Technische Berichte in Digitaler Forensik

Herausgegeben vom Lehrstuhl für Informatik 1 der Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) in Kooperation mit dem Masterstudiengang Digitale Forensik (Hochschule Albstadt-Sigmaringen, FAU, Goethe-Universität Frankfurt am Main)

Über den Einfluss von SSD-Technologie auf die Datenträger-Forensik

Karl Weber

31.3.2016

Technischer Bericht Nr. 9

Zusammenfassung

Ziel der dieser Arbeit ist es herauszuarbeiten, wie sich Unterschiede zwischen SSDs und HDDs auf forensische Untersuchungen und Datenwiederherstellungen auswirken. Zunächst werden anhand der technischen Grundlagen, auf denen Flashspeicher beruhen, die Besonderheiten dieser Speichertechnologie aufgezeigt. Im zweiten Schritt werden die Methoden und Funktionen vorgestellt, die innerhalb einer SSD in einem eigens dafür entwickelten Controller arbeiten, mit denen SSDs trotz dieser Besonderheiten in die Lage versetzt werden, HDDs zu ersetzen. Nach einer abschliessenden Gegenüberstellung von SSDs und HDDs, in der die wesentlichen Unterschiede in der Dateiorganisation innerhalb der Laufwerke herausgestellt werden, kann beurteilt werden, dass ohne neue Funktionen wie TRIM oder die bei SSDs weitverbreitete Selbst-Verschlüsselung sich SSDs und HDDs unter forensischen Gesichtspunkten wenig unterscheiden. Wird aber TRIM verwendet, sind gravierende Einschränkungen beim Wiederherstellen von gelöschten Dateien zu erwarten.

Entstanden als Masterarbeit im Rahmen des Studiengangs Digitale Forensik unter der Betreuung von Martin Rieger und Holger Morgenstern.

Kurzfassung

Seit einigen Jahren (ca. 2007) werden in vielen neuen Notebooks und Serversystemen Solid State Drives (SSDs) anstelle von bisher üblichen Hard Disk Drives (HDDs) eingebaut, ebenso werden oft bei Nachrüstungen bestehender Systeme HDDs durch SSDs ersetzt, beides mit steigender Tendenz. Ziel der vorliegenden Arbeit ist es herauszuarbeiten, wie sich Unterschiede zwischen der auf Flashspeicher basierenden SSD und der mit Magnetspeicherplatten arbeitenden HDD auf forensische Untersuchungen und Datenwiederherstellungen auswirken. Zunächst werden anhand der technischen Grundlagen, auf denen Flashspeicher beruhen, die Besonderheiten dieser Speichertechnologie aufgezeigt. Im zweiten Schritt werden die Methoden und Funktionen vorgestellt, die innerhalb einer SSD in einem eigens dafür entwickelten Controller arbeiten, mit denen SSDs trotz dieser Besonderheiten in die Lage versetzt werden, HDDs zu ersetzen.

Nach einer abschliessenden Gegenüberstellung von SSDs und HDDs, in der die wesentlichen Unterschiede in der Dateiorganisation innerhalb der Laufwerke herausgestellt werden, kann beurteilt werden, dass ohne neue Funktionen wie TRIM oder die bei SSDs weitverbreitete Selbst-Verschlüsselung sich SSDs und HDDs unter forensischen Gesichtspunkten wenig unterscheiden. Wird aber TRIM verwendet, sind gravierende Einschränkungen beim Wiederherstellen von gelöschten Dateien zu erwarten.

Abstract

For a few years (since around 2007), Solid State Drives (SSDs) are built into new notebooks and server systems instead of the traditionally prevalent Hard Disk Drives (HDDs). Similarly, HDDs are replaced with SSDs during retrofittings of existing systems, both with upward tendency. The aim of this work is to identify implications caused by the differences between SSDs, based on flash memory, and HDDs, based on magnet storage disks, with respect to forensic investigations and data recovery. Firstly, the special features of this storage technology are explored using the technical foundations of flash memory. Afterwards, the methods and functions working inside a specially developed controller in a SSD that enable them to replace HDDs despite those special features are introduced.

After a concluding comparison of SSDs and HDDs with focus on the significant differences in file organization inside the drives we can assess that without new functions such as TRIM or the (with SSDs commonly used) self encryption of devices the differences between SSDs and HDDs are small in the perspective of forensics. However, if TRIM is used, serious limitations are to be expected when trying to recover deleted files.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

16LC Sixteen-level Cells

8LC Eight-level Cells

AHCI Advanced Host Controller Interface

ATA AT Attachment

AT Advanced Technology, ein Formfaktor für Gehäuse und Platinen in Rechnern

BCH Bose, Chaudhuri und Hocquenheim, ein Fehlerkorrekturcode

BGA Ball Grid Array

BIOS Basic Input Output System

BL Bitline

CD Compact Disc

CF-Card CompactFlash Card, ein digitales Speichermedium

CPU Central Processing Unit

DEK Data Encryption Key

DRAM Dynamic Random Access Memory, eine Speichertechnologie mit wahlfreiem Zugriff

DVD Digital Versatile Disc

ECC Error-Correction-Code

EEPROM Electrically Erasable Programmable Read-Only Memory

Gbps Gigabytes per Second

GC Garbage-Collection

GiByte Gibibyte, $1Gib = 1024^3\ Bytes$

HDD Hard-Disk-Drive

HPA Host Protected Area

KEK Key Encryption Key

KiByte Kibibyte, $1KiB = 1024\ Bytes$

LBA Logical Block Address

MB Megabyte, $1MB = 10^6\ Bytes$

MEK Media Encryption Key

MFT Master File Table

MiByte Mebibyte, $1MiB = 1024^2 Bytes$

MLC Multi-level Cells

mSATA Mini-SATA

NAND Not-And, u.a. ein Logikgatter und eine Speichertechnologie

NOR Not-Or, u.a. ein Logikgatter und eine Speichertechnologie

NTFS New Technology File System

NVM Non-Volatile-Memory, ein nicht-flüchtiger Speicher

PBA Physical Block Address

PCB Printed Circuit Board

PCIe Peripheral Component Interconnect Express, ein Standard zur Verbindung von Peripheriegeräten mit dem Chipsatz des Computers

RAM Random-Access Memory, ein Speicher mit wahlfreiem Zugriff

SAS Serial Attached SCSI, ein Standard zur Kommunikation mit Speichergeräten, vor allem im Serverbereich üblich

SATA Serial ATA, ein Standard zur Kommunikation mit Speichergeräten

SCSI Small Computer System Interface, eine Familie von Protokollen zur Verbindung von Peripheriegeräten und Computern

SD-Cards Secure Digital Memory Card, ein digitales Speichermedium

SED Self-Encrypting Drive

SLC Single-level Cells

SSD Solid-State-Drive

TB Terabyte, $1TB = 10^{12}\ Bytes$

TiByte Tebibyte, $1TiB = 1024^4\ Bytes$

TSOP Thin Small Outline Package

USB Universal Serial Bus, ein Standard zur Kommunikation mit Peripheriegeräten

V_{TH} Threshold Voltage, eine Schwellspannung

WL Wordline

1 Einleitung

Seit nunmehr 60 Jahren, als IBM im Mai 1955 ein Gerät ankündigte, auf das ca. 5 Millionen Zeichen gespeichert werden konnten, gilt dieses „Harddisk“ (Festplatte) genannte Gerät als *das* Speichermedium in der Datenverarbeitung. Erstmals konnten Daten dauerhaft auf ein Medium geschrieben und im wahlfreien Zugriff wieder gelesen werden. Zunächst ausschließlich in Großrechnern eingesetzt, fand sie erst mit der Einführung und Verbreitung der Personalcomputer (ab ca. 1980) den Weg als Massenspeicher in den Massenmarkt.

In dieser Zeit haben sich Festplatten stark weiterentwickelt, unter anderem in den folgenden Aspekten:

- die Speicherkapazitäten wurden immer größer (Faktor $\approx 10^6$, von 5 MB[1] auf ca. 6 TB[2])
- die Formate (Baugrößen) wurden immer kleiner
- die Performance (vor allem die Datenübertragungsrate) wurde immer besser (Faktor $\approx 10^4$, von 8800 Bytes/s [3] zu 88 MiBytes/s [4]).

Was sich *nicht* verändert hat, ist die grundsätzliche Technik

- Daten auf die Festplatte zu schreiben
- sie wieder von dort zu lesen
- einmal geschriebene Daten zu ändern
- und sie schließlich zu löschen.

[1] Vgl. Chen, Ben M. / Lee, Tong H. / Peng, Kemao / Venkataramanan,Venkatakrishnan (2006): Hard Disk Drive Servo System (2nd. Edition), London: Springer, S. 5
[2] Vgl. Western Digital Corporation, , z. B. WD6001FXYZ http://www.wdc.com/wdproducts/library/SpecSheet/ENG/2879-800066.pdf [2015-10-25]
[3] Vgl. Chen, Ben M., a. a. O., S. 6
[4] Vgl. White Paper: Fujitsu PRIMERGY Servers Solid State Drives – FAQ (2014) http://docs.ts.fujitsu.com/dl.aspx?id=78858d6c-4c0f-479a-8ceb-705fe1938f4e [2015-06-06]

Die zu schreibenden Informationen werden durch unterschiedliche Magnetisierungen in einer magnetisierbaren Schicht in den einzelnen Scheiben einer Festplatte *dauerhaft* gespeichert.

Schon in den 1990er Jahren gab es Versuche, Massenspeicher auf der Basis von Halbleiterbausteinen als Alternative zu den Festplatten anzubieten[5]. Mit sogenannten Flash-EEPROM standen Chips zur Verfügung, die ebenso wie Festplatten Informationen dauerhaft (d. h. ohne Stromzufuhr) speichern können. Allerdings übertrafen die etablierten Festplatten die Flash-Speicher in den Bereichen Kapazität, Performance (hier besonders die Übertragungsrate, nicht die Latenzzeiten) und Preis noch um Längen. Erste Geräte kamen ungefähr im Jahr 2007 auf den Markt. Mehrere Firmen entwickelten Datenträger, die vor allem durch parallel[6] angeordnete Flash-Bausteine die Datenübertragungsrate von Festplatten erreichten und inzwischen auch übertreffen, bei den Zugriffszeiten sogar um mehrere Zehnerpotenzen schneller sind[7]. Durch gesunkene Preise und gesteigerte Kapazitäten stellen diese SSD (Solid-State-Drive) genannten Laufwerke nunmehr eine ernsthafte Konkurrenz für (Magnet-) Festplatten dar. Die heute am Markt verfügbaren SSD–Laufwerke sind u. a. mit den bekannten Schnittstellen (SAS, SATA) ausgestattet und in den gängigen Formaten erhältlich. Sie können sowohl in Neugeräten als auch in bestehenden Systemen im Austausch für Festplatten ohne größere Anpassungen eingesetzt werden. Um die Geschwindigkeitsvorteile von SSDs besser ausschöpfen zu können, sind SSDs vor allem für Enterprise-Anwendungen auch mit PCIe-Schnittstelle erhältlich, die den Performance-Flaschenhals, den die herkömmlichen Schnittstellen bei High-End SSDs inzwischen darstellen, durch den direkten Anschluss an den Hauptspeicher, eliminieren[8].

Als zentrale und oft auch einzige Speicherorte nicht-flüchtiger Daten in einem Rechnersystem sind Massenspeicher für forensische Untersuchungen von größter Wichtigkeit. Für diese Untersuchungen haben sich im Lauf der Jahre Vorgehensweisen etabliert, die von einer Vielzahl an Softwaretools, sowohl frei als auch auf dem kommerziellen Markt erhältlich, unterstützt werden. Dabei haben sie sich natürlich an den technischen Gegebenheiten und der Art und Weise, wie Daten auf den (Magnet-) Festplatten geschrieben, gelesen und gelöscht werden, orientiert und angepasst. Ziel der vorliegenden Arbeit ist es herauszuarbeiten, inwieweit mit der Verbreitung von SSD-Massenspeichern diese Vorgehensweisen und die damit verbundenen Werkzeuge weiterhin Gültigkeit haben bzw. an ihre Grenzen stoßen, oder ob sich durch SSDs neue Möglichkeiten für forensische Analysen ergeben.

[5] Vgl. Micheloni, R. / Marelli, A. / Eshghi. K.: Inside Solid State Drives (SSDs), Dordrecht: Springer, 2013, S. 2
[6] Vgl. Micheloni, R. / Crippa, L. / Marelli, A.: Inside NAND Flash Memories, Dordrecht: Springer, 2010, S. 38
[7] Vgl. ebenda S. 1
[8] Vgl. ebenda S. 39

Um hier Antworten geben zu können, ist es zunächst notwendig, die Funktionsweise und Architektur von SSD-Laufwerken zu verstehen und die sich daraus ergebenden wesentlichen Unterschiede zwischen herkömmlichen Festplatten und SSD-Laufwerken zu benennen. In einem weiteren Kapitel wird darauf resultierend auf die Datenrettung bzw. Datenwiederherstellung unter forensischer Sicht bei SSD-Laufwerken eingegangen. Es wird außerdem untersucht, welche Auswirkungen der Einsatz von SSD–spezifischen Funktionen wie der „TRIM“-Befehl, das „Over-Provisioning“ sowie die oft schon integrierte Verschlüsselungsfunktion darauf haben. Die Besonderheiten, die beim Löschen von SSD auftreten, werden in einem eigenen Kapitel besprochen.

Zur Vereinfachung der Schreibweise werden in den folgenden Kapiteln die Solid-State-Drives als SSD, die (Magnet-) Festplatten als HDD bezeichnet.

2 Technische Grundlagen

2.1 Aufbau von SSD-Datenträgern

Ein Flash-basierter SSD-Datenträger beinhaltet keinerlei bewegliche mechanische oder elektromechanische Teile (daher der Begriff *solid state*), er besteht vielmehr im Wesentlichen aus folgenden elektronischen Komponenten (siehe Abbildung 2.1)[1]:

- Flashspeicher
- Controller inkl. Host-Interface
- Puffer-Speicher
- passive Bauelemente (z. B. Stromversorgung, Sensoren)

Flashspeicher, auf dem die Daten einer SSD gespeichert werden, ist ein Halbleiterspeicher und gehört hierbei zu der Gruppe der sogenannten „Non-Volatile Memories“ (Nicht-flüchtige Speicher, NVM), die ihren Inhalt auch beim Wegfall der Stromversorgung nicht verlieren, im Gegensatz z. B. zu RAM-Speicher. Je nach interner Organisation der Speicherzellen innerhalb des Flashspeichers kann zwischen NAND- und NOR-Flashspeicher unterschieden werden. Aufgrund der höheren Zugriffsgeschwindigkeit und der linearen Adressierbarkeit bei den NOR-Flashspeichern eignen diese sich u. a. als Speicher von Boot-Code. In SSDs und anderen Flash-basierten Massenspeicherprodukten (SD-Cards, USB-Sticks, CF-Cards, usw.) kommen wegen ihrer wesentlich höheren Speicherdichte ausschließlich NAND-Flashspeicher zum Einsatz[2].

Der Controller ist das Bindeglied zwischen den Flashspeichern und dem Host, in dem die SSD als Massenspeicher eingesetzt wird und stellt dabei für beide Seiten geeignete Schnittstellen zur Verfügung. Des weiteren regelt er das Datenmanagement der Flashspeicher innerhalb der SSD[3]. Der Controller mit seinen Aufgaben wird in Abschnitt 2.1.2 detailliert betrachtet.

[1] Vgl. Micheloni, R. / Crippa, L., a.a.O., S. 39
[2] Vgl. Micheloni, R. / Marelli, A., a.a.O., S. 79
[3] Vgl. ebenda S. 26

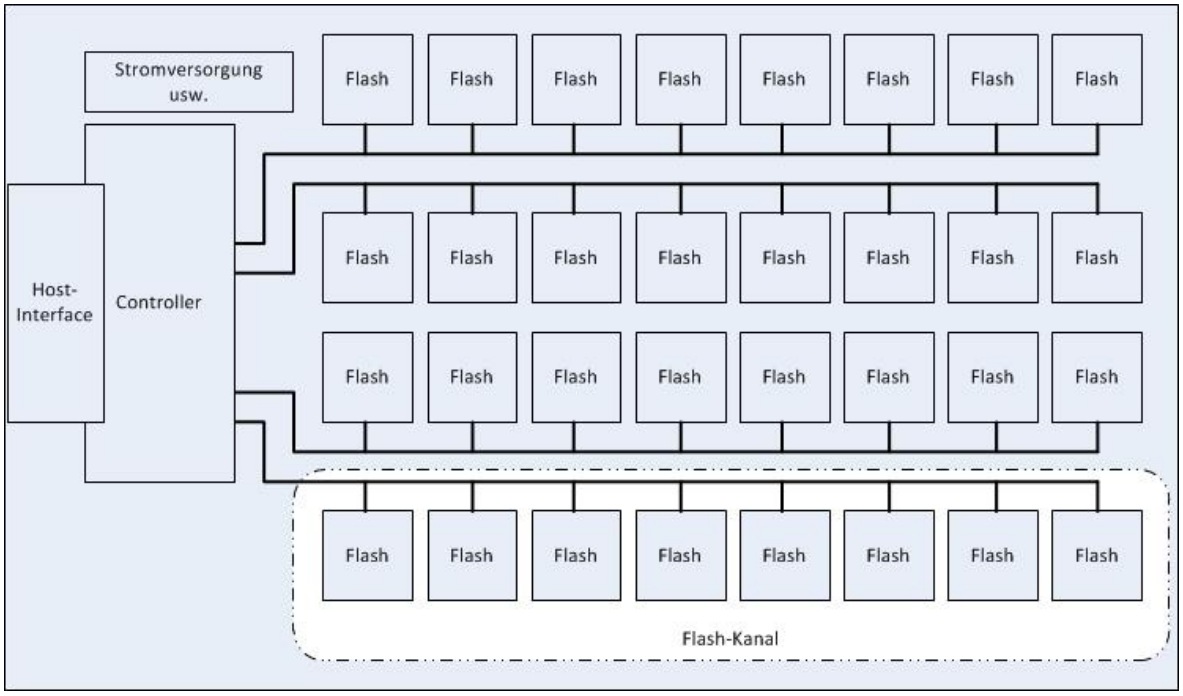

Abbildung 2.1: Blockschaltbild einer SSD

Um Daten, die in die Flashspeicher geschrieben oder von ihnen gelesen werden sollen, zwischenspeichern zu können, haben SSDs meist einen schnellen DRAM-Speicher als Data-Buffer integriert.

Diverse weitere Komponenten sind nicht SSD-spezifisch und sind z. B. für eine stabile Stromversorgung zuständig oder stellen als Sensoren für den Controller verschiedene Parameter zur Verfügung (z. B. Temperatur, Aktivitätssignale, usw.).

