Fachhochschule Hannover
Fakultät IV - Wirtschaft und Informatik
Studiengang Angewandte Informatik

MASTERARBEIT

Open-Source-Lösungen zur Wahrung der Datensicherheit und glaubhaften Abstreitbarkeit in typischen Notebook-Szenarien

Jussi Salzwedel

September 2009

Beteiligte

Erstprüfer

Prof. Dr. rer. nat. Josef von Helden
Fachhochschule Hannover
Ricklinger Stadtweg 120
30459 Hannover
E-Mail: josef.vonhelden@fh-hannover.de
Telefon: 05 11 / 92 96 - 15 00

Zweitprüfer

Prof. Dr. rer. nat. Carsten Kleiner
Fachhochschule Hannover
Ricklinger Stadtweg 120
30459 Hannover
E-Mail: carsten.kleiner@fh-hannover.de
Telefon: 05 11 / 92 96 - 18 35

Autor

Jussi Salzwedel
E-Mail: j-salzwedel@gmx.de

Erklärung

Hiermit erkläre ich, dass ich die eingereichte Masterarbeit selbständig und ohne fremde Hilfe verfasst, andere als die von mir angegebenen Quellen und Hilfsmittel nicht benutzt und die den benutzten Werken wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Ort, Datum Unterschrift

Inhaltsverzeichnis

1 Einführung **14**
- 1.1 Einleitung . . . 14
- 1.2 Motivation . . . 15
- 1.3 Typografische Konventionen . . . 18
- 1.4 Abgrenzung . . . 18

2 Anwendungsfälle **20**
- 2.1 A1: Notebook kommt abhanden . . . 20
- 2.2 A2: Notebook kommt abhanden und wird nach Verlust gefunden . . . 21
- 2.3 A3: Benutzen des Notebooks, ohne sensitive Daten preiszugeben . . . 22

3 Sicherheitsanforderungen **23**
- 3.1 S0: Open-Source . . . 24
- 3.2 S1: Gewährleistung der Vertraulichkeit . . . 24
- 3.3 S2: Überprüfbarkeit der Integrität . . . 24
- 3.4 S3: Glaubhafte Abstreitbarkeit . . . 25

4 Theoretische Grundlagen **26**
- 4.1 Verschlüsselungsverfahren . . . 26
  - 4.1.1 Advanced Encryption Standard . . . 27
  - 4.1.2 Blowfish . . . 28
  - 4.1.3 Camellia . . . 28
  - 4.1.4 CAST5 . . . 28
  - 4.1.5 CAST6 . . . 29
  - 4.1.6 Data Encryption Standard . . . 29
  - 4.1.7 Triple Data Encryption Standard . . . 29
  - 4.1.8 MARS . . . 29
  - 4.1.9 Rivest Cipher 6 . . . 30
  - 4.1.10 Twofish . . . 30
- 4.2 Betriebsmodi . . . 30
  - 4.2.1 Electronic Code Book . . . 31
  - 4.2.2 Cipher Block Chaining . . . 31

4.2.3 Cipher Feedback . . . 32
4.2.4 Output Feedback . . . 32
4.2.5 Counter . . . 32
4.2.6 CBC-Mask-CBC und ECB-Mix-ECB . . . 33
4.2.7 Counter with CBC-MAC . . . 33
4.2.8 Galois Counter Mode . . . 34
4.2.9 LRW . . . 34
4.2.10 Xor-Encrypt-Xor . . . 34
4.2.11 XEX-based Tweaked CodeBook with Cipher Text Stealing . . . 34
4.3 Initialisierungsvektoren . . . 35
4.3.1 Encrypted Salt-Sector Initialization Vector . . . 35
4.4 Hash-Funktionen . . . 35
4.4.1 Message-Digest Algorithm 2 . . . 37
4.4.2 Message-Digest Algorithm 4 . . . 37
4.4.3 Message-Digest Algorithm 5 . . . 37
4.4.4 RACE Integrity Primitives Evaluation Message Digest 128 . . . 38
4.4.5 RACE Integrity Primitives Evaluation Message Digest 160 . . . 38
4.4.6 Secure Hash Algorithm 1 . . . 39
4.4.7 Secure Hash Algorithm 224/256/384/512 . . . 39
4.4.8 Tiger . . . 40
4.5 Schlüsselgenerierung . . . 40
4.5.1 Password-Based Key Derivation Function 2 . . . 40
4.5.2 TKS1 . . . 40
4.6 Glaubhafte Abstreitbarkeit . . . 41
4.7 Trusted Computing . . . 43
4.7.1 Trusted Computing Platform Alliance und Trusted Computing Group . . . 43
4.7.2 Trusted Computing Group-Architektur . . . 43
4.7.3 Trusted Building Blocks . . . 44
4.7.4 Core Root of Trust for Measurement . . . 44
4.7.5 Dynamic Root of Trust for Measurement . . . 45
4.7.6 Trusted Platform Module . . . 45
4.7.7 Transitives Vertrauen . . . 48
4.7.8 Trusted Boot . . . 48
4.7.9 Secure Boot . . . 50
4.7.10 Trusted Software Stack . . . 50

5 Marktüberblick **52**
5.1 Open-Source Produkte . . . 52
5.1.1 Linux-spezifische Lösungen . . . 52
5.1.1.1 Cryptoloop . . . 53
5.1.1.2 loop-AES . . . 53

5.1.1.3 dm-crypt . . . 53
5.1.1.4 LUKS . . . 53
5.1.2 BSD-spezifische Lösungen . . . 54
5.1.2.1 GBDE . . . 54
5.1.2.2 GELI . . . 55
5.1.2.3 CGD . . . 55
5.1.2.4 Vnd . . . 55
5.1.2.5 Softraid CRYPTO . . . 55
5.1.2.6 OpenSolaris ZFS . . . 56
5.1.3 Windows-spezifische Lösungen . . . 56
5.1.3.1 FreeOTFE . . . 56
5.2 Kostenfreie Produkte . . . 57
5.2.1 Multiplattform-Lösungen . . . 57
5.2.1.1 TrueCrypt . . . 57
5.3 Kommerzielle Produkte . . . 59
5.3.1 BitLocker . . . 59
5.3.2 FileVault . . . 59
5.3.3 PGP Whole Disc Encryption . . . 60
5.3.4 Check Point Full Disc Encryption . . . 60
5.4 Gesamtübersicht der Produkte sowie verwendeter Algorithmen und Betriebsmodi . . . 60
5.5 Hardware-basierte Lösungen zur Integritätsmessung . . . 61
5.5.1 TrustedGRUB . . . 61
5.5.2 GRUB-IMA . . . 62
5.5.3 OSLO . . . 62

6 Analyse ausgewählter Lösungen 65
6.1 Adressierung der Sicherheitsanforderungen . . . 65
6.1.1 S0: Open-Source . . . 65
6.1.2 S1: Gewährleistung der Vertraulichkeit . . . 66
6.1.2.1 Architekturansatz . . . 66
6.1.2.2 Sicherheit der Algorithmen . . . 66
6.1.2.3 Sicherheit der Betriebsmodi . . . 67
6.1.2.4 Verwendete Schlüssellänge . . . 68
6.1.2.5 Implementierung . . . 69
6.1.2.6 Zusammenfassung . . . 73
6.1.3 S2: Überprüfbarkeit der Integrität . . . 74
6.1.3.1 Architekturansatz . . . 74
6.1.3.2 Implementierung . . . 74
6.1.4 S3: Glaubhafte Abstreitbarkeit . . . 75
6.2 Zusammenfassung . . . 75

7 Konzipierung eines Gesamtsystems **77**
7.1 Adressierung der Sicherheitsanforderungen . . . 77
7.1.1 Gewährleistung der Vertraulichkeit . . . 78
7.1.2 Überprüfung der Integrität . . . 79
7.1.3 Glaubhafte Abstreitbarkeit . . . 80
7.2 Produktzusammenstellung . . . 81
7.3 Verbleibende Lücken . . . 82
7.4 Lösungsansätze . . . 82

8 Exemplarische Realisierung **83**
8.1 Allgemeine Vorgaben . . . 83
8.2 Installation auf USB-Stick . . . 84
8.3 TrueCrypt Installation . . . 84
8.4 TrustedGRUB Installation . . . 87

9 Reflexion **89**
9.1 Erkenntnisse aus der exemplarischen Realisierung . . . 89
9.2 Erfüllung der Sicherheitsanforderungen . . . 90

10 Schlussbemerkung **91**

Quellenverzeichnis **93**

11 Anhang **104**
11.1 GRUB . . . 104

Abbildungsverzeichnis

4.1 Glaubhafte Abstreitbarkeit durch unterschiedliches Schlüsselmaterial . . 42
4.2 Trusted Building Blocks innerhalb einer PC-Plattform. Quelle: [ea07b] . . 44
4.3 Komponenten eines TPM. Quelle: Angelehnt an Abbildung 11.23 aus [Eck07] 47
4.4 Transitives Vertrauen. Quelle: Angelehnt an [ea07b] 49

5.1 Verstecktes Volume innerhalb eines Standard-Volume. Quelle: [Fou09] . . 58

Tabellenverzeichnis

5.1 Produkt x Verschlüsselungsverfahren-Matrix 63
5.2 Produkt x Betriebsmodus-Matrix 64

Abkürzungsverzeichnis

3DES	Triple-DES
AES	Advanced Encryption Standard
BSD	Berkley Software Distribution
CBC	Cipher Block Chaining
CCM	Counter with CBC-MAC
CFB	Cipher Feedback
CGD	CryptoGraphic Disc
CMC	CBC-Mask-CBC
CRT	Counter (Mode)
CRTM	Core Root of Trust for Measurement
CTS	Cipher Text Stealing
DES	Data Encryption Standard
DRTM	Dynamic Root for Trust for Measurement
ECB	Electronic Code Book
EME	ECB-Mix-ECB
ESSIV	Encrypted Salt-Sector Initialization Vector
FDE	Full Disc Encryption
FUSE	Filesystem in Userspace
GBDE	GEOM Based Disk Encryption
GCM	Galois Counter Mode
GNU	GNU's not Unix

GRUB	Grand Unified Bootloader
IMA	Integrity Measurement Architecture
LUKS	Linux Unified Key Setup
MAC	Message Authentication Code
MD2	Message-Digest Algorithm 2
MD4	Message-Digest Algorithm 4
MD5	Message-Digest Algorithm 5
NIST	National Institute of Standards and Technology
OFB	Output Feedback
OSLO	Open Secure Loader
OSS	Open Source Software
PBKDF2	Password-Based Key Derivation Function 2
PKCS	Public Key Cryptography Standards
RACE	Research and Development in Advanced Communications Technologies in Europe
RC6	Rivest Chipher 6
RIPEMD-128	RACE Integrity Primitives Evaluation Message Digest 128
RIPEMD-160	RACE Integrity Primitives Evaluation Message Digest 160
RSA	Asymmetrisches Kryptosystem
RTM	Root of trust for measurement
RTS	Root of trust for storage
RTR	Root of trust for reporting
SHA-1	Secure Hash Algorithm 1
SHA-224	Secure Hash Algorithm 224
SHA-256	Secure Hash Algorithm 256
SHA-384	Secure Hash Algorithm 384
SHA-512	Secure Hash Algorithm 512
TBB	Trusted Building Block

TCB	Tweaked Code Book
TCG	Trusted Computing Group
TCPA	Trusted Computing Platform Alliance
TKS1	An anti-forensic, two level, and iterated key setup scheme
TPM	Trusted Platform Module
TSS	Trusted Software Stack
vnd	vnode disk driver
XEX	Xor-Encrypt-Xor
XTS	XEX-TCB-CTS

Glossar

Entropie bezeichnet im Rahmen dieser Arbeit die scheinbare Menge an Zufall, d.h. nicht die tatsächlich vorhandene, sondern die, die ein uninformierter Beobachter erkennen kann.

Nonce „Used only once“ oder „number used once“ bezeichnet eine (meist zufällige) Zeichenfolge die ad hoc generiert und nur einmal verwendet wird, um vor allem Replay-Attacken vorzubeugen.

Public Domain ist in den USA nach [ea09a] ein rechtlicher Begriff und weist auf den vollständigen Rechtsverzicht des Rechteinhabers hin.

Public Key Cryptography Standards bezeichnet eine Reihe kryptographischer Spezifikationen, die u.a. von den RSA-Laboratorien entwickelt wurden. Weitere Details sind [Sec09] zu entnehmen.

Salt bezeichnet eine zufällige Bitfolge. Sie wird mit einem gegebenen Klartext konkateniert. Das Ergebnis dient als Eingabe für eine Hashfunktion. Hierdurch wird die Entropie der Eingabe erhöht.

Man gebe mir sechs Zeilen, geschrieben von dem redlichsten Menschen, und ich werde darin etwas finden, um ihn aufhängen zu lassen.

Armand-Jean du Plessis, duc de Richelieu (1585-1642)

1 Einführung

1.1 Einleitung

Die zunehmende Verbreitung von Notebooks und anderen mobilen Endgeräten führt dazu, dass private und geschäftliche Daten physisch nicht nur zu Hause respektive beim Arbeitgeber gespeichert, sondern mitgetragen werden. Dies erhöht die Wahrscheinlichkeit, dass Dritte unberechtigten Zugriff auf die sensitiven Daten erhalten können. Hieraus erwachsen für mobile Endgeräte folgende Sicherheitsanforderungen:

1. Die Vertraulichkeit der Daten muss gewährleistet bleiben.
2. Die Integrität der Daten muss gewährleistet und überprüft werden können.
3. Die Existenz von sensitiven Daten muss glaubhaft abgestritten werden können.

In dieser Arbeit werden vor allem Open Source Lösungen zur Sicherstellung der obigen Anforderungen analysiert. Dabei werden marktgängige Produkte sowohl hinsichtlich ihrer generellen Eignung als auch - in ausgewählten Fällen – hinsichtlich ihrer technischen Qualität beurteilt. Basierend auf den untersuchten Open Source Produkten wird ein Gesamtsystem konzipiert, das die o.g. Anforderungen so weit wie möglich erfüllt. Verbleibende Lücken werden aufgezeigt. Lösungsansätze zur Schließung dieser Lücken werden entworfen und exemplarisch realisiert.

1.2 Motivation

Seit dem 11. September 2001 ist der Einsatz von Überwachungstechnologien weltweit stark gestiegen. Diese Entwicklung öffnet die Tür hin zu Überwachungsgesellschaften. Eine Post-9/11-Paranoia hat irrationalen Problemlösungsansätzen sowohl in der Gesellschaft als auch in der

Politik[1] den Weg geebnet. Im Namen der Terrorismusbekämpfung werden mühsam erkämpfte Grund- und Freiheitsrechte fast schon im regelmäßigen Rhythmus abgeschafft. Der immense technologische Wandel bringt einen ebenso drastischen gesellschaftlichen Wandel mit sich und stellt Entscheidungsträger vor neue Herausforderungen. Gerade innerhalb der Politik sind Verantwortliche regelrecht überfordert. Politiker müssen lernen, mit der neuen digitalen Dimension des Lebens umzugehen, denn sie entwerfen die Rahmenbedinungen für eine digitale Welt, die sie kaum kennen. Die ihnen nachfolgende netzaffine Web 2.0 oder auch C64-Generation genannt, wächst hingegen mit einem ganz anderen Selbstverständnis der Digitalisierung auf: Das Notebook ist fast schon eine „Erweiterung“ des Gehirns, das Internet ein Kulturraum. Online-Erreichbarkeit ist genauso selbstverständlich wie die telefonische. Soziale Netze werden ins Internet getragen oder entstehen sogar dort. Digitale Inhalte werden mit einem Mausklick vervielfältigt. Spiele werden zu Tausenden im Internet gespielt. Dieser enorme Wandel bringt neue Herausforderungen und Probleme mit sich, die nach neuen gesellschaftlichen, aber auch technischen Lösungsansätzen verlangen. Ferner muss man sich vergegenwärtigen, dass die Informationstechnik das Potenzial einer Totalüberwachung aller Verhaltensweisen, Kontakte und Kommunikationsvorgänge hat - das gilt für Privatpersonen, wie für die

[1] Als repräsentatives Beispiel sei hier die Einführung des *Patriot Act* in den USA genannt. Peter Schaar in [Sch07] dazu:

> „Bereits einen Monat nach den Terroranschlägen verabschiedeten mit überwältigenden Mehrheiten zunächst der Senat (96 zu 1 Stimmen) und dann das Repräsentantenaus (337 zu 79) den 'Patriot Act', der Geheimdiensten und Polizeibehörden sehr weit gehende neue Befugnisse einräumt. Die beschlossenen Maßnahmen umfassen Vollmachten zum Abhören von Telefonen, zum Mitlesen von E-Mails und zum geheimen Zugriff auf alle möglichen privaten Datenbestände, von durch Telefongesellschaften gespeicherten Telekommunikationsdaten bis hin zu Dateien über das Leseverhalten in öffentlichen Bibliotheken. Niemals zuvor hatte es in den USA derart drastische Einschränkungen von Bürgerrechten ohne ausführliche Diskussion gegeben. Zahlreiche Senatoren und Abgeordnete beklagten noch kurz vor der Abstimmung, sie hätten nicht einmal die Gelegenheit gehabt, die ihnen nur wenige Tage zuvor zugeleiteten, mehrere hundert Seiten umfassenden Gesetzesentwürfe vollständig zu lesen.“

Noch prekärer sind die veröffentlichten Zahlen im Zusammenhang mit dem Patriot Act und in dem Jahr 2008 erfolgten Hausdurchsuchungen: Nach [Gri09] erfolgten 763 Hausdurchsuchungen, von denen lediglich drei im Zusammenhang mit Terrorismus stehen. 62% der Durchsuchungen erfolgten im Zusammenhang von Ermittlungen gegen Rauschgiftkriminalität, obwohl es sich um ein explizites Anti-Terror-Gesetz handelt. Ähnliche Maßnahmen wie die Verabschiedung des Patriot Act wurden hierzulande und weltweit ergriffen. Sie alle aufzuzählen, würde den Rahmen dieser Arbeit sprengen.

Wirtschaft als auch für staatliche Stellen. Maßnahmen zum Schutz vor dieser Bedrohung sind überfällig. Genau hier setzt diese Arbeit in Form eines Beitrags zum Datenselbstschutz an. Sie bietet einen Baustein im Gesamtbild der technischen Lösungen an: Die Anzahl mobiler Endgeräte nimmt stetig zu. Sensitive Daten - sowohl geschäftliche als auch private - werden auf tragbaren Datenträgern mitgenommen und bedürfen daher des Schutzes vor unberechtigtem Zugriff durch Dritte.

Als *Privatsphäre* bezeichnet man den Raum des individuellen Rückzugs. Sie ist unverzichtbare Vorraussetzung einer freien Meinungsbildung und muss daher geschützt bleiben. Dieser Raum bietet die Möglichkeit unbeobachtet und unzensiert über seine Erfahrungen und Einstellungen zu reflektieren und sich mit anderen auszutauschen. Sie bietet die Grundlage für das autonome Individuum und eine freie Öffentlichkeit. Totalitäre Systeme versuchen die öffentliche Sphäre und die Privatsphäre vollständig zu kontrollieren.

Das Recht auf *informationelle Selbstbestimmung* bezeichnet das Recht, selbst über die Preisgabe und Verwendung seiner personenbezogenen Daten zu bestimmen. Sie wurde vom Bundesverfassungsgericht im berühmten Volkszählungsurteil [Bun83] von 1983 als Grundrecht anerkannt. Sie leitet sich aus dem allgemeinen Persönlichkeitsrecht und der Menschenwürde ab. Wikipedia[2] fasst die Kernaussage in [div09a] gelungen zusammen

> „Die freie Selbstbestimmung bei der Entfaltung der Persönlichkeit werde gefährdet durch die Bedingungen der modernen Datenverarbeitung. Wer nicht wisse oder beeinflussen könne, welche Informationen bezüglich seines Verhaltens gespeichert und vorrätig gehalten werden, werde aus Vorsicht sein Verhalten anpassen (siehe auch Panoptismus). Dies beeinträchtige nicht nur die individuelle Handlungsfreiheit sondern auch das Gemeinwohl, da ein freiheitlich demokratisches Gemeinwesen der selbstbestimmten Mitwirkung seiner Bürger bedürfe. Mit dem Recht auf informationelle Selbstbestimmung wären eine Gesellschaftsordnung und eine diese ermöglichende Rechtsordnung nicht vereinbar, in der Bürger nicht mehr wissen können, wer was wann und bei welcher Gelegenheit über sie weiß.“

Wer die Privatsphäre und die informationelle Selbstbestimmung stärken will, erhält zumeist als Gegenargument zu hören: „Ich habe nichts zu verbergen.“ Das dem nicht so ist, zeigen laut [Sch07] die jährlich Tausenden von Beschwerden von Bürgern bei den Datenschutzbehörenden über den fehlerhaften respektive missbräuchlichen Umgang

[2] Anmerkung zum Zitieren aus der Wikipedia: Jeder kann Wikipedia-Artikel verändern oder verfälschen. Daher kann Wikipedia nicht immer als verlässliche Quelle für wissenschaftliche Arbeiten angesehen werden. Hinzu kommt der oft geäußerte und berechtigte Vorwurf des „Digitalen Maoismus“, in der sich das Halbwissen der Masse, statt das Fachwissen der Minderheit durchsetzt. Dennoch möchte ich in diesem Fall von den berechtigten Vorwürfen abweichen und die Wikipedia unverändert zitieren, weil die Kernaussage zum Thema der informationellen Selbstbestimmung meines Erachtens kaum besser zusammengefasst werden kann.

mit persönlichen Daten wie bspw. sensible medizinische Angaben oder Daten, die unter dem besonderen Schutz des Sozialgeheimnisses stehen. Die Beschwerden widerlegen laut [Sch07] ganz praktisch „die These, man habe nichts zu befürchten, weil man ja nichts zu verbergen habe.“ (S. 23)

Georg Orwells Roman „1984“ gilt als literarisches Symbol für die Gefahren des Überwachungsstaats. Nicht die Technologie an sich ist das Problem, so Orwell, sondern die Gefährdung der Demokratie und der Selbstbestimmung durch totalitäre Ideologien und Mächte, die sich der zunehmenden Überwachungstechnik bedienen könnten. Benjamin Franklin wird nachgesagt folgenden Satz im Jahr 1775 geäußert zu haben: „Wer grundlegende Freiheiten aufgibt, um vorübergehend ein wenig mehr Sicherheit zu gewinnen, verdient weder Freiheit noch Sicherheit.“ Daten, die für sich genommen belanglos erscheinen können in anderen Zusammenhängen durchaus bedeutsam werden. Das Bundesverfassungsgericht sagte dazu im Volkszählungsurteil unter C. II. 2.:

> „Die Verfassungsbeschwerden geben keinen Anlaß zur erschöpfenden Erörterung des Rechts auf informationelle Selbstbestimmung. Zu entscheiden ist nur über die Tragweite dieses Rechts für Eingriffe, durch welche der Staat die Angabe personenbezogener Daten vom Bürger verlangt. Dabei kann nicht allein auf die Art der Angaben abgestellt werden. Entscheidend sind ihre Nutzbarkeit und Verwendungsmöglichkeit. Diese hängen einerseits von dem Zweck, dem die Erhebung dient, und andererseits von den der Informationstechnologie eigenen Verarbeitungsmöglichkeiten und Verknüpfungsmöglichkeiten ab. Dadurch kann ein für sich gesehen belangloses Datum einen neuen Stellenwert bekommen; insoweit gibt es unter den Bedingungen der automatischen Datenverarbeitung kein 'belangloses' Datum mehr.“ [TB09]

Damit räumte das Gericht mit der Vorstellung auf, es gebe „freie“ Daten, die nicht schützenswert seien.

Im Jahr 2005 setze sich die Diskussion unter neuen Vorzeichen im Kontext der Online-Durchsuchung fort. Es kam zu einer Verfassungsbeschwerde gegen die Vorschriften im Verfassungsschutzgesetz von Nordrhein-Westfalen zur Online-Durchsuchung. Im Februar 2008 formulierte das Bundesverfassungsgerichts schließlich das *Grundrecht auf Gewährleistung der Vertraulichkeit und Integrität informationstechnischer Systeme* und passte somit das Recht auf informationelle Selbstbestimmung ans 21. Jahrhundert an. Private Computer und somit auch Notebooks enthalten meist besonders sensible Daten, die einen tiefen Einblick in die Persönlichkeit der Betroffenen geben. Der Schutz des absoluten Kernbereichs der Privatsphäre und somit der Daten vor staatlichen aber auch wirtschaftlichen Eingriffen ist somit unverzichtbar.

Neben dem Bedürfnis einer Privatperson nach dem Schutz seiner Privatsphäre existieren ebenso bedeutende Anforderungen seitens der Wirtschaft: Unternehmen haben ein großes Interesse, Firmengeheimnisse zu schützen und Wirtschaftsspionage effektiv vorzubeugen.

Abwerbung durch Headhunter, Mitarbeiterüberwachung oder die Übernahme sensibler staatlicher Aufgaben wie im Bereich des Gesundheitswesen oder des Verteidigungssektors durch die Wirtschaft sind nur einige weitere Beispiele, die die Notwendigkeit „sicherer Notebooks“ für Unternehmen aufzeigen.

Der Datenschutz muss genauso wie alle anderen Sicherheitsaspekte bereits in das Design von IT-System eingebunden werden. Diese sollten nicht als Add-On nachträglich auf ein fertiges System gestülpt werden. Hieraus ergeben sich ganz konkrete Anforderungen an IT-Systeme, die bereits in einer Anforderungsanalysephase berücksichtigt werden müssen.

1.3 Typografische Konventionen

Hervorhebungen erfolgen in *kursiver* Schrift. Datei- und Verzeichnisnamen, Shell-Befehle und Funktionsbezeichungen erfolgen in `dieser TrueType-Schriftart`.

