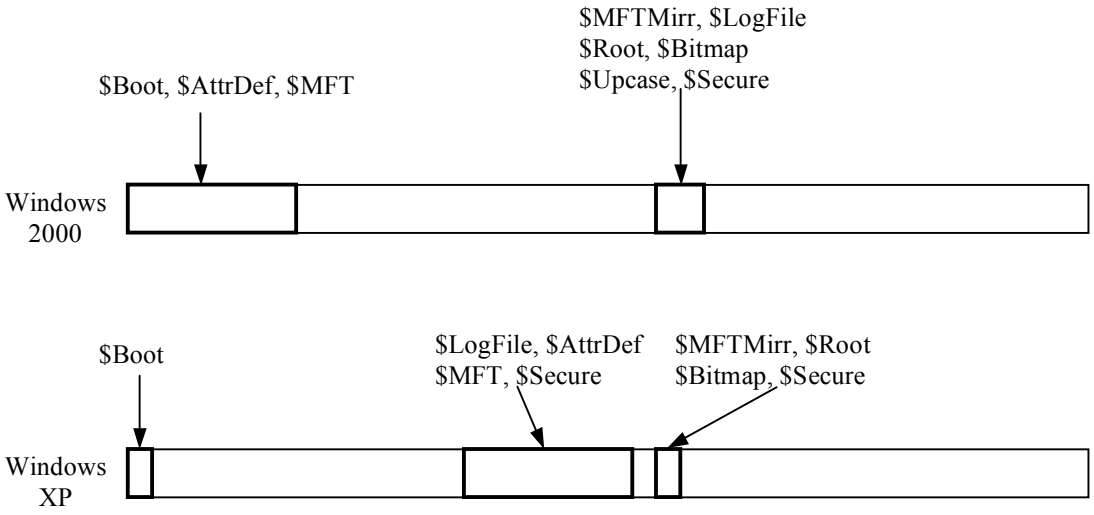


Abb. 4.14 Layout der Dateisystem-Metadaten je nach Betriebssystem Version [Car05 Figure 12.2 Seite314]

$Bitmap Datei

In dieser Metadaten Datei des Dateisystems, die sich im MFT-Eintrag 6 befindet, wird der Zuteilungsstatus jedes Clusters gespeichert. Für diesen Zweck gibt es im $DATA Attribut dieser Datei für jeden Cluster ein Bit, welches anzeigt, ob der Cluster zugeteilt ist oder nicht. Ist ein Bit, dessen Position in diesem Attribut der Adresse eines Clusters entspricht, auf 1 gesetzt, so bedeutet dies, dass dieser Cluster einem Attribut bzw. einer Datei zugeteilt ist.

$BadClus Datei

Sollte von Windows ein beschädigter Cluster entdeckt werden, so wird das in der Datei $BadClus gespeichert. Für das Speichern dieser Daten besitzt die Datei ein zweites $DATA Attribut mit dem Namen $Bad. Dieses Attribut besitzt, wie das $DATA Attribut der $Bitmap Datei, ein Bit für jeden Cluster des Dateisystems, jedoch handelt es sich bei dieser Datei um ein *sparse file*, und so werden nur jene Bits auf der Festplatte gespeichert, welche einem beschädigten Cluster zugeordnet sind.

Zuteilungsalgorithmus

Die Strategie zum Auffinden nicht zugeteilter Cluster ist von Betriebssystem zu Betriebssystem unterschiedlich. So verwendet nach Beobachtungen von Brian Carrier [Car05 Seite 313] beispielsweise Windows XP den *best fit* Algorithmus. Dieser Algorithmus versucht, zu den angeforderten Clustern einen freien Bereich zu finden, der annähernd dieselbe Größe bzw.

Anzahl von Clustern besitzt. Das bedeutet, dass bei mehreren Blöcken von freien Clustern derjenige ausgewählt wird, in den die benötigten Cluster am besten passen.

4.2.2.3 Metadata Kategorie

In NTFS werden die Metadaten der Dateien und Ordner, welche diese beschreiben, in Attributen gespeichert. Die wichtigsten Attribute, die diese Daten enthalten, werden in diesem Kapitel näher beschrieben.

$STANDARD_INFORMATION Attribut

Dieses Attribut existiert in jeder Datei und jedem Ordner und beinhaltet die wichtigsten Metadaten. Beispielsweise finden sich hier die primären Zeit- und Datumsstempel sowie Informationen über Besitzer, Sicherheit und Quotas. Keine dieser Daten sind essenziell für das Speichern von Dateien, werden jedoch von vielen Application-Level Features, die Microsoft anbietet, benötigt.

Wie bereits in Tab. 4.2 angeführt, besitzt dieses Attribut die Default-ID 16 und hat eine fixe Größe von 48 Bytes in Windows NT und 72 Bytes ab Windows 2000. Da Microsoft die Attribute der MFT-Einträge geordnet speichert, befindet sich dieses Attribut mit der niedrigsten Typ-ID an der ersten Stelle des MFT-Eintrags.

In diesem Attribut werden vier 64-Bit-Datums- und Zeitstempel gespeichert. Jeder dieser Werte repräsentiert die Zeit, welche seit dem 1. Jänner 1601 UTC bis zum jeweiligen Ereignis vergangen ist, mit einer Genauigkeit von Hundert Nanosekunden. Konkret handelt es sich dabei um folgende Zeitwerte:

- *Creation Time*: Zu dieser Zeit wurde die Datei angelegt.
- *Modified Time*: Der Zeitpunkt, an dem der Inhalt des $DATA oder des $INDEX Attributes zuletzt modifiziert wurde.
- *MFT Modified Time*: Der Zeitpunkt, an dem die Metadaten einer Datei zuletzt modifiziert wurden. Zu beachten ist hierbei, dass dieser Wert in Windows bei den Dateieigenschaften nicht angezeigt wird.
- *Accessed Time*: Der Zeitpunkt des letzten Zugriffs auf den Dateiinhalt.

Neben diesen Zeit- und Datumsangaben werden ebenfalls Flags, die angeben ob die Datei schreibgeschützt, eine Systemdatei oder eine Archivdatei ist, in diesem Attribut gespeichert. Zusätzlich beinhaltet es weitere Flags, die anzeigen, ob es sich um eine sparse, eine komprimierte oder eine verschlüsselte Datei handelt.

Seit NTFS-Version 3.0 gibt es vier weitere Felder, welche für Sicherheitsinformationen und Application-Level Features verwendet werden. Zu diesen Werten zählt die Besitzeridentität,

welche mit dem Besitzer der Datei übereinstimmt und ebenfalls für das Quotamanagement verwendet wird, die Anzahl der Bytes, die diese Datei zu den Quotas beiträgt, die Security-ID, welche als Index für die $Secure Datei und für die Zugriffskontrollregeln verwendet wird, sowie die *update sequence number* (USN), welche bei aktiviertem Journaling die Nummer des letzen Eintrags dieser Datei enthält.
Alles in allem enthält dieses Attribut viele interessante, aber für das Dateisystem nicht essenzielle Metadaten. Wobei bei den Zeiten darauf vertraut werden muss, dass sie vom Betriebssystem korrekt gesetzt werden.

$FILE_NAME Attribut

Jede Datei und jeder Ordner besitzt mindestens ein $FILE_NAME Attribut. Zusätzlich zu dem Attribut im MFT-Eintrag einer Datei bzw. eines Ordners wird im Index des übergeordneten Ordners eine weitere Instanz gespeichert. Das bedeutet jedoch nicht, dass beide Attribute dieselben Daten enthalten. Wie bereits in Tab. 4.2 angeführt, ist die ID dieses Attributes 48 und es hat eine variable Größe, die von der Länge des Dateinamens abhängt, oder genauer gesagt 66 Bytes fixe Größe plus die Länge des Namens. Gespeichert wird der Name in UTF-16 Unicodewerten und muss einem der möglichen Namensräume genügen. Zu diesen Namensräumen zählt das 8.3 DOS-Format, das Win32-Format und POSIX. Üblicherweise versucht Windows, für jede Datei ein $FILE_NAME Attribut im 8.3 DOS-Format anzulegen. Aus diesem Grund haben viele Dateien neben dem richtigen Namen auch noch einen im 8.3 DOS-Format. Dieses Verhalten kann jedoch deaktiviert werden, beispielsweise unter Windows XP über das Setzen des Registry Eintrags `HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\FileSystem\NtfsDisable8dot3NameCreation` auf den Wert 1 vgl. [MS07-2]. Zu beachten ist jedoch, dass jeder dieser Namensräume verschiedene Restriktionen bei den verwendbaren Zeichen aufweist. Ein weiterer wichtiger Wert in diesem Attribut ist die Dateireferenz auf den Ordner, in dem sich die Datei befindet. Weiters enthält dieses Attribut ebenfalls die vier zusätzlichen Felder und eine Menge von Flags über den Status der Datei wie das $STANDARD_INFORAMTION Attribut. Neben diesen Feldern gibt es noch zwei weitere, welche die tatsächliche und die benötigte Größe angeben. Laut [Car05 Seite 318] sind diese Felder jedoch auf 0 gesetzt.
Zusammengefasst sind von diesem Attribut nur der Dateiname und die Dateireferenz des übergeordneten Ordners wichtig. Aus diesen Informationen ist es möglich, den gesamten Pfad einer Datei oder eines Ordners zu rekonstruieren.

$DATA Attribut

Dieses Attribut hat keine spezielle Struktur oder definierten Werte und wird für das Speichern jeglicher Inhaltsdaten verwendet. Wie bereits in Tab. 4.2 angeführt, ist die ID dieses Attributes 128 und für die Größe gibt es keinen definierten Wert. Dieses Attribut wird für jede Datei angelegt und besitzt per Default keinen Namen. Es ist jedoch möglich, weitere $DATA Attribute anzugeben, jedoch müssen diese dann mit einem Namen versehen werden. Für diese weiteren $DATA Attribute, welche auch ADS (*alternate data streams*) genannt werden, gibt es eine Vielzahl von Anwendungsmöglichkeiten. So kann etwa ein Benutzer unter Windows zu einer Datei weitere Dateiinformationen angeben, die dann als ADS gespeichert werden, oder es kann beispielsweise ein Virenscanner oder ein Backupprogramm ADS für das Markieren einer Datei verwenden. Natürlich kann ein ADS auch für das Verstecken von Daten verwendet werden.

$ATTRIBUTE_LIST Attribut

Eine Datei oder ein Ordner kann bis zu 65536 Attribute besitzen, wobei nicht alle dieser Attribute in einem MFT-Eintrag Platz haben. Aus diesem Grund wird im $ATTRIBUTE_LIST Attribut für jedes Attribut der dazugehörige MFT-Eintrag gespeichert. Mit einer ID von 32, vgl. Tab. 4.2, ist sichergestellt, dass dieses Attribut immer im ersten MFT-Eintrag einer Datei gespeichert ist. Konkret beinhaltet dieses Attribut eine Liste aller Attribute der Datei, inklusive sich selbst, wobei jeder Eintrag die Typ-ID und die Adresse des MFT-Eintrags, in welchem das Attribut gespeichert ist, beinhaltet. Jeder dieser erweiterten MFT-Einträge speichert in seinem Header die Adresse des Basis MFT-Eintrags.

Damit das Ganze noch etwas komplizierter wird, kann es vorkommen, dass ein Attribut so weit fragmentiert wird, dass schon die Speicherung seiner *cluster runs*, vgl. Kapitel 4.2.1.2, mehrere MFT-Einträge benötigt. Falls dieser Fall eintritt, normalerweise nur bei $DATA Attributen, so erhält jeder weitere MFT-Eintrag ein ganz normales $DATA Attribut. Der Attribut-Header besitzt ein Feld, welches die *Virtual Cluster Number* (VCN) des *cluster runs* beinhaltet.

$SECURITY_DESCRIPTOR Attribut

Dieses Attribut existiert zwar in jeder Version von NTFS, meist aus Gründen der Abwärtskompatibilität, besitzt die ID 80, vgl. Tab. 4.2, wird jedoch nur von Windows NT verwendet. Sicherheitsdeskriptoren (*security descriptors*) werden von Windows für das Beschreiben der Zugriffskontrolle von Dateien und Ordner eingesetzt. Nachdem es oft vorkommt, dass mehre-

re Dateien dieselbe Beschreibung der Zugriffskontrolle besitzen, verwenden die neueren Versionen von NTFS eine eigene Metadaten Datei, in der die Sicherheitsdeskriptoren gespeichert werden. Diese Datei ist die $Secure Datei und befindet sich ab NTFS-Version 3.0 im MFT-Eintrag 9.
Für das Auffinden eines Sicherheitsdeskriptors wird eine eigene 32-Bit Security-ID verwendet. Diese *Security ID* darf jedoch nicht mit dem *Security Identifier* (SID) eines Benutzers verwechselt werden. Konkret besteht der Unterschied darin, dass die Security-ID nur im Dateisystem, der SID hingegen im ganzen System eindeutig ist.
Die $Secure Datei beinhaltet 2 Indizes, den $SDH und den $SII und ein $DATA Attribut, welches die aktuellen Sicherheitsdeskriptoren enthält. Ist von einer Datei oder eines Ordners die Security-ID bekannt, so wird der $SII Index, welcher nach den Security-IDs sortiert ist, zum Auffinden des Sicherheitsdeskriptors im $DATA Attribut verwendet. Der $SDH Index speichert die Hashwerte der gespeicherten Sicherheitsdeskriptoren und wird beim Ändern oder neu Anlegen eines Sicherheitsdeskriptors für das Suchen bereits vorhandener Sicherheitsdeskriptoren verwendet. Falls ein gesuchter Hashwert nicht vorhanden ist, wird ein neuer Sicherheitsdeskriptor erstellt, im $DATA Attribut gespeichert und die beiden Indizes aktualisiert.

4.2.2.4 File Name Kategorie

Nachdem in den vorherigen Kapiteln die Organisation und der Speicherort von Dateien und Ordner beschrieben wurden, geht es in diesem Kapitel darum, wie von einem Dateinamen auf die betreffende Datei bzw. deren Inhalt zugegriffen werden kann. Der Inhalt eines Ordners wird in einem Index, wie er in Kapitel 4.2.1.5 beschrieben wurde, verwaltet und in den $INDEX_ROOT und $INDEX_ALLOCATION Attributen gespeichert.

Ordner-Index

Jeder NTFS-Ordner hat einen normalen MFT-Eintrag mit einem speziellen Flag in dessen Header und in den Attributen $STANDARD_INFORMATION und $FILE_NAME. Die Einträge im Dateiindex enthalten eine Dateireferenzadresse und ein $FILE_NAME Attribut, welches, wie in Kapitel 4.2.2.3 beschrieben, den Dateinamen, die Dateigröße, Zeitstempel und weitere Flags enthält. Wobei die Dateigröße und die Zeitstempel in diesem Attribut bei Veränderungen von Windows aktualisiert werden.
Falls Windows so konfiguriert ist, dass für jede Datei ein 8.3 DOS-Name-Space Name benötigt wird, dann existieren mehrere $FILE_NAME Attribute im Index.

Die Abb. 4.15 zeigt einen einfachen Ordnerindex, bei dem zwei Indexdatensätze verwendet werden. Der Wert „X“ bedeutet, dass dieser Eintrag leer ist und alle dahinter Folgenden ebenfalls. Die Datei mit dem Namen qqq.txt wurde bereits gelöscht und der Eintrag ccc.txt wurde nach dem Einfügen weiterer Einträge im Zuge der Umsortierung an eine andere Stelle kopiert. Wird für eine Datei ein zweiter Dateiname wegen der Unterstützung eines weiteren Name Spaces, benötigt, so wird dieser Eintrag wie der zweite Eintrag der Datei eeeeeeeeeeee.txt, welcher im 8.3 DOS-Format ist, hinter dem ersten Eintrag eingefügt.

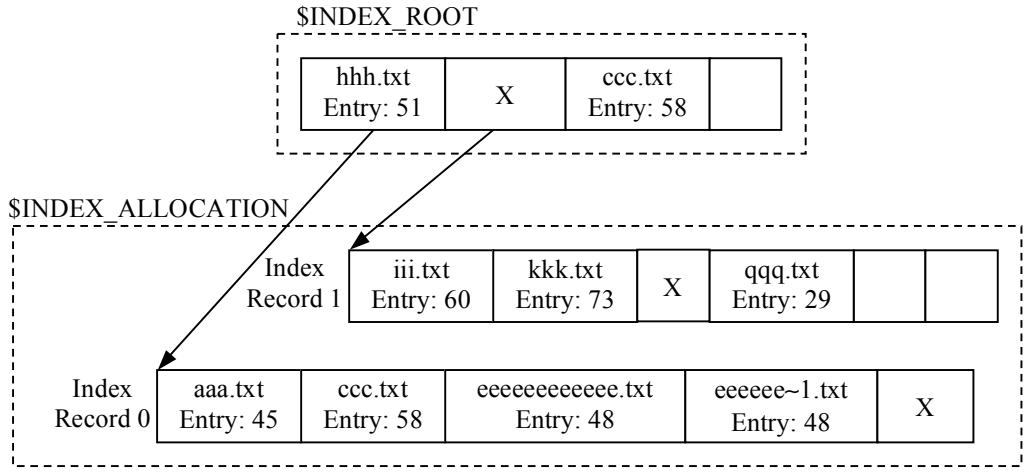


Abb. 4.15 Einfacher Ordnerindex [Car05 Figure 12.9 Seite 334]

Die Auflistung der Dateien des Ordners aus Abb. 4.15:

```
Verzeichnis von F:\Ordner-Index

09.08.2008  17:07    <DIR>          .
09.08.2008  17:07    <DIR>          ..
09.08.2008  17:04                 0 aaa.txt
09.08.2008  17:05                 0 eeeeeeeeeeee.txt
09.08.2008  17:03                 0 hhh.txt
09.08.2008  17:04                 0 iii.txt
09.08.2008  17:04                 0 kkk.txt
09.08.2008  17:04                 0 qqq.txt
               6 Datei(en)              0 Bytes
               2 Verzeichnis(se),    201.673.216 Bytes frei
```

Root-Directory

Wie bereits in Tab. 4.1 angegeben, befindet sich das Root-Directory mit dem Namen ‚.’ im MFT-Eintrag 5 und besitzt die Standardattribute $INDEX_ROOT, $INDEX_ALLOCATION und $BITMAP. In diesem Ordner befinden sich alle Metadatendateien des Dateisystems und wird für die Angabe des vollständigen Pfades einer Datei oder eines Ordners benötigt.

Weitere Dateizugriffsmöglichkeiten

Über die Funktion der sogenannten *hard links* können in NTFS mehrere Namen für eine Datei angegeben werden. Ein Hardlink zeigt auf den MFT-Eintrag der Datei, unterscheidet sich nicht vom originalen Dateinamen und befindet sich im Index des Ordners, in welcher die Datei gespeichert ist. Im Header des MFT-Eintrags der Datei befindet sich ein Linkzähler (*link counter*), welcher nach jedem Anlegen eines Hardlinks um 1 inkrementiert wird. Dieser Zähler zeigt nicht nur an, wie viele Hardlinks auf eine Datei zeigen, sondern er verhindert auch das Löschen eines MFT-Eintrags solange noch mindestens ein Hardlink darauf verweist. Sprich, ein MFT-Eintrag wird erst gelöscht, wenn durch Entfernen aller Hardlinks der Linkzähler wieder auf 0 gesetzt wurde.

Der MFT-Eintrag einer Datei enthält zusätzlich zu seinem $FILE_NAME Attribut ein weiteres $FILE_NAME Attribut für jeden Hardlink auf diese Datei bzw. diesen Eintrag. Zu beachten ist, dass ein Hardlink auf eine Datei nur innerhalb eines Volumes angelegt werden kann.

Mit Version 3 von NTFS wurden so genannte *reparse points*, welche zum Verlinken von Dateien, Ordnern und Volumes verwendet werden können, eingeführt. Dabei handelt es sich um spezielle Dateien oder Ordner, in welchen Informationen über das verlinkte Objekt gespeichert werden. Ein *reparse point* kann auf ein Objekt, Datei oder Ordner, welches sich im gleichen Volume, auf einem anderen Volume oder auf einem Netzwerkserver befindet, zeigen. Eine weitere Verwendung von *reparse points* ist das Mounten von Volumes als Ordner anstelle eines logischen Laufwerks mit eigenem Laufwerksbuchstaben, oder kurz gesagt: `/files` anstelle von `D:/` .

Ebenfalls wurde mit Version 3 von NTFS eine neue Möglichkeit für das Adressieren von Dateien und Ordner eingeführt. Diese Möglichkeit verwendet nicht die übliche Adressierung über Ordner und Dateinamen bzw. MFT-Einträgen, sondern verwendet eine eindeutige 128-Bit Objekt-ID (*object identifier*). Dieser Wert wird der Datei oder dem Ordner entweder vom Betriebssystem oder von einem Programm zugeordnet. Verwendet werden diese Object-IDs sowohl für Dateien und Ordner, als auch für eingebettete Dateien. Sie stellen sicher, dass ein Objekt mit dieser ID nach dem Umbenennen und Verschieben, auch auf ein anderes Volume, noch gefunden wird. Bekommt eine Datei oder ein Ordner eine Objekt-ID, so wird diese neben eventuellen Informationen über die ursprüngliche Position, im $OBJECT_ID Attribut gespeichert.

Zugegriffen wird auf Dateien mit einer Objekt-ID über die Zugriffsadresse, welche sich im Index `\$Extend\$ObjId` befindet.

4.2.2.5 Application Kategorie

In NTFS wurden viele Funktionen der Application Kategorie ins Dateisystem integriert. Wobei diese Funktionen nicht für die Hauptfunktionen des Dateisystems, das Speichern und wieder Finden von Dateien, sondern für das effizientere Arbeiten des Betriebssystems sowie von Programmen verwendet werden. In NTFS gehören zu diesen Funktionen *Disk Quotas*, *Logging* und *Change Journaling*.

Disk Quotas

Disk Quotas können von einem Administrator konfiguriert werden und limitieren den Speicherplatz, welcher von einem Benutzer verwendet werden darf. Die Informationen über die Quotas werden zu einem Teil im Dateisystem und zum anderen Teil in programmspezifischen Dateien, etwa in der Windows Registry gespeichert. Vor NTFS-Version 3 wurden die Quota Informationen in der Datei $Quota, die sich im MFT-Eintrag 9 befand, gespeichert. Seit Version 3 befindet sich diese Datei im Ordner `\$Extend` und kann in jedem MFT-Eintrag gespeichert werden.
Für das Verwalten der Quota-Informationen enthält die $Quota Datei zwei Indizes mit den Namen $O und $Q. Der Index $O verknüpft die Benutzer-ID mit der Security-ID und der Index $Q verknüpft die Benutzer-ID mit den Werten der verbrauchten und noch zur Verfügung stehenden Bytes.

Dateisystemjournaling

Für die Erhöhung der Stabilität des Dateisystems wird oft in einem Journal gespeichert, welche Operationen auf den Dateien und Ordnern durchgeführt werden. Aus diesem Journal kann das Betriebssystem bei Bedarf Fehler des Dateisystems, welche aufgrund eines Systemabsturzes während durchgeführter Schreiboperationen des Dateisystems auftraten, korrigieren. NTFS besitzt ebenfalls diese Funktionalität, wird von Microsoft jedoch als *Logging* bezeichnet. Die dazu benötigten Daten werden in der Datei $LogFile, welche sich im MFT-Eintrag 2 befindet und keine Spezialattribute enthält, gespeichert.
Das $DATA Attribut dieser Datei besteht aus dem Restart-Bereich (*Restart Area*) und dem Logging-Bereich (*Logging Area*). Im Restart-Bereich werden zwei Kopien einer Datenstruktur, welche vom Betriebssystem für das Aufräumen nach einem Crash verwendet werden, gespeichert. Diese Datenstrukturen enthalten ebenfalls einen Zeiger auf die letzte erfolgreich abgeschlossene Transaktion die im Logging-Bereich gespeichert ist.

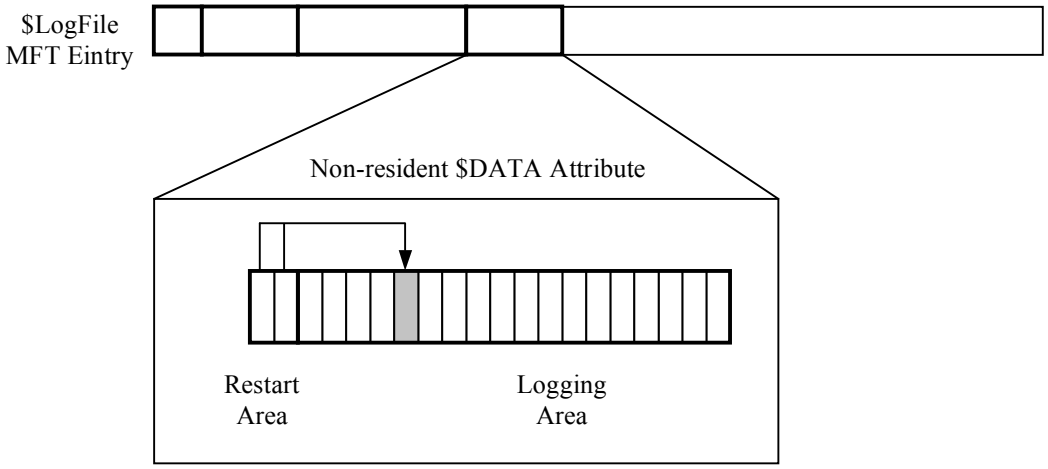


Abb. 4.16 Layout der $LogFile Datei mit den Logginginformationen im $DATA Attribut.[Car05 Figure12.11 Seite 341]

Die Logging Area besteht aus einer Reihe von Einträgen welche eine eindeutige und aufsteigend vergebene 64-Bit logical sequence number (LSN) besitzen. Die Logging Area besitzt eine fixe Größe und wird als Ringbuffer verwendet. Sprich nach dem Erreichen des letzten Eintrages werden am Beginn nicht mehr benötigte Einträge überschrieben. Aus diesem Grund zeigt die LSN nicht die Position, sondern den Zeitpunkt der Erstellung eines Eintrags an. Mit dem Program `chkdisk` und dem Parameter `/L` kann die Größe der $LogFile Datei angezeigt und mit `/L:`*`Größe`* geändert werden. [MSTN03-3]

Zu den Ereignissen, die in dieser Datei gespeichert werden, zählen unter anderem:

- Anlegen einer Datei oder eines Ordners
- Änderungen am Inhalt einer Datei oder eines Ordners
- Umbenennungen von Dateien oder Ordner
- Jegliche Änderungen an den Daten, die im MFT-Eintrag einer Datei oder eines Ordners gespeichert sind

Der meist verwendete Eintrag im Logging-Bereich ist der Update-Eintrag (*update record*). Dieser Eintrag wird für das Beschreiben der Dateisystemtransaktionen vor und nach dem Durchführen dieser verwendet. Viele Transaktionen werden in kleinere Teile unterteilt und benötigen dann für jeden Teil einen eigenen Update-Eintrag.

Neben der LSN besitzt ein Update-Eintrag zwei Felder. In einem dieser Felder, dem so genannten *redo field*, werden die Informationen über die durzuführende Operation gespeichert. Im anderen Feld wird genau das Gegenteil, nämlich die Information über das undo dieser Operation gespeichert. Beide Felder werden vor dem Durchführen der Transaktion erzeugt und

gespeichert. Wurde die Transaktion erfolgreich beendet, so wird ein weiterer Update-Eintrag, der sogenannte *Commit Record* eingefügt.

Der zweite Typ von Einträgen ist der Checkpoint-Eintrag (*checkpoint record*). Dieser Eintrag wird von Windows alle 5 Sekunden erzeugt, und gibt an wo Windows mit dem Verifizieren des Dateisystems Beginnen soll. Für diesen Zweck wird die LSN dieses Eintrages im Restart-Bereich der $LogFile Datei gespeichert.

Für das Verifizieren des Dateisystems lokalisiert das Betriebssystem den letzten Checkpoint-Eintrag und sucht nach gestarteten Transaktionen. Bei den abgeschlossenen Transaktionen kann mithilfe des *redo* Feldes überprüft werden ob die Daten korrekt auf der Festplatte gespeichert sind. Konnte eine Transaktion nicht abgeschlossen werden, so verwendet das Betriebssystem das *undo* Feld um die Daten auf der Festplatte in einen konsistenten Zustand zu bringen.

Die folgende Abbildung zeigt die $LogFile Datei mit zwei Transaktionen nach dem letzten Checkpoint-Eintrag, wobei eine dieser Transaktionen nicht mit einem *Commit Record* abgeschlossen wurde.

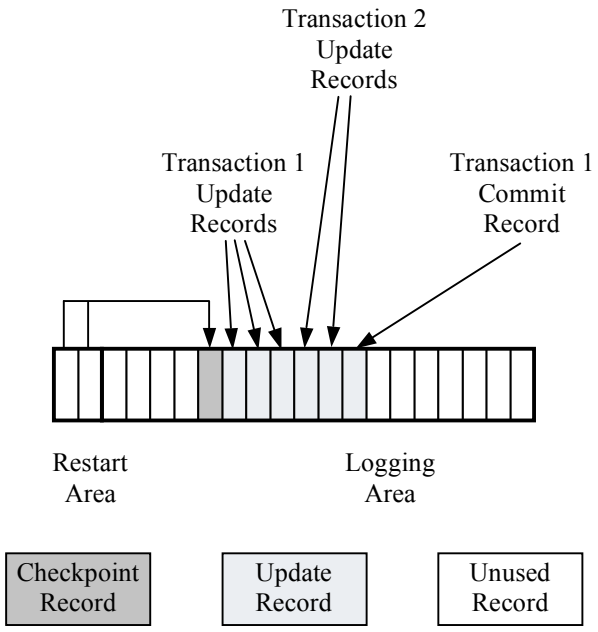


Abb. 4.17 Beispiel Transaktionen der $LogFile Datei vgl. [Car05 Figure 12.12 Seite 342]

Change Journal

Das *Change Journal* bietet ab NTFS-Version 3 für Programme die Möglichkeit über Veränderungen an Dateien und Ordner ohne das Scannen der Festplatte informiert zu werden. Diese Funktion, welche standardmäßig nicht aktiviert ist, kann von jedem Programm aktiviert bzw. deaktiviert werden. Damit ein Programm, welches das Journal aktiviert hat, erfährt, dass sich

der Status der Funktion geändert hat, besitzt das Journal eine 64-Bit Nummer, die bei jedem De- bzw. Aktivieren geändert wird, wobei jedoch beachtet werden muss, dass Windows die Journaldatei nach dem Deaktivieren löscht.
Gespeichert wird das Change Journal in der Datei $UsrJrnl, welche sich im Ordner `\$Extend` befindet. Diese Datei beinhaltet zwei $DATA Attribute, wobei im ersten, mit dem Namen $Max, die grundlegenden Daten über das Journal und im zweiten, mit dem Namen $J, die aktuellen Journaleinträge, welche keine fixe Größe, da diese von der Länge des Dateinamens abhängt, besitzen, gespeichert werden. Jeder Eintrag besitzt eine 64-Bit *Update Sequenz Number (USN)*, welche als Index verwendet und im $STANDARD_INFORMATION Attribut der jeweiligen Datei gespeichert wird. Da die Journaleinträge keine fixe Größe besitzen, ist die USN gleichzeitig der Byteoffset im Index, in welchem nur der Typ der Änderung gespeichert wird.
Da die USN ein Byteoffset ist, werden neue Einträge in den Index immer am Ende der Datei angefügt. Windows beschränkt jedoch die Größe der Journaldatei. Damit diese beiden Eigenschaften in einer Datei vereint werden können, werden nach dem Erreichen der maximalen Größe die Einträge am Beginn der Datei gelöscht und die Datei in eine *sparse* Datei umgewandelt.

4.2.3 Möglichkeiten der Datenwiederherstellung

Wie zuvor beim FAT-Dateisystem, Kapitel 4.1.3, muss auch in NTFS vor der Beschreibung der Wiederherstellungsmöglichkeiten, der Löschvorgang betrachtet werden.

4.2.3.1 Löschen von Dateien

Ist die betreffende Datei über die MFT und die Ordnerindizes gefunden, so muss für das Löschen nur mehr der Dateiname aus dem Ordnerindex entfernt, sowie die verwendeten Cluster und der MFT-Eintrag der Datei als ‚nicht zugeteilt' markiert werden. Falls notwendig wird der Index des Ordners noch neu sortiert.

4.2.3.2 Wiederherstellung von Daten

In NTFS können, im Unterschied zu anderen Dateisystemen, gelöschte Dateien sehr einfach wiederhergestellt werden. Der Grund hierfür ist, dass die in den MFT-Einträgen gespeicherten Zeiger und Verweise bei den NTFS-Implementierungen nicht gelöscht werden. Es kann jedoch passieren, dass ein MFT-Eintrag durch das Neuanlegen einer Datei überschrieben wird und so die noch vorhandenen Zeiger und Verweise gelöscht werden. Eine weitere Schwierigkeit, die dabei auftreten kann, ist, dass der Dateiname aus dem Ordner, in dem die Datei ge-

speichert war, durch das Umsortieren des Indexes gelöscht wird. Dies beeinträchtigt die Wiederherstellung einer Datei nur insofern, als dass die gefundene Datei eventuell keinem Ordner mehr zugeteilt werden kann.

Prinzipiell können gelöschte Dateien über das Absuchen der MFT nach nicht zugeteilten Einträgen gefunden werden. Falls einer dieser Einträge von einer gelöschten Datei stammt, kann über das darin enthaltene $FILE_NAME Attribut die Dateireferenzadresse auf den Ordner, in dem die Datei gespeichert war, referenziert, und der Pfad zu der Datei rekonstruiert werden. Falls die Cluster in denen die Daten gespeichert waren noch nicht überschrieben wurden, können die Dateien über die noch vorhandenen Zeiger wiederhergestellt werden.

Hatte diese Datei viele Attribute, so kann es möglich sein, dass für das korrekte Wiederherstellen die weiteren MFT-Einträge gesucht werden müssen. Windows verwendet für das Zuteilen neuer MFT-Einträge eine *first-available* Strategie, aus diesem Grund werden MFT-Einträge mit einer niedrigen Adresse öfter überschrieben als solche mit hohen Adressen. Diese Zuteilungsstrategie der MFT-Einträge darf jedoch nicht mit der für Cluster verwechselt werden.

Ebenfalls kann das Dateisystemlog und das Channge Journal, welches jedoch nicht immer aktiviert ist, nach Einträgen von Löschoperationen durchgesucht werden. Über diese Einträge kann dann versucht werden die betroffenen Dateien zu identifizieren und eventuell wiederherzustellen.

4.3 Ext3

Die Wurzeln von Ext3 liegen im UNIX File System (UFS). Es ist eine Erweiterung des Dateisystems Ext2 um die Funktionalität des Journaling. Die Weiterentwicklung von UFS besteht in erster Linie in dessen Vereinfachung und im Entfernen nicht mehr benötigter Komponenten. Obwohl Ext3 unter Linux weit verbreitet ist, und viele Distributionen es als Default-Dateisystem verwenden, gibt es ebenfalls Treiber[10] für Windows, die es erlauben, ein Ext3 bzw. Ext2 formatiertes Volume zu mounten und wie jedes andere Laufwerk zu verwenden.

4.3.1 Allgemeine Beschreibung

Das Design von Ext3 ist so ausgelegt, dass das Dateisystem schnell und zuverlässig arbeitet. Um dies zu erreichen, werden Kopien wichtiger Datenstrukturen über die gesamte Partition verteilt und die Daten einer Datei so gespeichert, dass die Leseköpfe der Festplatte beim Lesen der Datei wenig bewegt werden müssen. Am Beginn des Dateisystems kann sich, optional, ein reservierter Bereich befinden. Der Rest wird in Teile, den so genannten Blockgruppen (*block groups*), unterteilt. Jede dieser Blockgruppen, mit Ausnahme der Letzten, besitzt gleich viele Blöcke, welche für das Speichern der Dateinamen, der Metadaten und des Dateiinhaltes verwendet werden.

Die grundlegenden Layoutinformationen werden in einer eigenen Datenstruktur am Beginn des Dateisystems, dem sogenannten *Superblock,* gespeichert. Die Dateiinhalte werden in zusammenhängenden Sektoren, den sogenannten Blöcken (*blocks*), und die Metadaten der Dateien und Ordner in eigenen Datenstrukturen, den *Inodes*, gespeichert. Die Inodes besitzen eine fixe Größe und werden in einer eigenen Inodetabelle, welche in jeder Blockgruppe existiert, gespeichert. Ordner besitzen eine einfache Datenstruktur, in welcher die Namen und die Zeiger auf die Inodeeinträge der Dateien dieses Ordners gespeichert werden. Die Zusammenhänge des soeben Beschriebenen werden in der folgenden Abbildung dargestellt.

[10] Ext2 Installable File System For Windows, http://www.fs-driver.org

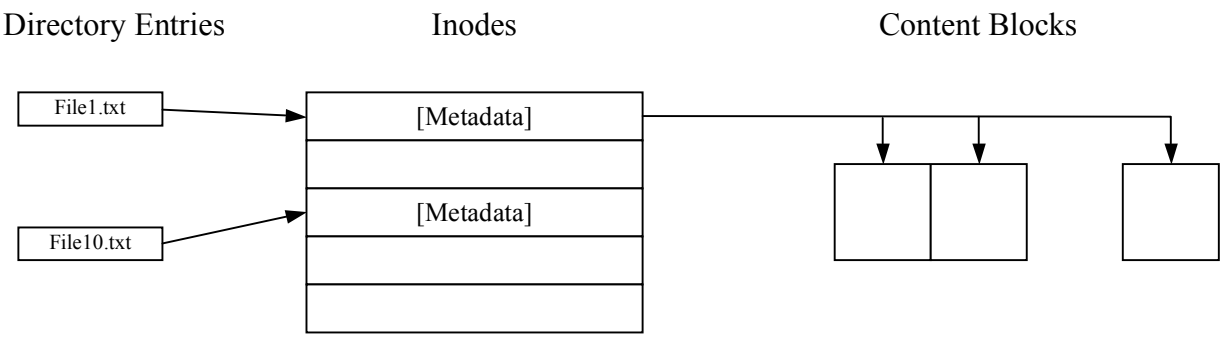


Abb. 4.18 Zusammenhänge zwischen Dateieinträgen, Inodes und Datenblöcken [Car05 Figure 14.1 Seite 398]

In Ext3 gibt es eine Reihe weiterer optionaler Features, welche in drei Gruppen unterteilt werden können. Entscheidend für die Unterteilung ist die Kompatibilität der Features zu Dateisystemimplementierungen, welche diese, weil optional, nicht unterstützen. So umfasst die erste Gruppe die voll kompatiblen Features, welche bei Nicht-Implementierung dem erfolgreichen Mounten und Verwenden nicht im Weg stehen. Zu diesen Features zählt etwa eine andere Zuteilungsstrategie oder ein aktiviertes Journal. Die zweite Gruppe umfasst jene Features, die ein mounten verhindern, falls eines davon aktiviert, jedoch nicht implementiert, ist. Dazu zählt etwa die Verschlüsselung oder die Komprimierung des Dateisystems. Zuletzt gibt es noch eine Menge von Features, welche bei Nicht-Implementierung das Mounten zwar nicht verhindern, aber die Verwendung einschränken, was meist eine read-only Verwendung des Dateisystems zur Folge hat. Beispielsweise fällt die Verwendung von B-trees anstelle unsortierter Listen für das Speichern der Dateieinträge in Ordnern in diese Gruppe.
Zu erwähnen ist weiters, dass es in Ext3 viele experimentelle Features, welche bei den meisten Distributionen nicht aktiviert bzw. installiert sind, gibt. Im Zuge einer forensischen Untersuchung muss jedoch bedacht werden, dass einige dieser Features von Administratoren oder experimentierfreudigen Benutzern aktiviert worden sein könnten.

4.3.2 Kategorisierung der Daten

Nach der Einleitung folgt in diesem Kapitel, wie zuvor bei FAT, Kapitel 4.1, und NTFS, Kapitel 4.2, die Unterteilung der gespeicherten Daten nach dem Schema aus Kapitel 3 bzw. Kapitel 3.2.

4.3.2.1 File System Kategorie

Diese Kategorie beinhaltet alle Daten, die für die Beschreibung des Dateisystems benötigt werden. Diese Daten werden in Ext3 in zwei verschiedenen Datenstrukturen gespeichert. Das ist auf der einen Seite der Superblock mit den grundlegenden Layoutinformationen, welcher

am Beginn des Dateisystems gespeichert wird und mit dem Bootsektor in FAT oder NTFS verglichen werden kann. Auf der anderen Seite gibt es für jede Blockgruppe einen Gruppendeskriptor (*group descriptor*), welcher das Layout der Blockgruppe beschreibt. Gespeichert werden diese Gruppendeskriptoren in der Gruppendeskriptortabelle (*group descriptor table*), welche im ersten Block nach dem Superblock gespeichert ist. Für die Stabilität des Dateisystems werden Backupkopien dieser Datenstrukturen über das gesamte Volume verteilt gespeichert.

Superblock

Der Superblock besitzt eine Größe von 1024 Bytes und beginnt nach den ersten 1024 Bytes eines Volumes. Wie in der Einleitung schon erwähnt, enthält der Superblock keinen Bootcode sondern lediglich Konfigurationswerte. Zu den gespeicherten Konfigurationswerten zählen die Blockgröße, die Anzahl der Blöcke, die Anzahl der Blöcke in einer Blockgruppe, die Anzahl der reservierten Blöcke vor der ersten Blockgruppe sowie die Anzahl der Inodes und der Inodes pro Blockgruppe. Neben diesen Konfigurationswerten werden noch weitere Daten, wie der Name des Volumes, Zeitstempel des letzten Schreibzugriffs, sowie wann und unter welchem Pfad das Volume das letzte Mal gemountet wurde, gespeichert. Ebenfalls gespeichert wird, ob das Dateisystem einer Konsistenzüberprüfung unterzogen werden muss oder nicht, und wie viele Blöcke und Inodes noch frei sind.
Der Superblock gibt auch Auskunft darüber, welche Features, wie in der Einleitung beschrieben, aktiviert sind. Die vorhin angesprochenen Backupkopien werden typischerweise im ersten Block jeder Blockgruppe gespeichert, wobei das aber von der jeweiligen Version des Betriebssystems abhängt.
Ein weiterer wichtiger Punkt ist, dass in Linux nicht nur über den Devicenamen sondern auch über den Volumenamen auf ein Dateisystem referenziert werden kann. Für diesen Zweck muss in der Konfigurationsdatei `/etc/fstab` mit „`LABEL=`*`name`*“ auf das Volume verwiesen werden.

Blockgruppen-Deskriptortabelle

Die Gruppendeskriptortabelle, in welcher sich die Gruppendeskriptoren für jede Blockgruppe des Dateisystems befinden, wird im ersten Block nach dem Superblock gespeichert. In diesen Gruppendeskriporen wird angegeben wo in einem Block die verschiedenen Datenstrukturen, wie Superblocks, Gruppendeskriptortabellen, Blockbitmaps, Inodetabellen und Inodebitmaps, welche neben den Dateidaten in den Blöcken gespeichert sind, gefunden werden können.

In der Blockbitmap wird der Zuteilungsstatus jedes Blocks in der Blockgruppe gespeichert. Da die Größe eines Blocks in Linux so angegeben wird, dass die Anzahl der Bits in einem Block genau der Anzahl von Blöcken in einer Blockgruppe entspricht, benötigt die Blockbitmap genau einen Block. Aufgrund dieser Gegebenheit lässt sich die Größe der Blockbitmap, und damit auch die eines Blockes, in Bytes einfach mit, „Anzahl der Blöcke einer Blockgruppe dividiert durch 8“, berechnen.
Der Zuteilungsstatus der Inodes wird in der Inodebitmap gespeichert. Die Größe dieser Bitmap kann durch das Dividieren der Anzahl der Inodes durch 8 berechnet werden. Im Allgemeinen sind in einer Blockgruppe weniger Inodes wie Blöcke angelegt, wobei die Anzahl beim Erzeugen des Dateisystems vom Benutzer angegeben werden kann.
Die Blockadressen der Blockbitmap, der Inodebitmap sowie die des Superblocks werden im Gruppendeskriptor gespeichert.
Wie die einzelnen Blöcke in einer Blockgruppe verteilt sind, zeigt die folgende Abbildung.

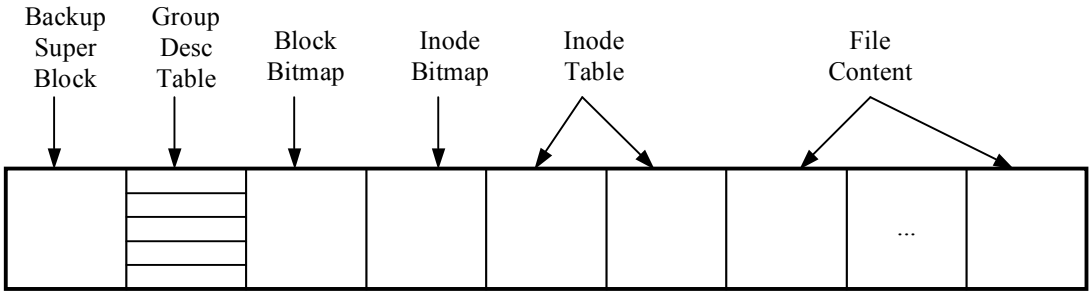


Abb. 4.19 Einfache Blockgruppe vgl. [Car05 Figure 14.3 Seite 401]

Bootcode

Soll ein Volume für das Booten verwendet werden, so kann in den ersten 1024 Bytes der benötigte Bootcode, welcher vom Bootcode im MBR (*Master Boot Record*) gestartet wird, gespeichert werden. Eine andere Möglichkeit, wie von vielen Linux-Distributionen verwendet, ist das Installieren eines *boot loader* im MBR, welcher direkt den Betriebssystem-Kernel startet.

4.3.2.2 Content Kategorie

In dieser Kategorie werden die Daten der Dateien und Ordner gespeichert. Wie zuvor schon erwähnt werden diese Daten in zusammenhängenden Sektoren, den Blöcken, gespeichert. Diese Blöcke können mit Cluster in NTFS oder FAT verglichen werden. Die Größe der Blöcke kann entweder 1024, 2048, 4096 oder 8192 Bytes sein und wird im Superblock angege-

ben. Aus der Blockgröße und den im Kernel angegeben Limits ergeben sich die maximalen Größen für Dateien und das Dateisystem, welche in der folgenden Tabelle aufgelistet werden.

Blockgröße	1 KB	2 KB	4 KB	8 KB
Max. Dateigröße	16 GB	256 GB	2048 GB	2048 GB
Max. Dateisystem-größe	2047 GB	8192 GB	16384 GB	32768 GB

Anzumerken ist hierbei noch, dass die Blockgröße von der Größe der Kernelpages abhängt. So können etwa Blockgrößen von 8192 Bytes nur auf Systemen, welche große Kernelpages erlauben, etwa Alpha-Systeme, verwendet werden.
Jeder dieser Blöcke erhält eine Adresse, welche beim ersten Block im Dateisystem bei 0 beginnt, und ist einer Blockgruppe zugeteilt, ausgenommen jener Blöcke, die sich im reservierten Bereich nach dem Superblock befinden. Für das Herausfinden, zu welcher Blockgruppe ein Block gehört, kann nach [Car05 Seite 409] folgende Formel verwendet werden:

```
group = (block – FIRST_DATA_BLOCK) / BLOCKS_PER_GROUP
```

Damit für das Speichern neuer Daten geeignete Blöcke gefunden werden, wird, wie zuvor in Kapitel 4.3.2.1 beschrieben, in der Blockbitmap nach freien Blöcken gesucht. Obwohl jedes Bit in der Blockbitmap genau einem Block entspricht, muss zuerst die Adresse eines Blocks in Relation zum Beginn der Blockgruppe ermittelt werden. Diese Adresse kann auch als *logical group address* bezeichnet werden. Die Berechnung des ersten Blocks einer Blockgruppe kann nach [Car05 Seite 409] mit folgender Formel durchgeführt werden.

```
first_block = group * BLOCKS_PER_GROUP + FIRST_DATA_BLOCK
```

Der Offset eines Blockes innerhalb einer Blockgruppe ergibt sich anschließend als

```
Offset = block – first_block
```

Zum Verständnis der beiden Formeln ein kleines Beispiel. Bei einer Blockgröße von 4096 Bytes und 32768 Blöcke pro Blockgruppe und der Annahme, dass kein reservierter Bereich existiert, ergibt sich für den Block 45000 ein Offset von 12232 Bits. Daraus ergibt sich, dass sich das Bit für diesen Block im Byte 1529 des Blocks befindet.

Zuteilungsalgorithmus

Das Zuteilen von Blöcken erfolgt in Linux abhängig davon, für welchen Zweck ein Block benötigt wird. So wird etwa für das Zuteilen eines neuen Blocks für Inodes eine *first available* Strategie verwendet und versucht einen Block in derselben Blockgruppe, wie die anderen zusammengehörigen Inodes zu finden. Benötigt eine existierende Datei mehr Platz, so wird eine *next available* Strategie verwendet. Des Weiteren gibt es ein Dateisystemfeature, bei welchem für Ordner im Vorhinein Blöcke zugeteilt werden, damit diese bei größerem Platzbedarf nicht fragmentiert werden. Mit diesen unterschiedlichen Vorgehensweisen wird versucht, die Bewegung des Schreib/Lesekopfs der Festplatte beim Lesen einer Datei zu minimieren. Sollte in einer Blockgruppe kein Platz mehr sein, so wird die Suche auf eine andere Blockgruppe ausgedehnt. Wird die Größe eines neu zugeteilten Block nicht zur Gänze benötigt, so werden die restlichen Bytes mit Nullen überschrieben um Daten im *slack space* zu vermeiden. Wobei die Art und Weise bei der Suche nach freien Blöcken nicht einheitlich, sondern vom verwendeten Betriebssystem abhängig ist, genau wie die Möglichkeit einen Teil des Dateisystems für den Root-Benutzer zu reservieren.

4.3.2.3 Metadata Kategorie

In dieser Kategorie werden alle Daten gespeichert, welche die Dateien und Ordner im Dateisystem beschreiben. Die primären Metadaten werden in Inodes, zusätzliche in Attributen gespeichert.

Inodes

Ein Inode (*Index Node*) ist eine Datenstruktur zum Speichern von Metadaten welche zu jeder Datei und jedem Ordner existiert. Alle Inodes besitzen dieselbe Größe, welche im Superblock angegeben wird, und eine Adresse beginnend bei 1. Ebenfalls im Superblock angegeben ist die Anzahl der Inodes, die jeder Blockgruppe zugeteilt und in dieser in der Inode Table gespeichert werden. Mit der Adresse des Inodes kann, über die folgende Formel aus [Car05 Seite 413], die Blockgruppe, in welcher dieses Inode gespeichert ist, berechnet werden.

```
group = (inode - 1) / INODES_PER_GROUP
```

Die ersten zehn Inodes sind typischerweise reserviert und als zugeteilt markiert. Es besitzt jedoch nur der Inode 2, welcher für das Root-Directory verwendet wird, eine spezielle Funktion. Von den Anderen wird etwa der Inode 1 für das Aufzeichnen beschädigter Blöcke und der Inode 8 für das Journal verwendet. Wobei diese im Kernel nicht speziell behandelt und

der verwendete Inode sowie die Adresse des ersten nicht reservierten Inodes im Superblock angegeben werden. Der erste nicht reservierte Inode, typischerweise Inode 11, wird häufig für den `lost+found` Ordner, der von Programmen, welche die Konsistenz des Dateisystems überprüfen, zum Speichern zugeteilter Inodes ohne Dateinamen verwendet wird, verwendet. Diese Inodes erhalten für diesen Zweck einen neuen Namen und werden anschließend in diesen Ordner verschoben.

Jeder Inode besitzt eine fixe Anzahl an Feldern, in denen die Dateigröße, Informationen über den Besitzer sowie Zeitstempel gespeichert werden. Sollten diese Felder nicht ausreichen, so gibt es die Möglichkeit zusätzliche Attribute, welche später beschrieben werden, zu verwenden. Die Dateigröße wird in einem 64-Bit Feld gespeichert, wobei jedoch beachtet werden muss, dass ältere Versionen nur 32-Bit verwenden und daher die Dateigröße auf 4 GB begrenzt ist. Genauer betrachtet wird bei den neueren Versionen ein ungenutztes Feld für die oberen 32 Bit verwendet. Sollte eine Datei im Dateisystem existieren, bei der für die Speicherung der Größe mehr als 32 Bit benötigt werden, so wird ein spezielles Flag, welches die read-only Kompatibilität anzeigt, gesetzt.

Für die Benutzerinformationen werden die, in Unix-Systemen für jeden Benutzer angelegten, Benutzer und Gruppen-IDs verwendet. Die Verknüpfung dieser IDs mit den dazugehörigen Namen erfolgt für die Benutzer-ID in der `/etc/passwd` Datei und für die Gruppen-ID in der `/etc/groups` Datei. Zu beachten ist jedoch, dass die in einem Inode gespeicherten IDs nicht bedeuten, dass diese Datei von dem dazugehörigen Benutzer angelegt wurde. Denn die IDs könne jederzeit, beispielsweise mit dem `chown` und `chgrp` Kommando, geändert werden, oder es kann auch vorkommen, dass in der `/etc/passwd` Datei kein Name zu einer Benutzer-ID gespeichert ist.

Zu den gespeicherten Zeitstempeln zählen die Zugriffszeit, *access time*, die Modifizierungszeit, *modification time*, die Änderungszeit, *change time*, und die Löschzeit, *deletion time*, welche in Sekunden seit dem 1. Jänner 1970 UTC gespeichert werden. Die *modification*, *access* und *change* Zeiten gibt es auch im UFS, wo sie als MAC-Zeiten zusammengefasst werden. Konkret bedeuten die verschiedenen Zeiten folgendes:

- *(last) access time*: Zeitpunkt, zu dem auf den Content einer Datei das letzte Mal zugegriffen wurde.
- *modification time*: Zeitpunkt, zu dem der Content einer Datei das letzte Mal geändert wurde.
- *change time*: Zu dieser Zeit wurden die Metadaten einer Datei zuletzt geändert.
- *deletion time*: Zu diesem Zeitpunkt wurde die Datei gelöscht.

Wird eine Datei neu angelegt, so werden die access, modification und change Zeiten auf die aktuelle Zeit gesetzt. Zu beachten ist jedoch, dass es in Linux und anderen Unix-Systemen mit dem `touch` Kommando einfach möglich ist, die access und modification Zeiten nach belieben zu ändern, jedoch mit Auswirkung auf die change Zeit.