2.1.1 NAND-Speicher

2.1.1.1 Die NAND-Speicherzelle

NAND-Flashspeicher gehört, wie bereits erwähnt, zu den „Non-Volatile Memories“. Er besteht aus Transistoren, die meistens auf der „Floating Gate Technology“ basieren (siehe Abbildung 2.2). Diese sind in der Lage, auf ein durch eine Oxid-Schicht elektrisch isoliertes Gate durch den Quanten-mechanischen Tunneleffekt Elektronen zu schieben, die sich dann durch die elektrische Isolierung wie in einer Falle („Trap“) befinden und auch bei Wegfall aller angelegten Spannungen dort verbleiben. Die durch die Elektronen verursachte Ladung des Gates bewirkt eine Änderung (Erhöhung) der Schwellspannung V_{TH}, bei der der Transistor auf der Source-Drain-Strecke leitend wird. Diese „eigentlich analogen“ Vorgänge im Transistor können digital ausgewertet werden. Dabei gilt:

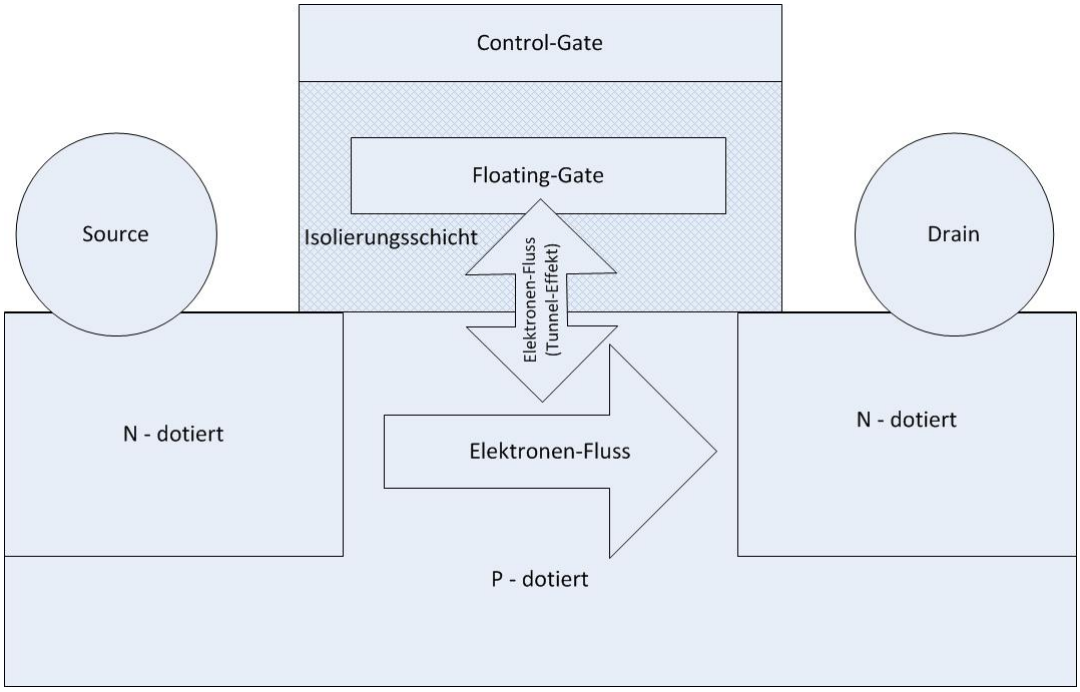

Abbildung 2.2: Flash-Speicherzelle

- Schreiben (Programmieren) der Speicherzelle: das Einbringen von Elektronen auf das Floating Gate (Zustand: logische „0“)
- Löschen der Speicherzelle: das Entfernen von Elektronen von dem Floating Gate (Zustand: logische „1“)
- Lesen einer logischen „0“: die Source-Drain-Strecke ist nicht leitend beim Anlegen einer Spannung oberhalb der „normalen“ Schwellspannung V_{TH}.
- Lesen einer logischen „1“: die Source-Drain-Strecke ist leitend beim Anlegen einer Spannung oberhalb der „normalen“ Schwellspannung V_{TH}.

Dabei sind Programmieren und Löschen einer Speicherzelle zwei völlig unterschiedliche Vorgänge, nicht einfach zwei Operationen mit gegensätzlichen Vorzeichen.

In diesen Speicherzellen kann demnach 1 Bit gespeichert werden. Durch exaktes, kontrolliertes Eindringen von Ladungen in das isolierte Gate können auch mehrere Änderungszustände der Schwellspannung erzeugt und wieder ausgelesen werden, mit dem Effekt, dass in diesen Zellen mehrere Bits gespeichert werden können. Folgende verschiedene NAND-Speicherzellen werden in SSDs verwendet:

- Single-level Cells (SLC): speichern über 2 Ladungszustände 1 Bit pro Zelle
- Multi-level Cells (MLC): speichern über 4 Ladungszustände 2 Bits pro Zelle
- 8-level Cells (8LC): speichern über 8 Ladungszustände 3 Bits pro Zelle (auch unter dem Namen Triple-level Cells, TLC, bekannt)

- 16-level Cells (16LC): speichern über 16 Ladungszustände 4 Bits pro Zelle

Durch Verwenden von Speicherzellen mit mehr Zuständen kann die Bitdichte erhöht und somit können größere Kapazitäten pro mm^2 angeboten werden, allerdings erfordern sowohl das Schreiben und Löschen als auch das Auslesen der Speicherzustände eine höhere Genauigkeit, was sich dann negativ u.a. auf die Performance auswirkt.

Das Durchdringen der Isolierungsschicht mit Elektronen ermüdet diese Schicht (to „wear out“). Dies bedeutet, dass nach einer endlichen Anzahl von Schreib- und Löschvorgängen (program- / erase-cycles) die Zahl der Elektronen, die die Isolierungsschicht durchqueren können, abnimmt und damit die Schwellspannung V_{TH} sich nicht mehr genügend und zuverlässig ändert, um beim Lesevorgang einen ausreichend signifikanten Unterschied feststellen zu können. Dieser Effekt ist umso stärker, je mehr Ladungszustände (Levels) in einer Speicherzelle vorkommen. Die Anzahl von möglichen Lösch- / Schreibzyklen beträgt bei:[4]

- SLC: bis zu 100.000
- MLC: zwischen ca. 3.000 und ca. 30.000
- TLC: zwischen ca. 300 und ca. 3000.

Die Werte differieren je nach Hersteller, Strukturgrößen und anderen Parametern.

Als Alternative zu den Speicherzellen in Floating-Gate Technologie gibt es NAND-Memory auf der Basis der sogenannten Charge-Trapping Technologie. Der Vorteil liegt vor allem in der höheren Speicherdichte und einer geringeren Ermüdung der Oxidschicht[5].

Ab einer Strukturgröße von ca. 10nm stößt eine weitere Verdichtung bei Verwendung von Zellen sowohl auf Basis der Floating-Gate als auch der Charge-Trapping Technologie an ihre Grenzen, da die einzelnen Zellen sich zu stark gegenseitig beeinflussen. Als Möglichkeit einer weiteren Steigerung der Packungsdichte haben einzelne Hersteller Verfahren entwickelt, mit denen sich Speicherzellen nicht nur in einer Ebene, sondern auch vertikal zusammenschalten lassen[6,7].

2.1.1.2 Flashspeicher-Organisation

Die beschriebenen Speicherzellen werden innerhalb des Flashspeichers in Einheiten gruppiert, bei denen der NAND-String die kleinste Einheit darstellt. Er wird in der

[4] Vgl. White Paper: Fujitsu, a.a.O. S. 4
[5] Vgl. Micheloni, R. / Marelli, A., a.a.O., S. 113f
[6] Vgl. http://www.samsung.com/us/business/oem-solutions/pdfs/V-NAND_technology_WP.pdf [2015-12-01]
[7] Micheloni, R. / Marelli, A., a.a.O., S. 117f

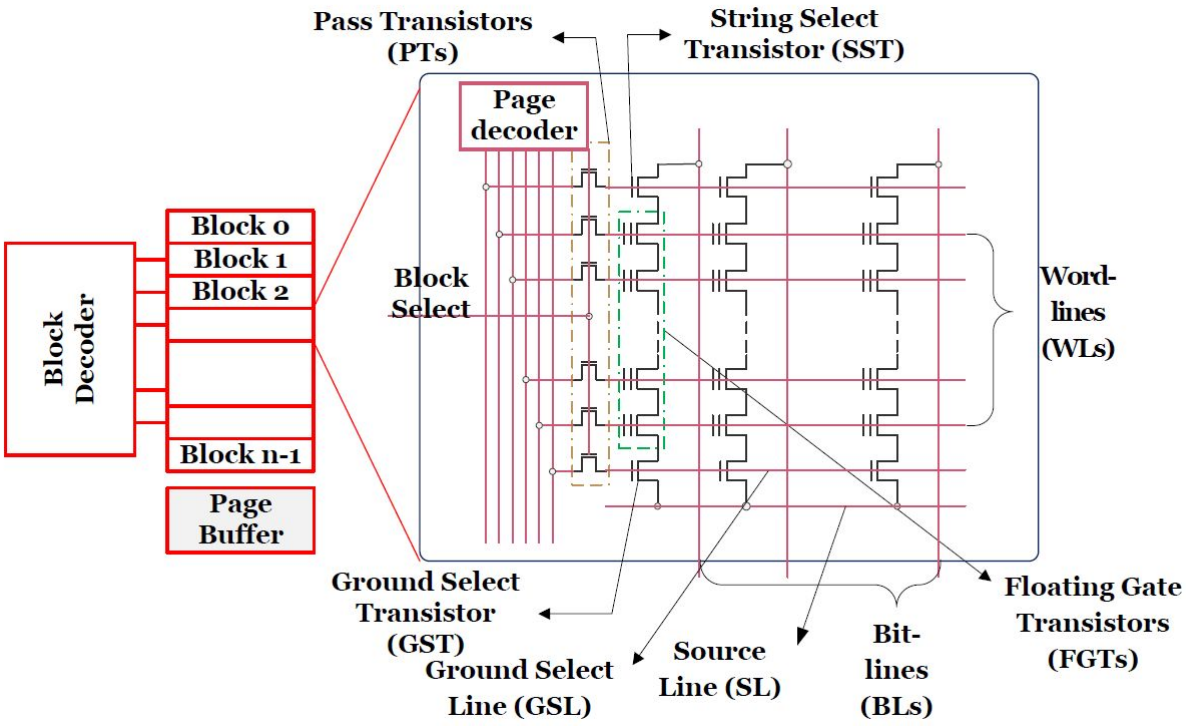

Abbildung 2.3: NAND-Array

Regel aus 32, 64 oder 128 Speicherzellen gebildet, die hintereinander geschaltet sind. NAND-Strings wiederum werden in NAND-Arrays zusammengefasst, so dass sie eine Matrix ergeben, in der mit Hilfe einer Reihe von zusätzlichen (Select-) Transistoren die Speicherzellen adressierbar sind (siehe Abbildung 2.3)[8].

Die Leitung, die die Source- und Drain-Kontakte innerhalb eines NAND-Strings verbindet, ergibt die sogenannte Bitline (BL), die miteinander verbundenen Control-Gates der „parallel“ angeordneten NAND-Strings bilden eine Wordline (WL). Damit ergeben sich die zwei wichtigsten Einheiten, mit denen in Flashspeichern und damit auch in SSDs gearbeitet werden kann:

- *Pages* und
- *Blocks*

Die Speicherzellen, die in einer Wordline verbunden sind, bilden eine Page, die Zellen mehrerer Wordlines (also mehrere Pages) werden zu einem Block zusammengefasst. Eine Page ist die kleinste Einheit, die in NAND-Flashspeichern gelesen bzw. geschrieben (programmiert) werden kann, ein Block ist die kleinste Einheit, die gelöscht werden kann.

[8]Bild entnommen aus http://electronics.stackexchange.com/questions/114726/how-do-nand-flash-memory-writes-work [2015-12-01]

Je nach verwendetem Zelltyp (SLC, MLC, TLC, 16LC) und je nach Strukturgröße kommen in aktuellen SSDs Pages mit folgenden Größen vor:

- 2 KiB, 4 KiB, 8 KiB und 16 KiB

Blöcke bestehen z. B. aus 64, 128, 256 oder 512 Pages, so dass sich mögliche Blockgrößen von 128 KiB bis 8 MiB ergeben[9,10,11].

Die nächstgrößeren Einheiten, die in einer NAND-Struktur vorkommen, sind *Plane* und *Die*. Mehrere Blöcke bilden eine Plane, und mehrere Planes ein Die. Ein Die ist die kleinste Einheit bei Flashspeichern, auf der unabhängig voneinander Befehle (Lesen, Schreiben, usw.) abgearbeitet werden können. Innerhalb eines Dies können bestimmte Operationen auf den darauf befindlichen Planes zeitgleich ausgeführt werden (z.B. Multi-Plane read)[12]. Ein oder mehrere Dies wiederum sind in einem *Package*, z. B. in Form eines *TSOP* (Thin Small Outline Package) oder eines *BGA* (Ball Grid Array) untergebracht. Der Gesamtkapazität gemäß sind auf dem PCB (Printed Circuit Board) einer SSD entsprechend viele solcher TSOPs oder BGAs aufgelötet.

Eine der momentan (Dezember 2015) größten erhältlichen SSDs ist die SSD 850 2 TiB von Samsung. Es gibt sie als PRO-Version mit 2-bit MLC-Speicherzellen bzw. als EVO-Version mit 3-bit MLC (TLC)-Speicherzellen. Ausgehend von der Anzahl der Packages (8) ist folgender logischer Aufbau möglich und wahrscheinlich[13]:

- Bytes per Page: 8 KiB
- Pages per Block: 512
- Bytes per Block: 4 MiB
- Blocks per Plane: 2048
- Bytes per Plane: 8 GiB
- Planes per Die: 2
- Bytes per Die: 16 GiB
- Dies per Package: 16
- Bytes per Package: 256 GiB
- 8 Packages (TSOPs): 8 * 256 GiB = 2TiB

[9] Vgl. Micheloni, R. / Marelli, A., a.a.O., S. 20ff
[10] Vgl. Micheloni, R. / Crippa, L., a.a.O., S. 20ff
[11] Vgl. Heidrich, S.: Hyperstone 2015: New flash management architecture enables MLC for industrial storage http://files.iccmedia.com/magazines/basfeb15/basfeb15-p25.pdf [2015-12-01]
[12] Vgl. Micheloni, R. / Crippa, L., a.a.O., S. 32
[13] Leider ist der genaue logische Aufbau weder im Internet noch bei den Herstellern direkt zu erfahren, er gilt als vertraulich.

2.1.1.3 Lesen von Daten

Wie im vorigen Abschnitt erwähnt, ist eine Page die kleinste Einheit, die in einer SSD gelesen werden kann. Zum Lesen einer Page stehen verschiedene Commands zur Verfügung, bei denen als Parameter u. a. die Adresse der zu lesenden Page und die Adresse des Blocks, in dem sich die betreffende Page befindet, übergeben werden[14]. Je nach Größe der Page (2 KiB bis 16 KiB) und je nach Clustergröße (z. B. 4 KiB) bzw. physikalischer Sektorgröße (512 Byte oder 4KiB) des verwendeten Dateisystems bedeutet das, dass evtl. mehrere Pages pro Leseanforderung des Hosts gelesen oder die gelesenen Daten aufgeteilt werden müssen. Die transparente Zuordnung der Cluster bzw. logischen Blockadressen des Dateisystems (LBA) des Hosts zu den Pages ist Aufgabe des SSD-Controllers. Mehr dazu in Abschnitt 2.1.2.2.

2.1.1.4 Schreiben von Daten

Wie beim Lesen ist eine Page die kleinste Einheit, die in einer SSD geschrieben („programmiert“) werden kann. Schreiben bedeutet technisch (siehe Abschnitt 2.1.1.1) das Einbringen von Ladungen auf das Floating Gate, was eine Änderung einer logischen „1“ in eine logische „0“ ergibt. Somit können nur Zellen einer zuvor gelöschten Page (logisch „1“) geschrieben werden. Dazu werden einem Write-Command (wie beim Lesen) als Parameter die Adresse der zu schreibenden Page und die Adresse des Blocks, in dem sich die betreffende Page befindet, übergeben und in weiteren Zyklen die zu schreibenden Daten[15].

Sollen nun Daten einer Page geändert werden, so kann nicht einfach dieselbe Page überschrieben werden. Es gibt keine „Update“ - Operation bei Flashspeichern. Folgende Vorgehensweisen sind denkbar:

1. Alle Pages des betreffenden Blocks werden in einen Buffer (z.B. DRAM) gelesen, dort werden die Daten entsprechend geändert, der Block wird gelöscht und die Pages werden neu geschrieben (siehe Abbildung 2.4, Version 1).

2. Alle Pages des betreffenden Blocks werden in einen Buffer (z.B. DRAM) gelesen, dort werden die Daten entsprechend geändert, die Pages werden in einem zuvor gelöschten Block neu geschrieben. Alle Pages im bisherigen Block werden als ungültig markiert (siehe Abbildung 2.4, Version 2).

3. Die zu ändernde Page wird in einen Buffer (z.B. DRAM) gelesen, dort geändert und anschließend in eine seit dem letzten Löschen noch nicht benutzte Page

[14] Vgl. Micheloni, R. / Crippa, L., a.a.O., S. 29
[15] Vgl. ebenda S. 33f

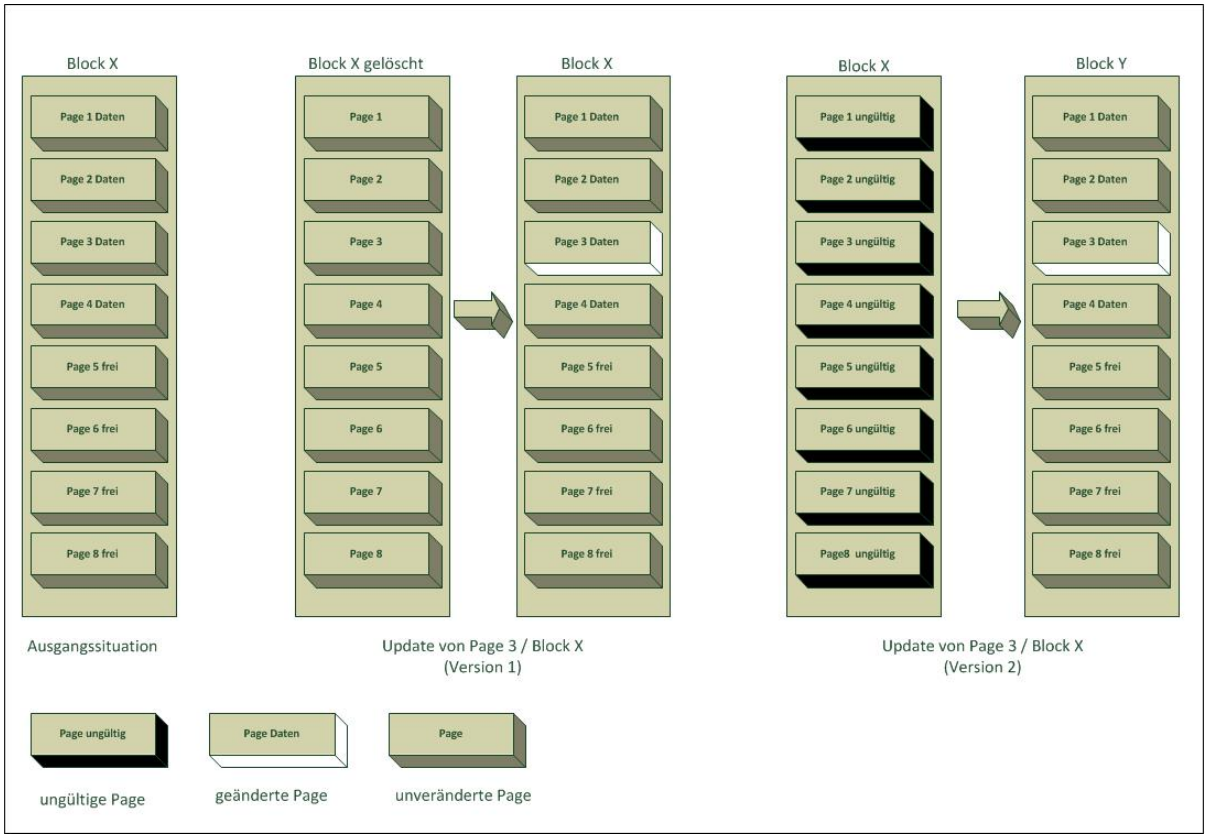

Abbildung 2.4: Update einer Page Version 1 / 2

desselben Blocks geschrieben. Die bisherige Page wird als ungültig markiert (siehe Abbildung 2.5, Version 3).

