Seminar

Betriebssysteme

NTFS

Informatik

Sommersemester 2006

Inhalt

Einleitung – das „neue“ Dateisystem

NTFS steht für New Technology File System. Dieses Dateisystem wurde mit Windows NT eingeführt. Gegenüber dem alten bis dahin verwendeten Dateisystem für Windows (FAT) bietet NTFS einige erhebliche Vorteile.
Das Dateisystem wurde vollständig neu entwickelt und ist nicht eine Erweiterung oder Abwandlung eines schon bekannten Dateisystems.
Ein Dateisystem ist (nach Definition von Microsoft): Ein Teil des Betriebssystems, was bestimmt, wie Dateien benannt, gespeichert und organisiert werden. Ein Dateisystem managed Dateien und Ordner und die Informationen die benötigt werden, diese zu lokalisieren und auf diese zuzugreifen.
Die alten Systeme MS-Dos, Windows 95, 98 und ME arbeiten ausschließlich mit FAT. Alle neueren Microsoft Betriebssysteme kommen mit NTFS, aber auch noch mit dem alten FAT (File Allocation Table) Dateisystem zurecht. Sie können auch parallel mit unterschiedlichen Dateisystemen arbeiten. NTFS wird ab Windows NT 4.0, Windows 2000, Windows XP unterstützt.
Zu den Vorteilen von NTFS gehören lange Dateinamen, erweiterte Dateiattribute und sehr große Speicherkapazitäten (bis zu 2^{64} Byte – also etwa 17 Milliarden GByte). Laut Microsoft arbeitet das System zuverlässiger und erhöht die Sicherheit. NTFS bietet die Möglichkeit der Datenkompression, der Datenverschlüsselung und teilweise der Datenwiederherstellung.
Das System ist ein großes komplexes Gebiet, und so kann hier nur ein erster Eindruck vermittelt werden.

Grundlegende Konzepte

NTFS verwendet 64-Bit-Adressen und kann deswegen, wie schon eben erwähnt 2^{64} Byte große Partitionen verwalten. Es gibt keine Beschränkungen auf die Größe einer Datei oder eines Verzeichnisses. Diesen Vorteil merkt man zum Beispiel schon, wenn man selbst Filme schneidet, denn FAT erlaubt nur eine maximale Dateigröße von 4GB.
Die Dateinamen werden bei NTFS im Unicode angegeben, wodurch es möglich ist, dass auch in Griechenland, Japan, Russland, usw. Dateinamen in der Landessprache verwendet werden können.
Theoretisch unterstützt NTFS auch die Unterscheidung von Groß- und Kleinschreibung, allerdings unterstützt das Win32 API (Application Programming Interface) diese Unterscheidung nicht. Dateinamen dürfen maximal 255 Zeichen lang sein und vollständige Pfadangaben 32767 Zeichen.
Bei Unix oder FAT sind Dateien lediglich lineare Sequenzen von Bytes. Im NTFS bestehen Dateien aus mehreren Attributen. Jedes Attribut wird durch einen so genannten Bytestrom repräsentiert. Die Namen der Ströme setzen sich aus dem Dateinamen, einem Doppelpunkt und dem Stromnamen zusammen *datei:strom1*. Die meisten Dateien haben nur kurze Ströme wie den Dateinamen und einen langen Strom, der die Daten enthält.
Das Prinzip der Verwendung mehrerer Ströme ist von Apple Macintosh abgeschaut. Bei Apple haben Dateien immer zwei Ströme: einen für Daten und den anderen für Ressourcen. NTFS Server sind dadurch auch in der Lage Macintosh-Clients zu bedienen.
Das Multiple-Strom-Prinzip hat aber auch noch weitere Vorteile. Ein Bildbearbeitungsprogramm kann einen Strom für die Hauptdaten des Bildes verwenden und einen anderen für eine kleine Vorschau.
Ein Strom kann maximal 2^{64} Byte lang sein. Das ist sehr lang. Binär ausgedruckt mit einer Zeichenbreite von einem Millimeter, ergäbe sich eine Länge von 15 Lichtjahren.

Wie bei Unix handelt es sich bei NTFS auch um ein hierarchisch aufgebautes Dateisystem, welches in verschiedene Ebenen eingeteilt (Festplatte, Partitionen, Ordner, Unterordner, Dateien) ist. Man muss allerdings beachten, dass nicht wie bei Unix der Slash „/“ sondern der Backslash „\“ als Separator der einzelnen Ebenen benutzt wird.