1.4 Abgrenzung

Im Rahmen dieser Arbeit liegt der Fokus auf Software-Lösungen. Hardwareunterstützte Ansätze im Bereich der Full Disc Encryption (FDE), wie sie innerhalb der *TCG Storage Specification* genannt werden, sind nicht Gegenstand der Betrachtung. Zum einen ist der Standard relativ jung, zum anderen sind erst zum Zeitpunkt der Erstellung dieser Arbeit die ersten Festplatten auf dem Markt, so dass eine ausführliche Betrachtung nicht stattfinden kann[3].

Hardwarebasierte-Attacken fallen ebenfalls aus dem Rahmen der Betrachtung, weil sie stärker im Bereich der Elektrotechnik anzusiedeln sind und daher ganz andere Schutzmaßnahmen erfordern.

[3] Kryptographie-Hardware wird in 99% aller Fälle proprietär realisiert. Die verwendeten Algorithmen können offen sein, die Spezifikation des Systems - wie im Falle von TCG Storage Specification - ebenfalls, dennoch bleibt die Hardware geschlossen. Der Trend hin zu mehr offener Hardware wäre an dieser Stelle begrüßenswert. Selbstverständlich gilt dies nicht nur für Kryptographie-Hardware, sondern auch für andere Komponenten, wie bspw. CPU etc. Damit wäre die Überprüfbarkeit der Hardware gegeben. Es existieren vereinzelte Ansätze, die in diese Richtung weisen - selbst namhafte Hersteller, wie SUN haben ihre SPARC-Architektur in Form von OpenSPARC unter GPL offengelegt. Dennoch ist die Entwicklung im Bereich der Hardwareherstellung weit entfernt von dem Wandel von proprietären Lösungen hin zu offenen Lösungen, wie er sich innerhalb der Softwarelandschaft vollzieht. Außerdem stellt sich die Frage: Wie produziert man seine eigene und damit vertrauenswürdige Hardware?

Für Hardware-basierte Angriffe gilt außerdem folgender allgemeiner Grundsatz: *Wenn ein Angreifer physikalischen Zugriff auf das Notebook erhält* ***und*** *der Benutzer das Notebook nach diesem Zeitpunkt nutzt, dann gibt es* ***kein*** *System, welches das Notebook vor Kompromittierung schützt.*

We should treat personal electronic data with the same care and respect as weapons-grade plutonium – it is dangerous, long-lasting and once it has leaked there's no getting it back.

Cory Doctorow (* 1971)

2 Anwendungsfälle

Im Folgenden werden die für diese Arbeit relevanten Anwendungsfälle beschrieben. Sie dienen als Motivation für diese Arbeit und zeigen die Notwendigkeit der zu erarbeitenden Schutzmaßnahmen auf. Die genannten Anwendungsfälle erheben keinen Anspruch auf Vollständigkeit, weisen aber auf die Kernanforderungen hin und finden sich innerhalb der aktuellen Berichterstattung in den einschlägigen Medien wieder. Nachfolgend wird der Begriff *Notebook* synonym für *mobiles Endgerät* verwendet. Es wird bewusst der Begriff *Notebook* verwendet, um die Anwendungsfälle möglichst konkret zu halten. In jedem Anwendungsfall wird davon ausgegangen, dass das Notebook sensitive und schützenswerte Daten enthält.

2.1 A1: Notebook kommt abhanden

In diesem Anwendungsfall kommt das Notebook abhanden. Dies kann bspw. durch Diebstahl oder Unachtsamkeit beim Transport zustande kommen. Es wird davon ausgegangen, dass der Benutzer nicht mehr in den Besitz seines Notebooks kommt. *Die Informationen müssen derart geschützt sein, dass unberechtigte Parteien keinen Zugriff auf diese erhalten können.* Aktuelle und vergangene Beispiele belegen, dass dieser Anwendungsfall regelmäßig eintritt und vor allem personenbezogene Daten in die Hände Dritter gelangen:

- Nach Angaben von *The Irish Times* wurden am 05. Juni 2009 vier Laptops der Fir-

ma Bord Gáis[1] gestohlen [div09b]. Einer der gestohlenen Laptops enthielt u.a. die Bankverbindungsdaten von 75000 Kunden. Die Daten waren nicht verschlüsselt.

- Am 06. Juni 2008 berichtete die Stanford University, dass sie ein Laptop mit personenbezogenen Daten von 60000 aktiven und ehemaligen Mitarbeitern vermisst [Liv08].
- Heise online berichtete am 02. Januar 2008 über den sprunghaften Anstieg von Datenklau-Opfern in den USA und erwähnt explizit als eine Ursache hierfür das Verlieren oder Gestohlen werden von Laptops [Zie08].
- Heise online berichtete am 17. Februar 2007, dass beim FBI 160 Laptops im Zeitraum von 44 Monaten verschwunden sind oder gestohlen wurden. Das Prekäre an dem Vorfall ist, dass das FBI nicht genau weiss, was auf den Laptops gespeichert war. Sie räumen ein, dass es sich bei den Daten um brisante Informationen zu Fällen, Personen, FBI-Aktionen oder Software zur Erstellung von Büro-Ausweisen handeln kann [Ebe07].
- Heise online berichtete am 14. Dezember 2006, dass der Firma Boeing ein Notebook mit personenbezogenen Daten von 382000 aktiven und ehemaligen Mitarbeitern abhanden gekommen ist. Boing konnte nicht sagen, ob die Daten verschlüsselt waren oder nicht. Laut Heise online „gebe es keine unternehmensweite Richtlinie für die Verschlüsselung von Mitarbeiterdaten“ [Wil06].

Private Notebooks kommen genau so oft abhanden wie geschäftliche Notebooks, werden aber nicht im Rahmen der Berichterstattung in den Medien erwähnt und fallen daher oft aus der Betrachtung.

2.2 A2: Notebook kommt abhanden und wird nach Verlust gefunden

In diesem Anwendungsfall kommt das Notebook wie in Anwedungsfall A1 abhanden, mit dem Unterschied, dass der Benutzer das Notebook nach einem Zeitraum Δt zurückerhält. Nun stellt sich die zentrale Frage: *Woher weiss der Benutzer, dass er dem Notebook softwareseitig vertrauen kann?* Innerhalb des Zeitraums Δt könnte ein Dritter das System modifiziert und bspw. ein Trojanisches Pferd[2] oder ein Rootkit[3] installiert haben.

[1] Bord Gáis Éireann, kurz Bord Gáis, ist der größte Gaslieferant der Republik Irland.

[2] Ein Trojanisches Pferd ist ein Computerprogramm, welches vorgibt eine nützliche Anwendung zu sein, aber im Hintergrund ohne Wissen des Anwenders andere Funktionen ausführt.

[3] Ein Rootkit ist ein Computerprogramm oder eine Sammlung von Computerprogrammen, die nach Einbruch auf ein kompromittiertes System installiert wird. Sie dient dazu, den Angriff bzw. weitere ggf. folgende Zugriffe bereits auf der Ebene des Kernels zu verbergen.

2.3 A3: Benutzen des Notebooks, ohne sensitive Daten preiszugeben

In diesem Anwendungsfall geht es um die Fragestellung, *wie ein Benutzer das Notebook benutzen kann, ohne Zugriff auf sensitive Daten zu gewähren* ***und*** *in der Lage ist, die Existenz solcher Daten erfolgreich abzustreiten.* Folgende Beispiele seien hierzu genannt:

- Ein Mitarbeiter einer Menschenrechtsorganisation protokolliert im Rahmen seiner Tätigkeit Menschenrechtsverletzungen innerhalb eines totalitären Regimes. Der Mitarbeiter wird festgenommen und mit nicht-rechtsstaatlichen Mitteln verhört, so dass er u.a. dazu gezwungen wird, sich an seinem Notebook anzumelden.
- Nach [Sch08] und [Fra08] hat der US-Zoll bei der Einreise in die USA das Recht, den Einreisenden zu bitten, sich am Notebook anzumelden. Wird dem nicht nachgekommen, kann die Einreise verweigert werden.
- In Großbritannien ist im Oktober 2007 das *Section Three of the Regulation of Investigatory Powers Act (RIPA)* in Kraft getreten. Es erlaubt der Polizei auf die Herausgabe von Schlüsselmaterial zu verschlüsselten Daten zu drängen. Wer dem nicht nachkommt, kann laut Gesetzestext [otUK00] mit bis zu zwei Jahren Freiheitsentzug bestraft werden. Bereits einen Monat nachdem in Kraft treten des Gesetzes gab es den ersten „kuriosen“ Fall dazu: Tierschützer wurden von den Behörden aufgefordert, entweder ihre Daten zu entschlüsseln und diese den Behörden zu geben oder das Schlüsselmaterial den Behörden bereitzustellen [War07]. Im August 2009 berichtete heise online, dass es im Rahmen von RIPA sogar zu der ersten Passwort-Erzwingungshaft gekommen ist. [Wil09]

Diese und andere Vorfälle haben die Diskussion um *deniable encryption* erneut entfacht, die Notwendigkeit technologischer Maßnahmen aufgezeigt und die Entwicklung von Werkzeugen forciert.

Ohne Sicherheit ist keine Freiheit.

Wilhelm von Humboldt (1767-1835)

3 Sicherheitsanforderungen

In diesem Kapitel werden Anforderungen aus den Anwendungsfällen und daraus implizit verbundene Schutzziele abgeleitet. Die Anforderungen lassen sich in allgemeine Anforderungen einerseits und Sicherheitsanforderungen andererseits unterteilen. Die allgemeinen Anforderungen bilden den Rahmen, die Sicherheitsanforderungen den Kern dieser Arbeit.

Implizit gelten folgende Rahmenbedingungen für die Arbeit:

Notebook als Hardwareplattform Das Thema der Arbeit ist auf Notebooks ausgerichtet, allgemein gesehen auf mobile Endgeräte. Nicht betrachtet werden stationäre PCs oder Server.

X86-Architektur Die Software-Lösungen werden im Umfeld von X86-Architekturen betrachtet. Andere Hardware-Plattformen werden nicht betrachtet.

Als allgemeine Anforderung gilt:

Benutzerfreundlichkeit Die Lösung muss benutzerfreundlich sein, damit sie laufend und einfach genutzt werden kann. Idealerweise gibt sie dem Nutzer eine hohe Nutzungsqualität bei der Interaktion mit dem System.

Die in Kapitel 2 genannten Anwendungsfälle betreffen die Schutzziele *Vertraulichkeit*, *Integrität* und *Abstreitbarkeit* die nachfolgend im Detail untersucht werden. Aus den Anwendungsfällen lassen sich die nachfolgenden Sicherheitsanforderungen ableiten.

3.1 S0: Open-Source

Die Betrachtung liegt auf Open-Source-Lösungen. Diese Anforderung soll sowohl für das Produkt selbst, als auch für die Betriebssystemplattform gelten. Unter Open-Source Produkten werden im Rahmen dieser Arbeit Softwareprodukte verstanden, die der Open Source Definition [Ini] der Open Source Initiative[1] genügen. Das Studieren und Analysieren des Quellcodes ermöglicht es, gerade im Kontext einer sicherheitsrelevanten Betrachtung, die Implementation nach sicherheitskritischen Gesichtspunkten zu überprüfen.

3.2 S1: Gewährleistung der Vertraulichkeit

Gewährleistung der Vertraulichkeit bedeutet, dass lediglich autorisierte Benutzer Zugriff auf gespeicherte Daten erhalten. Der Zugriff auf sensitive Daten durch Dritte ist nicht möglich. Das BSI definiert Vertraulichkeit in [fSidI09] wie folgt:

> „Vertraulichkeit ist der Schutz vor unbefugter Preisgabe von Informationen. Vertrauliche Daten und Informationen dürfen ausschließlich Befugten in der zulässigen Weise zugänglich sein.“ (S. 10)

Dieses Schutzziel korrespondiert mit Anwendungsfall A1 aus Kapitel 2.1.

3.3 S2: Überprüfbarkeit der Integrität

Überprüfbarkeit der Integrität bedeutet, dass Daten und Programme nicht unbemerkt verändert werden dürfen. Sollte es zu einer Änderung von Daten oder Programmen durch Dritte kommen, muss dies für den Anwender ersichtlich sein. Eine Manipulation der Daten und Programme durch Dritte kann nicht unterbunden werden. Somit ist eine Änderung der Daten und Programme durch Dritte möglich. Sollte dieser Fall eintreten, muss der Anwender in der Lage sein, die Integrität seiner Daten und Programme zu überprüfen und somit Änderungen durch Dritte festzustellen. Das BSI definiert Integrität in [fSidI09] wie folgt:

> „Integrität bezeichnet die Sicherstellung der Korrektheit (Unversehrtheit) von Daten und der korrekten Funktionsweise von Systemen. Der Begriff Integrität drückt aus, dass die Daten vollständig und unverändert sind. Der Verlust der Integrität von Informationen bedeutet daher, dass die Daten unerlaubt verändert wurden.“ (S. 10)

[1] Die Open Source Initiative (OSI) ist eine Organisation, die Open-Source-Software fördert. Weitere Details können `http://www.opensource.org/` entnommen werden.

Dieses Schutzziel korrespondiert mit Anwendungsfall A2 aus Kapitel 2.2.

3.4 S3: Glaubhafte Abstreitbarkeit

Glaubhafte Abstreitbarkeit (engl. plausible deniability) heisst im Kontext dieser Arbeit, dass die Existenz von Daten, also das Vorhandensein von Informationen abgestritten werden kann. Es ist das Gegenteil von Nichtabstreitbarkeit (engl. non-repudiation). Dieses Schutzziel korrespondiert mit Anwendungsfall A3 aus Kapitel 2.3.

Es gibt nichts Praktischeres als eine gute Theorie.
Immanuel Kant (1724-1804)

4 Theoretische Grundlagen

4.1 Verschlüsselungsverfahren

Unter Verschlüsselung versteht man die Transformation von Klartext in Geheimtext, unter Entschlüsselung die Umkehrfunktion. Im Bereich der Verschlüsselungsverfahren unterscheidet man zwischen symmetrischen und asymmetrischen Verschlüsselungsverfahren:

Symmetrische Verschlüsselungsverfahren zeichnen sich dadurch aus, dass alle beteiligten Parteien denselben Schlüssel S besitzen und verwenden:
V_S sei Verschlüsselungsfunktion mit Schlüssel S
E_S sei Entschlüsselungsfunktion zu V_S
K sei Klartext
G sei Geheimtext
Dann lässt sich die Verschlüsselung schreiben als:

$$G = V_S(K)$$

Die Entschlüsselung lässt sich schreiben als:

$$K = E_S(G)$$

Asymmetrische Verschlüsselungsverfahren verwenden pro Partei ein Schlüsselpaar, das aus einem öffentlichen und privaten Schlüssel besteht. Die Kernidee ist, dass der öffentliche Schlüssel öffentlich ausgetauscht werden kann, eine verschlüsselte Nachricht aber nur mit dem privaten, also geheimen Schlüssel entschlüsselt werden kann.
$S_{A_{pub}}$ und $S_{A_{priv}}$ seien öffentlicher und privater Schlüssel der Partei A
$S_{B_{pub}}$ und $S_{B_{priv}}$ seien öffentlicher und privater Schlüssel der Partei B
V_S sei Verschlüsselungsfunktion mit Schlüssel S
E_S sei Entschlüsselungsfunktion zu V_S
K sei Klartext einer Nachricht von Partei A an Partei B
G sei Geheimtext
Dann lässt sich die Verschlüsselung der Nachricht K von Partei A an Partei B schreiben als:

$$G = V_{S_{B_{pub}}}(K)$$

Die Entschlüsselung lässt sich schreiben als:

$$K = E_{S_{B_{priv}}}(G)$$

Nachfolgend wird ein Überblick über die im Rahmen dieser Arbeit genannten und betrachteten Verschlüsselungsverfahren gegeben. Pro Verschlüsselungsverfahren werden die Produkte genannt, die das jeweilige Verfahren verwenden.

4.1.1 Advanced Encryption Standard

Der Advanced Encryption Standard (AES) ist eine symmetrische Blockchiffre, die von Joan Daemen und Vincent Rijmen im Jahr 1998 veröffentlicht und im Jahr 2000 als AES zertifiziert wurde. Es verwendet den Rijndael-Algorithmus mit 128 Bit Blocklänge. Die Schlüssellänge kann zwischen den Werten 128 Bit, 192 Bit und 256 Bit gewählt werden. Der Algorithmus wurde vom NIST[1] als Nachfolger für DES respektive 3DES herausgegeben.

AES wird u.a. in IEEE802.11i, SSH, IPSec eingesetzt. AES wird in den im Rahmen dieser Arbeit untersuchten Produkten TrueCrypt, FreeOTFE, loop-AES, GBDE, GELI, CGD, OpenSolaris, FileVault, BitLocker, PGP WDE und Check Point FDE verwendet. AES gilt als ausreichend sicher und ist performant. Dennoch zeigen jüngste Angriffe auf AES ([BKN09] und [BK09]), dass auch dieser Algorithmus irgendwann gebrochen wird. Weitere Details zum Algorithmus können der Spezifikation [Sta01] entnommen werden.

[1] Das National Institute of Standards and Technology ist in den USA u.a. für Standardisierungsprozesse verantwortlich.

4.1.2 Blowfish

Blowfish ist eine symmetrische Blockchiffre, die von Bruce Schneier im Jahr 1993 veröffentlicht wurde. Die Blockgröße liegt bei 64 Bit, die Schlüssellänge zwischen 32 und 448 Bit. Das Verfahren basiert auf einem Feistelnetzwerk[2]. Bruce Schneier führt auf seiner Webseite[3] zahlreiche Produkte auf, die Blowfish einsetzen. Blowfish wird in den im Rahmen dieser Arbeit untersuchten Produkten FreeOTFE, GELI, CGD und vnd verwendet. Weitere Details können der Spezifikation [Sch93] entnommen werden.

4.1.3 Camellia

Camellia ist eine symmetrische Blockchiffre, die von Mitsubishi und NTT[4] entwickelt und im Jahr 2000 veröffentlicht wurde. Die Blockgröße liegt bei 128 Bit. Die Schlüssellänge kann zwischen 128 Bit, 192 Bit und 256 Bit gewählt werden. Das Verfahren basiert auf einem Feistelnetzwerk. Camellia wurde u.a. vom EU-Projekt NESSIE[5] als empfohlener Algorithmus ausgewählt. Camellia wird in dem im Rahmen dieser Arbeit untersuchtem Produkt GELI verwendet. Weitere Details lassen sich RFC 3717 [MNM04] oder der NTTs Camellia-Webseite[6] entnehmen.

4.1.4 CAST5

CAST5 - oder auch CAST-128 genannt - ist eine symmetrische Blockchiffre, die von Carlisle Adams, Stafford Tavares im Jahr 1996 veröffentlicht wurde. Der Name setzt sich aus den Anfangsbuchstaben der Namen der Entwickler zusammen. Der Algorithmus basiert auf einem Festelnetzwerk. Die Blockgröße liegt bei 64 Bit, die Schlüssellänge zwischen 40 und 128 Bit. CAST5 wird in dem im Rahmen dieser Arbeit untersuchtem Produkt FreeOTFE verwendet. Weitere Details lassen sich RFC 2144 [Ada97] entnehmen.

[2] Feistelnetzwerk oder auch Feistelchiffre genannt ist eine von Horst Feistel (1915-1990) entwickelte Struktur für Blockchiffren. Feistelnetzwerke werden in zahlreichen Verschlüsselungsverfahren eingesetzt und besitzen folgenden Aufbau: Jeder Block wird in zwei Hälften L_0 und R_0 geteilt. Für die folgenden i Runden gilt mit f als Transformationsfunktion und K_i als Rundenschlüssel: $L_i = R_{i-1} \wedge R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$. Der verschlüsselte Text ist die Konkatenation aus L_n und R_n am Ende der Runden. Die Entschlüsselung kommt ohne eine Umkehrfunktion f^{-1} aus und lässt sich beschreiben als: $R_{i-1} = L_i \wedge L_{i-1} = R_i \oplus f(L_i, K_i)$.

[3] http://www.schneier.com/blowfish-products.html

[4] Nippon Telegraph and Telephone

[5] New European Schemes for Signatures, Integrity and Encryption

[6] http://info.isl.ntt.co.jp/crypt/eng/camellia/index.html

4.1.5 CAST6

CAST6 - oder auch CAST-256 genannt - ist eine symmetrische Blockchiffre, die auf CAST5 aufbaut. Zusätzlich zu den Entwicklern Carlisle Adams, Stafford Tavares von CAST5 haben auch Howard Heys und Michael Wiener zum Design beigetragen. Der Algorithmus wurde im Jahr 1998 veröffentlicht. Es handelt sich dabei um ein Feistelnetzwerk. Die Blockgröße wurde im Gegensatz zu CAST5 auf 128 Bit verdoppelt. Die Schlüsselgrößen liegen nun bei 128 Bit, 160 Bit, 192 Bit, 224 Bit und 256 Bit. CAST6 wird in dem im Rahmen dieser Arbeit untersuchtem Produkt FreeOTFE verwendet. Weitere Details können RFC 2612 [AG99] entnommen werden.

4.1.6 Data Encryption Standard

Der Data Encryption Standard (DES) ist eine symmetrische Blockchiffre aus dem Jahr 1975. Entwickelt wurde es von Horst Feistel und zahlreichen Mitarbeitern der IBM. DES nutzt das Schema von Feistel. Die Blockgröße beträgt 64 Bit. Die Schlüssellänge liegt bei 56 Bit und konnte daher bereits per Brute-Force-Angriff gebrochen werden. DES wird u.a. in Geldautomaten und Sprechfunkgeräten verwendet [Kub09]. In der Softwarelandschaft wurde DES in den letzten Jahren zunehmend durch 3DES bzw. AES ersetzt, so dass man heute kaum noch aktuelle Software findet, die DES einsetzt. DES wird in dem im Rahmen dieser Arbeit untersuchtem Produkt FreeOTFE verwendet. Weitere Details können der Spezifikation [Sta99] entnommen werden.

4.1.7 Triple Data Encryption Standard

Triple-DES (3DES) bezeichnet die dreifache Ausführung von DES mit drei unabhängigen Schlüsseln. Das Verfahren wurde im Jahr 1981 von Ralph Merkle und Martin Hellman vorgeschlagen. Die Schlüssellänge ist mit 168 Bits dreimal so groß wie bei DES und erhöht den Rechenaufwand eines Brute-Force-Angriffs im Vergleich zu DES um das 2^{112}-fache. 3DES wird in den im Rahmen dieser Arbeit untersuchten Produkten FreeOTFE, GELI und CGD verwendet. Weitere Details können der Spezifikation zu DES [Sta99] entnommen werden.

4.1.8 MARS

MARS ist eine Blockchiffre, die auf einem Feistelnetzwerk aufbaut. Sie wurde von D. Coppersmith im Jahr 1998 veröffentlicht. Die Blockgröße beträgt 128 Bit, die Schlüssellänge kann Werte zwischen 128 Bit und 448 Bit in jeweils 32 Bit Schritten annehmen.

MARS wird in dem im Rahmen dieser Arbeit untersuchtem Produkt FreeOTFE verwendet. Weitere Details können der Veröffentlichung zu MARS [BCD+99] entnommen werden.

4.1.9 Rivest Cipher 6

Rivest Cipher 6 (RC6) ist eine symmetrische Blockchiffre, die von Ronald Rivest im Jahr 1998 veröffentlicht wurde. RC6 basiert auf RC5. Die Blockgröße liegt bei 128 Bit, Schlüssellängen bei 128 Bit, 192 Bit und 256 Bit. RC6 wird in dem im Rahmen dieser Arbeit untersuchtem Produkt FreeOTFE verwendet. Weitere Details können der Spezifikation [RRSY98] entnommen werden.

4.1.10 Twofish

Twofish ist eine symmetrische Blockchiffre, die von Bruce Schneier, Niels Ferguson, John Kelsey, Doug Whiting, David Wagner und Chris Hall im Jahr 1998 veröffentlicht wurde. Twofish ist der Nachfolger von Blowfish. Die Blockgröße beträgt analog zu AES 128 Bit, die Schlüsselängen liegen ebenfalls bei 128 Bit, 192 Bit und 265 Bit. Der Algorithmus ist unter Public Domain und findet daher in diversen Open-Source-Projekten Verwendung. Bruce Schneier führt auf seiner Webseite[7] zahlreiche Produkte auf, die Twofish einsetzen. Twofish wird in den im Rahmen dieser Arbeit untersuchten Produkten TrueCrypt und FreeOTFE verwendet. Weitere Details können der Veröffentlichung des Standards [SKW+98] entnommen werden.

4.2 Betriebsmodi

Blockchiffren arbeiten auf Blöcken fester Länge. Da eine Klartextnachricht eine beliebige Länge haben kann, muss genau festgelegt werden, wie die Nachricht verschlüsselt wird, so dass sie unter Verwendung des gleichen Schlüssels immer den gleichen Chiffretext ergibt. Dies wird durch die Betriebsmodi beschrieben. Sie wurden zusätzlich so entworfen, dass sie die Vertraulichkeit respektive Integrität der Nachricht gewährleisten.

Festplattenverschlüsselungssysteme zerlegen zum Verschlüsseln der Festplatte die gesamte Platte in logische Einheiten, meist Sektoren. Die Einheit Sektor wird deshalb präferiert, weil es die kleinste logische Einheit auf einer Festplatte darstellt und zusätzlich der physikalische Zugriff auf einen Sektor atomar erfolgt. Die Einteilung in logische Einheiten hat den Vorteil, dass bspw. unter Verwendung von Verfahren wie Cipher Block

[7] http://www.schneier.com/twofish-products.html

Chaining (s.u.) nicht die gesamte Festplatte neu verschlüsselt werden muss, wenn sich bspw. nur der erste Klartextblock ändert.

Es folgt eine Übersicht der wichtigsten Betriesmodi. Diese werden kurz beschrieben und die Produkte aus der Marktübersicht in Kapitel 5 genannt, in denen sie Verwendung finden.