4. Die zu ändernde Page wird in einen Buffer (z.B. DRAM) gelesen, dort geändert und anschließend in eine seit dem letzten Löschen noch nicht benutzte Page eines anderen Blocks geschrieben. Die bisherige Page wird als ungültig markiert (siehe Abbildung 2.5, Version 4).

Die Abbildungen 2.4 und 2.5 zeigen diese Möglichkeiten beim Ändern einer Page anhand folgenden Beispiels:

Ausgangssituation: in Block X sind die Pages 1, 2, 3, 4 beschrieben, Page 3 soll geändert werden

- (Version 1): Lesen aller Pages in Block X in den Buffer, Update der Daten in Page 3, Löschen des Blocks X, Schreiben der Pages 1, 2, 3, 4 in Block X
- (Version 2): Lesen aller Pages in Block X in den Buffer, Update der Daten in Page 3, Markieren aller Pages des Blocks X als ungültig, Schreiben der Pages 1, 2, 3, 4 in Block Y
- (Version 3): Lesen von Page 3 in Block X in den Buffer, Update der Daten in

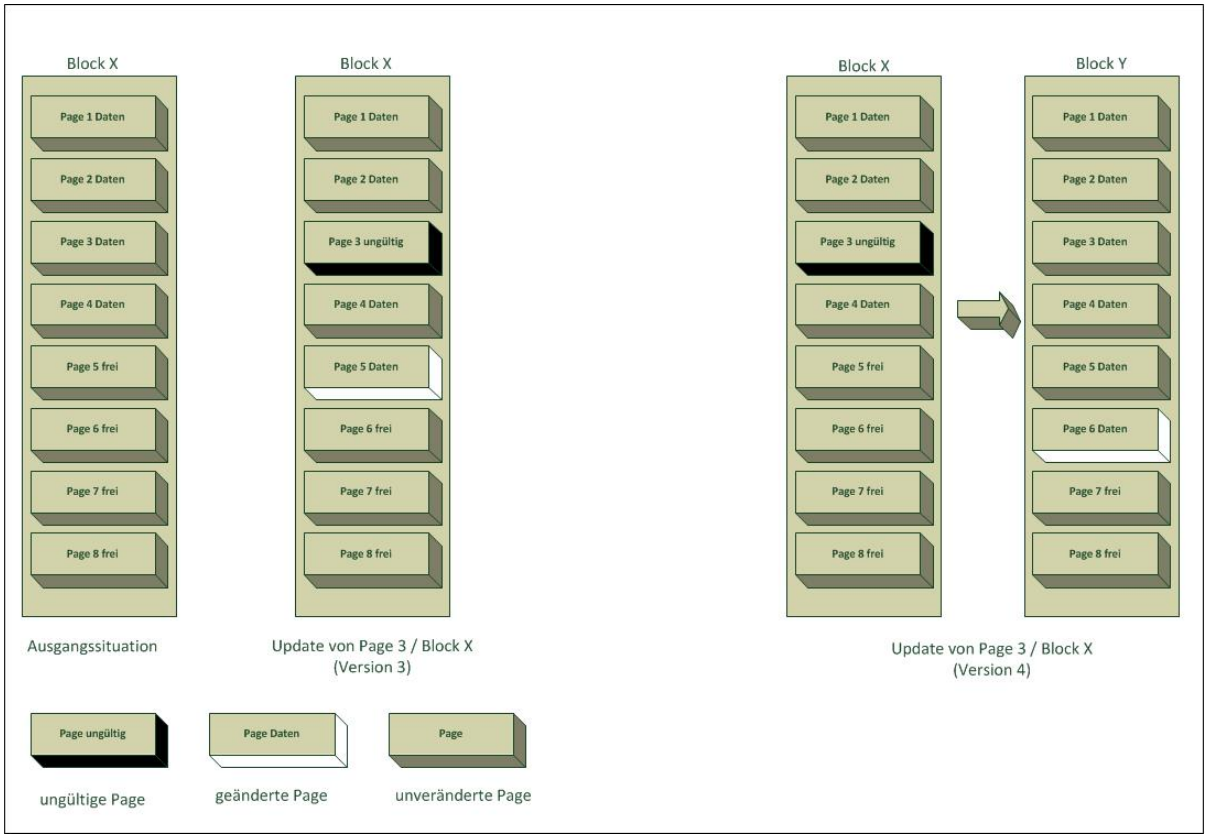

Abbildung 2.5: Update einer Page Version 3 / 4

Page 3, Markieren der Page 3 als ungültig, Schreiben der Daten im Buffer in Page 5 in Block X

- (Version 4): Lesen von Page 3 in Block X in den Buffer, Update der Daten in Page 3, Markieren der Page 3 in Block X als ungültig, Schreiben der Daten im Buffer in Page 6 in Block Y

Man sieht, dass die beschriebenen Möglichkeiten, die Daten einer Page zu ändern, deutlich mehr Aufwand benötigen, als der eigentliche Schreibvorgang einer Page in einem gelöschten Block. Dieser Mehraufwand beim Schreiben wird oft als *Write-Amplifikation* bezeichnet und kann wie folgt rechnerisch beschrieben werden:[16]

$$\text{Write Amplifikation} = \frac{\text{Tatsächliches Auf Datenträger Geschriebenes Datenvolumen}}{\text{Eigentlich Zu Schreibendes Datenvolumen}}$$

Hinzu kommt der Aufwand, die (Mehr-) Daten zu lesen und vor allem das eventuelle Löschen eines Blocks. Damit eine SSD eine eher gleichmäßig hohe und verlässliche Schreibperformance bieten kann, sind einige Anstrengungen notwendig. Dies sind Kernaufgaben des Controllers, die in Abschnitt 2.1.2 beschrieben werden.

[16]Vgl. White Paper: Fujitsu, a.a.O., S. 5

2.1.1.5 Löschen von Daten

Löschen von Daten in einem Flashspeicher ist notwendig, um überhaupt neue Daten auf den Speicher schreiben zu können. Der Löschvorgang ist, wie schon erwähnt, nicht einfach die Umkehrung des Schreibvorgangs. Er erfordert das Anlegen einer hohen Spannung in mehreren Schritten, die Abschirmung benachbarter Bereiche von Zellen vor ungewollter Beeinflussung sowie das Überprüfen (Lesen) aller Zellen, ob der Löschvorgang erfolgreich war. Dies nimmt deutlich mehr Zeit in Anspruch als ein reiner Lese- oder Schreibvorgang (Größenordnung: mehrere Millisekunden[17,18]). Deshalb ist es effizienter, viele Zellen in einer größeren Einheit parallel zu löschen. In NAND-Flashspeichern ist diese Einheit ein Block, der, wie in Abschnitt 2.1.1.2 erwähnt, aus 64 bis 512 Pages besteht und bis zu 8 MiB groß sein kann. Durch das Löschen eines ganzen Blocks stehen danach alle Pages dieses Blocks wieder für Schreibvorgänge zur Verfügung.

2.1.2 Aufgaben des Controllers

Der Controller ist das Herzstück einer SSD, sein Design und seine Leistungsfähigkeit sind wesentlich verantwortlich für die Schreib- und Leseperformance, die Datenintegrität und die Lebensdauer der Speicherzellen und damit der gesamten SSD. Diese Anforderungen stehen oft in Konkurrenz zueinander, so dass in vielem ein Kompromiss gefunden werden muss, um allen Anforderungen gerecht werden zu können. Zur Bewältigung seiner oft zeitkritischen Aufgaben besteht ein SSD-Controller in der Regel aus einem Standard-Prozessor sowie spezieller Hardware zusammen mit entsprechender Firmware, in der ausgeklügelte Algorithmen ausgeführt werden[19].

Wie in Abbildung 2.6[20] ersichtlich, muss der Controller zusätzlich zu seinen *klassischen* Funktionen auch Aufgaben erledigen, die aus den in den vorigen Abschnitten beschriebenen SSD-spezifischen Methoden des Schreibens, Löschens und der sonstigen Datenorganisation herrühren. Folgende Funktionsblöcke sind innerhalb des Controllers vorhanden:

- Host-Interface als Schnittstelle zum Host
- Flash-Interface als Schnittstelle zum Flashspeicher
- Bad Block Management

[17] Vgl. Micheloni, R. / Crippa, L., a.a.O., S. 36
[18] Vgl. Micheloni, R. / Marelli, A., a.a.O., S. 159
[19] Vgl. Micheloni, R. / Crippa, L., a.a.O., S. 40
[20] Vgl. Micheloni, R. / Marelli, A., a.a.O., S. 176

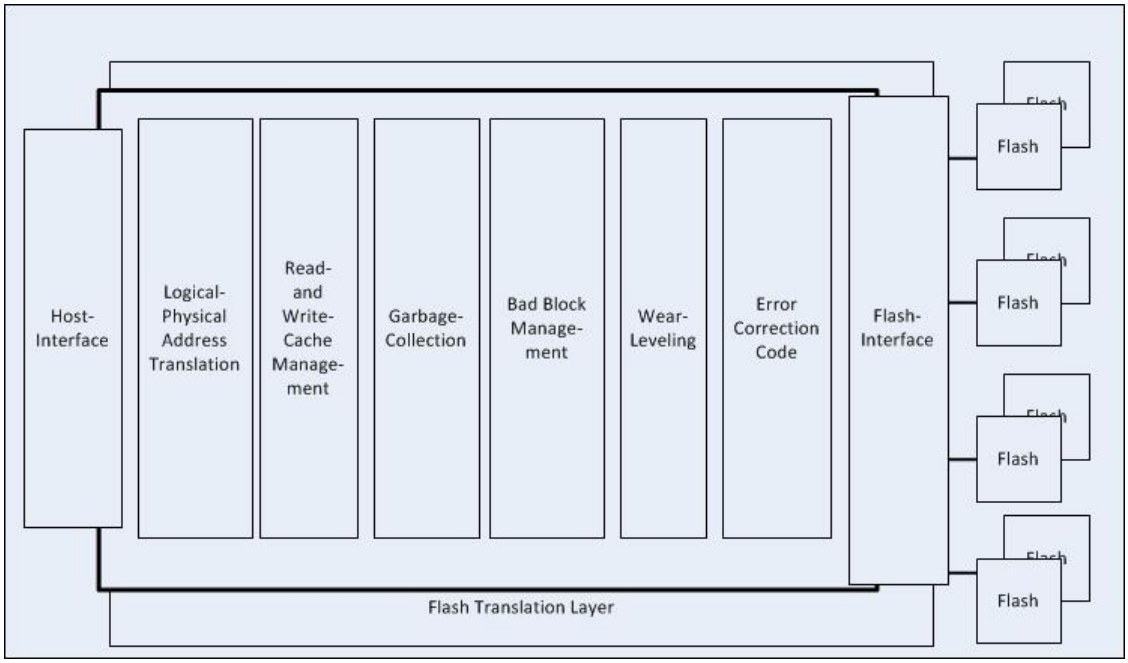

Abbildung 2.6: Controller-Funktionen

- Logical to Physical Address Translation (Umsetzung von logischen in physikalische Adressen)
- Read- and Write-Cache Management
- Garbage-Collection
- Wear-Levelling
- Error Correction

Klassische Aufgaben (das sind die, die auch z. B. in HDD-Controllern implementiert sind) sind:

- Bereitstellen eines geeigneten Interfaces zum und vom Host im Industriestandard, z. B. SATA, SAS: Entgegennahme, Interpretation und Umsetzung der Lese- und Schreibanforderungen vom und zum Host in entsprechende Tasks innerhalb des Flashspeichers
- Bad-Block Management: auch Flashspeicher sind nicht fehlerfrei. Das Bad-Block Management verwaltet eine bereits beim Fertigungsprozess erstellte Liste nicht zuverlässiger Blöcke, führt diese Liste im Betrieb weiter und sorgt so für den transparenten Ersatz defekter Speicherblöcke[21].

Die anderen Funktionen haben eher SSD-spezifische Aspekte und werden in den nächsten Abschnitten näher erläutert.

[21] Vgl. Micheloni, R. / Marelli, A., a.a.O., S. 30

2.1.2.1 Read- / Write-Cache

Obwohl mit Flashspeicher sowohl beim Lesen als auch beim Schreiben sehr hohe Geschwindigkeiten erzielt werden, ist er doch gegenüber z. B. dem (flüchtigen) DRAM um Größenordnungen langsamer. Deshalb sind SSDs in der Regel mit einem schnellen und großen DRAM-Cache[22] ausgestattet, der sowohl Schreib- als auch Leseoperationen zwischenspeichert und somit beschleunigt. Werden Daten, die sich schnell ändern, im Schreib-Cache gehalten und nur dort geändert (und damit seltener in den Flashspeicher zurückgeschrieben), wird die Schreibperformance nicht nur gesteigert, sondern es werden Lösch- und Schreibzyklen vermieden und damit die Lebensdauer der Flashspeicherzellen erhöht.

Bis vor wenigen Jahren arbeiteten gängige Betriebssysteme ausschließlich mit einer Sektorgröße von 512 Bytes als kleinste Einheit für Lese- und Schreibzugriffe auf ihre Datenträger, unabhängig von der in vielen Dateisystemen verwendeten Clustergröße von z. B. 4 KiB. Vor allem die Steigerung der Kapazität von Massenspeichern auf über 2 TiB, aber auch ein besseres Verhältnis von Brutto- zu Nettokapazität und mehr Möglichkeiten bei Fehlerkorrekturen (der Platz pro Sektor für ECC kann erhöht werden), bewogen die Hersteller von Massenspeichern dazu, die Sektorgröße von 512 Bytes abzulösen und auf 4 KiB zu erhöhen. Neueste Betriebssysteme[23] können inzwischen auch mit der neuen Sektorgröße arbeiten, bei älteren emulieren diese Datenträger gegenüber dem Betriebssystem eine Sektorgröße von 512 Byte[24].

Da, wie bereits erwähnt, bei SSDs die kleinste Einheit für Lese- und Schreibvorgänge eine Page mit einer Größe von 2 KiB bis 16 KiB ist, passt hier die vom Betriebssystem erwartete Sektorgröße meist nicht mit der Größe einer Page zusammen, egal ob das Betriebssystem mit 512 Bytes oder auch mit 4 KiB Sektorgröße arbeiten kann. Soll nun nur ein Teil einer Page geschrieben werden, z.B. 512 Byte bei einer Pagegröße von 8 KiB, stehen die ungenutzten 93,75% der Daten dieser Page für einen weiteren Schreibvorgang erst wieder nach einem Löschen des gesamten Blocks zur Verfügung, da jeder weitere Schreibvorgang benachbarte Zellen beeinträchtigen kann („program disturbance“). Die durch solche ineffizienten Schreibvorgänge verursachte Fragmentierung des Flashspeichers hat als Folge eine verminderte Schreibperformance und eine hohe Anzahl von Löschzyklen. Da die Sektorgröße der Betriebssysteme eben nicht beliebig anpassbar ist, wird auch in den Fällen, in denen die Sektorgröße nicht der Größe ei-

[22] z. B. 1024 MB DDR2 SDRAM-Cache bei der SAMSUNG SSD 850 PRO 1TB, http://www.samsung.com/global/business/semiconductor/minisite/SSD/downloads/document/Samsung_SSD_850_PRO_Data_Sheet_rev_2_0.pdf [2015-12-01]

[23] z. B. Windows 8, Windows Server 2012, Linux ab Kernel Version 2.6.31

[24] Vgl. International Disk Drive Equipment and Materials Association (IDEMA): Advanced Format (AF) Technology:
http://www.idema.org/?page_id=98 [2015-12-15]

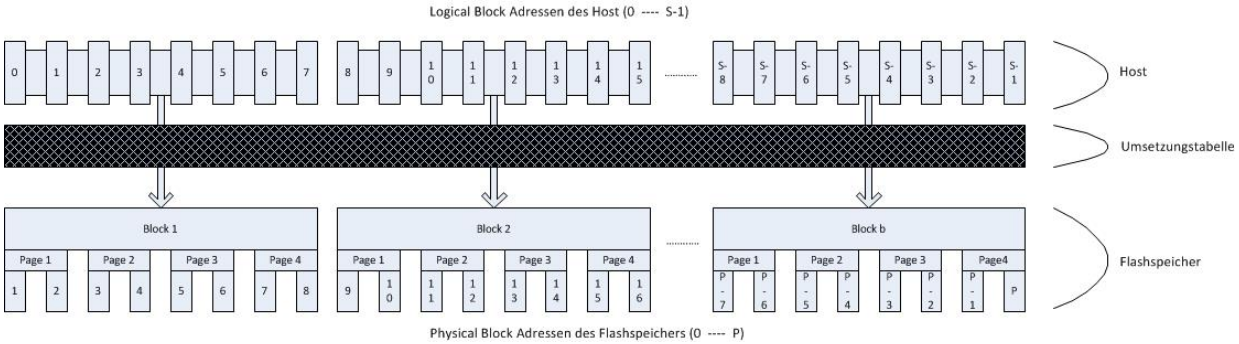

Abbildung 2.7: Block-basierende Umsetzung von LBA zu PBA

ner Page oder einem Vielfachen davon entspricht, versucht, Schreibvorgänge im Cache zusammenzufassen, um unnötige Lösch- / Schreibzyklen zu vermeiden[25].

2.1.2.2 Umsetzung Logical-Block-Adressen nach Physical-Block-Adressen

Betriebssysteme bzw. ihre Dateisysteme sehen den Speicher ihrer angeschlossenen Massendatenträger als serielle Aneinanderreihung sogenannter *Logischer Blöcke* mit fester Größe, die, wie oben schon erwähnt, entweder 512 Byte (bisherige Betriebssysteme) oder 4 KiB (neuere Betriebssysteme) groß sind. Sie werden, mit 0 beginnend, durchnummeriert, der tatsächliche physikalische Aufbau des Datenträgers (z. B. die Aufteilung in Zylinder, Köpfe und Sektoren bei HDDs, Pages und Blöcke bei SSDs) wird dabei nicht beachtet. Vom Betriebssystem aus gesehen, werden Daten transparent in einem bestimmten logischen Block (Logical Block Address, LBA) geschrieben, gelesen und eventuell geändert.

Die in den vorigen Abschnitten beschriebenen Verfahren, wie Daten auf SSDs geschrieben und geändert werden, bedingen allerdings, dass es keine feste Zuordnung dieser logischen Blöcke des Betriebssystems zu physikalischen Einheiten in den Flashspeichern bei SSDs geben kann. Vielmehr unterhält der Controller Umsetzungstabellen, die ständig auf dem Laufenden gehalten werden müssen, immer dann aktualisiert werden, wenn durch Ändern von Daten diese Daten in neue Pages bzw. in neue Pages in einem anderen Block geschrieben werden. Dabei gibt es verschiedene Ansätze:

- Block-basierende Umsetzungstabelle: ein Block des Flashspeichers wird der Anzahl von Logical Blocks des Betriebssystems zugeordnet, die seiner Größe entsprechen, z. B. 8192 Logical Blocks bei einer Blockgröße von 4 MiB und einer LBA-Größe von 512 Bytes. Somit entspricht der erste Sektor solch einer Gruppe von Sektoren immer der ersten (Sub-) Page des gerade zugewiesenen Flash-Blocks (siehe Abbildung 2.7).