API Aufrufe

API steht für Application Programming Interface. Die API ist eine Programmierschnittstelle, die von einem System anderen Programmen zur Verfügung gestellt wird, um diese an das System anzubinden.
Die Win32-API-Funktionsaufrufe sind den Pendants von Unix sehr ähnlich.

Win32-API-Funktion	**Unix**	**Beschreibung**
CreateFile	open	Erzeugen od. Öffnen einer Datei, Rückgabe eines Handles
DeleteFile	unlink	Löschen einer Datei
CloseHandle	close	Schließen einer Datei
ReadFile	read	Daten aus Dateien einlesen
WriteFile	write	Daten in Dateien schreiben
SetFilePointer	lseek	Lesezeiger an bestimmte Position der Datei setzen
GetFileAttributes	stat	Dateiattribute ermitteln
LockFile	fcntl	Bereich einer Datei gegen mehrfachen Zugriff sperren
UnlockFile	fcntl	Gesperrten Bereich wieder freigeben

Die Aufrufe am Beispiel *CreateFile* erklärt: *CreateFile* wird benutzt um eine neue Datei zu erzeugen. Der Aufruf liefert einen Handle darauf zurück. Ein Handle ist ein Ergebnis eines Win32-API-Aufrufs, und kann dazu verwendet werden Operationen auf das Objekt auszuführen, welches durch den Aufruf erzeugt wurde. *CreateFile* muss auch verwendet werden um schon existierende Dateien zu öffnen.
Den Aufrufen können noch verschiedene Parameter mit übergeben werden. Auch hier möchte ich nur kurz die sieben Parameter von *CreateFile* nennen:

1. Ein Zeiger zum Namen der zu erstellenden bzw. zu öffnenden Datei
2. Flags mit Informationen darüber, ob die Datei gelesen und/oder beschrieben werden kann
3. Flags mit Informationen darüber, ob mehrere Prozesse die Datei gleichzeitig öffnen können
4. Ein Zeiger zum Sicherheitsbezeichner mit Informationen, wer auf die Datei zugreifen kann
5. Flags mit Informationen darüber, was zu tun ist, wenn die Datei existiert, bzw. nicht existiert
6. Flags in Bezug auf Attribute, z.B. Archivieren, Komprimieren usw.
7. Das Handle einer Datei, deren Attribute für die neue Datei geklont werden sollen

Dies sind nur die wichtigsten Aufrufe. Es gibt noch erheblich mehr.

Die für die Verzeichnisstruktur zuständigen wichtigen Aufrufe sind:

Win32-API-Funktion	**Unix**	**Beschreibung**
CreateDirectory	mkdir	Neues Verzeichnis erstellen
RemoveDirectory	rmdir	Leeres Verzeichnis löschen
FindFirstFile	opendir	Verzeichnis öffnen und Einträge auslesen
FindNextFile	readdir	Nächsten Verzeichniseintrag lesen
MoveFile	rename	Datei von einem Verzeichnis in ein anderes verschieben
SetCurrentDirectory	chdir	Aktuelles Arbeitsverzeichnis wechseln

Hierzu muss eigentlich nicht viel gesagt werden.

Realisierung des Win-2000-Dateisystems

Struktur des Dateisystems – MFT

Eine Festplatte kann unterteilt werden in einzelne Partitionen. Diese können auf das NTFS-Format formatiert werden. Jede Partition enthält dann Dateien, Verzeichnisse, Bitmaps und weitere Datenstrukturen. Mit Bitmaps werden hier Folgen von voneinander unabhängigen Bits bezeichnet, die jeweils die gleiche Bedeutung haben. Zum Beispiel ein „Lageplan“, welche Bereiche des Volumes belegt sind.
Jedes Volume ist eine lineare Anordnung von Blöcken (auch Cluster genannt). Dies Cluster haben eine Größe von 512 Byte bis zu 64 KB. Als Kompromiss zwischen großen Blöcken (effizienter Datentransfer) und kleinen Blöcken (geringe Fragmentierung) haben die meisten Volumes eine Clustergröße von 4 KB.
Das Herz des NTFS ist die so genannte Master File Table MFT. Diese Datenstruktur ist eine lineare Sequenz von Clustern, welche 1KB groß sind (unabhängig von der Clustergröße des restlichen Volumes). Die MFT ist im Grunde eine Datenbank, deren Einträge eine Datei oder ein Verzeichnis auf dem Volume beschreiben. Die Einträge bestehen aus den Attributen der Datei (Name, Zeitstempel) und eine Liste von Clusteradressen, wo die Datei gespeichert ist. Sollte eine Datei so groß sein, dass die Clusteradressen nicht alle in einem MFT Eintrag gespeichert werden können, werden weitere Einträge verwendet, und der erste Eintrage (Base-Record) zeigt auf die anderen.