Im *mode* Feld eines Inodes werden der Typ einer Datei und wesentliche Werte über Berechtigungen gespeichert. Dieses Feld ist von zentraler Bedeutung, da es in Linux bzw. Unix-Systemen ein weites Anwendungsgebiet für Dateien gibt. So werden Dateien neben der Verwendung als normale Dateien und Ordner ebenfalls für das Ansprechen von *block devices*, *character devices*, *named pipes*, *Unix sockets* und als *symbolic links* verwendet. Kurze Erklärung der verschiedenen Verwendungen:

- *block devices*: Geräte wie Festplatten oder USB-Sticks, von welchen Daten nur als Vielfaches der physikalischen Speicherblöcke gelesen oder geschrieben werden können.
- *character devices*: In diese Kategorie fallen etwa Tastaturen oder andere Geräte, auch *row devices* genannt, die zeichenweise gelesen bzw. beschrieben werden können.
- *named pipes*: Diese Funktionalität wird für das Austauschen von Informationen zwischen Prozessen verwendet und ist als unidirektionale Kommunikation nach dem FIFO-Prinzip implementiert. In dem ein Prozess die „Datei“ öffnet und darin Daten speichert, werden diese anderen Prozessen zur Verfügung gestellt und können anschließend gelesen werden. Gespeichert werden die Daten dieser Dateien nicht direkt auf der Festplatte, sondern im Kernelspeicher.
- *Unix sockets*: Diese Funktionalität wird ebenfalls für das Austauschen von Informationen zwischen Prozessen verwendet, ist jedoch als bidirektionale Kommunikation implementiert. Wie bei den *named pipes* werden die Daten hier ebenfalls nicht auf der Festplatte gespeichert.
- *symbolic link*: Dieser Datei Typ stellt einen Softlink zu einer anderen Datei oder einem anderen Ordner dar.

Hardware-Geräte werden zwar als Dateien angezeigt, enthalten jedoch nur einen oder mehrere Dateinamen um auf diese zuzugreifen. In den Inodes dieser Dateien werden weitere Informationen zu den Geräten gespeichert.

Zu den, neben dem Dateityp, gespeicherten Berechtigungen zählen Rechte für das Lesen, Schreiben und Ausführen von Dateien, welche für den Besitzer, die Benutzergruppe und alle Benutzer gespeichert werden. Durch die Aufteilung der Berechtigungen kann zum Beispiel

angegeben werden, dass nur ein bestimmter Benutzer oder eine bestimmte Benutzergruppe auf eine Datei oder einen Ordner lesend und schreibend zugreifen darf. Es ist aber der Implementierung des Betriebssystems überlassen, ob es diese Berechtigungen anwendet oder nicht.
Weiters beinhaltet das *mode* Feld des Inodes Bits für spezielle Aufgaben. Zu diesen Bits gehört das *sticky bit*, welches bei einer ausführbaren Datei angibt, dass nach dem Beenden nicht alle Daten aus dem Speicher gelöscht werden. Angewandt auf einen Ordner verhindert es das Löschen des Inhalts durch alle Benutzer außer dem Besitzer. Soll ein Programm bzw. ein Prozess unter einem anderen Benutzer laufen, so kann über das Setzen des *set user ID* (SUID) und des *set group ID* (SGID) Bits angegeben werden, unter welchem Benutzer einer Gruppe der Prozess bzw. das Programm laufen soll. Über diesen Mechanismus ist es Benutzern mit geringeren Rechten möglich, Programme beispielsweise mit Administratorrechten zu starten. Angewendet auf Ordner bewirkt das SGID-Bit, dass alle Dateien in diesem Ordner dieselbe *group ID* zugewiesen bekommen.
Der Linkzähler eines Inodes speichert die Anzahl der Dateien, die auf diesen Inode zeigen und verhindert das Löschen des Inodes, falls der Wert größer als Null (>0) ist. Wobei der Zeitpunkt des Löschens davon abhängt, ob zu der Zeit, zu welcher der Link Zähler = 0 geworden ist, noch ein Prozess den Inode geöffnet hat oder nicht. Wenn ja, dann wird aus dem Inode ein verwaister Inode der erst nach dem Schließen gelöscht wird. Ansonsten wird dieser sofort gelöscht.
Ebenfalls beinhaltet jeder Inode eine Generierungs-ID, welche vom *Network File System* (NFS) benutzt wird, um herauszufinden, wann eine neue Datei angelegt wurde. Wird ein Inode einer neuen Datei zugeteilt, so wird der Wert dieser ID, ähnlich wie bei der *sequence number* in NFTS, erhöht. Zusätzlich ist diese ID in Linux mit einem *event counter* verknüpft.

Blockzeiger

Die Art und Weise wie in Ext3 die Daten über die benutzten Blöcke einer Datei gespeichert werden, zeigt, dass das Design dieses Dateisystem für die effiziente Verarbeitung kleiner Dateien ausgelegt ist. Aus diesem Grund werden pro Inode Zeiger auf die ersten 12 Blöcke einer Datei, welche *direkte Zeiger* / *direct pointer* genannt werden, gespeichert. Benötigt eine Datei mehrere Blöcke, so wird ein neuer Block, in welchem die weiteren Zeiger, als 4-Byte Adresse, gespeichert werden, zugeteilt. Der Zeiger auf diesen Block, welcher im Inode gespeichert ist, wird als *indirekter Blockzeiger* / *indirect block pointer* bezeichnet. Reichen die Blöcke, die auf diese Weise adressiert werden können, nicht aus, so wird ein *doppelt indirekter Blockzeiger* / *double indirect block pointer* verwendet. In diesem Fall enthält der Block, auf den der

Zeiger aus dem Inode zeigt, eine Liste weiterer Zeiger auf Blöcke, in denen Zeiger auf die Datenblöcke gespeichert werden. Reichen die so adressierbaren Blöcke noch immer nicht aus, besteht noch die Möglichkeit eines *dreifach indirekten Blockzeiger* / *triple indirect block pointer*. Dabei wird im Inode ein Zeiger auf einen Block, in welchem doppelt indirekte Blockzeiger gespeichert sind, gespeichert. Die folgende Abbildung in Anlehnung an [Car05 Seite 416 Figure 14.5] zeigt die Zusammenhänge der zuvor beschriebenen Zeigerstrukturen.

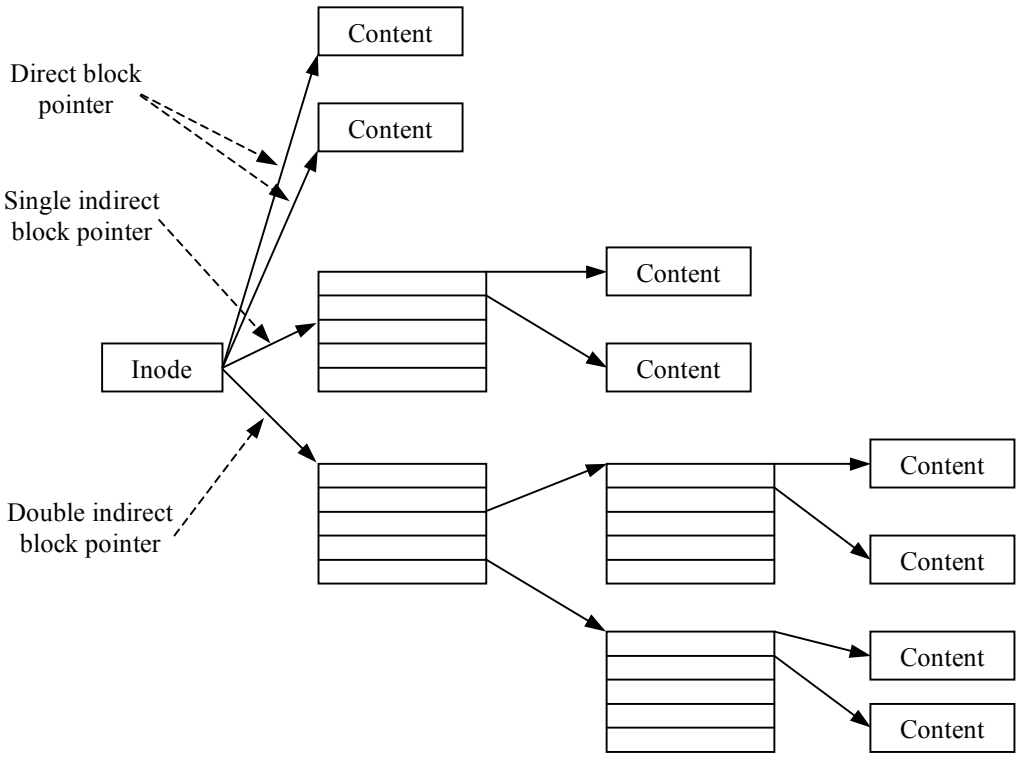


Abb. 4.20 Darstellung der verschiedenen Blockzeigervarianten

In Ext3 gibt es ebenfalls die Möglichkeit, Blöcke einer Datei, welche nur aus Nullen bestehen, als sogenannte *sparse* Blöcke zu markieren. Diese Markierung erfolgt einfach damit, dass die Adresse im Zeiger, der auf den Block zeigen soll, auf den Wert 0 gesetzt wird. Wenn so ein Block gelesen oder kopiert werden soll, werden die nicht gespeicherten 0en vom Betriebssystem generiert und der lesende bzw. kopierende Prozess merkt nicht, dass es sich bei diesem Block um einen *sparse* Block handelt. Im Unterschied zur Komprimierung auf Dateisystemebene kann ein Block nur dann zu einem *sparse* Block komprimiert werden, wenn der gesammte Block aus Nullen besteht.

Attribute

In Ext3 gibt es eine Menge von Attributen, welche in jedem Inode im *flag* Feld gespeichert werden und ein spezielles Dateisystemverhalten, beschränkt auf einzelne Dateien, erlauben. Zu den Funktionen die über Attribute aktiviert werden, zählen: Das sichere Löschen, das Wiederherstellen, das Komprimieren, das unmittelbare Schreiben auf die Festplatte (*synchronous updates*), das Einfrieren des aktuellen Dateiinhalts (*immutability*), das Einschränken der Editierfunktionen auf das Hinzufügen von Daten (*append-only*), das Verhindern des Speicherns bei der Ausführung des `dump` Kommandos (*dumpable*), das Verhindern der Aktualisierung der *access time* (*no-atime*), das Aktivieren indizierter Ordner (*indexed directories*) und das Aktivieren des Datenjournaling (*data-journaling*). Die Unterstützung dieser Attribute hängt, da einige experimentell sind oder nicht mehr verwendet werden, von der Version des Betriebssystems ab. Mit dem Kommando `lsattr` können die gesetzten Attribute einer Datei oder eines Ordners angezeigt und mit dem `chattr` Kommando geändert werden.

Zu den bereits erwähnten Attributen gibt es noch erweiterte Attribute, wobei diese unter Linux nur dann verwendet werden können, wenn das Dateisystem mit der `user_xattr` Option gemountet und ein Kernel ab Version 2.4 verwendet wird. Soll diese Option standardmäßig verwendet werden, so muss diese in der `/etc/fstab` Datei eingetragen werden. Ein erweitertes Attribut ist eine Liste von Paaren, bestehend aus einem Namen und einem Wert, welche vom Benutzer mit dem `setfattr` Kommando erzeugt werden können. Die erweiterten Attribute werden in einem Block, welcher zur Datei hinzugefügt wird, gespeichert. Verwenden mehrere Dateien dieselben erweiterten Attribute, so verwenden diese denselben Block.

Neben den Benutzern werden die erweiterten Attribute auch vom Betriebssystem zum Beispiel für die POSIX *access control lists* (ACL) verwendet. Diese ACL sind eine erweiterte Methode für das Beschreiben von Zugriffsrechten auf Dateien im Vergleich zu der Methode, die normalerweise in Ext3 verwendet wird. Konkret werden die ACLs anstelle von Unix-Gruppen für die Beschreibung der Zugriffsrechte verwendet. Mit dem Kommando `setfacl` kann ein Benutzter eine ACL für eine Datei angeben. Damit diese vom Betriebssystem berücksichtigt wird, muss das Dateisystem mit der Option `acl` gemountet werden.

4.3.2.4 File Name Kategorie

Diese Kategorie beschreibt die Art und Weise wie in Ext3 die Namen von Dateien und Ordnern gespeichert und diesen zugeordnet werden. Neben der normalen Zuordnung der Namen über die Ordnereinträge gibt es noch die Möglichkeit der Hard- und Softlinks sowie Hash-Trees.

Ordnereinträge

Wie zuvor im Abschnitt über die Inodes erwähnt, werden Ordner wie Dateien, bei denen im *mode* Feld des Inodes angegeben ist, dass es sich um einen Ordner handelt, gespeichert. Anstelle der Dateidaten werden in den Blöcken von Ordnern sogenannte Ordnereinträge / *directory entries* gespeichert. Diese Ordnereinträge sind einfache Datenstrukturen in welchen Dateinamen mit den dazugehörigen Adressen der Inodes, welche die Metadaten der Dateien enthalten, gespeichert werden. Die Größe eines Ordners hängt nur von der Anzahl der zugeordneten Blöcke, jedoch nicht von der Anzahl der Dateien, ab.

Die ersten beiden Ordnereinträge eines Ordners sind für die . und .. Ordner, welche für den Ordner selbst und für den Überordner stehen. Anschließend folgen die Ordnereinträge für die restlichen Dateien und Unterordner.

Der Name einer Datei kann zwischen Einem und 255 Zeichen lang sein und beeinflusst den Speicherplatz, den ein Ordnereintrag benötigt. Die Länge der Ordnereinträge ist die Summe aus der Länge des Namens und den benötigten 8 Byte der statischen Felder aufgerundet auf ein Vielfaches von 4. Da die Ordnereinträge sequenziell hintereinander gespeichert werden, beinhaltet jeder ein Feld, in dem seine Länge und damit der Beginn des Nächsten gespeichert werden, wobei der letzte Ordnereintrag auf das Blockende zeigt. Wird eine Datei oder ein Ordner gelöscht, so wird der Zeiger des vorherigen Ordnereintrags auf den nächsten Eintrag nach dem gelöschten Eintrag gesetzt. Durch dieses Vorgehen werden die Namen gelöschter Dateien oder Ordner nicht gelöscht, sondern lediglich beim Auflisten des Ordnerinhaltes ausgeblendet. Die soeben beschriebenen Zusammenhänge zeigt die folgende Abbildung.

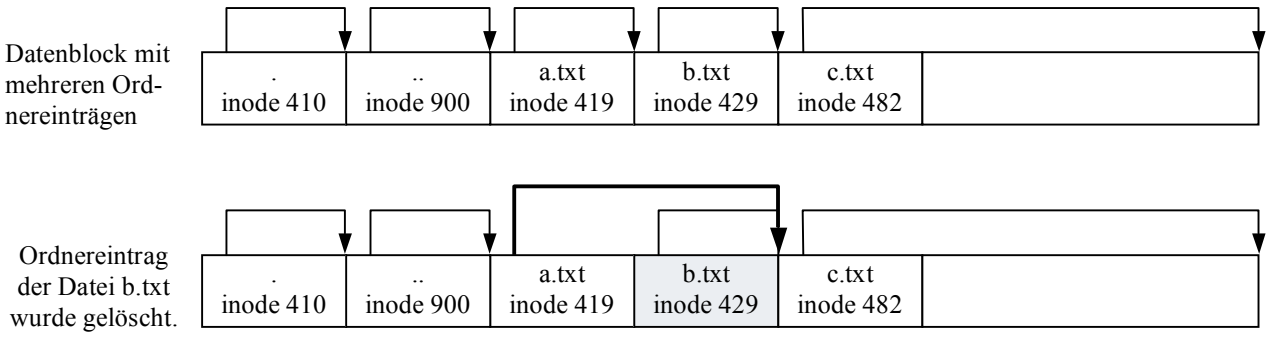


Abb. 4.21 Schematische Darstellung der Ordnereinträge in einem Datenblock vgl. [Car05 Figure 14.6 Seite 424]

Wenn ein neuer Eintrag erstellt werden soll, vergleicht das Betriebssystem die Längen der vorhandenen Einträge mit der tatsächlichen Länge um einen freien Speicher für den neuen Eintrag zu finden.

Aktuell gibt es zwei Versionen von Ordnerstrukturen. Eine alte Version, die nur den Namen, die Inodeadresse und die Länge beinhaltet und eine neue Version, die ein Byte des Feldes, in dem die Länge des Dateinamens gespeichert ist, verwendet, um den Typ der Datei, wie etwa Datei, Ordner oder Block Device, zu speichern.

Hard/Softlinks und Mount-Poinsts

Durch das Unterstützen von Hard- und Softlinks in Ext3 besitzt der Benutzer die Möglichkeit, einer Datei oder einem Ordner mehrere Namen zuzuordnen. Ein Hardlink ist ein zusätzlicher Name für eine Datei oder einen Ordner im gleichen Dateisystem. Für das Anlegen eines Hardlinks erzeugt das Betriebssystem einen neuen Ordnereintrag, lässt diesen auf den Inode der Datei zeigen und inkrementiert den Linkzähler im Inode um Eins. Nach dem Anlegen eines Hardlinks kann nicht mehr festgestellt werden, ob es sich um einen Link oder um den Originaldateinamen handelt. Wie schon erwähnt, wird eine Datei erst dann gelöscht, wenn alle Hardlinks gelöscht und somit der Linkzähler wieder auf Null gesetzt wurde. Die . und .. Einträge eines Ordners sind ebenfalls Hardlinks auf den Ordner selbst bzw. auf dessen Überordner.

Softlinks erzeugen ebenfalls weitere Namen für eine Datei oder einen Ordner. Im Unterschied zu den Hardlinks können diese jedoch über verschiedene Dateisysteme verteilt gespeichert sein bzw. auf Dateien oder Ordner eines anderen Dateisystems zeigen. Das Betriebssystem verwendet für die Erzeugung eines Softlinks eine Datei vom Type *symbolic link*, in welcher der Pfad auf die zu zeigende Datei bzw. den zu zeigenden Ordner gespeichert wird. Je nach Länge des Pfades wird dieser entweder im Inode, wenn der Pfad weniger als 60 Zeichen lang ist, oder in einem Datenblock der Datei gespeichert. Anders als bei einem Hardlink, kann bei einem Softlink die Datei auf die einer oder mehrere Softlinks zeigen immer gelöscht werden. Es wird dann nur vom Betriebssystem ein Fehler gemeldet wenn über den Softlink auf diese Datei zugegriffen werden soll. Nachdem das Erzeugen eines Softlinks die betreffende Datei bzw. deren Inode nicht verändert, gibt es auch beim Löschen eines Softlinks keine Einschränkungen. Das bedeutet, dass die Datei auf die der Softlink zeigt, nicht verändert wird und weiter existiert.

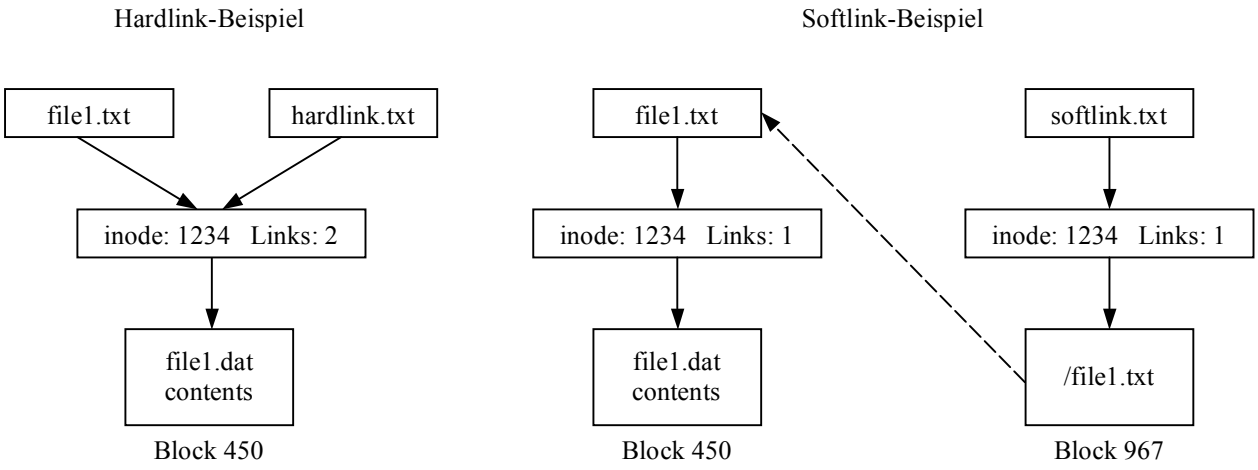


Abb. 4.22 Beispiele eines Hard- und eines Softlinks vgl. [Car05 Figure 14.7 Seite 427]

Hash-Trees

Bei der Erzeugung des Dateisystems kann der Benutzer auswählen, dass anstelle der ungeordneten Liste Hash-Trees für die Organisation der Dateien verwendet werden sollen. Entschließt sich der Benutzer für die Hash-Trees, so wird im Superblock ein Flag für die Kompatibilität dieses Features gesetzt. Die Kompatibilität rührt daher, da die Hash-Trees ebenfalls Dateieinträge verwenden. Ein Hash-Tree kann mit den B-Trees, welche in NTFS Verwendung finden, verglichen werden. Der Unterschied besteht nur darin, dass bei Hash-Trees anstelle der Dateinamen die Hashwerte dieser für die Sortierung verwendet werden.

Organisiert ein Ordner seine Ordnereinträge mit einem Hash-Tree, so benötigt dieser mehrere Blöcke, welche die Knoten in diesem Tree darstellen. Jeder dieser Knoten enthält Dateien, bei denen der Hashwert des Dateinamens in einem bestimmten Bereich liegt. Der erste Block eines Ordners wird der Wurzelknoten und beinhaltet neben den Einträgen der . und .. Ordner noch Knotendeskriptoren in welchen ein Hashwert und eine Blockadresse gespeichert sind. Diese Knotendeskriptoren werden vom Betriebssystem dazu verwendet, zu einem Hashwert den dazugehörigen Block zu finden.

Der Vorteil besteht nun darin, dass Betriebssysteme, die Hash-Trees nicht unterstützen, den Inhalt eines Ordners einfach sequenziell, ohne die Sortierung zu bemerken, auslesen können.

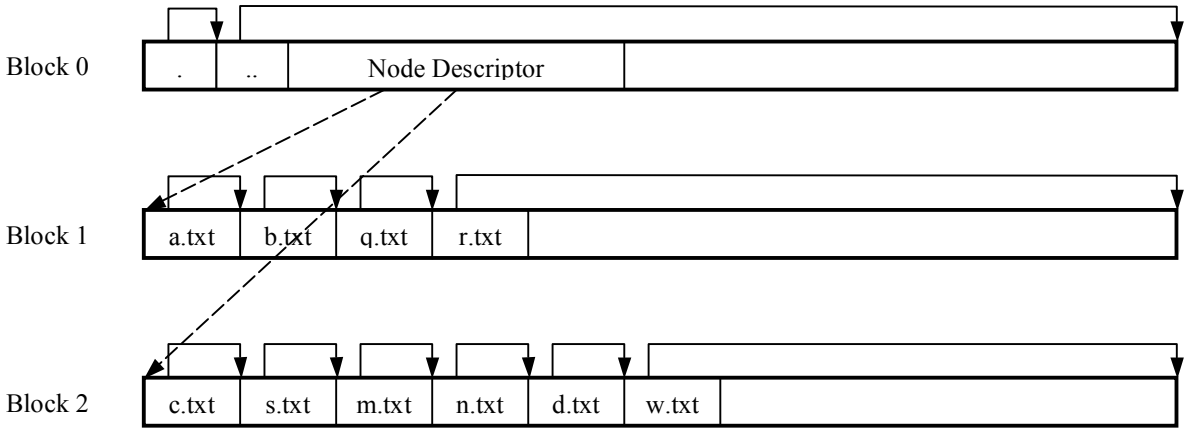


Abb. 4.23 Darstellung eines Ordners der 3 Blöcke für das Speichern eines Hash-Trees verwendet. vgl. [Car05 Figure 14.9 Seite 429]

4.3.2.5 Application Kategorie

Von den möglichen Features dieser Kategorie wurde in Ext3 nur das Dateisystemjournaling ins Dateisystem integriert. Features wie Quotas verwenden für das Speichern ihrer Daten normale Dateien.

Dateisystemjournaling

Typischerweise wird in Ext3 der Inode 8 für das Speichern des Journals verwendet, es kann jedoch im Superblock ein anderer Speicherort angegeben werden. Wird das Journaling verwendet, so wird im Superblock ein Kompatibilitäts-Flag für dieses Feature gesetzt, außer, es wird im Superblock angegeben, dass ein externes Journal verwendet wird. In diesem Fall wird ein Inkompatibilitäts-Flag im Superblock gesetzt.

Das Journaling in Ext3 arbeitet blockorientiert, was bedeutet, dass bereits die Änderung eines einzigen Bits in einem Inode dazu führt, dass der gesamte Block, in welchem das Inode gespeichert ist, im Journal gespeichert wird. Ein anderer Ansatz wäre der *Record Level* Ansatz, bei welchem nur die jeweiligen Inodes, anstelle der gesamten Datenblöcke, im Journal gespeichert werden.

Der erste Block des Journals beinhaltet den Superblock des Journals, in welchem allgemeine Informationen gespeichert werden. Die weiteren Blöcke, welche dieselbe Größe wie die Dateisystemblöcke haben sollten, werden für die Journaleinträge verwendet. Organisiert wird das Journal wie ein Ringspeicher was bedeutet, dass das Journal nur eine begrenzte Anzahl von Blöcken speichert und nach Erreichen des letzen Blocks von vorne mit dem Überschrei-

ben der ältesten Blöcke beginnt. Wobei in Linux das Journal bei jedem mounten des Dateisystems neu gestartet wird.

Alle Updates des Dateisystems werden im Journal als Transaktion, von denen jede eine sequenzielle Transaktionsnummer erhält, gespeichert. In den Blöcken des Journals werden entweder administrative oder Dateisystem-Updatedaten gespeichert. Jede Transaktion startet mit einem Deskriptorblock in welchem die Transaktionsnummer und eine Liste der Dateisystemblöcke, welche geändert wurden, gespeichert werden. Nach dem Deskriptorblock folgen die Datenblöcke, welche in der Liste des Deskriptorblocks aufgelistet sind. Den Abschluss der Transaktion bildet ein Commitblock mit derselben Transaktionsnummer. Die folgende Abbildung zeigt eine schematische Darstellung eines Journals mit zwei Transaktionen.

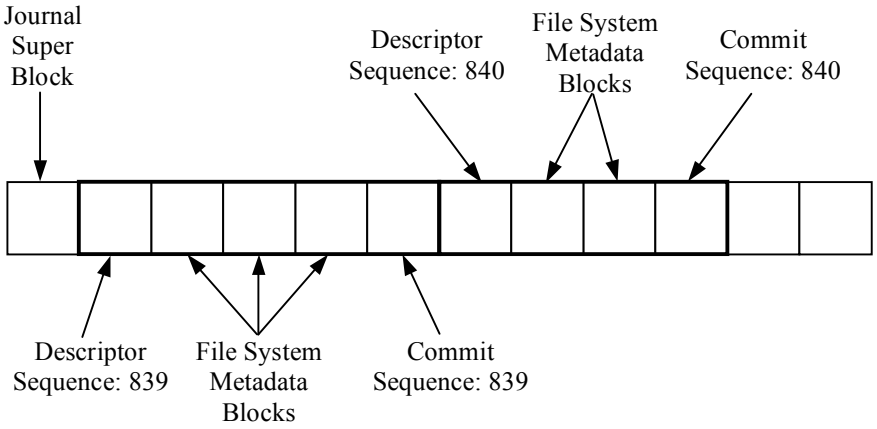


Abb. 4.24 Ext3 Journal Darstellung mit 2 Transaktionen [Car05 Figure 14.13 Seite 438]

Im Superblock des Journals wird die Position und die Transaktionsnummer des ersten Deskriptorblocks gespeichert. Sollte ein Deskriptorblock mit dem Speicher nicht auskommen, so kann ein weiterer Deskriptorblock mit der gleichen Transaktionsnummer zugeteilt werden. Es gibt auch die Möglichkeit einen Dateisystemblock, welcher im Journal gespeichert ist, von der Wiederherstellung im Fehlerfall auszuschließen. Dies geschieht unter Verwendung eines so genannten *annullier* bzw. *revoke* Blocks, welcher eine Transaktionsnummer und eine Liste der auszuschließenden Blöcke beinhaltet. Sollte aufgrund eines Crashs ein Recovery notwendig sein, so werden alle Blöcke, die im Revokeblock aufgelistet sind, nicht wiederhergestellt.
Details über die Datenstruktur des Journals können im Kapitel 15 in [Car05 ab Seite 449] nachgelesen werden.

In Ext3 gibt es drei verschiedene Modi, in denen das Journaling durchgeführt werden kann.

- *Full* (`data=journal`): Speichert die Metadaten und den Inhalt einer Datei zuerst im Journal und erst anschließend im Dateisystem. Die hohe Zuverlässigkeit wird durch

die verminderte Geschwindigkeit wegen des doppelten Zugriffs auf die Festplatte gemindert. Der große Vorteil dieses Modus ist, dass nach einem Crash Dateien, welche schon im Journal, jedoch noch nicht im Dateisystem gespeichert sind, wiederhergestellt werden können.

- *Writeback* (`data=writeback`): Dieser Modus speichert nur die Metadaten der Dateien im Journal. Wobei jedoch beachtet werden muss, dass Ext3 mit dem Commit der Journaleinträge nicht auf den Abschluss der Speicheroperation auf der Festplatte wartet. Der Vorteil der schnellen Verarbeitung wird durch die Möglichkeit, dass nach einem Crash Dateien alte Daten enthalten können, gemindert.
- *Ordered* (`data=ordered`): Arbeitet wie der Modus *Writeback*, jedoch mit dem Unterschied, dass das Commit im Journal erst nach dem Beenden der Speicheroperation auf der Festplatte gesetzt wird. Diese Stufe ist aufgrund der Kombination aus hoher Geschwindigkeit und Datensicherheit die Standardeinstellung in Linux.

Soll ein anderer Modus als der *Ordered* Modus verwendet werden, so kann dies entweder über die Option `-o` des `mount` Befehls, `mount -t ext3 -o data=writeback /dev/hda2 /mnt/hda2`, oder über einen Eintrag in der `/etc/fstab` Datei angegeben werden. Beispielsweise kann der vorherige `mount` Befehl wie folgt in der `/etc/fstab` Datei angegeben werden: `/dev/hda2 /mnt/hda2 ext3 data=writeback 1 0`

4.3.3 Möglichkeiten der Datenwiederherstellung

Wie zuvor schon bei den Dateisystemen FAT, Kap. 4.1.3, und NTFS, Kap. 4.2.3, beginnt dieses Kapitel bei Ext3 ebenfalls mit dem Löschvorgang von Dateien, um die anschließende Wiederherstellung leichter zu verstehen.

4.3.3.1 Löschen von Dateien

Soll eine Datei von der Festplatte gelöscht werden, so muss zuerst der Ordner, in dem sie gespeichert ist, über den Superblock und den genauen Pfad im Dateisystem gefunden werden. Sprich es müssen der Ordnereintrag, der Inode und die Blöcke, in denen die Daten gespeichert sind, gefunden werden. Nach der Lokalisierung der Datei wird der Zeiger des vorherigen Ordnereintrags so geändert, dass er entweder auf den Beginn des übernächsten Ordnereintrags oder auf das Ende des Blocks zeigt und so den Ordnereintrag der zu löschenden Datei aus dem Ordner entfernt, vergleiche dazu Abb. 4.21. Als nächstes wird der Linkzähler im Inode der Datei um 1 dekrementiert, und im Fall, dass dieser anschließend 0 ist, wird der gesamte

Inode freigegeben. Diese Freigabe erfolgt über das Setzen des dazugehörigen Bits in der Inodebitmap auf 0. Anschließend wird die Anzahl der freien Inodes im Deskriptorblock und im Superblock korrigiert. Die letzte Aufgabe besteht nun darin, die verwendeten Blöcke der Datei freizugeben. Dies geschieht über das Setzen des dazugehörigen Bits in der Blockbitmap auf 0 und durch das Löschen der Blockzeiger im Inode. Während des Freigebens der Blöcke wird nach jedem Block die Größe der Datei sowie die Anzahl der freien Blöcke im Superblock und im Gruppendeskriptorblock geändert.
Alle durchgeführten Änderungen werden im Journal gespeichert und bewirken Änderungen an den Zeitstempeln der Datei und des Ordners.

4.3.3.2 Wiederherstellung von Dateien

Nach dem Löschen einer Datei existiert der Zeiger vom Dateinamen auf den Inode zwar weiterhin, die Blockzeiger wurden jedoch während des Löschvorgangs überschrieben. Aus diesem Grund muss für das Wiederherstellen von Daten auf die Technik des *Data Carvings*, wie in Kapitel 3.3.3 beschrieben, zurückgegriffen werden. Da die meisten Betriebssysteme, die Datenblöcke einer Datei in der Blockgruppe, in welcher auch der Inode dieser Datei gespeichert ist, speichern, kann das *Data Carving* zuerst nur auf diese Blockgruppe beschränkt, und erst bei Misserfolg auf das gesamte Dateisystem ausgedehnt werden. Es sollte jedoch beachtet werden, dass die indirekten Blockzeiger noch zwischen den Daten existieren können. Wenn im Idealfall nur zusammenhängende Blöcke zugeteilt wurden, kann sich der erste indirekte Blockzeiger im dreizehnten Block befinden. Über die Blockgröße kann dann der nächste indirekte Blockzeiger berechnet und die Datei eventuell wiederhergestellt werden.
Ein weiterer Ansatz für das Wiederherstellen von Dateien ist, im Journal nach älteren Versionen von Inode-Blöcken zu suchen, und über diese den Inode einer Datei und damit die gelöschten Blockzeiger wiederherzustellen. Dies funktioniert jedoch nur bei Dateien die erst kürzlich gelöscht, und deren Einträge im Journal noch nicht überschrieben wurden.
Die Analyse des Dateisystemjournals beginnt beim Auslesen der Journal-Adresse aus dem Superblock. Nach der Lokalisierung des Journals erhält man über die Verarbeitung des Journal-Superblocks die Startadresse der ersten gültigen Transaktion. Der Deskriptorblock dieser Transaktion gibt an welche Blöcke in dieser Transaktion verändert wurden. Wird nun ein bestimmter Block gesucht, so kann dies über das Durchsuchen aller Deskriptorblöcke nach dem betreffenden Block erfolgen. Das Journal kann ebenfalls nach bestimmten Schlüsselwörtern oder Werten durchsucht werden. Anschließend kann über den Gruppendeskriptor der dazugehörige Block ausfindig gemacht werden. Handelt es sich dabei um den Block einer Inodetabelle, so kann darin noch der Blockzeiger einer gelöschten Datei enthalten sein. Falls das Lö-

schen der Datei nicht zu lange zurück liegt kann es möglich sein, dass diese Datei über den gefundenen Blockzeiger wieder gelesen werden kann. Es muss jedoch beachtet werden, dass das Journal durch das Speichern ganzer Blöcke viele alte Inodes enthält

5 Erstellen von Festplattenimages: Kurzanleitung

Dieses Kapitel beschreibt, wie die Images vor der Analyse erstellt werden. Wobei hier bewusst, um die Images mit jedem Forensik-Programm verwenden zu können, auf ein proprietäres Format verzichtet wird. Das Erstellen wird daher anhand der Verwendung des kleinen Unix-Programms `dd`, welches in Kapitel 2.3.2 beschreiben wurde, vorgestellt. Dieses Programm setzt jedoch eine gewisse Kenntnis über die Verwendung der Kommandozeile unter Unix/Linux und Windows, sowie Kenntnis darüber, wie bei den einzelnen Systemen auf Laufwerke zugegriffen werden kann, voraus.

5.1 Parameter für die Verwendung des Programms dd

Vor dem Eingehen auf konkrete Beispiele werden hier zuerst die wichtigsten Parameter für die Verwendung dieses Programms angeführt. Dazu sei gesagt, dass die kompletten Parameter in der Unix Manual Page für `dd`, über die Eingabe von `man dd` angezeigt werden können.

```
if= ... Eingabe Datei / Device
of= ... Ausgabe Datei / Device
bs= ... Blockgröße
skip= ... Anzahl der zu überspringenden Blöcke in der Eingabe
seek= ... Anzahl der zu überspringenden Blöcke in der Ausgabe
count= ... Anzahl der zu verarbeitenden Blöcke
```

Für das Erzeugen von Images sind primär die beiden ersten Parameter wichtig, wobei diese durch die Angabe von Unix-Pipes ersetzt werden können. Für die genaue Verwendung siehe nachfolgende Beispiele.
Da es, wie in Kapitel 2.3 erwähnt, einige Versionen dieses Programms gibt, die sich im Funktionsumfang unterscheiden, gibt es bei diesen auch einige zusätzliche Parameter, die, da sie für die Übungsimages nicht relevant sind, hier nicht anfgeführt werden.
Wird dieses Programm unter Windows verwendet, muss zuerst mithilfe des in Windows enthaltenen Programms `mountvol` die korrekte Laufwerks- bzw. Partitionsbezeichnung ermittelt werden. Für das Laufwerk C: zum Beispiel durch die Eingabe der folgenden Befehlszeile:

```
mountvol C: /L
```

Beispiel Ausgabe:

```
\\?\Volume{c30db3c3-9b47-11db-a3e2-806d6172696f}\
```

5.2 Lokales Speichern

Für den Fall, dass die zu untersuchende Festplatte aus dem betroffenen Computer ausgebaut und im Labor in Ruhe untersucht werden kann, empfiehlt sich die Festplatte über ein Gerät, welches den Schreibzugriff auf das Laufwerk verhindert, an einen Computer anzuschließen und das Image dieser Festplatte lokal zu speichern. Dafür reicht es, wenn im Parameter `of` des Programms `dd` der Speicherort und der Dateiname auf einem lokalen Laufwerk angegeben wird.

Auf einem Linux-Rechner wäre das zum Beispiel:

```
dd if=/dev/hdd of=/mnt/forensic_images/fatHarddisk.dd bs=1M
```

Hier wird die gesamte Festplatte `hdd` auf einer lokalen Festplatte, welche unter dem Mountpoint `/mnt/forensic_images` eingehängt ist, in dem Image `fatHarddisk.dd` gespeichert. Über den Parameter `bs=1M` wird noch angegeben, dass die Daten in Blöcken der Größe 1 Mb kopiert werden.

Dasselbe würde unter Windows zum Beispiel wie folgt angegeben werden:

```
dd if= \\?\Volume{7b03db00-ccaf-11db-a850-806d6172696f}\
of=D:\forensicImages\fatHarddisk.dd bs=1M
```

Mit dem Programm `dd` ist es auch möglich, aus einem Festplattenimage einzelne Partitionen für eine getrennte Weiterverarbeitung zu extrahieren. Damit diese richtig extrahiert werden können, müssen zuerst Informationen über die vorhandenen Partitionen gesammelt werden. Für diesen Zweck kann man das Programm `mmls` aus der Programmsammlung des Sleuth Kit, siehe Kapitel 2.2.2, verwenden. Das folgende Beispiel zeigt, wie mit diesem Programm die Partitionstabelle eines Images angezeigt werden kann.

```
./mmls -t dos /Users/thomas/UNI/SS07/Diplomarbeit/Images/private-
Images/winme-fat32.dd
DOS Partition Table
Offset Sector: 0
Units are in 512-byte sectors

     Slot    Start        End          Length       Description
00:  -----   0000000000   0000000000   0000000001   Primary Table (#0)
01:  -----   0000000001   0000000062   0000000062   Unallocated
02:  00:00   0000000063   0008401994   0008401932   Win95 FAT32 (0x0B)
03:  -----   0008401995   0008404823   0000002829   Unallocated
```

Damit von diesem Image die FAT32-Partition extrahiert werden kann, sind zwei Informationen wichtig: 1. Wo beginnt die Partition und 2. Wo endet bzw. wie groß ist sie. Der folgende Aufruf des Programms `dd` zeigt, wie durch geschickte Wahl der Parameter diese Partition ohne Rechenaufwand aus dem Image extrahiert wird.

```
dd if=winme-fat32.dd of=fat32-partition.dd bs=512 skip=63 count=8401932
```

5.3 Entferntes Speichern

Kann die betroffene Festplatte nicht aus dem zu untersuchenden System ausgebaut werden, zum Beispiel wegen benötigter Hardware, kein Zugang oder Ähnliches, so können die Daten auch über eine Netzwerkverbindung übertragen werden. In Kombination mit dem Programm `netcat / nc` oder mit `cryptcat` für eine verschlüsselte Übertragung können die Daten, unter Verwendung von Unix-Pipes, zu einem beliebigen Computer im Netzwerk übertragen werden.
Damit man auf diesen Computern keine relevanten Daten zerstört, empfiehlt es sich, diesen Computer, sofern möglich, mit einer bootfähigen Linux CD / DVD zu booten und mit den Tools dieser CD die Images zu erstellen. Hierfür eignen sich hervorragend die in Kapitel 2.2 vorgestellten Forensik Boot CDs.
Zur Veranschaulichung ein Beispiel für die Verwendung von `nc` in Kombination mit Unix-Pipes.

Zuerst muss der zu empfangende Computer für die Kommunikation eingerichtet werden:

```
nc -l -p 8000 | dd of=/mnt/forensic_images/fatImage.dd
```

Nachdem der Empfänger eingerichtet wurde, kann das Senden mit folgendem Befehl gestartet werden:

```
dd if=/dev/hdd bs=1M | nc 192.168.1.15 8000
```

6 Festplattenimages für die Szenarien

Aufgrund der Forderung nach kleinen Images für die Szenarien wurden die Images mithilfe des Programms `Parallels Desktop` für `Mac OS X`, mit welchem virtuelle Maschinen erstellt und gestartet werden können, erzeugt. Möglich wird dies, da eine virtuelle Festplatte, die mit einer fixen Größe angelegt wurde, als Datei gespeichert wird und dieselbe Struktur wie eine normale Festplatte aufweist. Weitere Vorteile dieser virtuellen Festplatten sind die Initialisierung mit Nullen und die Möglichkeit, diese Festplatten während des laufenden Systems durch Schalten in den Stand-by-Modus zu untersuchen.

6.1 Anleitung zum Erstellen einer virtuellen Festplatte

Der folgende einfache Workshop zeigt, wie unter `Parallels Desktop` zu einer bestehenden virtuellen Maschine eine neue virtuelle Festplatte hinzugefügt wird.

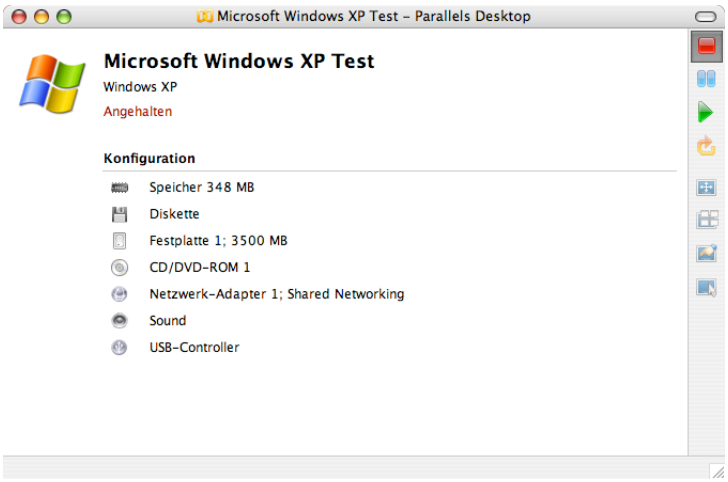


Abb. 6.1 Konfigurationsübersicht einer vorhandenen virtuellen Maschine

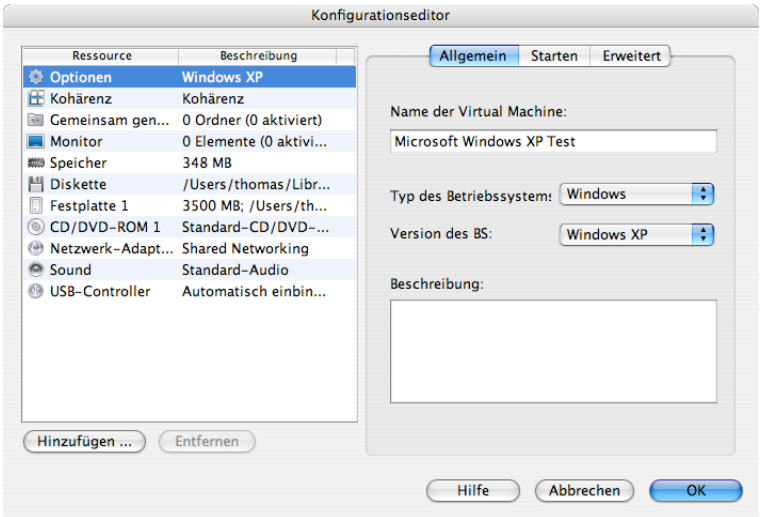


Abb. 6.2 Konfigurationseditor der virtuellen Maschine.

Durch Klicken auf "Hinzufügen" kann ein neues Gerät hinzugefügt werden.

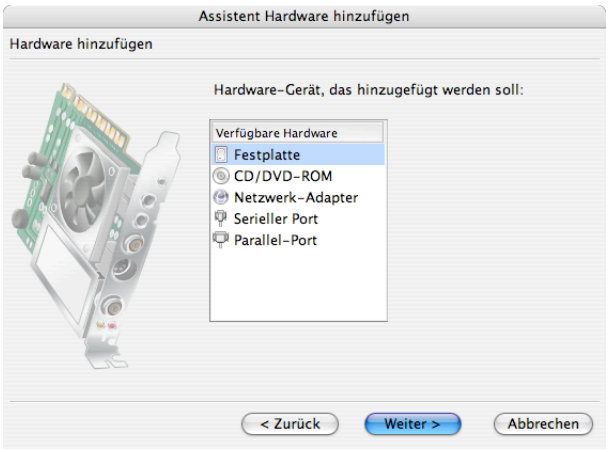


Abb. 6.3 Auswahl des zu hinzufügenden Gerätes.

In diesem Fall sollte man eine Festplatte hinzufügen.

Abb. 6.4 Angeben, dass eine neue virtuelle Festplatte angelegt werden soll.

Abb. 6.5 Angabe der Größe und der Art der virtuellen Festplatte.

Wird das einfache Format ausgewählt, so enthält die virtuelle Festplatte dieselbe Struktur wie eine physikalische Festplatte, und ist damit gleichwertig wie das Image einer Festplatte.

Abb. 6.6 Angabe des Speicherotes

Abb. 6.7 Erstellen der Festplatte

Der Assistent erzeugt nun eine Datei in der angegebenen Größe, welche nur 0en enthält.

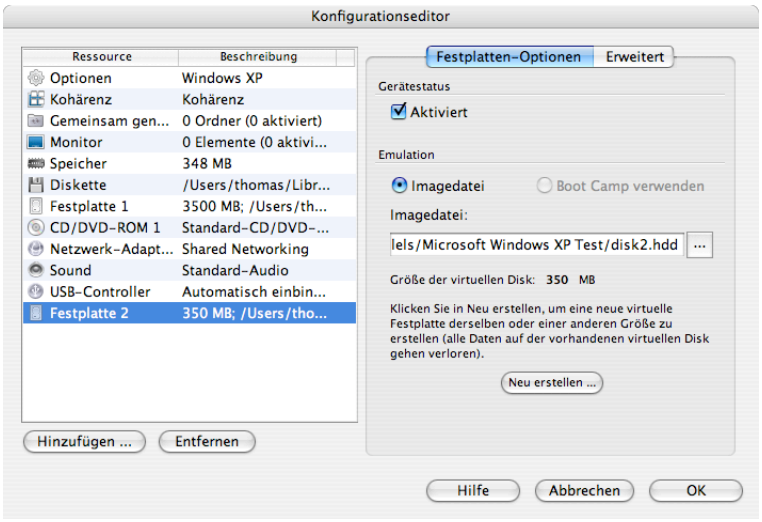


Abb. 6.8 Konfigurationseditor mit der neu erzeugten Festplatte

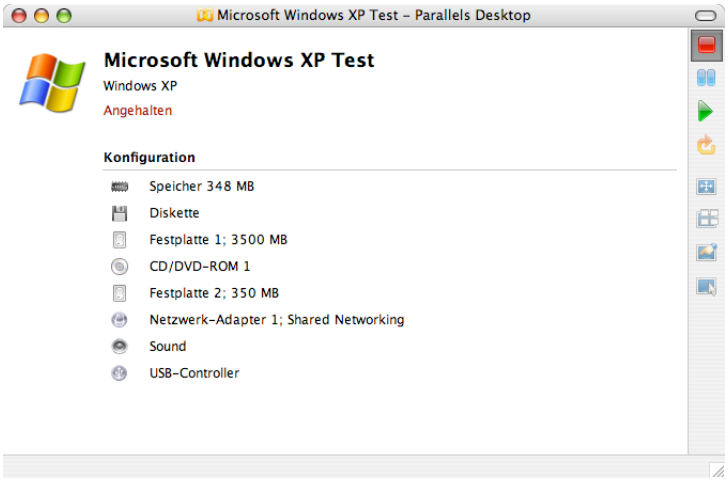


Abb. 6.9 Neue Konfigurationsübersicht

Nach dem Neustart dieser virtuellen Maschine wird die neu angelegte Festplatte vom Betriebssystem als neu erkannt und muss, genau wie eine neue physikalische Festplatte, partitioniert und formatiert werden. Unter Windows XP beispielsweise wird das mit der Datenträgerverwaltung im Programm `Computerverwaltung` durchgeführt.

Nachdem die Festplatte partitioniert und formatiert wurde, können die vorbereiteten Daten auf diese Festplatte kopiert und die geplanten Aktionen wie anlegen, ADS erstellen, löschen, verschlüsseln oder komprimieren durchgeführt werden. Die Resultate dieser Aktionen kann man anschließend mit einem in der virtuellen Maschine installierten Forensik-Programm analysieren. Wurden alle Aktionen zur Zufriedenheit ausgeführt, muss lediglich die Datei der virtuellen Festplatte kopiert und die Dateiendung, für eine leichtere Erkennbarkeit, in `dd` geändert werden.

7 Beschreibung der Übungsszenarien

Die Beschreibung der Übungsszenarien bildet den Hauptteil dieser Arbeit und beginnt mit einer Übersicht über die folgenden Szenarien.

7.1 Übersicht

Vor der detaillierten Beschreibung der Szenarien in Kapitel 7.2 gibt die folgende Tabelle einen Überblick über die Aufgaben der Szenarien.

	Imagename(n)	**Was ist zu tun**
Szenario 1: Einführung	fat-img-kw.dd, ntfs-img-kw-1.dd, ext3-img-kw-1.dd, daylight.dd, 6-fat-undel.dd, 7-ntfs-undel.dd	Einarbeitung in Forensik-Werkzeuge anhand der Images und Aufgaben von Brian Carrier. Dabei liegt der Schwerpunkt bei der Verwendung der Suchfunktion dieser Werkzeuge.
Szenario 2: FAT32 in Windows ME	Szenario-2-winme-FAT32.dd	Suchen und Wiederherstellen gelöschter Dateien, Ordner und Unterordner in einem Windows ME Image mit FAT32-Formatierung.
Szenario 3: FAT32 in openSUSE 10.3	Szenario-3-openSUSE103-FAT32.dd	Suchen und Wiederherstellen gelöschter Dateien, Ordner und Unterordner in einem openSUSE10.3 Image mit FAT32-Formatierung.
Szenario 4: Allocation-Algorithmus	Szenario-4-winxp-FAT32.dd Szenario-4-openSUSE103-FAT32.dd	Suchen und Wiederherstellen gelöschter Dateien um den Zuteilungs- / Allocation-Algorithmus des FAT-Treibers in Windows XP und/oder openSUSE10.3 zu vergeichen.
Szenario 5: NTFS mit ADS unter Windows XP	Szenario-5-winxp-NTFS.dd	Suchen und Wiederherstellen von Dateien im Allgemeinen und im Speziellen von ADS Daten in Dateien und Ordner.
Szenario 6: Einführung in EnCase Forensics	Hunter XP.E01	Bei diesem Szenario wird die Funktionalität des Programms `EnCase` für die Analyse ausgenutzt.
Szenario 7: EnCase vs. Konkurenz	Hunter XP.E01	Hier wird das Image von Szenario 6 mit einem anderen Forensik-Programm untersucht. Dabei werden zwei Aufgaben aus Szenario 6 durchgeführt und die Ergebnisse mit denen aus Aufgabe 6 verglichen.
Szenario 8: Live View	Szenario-8-winxp-NTFS.dd	Mit Hilfe von `LiveView` und `VMware Server` wird der Startvorgang eines Computers untersucht.

Szenario 9: Ext3 in openSUSE 10.3	Szenario-9-openSUSE103-ext3.dd	Suchen und Wiederherstellen gelöschter Dateien, Ordner und Unterordner in einem openSUSE10.3 Image mit Ext3-Formatierung.
Szenario 10: Verschlüsselung und Komprimierung in NTFS	Szenario-10-winxp-NTFS.dd	Analyse der Komprimierungs- und Verschlüsselungs-Funktion von NTFS unter Windows XP. Dabei liegt das Hauptaugenmerk darauf den Inhalt der verschlüsselten und komprimierten Dateien wenn möglich auszulesen.
Szenario 11: Recovery einer Formatierung	Szenario-11-winxp-NTFS.dd	Suchen und Wiederherstellen von Dateien, die vor einer Formatierung auf der Festplatte waren.
Szenario 12: Recovery beschädigter Sektoren	Szenario-12-winxp-NTFS.dd	Fehlerhafte Sektoren mithilfe eines Hexeditors wiederherstellen, um so das Image untersuchen zu können.

Tab. 7.1 Szenarienübersicht

7.2 Detailausarbeitung

7.2.1 Szenario 1: Einführung

- **Übungsangabe**

Brian Carrier, der Autor des Buches „File System Forensic Analysis“ hat eine Menge von kleinen Testimages unter der GPL Lizenz veröffentlicht, zu finden unter http://dftt.sourceforge.net. Diese Images eignen sich hervorragend für das Einarbeiten in und Testen von Forensik-Programmen, da er auch gleich eine Beschreibung der zu suchenden Daten beigefügt hat. Je nachdem, wie viel Zeit die Übungsteilnehmer für die Ausarbeitung eines Szenarios benötigen, können in einer Übungseinheit mehrere dieser Szenarien bearbeitet werden.

Für den Bericht sollten Sie dokumentieren, wie erfolgreich Sie diese Szenarien bearbeiten konnten, wurde alles gefunden, wurde etwas nicht gefunden, waren die Prüfsummen korrekt, welche Schwierigkeiten traten auf, etc.

- **Image Details**

Die Images für dieses Szenario befinden sich im Ordner `Szenario 1` der Images-DVD. Jedes Image befindet sich in einem eigenen Unterordner, welcher weiters eine Beschreibung zu dem Image und eine Kopie der GPL enthält.

7.2.2 Szenario 2: FAT32 in Windows ME

- **Übungsangabe**

Sie erhalten für dieses Übungsszenario ein Festplattenimage im raw / dd Format. Dieses Image ist ein genaues Abbild einer Festplatte im MBR-Layout[11], welches eine FAT32-Partition enthält die als Datenpartition verwendet und unter Windows ME formatiert wurde.
Die Festplatte, von der das Image erzeugt wurde, stammt von einer Polizeirazzia. Es wird vermutet, dass der Besitzer vor dem Eintreffen der Polizei möglicherweise ermittlungsrelevante Daten gelöscht hat. Über das genaue Verbrechen sind jedoch keine Details bekannt.
Ihre Aufgabe besteht nun darin, mit Hilfe eines Computer-Forensik-Programms nach diesen Daten zu suchen, und die für die weiteren Ermittlungen wichtigen Daten, wenn möglich, zu extrahieren und zu sichern. Weiters sollen Sie Ihre Arbeitsschritte dokumentieren und einen Bericht erstellen. Dafür sollen Sie die Möglichkeiten des verwendeten Forensik-Programms ausnutzen.