[25] Vgl. Micheloni, R. / Marelli, A., a.a.O., S. 185f

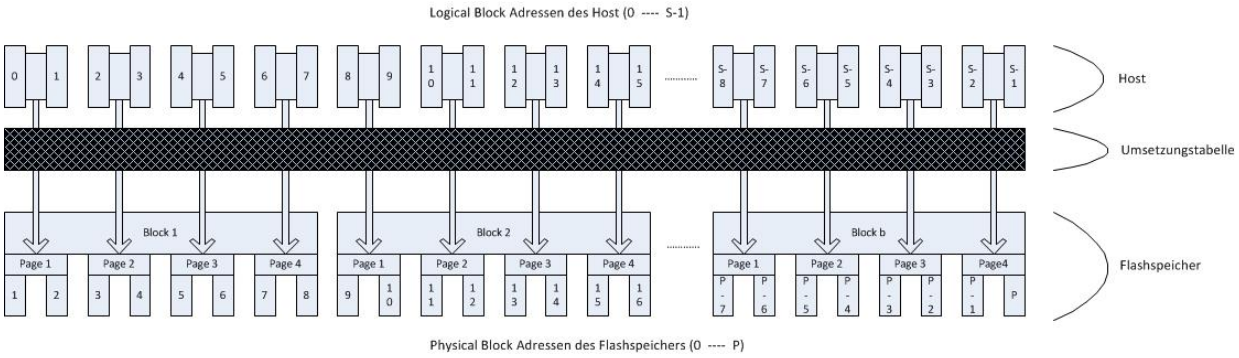

Abbildung 2.8: Page-basierende Umsetzung von LBA zu PBA

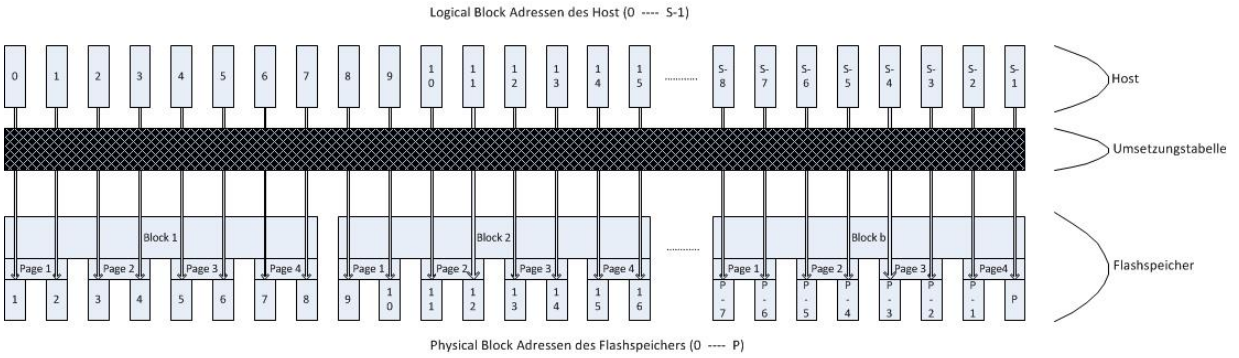

Abbildung 2.9: Sub-Page-basierende Umsetzung von LBA zu PBA

- Page-basierende Umsetzungstabelle: einer Page des Flashspeichers wird die Anzahl von Logical Blocks des Betriebssystems gegenübergestellt, die ihrer Größe entsprechen, z. B. 8 Logical Blocks bei einer Blockgröße von 4 KiB und einer LBA-Größe von 512 Bytes. Dabei kann jede Page des gesamten Flashspeichers jeder Gruppe von logischen Sektoren des Betriebssystems zugeordnet werden, unabhängig davon, in welchem Flashspeicher-Block sie sich befindet. Die einzelnen Sektoren entsprechen Sub-Pages. Im Sonderfall von einer Pagegröße von 4 KiB und einem Betriebssystem, welches 4 KiB als LBA führt, kann so eine Tabelle mit einer 1:1 Beziehung entstehen (siehe Abbildung 2.8).

- Sub-Page-basierende Umsetzungstabelle: hier wird einem logischen Block des Betriebssystems eine entsprechend gleich große Sub-Page zugeordnet. Zu beachten ist hier, dass natürlich beim Schreiben bzw. Ändern von Sektoren immer eine ganze Page geschrieben wird, beim Lesen aber nur der Teil gelesen und übertragen werden muss, der benötigt wird (siehe Abbildung 2.9).

Je feiner die Umsetzungstabelle aufgebaut ist, desto größer ist sie, nimmt dadurch mehr Platz in Anspruch und es wird mehr Aufwand benötigt, sie zu durchsuchen und zu aktualisieren. Auf der anderen Seite ist die Kapazitätsausnutzung höher und sie bietet eine größere Flexibilität für verschiedene Anwendungsszenarien. Weiterhin hat die Art der Umsetzung einen großen Einfluss auf die Schreibperformance und Abnutzung

der Speicherzellen: bei Block-basierender Umsetzungstabelle und Überschreibanforderungen in der Größe von einem Sektor muss ein gesamter Block gelesen und in einem neuen Block geschrieben werden. Die Write-Amplifikation (siehe Abschnitt 2.1.1.4) kann dabei Werte weit über 1000 annehmen.

Die Abbildungen 2.7, 2.8 und 2.9 zeigen, wie mit den verschiedenen Methoden die Zuordnung (schraffierter Bereich) der logischen Blockadressen des Hosts (obere Reihe) zu den physikalischen Adressen des Flashspeichers (untere Reihe) durchgeführt wird. Man kann erkennen, dass je nach Feinheitsgrad der Basis der Zuordnung die Umsetzungstabelle größer wird, aber im Gegenzug auch mehr Flexibilität bietet.

Die Umsetzungstabellen werden in der Regel im Cache gehalten und bearbeitet und müssen beim Abschalten der SSD zuverlässig in den Flashspeicher selbst geschrieben werden, da die Integrität dieser Tabellen essentiell wichtig für die Daten-Integrität der gesamten SSD ist.[26,27].

2.1.2.3 Garbage-Collection

Hinter der Funktion *Garbage-Collection* steckt die wohl wichtigste Aufgabe des Controllers einer SSD. Wichtig vor allem in dem Sinn, da ohne sie eine SSD zu dem Zeitpunkt dramatisch bei der Schreibperformance einbrechen würde, an dem keine freien, d. h. gelöschten Pages für neue Schreibvorgänge mehr zur Verfügung stehen. Das bedeutet nicht, dass dann die gesamte Kapazität der SSD aus Sicht des Betriebssystems aufgebraucht ist, vielmehr sind alle möglichen Pages mit gültigen (oder aktiven) bzw. mit ungültigen (oder inaktiven)[28] Daten gefüllt. Auch die verwendete Methode zur Umsetzung von logischen in physikalische Adressen hat Einfluss auf den Grad, wie weit die Kapazität erschöpft ist, z. B. kann es bei Block-basierender Umsetzung vorkommen, dass nur eine Page pro Block belegt ist und die übrigen eigentlich frei sind, aber nicht für Pages anderer Blöcke verwendet werden können. Je nach Basis der Umsetzung von LBA nach PBA (Block, Page oder Sub-Page) ist die Kapazitätsausnutzung höher oder niedriger.

Müssen jetzt Daten geändert werden, kann dies nur über das aufwändige Verfahren - *read, modify, erase, write* - geschehen, wie in Abbildung 2.4, Version 1 oder Abbildung 2.5, Version 3 beschrieben.

Um diese Situation zu vermeiden, wendet der SSD-Controller die sogenannte *Garbage-Collection*[29] an. Hierbei *sammelt* der Garbage-Collector gültige Pages eines oder meh-

[26] Vgl. Micheloni, R. / Marelli, A., a.a.O., S. 28
[27] Vgl. Heidrich, S., a.a.O.
[28] Ungültige (inaktive) Pages werden oft auch als *stale* bezeichnet.
[29] Garbage-Collection heißt übersetzt ungefähr *Einsammeln von Unrat* und ist auch an anderer Stelle in der Informationstechnologie bekannt, z. B. im Memory Management einer Software.

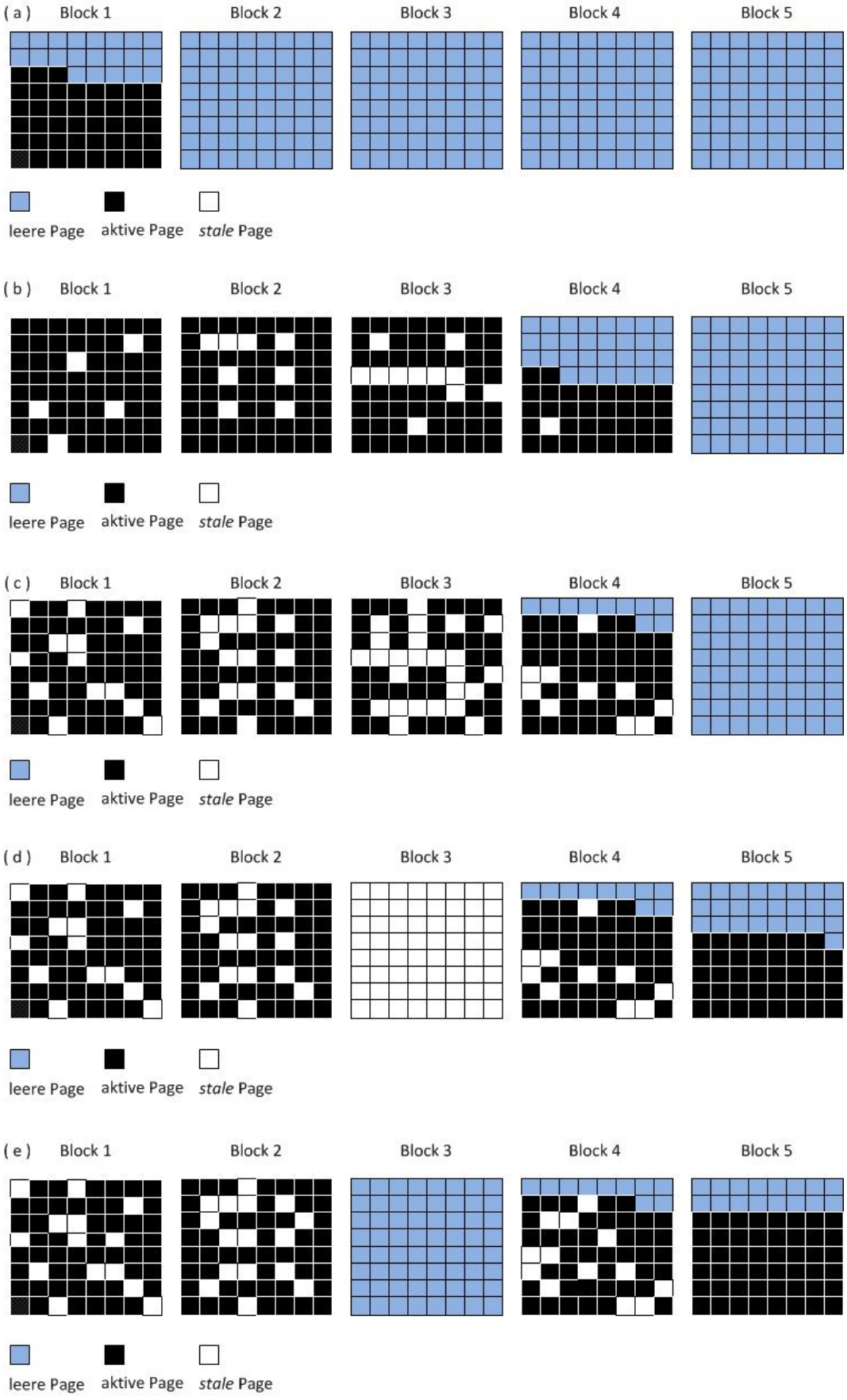

Abbildung 2.10: Beispiel von Garbage-Collection

rerer Blöcke *ein*, fasst sie neu zusammen und schreibt sie in einen freien, gelöschten Block (unter Berücksichtigung der verwendeten Umsetzungsmethode von logischen in physikalische Adressen). Übrig bleiben die *stale* Pages, deren Blöcke dann in einem weiteren Vorgang gelöscht werden können und damit wieder für neue Schreibvorgänge zur Verfügung stehen[30].

Abbildung 2.10 zeigt vereinfacht die Vorgehensweise in einem Beispiel mit 5 Blöcken und jeweils 64 Pages. Die Pages werden nach und nach mit Daten beschrieben, beginnend mit Block 1 (a). Ist ein Block voll, werden Pages der nächsten Blöcke (2, 3 und 4) verwendet (b) bzw. (c). Müssen die Daten einer Page geändert werden, wird die bisherige Page als *stale* markiert und der neue Inhalt in freie Pages geschrieben (b) bzw. (c). Bevor keine freien Blöcke mehr zur Verfügung stehen, fasst der Garbage-Collector die aktiven Pages von Block 3 ein und kopiert sie in den noch freien Block 5 (d). Block 3 enthält nun keine aktiven Pages mehr und kann bei nächster Gelegenheit gelöscht werden (e). Somit stehen wieder gelöschte Pages zum neu Beschreiben bereit[31].

Garbage-Collection kann im Hintergrund stattfinden (sogenannte Background GC). Wenn eine SSD gerade keine Aufgaben vom Host zu erledigen hat, sich also im Idle-Zustand befindet, kann Garbage-Collection durchgeführt werden. Damit hat eine SSD immer ausreichend gelöschte Blöcke zum performanten Schreiben zur Verfügung. Allerdings wird dies mit einem erhöhten Verschleiß erkauft, da auch unnötige Lösch- / Schreibzyklen durchgeführt werden, wenn Pages kopiert werden, die eventuell in Kürze als *stale* markiert würden. Background GC findet oft in Consumer-SSDs Anwendung, da hier in der Regel mit mehr Idle-Zeit gerechnet werden kann.

Die andere Möglichkeit ist das Foreground-GC (auch Echtzeit GC genannt). Hierbei wird die Garbage-Collection erst durchgeführt, wenn der neue gelöschte Platz wirklich gebraucht wird, mithin werden so unnötige Lösch- / Schreibzyklen vermieden. Dies wird eher in Enterprise-SSDs verwendet, in denen viele Schreibaufträge durchgeführt werden und Idle-Zustände seltener vorkommen.

Welche Methode bzw. welche Kombination angewendet wird, ist in der Regel für eine SSD nicht dokumentiert und gilt ebenso als vertraulich.

2.1.2.4 Wear-Levelling

Eine wesentliche Eigenschaft einer Flashspeicherzelle ist der Umstand, dass ihre Lebensdauer begrenzt ist. Das unterscheidet sie von eigentlich allen anderen (Halbleiter-)

[30] Vgl. Micheloni, R. / Marelli, A., a.a.O., S. 29

[31] Der Begriff Garbage-Collection wird hier eigentlich nicht richtig verwendet, denn nicht der *Unrat*, sondern die *wertvollen Stücke* werden eingesammelt, der *Unrat* wird liegen gelassen.

Speichertechnologien. Diese Begrenzung ist nicht zeitlicher Natur, sondern jedes Löschen und Schreiben einer Speicherzelle beschädigt („schwächt“) die Oxid-Schicht, die die Elektronen (die letztendlich die digitale Information darstellen) zum Floating-Gate bzw. zur Charge-Trap passieren (siehe Abschnitt 2.1.1.1 sowie Abbildung 2.2).

Die Anzahl von möglichen Lösch- / Schreibzyklen ist wesentlich abhängig von der verwendeten Speicherzelle (SLC, MLC, TLC usw.), sie ist beim SLC-Typ deutlich höher (bis zu 100.000 Zyklen) als bei den heute weiter verbreiteten MLC bzw. TLC-Typen (teilweise nur wenige hundert Zyklen[32]). Wird im Extremfall eine SSD wie eine CD / DVD verwendet, also nur von ihr gelesen, nachdem sie einmal geschrieben wurde, hat diese begrenzte Lebensdauer praktisch keine Auswirkung und bräuchte auch nicht weiter betrachtet zu werden. Im anderen Extremfall, in dem alle Daten ständig verändert werden oder die SSD nach dem Schreiben gleich wieder gelöscht und neu beschrieben wird, kann man sich gut vorstellen, dass im Falle von wenigen hundert erlaubten Lösch- / Schreibzyklen die Speicherzellen in absehbarer Zeit nicht mehr zuverlässig ihre Informationen ablegen und lesen können.

In der Praxis der Datenträgernutzung werden diese beiden Extremfälle sicher keine Anwendung finden, sondern eine beliebige Kombination aus beiden Nutzungen. Häufig sich ändernde Datenbereiche, z.B. Loggingdateien, Speicherauslagerungsdateien und Datenbanken stehen Daten gegenüber, die sich nach dem erstmaligen Schreiben auf den Datenträger eher selten ändern. Dies sind z. B. Betriebssystems- und Applikationsdateien, die sich nach ihrer Installation äußerst selten ändern, auch Bild- und Filmdateien, die im Consumerbereich große Teile von Datenträgern füllen. Um zu verhindern, dass bei SSDs häufig sich verändernde Dateien wenige Datenblöcke extrem schnell abnutzen lassen und dann vorzeitig ausfallen, sorgt der Controller dafür, dass die Abnutzung so gleichmäßig wie möglich über alle Blöcke verteilt wird. Dieser Vorgang nennt sich Wear-Levelling („Abschwächungsverteilung“).

Das Wear-Levelling und die zwei vorgenannten Controller-Funktionen der Garbage-Collection (siehe Abschnitt 2.1.2.3) bzw. der Umsetzung von logischen in physikalische Adressen (siehe Abschnitt 2.1.2.2) arbeiten nicht isoliert voneinander, sondern sie sind vielmehr eng miteinander verzahnt. Wird bei der Garbage-Collection ein neuer Block mit *eingesammelten* Pages neu beschrieben, dann wird unter Wear-Levelling-Aspekten derjenige als neuer Block ausgewählt, der noch wenige Lösch- / Schreibzyklen erfahren hat. Da sich durch Wear-Levelling-Aktivitäten auch die Zuordnung von LBA in PBA ändert, ist ein performantes Ändern in Echtzeit der Umsetzungstabelle notwendig.

Der Controller unterhält für das Wear-Levelling eine Tabelle, in der die Lösch- / Schreibzyklen jedes Blocks protokolliert werden.

[32] Vgl. Micheloni, R. / Marelli, A., a.a.O., S. 234

Die Art und Weise, wie Wear-Levelling durchgeführt wird, lässt sich in zwei grundsätzliche Arten unterscheiden:

- dynamisches Wear-Levelling bzw.
- statisches Wear-Levelling

Dynamisches Wear-Levelling findet statt, wenn neue Pages oder Pages durch Änderungsanforderung des Hosts in neue Blöcke geschrieben werden müssen und diese unter Wear-Levelling Gesichtspunkten bestimmt werden, ähnlich wie bei der oben beschriebenen Garbage-Collection. Wäre dies die einzige Wear-Levelling-Strategie, könnten dabei allerdings Blöcke, die sich nie oder sich äußerst selten ändernde Daten enthalten, nicht mit berücksichtigt werden, eine gleichmässige Verteilung wäre nicht möglich. Statisches Wear-Levelling dagegen erfasst auch solche Blöcke und sorgt damit für eine weitaus bessere Verteilung der Belastung durch Lösch- / Schreibzyklen. Diese Art von Wear-Levelling verursacht allerdings selbst Lösch- und Schreibzugriffe. Um diese zusätzlichen Lösch- / Schreibzyklen so gering wie möglich zu halten und dabei möglichst wenige Ressourcen des Controllers zu verbrauchen, sind ausgeklügelte Verfahren notwendig. Zwei solcher Verfahren, der

- *Evenness-Aware Algorithm* und der
- *Dual-Pool Algorithm*

sind Methoden, die unter Anwendung von unterschiedlichen Ansätzen eine gleichmäßige Verteilung der Lösch- / Schreibzyklen über alle Blöcke einer SSD erreichen und dabei auch diejenigen Blöcke mit einbeziehen, auf denen sich Daten mit wenigen Änderungen befinden. Sie im Detail[33] zu beschreiben, würde den Rahmen dieser Arbeit bei weitem sprengen.