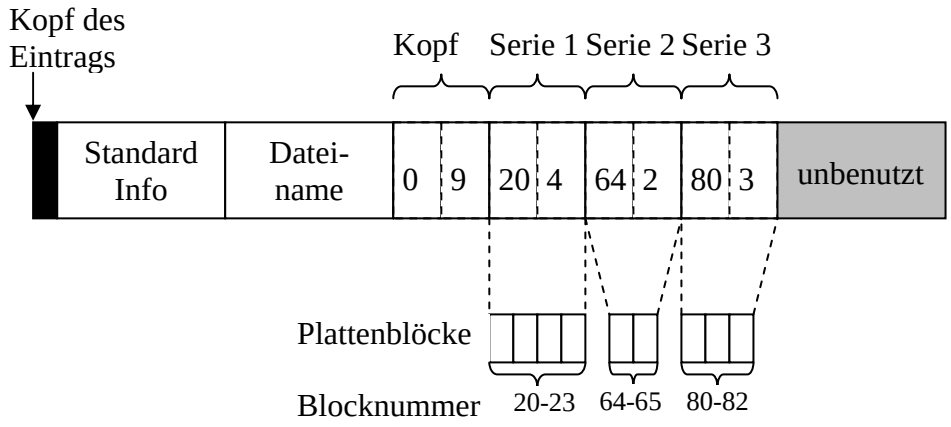
Die MFT ist selbst eine Datei und kann irgendwo auf dem Volume stehen. Der Aufbau der MFT sieht wie folgt aus:

Systemdatei	Dateiname	MFT-Record	Beschreibung
Master File Table	$Mft	0	Beschreibt MFT selbst, insb. wo die Blöcke der MFT gespeichert sind
Master File Table Kopie	$MftMirr	1	Sicherheitskopie des vorderen Teils der MFT
LogFile	$LogFile	2	Strukturelle Änderungen werden erst eingetragen, dann ausgeführt. Wiederherstellbar bei fehlerhafter Ausführung
Volume	$Volume	3	Enthält Informationen über Volume-Größe, Bezeichnung usw.
Attribut Definitionen	$AttrDef	4	Definitionen der Attribute
Wurzelverzeichnis	$	5	Wurzelverzeichnis
Cluster Bitmap	$Bitmap	6	Gibt an, welche Cluster frei oder belegt sind
Boot-Sektor	$Boot	7	Bootcode des Betriebssystems
Schlechte Cluster	$BadClus	8	aussortierte schlechte Cluster, werden in Bitmap geschrieben und als unebnutzbar markiert
Sicherheitsdatei	$Secure	9	Benutzerbeschränkungen verwaltet
Upcase Table	$Upcase	10	Verwaltet eine Liste von 40000 Unicodezeichen
Erweiterungen	$Extend	11	Verschiedene Dateien mit Informationen über Plattenkontingente, Wiedereinstiegspunkte usw.
		12-15	Reserviert falls in Zukunft benötigt
		16	erste Benutzerdatei