4.2.1 Electronic Code Book

Das Electronic Code Book (ECB) Modus ist eine Betriebsart für Blockverschlüsselungen. In diesem Modus wird jeder Klartextblock sequentiell und unabhängig voneinander unter Verwendung des geheimen Schlüssels in einen Geheimtextblock überführt. Da keine Verkettung zwischen den einzelnen Blöcken erfolgt, führen gleiche Klartextblöcke unter Verwendung des gleichen Schlüssels zu gleichen Geheimtextblöcken. Somit ist ECB anfällig für statistische Analysen. Der Name ECB hat seinen Ursprung darin, dass Codebücher (code books) über die Zuordnung von Geheimtextblöcken und Klartextblöcken erstellt werden können.

4.2.2 Cipher Block Chaining

Der Cipher Block Chaining (CBC) Modus ist eine Betriebsart für Blockverschlüsselungen. Das Verfahren ist wie ECB aufgebaut, mit dem Unterschied, dass jeder Klartextblock zusätzlich vor dem Verschlüsseln mit dem Geheimtextbock des Vorgängers per XOR verknüpft wird. Beim Entschlüsseln wird jeder Block nach dem Entschlüsseln mit dem Geheimtextblock des Vorgängers per XOR verknüpft. Die Verschlüsselung erfolgt also „rekursiv“: Um Block n verschlüsseln zu können, müssen alle vorherigen Blöcke, also die Blöcke 0 bis $n-1$, verschlüsselt worden sein. Für den ersten Block wird in beiden Fällen ein Initialisierungsvektor (IV) verwendet. Durch die Verkettung der Blöcke wird ein Angriff über Codebücher ad absurdum geführt. Daher gilt CBC als wesentlich sicherer als ECB.

Produkte, die im Rahmen dieser Arbeit in Kapitel 5 betrachtet werden und CBC einsetzen sind u.a. Cryptoloop, dm-crypt, GBDE, GELI, CGD, FileVault und BitLocker.

Clemens Fruhwirth führt in [Fru05] verschiedene Arten von Angriffen auf CBC im Kontext von Festplattenverschlüsselungssystemen auf: Durchsickern von Inhalten, Wasserzeichenangriffe, Durchsickern von Veränderungen, Veränderbarkeit von Inhalten, Blöcke können beliebig verschoben werden (Copy&Paste-Angriff) etc. Diese Beispiel zeigen daher die Notwendigkeit von den weiter unten beschriebenen Verfahren CMC, EME und LRW auf.

4.2.3 Cipher Feedback

Der Cipher Feedback (CFB) Modus ist eine Betriebsart für Blockverschlüsselungen. Das Verfahren kann aber auch als Stromverschlüsselung eingesetzt werden. In diesem Modus wird jeder Klartextblock sequentiell mit dem Ergebnis aus der Verschlüsselung per XOR verknüpft. Die Eingabe für die Verschlüsselung ist der Geheimtextblock des Vorgängers. Für den ersten Block wird ein Initialisierungsvektor (IV) verwendet.

Keines der im Rahmen der Marktübersicht in Kapitel 5 genannten Produkte verwendet CFB explizit. Dies schließt allerdings den möglichen Einsatz von CFB innerhalb von Kapitel 5.1.1 vorgestellten generischen Linux-Lösungen nicht aus. Diese verwenden das Linux Crypto API, welches den Einsatz beliebiger Algorithmen und Betriebsmodi ermöglicht.

4.2.4 Output Feedback

Der Output Feedback (OFB) Modus ist eine Betriebsart für Blockverschlüsselungen. Das Verfahren kann aber auch als Stromverschlüsselung eingesetzt werden. In diesem Modus wird jeder Klartextblock sequentiell mit dem Ergebnis aus der Verschlüsselung per XOR verknüpft. Die Eingabe für die Verschlüsselung ist im Unterschied zu CFB nicht der Geheimtextblock des Vorgängers, sondern das Ergebnis der Verschlüsselung des Vorgängers.

Keines der im Rahmen der Marktübersicht in Kapitel 5 genannten Produkte verwendet OFB explizit. Dies schließt wie in Kapitel 4.2.3 bereits erwähnt, den Einsatz von OFB innerhalb von Kapitel 5.1.1 vorgestellten generischen Linux-Lösungen nicht aus.

4.2.5 Counter

Statt einen Klartextblock als Eingabe für die Blockchiffrierung zu verwenden, nimmt man einen Zähler, auch Counter genannt, als Eingabe. Das Ergebnis der Verschlüsselung wird per XOR mit dem Klartextbock verknüpft und man erhält den Geheimtextblock. Nach jeder Blockverschlüsselung führt man eine Funktion f auf den Zähler aus, die die Eingenschaft hat, eine Sequenz auszugeben, die sich möglichst lange nicht wiederholt. Dies kann bspw. etwas Einfaches wie das Erhöhen des Zählers um den Wert eins sein. Ergänzend kann der Einsatz eines Initialisierungsvektors in Form einer Nonce erfolgen. Der Vorteil des Counter-Modus ist, dass man den i-ten Schlüsselblock k_i erzeugen kann, ohne dazu alle vorherigen Schlüsselblöcke generieren zu müssen. Der Zähler wird einfach auf den i-ten internen Zustand gesetzt und der gewünschte Block erzeugt, mit dem dann der Klartext per XOR verknüpft wird. Das Gleiche gilt für das Entschlüsseln: es ist

möglich jeden beliebigen Block zu entschlüsseln, ohne die vorherigen oder nachfolgenden Blöcke zu kennen. Diese Eigenschaft des beliebigen Zugriffs (engl. *random access*) ist eine ideale Voraussetzung im Kontext von Festplattenverschlüsselungssystemen, weil Festplattenzugriffe in der Regel als *random access* erfolgen.

Von den im Rahmen dieser Arbeit in Kapitel 5 betrachteten Produkten, wird Counter Mode von OpenSolaris innerhalb von ZFS crypto verwendet [Mof08].

4.2.6 CBC-Mask-CBC und ECB-Mix-ECB

Shai Haveli und Phillip Rogaway stellten in [HR03b] im Juni 2003 das CBC-Mask-CBC (CMC) Verfahren und einen Monat später in [HR03a] das ECB-Mix-ECB (EME) Verfahren vor. CMC besteht aus drei Schritten: Erst wird CBC angewandt, anschließend wird der Geheimtext mittels XOR maskiert und schließlich CBC rückwärts angewandt. Im Unterschied zu CMC ist EME parallelisierbar und besteht aus zwei Schichten ECB und einer leichtgewichtigen Mixing-Funktion. Sowohl CMC als auch EME führen zu einer gegenseitigen Abhängigkeit der Blöcke. Dadurch werden die Schwächen von CBC adressiert: CMC und EME verhindern

- das Verschieben von Blöcken
- das Manipulieren von Blöcken
- und Wasserzeichenangriffe.

Keines der im Rahmen der Marktübersicht in Kapitel 5 genannten Produkte verwendet CMC oder EME explizit. Dies schließt den Einsatz von den genannten Betriebsmodi innerhalb von Kapitel 5.1.1 vorgestellten generischen Linux-Lösungen nicht aus. Diese verwenden das Linux Crypto API, das den Einsatz beliebiger Algorithmen und Betriebsmodi ermöglicht. Clemens Fruhwirth hat bspw. eine Test-Implementation von EME für Linux geschrieben [Fru04a]. Weitere Details zu CMC und EME können den beiden oben genannten Veröffentlichungen entnommen werden.

4.2.7 Counter with CBC-MAC

Counter with CBC-MAC (CCM) Modus ist eine Betriebsart für Blockverschlüsselungen. CCM wurde entworfen, um die Authentizität und Geheimhaltung einer Nachricht zu gewährleisten. Cipher Block Chaining Message Authentication Code (CBC-MAC) ist ein Verfahren, um Message Authentication Codes (MAC) aus einer Blockchiffe für die Integritätssicherung zu generieren. CCM wird innerhalb von IEEE 802.11i und IPSec verwendet. Weitere Details können RFC 3610 [WHF03] entnommen werden.

4.2.8 Galois Counter Mode

Galois Counter Mode (GCM) ist eine Betriebsart für Blockverschlüsselungen. GCM wurde entworfen, um die Authentizität und Geheimhaltung einer Nachricht zu gewährleisten. Es verknüpft den Counter-Betriebsmodus zur Verschlüsselung mit dem Galois-Modus zum Nachweis der Authentizität. GCM wird u.a. in IPSec verwendet. Weitere Details können der Spezifikation [Dwo07] entnommen werden.

4.2.9 LRW

LRW ist der Kurzname für LRW-AES und leitet sich aus den Anfangsbuchstaben der Erfinder Liskov, Rivest und Wagner ab. In [LRW02] haben sie im Jahr 2002 eine modifizierbare Blockchiffre veröffentlicht. Modifizierbar deshalb, weil nicht nur der Klartextblock und der Schlüssel als Eingabe für die Verschlüsselung dienen, sondern ein weiterer Wert („tweak“) hinzukommt. Dieser Wert hat den gleichen Zweck, wie Initialisierungsvektoren bei CBC. Von den im Rahmen dieser Arbeit in Kapitel 5 betrachteten Produkten, wird LRW in FreeOTFE und dm-crypt verwendet. Weitere Details können der oben genannten Veröffentlichung entnommen werden.

4.2.10 Xor-Encrypt-Xor

Xor-Encrypt-Xor (XEX) ist ein weiterer modifizierbarer Modus und wurde von Rogaway entworfen, um hintereinander liegende Blöcke möglichst schnell verarbeiten und Schwächen aus LRW beheben zu können. Weitere Details zu XEX können dem Artikel zur Theorie der Festplattenverschlüsselung in der englischen Wikipedia unter [ea09c] entnommen werden.

4.2.11 XEX-based Tweaked CodeBook with Cipher Text Stealing

XTS steht für „XEX-based Tweaked Code Book (TCB) with Cipher Text Stealing (CTS)“. Die Abkürzung leitet sich daher aus XEX-TCB-CTS ab, wobei der dritte Buchstabe durch das S ersetzt wird, um Verwechselungen mit XTC[8] zu vermeiden. XTS ist inzwischen als IEEE P1619 Standard for Cryptographic Protection of Data on Block-Oriented Storage Devices standardisiert. Von den im Rahmen dieser Arbeit in Kapitel 5 betrachteten Produkten, wird XTS in TrueCrypt, dm-crypt, Softraid CRYPTO verwendet. Weitere Details hierzu können der IEEE Security in Storage Working Group Webseite[9] entnommen werden.

[8] XTC ist die Abkürzung für die Droge „ecstacy“.
[9] https://siswg.net/index.php

4.3 Initialisierungsvektoren

Unter einem Initialisierungsvektor versteht man eine Menge an Zufallsdaten, meist IV genannt, die innerhalb mancher Betriebsmodi benötigt werden, um vorrangig Known-Plaintext-Attacken[10] abzuwehren.

4.3.1 Encrypted Salt-Sector Initialization Vector

Encrypted Salt-Sector Initialization Vector (ESSIV) ist ein von Clemens Fruhwirth in [Fru05] vorgestelltes Verfahren, um Initialisierungsvektoren für Blockverschlüsselungen *im Kontext von Festplattenverschlüsselungssystemen* zu generieren. Das Verfahren ist besonders resistent gegen Wasserzeichenangriffe. Mathematisch ausgedrückt:

IV sei Initialisierungsvektor
sektor sei logische Einheit des Datenträgers
V_s sei Verschlüsselungsfunktion mit Eingabe s als Salt
S sei Schlüssel
H sei eine Hash-Funktion, die Hash-Werte mit einer für V passenden Länge produziert, bspw. 128 Bit für AES

$$IV(sektor) = V_s(sektor) \wedge s = H(S)$$

Von den im Rahmen dieser Arbeit in Kapitel 5 betrachteten Produkten, wird ESSIV in FreeOTFE und dm-crypt verwendet.

4.4 Hash-Funktionen

Prüfsummen werden in Abhängigkeit der zu bearbeitenden Daten gebildet, d.h. die Daten dienen als Eingabe für die Prüfsummenfunktion. Das charakteristische Kennzeichen einer Prüfsumme ist seine Länge. Im Verhältnis zu den Daten ist die Prüfsumme meist sehr kurz, damit nicht unnötig Bandbreite durch die Übermittlung von Metadaten verwendet wird. Die Hash-Funktionen lassen sich gliedern in:

Schlüssellose Hash-Funktionen weisen keine Abhängigkeit zu einem innerhalb einer Verschlüsselung verwendeten Schlüssel auf und sind:

- Einweg-Hash-Funktionen

[10] In einem Known Plaintext-Angriff kennt der Angreifer sowohl Geheimtext als auch Klartext der Nachricht und versucht durch dieses Wissen den Schlüssel zu errechnen.

- Kollisionsresistente Hash-Funktionen (Untermenge der Einweg-Hash-Funktionen)

Schlüsselabhängige Hash-Funktionen hingegen weisen eine Abhängigkeit zu Schlüsselmaterial einer im Zusammenhang der Hash-Funktion durchgeführten Verschlüsselung auf und sind:

- Message Authentication Codes (MAC)
- Keyed-Hash Message Authentication Code (HMAC)

Mathematisch lassen sich Hash-Funktionen wie folgt beschreiben:
Sei H Hash-Funktion.
Sei x Funktionsargument.
Sei y Funktionswert.
Für Einweg-Hash-Funktionen muss dann gelten:

- Zu einem gegebenen $y = H(x)$ ist es praktisch unmöglich ein x zu finden (Einwegfunktion).
- Zu einem gegebenen x ist es praktisch unmöglich ein x' zu finden, so dass gilt: $H(x) = H(x') \wedge x \neq x'$ (schwache Kollisionsresistenz).

Für kollisionsresistente Hash-Funktionen muss *zusätzlich* gelten:

- Es ist praktisch unmöglich x und x' so zu finden, dass gilt: $H(x) = H(x') \wedge x \neq x'$ (starke Kollisionsresistenz).

Hash-Funktionen sind Funktionen, die eine Abbildung aus der Quellmenge in die Zielmenge führen und dabei die nachfolgenden Kriterien erfüllen:

- Datenreduktion
- Deterministisches Chaos
- Surjektivität
- Effizienz

Zusätzlich müssen folgende Bedingungen gelten:

- Es ist praktisch nahezu unmöglich, für ein Element aus der Zielmenge das ursprüngliche Element aus der Quellmenge zu berechnen.
- Die Anzahl von effizient berechenbaren Kollisionen muss so gering wie möglich gehalten werden.

Hash-Funktionen werden im Rahmen von Festplattenverschlüsselungssystemen als Pseudozufallszahlengeneratoren und zur Prüfung der Integrität von Daten verwendet. Zusätzlich werden sie eingesetzt, um einen Schlüsselraum besser auszunutzen.

4.4.1 Message-Digest Algorithm 2

Message-Digest Algorithm 2 (MD2) ist eine Hash-Funktion, die 128 Bit lange Hash-Werte erzeugt. Sie ist für 8 Bit Architekturen optimiert und wurde 1989 von Ronald Rivest entworfen und veröffentlicht. Weitere Details sind RFC 1319 [Kal92] zu entnehmen. Seit Frédéric Muller seinen Angriff auf MD2 in [Mul04] veröffentlicht hat, gilt MD2 als nicht mehr sicher. Neben seinem Angriff gibt es weitere wirksame Angriffe gegen MD2.

4.4.2 Message-Digest Algorithm 4

Message-Digest Algorithm 4 (MD4) ist eine Hash-Funktion, die 128 Bit lange Hash-Werte erzeugt. Sie ist im Gegensatz zu MD2 für 32 Bit Architekturen optimiert. MD4 wurde, wie MD2 von Ronald Rivest entworfen. Er hat den Algorithmus im Jahr 1990 veröffentlicht. Weitere Details sind RFC 1320 [Riv92a] zu entnehmen. Bert den Boer und Antoon Bosselaers wiesen bereits ein Jahr später in [dBB91] auf Schwächen in MD4 hin. In den nachfolgenden Jahren wurden weitere Angriffe auf MD4 veröffentlicht, so dass MD4 heute nicht mehr als sicher angesehen werden kann. Kein Produkt, der im Rahmen der Marktübersicht in Kapitel 5 genannten Produkte, verwendet MD4 explizit. Dies schließt wie in 4.2.3 erwähnt, den Einsatz von MD4 innerhalb von Kapitel 5.1.1 vorgestellten generischen Linux-Lösungen nicht aus.

4.4.3 Message-Digest Algorithm 5

Message-Digest Algorithm 5 (MD5) ist eine Hash-Funktion, die 128 Bit lange Hash-Werte erzeugt. MD5 wurde, wie bereits MD2 und MD4 ebenfalls von Ronald Rivest entworfen und im Jahr 1992 veröffentlicht. Weitere Details sind RFC 1321 [Riv92b] zu entnehmen. MD5 ist weit verbreitet innerhalb der Softwarelandschaft, vor allem der Open-Source Softwarelandschaft. Bert den Boer und Antoon Bosselaers veröffentlichten im Jahr 1993 in [dBB93] den ersten Angriff auf MD5. Drei Jahre später folgte der Angriff von Hans Dobbertin, der in [Dob96] eine Kollision der Kompressionsfunktion von MD5 erzeugte. Experten innerhalb der Kryptographie-„Gemeinde“ als auch die Fachliteratur empfahlen nun den Wechsel weg von MD5 hin zu SHA-1 und RIPEMD-160. Die Netzgemeinde wurde zunehmend sensibilisiert, zahlreiche Open-Source-Projekte wechselten von MD5 zu SHA-1. Neun Jahre später gelang einer chinesischen Gruppe von Forschern schließlich die Erzeugung von Kollisionen für MD5. Sie veröffentlichten ihre Ergebnisse in [WFLY04]. Vier Jahre später stellte eine Gruppe von Hackern[11] Ende 2008 auf dem

[11] Der Begriff *Hacker* ist im ursprünglichen Sinne zu verstehen, d.h. wie u.a. in RFC 1392 [MP93] definiert.

25. Chaos Communication Congress (25C3)[12] einen Aufsehen erregenden Angriff auf die PKI von SSL vor[13] und nutzte hierbei MD5 Kollisionen ausgenutzt. Eine umfangreiche Beschreibung des Angriffs befindet sich in [SSA+08]. Der Gruppe ist es gelungen, ein gefälschtes CA-Zertifikat zu erstellen, welches von allen Browsern akzeptiert wurde, weil es offenbar von einer legitimen Wurzel-CA unterzeichnet wurde. Die Brisanz dieses Angriffs zeigt zum einen wie weit verbreitet MD5 immer noch ist und zum anderen, dass Kollisionen nicht nur theoretisch auf dem Papier, sondern auch in der Praxis möglich sind. Marc Bevand hat in seinem Vortrag [Bev09] auf der Black Hat USA 2009 am 30. Juli 2009 vorgestellt, wie GPUs verwendet werden können, um Kollisionen noch effizienter berechnen zu können. Von den im Rahmen dieser Arbeit in Kapitel 5 betrachteten Produkten wird MD5 in FreeOTFE, loop-AES, GBDE und GELI verwendet.

4.4.4 RACE Integrity Primitives Evaluation Message Digest 128

RACE[14] Integrity Primitives Evaluation Message Digest 128 (RIPEMD-128) ist eine Hash-Funktion, die 128 Bit lange Hash-Werte erzeugt. Der Algorithmus wurde von Antoon Bosselaers, Bart Preneel und Hans Dobbertin als Nachfolger für RIPEMD entwickelt und im Jahr 1996 veröffentlicht. RIPEMD basiert auf MD4. Die Sicherheit von RIPEMD war in Frage gestellt worden[15]. Daher bestand die Notwendigkeit eines Nachfolgers. Zusätzlich zur 128 Bit Version gibt es auch eine 256 Bit Version (RIPEMD-256), die sich lediglich in der Länge des Hash-Wertes unterscheidet. Von den im Rahmen dieser Arbeit in Kapitel 5 betrachteten Produkten, wird RIPEMD-128 in FreeOTFE verwendet.

4.4.5 RACE Integrity Primitives Evaluation Message Digest 160

RACE Integrity Primitives Evaluation Message Digest 160 (RIPEMD-160) ist eine Hash-Funktion, die 160 Bit lange Hash-Werte erzeugt. Der Algorithmus wurde von Antoon Bosselaers, Bart Preneel und Hans Dobbertin im Jahr 1996 veröffentlicht. Es gibt analog zu RIPEMD-128, ebenfalls eine Version, die einen doppelt so langen Hash-Wert generiert und sonst keine Unterschiede aufweist: RIPEMD-320. Hierdurch wird lediglich die Wahrscheinlichkeit für Kollisionen gesenkt. RIPEMD-160 wurde im Unterschied zu

[12] http://events.ccc.de/congress/2008/

[13] Das Vortragsvideo befindet sich unter http://chaosradio.ccc.de/25c3_m4v_3023.html

[14] Research and Development in Advanced Communications Technologies in Europe (RACE) ist ein Programm, welches 1988 von der Europäische Komission ins Leben gerufen wurde, um den Ausbau von Breitband-Internet in Europa zu fördern.

[15] Im Jahr 2004 ist eine Kollision in [WFLY04] veröffentlicht worden. Somit haben sich die Befürchtungen um die Sicherheit von RIPEMD bestätigt.

SHA-1 offen innerhalb der Kryptographie-Gemeinde entwickelt. RIPEMD-160 weist eine ähnliche Performanz, wie SHA-1 auf und bietet somit eine „offenere“ Alternative zu SHA-1. Von den im Rahmen dieser Arbeit in Kapitel 5 betrachteten Produkten, wird RIPEMD-160 bspw. in TrueCrypt verwendet. Weitere Details können der Spezifikation [DBP96] entnommen werden.

4.4.6 Secure Hash Algorithm 1

Secure Hash Algorithm 1 (SHA-1) ist eine Hash-Funktion, die 160 Bit lange Hash-Werte erzeugt. Im Jahr 1993 wurde der Secure Hash Standard, FIPS[16] PUB 180 vom NIST veröffentlicht. Dieser Algorithmus wird heute als SHA-0 bezeichnet. NSA[17] hat wegen Sicherheitsbedenken eine Überarbeitung des Algorithmus vorgenommen und im Jahr 1995 eine überarbeitete Version, FIPS PUB 180-1 veröffentlicht. Dieser Algorithmus wird heute als SHA-1 bezeichnet. SHA-0 und SHA-1 basieren auf ähnlichen Entwurfsmustern wie MD4 und MD5. Weitere Details sind RFC 3174 [DEJ01] zu entnehmen. Erste Angriffe gegen SHA-1 wurden im Jahr 2005 veröffentlicht, wobei bis heute keine Kollisionen bekannt sind. Im August 2005 haben Xiaoyun Wang, Andrew Yao und Frances Yao einen Kollisionsangriff mit einem Aufwand von 2^{63} veröffentlicht. Derzeit versucht SHA-1 Collision Search Graz[18] mit Hilfe von verteiltem Rechnen[19] eine SHA-1 Kollision zu finden. Den jüngsten Angriff auf SHA-1 haben Cameron McDonald, Philip Hawkes und Josef Pieprzyk im Juni 2009 in [MHP09] veröffentlicht, aber auf Grund von nicht erfüllten Annahmen inzwischen zurückgezogen. Sie arbeiten an einer neuen Veröffentlichung. Von den im Rahmen dieser Arbeit in Kapitel 5 betrachteten Produkten, wird SHA-1 bspw. in FreeOTFE verwendet.

4.4.7 Secure Hash Algorithm 224/256/384/512

Secure Hash Algorithm 224/256/384/512 gehören zu der Familie der SHA-2 Algorithmen und erzeugen im Gegensatz zu SHA-1 lediglich längere Hash-Werte: 224, 256, 384 respektive 512 Bit. Von den im Rahmen dieser Arbeit in Kapitel 5 betrachteten Produkten, wird SHA-512 bspw. in TrueCrypt und FreeOTFE verwendet.

[16] Federal Information Processing Standard (FIPS) sind von der US-Regierung veröffentlichete Standards
[17] National Security Agency (NSA) ist der größte Nachrichtendienst der USA.
[18] `http://boinc.iaik.tugraz.at/sha1_coll_search/`
[19] `http://distributed.net/` führt seit Jahren ähnliche Projekte durch.

4.4.8 Tiger

Tiger ist eine Hash-Funktion, die 192 Bit lange Hash-Werte erzeugt. Der Algorithmus wurde von Ross Anderson und Eli Biham im Jahr 1995 entwickelt und ein Jahr später in [AB96] veröffentlicht. Tiger ist speziell für 64 Bit Architekturen optimiert. Von den im Rahmen dieser Arbeit in Kapitel 5 betrachteten Produkten, wird Tiger in FreeOTFE verwendet.

4.5 Schlüsselgenerierung

Schlüssel bilden im Rahmen der Kryptographie neben der eigentlichen Implementierung häufig das schwächste Glied in der Gesamtkette der verwendeten Komponenten. Menschen neigen dazu, Passwörter zu verwenden, die leicht zu merken statt kryptografisch sicher sind. Selbst Kryptographie-Experte Bruce Schneier schreibt in seinem Blog, dass er gegen zahlreiche Passwort-Regeln verstößt [Sch09]. Daher wurden Verfahren vorgeschlagen, die die Qualität des Schlüsselmaterials verbessern. Nachfolgend soll eines der Verfahren vorgestellt werden. Zusätzlich muss das nachträgliche Ändern von Schlüsselmaterial möglich sein. Hierzu wird ebenfalls ein Verfahren vorgestellt.