Bei einem Anwendungsprofil, das sowohl häufiges Ändern von Daten als auch gleichzeitig das Vorhandensein von großen Mengen von sich nie ändernden Daten beinhaltet, ist das statische Wear-Levelling die größte Ursache für häufiges Verschieben von Blöcken innerhalb der SSD. Besteht der Inhalt einer SSD mehrheitlich aus sich häufig ändernden Daten, hat das statische Wear-Levelling weniger zu arbeiten, da das dynamische Wear-Levelling (incl. Garbage-Collection) die gleichmäßige Verteilung direkt beim Schreiben bzw. Ändern der Daten übernimmt.

2.1.2.5 Error-Correction-Code (ECC)

ECC ist ein Fehlerkorrekturverfahren, mit dem die Richtigkeit einer Information bei der Datenübertragung bzw. Datenspeicherung überprüft und gegebenenfalls auch kor-

[33] Vgl. Micheloni, R. / Marelli, A., a.a.O., S. 233-255

rigiert werden kann. Solche Verfahren werden in vielen Bereichen der IT eingesetzt, um die Zuverlässigkeit eines Speichers oder einer Übertragung zu erhöhen. Bei der Speicherung von Daten werden zu diesem Zweck vor dem Schreiben redundante Informationen in Form von zusätzlichen Bits (sogenannten *Parity Bits*) erzeugt und mit den Daten gespeichert. Beim Lesen der Daten werden diese Parity Bits erneut berechnet und mit den gespeicherten verglichen. Sind sie gleich, können die gelesenen Daten als fehlerfrei angesehen werden. Ist dies nicht der Fall, kann je nach Aufwändigkeit des eingesetzten Verfahrens der oder die Fehler korrigiert werden. Kann nicht zweifelsfrei erkannt werden, was zu korrigieren ist, gilt der Lesevorgang als gescheitert.

Flashspeicher ist von Haus aus ein Speicher mit einer eher geringen Zuverlässigkeit[34]. Eine Ursache, die die Zuverlässigkeit eines Lesevorgangs beeinträchtigen kann, ist die Unsicherheit bei der Bestimmung der genauen Schwellspannung V_{TH}(siehe Abschnitt 2.1.1.1). Je kleiner die Strukturgröße der Zellen und je mehr MLC- und TLC-Speicherzellen (bei denen die Schwellspannung entsprechend der Anzahl von Bitzuständen pro Speicherzelle verschiedene Werte annehmen kann) Verwendung finden, desto wichtiger ist eine leistungsfähige und performante Fehlerkorrektur[35].

Bei SSDs werden vor allem die ECC-Verfahren nach Hamming, Reed-Solomon und nach Bose, Chaudhuri und Hocquenheim (BCH) eingesetzt[36].

Die zusätzlichen Parity Bits, die für ECC benötigt werden, befinden sich in der sogenannten *Spare Area* einer Page. Gängige Werte der Größe einer solchen Spare Area sind z.B. 64 Byte[37] bei einer Page von 2048 Byte oder 128 Byte[38] bei einer Page von 4096 Byte. In der Spare Area werden neben den Parity Bits für ECC (z. B. 3 Byte) auch Verwaltungsinformationen für andere Controller-Funktionen wie das Bad-Block Management gespeichert.

Beim ECC wie auch bei den vorgenannten Controller-Funktionen

- Wear-Levelling
- Garbage-Collection
- Umsetzung Logical-Block-Adressen nach Physical-Block-Adressen

muss das Design des Controllers die verschiedenen, konkurrierenden Anforderungen nach Performance, Zuverlässigkeit und niedrigem Ressourcenverbrauch (z. B. Verbrauch von CPU, Strom, usw.) berücksichtigen, was nicht immer einfach ist. Wie schon

[34] Vgl. Micheloni, R. / Marelli, A., a.a.O., S. 259
[35] Vgl. Micheloni, R. / Crippa, L., a.a.O., S. 43
[36] Vgl. Spansion: What Types of ECC Should Be Used on Flash Memory? https://www.spansion.com/Support/Application%20Notes/Types_of_ECC_Used_on_Flash_AN.pdf [2015-12-16]
[37] Vgl. Micheloni, R. / Crippa, L., a.a.O., S. 27
[38] Vgl. Micheloni, R. / Marelli, A., a.a.O., S. 290

erwähnt, sind die in den Controller-Funktionen letztendlich verwendeten Algorithmen nicht veröffentlicht und gelten als vertraulich.

2.2 Gegenüberstellung von SSD zu HDD

„SSDs are storage devices that use nonvolatile solid-state memory (usually flash memory) to emulate HDDs.“[39].

Dass SSDs überhaupt als Alternative zu HDDs auf dem Datenträgermarkt angenommen wurden, ist vor allem dem Umstand zu verdanken, dass eine SSD sich nach aussen so verhalten kann, als wäre sie eine HDD. Um dieses Verhalten trotz der besonderen Eigenschaften von Flashspeicher wie

- das Löschen vor jedem Schreiben,
- die begrenzte Anzahl von Lösch- / Schreibzyklen und
- die notwendige Umsetzung von logischen Sektoren des Betriebssystems in die physikalischen Pages, Sub-Pages und Blöcke der SSD

zu erreichen, ist ein enormer Aufwand notwendig, was in den Abschnitten in 2.1.2 zu den Controller-Funktionen deutlich wurde.

Wie in der Einleitung bereits erwähnt, waren die ersten SSDs, die ab ca. 2007 angeboten wurden, lange Zeit

- viel teurer als HDDs
- hatten deutlich geringere Kapazitäten als HDDs
- und die niedrigeren Latenzzeiten alleine hätten für eine Akzeptanz nicht gereicht.

Nur die Kompatibilität bei

- den Interfaces (SATA und SAS)
- den Formfaktoren (vor allem 2.5“)
- der nach aussen identischen logischen Struktur der LBAs

[39] „SSDs sind Speichergeräte, die nicht-flüchtigen Festwertspeicher (in der Regel Flash-Speicher) verwenden, und dabei HDDs emulieren.“
IBM Knowledge Center: Considerations for solid-state drives (SSDs)
http://www-01.ibm.com/support/knowledgecenter/HW4M4/p8ebj/arebjsolidstatedrives.htm
[2015-12-05]

machten sowohl den Einsatz in neuen Systemen als auch einen Austausch in bestehenden Geräten *ohne Anpassungen* (d. h. keine neuen Treiber, keine anderen Kabel) möglich. Inzwischen haben sich die SSDs bei Preis und Kapazität den HDDs weiter genähert, bei der Performance sind HDDs in der Regel abgeschlagen.

Trotz der o. g. Kompatibilität gibt es Unterschiede zwischen SSDs und HDDs, die in den nächsten Abschnitten detaillierter erläutert werden.

2.2.1 Formfaktoren

Die Baugrößen und -formen bei HDDs sind weitgehend bestimmt durch den Platzbedarf der verwendeten Scheiben, des Spindelmotors und des Aktuators. Deshalb sind hier nur Veränderungen möglich, wenn auch die genannten Bauteile weiter verkleinert werden. Gängige Formfaktoren bei HDDs sind 3,5“ und 2,5“, seltener 1,8“.

Aus den oben genannten Kompatibilitätsgründen sind SSDs vor allem im Formfaktor 2,5“ erhältlich. Durch die ausschließliche Verwendung von elektronischen Bauteilen, die (zumindest im Prinzip) in beliebiger Anordnung auf einem PCB aufgelötet werden, sind aber auch völlig neue Bauformen und vor allem Bauhöhen möglich, die dann ihrerseits nicht mehr designmitbestimmend sind, z. B. bei Notebooks mit sehr geringer Bauhöhe.

2.2.2 Interfaces

Die sowohl bei HDDs als auch bei SSDs mehrheitlich verwendeten Interfaces SATA (vor allem in Desktops und in Consumer-PCs) und SAS (für Serverapplikationen) sind ursprünglich für HDDs entwickelt und im Lauf der Zeit erweitert worden. Von

- SATA 1.0 mit 1.5 Gbps über
- SATA 2.0 mit 3.0 Gbps bis zum heutigen Stand
- SATA 3.0 mit 6 Gbps[40].

bzw.

- SAS mit bis zu 12 Gbps[41].

[40] Vgl. SATA Product Features https://sata-io.org/sites/default/files/images/SATA-IO-English-Brochure-May-2009.pdf [2016-01-01]

[41] Vgl. SAS Technology Roadmap - August 2015 http://www.scsita.org/library/presentations/SAS%20Technology%20Roadmap%20-%20August%202015,%20with%20notes.pdf [2016-01-01]

Aktuelle SSDs stoßen bei diesen Interfaces inzwischen an deren Performance-Grenzen, so dass mittlerweile die Interfaces einen Flaschenhals für SSDs darstellen. Um diesen zu umgehen, wurden speziell für SSDs neue Interfaces entwickelt, mit denen SSDs direkt an den PCIe Host-Bus angeschlossen werden. Damit sind Transferraten von bis zu 4 GBytes/s möglich[42]. SSDs mit PCIe-Interface sind in der Form einer PCIe-Karte erhältlich, in Notebooks (vor allem in solche mit sehr geringer Bauhöhe) werden PCIe-SSDs im Format M.2 bzw. mSATA eingebaut, die in Größe und Form eher einem gewöhnlichen Speicherriegel ähneln, als dem Gehäuse einer HDD.

Für SSDs mit einem Anschluss in der Form von mSATA bzw. M.2 gibt es Adapter auf den normalen SATA-Anschluss. Um *forensische Images* (siehe Abschnitt 3.5 *Erstellen einer forensischen Kopie*) von zu untersuchenden Datenträgern erstellen zu können, werden sogenannte *Writeblocker* (Interfaces, die nur Lesezugriffe zur SSD zulassen) an den SATA-Anschluss angeschlossen. Für SSDs als PCIe-Karte sind derzeit keine Writeblocker verfügbar.

2.2.3 Sektor- / Blockmanagement

2.2.3.1 Zuordnung Logical Block Adressen zu physikalischen Einheiten

Wie bereits in Abschnitt 2.1.2.2 erläutert, sehen aktuelle Betriebssysteme den Speicher ihrer angeschlossenen Datenträger als serielle Aneinanderreihung der kleinsten Speichereinheiten, ohne dabei die interne Organisation des jeweiligen Gerätes kennen und berücksichtigen zu müssen. Diese Einheiten werden auch logische Blöcke (Logical Blocks) genannt, und sind, je nach Betriebssystemsplattform und -version, entweder 512 Bytes oder 4096 Bytes gross. Die Abstraktion von der Geometrie der Datenträger, die durch die Verwendung der Logical Block Adressierung (LBA) möglich ist, macht die Adressierung unabhängig z. B. von der bei HDDs gegebenen Aufteilung in Zylinder (Cylinders), Spuren (Tracks) und Sektoren (Sectors) oder der bei SSDs üblichen Organisation in Pages und Blöcke.

Dabei gibt es bei HDDs eine Zuordnung der logischen Blöcke des Betriebssystems zu den einzelnen physikalischen Sektoren. Beginnend mit Sektor 1 in Spur 0 Zylinder 0[43] (dem „äussersten“ Sektor), der die logische Blockadresse 0 (LBA 0) erhält, werden die Sektoren weiter „nach innen“ gezählt, bis zum letzten Sektor der letzten Spur im letzten Zylinder, der die höchste logische Blockadresse $\mathrm{LBA}_{\mathrm{max}}$ erhält. Diese lässt sich berechnen mit:

[42] Vgl. Micheloni, R. / Marelli, A., a.a.O., S. 39

[43] Zu beachten ist, dass bei HDDs üblicherweise Zylinder und Spuren ab 0, jedoch Sektoren ab 1 gezählt werden.

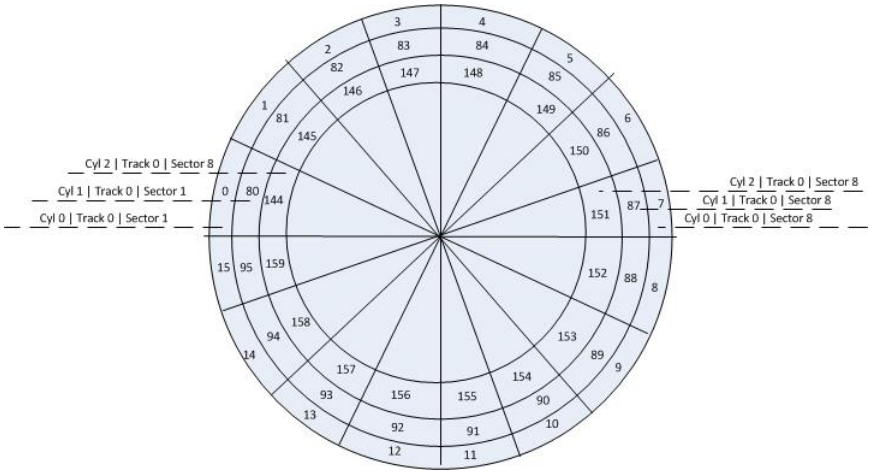

Abbildung 2.11: Zählung LBA und Zusammenhang mit Zylindern, Spuren und Sektoren

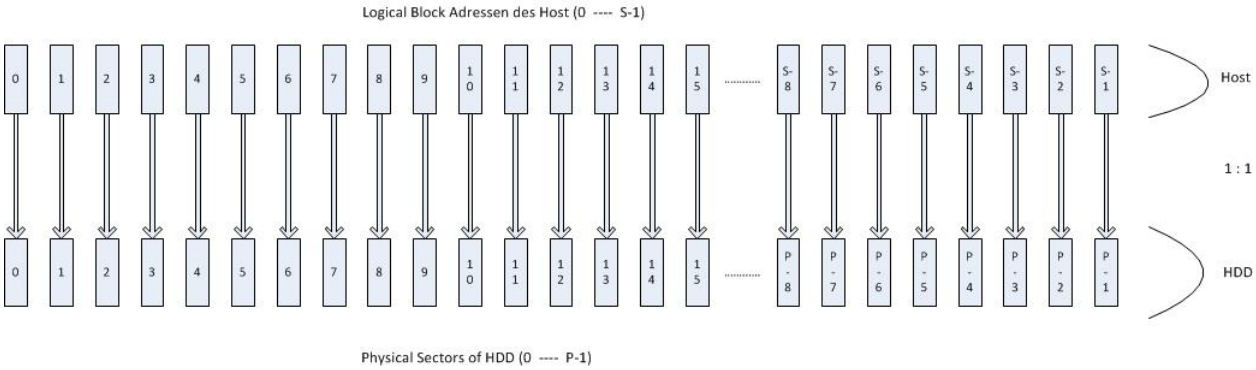

Abbildung 2.12: Zuordnung LBA zu physikalischen Sektoren bei HDDs

$$LBA_{max} = (CYL \times TRACK \times SECT) - 1$$

wobei

- CYL die Anzahl aller Zylinder
- TRACK die Anzahl aller Spuren eines Zylinders und
- SECT die Anzahl aller Sektoren einer Spur ist.

Abbildung 2.11 zeigt vereinfacht die Sektoren der ersten 3 Zylinder der obersten Scheibe einer HDD mit 2 Platten, 4 Spuren pro Zylinder und 16 Sektoren pro Spur. Man erkennt, wie die Zählung der logischen Blöcke mit 0 bei Cyl 0 | Track 0 | Sect 1 beginnt und bei Cyl 2 | Track 0 | Sect 15 mit LBA 159 endet. Die in der Abbildung „fehlenden“ Sektoren befinden sich auf der Rückseite der gezeigten Scheibe und auf Vorder- und Rückseite der 2. Scheibe.

Da die Reihenfolge von Zylindern, Spuren und Sektoren einer HDD sich nicht ändert[44], bleibt auch die Zuordnung einer logischen Blockadresse zu einer physikalischen Adresse fest. In Abbildung 2.12 ist diese 1 : 1 Zuordnung dargestellt. Die physikalische Adresse eines Sektors (C | T | S) lässt sich aus einer gegebenen logischen Blockadresse (LBA) berechnen mit:

$C = LBA/(TRACK \times SECT)$

$T = (LBA/SECT) \bmod TRACK$

$S = (LBA \bmod SECT) + 1$

wobei

- CYL die Anzahl aller Zylinder
- TRACK die Anzahl aller Spuren eines Zylinders und
- SECT die Anzahl aller Sektoren einer Spur
- C der physikalische Zylinder
- T die physikalische Spur und
- S der physikalische Sektor ist.

Diese Berechnungen sind insofern vereinfacht, da moderne HDDs keine einheitliche Sektordichte über alle Spuren haben, sondern sich in den äusseren Spuren mehr Sektoren befinden (da mehr Platz vorhanden ist) als in den inneren Spuren (da weniger Platz vorhanden ist). Die HDD ist dabei in mehrere Zonen eingeteilt, innerhalb einer Zone ist die Anzahl der Sektoren pro Spur konstant. Für die feste Zuordnung von logischen Blockadressen zu physikalischen Sektoren ist dies aber unerheblich.

Im Gegensatz dazu gibt es bei SSDs keine feste Zuordnung von logischen Blockadressen des Betriebssystems zu den physikalischen Einheiten von (Sub-) Pages und Blöcken. Wie in Abschnitt 2.1.2.2 ausführlich beschrieben, ist diese Umsetzung ein interner Mechanismus der SSD, der ohne jegliche Einflussmöglichkeit des Hosts vollkommen im Verborgenen abläuft. Die Umsetzungstabelle befindet sich innerhalb der SSD, steht sie nicht zur Verfügung, liegen die gespeicherten Daten völlig unstrukturiert vor.

[44] Nicht berücksichtigt ist hier das Bad Block Management, das eine Verschiebung einzelner Sektoren innerhalb einer Spur bzw. innerhalb benachbarter Spuren bewirken kann.

2.2.3.2 Schreiben und Ändern von Daten

Die kleinste Einheit zum Lesen und Schreiben bei HDDs ist, wie schon erwähnt, ein Sektor, der eine Größe von 512 Bytes bzw. bei neueren HDDs von 4096 Bytes hat. Ein Sektor kann beliebig oft (neu) beschrieben werden, dabei ist es unerheblich, ob sich bereits Daten auf dem Sektor befinden oder nicht. Die einmal geschriebenen Daten bleiben so lange auf diesem Sektor erhalten, bis sie durch einen weiteren Schreibvorgang überschrieben werden.

Einen dezidierten Löschvorgang gibt es bei HDDs nicht. Sollen Daten auf einer HDD gelöscht werden, so dass sie für einen Lesevorgang nicht mehr zur Verfügung stehen, müssen die entsprechenden Sektoren explizit überschrieben werden, z. B. mit einer Folge von 0 oder 1 oder mit einem zufälligen Bitmuster.

Wie in den Abschnitten 2.1.1.4 (Schreiben von Daten) und 2.1.2.3 (Garbage-Collection) erläutert, können bei SSDs durch die Notwendigkeit, dass nur in gelöschte Pages geschrieben werden kann, die Daten einzelner Pages in weitere Pages kopiert werden. Die Pages mit den ursprünglichen Daten werden für das Betriebssystem ausgeblendet und sind deshalb auch nicht mehr im Zugriff. Bei häufigem Ändern von Daten können sich so mehrere Generationen an verschiedenen Orten innerhalb des Flashspeichers einer SSD befinden. Auch dieser Mechanismus ist nicht beeinflussbar, so dass keinerlei quantitative Aussage über das Vorhandensein von Kopien gemacht werden kann.