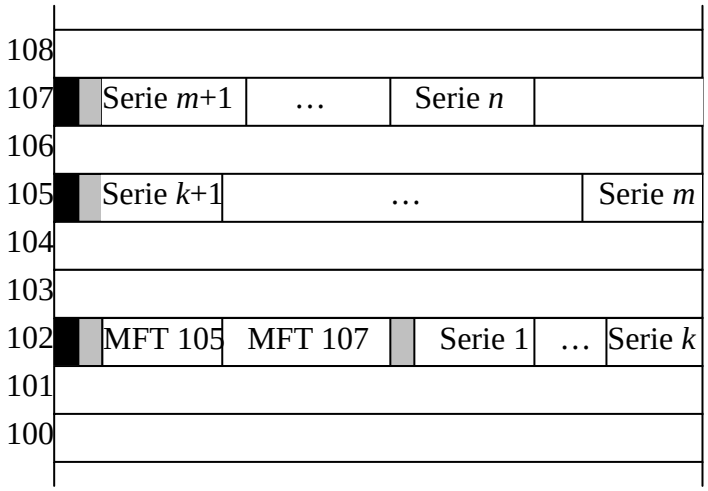
Jeder dieser ersten Einträge beschreibt eine normale Datei mit Attributen und Datenblöcken, so wie jede andere Datei auch. Um anzuzeigen, dass es sich um Metadaten (als **Metadaten** bezeichnet man allgemein Daten, die Informationen über andere Daten enthalten) handelt wird vor den Dateinamen ein $-Zeichen gesetzt.
Jeder Eintrag in der MFT besteht aus einem Eintragskopf und einer Sequenz von (Attributkopf, Attributwert)-Paaren. Die Attribute müssen in einer festen Reihenfolge in der MFT stehen.
Es gibt 13 Attribute die von NTFS vordefiniert sind. Diese sind:

Attribut	Beschreibung
Standardinformation	Zeitstempel, Informationsbits usw.
Dateiname	Dateiname in Unicode
Security-Deskriptor	veraltet – Informationen jetzt unter $Extend$Secure
Attributliste	Ort zusätzlicher MFT-Einträge (wenn nötig)
Objekt-ID	64 Bit Dateiidentifizierung pro Volume
Reparse	Für Montieren und symbolische Links
Volumenname	Name des Volumes
Volumeninformation	Volumenversion
Indexwurzel	Für Verzeichnisse genutzt
Indexallokation	Für sehr große Verzeichnisse genutzt
Bitmap	Für sehr große Verzeichnisse genutzt
Logging-Hilfssystem	Kontrolliert das Logging in $LogFile
Daten	Datenstrom

Bei dem Datenattribut kennzeichnet der Attributwert die Anzahl der Cluster, die diese Datei beschreiben. Der Attributkopf enthält Sicherheitsbeschreibungen und Datentyp.
Ein MFT Eintrag für eine Datei sieht zum Beispiel folgendermaßen aufgebaut:

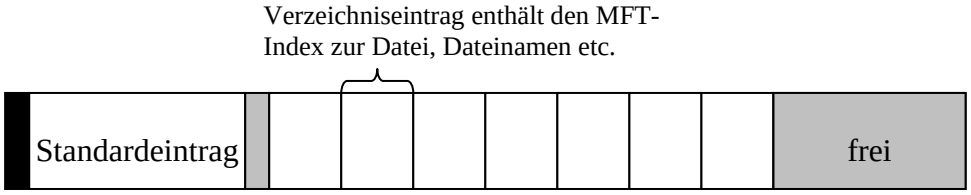


Hinter dem Dateinamen gibt es den Kopf, welcher das Offset angibt und die Anzahl der zur Datei gehörenden Cluster. Hinter dem Kopf kommen die Serien, die beschreiben, in welchen Clustern der Partition die Dateifragmente gefunden werden. Die Anzahl der Serien hängt davon ab, wie gut der Disk-Block-Allocator beim Anlegen der Datei gearbeitet hat. Im schlimmsten Fall kann eine Datei mit *n* Blöcken *n* Serien haben. Wenn keine Blöcke gespeichert werden sollen (wie im Kopf) wird der erste Eintrag gleich Null gesetzt. Dies dient dazu zu beschreiben, wie viele Blöcke (unkomprimiert) benötigt werden.
In den meisten Fällen passen natürlich nicht die gesamten Daten in den Eintrag des Datenattributs. Geschieht dies, werden mehrere MFT-Einträge verwendet. Besteht eine Datei also aus zu vielen Serien um in einen Eintrag zu passen, wird berechnet, wie viele Erweiterungseinträge benötigt werden. Die Indizes dieser werden dann in den Basiseintrag geschrieben. Der restliche Platz im Basiseintrag wird schon für die ersten Serien verwendet.