4.5.1 Password-Based Key Derivation Function 2

Password-Based Key Derivation Function 2 (PBKDF2) ist Teil von Password-Based Cryptography Specification 5 (PKCS #5) und wurde von RSA Laboratories im Jahr 1999 spezifiziert. PBKDF2 wird verwendet, um Schlüsselmaterial abzuleiten. Hierzu wird die Eingabe um einen Salt ergänzt. Das konkatenierte Ergebnis dient als Eingabe für eine Hash-Funktion. Dadurch wird die Entropie der Eingabe erhöht und somit werden Wörterbuch- und Brute-Force-Angriffe erschwert. Das Ergebnis kann als (meist stärkerer) kryptographischer Schlüssel für die eigentliche Verschlüsselung verwendet werden. Von den im Rahmen dieser Arbeit in Kapitel 5 betrachteten Produkten, wird PBKDF2 bspw. in TrueCrypt verwendet. Weitere Details sind der Spezifikation [Lab99] oder RFC 2898 [Kal00] zu entnehmen.

4.5.2 TKS1

TKS1 ist ein von Clemens Fruhwirth im Jahr 2004 in [Fru04c] vorgestelltes Verfahren, das eine zweistufige Schlüsselhierarchie vorschlägt. Das Verfahren ermöglicht das Ändern von Kennwörtern und adressiert das Problem, dass Daten auf magnetischen Datenträgern

schwer zuverlässig zu löschen sind. Von den im Rahmen dieser Arbeit in Kapitel 5 betrachteten Produkten ist LUKS nach dem TKS1-Template entwickelt worden.

4.6 Glaubhafte Abstreitbarkeit

Abstreitbare Verschlüsselung[20] erlaubt es, die Existenz von verschlüsselten Daten abzustreiten. Ziel ist es, sensitive Daten zu schützen, selbst wenn man zur Herausgabe von Schlüsselmaterial gezwungen wird. Hierzu werden unterschiedliche Daten mit unterschiedlichen Schlüsseln verschlüsselt. Meist werden wie in Abbildung 4.1 scheinbar sensitive Daten mit einem Schlüssel verschlüsselt und die eigentlich zu schützenden Daten mit einem anderen Schlüssel. Die in der Abbildung dargestellte logische Sicht mit zwei getrennten Datenbereichen ist nur für den Benutzer sichtbar, weil er das Schlüsselmaterial besitzt. Ein Angreifer kann von außen nicht erkennen, dass zwei getrennte Datenbereiche vorliegen. Erzwingt ein Angreifer die Herausgabe des Schlüssels, dann wird nur der Schlüssel zu den scheinbar sensitiven Daten herausgegeben. Die Existenz weiterer verschlüsselter Daten kann dadurch abgestritten werden. Hierbei bedient man sich der Tatsache, dass verschlüsselte Daten und echte Zufallszahlen nach außen hin gleich aussehen und somit für den Angreifer nicht ersichtlich ist, ob weitere verschlüsselte Daten vorliegen oder nicht. Der Angreifer sieht Zufallszahlen. Damit das Konzept aufgeht, muss das System beim Einrichten den Datenträger *immer* vollständig mit Zufallszahlen hoher Entropie füllen. Ein guter Verschlüsselungsalgorithmus zeichnet sich dadurch aus, dass das verschlüsselte Material ebenfalls eine hohe Entropie aufweist und somit von Zufallszahlen nicht unterschieden werden kann.

Um das Konzept der glaubhaften Abstreitbarkeit zu vollenden, kann ein One-Time-Pad verwendet werden. Hiermit kann aus *jedem* Geheimtext *jeder* Klartext erzeugt werden, wenn nur der entsprechende Schlüssel verwendet wird. Mathematisch lässt sich das wie folgt ausdrücken:

S sei One-Time-Pad
$K_{sensitiv}$ sei sensitiver Klartext
$K_{scheinbar}$ sei scheinbar sensitiver Klartext
$S_{scheinbar}$ sei scheinbarer Schlüssel
G sei Geheimtext

Der Klartext $K_{sensitiv}$ wird unter Verwendung des One-Time-Pad S verschlüsselt:

$$K_{sensitiv} \oplus S = G$$

[20] Engl.: deniable encryption

Der Benutzer des Systems kennt den Schlüssel S und kann somit den Geheimtext G wie folgt entschüsseln:

$$G \oplus S = K_{sensitiv}$$

Der Benutzer des Systems definiert $S_{scheinbar}$ wie folgt:

$$S_{scheinbar} = G \oplus K_{scheinbar}$$

Ein Angreifer erzwingt die Herausgabe des Schlüssels S. Der Benutzer gibt aber $S_{scheinbar}$ heraus. Da S ein One-Time-Pad ist und somit aus (echten) Zufallszahlen besteht, gibt es keine Möglichkeit nachzuweisen, dass $S_{scheinbar}$ nicht der richtige Schlüssel ist. Der Angreifer entschlüsselt somit den Geheimtext G unter Verwendung von Schlüssel $S_{scheinbar}$:

$$G \oplus S_{scheinbar} = G \oplus G \oplus K_{scheinbar} = K_{scheinbar}$$

Das One-Time-Pad implementiert somit eine perfekt abstreitbare Verschlüsselung. Neben der Theorie ist die Frage nach der Implementierbarkeit und Benutzbarkeit eines One-Time-Pad ebenso wichtig und wie Bruce Schneier in [Sch02] argumentiert, längst nicht so trivial.

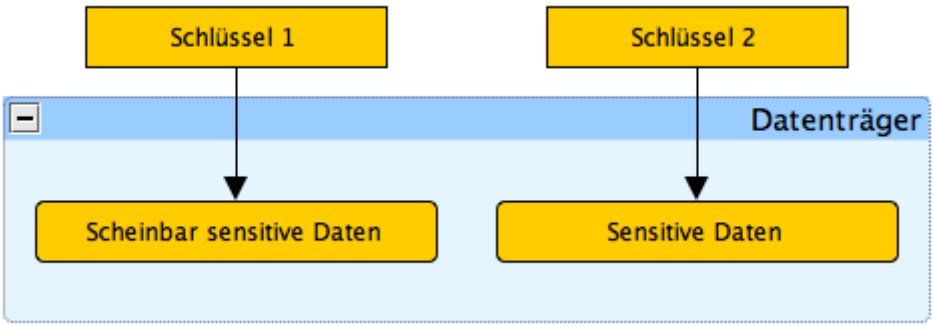

Abbildung 4.1: Glaubhafte Abstreitbarkeit durch unterschiedliches Schlüsselmaterial

[Kar07] unterteilt abstreitbare Verschlüsselung in statische und dynamische Verfahren. Das statische Verschlüsselungsverfahren erlaubt Änderungen an verschlüsselten Daten nur nach Entschlüsselung aller versteckten Nachrichten. Das dynamische Verschlüsselungsverfahren erlaubt es hingegen eine verschlüsselte Nachricht zu ändern, ohne wissen zu müssen, ob weitere versteckte Nachrichten vorliegen.

Das Konzept der abstreitbaren Verschlüsselung wurde erstmalig in [AWD] implementiert und später in [CDNO97] theoretisch fundiert.

4.7 Trusted Computing

Nach [vH08] bezieht sich der Begriff *Trusted Computing* erst einmal auf eine Sammlung offener und nicht-proprietärer Spezifikationen der Trusted Computing Group[21] (TCG). Das verfolgte Ziel ist es, die Sicherheit von IT-Systemen durch den Einsatz von vertrauenswürdiger Hard- und Software zu erhöhen. Ein zentraler Bestandteil einer Trusted Platform ist das Trusted Platform Module (TPM). Im Rahmen dieser Arbeit wird Trusted Computing beleuchtet, weil es vielversprechende Ansätze bietet, um das Problem des vertrauenswürdigen Bootvorgangs und somit die Sicherheitsanforderung S2 zu adressieren.

4.7.1 Trusted Computing Platform Alliance und Trusted Computing Group

Die Trusted Computing Platform Alliance (TCPA) wurde 1999 von Compaq, HP, IBM, Intel und Microsoft gegründet. Ziel der Allianz war nach [Eck07] die Spezifikation von Hard- und Softwarestandards, um das Vertrauen in Computer-Plattformen im Kontext von E-Business-Transaktionen zu erhöhen. Die Zahl der Mitglieder wuchs rapide an, so dass man im Jahr 2003 bereits über 200 Mitglieder verzeichnete. Alle Entscheidungen mussten einstimmig gefällt werden. Bei der großen Zahl von Mitgliedern war dies ein wachsendes Problem und der Auslöser dafür, dass AMD, HP, Intel und Microsoft im Jahr 2003 die Trusted Computing Group (TCG) gründeten.

4.7.2 Trusted Computing Group-Architektur

Die TCG Specification Architecture Overview [ea07b] beschreibt im Rahmen einer TCG-Architektur verschiedene Bestandteile, u.a. die sogenannten *Roots of Trust* . Eine Trusted Platform enthält meist drei Wurzeln des Vertrauens:

Root of trust for measurement (RTM) ist in der Lage zuverlässige Integritätsmessungen, bspw. während des Boot-Vorgangs, durchzuführen. Hierbei bildet das Core Root of Trust for Measurement (CRTM) eine Ausnahmerolle für PC-Architekturen. Das CRTM besteht aus ausführbarem Code, der als Teil des BIOS gespeichert (oder im TPM integriert) ist und von einer externen dritten Instanz zertifiziert wurde. Es wird bei jedem Neustart des Systems als Erstes ausgeführt. Zusätzlich bildet das RTM die Wurzel für die transitive Vertrauenskette.

[21] Die Trusted Computing Group ist ein Konsortium aus zahlreichen Hard- und Softwareherstellern, die Standards für die Trusted Computing Platform entwickeln. Weitere Details sind `http://www.trustedcomputinggroup.org` zu entnehmen.

Root of trust for storage (RTS) ist in der Lage, Integritätsmesswerte und Schlüssel sicher zu speichern.

Root of trust for reporting (RTR) ist in der Lage, zuverlässig über die im RTS abgelegten Informationen zu berichten (Attestierung).

Das TPM implementiert hierbei das RTS und das RTR.

Ergänzend zum TPM gibt es die Trusted Software Stack (TSS). Sie stellt eine Standard-API für Anwender bereit. Über diese API können die durch das TPM bereitgestellten Funktionen und Dienste genutzt werden.

4.7.3 Trusted Building Blocks

Als Trusted Building Blocks (TBB) bezeichnet man alle Bestandteile der *Roots of Trust*, die keine abgeschirmten Umgebungen oder geschützten Eigenschaften aufweisen. Üblicherweise enthalten sie lediglich die Instruktionen des RTM sowie TPM Initialisierungsfuktionen und sind zusätzlich meist plattformspezifisch. Die TBB einer PC-Plattform sind die in Abbildung 4.2 fett markierten Bestandteile.

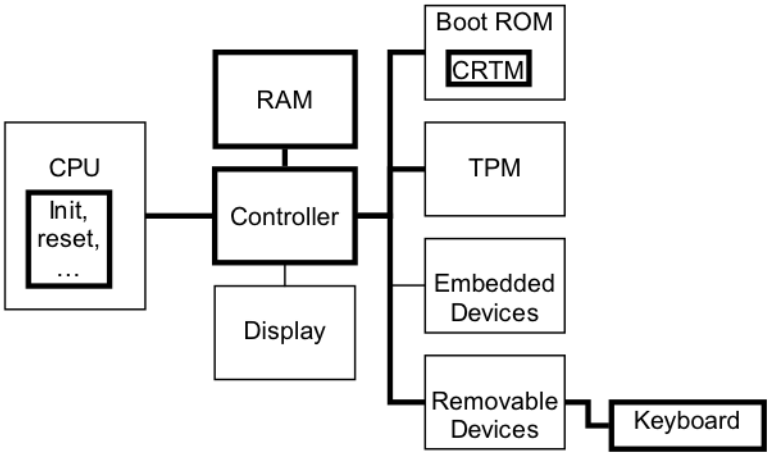

Abbildung 4.2: Trusted Building Blocks innerhalb einer PC-Plattform. Quelle: [ea07b]

4.7.4 Core Root of Trust for Measurement

Das Core Root of Trust for Measurement (CRTM) ist das *Static Root of Trust for Measurement* und bezeichnet die CPU, die nach einem Neustart Programmcode im BIOS ausführt: Es wird ein Messvorgang über einzelne Systemzustände außerhalb der Trusted Platform durchführt, die Ergebnisse der Messvorgänge werden innerhalb der Trusted

Platform ablegt. Die RTM ist abstrakt betrachtet eine Funktion, die ausgeführt wird, wenn die bisherige Ausführugshistorie der Plattform keinen Einfluss auf die zukünftige Ausführung der Plattform haben kann. Um diese Funktionalität zu gewährleisten, wurde sie in der zeitlichen Abfolge an den unmittelbaren Beginn des Bootvorgangs gelegt. Die Implementation des CRTM erfolgt im BIOS, obwohl die Spezifikation es offen lässt, diese im TPM selbst umzusetzen. Die Integration im BIOS kann auf zwei Arten erfolgen:

1. **Das BIOS Boot Block enthält das CRTM.**

   Vorteile: Das CRTM ist ein kleiner und überschaubarer Teil des TBB. Andere Bereiche des BIOS können wie gewohnt aktualisiert werden.

   Nachteil: Das CRTM muss die anderen Bereiche des BIOS überwachen.

2. **CRTM liegt an beliebiger Stelle im BIOS.**

   Vorteil: Das gesamte BIOS wird zum Teil des TBB.

   Nachteil: Aktualisierungen können ausschließlich durch den Hersteller der Plattform erfolgen.

4.7.5 Dynamic Root of Trust for Measurement

Das Dynamic Root of Trust for Measurement (DRTM) trägt der Tatsache Rechnung, dass isolierte Ausführungsumgebungen im Zuge von Virtualisierungslösungen hoch- und heruntergefahren werden können. Dennoch gilt hier das Gleiche wie für CRTM: Die bisherige Ausführungshistorie einer isolierten Ausführungsumgebung darf keinen Einfluss auf die zukünftigee Ausführung anderer isolierter Ausführungsumgebungen haben. DRTM bezeichnet somit die CPU nach dem Neustart der Ausführungsumgebung. Das DRTM misst die Ausführungsumgebung vor ihrem Start.

4.7.6 Trusted Platform Module

Nach [Pro06] bietet ein Trusted Platform Module (TPM) eine isolierte Umgebung an, die den Zugriff auf die Daten in der Umgebung beschränkt. Zusätzlich muss ein TPM in der Lage sein, diese Daten auch bei ausgeschalteter Plattform zu schützen.

Nach [CYC07] (Seite 10) gewährleistet das TPM folgende Ziele:

- Private Schlüssel verlassen das TPM nie.
- Schadsoftware wird immer erkannt.
- Es wird verhindert, dass Schadsoftware Zugriff auf die privaten Schlüssel erhält.

- Schlüsselmaterial kann durch einen physischen Angriff nur schwer ermittelt werden.

Das TPM erreicht diese Ziele durch drei Gruppen von Funktionen:

Public Key Authentifizierung Private Schlüssel werden im Chip erstellt und können das TPM niemals in unverschlüsselter Form verlassen.

Funktionen zur Integritätsmessung Im Rahmen des Trusted Boot werden Hash-Werte über Bestandteile der Boot-Software gebildet und während des Boot-Prozesses in PCRs abgelegt. Anschließend können Daten mit der gebildeten PCR-Konfiguration versiegelt werden. Die versiegelten Daten können nur dann entsiegelt werden, wenn die PCRs die gleichen Werte wie beim Versiegeln aufweisen. Denkbar wäre bspw. ein System, welches nur dann seinen Bootvorgang abschließt, wenn die Integrität des Systems sichergestellt wurde. Alternativ könnte Schlüsselmaterial für eine verschlüsselte Partition an bestimmte PCR-Werte gebunden werden, so dass die Partition nur dann entschlüsselt werden kann, wenn genau diese vorgegeben PCR-Werte vorliegen. Hierzu wird die `seal`-Operation verwendet, die Schlüsselmaterial an bestimmte PCR-Werte bindet.

Funktionen zur Attestierung Das TPM ist in der Lage auf Anfragen durch Dritte, Integritätsaussagen über die Plattform zu attestieren. Hierzu werden entweder Attestation Identity Keys (AIK) oder das mit der TPM Spezifikation 1.2 eingeführte Direct Anonymous Attestation[22] (DAA) verwendet.

Die TPM-Architektur besteht nach [Eck07] aus den in Abbildung 4.3 dargestellten Komponenten, die über einen internen Nachrichtenbus miteinander kommunzieren und nachfolgend kurz beschrieben werden. Die Kommunikation mit der Außenwelt erfolgt über eine IO-Schnittstelle.

Nicht-flüchtiger Speicher Dieser Speicher enthält ein 160 Bit Data Integrity Register (DIR) zum Ablegen sicherheitskritischer Daten (bspw. Platform Endorsment-Zertifikat des Herstellers).

Platform Configuration Register (PCR) Innerhalb der PCR können bis zu 16 Hash-Werte mit der Länge von 160 Bit abgelegt werden. Sie werden verwendet, um beim Bootvorgang Messwerte über Teile der Systemkonfiguration sicher abzulegen. Nach einem erfolgreichem Boot-Vorgang können sie mit Referenzwerten verglichen werden, um Aussagen über die Integrität der Plattform zu treffen. Hash-Werte können nur über die `extend`-Operation geschrieben und **nicht** direkt gesetzt werden. Durch diese Vorgehensweise wird das direkte Setzen von PCR-Werten durch einen Angreifer unterbunden. Die `extend`-Operation arbeitet wie folgt:
Sei H Hash-Funktion.

[22] DAA ermöglicht die entfernte Authentisierung eines TPMs unter Beibehaltung der Privatsphäre des Benutzers. Hierbei wird ein Zero-Knowledge-Protokoll verwendet. Weitere Details sind [BCC04] zu entnehmen.

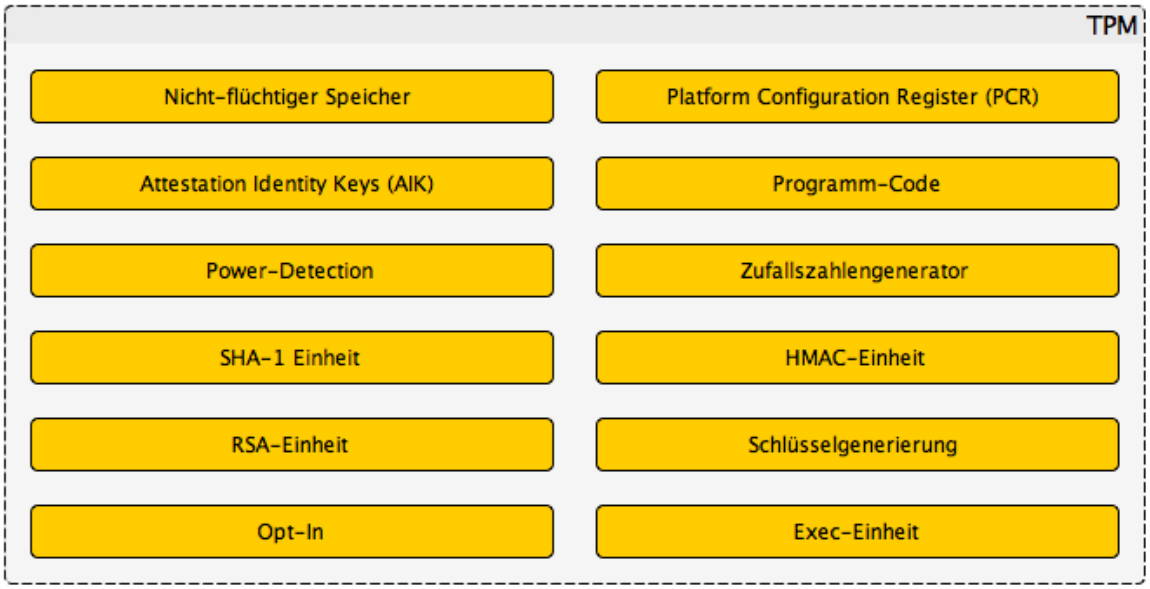

Abbildung 4.3: Komponenten eines TPM. Quelle: Angelehnt an Abbildung 11.23 aus [Eck07]

Sei x neuer Hash-Wert.

$$pcr_{i_{neu}} = H(pcr_{i_{alt}}|x)$$

Attestation Identity Keys (AIK) Diese 2048 Bit langen Signaturschlüssel werden zum Signieren von Daten verwendet, so dass ein außenstehender Dritter prüfen kann, ob es sich bei dem TPM um einen echten TPM handelt. Hierzu wird eine PKI verwendet, die ein CA benötigt.

Programm-Code Diese Einheit enthält eine Firmware, mit der es möglich ist, die Integrität der Geräte auf der TCG-Plattform zu überprüfen.

Power-Detection Diese Komponente überwacht die Energieversorgung der Plattform.

Zufallszahlengenerator Das TPM enthält einen Hardware-basierten Zufallszahlengenerator, der für alle kryptografischen Funktionen verwendet wird. Im Gegensatz zu einem Software-basierten Pseudo-Zufallszahlengenerator bietet dieser eine wesentlich bessere Entropiequelle für kryptografische Operationen.

SHA-1 Einheit Hiermit können SHA-1 Hash-Werte berechnet werden. Die Funktion kann sowohl innerhalb als auch von außerhalb des TPM verwendet werden.

HMAC-Einheit Diese Einheit berechnet Hash Message Authentication Code-Werte und kann nur intern vom TPM verwendet werden. HMACs sind Message Authentication Codes, die auf kryptografischen Hash-Funktionen basieren. HMAC ist in FIPS PUB 198 [Sta02] standardisiert.

RSA-Einheit Diese Einheit dient der RSA-Ver- und Entschlüsselung und dem Erstellen von RSA-Signaturen. Die Implementierung und die Datenformate folgen PKCS#1 [Lab02].

Schlüsselgenerierung Im TPM können sowohl symmetrische als auch asymmetrische Schlüssel generiert werden. Bei der Erstellung des Schlüssels muss angegeben werden, wofür der Schlüssel verwendet werden soll: Zum signieren, verschlüsseln etc.

Opt-In Die Aktivierung und Deaktivierung des Chips erfolgen durch die Opt-In Einheit. Eine Deaktivierung setzt die Dienste der TCG-Plattform außer Kraft. Zum Setzen des Opt-In Flags muss eine Autorisierung durch den Besitzer erfolgen oder die physische Präsenz an der Plattform nachgewiesen werden.

Exec-Einheit Diese Einheit führt Programmcode aus, um TPM-Befehle auszuführen, welche über den I/O-Port an das TPM übermittelt wurden.

4.7.7 Transitives Vertrauen

Das transitive Vertrauen ermöglicht die Erweiterung der Vertrauensgrenze. Jede in Abbildung 4.4 dargestellte Schicht ist verantwortlich, die Vertrauenswürdigkeit der nachfolgenden Schicht zu bestimmen und die gemessenen Integritätswerte dem TPM zu berichten. Konkret heisst das: Die CPU führt den CRTM-Code aus. Dieser misst den Betriebssystemladercode und schreibt den Messwert in ein PCR. Anschließend wird im zweiten Schritt die Ausführung an den Betriebssystemlader gegeben. Dieser misst in einem dritten Schritt den Betriebssystemcode, schreibt den Messwert in ein PCR und gibt die Ausführung an den Betriebssystemcode weiter. Die CPU führt nun Betriebssystemcode aus, die die Aufgabe hat, den Anwendungscode zu messen. Der Messwert wird in einem PCR ablegt und schließlich die Kontrolle an das Anwendungsprogramm gegeben. Durch dieses Vorgehen und die Existenz einer vertrauenswürdigen Wurzel ist es möglich, eine Kette des transitiven Vertrauens aufzubauen, so dass man im Nachhinein verlässliche Integritätswerte in den PCRs vorliegen hat.

4.7.8 Trusted Boot

Trusted Boot bezeichnet einen vertrauenswürdigen Bootvorgang, der im Folgenden beschrieben wird.

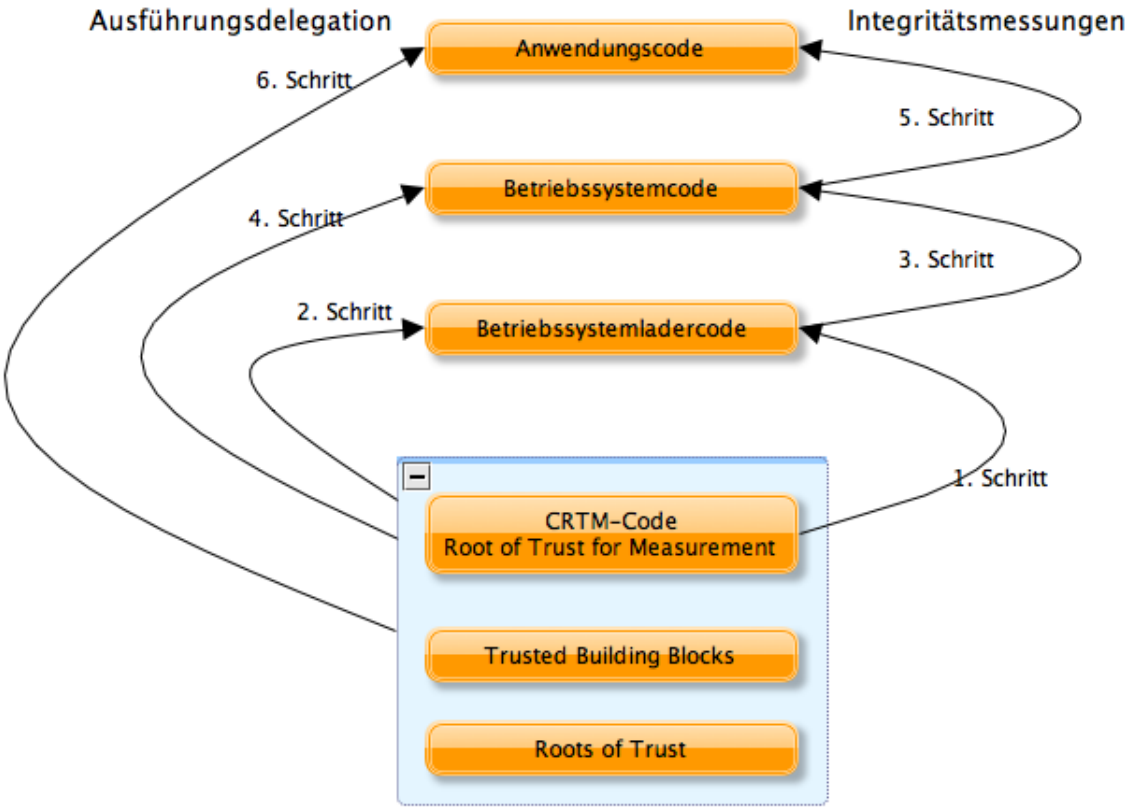

Abbildung 4.4: Transitives Vertrauen. Quelle: Angelehnt an [ea07b]

Ein klassischer Bootvorgang besteht meist aus folgenden Schritten, die nach dem Einschalten eines Computers durchgeführt werden:

1. Das BIOS führt diverse Aufgaben wie den Power On Self-Test (POST) durch. Anschließend sucht das BIOS nach Bootsektoren, im Falle von Festplatten nach dem Master Boot Record (MBR). Das MBR enthält einen Boot-Loader. Dieser wird vom BIOS ausgeführt.
2. Der Boot-Loader lädt den Betriebssystemkern und startet diesen.
3. Der Betriebssystemkern lädt Gerätetreiber und startet Dienste. Ist der Vorgang abgeschlossen, gilt das System als „hochgefahren“.