- **Image Details**

Name	`Szenario-2-winme-FAT32.dd`
Größe	`419.586.048 Byte`
MD5 Prüfsumme	`722F4A502E93B183C9ABDB563A5F1E9D`

- **Forensisch interessante Dateien und Ordner**

In diesem Abschnitt werden alle gelöschten Dateien und Ordner angegeben, welche im Zuge der forensischen Untersuchung gefunden werden sollen.

Angegebener Inhalt des Ordners gelöscht		
`\Dokumente\KV Angewandte Systemtheorie – Kryptographie\`		
Dateiname	Größe in Byte	MD5 Prüfsumme
`01 - Intro.pdf`	`555.147`	`DBB8A499D8A6FB200F65F3D6A2B1DC11`
`02 - DES.pdf`	`333.246`	`713A3D7C5179860C632818BA4368606F`
`03 - FromDES_ToAES.pdf`	`670.463`	`A442C23B2F7E8E68D7C973DDD2BD5935`
`04 - Streamciphers.pdf`	`370.164`	`9672DC6BE1B59C58CDE1D769577AD885`
`05 - AsymetricCrypto .pdf`	`327.872`	`1A6A5AC65DDCCE99868E22B51F8E3339`

[11] Der Ursprüngliche Name für diesen Partitionierungstyp war DOS Partition, doch mittlerweile wird dieser von Microsoft aber nur noch als MBR Partition bezeichnet.

06 - Signatures.pdf	511.361	3F2C6F4B2F9AFDC95F4CFECFB246FED5
07 - ECC.pdf	153.306	C5508A72297CB052F120E3B1D7068C79
08 - PGP.pdf	668.244	63E1AA9B564EAD28892261282EA42AC6
09 - ssl.pdf	406.166	3435C6244E44C36C35CB3485501C4B5C
10 – SecInternetPayment .pdf	1.413.042	4CE931E1D272F5676327A66A86D91026
HalloLeute.pdf	191.566	4AC54DAD2C20C9E48DFC7230B8A9705F
Mitschrift 29.11.06.doc	48.128	06547E9EAA747812A5CD9B0C41910FA5

Ordner + Inhalt gelöscht

\neue Dokumente\Systemtheorie 1\FMSCalc

Dateiname	Größe in Byte	MD5 Prüfsumme
FSMCalc.jar	167.638	E523A6B4607B61F753AC54AAAF88AF7A
FSMCalc.pdf	586.933	70CE8D6A978FB78C128C1530591C644E
FSMCalc.zip	1.208.407	687BE011172B57A6927DA1B974F4F930
jgraph.jar	157.121	E238E43F18BCBD0AFB020618BD610D18

Angegebene Dateien im Ordner gelöscht

\neue Dokumente\VL Pervasive Computing Infrastruktur

Dateiname	Größe in Byte	MD5 Prüfsumme
01 Pervasive Vision.pdf	5.397.571	12A534CECE924E784A277F9A4D64E8C4
04 reading 1 sarma+weis+ engels - rfid systems and security and privacy impli- cations.pdf	401	8386A4247CD7EA6963D47E45D38B1619
04 reading 2 roemer+schoch +mattern+duebendorfer - smart identification frame- works for ubiquitous comput- ing applications.pdf	401	8386A4247CD7EA6963D47E45D38B1619
04 reading 3 orr+abowd - the smart floor - a mechanism for natural user identifica- tion and tracking.pdf	401	8386A4247CD7EA6963D47E45D38B1619
05 reading 1 akyidiz2001_ wireless sensor networks_a survey.pdf	401	8386A4247CD7EA6963D47E45D38B1619
05 reading 2 cerpa2001_ habitat monitoring.pdf	401	8386A4247CD7EA6963D47E45D38B1619
05 reading 3 ewps2006_ platform survey.pdf	401	8386A4247CD7EA6963D47E45D38B1619

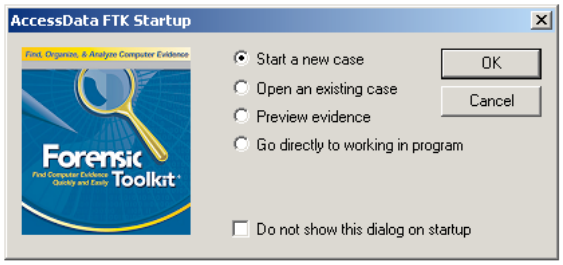
05 reading 4 haensel2006_ sensor networks.pdf	401	8386A4247CD7EA6963D47E45D38B1619
05 reading 5 inta2000_ directed diffusion.pdf	401	8386A4247CD7EA6963D47E45D38B1619
05 reading 6 krish2002_ critical density.pdf	401	8386A4247CD7EA6963D47E45D38B1619
06 a-survey-of-context.pdf	401	8386A4247CD7EA6963D47E45D38B1619
06 reading1 dey+abowd a conceptual framework.pdf	401	8386A4247CD7EA6963D47E45D38B1619
06 reading2 dey understanding and using context.pdf	401	8386A4247CD7EA6963D47E45D38B1619
06 reading3 chen+kotz a survey of context aware.pdf	401	8386A4247CD7EA6963D47E45D38B1619
07 reading 1 Molyneaux+ Kortuem - Ubiquitous Displays in dynamic environments - Issues and Opportunities.pdf	401	8386A4247CD7EA6963D47E45D38B1619
07 reading 2 plaue+miller +stasko - is a picture worth a thousand words - evaluation of information awareness displays.pdf	401	8386A4247CD7EA6963D47E45D38B1619
07 reading 3 ferscha+vogl - the webwall.pdf	401	8386A4247CD7EA6963D47E45D38B1619
09 PrehensileMovementsOfThe-HumanHand Napier 1956.pdf	401	8386A4247CD7EA6963D47E45D38B1619
09 reading1 fishkin taxonomy.pdf	401	8386A4247CD7EA6963D47E45D38B1619
09 reading2 sharlin et al TUIs humans spatiality.pdf	401	8386A4247CD7EA6963D47E45D38B1619
09 reading3 shaer et al TAC paradigm.pdf	401	8386A4247CD7EA6963D47E45D38B1619

Anmerkung: Bei den Dateien mit der Größe von 401 Bytes handelt es sich in Wirklichkeit um generierte Fehlermeldungen eines Webservers im HTML-Format. Diese Fehlermeldungen kommen daher, da es mit der Authentifizierung am Download-Server Probleme gab.

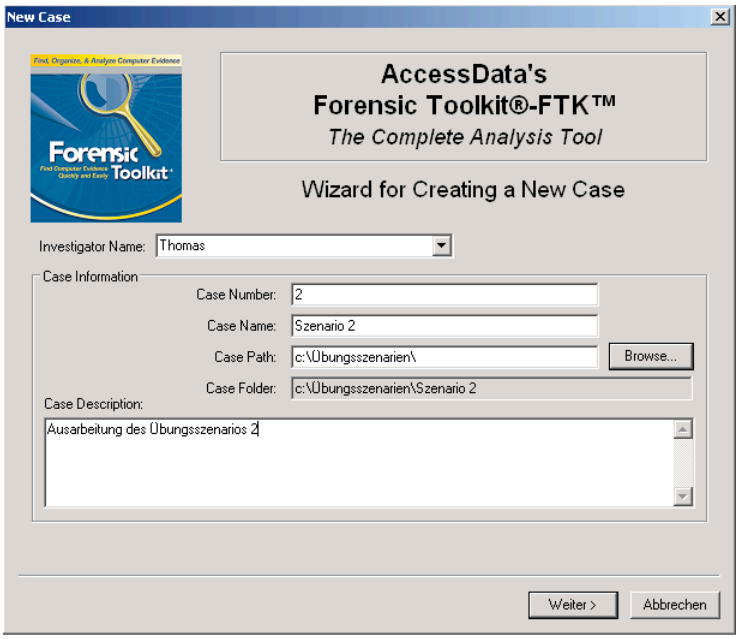
Ordner + Inhalt gelöscht		
\neue Dokumente\ KV Sicherheitsaspekte in der Informatik\12_IDS (Kofler)		
Dateiname	Größe in Bytes	MD5 Prüfsumme
12b_IDS_Executive-View.pdf	203.064	0AFA7EB6A74C220BD389BA486F5BFA03
12b_IDS_Executive-View_notes.pdf	193.585	293E7AA0B4E79236B344CDF317FE84B1
12c_IDS_Admin-View.pdf	2.888.255	3A0F50C6F3C68255680D1BE3C8EA9502
12c_IDS_Admin-View_notes.pdf	2.174.543	C90BCCC9648CA20760AF629911C2B3F3
12d_IDS_Admin-View_FAQs.pdf	154.155	8428BAAC0BE6B5DD7770ACDD1F978D23
12e_IDS_Expert-View.pdf	502.665	B0B9F452DAF4B36645A01513BBA088DC
12e_IDS_Expert-View_notes.pdf	423.861	E0BED097341932C34AE746B44A1B5632
12g_IDS_Literatur.pdf	397.393	EC17F82E8D2414CABB30D11E78593527

- **Forensische Untersuchung**

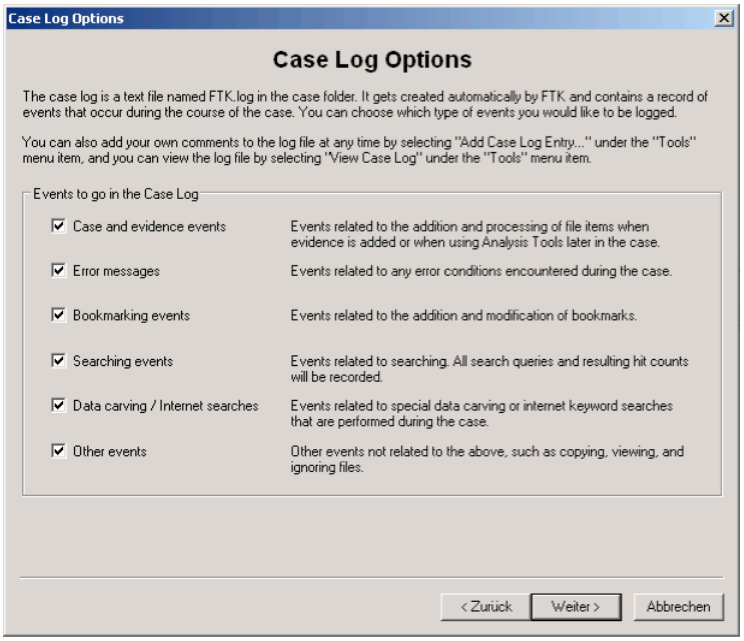
Damit man das Image dieses Szenarios untersuchen kann, müssen Sie einen neuen Fall anlegen. Für diese Aufgabe bietet das Programm `Forensic Toolkit` einen eigenen Wizard der nach dem Starten des Programms ausgeführt werden kann.



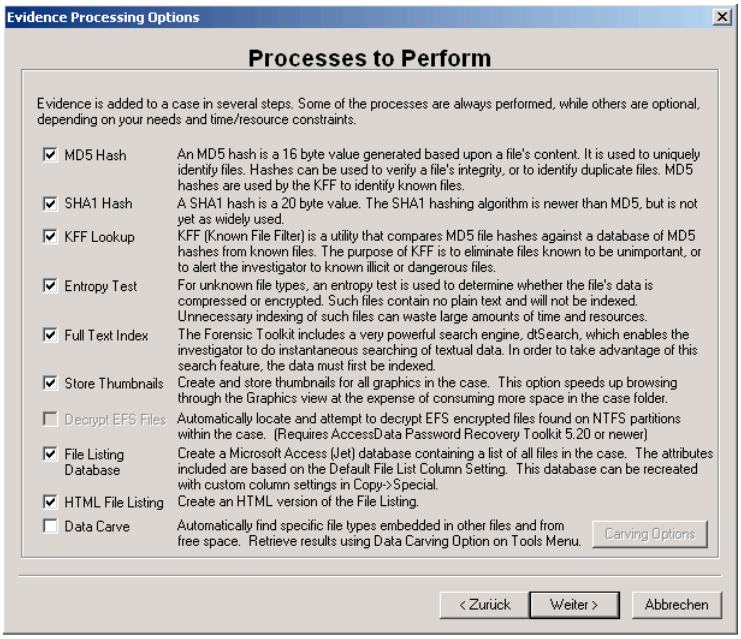
Die folgenden Screenshots zeigen die wichtigsten Schritte des Wizards, bei denen Einstellungen gemacht werden müssen, alle anderen Schritte können mit den vorgegebenen Werten übernommen werden, mit Ausnahme der Seite mit den Angaben zur untersuchenden Person.

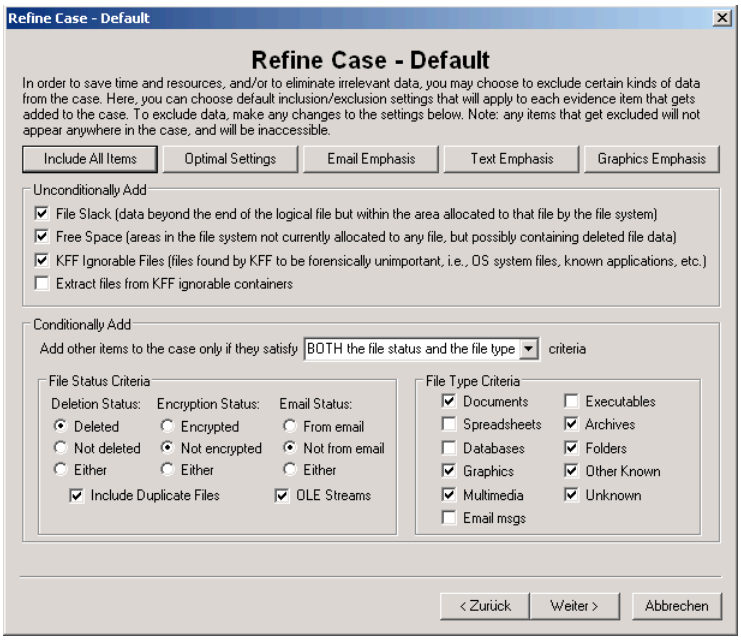


Zuerst müssen die allgemeinen Daten des neuen Falls angegeben werden.

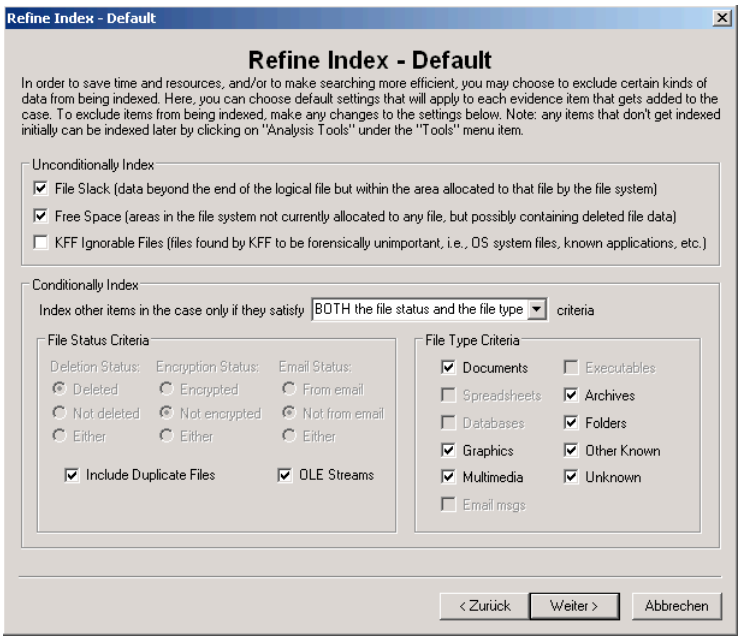


Nach der Angabe der Logging-Optionen werden jene Funktionen ausgewählt, welche nach Abschluss des Wizards durchgeführt werden sollen. Wobei hier aus Gründen von Zeit und Übersichtlichkeit auf die Funktion des `Data Carving` verzichtet wird.

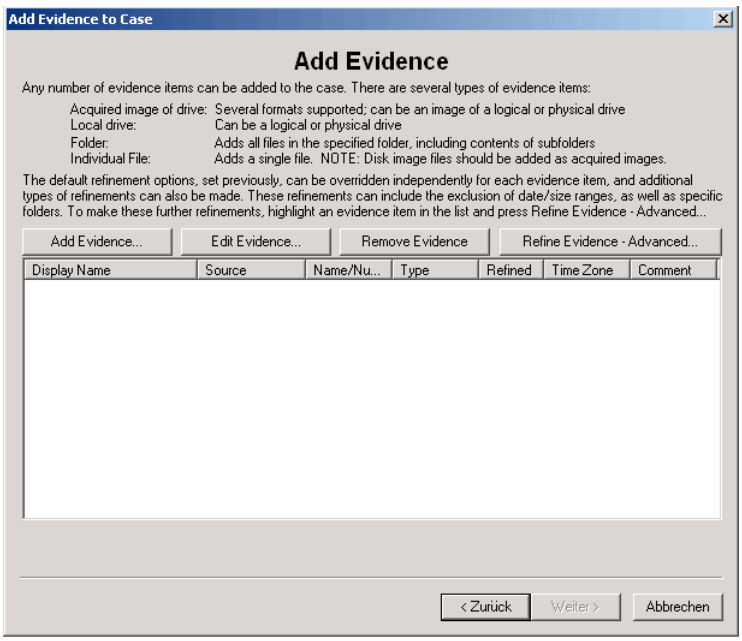


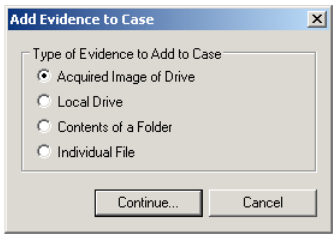


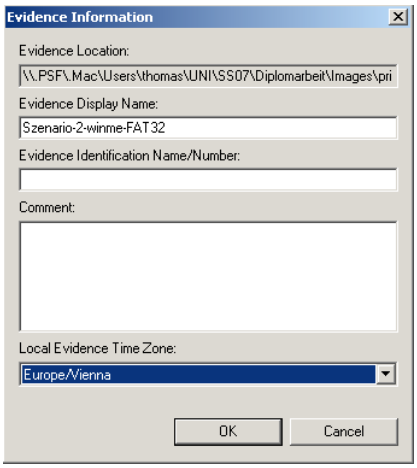
In diesem Schritt kann die Suche auf bestimmte Dateitypen und den Status einer Datei eingeschränkt werden. Die hier gezeigten Einstellungen sind für die in diesem Szenario zu suchenden Dateien optimiert. Die Optimierung liegt ganz einfach darin, die Suche nur auf die vorhandenen Dateitypen und auf gelöschte Dateien einzuschränken.

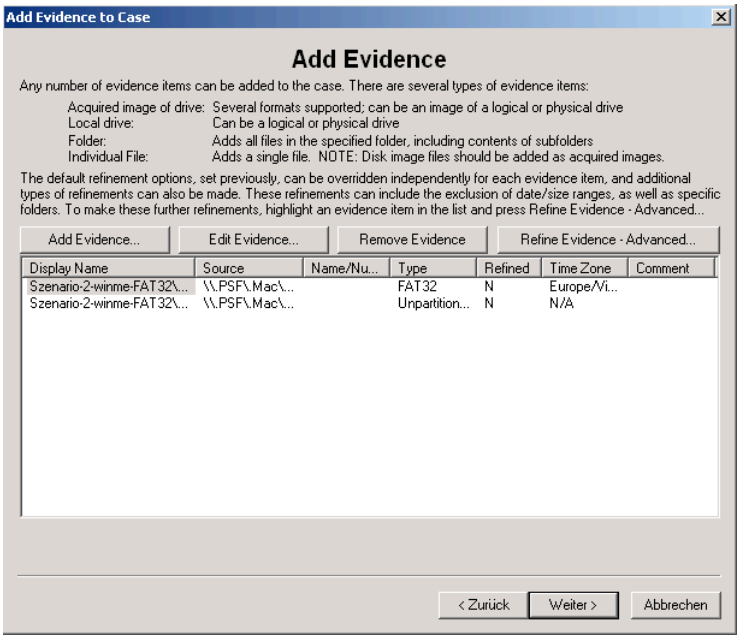


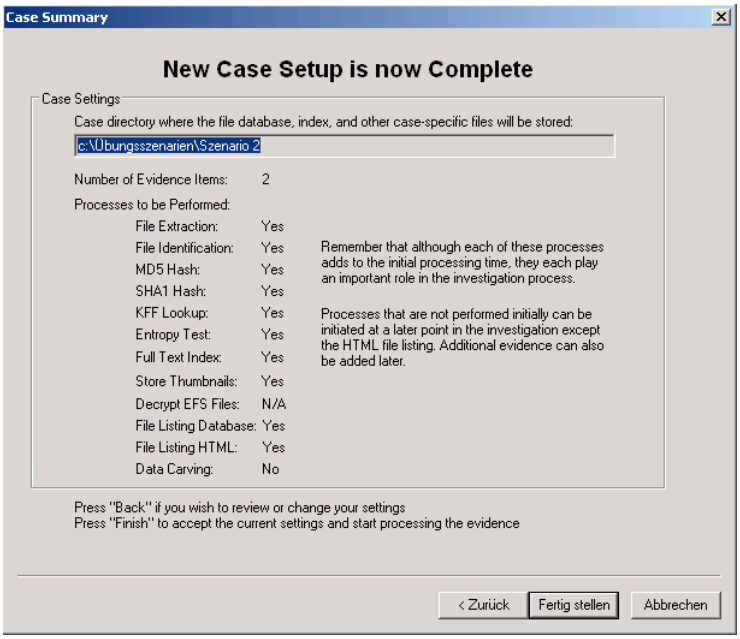
Nach dem Auswählen der Kriterien für den Index muss nur noch das Image durch Klicken auf den Button `Add Evidence...` als Beweis hinzugefügt werden. Vor dem Auswählen des Beweises muss noch entschieden werden welche Art von Beweis hinzugefügt werden soll. Da es bei diesem Szenario um die Untersuchung eines Images geht, ist hier „Aquierd Image of Drive“ auszuwählen. Mit dem Klicken auf den „Continue...“ Button öffnet sich ein Standardfenster des Betriebssystems für das Auswählen einer Datei. Bevor das Image endgültig als Beweis in die Liste im „Add Evidence“ Fenster aufgenommen wird, können im „Evidence Information“ Fenster noch ergänzende Informationen über den Beweis angegeben werden.









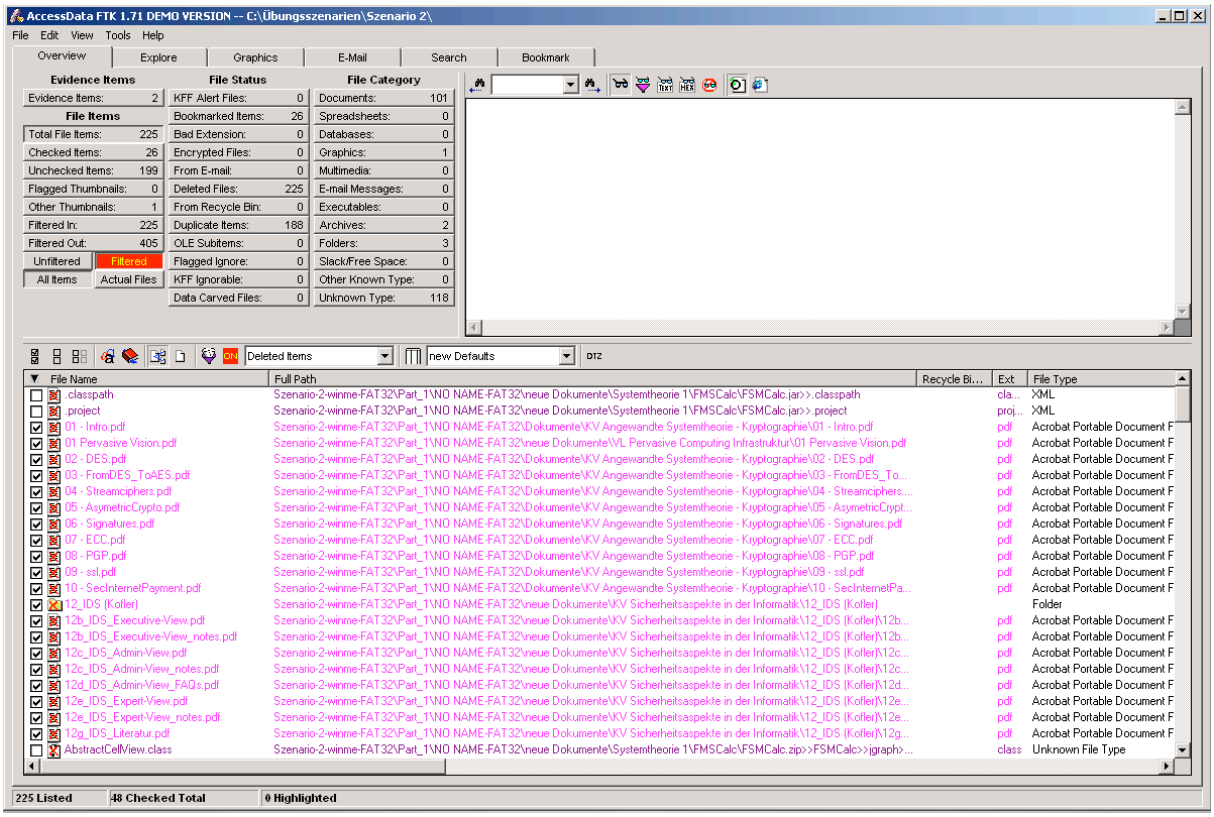


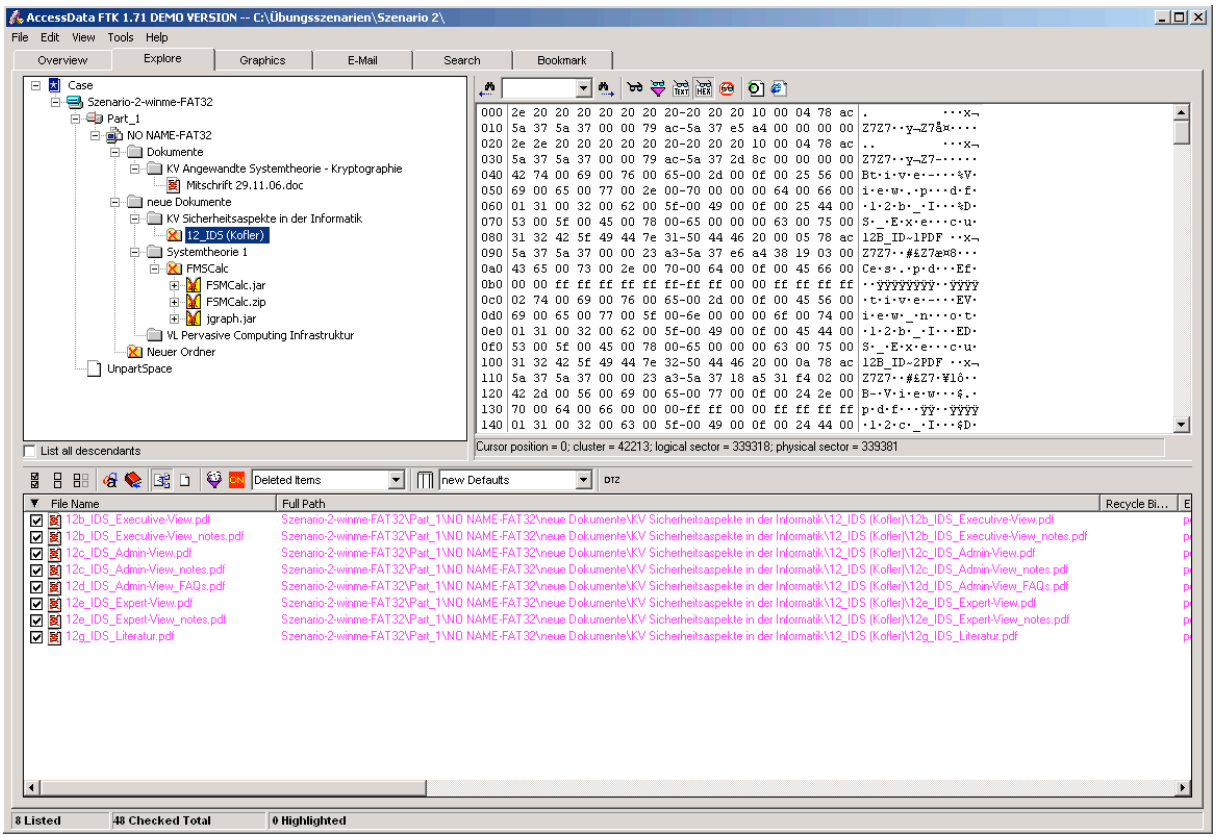
Nachdem alle Informationen angegeben wurden, kann über das Klicken auf den Button `Fertig stellen` mit der Analyse des Images begonnen werden.

- **Ergebnisse der forensischen Untersuchung**

Übersicht

Bevor die gefundenen Dateien und Ordner detailliert aufgelistet werden, zeigen die beiden folgenden Screenshots einen Teil der Ergebnisse wie sie im Programm `Forensic Toolkit` angezeigt werden. Der erste Screenshot zeigt die Übersichtsdarstellung mit einem Teil der gelöschten Dateien welche gefunden wurden. Im zweiten Screenshot ist die Explorerdarstellung mit bereits geöffneten Ordnern zu sehen. Bei einem dieser Ordner ist zusätzlich noch der gelöschte Inhalt, welcher bei derAnalyse gefunden wurde, zu sehen.





Liste der gefundenen Dateien und Ordner

Im Ordner `\Dokumente\KV Angewandte Systemtheorie - Kryptographie\` gefundene Dateien:

Dateiname	Größe in Byte	MD5 Prüfsumme	Fehlerfrei
01 - Intro.pdf	555.147	DBB8A499D8A6FB200F65F3D6A2B1DC11	ja
02 - DES.pdf	333.246	713A3D7C5179860C632818BA4368606F	ja
03 - FromDES_To AES.pdf	670.463	A442C23B2F7E8E68D7C973DDD2BD5935	ja
04 - Streamci-phers.pdf	370.164	9672DC6BE1B59C58CDE1D769577AD885	ja
05 - AsymetricC-rypto.pdf	327.872	1A6A5AC65DDCCE99868E22B51F8E3339	ja
06 - Signatures .pdf	511.361	3F2C6F4B2F9AFDC95F4CFECFB246FED5	ja
07 - ECC.pdf	153.306	C5508A72297CB052F120E3B1D7068C79	ja
08 - PGP.pdf	668.244	63E1AA9B564EAD28892261282EA42AC6	ja
09 - ssl.pdf	406.166	3435C6244E44C36C35CB3485501C4B5C	ja
10 - SecInter-netPayment.pdf	1.412.756	4CE931E1D272F5676327A66A86D91026	ja
HalloLeute.pdf	191.566	4AC54DAD2C20C9E48DFC7230B8A9705F	ja
Mitschrift 29.11.06.doc	48.128	06547E9EAA747812A5CD9B0C41910FA5	ja

Ordner `\neue Dokumente\ KV Sicherheitsaspekte in der Informatik\12_IDS (Kofler)` mit Inhalt gefunden.

Dateiname	Größe in Bytes	MD5 Prüfsumme	Fehlerfrei
12b_IDS_Executive-View.pdf	203.064	0AFA7EB6A74C220BD389BA486F5BFA03	ja
12b_IDS_Executive-View_notes.pdf	193.585	293E7AA0B4E79236B344CDF317FE84B1	ja
12c_IDS_Admin-View.pdf	2.888.255	3A0F50C6F3C68255680D1BE3C8EA9502	ja
12c_IDS_Admin-View_notes.pdf	2.174.543	C90BCCC9648CA20760AF629911C2B3F3	ja
12d_IDS_Admin-View_FAQs.pdf	154.155	8428BAAC0BE6B5DD7770ACDD1F978D23	ja
12e_IDS_Expert-View.pdf	502.665	B0B9F452DAF4B36645A01513BBA088DC	ja
12e_IDS_Expert-View_notes.pdf	423.861	E0BED097341932C34AE746B44A1B5632	ja
12g_IDS_Literatur.pdf	397.393	EC17F82E8D2414CABB30D11E78593527	ja

Ordner `\neue Dokumente\Systemtheorie 1\FMSCalc` mit Inhalt gefunden

Dateiname	Größe	MD5 Prüfsumme	Fehlerfrei
FSMCalc.jar	167.638	E523A6B4607B61F753AC54AAAF88AF7A	ja
FSMCalc.pdf	586.933	70CE8D6A978FB78C128C1530591C644E	ja
FSMCalc.zip	1.208.407	687BE011172B57A6927DA1B974F4F930	ja
jgraph.jar	157.121	E238E43F18BCBD0AFB020618BD610D18	ja

Dateien im Ordner `\neue Dokumente\VL Pervasive Computing Infrastruktur` gefunden

Dateiname	Größe in Bytes	MD5 Prüfsumme	Fehler-frei
01 Pervasive Vision.pdf	5.397.571	12A534CECE924E784A277F9A4D64E8C4	ja
04 reading 1 sarma+weis +engels - rfid systems and security and privacy implications.pdf	401	8386A4247CD7EA6963D47E45D38B1619	ja

04 reading 2 roemer+scho schoch+mattern+duebendorf er - smart identification frameworks for ubiquitous computing applications .pdf	401	8386A4247CD7EA6963D47E45D38B1619	ja
04 reading 3 orr+abowd - the smart floor - a me- chanism for natural user identification and tra- cking.pdf	401	8386A4247CD7EA6963D47E45D38B1619	ja
05 reading 1 aky- ildiz2001_wireless sensor networks_a survey.pdf	401	8386A4247CD7EA6963D47E45D38B1619	ja
05 reading 2 cerpa2001_ habitat monitoring.pdf	401	8386A4247CD7EA6963D47E45D38B1619	ja
05 reading 3 ewps2006_ platform survey.pdf	401	8386A4247CD7EA6963D47E45D38B1619	ja
05 reading 4 haensel2006 _sensor networks.pdf	401	8386A4247CD7EA6963D47E45D38B1619	ja
05 reading 5 inta2000_ directed diffusion.pdf	401	8386A4247CD7EA6963D47E45D38B1619	ja
05 reading 6 krish2002_ critical density.pdf	401	8386A4247CD7EA6963D47E45D38B1619	ja
06 a-survey-of-context .pdf	401	8386A4247CD7EA6963D47E45D38B1619	ja
06 reading1 dey+abowd a conceptual framework.pdf	401	8386A4247CD7EA6963D47E45D38B1619	ja
06 reading2 dey under- standing and using con- text.pdf	401	8386A4247CD7EA6963D47E45D38B1619	ja
06 reading3 chen+kotz a survey of context aware .pdf	401	8386A4247CD7EA6963D47E45D38B1619	ja
07 reading 1 Molyneaux +Kortuem - Ubiquitous Displays in dynamic envi- ronments - Issues and Op- portunities.pdf	401	8386A4247CD7EA6963D47E45D38B1619	ja

07 reading 2 plaue+miller+stasko - is a picture worth a thou-sand words – evaluation of information awareness displays.pdf	401	8386A4247CD7EA6963D47E45D38B1619	ja
07 reading 3 ferscha+vogl - the webwall.pdf	401	8386A4247CD7EA6963D47E45D38B1619	ja
09 PrehensileMovement-sOfTheHumanHand Napier 1956.pdf	401	8386A4247CD7EA6963D47E45D38B1619	ja
09 reading1 fishkin tax-onomy.pdf	401	8386A4247CD7EA6963D47E45D38B1619	ja
09 reading2 sharlin et al TUIs humans spatiality .pdf	401	8386A4247CD7EA6963D47E45D38B1619	ja
09 reading3 shaer et al TAC paradigm.pdf	401	8386A4247CD7EA6963D47E45D38B1619	ja

Ordner `Neuer Ordner` im Stammverzeichnis gefunden.

Dieser Ordner besitzt keinen Inhalt und wurde von Windows automatisch beim Erstellen eines neuen Ordners angelegt und nach dessen Umbenennung automatisch gelöscht.

- **Zusammenfassung**

Alle gelöschten Dateien und Ordner dieses Szenarios konnten korrekt gefunden und wiederhergestellt werden.

7.2.3 Szenario 3: FAT32 in openSUSE 10.3

- **Übungsangabe**

Für dieses Szenario erhalten Sie ein raw / dd Image einer Festplatte, welche aus einer FAT32-Partition besteht und in einem Computer, auf welchem openSUSE 10.3 installiert war, zum Speichern von Daten verwendet wurde.

Aus Versehen hat der Besitzer dieses Computers mehrere Ordner samt Inhalt gelöscht. Erst nach dem nächsten Einschalten bemerkte der Besitzer, dass er einige wichtige Dokumente gelöscht hatte.

Ihre Aufgabe besteht nun darin, mit einem Computer-Forensik-Programm nach diesen Dateien zu suchen und falls möglich wiederherzustellen. Ihre Tätigkeiten sollen Sie wieder zu einem Bericht zusammenfassen.

- **Image Details**

Name	Szenario-3-openSUSE103-FAT32.dd
Größe	157.409.280 Byte
MD5 Prüfsumme	69BA0F6AEAD441F1AA645B6679DEFE67

- **Forensisch interessante Dateien und Ordner**

In diesem Abschnitt werden alle gelöschten Dateien und Ordner angegeben, welche im Zuge der forensischen Untersuchung gefunden werden sollen.

Inhalt des Ordners gelöscht		
Dokumente\KV Angewandte Systemtheorie – Kryptographie		
Dateiname	Größe in Byte	MD5 Prüfsumme
01 - Intro.pdf	555.147	DBB8A499D8A6FB200F65F3D6A2B1DC11
02 - DES.pdf	333.246	713A3D7C5179860C632818BA4368606F
03 - FromDES_ToAES.pdf	670.463	A442C23B2F7E8E68D7C973DDD2BD5935
04 - Streamciphers.pdf	370.164	9672DC6BE1B59C58CDE1D769577AD885
05 - AsymetricCrypto.pdf	327.872	1A6A5AC65DDCCE99868E22B51F8E3339
06 - Signatures.pdf	511.361	3F2C6F4B2F9AFDC95F4CFECFB246FED5
07 - ECC.pdf	153.306	C5508A72297CB052F120E3B1D7068C79
08 - PGP.pdf	668.244	63E1AA9B564EAD28892261282EA42AC6
09 - ssl.pdf	406.166	3435C6244E44C36C35CB3485501C4B5C
10 - SecInternetPayment .pdf	1.413.042	4CE931E1D272F5676327A66A86D91026
HalloLeute.pdf	191.566	4AC54DAD2C20C9E48DFC7230B8A9705F
Mitschrift 29.11.06.doc	48.128	06547E9EAA747812A5CD9B0C41910FA5

Ordner + Inhalt gelöscht		
Dokumente\Seminar - Netzwerke – Security		
Dateiname	Größe in Byte	MD5 Prüfsumme
1.Intro.pdf	133.565	701D6EA87F7B72F3BA1C3994991B2822
2.Networks.pdf	376.297	EFB288CF96E3F492C77187DEFE526509

3.Vulnerability Analysis.pdf	380.177	751E164066D23341D0B10976F06B1530
4.NAM.pdf	214.495	25F279F91150AC677B7BB49A9411601D
5.Google Hacking.pdf	1.138.327	F497AD7B16F87D6C3DBC0F0695BF23D6
6.CVSS.pdf	551.583	9E2F4271606746E2683F2E8A674F689C
2005_20online.pdf	419.517	E14402CD517111948647227D90134631
8021X_IPsec.doc	231.424	2FD7EFA171E1A754C9310535F420E6DD
92216901.PDF	2.584.952	B2B38D4A3CDEBAD26E2B43BC916C09D8
cg0705.mspx.htm	39.029	26ADC88A153A96D8DE7E4CB67D14124D
diverse Links.txt	877	36C748CF6B43DCD2B0D97C5EA28AEA48
include.pdf	136.517	5B61D170CB1CCD3A1C5A751C58EAE85E
Mögliche Seminararbeitsthemen .doc	19.968	1D0E426E85A6E5FFF91D1C52E86DC36F
NAP_IPsec.doc	445.440	16FBD9EAA1C5469E87700FE4F648A049
NAP_Policy.doc	527.872	0F3D3F77BCF8C60340B882D068349067
NAP-Pic1.jpg	15.256	8129A1D9CA03F49A3E315559FC611040
NAP-Pic2.jpg	17.690	A729A8DD90F0C39DDD0E6F96621FC706
NAPArch.doc	470.016	EC31DE70C78A03FA11D457CFFF107C61
NAPIntro.doc	234.496	168FBE16DE76E0CC127510761CD30011
Open_Standards_for_Integrity-Based_AccessControl.pdf	220.249	4BEDBF72898AE287C898AF9BB787BBF1
Screen1-Detail.bmp	791.898	2B9C6C5B91EC8C42AC8BE2F4C4BD59F1
Screen1.bmp	2.359.350	60EC289902BF709F57F841A452A4E516
Screen2-Detail.bmp	605.238	DC6649B66B450D24D7E4BC53642FF98D
Screen2.bmp	2.359.350	5B441F14173278E3697EC2F0A1739824
Securtiy_CNAC_032004.pdf	436.806	4B4F0DB4FFD16830B9494FAB5B7D88B0
Seminararbeit.doc	1.003.520	475A4C1E52E4F804037D14C6A1E46DDB
TCG_1_0_Architecture_Overview .pdf	585.123	459AEC22DEE9C2B33F4A7717D968823A
The Laws of Vulnerabilities 2005 Edition.pdf	421.988	C36882E7747A6192935CBF217D7E9764
Thumbs.db	22.016	CE22CCD1AAE89D39FA8F24F3604DB772
TNC_Architecture_v1_0_r4.pdf	354.493	68FF5BE864433BC059C38C399819C303
TNC_NI-collateral_10_may_(2) .pdf	174.864	74943AAC6BC1955F5EBC671810F8F3F6

Ordner + Inhalt gelöscht		
Dokumente\Seminar - Netzwerke - Security\Cisco		
Dateiname	**Größe in Byte**	**MD5 Prüfsumme**
cdccont_0900aecd80217e26.pdf	2.396.587	8684A9C05065BA089CC58C87B97C1D38
cdccont_0900aecd80234ef4.pdf	602.417	622574D247BD75572C1186486591155F
Securtiy_CNAC_032004.pdf	436.806	4B4F0DB4FFD16830B9494FAB5B7D88B0

Ordner + Inhalt gelöscht		
Dokumente\Seminar - Netzwerke - Security\cg0705.mspx_files		
Dateiname	Größe in Byte	MD5 Prüfsumme
ADSAdClient31.htm	868	FC90E1EA430AD61272382AD33D915ECA
arrow_px_down.gif	53	A50A5A7BE6297D534E0A9559C583C217
arrow_px_up.gif	53	FBAE595C25E4EB510CE17F4EA17C8760
arrowLTR.gif	821	00BA9FF02078EE89D912B07B8E7D3838
broker.js	4.571	3978DB0D26A384B5622DE03A87C87C6F
cableguy.gif	30.792	0E94B6586D8A5D58C143F471C483F376
cg07050.jpg	15.256	8129A1D9CA03F49A3E315559FC611040
cg07051.jpg	17.690	A729A8DD90F0C39DDD0E6F96621FC706
css_002.css	2.572	3E1403BF57EE32B3F5D4377C5AC7A066
css.css	2.713	871B7056072A4C81CC42629B3940E1C3
gradient_002.jpg	869	C356C7D8C367BBBDCD157766585D1DC6
gradient.jpg	1.619	C64CD96C7A3A133C4B6991A1608B1144
menujs	34.334	67EDB22B47CF2D195B29CAF19AAB7EB7
ratings.htm	8.697	DDDD7A9B0F0011842925FCAEB35F1C71
SiteRecruit_PageConfigura-tion_2943mt33-3089mt-2944mt 1.js	2.910	865E579A119E41BF2993417676BACAAA
TechNetB_masthead_ltr.gif	3.576	5249D778EB3E80146D0A6401395300DC
templatecss.css	11.290	B5534574022FFDAA179790AC3B09BEF3
text.jpg	1.908	CA86E0AE62452F4F1103D034FE337B58
Thumbs.db	8.192	22FCDA2241363F882061ECBEFAC8B204
trans_pixel.gif	44	6D69C4DE0545F6FC9BA3DCD899E29501

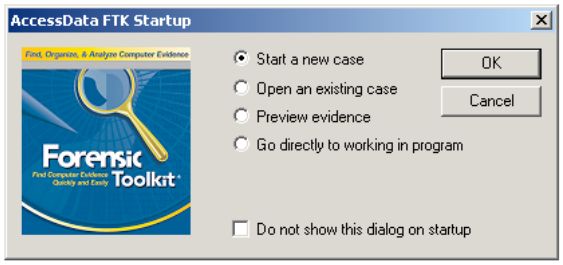
Ordner + Inhalt gelöscht		
Dokumente\Seminar - Netzwerke - Security\cg0705.mspx_files\ADSAdClient31 _data		
Dateiname	Größe in Byte	MD5 Prüfsumme
0000053432_0000000000000002 65828.gif	10.099	14A168D6573518C13C57971CDE8850EB

Ordner + Inhalt gelöscht		
Dokumente\Seminar - Netzwerke - Security\cg0705.mspx_files\ratings_data		
Dateiname	Größe in Byte	MD5 Prüfsumme
ContentRating.css	420	F02DC615EF1E9293C74330D6D59BB14D
rtg_email.gif	1.010	B201A23EC07D55866FAB17A62030774E
rtg_print.gif	574	101A0AC63030B36FCE0421D896940AEF
rtg_save.gif	567	7D20B965EA147E6DC2B580CE6AE71778

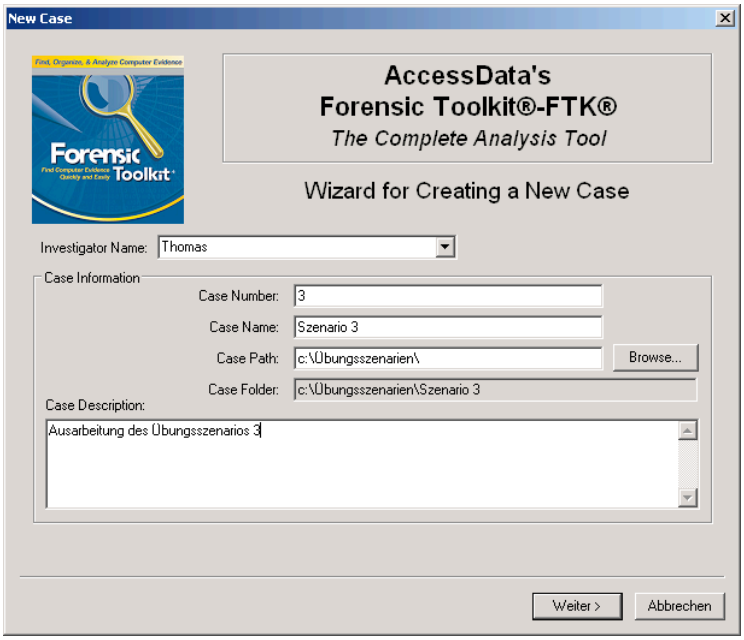
Dateien im Ordner gelöscht		
Dokumente\SDK-Docs-Tutorials		
Dateiname	Größe in Byte	MD5 Prüfsumme
Java_Bücherliste.doc (Image022_bearbeitet.jpg)	470.012	71DA5B4AC0C7E5E966EE1B8AAE08E00E
how_to_use_the_javadoc.doc (Image030.jpg)	978.997	94F608C1B3C764C6FC0C4B5DEA9F609E

- **Forensische Untersuchung**

Damit man das Image dieses Szenarios untersuchen kann, müssen Sie einen neuen Fall anlegen. Für diese Aufgabe bietet das Programm `Forensic Toolkit` einen eigenen Wizard der nach dem Starten des Programms ausgeführt werden kann.



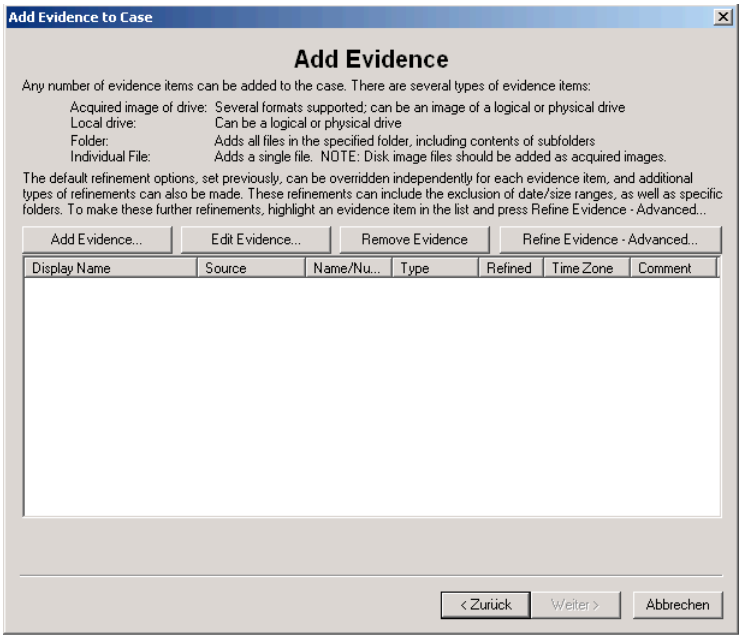
Die folgenden Screenshots zeigen die wichtigsten Schritte des Wizards bei denen Einstellungen gemacht werden müssen, alle anderen Schritte können mit den vorgegebenen Werten übernommen werden, mit Ausnahme der Seite mit den Angaben zur untersuchenden Person.

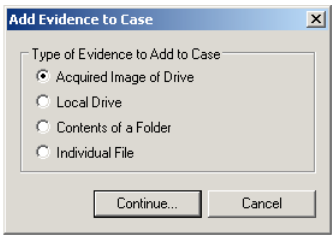


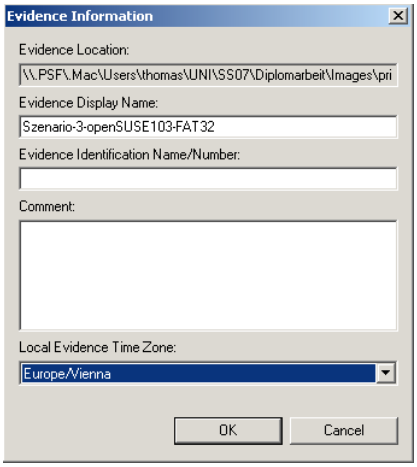
Zuerst müssen die allgemeinen Daten des neuen Falls angegeben werden.

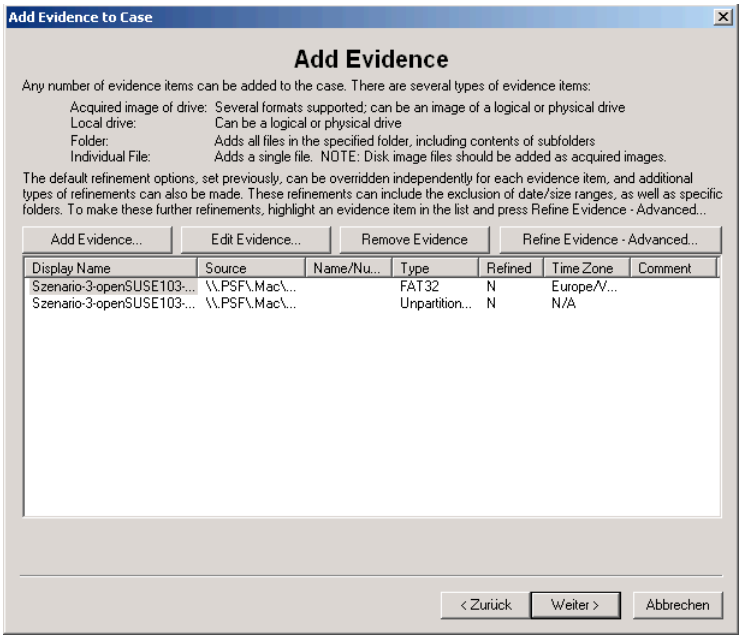
Für die weiteren nicht Szenario spezifischen Schritte wird hier auf die detaillierten Angaben im Kapitel 7.2.2 (Szenario 2: FAT32 in Windows ME) verwiesen. Nach diesen Schritten

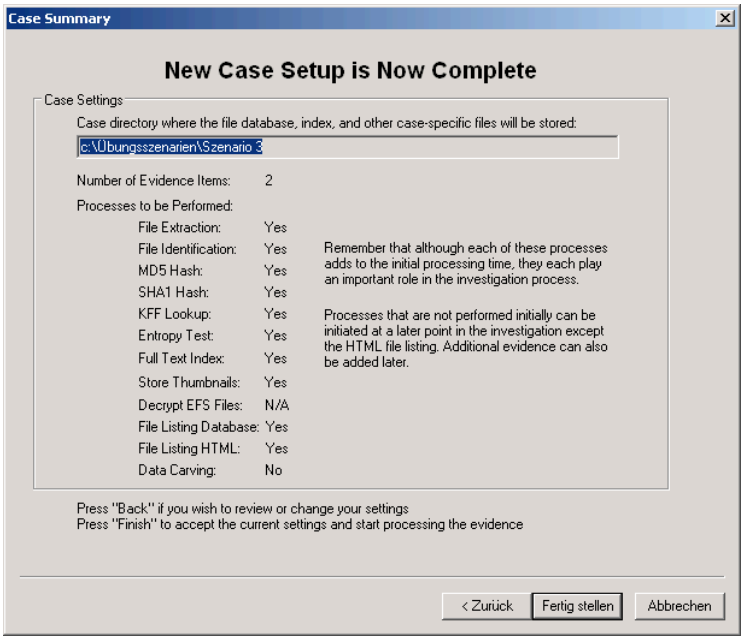
muss nur noch das Image durch Klicken auf den Button `Add Evidence...` als Beweis hinzugefügt werden.











Nachdem alle Informationen angegeben wurden, kann über das Klicken auf den Button `Fertig stellen` mit der Analyse des Images begonnen werden.