Sollte der Fall auftreten, dass noch nicht mal alle Indizes der Erweiterungseinträge in einen Eintrag passen, dann wird die Liste der Indizes nicht in der MFT gespeichert, sondern ausgelagert auf der Platte gespeichert.

Ein MFT-Eintrag für ein kleines Verzeichnis – in Microsoft-Sprache ein Ordner – besteht aus einzelnen Einträgen, die den MFT-Index zu einer Datei oder einem Unterverzeichnis, die Länge des Dateinamens, den Dateinamen und weitere Felder enthalten. Die einzelnen Einträge haben eine feste Struktur und Länge. Der Dateiname ist allerdings variabel.
Das Suchen nach einem Eintrag in einem Verzeichnis besteht darin, die Dateinamen der Reihe nach zu untersuchen.

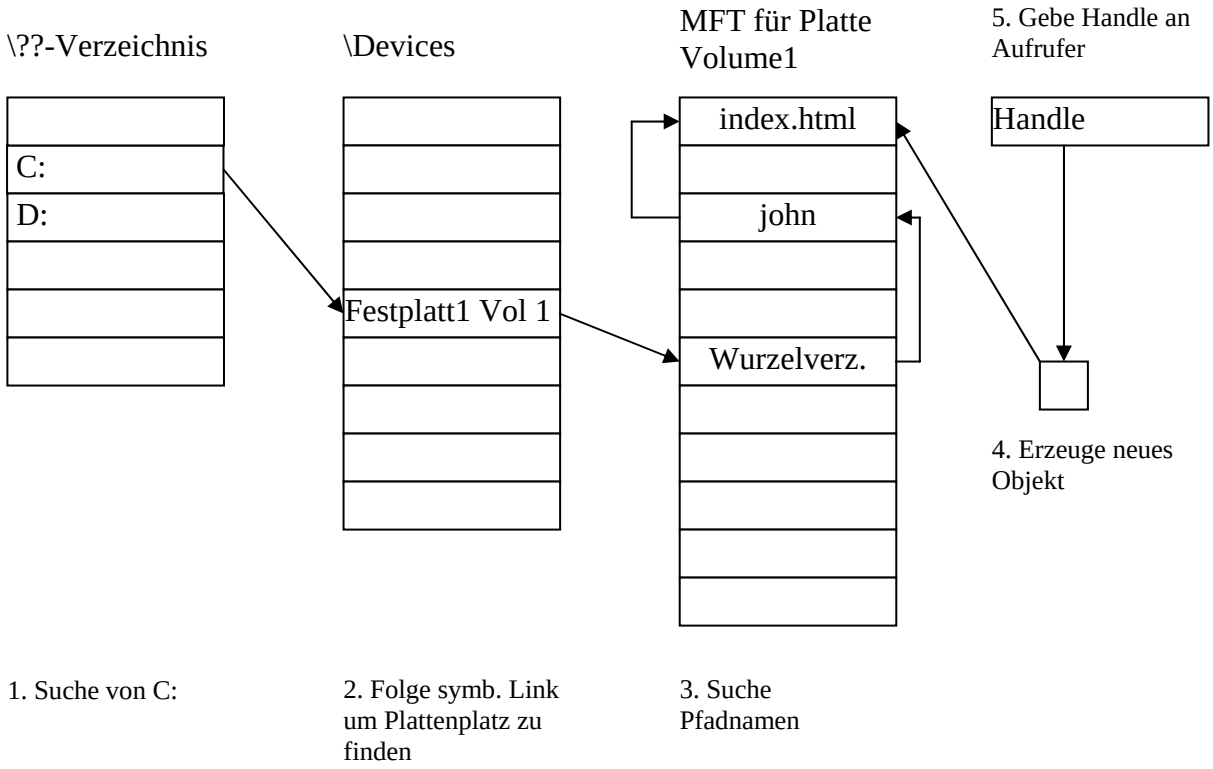


Für große Verzeichnisse wird ein anderes Format verwendet. Die Dateien werden dann nicht linear sondern in einem Baum aufgelistet. Hierdurch wird das Einfügen neuer Namen und Dateien an der richtigen Stelle ermöglicht.