Nach diesen Schritten kann der Benutzer schließlich mit dem System interagieren. Die Frage, die sich im Kontext von Trusted Boot an dieser Stelle ergibt, ist: Woher weiß der Benutzer, dass er dem System nach dem Hochfahren vertrauen kann? Ein Angreifer könnte einen der oben genannten Schritte so modifiziert haben, dass er die Kontrolle über den Computer erlangt. Softwareseitig ist es zur Laufzeit unmöglich zu überprüfen,

ob das soeben gestartete System vertrauenswürdig ist oder nicht.

Die TCG adressiert dieses Problem durch die Herstellung der *Chain of Trust*, die sich über den gesamten Bootvorgang erstreckt: Das CRTM bildet den Vertrauensanker. Es ist das erste Glied in der Kette transitiven Vertrauens. Es misst das restliche BIOS und speichert den aus der Messung erhaltenen Messwert in einem PRC ab, bevor es die Kontrolle an das restliche BIOS übergibt. Das restliche BIOS führt nun weitere Messungen durch und misst die BIOSe eingebauter Steckkarten und anderer Hardware. Die Messwerte werden in PCRs abgelegt, anschließend wird die Kontrolle an die externen BIOSe übergeben. Nach dem das BIOS die Kontrolle zurückerlangt hat, wird der Boot-Loader gemessen. Der Messwert wird ebenfalls in einem PCR abgelegt, bevor die Kontrolle an den Boot-Loader übergeben wird. Dieser misst den Kernel, speichert den Messwert in einem PCR und übergibt anschließend die Kontrolle an den Kernel. Der Kernel misst nun Teile des Betriebssystems, speichert die Messwerte in PCRs ab und führt anschließend die gemessenen Programme aus.

Die Verkettung der Messungen aller Einzelkomponenten **verbunden** mit der Möglichkeit Hash-Werte lediglich mittels `extend` Operation in die PCRs schreiben zu können, bildet die Grundlage für die *Chain of Trust*. Wären die Messungen nicht lückenlos verkettet, könnte ein Angreifer ein nicht vertrauenswürdiges Programm in die Bootsequenz einbauen und somit seinen Code zum Laufen bringen. Könnten die PCR-Werte mittels einer `set` Operation o.Ä. gesetzt werden, könnte ein Angreifer einen manipulierten PCR-Wert setzen und somit einen gültigen Integritätszustand vortäuschen. Durch das Auslesen der im TPM abgelegten PCR-Werte kann nach einem Bootvorgang verlässlich überprüft werden, ob die Plattform in einem vertrauenswürdigen Zustand ist. Diese Fähigkeit, nach dem Bootvorgang eine verlässliche Aussage über die Plattformintegrität treffen zu können, bezeichnet man als *Trusted Boot*.

4.7.9 Secure Boot

Ein Secure Boot unterscheidet sich von einem Trusted Boot dadurch, dass das System ausschließlich in einen vertrauenswürdigen Zustand gebootet werden kann. Beim Booten wird analog zum Trusted Boot die Chain of Trust aufgebaut. Zusätzlich werden die Messwerte mit Referenzwerten verglichen. Im Falle einer Abweichung wird der Bootvorgang abgebrochen. Secure Boot adressiert zusammen mit Trusted Boot die Sicherheitsanforderung S2.

4.7.10 Trusted Software Stack

Das Trusted Software Stack (TSS) ist ein umfangreiches Softwarepaket, das sich in verschiedene Komponenten gliedert. Das TSS verfolgt laut Spezifikation [Cha06] folgende

Ziele:

- Einen definierten Eintrittspunkt zu TPM-Funktionen für Applikationen bereitstellen
- TPM-Zugriff synchronisieren
- Aufbau von Funktionsaufrufen inklusiver ihrer Interna (Byte-Reihenfolge etc.) verbergen
- Verwalten von TPM-Ressourcen

Hierbei soll vor allem die Unabhängigkeit von sowohl der Hardware-Plattform als auch vom Betriebssytem gewährleistet werden, um die Interoperabilität zu erhöhen.

Das TCG Software Stack (TSS) spezifiziert eine Standard-API zum Zugriff auf das TPM. Sie abstrahiert die konkreten Implementationsdetails, so dass Anwendungsentwickler sich nicht um die Details der konkreten TPM-Implementation kümmern müssen, sondern interoperabele Software entwickeln können. Weitere Details sind der Spezifikation [ea07a] zu entnehmen.

There are two kinds of cryptography in this world: cryptography that will stop your kid sister from reading your files, and cryptography that will stop major governments from reading your files.

Bruece Schneier (* 1963)

5 Marktüberblick

Die im Rahmen dieser Arbeit genannten Sicherheitsanforderungen sind nicht neu und teileweise schon durch Produkte adressiert. Der Bedarf an umfassenden FDE-Lösungen wächst beständig. Daher gibt es bereits einen Markt für solche Produkte. Der nachfolgende Marktüberblick hilft dabei, den aktuellen Markt einordnen und nach Kriterien wie Open-Source Software (OSS), kommerzielles Produkt und Betriebssystemplattform gruppieren zu können. Am Ende des Kapitels werden Boot-Umgebungen zur Integritätsmessung betrachtet. Es handelt sich dabei nicht um explizite FDE-Lösungen, sondern um Produkte, die die Sicherheitsanforderung S2 adressieren. Alle Produkte werden kurz beschrieben, Vor- und Nachteile sowie Schwerpunkte werden aufgezeigt.

5.1 Open-Source Produkte

Unter Open-Source Produkten werden Softwareprodukte verstanden, die der Open Source Definition der Open Source Initiative genügen.

5.1.1 Linux-spezifische Lösungen

Nachfolgend werden Lösungen aufgeführt, die ausschließlich unter GNU/Linux laufen und auf den Linux-Kernel zugeschnitten sind.

5.1.1.1 Cryptoloop

Cryptoloop ist ein Festplatten-Verschlüsselungs-Modul für Linux Kernel Version 2.6. Es ist Teil der Device Mapper Infrastruktur[1] und kann somit auf beliebige Gerätedateien angewendet werden. Zum Verschlüsseln werden die durch das Linux Crypto API bereitgestellten Algorithmen verwendet. Markku-Juhani Olavi Saarinen beschreibt in [Saa04b] und [Saa04a] einen Wasserzeichen-Angriff auf Cryptoloop. Markus Reichelt beschreibt daher auf seiner Webseite *Why Mainline Cryptoloop Should Not Be Used* [Rei04], warum Cryptoloop nicht mehr verwendet werden sollte. Cryptoloop adressiert die in Kapitel 3 formulierte Sicherheitsanforderung S1. Weitere Details zu Cryptoloop können dem Cryptoloop-HOWTO [Hö04] entnommen werden.

5.1.1.2 loop-AES

Loop-AES ist ein Kernelmodul für den Linux Kernel Version 2.6 und stellt die Verschlüsselung mittels AES auf einem Gerät (*device*) bereit. Es wird seit 2001 von Jari Ruusu entwickelt und weitergepflegt. Loop-AES adressiert die in Kapitel 3 formulierte Sicherheitsanforderung S1. Weitere Details können loop-AES.README [Ruu09] entnommen werden.

5.1.1.3 dm-crypt

Unter dem Betriebssystem Linux bietet das dm-crypt Subsystem die Möglichkeit, Datenträger transparent zu verschlüsseln. Das Subsystem ist seit der Kernel Version 2.6.4 Teil der Device Mapper Infrastruktur. Somit können Partitionen oder auch beliebige Gerätedateien verschlüsselt werden. Zum Verschlüsseln werden die durch das Linux Crypto API bereitgestellten Algorithmen verwendet. Dm-crypt adressiert die in Kapitel 3 formulierte Sicherheitsanforderung S1. Weitere Details können der Entwickler-Webseite[2] oder dem dm-crypt Wiki[3] entnommen werden.

5.1.1.4 LUKS

Linux Unified Key Setup (LUKS) ist ein von Clemens Fruhwirth definierter Standard zum Verschlüsseln von Festplatten unter Linux. LUKS ist somit kein Produkt zum Verschlüsseln von Festplatten, wird aber im Rahmen dieser Arbeit erwähnt, weil es sich als

[1] Device Mapper ist ein generisches Framework, um Blockgeräte aufeinander abzubilden.
[2] http://www.saout.de/misc/dm-crypt/
[3] http://www.saout.de/tikiwiki/tiki-index.php

Standard durchgesetzt hat, innerhalb der Linux-Distributionen verbreitet ist und im Zusammenspiel mit `dm-crypt` betrachtet werden sollte. LUKS erhöht die Interoperabilität unter Linux-Distributionen und erlaubt ein umfangreicheres Schlüssel-Management. Die Referenzimplementation von LUKS unter Linux erweitert `cryptsetup` und verwendet `dm-crypt` zur Festplattenverschlüsselung. Unter Windows können mittels LUKS verschlüsselte Festplatten mit Hilfe von FreeOTFE (s. Kapitel 5.1.3.1) verwendet werden. LUKS wurde an Hand von TKS1 (s. Kapitel 4.5.2) entworfen. Weitere Details können der Projekt-Webseite[4] entnommen werden.

5.1.2 BSD-spezifische Lösungen

Nachfolgend werden Lösungen aufgeführt, die innerhalb der BSD-Welt verwendet werden. Berkeley Software Distribution (BSD) basiert auf AT&Ts Unix und ist im Jahr 1977 entstanden. Die populärsten Vertreter der BSD-Familie sind heute FreeBSD, NetBSD und OpenBSD. Alle BSD-Unixe stehen im Gegensatz zu GNU/Linux unter der BSD-Lizenz, die kein Copyleft[5]-Prinzip kennt.

5.1.2.1 GBDE

GEOM Based Disk Encryption ist ein Block-orientierter Gerätetreiber für FreeBSD, der transparente Ver- und Entschlüsselung anbietet. Im Unterschied zu anderen Lösungen, die über die Betriebsmodi versuchen, Wasserzeichenangriffe zu erschweren oder zu unterbinden, geht GBDE andere Wege: In GBDE wird jeder Sektor mit einem eigenen zufälligen Schlüssel verschlüsselt. Wenn ein Angreifer zu einem beliebigen Zeitpunkt Zugriff auf ein Abbild der Festplatte hätte, könnte er keine Rückschlüsse auf die enthaltenen Daten ziehen, weil kein Schlüssel mehrfach verwendet wurde. GBDE verwendet AES zum Verschlüsseln der Daten und MD5 zum Erstellen von Schlüsselmaterial. GBDE adressiert die in Kapitel 3 formulierte Sicherheitsanforderung S1. Weitere Details lassen sich dem FreeBSD Handbook [Gre09] und Poul-Henning Kamps lesenswerter Veröffentlichung zu GBDE [Kam03] entnehmen. Des Weiteren schreibt Poul-Henning Kamps in seinem Papier unter Absatz 7.3:

> „This could be used to implement 'plausible denial' by embedding the GBDE partition inside some data which credibly can be claimed to be something else.“

und signalisiert damit, dass eine abstreitbare Verschlüsselung mittels GBDE implementiert und somit Sicherheitsanforderung S3 aus Kapitel 3 adressiert werden könnte.

[4] `http://code.google.com/p/cryptsetup/`
[5] Zu Copyleft siehe `http://www.gnu.org/copyleft/copyleft.de.html`.

5.1.2.2 GELI

In FreeBSD 6.0 wurde eine neue kryptographische GEOM[6]-Klasse namens GELI eingeführt. GELI unterscheidet sich von dem älteren GEOM und unterstützt das Crypto-Framework sowie verschiedene kryptographische Algorithmen wie AES, Blowfish, 3DES und Camellia. Außerdem erlaubt es die Verschlüsselung der Wurzel-Partition und den Einsatz mehrerer unterschiedlicher Schlüssel. GELI adressiert die in Kapitel 3 formulierte Sicherheitsanforderung S1. Weitere Details lassen sich dem FreeBSD Handbook [Gre09] entnehmen.

5.1.2.3 CGD

CryptoGraphic Disc ist ein Pseudo-Gerätetreiber für NetBSD. Es liegt über einem Gerätetreiber und bietet somit transparentes Ver- und Entschlüsseln an. CGD unterstützt AES-CBC, Blowfish-CBC und 3DES-CBC. CGD adressiert die in Kapitel 3 formulierte Sicherheitsanforderung S1. Weitere Details sind dem NetBSD Guide [ea08a] und Roland C. Dowdeswell und John Ioannidis Veröffentlichung zu CGD [DI03] zu entnehmen.

5.1.2.4 Vnd

OpenBSD Vnode disk driver (`vnd`) lässt eine Datei nach außen wie eine Festplatte erscheinen. Dies wird verwendet für Swap-Dateien oder um Abbilder für Disketten zu erstellen. Safe vnode disk driver (`svnd`) unterscheidet sich von `vnd` lediglich darin, dass `svnd` einen gepufferten Cache verwendet. Mittels `vnconfig` lässt sich ein `vnode` so konfigurieren, dass dieser zum Ver- und Entschlüsseln einer Festplatte verwendet werden kann. Hierbei wird zum Verschlüsseln Blowfish-CBC und zum Generieren von Schlüsselmaterial eine SALT-Datei verwendet. Vnd adressiert die in Kapitel 3 formulierte Sicherheitsanforderung S1. Weitere Details sind den Hilfeseiten zu `vnd` [ea08b] und `vnconfig` [Kra09] zu entnehmen.

5.1.2.5 Softraid CRYPTO

OpenBSD Softraid dient dazu ein RAID System in Software zu simulieren. Neben RAID 0/1/4/5 exisitert die Option CRYPTO, die Daten auf einem Datenblock verschlüsselt. Hierzu wird das Crypto API verwendet, wobei zum Verschlüsseln AES und als Betriebsmodus XTS verwendet werden. Softraid CRYPTO adressiert die in Kapitel 3 formulierte

[6]Für mehr Informationen zu GEOM siehe GEOM Man Page [Kam06].

Sicherheitsanforderung S1. Weitere Details sind der Hilfeseite zu `softraid` [Pee09] zu entnehmen.

5.1.2.6 OpenSolaris ZFS

OpenSolaris bietet in seinen Entwickler-Releases seit April 2008 Verschlüsselung für ZFS an [Mof09]. ZFS ist ein viel versprechendes Dateisystem der Firma SUN. Es adressiert zahlreiche Anforderungen an moderne Dateisysteme. ZFS wurde zum ersten Mal im Juni 2006 veröffentlicht und wird neben OpenSolaris von FreeBSD und Apple Mac OS X unterstützt. Sun plant nach [Mof07] die Implementation folgender Features:

- Gewährleistung der Integrität durch Fletcher[7] und SHA-256
- Verschlüsselung im Prototypen mittels AES-CBC
- Verschlüsselung in Produktion mittels CCM/GCM.

Laut dem Projektplan[8] für *ZFS Encrypted Datasets PSARC 2007/261* ist mit einer ersten Implementation in OpenSolaris im vierten Quartal 2009 zu rechnen.

5.1.3 Windows-spezifische Lösungen

Nachfolgend wird der Vollständigkeits halber eine Lösung aufgeführt, die ausschließlich unter Microsoft Windows läuft.

5.1.3.1 FreeOTFE

Free On the Fly Encryption (FreeOTFE) weist eine modulare Architektur auf, so dass Drittanbieter Algorithmen bei Bedarf entwickeln können. FreeOTFE unterstützt etwas mehr Verschlüsselungsverfahren und Hash-Funktionen als TrueCrypt. Zum Verschlüsseln werden zu den bei TrueCrypt verwendeten Verfahren AES, Serpent und Twofish zusätzlich Blowfish, CAST5, CAST6, DES, 3DES, MARS und RC6 unterstützt. Zu den bei TrueCrypt verwendeten Hash-Funktionen RIPEMD-160, SHA-512, Whirlpool werden zusätzlich MD2, MD4, MD5, RIPEMD-128, SHA-1, SHA-224, SHA-256, SHA-384 und Tiger unterstützt. An Betriebsmodi werden zu dem aus TrueCrypt bekannten XTS zusätzlich CBC und LRW unterstützt. Diese Vielfalt zeigt deutlich, dass der Schwerpunkt von FreeOTFE in der Unterstützung von möglichst vielen Verfahren liegt. Daher ist FreeOTFE ferner kompatibel zu unter Linux verwendeten verschlüsselten Volumes (LUKS, dm-crypt und cryptoloop). Zusätzlich bietet es einen „portable mode“ an, in

[7] Zu Fletcher siehe RFC 1146 Appendix I und II in [ZP90].
[8] `http://www.opensolaris.org/os/project/zfs-crypto/plan/`

welchem weder Software installiert noch Administratorrechte notwendig sind. Weitere Details können der Produkt-Webseite[9] entnommen werden.

5.2 Kostenfreie Produkte

Nachfolgend wird ein Produkt aufgeführt, welches zwar kostenfrei erhältlich ist und den Quellcode zur Verfügung stellt, dennoch nicht als Open-Souce Software gemäß OSI Open Source Definition bezeichnet werden kann und daher getrennt von den oben genannten Open-Source Produkten betrachtet werden muss.

5.2.1 Multiplattform-Lösungen

Nachfolgend wird ein Produkt, welches unter mehr als einem Betriebssystem läuft, aufgeführt.

5.2.1.1 TrueCrypt

TrueCrypt ermöglicht das Verschlüsseln von sogenannten Volumes sowie Partitionen und implementiert abstreitbare Verschlüsselung. TrueCrypt unterstützt folgende symmetrische Verschlüsselungsverfahren: AES, Serpent und Twofish. Die genannten Verschlüsselungsverfahren können wie folgt kaskadiert werden: AES-Twofish, AES-Twofish-Serpent, Serpent-AES, Serpent-Twofish-AES, Twofish-Serpent. TrueCrypt unterstützt folgende Hash-Funktionen: RIPEMD-160, SHA-512, Whirlpool. Als Betriebsmodus verwendet TrueCrypt XTS.

TrueCrypt speichert keine Metadaten in unverschlüsselter Form auf der Festplatte. Das Verschlüsselungsverfahren und die verwendete Hash-Funktion müssen daher mittels Brute-Force Verfahren ermittelt werden. Dies ist als Sicherheitsmerkmal bewusst so von den TrueCrypt-Entwicklern entworfen worden.

TrueCrypt implementiert einen Container-Ansatz. Ein Container, oder auch Volume genannt, dient als logischer Behälter für Daten. Diese können vom Betriebssystem eingebunden werden und ausgehangen werden. Unter Windows werden einzelne Container als Laufwerke geführt, unter Linux und Mac OS als Verzeichnisse eingebunden. Container können entweder als Dateien oder als Partitionen realisiert werden.

[9] http://www.freeotfe.org/

Ein TrueCrypt-Container wird initial bei der Erstellung mit Zufallszahlen hoher Entropie[10] beschrieben. Dies ist **essentiell**, um das Konzept der abstreitbaren Verschlüsselung zu implementieren. Ein Container kann, wie in Abbildung 5.1 dargestellt, einen weiteren versteckten Container enthalten. Von außen betrachtet ist es nicht möglich zu erkennen, ob der äußere Container einen weiteren Container enthält oder nicht, weil die initial geschriebenen Zufallszahlen sich nicht von verschüsselten Daten unterscheiden. Damit implementiert TrueCrypt das Prinzip der glaubhaften Abstreitbarkeit und adressiert somit die in Kapitel 3 formulierten Sicherheitsanforderungen S1 und S3.

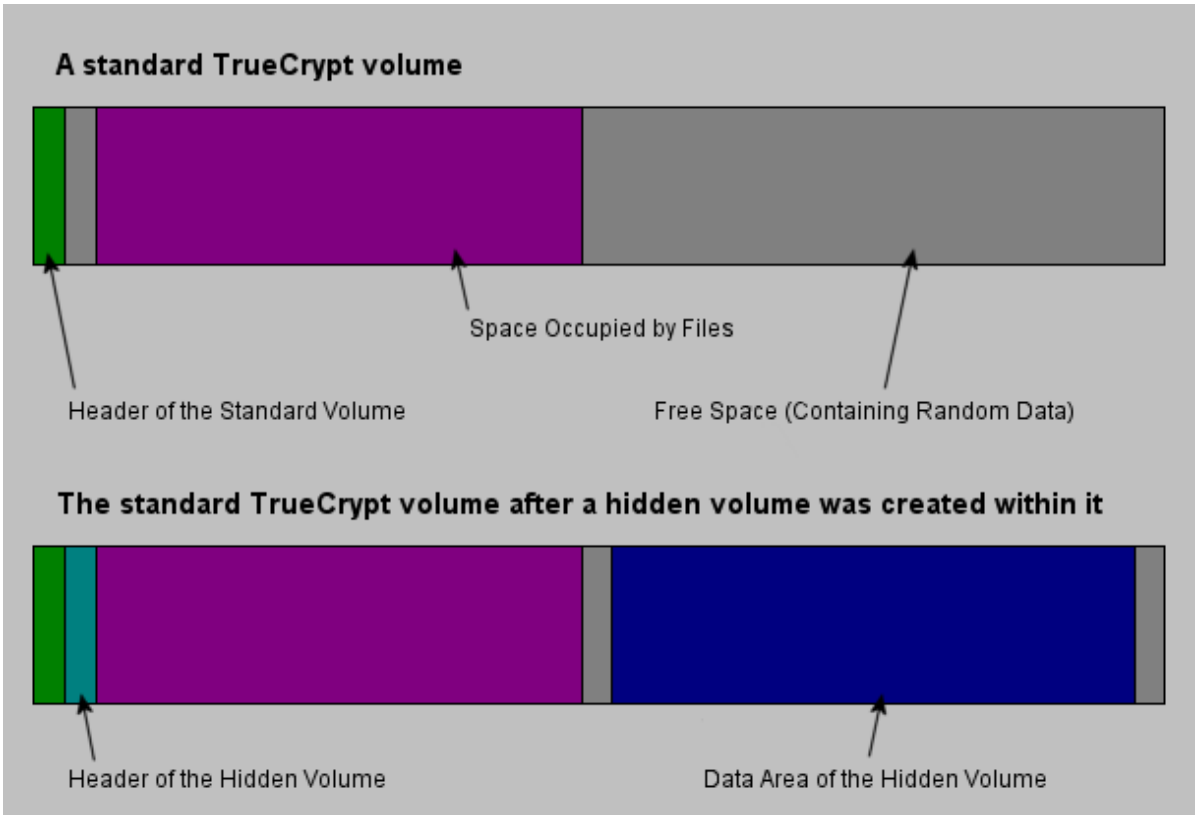

Abbildung 5.1: Verstecktes Volume innerhalb eines Standard-Volume. Quelle: [Fou09]

Das Schlüsselmaterial wird aus Zufallszahlen generiert. Das Benutzerkennwort wird lediglich verwendet, um das Schlüsselmaterial zu entschlüsseln. Das eigentliche Schlüssel-

[10] Unter Windows werden hierzu Daten aus verschiedenen Quellen einbezogen: Mausbewegungen während der Container-Erstellung, Tastatureingaben wie welche Tasten und Dauer zwischen den Eingaben, Leistungscharakteristik der Festplatte, Statistiken der Netzwerkschnittstelle, Windows CryptoAPI, periodische Stichproben offener Dateien und offener Handler etc. Unter MacOS und Linux wird der im Betriebssystem eingebaute Random Number Generator /dev/random respektive /dev/urandom verwendet. Anschließend werden die ermittelten Zufallszahlen durch eine *Pool Mixing Funktion* gereicht, um die Diffusion zu erhöhen. Diffusion bedeutet, dass jedes Klartextbit auf möglichst viele Bits des Chiffretextes verteilt wird.

material unterliegt somit nicht den Nachteilen, die durch ein schlecht gewähltes Benutzerkennwort entstehen (zu kurz, Wörterbuch-Attacken möglich etc.).

Die TrueCrypt-Lizenz erfüllt nicht die Open Source Definition der OSI und wird bspw. daher von den bedeutenden Linux-Distributionen (Debian [Qua06], Ubuntu [Fut07], Fedora [Cal08], openSUSE [Jae08] und Gentoo [Ber09]) als „nicht-frei“ betrachtet. Das MacMacken Blog führt unter [ea09b] mehrere Gründe gegen den Einsatz von TrueCrypt auf und thematisiert u.a. die Problematik um die OSS-Fragestellung.

Weitere Details können der TrueCrypt-Webseite[11] entnommen werden.

5.3 Kommerzielle Produkte

Um den Marktüberblick abzurunden, wird ein kurzer Blick auf ausgewählte kommerzielle Produkte geworfen.