- **Ergebnisse der forensischen Untersuchung**

Liste der gefundenen Dateien und Ordner

Im Ordner `\Dokumente\KV Angewandte Systemtheorie - Kryptographie\` gefundene Dateien:

Dateiname	Größe in Byte	MD5 Prüfsumme	Fehlerfrei
01 - Intro.pdf	555.147	DBB8A499D8A6FB200F65F3D6A2B1DC11	ja
02 - DES.pdf	333.246	713A3D7C5179860C632818BA4368606F	ja
03 - FromDES_To AES.pdf	670.463	A442C23B2F7E8E68D7C973DDD2BD5935	ja
04 - Streamci-phers.pdf	370.164	9672DC6BE1B59C58CDE1D769577AD885	ja
05 - AsymetricCryp-to.pdf	327.872	1A6A5AC65DDCCE99868E22B51F8E3339	ja
06 - Signatures .pdf	511.361	3F2C6F4B2F9AFDC95F4CFECFB246FED5	ja
07 - ECC.pdf	153.306	C5508A72297CB052F120E3B1D7068C79	ja
08 - PGP.pdf	668.244	63E1AA9B564EAD28892261282EA42AC6	ja
09 - ssl.pdf	406.166	3435C6244E44C36C35CB3485501C4B5C	ja
10 - SecInternet-Payment.pdf	1.412.756	4CE931E1D272F5676327A66A86D91026	ja
HalloLeute.pdf	191.566	4AC54DAD2C20C9E48DFC7230B8A9705F	ja

Mitschrift 29.11.06.doc	48.128	06547E9EAA747812A5CD9B0C41910FA5	ja

Vom gelöschten Ordner `Dokumente\Seminar - Netzwerke - Security\` konnten nur folgende Unterordner- und Dateinamen, jedoch kein Dateiinhalt, wiederhergestellt werden.

Ordnername
ADSAdClient31_data
cg0705.mspx_files
Cisco
ratings_data

Im Ordner `\Dokumente\Seminar - Netzwerke - Security\` gefundene Dateien:

Dateiname	Größe in Byte	MD5 Prüfsumme	Fehlerfrei
4.NAM.pdf	0	keine Daten für die Berechnung	nein
arrow_px_down.gif	0	keine Daten für die Berechnung	nein
arrow_px_up.gif	0	keine Daten für die Berechnung	nein
cdccont_0900aecd80217e26.pdf	0	keine Daten für die Berechnung	nein
cdccont_0900aecd80234ef4.pdf	0	keine Daten für die Berechnung	nein
gradient.jpg	0	keine Daten für die Berechnung	nein
NAP-Pic1.jpg	0	keine Daten für die Berechnung	nein
NAPArch.doc	0	keine Daten für die Berechnung	nein
Screen1-Detail.bmp	0	keine Daten für die Berechnung	nein
Securtiy_CNAC_032004.pdf	0	keine Daten für die Berechnung	nein

- **Zusammenfassung**

Das Ergebnis dieses Szenarios zeigt, dass nicht alle gelöschten Dateien gefunden wurden. Von den gefundenen konnten auch nur die Dateien aus dem Ordner `\Dokumente\KV Angewandte Systemtheorie - Kryptographie\` korrekt wiederhergestellt werden. Bei allen anderen Dateien wurde lediglich der Dateiname gefunden. Wobei hier auch nicht alle Dateien den richtigen Ordnern zugeordnet werden konnten. Dabei handelt es sich aber offensichtlich um ein Problem des `Forensic Toolkits`, da dieses Problem mit dem `Authopsy Forensic Browser` nicht auftritt.

Alle anderen Daten wurden vom Betriebssystem so weit überschrieben, dass sie auch nicht bei der manuellen Suche mit einem Hexeditor oder unter Anwendung der `Data Carving` Funktion gefunden werden können.

7.2.4 Szenario 4: Allocation-Algorithmus

- **Übungsangabe**

Für dieses Szenario erhalten Sie zwei raw / dd Images von Festplatten, welche jeweils eine mit FAT32 formatierte Partition enthalten. Je eine dieser Festplatten wurde unter Windows XP und unter openSUSE 10.3 verwendet. Wobei diese Festplatten vor der Formatierung fabriksneu, sprich ohne jegliche Daten waren.

Die Motivation für diese Aufgabe ist die Untersuchung des Zuteilungs- / Allocation-Algorithmus. Aus diesem Grund wurden nach dem Löschen einiger Dateien weitere Dateien auf die Festplatte kopiert und von diesen wieder einige gelöscht.

Ihre Aufgabe besteht nun darin, mit Hilfe eines Computer-Forensik-Programms die Images nach diesen gelöschten Dateien zu durchsuchen. Die gefundenen Dateien sollen anschließend wenn möglich extrahiert und gespeichert werden. Wie bei einer richtigen forensischen Untersuchung sollen Sie einen Bericht über Ihre Tätigkeiten erstellen. Führen Sie in diesem Bericht ebenfalls Ihre Untersuchungsergebnisse über den Zuteilungs- / Allocation-Algorithmus an.

- **Image Details**

Name	`Szenario-4-openSUSE103-FAT32.dd`
Größe	`367.460.352 Byte`
MD5 Prüfsumme	`2FDFCD370AA69C0638DBF1EDE1281E9F`

Name	`Szenario-4-winxp-FAT32.dd`
Größe	`367.460.352 Byte`
MD5 Prüfsumme	`8617488CF5B7CD12195CB3BFFB899F88`

- **Forensisch interessante Dateien und Ordner**

In diesem Abschnitt werden alle gelöschten Dateien und Ordner angegeben, welche im Zuge der forensischen Untersuchung gefunden werden sollen.

Datei-Log:

1. Ordner `Bilder` und `Dokumente` auf das Laufwerk kopieren
2. Dateien aus dem Ordner `\Dokumente\KV Angewandte Systemtheorie – Kryptographie\` wie angegeben löschen

3. Ordner `neue Bilder` kopieren
4. Ordner `neue Dokumente` anlegen und Ordner `KV Requirements Engineering`, `Systemtheorie 1` und `VL Pervasive Computing Infrastruktur` kopieren
5. Inhalt und Ordner `\neue Dokumente\Systemtheorie 1\FMSCalc` löschen
6. Dateien aus dem Ordner `\neue Dokumente\VL Pervasive Computing Infrastruktur` wie angegeben löschen
7. Ordner `KV Sicherheitsaspekte in der Informatik` und `Netzwerkadministration` kopieren
8. Inhalt und Ordner `\neue Dokumente\ KV Sicherheitsaspekte in der Informatik\12_IDS (Kofler)` löschen

Angegebener Inhalt des Ordners gelöscht		
`\Dokumente\KV Angewandte Systemtheorie – Kryptographie\`		
Dateiname	Größe in Byte	MD5 Prüfsumme
`01 - Intro.pdf`	555.147	`DBB8A499D8A6FB200F65F3D6A2B1DC11`
`02 - DES.pdf`	333.246	`713A3D7C5179860C632818BA4368606F`
`03 - FromDES_ToAES.pdf`	670.463	`A442C23B2F7E8E68D7C973DDD2BD5935`
`04 - Streamciphers.pdf`	370.164	`9672DC6BE1B59C58CDE1D769577AD885`
`05 - AsymetricCrypto .pdf`	327.872	`1A6A5AC65DDCCE99868E22B51F8E3339`
`06 - Signatures.pdf`	511.361	`3F2C6F4B2F9AFDC95F4CFECFB246FED5`
`07 - ECC.pdf`	153.306	`C5508A72297CB052F120E3B1D7068C79`
`08 - PGP.pdf`	668.244	`63E1AA9B564EAD28892261282EA42AC6`
`09 - ssl.pdf`	406.166	`3435C6244E44C36C35CB3485501C4B5C`
`10 - SecInternetPay- ment.pdf`	1.413.042	`4CE931E1D272F5676327A66A86D91026`
`HalloLeute.pdf`	191.566	`4AC54DAD2C20C9E48DFC7230B8A9705F`
`Mitschrift 29.11.06.doc`	48.128	`06547E9EAA747812A5CD9B0C41910FA5`

Ordner + Inhalt gelöscht		
`\neue Dokumente\Systemtheorie 1\FMSCalc`		
Dateiname	Größe in Byte	MD5 Prüfsumme
`FSMCalc.jar`	167.638	`E523A6B4607B61F753AC54AAAF88AF7A`
`FSMCalc.pdf`	586.933	`70CE8D6A978FB78C128C1530591C644E`
`FSMCalc.zip`	1.208.407	`687BE011172B57A6927DA1B974F4F930`
`jgraph.jar`	157.121	`E238E43F18BCBD0AFB020618BD610D18`

Angegebene Dateien im Ordner gelöscht		
\neue Dokumente\VL Pervasive Computing Infrastruktur		
Dateiname	Größe in Byte	MD5 Prüfsumme
01 Pervasive Vision.pdf	5.397.571	12A534CECE924E784A277F9A4D64E8C4
04 reading 1 sarma+weis+ engels - rfid systems and security and privacy impli- cations.pdf	401	8386A4247CD7EA6963D47E45D38B1619
04 reading 2 roemer+schoch +mattern+duebendorfer - smart identification frame- works for ubiquitous comput- ing applications.pdf	401	8386A4247CD7EA6963D47E45D38B1619
04 reading 3 orr+abowd - the smart floor - a mechanism for natural user identifica- tion and tracking.pdf	401	8386A4247CD7EA6963D47E45D38B1619
05 reading 1 akyildiz2001_ wireless sensor networks_a survey.pdf	401	8386A4247CD7EA6963D47E45D38B1619
05 reading 2 cerpa2001_ habitat monitoring.pdf	401	8386A4247CD7EA6963D47E45D38B1619
05 reading 3 ewps2006_plat- form survey.pdf	401	8386A4247CD7EA6963D47E45D38B1619
05 reading 4 haensel2006_ sensor networks.pdf	401	8386A4247CD7EA6963D47E45D38B1619
05 reading 5 inta2000_ directed diffusion.pdf	401	8386A4247CD7EA6963D47E45D38B1619
05 reading 6 krish2002_crit- ical density.pdf	401	8386A4247CD7EA6963D47E45D38B1619
06 a-survey-of-context.pdf	401	8386A4247CD7EA6963D47E45D38B1619
06 reading1 dey+abowd a con- ceptual framework.pdf	401	8386A4247CD7EA6963D47E45D38B1619
06 reading2 dey understand- ing and using context.pdf	401	8386A4247CD7EA6963D47E45D38B1619
06 reading3 chen+kotz a sur- vey of context aware.pdf	401	8386A4247CD7EA6963D47E45D38B1619

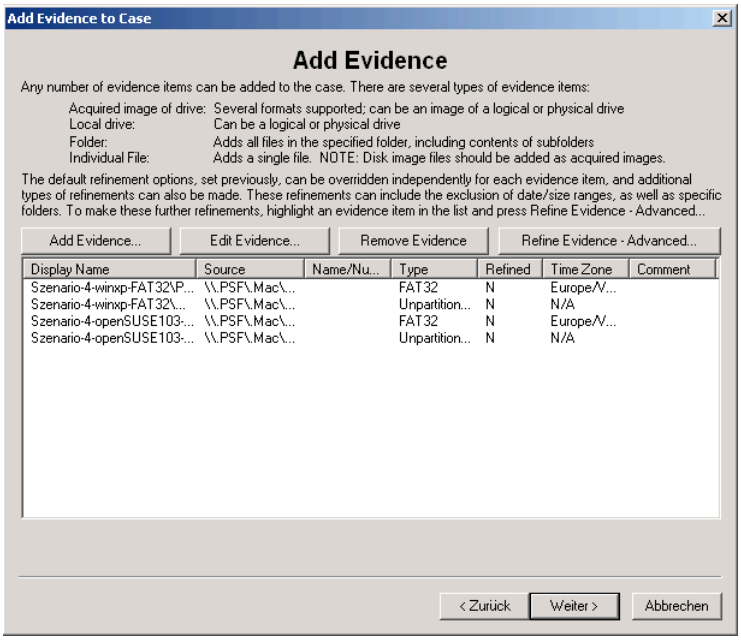
07 reading 1 Molyneaux+ Kortuem - Ubiquitous Displays in dynamic environments - Issues and Opportunities.pdf	401	8386A4247CD7EA6963D47E45D38B1619
07 reading 2 plaue+miller+ stasko - is a picture worth a thousand words - evaluation of information awareness displays.pdf	401	8386A4247CD7EA6963D47E45D38B1619
07 reading 3 ferscha+vogl - the webwall.pdf	401	8386A4247CD7EA6963D47E45D38B1619
09 PrehensileMovementsOfTheHumanHand Napier 1956.pdf	401	8386A4247CD7EA6963D47E45D38B1619
09 reading1 fishkin taxonomy.pdf	401	8386A4247CD7EA6963D47E45D38B1619
09 reading2 sharlin et al TUIs humans spatiality.pdf	401	8386A4247CD7EA6963D47E45D38B1619
09 reading3 shaer et al TAC paradigm.pdf	401	8386A4247CD7EA6963D47E45D38B1619

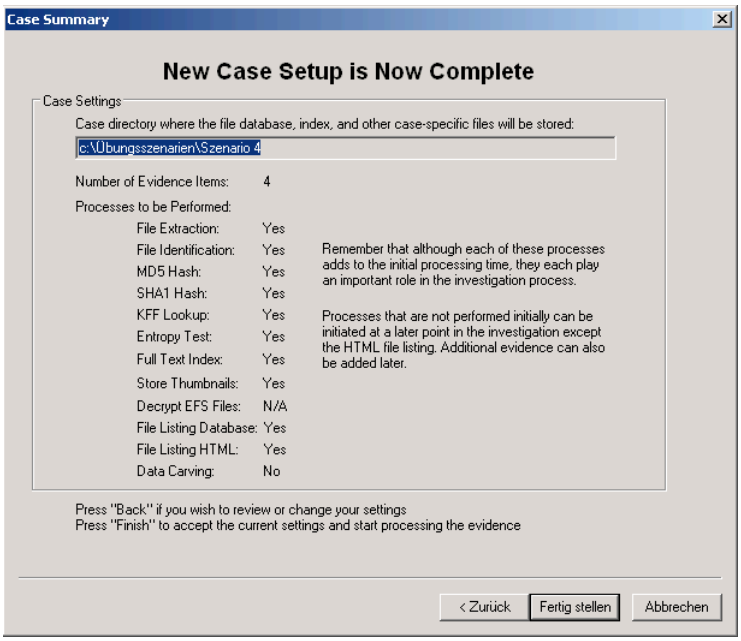
Ordner + Inhalt gelöscht		
\neue Dokumente\ KV Sicherheitsaspekte in der Informatik\12_IDS (Kofler)		
Dateiname	Größe in Bytes	MD5 Prüfsumme
12b_IDS_Executive-View.pdf	203.064	0AFA7EB6A74C220BD389BA486F5BFA03
12b_IDS_Executive-View_notes.pdf	193.585	293E7AA0B4E79236B344CDF317FE84B1
12c_IDS_Admin-View.pdf	2.888.255	3A0F50C6F3C68255680D1BE3C8EA9502
12c_IDS_Admin-View_notes.pdf	2.174.543	C90BCCC9648CA20760AF629911C2B3F3
12d_IDS_Admin-View_FAQs.pdf	154.155	8428BAAC0BE6B5DD7770ACDD1F978D23
12e_IDS_Expert-View.pdf	502.665	B0B9F452DAF4B36645A01513BBA088DC
12e_IDS_Expert-View_notes.pdf	423.861	E0BED097341932C34AE746B44A1B5632
12g_IDS_Literatur.pdf	397.393	EC17F82E8D2414CABB30D11E78593527

- **Forensische Untersuchung**

Da die Untersuchung des Images für dieses Szenario geleich wie bei den beiden vorherigen Szenarios erfolgt, wird für die Details dieser auf das Kapitel 7.2.2 (Szenario 2: FAT32 in Windows ME) verwiesen.

Zusammenfassend hier noch die Auflistung der hinzugefügten Beweise.





Nachdem alle Informationen angegeben wurden, kann über das Klicken auf den Button `Fertig stellen` mit der Analyse des Images begonnen werden.

- **Ergebnisse der forensischen Untersuchung**

Liste der gefundenen Dateien und Ordner in `Szenario-4-openSUSE103-FAT32.dd`

Im Ordner `\Dokumente\KV Angewandte Systemtheorie - Kryptographie\` wurden folgende Dateien gefunden:

Dateiname	Größe in Byte	MD5 Prüfsumme	Fehlerfrei
01 - Intro.pdf	555.147	DBB8A499D8A6FB200F65F3D6A2B1DC11	ja
02 - DES.pdf	333.246	713A3D7C5179860C632818BA4368606F	ja
03 - FromDES_To AES.pdf	670.463	A442C23B2F7E8E68D7C973DDD2BD5935	ja
04 - Stream-ciphers.pdf	370.164	9672DC6BE1B59C58CDE1D769577AD885	ja
05 – Asymetric Crypto.pdf	327.872	1A6A5AC65DDCCE99868E22B51F8E3339	ja
06 – Signatures .pdf	511.361	3F2C6F4B2F9AFDC95F4CFECFB246FED5	ja
07 - ECC.pdf	153.306	C5508A72297CB052F120E3B1D7068C79	ja
08 - PGP.pdf	668.244	63E1AA9B564EAD28892261282EA42AC6	ja
09 - ssl.pdf	406.166	3435C6244E44C36C35CB3485501C4B5C	ja
10 - SecInternet-Payment.pdf	1.412.756	4CE931E1D272F5676327A66A86D91026	ja
HalloLeute.pdf	191.566	4AC54DAD2C20C9E48DFC7230B8A9705F	ja
Mitschrift 29.11.06.doc	48.128	06547E9EAA747812A5CD9B0C41910FA5	ja

Im Ordner `\neue Dokumente\ KV Sicherheitsaspekte in der Informatik\`wurde der gelöschte Ordner `12_IDS (Kofler)` mit folgenden Dateien gefunden:

Dateiname	Größe in Byte	MD5 Prüfsumme	Fehlerfrei
12b_IDS_Executive-View.pdf	203.064	0AFA7EB6A74C220BD389BA486F5BFA03	ja
12b_IDS_Executive-View_notes.pdf	193.585	293E7AA0B4E79236B344CDF317FE84B1	ja
12c_IDS_Admin-View .pdf	2.888.255	3A0F50C6F3C68255680D1BE3C8EA9502	ja
12c_IDS_Admin-View_ notes.pdf	2.174.543	C90BCCC9648CA20760AF629911C2B3F3	ja
12d_IDS_Admin-View_ FAQs.pdf	154.155	8428BAAC0BE6B5DD7770ACDD1F978D23	ja
12e_IDS_Expert-View .pdf	502.665	B0B9F452DAF4B36645A01513BBA088DC	ja
12e_IDS_Expert-View_ notes.pdf	423.861	E0BED097341932C34AE746B44A1B5632	ja
12g_IDS_Literatur.pdf	397.393	EC17F82E8D2414CABB30D11E78593527	ja

Im Ordner `\neue Dokumente\Systemtheorie 1\`wurde der gelöschte Ordner `FMSCalc` mit folgenden Dateien gefunden:

Dateiname	Größe in Byte	MD5 Prüfsumme	Fehlerfrei
FSMCalc.jar	167.638	E523A6B4607B61F753AC54AAAF88AF7A	ja
FSMCalc.pdf	586.933	70CE8D6A978FB78C128C1530591C644E	ja
FSMCalc.zip	1.208.407	687BE011172B57A6927DA1B974F4F930	ja
jgraph.jar	157.121	E238E43F18BCBD0AFB020618BD610D18	ja

Im Ordner `\neue Dokumente\VL Pervasive Computing Infrastruktur \`wurden folgenden Dateien gefunden:

Dateiname	Größe in Byte	MD5 Prüfsumme	Fehlerfrei
01 Pervasive Vision.pdf	5.397.571	12A534CECE924E784A277F9A4D64E8C4	ja
04 reading 1 sarma+weis+ engels - rfid systems and security and privacy implications.pdf	401	8386A4247CD7EA6963D47E45D38B1619	ja
04 reading 2 roemer+ schoch+mattern+dueben- dorfer - smart identifi- cation frameworks for ubiquitous computing ap- plications.pdf	401	8386A4247CD7EA6963D47E45D38B1619	ja
04 reading 3 orr+abowd - the smart floor - a me- chanism for natural user identification and tra- cking.pdf	401	8386A4247CD7EA6963D47E45D38B1619	ja
05 reading 1 akyildiz2001 _wireless sensor networks _a survey.pdf	401	8386A4247CD7EA6963D47E45D38B1619	ja
05 reading 2 cerpa2001_ habitat monitoring.pdf	401	8386A4247CD7EA6963D47E45D38B1619	ja
05 reading 3 ewps2006_ platform survey.pdf	401	8386A4247CD7EA6963D47E45D38B1619	ja
05 reading 4 haensel2006 _sensor networks.pdf	401	8386A4247CD7EA6963D47E45D38B1619	ja
05 reading 5 inta2000_ directed diffusion.pdf	401	8386A4247CD7EA6963D47E45D38B1619	ja
05 reading 6 krish2002_ critical density.pdf	401	8386A4247CD7EA6963D47E45D38B1619	ja
06 a-survey-of-context .pdf	401	8386A4247CD7EA6963D47E45D38B1619	ja

06 reading1 dey+abowd a conceptual framework.pdf	401	8386A4247CD7EA6963D47E45D38B1619	ja
06 reading2 dey under-standing and using con-text.pdf	401	8386A4247CD7EA6963D47E45D38B1619	ja
06 reading3 chen+kotz a survey of context awa-re.pdf	401	8386A4247CD7EA6963D47E45D38B1619	ja
07 reading 1 Molyneaux+ Kortuem - Ubiquitous Dis-plays in dynamic environ-ments - Issues and Oppor-tunities.pdf	401	8386A4247CD7EA6963D47E45D38B1619	ja
07 reading 2 plaue+miller +stasko - is a picture worth a thousand words - evaluation of information awareness displays.pdf	401	8386A4247CD7EA6963D47E45D38B1619	ja
07 reading 3 ferscha+vogl - the webwall.pdf	401	8386A4247CD7EA6963D47E45D38B1619	ja
09 PrehensileMovementsOf TheHumanHand Napier 1956.pdf	401	8386A4247CD7EA6963D47E45D38B1619	ja
09 reading1 fishkin tax-onomy.pdf	401	8386A4247CD7EA6963D47E45D38B1619	ja
09 reading2 sharlin et al TUIs humans spatiality .pdf	401	8386A4247CD7EA6963D47E45D38B1619	ja
09 reading3 shaer et al TAC paradigm.pdf	401	8386A4247CD7EA6963D47E45D38B1619	ja

Liste der gefundenen Dateien und Ordner in `Szenario-4-winxp-FAT32.dd`

Im Ordner `\Dokumente\KV Angewandte Systemtheorie - Kryptographie\` wurden folgende Dateien gefunden:

Dateiname	Größe in Byte	MD5 Prüfsumme	Fehlerfrei
01 - Intro.pdf	0	keine Daten für die Berechnung	nein
02 - DES.pdf	0	keine Daten für die Berechnung	nein
03 - FromDES_To AES.pdf	0	keine Daten für die Berechnung	nein

04 - Streamci-phers.pdf	0	keine Daten für die Berechnung	nein
05 - Asymetric-Crypto.pdf	0	keine Daten für die Berechnung	nein
06 - Signatures .pdf	0	keine Daten für die Berechnung	nein
07 - ECC.pdf	0	keine Daten für die Berechnung	nein
08 - PGP.pdf	0	keine Daten für die Berechnung	nein
09 - ssl.pdf	0	keine Daten für die Berechnung	nein
10 - SecInternet-Payment.pdf	0	keine Daten für die Berechnung	nein
HalloLeute.pdf	0	keine Daten für die Berechnung	nein
Mitschrift 29.11.06.doc	0	keine Daten für die Berechnung	nein

Im Ordner `\neue Dokumente\ KV Sicherheitsaspekte in der Informatik\` wurde der gelöschte Ordner `12_IDS (Kofler)` mit folgenden Dateien gefunden:

Dateiname	Größe in Byte	MD5 Prüfsumme	Fehlerfrei
12b_IDS_Executive-View.pdf	203.064	0AFA7EB6A74C220BD389BA486F5BFA03	ja
12b_IDS_Executive-View_notes.pdf	193.585	293E7AA0B4E79236B344CDF317FE84B1	ja
12c_IDS_Admin-View .pdf	2.888.255	3A0F50C6F3C68255680D1BE3C8EA9502	ja
12c_IDS_Admin-View_ notes.pdf	2.174.543	C90BCCC9648CA20760AF629911C2B3F3	ja
12d_IDS_Admin-View_ FAQs.pdf	154.155	8428BAAC0BE6B5DD7770ACDD1F978D23	ja
12e_IDS_Expert-View .pdf	502.665	B0B9F452DAF4B36645A01513BBA088DC	ja
12e_IDS_Expert-View_ notes.pdf	423.861	E0BED097341932C34AE746B44A1B5632	ja
12g_IDS_Literatur.pdf	397.393	EC17F82E8D2414CABB30D11E78593527	ja

Im Ordner `\neue Dokumente\Systemtheorie 1\`wurde der gelöschte Ordner `FMSCalc` ohne Inhalt gefunden.

Im Ordner `\neue Dokumente\VL Pervasive Computing Infrastruktur \`wurden folgenden Dateien gefunden:

Dateiname	Größe in Byte	MD5 Prüfsumme	Fehler-frei
01 Pervasive Vision.pdf	0	keine Daten für die Berechnung	nein
04 reading 1 sarma+weis+ engels - rfid systems and security and privacy impli- cations.pdf	401	8386A4247CD7EA6963D47E45D38B1619	ja
04 reading 2 roemer+schoch+ mattern+duebendorfer - smart identification frameworks for ubiquitous computing ap- plications.pdf	401	8386A4247CD7EA6963D47E45D38B1619	ja
04 reading 3 orr+abowd - the smart floor - a mechanism for natural user identifica- tion and tracking.pdf	401	8386A4247CD7EA6963D47E45D38B1619	ja
05 reading 1 akyildiz2001_ wireless sensor networks_a survey.pdf	401	8386A4247CD7EA6963D47E45D38B1619	ja
05 reading 2 cerpa2001_ habitat monitoring.pdf	401	8386A4247CD7EA6963D47E45D38B1619	ja
05 reading 3 ewps2006_ platform survey.pdf	401	8386A4247CD7EA6963D47E45D38B1619	ja
05 reading 4 haensel2006_ sensor networks.pdf	401	8386A4247CD7EA6963D47E45D38B1619	ja
05 reading 5 inta2000_di- rected diffusion.pdf	401	8386A4247CD7EA6963D47E45D38B1619	ja
05 reading 6 krish2002_crit- ical density.pdf	401	8386A4247CD7EA6963D47E45D38B1619	ja
06 a-survey-of-context.pdf	401	8386A4247CD7EA6963D47E45D38B1619	ja
06 reading1 dey+abowd a con- ceptual framework.pdf	401	8386A4247CD7EA6963D47E45D38B1619	ja
06 reading2 dey understand- ing and using context.pdf	401	8386A4247CD7EA6963D47E45D38B1619	ja
06 reading3 chen+kotz a sur- vey of context aware.pdf	401	8386A4247CD7EA6963D47E45D38B1619	ja
07 reading 1 Molyneaux+ Kortuem - Ubiquitous Dis- plays in dynamic environ- ments - Issues and Opportu- nities.pdf	401	8386A4247CD7EA6963D47E45D38B1619	ja

07 reading 2 plaue+miller+ stasko - is a picture worth a thousand words - evaluation of information awareness displays.pdf	401	8386A4247CD7EA6963D47E45D38B1619	ja
07 reading 3 ferscha+vogl - the webwall.pdf	401	8386A4247CD7EA6963D47E45D38B1619	ja
09 PrehensileMovementsOfThe-HumanHand Napier 1956.pdf	401	8386A4247CD7EA6963D47E45D38B1619	ja
09 reading1 fishkin taxonomy .pdf	401	8386A4247CD7EA6963D47E45D38B1619	ja
09 reading2 sharlin et al TUIs humans spatiality.pdf	401	8386A4247CD7EA6963D47E45D38B1619	ja
09 reading3 shaer et al TAC paradigm.pdf	401	8386A4247CD7EA6963D47E45D38B1619	ja

Im Root-Directory wurde noch ein gelöschter Ordner mit dem Namen `Neuer Ordner` gefunden. Dieser Ordner wurde vom Windows-Explorer während dem Anlegen eines neuen Ordners angelegt und nach dessen Umbenennung gelöscht.

- **Zusammenfassung**

Dieses Szenario zeigt, dass die FAT-Implementierungen bzw. die verwendeten Zuteilungsalgorithmen bei diesen beiden Systemen unterschiedlich sind. So konnten etwa beim Linux Image alle Dateien wiederhergestellt werden. Im Gegensatz dazu konnten beim Windows XP Image nur die zuletzt gelöschten und einige ganz kleine Dateien wiederhergestellt werden.
Für das Linux System lässt sich sehr leicht erkennen, dass hier ein *next available* Algorithmus verwendet wurde.
Um für Windows XP eindeutig feststellen zu können welcher Algorithmus verwendet wird, muss das `Szenario-4-winxp-FAT32.dd` Image noch einmal wie in Kapitel 7.2.2 (Szenario 2: FAT32 in Windows ME), jedoch ohne Einschränkung auf nur gelöschte Dateien, untersucht werden. Vergleicht man hier nun die Startcluster der Dateien in der Reihenfolge wie sie laut der Angabe im Datei-Log erzeugt und gelöscht wurden, so erkennt man speziell bei den kleinen gelöschten Dateien, dass von Windows XP ein *best fit* Algorithmus verwendet wurde.

7.2.5 Szenario 5: NTFS mit ADS unter Windows XP

- **Übungsangabe**

Für dieses Übungsszenario erhalten Sie wieder ein Image im raw / dd Format. Die in diesem Image enthaltene Festplatte im MBR-Layout besteht aus einer mit NTFS formatierten Partition, die in einem Rechner mit Windows XP als Datendisk verwendet wurde.
Da es im NTFS-Dateisystem möglich ist, zu Dateien oder Ordner alternative Datenströme (*ADS*) hinzuzufügen, soll in diesem Szenario speziell nach gelöschten Dateien und Ordnern gesucht werden, welche solche alternative Datenströme beinhaltet haben.
Nachdem Sie diese Daten gefunden haben, sollen Sie versuchen, diese so zu extrahieren, sodass sie mit einem für diesen Datentypen geeigneten Programm geöffnet werden können.
Verwenden Sie für die Suche nach diesen Daten ebenfalls ein Computer-Forensik-Programm und erstellen Sie einen Bericht über Ihre Tätigkeiten.

- **Image Details**

Name	`Szenario-5-winxp-NTFS.dd`
Größe	`367.460.352 Byte`
MD5 Prüfsumme	`2EF2CD9BAF942D7C15BC52AC3DB555C1`

- **Forensisch interessante Dateien und Ordner**

In diesem Abschnitt werden alle gelöschten Dateien und Ordner angegeben, welche im Zuge der forensischen Untersuchung gefunden werden sollen. Die Bezeichnung ADS beutet hier *Alternate Data Stream*. Ein ADS ist ein Feature von NTFS, welches erlaubt zu einer Datei oder einem Ordner weitere zusätzliche Daten zu speichern. Details darüber in Kapitel 4.2 NTFS.

Ordner, in welchem die Dateien enthalten sind			
`\Dokumente\Seminar - Netzwerke - Security`			
Dateiname	Größe in Byte	MD5 Prüfsumme	ADS von Datei
`Screen1-Detail.bmp`	`791.898`	`2B9C6C5B91EC8C42AC8BE2F4C4BD59F1`	`Screen1.bmp`
`Screen2-Detail.bmp`	`605.238`	`DC6649B66B450D24D7E4BC53642FF98D`	`Screen2.bmp`

Ordner, in welchem die Dateien enthalten sind			
\Dokumente\SDK-Docs-Tutorials			
Dateiname	Größe in Byte	MD5 Prüfsumme	ADS von Datei
jwstutorial20.zip	2.427.310	2A9776EAF04FD078A12EF3B6134ED721	jwsdp-2_0-ant-docs.zip

Ordner, in welchem die Dateien enthalten sind			
\Bilder			
Dateiname	Größe in Byte	MD5 Prüfsumme	ADS von Datei
Image002.jpg	352.784	F0C0E3BEDB439F2F40795863B19DDA4C	Beispielbil-der.lnk

Ordner, in welchem die Dateien enthalten sind			
\Dokumente\Seminar - Netzwerke - Security			
Dateiname	Größe in Byte	MD5 Prüfsumme	ADS von Ordner
2.Networks.pdf	376.297	EFB288CF96E3F492C77187DEFE526509	Cisco

Angegebene Dateien im Ordner gelöscht			
\Dokumente\KV Angewandte Systemtheorie - Kryptographie			
Dateiname	Größe in Bytes	MD5 Prüfsumme	Bemerkung
06 - Signatures.pdf	511.361	3F2C6F4B2F9AFDC95F4CFECFB246FED5	
Mitschrift 29.11.06.doc	48.128	06547E9EAA747812A5CD9B0C41910FA5	ADS der Datei 06 - Signatures.pdf

Angegebene Datei im Ordner gelöscht			
\Bilder			
Dateiname	Größe in Bytes	MD5 Prüfsumme	Bemerkung
Image040.jpg	562.769	0E7AC1684009838B5085A71C2B5EA49D	
Image039.jpg	492.559	C37B868A47B7EDED2BDEC6365525C295	ADS der Datei Image040.jpg

Weiters wurden folgende Dateien gelöscht.

Dateien im Ordner gelöscht		
\Dokumente\KV Angewandte Systemtheorie - Kryptographie		
Dateiname	Größe in Bytes	MD5 Prüfsumme
05 - AsymetricCrypto .pdf	327.872	1A6A5AC65DDCCE99868E22B51F8E3339
07 - ECC.pdf	153.306	C5508A72297CB052F120E3B1D7068C79
08 - PGP.pdf	668.244	63E1AA9B564EAD28892261282EA42AC6
09 - ssl.pdf	406.166	3435C6244E44C36C35CB3485501C4B5C
10 - SecInternetPay-ment.pdf	1.413.042	4CE931E1D272F5676327A66A86D91026
HalloLeute.pdf	191.566	4AC54DAD2C20C9E48DFC7230B8A9705F

Dateien im Ordner gelöscht		
\Dokumente\SDK-Docs-Tutorials		
Dateiname	Größe in Byte	MD5 Prüfsumme
j2ee-5_0-beta-doc-tutorial.zip	12.276.520	F193EC2247C1C76AAC2D44748AB60A2F

Dateien im Ordner gelöscht		
\Dokumente\Seminar - Netzwerke – Security\cg0705.mspx_files		
Dateiname	Größe in Bytes	MD5 Prüfsumme
ADSAdClient31.htm	868	FC90E1EA430AD61272382AD33D915ECA
arrow_px_down.gif	53	A50A5A7BE6297D534E0A9559C583C217
arrow_px_up.gif	53	FBAE595C25E4EB510CE17F4EA17C8760
arrowLTR.gif	821	00BA9FF02078EE89D912B07B8E7D3838
broker.js	4.571	3978DB0D26A384B5622DE03A87C87C6F
cableguy.gif	30.792	0E94B6586D8A5D58C143F471C483F376

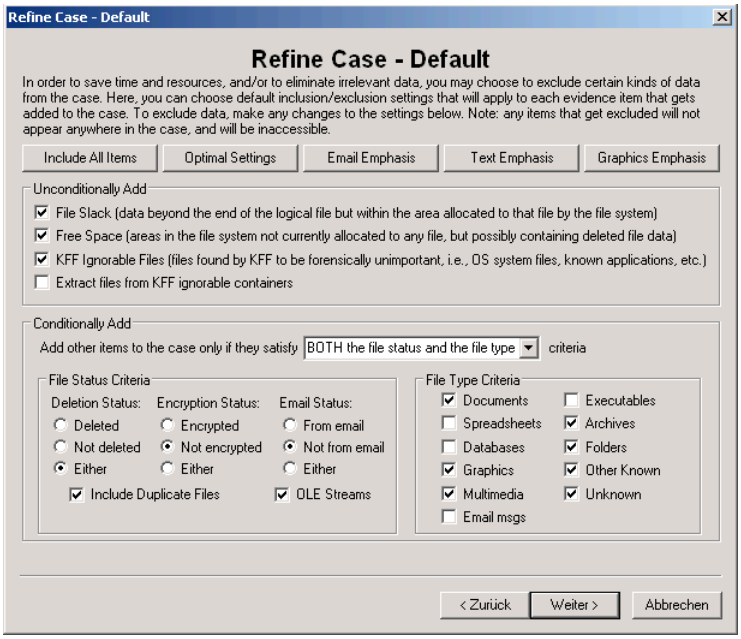
Ordner + Inhalt gelöscht (durch löschen des Ordners)		
\Dokumente\Seminar - Netzwerke – Security\cg0705.mspx_files\ratings_data		
Dateiname	Größe in Byte	MD5 Prüfsumme
ContentRating.css	420	F02DC615EF1E9293C74330D6D59BB14D
rtg_email.gif	1.010	B201A23EC07D55866FAB17A62030774E
rtg_print.gif	574	101A0AC63030B36FCE0421D896940AEF
rtg_save.gif	567	7D20B965EA147E6DC2B580CE6AE71778

Ordner + Inhalt gelöscht		
\neue Bilder		
Dateiname	Größe in Bytes	MD5 Prüfsumme
Image050.jpg	504.864	6C455C48DED3ABA6C275BA38D66F3835
Image051.jpg	460.033	4505C879A31416AF1BDD858F888A28DF
Image052-bearbeitet.jpg	1.224.896	CB6A72603C198555FD9AA52FF9C9A621
Image053.jpg	390.427	B4C8F57484C357B811C6FB56C4B001C6

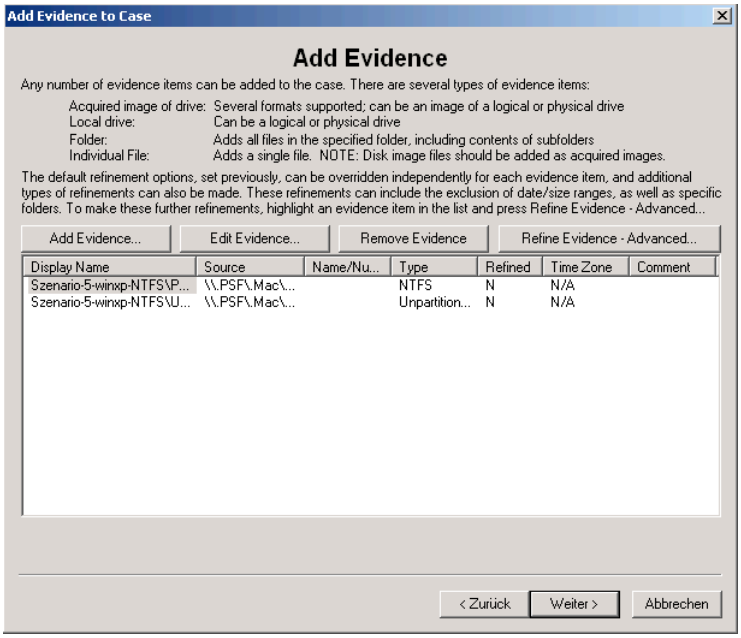
- **Forensische Untersuchung**

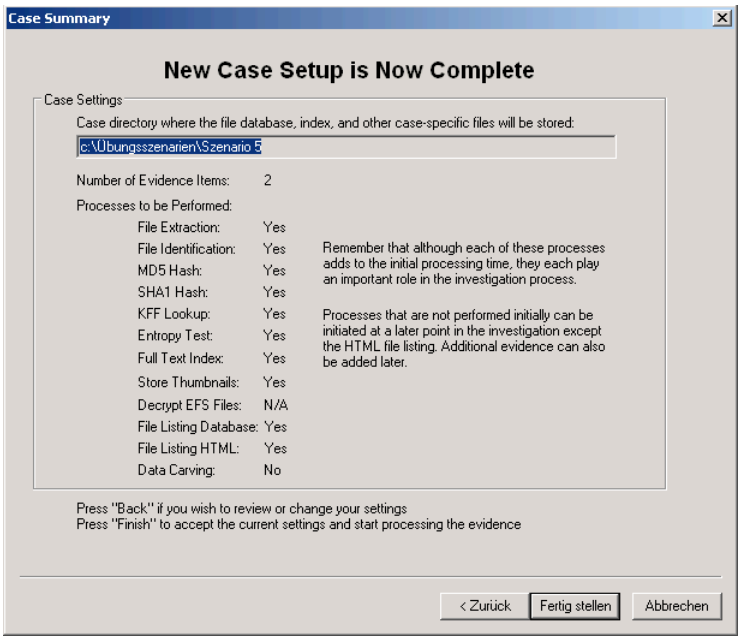
Da die Untersuchung des Images für dieses Szenario gleich wie bei den beiden vorherigen Szenarios erfolgt, wird für die Details dieser auf das Kapitel 7.2.2 (Szenario 2: FAT32 in Windows ME) verwiesen.

In diesem Schritt muss jedoch im Unterschied zu den anderen Szenarien die Suche auf alle Dateien, gelöschte und nicht gelöschte, ausgedehnt werden.



Zusammenfassend hier noch die Auflistung der hinzugefügten Beweise.

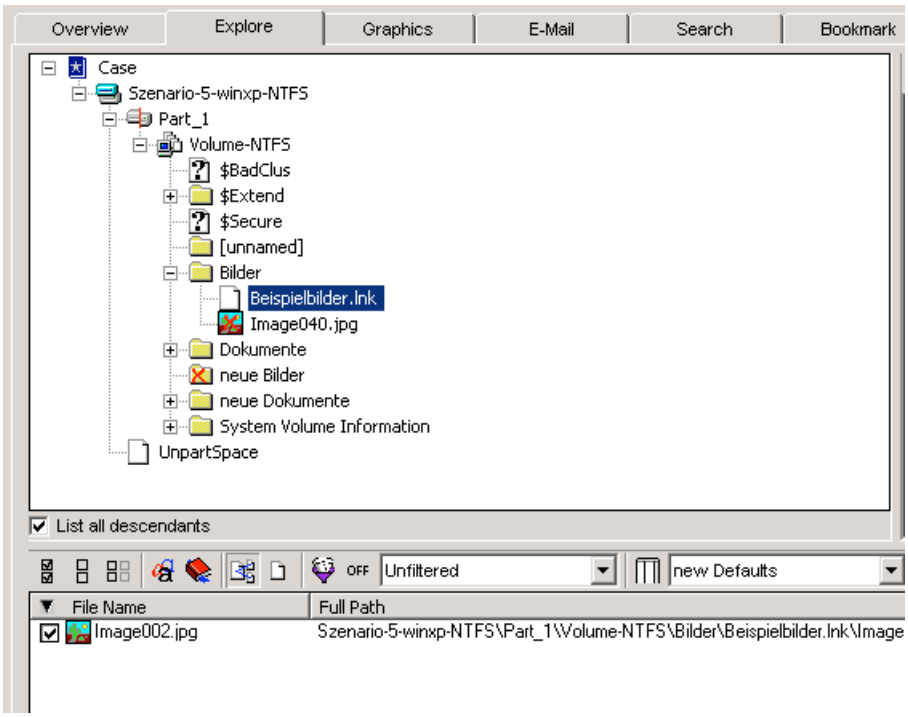




Nachdem alle Informationen angegeben wurden, kann über das Klicken auf den Button `Fertig stellen` mit der Analyse des Images begonnen werden.

- **Ergebnisse der forensischen Untersuchung**

Da das `Forensic Toolkit` keine spezielle Methode für das Filtern bzw. Auflisten von ADS zur Verfügung stellt, müssen diese Daten manuell in der Explorer-Sicht gesucht werden. Der folgende Screenshot zeigt an einer Datei, wie deren ADS angezeigt wird.



Anzeige einer Datei, die als ADS einer anderen Datei gespeichert ist.

In anderen forensischen Analyseprogrammen werden die ADS-Dateien auch mit der folgenden Syntax angezeigt: `Dateiname:ADS-Name`. Diese Art der Anzeige wird zum Beispiel im `Sleuth Kit` verwendet.

Gefundene ADS-Dateien nicht gelöschter Dateien

Die Datei `\Bilder\Beispielbilder.lnk` beinhaltet folgende ADS-Datei

Dateiname	Größe in Byte	MD5 Prüfsumme
`Image002.jpg`	`352.784`	`F0C0E3BEDB439F2F40795863B19DDA4C`

Die Datei `\Dokumente\SDK-Docs-Tutorials\jwsdp-2_0-ant-docs.zip` beinhaltet folgende ADS-Datei

Dateiname	Größe in Byte	MD5 Prüfsumme
`jwstutorial20.zip`	`2.427.310`	`2A9776EAF04FD078A12EF3B6134ED721`

Die Datei `\Dokumente\Seminar - Netzwerke – Security\Screen1.bmp` beinhaltet folgende ADS-Datei

Dateiname	Größe in Byte	MD5 Prüfsumme
`Screen1-Detail.bmp`	`791.898`	`2B9C6C5B91EC8C42AC8BE2F4C4BD59F1`

Die Datei `\Dokumente\Seminar - Netzwerke – Security\Screen2.bmp` beinhaltet folgende ADS-Datei

Dateiname	Größe in Byte	MD5 Prüfsumme
`Screen2-Detail.bmp`	`605.238`	`DC6649B66B450D24D7E4BC53642FF98D`

Anmerkung

Die Datei `2.Networks.pdf` im Ordner `\Dokumente\Seminar - Netzwerke – Security\Cisco\` wird mit dem `Forensic Toolkit` nicht als ADS des Ordners, sondern als normaler Inhalt angezeigt.

Gefundene ADS-Dateien gelöschter Dateien

Die Datei `\Bilder\Image040.jpg` beinhaltet folgende ADS-Datei

Dateiname	Größe in Byte	MD5 Prüfsumme
`Image039.jpg`	`492.559`	`C37B868A47B7EDED2BDEC6365525C295`

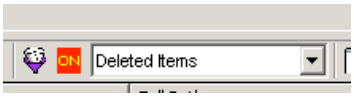
Die Datei `\Dokumente\KV Angewandte Systemtheorie - Kryptographie\06 - Signatures.pdf` beinhaltet folgende ADS-Datei

Dateiname	Größe in Byte	MD5 Prüfsumme
Mitschrift 29.11.06.doc	48.128	06547E9EAA747812A5CD9B0C41910FA5

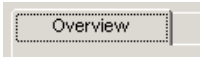
Liste der gefundenen gelöschten Dateien und Ordner

Damit das Auffinden der gelöschten Dateien einfacher wird, kann die Funktionalität des Dateifilters eingesetzt werden. Für das Herausfiltern der gelöschten Dateien kann die vorhandene Einstellung für das Anzeigen gelöschter Dateien verwendet werden.

Den Dateifilter können Sie entweder durch das Klicken auf das rechte Symbol des folgenden Screenshots anpassen, oder Sie können eine vorgegebene Einstellung aus dem Drop-Down Menü neben dem `ON` Symbol auswählen.



Aktiviert wird der Dateifilter über das Klicken auf den Button `Filtered` auf der Übersichtsseite (`Overview`).



Die Aktivierung des Buttons `Actual Files` bewirkt, dass nur die vollständigen Dateien, nicht aber die eingebetteten Dateien, wie der Inhalt von Zip-Dateien, E-Mails oder OLE-Streams angezeigt werden.

Im Ordner `\Bilder\` gefundene Datei:

Dateiname	Größe in Byte	MD5 Prüfsumme	Fehlerfrei
Image040.jpg	562.769	0E7AC1684009838B5085A71C2B5EA49D	ja

Im Ordner `\Dokumente\KV Angewandte Systemtheorie - Kryptographie\` gefundene Dateien:

Dateiname	Größe in Byte	MD5 Prüfsumme	Fehlerfrei
05 - Asymetric Crypto.pdf	327.872	1A6A5AC65DDCCE99868E22B51F8E3339	ja
06 - Signatures .pdf	511.361	3F2C6F4B2F9AFDC95F4CFECFB246FED5	ja

07 - ECC.pdf	153.306	C5508A72297CB052F120E3B1D7068C79	ja
08 - PGP.pdf	668.244	63E1AA9B564EAD28892261282EA42AC6	ja
09 - ssl.pdf	406.166	3435C6244E44C36C35CB3485501C4B5C	ja
10 - SecInternet-Payment.pdf	1.412.756	4CE931E1D272F5676327A66A86D91026	ja
HalloLeute.pdf	191.566	4AC54DAD2C20C9E48DFC7230B8A9705F	ja

Im Ordner `\Dokumente\SDK-Docs-Tutorials\` gefundene Datei

Dateiname	Größe in Byte	MD5 Prüfsumme	Fehlerfrei
j2ee-5_0-beta-doc-tutorial.zip	12.276.520	F193EC2247C1C76AAC2D44748AB60A2F	ja

Im Ordner `\Dokumente\Seminar - Netzwerke – Security\cg0705.mspx_files\` gefundene Dateien:

Dateiname	Größe in Byte	MD5 Prüfsumme	Fehlerfrei
ADSAdClient31.htm	868	FC90E1EA430AD61272382AD33D915ECA	ja
arrow_px_down.gif	53	A50A5A7BE6297D534E0A9559C583C217	ja
arrow_px_up.gif	53	FBAE595C25E4EB510CE17F4EA17C8760	ja
arrowLTR.gif	821	00BA9FF02078EE89D912B07B8E7D3838	ja
broker.js	4.571	3978DB0D26A384B5622DE03A87C87C6F	ja
cableguy.gif	30.792	0E94B6586D8A5D58C143F471C483F376	ja

Weiters wurde in diesem Ordner der gelöschte Ordner `\ratings_data` mit folgenden Dateien gefunden.

Dateiname	Größe in Byte	MD5 Prüfsumme	Fehlerfrei
$I30	4.096	E80C95B223972320B00DEC7DB7A9E8E1	*
ContentRating.css	420	F02DC615EF1E9293C74330D6D59BB14D	ja
rtg_email.gif	1.010	B201A23EC07D55866FAB17A62030774E	ja
rtg_print.gif	574	101A0AC63030B36FCE0421D896940AEF	ja
rtg_save.gif	567	7D20B965EA147E6DC2B580CE6AE71778	ja

Ordner `\neue Bilder` im Stammverzeichnis mit folgendem Inhalt gefunden.

Dateiname	Größe in Byte	MD5 Prüfsumme	Fehlerfrei
$I30	4.096	489731EBB561C8ED8C6951C84401B8D6	*
Image050.jpg	504.864	6C455C48DED3ABA6C275BA38D66F3835	ja
Image051.jpg	460.033	4505C879A31416AF1BDD858F888A28DF	ja
Image052-bear-beitet.jpg	1.224.896	CB6A72603C198555FD9AA52FF9C9A621	ja
Image053.jpg	390.427	B4C8F57484C357B811C6FB56C4B001C6	ja

* Angabe nicht möglich, da die Datei eine Indexdatei des Dateisystems ist und vor dem Löschen nicht sichtbar war. Genau genommen handelt es sich hier um das $INDEX_ALLOCATION Attribut des Ordners. Details darüber im Kapitel 4.2.1.3.

Weitere interessante Untersuchung

Erweiterte Analyse von Bildern: Suche, welche Bilder mit Adobe Photoshop bearbeitet wurden. Extrahieren von eingebeteten Thumbnails der Datei `Image022_bearbeitet.jpg` + Anzeige eingebeteter Metainformationen (erstellt mit Photoshop).

- **Zusammenfassung**

Alle gelöschten Dateien und Ordner dieses Szenarios konnten korrekt gefunden und wiederhergestellt werden.

7.2.6 Szenario 6: Einführung in EnCase Forensics

- **Übungsangabe**

Für dieses Szenario soll die Testversion des Programms EnCase Forensics verwendet werden. Das zu analysierende Image gehört zum Lieferumfang der Testversion und ist kein raw Image, sondern ein Image im EnCase Format (.E01) welches eine Windows XP Installation beinhaltet.

Nach einer kurzen Einarbeitungsphase sollen Sie versuchen, folgende Aufgaben zu lösen:

- Anzeigen gelöschter Dateien durch die Anwendung eines Filters
- Suche nach Bildern in den temporären Dateien durch das Erstellen einer eigenen Query
- Anzeige aller Bilder des Images
- Berechnen der Hashwerte
- Erstellen von „Keywords“ und anschließendes Suchen nach diesen
- Suche nach Internet History Daten
- Suche nach E-Mails

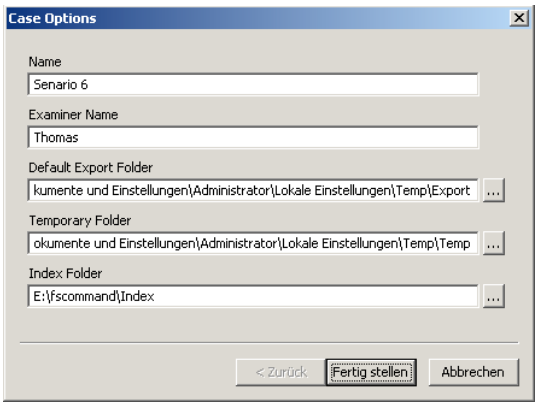
Wie schon bei den vorherigen Szenarien sollen Sie wieder einen Bericht über ihre Tätigkeiten erstellen. Da der generierte Bericht aller Tätigkeiten, mit den extrahierten Dateien, sehr groß wird (ca. 2GB), ist es ausreichend, wenn sich der Bericht auf einen Teil der Aufgabe beschränkt.

- **Image Details**

Name	`Hunter XP.E01`
Größe	`585.230.499 Byte`
MD5 Prüfsumme	`8B40554177D2DD0B385A253EEF9A49E8`

- **Forensische Untersuchung**

Damit bei der Demoversion die Untersuchung gestartet werden kann, muss das Image per Drag & Drop in das Programm gezogen werden. Anschließend sind noch die Daten über den Fall einzugeben.

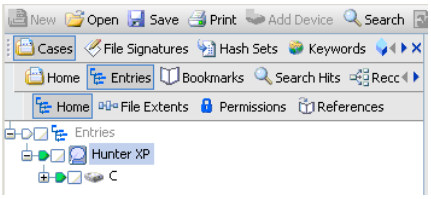


Nach dem Klicken auf `Fertig stellen` beginnt das Programm mit dem Einlesen und Verifizieren des Images. Dieser Vorgang nimmt je nach Computer einige Minuten in Anspruch. Nachdem dieser Vorgang beendet ist, kann mit der Analyse des Images begonnen werden.

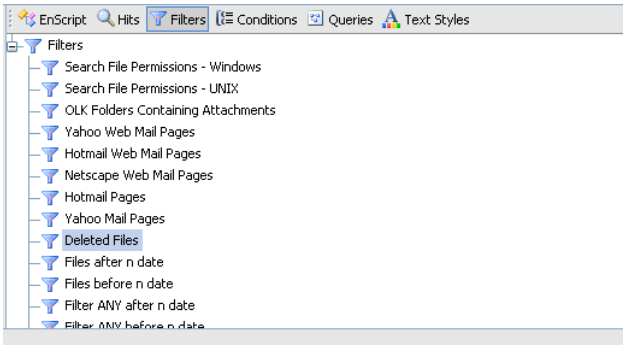
Lösung der Aufgaben

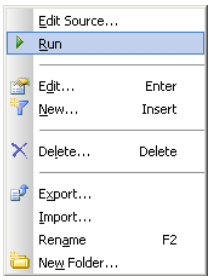
- **Anzeigen gelöschter Dateien durch die Anwendung eines Filters**

Da der benötigte Filter für diese Aufgabe schon vorhanden ist, ist es einfach, diesen anzuwenden. Zuerst muss, wie in dem Bildschirmausschnitt gezeigt, das zu untersuchende Beweismittel, in unserem Fall der Inhalt des Images, durch Anklicken des Fünfeckes links neben der Bezeichnung `Hunter XP` markiert werden. Wenn das Fünfeck grün dargestellt ist, ist die Markierung aktiv.