2.3 Unterschiede von SSDs und HDDs / Zusammenfassung

Aus den in Abschnitt 2.1 erläuterten Techniken und Funktionen einer SSDs und den in Abschnitt 2.2 gegenübergestellten Eigenschaften von HDDs ergeben sich folgende wesentlichen Unterschiede zwischen SSDs und HDDs:

- Formfaktoren: die kompaktere Bauweise von SSDs macht neue, kleinere Formfaktoren möglich
- Interfaces: die ursprünglich für HDDs entwickelten Interfaces SATA und SAS stellen für SSDs inzwischen einen Flaschenhals dar. Für neue und schnellere Interfaces (PCIe) stehen eventuell noch keine Möglichkeiten zur Verfügung, forensische Images zu erstellen.
- Blockmanagement: Kann nur direkt auf die jeweiligen Datenträgermedien zugegriffen werden (Plattenstapel bei HDDs, Flash-Chips bei SSDs), stehen bei HDDs

die Daten durch die feste Beziehung von LBA zu physikalischen Einheiten in ihrer kompletten Struktur zur Verfügung. Bei SSDs liegen die Inhalte der einzelnen Pages durch die fehlende Umsetzung von LBA nach PBA völlig unstrukturiert vor, d. h. in hohem Mass fragmentiert verteilt über den gesamten Flashspeicher[45].

- Schreiben / Ändern / Löschen von Daten: Sowohl bei HDDs als auch bei SSDs[46] bleiben einmal geschriebene Daten erhalten, bis sie durch Initiative des Betriebssystems überschrieben werden. Bei SSDs können Kopien bzw. ältere Stände von Daten neben den aktuellen Daten selbst über den gesamten Flashspeicher verteilt sein. Diese sind für das Betriebssystem nicht sichtbar, da die entsprechenden (Sub-)Pages aus der Umsetzung LBA nach PBA ausgeblendet sind. Sie könnten nur direkt aus den Flash-Chips ausgelesen werden.

[45] Sowohl für HDDs als auch für SSDs ist das direkte Auslesen von Daten von den jeweiligen Medien äusserst aufwändig und nur von darauf spezialisierten Firmen wie z. B. *KrollOntrack* oder *SalvageData Recovery* durchführbar.

[46] Ausnahme bei SSD: wird die Funktion TRIM verwendet, ist dies bei SSDs nicht der Fall. Diese Funktion wird wegen ihres großen Einflusses in einem eigenen Abschnitt (3.2) behandelt.

3 Datenwiederherstellung bei SSD

Im vorigen Kapitel sind die technischen Grundlagen für die Mechanismen, die innerhalb einer SSD ablaufen, im Einzelnen erläutert worden. Es wurde auch erwähnt, dass die Details der Algorithmen, die letztendlich im Controller einer SSD implementiert sind, allerdings nicht dokumentiert und nicht veröffentlicht werden. Man muss also eine SSD auch als eine Art Black-Box betrachten, deren interne Arbeitsweise wenigstens teilweise im Dunkeln liegt[1].

Für forensische Untersuchungen an Datenträgern, die sich mit den *sichtbaren* Dateien beschäftigen, z. B. das Auswerten von Logdateien, Datenbanken, Bilddateien, Dokumenten usw., ist dieses Verhalten einer SSD als eine (teilweise) Black-Box eher unerheblich, da die SSD sich hier wie eine HDD verhält und das Betriebssystem eine transparente Sicht und den vollen Zugriff auf seine Daten hat. Im Gegensatz dazu sind für forensische Analysen gerade die Bereiche eines Datenträgers von Interesse, die im Verborgenen liegen, also z. B. direkt von Benutzern, dem Betriebssystem oder Programmen gelöschte Dateien[2] oder versteckte Daten (File Slack / Drive Slack[3]).

Zunächst wird der Einfluss der vorgenannten SSD-Controllerfunktionen auf diese Bereiche untersucht. Diese Controllerfunktionen lassen sich weder vom Host noch an der SSD abschalten oder auf irgendeine Art und Weise beeinflussen. Die TRIM-Funktion dagegen, die, abhängig vom Betriebssystem, der eingesetzten SSD und des verwendeten Host-Controllers, seit ca. 2009 verfügbar ist, stellt eine spezielle Verbindung des Betriebssystems zum SSD-Controller her. Die Auswirkungen auf den Umgang mit gelöschten Dateien innerhalb einer SSD sind enorm und werden deswegen in einem eigenen Abschnitt behandelt.

Welchen Einfluss das sogenannte Over-Provisioning und die in vielen SSDs verfügbare (Selbst-) Verschlüsselung auf forensische Analysen haben, ist Thema weiterer Abschnitte; zum Schluss dieses Kapitels wird noch untersucht, was beim Erstellen einer forensischen Kopie eines SSD-Datenträgers zu beachten ist.

[1] Vgl. Bonetti, G. / Viglione, M. / Frossi, A / Maggi, F. / Zanero, S. (2013): A Comprehensive Black-box Methodology for Testing the Forensic Characteristics of Solid-state Drives. 29th Annual Computer Security Applications Conference (ACSAC '13), (pp. 269-278): Abs. 2.1 http://www.syssec-project.eu/m/page-media/3/ssdforensics-ACSAC-final.pdf [2016-01-02]

[2] Vgl. Geschonneck, A. (2008): Computer Forensik, 3. Aufl., Heidelberg: dpunkt verlag, S. 101

[3] Vgl. ebenda, S. 104

3.1 Auswirkungen der SSD-Controllerfunktionen auf Datenwiederherstellung (ohne TRIM)

Die in Abschnitt 2.1.2 vorgestellten Funktionen eines SSD-Controllers regeln innerhalb einer SSD das Datenmanagement. Sie sind größtenteils den besonderen Eigenschaften des Flashspeichers geschuldet und sind daraufhin optimiert, dem Betriebssystem gegenüber eine HDD mit hoher Performance zu emulieren. Beim Design dieser Funktionen steht der effiziente Umgang mit (aus Sicht des Controllers) aktiven Daten im Fokus. Wie sich diese Funktionen auf (aus Sicht des Betriebssystems möglicherweise) gelöschte oder nicht benutzte Bereiche und die Möglichkeiten, sie wiederherzustellen, auswirkt, ist eher nebensächlich.

Bei Fragen zur Wiederherstellung von gelöschten Objekten sind von besonderem Interesse:

- vom Benutzer / Betriebssystem eines PCs gelöschte Dateien
- die Möglichkeiten zur Wiederherstellung nach einem (High-Level-) Format einer SSD
- der Umgang mit möglichen Daten-Fragmenten wie File Slack / Drive Slack / RAM Slack
- bei NTFS: MFT und sehr kleine Dateien[4]

Löschen von Dateien / Formatieren eines Datenträgers Wenn Dateien des Filesystems gelöscht werden, dann markiert der Host die Einträge in den Verwaltungseinheiten des Filesystems als gelöscht, die Einträge als solche und vor allem die Daten selber werden vom Filesystem zunächst nicht angetastet. Sie sind weiter an dem bisherigen Ort praktisch unverändert vorhanden. Aus der Sicht des Betriebssystems sind diese logischen Blöcke der gelöschten Datei(en) wieder frei verfügbar und können bei Bedarf überschrieben werden. Der angeschlossene Datenträger (HDD oder SSD) bzw. dessen Controller bekommt davon nichts mit.

Führt der SSD-Controller nun (dynamisches) Wear-Levelling und / oder Garbage-Collection durch, stellen sich die vom Host als gelöscht angesehenen logischen Blöcke für den Controller weiterhin als gültige Pages dar, er wird sie beim Kopiervorgang der Garbage-Collection genau so behandeln wie andere gültige Pages. Dasselbe gilt für eine

[4] Vgl. Microsoft Corporation: MasterFileTable https://msdn.microsoft.com/de-de/library/windows/desktop/aa365230(v=vs.85).aspx [2016-01-02]

mit Quick-Format gelöschte SSD, auch hier erhält der SSD-Controller und damit der Garbage-Collector keine Information über den Löschvorgang[5].

Slack-Bereiche Sogenannte Slack-Bereiche entstehen im Filesystem, da in den seltensten Fällen die Größe einer Datei exakt der Größe eines Clusters oder einem Vielfachen davon entspricht. Die Größe des freien Platzes zwischen dem Ende der Datei und dem Ende des letzten Clusters der Datei entspricht der Differenz aus der Summe der von einer Datei belegten Cluster und der tatsächlichen Dateigröße. Dieser Bereich wird *File Slack* genannt, der sich wiederum in den *RAM Slack* (Rest des letzten logischen Blocks, der noch teilweise von der Datei belegt ist) und den *Drive Slack* (die nicht benutzten logischen Blöcke des letzten Clusters der Datei) aufteilt. Abbildung 3.1 zeigt File, RAM und Drive Slack am Beispiel einer Datei, deren Dateiende innerhalb des 6. logischen Blocks des letzten Clusters liegt.

Die Daten des RAM Slack werden benötigt, um den letzten von einer Datei belegten logischen Block auf die volle Größe aufzufüllen. Diese Füllbytes können je nach Betriebssystem z. B. Fragmente aus einem Bereich des Hauptspeicher sein und damit wertvolle Informationen beinhalten, die bei einer forensischen Untersuchung von großem Wert sein können, u. U. Fragmente von Dokumenten, Bildern oder aber Daten wie Passwörter, die gar nie auf einen Datenträger gespeichert werden. Bei einer Anzahl von 100.000 Dateien auf einem Datenträger und einer Sektorgröße von 512 Byte kann die Gesamtgröße des RAM Slack bis zu 50 MByte groß sein, bei einer Sektorgröße von 4096 Byte bis zu 400 MByte.

Beim Schreiben einer Datei werden die logischen Blöcke, die den Drive Slack bilden, nicht neu geschrieben, sondern der beim Zeitpunkt des Schreibens darin befindliche Inhalt dieser Blöcke *ist* der Drive Slack[6].

Bezüglich des RAM Slack verhält sich eine SSD analog einer HDD. Da sowohl Lese- als auch Schreibvorgänge immer auf Basis einer Page stattfinden, wird der RAM Slack auch bei Kopiervorgängen, die aufgrund von Garbage-Collection oder Wear-Levelling durchgeführt werden, immer in diesem logischen Block (aus Sicht des Betriebssystems) beibehalten. Er steht somit für forensische Untersuchungen zur Verfügung.

Anders sieht es bei dem Drive Slack aus. Da bei SSDs nur auf Pages in vorher gelöschten Blöcken geschrieben werden kann, sind die (Sub-) Pages, die nach dem Schreiben einer Datei einen möglichen Drive Slack bilden, immer leer. Bei HDDs können an dieser Stelle Reste von früher geschriebenen und gelöschten Dateien sein.

[5]Vgl. Bonetti, G., a.a.O., Abs. 4.2
[6]Vgl. Geschonneck, A., a.a.O, S. 105

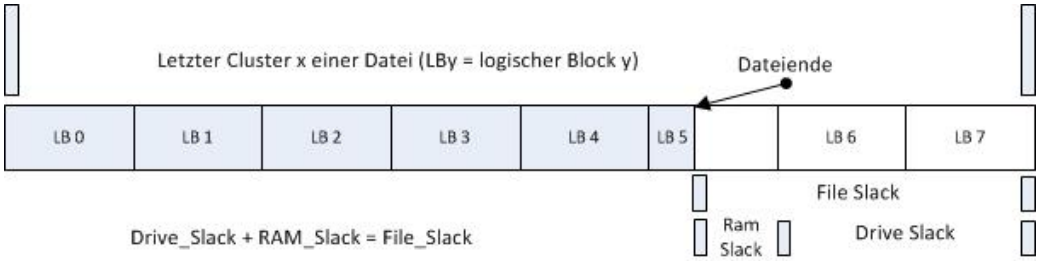

Abbildung 3.1: Beispiel für Slack-Bereiche einer Datei

NTFS: MasterFileTable und kleine Dateien Die MasterFileTable (MFT) bei NTFS ist die Datei mit den Verwaltungseinträgen des Filesystems, sie wird beim Erstellen einer Partition zweifach angelegt. Pro Datei bzw. Verzeichnis wird ein Eintrag hinzugefügt, beim Löschen einer Datei oder eines Verzeichnisses wird der Eintrag als frei markiert und kann neu verwendet werden. Eine MFT kann demnach nur größer werden, nicht kleiner. Werden bei NTFS kleine Dateien gespeichert (< 982 Byte), so werden die Inhalte dieser Dateien direkt in der MFT abgelegt, und nicht ausserhalb in eigenen Clustern. Beim Löschen solch einer Datei in einer SSD verhält es sich hier genau wie bei einer HDD, die Datei lässt sich einfach wiederherstellen[7].

Bei der Frage zur Wiederherstellbarkeit stellen unter Umständen kleine Dateien eine Besonderheit dar, die aufgrund ihrer geringen Größe nur im Read- / Write-Cache bearbeitet werden. Existieren diese Dateien nur während eines Power-Up-Zyklus der SSD bzw. des Host, d. h. die Dateien werden vor dem Shutdown des Host vom Betriebssystem wieder gelöscht, ist es möglich, dass sie gar nie in den Flashspeicher geschrieben wurden und dort natürlich auch keine Spuren hinterlassen haben. Somit können sie nicht wiederhergestellt werden (siehe Abschnitt 2.1.2.1 Read- / Write-Cache)[8].

Wie man sieht, werden allein durch das Ausführen von Funktionen des SSD-Controllers, vor allem Garbage-Collection und Wear-Levelling, *keinerlei* Daten auf einer SSD gelöscht, egal ob diese Daten aus der Sicht des Betriebssystems gültige Daten oder gelöschte Daten sind. Es gibt Veröffentlichungen[9]über forensische Versuche, bei denen die Autoren durch Versuche herausgefunden haben, dass Garbage-Collection alleine in der Lage wäre, Dateien, die vom Betriebssystem gelöscht wurden, auch auf der SSD zu löschen. Explizit dieser Aussage wird in weiteren Veröffentlichungen[10,11] widersprochen.

[7]Vgl. Microsoft TechNet: The Four Stages of NTFS File Growth.
http://blogs.technet.com/b/askcore/archive/2009/10/16/the-four-stages-of-ntfs-file-growth.aspx [2015-12-15]
[8]Vgl. Bonetti, G., a.a.O., Abs. 3.2
[9]Vgl. Bell, G. / Boddington, R. (2010): Solid State Drives: The Beginning of the End for Current Practice in Digital Forensic Recovery?
Journal of Digital Forensics, Security and Law, 5 (3), (pp. 1-20): Abs. 4.1
http://ojs.jdfsl.org/index.php/jdfsl/article/viewFile/21/45 [2015-12-18]
[10]Vgl. Bonetti, G., a.a.O., Abs. 4.2
[11]Vgl. Gubanov, Y. / Afonin, O.: SSD Forensics 2014 Recovering Evidence from SSD Drives: Under-

3.2 Funktionsweise und Auswirkungen der *TRIM*-Funktion

3.2.1 Grundlagen

Mit der Einführung der Funktion TRIM in 2009 in Verbindung mit SSDs gab es erstmals die Möglichkeit, dass das Betriebssystem bzw. das Filesystem mit einer angeschlossenen SSD *kommunizieren* konnte. Durch die Abstraktion der physikalischen Speicherplätze des Flashspeichers in logische Blockadressen des Betriebssystems erhielt der Controller einer SSD bis dahin keine Informationen darüber, wenn das Betriebssystem Dateien in seinem Filesystem gelöscht hat. Zwar liegen die Verwaltungsdateien der verschiedenen Filesysteme ebenfalls auf dem Datenträger, die gesicherte Interpretation der Strukturen des gerade verwendeten Filesystems durch den SSD-Controller selbst würde jedoch schon an der großen Anzahl von verschiedenen Filesystemen und deren Versionen scheitern.

TRIM ist aber nicht überall verfügbar, sondern funktioniert nur, wenn sowohl

- die SSD,
- das Betriebssystem und gegebenenfalls das zugehörige Filesystem als auch
- der SATA - Controller

die TRIM-Funktionalität unterstützen und im Falle der Betriebssysteme und SATA-Controller die Funktion auch eingeschaltet ist

Aktuelle SSDs unterstützen TRIM generell, im Einzelfall ist das in den Laufwerks-Spezifikationen dokumentiert. Folgende Kombinationen von Betriebssystemen und Dateisystemen ermöglichen die Verwendung von TRIM[12]:

- Windows in Version 7 / 8 / 10 mit NTFS
- Windows Server 2008 R2 mit NTFS
- Linux ab Kernel 2.6.33 mit ext4
- Mac OS X Snow Leopard v. 10.6.8 (nur vorinstallierte SSDs)

Der verwendete SATA-Controller muss außerdem im Modus *AHCI* betrieben werden.

standing TRIM, Garbage Collection and Exclusions, S. 11
http://belkasoft.com/download/info/SSD%20Forensics%202014.pdf [2016-01-02]

[12] Vgl. Samsung Electronics Co.: TRIM for Data Centers: http://www.samsung.com/global/business/semiconductor/minisite/SSD/downloads/document/Samsung_SSD_845DC_09_TRIM.pdf [2015-12-13]

3.2.2 Funktionsweise

TRIM ist eine ATA Funktion[13], über die das Betriebssystem dem angeschlossenen Datenträger (SSD) eine Liste mit von ihm gelöschten Logischen Blockadressen mitteilen kann. Mit Hilfe dieser Information kann der SSD-Controller die Garbage-Collection und das Wear-Levelling deutlich effizienter ausführen. Der Controller kann die vom Betriebssystem gelöschten logischen Blöcke genau so als *stale* (inaktiv) markieren wie die Pages, die durch Schreibvorgänge (Ändern) inaktiv geworden sind. Beim nächsten Durchlauf der Garbage-Collection müssen beim Zusammenfassen der aktiven Pages diese Pages nicht mehr berücksichtigt werden, womit weniger Pages auf dem Flashspeicher bewegt werden, was Performance spart und unnötige Lösch- / Schreibzyklen vermeidet (siehe Abschnitt 2.1.2.3 Garbage-Collection).

Das TRIM-Command wird nicht nur dann an den SSD-Controller geschickt, wenn das Betriebssystem Dateien gelöscht hat, sondern auch beim Ausführen von Formatierungs- bzw. Partitionierungsaktionen. Das bedeutet, dass beim Formatieren oder Löschen einer gesamten Partition der SSD-Controller alle sich darin befindenden logischen Blöcke löschen kann[14].

Das Betriebssystem hat aber weiterhin keinen Einfluss auf den Zeitpunkt oder sonstige Parameter, mit denen die Garbage-Collection ausgeführt wird. Es übermittelt nur die Adressen und bekommt auch keine Rückmeldung über etwaige Aktionen innerhalb der SSD. Es ist sozusagen nur unterstützend tätig.

3.2.3 Auswirkungen

Wenn die o. g. Voraussetzungen (kompatibles Betriebssystem, SSD und SATA-Controller) alle erfüllt und die Funktionen richtig implementiert sind, kann die TRIM-Funktion eine enorme Auswirkung auf den Umgang eines SSD-Controllers mit vom Betriebssystem gelöschten logischen Blöcken haben. Auf einer SSD, bei der TRIM ständig angewendet wird, sind keine Bereiche mit Dateien mehr vorhanden, die nach dem Löschen wiederhergestellt werden können. TRIM hat wahrscheinlich den größten negativen Einfluss auf forensische Untersuchungen bei SSDs, da es genau das verhindert, was bei einer forensischen Analyse den größten Nutzen bringt: die Wiederherstellung von logisch gelöschten, aber auf den Speichermedien noch vorhandenen Daten.