Dateinamensuche

Wenn ein Programm nun auf eine Datei zugreift, macht es einen Aufruf wie `CreateFile(„C:\john\index.html“,…)`
Dieser Aufruf wird dann weitergeleitet an die Bibliothek *kernel32.dll*. Dort wird *\??* vor den Namen gehängt. Somit ergibt sich der Aufruf:
`\??\C:\john\index.html`
Dieser Name wird dann an den NTFS Aufruf *NtFileCreate* weitergeleitet.
Die Suche nach der Datei startet dann auf der obersten Ebene. Die Partition C: wird gefunden. Diese ist allerdings nur ein symbolischer Verweis auf einen anderen Teil des Namensraums des Objektmanagers, nämlich auf das Verzeichnis *\Device*. In diesem Verzeichnis gibt es dann *\HarddiskVolume1*, welches die erste Partition auf der Platte ist.
Nun weiß der Rechner, welche MFT verwendet werden muss.
Die Suche wird nun im Wurzelverzeichnis fortgesetzt, welches mit Hilfe des Eintrags 5 der MFT gefunden wird. In diesem Verzeichnis wird die Zeichenkette `john` gesucht. Wenn diese gefunden wurde, wird der Index auf die MFT für diesen Eintrag zurückgeliefert und es wird in diesem Eintrag weitergesucht nach der Zeichenkette „`index.html`“.

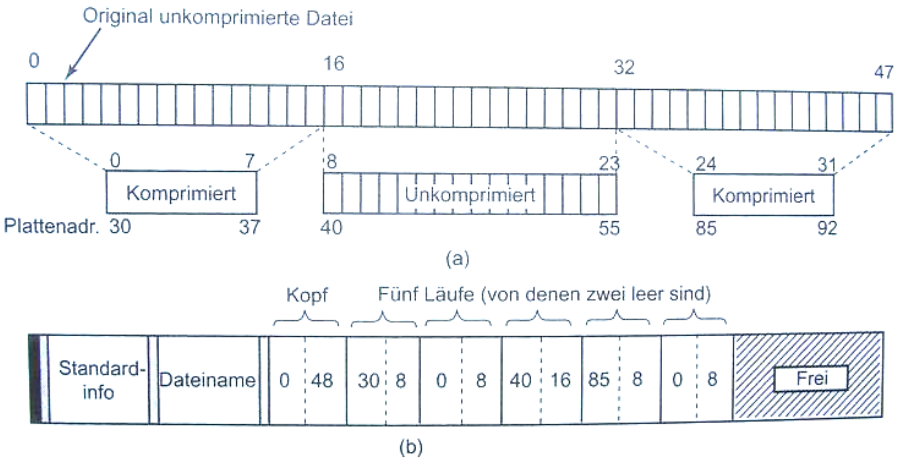
Sobald dieser Eintrag gefunden wurde, erzeugt der Objektmanager ein neues unbenanntes Objekt, welches den Index des MFT-Eintrags der Datei enthält. Das aufrufende Programm erhält dann ein Handle auf dieses Objekt.
Sollte die Datei noch von einem weiteren Thread gleichzeitig verwendet werden, wird ein weiteres Objekt erzeugt.



Dateikompression

NTFS kann Dateien im Hintergrund komprimieren. Die Prozesse die eine Datei erzeugen bzw. auf eine zugreifen bekommen von dieser Komprimierung gar nichts mit. Die Komprimierung wird im Kompressionsmodus durchgeführt und die Datei dann erst auf die Platte geschrieben, bzw. die Datei wird beim Zugriff erst dekomprimiert, bevor sie an das aufrufende Programm weitergeleitet wird.
Bei der Komprimierung werden zuerst die ersten 16 Blöcke einer Datei untersucht. Sollten sie komprimierbar sein, also in 15 oder weniger Blöcken zusammenfassbar, werden diese dann (möglichst in einer Serie) gespeichert. Ansonsten werden die ersten 16 Blöcke umkomprimiert gespeichert. Anschließend werden die Blöcke 16-31 untersucht, usw.
Ein MFT-Eintrag einer komprimierten Datei sieht ähnlich dem einer unkomprimierten Datei aus, nur dass die fehlenden Plattenblöcke mit der Adresse 0 gespeichert werden.
Wenn die ersten 16 Blöcke auf 8 komprimiert werden, heißt die Serie 1: (30,8) und Serie 2: (0,8).
Dies führt zu keiner Verwechslung, da im Plattenblock 0 keine Dateien gespeichert werden (enthält Bootsektor). Somit können die Dateien auch leicht wieder dekomprimiert werden.
Die Wahl, jeweils 16 Blöcke zu betrachten war ein Kompromiss zwischen effektiverer Komprimierung und Verlängerung des wahlfreien Zugriffs. Wenn NTFS nämlich den Block 35 einer komprimierten Datei sucht, muss zuerst alles dekomprimiert und gelesen werden, um Block 35 zu finden.