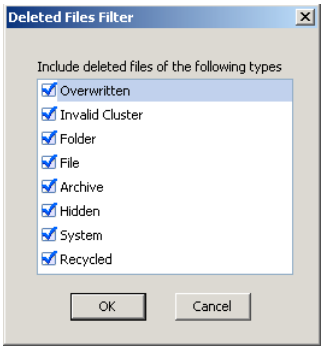


Im nächsten Schritt muss der gewünschte Filter aus der Liste der Filter ausgewählt und aktiviert werden. Die Aktivierung erfolgt entweder über einen Doppelklick oder einen Klick mit der rechten Maustaste auf den Filter und einen weiteren Klick auf `Run`, wie in den beiden Bildschirmausschnitten zu sehen ist.





Nach dem Starten des Filters öffnet sich das unten gezeigte Fenster in welchem die Dateien, die der Filter anzeigen soll, ausgewählt werden können.



Nach dem Klicken auf `OK` wird der Filter aktiv und das Resultat wird, wie in diesem Bildschirmausschnitt zu sehen ist, angezeigt.

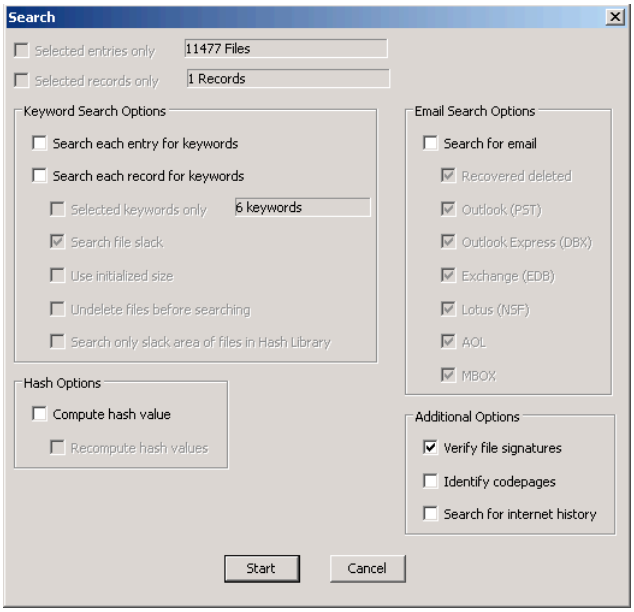
Refresh | Deleted Files | Display

Table | Report | Gallery | Timeline | Disk | Code

	Name	Filter	In Report	File Ext	File Type	File Category	Signature
1	003771500145_LG[1].gif	Deleted Files		gif	GIF	Picture	
2	102-0208_IMG[1].jpg	Deleted Files		jpg	JPEG	Picture	
3	102-0250_IMG.JPG.lnk	Deleted Files		lnk	Link	Windows	
4	104-0406_IMG.JPG	Deleted Files		JPG	JPEG	Picture	
5	104-0407_IMG.JPG	Deleted Files		JPG	JPEG	Picture	
6	104-0489_IMG.JPG.lnk	Deleted Files		lnk	Link	Windows	
7	_23749_moneybox0323_130[1].jpg	Deleted Files		jpg	JPEG	Picture	
8	_23749_msn0323_60[1].gif	Deleted Files		gif	GIF	Picture	
9	a5rfiles	Deleted Files					
10	action.txt	Deleted Files		txt	Text	Document	
11	addcn30.dll	Deleted Files		dll	Dynamic Link Library	Code\Library	
12	address	Deleted Files					
13	address	Deleted Files					
14	address	Deleted Files					
15	address.bak	Deleted Files		bak	Backup	Document	
16	address.dat	Deleted Files		dat	Data ASCII / Binary	Code\Library	
17	address.txt	Deleted Files		txt	Text	Document	
18	address2.txt	Deleted Files		txt	Text	Document	
19	AddressCitiesDB.PDB	Deleted Files		PDB			
20	AddressCompaniesDB.PDB	Deleted Files		PDB			
21	AddressCountriesDB.PDB	Deleted Files		PDB			
22	AddressStatesDB.PDB	Deleted Files		PDB			
23	AddressTitlesDB.PDB	Deleted Files		PDB			

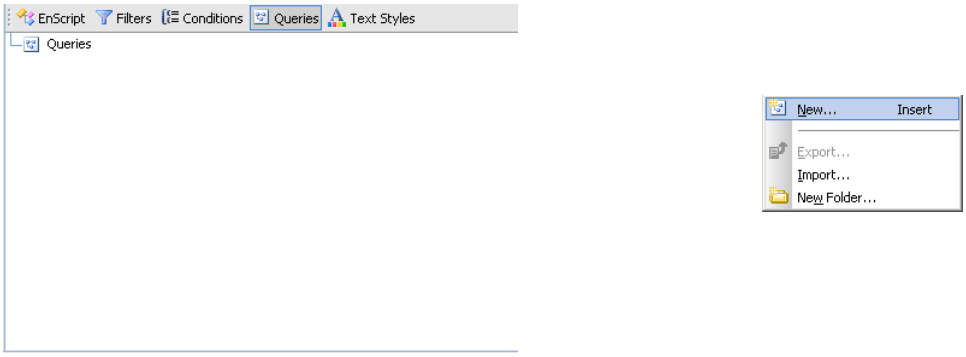
- **Suche nach Bildern in den temporären Dateien durch das Erstellen einer eigenen Query**

Für das Lösen dieser Aufgabe ist es zuerst notwendig, eine eigene Query zu erstellen. Damit diese Query erfolgreich ausgeführt werden kann, müssen zuerst die Signaturen der Dateien verifiziert werden. Dies wird, wie im folgenden Screenshot gezeigt, durch das Markieren der Funktion `Verify file signatures` im Startfenster der Suchfunktion erledigt. Mit dem Start der Suchfunktion wird mit der Verifizierung der Dateisignaturen begonnen.

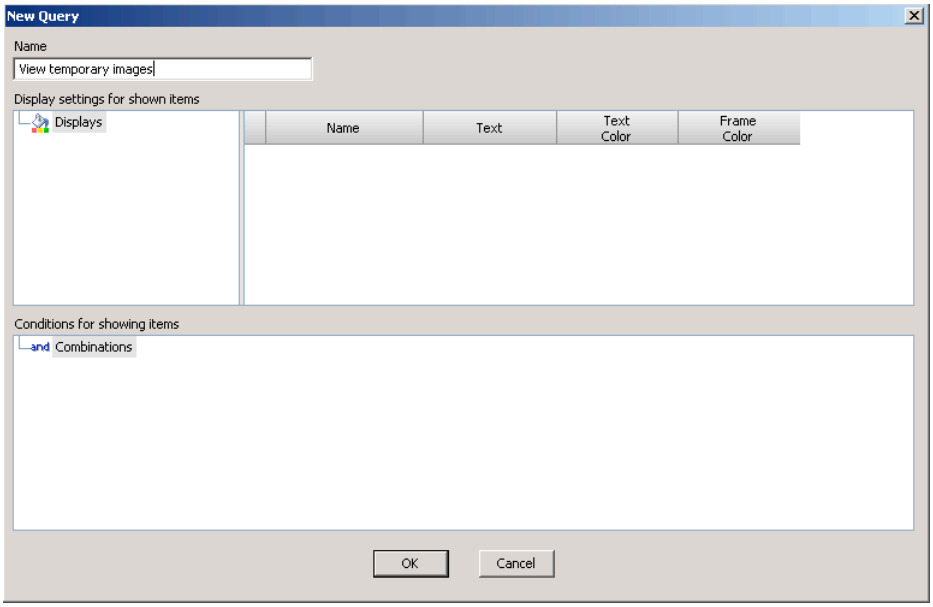


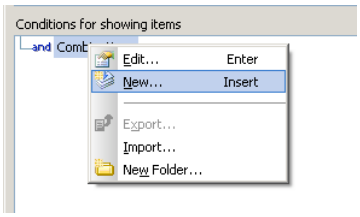
Während dieser Vorgang läuft, kann mit dem Erstellen der Query begonnen werden.

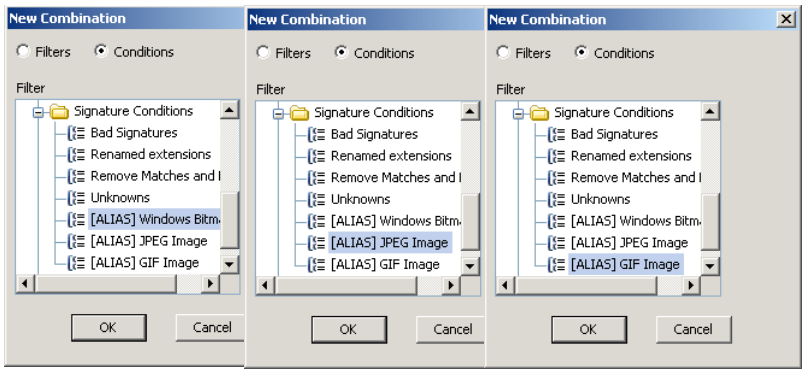
Eine neue Query wird durch einen Rechtsklick in das Feld der Queries und über die Auswahl der Funktion `New...` erstellt.



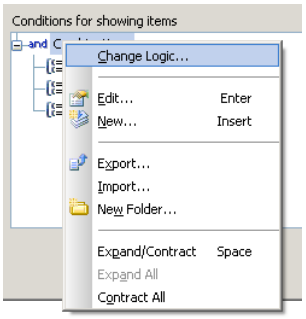
Diese Funktion öffnet ein Fenster, in dem Sie die neue Query konfigurieren können. Damit die neue Query die gestellte Aufgabe löst, müssen `Combinations` eingefügt werden. Die folgenden drei Screenshots bzw. Bildschirmausschnitte zeigen, wie eine neue Combination erstellt wird und welche `Combinations` notwendig sind.

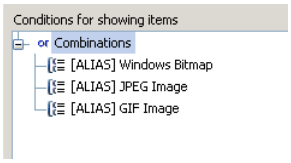




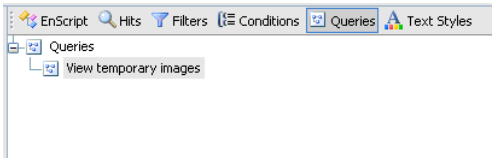


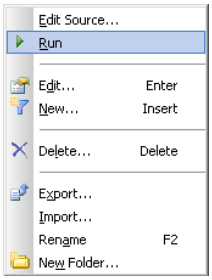
Nach dem Anlegen der `Combinations` muss noch die Verknüpfungslogig von UND auf ODER geändert werden, denn ein Bild kann nicht JPEG und Bitmap gleichzeitig sein.



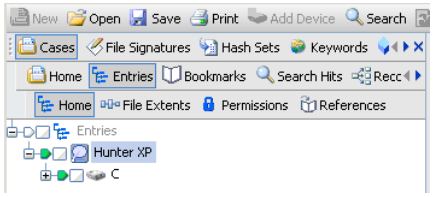


Das Klicken auf `OK` schließt das Bearbeitungsfenster und stellt die neue Query in der Liste der Queries zur Verfügung.

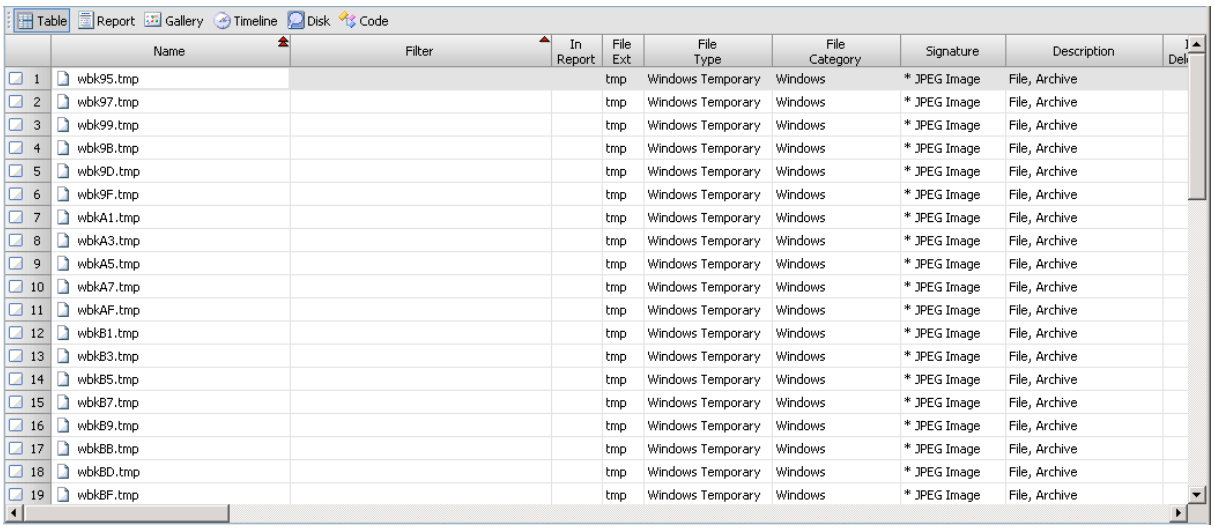




Bevor die neue Query entweder über einen Doppelklick oder einen Klick mit der rechten Maustaste auf den Filter und einen weiteren Klick auf `Run`, ausgeführt werden kann, muss noch überprüft werden, ob der richtige Beweis bzw. die richtige Partition markiert ist. Zur Erinnerung ein Bildschirmausschnitt, der die markierte Partition zeigt.



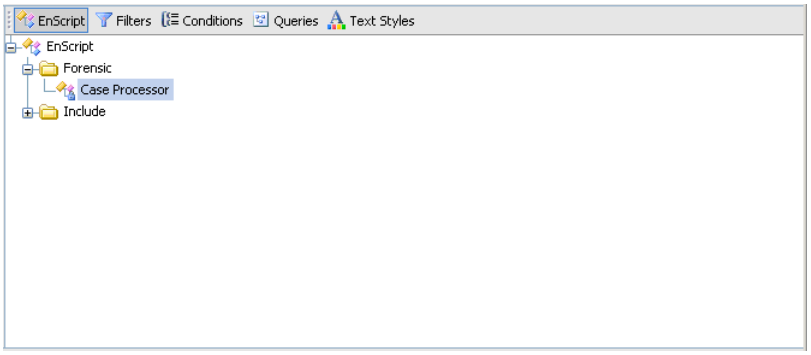
Im nächsten Bildschirmausschnitt wird ein Teil des Resultates dieser Query gezeigt.



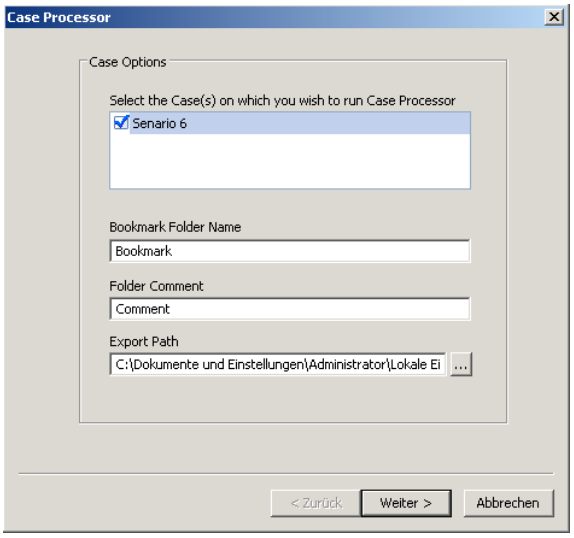
	Name	Filter	In Report	File Ext	File Type	File Category	Signature	Description
1	wbk95.tmp			tmp	Windows Temporary	Windows	* JPEG Image	File, Archive
2	wbk97.tmp			tmp	Windows Temporary	Windows	* JPEG Image	File, Archive
3	wbk99.tmp			tmp	Windows Temporary	Windows	* JPEG Image	File, Archive
4	wbk9B.tmp			tmp	Windows Temporary	Windows	* JPEG Image	File, Archive
5	wbk9D.tmp			tmp	Windows Temporary	Windows	* JPEG Image	File, Archive
6	wbk9F.tmp			tmp	Windows Temporary	Windows	* JPEG Image	File, Archive
7	wbkA1.tmp			tmp	Windows Temporary	Windows	* JPEG Image	File, Archive
8	wbkA3.tmp			tmp	Windows Temporary	Windows	* JPEG Image	File, Archive
9	wbkA5.tmp			tmp	Windows Temporary	Windows	* JPEG Image	File, Archive
10	wbkA7.tmp			tmp	Windows Temporary	Windows	* JPEG Image	File, Archive
11	wbkAF.tmp			tmp	Windows Temporary	Windows	* JPEG Image	File, Archive
12	wbkB1.tmp			tmp	Windows Temporary	Windows	* JPEG Image	File, Archive
13	wbkB3.tmp			tmp	Windows Temporary	Windows	* JPEG Image	File, Archive
14	wbkB5.tmp			tmp	Windows Temporary	Windows	* JPEG Image	File, Archive
15	wbkB7.tmp			tmp	Windows Temporary	Windows	* JPEG Image	File, Archive
16	wbkB9.tmp			tmp	Windows Temporary	Windows	* JPEG Image	File, Archive
17	wbkBB.tmp			tmp	Windows Temporary	Windows	* JPEG Image	File, Archive
18	wbkBD.tmp			tmp	Windows Temporary	Windows	* JPEG Image	File, Archive
19	wbkBF.tmp			tmp	Windows Temporary	Windows	* JPEG Image	File, Archive

- **Anzeige aller Bilder des Images**

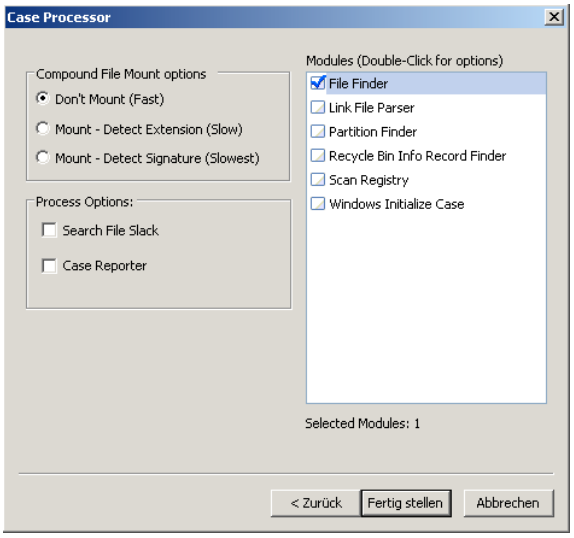
Für das Aufsuchen aller Bilder des Images wird der `Case Processor`, welcher als `EnScript` Anwendung mit `EnCase` mitgeliefert wird, verwendet.



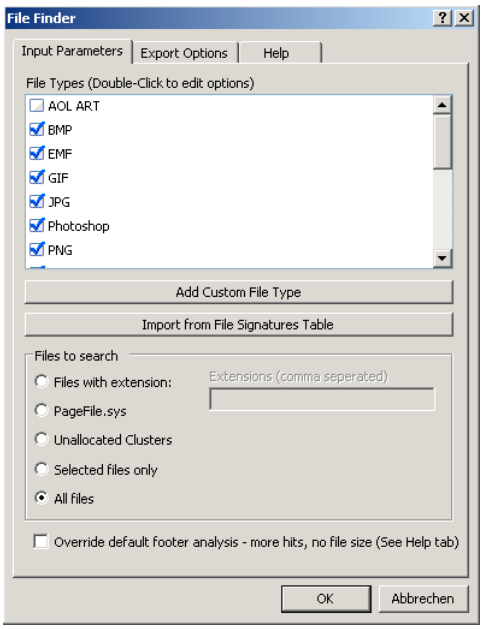
Gestartet wird der `Case Processor` über einen Doppelklick oder über die rechte Maustaste und `Run`. Da der `Case Processor` die Ergebnisse als Bookmarks speichert, müssen zu Beginn die Ordner der Bookmark-Struktur angegeben werden.



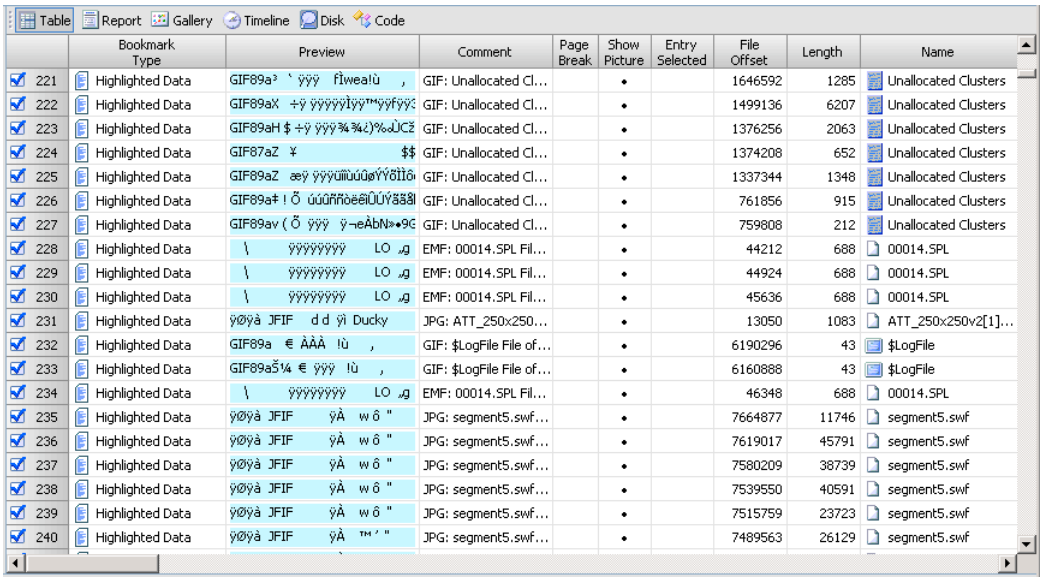
Für diese Aufgabe wird das `File Finder` Modul, wie im folgenden Screenshot gezeigt, verwendet.



Nach dem Starten des `File Finders` müssen noch die zu suchenden Dateitypen und die zu durchsuchenden Dateien angegeben werden.

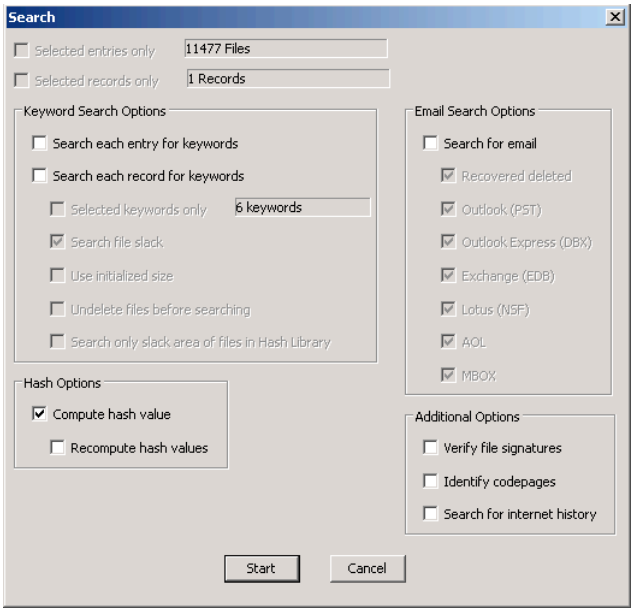


Nach Abschluss des Suchvorgangs befinden sich die Resultate bei den Bookmarks. Einen kleinen Teil des Resultates zeigt der folgende Bildschirmausschnitt.



- **Berechnen der Hashwerte**

Die Berechnung der Hashwerte ist einfach über die Suchfunktion zu erledigen. Hierfür muss nur im Suchdialogfenster die Funktion `Compute hash value` aktiviert und auf `Start` geklickt werden.

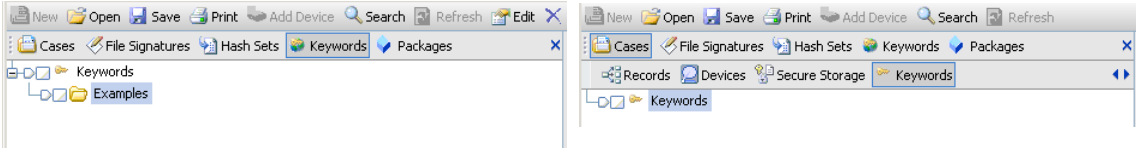


Nach Beendigung dieses Vorgangs werden die berechneten Hashwerte in der Listendarstellung angezeigt.

Code Page	Hash Value	Hash Set
0	5ead9d1c32a5deb91600b0783b7b4699	
0	2f35b6db61617b096bda629678f9cca7	
0	eed38cd7639fc1cb14f486e2f7d29546	
0	ec20cf24abcee99cbccd60f19913e0e8	
0	db36fb3c0170105aaa12b18f34a6b4a1	
0	4e86632343cae9ff63551a3baa9da3f5	
0	1de4d799f5ac1f0d1d63f69070fef0e4	
0	f4e3609bf826ee29f52efffb8a5dec9e	
0	f4e3609bf826ee29f52efffb8a5dec9e	
0	cec6f1262f2563db19574238c959d5f7	
0	5ead9d1c32a5deb91600b0783b7b4699	
0	2f35b6db61617b096bda629678f9cca7	
0	eed38cd7639fc1cb14f486e2f7d29546	
0	ec20cf24abcee99cbccd60f19913e0e8	
0	db36fb3c0170105aaa12b18f34a6b4a1	
0	4e86632343cae9ff63551a3baa9da3f5	
0	1de4d799f5ac1f0d1d63f69070fef0e4	
0	f4e3609bf826ee29f52efffb8a5dec9e	

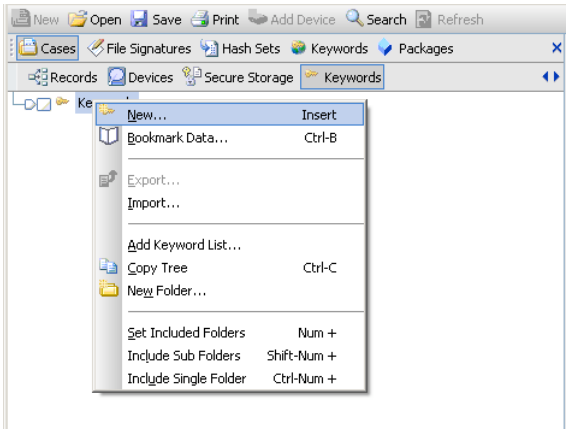
- **Erstellen von „Keywords“ und anschließendes Suchen nach diesen**

Zu einer wichtigen Funktionalität jedes Forensik-Programms zählt die Suche nach Keywords. In `EnCase` werden die Keywords in globale, welche für jeden Fall verwendet werden können, und in lokale, welche nur für einen speziellen Fall verwendet werden können, unterteilt.

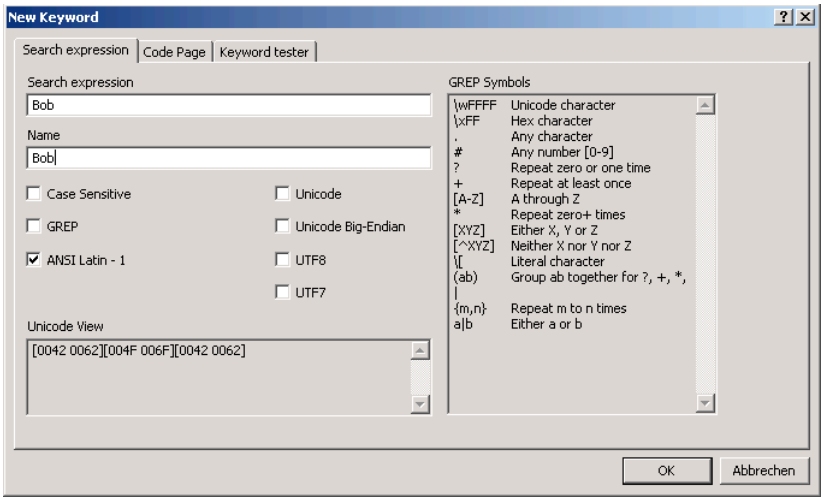


In diesem Beispiel wird die Verwendung lokaler Keywords gezeigt. Der Unterschied zu den globalen liegt nur an der Stelle, wo diese angelegt werden.

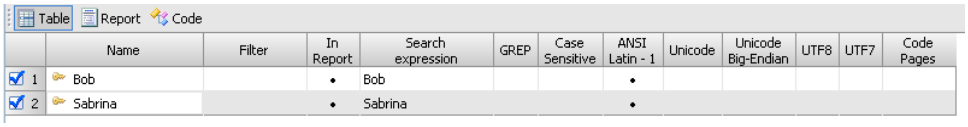
Wie der folgende Bildschirmausschnitt zeigt, werden die Keywords wieder über das Kontextmenü der rechten Maustaste und der Funktion `New...` angelegt.



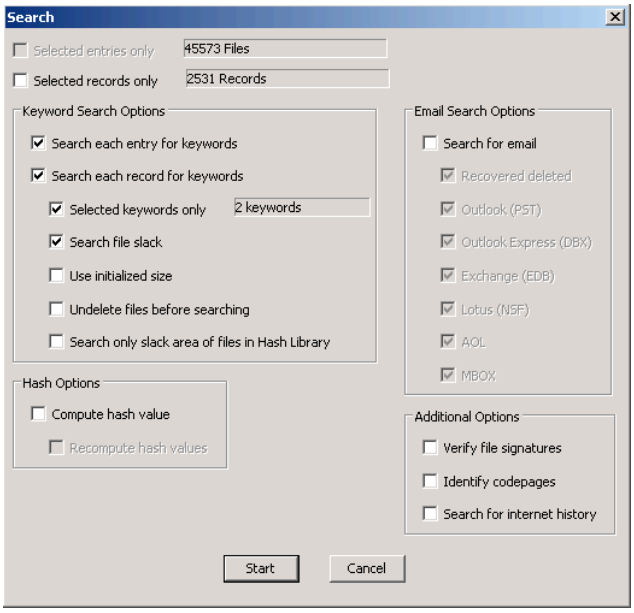
Anschließend öffnet sich ein Fenster für das Anlegen des neuen Keywords. Für diese Aufgabe ist es ausreichend, einfache Wörter als `Search expression` anzugeben.



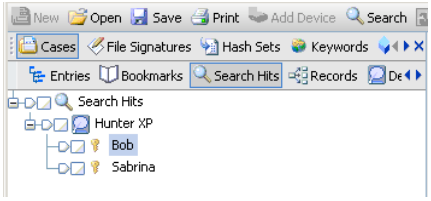
In diesem Beispiel wurden zwei Keywords, nach denen das Image durchsucht werden soll, angelegt. Damit gezielt nach diesen gesucht wird, müssen sie vor dem Starten der Suche markiert werden.

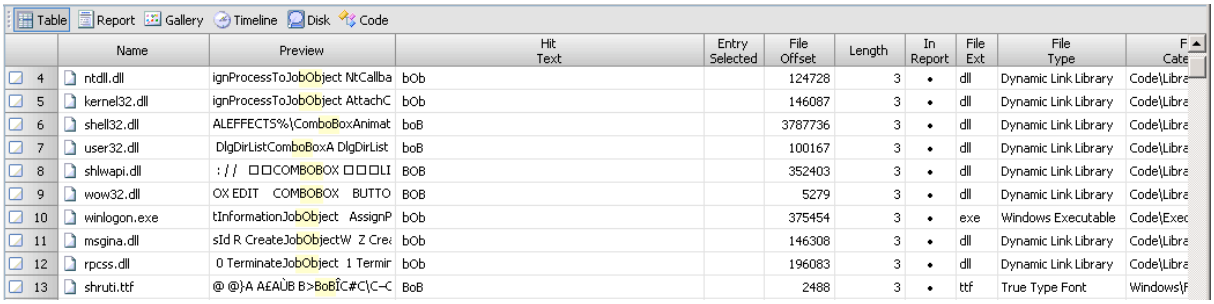


Im Dialog der Suchfunktion müssen Sie vor dem Suchen noch die `Keyword Search Options` anpassen. Wobei hier das Markieren der Funktion `Selected keywords only` am wichtigsten ist. Zu beachten ist weiters, dass die globalen Keywords nicht markiert sind.



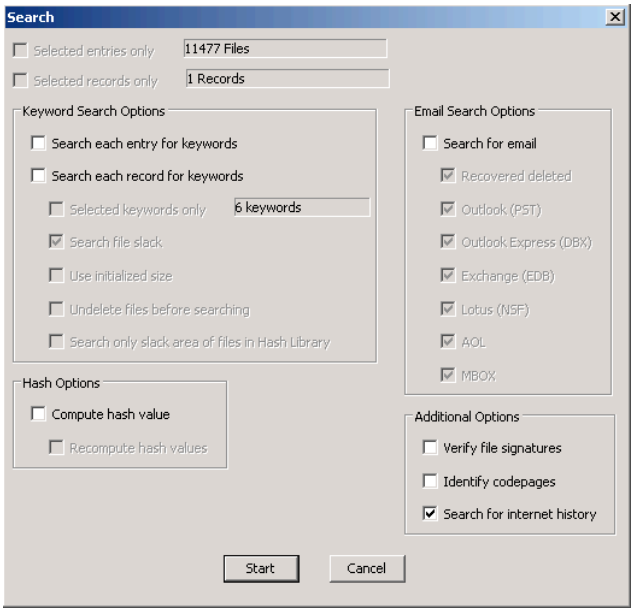
Die Ergebnisse der Suchanfragen werden unter den `Search Hits` nach den Keywords sortiert aufgelistet. Die beiden folgenden Bildschirmausschnitte zeigen einen Teil des Ergebnisses der Suche nach den zuvor angelegten Keywords.



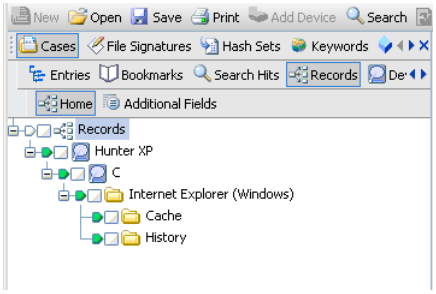


- **Suchen nach Internet History Daten**

Das Suchen nach Internet History Daten und Cache Inhalten erfolgt einfach über das Aktivieren der Funktion `Search for internet history` der Suche.

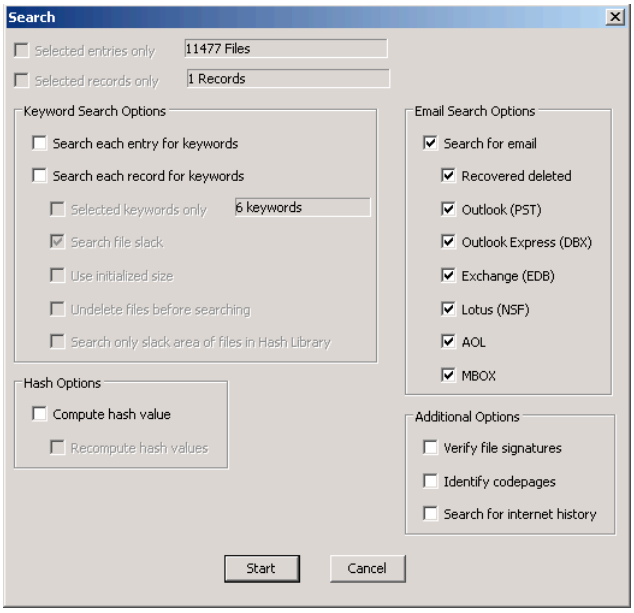


Die Ergebnisse dieser Suche werden unter den `Records` gespeichert und können dort angezeigt werden.

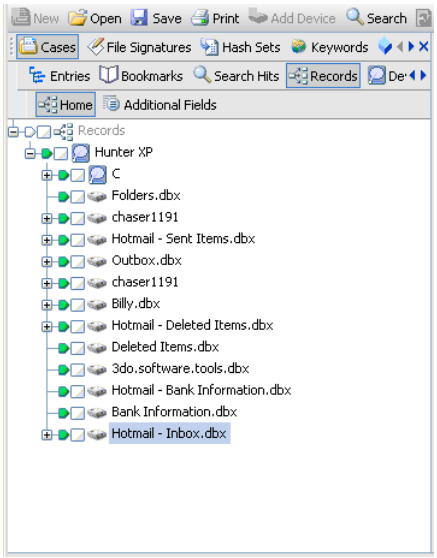


- **Suchen nach E-Mails**

Das Suchen nach E-Mail Daten erfolgt einfach über das Aktivieren der Funktion `Search for email` der Suche. Weiters kann noch angegeben werden, nach welchen Mailformaten gesucht werden soll.



Die Ergebnisse dieser Suche werden unter den `Records` gespeichert und können dort angezeigt werden.



- **Zusammenfassung**

Nach der Analyse dieses Demoimages ist es schwierig zu sagen, ob alle Daten gefunden wurden. Wobei das bei diesem Szenario auch nur zweitrangig ist, da es primär um die Handhabung des Programms EnCase und um die möglichen Ergebnisse geht.

7.2.7 Szenario 7: EnCase vs. Konkurenz

- **Übungsangabe**

Bei diesem Szenario soll dasselbe Image wie in Szenario 6 verwendet werden. Im Gegensatz zu Szenario 6 soll dieses Mal ein anderes Computer-Forensik-Programm für die Analyse verwendet werden.

Dabei versuchen Sie folgende Aufgaben zu lösen:

- Suche nach allen gelöschten Dateien und Ordner
- Suche nach Bildern allgemein und im Speziellen nach Bildern in den temporären Dateien

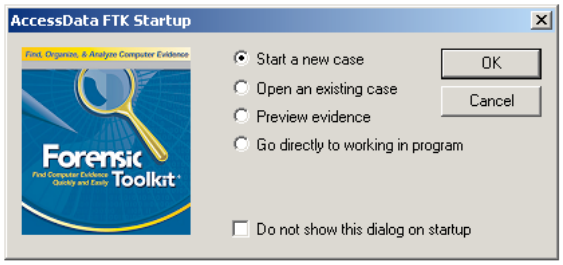
Im Bericht sollen Sie anführen ob es, und wenn ja welche, Unterschiede in den beiden Ergebnissen gibt.

- **Image Details**

Name	`Hunter XP.E01`
Größe	`585.230.499 Byte`
MD5 Prüfsumme	`8B40554177D2DD0B385A253EEF9A49E8`

- **Forensische Untersuchung**

Damit man das Image dieses Szenarios untersuchen kann, müssen Sie einen neuen Fall anlegen. Für diese Aufgabe bietet das Programm `Forensic Toolkit` einen eigenen Wizard der nach dem Starten des Programms ausgeführt werden kann.

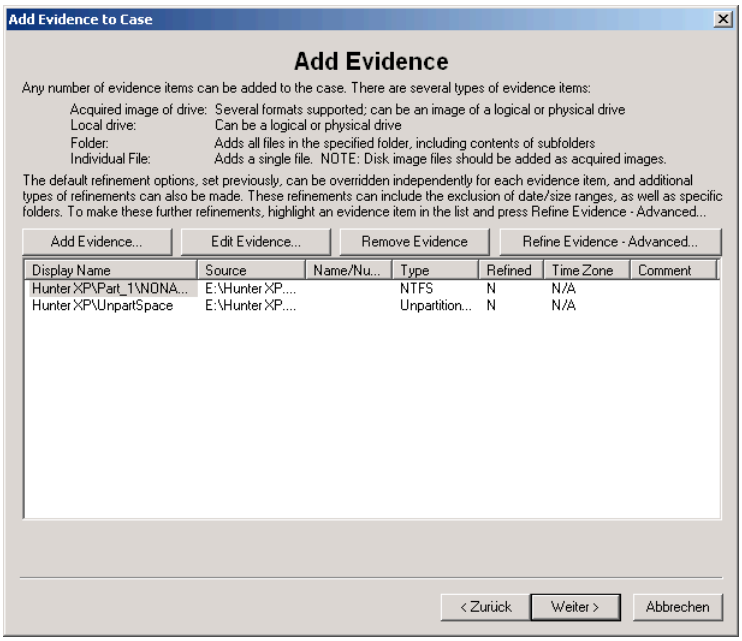


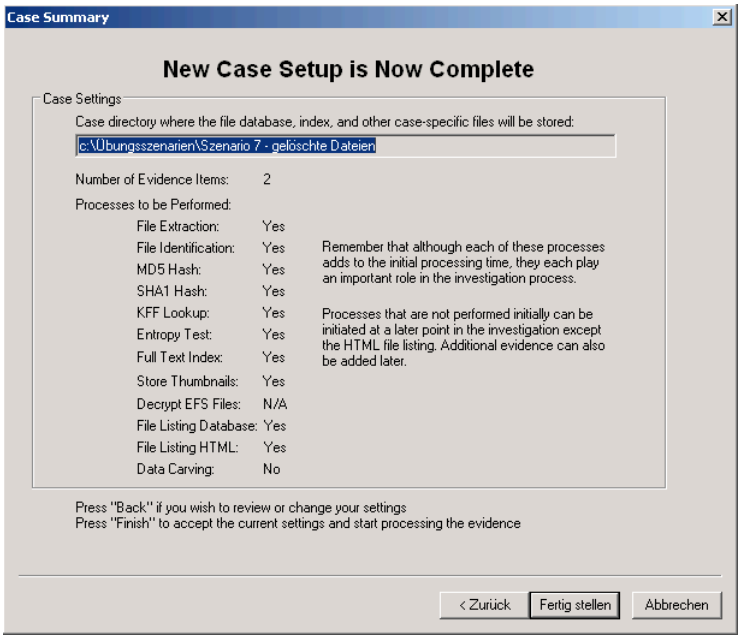
Für eine bessere Übersicht und da das Programm `Forensic Toolkit` in der Testversion nur 5000 Dateien pro Fall verarbeiten kann, wird diese Aufgabe in zwei Fälle unterteilt.

- **Suchen nach allen gelöschten Dateien und Ordner**

Da das Lösen dieser Aufgabe gleich erfolgt wie die Untersuchung in Szenario 2, wird für die Details auf das Kapitel 7.2.2 (Szenario 2: FAT32 in Windows ME) verwiesen.

Zusammenfassend hier noch die Auflistung der hinzugefügten Beweise.

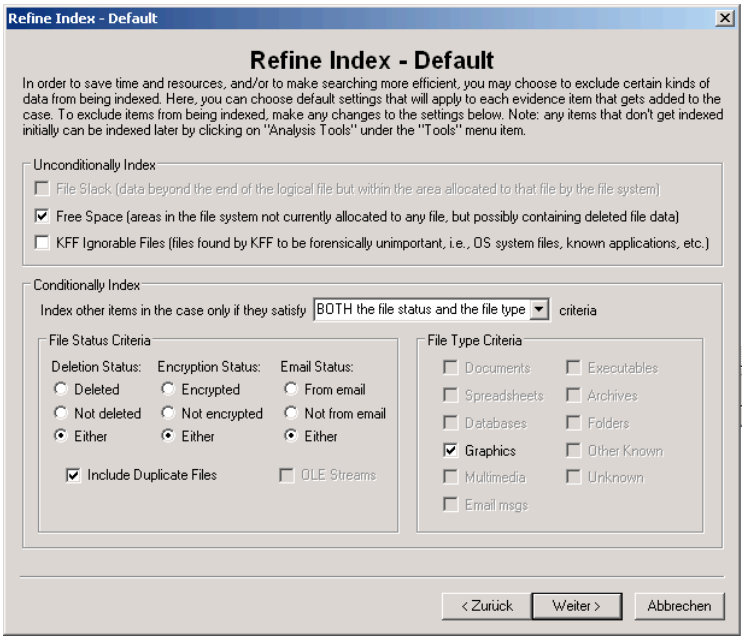




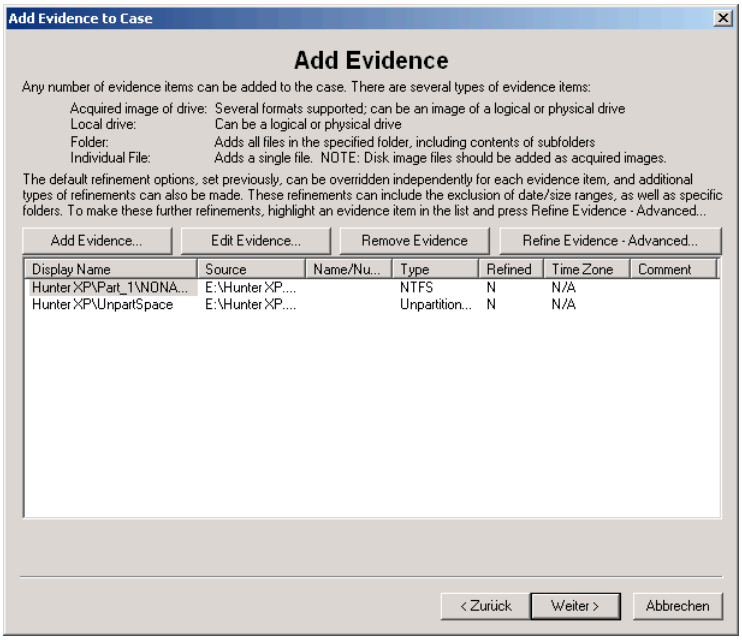
Nachdem alle Informationen angegeben wurden, kann über das Klicken auf den Button `Fertig stellen` mit der Analyse des Images begonnen werden.

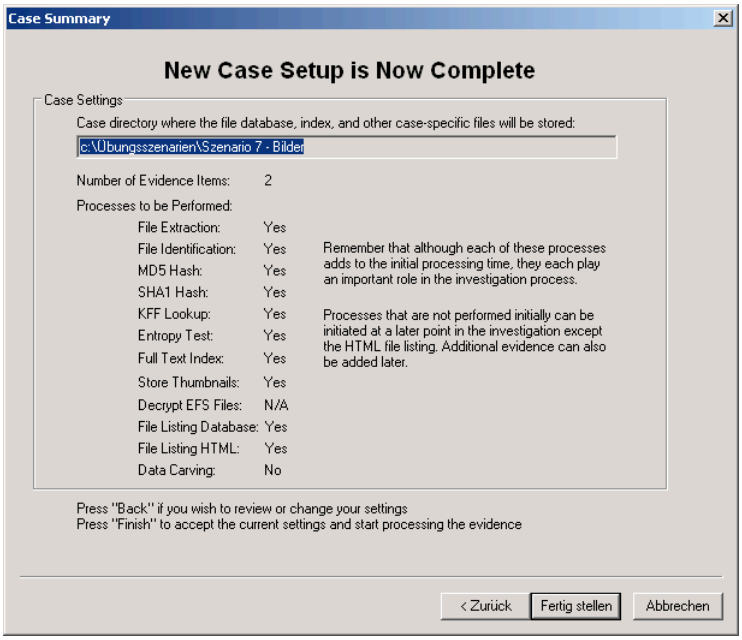
- **Suchen nach Bildern allgemein und im Speziellen nach Bildern in den temporären Dateien**

Da das Lösen dieser Aufgabe gleich erfolgt wie die Untersuchung in Szenario 2, wird für die Details auf das Kapitel 7.2.2 (Szenario 2: FAT32 in Windows ME) verwiesen. Wobei in dem hier gezeigten Schritt die Suche auf gelöschte und nicht gelöschte Bilddateien beschränkt werden soll.



Zusammenfassend hier noch die Auflistung der hinzugefügten Beweise.

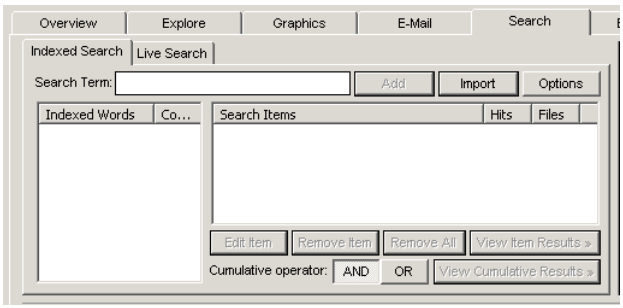


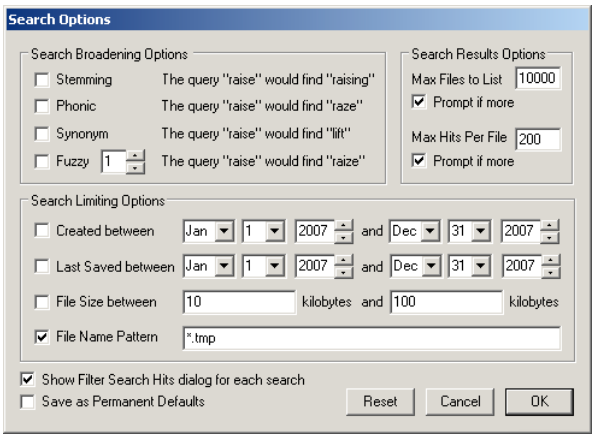


Nachdem alle Informationen angegeben wurden, kann über das Klicken auf den Button `Fertig stellen` mit der Analyse des Images begonnen werden.

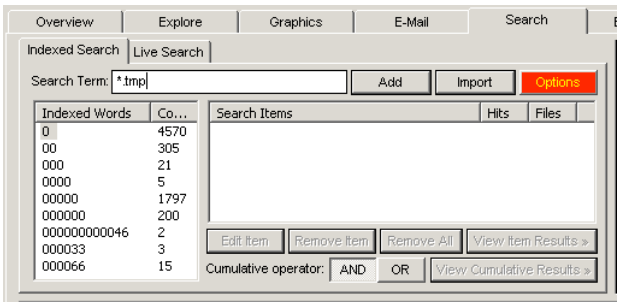
- **Suchen nach Bildern in den temporären Dateien**

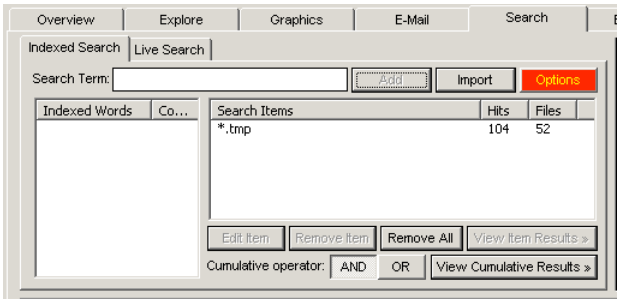
Nachdem die Analyse des Images abgeschlossen ist, werden alle gefundenen Bilder im Bereich `Images` aufgelistet. Damit aus diesen Dateien diejenigen herausgefunden werden, welche als temporäre Dateien gespeichert sind, verwendet man am besten die Suchfunktion.
Die folgende Anleitung beschreibt wie die Suchfunktion für dieses Beispiel verwendet wird.
Vor dem Starten der Suche wird unter den `Search Options` die Suche auf Dateien mit der Endung `.tmp` eingeschränkt. Den Dialog mit den Search Options erhält man über das Anklicken des `Options` Buttons.
Der Eintrag `*.tmp` im Feld `File Name Pattern` bewirkt die Einschränkung auf Dateien mit der Endung `.tmp`.



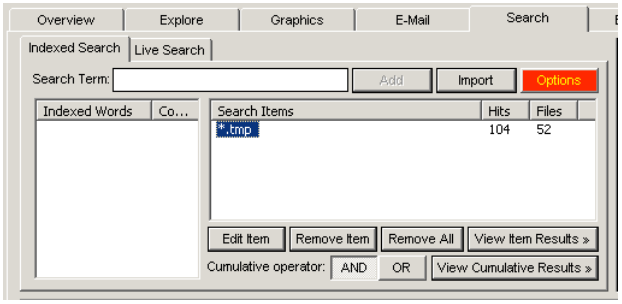


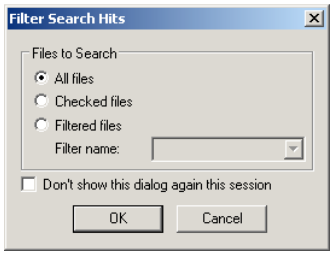
Der rot hinterlegte `Options` Button zeigt an, dass Suchoptionen aktiviert sind. Als `Search term` muss weiters `*.tmp` eingegeben und anschließend auf `Add` geklickt werden.



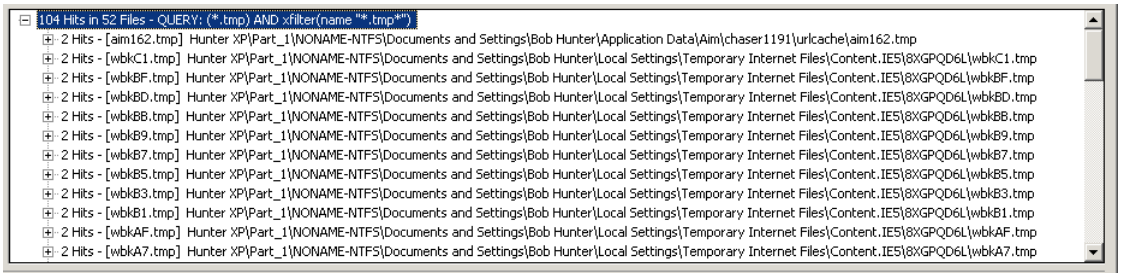


Für das Anzeigen der Suchresultate wird das `Search Item` markiert und anschließend auf `View Item Results >>` geklickt. Jetzt muss nur noch in dem Pop-Up Fenster angegeben werden welche Dateien verwendet werden sollen.

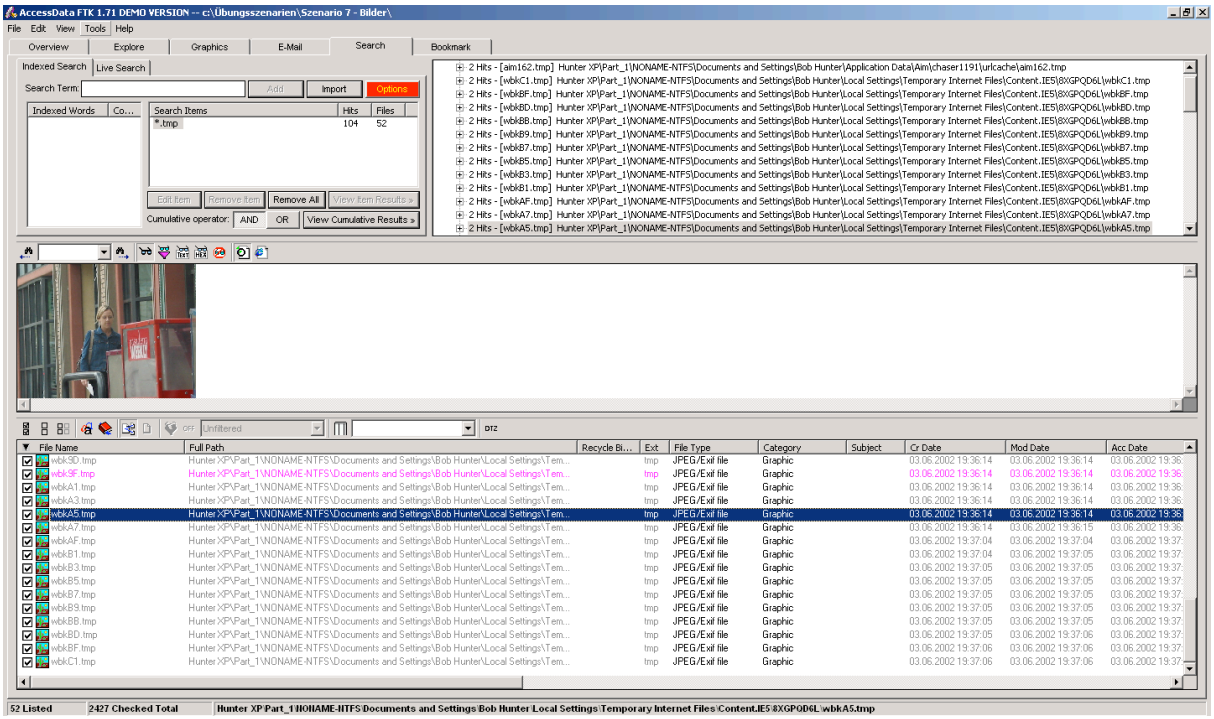




Mit einem Klick auf `OK` werden die Ergebnisse im Fenster rechts daneben, wie der folgende Bildschirmausschnitt zeigt, dargestellt.