5.3.1 BitLocker

BitLocker ist ein Produkt der Firma Microsoft zum Verschlüsseln von Laufwerken unter den Betriebssystemen Windows Vista Enterprise und Ultimate, Windows Server 2008 und Windows 7 Enterprise und Ultimate. BitLocker ermöglicht zusätzlich zum Verschlüsseln von Laufwerken die Integritätsprüfung beim Start des Betriebssystems. Hierbei kann ein vorhandenes TPM zum sicheren Ablegen von Schlüsselmaterial verwendet werden. Weitere Details lassen sich der Produktseite[12], dem BitLocker FAQ[13] und dem lesenswerten BitLocker-Leitfaden [SPT+08], der in Zusammenarbeit von BSI und dem Fraunhofer Institut für Sichere Informations-Technologie entstanden ist, entnehmen.

5.3.2 FileVault

FileVault ist Bestandteil von Mac OS X seit der Version 10.3. Es dient zum Verschlüsseln von Benutzerverzeichnissen, bietet also keine FDE-Lösung an. Zum Verschlüsseln wird AES-128 verwendet. Weitere Details zu FileVault lassen sich der Mac OS X 10.5 Hilfeseite zum Thema FileVault[14] entnehmen. Jacob Appelbaum und Ralf-Philipp Weinmann haben Ende 2006 auf dem 23. Chaos Communication Congress (23C3)[15] in ihrem

[11] http://www.truecrypt.org/
[12] http://www.microsoft.com/germany/windows/products/windowsvista/features/details/bitlocker.mspx
[13] http://technet.microsoft.com/de-de/library/cc766200.aspx
[14] http://docs.info.apple.com/article.html?path=Mac/10.5/en/8727.html
[15] http://events.ccc.de/congress/2006/

Vortrag[16] „Unlocking FileVault“ zahlreiche, teils gravierende Schwächen in FileVault aufgezeigt.

5.3.3 PGP Whole Disc Encryption

PGP Whole Disc Encryption der Firma PGP Corporation bietet eine FDE Lösung mit der Möglichkeit eine Smartcard- oder USB-Token-basierte Pre-Boot-Authentifizierung durchzuführen. Unterstützt werden diverse Windows Versionen und Mac OS X. Weitere Details sind der Produkt-Webseite[17] zu entnehmen.

5.3.4 Check Point Full Disc Encryption

Check Point Full Disc Encryption der Firma Check Point Software Technologies Ltd. ist eine FDE Lösung für die Betriebssysteme Windows, Mac OS X und Linux. Weitere Details sind der Produkt-Webseite[18] zu entnehmen.

5.4 Gesamtübersicht der Produkte sowie verwendeter Algorithmen und Betriebsmodi

Tabelle 5.1 gibt einen Überblick über die Produkte und verwendeten Verschlüsselungsverfahren. Es sind drei Produktgruppen deutlich zu erkennen:

1. Die größte Zahl der Produkte verwendet wenige ausgewählte Algorithmen (meist AES).
2. Die nächst größere Gruppe wird aus den Lösungen aus dem Linux-Umfeld gebildet. Das generische Linux Crypto API erlaubt das Einbinden diverser Algorithmen.
3. Schließlich fällt FreeOTFE durch die Unterstützung zahlreicher Algorithmen auf.

Tabelle 5.2 gibt einen Überblick über die Produkte und verwendeten Betriebsmodi.

[16] Informationen zum Vortrag sind http://events.ccc.de/congress/2006/Fahrplan/events/1642.en.html, den Vortrags-Folien [AW06] und dem Vortragsvideo unter http://chaosradio.ccc.de/23c3_m4v_1642.html zu entnehmen.
[17] http://www.pgp.com/de/products/wholediskencryption/
[18] http://www.checkpoint.com/products/datasecurity/pc/

5.5 Hardware-basierte Lösungen zur Integritätsmessung

Nachfolgend werden Hardware-basierte Lösungen zur Integritätsmessung betrachtet, wobei der Schwerpunkt auf Produkten aus dem Trusted Computing Umfeld liegt.

5.5.1 TrustedGRUB

TrustedGRUB erweitert GRUB[19] um die Fähigkeit einen vertrauenswürdigen Bootvorgang durchzuführen. Um die Funktionsweise von TrustedGRUB zu verstehen, sollte ein Basisverständnis von GRUB vorhanden sein. Eine Beschreibung von GRUB kann dem Anhang unter Kapitel 11.1 entnommen werden. TrustedGRUB setzt die Vertrauenskette, die durch das CRTM begonnen wurde, fort. Es erweitert Stage 1 so, dass es den ersten Sektor von Stage 2 prüft. Zusätzlich muss es alle Bestandteile, die Stage 2 verwendet, also Betriebssystemkern, GRUB-Module und Konfigurationsinformationen, ebenfalls messen. Weiterhin muss es Befehle, die ggf. vom Benutzer über die interaktive Shell eingegebenen werden, messen.

TrustedGRUB ist in der Lage zu ladende Komponenten vor dem Ladevorgang zu messen und die Messwerte in den PCRs des TPM abzulegen. Hierdurch kann die *Chain of Trust* auf einer mit einem TPM ausgestatteten Plattform auch während des Hochfahrens nahtlos sichergestellt werden.

Folgende PCRs werden belegt:

- PCR 4: MBR und stage1
- PCR 8: Boot-Loader stage2 Teil 1
- PCR 9: Boot-Loader stage2 Teil 2
- PCR 12: alle Kommandozeilenparameter aus menu.lst als auch die, welche über die interaktive GRUB-Shell eingegeben wurden
- PCR 13: alle Dateien, die mittels Checkfile überprüft wurden
- PCR 14: alle Dateien, die geladen wurden (bspw. Linux-Kernel, initrd, Module etc.)

Weitere Details sind der Projekt-Webseite[20] zu entnehmen.

[19] Grand Unified Bootloader
[20] http://sourceforge.net/projects/trustedgrub

5.5.2 GRUB-IMA

GRUB-IMA ist eine Trusted Computing-Erweiterung für GRUB. Es wurde von IBM für ihre Integrity Measurement Architecture (IMA) entwickelt. Die Hauptfunktionen von GRUB-IMA sind laut Projekt-Webseite[21]:

- Messung durchführen, während GRUB geladen wird.
- Stage 1 misst den ersten Sektor von stage 1.5 (oder stage 2). Stage 1 wird durch das BIOS gemessen.
- Der erste Sektor von stage 1.5 (respektive stage 2) misst die restlichen Sektoren. Stage 1.5 misst stage 2.
- GRUB misst nach dem Bootvorgang die Konfigurationsdatei `grub.conf`. Anschließend misst es weitere Dateien, die innerhalb der Konfigurationsdatei spezifiziert wurden.

5.5.3 OSLO

The Open Secure Loader (OSLO) ist ein Urladeprogramm, das SKINIT[22] für vertrauenswürdiges Booten verwendet. Das Changelog des Projektes [Kau07a] lässt allerdings vermuten, dass die Entwicklung im Jahr 2007 eingestellt wurde. Weitere Details können der Webseite zu OSLO[23] entnommen werden.

[21] http://sourceforge.jp/projects/openpts/wiki/GRUB-IMA

[22] AMDs Pacifica-Technologie mit Secure Virtual Machine Extensions SVM verfügt über Befehle zum Verwalten des Virtual Machine Control Block VMCB und zum Durchführen von SVM-Operationen. Hierzu zählt auch SKINIT, das den Prozessor für den Einsatz von Trusted Software vorbereitet. SKINIT entspricht grob dem, was bei Intel SENTER heisst.

[23] http://os.inf.tu-dresden.de/~kauer/oslo/

Produkt	AES	Blowfish	Twofish	Serpent	DES	3DES	CAST5	CAST6	MARS	RC6	Camellia
FreeOTFE	+	+	+	+	+	+	+	+	+	+	-
Cryptoloop	*	*	*	*	*	*	*	*	*	*	*
loop-AES	+	*	*	*	*	*	*	*	*	*	*
dm-crypt	*	*	*	*	*	*	*	*	*	*	*
GBDE	+	-	-	-	-	-	-	-	-	-	-
GELI	+	+	-	-	-	+	-	-	-	-	+
CGD	+	+	-	-	-	+	-	-	-	-	-
vnd	-	+	-	-	-	-	-	-	-	-	-
Softraid CRYPTO	+	-	-	-	-	-	-	-	-	-	-
OpenSolaris	+	-	-	-	-	-	-	-	-	-	-
TrueCrypt	+	-	+	+	-	-	-	-	-	-	-
FileVault	+	-	-	-	-	-	-	-	-	-	-
BitLocker	+	-	-	-	-	-	-	-	-	-	-
PGP WDE	+	-	-	-	-	-	-	-	-	-	-
Check Point FDE	+	-	-	-	-	-	-	-	-	-	-

Anmerkung: Die mit * gekennzeichneten Produkte verwenden das Linux Crypto-API und können somit alle vom Linux-Kernel angebotenen Algorithmen implementieren.

Tabelle 5.1: Produkt x Verschlüsselungsverfahren-Matrix

Produkt	ECB	CBC	Counter	CMC	EME	CCM	GCM	LRW	XEX	XTS
FreeOTFE	-	+	-	-	-	-	-	+	-	+
Cryptoloop	*	*	*	*	*	*	*	*	*	*
loop-AES	*	*	*	*	*	*	*	*	*	*
dm-crypt	*	*	*	*	*	*	*	*	*	*
GBDE	-	+	-	-	-	-	-	-	-	-
GELI	-	+	-	-	-	-	-	-	-	-
CGD	-	+	-	-	-	-	-	-	-	-
vnd	-	+	-	-	-	-	-	-	-	-
Softraid CRYPTO	-	-	-	-	-	-	-	-	-	+
OpenSolaris	-	+	-	-	-	**	**	-	-	-
TrueCrypt	-	-	-	-	-	-	-	-	-	+
FileVault	-	+	-	-	-	-	-	-	-	-
BitLocker	-	+	-	-	-	-	-	-	-	-
PGP WDE	-	+	-	-	-	-	-	-	-	-
Check Point FDE	?	?	?	?	?	?	?	?	?	?

Anmerkungen:

1. Die mit * gekennzeichneten Produkte verwenden das Linux Crypto-API und können somit alle vom Linux-Kernel angebotenen Betriebsmodi implementieren.
2. OpenSolaris enthält mit ** gekennzeichnete Betriesmodi. Diese werden laut [Mof07] in der bevorstehenden Produktionsversion implementiert werden.
3. Zu Check Point FDE lassen sich keine Aussagen tätigen, weil der Hersteller keine Angaben zu den Betriesmodi tätigt.

Tabelle 5.2: Produkt x Betriebsmodus-Matrix

Beware of bugs in the above code; I have only proved it correct, not tried it.

Donald Ervin Knuth (* 1938)

6 Analyse ausgewählter Lösungen

Aus den in Kapitel 5 genannten Produkten - mit Ausnahme von OpenSolaris ZFS crypto, weil es noch nicht vollständig fertiggestellt ist - wird eine Auswahl getroffen werden, die möglichst viele, idealerweise alle Sicherheitsanforderungen aus Kapitel 3 adressiert. Diese Produkte werden im Hinblick auf die konkrete Umsetzung der Sicherheitsanforderungen untersucht und wenn möglich, wird ein Produkt vorgeschlagen.

6.1 Adressierung der Sicherheitsanforderungen

Die in Kapitel 3 definierten Sicherheitsanforderungen werden im Hinblick auf die Umsetzung durch die Produkte untersucht.

6.1.1 S0: Open-Source

Alle in Kapitel 5.1 genannten Produkte sind Open-Source Produkte im Sinne der Open Source Definition der Open Source Initiative und erfüllen somit diese Anforderung. Zusätzlich wurde in Kapitel 3 gefordert, dass die Betriebssystemplattform unter der das Produkt läuft, ebenfalls ein OSS-Produkt ist. Damit fällt FreeOTFE aus der weiteren Betrachtung, weil es ausschließlich unter MS Windows läuft.

TrueCrypt nimmt wegen seiner nicht OSI-konformen Lizenz eine Sonderstellung ein. Es genügt, wie bereits in Kapitel 5.2.1.1 beschrieben, nicht der OSI Open-Source Definition.

Dennoch liegen die Quellen vor und können somit einer Sicherheitsanalyse unterzogen werden. TrueCrypt ist daher im Unterschied zu FreeOTFE in der weiteren Betrachtung enthalten.

6.1.2 S1: Gewährleistung der Vertraulichkeit

Die Gewährleistung der Vertraulichkeit wird durch *Verschlüsselung* adressiert. Festplattenverschlüsselungs-Produkte aus Kapitel 5 adressieren diese Sicherheitsanforderung unterschiedlich. Das Prinzip ist zwar bei allen Produkten das Gleiche, die verwendeten Algorithmen und Betriebsmodi sowie unterstützten Plattformen und Implementierungsdetails können jedoch stark variieren. Daher werden sie im Folgenden genauer betrachtet.

6.1.2.1 Architekturansatz

Architekturell lassen sich unterschiedliche Ansätze beobachten:

- TrueCrypt ist ein generisches Produkt, welches unter mehreren Betriebssystemplattformen läuft. Es bietet außerdem eine grafische Oberfläche zur vereinfachten Bedienung an. Die Anbindung an den Windows-Kernel geschieht über einen Gerätetreiber. Unter Linux und Mac OS wird FUSE[1] vewendet.
- Cryptoloop, loop-AES und dm-crypt sind Kernel-Module für den Linux-Kernel. GBDE, GELI, CGD, vnd und Softraid CRYPTO sind Gerätetreiber für die entsprechenden BSD-Kernels. Alle genannten Produkte sind betriebssystemabhängig und auf die unterliegende Plattform zugeschnitten. Alle Produkte sind architekturell auf Kernel-Ebene anzusiedeln und bieten daher den Vorteil, dass sie eingesetzt werden können, sobald der Kernel läuft. Es sind keine weiteren Dienste/Programme notwendig, um die Funktionsfähigkeit der Produkte zu gewährleisten.

Eine nahtlose Integration in das Betriebssystem ist zu begrüßen, weil damit die Architektur des Produkts schlank gehalten werden kann. Viele Aufgaben können an das Betriebssystem delegiert werden (Verschlüsselungsverfahren, Dateisystemzugriffe etc.). Dieses Argument spricht gegen TrueCrypt und für die anderen Produkte.

6.1.2.2 Sicherheit der Algorithmen

Die Sicherheit der Algorithmen kann nur durch die intensive Analyse dieser durch Kryptographie-Experten weltweit erfolgen. Je mehr Angriffe auf einen Algorithmus er-

[1] Filesystem in Userspace (FUSE) ist ein Kernel-Modul zum Auslagern von Dateisystemtreibern aus dem Kernel-Mode in den User-Mode. Dadurch können nicht-privilegierte Benutzer eigene Dateisysteme einbinden.

folglos durchgeführt werden, als desto sicherer gilt der Algorithmus mit der Zeit. Sobald erste erfolgreiche Angriffe gegen einen Algorithmus existieren, darf dieser nicht mehr als sicher eingestuft werden. Derzeit gelten Algorithmen wie AES, Blowfish, Camellia, CAST5, CAST6, MARS, RC6 und Twofish laut Aussage von NIST und anderen Institutionen als sicher. *Somit verwenden alle in der Marktübersicht genannten OSS-Produkte sichere Algorithmen.* Es lässt sich beobachten, dass die überwiegende Mehrheit der Produkte sich auf AES konzentiert, lediglich vnd verwendet Blowfish.

6.1.2.3 Sicherheit der Betriebsmodi

ECB ist, wie bereits in Kapitel 4.2.1 beschrieben, anfällig für statistische Analysen, weil keine Verkettung der Blöcke stattfindet.

CBC behebt dieses Problem durch die Verkettung der Blöcke. Nach [et.09] und Clemens Fruhwirths Ausführungen in [Fru04b] ist CBC anfällig für zahlreiche Angriffe:

- Wenn die IVs vorhersagbar sind, kann ein Angreifer den Einsatz des IV ad absurdum führen, in dem er die Klartextdaten so auswählt, dass die Verknüpfung mittels XOR Null ergibt. [et.09] empfiehlt daher beim Einsatz von CBC die Initialisierungsvektoren per Stromchiffre aus dem Schlüsselmaterial abzuleiten.
- Inhalte können durchsickern, vorausgesetzt der Angreifer findet zwei gleiche Geheimtextblöcke. Details zu dem Angriff sind [Fru04b] zu entnehmen.
- Es ist möglich, die Existenz bestimmter Daten innerhalb der verschlüsselten Daten nachzuweisen. Dieser Angriff wird als Wasserzeichenangriffe bezeichnet.
- Änderungen an Daten können durchsickern, wenn der Angreifer Zugriff auf die verschlüsselten Daten zu unterschiedlichen Zeitpunkten hatte und Änderungen aufzeichnen konnte. Details sind [Fru04b] zu entnehmen.
- Bis auf den ersten Block sind alle Klartextblöcke zu einem Gewissen grad formbar (engl.: malleable). Ein Angreifer kann bestimmte Bits im Klartext kippen, wenn er die entsprechenden Bits im Geheimtext des vorhergehenden Blocks kippt. Details sind [Fru04b] zu entnehmen.
- In CBC ist die Verschlüsselung eines Blocks nur von Geheimtextblock und vom Vorgänger abhängig. Diese können durch ein anderes vorhandenes Paar von Geheimtextblöcken ersetzt werden. Der erste Geheimtextblock wird nicht vorhersagbare Klartextdaten liefern. Der zweite Geheimtextblock wird als Klartextblock genau den gleichen Klartextblock aufweisen, den es ursprünglich hat. Dieser Angriff erlaubt es also Klartextblöcke an beliebige Stelle zu

kopieren und wird daher auch als Copy&Paste-Angriff bezeichnet.

CFB Nach [Ano09] ist CFB anfällig für Angriffe, wenn Initialisierungsvektoren mehfach verwendet werden. CFB und OFB werden, wie bereits in Kapitel 4.2.3 dargelegt, nicht in den untersuchten Produkten verwendet. Da sie eine Stromchiffre erzeugen, sind sie nicht nicht für den Einsatz mit blockorientierten Geräten geeignet.

CCM Gegen CCM existiert derzeit kein erfolgreicher Angriff. Jedoch formulieren Phillip Rogaway und David Wagner in [RW03] zahlreiche Kritikpunkte an CCM.

GCM Niels Ferguson zeigt in [Fer05] zwei Schwächen in der Authentisierung von GCM auf.

LRW Nach [Nas09] besitzt LRW bestimmte Schwächen, so dass es angegriffen werden kann. Daher wird seit 2006 empfohlen, es nicht mehr einzusetzen.

XEX ist unter bestimmten in [Min08] beschriebenen Umständen nicht mehr sicher.

XTS gilt als sicher und wurde daher im Dezember 2007 wie bereits in Kapitel 4.2.11 beschrieben, als IEEE P1619 standardisiert. Das NIST bat zwischen Juni und September 2008 um die Einreichung von Kommentaren zum Betriesmodus XTS-AES. Die Bedenken der Kryptoexperten sind in [SCS+08] zusammengefasst. XTS wird in TrueCrypt, dm-crypt und Softraid CRYPTO verwendet.

Zusammenfassend lässt sich festhalten: *Sichere Betriesmodi sind somit CCM und XTS. Somit kommen nur noch die Produkte TrueCrypt, dm-crypt und Softraid CRYPTO im Rahmen der weiteren Untersuchung in Betracht.*

6.1.2.4 Verwendete Schlüssellänge

Die verwendete Schlüssellänge trägt erheblich zur Sicherheit des eingesetzen Algorithmus bei. Mit zunehmender Schlüssellänge steigt der Umfang des Schlüsselraums (Menge aller möglichen Schlüssel) exponential an. Bei einem Brute-Force-Angriff werden alle Schlüssel des Schlüsselraums sequentiell ausprobiert. Je größer der Schlüsselraum ist, desto größer ist der Aufwand für einen Brute-Force-Angriff. Ziel ist es, die Schlüssellänge so groß zu wählen, dass die Wahrscheinlichkeit den Schlüssel mit akzeptablem Rechenaufwand (d.h. Zeit- und Kostenaufwand) zu finden, gegen Null geht.

Sowohl TrueCrypt, dm-crypt als auch Softraid CRYPTO erlauben den Einsatz ausreichend langer Schlüssel.

6.1.2.5 Implementierung

Die Implementierung trägt maßgeblich zur Sicherheit des eingesetzen Systems bei. Hierbei spielen u.a. folgende Kriterien eine Rolle, die nachfolgend pro Produkt kurz betrachtet werden.

Quellen Wie umfangreich sind die Quellen? Je umfangreicher ein Projekt, desto größer die Anzahl von Fehlern und somit auch sicherheitskritischen Fehlern. Welchen Eindruck erwecken die Quellen?[2]

Historie Wie ist die Produkthistorie? Ein Produkt, das im Laufe der Zeit durch zahlreiche Sicherheitslücken aufgefallen ist, wird nicht das Vertrauen der Benutzer stärken.

Komplexität „[..] complexity is the worst enemy of security." [Sch00] Die zunehmende Komplexität eines System erhöht die Wahrscheinlichkeit für Fehler und reduziert gleichzeitig die Verständlichkeit des Systems. Beides führt zu vermehrten Sicherheitslücken.

Kontext Meist werden Nachrichten kompromittiert, nicht indem die Verschlüsselung gebrochen wird, sondern bspw. weil nach der Verschlüsselung auf dem Quell- oder Zielsystem sensitive Daten weiterhin vorhanden sind oder bspw. weil das System bereits vor der Verschlüsselung kompromittiert wurde.

Bedienbarkeit Ein System, welches sich durch eine unnötig komplizierte Bedienung oder mangelnde Leistungsfähigkeit aufweist, ist nicht einsetzbar.

Das BSI nennt in [fSidI] weitere zu berücksichtigende Kriterien: Verlässliches Schlüsselmanagement, Fehlbedienungs- und Fehlfunktionssicherheit, Vorkehrungen gegen sicherheitsmindernde Manipulationen und Abwesenheit verdeckter kompromittierender Kanäle. Eine detaillierte Betrachtung dieser Kriterien entfällt auf Grund ihres Umfangs.

6.1.2.5.1 Cryptoloop

Quellen Cryptoloop liegt in den Quellen des Linux Kernels 2.6.31 unter `/drivers/block/cryptoloop.c` und ist 5005 Byte groß. Der Quellcode wirkt sehr schlank und aufgeräumt.

Historie Cryptoloop weist eine längere Historie im Linux-Kernel auf.

Komplexität Die Komplexität ist reduziert auf den Anwendungsfall eine Gerätedatei zu verschlüsseln und zu entschlüsseln.

[2] Hierbei kann es sich nur um einen ersten Eindruck handeln. Eine vollständige Quellcode-Analyse aller Produkte würde den Rahmen dieser Arbeit sprengen.

Kontext Als Kernel-Modul ist cryptoloop in die Linux-Kernel-Module Architektur eingebettet.

Bedienbarkeit Cryptoloop wird über die Kernel-Konfiguration und Kommandozeilenwerkzeuge gesteuert.

6.1.2.5.2 loop-AES

Quellen loop-AES Quellen sind 1,6 MB groß und erweitern einen vorhandenen Kernel um die loop-AES-Funktionen. AES wird in loop.c eingebaut. Konzeptionell wäre eine Trennung der Verschlüsselungsverfahren von der restlichen Funtionalität wünschenswert.

Historie loop-AES weist eine lange Historie von Änderungen seit 2001 auf.

Komplexität loop-AES weist eine ähnliche Komplexität wie Cryptoloop auf.

Kontext Als Kernel-Modul ist cryptoloop in die Linux-Kernel-Module Architektur eingebettet.

Bedienbarkeit loop-AES wird über die Linux Kernel-Konfiguration und Kommandozeilenwerkzeuge gesteuert.

6.1.2.5.3 dm-crypt

Quellen dm-crypt liegt in den Quellen des Linux Kernels 2.6.31 unter `/drivers/md/dm-crypt.c` und ist 32259 Byte groß. Initialisierungsvektoren werden in dm-crypt.c erstellt, die Verschlüsselung hingegen ist ausgelagert.

Historie dm-crypt ist im Gegensatz zu Cryptoloop und loop-AES relativ jung. Es ist im Linux-Kernel 2.6.4 hinzugefügt worden.

Komplexität dm-crypt ist in die Device Mapper Infrastruktur eingebunden und daher Teil eines komplexeren Gesamtsystems.

Kontext Als Kernel-Modul ist dm-crypt in die Linux-Kernel-Module Architektur eingebettet.

Bedienbarkeit dm-crypt wird über die Linux Kernel-Konfiguration und Kommandozeilenwerkzeuge gesteuert.

6.1.2.5.4 GBDE

Quellen GBDE liegt in den Quellen von FreeBSD 7.2 unter `/src/sbin/gbde/gbde.c` und ist 21943 Byte groß. Der Quellcode ist lesbar und rudimentär kommentiert.

Historie GBDE wurde im Oktober 2002 im CVS-Repository von FreeBSD angelegt. Die letzte Änderung ist auf Februar 2006 datiert. GBDE wird seit dem nicht weiterentwickelt.

Komplexität Die Komplexität beschränkt sich auf den Anwendungsfall eine Gerätedatei zu verschlüsseln und zu entschlüsseln als auch die Steuerung der Benutzerinteraktion.

Kontext GBDE ist in das GEOM-Framework und als Kommandozeilenwerkzeug in das Betriebssystem eingebunden.

Bedienbarkeit GBDE wird über die Kommandozeile gesteuert.

6.1.2.5.5 GELI

Quellen GELI liegt in den Quellen von FreeBSD 7.2 unter `/src/sbin/geom/class/eli/geom_eli.c` und ist 31291 Byte groß. Der Quellcode ist lesbar und rudimentär kommentiert.

Historie GELI wurde im Juli 2005 im CVS-Repository von FreeBSD angelegt. Die letzte Änderung ist vom August 2008. In der Zwischenzeit ist GELI laufend erweitert worden.