[13] Vgl. ATA/ATAPI Command Set 2, Abs. 7.10.3.2: TRIM http://www.t13.org/documents/uploadeddocuments/docs2009/d2015r2-ataatapi_command_set_-_2_acs-2.pdf [2015-12-29]
[14] Vgl. Gubanov, Y., a.a.O., S.4

SSD	Filesystem	Files written	Files recovered	Percentage
SSD A	NTFS	112790	0	0,00%
SSD A	ext4	110322	0	0,00%
SSD B	NTFS	101155	71607	70,79%
SSD B	ext4	99475	0	0,00%
SSD C	NTFS	112192	0	0,00%
SSD C	ext4	110124	0	0,00%

Tabelle 3.1: Files recoverability test results

In mehreren Veröffentlichungen[15,16] wurde das Verhalten von SSDs bei aktiviertem TRIM und dem Löschen von Dateien untersucht. Mehrere SSDs wurden mit Dateien gefüllt, die Dateien wurden anschließend gelöscht und in verschiedenen Zeitabständen wurde ermittelt, wie viele der gelöschten Dateien wiederhergestellt werden konnten. Die meisten Tests lieferten die Resultate, die erwartet wurden. Allerdings gab es auch Ergebnisse, die so nicht erklärbar waren. Tabelle 3.1 zeigt als Beispiel die Ergebnisse eines *File recoverability test*: bis auf die SSD B in Verbindung mit dem Filesystem NTFS (Wiederherstellungsrate 70,79%) konnten keine Dateien wiederhergestellt werden. In einer Veröffentlichung von Gubanov und Afonin[17] vermuten die Autoren, dass fehlerhafte Implementierungen, vor allem von TRIM bzw. Garbage-Collection, aber auch von den sonstigen Controller-Funktionen, für die nicht erwarteten Ergebnisse verantwortlich sein können. In derselben Veröffentlichung wird deswegen die Empfehlung ausgesprochen, bei forensischen Untersuchungen zuerst zu ermitteln, ob Funktionen wie TRIM überhaupt aktiv sind bzw. waren. Es gibt einige Konstellationen, bei denen TRIM gar nicht verwendet wird bzw. funktioniert, dadurch können durchaus wider Erwarten viele forensische Spuren gefunden und gesichert werden. Solche Konstellationen sind u.a.:

- SSDs mit PCIe-Interface
- RAID-Systeme
- inkonsistente Partitionen / Volumes
- SSDs, die über USB angeschlossen sind

[15]Vgl. Nisbet, A. / Lawrence, S. / Ruff, M. (2013): A Forensic Analysis And Comparison Of Solid State Drive Data Retention With Trim Enabled File Systems.
11th Australian Digital Forensics Conference (2013), (pp. 103-111).
http://ro.ecu.edu.au/cgi/viewcontent.cgi?article=1124&context=adf [2016-01-01]
[16]Vgl. Bonetti, G., a.a.O.
[17]Vgl. Gubanov, Y., a.a.O.

3.3 Funktionsweise und Auswirkungen von *Over-Provisioning*

3.3.1 Funktionsweise

In Abschnitt 2.1.2.3 Garbage-Collection wurde ein Szenario beschrieben, bei dem durch viele Schreibvorgänge nahezu alle Pages des Flashspeichers mit aktiven und inaktiven Daten gefüllt sind. Ist nun bei hoher Belastung der SSD der Controller nicht mehr in der Lage, mit Garbage-Collection inaktive Pages zusammenzufassen und die entsprechenden Blöcke nach dem Löschen für neue Schreibvorgänge zur Verfügung zu haben, oder beinhalten die Pages des Flashspeichers überwiegend aktive Daten, droht die Schreibperformance dramatisch einzubrechen, da der Controller beim Schreiben bzw. Ändern von Pages das aufwändige Schreibverfahren *read, modify, erase, write* durchführen muss[18]. Dadurch steigt die sogenannte Write-Amplifikation (siehe Erklärung und Definition in Abschnitt 2.1.1.4). Um hier den SSD-Controller und die Garbage-Collection zu unterstützen und gute Performancewerte zu erhalten, wird von vielen Herstellern sogenanntes *Over-Provisioning* angewendet.

Over-Provisioning bedeutet, dass der Flashspeicher größer ist, als die vom Host aktuell genutzte Kapazität der SSD. Der Bereich, der für Over-Provisioning verwendet wird, kann sich aus verschiedenen Teilen zusammensetzen, wie Abbildung 3.2 zeigt. Dies ist zum einen ein Bereich, der vom Hersteller dafür reserviert ist, nicht verändert werden kann und zu keiner Zeit in Sicht oder im Zugriff des Hosts ist (*statisches* oder *echtes* Over-Provisioning). Zum anderen kann der User weiteren Speicherplatz für (dynamisches) Over-Provisioning bereitstellen, indem er die ihm zur Verfügung stehende Kapazität nicht ausnutzt und beim Partitionieren einen beliebig großen Bereich freilässt. Bei SSDs, die HPA (Host Protected Area) unterstützen, kann damit zusätzlicher Platz für Over-Provisioning bereitgestellt werden, der dann ebenfalls vom Host nicht mehr gesehen wird (der aber natürlich wieder zurückgenommen werden kann).

Die durch das Over-Provisioning zusätzlich zur Verfügung stehenden Blöcke und Pages können vom SSD-Controller für Schreib- und Änderungsaufträge direkt benutzt werden, durch das Umsetzen von physikalischen in logische Adressen werden diese Pages und Blöcke bei Verwendung für den Host transparent zugreifbar und die anderen, nicht mehr aktiven Pages werden in den logischen Pool des Over-Provisioning verschoben, wo sie bei Gelegenheit gelöscht werden, um wieder für neue Schreibvorgänge zur Verfügung zu stehen. Durch die zwangsläufige Verkleinerung der nutzbaren Kapazität steht immer ein Bereich zur Verfügung, der nicht mit aktiven Pages gefüllt werden kann.

[18] Vgl. Micheloni, R. / Marelli, A., a.a.O., S. 72

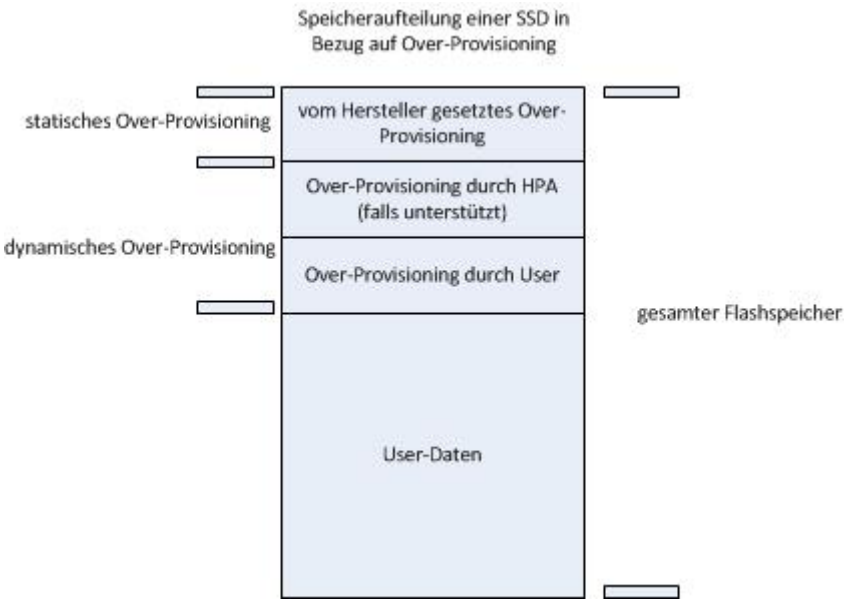

Abbildung 3.2: Speicherbereiche bei Over-Provisioning

Berechnet wird die Größe von Over-Provisioning nach folgender Formel:

$$\text{Over-Provisioning in \%} = \frac{\text{Over-Provisioning in GB}}{\text{Netto-Kapazität der Partition(en) in GB}}$$

Gängige voreingestellte Größen für Enterprise-SSDs sind 30%, für Consumer-SSDs sind ca. 7% -10% üblich. Diese Zahlen sind nicht zufällig gewählt. Bei Consumer-SSDs entsprechen die 7% ungefähr der Differenz von Gbyte und GiByte[19], bei Enterprise-SSDs kommt zu den 7% ein weiterer Bereich hinzu, der dem höheren Lastprofil, dem eine Enterprise-SSD ausgesetzt ist, geschuldet ist und auch entsprechend vermarktet wird. Letztendlich hängt der Wert des Over-Provisioning davon ab, welchen maximalen Wert von Write-Amplifikation man im Falle einer fast vollen SSD nicht überschreiten will.

Zu beachten ist, dass die dynamischen Bereiche, die für das Over-Provisioning bereitgestellt werden (also HPA und freier Platz durch kleinere Partitionen), vor dem Bereitstellen ausschließlich gelöschte Blöcke enthalten, damit der Controller sie für Garbage-Collection und Wear-Levelling vollumfänglich nutzen kann.

3.3.2 Auswirkungen

Zur Erhaltung der Schreibperformance auch bei steigendem Füllgrad des genutzten Flashspeichers ist das Over-Provisioning eine geeignete Methode. Unter forensischen Gesichtspunkten ist zu beachten, dass Pages, die bei der Garbage-Collection bzw.

[19] 1 GByte = 1000000000 Byte, 1 GiByte = 1024*1024*1024 Byte = 1073741824 Byte

dem Wear-Levelling in den Teil des vom Hersteller reservierten Bereichs des Over-Provisioning kopiert werden, nur über einen direkten Zugriff auf den Flash-Chip für eine forensische Untersuchung zur Verfügung stehen. Die anderen, dynamischen Bereiche sind für den Host sichtbar und unterliegen der gleichen Behandlung von aktiven und inaktiven Pages wie in den Abschnitten 3.1 und 3.2 beschrieben.

3.4 Integrierte Verschlüsselung bei SSD

Viele der heute[20] verfügbaren SSDs bieten die Möglichkeit, die auf ihnen zu speichernden Daten beim Speichern zu verschlüsseln. Anders als die schon seit längerem verfügbaren Softwareverschlüsselungen für Datenträger wie *TrueCrypt* oder *PGP Whole Disk Encryption,* die letztendlich Applikationen des Hosts sind, findet die Verschlüsselung für SSDs komplett innerhalb der SSD selbst statt. Sie ist eine weitere Funktion des Controllers und ist für den Host (wie die anderen Controller-Funktionen) vollkommen transparent. Diese Art der Verschlüsselung ist keine Erfindung für SSDs, es gibt sie sowohl für HDDs als auch für SSDs, bei SSDs ist sie aber deutlich weiter verbreitet[21]. Datenträger, bei denen der gesamte Verschlüsselungsvorgang innerhalb des Gerätes durchgeführt wird, heißen auch *Self-Encrypting Disk* oder *Self-Encrypting Drive* (SED).

Ein großer Sicherheitsvorteil von SEDs ist, dass die Schlüssel, die für die Verschlüsselung der Daten verwendet werden, sich zu jeder Zeit innerhalb der SSD-Controller-Logik befinden, also nie in den Speicher oder in Register der CPU übertragen werden. Damit können sie auch nicht von dort ausgelesen werden. Als Verschlüsselungsstandard wird von den meisten SEDs AES-256[22] verwendet, der als äusserst sicher gilt.

Eine SSD, die eine solche Verschlüsselung anbietet, hat diese bereits ab Werk aktiviert. Das bedeutet, dass ab dem ersten Schreiben von Daten auf die SSD diese Daten im Flashspeicher verschlüsselt werden. Hintergrund dafür ist, dass bei SSDs beim nachträglichen Verschlüsseln jede Page noch einmal geschrieben werden müsste, mit allen Konsequenzen, die in den Abschnitten über die Controllerfunktionen beschrieben sind. Der Schlüssel, mit dem ab Werk verschlüsselt wird, ist ein für jede SSD einmaliger Schlüssel. Er wird *Data Encryption Key (DEK)* oder auch *Media Encryption Key (MEK)* genannt und bei der Herstellung der SSD in der Controller-Logik erzeugt. Beim Einrichten der Verschlüsselung durch den User wird dann nur der DEK (bzw. MEK)

[20] Vgl. Tom Coughlin: Self-Encrypting Drive Market and Technology Report, S.4 http://www.tomcoughlin.com/Techpapers/Self%20Encrypted%20Storage%20Device%20Market%20and%20Technology%20Report%20Brochure,%20041215.pdf [2016-01-15]

[21] Vgl. ebenda S. 5

[22] Vgl. Micheloni, R. / Marelli, A., a.a.O., S. 349 / 354

mit dem *Key Encryption Key (KEK)* verschlüsselt. Ab diesem Zeitpunkt muss bei jedem Einschalten des PCs und der Festplatte erst der KEK eingegeben werden, mit dem (bei Richtigkeit) die SSD entsperrt und der DEK entschlüsselt wird. Danach kann der Controller Daten beim Schreiben und Lesen transparent ent- bzw. verschlüsseln. Wurde ein falscher Key eingegeben, bleibt die SSD gesperrt und kann keine Daten entgegennehmen und liefern[23]. Die Art und Weise, wie dieser KEK eingerichtet, abgefragt und geändert werden kann, ist abhängig vom PC, Server oder Laptop, in dem die SSD eingebaut ist. Oft sind es Features des BIOS, die diese Kommunikation übernehmen.

Diese zweistufige Verschlüsselung, die auch in anderen Bereichen verwendet wird, hat bei SSDs folgende Vorteile:

- die Verschlüsselung kann jederzeit ohne viel Aufwand eingerichtet werden
- der Schlüssel KEK kann nach Belieben geändert werden
- zum Löschen der SSD muss nur der KEK gelöscht werden
- zu keiner Zeit müssen Pages des Flashspeichers für Umschlüsselungsaktivitäten neu geschrieben werden

Für forensische Untersuchungen an SSDs stellt eine *ausgeschaltete* SSD mit *eingerichteter* Verschlüsselung wahrscheinlich die größte Hürde dar, gilt doch die Verschlüsselung selbst (zumindest wenn es sich um den Standard AES-256 handelt) als ungebrochen[24]. Ohne den KEK wird kein Zugriff auf Daten möglich sein. Befindet sich das Rechnersystem, in dem die SSD eingebaut ist, im Standby oder Ruhemodus, kann unter Umständen ein Zugang zu den Daten der SSD erreicht werden.[25]

Wird eine SED, die forensisch untersucht werden soll, im eingeschalteten und entsperrten Zustand angetroffen, können die Untersuchungen *an dem System, an dem sie angeschlossen ist*, bei vollem Zugriff durchgeführt werden, das Abziehen des Datenkabels und Neuverbinden an einem anderen System ist unter Umständen möglich[26].

Bevor bei einer SSD, die offensichtlich keine vom User eingerichtete Verschlüsselung hat, die Flash-Chips zum direkten Auslesen von dem PCB ausgelötet werden, sollte zunächst geklärt werden, ob sie eine *self-encrypting disk* ist. Ist das der Fall, ist die Verschlüsselung von Anfang an bereits aktiv und ein direktes Auslesen der Daten aus den Flash-Chips liefert keine verwertbaren Daten.

[23] Vgl. Müller, T. / Freiling, F. (2015): A Systematic Assessment of the Security of Full Disk Encryption.
IEEE Trans. Dependable Sec. Comput. 12(5): 491-503 (2015): Abs. 2.2.2
https://www.computer.org/csdl/trans/tq/preprint/06951337.pdf [2016-03-29]

[24] Vgl. Micheloni, R. / Marelli, A., a.a.O., S. 349 / 353

[25] Vgl. Müller T., a.a.O.

[26] Vgl. ebenda.

3.5 Erstellen einer forensischen Kopie einer SSD

Eine forensische Kopie eines Datenträgers ist ein bitweise erstelltes 1:1 Image des gesamten physischen Inhalts[27]. Das Erstellen einer solchen Kopie gehört zu den wichtigsten Tätigkeiten in einer forensischen Untersuchung. Um die Richtigkeit der Kopie sicherstellen und nachweisen zu können, wird der zu kopierende Datenträger über einen sogenannten Writeblocker an den Kopierrechner angeschlossen, mit dem gewährleistet wird, dass keine Write-Commands an den Datenträger gelangen. Zusätzlich werden vor und nach dem Kopiervorgang vom zu kopierenden Datenträger Hash-Werte erstellt, die untereinander und mit dem Hashwert der erstellten Kopie übereinstimmen müssen.

Diese Vorgehensweise ist mit HDD seit vielen Jahren mit relativ wenig Aufwand umsetzbar. SSDs dagegen verhalten sich in vielerlei Hinsicht anders als HDDs. „The SSD technology which is replacing magnetic hard drives inside many computers is neither simple, well understood, nor homogenous; rather it is complex, poorly documented, and highly heterogeneous. Worst of all, it is active - that is to say, the SSD may act under its own initiative, and may undertake quite remarkable (and highly evidence-destructive) actions even in the absence of write commands from a computer, potentially regardless of efforts by police and forensic analysts to prevent invalidation of evidence.“[28]

Dieses Zitat fasst in wenigen Sätzen vieles von dem zusammen, was in den Abschnitten des Kapitels 2 und 3 erläutert wurde. Im Bezug auf das Thema dieses Abschnitts ist wohl der wichtigste Satz: „Worst of all, it is active ....“ Bei den oben genannten Anforderungen an eine forensische Kopie sollte das zu untersuchende Objekt alles andere als „aktiv“ sein. Eine HDD erfüllt diese Bedingungen, sie verändert sich nicht[29], vor allem, wenn mit Hilfe eines Writeblockers jegliche Schreibzugriffe auch elektronisch verhindert werden . In einem SSD-Controller jedoch sind ständig mehrere Prozesse am Laufen, die durchaus auch Daten bewegen können, z. B. die Garbage-Collection und das Wear-Levelling. Wann diese Prozesse ablaufen, ist von aussen nicht zu beeinflussen.

Bei der Frage, ob diese Aktivitäten den zu kopierenden Datenträger während dem Erstellen einer forensischen Kopie unzulässig verändern können, muss wie in den Abschnitten 3.1 (Auswirkungen der SSD-Controllerfunktionen auf Datenwiederherstellung (ohne TRIM)) und 3.2 (Funktionsweise und Auswirkungen der *TRIM*-Funktion) zwischen SSDs mit aktiver und inaktiver TRIM-Funktion unterschieden werden.

Ist TRIM aktiv, teilt das Betriebssystem dem SSD-Controller über ein entsprechendes ATA-Command[30] beim Löschen von Dateien die logischen Blockadressen mit, die aus

[27] Vgl. Geschonneck, A., a.a.O., S. 87f
[28] Bell, G., a.a.O., Abs. 2
[29] Abgesehen vom Ersetzen eines Sektors im Rahmen des Bad-Block Managements
[30] Vgl. ATA/ATAPI Command Set 2, a.a.O., Abs. 7.10.3.2 TRIM

Sicht des Betriebssystems auch physikalisch gelöscht werden können. Auf den Zeitpunkt, an dem die entsprechenden Pages des Flashspeichers dann tatsächlich gelöscht werden, hat das Betriebssystem keinen Einfluss, das erledigt der SSD-Controller im Rahmen der Garbage-Collection.Wird eine SSD sichergestellt, bei der unmittelbar vor dem Ausschalten Dateien gelöscht oder ein Format-Befehl ausgeführt wurden, wird die Garbage-Collection zu einem nicht bestimmbaren Zeitpunkt nach dem nächsten Einschalten diese vom Betriebssystem mitgeteilten logischen Blockadressen löschen, unabhängig davon, ob die SSD an einen Host angeschlossen ist bzw. unabhängig davon, ob zwischen Host und SSD ein Writeblocker geschaltet ist. Somit kann nicht ausgeschlossen werden, dass sich der Inhalt der SSD *vor*, *während* oder auch *nach* einem Erstellen einer Kopie ändert. Ist die Garbage-Collection während oder nach dem Erstellen einer Kopie tätig, kann durch Erzeugen und Vergleichen von Hashwerten des Originals und der Kopie zumindest festgestellt werden, dass eine Änderung stattgefunden hat. Führt der SSD-Controller Garbage-Collection vor den Erstellen einer Kopie aus (z.B. wenn die SSD lange genug im IDLE-Zustand betrieben wurde), ist eine Änderung nicht feststellbar. Wird vermutet, dass TRIM bei einer zu untersuchenden SSD aktiv war, sollte diese SSD erst unmittelbar vor dem Starten des Kopiervorgangs eingeschaltet werden, um zumindest das aktive Ändern der SSD feststellen zu können.