Der gesamte Screenshot zeigt die Anordnung der einzelnen Elemente und einen Ausschnitt aus der Ergebnisliste.



- **Ergebnisse der forensischen Untersuchung**

Aufgrund der umfangreichen Ergebnisse werden diese hier nicht gesondert angeführt, sondern können im generierten Bericht, welcher mit dem Forensik-Programm erzeugt wurde, angesehen werden.

- **Zusammenfassung**

Der Vergleich der Ergebnisse mit denen derselben Aufgaben aus Szenario 6 zeigt, dass beide Programme exakt dieselben Dateien finden. Dies kann über den Vergleich der Dateinamen und der Prüfsummen nachgeprüft werden.

7.2.8 Szenario 8: Live View

- **Übungsangabe**

Ziel dieses Szenarios ist es zu zeigen, wie wichtig die Verwendung korrekt erstellter Images ist, und was sich an einem System nur durch einen Bootvorgang ändert.
Für diesen Zweck soll aus einem raw Image mit dem Programm `Live View` ein lauffähiges `VMware` Image erzeugt und anschließend als virtuelle Maschine gestartet werden. Nach dem Starten sollen Sie sich mit einem vorinstallierten Forensik-Programm die MAC-Änderungen zum Beispiel am `Windows` Ordner oder am Ordner für die Benutzereinstellungen ansehen. Nach dieser Analyse können einzelne Abschnitte der Festplatte bzw. Dateien mit der Hexeditor-Ansicht zwischen dem Original-Image und dem gestarteten System verglichen werden.

- **Image Details**

Name	`Szenario-8-winxp-NTFS.dd`
Größe	`1.573.060.608 Byte`
MD5 Prüfsumme	`366331EA95BE0CE1761B00DF87CD020C`

- **Installation von LiveView**

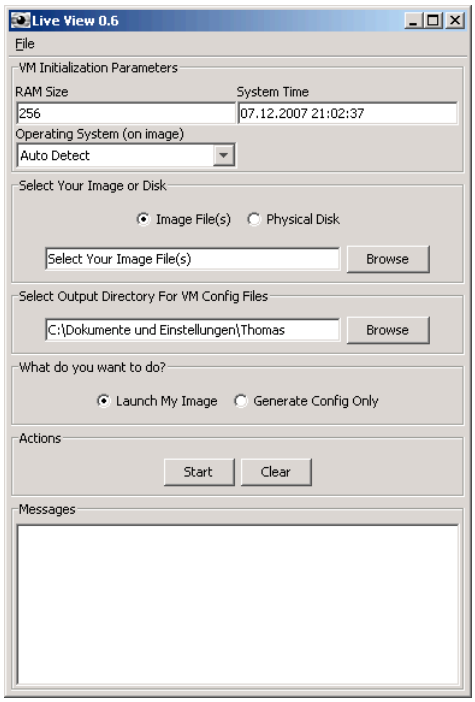
Die Verwendung von `LiveView` benötigt einige zusätzliche Programme. Fehlen diese Programme, so installiert das Installationsprogramm, bei vorhandener Internetverbindung, selbstständig diese Programme. Damit dieser Vorgang nicht jeder Student durchführen muss, befinden sich auf der Installations-CD alle benötigten Programme. Wobei jedoch darauf zu achten ist, dass diese Programme vor der Installation von `LiveView` installiert werden. Konkret handelt es sich dabei um:

- Java-Laufzeitumgebung
- VMware Disk-Mount Utility
- VMware Server

Damit man die Installation einfach durchführen kann, befinden sich auf der Installations-CD zwei Batchdateien, eine mit Java-Installation und eine ohne, für die Installation aller Programme inklusive `LiveView`. Da für den `VMware Server` eine Lizenz erforderlich ist, befindet sich auf der Installations-CD weiters eine Text-Datei mit Lizenzschlüsseln, welche von `VMware` angefordert wurden.

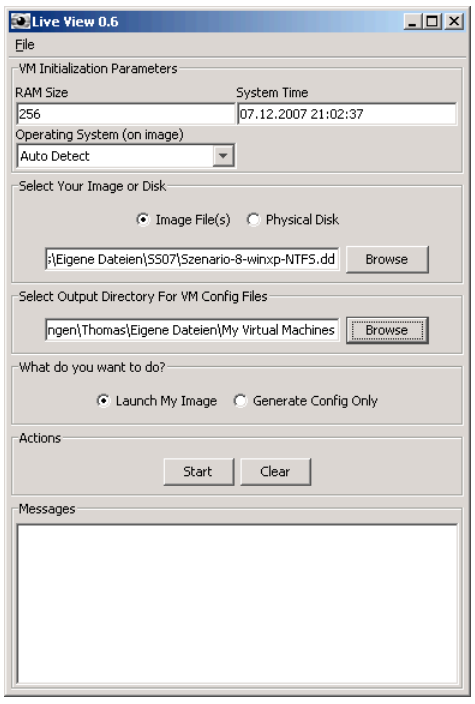
- **Verwendung von LiveView**

Der folgende Screenshot zeigt die Benutzeroberfläche von `LiveView`.

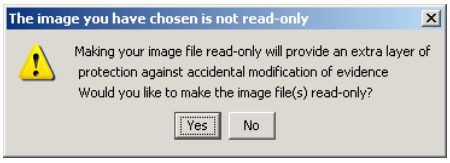


Für dieses Übungsszenario ist es ganz wichtig, dass Sie in das Feld `System Time`, welches das Datum enthält mit dem die neue virtuelle Maschine gestartet wird, das Datum `07.12.2007` eintragen, denn die Windows Installation wurde nach der Installation nicht bei Microsoft registriert und würde daher bei einem anderem Datum nicht ohne diese Registrierung starten.

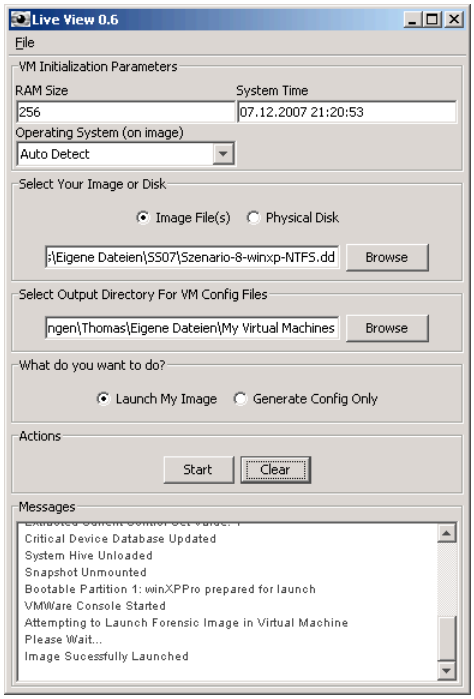
Anschließend ist nur noch das zu verwendende Image und der Speicherort für die Konfiguration der virtuellen Maschine auszuwählen. Durch das Klicken auf den `Start` Button wird mit der Konfiguration der neuen virtuellen Maschine begonnen.



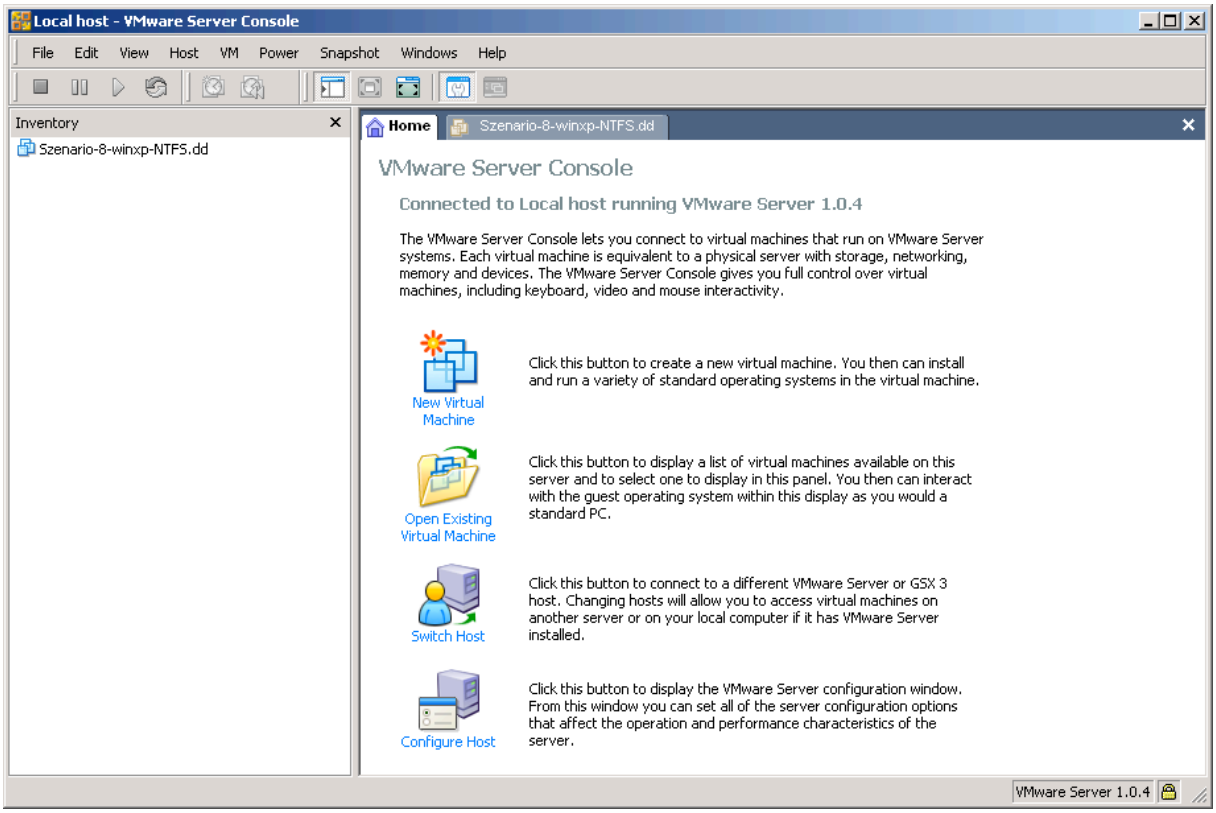
Falls das verwendete Image nicht schreibgeschützt ist, kommt eine Programmmeldung die nachfragt, ob das Image schreibgeschützt werden soll oder nicht. Das Setzen des Schreibschutzes würde hier nur einen zusätzlichen Schutz bieten, da der Inhalt des Image gelesen aber nicht verändert wird. Alle Änderungen werden in einer separaten Konfigurations-Datei der virtuellen Maschine gespeichert.



Wenn die Konfiguration ohne Fehler beendet werden konnte, wird die neue virtuelle Maschine gestartet.



Dieser Screenshot zeigt die generierte virtuelle Maschine in der `VMware Server Console`.

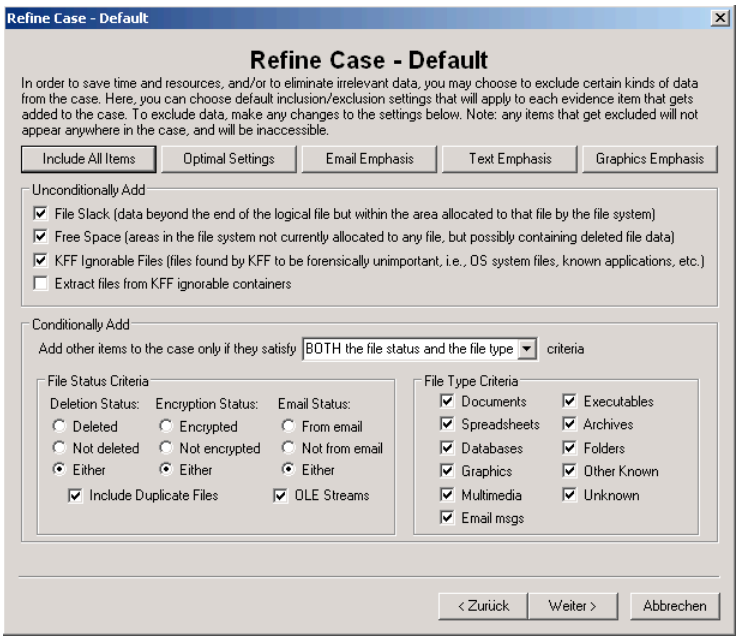


Der erste Start der neuen virtuellen Maschine wird, abhängig vom verwendeten Computer, ein paar Minuten dauern, da Windows zuerst die neu gefundene Hardware installiert. Um ein komfortableres Arbeiten zu erreichen, vor allem die Verwendung der Maus, sollten noch die `VMware Tools` installiert werden. Dafür müssen Sie einfach bei laufender virtueller Maschine im Menü `VM` den Befehl `Install Vmware Tools...` auswählen. Nach der Installation der `VMware Tools` sollte das Datum um einen Tag vor gestellt und die virtuelle Maschine neu gestartet werden. Das Umstellen des Datums hat den Sinn, dass bei der anschließenden Analyse die Zeitstempel eindeutig dem Startvorgang zugeordnet werden können.

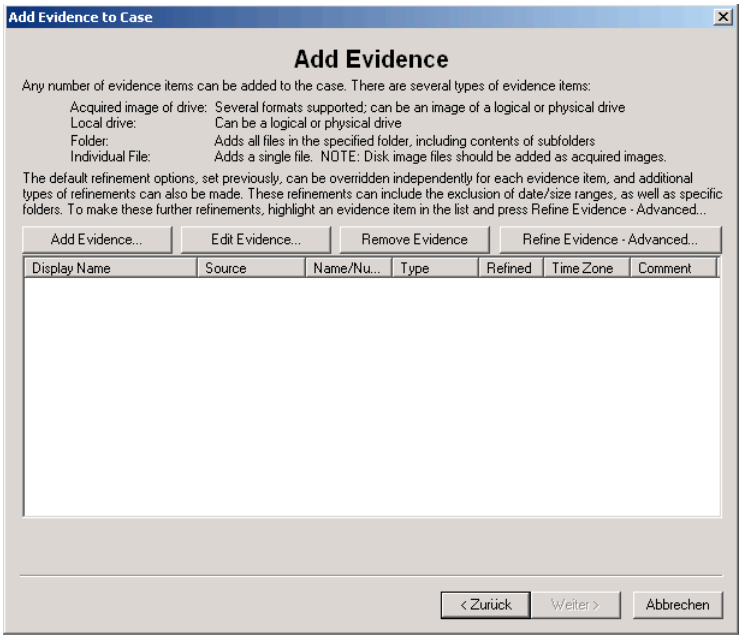
- **Forensische Analyse**

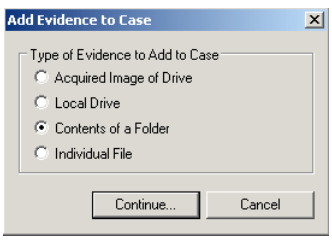
Damit Ordner, welche vom Startvorgang betroffen sind, untersucht werden können, müssen Sie zuerst die vorhin erzeugte virtuelle Maschine starten. Anschließend legen Sie einen neuen Fall für die forensische Analyse an. Für diese Aufgabe bietet das vorinstallierte Programm `Forensic Toolkit` einen eigenen Wizard, der nach dem Starten des Programms gestartet werden kann.

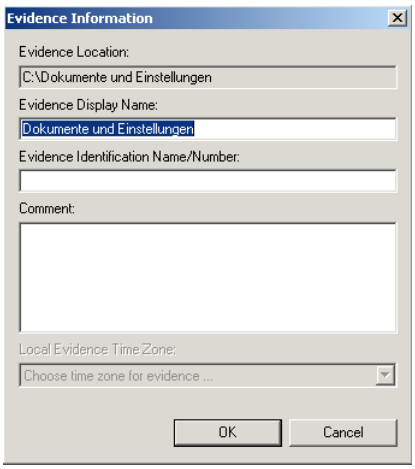
Für die weiteren nicht Szenario spezifischen Schritte wird hier auf die detaillierten Angaben im Kapitel 7.2.2 (Szenario 2: FAT32 in Windows ME) verwiesen. Wobei der erste Unterschied darin besteht, dass die Suche auf keine speziellen Dateien eingeschränkt wird.

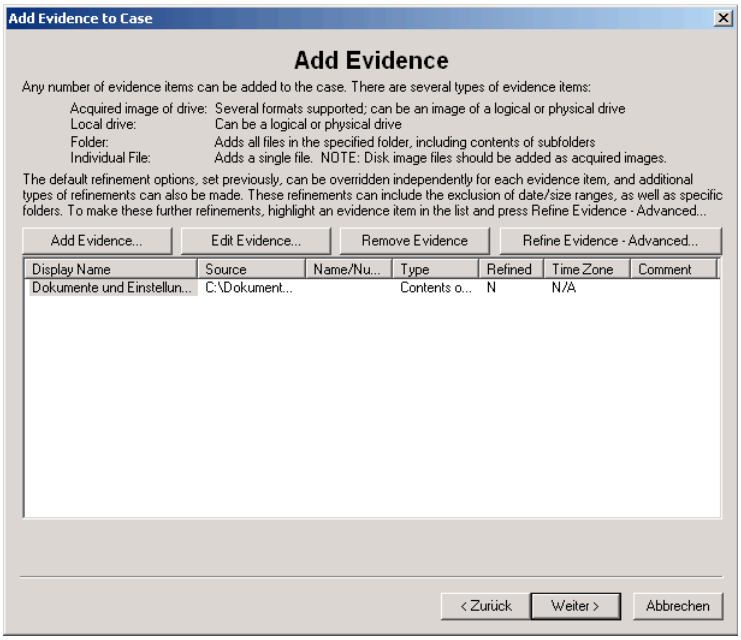


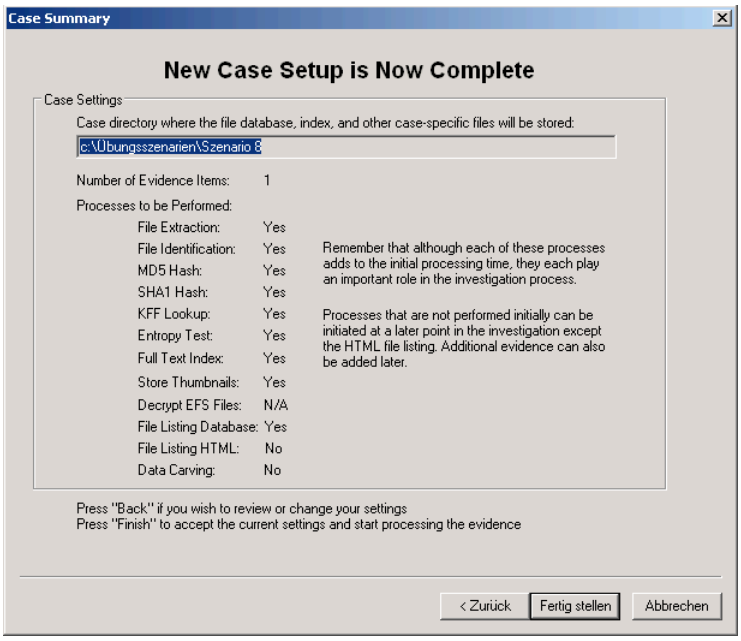
Der zweite Unterschied besteht darin, dass hier nicht ein Image, sondern ein oder mehrere Ordner der virtuellen Festplatte als Beweis verwendet werden. Dafür muss nach dem Klicken auf den `Add Evidence...` Button `Contents of a Folder` anstelle von `Acquired Image or Drive` ausgewählt werden.









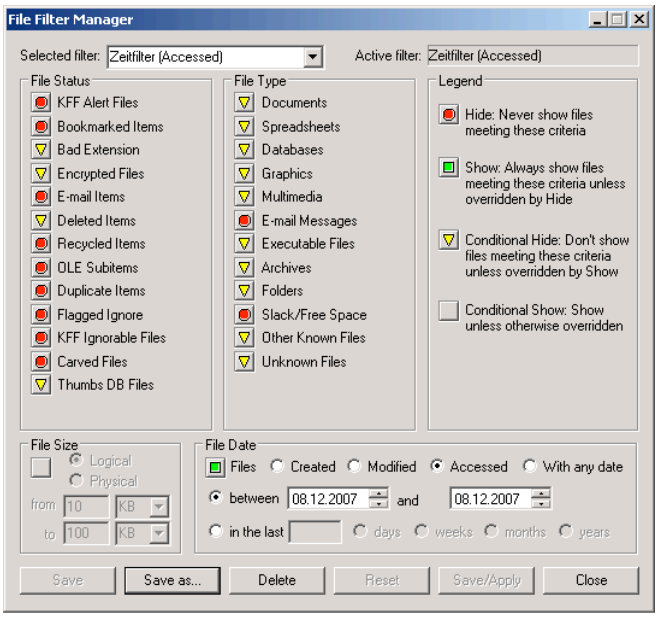


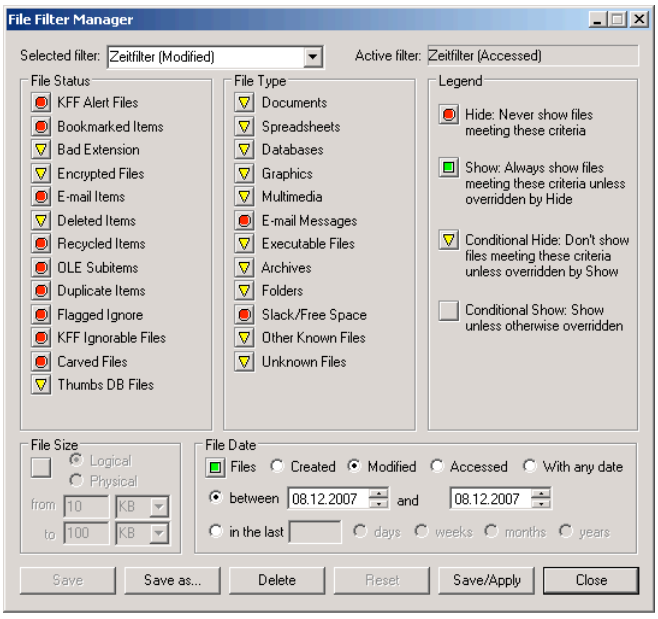
Nachdem alle Informationen angegeben wurden, kann über das Klicken auf den Button `Fertig stellen` mit der Analyse der Beweise begonnen werden.

- **Hinweise für die forensische Untersuchung**

Da es im Programm `Forensic Toolkit` keine Zeitlinien-Ansicht wie beim Programm EnCase gibt, muss hier auf die Funktion der Dateifilter ausgewichen werden. Nachdem es für diesen Fall noch keinen vorgefertigten Filter gibt, ist ein eigener, über das Klicken auf das Symbol des `File Filter Managers`, im Bild ganz links, zu erstellen.

Die folgenden beiden Screenshots zeigen die Einstellungen für das Filtern der Dateien, auf die zuletzt zugegriffen und welche zuletzt bearbeitet wurden. Es ist darauf hinzuweisen, dass bei der Korrektheit der Zeiten auf das Betriebssystem vertraut werden muss.



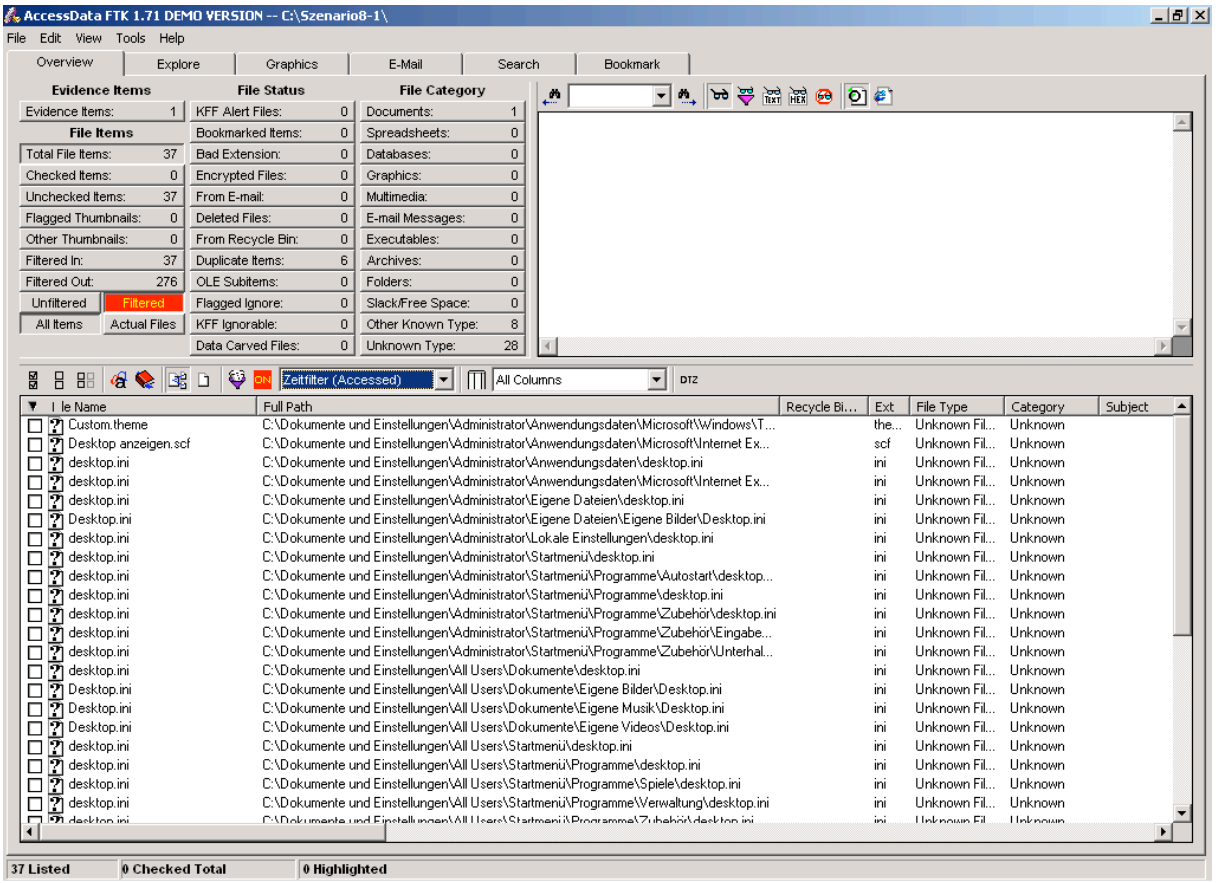


Mit dem Klicken auf `Save as...` wird der neue Filter gespeichert und kann angewendet werden.

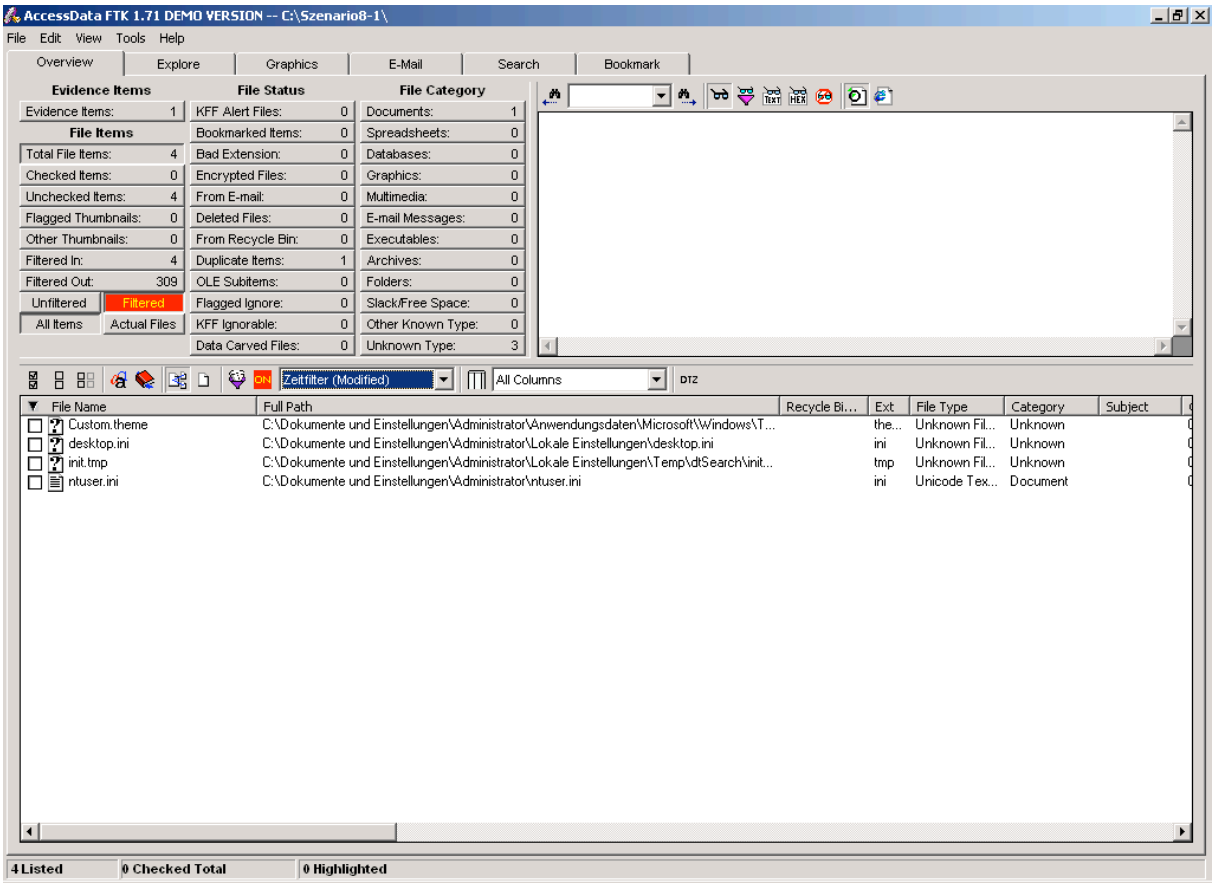
- **Zusammenfassung**

Zusammenfassend hier je zwei Screenshots die die Anwendung der zuvor beschriebenen Dateifilter auf den `Dokumente und Einstellungen` und den `Windows` Ordner zeigen. Bei der Untersuchung des `Windows` Ordner ist zu beachten, dass dieser mehr als 5000 Dateien enthält,

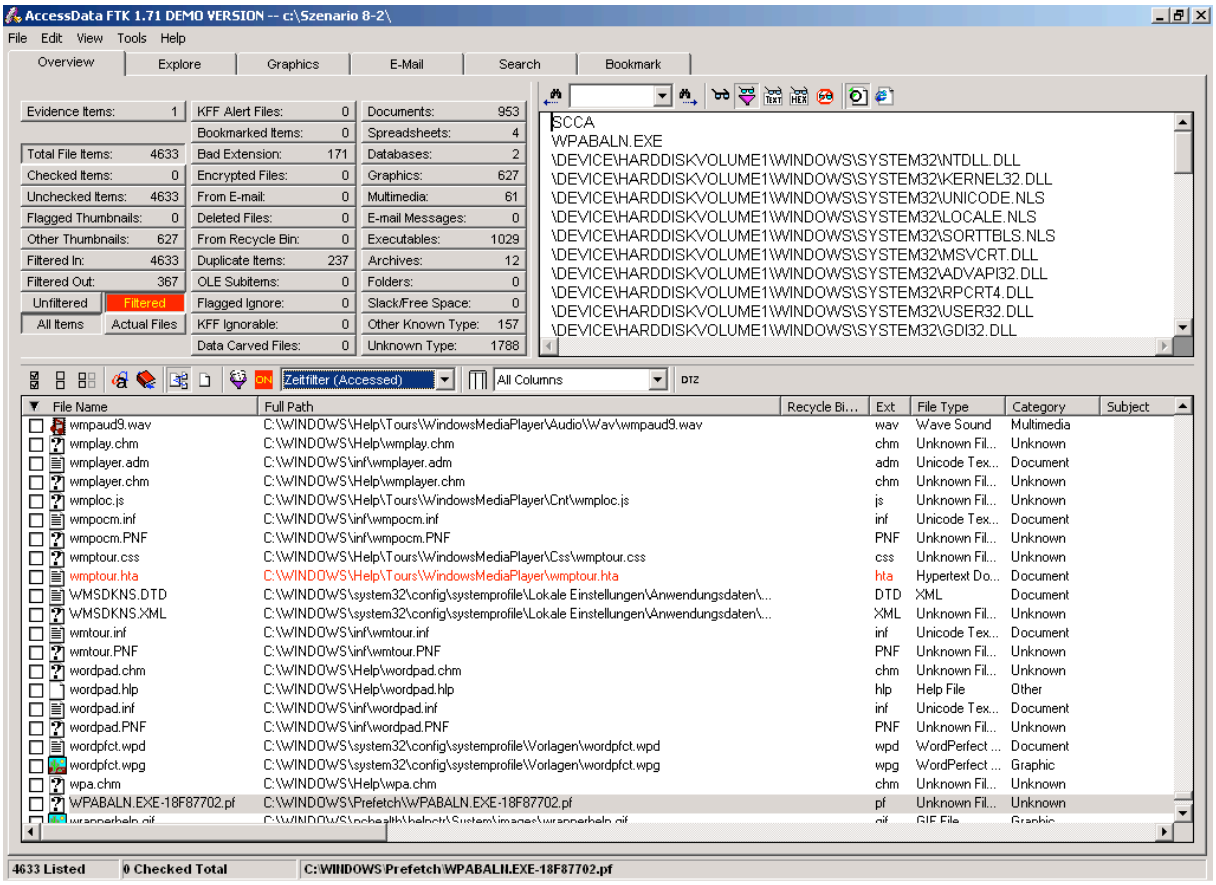
und so mit der Demo Version des `Forensic Toolkits` nicht alle Dateien angezeigt werden können. Als Alternative kann man die Analyse aufteilen, und jeweils nur ein paar Unterordner als Beweise verwenden.



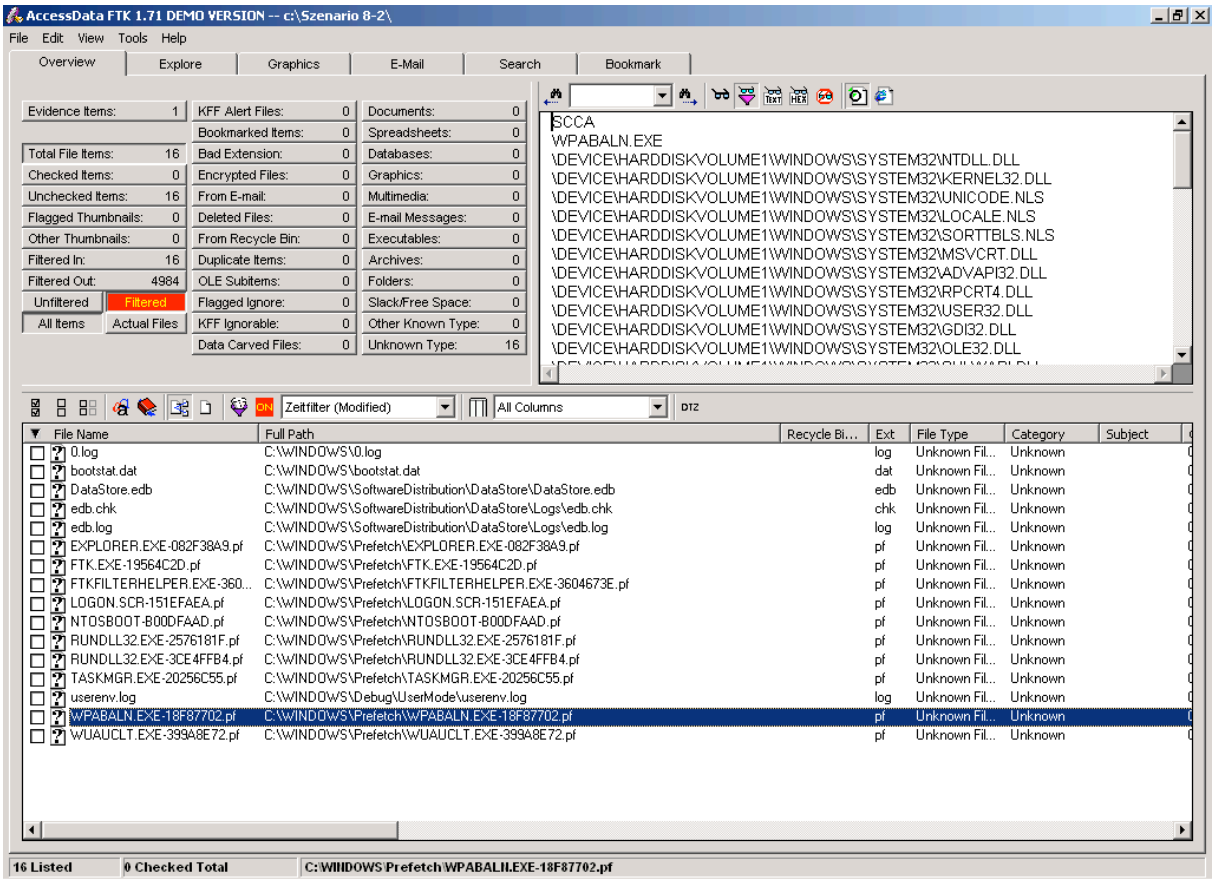
Ausschnitt aus der Auflistung der Dateien mit einem neuen Accessed-Zeitstempel des `Dokumente und Einstellungen` Ordners.



Auflistung der Dateien mit einem neuen Modified-Zeitstempel des `Dokumente und Einstellungen` Ordners.



Auschnitt aus der Auflistung der Dateien mit einem neuen Accessed-Zeitstempel des `Windows` Ordners.



Auflistung der Dateien mit einem neuen Modified-Zeitstempel des `Windows` Ordners.

7.2.9 Szenario 9: Ext3 in openSUSE 10.3

- **Übungsangabe**

Für dieses Szenario erhalten Sie ein Image im raw / dd Format. Hierbei handelt es sich diesmal nicht um eine Windows-Festplatte sondern um eine Linux-Festplatte. Diese enthält eine mit Ext3 formatierte Partition, welche als zusätzliche Datendisk verwendet wurde.

Diese Datendisk gehörte zu den Freigaben eines Samba Servers. Aus diesem Grund hatten auch mehrere Benutzer Lese- und Schreibrechte auf dieses Laufwerk. Einer dieser Benutzer löschte einige Dateien und Ordner dieser Freigabe. Da diese gelöschten Daten für einen anderen Benutzer von Bedeutung waren, und aus Versehen auf dieses Laufwerk kopiert wurden, besteht nun Ihre Aufgabe darin, dieses Laufwerk zu untersuchen. Dokumentieren Sie, wie auch bei den vorherigen Szenarien, welche gelöschten Dateien und Ordner gefunden und wiederhergestellt werden konnten.

Für Ihre Analyse und für das Erstellen eines Berichtes sollen Sie wieder ein Computer-Forensik-Programm verwenden.

- **Image Details**

Name	Szenario-9-openSUSE103-ext3.dd
Größe	210.051.072 Byte
MD5 Prüfsumme	F702E571153667B57F3B580BDDBE2D38

- **Forensisch interessante Dateien und Ordner**

In diesem Abschnitt werden alle gelöschten Dateien und Ordner angegeben, welche im Zuge der forensischen Untersuchung gefunden werden sollen.

Angegebene Dateien im Ordner gelöscht		
/Bilder		
Dateiname	**Größe in Byte**	**MD5 Prüfsumme**
Blick auf den Wildensee vom Rinnerkogel.JPG	293.902	46EAEE7075DFA2114E67C46969555C30
Image002.jpg	352.784	F0C0E3BEDB439F2F40795863B19DDA4C
Image003.jpg	401.615	8992CDB9537DB2AD06C7EF49EA7A1526
Image004.jpg	370.495	35A1B1303A61BF5BC92084DD500AA398

Angegebene Dateien im Ordner gelöscht		
/Dokumente/SDK-Docs-Tutorials		
Dateiname	**Größe in Byte**	**MD5 Prüfsumme**
j2ee-5_0-beta-doc-tutorial.zip	12.276.520	F193EC2247C1C76AAC2D44748AB60A2F
jwstutorial20.zip	2.427.310	2A9776EAF04FD078A12EF3B6134ED721

Ordner + Inhalt gelöscht		
/neue Dateien/neue Bilder		
Dateiname	**Größe in Bytes**	**MD5 Prüfsumme**
Image050.jpg	504.864	6C455C48DED3ABA6C275BA38D66F3835
Image051.jpg	460.033	4505C879A31416AF1BDD858F888A28DF
Image052-bearbeitet.jpg	1.224.896	CB6A72603C198555FD9AA52FF9C9A621
Image053.jpg	390.427	B4C8F57484C357B811C6FB56C4B001C6

Ordner + Inhalt gelöscht		
/neue Dateien/neue Dokument/ KV Requirements Engineering		
Dateiname	**Größe in Bytes**	**MD5 Prüfsumme**
RE-JKU-0.pdf	235.513	B0FCBE305124C4409D2106542D9660D2
RE-JKU-1a-RESpiral.pdf	107.838	9E8586F85D8FC7C3AC47D996B37CF702
RE-JKU-1b-WinWin.pdf	1.520.704	3DCDE14135D1FA40B5C63909CFA71D28

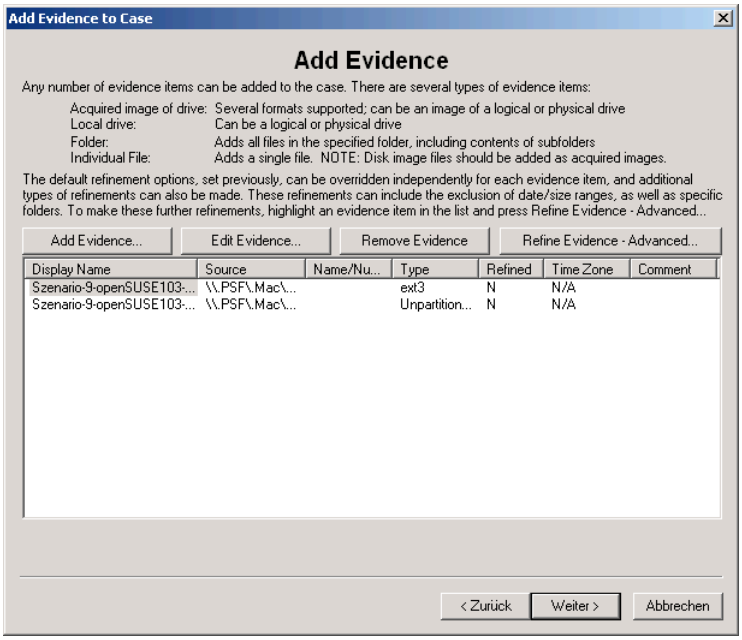
RE-JKU-1c-TypesOfRequirements.pdf	86.558	D17485EBCCF9D89F53817FA6ADD4BF76
RE-JKU-2a.pdf	724.578	C677BCD144EE0FF36BE14441F89D8C7E
RE-JKU-2b.pdf	743.825	9165A0729E4E680D8C25FC4F1F690BDC
RE-JKU-3.zip	4.455.068	17C2A71FC01492B3A501EC241729E622
RE-JKU-4.zip.zip	2.302.519	70C737D32C143DB937F484364C1C28EF
SeCSE@Linz.pdf	4.531.981	7C72152C7929905727C409DC5F777BB3
template.pdf	848.787	73D4F61BC3856485F7A4A01FE825098F

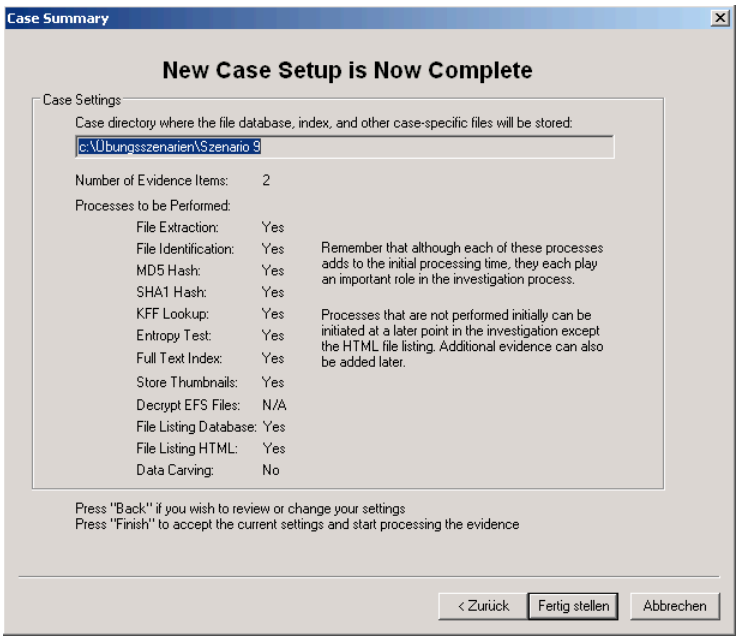
Ordner + Inhalt gelöscht		
/neue Dateien/neue Dokument/KV Requirements Engineering/RE-JKU-3		
Dateiname	Größe in Bytes	MD5 Prüfsumme
01 Volere-Template.pdf	889.310	85007F2EFA790520693CED29C45308FF
ASERegularPaper#7_ Neumueller_Christian.pdf	268.058	BAA71660DF59AE1DB9F7011D36B5698B
RAS-TUWIEN-3a.pdf	206.813	02CBBD0D8C0D6D050645F4EA1017CE2A
RAS-TUWIEN-3b.pdf	39.654	AD5F74C25692A535C6244FBEA6D73B97
RAS-TUWIEN-3c.pdf	1.210.831	A9DC94FC0F575E19DCCC842CA4B3FFD6
RAS-TUWIEN-3d.pdf	1.630.939	814B50F1A5519DEDE489FE6E04138A46
RAS-TUWIEN-3e.pdf	729.571	87AD030782034D2E5DC8264E7FA2EDD0

Ordner + Inhalt gelöscht		
/neue Dateien/neue Dokument/KV Requirements Engineering\RE-JKU-4.zip		
Dateiname	Größe in Bytes	MD5 Prüfsumme
ASE Halling Gruenbacher Biffl.pdf	325.764	9EDC3A571CF95EDD4C848CE94B9A0CC2
DetReadTechnDtCBR001021.doc	40.448	299E2C1DF3A451AC3CABEC8DA84CAA85

- **Forensische Untersuchung**

Da die Untersuchung des Images für dieses Szenario gleich wie in Szenario 2 erfolgt, wird für die Details dieser auf das Kapitel 7.2.2 (Szenario 2: FAT32 in Windows ME) verwiesen. Zusammenfassend hier noch die Auflistung der hinzugefügten Beweise.

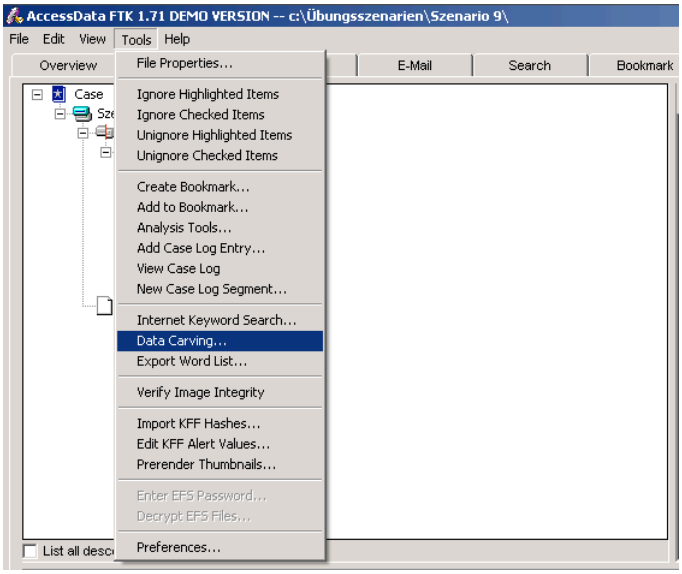




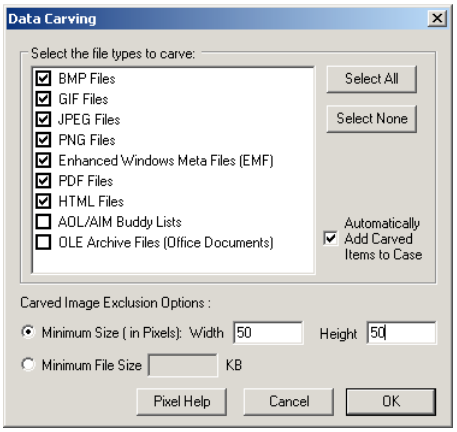
Nachdem alle Informationen angegeben wurden, kann über das Klicken auf den Button `Fertig stellen` mit der Analyse des Images begonnen werden.

Da bei der Untersuchung des Images keine gelöschten Daten gefunden wurden, wird die Funktion des `Data Carvings`, also das Suchen nach Daten anhand deren Signaturen, auf den freien Speicher angewandt.

Zu finden ist diese Funktion, welche bereits zuvor im Wizard aktiviert werden hätte können, im Menü Tools.



Bevor mit der Suche begonnen werden kann, muss noch angegeben werden, nach welchen Dateitypen gesucht werden soll. Gilt es auch Bilder zu suchen, so muss weiters eine minimale Größe dieser entweder in `kB` oder `Pixel` angegeben werden.



- **Ergebnisse der forensischen Untersuchung**

Von den angegebenen Dateien konnten auch mit Unterstützung des `Data Carvings` keine korrekt wiederhergestellt werden. Es konnten jedoch einige gelöschte Bilder und PDF-Dokumente mit mehr oder weniger Fehlern wiederhergestellt werden. Weiters wurde ebenfalls eine Menge eingebetteter Bilder der gelöschten PDF- und Zip-Dateien gefunden. Die Analyse des freien Speichers mit dem Programm `Foremost` ergab ebenfalls die selbe Anzahl an gefundenen Dateien.

Da die Liste der gefundenen Dateien 110 Elemente umfasst, wird sie hier nicht gesondert angeführt, sondern kann im generierten Bericht zu diesem Szenario angesehen werden. Die Analyse dieses Images mit dem `Autopsy Forensic Browser` ergibt zum Unterschied zwar die Namen der gelöschten Dateien bei denen der Ordner noch existiert, aber es konnte ebenfalls keine dieser Dateien wiederhergestellt werden.

- **Zusammenfassung**

Zusammenfassend lässt sich sagen, dass die gelöschten Daten zwar ein wenig überschrieben wurden, aber sich zum Größten Teil noch auf der Festplatte befinden. Und daher unter Zuhilfenahme der `Data Carving` zum Teil wiederhergestellt werden können. Schwierig ist jedoch die Zuweisung des korrekten Dateinamens, da er entweder gar nicht gefunden oder ohne Zeiger auf die Datei gefunden wird.

7.2.10 Szenario 10: Verschlüsselung und Komprimierung in NTFS

- **Übungsangabe**

Bei diesem Szenario geht es um erweiterte NTFS-Optionen. Sie erhalten für dieses Szenario ein raw / dd Image einer Festplatte, welche eine NTFS-Partition enthält, die als Datenpartition in einem Windows XP PC verwendet wurde.
Bei der Speicherung der Daten auf dieser Festplatte wurden die erweiterten NTFS-Funktionen für das Komprimieren und Verschlüsseln von Dateien und Ordnern verwendet.
Ihre Aufgabe besteht nun darin, mit einem Computer-Forensik-Programm zu versuchen, die Inhalte dieser Ordner und Daten anzuzeigen, weiters wurden einige dieser komprimierten oder verschlüsselten Dateien und Ordner gelöscht und Sie sollen versuchen diese zu finden und wenn möglich zu extrahieren.
Geben Sie in ihrem Bericht unter anderem an, ob es mit dem von Ihnen verwendeten Programm möglich ist, solche Dateien und Ordner zu lesen bzw. zu finden.

- **Image Details**

Name	`Szenario-10-winxp-NTFS.dd`
Größe	`388.104.192 Byte`
MD5 Prüfsumme	`D887414D60D8B901E183CA29A8D0E9BE`

- **Forensisch interessante Dateien und Ordner**

In diesem Abschnitt werden alle gelöschten Dateien und Ordner angegeben, welche im Zuge der forensischen Untersuchung gefunden werden sollen.
Bevor die gelöschten Dateien und Ordner angegeben werden zuerst eine Auflistung der Ordner die verschlüsselt oder komprimiert sind. Wobei die Kombination von Verschlüsselung und Komprimierung auf Dateisystemebene nicht möglich ist.