Komplexität Verschlüsselungsalgorithmen u.Ä. sind ausgelagert.

Kontext GELI ist in das GEOM-Framework und als Kommandozeilenwerkzeug in das Betriebssystem eingebunden.

Bedienbarkeit GELI wird über die Kommandozeile gesteuert.

6.1.2.5.6 CGD

Quellen CGD liegt in den Quellen von NetBSD 5.0.1 unter `/src/sys/dev/cgd.c` und ist 20657 Byte groß. Die Verschlüsselungsalgorithmen sind in `/src/sys/dev/cgd_crypto.c` und sollen später in ein generisches Kryptographie-Framework ausgelagert werden. Der Quellcode ist lesbar und kommentiert.

Historie CGD wurde im Oktober 2002 im CVS-Repository von NetBSD angelegt. Die letzte Änderung ist vom September 2009. In der Zwischenzeit sind laufend Anpassungen vorgenommen worden.

Komplexität Die Verschlüsselungsalgorithmen sind zwar ausgelagert, aber noch nicht in ein eigenes Crypto-Framework.

Kontext CGD ist als Kommandozeilenwerkzeug in NetBSD als Betriebssystem eingebunden.

Bedienbarkeit CGD wird über die Kommandozeile gesteuert.

6.1.2.5.7 vnd

Quellen vnd liegt in den Quellen von OpenBSD 4.5 unter `/src/sys/dev/vnd.c` und ist 27322 Byte groß. Der Verschlüsselungsalgorithmus ist im Crypto-Framework ausgelagert. Der Quellcode ist in typischer OpenBSD-Manier aufgeräumt, lesbar und kommentiert.

Historie vnd wurde im Oktober 1995 im CVS-Repository von OpenBSD angelegt. Die letzte Änderung ist vom September 2009. In der Zwischenzeit sind zahlreiche Änderungen vorgenommen worden.

Komplexität Die Funktion zur Verschlüsselung innerhalb `vnd.c` besteht aus einem Dutzend Zeilen und ist somit äußerst schlank gehalten.

Kontext vnd ist Bestandteil des Kernsystems von OpenBSD.

Bedienbarkeit vnd wird über ein Kommandozeilenwerkzeug gesteuert.

6.1.2.5.8 Softraid CRYPTO

Quellen Softraid CRYPTO liegt in den Quellen von OpenBSD 4.5 unter `/src/sys/dev/softraid_crypto.c` und ist 19281 Byte groß.

Historie Softraid CRYPTO wurde im November 2007 in dem CVS-Repository von OpenBSD angelegt. Die letzte Änderung ist vom August 2009. In der Zwischenzeit sind einige Änderungen vorgenommen worden.

Komplexität Die Komplexität beschränkt sich auf den Anwendungsfall, eine Gerätedatei zu verschlüsseln und zu entschlüsseln.

Kontext Softraid CRYPTO ist Bestandteil des Software RAID Systems von OpenBSD.

Bedienbarkeit Softraid CRYPTO wird über Kommandozeilenwerkzeuge gesteuert.

6.1.2.5.9 TrueCrypt

Quellen TrueCrypt 6.2a Quellen für Linux und Mac sind 7,4 MB groß, die Quellen für die Windows-Version sind 4,8 MB groß. Der Quellcode ist in Form von zahlreichen Ordnern und Dateien strukturiert. Der Quellcode ist in C++, C und Assembler geschrieben, ist lesbar, aber kaum kommentiert. Dies liegt vermutlich daran, dass der Quellcode lediglich von den beiden Hauptentwicklern und nicht von einer größeren Community gepflegt wird und daher logischerweise für die Hauptentwickler bekannt ist.

Historie Version 1.0 wurde im Februar 2004 veröffentlicht. Die aktuelle Version 6.2a ist vom Juni 2009.

Komplexität TrueCrypt ist aus den folgenden Gründen im Vergleich zu den oben genannten Produkten komplexer:

1. TrueCrypt ist nicht in ein Betriebssystem integriert, sondern plattformunabhängig und muss somit mehr Funktionalität selbst implementieren.
2. TrueCrypt bietet eine vollständige grafische Benutzeroberfläche an.
3. TrueCrypt verwendet FUSE. Das ermöglicht den Einsatz auf diversen Betriebssystemen, ohne sich um die Low-Level-Details der Dateisysteme kümmern zu müssen, steigert aber die Komplexität, weil eine weitere Schicht beim Dateisystemzugriff ins Spiel kommt.

Kontext TrueCrypt ist ein eigenes abgeschlossenes System. In der Ausführung verlässt es sich auf die korrekte Funktionsfähigkeit des darunterliegenden Betriebssystems.

Bedienbarkeit TrueCrypt lässt sich auf Grund seiner grafischen Benutzeroberfläche leicht bedienen. Die Steuerung per Kommandozeile ist ebenfalls möglich. Die Leistungsfähigkeit ist auf Grund von FUSE nicht so hoch wie bei den anderen Produkten, weil Daten zwischen Kernelspace und Userspace kopiert werden müssen.

6.1.2.5.10 Zusammenfassung

- Unter Linux sollte dm-crypt verwendet werden an Stelle von Cryptoloop oder loop-AES. Dm-crypt ermöglicht als Kernelmodul außerdem die Verschlüsselung des Wurzelverzeichnisses. Muss das Wurzelverzeichnis nicht verschlüsselt werden, kann unter Linux TrueCrypt verwendet werden.
- Unter FreeBSD sollte GELI verwendet werden. GBDE wird nicht mehr aktiv weiterentwickelt.
- Unter NetBSD kann lediglich CGD verwendet werden.
- Unter OpenBSD kann vnd als auch Softraid CRYPTO verwendet werden.

6.1.2.6 Zusammenfassung

Da GELI, CGD und vnd wegen der Verwendung unsicherer Betriebsmodi bereits in Kapitel 6.1.2.3 aus der Betrachtung gefallen sind, bleiben TrueCrypt, dm-crypt und Softraid CRYPTO als potentielle Kandidaten zur Gewährleistung der Vertraulichkeit übrig.

6.1.3 S2: Überprüfbarkeit der Integrität

Keines der untersuchten Produkte zur Verschlüsselung der Festplatte stellt Mechanismen zur Überprüfung der Integrität bereit. Der naheliegendste Ansatz zur Überprüfung der Integrität eines Systems ist die Verwendung von Hash-Werten zur Vermessung dieses Systems. Hierbei lassen sich die Ansätze in Software-basierte und Hardware-basierte unterteilen. Im Folgenden werden die in Kapitel 5.5 genannten Hardware-basierten Ansätze TrustedGRUB, GRUB-IMA und OSLO betrachtet.

6.1.3.1 Architekturansatz

Architekturell setzen TrustedGRUB und GRUB-IMA auf die CRTM. OSLO setzt auf die AMD-eigene Erweiterung SKINIT und DRTM. Die Vorteile von OSLO gegenüber TrustedGRUB und GRUB-IMA sind in [Kau07b] dargelegt.

6.1.3.2 Implementierung

In Hinblick auf die Implementierung werden die gleichen Kriterien wie in Kapitel 6.1.2.5 verwendet.

6.1.3.2.1 TrustedGRUB

Quellen TrustedGRUB 1.1.3 Quellen sind 4,8 MB groß.

Historie TrustedGRUB hat seit Version 0.1 bis Version 1.1.3 zahlreiche Änderungen erlebt.

Komplexität Die Komplexität beschränkt sich auf Messungen während der Bootsequenz.

Kontext TrustedGRUB erweitert GRUB und ist somit in die GRUB-Architektur eingebettet.

Bedienbarkeit TrustedGRUB muss für aktuelle Distributionen per Hand angepasst und eingerichtet werden.

6.1.3.2.2 GRUB-IMA

Quellen GRUB-IMA 0.97-38 Patch für Fedora 10 ist 116 KB groß.

Historie Die Produkthistorie ist nicht bekannt.

Komplexität Die Komplexität beschränkt sich auf Messungen während der Bootsequenz.

Kontext GRUB-IMA erweitert GRUB und ist somit in die GRUB-Architektur eingebettet.

Bedienbarkeit GRUB-IMA muss für aktuelle Distributionen per Hand angepasst und eingerichtet werden.

6.1.3.2.3 OSLO

Quellen OSLO 0.4.5 Quellen sind 164 KB groß. (OSLO wirbt damit, dass die Quellen gerade mal aus ca. 1000 Zeilen bestehen und die Binärdatei 4 KB groß ist.)

Historie OSLO 0.2 wurde im März 2006 veröffentlicht, die aktuelle Version 0.4.5 ist vom Juni 2007.

Komplexität Die Komplexität beschränkt sich ähnlich zu TrustedGRUB und GRUB-IMA auf Messungen während der Bootsequenz.

Kontext OSLO setzt eine AMD CPU voraus.

Bedienbarkeit OSLO wird per Hand eingerichtet und über die Kommandozeile gesteuert.

6.1.4 S3: Glaubhafte Abstreitbarkeit

Von allen in Kapitel 5 untersuchten Produkten implementiert lediglich TrueCrypt das Konzept der abstreitbaren Verschlüsselung und adressiert somit Sicherheitsanforderung S3.

6.2 Zusammenfassung

Zusammenfassend lässt sich für die Sicherheitsanforderungen festhalten:

S0: Open-Source wird von allen in Kapitel 5.1 genannten Produkten adressiert. TrueCrypt nimmt hierbei eine Sonderstellung ein.

S1: Gewährleistung der Vertraulichkeit wird von dm-crypt und Softraid CRYPTO adressiert.

S2: Überprüfbarkeit der Integrität wird von TrustedGRUB, GRUB-IMA und OSLO adressiert.

S3: Glaubhafte Abstreitbarkeit wird lediglich von TrueCrypt erfüllt.

Es ist schnell ersichtlich, dass derzeit keine OSS-Lösung am Markt existiert, die allen Sicherheitsanforderungen S1, S2 und S3 gerecht wird. Eine eigene Lösung bzw. die Zusammenstellung ausgewählter Produkte ist somit notwendig, um alle Sicherheitsanforderungen zu bedienen.

Nun kann man sich fragen: Warum gibt es noch kein OSS-Produkt, welches alle Sicherheitsanforderungen adressiert? Mögliche Antworten sind:

1. Trusted Computing ist noch relativ jung - die OSS-Szene möchte erst beobachten, wie sich die Technologie in kommerziellen Betriebssystemen behaupten wird.
2. Dem Trusted Computing haftet innerhalb der OSS-Szene weiterhin der Ruf des „DRM enabler“ an. Daher werden Technologien aus dem Trusted Computing Umfeld gemieden.
3. Der Einsatz von abstreitbarer Verschlüsselung ist umstritten, weil bereits der Einsatz ein Indiz für die Existenz vertraulicher Daten darstellt.

If you think you can solve your security problems, then you don't understand the problems and you don't understand the technology.

Bruce Schneier (* 1963)

7 Konzipierung eines Gesamtsystems

In diesem Kapitel wird ein Gesamtsystem konzipiert, welches die in Kapitel 3 gestellten Sicherheitsanfordernungen erfüllt. Hierzu werden Produkte aus Kapitel 5 optimal zusammengestellt, verbleibende Lücken aufgezeigt und Lösungsansätze dargelegt.

7.1 Adressierung der Sicherheitsanforderungen

Das Gesamtkonzept adressiert die Sicherheitsanforderungen aus Kapitel 3 wie folgt:

Gewährleistung der Vertraulichkeit wird adressiert durch *Verschlüsselung*. In Frage kommende Produkte sind TrueCrypt, dm-crypt und Softraid CRYPTO.

Überprüfung der Integrität wird adressiert durch die *Bildung von Prüfsummen bzw. Hash-Werten*, die entweder über die gesamte Festplatte oder sequentiell über Teile des Systems (erst Boot-Loader, dann Kernel, dann Rest) gebildet werden. Die ermittelten Hash-Werte werden mit vorhandenen korrekten Referenzwerten verglichen. Anschließend können Aussagen über die Integrität des Systems getroffen werden. Die Bildung von Hash-Werten kann durch selbsterstellte Skripte oder unter Verwendung der genannten Hardware-basierten Lösungen aus dem Trusted Computing Umfeld erfolgen.

Glaubhafte Abstreitbarkeit wird adressiert durch das Konzept der *abstreitbaren Verschlüsselung*. Als Produkt kommt nur TrueCrypt in Frage. Alle anderen Produkte

implementieren keine abstreitbare Verschlüsselung.

Im Folgenden wird die Adressierung der Sicherheitsanforderungen im Detail ausgeführt.

7.1.1 Gewährleistung der Vertraulichkeit

Die Gewährleistung der Vertraulichkeit wird durch die *Verschlüsselung* erreicht. Hierbei gibt es unterschiedliche Vorgehensweisen, deren Vor- und Nachteile kurz beleuchtet werden: Die Sicherheitsanforderung zur Gewährleistung der Vertraulichkeit ist optimal erfüllt, wenn die *gesamte Festplatte* mittels *bewährter Algorithmen* verschlüsselt ist. Das Verschlüsseln der gesamten Festplatte wird als Full Disc Encryption (FDE) bezeichnet und ist deshalb sinnvoll, weil dadurch keine Informationen Preis gegeben werden. Werden nur einzelne Teile der Festplatte verschlüsselt, dann bieten die unverschlüsselten Bereiche einen Angriffsvektor für einen potentiellen Angreifer an: Das Ausspähen ggf. vertraulicher Daten ist möglich. Das System kann modifiziert werden bspw. durch die Installation eines Rootkits. Grundsätzlich sollten bewährte Algorithmen eingesetzt werden, weil die Geschichte der Kryptographie immer wieder zeigt, dass der Einsatz von nicht bewährten oder sogar proprietären Algorithmen zum Verlust der Vertraulichkeit führt. Proprietäre Algorithmen werden meist innerhalb kürzester Zeit gebrochen. Die Sicherheit eines Verschlüsselungsverfahrens beruht auf der Geheimhaltung des Schlüssels, nicht des Algorithmus. Dieses Prinzip wird auch als *Kerckhoffs' Prinzip* bezeichnet.

Beim Verschlüsseln der Festplatte werden von den in Kapitel 5 genannten Produkten und den damit verbundenen Betriebssystemplattformen unterschiedliche Vorgehensweisen verfolgt:

Gesamte Festplatte verschlüsseln Die gesamte Festplatte wird verschlüsselt. Wenn ein bewährter Algorithmus verwendet wird, kann ein Angreifer keine sinnvollen Aussagen über die gespeicherten Daten tätigen. Diese weisen die gleiche Entropie wie Zufallszahlen auf und bieten somit keinen Informationsgehalt. Die Daten auf der Festplatte enthalten für den Angreifer keine erkennbare Struktur. Einzig die Größe der verschlüsselten Daten ist nach außen hin sichtbar. Um diesen Ansatz zu implementieren, muss von einem externen Medium (bspw. USB-Stick) gebootet werden. Das Booten von der Festplatte ist nicht möglich, weil keine unverschlüsselten Daten und somit auch kein Boot-Loader vorliegen kann.

Gesamte Festplatte bis auf Boot-Loader verschlüsseln Die gesamte Festplatte wird bis auf den Boot-Loader verschlüsselt. Das Vorgehen wird auch als Pre-Boot Authentication genannt. Vor dem eigentlichen Bootvorgang wird eine Authentifizierung durchgeführt. Im Open-Source-Umfeld implementiert lediglich TrueCrypt dieses Vorgehen, allerdings nur unter Microsoft Windows. Im kommerziellen Umfeld wird dieser Ansatz von zahlreichen Produkten verfolgt. Einem Angreifer ist lediglich ersichtlich, dass ein FDE-Produkt eingesetzt wird sowie um welches Produkt es

sich handelt. Zusätzlich kann ein Angreifer die Pre-Boot-Umgebung angreifen und bspw. so modifizieren, dass ein Zugriff auf Schlüsselmaterial nach einer Authentifizierung durch einen legitimen Benutzer möglich wird und somit anschließend das Gesamtsystem kompromittieren.

Gesamte Festplatte bis auf Boot-Loader und Kernel verschlüsseln Hierbei wird die gesamte Festplatte bis auf den Boot-Loader und den Kernel sowie ggf. weitere zum Booten benötigte Komponenten, wie `initrd` unter Linux, verschlüsselt. Der Kernel ist nach seinem Start in der Lage, den Rest der Festplatte zu entschlüsseln. Diese Vorgehensweise ist derzeit der de facto Standard innerhalb diverser Linux-Distributionen. Einem Angreifer werden mit diesem Vorgehen relativ viele Informationen und somit auch mehr Angriffsvektoren gegeben. Eine Modifikation des Kernels inklusive der Installation eines Rootkits ist möglich. Damit kann das Gesamtsystem nach einer Authentifizierung durch einen legitimen Benutzer kompromittiert werden.

Bestimmte Partitionen verschlüsseln Hierbei wird die Verschlüsselung der Festplatte auf verschiedene Partitionen beschränkt, bspw. auf die Benutzerverzeichnisse. Dieses Vorgehen wird bspw. von Apples FileVault implementiert und innerhalb diverser Linux-Distributionen während der Installation als Option angeboten. Einem Angreifer sind alle nicht verschlüsselten Daten wie Programme, Bibliotheken, Logdateien, Konfigurationsdateien etc., zugänglich. Die Gewährleistung der Vertraulichkeit ist folglich geringer als in den bereits genannten Vorgehensweisen. Zusätzlich kann ein Angreifer das gesamte Betriebssystem modifizieren und somit das Gesamtsystem kompromittieren.

7.1.2 Überprüfung der Integrität

Die Überprüfung der Integrität wird durch die Bildung von Hash-Werten adressiert. Sie werden entweder sequentiell über Bestandteile des Gesamtsystems (Boot-Loader, Kernel etc.), einzelne Partitionen oder über die gesamte Festplatte gebildet. Anschließend können sie mit Referenzwerten verglichen werden, um Aussagen über die Integrität des Systems treffen zu können.

Unter Verwendung eines TPM kann ein Trusted Boot realisiert werden und damit die Integrität des Systems hardware-gestützt überprüft werden. Hierbei wird eine Chain of Trust ab dem CRTM bis zum Anwendungsprogramm aufgebaut. Diese lückenlose Vertrauenskette und die Möglichkeit Integritätsmesswerte im TPM so hinterlegen zu können, dass sie nicht modifiziert werden können, ermöglicht es, verlässliche Aussagen über die Integrität des laufenden Systems zu treffen.

Alternativ kann nur die Verwendung externer vertrauenswürdiger Bootmedien, bspw. ein USB-Stick, den man immer mit sich trägt, die Anforderung zur Überprüfung der

Integrität gewährleisten. Denn ein Bootvorgang innerhalb einer kompromittierten Bootumgebung führt unweigerlich dazu, dass das laufende System als kompromittiert und somit nicht vertrauenswürdig betrachtet werden kann. Die Chain of Trust ist gebrochen. *Die Überprüfung der Integrität muss also durch eine vertrauenswürdige Umgebung erfolgen.*

Ziel ist es, dass System so aufzusetzen, dass eine Veränderung durch Dritte in jedem Fall ersichtlich wird. Dies kann durch den Einsatz bewährter Verschlüsselungsverfahren/-modi und vertrauenswürdiger Boot-Umgebungen erreicht werden. Werden verschlüsselte Datenbereiche modifiziert, dann führt das zur Korruption der verschlüsselten Daten. Nicht verschlüsselte Datenbereiche können durch vertrauenswürdige Boot-Umgebungen vermessen werden. Somit sind auch Aussagen über die Integrität verschlüsselter und nicht verschlüsselter Datenbereiche möglich.

7.1.3 Glaubhafte Abstreitbarkeit

Die glaubhafte Abstreitbarkeit wird mittels *abstreitbarer Verschlüsselung* implementiert. In der Umsetzung gibt es zwei Ansätze:

1. Die gesamte Festplatte wird verschlüsselt. Von außen betrachtet enthält die Festplatte statistisches Rauschen bestehend aus Zufallszahlen. Hierbei muss die Verschlüsselung so implementiert werden, dass keine Dateisystem-Header oder andere Metainformationen ersichtlich sind.

2. Die Festplatte enthält ganz offensichtlich ein verschlüsseltes System, erkenntlich bspw. an der TrueCrypt-Passworteingabeaufforderung. Es ist jedoch nach außen hin nicht ersichtlich, ob weitere Systeme/Container innerhalb der verschlüsselten Daten vorliegen.

Beide Ansätze verlangen den Einsatz von Verschlüsselung. Verschlüsselung ist somit eine notwendige Bedingung für abstreitbare Verschlüsselung.

Alternativ könnte man argumentieren, dass man glaubhafte Abstreitbarkeit über Steganographie realisieren kann. Die Kryptographie und Steganographie verfolgen jedoch unterschiedliche Ziele. Das Ziel der Kryptographie ist die Geheimhaltung. Das Ziel der Steganographie ist die vertrauliche Geheimhaltung durch Verbergen der Geheimhaltung. Steganographische Verfahren lassen sich unterteilen in Verfahren, die darauf basieren, dass das Verfahren nicht nach außen hin bekannt ist und in Verfahren, die zusätzlich Verschlüsselung verwenden, um der Nachricht günstige statistische Merkmale zu geben. Somit sind Produkte wie TrueCrypt und FreeOTFE steganographische Produkte, die Verschlüsselung verwenden. Da die Grenzen zwischen Steganographie und Kryptographie innerhalb der Fachliteratur unterschiedlich diskutiert werden und fliessend sind,

findet keine weitere Betrachtung des Themas Steganographie innerhalb dieser Arbeit statt.

7.2 Produktzusammenstellung

Ist man darauf angewiesen, dass Sicherheitsanforderung S0 erfüllt ist[1], gibt es derzeit kein Produkt, welches gleichzeitig Sicherheitsanforderung S3 adressiert. Somit ist es nicht möglich, alle in Kapitel 3 definierten Sicherheitsanforderungen zu erfüllen. Akzeptiert man die Sonderstellung der TrueCrypt-Lizenz und beschränkt sich auf die durch den offenen Quellcode gegebene Möglichkeit diesen auf sicherheitskritische Aspekte analysieren zu können, dann lässt sich eine Produktzusammenstellung wie folgt bilden:

1. S0 impliziert den Einsatz eines OSS-Betriebssystems.
2. Die Verwendung von TrueCrypt impliziert die Verwendung von Microsoft Windows, GNU/Linux oder Mac OS.
3. (1.) und (2.) bedingen zusammengenommen den Einsatz von GNU/Linux als Betriebssystemplattform. Somit fällt auch Softraid CRYPTO aus der Betrachtung.
4. Die Überprüfung der Integrität bedingt die Existenz einer vertrauenswürdigen Boot-Umgebung. Diese kann erreicht werden durch
   - Einsatz eines TPM mittels der Produkte aus dem Trusted Computing Umfeld (TrustedGRUB, GRUB-IMA, OSLO) oder
   - das Booten von einem externen vertrauenswürdigen Medium, bspw. USB-Stick, CD-ROM etc. Die Idee hierbei ist, dass externe Boot-Medien bspw. permanent mitgetragen werden können und somit nie unbemerkten Modifikationen durch unbefugte Dritte unterliegen.

Zusammenfassend lässt sich festhalten:

Betriebssystem: GNU/Linux

Verschlüsselung: dm-crypt

Schlüsselverwaltung: LUKS (implizit)

Abstreitbare Verschlüsselung: TrueCrypt

Überprüfung der Integrität: vertrauenswürdige Boot-Umgebung

[1] Zahlreiche Linux-Distributionen verfolgen das Ziel, ausschließlich OSI kompatible Software zu paketieren. Sie können somit nur Software in ihre Distribution aufnehmen, welche die Open Source Definition der OSI erfüllt.

7.3 Verbleibende Lücken

In der oben genannten Produktzusammenstellung bleiben folgende Lücken bestehen:

1. Die Interoperabilität der Produkte ist nicht implementiert. D.h. kein OSS-Produkt zum Verschlüsseln der Festplatte ist dafür ausgelegt, die Funktionalitäten eines TPM zu nutzen. Diese Brücke muss bis heute selbst implementiert werden.
2. Kein oben genanntes Produkt verwendet das TPM als Speicher für Schlüsselmaterial. Die genannten Produkte legen ihre Schlüssel entweder auf der Festplatte oder einem externen Datenspeicher wie bspw. einem USB-Stick ab. Schlüssel, die auf der Platte liegen, sind somit Brute-Force-Angriffen ausgesetzt. Bei Verwendung eines TPM wäre dies nicht der Fall, weil das TPM Maßnahmen gegen Angriffe mittels Brute-Force-Methode implementiert.
3. Kein oben genanntes FDE-Produkt verwendet eine vertrauenswürdige Boot-Umgebung. Aktuelle Beispiele wie das Stoned Bootkit[2] zeigen die Notwendigkeit für einen Secure Boot auf: Trotz des Einsatzes von TrueCrypt ist es möglich, das MBR so zu modifizieren, dass ein Bootkit implementiert werden kann.

7.4 Lösungsansätze

Um die oben aufgezeigten Lücken zu schließen, sind folgende Lösungsansätze denkbar:

1. Die Interoperabilität muss derzeit selbst implementiert werden. TrueCrypt, dm-crypt und LUKS müssten daher um eine TPM-Anbindung erweitert werden.
2. Bestehende Produkte müssten so erweitert werden, dass sie das TPM benutzen, um Schlüsselmaterial für die Festplattenverschlüsselung abzulegen bzw. an einen ganz bestimmten Integritätszustand des Systems per `seal`-Operation zu binden. Hierbei muss man allerdings Folgendes beachten: Ein System ist in einem Zustand, der mit ganz bestimmten PCR-Werten korrespondiert, wenn es in der Lage ist, einen Wert zu entschlüsseln, der an genau diese Kombination von PCR-Werten gebunden wurde. Ein Angreifer könnte aber in den Besitz des Ergebnisses kommen und dieses im Falle einer erneuten Entschlüsselungsanfrage vorweisen (Replay-Angriff). Daher sollte nach [CYC07] zusätzlich die im TPM implementierte Funktionalität zum Signieren eingesetzt werden, weil eine Signatur außerhalb des TPM nicht gefälscht werden kann.
3. Vertrauenswürdige Boot-Umgebung mittels Integritätsüberprüfung unter Benutzung eines TPM verwenden.