(a) Komprimierungsbeispiel einer Datei mit 48 Blöcken. (b) der MFT-Eintrag dieser Datei



Dateiverschlüsselung

Die Möglichkeit Daten zu verschlüsseln ist heutzutage sehr wichtig. Denn es gibt immer mehr sensible Daten, die wir elektronisch speichern, von denen wir nicht möchten, dass sie andere Personen in die Hände bekommen. Aber gerade dies ist häufig schnell geschehen.
Windows 2000 bietet einem die Möglichkeit, Dateien und Verzeichnisse zu verschlüsseln, so dass diese bei einem Start mit einer MS-DOS-Bootdiskette, bei Diebstahl oder dergleichen die nicht gelesen werden können.
Dateien, die in verschlüsselte Verzeichnisse kopiert werden, werden auch automatisch verschlüsselt. Die Verschlüsselung selbst übernimmt nicht NTFS sondern das so genannte EFS (Encrypting File System), das sich zwischen dem NTFS und dem Benutzerprozess befindet.
Die Dateien werden dann mit einem symmetrischen Verfahren verschlüsselt. Der angewandte Verschlüsselungsalgorithmus ist ähnlich dem DES mit einer Schlüssellänge von 128 Bit. Es können allerdings mit EFS auch andere Algorithmen verwendet werden. Jede Datei wird mit einem anderen Schlüssel verschlüsselt. Somit ist gewährleistet, dass bei Kompromittierung eines Schlüssels nur eine Datei gelesen werden kann.
Die Schlüssel müssen natürlich auch auf der Platte gespeichert werden, und zwar nicht im Klartext, sondern sie müssen ebenfalls verschlüsselt werden. Hierzu wird eine Public-Key-Verschlüsselung verwendet. Die Schlüssel werden alle in der Registry gespeichert, und jeweils mit dem öffentlichen Schlüssel des Benutzers verschlüsselt.
Zum Entschlüsseln der Dateien wird der verschlüsselte Schlüssel aus der Registry geholt und kann dann mit dem privaten Schlüssel des Benutzers entschlüsselt werden. Der private Schlüssel sollte am besten nicht auf der Platte sondern zum Beispiel auf einer eingelegten Smartcard gespeichert sein. Jedoch kommt Windows 2000 damit nicht zurecht.
Aus diesem Grund wird ein Schlüsselpaar (öffentlicher Schlüssel, privater Schlüssel) erzeugt, sobald der Benutzer das erste Mal eine Datei verschlüsseln will. Der private Schlüssel wird dann auf einer Diskette gespeichert, welche nochmals symmetrisch verschlüsselt ist. Der Schlüssel hierfür leitet sich von dem Passwort des Benutzers ab. Wenn sich der Benutzer dann später anmeldet, erhält das System den Schlüssel, um den privaten Schlüssel zu erhalten, mit welchem dann die Dateien entschlüsselt werden können. Der private Schlüssel wird für die Sitzung in den virtuellen Adressraum geschrieben und anschließend immer gelöscht.
Das ganze klingt recht kompliziert hat aber mehrere Gründe. Der Hauptgrund besteht darin, dass es viel zu aufwändig wäre, jede Datei mit einem PK-Verfahren zu verschlüsseln. Es ist sehr viel effizienter ein symmetrisches Verschlüsselungsverfahren zu verwenden, um die Daten zu verschlüsseln, und eine PK-Verschlüsselung, um die kurzen Keys zu verschlüsseln.

Quellen

- Computerarchitektur, Tanenbaum/Goodman, München 2001
- Moderne Betriebssysteme, Tanenbaum, München 2002
- Betriebssysteme, Stallings, München 2003
- Betriebssysteme, Brause, Berlin 2001
- http://technet2.microsoft.com/WindowsServer/en/Library/81cc8a8a-bd32-4786-a849-03245d68d8e41033.mspx?mfr=true
- http://www.wikipedia.de