Verschlüsselte Ordner

```
Dokumente\KV Angewandte Systemtheorie – Kryptographie
neue Dokumente
neue Dokumente\KV Requirements Engineering
neue Dokumente\KV Sicherheitsaspekte in der Informatik
neue Dokumente\Netzwerkadministration
neue Dokumente\Systemtheorie 1
neue Dokumente\VL Pervasive Computing Infrastruktur
```

Komprimierter Ordner

```
Dokumente\Seminar - Netzwerke – Security
```

Gelöschte Dateien

Angegebene Dateien im Ordner gelöscht		
Dokumente\SDK-Docs-Tutorials		
Dateiname	Größe in Byte	MD5 Prüfsumme
j2ee-5_0-beta-doc-tutorial.zip	12.276.520	F193EC2247C1C76AAC2D44748AB60A2F
javadocs.zip	8.015.165	A2225057AF5C02208BA985E9DEE68399

Angegebene Dateien im Ordner gelöscht		
Dokumente\Seminar - Netzwerke – Security		
Dateiname	Größe in Byte	MD5 Prüfsumme
3.Vulnerability Analysis.pdf	380.177	751E164066D23341D0B10976F06B1530
5.Google Hacking.pdf	1.138.327	F497AD7B16F87D6C3DBC0F0695BF23D6

Angegebene Dateien im Ordner gelöscht		
Dokumente\Seminar - Netzwerke – Security\cg0705.mspx_files		
Dateiname	Größe in Byte	MD5 Prüfsumme
broker.js	4.571	3978DB0D26A384B5622DE03A87C87C6F
cableguy.gif	30.792	0E94B6586D8A5D58C143F471C483F376
cg07050.jpg	15.256	8129A1D9CA03F49A3E315559FC611040
ratings.htm	8.697	DDDD7A9B0F0011842925FCAEB35F1C71

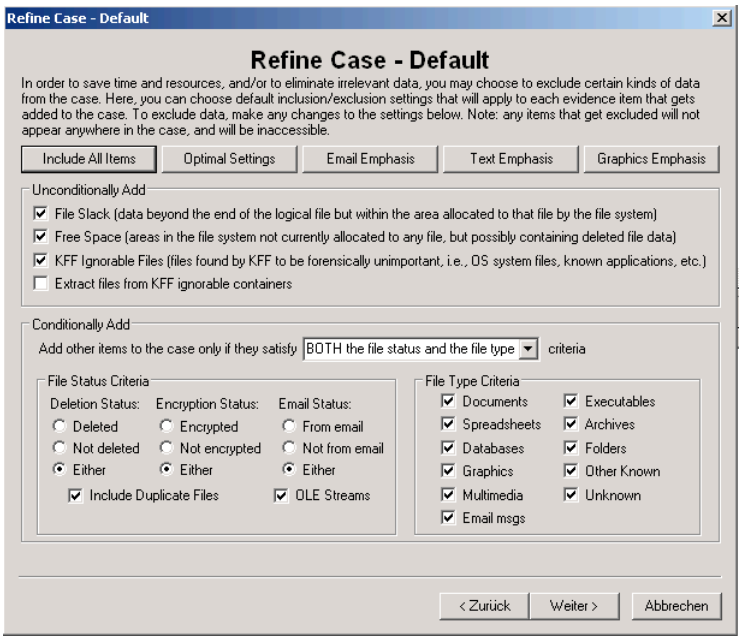
Ordner + Inhalt gelöscht		
Dokumente\Seminar - Netzwerke – Security\cg0705.mspx_files\ratings_data		
Dateiname	Größe in Byte	MD5 Prüfsumme
ContentRating.css	420	F02DC615EF1E9293C74330D6D59BB14D
rtg_email.gif	1.010	B201A23EC07D55866FAB17A62030774E
rtg_print.gif	574	101A0AC63030B36FCE0421D896940AEF
rtg_save.gif	567	7D20B965EA147E6DC2B580CE6AE71778

Verschlüsselter Ordner + Inahlt gelöscht		
neue Dokument\Netzwerkadministration		
Dateiname	Größe in Bytes	MD5 Prüfsumme
NetAdm_c_Server_versus_Workstation Linz.pdf	195598	8D287EEEB85FFAAE14305A19F4B7818C
NetAdm_d_Workstation Linz.pdf	1634160	E5692468A5026C9983EFD3B06B0A5E08
NetAdm_e_Installation Linz.pdf	476829	D483F6997FB21530ED8B40B88DCC7A51
NetAdm_f_MMC_Eventlog_Perf Linz.pdf	677811	B7DB9EB5A131EC19F9815C73C6E7C239
NetAdm_g_Diskadmin Linz.pdf	540072	7A7C0BCA6EE98C918E628B03EC08ECFA
NetAdm_h_UserManagement0 Linz.pdf	2231049	C845CD392F609AE3162ADAD9937A9972
NetAdm_i_NTFS Linz.pdf	1501695	18DB50EC8947445B97F980DB5398432D
NetAdm_j_Printing STP.pdf	750913	53C27451FE9D626A47C9D74E01D38CAD
NetAdm_k_Shares Linz.pdf	836034	A1C2B66A4FB8A345CAB4A1F2ED44F54D
NetAdm_l_ADS.pdf	897738	1BC450F278DBD5997ED97282C6555C2B
NetAdm_m_Mailserver.pdf	1458868	4CE25C8FA1CCAB5D0FC619328301B706
NetAdm_Mailserver.pdf	1458868	4CE25C8FA1CCAB5D0FC619328301B706
NetAdm_n_DHCP.pdf	1306896	5CC17C78F3E01238F02876A07620551C
NetAdm_o_DNS (Netzwerke).pdf	876533	4D1193778B6CB27124A23B6D998124D9
NetAdm_o_DNS.pdf	939640	943516FCB381974EE5472C0300478877
NetAdm_p_IIS.pdf	739469	D644187B044F6AC09247D0F66B5BEE49
NetAdm_q_ActiveContent.pdf	1502990	83CA66ABBF3EF0A2E428F48845F2534A
NetAdm_r_SNMP.pdf	1081460	B122DE8ACDED64012264FDC23A4C27F8

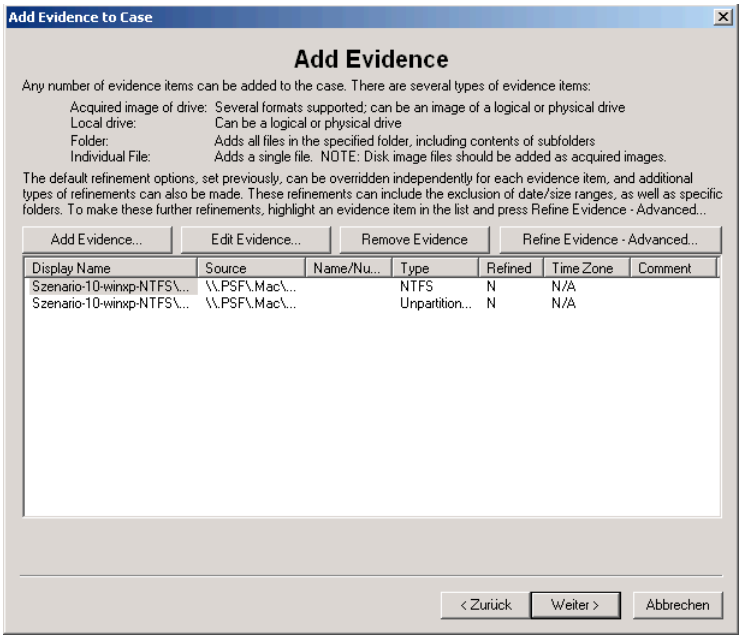
- **Forensische Untersuchung**

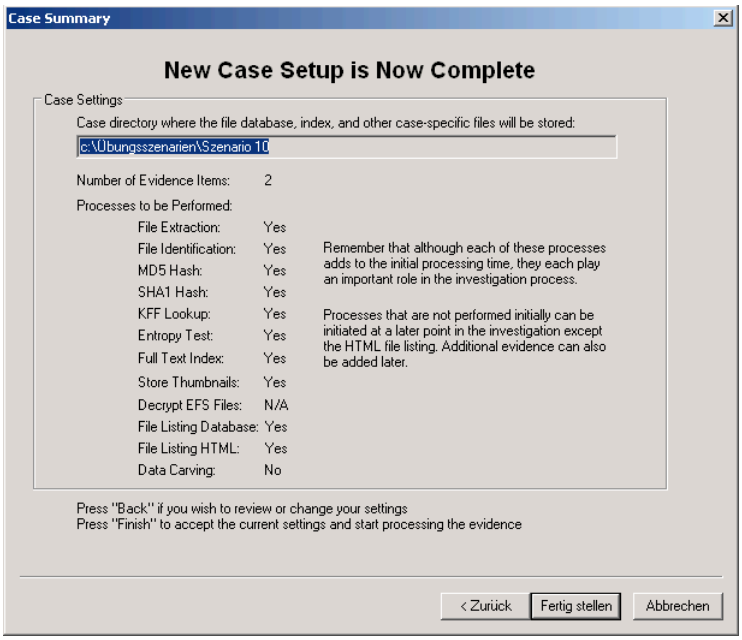
Die Untersuchung dieses Images erfolgt gleich wie die Untersuchung in Szenario 2, daher wird für die Details auf das Kapitel 7.2.2 (Szenario 2: FAT32 in Windows MESzenario 2: FAT32 in Windows ME) verwiesen.

Ein kleiner Unterschied zum Szenario 2 ist, dass hier die Suche nicht eingeschränkt wird.



Zusammenfassend hier noch die Auflistung der hinzugefügten Beweise.

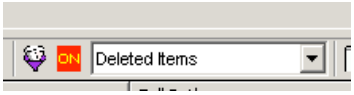




Nachdem alle Informationen angegeben wurden, kann über das Klicken auf den Button `Fertig stellen` mit der Analyse des Images begonnen werden.

- **Ergebnisse der forensischen Untersuchung**

Über das Einsetzen des Dateifilters wird das Auffinden der gelöschten Dateien einfacher, denn mit dem Verwenden der vorhandenen Einstellung für das Anzeigen gelöschter Dateien werden diese einfach angezeigt. Diese vorgegebenen Einstellungen können über das Drop-Down Menü neben dem `ON` Symbol des folgenden Screenshots ausgewählt werden. Soll der Dateifilter angepasst werden, so kann dieser über das Klicken auf das rechte Symbol angepasst werden.



Aktiviert wird der Dateifilter über das Klicken auf den Button `Filtered` auf der Übersichtsseite (`Overview`).

Die Aktivierung des Buttons `Actual Files` bewirkt, dass nur die vollständigen Dateien, nicht aber die eingebetteten Dateien, wie der Inhalt von Zip-Dateien, E-Mails oder OLE-Streams angezeigt werden.

Liste der gefundenen gelöschten Dateien und Ordner

Im Ordner `\Dokumente\SDK-Docs-Tutorials\` gefundene Datei

Dateiname	Größe in Byte	MD5 Prüfsumme	Fehlerfrei
j2ee-5_0-beta-doc-tutorial.zip	12.276.520	F193EC2247C1C76AAC2D44748AB60A2F	ja
javadocs.zip	8.015.165	A2225057AF5C02208BA985E9DEE68399	ja

Im Ordner `\Dokumente\Seminar - Netzwerke – Security\` gefundene Dateien:

Dateiname	Größe in Byte	MD5 Prüfsumme	Fehlerfrei
3.Vulnerability Analysis.pdf	380.177	751E164066D23341D0B10976F06B1530	ja
5.Google Hack-ing.pdf	1.138.327	F497AD7B16F87D6C3DBC0F0695BF23D6	ja

Im Ordner `\Dokumente\Seminar - Netzwerke – Security\cg0705.mspx_files\` gefundene Dateien:

Dateiname	Größe in Byte	MD5 Prüfsumme	Fehlerfrei
broker.js	4.571	3978DB0D26A384B5622DE03A87C87C6F	ja
cableguy.gif	30.792	0E94B6586D8A5D58C143F471C483F376	ja
ratings.htm	8.697	DDDD7A9B0F0011842925FCAEB35F1C71	ja

Der gelöschte Ordner `\Dokumente\Seminar - Netzwerke – Security\cg0705.mspx_file\ratings_data\` und folgende gelöschte Dateien in diesem Ordner wurden gefunden:

Dateiname	Größe in Byte	MD5 Prüfsumme	Fehlerfrei
$I30	4.096	DD10E554EF4545ED387892F954BE2CBC	*
ContentRating.css	420	F02DC615EF1E9293C74330D6D59BB14D	ja
rtg_email.gif	1.010	B201A23EC07D55866FAB17A62030774E	ja
rtg_print.gif	574	101A0AC63030B36FCE0421D896940AEF	ja
rtg_save.gif	567	7D20B965EA147E6DC2B580CE6AE71778	ja

* Angabe nicht möglich, da die Datei eine Indexdatei des Dateisystems ist und vor dem Löschen nicht sichtbar war. Genau genommen handelt es sich hier um das $INDEX_ALLOCATION Attribut des Ordners `\ratings_data`.

Im Ordner `\System Volume Information\` gefundene Datei:

Dateiname	Größe in Byte	MD5 Prüfsumme	Fehlerfrei
EFS0.LOG	1.536	2BE308D03EEB4927CC5630CF3235A8D0	*

* Angabe nicht möglich, da es sich hierbei um eine Logdatei des NTFS-Verschlüsselungsverfahren handelt.

Im Ordner `\neue Dokumente\VL Pervasive Computing Infrastruktur\` gefundene Datei:

Dateiname	Größe in Byte	MD5 Prüfsumme	Fehlerfrei
EFS0.TMP	672.114	BD09C9378C6D6F673B80A09F89F548AB	*

* Ein Vergleich der Prüfsumme mit denen aus der Prüfsummensammlung ergibt, dass es sich hierbei um die Datei „Vorbesprechung2006.pdf“ handelt. Der Ursprung dieser temporären Datei liegt im Verschlüsselungsverfahren von NTFS.

Der gelöschte Ordner `neue Dokument\Netzwerkadministration` wurde mit folgendem Inhalt gefunden.

Dateiname	Größe in Bytes	MD5 Prüfsumme	Fehlerfrei
$EFS	576	E81BB2C9AE17DCBA4E7868321AE52762	*
$I30	8192	A19B84C8C72F4FB2F7D3F82F6A68BFA9	*
NetAdm_c_Server_versus_Workstation Linz.pdf	195598	1141195A80D9934C0A15E4E2834A00DD	+
NetAdm_d_Workstation Linz.pdf	1634160	3818D743F93D6C7CE325B6ADEC61D21A	+
NetAdm_e_Installation Linz.pdf	476829	36F096E122A40420B114547DC7E7F556	+
NetAdm_f_MMC_Eventlog_Perf Linz.pdf	677811	448B4EBB625BF1D73DCA7460974A08D	+
NetAdm_g_Diskadmin Linz.pdf	540072	446012476CB7FCE2FFAEB93CEDBFF8F1	+
NetAdm_h_UserManagement0 Linz.pdf	2231049	5A1BA0A64677EB2A670EE83233EFFB15	+
NetAdm_i_NTFS Linz.pdf	1501695	10BBEFFA4A8AFD3FF6B8BAB828C10B66	+
NetAdm_j_Printing STP.pdf	750913	F84BAAB364E7A4A6D0D11A8910CD323C	+
NetAdm_k_Shares Linz.pdf	836034	FF02EAC75A40D5467964924A512DC551	+
NetAdm_l_ADS.pdf	897738	F0FDF3AC918C7A6ECAD200B719219A42	+
NetAdm_m_Mailserver.pdf	1458868	FE4BF93DC141640B00ADE279A9F76990	+
NetAdm_Mailserver.pdf	1458868	77531B7B3AF5E2F82E11E3B30BF82D4E	+
NetAdm_n_DHCP.pdf	1306896	937048DB06BEA64F903669AF1B182AA3	+
NetAdm_o_DNS(Netzwerke).pdf	876533	B35CC9C9949F24FDD11C8E453C5A1101	+

NetAdm_o_DNS.pdf	939640	70F5ADBE4AEBF0E448185FF36D65277F	+
NetAdm_p_IIS.pdf	739469	13A9814ACDE4E45F8F57B72930833447	+
NetAdm_q_ActiveContent.pdf	1502990	E702DD037F056C5A003976AB7FCF02DE	+
NetAdm_r_SNMP.pdf	1081460	D2400BFECD74B73D61A4C4125DF74E9B	+

* Angabe nicht möglich, da es sich hier um Dateien des NTFS-Verschlüsselungsverfahrens handelt.

+ Angabe nicht möglich, da es sich hier um verschlüsselte Dateien handelt die nicht entschlüsselt und mit den Original-Dateien verglichern werden können.

- **Zusammenfassung**

Die Analyse der verschlüsselten Ordner und Dateien war nur so weit möglich, als dass die Datei- und Ordner-Namen gefunden und aufgelistet werden konnten. Die gelöschten Dateien konnten alle wiederhergestellt werden, wobei bei den verschlüsselten der Inhalt nicht analysiert werden konnte, da die Verschlüsselung nicht ohne Hilfsmittel gebrochen werden kann.
Mit dem komprimierten Ordner gab es hingegen keine Probleme, das heißt, es konnten alle enthaltenen Dateien gefunden und wiederhergestellt werden.

7.2.11 Szenario 11: Recovery einer Formatierung

- **Übungsangabe**

Mit diesem Szenario sollen weitere Besonderheiten von NTFS behandelt werden. Sie erhalten wieder ein raw / dd Image einer Festplatte, welche aus einer NTFS-Partition besteht. Die Besonderheit bei dieser Partition ist, dass sie direkt vor dem Erstellen des Images neu mit NTFS formatiert wurde, und noch keine Daten enthält. Sie können jedoch davon ausgehen, dass für das Formatieren nur die Methode Quickformat verwendet wurde. Aus diesem Grund und unter Kenntnis darüber, wie NTFS organisiert ist, sollte es möglich sein, noch vorhandene Daten, welche sich vor der Formatierung auf der Festplatte befunden haben, zu finden und möglicherweise zu extrahieren.
Geben Sie in Ihrem Bericht an, welche Daten Sie gefunden haben und welche Sie wiederherstellen konnten.

- **Image Details**

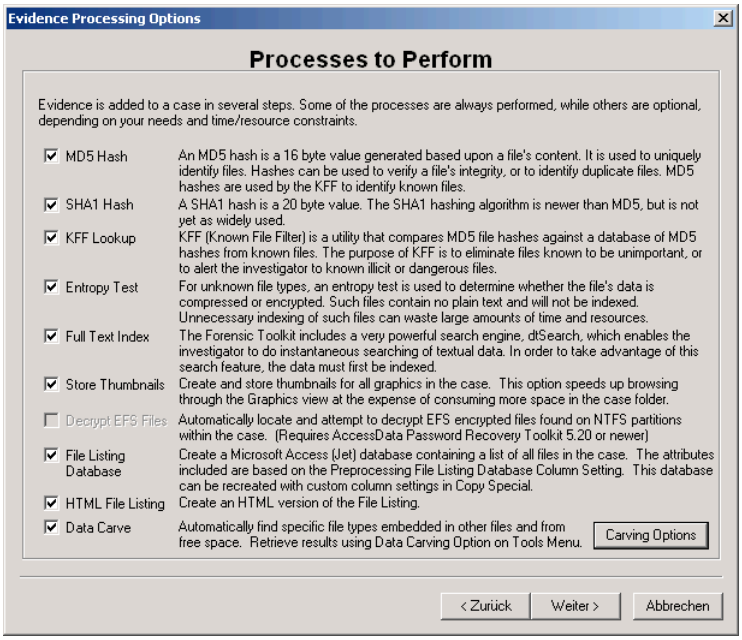
Name	`Szenario-11-winxp-NTFS.dd`
Größe	`367.460.352 Byte`
MD5 Prüfsumme	`5A506CE631825CF2D1B9A5D52D5138E9`

- **Forensisch interessante Dateien und Ordner**

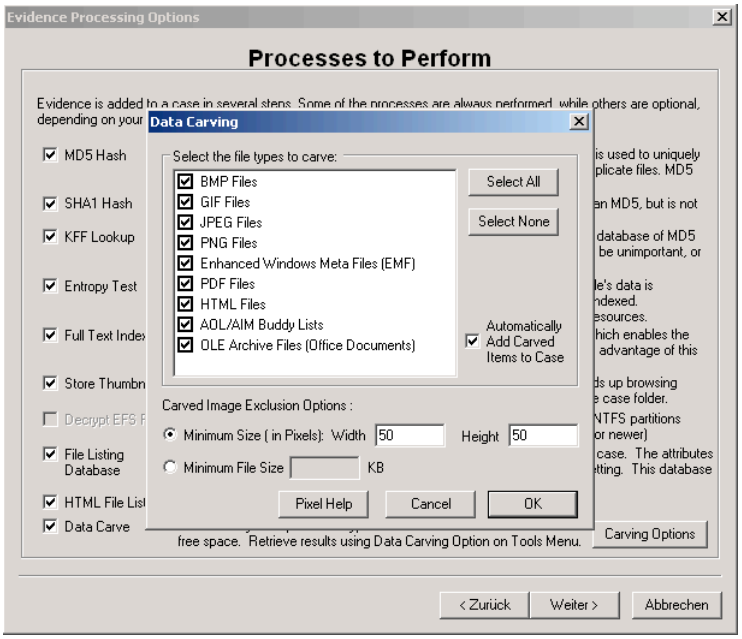
Insgesammt beinhaltete die Festplatte vor der Formatierung 290 vom Benutzer angelegt Dateien.

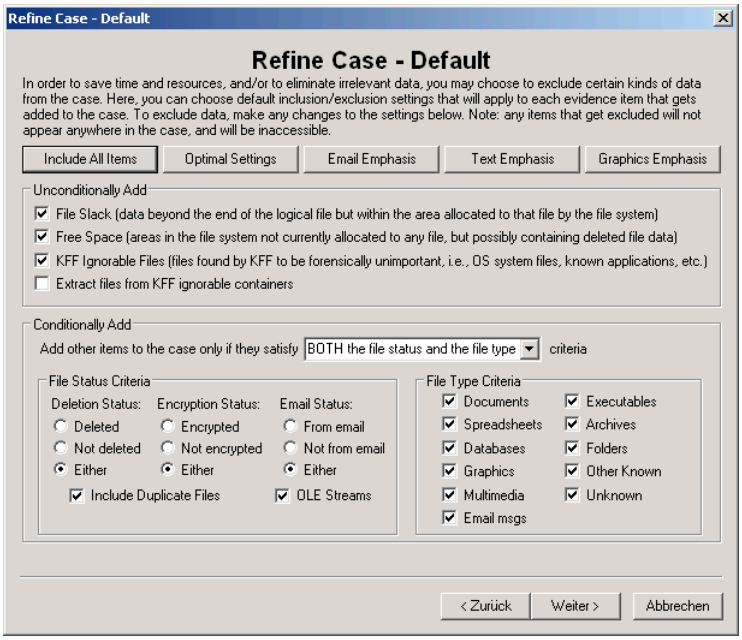
- **Forensische Untersuchung**

Die ersten Schritte der Untersuchung dieses Images erfolgen gleich wie die in Kapitel 7.2.2 (Szenario 2: FAT32 in Windows MESzenario 2: FAT32 in Windows ME).
Nach der Angabe der Logging-Optionen werden jene Funktionen ausgewählt, welche nach Abschluss des Wizards durchgeführt werden sollen. Anders als bei den vorherigen Szenarien wird für dieses Szenario die Funktion des `Data Carvings` verwendet, um die Dateien im freien Speicher zu finden. Nachdem der Inhalt des Images nicht bekannt ist, werden beim Data Carving alle möglichen Dateitypen angegeben. Die einzige sinnvolle Einschränkung ist das Angeben einer Mindestgröße für Bilder.

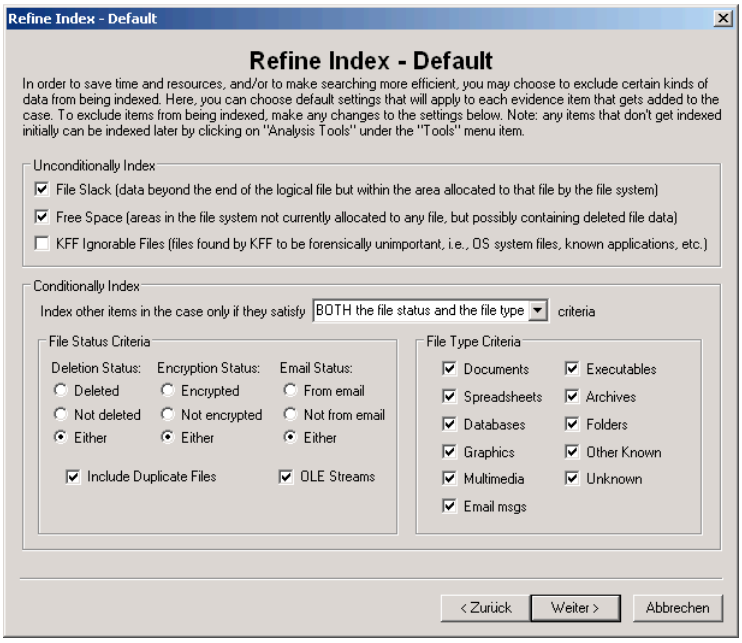


Angabe der Optionen für das `Data Carving` welche über den Button `Carving Options` erreichbar sind.

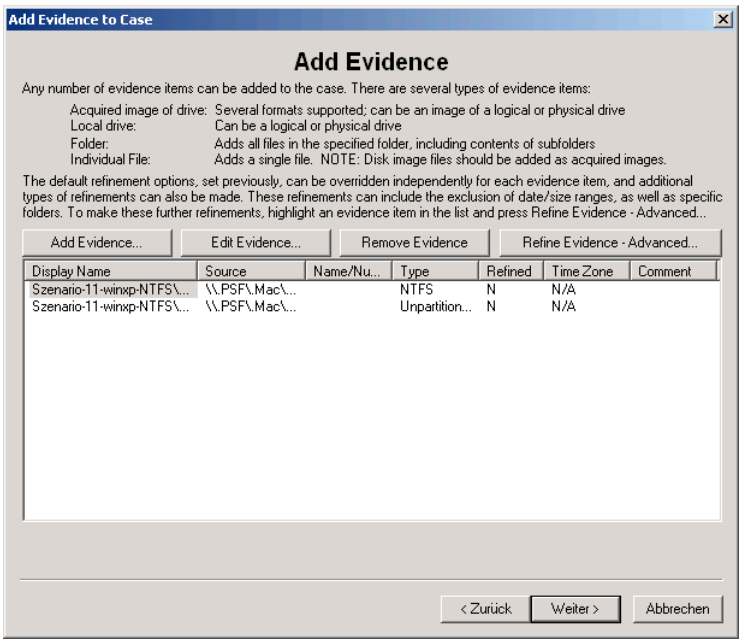


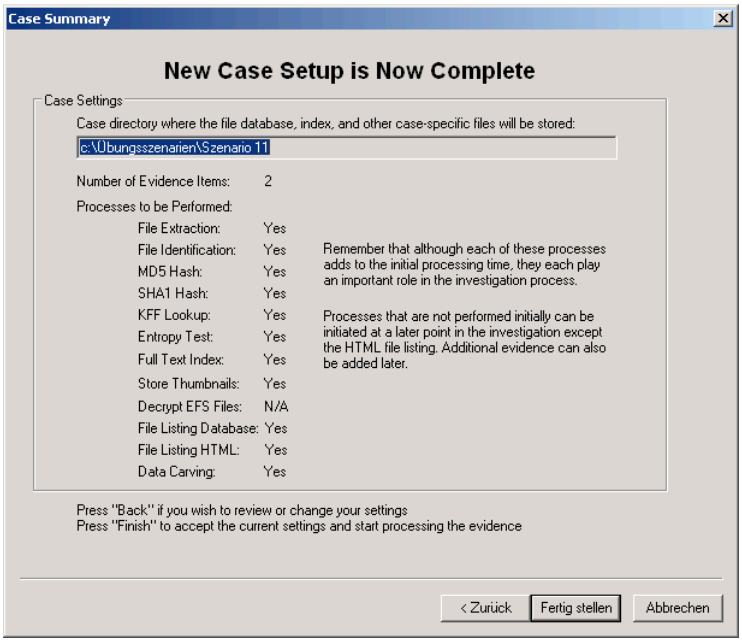


In diesem Schritt kann die Suche auf bestimmte Dateitypen und den Status einer Datei eingeschränkt werden. Die hier gezeigten Einstellungen sind für die in diesem Szenario zu suchenden Dateien optimiert.



Zusammenfassend hier noch die Auflistung der hinzugefügten Beweise.





Nachdem alle Informationen angegeben wurden, kann über das Klicken auf den Button `Fertig stellen` mit der Analyse des Images begonnen werden.

- **Ergebnisse der forensischen Untersuchung**

Da mit dem Formatieren der Festplatte die MFT neu geschrieben wurde, konnten keine originalen Dateinamen wiederhergestellt werden.

Die Analyse fand insgesammt 425 Dateien, welche hier nicht gesondert angeführt werden. Da die Dateien mithilfe der Data Carving Funktion gefunden wurden, kann nicht genau gesagt werden, ob es sich hierbei um alle zu findenden Dateien handelt. Bei diesen Dateien sind ebenfalls die in den PDF- und Zip-Dateien enthaltenen Dateien, welche gefunden wurden, in-

kludiert. Eine gesamte Liste der gefundenen Dateien kann dem generierten Bericht entnommen werden.

- **Zusammenfassung**

Von den gelöschten Dateien wurden zwar einige gefunden, der Vergleich der Prüfsummen ergibt jedoch, dass diese nicht korrekt wiederhergestellt werden können. Weiters konnte auch kein Dateiname rekonstruiert werden.

7.2.12 Szenario 12: Recovery beschädigter Sektoren

- **Übungsangabe**

Dieses Szenario ist im Vergleich zu den anderen Szenarien nicht mehr trivial, da es einiges an Wissen über die physikalische Speicherung von Daten auf der Festplatte und Übung im Umgang mit Hexeditoren voraussetzt.
Das zu bearbeitende Festplattenimage stammt von einer Festplatte, bei welcher einige Sektoren beschädigt wurden. Unter diesen Sektoren befinden sich auch der Sektor 0 und der Sektor 63. Es kann davon ausgegangen werden, dass diese Festplatte mit NTFS formatiert wurde.
Bevor Sie sich auf die Suche nach gelöschten und verdächtigen Daten machen können, müssen Sie zuerst versuchen das Image mit Hilfe eines Hexeditors so zu bearbeiten, dass Sie keine ermittlungsrelevanten Daten zerstören, aber es anschließend möglich ist, das Image mit einem Computer-Forensik-Programm zu öffnen und zu analysieren.
Nachdem Sie das Image erfolgreich mit einem Forensik-Programm geöffnet haben, können Sie noch nach möglicherweise gelöschten Daten suchen.
Für die Dokumentation sollen Sie genau angeben, wie Sie bei der Wiederherstellung der Daten vorgegangen sind.

- **Image Details**

Name	`Szenario-12-winxp-NTFS.dd`
Größe	`367.460.352 Byte`
MD5 Prüfsumme	`5C71F436D8B8EF7E02D50018319C7E5A`

- **Forensisch interessante Dateien und Ordner**

In diesem Abschnitt werden alle gelöschten Dateien und Ordner angegeben, welche im Zuge der forensischen Untersuchung gefunden werden sollen.

Ordner, in welchem die Dateien enthalten sind			
\Dokumente\Seminar - Netzwerke - Security			
Dateiname	Größe in Byte	MD5 Prüfsumme	ADS von Datei
Screen1-Detail.bmp	791.898	2B9C6C5B91EC8C42AC8BE2F4C4BD59F1	Screen1.bmp
Screen2-Detail.bmp	605.238	DC6649B66B450D24D7E4BC53642FF98D	Screen2.bmp

Ordner, in welchem die Dateien enthalten sind			
\Dokumente\SDK-Docs-Tutorials			
Dateiname	Größe in Byte	MD5 Prüfsumme	ADS von Datei
jwstutori-al20.zip	2.427.310	2A9776EAF04FD078A12EF3B6134ED721	jwsdp-2_0-ant-docs.zip

Ordner, in welchem die Dateien enthalten sind			
\Bilder			
Dateiname	Größe in Byte	MD5 Prüfsumme	ADS von Datei
Image002.jpg	352.784	F0C0E3BEDB439F2F40795863B19DDA4C	Beispielbilder.lnk

Ordner, in welchem die Dateien enthalten sind			
\Dokumente\Seminar - Netzwerke - Security			
Dateiname	Größe in Byte	MD5 Prüfsumme	ADS von Ordner
2.Networks.pdf	376.297	EFB288CF96E3F492C77187DEFE526509	Cisco

Angegebene Dateien im Ordner gelöscht			
\Dokumente\KV Angewandte Systemtheorie - Kryptographie			
Dateiname	Größe in Bytes	MD5 Prüfsumme	Bemerkung
06 - Signatu-res.pdf	511.361	3F2C6F4B2F9AFDC95F4CFECFB246FED5	
Mitschrift 29.11.06.doc	48.128	06547E9EAA747812A5CD9B0C41910FA5	ADS

Angegebene Datei im Ordner gelöscht			
\Bilder			
Dateiname	Größe in Bytes	MD5 Prüfsumme	Bemerkung
Image040.jpg	562.769	0E7AC1684009838B5085A71C2B5EA49D	
Image039.jpg	492.559	C37B868A47B7EDED2BDEC6365525C295	ADS

Weiters wurden folgende Dateien gelöscht.

Dateien im Ordner gelöscht		
\Dokumente\KV Angewandte Systemtheorie - Kryptographie		
Dateiname	Größe in Bytes	MD5 Prüfsumme
05 - AsymetricCrypto.pdf	327.872	1A6A5AC65DDCCE99868E22B51F8E3339
07 - ECC.pdf	153.306	C5508A72297CB052F120E3B1D7068C79
08 - PGP.pdf	668.244	63E1AA9B564EAD28892261282EA42AC6
09 - ssl.pdf	406.166	3435C6244E44C36C35CB3485501C4B5C
10 - SecInternetPay-ment.pdf	1.413.042	4CE931E1D272F5676327A66A86D91026
HalloLeute.pdf	191.566	4AC54DAD2C20C9E48DFC7230B8A9705F

Ordner, in welchem die Dateien enthalten sind		
\Dokumente\SDK-Docs-Tutorials		
Dateiname	Größe in Byte	MD5 Prüfsumme
j2ee-5_0-beta-doc-tutori-al.zip	12.276.520	F193EC2247C1C76AAC2D44748AB60A2F

Ordner, in welchem die Dateien enthalten sind		
\Dokumente\Seminar - Netzwerke – Security\cg0705.mspx_files		
Dateiname	Größe in Bytes	MD5 Prüfsumme
ADSAdClient31.htm	868	FC90E1EA430AD61272382AD33D915ECA
arrow_px_down.gif	53	A50A5A7BE6297D534E0A9559C583C217
arrow_px_up.gif	53	FBAE595C25E4EB510CE17F4EA17C8760
arrowLTR.gif	821	00BA9FF02078EE89D912B07B8E7D3838
broker.js	4.571	3978DB0D26A384B5622DE03A87C87C6F
cableguy.gif	30.792	0E94B6586D8A5D58C143F471C483F376

Ordner + Inhalt gelöscht		
\Dokumente\Seminar - Netzwerke – Security\cg0705.mspx_files\ratings_data		
Dateiname	Größe in Byte	MD5 Prüfsumme
ContentRating.css	420	F02DC615EF1E9293C74330D6D59BB14D
rtg_email.gif	1.010	B201A23EC07D55866FAB17A62030774E
rtg_print.gif	574	101A0AC63030B36FCE0421D896940AEF
rtg_save.gif	567	7D20B965EA147E6DC2B580CE6AE71778

- **Forensische Untersuchung**

Reparieren des Images

1. Die Analyse der Partitionstabelle mit dem Programm `mmls` ergibt, dass der Startsektor der ersten Partition untypischerweise Sektor 64 ist und dass kein Typ für das Dateisystem gesetzt ist.

```
DOS Partition Table
Offset Sector: 0
Units are in 512-byte sectors

     Slot    Start        End          Length       Description
00:  -----   0000000000   0000000000   0000000001   Primary Table (#0)
01:  -----   0000000001   0000000063   0000000063   Unallocated
02:  00:00   0000000064   0000706860   0000706797   Empty (0x00)
03:  -----   0000706861   0000717695   0000010835   Unallocated
```

2. Korrigieren der Partitionstabelle: Die Partitionstabelle einer MBR-Festplatte befindet sich in den letzten 66 Bytes des ersten Sektors, wobei jeder Eintrag 16 Bytes lange ist und die letzten beiden Bytes das Ende markieren. Da diese Festplatte nur eine Partition besitzt, sind nur die Bytes 446-461 von Interesse.

```
000000416  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
000000432  00 00 00 00 00 2C 44 63 00 00 06 22 00 00 00 01   .....,Dc..."....
000000448  01 00 00 FE 3F 2B 40 00 00 00 ED C8 0A 00 00 00   ....?+@.........
000000464  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
000000480  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
000000496  00 00 00 00 00 00 00 00 00 00 00 00 00 00 55 AA   ..............U.
```

   Zuerst wird der Typ der Partition, welcher im Byte 4 dieses Eintrags angegeben wird, von 0x00 auf 0x07, der Wert für NTFS, gesetzt.

   Vorher: `000000448  01 00 00 FE 3F 2B 40 00 00 00 ED C8 0A 00 00 00  ....?+@.........`

   Nachher: `000000448  01 00 07 FE 3F 2B 40 00 00 00 ED C8 0A 00 00 00  ....?+@.........`

   Anschließend muss noch der Startsektor, welcher im Normalfall der Sektor 63 für die erste Partition ist, richtig angegeben werden. Für diesen Zweck müssen die Bytes 8-11 dieses Eintrages auf den Wert 63, in Hex 0x3f gesetzt werden. Wobei hier zu beachten ist, dass die Daten im Little Endian Format gespeichert sind.

   Vorher: `000000448  01 00 07 FE 3F 2B 40 00 00 00 ED C8 0A 00 00 00  ....?+@.........`

   Nachher: `000000448  01 00 07 FE 3F 2B 3F 00 00 00 ED C8 0A 00 00 00  ....?+?.........`

3. Suchen nach dem Bootsektor: Der Bootsektor der ersten Partition sollte, wie in der Partitionstabelle angegeben, der Sektor 63 sein. Damit dieser gefunden werden kann, gibt es zwei Möglichkeiten: Nach dem OEM-String `NTFS` suchen, oder, unter Verwendung des

Wissens über die Größe des Sektors (512 Bytes), den Offset berechnen. `512 x 63 = 32256.`

```
000032256  EB 52 90 4E 54 46 53 20 20 20 20 00 02 01 00 00   .R.NTFS    .....
000032272  00 00 00 00 00 F8 00 00 3F 00 FF 00 3F 00 00 00   ........?...?...
000032288  00 00 00 00 80 00 80 00 EC C8 0A 00 00 00 00 00   ................
000032304  4F 98 03 00 00 00 00 00 76 64 05 00 00 00 00 00   O.......vd......
000032320  02 00 00 00 08 00 00 00 C1 E5 76 3C 0B 77 3C D6   ..........v<.w<.
000032336  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
000032352  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
000032368  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
000032384  08 CD 13 73 05 B9 FF FF 8A F1 66 0F B6 C6 40 00   ...s......f...@.
000032400  00 00 00 00 00 00 00 00 00 CD C0 ED 06 41 66 0F   .............Af.
000032416  B7 C9 66 00 00 00 00 00 00 00 00 00 00 AA 55 8A   ..f...........U.
000032432  16 24 00 CD 13 72 0F 81 FB 55 AA 75 09 F6 C1 01   .$...r...U.u....
000032448  74 04 FE 06 14 00 C3 66 60 1E 06 66 A1 10 00 66   t......f`..f...f
000032464  03 06 1C 00 66 3B 06 20 00 0F 82 3A 00 1E 66 6A   ....f;. ...:..fj
000032480  00 66 00 00 00 00 00 00 00 00 00 00 00 00 00 00   .f..............
000032496  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
000032512  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
000032528  66 58 66 58 1F EB 2D 66 33 D2 66 0F B7 0E 18 00   fXfX..-f3.f.....
000032544  66 F7 F1 FE C2 8A CA 66 8B D0 66 C1 EA 10 F7 36   f......f..f....6
000032560  1A 00 00 00 00 00 00 00 00 00 00 00 00 00 CC B8   ................
000032576  01 02 CD 13 0F 82 19 00 8C C0 05 20 00 8E C0 66   ........... ...f
000032592  FF 06 00 00 00 00 00 00 00 00 00 00 07 1F 66 61   ..............fa
000032608  C3 A0 F8 01 E8 09 00 A0 FB 01 E8 03 00 FB EB FE   ................
000032624  B4 01 8B F0 AC 3C 00 74 09 B4 0E BB 07 00 CD 10   .....<.t........
000032640  00 00 00 00 00 46 65 68 6C 65 72 20 62 65 69 6D   .....Fehler beim
000032656  20 4C 65 73 65 00 00 00 65 73 20 44 61 74 65 6E    Lese...es Daten
000032672  74 72 84 67 65 72 00 00 00 00 00 4C 44 52 20   tr.ger......LDR
000032688  66 65 68 6C 74 00 0D 00 00 00 4C 44 52 20 69 73   fehlt.....LDR is
000032704  74 20 6B 6F 6D 70 72 00 00 00 65 72 74 00 0D 0A   t kompr...ert...
000032720  00 00 00 00 00 00 72 74 20 6D 00 00 00 53 74 72   ......rt m...Str
000032736  67 2B 00 00 00 00 45 6E 74 66 0D 0A 00 00 00 00   g+....Entf......
000032752  00 00 00 00 00 00 00 00 00 00 00 00 00 00 55 AA   ..............U.
```

4. Backupkopie des Bootsektors suchen: Aufgrund der Beschädigung am Bootsektor im Bereich des Bootcodes, ist ein Starten von der Festplatte nach dem Korrigieren der Partitionstabelle nicht möglich. Aus diesem Grund soll eine Reparatur des Bootsektors versucht werden. Mit dem Wissen, dass bei einer mit NTFS formatierten Partition am Ende des Dateisystems eine Backupkopie des Bootsektors gespeichert wird, kann der beschädigte Bootsektor leicht wiederhergestellt werden. Mit der Information über die Größe der Partition aus der Partitionstabelle kann die Position dieser Kopie leicht errechnet werden. Dafür werden folgende Daten benötigt:
   - Beginn der Partition: Sektor 63 bzw. Offset 32256
   - Größe der Partition: 706797 Sektoren, Bytes 12-15 eines Eintrags in der Partitionstabelle

   Daraus errechnet sich der Offset wie folgt:

   `(706797 - 1) x 512 + 32256 = 361911808`

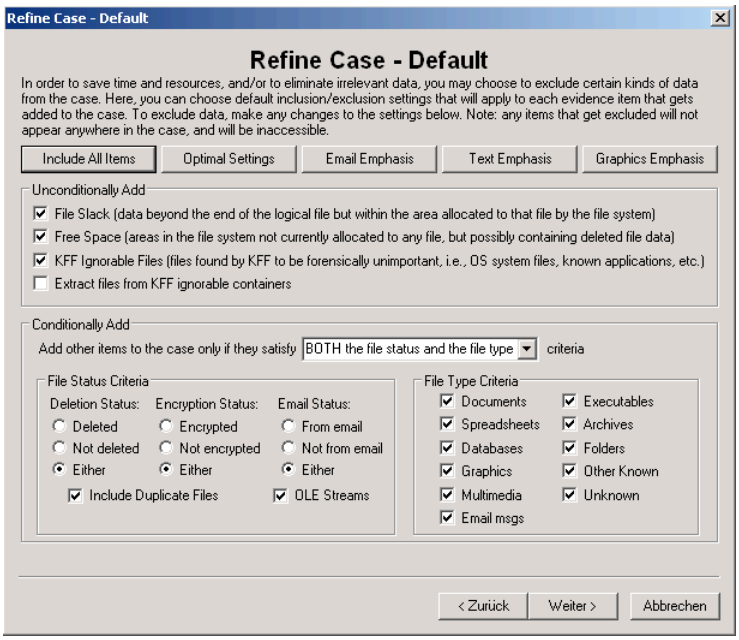
```
361911808  EB 52 90 4E 54 46 53 20 20 20 20 00 02 01 00 00   .R.NTFS    .....
361911824  00 00 00 00 00 F8 00 00 3F 00 FF 00 3F 00 00 00   ........?...?...
361911840  00 00 00 00 80 00 80 00 EC C8 0A 00 00 00 00 00   ................
361911856  4F 98 03 00 00 00 00 00 76 64 05 00 00 00 00 00   O.......vd......
361911872  02 00 00 00 08 00 00 00 C1 E5 76 3C 0B 77 3C D6   ..........v<.w<.
361911888  00 00 00 00 FA 33 C0 8E D0 BC 00 7C FB B8 C0 07   .....3.....|....
361911904  8E D8 E8 16 00 B8 00 0D 8E C0 33 DB C6 06 0E 00   ..........3.....
361911920  10 E8 53 00 68 00 0D 68 6A 02 CB 8A 16 24 00 B4   ..S.h..hj....$..
361911936  08 CD 13 73 05 B9 FF FF 8A F1 66 0F B6 C6 40 66   ...s......f...@f
361911952  0F B6 D1 80 E2 3F F7 E2 86 CD C0 ED 06 41 66 0F   .....?.......Af.
361911968  B7 C9 66 F7 E1 66 A3 20 00 C3 B4 41 BB AA 55 8A   ..f..f. ...A..U.
361911984  16 24 00 CD 13 72 0F 81 FB 55 AA 75 09 F6 C1 01   .$...r...U.u....
361912000  74 04 FE 06 14 00 C3 66 60 1E 06 66 A1 10 00 66   t......f`..f...f
361912016  03 06 1C 00 66 3B 06 20 00 0F 82 3A 00 1E 66 6A   ....f;. ...:..fj
361912032  00 66 50 06 53 66 68 10 00 01 00 80 3E 14 00 00   .fP.Sfh.....>...
361912048  0F 85 0C 00 E8 B3 FF 80 3E 14 00 00 0F 84 61 00   ........>.....a.
361912064  B4 42 8A 16 24 00 16 1F 8B F4 CD 13 66 58 5B 07   .B..$.......fX[.
361912080  66 58 66 58 1F EB 2D 66 33 D2 66 0F B7 0E 18 00   fXfX..-f3.f.....
361912096  66 F7 F1 FE C2 8A CA 66 8B D0 66 C1 EA 10 F7 36   f......f..f....6
361912112  1A 00 86 D6 8A 16 24 00 8A E8 C0 E4 06 0A CC B8   ......$.........
361912128  01 02 CD 13 0F 82 19 00 8C C0 05 20 00 8E C0 66   ........... ...f
361912144  FF 06 10 00 FF 0E 0E 00 0F 85 6F FF 07 1F 66 61   ..........o...fa
361912160  C3 A0 F8 01 E8 09 00 A0 FB 01 E8 03 00 FB EB FE   ................
361912176  B4 01 8B F0 AC 3C 00 74 09 B4 0E BB 07 00 CD 10   .....<.t........
361912192  EB F2 C3 0D 0A 46 65 68 6C 65 72 20 62 65 69 6D   .....Fehler beim
361912208  20 4C 65 73 65 6E 20 64 65 73 20 44 61 74 65 6E    Lesen des Daten
361912224  74 72 84 67 65 72 73 00 0D 0A 4E 54 4C 44 52 20   tr.gers...NTLDR
361912240  66 65 68 6C 74 00 0D 0A 4E 54 4C 44 52 20 69 73   fehlt...NTLDR is
361912256  74 20 6B 6F 6D 70 72 69 6D 69 65 72 74 00 0D 0A   t komprimiert...
361912272  4E 65 75 73 74 61 72 74 20 6D 69 74 20 53 74 72   Neustart mit Str
361912288  67 2B 41 6C 74 2B 45 6E 74 66 0D 0A 00 00 00 00   g+Alt+Entf......
361912304  00 00 00 00 00 00 00 00 83 A8 B6 CE 00 00 55 AA   ..............U.
```

Mit einer weiteren Suche nach dem OEM-String `NTFS` kann dieser Sektor ebenfalls gefunden werden. Wobei das bei einem Image, welches einige Gigabyte groß ist, aufgrund des häufigen Vorkommens dieses Strings, erheblich zeitaufwändiger ist, als die Berechnung.

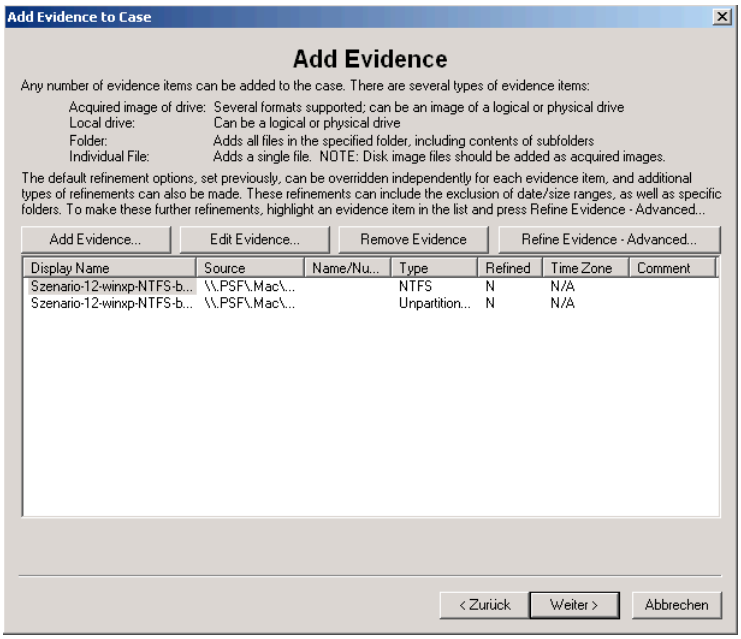
5. Wiederherstellen des Bootsektors: Für die Wiederherstellung wird einfach der Backup-Bootsektor mit dem Hexeditor kopiert und an der Stelle des beschädigten Sektors eingefügt. Bevor das Image nun einer forensischen Untersuchung unterzogen werden kann, muss es noch gespeichert werden.

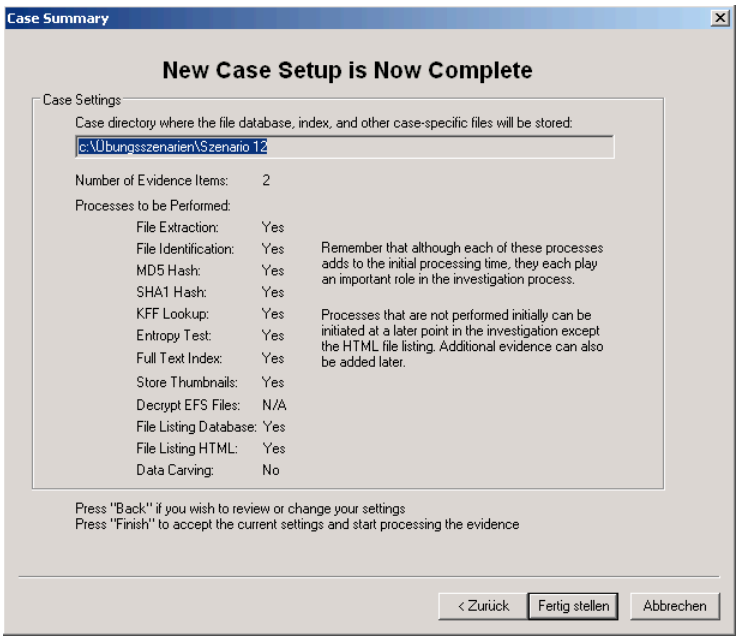
Analyse des Images

Da die Untersuchung des Images für dieses Szenario gleich wie in Szenario 2 erfolgt, wird für die Details dieser auf das Kapitel 7.2.2 (Szenario 2: FAT32 in Windows ME) verwiesen. Im Unterschied zum Szenario 2 wird hier die Suche nicht auf bestimmte Dateien eingschränkt.



Zusammenfassend hier noch die Auflistung der hinzugefügten Beweise.

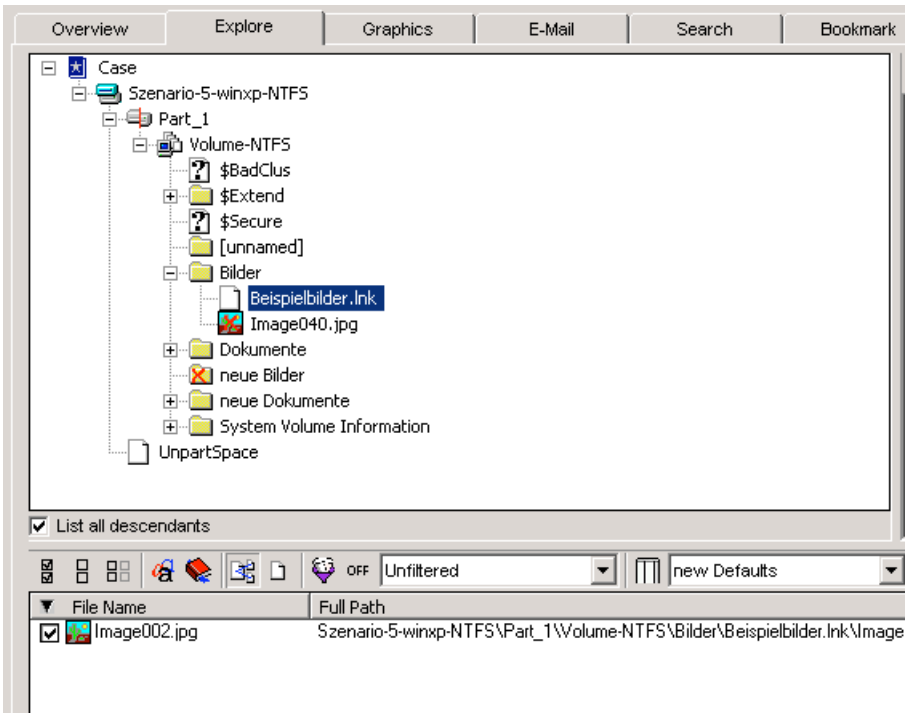




Nachdem alle Informationen angegeben wurden, kann über das Klicken auf den Button `Fertig stellen` mit der Analyse des Images begonnen werden.

- **Ergebnisse der forensischen Untersuchung**

Da das `Forensic Toolkit` keine spezielle Methode für das Filtern bzw. Auflisten von ADS zur Verfügung stellt, müssen diese Daten manuell in der Explorer-Sicht gesucht werden. Der folgende Screenshot zeigt an einer Datei, wie deren ADS angezeigt wird.



Anzeige einer Datei, die als ADS einer anderen Datei gespeichert ist.

In anderen forensischen Analyseprogrammen werden die ADS-Dateien auch mit der folgenden Syntax angezeigt: `Dateiname:ADS-Name`. Diese Art der Anzeige wird zum Beispiel im `Sleuth Kit` verwendet.

Gefundene ADS Dateien nicht gelöschter Dateien

Die Datei `\Bilder\Beispielbilder.lnk` beinhaltet folgende ADS-Datei

Dateiname	Größe in Byte	MD5 Prüfsumme
`Image002.jpg`	352.784	F0C0E3BEDB439F2F40795863B19DDA4C

Die Datei `\Dokumente\SDK-Docs-Tutorials\jwsdp-2_0-ant-docs.zip` beinhaltet folgende ADS-Datei

Dateiname	Größe in Byte	MD5 Prüfsumme
`jwstutorial20.zip`	2.427.310	2A9776EAF04FD078A12EF3B6134ED721

Die Datei `\Dokumente\Seminar - Netzwerke – Security\Screen1.bmp` beinhaltet folgende ADS-Datei

Dateiname	Größe in Byte	MD5 Prüfsumme
`Screen1-Detail.bmp`	791.898	2B9C6C5B91EC8C42AC8BE2F4C4BD59F1

Die Datei `\Dokumente\Seminar - Netzwerke – Security\Screen2.bmp` beinhaltet folgende ADS-Datei

Dateiname	Größe in Byte	MD5 Prüfsumme
`Screen2-Detail.bmp`	605.238	DC6649B66B450D24D7E4BC53642FF98D

Anmerkung

Die Datei `2.Networks.pdf` im Ordner `\Dokumente\Seminar - Netzwerke – Security\Cisco\` wird mit dem `Forensic Toolkit` nicht als ADS des Ordners, sondern als normaler Inhalt angezeigt.