Ist TRIM nicht aktiv, sind die beschriebenen Controller-Funktionen genauso unbeeinflussbar tätig, was den Zeitpunkt angeht. Wie in Abschnitt 3.1 gezeigt, werden dabei aber keine Speicherbereiche gelöscht, sondern ausschließlich „umgeschichtet“. Die Erstellung einer forensischen Kopie ist also möglich.

4 „Sicheres Löschen“ von SSDs

Mit steigendem Bewusstsein für einen wirksamen Schutz von persönlichen, vertraulichen und geheimen Daten vor unberechtigtem Zugriff, Gebrauch, Missbrauch und Veränderung, steigt auch die Notwendigkeit, solche Daten *sicher und vollständig* von nicht mehr gebrauchten und verwendeten Datenträgern löschen zu können. Dies betrifft Privatpersonen mit ihren persönlichen Daten auf Heim-PCs, vor allem aber IT- und Datensicherheitsverantwortliche in Unternehmen, Behörden und öffentlichen Einrichtungen (z. B. des Gesundheitswesens, von Schulen etc.). Für HDDs und SSDs, die defekt bzw. auszutauschen sind oder mitsamt dem Rechnersystem ausser Betrieb genommen werden, gibt es deshalb entsprechende Verfahrensvorschriften[1], die gewährleisten sollen, dass auf den so behandelten Datenträgern beim Weiterverwenden oder nach der Entsorgung keinerlei Daten mehr wiederhergestellt werden können.

Diese Verfahren beinhalten z. B.:

- physikalische Vernichtung
- mehrfaches Überschreiben von einzelnen Dateien
- mehrfaches Überschreiben des gesamten sichtbaren physikalischen Datenbereichs
- Verwenden von Interface-spezifischen Befehlen zum Löschen des gesamten Datenbereichs
- Degaussing

Manche für HDDs entwickelten Verfahren sind für SSDs nicht oder nicht vollständig geeignet[2]. Z. B. hat die bei HDDs hochwirksame Methode des Degaussings bei SSDs keinerlei Auswirkung. Laut der Veröffentlichung von Wei et al. wurden verschiedene Flash-Chips von SSDs in einen für die NSA zertifizierten Degausser verschiedenen Feldern ausgesetzt, die Daten in den Flash-Chips blieben vollständig erhalten[3].

[1] Vgl. NIST: Guidelines for Media Sanitization
http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-88r1.pdf [2015-12-28]

[2] Vgl. Wei, M. / Grupp, L. / Spada, F. / Swanson, S. (2011): Reliably Erasing Data from Flash-Based Solid State Drives.
9th USENIX conference on File And Storage Technologies (FAST'11), (pp. 8-20).
http://cseweb.ucsd.edu/~m3wei/assets/pdf/LISA2011-sanitize.pdf [2015-07-15]

[3] Vgl. ebenda Abs. 3.2.3

Wei et al. haben bei verschiedenen SSDs folgende Methoden zur sicheren Löschung getestet:

- mehrfaches Überschreiben des gesamten sichtbaren physikalischen Datenbereich
- Verwenden von Interface-spezifischen Befehlen zum Löschen des gesamten Datenbereichs
- Verschlüsselung
- mehrfaches Überschreiben von einzelnen Dateien

Verifiziert wurden die Ergebnisse durch direktes Auslesen der Flashspeicher.

Das mehrfache Überschreiben des gesamten sichtbaren physikalischen Datenbereichs führt in vielen Fällen zu einer vollständigen Löschung der SSD, oft sind 2 Durchgänge ausreichend, teilweise waren aber deutlich mehr Durchgänge notwendig. Diese Methode gilt deswegen als unsicher und nimmt ausserdem sehr viel Zeit in Anspruch. Wurde eine SSD mit Hilfe des ATA-Befehls SECURITY ERASE UNIT[4] gelöscht, führte auch dies in mehreren Fällen zu einer zuverlässig gelöschten SSD, der Befehl war aber in manchen SSDs offensichtlich nicht richtig implementiert.

Wie auch in Abschnitt 3.4 *Integrierte Verschlüsselung bei SSD* bereits erwähnt, ist es bei verschlüsselten SSDs ausreichend, den *Key Encrypting Key* zu löschen, um eine sichere Löschung der SSD zu gewährleisten. Wei et al. weisen allerdings darauf hin, dass dies wiederum allein von der richtigen Implementierung der Schlüssellöschung innerhalb der SSD abhängt und das Ergebnis im Prinzip nicht zu überprüfen ist.

Sollen nur einzelne Dateien sicher gelöscht werden, sind die Techniken, die hier für HDDs verwendet werden (das mehrfache Überschreiben von einzelnen Dateien), vollkommen wirkungslos. Durch die für SSDs beschriebenen Verfahren, wie Dateien neu geschrieben werden, ist dies nicht überraschend.

Degaussing ist für HDDs ausreichend und effizient für eine sichere Löschung, allerdings ist die HDD danach nicht mehr weiterverwendbar, was aus ökonomischen Gesichtspunkten nicht immer sinnvoll erscheint. Für SSDs gibt es auch eine Möglichkeit der physikalischen Vernichtung, dabei muss beachtet werden, ab welcher Partikelgröße eine Wiederherstellung auch von Teilen von Pages nicht mehr möglich ist[5].

Leider konnten keine Informationen gefunden werden, inwieweit die Bereiche, die für Over-Provisioning verwendet werden und vom Hersteller reserviert sind (siehe Abschnitt 3.3.1), von den hier vorgestellten Löschtechniken erfasst werden.

[4] Vgl. ATA/ATAPI Command Set 2, a.a.O., Abs. 7.46: SECURITY ERASE UNIT

[5] Vgl. Swanson, S. (2011): Destroying Flash Memory-Based Storage Devices. University of California, CSE; Engineering technical report cs2011-0968. http://cseweb.ucsd.edu/~swanson/papers/TR-cs2011-0968-Grind.pdf [2015-10-13]

5 Fazit

Auch wenn Festplatten sicher noch eine Weile den Datenträgermarkt dominieren werden, so steigt der Anteil von SSDs am Gesamtvolumen doch stetig an[1]. Durch die Kompatibilität der Interfaces und Formfaktoren können nicht nur neue Systeme mit SSDs ausgestattet werden, sondern auch in bestehenden Desktops und Laptops SSDs HDDs auf einfache Art und Weise ersetzen. Neue und kleinere Bauformen führen zudem dazu, dass gerade neue (Ultra-) Notebooks nicht mehr auf die bisherigen Formfaktoren von z. B. 2.5“ Festplatten Rücksicht nehmen müssen. Eine Annäherung der SSDs in Preis und Kapazität an HDDs wird die Kunden freuen, da sie schnellere, kleinere, leisere und robustere Datenträger bekommen, die auch noch weniger Leistung benötigen. Man kann davon ausgehen, dass SSDs dadurch mehr Marktanteile gewinnen werden. Die weitergehende Verbreitung von SSDs wird also sowohl die Organisationen, die Datenträger forensisch untersuchen, als auch die Firmen, die professionelle Dienste in der Datenwiederherstellung anbieten, auch in Zukunft mit neuen Herausforderungen beschäftigen.

Die Unterschiede zwischen SSDs und HDDs sind größtenteils auf zwei (negative) Eigenschaften der Flashspeicher zurückzuführen. Die eine ist die begrenzte Lebensdauer der Flashzellen: nach einer endlichen Anzahl von Lösch- und Schreibvorgängen verliert die Zelle ihre Zuverlässigkeit und kann nicht mehr für Speichervorgänge verwendet werden. Die andere ist die Einschränkung, dass Flashzellen nicht beliebig geschrieben werden können, sondern der Schreibvorgang nur in einer Richtung möglich ist. Danach muss die Zelle gelöscht werden. Das Löschen kann aber nur in größeren Einheiten durchgeführt werden und dauert deutlich länger als das Schreiben.

Um Flashspeicher trotz dieser Einschränkungen in die Lage zu versetzen, mit HDDs konkurrieren zu können und im Prinzip dem Host gegenüber eine HDD zu emulieren, sind Controller mit ausgeklügelten Funktionen entwickelt worden, die diese Schwächen nach aussen kaschieren. Leider sind diese Controller schlecht dokumentiert und unterliegen im Detail in der Regel dem Firmengeheimnis. Sie sind von ausserhalb der SSD nicht beeinflussbar und bilden eine eigene, aktive Einheit. Wie wir gesehen haben, ver-

[1] Vgl. Heise: News | Server & Storage: http://www.heise.de/ct/ausgabe/2016-1-News-Server-Storage-3047977.html [2016-01-25]

hält sich solch eine SSD unter forensischen Aspekten weitgehend wie eine HDD, einmal geschriebene Daten werden durch die SSD nicht gelöscht.

Um den Nachteil, dass in Flashspeicher Zellen nicht überschrieben, sondern erst gelöscht werden müssen, weiter abzuschwächen, wurde 2009 die TRIM-Funktion eingeführt. Mit Hilfe dieser Funktion kann der Host dem SSD-Controller diejenigen logischen Blöcke mitteilen, die aus seiner Sicht, z. B. nach dem Löschen einer Datei oder dem Formatieren einer Partition, nicht mehr benötigt werden und vollständig gelöscht werden können, also auch physikalisch in den Flashzellen. Ist diese Funktion aktiv und richtig implementiert, werden viele Spuren, die sonst auf den Datenträgern zu finden sind, unwiederbringlich gelöscht. Eine SSD wird dadurch zu einer aktiven Einheit, die sehr wohl auch ohne den Host Daten im Flashspeicher löschen kann.

TRIM ist nur in bestimmten Betriebs- und Filesystemen implementiert, zusätzlich müssen Hostcontroller und SSD selbst die Funktion auch unterstützen. Oft sind nicht alle Bedingungen erfüllt oder die TRIM-Funktion ist im SSD-Controller nicht richtig implementiert. Dies bedeutet, dass in einer SSD nicht zwingend keine Spuren zu finden sind, sondern es sich lohnen kann, trotz offensichtlich aktiviertem TRIM nach auswertbaren Spuren zu suchen.

Die bei immer mehr SSDs angebotene Funktion der Selbst-Verschlüsselung stellt eine echte Hürde für forensische Untersuchungen dar. Ist eine SED ausgeschaltet und die Verschlüsselung aktiviert, ist ohne die Kenntnis des *Key Encrypting Key* kein Zugriff auf die Daten möglich. Zu beachten ist ausserdem, dass bei SEDs die Daten im Flashspeicher von Anfang an verschlüsselt werden, egal ob die Verschlüsselung für den User letztendlich eingerichtet wurde oder nicht. Dies hat zur Konsequenz, dass durch ein direktes Auslesen der Flashspeicher-Chips ausschließlich verschlüsselte Daten gewonnen werden können. Dies gilt auch für solche Datenbereiche, die ausserhalb des Bereichs liegen, die vom Host aus sichtbar sind, z.B. Bereiche des Over-Provisioning.

Auf der anderen Seite machen es diese Eigenschaften aber auch nicht trivial, SSDs sicher und komplett zu löschen. Deshalb gilt auch hier, dass bei verschiedenen SSDs das (vermeintlich) sichere Löschen unter Umständen nicht vollständig durchgeführt wurde und auf den SSDs noch verwertbare Spuren zu finden sein können.

6 Ausblick

Wie wir gesehen haben, hat die TRIM-Funktion schon heute den größten negativen Einfluss auf die Datenträgerforensik. Es muss damit gerechnet werden, dass sowohl für weitere Betriebssysteme und Interfaces als auch für RAID-Systeme die Möglichkeit geschaffen wird, dem SSD-Controller mitzuteilen, welche physikalischen Bereiche er löschen kann, ob innerhalb einer Ausweitung des TRIM-Befehls oder mit einer anderen Funktionalität. Zudem wird wohl auch die heute teilweise noch fehlerhafte Implementierung der TRIM-Funktion eher verbessert werden, so dass es in Zukunft mehr SSD-Datenträger geben wird, die weniger Spuren für forensische Untersuchungen bieten.

Auch beim Versuch, eine forensische Kopie einer sichergestellten SSD zu erzeugen, wird dies unter Umständen von der TRIM-Funktion verhindert, da nicht ausgeschlossen werden kann, dass der SSD-Controller beim Kopieren noch Aktivitäten im Rahmen einer Garbage-Collection durchführt, die durch TRIM zumindest ausgelöst wurden. Hier wäre hilfreich, wenn die Datenträgerindustrie bzw. ihre Verbände eine industrieweite Möglichkeit schaffen würden, dass solche Aktivitäten im Falle einer forensischen Untersuchung abgeschaltet werden könnten.

Was kommt nach der SSD? Auch die SSD wird irgendwann durch eine neuere und bessere Technik Konkurrenz erhalten, die vor allem ihre beschriebenen Schwächen eliminiert. Es gibt Forschungen, die die Vorteile von heutigem SRAM und NAND-Speicher zu neuen Technologien verschmelzen, so dass Memory und Storage in einem PC zusammenwachsen, was nicht nur neue forensische Fragen aufwerfen wird. Die Stichworte dieser neuen Technologien sind ReRAM (RRAM), FRAM, MRAM und PCRAM.

Quellen

American National Standard, T13: ATA/ATAPI Command Set 2:
http://www.t13.org/documents/uploadeddocuments/docs2009/d2015r2-ataatapi_command_set_-_2_acs-2.pdf [2015-12-29]

Bell, G. / Boddington, R. (2010): Solid State Drives: The Beginning of the End for Current Practice in Digital Forensic Recovery?
Journal of Digital Forensics, Security and Law, 5 (3), (pp. 1-20).
http://ojs.jdfsl.org/index.php/jdfsl/article/viewFile/21/45 [2015-12-18]

Bonetti, G. / Viglione, M. / Frossi, A / Maggi, F. / Zanero, S. (2013): A Comprehensive Black-box Methodology for Testing the Forensic Characteristics of Solid-state Drives.
29th Annual Computer Security Applications Conference (ACSAC '13), (pp. 269-278)
http://www.syssec-project.eu/m/page-media/3/ssdforensics-ACSAC-final.pdf
[2016-01-02]

Chen, Ben M. / Lee, Tong H. / Peng, Kemao / Venkataramanan,Venkatakrishnan (2006): Hard Disk Drive Servo System (2nd. Edition), London: Springer

Coughlin, T.: Self-Encrypting Drive Market and Technology Report
http://www.tomcoughlin.com/Techpapers/Self%20Encrypted%20Storage%20Device%20Market%20and%20Technology%20Report%20Brochure,%20041215.pdf [2016-01-10]

embedded world 2015: Heidrich, S., Hyperstone 2015: New flash management architecture enables MLC for industrial storage.
http://files.iccmedia.com/magazines/basfeb15/basfeb15-p25.pdf [2015-12-01]

Fujitsu Corporation: White Paper: Fujitsu PRIMERGY Servers Solid State Drives – FAQ (2014)
http://docs.ts.fujitsu.com/dl.aspx?id=78858d6c-4c0f-479a-8ceb-705fe1938f4e
[2015-06-06]

Geschonneck, A. (2008): Computer Forensik, 3. Aufl., Heidelberg: dpunkt verlag

Gubanov, Y. / Afonin, O.: SSD Forensics 2014 Recovering Evidence from SSD Drives: Understanding TRIM, Garbage Collection and Exclusions
http://belkasoft.com/download/info/SSD%20Forensics%202014.pdf [2016-01-02]

Heise: News | Server & Storage
http://www.heise.de/ct/ausgabe/2016-1-News-Server-Storage-3047977.html
[2016-01-25]

IBM Knowledge Center: Considerations for solid-state drives (SSDs)
http://www-01.ibm.com/support/knowledgecenter/HW4M4/p8ebj/arebjsolidstatedrives.htm
[2015-12-05]

International Disk Drive Equipment and Materials Association (IDEMA): Advanced Format (AF) Technology
http://www.idema.org/?page_id=98 [2015-12-15]

Micheloni, R. / Crippa, L. / Marelli, A.: Inside NAND Flash Memories.
Dordrecht: Springer, 2010.

Micheloni, R. / Marelli, A. / Eshghi. K.: Inside Solid State Drives (SSDs).
Dordrecht: Springer, 2013.

Microsoft Corporation: MasterFileTable
https://msdn.microsoft.com/de-de/library/windows/desktop/aa365230(v=vs.85).aspx
[2016-01-02]

Microsoft TechNet: The Four Stages of NTFS File Growth
http://blogs.technet.com/b/askcore/archive/2009/10/16/the-four-stages-of-ntfs-file-growth.aspx
[2015-12-15]

Müller, T. / Freiling, F. (2015): A Systematic Assessment of the Security of Full Disk Encryption.
IEEE Trans. Dependable Sec. Comput. 12(5): 491-503 (2015)
https://www.computer.org/csdl/trans/tq/preprint/06951337.pdf [2016-03-29]

Nisbet, A. / Lawrence, S. / Ruff, M. (2013): A Forensic Analysis And Comparison Of Solid State Drive Data Retention With Trim Enabled File Systems.
11th Australian Digital Forensics Conference (2013), (pp. 103-111).
http://ro.ecu.edu.au/cgi/viewcontent.cgi?article=1124&context=adf [2016-01-01]

NIST: Guidelines for Media Sanitization
http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-88r1.pdf
[2015-12-28]

Samsung Electronics Co.: Samsung SSD 850 PRO Datasheet
http://www.samsung.com/global/business/semiconductor/minisite/SSD/downloads/document/Samsung_SSD_850_PRO_Data_Sheet_rev_2_0.pdf [2015-12-01]

Samsung Electronics Co.: TRIM for Data Centers
http://www.samsung.com/global/business/semiconductor/minisite/SSD/downloads/document/Samsung_SSD_845DC_09_TRIM.pdf [2015-12-13]

Samsung Electronics Co.: V-NAND technology
http://www.samsung.com/us/business/oem-solutions/pdfs/V-NAND_technology_WP.pdf [2015-12-01]

SAS Technology Roadmap - August 2015
http://www.scsita.org/library/2015/10/serial-attached-scsi-technology-roadmap.html [2016-01-01]

Serial ATA International Organization (SATA-IO): SATA Product Features
https://sata-io.org/sites/default/files/images/SATA-IO-English-Brochure-May-2009.pdf [2016-01-01]

Spansion Inc: What Types of ECC Should Be Used on Flash Memory?
https://www.spansion.com/Support/Application%20Notes/Types_of_ECC_Used_on_Flash_AN.pdf [2015-12-16]

Swanson, S. (2011): Destroying Flash Memory-Based Storage Devices:
University of California, CSE; Engineering technical report cs2011-0968.
http://cseweb.ucsd.edu/~swanson/papers/TR-cs2011-0968-Grind.pdf [2015-10-13]

Wei, M. / Grupp, L. / Spada, F. / Swanson, S. (2011): Reliably Erasing Data from Flash-Based Solid State Drives:
9th USENIX conference on File And Storage Technologies (FAST'11), (pp. 8-20).
http://cseweb.ucsd.edu/~m3wei/assets/pdf/LISA2011-sanitize.pdf [2015-07-15]

Western Digital Corporation: WD Re:
http://www.wdc.com/wdproducts/library/SpecSheet/ENG/2879-800066.pdf [2015-10-25]