Gefundene ADS Dateien gelöschter Dateien

Die Datei `\Bilder\Image040.jpg` beinhaltet folgende ADS-Datei

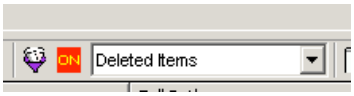
Dateiname	Größe in Byte	MD5 Prüfsumme
`Image039.jpg`	492.559	C37B868A47B7EDED2BDEC6365525C295

Die Datei `\Dokumente\KV Angewandte Systemtheorie - Kryptographie\06 - Signatures.pdf` beinhaltet folgende ADS-Datei

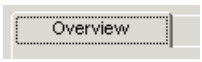
Dateiname	Größe in Byte	MD5 Prüfsumme
Mitschrift 29.11.06.doc	48.128	06547E9EAA747812A5CD9B0C41910FA5

Liste der gefundenen gelöschten Dateien und Ordner

Über das Einsetzen des Dateifilters wird das Auffinden der gelöschten Dateien einfacher, denn mit dem Verwenden der vorhandenen Einstellung für das Anzeigen gelöschter Dateien werden diese einfach angezeigt. Diese vorgegebenen Einstellungen können über das Drop-Down Menü neben dem `ON` Symbol des folgenden Screenshots ausgewählt werden. Soll der Dateifilter angepasst werden, so kann dieser über das Klicken auf das rechte Symbol angepasst werden.



Aktiviert wird der Dateifilter über das Klicken auf den Button `Filtered` auf der Übersichtsseite (`Overview`).



Die Aktivierung des Buttons `Actual Files` bewirkt, dass nur die vollständigen Dateien, nicht aber die eingebetteten Dateien, wie der Inhalt von Zip-Dateien, E-Mails oder OLE-Streams angezeigt werden.

Im Ordner `\Bilder\` gefundene Datei:

Dateiname	Größe in Byte	MD5 Prüfsumme	Fehlerfrei
Image040.jpg	562.769	0E7AC1684009838B5085A71C2B5EA49D	ja

Im Ordner `\Dokumente\KV Angewandte Systemtheorie - Kryptographie\` gefundene Dateien:

Dateiname	Größe in Byte	MD5 Prüfsumme	Fehlerfrei
05 - Asymetric-Crypto.pdf	327.872	1A6A5AC65DDCCE99868E22B51F8E3339	ja
06 - Signa-tures.pdf	511.361	3F2C6F4B2F9AFDC95F4CFECFB246FED5	ja
07 - ECC.pdf	153.306	C5508A72297CB052F120E3B1D7068C79	ja

08 - PGP.pdf	668.244	63E1AA9B564EAD28892261282EA42AC6	ja
09 - ssl.pdf	406.166	3435C6244E44C36C35CB3485501C4B5C	ja
10 - SecInternet-Payment.pdf	1.412.756	4CE931E1D272F5676327A66A86D91026	ja
HalloLeute.pdf	191.566	4AC54DAD2C20C9E48DFC7230B8A9705F	ja

Im Ordner `\Dokumente\SDK-Docs-Tutorials\` gefundene Datei

Dateiname	Größe in Byte	MD5 Prüfsumme	Fehlerfrei
j2ee-5_0-beta-doc-tutorial.zip	12.276.520	F193EC2247C1C76AAC2D44748AB60A2F	ja

Im Ordner `\Dokumente\Seminar - Netzwerke - Security\cg0705.mspx_files\` gefundene Dateien:

Dateiname	Größe in Byte	MD5 Prüfsumme	Fehlerfrei
ADSAdClient31.htm	868	FC90E1EA430AD61272382AD33D915ECA	ja
arrow_px_down.gif	53	A50A5A7BE6297D534E0A9559C583C217	ja
arrow_px_up.gif	53	FBAE595C25E4EB510CE17F4EA17C8760	ja
arrowLTR.gif	821	00BA9FF02078EE89D912B07B8E7D3838	ja
broker.js	4.571	3978DB0D26A384B5622DE03A87C87C6F	ja
cableguy.gif	30.792	0E94B6586D8A5D58C143F471C483F376	ja

Weiters wurde in diesem Ordner der gelöschte Ordner `\ratings_data` mit folgenden Dateien gefunden.

Dateiname	Größe in Byte	MD5 Prüfsumme	Fehlerfrei
$I30	4.096	E80C95B223972320B00DEC7DB7A9E8E1	*
ContentRating.css	420	F02DC615EF1E9293C74330D6D59BB14D	ja
rtg_email.gif	1.010	B201A23EC07D55866FAB17A62030774E	ja
rtg_print.gif	574	101A0AC63030B36FCE0421D896940AEF	ja
rtg_save.gif	567	7D20B965EA147E6DC2B580CE6AE71778	ja

Ordner `\neue Bilder` im Stammverzeichnis mit folgendem Inhalt gefunden.

Dateiname	Größe in Byte	MD5 Prüfsumme	Fehlerfrei
$I30	4.096	489731EBB561C8ED8C6951C84401B8D6	*
Image050.jpg	504.864	6C455C48DED3ABA6C275BA38D66F3835	ja
Image051.jpg	460.033	4505C879A31416AF1BDD858F888A28DF	ja
Image052-bearbeitet.jpg	1.224.896	CB6A72603C198555FD9AA52FF9C9A621	ja
Image053.jpg	390.427	B4C8F57484C357B811C6FB56C4B001C6	ja

* Angabe nicht möglich, da die Datei eine Indexdatei des Dateisystems ist und vor dem Löschen nicht sichtbar war. Genau genommen handelt es sich hier um das $INDEX_ALLOCATION Attribut des Ordners. Details darüber im Kapitel 4.2.1.3.

- **Zusammenfassung**

Nach einer erfolgreichen Korrektur des Images, wie sie in der Anleitung beschrieben wurde, ist es möglich, alle gelöschten Dateien und Ordner zu finden und fehlerfrei wiederherzustellen.

8 Zusammenfassung

Diese Arbeit beschreibt ein allgemeines Dateisystemmodell und vergleicht anschließend die gängigen Dateisysteme FAT, NTFS und Ext3 mit diesem Modell. Diese Vergleiche bieten einen guten Überblick über den Aufbau und die Funktionsweise dieser Dateisysteme, wobei hier nicht alle verwendeten Datenstrukturen der Dateisysteme beschrieben werden. Die genaue Beschreibung derselben findet sich entweder in der Dokumentation des Dateisystems oder in weiterführender Literatur.

Nach dem Lösen der Übungsszenarien sollten die Studenten in der Lage sein mit dem verwendeten Computer-Forensik-Programm umzugehen und etwas schwierigere Fälle lösen zu können. Um das Wissen über Dateisysteme und deren Datenstrukturen zu vertiefen, ist es den Studenten möglich, sich in weiteren Szenarien mit der detaillierten Analyse einzelner Datenstrukturen eines Dateisystems zu beschäftigen. Interessant wäre beispielsweise die Untersuchung des Dateisystemjournals in Kombination mit der genauen Analyse der Metadatenstrukturen, so wie die Suche nach Einbrechern in ein System unter Zuhilfename der Logfiles des Intrusion Detection Systemen bzw. der Firewall.

Persönliche Erfahrung und Ausblick

Nachdem ich mich zuvor mit dem Thema Computer-Forensik noch nicht beschäftigt hatte, dauerte die Einarbeitungsphase, welche ebenfalls das Suchen und Ausprobieren der verschiedenen Forensik-Programme umfasste, etwas länger. Da ich für das Erzeugen der Images keine kleine Festplatte mit ein paar hundert Megabyte hatte, behalf ich mir mit den virtuellen Festplatten des Programms `Parallels Desktop`. Diese Entscheidung erwies sich sehr schnell als äußerst praktikabel, da ich die damit erzeugten virtuellen Festplatten ohne großen Aufwand gleich als Images verwenden konnte. Ein weiterer Vorteil dieser Methode bestand darin, dass ich die Images während des Erstellungsprozesses untersuchen und bei Bedarf einfach ändern konnte. Trotz dieser Vorteile nahmen die Erzeugung der Images und die Ausarbeitung der Musterlösungen einen Großteil der Zeit in Anspruch.

Die Schwierigkeiten bei der Ausarbeitung der Dateisystembeschreibung waren das Recherchieren der Literatur sowie das Verstehen und Einarbeiten in die einzelnen Dateisysteme aufgrund der vielen verwendeten Datenstrukturen für das Speichern der verschiedenen Daten. Bei den Recherchen lag die Herausforderung darin, dass die Dokumentationen vom Hersteller teilweise gar nicht zugänglich oder mit der des Betriebssystems vermischt sind. Für das Verstehen der Funktionsweise der Dateisysteme war es für mich sehr hilfreich zuerst das allge-

meine Model auszuarbeiten und aufbauend auf dieses die konkreten Dateisysteme zu beschreiben. Weiters könnte der theoretische Teil um eine detaillierte Beschreibung der Unterschiede sowie der Stärken und Schwächen der beschriebenen Dateisysteme ergänzt werden.
Persönlich lernte ich durch die Beschäftigung mit diesem Thema viel über die drei beschriebenen Dateisysteme, die Anwendung forensischer Werkzeuge, das Erstellen von Festplattenimages und darüber hinaus, was bei einer forensischen Untersuchung beachtet werden muss.

Abbildungsverzeichnis

Literatur

[Bar02] Michael Barba, „Computer Forensic Investigations“, ASIS Presentation 2002, http://www.computer-forensic.com/old_site/presentations/ASIS_Presentation.pdf

[Buc03] Johannes Buchmann, „Einführung in die Kryptographie“, 3. erweiterte Auflage 2004 (September 2003), Springer Verlag Berlin, ISBN 3540405089

[Bui03] Sonia Bui, Michelle Enyeart, Jenghuei Luong, „Issues in Computer Forensics“, 22.05.2003, http://www.cse.scu.edu/~jholliday/COEN150sp03/projects/Forensic%20Investigation.pdf

[Car05] Brian Carrier, „File System Forensic Analysis“, Adision-Wesley Perason Education 2005, ISBN 0-321-26817-2

[CERT05] US-CERT – United States Computer Emergency Readiness Team, „Computer Forensics“, 2005, http://www.us-cert.gov/reading_room/forensics.pdf

[Eck04] Dr. Knut Eckstein, „Forensics for Advanced UNIX File Systems“, IEEE/USMA IA Workshop 2004

[Ges06] Alexander Geschonneck, „Computer Forensik“, 2., aktualisierte Auflage 2006, dpunkt Verlag, ISBN 3-89864-379-4

[Haa01] Norman Haase, „Computer Forensics: Introduction to Incident Response and Investigation of Windows NT/2000“, 4. Dezember 2001, SANS Institute 2001

[Ham06] Oliver Hamel, „ext3, Grundlagen und Tuning“, Linux Enterprise Ausgabe 6 2006, http://entwickler.com/itr/online_artikel/psecom,id,868,nodeid,9.html

[IT-Lexikon] IT-Lexikon: Fachwissen für IT-Professionals, „Definition: Computer-Forensik“, http://www.itwissen.info/definition/lexikon/Computer-Forensik-computer-forensics.html, 12. Juni 2008

[Joh01] Michael K. Johnson, „Whitepaper: Red Hat’s New Journaling File System: ext3“, Red Hat 2001, http://www.redhat.com/support/wpapers/redhat/ext3/index.html

[Ker-1] ext2.txt, Kerneldokumentation, Im Unterordner `Documentation/filesystems/ext2.txt` des Kernel Source Codes

[Ker-2] ext3.txt, Kerneldokumentation, Im Unterordner `Documentation/filesystems/ext3.txt` des Kernel Source Codes

[Kil97] Joe Kilian, Phillip Rogaway, „How to Protect DES Against Exhaustive Key Search“, 28. Juli 1997, http://wwwcsif.cs.ucdavis.edu/~rogaway/papers/desx.ps

[Kof04] Michael Kofler, „Linux: Installation, Konfiguration, Anwendung“ 7., [vollst. überarb.] Auflage 2004, Addison-Wesley, ISBN 3-8273-2158-1

[Linux-NTFS] Linux-NTFS Projekt, http://www.linux-ntfs.org

[MS00] Microsoft Corporation, „FAT32 File System Specification“, Version 1.03 6.12.2000, fatgen103.doc, http://download.microsoft.com/download/1/6/1/161ba512-40e2-4cc9-843a-923143f3456c/fatgen103.doc

[MS02-1] Microsoft Knowledge Base Artikel 314878, „Standard-Clustergröße für FAT- und NTFS-Dateisysteme“, http://support.microsoft.com/kb/314878

[MS03-1] Microsoft Knowledge Base Artikel 299648, „Beschreibung von NTFS-Datums- und Zeitstempeln für Dateien und Ordner“, http://support.microsoft.com/kb/299648

[MS03-2] Microsoft Knowledge Base Artikel 205524, „Erstellen und Bearbeiten von NTFS-Abzweigungspunkten (NTFS Junction Points)“, http://support.microsoft.com/kb/205524

[MS03-3] Microsoft Knowledge Base Artikel 245725, „HOW TO: Recover an Accidden-tally Deleted NTFS or FAT32 Dynamic Volume“, http://support.microsoft.com/kb/245725

[MS03-4] Microsoft Knowledge Base Artikel 314463, „Beschränkungen des FAT32-Da-teisystems in Windows XP“, http://support.microsoft.com/kb/314463

[MS04-1] Microsoft Knowledge Base Artikel 174619, „ So reserviert NTFS Speicher-platz für die Master-Dateitabelle“, http://support.microsoft.com/kb/174619

[MS04-2] Microsoft Knowledge Base Artikel 121517, „ Wie geht man vor, wenn der NTFS-Bootsektor beschädigt ist?“, http://support.microsoft.com/kb/121517

[MS04-3] Microsoft Knowledge Base Artikel 153973, „NT: Wiederherstellen des NTFS-Boot-Sektors bei NTFS-Partitionen“, http://support.microsoft.com/kb/153973

[MS04-4] Microsoft Knowledge Base Artikel 253845, „ Neue Fähigkeiten und Features des Dateisystems NTFS 3.0“, http://support.microsoft.com/kb/253845

[MS05-1] Microsoft Knowledge Base Artikel 100108, „Übersicht über Dateisysteme FAT, HPFS und NTFS“, http://support.microsoft.com/kb/100108

[MS05-2] Microsoft Knowledge Base Artikel 184006, „Für das FAT32-Dateisystem gel-tende Beschränkungen“, http://support.microsoft.com/kb/184006

[MS05-3] Microsoft Knowledge Base Artikel 69912, „Zusammenfassende Darstellung der MS-DOS-Partitionierung, http://support.microsoft.com/kb/69912/

[MS05-4] Microsoft Knowledge Base Artikel 315688, „Probleme mit dem Festplattenspeicher auf NTFS-Volumes unter Windows XP identifizieren und beheben“, http://support.microsoft.com/kb/315688

[MS06-1] Microsoft Knowledge Base Artikel 140418, „Detaillierte Erläuterungen zum FAT-Bootsektor“, http://support.microsoft.com/kb/140418/

[MS06-2] Microsoft Knowledge Base Arikel 835840, „Data recovery“, http://support.microsoft.com/kb/835840/EN-GB/

[MS06-3] Microsoft Knowledge Base Artikel 303079, „Probleme mit dem Festplattenspeicher auf NTFS-Volumes identifizieren und beheben“, http://support.microsoft.com/kb/303079

[MS06-4] Microsoft Knowledge Base Artikel 154997, „Beschreibung des FAT32 File Sytems“, http://support.microsoft.com/kb/154997

[MS06-5] Microsoft Knowledge Base Artikel 241201, „ Sichern des privaten EFS-Schlüssels für den Wiederherstellungsagenten in Windows Server 2003, Windows 2000 und Windows XP“, http://support.microsoft.com/kb/241201

[MS07-1] Microsoft Knowledge Base Artikel 140365, „Standardclustergröße für FAT und NTFS“, http://support.microsoft.com/kb/140365

[MS07-2] Microsoft Knowledge Base Artikel 121007, „Deaktivieren der 8.3-Namenserstellung auf NTFS-Partitionen“, http://support.microsoft.com/kb/121007

[MS07-3] Microsoft Knowledge Base Artikel 310749, „Neue Möglichkeiten und Features des Dateisystems NTFS 3.1“, http://support.microsoft.com/kb/310749

[MS07-4] Microsoft Knowledge Base Artikel 329741, „Encrypting File System (EFS) files appear corrupted when you open them“, http://support.microsoft.com/kb/329741/en-us

[MSDN-1] Microsoft Developer Network Library, „File Times“, http://msdn2.microsoft.com/en-us/library/ms724290.aspx

[MSDN-2] Microsoft Developer Network Library, „Filetime“, http://msdn2.microsoft.com/en-us/library/ms724284.aspx

[MSDN-3] Microsoft Developer Network Library, „Naming a File“, http://msdn2.microsoft.com/en-us/library/aa365247.aspx

[MSTN03-1] Microsoft TechNet, “How NTFS Works“, 28.März 2003, http://technet2.microsoft.com/WindowsServer/en/library/8cc5891d-bf8e-4164-862d-dac5418c59481033.mspx?mfr=true

[MSTN03-2] Microsoft TechNet, “What is NTFS?“, 28.März 2003, http://technet.microsoft.com/en-us/library/cc778410.aspx

[MSTN03-3] Microsoft TechNet, “How NTFS Works“, 28.März 2003, http://technet.microsoft.com/en-us/library/cc781134.aspx

[MSTN05] Microsoft TechNet, „NTFS compared to FAT and FAT32“, 21.01 2005, http://technet2.microsoft.com/windowsserver/en/library/61db3cb9-3c46-487a-a199-23c0d6572fc01033.mspx?mfr=true

[Nel04] Bill Nelson, Amelia Phillips, Frank Enfinger, Christopher Steuart, „Computer Forensics and Investigations“, März 2004, ISBN 1-59200382-6, Course Technology

[ntfs.com] Informationen über die Dateisysteme FAT und NTFS inklusive WinFS. www.ntfs.com

[Ott02] Thomas Ottmann, Peter Widmayer, „Algorithmen und Datenstrukturen“ 4. Auflage 2002, Spektrum Akademischer Verlag, ISBN 3827410290

[Pos] Brien Posey, TechRepublic's Windows Support Professional (TechRepublic.com), „Choosing Between FAT and NTFS“, http://www.microsoft.com/technet/archive/ittasks/deploy/fat.mspx?mfr=true

[Rog96] Phillip Rogaway, „The Security of DESX“, Draft 2.0, 5. Juli 1996, http://www.cs.ucdavis.edu/~rogaway/papers/cryptobytes.ps

[Rus01] Charlie Russel, „NTFS vs. FAT: Which Is Right for You?“, 1. Oktober 2001, http://www.microsoft.com/windowsxp/using/setup/expert/russel_october01.mspx

[Sch05] Bruce Schneier, „Angewandte Kryptographie - Der Klassiker“ 2005, Pearson Studium, ISBN 3827372283

[Sch07] Klaus Schmeh, „Kryptographie“, 3. Überarbeitete und erweiterte Auflage 2007, dpunkt Verlag, ISBN 3898644359

[Sve05] Anders Svensson, „Computer Forensics Applied to Windows NTFS Computers“, Master thesis, Stockholm's University / Royal Institute of Technology, Kista, Stockholm, Sweden, April 2005

[Twe00] Stephen C. Tweedie, „EXT3, Journaling Filesystem“, 20. Juli 2000, http://olstrans.sourceforge.net/release/OLS2000-ext3/OLS2000-ext3.html

[Twe98] Stephen C. Tweedie, „Journaling the Linux ext2fs Filesystem“, LinuxExpo98, ftp://ftp.kernel.org/pub/linux/kernel/people/sct/ext3/journal-design.ps.gz

[Wri01] Howard Wright, „The Encrypting File System – How Secure is It?“, SANS Security Essentials (GSEC) Version 1.2f, 2.11.2001, http://www.sans.org/reading_room/whitepapers/win2k/211.php

Anmerkung: Wenn nicht anders angegeben, sind alle Links zuletzt am 5. Mai 2008 besucht worden.

Anhang

A. MD5 Prüfsummen der Dateien aus der Dateisammlung

In diesem Kappitel werden alle Dateien die bei der Erstellung der Images für die Szenarien 2 - 5 und 8 – 12 verwendet wurden mit Größe und Prüfsumme sortiert nach dem enthaltenen Ordner aufgelistet.

Ordner: `Bilder`

Dateiname	Größe in Byte	MD5 Prüfsumme
Blick auf den Wildensee vom Rinnerkogel.JPG	293.902	46EAEE7075DFA2114E67C46969555C30
Dachstein vom Rinnerko-gel.JPG	56.521	EF1B6461A9905EDDA2081FEC816D467D
Dachstein.jpg	496.538	0636586E7554A96A32A2AB97544CD428
Gipfelkreuz Rinnerko-gel.JPG	165.897	64659A420FB7A37CF17D6371C8C6B2CD
Gosaukamm + Dachstein von Katrin.JPG	51.049	99C701E63073E53EB24CBB42F360B022
Image002.jpg	352.784	F0C0E3BEDB439F2F40795863B19DDA4C
Image003.jpg	401.615	8992CDB9537DB2AD06C7EF49EA7A1526
Image004.jpg	370.495	35A1B1303A61BF5BC92084DD500AA398
Image012.jpg	278.153	4408BC82013C350DD2063B8D43CEB82E
Image013.jpg	382.099	AD0D3DAE415BF1C5637E3F2573AAD451
Image016.jpg	488.279	C3795C0E33D4B49EE74E9A2F17C908E9
Image020.jpg	465.511	30955B64C2686396A0F081DD4608B34D
Image022_bearbeitet.jpg	978.997	94F608C1B3C764C6FC0C4B5DEA9F609E
Image028.jpg	337.195	4591C40954AD6A1FEEF4F17FA8971577
Image029.jpg	360.935	AB4A403E493A79628D0974882671DF0A
Image030.jpg	470.012	71DA5B4AC0C7E5E966EE1B8AAE08E00E
Image031.jpg	383.440	D332FE389E72A38CC9F9A5B1000DF73F
Image039.jpg	492.559	C37B868A47B7EDED2BDEC6365525C295
Image040.jpg	562.769	0E7AC1684009838B5085A71C2B5EA49D

Ordner: `Dokumente/KV Angewandte Systemtheorie – Kryptographie`

Dateiname	Größe in Byte	MD5 Prüfsumme
01 - Intro.pdf	555.147	DBB8A499D8A6FB200F65F3D6A2B1DC11
02 - DES.pdf	333.246	713A3D7C5179860C632818BA4368606F

03 - FromDES_ToAES.pdf	670.463	A442C23B2F7E8E68D7C973DDD2BD5935
04 - Streamciphers.pdf	370.164	9672DC6BE1B59C58CDE1D769577AD885
05 - AsymetricCrypto.pdf	327.872	1A6A5AC65DDCCE99868E22B51F8E3339
06 - Signatures.pdf	511.361	3F2C6F4B2F9AFDC95F4CFECFB246FED5
07 - ECC.pdf	153.306	C5508A72297CB052F120E3B1D7068C79
08 - PGP.pdf	668.244	63E1AA9B564EAD28892261282EA42AC6
09 - ssl.pdf	406.166	3435C6244E44C36C35CB3485501C4B5C
10 - SecInternetPay-ment.pdf	1.413.042	4CE931E1D272F5676327A66A86D91026
HalloLeute.pdf	191.566	4AC54DAD2C20C9E48DFC7230B8A9705F
Mitschrift 29.11.06.doc	48.128	06547E9EAA747812A5CD9B0C41910FA5

Ordner: Dokumente/Seminar - Netzwerke – Security

Dateiname	Größe in Byte	MD5 Prüfsumme
1.Intro.pdf	133.565	701D6EA87F7B72F3BA1C3994991B2822
2.Networks.pdf	376.297	EFB288CF96E3F492C77187DEFE526509
3.Vulnerability Analysis.pdf	380.177	751E164066D23341D0B10976F06B1530
4.NAM.pdf	214.495	25F279F91150AC677B7BB49A9411601D
5.Google Hacking.pdf	1.138.327	F497AD7B16F87D6C3DBC0F0695BF23D6
6.CVSS.pdf	551.583	9E2F4271606746E2683F2E8A674F689C
2005_20online.pdf	419.517	E14402CD517111948647227D90134631
8021X_IPsec.doc	231.424	2FD7EFA171E1A754C9310535F420E6DD
92216901.PDF	2.584.952	B2B38D4A3CDEBAD26E2B43BC916C09D8
cg0705.mspx.htm	39.029	26ADC88A153A96D8DE7E4CB67D14124D
diverse Links.txt	877	36C748CF6B43DCD2B0D97C5EA28AEA48
include.pdf	136.517	5B61D170CB1CCD3A1C5A751C58EAE85E
Mögliche Seminararbeitsthemen.doc	19.968	1D0E426E85A6E5FFF91D1C52E86DC36F
NAP_IPsec.doc	445.440	16FBD9EAA1C5469E87700FE4F648A049
NAP_Policy.doc	527.872	0F3D3F77BCF8C60340B882D068349067
NAP-Pic1.jpg	15.256	8129A1D9CA03F49A3E315559FC611040
NAP-Pic2.jpg	17.690	A729A8DD90F0C39DDD0E6F96621FC706
NAPArch.doc	470.016	EC31DE70C78A03FA11D457CFFF107C61
NAPIntro.doc	234.496	168FBE16DE76E0CC127510761CD30011
Open_Standards_for_IntegrityBased_AccessControl.pdf	220.249	4BEDBF72898AE287C898AF9BB787BBF1
Screen1-Detail.bmp	791.898	2B9C6C5B91EC8C42AC8BE2F4C4BD59F1
Screen1.bmp	2.359.350	60EC289902BF709F57F841A452A4E516
Screen2-Detail.bmp	605.238	DC6649B66B450D24D7E4BC53642FF98D
Screen2.bmp	2.359.350	5B441F14173278E3697EC2F0A1739824
Securtiy_CNAC_032004.pdf	436.806	4B4F0DB4FFD16830B9494FAB5B7D88B0
Seminararbeit.doc	1.003.520	475A4C1E52E4F804037D14C6A1E46DDB
TCG_1_0_Architecture_Overview.pdf	585.123	459AEC22DEE9C2B33F4A7717D968823A
The Laws of Vulnerabilities 2005	421.988	C36882E7747A6192935CBF217D7E9764

The Laws of Vulnerabilities 2005 Edition.pdf	421.988	C36882E7747A6192935CBF217D7E9764
Thumbs.db	22.016	CE22CCD1AAE89D39FA8F24F3604DB772
TNC Architecture v1 0 r4.pdf	354.493	68FF5BE864433BC059C38C399819C303

Ordner: Dokumente/Seminar - Netzwerke – Security/Cisco

Dateiname	Größe in Byte	MD5 Prüfsumme
cdccont_0900aecd80217e26.pdf	2.396.587	8684A9C05065BA089CC58C87B97C1D38
cdccont_0900aecd80234ef4.pdf	602.417	622574D247BD75572C1186486591155F
Securtiy_CNAC_032004.pdf	436.806	4B4F0DB4FFD16830B9494FAB5B7D88B0

Ordner: Dokumente/Seminar - Netzwerke – Security/cg0705.mspx_files

Dateiname	Größe in Byte	MD5 Prüfsumme
ADSAdClient31.htm	868	FC90E1EA430AD61272382AD33D915ECA
arrow_px_down.gif	53	A50A5A7BE6297D534E0A9559C583C217
arrow_px_up.gif	53	FBAE595C25E4EB510CE17F4EA17C8760
arrowLTR.gif	821	00BA9FF02078EE89D912B07B8E7D3838
broker.js	4.571	3978DB0D26A384B5622DE03A87C87C6F
cableguy.gif	30.792	0E94B6586D8A5D58C143F471C483F376
cg07050.jpg	15.256	8129A1D9CA03F49A3E315559FC611040
cg07051.jpg	17.690	A729A8DD90F0C39DDD0E6F96621FC706
css_002.css	2.572	3E1403BF57EE32B3F5D4377C5AC7A066
css.css	2.713	871B7056072A4C81CC42629B3940E1C3
gradient_002.jpg	869	C356C7D8C367BBBDCD157766585D1DC6
gradient.jpg	1.619	C64CD96C7A3A133C4B6991A1608B1144
menujs	34.334	67EDB22B47CF2D195B29CAF19AAB7EB7
ratings.htm	8.697	DDDD7A9B0F0011842925FCAEB35F1C71
SiteRecruit_PageConfiguration _2943mt33-3089mt-2944mt1.js	2.910	865E579A119E41BF2993417676BACAAA
TechNetB_masthead_ltr.gif	3.576	5249D778EB3E80146D0A6401395300DC
templatecss.css	11.290	B5534574022FFDAA179790AC3B09BEF3
text.jpg	1.908	CA86E0AE62452F4F1103D034FE337B58
Thumbs.db	8.192	22FCDA2241363F882061ECBEFAC8B204
trans_pixel.gif	44	6D69C4DE0545F6FC9BA3DCD899E29501

Ordner: `Dokumente/Seminar - Netzwerke – Security/cg0705.mspx_files/ADSAdClient31_data`

Dateiname	Größe in Byte	MD5 Prüfsumme
0000053432_00000000000000265828.gif	10.099	14A168D6573518C13C57971CDE8850EB

Ordner: `Dokumente/Seminar – Netzwerke – Security/cg0705.mspx_files/ratings_data`

Dateiname	Größe in Byte	MD5 Prüfsumme
ContentRating.css	420	F02DC615EF1E9293C74330D6D59BB14D
rtg_email.gif	1.010	B201A23EC07D55866FAB17A62030774E
rtg_print.gif	574	101A0AC63030B36FCE0421D896940AEF
rtg_save.gif	567	7D20B965EA147E6DC2B580CE6AE71778

Ordner: `Dokumente/SDK-Docs-Tutorials`

Dateiname	Größe	MD5 Prüfsumme
j2ee-5_0-beta-doc-tutorial.zip	12.276.520	F193EC2247C1C76AAC2D44748AB60A2F
javadocs.zip	8.015.165	A2225057AF5C02208BA985E9DEE68399
jdk-1_5_0-doc.zip	46.191.338	06D108655466460E12BC9018C5FEE863
jwsdp-2_0-ant-docs.zip	7.318.868	90B9EE408B0D1F7E72277BFF7300E9DD
jwstutorial20.zip	2.427.310	2A9776EAF04FD078A12EF3B6134ED721
Java_Bücherliste.doc (Image022_bearbeitet.jpg)	470.012	71DA5B4AC0C7E5E966EE1B8AAE08E00E
how_to_use_the_javadoc.doc (Image030.jpg)	978.997	94F608C1B3C764C6FC0C4B5DEA9F609E

MD5 Prüfsummen der einzelnen Imagedateien

Dateiname	Größe in Byte	MD5 Prüfsumme
Szenario-2-winme-FAT32.dd	419.586.048	722F4A502E93B183C9ABDB563A5F1E9D
Szenario-3-openSUSE103-FAT32.dd	157.409.280	69BA0F6AEAD441F1AA645B6679DEFE67
Szenario-4-openSUSE103-FAT32.dd	367.460.352	2FDFCD370AA69C0638DBF1EDE1281E9F
Szenario-4-winxp-FAT32.dd	367.460.352	8617488CF5B7CD12195CB3BFFB899F88

Szenario-5-winxp-NTFS.dd	367.460.352	2EF2CD9BAF942D7C15BC52AC3DB555C1
Hunter XP.E01	585.230.499	8B40554177D2DD0B385A253EEF9A49E8
Szenario-8-winxp-NTFS.dd	1.573.060.608	366331EA95BE0CE1761B00DF87CD020C
Szenario-9-openSUSE103-ext3.dd	210.051.072	F702E571153667B57F3B580BDDBE2D38
Szenario-10-winxp-NTFS.dd	388.104.192	D887414D60D8B901E183CA29A8D0E9BE
Szenario-11-winxp-NTFS.dd	367.460.352	5A506CE631825CF2D1B9A5D52D5138E9
Szenario-12-winxp-NTFS.dd	367.460.352	5C71F436D8B8EF7E02D50018319C7E5A

Ordner: Netzwerkadministration

Dateiname	Größe in Byte	MD5 Prüfsumme
NetAdm_c_Server_versus_Work-station Linz.pdf	195.598	8D287EEEB85FFAAE14305A19F4B7818C
NetAdm_d_Workstation Linz.pdf	1.634.160	E5692468A5026C9983EFD3B06B0A5E08
NetAdm_e_Installation Linz.pdf	476.829	D483F6997FB21530ED8B40B88DCC7A51
NetAdm_f_MMC_Eventlog_Perf Linz.pdf	677.811	B7DB9EB5A131EC19F9815C73C6E7C239
NetAdm_g_Diskadmin Linz.pdf	540.072	7A7C0BCA6EE98C918E628B03EC08ECFA
NetAdm_h_UserManagement0 Linz.pdf	2.231.049	C845CD392F609AE3162ADAD9937A9972
NetAdm_i_NTFS Linz.pdf	1.501.695	18DB50EC8947445B97F980DB5398432D
NetAdm_j_Printing STP.pdf	750.913	53C27451FE9D626A47C9D74E01D38CAD
NetAdm_k_Shares Linz.pdf	836.034	A1C2B66A4FB8A345CAB4A1F2ED44F54D
NetAdm_l_ADS.pdf	897.738	1BC450F278DBD5997ED97282C6555C2B
NetAdm_Mailserver.pdf	1.458.868	4CE25C8FA1CCAB5D0FC619328301B706
NetAdm_m_Mailserver.pdf	1.458.868	4CE25C8FA1CCAB5D0FC619328301B706
NetAdm_n_DHCP.pdf	1.306.896	5CC17C78F3E01238F02876A07620551C
NetAdm_o_DNS (Netzwerke).pdf	876.533	4D1193778B6CB27124A23B6D998124D9
NetAdm_o_DNS.pdf	939.640	943516FCB381974EE5472C0300478877
NetAdm_p_IIS.pdf	739.469	D644187B044F6AC09247D0F66B5BEE49
NetAdm_q_ActiveContent.pdf	1.502.990	83CA66ABBF3EF0A2E428F48845F2534A
NetAdm_r_SNMP.pdf	1.081.460	B122DE8ACDED64012264FDC23A4C27F8

Ordner: Systemtheorie 1

Dateiname	Größe in Byte	MD5 Prüfsumme
01-intro-slides.pdf	88.267	EEBB6D2959490674AF9DF39008C2B69B
02-ea-slides.pdf	110.532	6C52CBD062D57C2F38098DEA50175E45
03-sim-slides.pdf	93.951	F8A6A2EDAA7DB9023E554967DC7280CB
04-reach-slides.pdf	173.255	693C20D6BC8AB70C0F63491BD358BBD9
05-po-slides.pdf	126.912	B0E00B60243A6AFE79178FD1406BE7FE
06-fair-slides.pdf	99.581	8ADC68B2F9D74260F5FB319B1DE25926

07-aigs-slides.pdf	156.888	BDEAE424C384E8F19406CDBDFC775DDA
08-bdds-slides.pdf	147.086	98051EA97B6B8322E3519514EEEC6710
aig4teaching.zip	7.355	D45DD7529B9F66E3FFDB8563774F0BE1
aigs-handout-2x2.pdf	165.137	FAB95912B4D2B0B5590802F83B13AC2D
bdd4teaching.zip	5.868	3FDE19144BEC008EB7966FB92E6BF15C
bdds-handout-2x2.pdf	155.235	8F305C13034497AB4BC4FF7E2D328D40
crossing.pml	803	7990CDE98C9CA1033D8EA63AE837D9C2
ea-handout-2x2.pdf	120.617	04789E77BAF50497A74606F07B064074
exercise24.pml	1.702	F4801947B4DA9CECC68DF1A0A017D819
fair-handout-2x2.pdf	108.207	4FECE1985BF41BF83BFB0EE812CAB9FC
fs1-ex1.pdf	47.813	EF528BB5F2DB262F54E7E3CB44832157
fs1-ex10.pdf	45.371	EC4B80266C0AAC1A2BA9BFDC711FD204
fs1-ex11.pdf	49.372	0927FDCF94B3FFAC51B0DD4842D40C31
fs1-ex2.pdf	68.630	562E7C9ACB033EC435A8447507601529
fs1-ex3.pdf	116.557	B56323C889F26323C83739D64A5CEC47
fs1-ex4.pdf	124.149	E999F6ABADB6CD990DDCA06B61B89253
fs1-ex5.pdf	119.098	02F1255E2C8263AE0D7564D9E1FE80B4
fs1-ex6.pdf	100.647	B5B0DA770C0FCDCDB3F8B6DE72C49D4F
fs1-ex7.pdf	65.066	A34E6F080F0E46122EF45F4DA8D8EBF4
fs1-ex8.pdf	69.040	B0F9229400584102469 16DE838389611
fs1-ex9.pdf	63.487	055C44A041EDB7AAF10AE1C2C7570663
fs1-handout-2x2.pdf	558.663	F7157AE25B05114E9EBF78510420C9CE
fs1-slides.pdf	548.860	CA8FCAAD627E3BF8F5DE2FA4798FBD40
intro-handout-2x2.pdf	92.005	0C07B0F00B452DBEF1A125950C89FF25
lg-0.9.tar.gz	58.478	964437BDB6CD268F0BC22738DCCAB05C
mutex.pml	993	07C19754FFFB2A6AE69AA82E56D32A6D
philosophers.pml	2.049	FC29C40A5594AD4B6FD208F06D17320F
philosophers_deadlock.pml	1.876	069C499CA24A1CED980A1F13D4F685B5
po-handout-2x2.pdf	135.507	C4E4DB7066FDC789502C83137CBF55AE
reach-handout-2x2.pdf	181.693	99A3B232F0FB94A9A9C6F877FBA36C94
sim-handout-2x2.pdf	100.511	09CF7053FFA0B4B7E1A4AE66DAE384DB
sources.zip	1.074.201	F803CBB2FF888C1F435C9D3A84252220
testhash.c	16.158	1B8C5E43D54C616CF9CD112CC0F54799
testhashex.c	17.103	782E85AC2BE62F00A03E1047E9B8826F

Ordner: `Systemtheorie/FMSCalc`

Dateiname	Größe in Bytes	MD5 Prüfsumme
FSMCalc.jar	167.638	E523A6B4607B61F753AC54AAAF88AF7A
FSMCalc.pdf	586.933	70CE8D6A978FB78C128C1530591C644E
FSMCalc.zip	1.208.407	687BE011172B57A6927DA1B974F4F930
jgraph.jar	157.121	E238E43F18BCBD0AFB020618BD610D18

Ordner: `VL Pervasive Computing Infrastruktur`

Dateiname	Größe in Bytes	MD5 Prüfsumme
01 Pervasive Vision.pdf	5.397.571	12A534CECE924E784A277F9A4D64E8C4
01 reading 1 weiser 21st century.pdf	35.761	B4E85DCA8AA462B640DECBF6E553EC2F
01 reading 2 weiser+brown age of calm.pdf	37.398	28F3AB11BB5C8ED2DB8C7968D6C5D5A9
01 reading 3 ferscha what is pervasive computing.pdf	175.659	74F17B5D72C6085D8B4AF123E0888210
02 Spontane Vernetzung.pdf	455.495	7D9E38DDEAF1E7B8AF1BE10E80BC224E
02 reading 1 frodigh++ wireless ad hoc networking .pdf	714.914	F1FE10FE7CBE0EB5733F861D3C3606FA
02 reading 2 akyildiz++ wireless sensor networks - a survey .pdf	271.921	F9F3B1FA5EE53D8408C5A4A8AB6C4DBB
02 reading 3 tseng++ location awareness in ad hoc wireless mobile networks.pdf	497.870	6A357592579A740219EDDEC02143C61C
03 Localization.pdf	1.341.909	3EE3B459829B718BABC9408F61040559
03 reading 1 hightower+ borriello - a survey and taxonomy of location systems.pdf	1.950.443	2A22258A4DE04FC274A27365EAD2F661
03 reading 2 indulska+ sutton - location management in pervasive systems.pdf	230.444	664620B29A02919670B9B744A800E0CF
03 reading 3 hazas+scott +krumm - location-aware computing comes of age.pdf	91.538	B7961BCA19C0A77F23958148D8D763D1
03 supplementary 4 dobson - a taxonomy for thinking about location.pdf	111.939	9E1F59F91A51D86FD63D5B5ECB1F8332
04 identifikation.pdf	3.845.481	EEFC8587B330B2DA57D7B24F5D070FDC
04 reading 1 sarma+weis+ engels - rfid systems and security and privacy implications.pdf	401	8386A4247CD7EA6963D47E45D38B1619

04 reading 2 roemer+schoch +mattern+duebendorfer - smart identification frameworks for ubiquitous computing applications.pdf	401	8386A4247CD7EA6963D47E45D38B1619
04 reading 3 orr+abowd - the smart floor - a mechanism for natural user identification and tracking.pdf	401	8386A4247CD7EA6963D47E45D38B1619
05 GPS_Timing.pdf	320.480	EA2656F7B4682B9CDEBF0C95BDAD31A2
05 Wireless Sensor Networks.pdf	2.233.089	133DE93B2875B073FDC925AB26BC7A63
05 reading 1 akyildiz2001 _wireless sensor networks_a survey.pdf	401	8386A4247CD7EA6963D47E45D38B1619
05 reading 2 cerpa2001_ habitat monitoring.pdf	401	8386A4247CD7EA6963D47E45D38B1619
05 reading 3 ewps2006_ platform survey.pdf	401	8386A4247CD7EA6963D47E45D38B1619
05 reading 4 haensel2006_ sensor networks.pdf	401	8386A4247CD7EA6963D47E45D38B1619
05 reading 5 inta2000_ directed diffusion.pdf	401	8386A4247CD7EA6963D47E45D38B1619
05 reading 6 krish2002_ critical density.pdf	401	8386A4247CD7EA6963D47E45D38B1619
06 a-survey-of-context.pdf	401	8386A4247CD7EA6963D47E45D38B1619
06 context.pdf	1.219.882	E97A66F6BFD6798D0C027A340B43EEF6
06 reading1 dey+abowd a conceptual framework.pdf	401	8386A4247CD7EA6963D47E45D38B1619
06 reading2 dey understanding and using context.pdf	401	8386A4247CD7EA6963D47E45D38B1619
06 reading3 chen+kotz a survey of context aware.pdf	401	8386A4247CD7EA6963D47E45D38B1619
07 displays.pdf	6.020.327	6FDB1691D70998B93CBC0A0A71F23E00
07 reading 1 Molyneaux+ Kortuem - Ubiquitous Displays in dynamic environments - Issues and Opportunities.pdf	401	8386A4247CD7EA6963D47E45D38B1619

07 reading 2 plaue+miller+ stasko - is a picture worth a thousand words - evaluation of information awareness displays.pdf	401	8386A4247CD7EA6963D47E45D38B1619
07 reading 3 ferscha+vogl - the webwall.pdf	401	8386A4247CD7EA6963D47E45D38B1619
08 wearables.pdf	4.291.758	2F97299A50204C55440334EB211596F3
09 PrehensileMovement-sOfTheHumanHand Napier 1956.pdf	401	8386A4247CD7EA6963D47E45D38B1619
09 reading1 fishkin taxon-omy.pdf	401	8386A4247CD7EA6963D47E45D38B1619
09 reading2 sharlin et al TUIs humans spatiality.pdf	401	8386A4247CD7EA6963D47E45D38B1619
09 reading3 shaer et al TAC paradigm.pdf	401	8386A4247CD7EA6963D47E45D38B1619
09 tangible.pdf	2.555.586	ED6FDAAD4EF3EF09E42A6C4C63BFD95E
0x brain_computer_interfa-ces_21nov2006_ersetzt_WS22 nov2006.pdf	52.054	A43ECA169C411E02F3F5C90E2D534C55
Unterlagen.txt	17	7CB67784B0026390CA15F31E01FF9BC1
Vorbesprechung2006.pdf	672.114	BD09C9378C6D6F673B80A09F89F548AB

Ordner: neue Bilder

Dateiname	Größe in Bytes	MD5 Prüfsumme
Image050.jpg	504.864	6C455C48DED3ABA6C275BA38D66F3835
Image051.jpg	460.033	4505C879A31416AF1BDD858F888A28DF
Image052-bearbeitet.jpg	1.224.896	CB6A72603C198555FD9AA52FF9C9A621
Image053.jpg	390.427	B4C8F57484C357B811C6FB56C4B001C6

Ordner: neue Dokument\ KV Requirements Engineering\

Dateiname	Größe in Bytes	MD5 Prüfsumme
RE-JKU-0.pdf	235.513	B0FCBE305124C4409D2106542D9660D2
RE-JKU-1a-RESpiral.pdf	107.838	9E8586F85D8FC7C3AC47D996B37CF702
RE-JKU-1b-WinWin.pdf	1.520.704	3DCDE14135D1FA40B5C63909CFA71D28
RE-JKU-1c-TypesOfRe-	86.558	D17485EBCCF9D89F53817FA6ADD4BF76

quirements.pdf		
RE-JKU-2a.pdf	724.578	C677BCD144EE0FF36BE14441F89D8C7E
RE-JKU-2b.pdf	743.825	9165A0729E4E680D8C25FC4F1F690BDC
RE-JKU-3.zip	4.455.068	17C2A71FC01492B3A501EC241729E622
RE-JKU-4.zip.zip	2.302.519	70C737D32C143DB937F484364C1C28EF
SeCSE@Linz.pdf	4.531.981	7C72152C7929905727C409DC5F777BB3
template.pdf	848.787	73D4F61BC3856485F7A4A01FE825098F

Ordner: neue Dokument\ KV Requirements Engineering\RE-JKU-3

Dateiname	Größe in Bytes	MD5 Prüfsumme
01 Volere-Template.pdf	889.310	85007F2EFA790520693CED29C45308FF
ASERegularPaper#7_ Neumueller_ Christian.pdf	268.058	BAA71660DF59AE1DB9F7011D36B5698B
RAS-TUWIEN-3a.pdf	206.813	02CBBD0D8C0D6D050645F4EA1017CE2A
RAS-TUWIEN-3b.pdf	39.654	AD5F74C25692A535C6244FBEA6D73B97
RAS-TUWIEN-3c.pdf	1.210.831	A9DC94FC0F575E19DCCC842CA4B3FFD6
RAS-TUWIEN-3d.pdf	1.630.939	814B50F1A5519DEDE489FE6E04138A46
RAS-TUWIEN-3e.pdf	729.571	87AD030782034D2E5DC8264E7FA2EDD0

Ordner: neue Dokument\ KV Requirements Engineering\RE-JKU-4.zip

Dateiname	Größe in Bytes	MD5 Prüfsumme
ASE Halling Gruenbacher Biffl.pdf	325764	9EDC3A571CF95EDD4C848CE94B9A0CC2
DetReadTechnDtCBR001021.doc	40448	299E2C1DF3A451AC3CABEC8DA84CAA85
DetReadTechnDtDsgrTut001021.doc	37888	53B61BF6FE0E74AEA4BFDDEB309F51D3
DetReadTechnDtTstrTut001021.doc	41472	9F8B673DD4895B6F4E860A1204911637
DetReadTechnDtUserTut001021.doc	38400	AFE65100C9628E8C0D9FC74B63965FFF
REundPL.pdf	2491538	17453CBB6655B1EEA083E445B7B76A8E
Vortrag-Xavier-Franch-Dez18-JKU.pdf	31396	DA21DBDFBDE5D55B708C8048B2E62F4E

Ordner: neue Dokument\ KV Sicherheitsaspekte in der Informatik

Dateiname	Größe in Bytes	MD5 Prüfsumme
00_Basics.pdf	331405	C53E7C07667220B371E8101E3DE38DCB
00_Basics_notes.pdf	299286	C6D6CD1D8D31F8F54365AF62B7A70745
01_Einleitung.pdf	741924	3AE36442333F1FA33BD8DE30DAD2C24C
01_Einleitung_notes.pdf	602442	4C5B5247B347A37ADFE410E4A2F11BFC

02_Security_Policy.pdf	840505	6F7B10E2D6A918EE03D39B58C92825DA
02_Security_Policy_notes.pdf	719787	55174C18D4AAC2A7B239EC1CCD111B85
03_Network_Sniffer.pdf	430293	20BFECE28576AC7219B6CB576A713957
03_Network_Sniffer_notes.pdf	375160	B00D1BA0EF3318616607D3783658EAA6
04_Logging.pdf	415764	82DD34C103D1D05C0260184E36EA0756
04_Logging_notes.pdf	368177	F944A0FC53A50BF86047D1A19F5BDFF0
05_Inside_the_Fence.pdf	608406	FCD892DDE93C50E4CA5AACE834512221
05_Inside_the_Fence_notes.pdf	513020	7650308EA9230224E266B2A338BF4F30
06_Encryption_Basics.pdf	641631	498E1A5A47DA4A79AAC6CE3C66D4103B
06_Encryption_Basics_notes.pdf	601984	9DF433017CF2BA52E24C6A4E2FA3CA91
07_Authentication.pdf	1208985	CCBC410A54384EA709BFBC92ED6658B4
07_Authentication_notes.pdf	1055744	2251F3DC72D4D634B813BB4FFFD1A16A
08_PKI.pdf	1054489	93A7089A59DB8F2324CFA1AA5F92721F
08_PKI_notes.pdf	946035	D649A023B0BCA2CA517F5BCBDC7D089C
09_VPN.pdf	1130955	7C5AFCC1A0A0FA9D8E7ED3B14584C6F4
09_VPN_notes.pdf	996444	FB44EBA4A90057AD31090D81AB2004DC
10_Firewalls.pdf	2259952	17E8E35470927C7056D5622420B4734C
10_Firewalls_notes.pdf	2062693	6A1ADF8670555C945F646E180C110C02
11_Content_Analysis.pdf	1079142	6B3725BFEF6AE410B76B6D7AA9412534
11_Content_Analysis_notes.pdf	979960	D70F8BFDE59108B657E737760D5EC7CF
13_Secure_Client_and_Server.pdf	664188	4D1ED2228E42A0B4CFDD171229534A35
13_Secure_Client_and_Server_notes.pdf	690505	119DBE958ECAA84FA140C9F1D718B02E
14_Testing_Security.pdf	1029364	98516849FC20C7B1E74D60D1B1DBC815
14_Testing_Security_notes.pdf	894289	AAE6A5843B35792105B8A50989EA64C9
15_Availability.pdf	1391762	68540E3006FB080D4E15E83094EDD8DF
16_Security_Control.pdf	917470	92D05529880B8F06A4FC94F705BDBC84

Ordner: neue Dokumente\ KV Sicherheitsaspekte in der Informatik\12_IDS (Kofler)

Dateiname	Größe in Bytes	MD5 Prüfsumme
12b_IDS_Executive-View.pdf	203064	0AFA7EB6A74C220BD389BA486F5BFA03
12b_IDS_Executive-View_notes.pdf	193585	293E7AA0B4E79236B344CDF317FE84B1
12c_IDS_Admin-View.pdf	2888255	3A0F50C6F3C68255680D1BE3C8EA9502
12c_IDS_Admin-View_notes.pdf	2174543	C90BCCC9648CA20760AF629911C2B3F3
12d_IDS_Admin-View_FAQs.pdf	154155	8428BAAC0BE6B5DD7770ACDD1F978D23
12e_IDS_Expert-View.pdf	502665	B0B9F452DAF4B36645A01513BBA088DC
12e_IDS_Expert-View_notes.pdf	423861	E0BED097341932C34AE746B44A1B5632
12g_IDS_Literatur.pdf	397393	EC17F82E8D2414CABB30D11E78593527

Ordner: `neue Dokument\Netzwerkadministration`

Dateiname	Größe in Bytes	MD5 Prüfsumme
NetAdm_c_Server_versus_Workstation Linz.pdf	195598	8D287EEEB85FFAAE14305A19F4B7818C
NetAdm_d_Workstation Linz.pdf	1634160	E5692468A5026C9983EFD3B06B0A5E08
NetAdm_e_Installation Linz.pdf	476829	D483F6997FB21530ED8B40B88DCC7A51
NetAdm_f_MMC_Eventlog_Perf Linz.pdf	677811	B7DB9EB5A131EC19F9815C73C6E7C239
NetAdm_g_Diskadmin Linz.pdf	540072	7A7C0BCA6EE98C918E628B03EC08ECFA
NetAdm_h_UserManagement0 Linz.pdf	2231049	C845CD392F609AE3162ADAD9937A9972
NetAdm_i_NTFS Linz.pdf	1501695	18DB50EC8947445B97F980DB5398432D
NetAdm_j_Printing STP.pdf	750913	53C27451FE9D626A47C9D74E01D38CAD
NetAdm_k_Shares Linz.pdf	836034	A1C2B66A4FB8A345CAB4A1F2ED44F54D
NetAdm_l_ADS.pdf	897738	1BC450F278DBD5997ED97282C6555C2B
NetAdm_m_Mailserver.pdf	1458868	4CE25C8FA1CCAB5D0FC619328301B706
NetAdm_Mailserver.pdf	1458868	4CE25C8FA1CCAB5D0FC619328301B706
NetAdm_n_DHCP.pdf	1306896	5CC17C78F3E01238F02876A07620551C
NetAdm_o_DNS (Netzwerke).pdf	876533	4D1193778B6CB27124A23B6D998124D9
NetAdm_o_DNS.pdf	939640	943516FCB381974EE5472C0300478877
NetAdm_p_IIS.pdf	739469	D644187B044F6AC09247D0F66B5BEE49
NetAdm_q_ActiveContent.pdf	1502990	83CA66ABBF3EF0A2E428F48845F2534A
NetAdm_r_SNMP.pdf	1081460	B122DE8ACDED64012264FDC23A4C27F8

B. Lebenslauf / Curriculum Vitae

PERSÖNLICHE INFORMATION

Thomas Weisshaar BSc
Seegasse 4
4864 Attersee
tweiss@gmx.at

AUSBILDUNG

1988 – 1992	Volksschule	Attersee
1992 – 1996	Hauptschule	St. Georgen i. A.
1996 – 2001	HTL Fachrichtung Elektrotechnik, Umwelttechnik und Leistungselektronik	Braunau a. Inn
2001 – 2008	Diplomstudium Informatik	Uni Linz
2008	Bachelorstudium Informatik	Uni Linz
ab 2008	Masterstudium Informatik	Uni Linz

BERUFSERFAHRUNG

Juli – August 1998 Energie AG Vöcklabruck
Ferialpraktikant, Abteilung für Fernwärme

Juni – August 2000 Ebewe Ges. m. b. H. Nfg. KG Unterach a. A.
Ferialpraktikant, Werkstätte Verpackung

Juli – August 2002 Stern & Hafferl, Atterseeschifffahrt Attersee
Matrose

Juli – August 2003 Stern & Hafferl, Atterseeschifffahrt Attersee
Matrose

Juli – August 2004 Stern & Hafferl, Atterseeschifffahrt Attersee
Matrose

September 2004 Ebewe Ges. m. b. H. Nfg. KG Unterach a. A.
Ferialangestellter IT-Abteilung

Juli – August 2005 Stern & Hafferl, Atterseeschifffahrt Attersee
Matrose

Juli – September 2006 Stern & Hafferl, Atterseeschifffahrt Attersee
Matrose

WS 2006 Institut für Informationsverarbeitung und Mikroprozessortechnik FIM, Uni Linz, Tutor für C++ Praktikum	Linz
Juni – September 2007 Stern & Hafferl, Atterseeschifffahrt Matrose	Attersee
Seit Juli 2008 Dr. Bernhard Harich, IT-Betreuung	Bad Ischl

C. Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Masterarbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt bzw. die wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht habe.
Des weiteren versichere ich, dass ich diese Masterarbeit weder im In- noch im Ausland in irgendeiner Form als Prüfungsarbeit vorgelegt habe.

Thomas Weisshaar