[2] http://www.stoned-vienna.com/

Never trust a computer you can't throw out a window.

Steve Wozniak (* 1950)

8 Exemplarische Realisierung

Im Rahmen der Arbeit ist eine exemplarische Realisierung für den Einsatz sowohl ohne TPM als auch mit TPM durchgeführt worden. Die Installation erfolgte auf einem IBM ThinkPad T43 (Type 2668) unter Verwendung der Linux-Distributionen Debian[1] 5.0.1 (i386) und Ubuntu[2] 9.04 (32 Bit).

8.1 Allgemeine Vorgaben

Für beide Ansätze mit und ohne Verwendung eines TPM gilt:

- Boot-Umgebung (/boot), unverschlüsselt
- verschlüsselte Swap-Partition (optional)
- TrueCrypt-Container für abstreitbare Verschlüsselung

Unter Verwendung eines TPM gilt zusätzlich:

- Wurzelverzeichnis (/) verschlüsselt mit LUKS/dm-crypt
- Integritätsüberprüfung mittels TPM
- Schlüsselmaterial im TPM gebunden an eine integre Boot-Umgebung (optional)

[1] http://www.debian.org/
[2] http://www.ubuntu.com/

Ist kein TPM vorhanden, wird die vertrauenswürdige Boot-Umgebung auf ein externes Medium - in diesem Fall ein USB-Stick - installiert. Es gilt:

- Boot-Umgebung (/boot) auf USB-Stick
- Wurzelverzeichnis (/) auf USB-Stick (optional verschlüsselt)
- Schlüsselmaterial auf USB-Stick und/oder Kennwort verwenden

8.2 Installation auf USB-Stick

Die eingebaute Festplatte liegt unter `/dev/sda`, der USB-Stick liegt unter `/dev/sdb`.

- Partitionierung:
  1. Partition: /dev/sdb1, Mountpoint: boot, Dateisystemtyp: ext2
  2. Partition: /dev/sdb5, Mountpoint: /, Dateisystemtyp: ext3
- Es wird keine swap-Partition auf dem USB-Stick angelegt, um nicht zu viele Schreib- und Lesezugriffe auf dem USB-Stick durchzuführen.
- GRUB wird in /dev/sdb installiert.

8.3 TrueCrypt Installation

Nach erfolgreicher Betriebssysteminstallation wird TrueCrypt installiert. Nachfolgende Partitionierung bzw. Gerätebezeichnungen gelten für die in Kapitel 8.2 beschriebene Lösung mittels USB-Stick. Das Vorgehen ist für die TPM-basierte Lösung analog, es müssen lediglich die Gerätebezeichnungen angepasst werden.

Es wird eine Partition mit der Größe von 1 GB für das *outer volume* in /dev/sda1 erstellt. Diese Größenbeschränkung dient lediglich der Zeitersparnis beim initialen Füllen der Partition mit Zufallszahlen. Sinnvoller ist es, die gesamte Festplatte (/dev/sda) zu verwenden, denn selbst die Partitionstabelle bietet einem Angreifer einen Angriffsvektor. Zum einen lässt sie sich durch einen Angreifer verändern, zum anderen gibt sie ggf. Rückschlüsse auf Datenmengen. Ist die gesamte Festplatte nach außen hin mit Rauschen gefüllt, führt eine Modifikation von Daten „lediglich“ zum Verlust von Daten.

Die nachfolgenden Schritte orientieren sich am Leitfaden zum Einrichten von *hidden volumes* aus dem Ubuntu Dokumentationswiki [ryu08]. Um ein *hidden volume* zu erstellen, müssen folgende Befehle als `root` abgesetzt werden.

Zuerst wird ein *outer volume* erstellt:

```
truecrypt --text --filesystem=none --volume-type=normal --
    encryption=AES --hash=RIPEMD-160 --random-source=/dev/urandom
     -c /dev/sda1
```

TrueCrypt fordert zur Eingabe eines Kennworts und/oder Angabe einer Schlüsseldatei auf.

Volume im device mapper mappen ohne, dass das Volume in das Dateisystem eingebunden wird:

```
truecrypt --filesystem=none /dev/sda1
```

Mittels

```
truecrypt --text --list
```

lässt sich ermitteln, wo das Volume im device mapper gemappt wurde. Als Ausgabe erhält man:

```
1: /dev/sda1 /dev/mapper/truecrypt1 -
```

Nun muss das *outer volume* mit FAT formatiert werden:

```
mkfs.vfat /dev/mapper/truecrypt1
```

Volume auswerfen:

```
truecrypt -d
```

Nun wird das *hidden volume* mit einer Beispielgröße von 250 MB innerhalb des *outer volume* erstellt:

```
truecrypt --text --filesystem=none --volume-type=hidden --size
    =262144000 --encryption=AES --hash=RIPEMD-160 --random-source
    =/dev/urandom -c /dev/sda1
```

TrueCrypt fragt erneut nach einem Kennwort und/oder Angabe einer Schlüsseldatei. Dieses Mal für das *hidden volume*.

Volume im device mapper mappen ohne, dass das Volume in das Dateisystem eingebunden wird:

```
truecrypt --filesystem=none /dev/sda1
```

TrueCrypt fordert zur Eingabe des Kennwortes auf. An dieser Stelle muss das Kennwort für das *hidden volume* angegeben werden.

Mittels

```
truecrypt --text --list
```

lässt sich ermitteln, wo das Volume im device mapper gemappt wurde. Als Ausgabe erhält man erneut:

```
1: /dev/sda1 /dev/mapper/truecrypt1 -
```

Nun muss das *hidden volume* mittels eines für den Linux Befehl mount bekannten Dateisystems (bspw. ext3) formatiert werden:

```
mkfs.ext3 /dev/mapper/truecrypt1
```

Volume auswerfen:

```
truecrypt -d
```

Verzeichnis zum Einbinden anlegen und *outer volume* so einbinden, dass das *hidden volume* geschützt bleibt.

```
mkdir /mnt/tc
truecrypt -P /dev/sda1 /mnt/tc
```

Dateien in das *outer volume* schreiben:

```
cp ubuntu.iso /mnt/tc
```

Das *outer volume* aushängen:

```
truecrypt -d
```

Nun kann das *hidden volume* eingebunden und mit sensitiven Daten beschrieben werden:

```
truecrypt /dev/sda1 /mnt/tc
cp sensitive.daten /mnt/tc
```

Schließlich das *hidden volume* aushängen:

```
truecrypt -d
```

Nun sind das *hidden volume* und das *outer volume* erfolgreich eingerichtet und mit Daten befüllt.

8.4 TrustedGRUB Installation

Es müssen TPM-Pakete und Compiler-Pakete zum Bau von TrustedGRUB installiert werden:

```
(als root:)
aptitude install tpm-tools trousers automake autoconf gcc make
```

Das GRUB-Verzeichnis unter /boot muss gesichert werden und GRUB anschließend deinstalliert werden. Weiter unten wird TrustedGRUB installiert.

```
(als root:)
mv /boot/grub /boot/grub_old
dpkg -r grub
```

TrustedGRUB kompilieren:

```
tar xzvf TrustedGRUB-1.1.3.tgz
cd TrustedGRUB-1.1.3
./build_tgruib.sh
cd TrustedGRUB-1.1.3
patch < 008_all_grub-0.97-AM_PROG_AS.patch
./configure
make
```

Anm.: Sollte das make nicht korrekt durchlaufen, müssen ggf. vorher CFLAGS für Stage 1 und 2 um `-fno-stack-protector` ergänzt werden.

TrustedGRUB installieren:

```
(als root:)
cd <userhome>/TrustedGRUB-1.1.3/TrustedGRUB-1.1.3
make install
mkdir /boot/grub
cp ../default /boot/grub/
cp stage1/stage1 /boot/grub
cp stage2/stage2 /boot/grub
cp /boot/grub_old/menu.lst /boot/grub
grub
 root (hd0,0)
 setup (hd0)
 quit
```

Neustart des Systems:

```
(als root:)
reboot
```

Die TrustedGRUB-Installation lässt sich initial testen, in dem beim Bootvorgang im GRUB-Auswahlmenü durch Betätigen der Taste `c` in den Kommandozeilenmodus gewechselt wird. Dort können explizite TrustedGRUB-Befehle wie `checkfile` oder `sha1` auf ihre Ausführung getestet werden.

TrustedGRUB benötigt eine Datei, die als `checkfile` bezeichnet wird und SHA1-Werte zu genannten Dateien enthält. Die Datei enthält Tupel beliebiger Länge bestehend aus Integritätsmesswert und Dateipfad. Ein Tupel steht in genau einer Zeile, die mit genau einem Zeilenumbruchzeichen („\n“) endet. Die Gesamtgröße der Datei ist allerdings auf 8096 Bytes beschränkt. Jede Zeile ist wie folgt zusammengesetzt:

```
<SHA1-Wert><Whitespace-Zeichen><(hdX,Y)/Pfad>
```

SHA1-Wert der Datei (40 Byte)

Whitespace-Zeichen als Trennzeichen

(hdX,Y)/Pfad als vollqualifizierte GRUB-Pfadangabe zur Datei mit: X = Nummer der Festplatte (0..n), Y = Nummer der Partition (0..n), Pfad = Pfad zur Datei

Als Beispiel sei ein Checkfile mit drei Einträgen (Kernel, initial ramdisk, GRUB Menü) genannt. Alle Dateien liegen auf der ersten Festplatte in der ersten Partition.

```
4af1fdc7c2d425c6c13995ede91c77a417c2fa6f (hd0,0)/vmlinuz-2.6.26-2-686
8bbc58e1d05e4003ad30ad010edd8ccba30f2a9e (hd0,0)/initrd.img-2.6.26-2-686
c0d99cef6edc8d33a6f1a29f3ca2542798a65ebe (hd0,0)/grub/menu.lst
```

Hinweis: Das Checkfile ist eine beliebte Fehlerquelle. TrustedGRUB ist sehr empfindlich, was die Syntax der Datei angeht. Der Aufbau muss sich genau an die oben genannten Regeln halten, sonst bricht TrustedGRUB den Boot-Vorgang ab.

When privacy is outlawed only outlaws will have privacy.
Philip R. „Phil“ Zimmermann Jr. (* 1954)

9 Reflexion

9.1 Erkenntnisse aus der exemplarischen Realisierung

Nach der exemplarischen Realisierung lässt sich konstatieren:

1. Die Integrität des Checkfile wird durch TrustedGRUB nicht überprüft. Ein Angreifer könnte somit den Kernel durch einen Kernel mit eingebautem RootKit auswechseln und den korrespondierenden SHA1-Wert im Checkfile ersetzen. Der Bootvorgang würde keinen Fehler melden. Wie bereits in Kapitel 5.5.1 beschrieben, enthält PCR 13 alle Messungen aus dem Checkfile-Vorgang. Somit würde der Angriff durch einen anderen Wert in PCR 13 auffallen. [MS08] schlägt daher vor, Daten mittels `seal`-Operation an PCR 13 zu versiegeln. Sobald später die `unseal`-Operation ausgeführt werden würde, wäre die Manipulation ersichtlich.
2. Eine Überprüfung weiterer Betriebssystemkomponenten durch den Kernel erfolgt nicht. Somit reicht die *Chain of Trust* lediglich bis zum Kernel. Abhilfe verschafft die Verschlüsselung der restlichen Betriebssystembestandteile: Wurzelverzeichnis mittels dm-crypt verschlüsseln.

9.2 Erfüllung der Sicherheitsanforderungen

Wie werden die Sicherheitsanforderungen durch die exemplarische Realisierung adressiert?

S0: Open-Source Alle eingesetzen Produkte erfüllen die Anforderung nach OSS. Lediglich TrueCrypt nimmt eine Sonderstellung auf Grund seiner Lizenzproblematik ein.

S1: Gewährleistung der Vertraulichkeit Die Verschlüsselung mittels dm-crypt adressiert diese Anforderung.

S2: Überprüfbarkeit der Integrität Durch den Einsatz einer vertrauenswürdigen Boot-Umgebung ist es möglich, die Integrität der Daten zu überprüfen.

S3: Glaubhafte Abstreitbarkeit TrueCrypt implementiert das Konzept der abstreitbaren Verschlüsselung und erfüllt somit diese Sicherheitsanforderung.

Datenschutz ist unerlässliche Voraussetzung für eine demokratisch verantwortbare Informationsgesellschaft.

Hartmut Lubomierski (* 1943)

10 Schlussbemerkung

Es verwundert, dass es keine Off-the-shelf-Lösung gibt, die alle in Kapitel 3 genannten Sicherheitsanforderungen in einem Produkt vereint umsetzt. Je mehr mobile Endgeräte Einzug in unseren Alltag erhalten, und daher zunehmend sensitive Daten über uns speichern, desto stärker wird die Nachfrage nach umfassenden Lösungen, die Teile oder alle genannten Sicherheitsanforderungen adressieren.

Trusted Computing ist eine noch relativ junge Technologie im Bereich der IT-Industrie. Der konkrete Nutzen ist für viele noch nicht ersichtlich, die Akzeptanz für diese Technologie stellenweise gering. Konzepte wie die *Chain of Trust* sind ggf. gar nicht erfüllbar, weil Integritätsmessungen auch zur Laufzeit des Betriebssystems notwendig wären. Derzeit erlaubt die *Chain of Trust* Aussagen über zeitlich zurückliegende Messungen. Wenn aber die bereits vermessene Software zur Laufzeit kompromittiert wird, bspw. durch einen Bufferoverflow, kann auch keine *Chain of Trust* helfen. Sicherheitslücken wird es immer geben, daher wird gängige Software immer kompromittiert werden[1].

Wer kein TPM besitzt, einem TPM nicht vertraut oder aus anderen Gründen kein TPM verwenden möchte, kann sich mit Hilfe von *coreboot*[2] sein eigenes BIOS nach eigenen Sicherheitsanforderungen erstellen und somit ggf. eine vertrauenswürdige Boot-Umgebung aufbauen.

[1] Eine Ausnahme bildet Software, die einer Spezifikation genügt und dies per Korrektsheitsbeweis nachgewiesen wurde. Meist handelt es sich hierbei um Spezialsoftware, die auf eine konkrete Aufgabe zugeschnitten ist und nicht um umfangreiche Linux-Distributionen.

[2] http://www.coreboot.org/Welcome_to_coreboot

Das Konzept der abstreitbare Verschlüsselung scheint - so der Eindruck nach dem Lesen diverser Artikel, Webseiten, Foreneinträge, Blogposts etc. - nicht unumstritten zu sein. Der Gegenstand, an dem sich die Meinungen teilen, ist die Frage, ob nicht bereits der Einsatz eines Produktes, welches das Konzept der abstreitbaren Verschlüsselung umsetzt, ein starkes Indiz für die Existenz sensitiver Daten ist. [Sou09] argumentiert daher, dass diese Fragestellung wegfiele, wenn sich so ein Produkt als Standard etablieren würde. Damit wäre die Existenz eines solchen Produktes kein Indiz mehr. Alle namhaften Betriebssystemhersteller als auch die Open-Source-Community müssten das Konzept der abstreitbaren Verschlüsselung in ihren Betriebssystemen implementieren. Ob dies jemals geschehen wird, bleibt fraglich.

Sobald ZFS Crypto und die TPM-Unterstützung in OpenSolaris fertiggestellt sind, bietet sich OpenSolaris als weitere Lösung zur Umsetzung der Sicherheitsanforderungen an. Es bleibt allerdings offen, ob ZFS jemals das Konzept der abstreitbaren Verschlüsselung implementieren wird. Dafür wird es nach Aussage der TrueCrypt-Webseite[3] vielleicht irgendwann TrueCrypt für OpenSolaris geben.

Philippe Quéau, Direktor der UNESCO-Abteilung für Information und Informatik postulierte 1998 eine ethische Vision der Informationsgesellschaft und sagte:

> „Der Schutz des Privatlebens ist am Ende dieses Jahrhunderts zu einer der wichtigsten Aufgaben bei den Menschenrechten geworden. Sie hat mit den Grundlagen der Menschenwürde und dem heiligen Wesen der menschlichen Person zu tun, die aus kommerziellen und politischen Zwecken durch gefährliche Formen des Eindringen bedroht werden.“ [Que98]

Peter Schaar fordert in [Sch07], dass der Datenschutz um eine Datenschutztechnologie ergänzt werden muss. Diese Arbeit ist ein Beitrag hierzu. Sie greift das Thema der Notebook-Sicherheit auf, aggregiert die benötigten theoretische Grundlagen, liefert einen Marktüberblick, analysiert ausgewählte Lösungen und entwirft schließlich ein Gesamtkonzept zur Adressierung der formulierten Sicherheitsanforderungen. Die exemplarische Realisierung und anschließende Reflexion bieten eine Grundlage für weitere Forschungs- und Entwicklungsarbeiten zur Schaffung einer allumfassenden Lösung.

[3] http://www.truecrypt.org/misc/opensolaris

Quellenverzeichnis

[AB96] Ross Anderson and Eli Biham. Tiger: A Fast New Hash Function. http://www.cs.technion.ac.il/~biham/Reports/Tiger/tiger/tiger.html, Februar 1996. [Online; Stand 24. August 2009].

[Ada97] C. Adams. The CAST-128 Encryption Algorithm. http://www.rfc-editor.org/rfc/rfc2144.txt, Mai 1997. [Online; Stand 24. August 2009].

[AG99] C. Adams and J. Gilchrist. The CAST-256 Encryption Algorithm. http://www.rfc-editor.org/rfc/rfc2612.txt, Juni 1999. [Online; Stand 24. August 2009].

[Ano09] Suno Ano. Full-disk Encryption. http://sunoano.name/ws/public_xhtml/dm-crypt_luks.html#sec14, August 2009. [Online; Stand 3. September 2009].

[AW06] Jacob Appelbaum and Ralf-Philipp Weinmann. Unlocking FileVault - An analysis of Apple's diskencryption system. http://events.ccc.de/congress/2006/Fahrplan/attachments/1244-23C3VileFault.pdf, Dezember 2006. [Online; Stand 26. August 2009].

[AWD] Julian Assange, Ralf P. Weinmann, and Suelette Dreyfus. Rubberhose cryptographically deniable transparent disk encryption system. http://iq.org/~proff/rubberhose.org/. [Online; Stand 17. September 2009].

[BCC04] Ernie Brickell, Jan Camenisch, and Liqun Chen. Direct Anonymous Attestation. http://www.hpl.hp.com/techreports/2004/HPL-2004-93.pdf, Mai 2004. [Online; Stand 25. August 2009].

[BCD+99] Carolynn Burwick, Don Coppersmith, Edward D'Avignon, Rosario Gennaro, Shai Halevi, Charanjit Jutla, Stephen M. Matyas Jr., Luke O'Connor, Mohammad Peyravian, David Safford, and Nevenko Zunic. MARS - a candidate cipher for AES. http://researchweb.watson.ibm.com/security/mars.ps, September 1999. [Online; Stand 25. August 2009].

[Ber09] Donnie Berkholz. truecrypt has a dangerous license. http://bugs.gentoo.org/show_bug.cgi?id=241650, Februar 2009. [Online; Stand 16. September 2009].

[Bev09] Marc Bevand. MD5 Chosen-Prefix Collisions on GPUs. http://perso.epita.fr/~bevand_m/talks/bevand-md5-chosen-prefix-slides.pdf, Juli 2009. [Online; Stand 25. August 2009].

[BK09] Alex Biryukov and Dmitry Khovratovich. Related-key Cryptanalysis of the Full AES-192 and AES-256. http://eprint.iacr.org/2009/317.pdf, Juni 2009. [Online; Stand 24. August 2009].

[BKN09] Alex Biryukov, Dmitry Khovratovich, and Ivica Nikolić. Distinguisher and Related-Key Attack on the Full AES-256 (Extended Version). http://eprint.iacr.org/2009/241.pdf, Mai 2009. [Online; Stand 24. August 2009].

[Bun83] Das Bundesverfassungsgericht. BVerfGE 65, 1 - Volkszählung. http://www.servat.unibe.ch/law/dfr/bv065001.html, Dezember 1983. [Online; Stand 24. August 2009].

[Cal08] Tom Callaway. TrueCrypt licensing concern. http://lists.freedesktop.org/archives/distributions/2008-October/000276.html, Oktober 2008. [Online; Stand 16. September 2009].

[CDNO97] Ran Canetti, Cynthia Dwork, Moni Naor, and Rafail Ostrovsky. Deniable Encryption. http://eprint.iacr.org/1996/002.ps, Juni 1997. [Online; Stand 19. September 2009].

[Cha06] David Challener. TCG Software Stack (TSS) Specification Version 1.2 Level 1. http://www.trustedcomputinggroup.org/resources/tcg_software_stack_tss_specification, Januar 2006. [Online; Stand 24. August 2009].

[CYC07] David Challener, Kent Yoder, and Ryan Catherman. *A Practical Guide to Trusted Computing*. IBM Press, 2007.

[dBB91] Bert den Boer and Antoon Bosselaers. An Attack on the Last Two Rounds of MD4. ftp://ftp.esat.kuleuven.ac.be/pub/cosic/bosselae/md4rnd23.ps.gz, Oktober 1991. [Online; Stand 24. August 2009].

[dBB93] Bert den Boer and Antoon Bosselaers. Collisions for the compression function of MD5. http://www.esat.kuleuven.ac.be/~cosicart/pdf/AB-9300.pdf, Juli 1993. [Online; Stand 24. August 2009].

[DBP96] Hans Dobbertin, Antoon Bosselaers, and Bart Preneel. RIPEMD-160: A Strengthened Version of RIPEMD. http://homes.esat.kuleuven.be/

~cosicart/pdf/AB-9601/AB-9601.pdf, April 1996. [Online; Stand 25. August 2009].

[DEJ01] 3rd D. Eastlake and P. Jones. US Secure Hash Algorithm 1 (SHA1). http://www.rfc-editor.org/rfc/rfc3174.txt, September 2001. [Online; Stand 24. August 2009].

[DI03] Roland C. Dowdeswell and John Ioannidis. The CryptoGraphic Disk Driver. http://www.imrryr.org/~elric/cgd/cgd.pdf, Juni 2003. [Online; Stand 24. August 2009].

[div09a] diverse. Informationelle Selbstbestimmung. http://de.wikipedia.org/w/index.php?title=Informationelle_Selbstbestimmung&oldid=62464744, Juli 2009. [Online; Stand 24. August 2009].

[div09b] diverse. Stolen laptop 'difficult to access'. http://www.irishtimes.com/newspaper/breaking/2009/0618/breaking29.htm, Juni 2009. [Online; Stand 24. August 2009].

[Dob96] Hans Dobbertin. Cryptanalysis of MD 5 Compress. http://www.epm.ornl.gov/~dunigan/md5crack.ps, Mai 1996. [Online; Stand 24. August 2009].

[Dwo07] Morris Dworkin. Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC. http://csrc.nist.gov/publications/nistpubs/800-38D/SP-800-38D.pdf, November 2007. [Online; Stand 25. August 2009].

[ea07a] David Challener et al. TCG Software Stack (TSS) Specification Version 1.2 Level 1 Errata A. http://www.trustedcomputinggroup.org/files/resource_files/6479CD77-1D09-3519-AD89EAD1BC8C97F0/TSS_1_2_Errata_A-final.pdf, März 2007. [Online; Stand 25. August 2009].

[ea07b] David Grawrock et al. TCG Specification Architecture Overview Revision 1.4. http://www.trustedcomputinggroup.org/resources/tcg_architecture_overview_version_14, August 2007. [Online; Stand 24. August 2009].

[ea08a] Federico Lupi et al. The cryptographic device driver (CGD). http://www.netbsd.org/docs/guide/en/chap-cgd.html, Februar 2008. [Online; Stand 26. August 2009].

[ea08b] Wolfram Schneider et al. OpenBSD Programmer's Manual - vnd - vnode disk driver. http://www.openbsd.org/cgi-bin/man.cgi?query=vnd&apropos=0&sektion=0&manpath=OpenBSD+Current&arch=i386&format=html, März 2008. [Online; Stand 26. August 2009].

[ea09a] Howwi et. al. Gemeinfreiheit. http://de.wikipedia.org/w/index.

php?title=Gemeinfreiheit&oldid=64763791#Public_Domain, September 2009. [Online; Stand 25. September 2009].

[ea09b] Martin Steiger et al. 6+ Gründe gegen TrueCrypt. http://www.macmacken.com/2009/05/30/6-gruende-gegen-truecrypt/, Mai 2009. [Online; Stand 16. September 2009].

[ea09c] Matthew V Ball et. al. Disk encryption theory. http://en.wikipedia.org/w/index.php?title=Disk_encryption_theory&oldid=308526372#XEX, August 2009. [Online; Stand 26. September 2009].

[Ebe07] Adolf Ebeling. Laptop-Schwund beim FBI. http://www.heise.de/newsticker/Laptop-Schwund-beim-FBI--/meldung/85465, Februar 2007. [Online; Stand 24. August 2009].

[Eck07] Claudia Eckert. *IT-Sicherheit: Konzepte - Verfahren - Protokolle*. Oldenbourg, 2007.

[et.09] Matthew V Ball et.al. Disk encryption theory. http://en.wikipedia.org/w/index.php?title=Disk_encryption_theory&oldid=308526372, August 2009. [Online; Stand 1. September 2009].

[Fer05] Niels Ferguson. Authentication weaknesses in GCM. http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/comments/CWC-GCM/Ferguson2.pdf, Mai 2005. [Online; Stand 27. September 2009].

[Fou09] TrueCrypt Foundation. TrueCrypt - Free Open-Source Disk Encryption - Documentation - Hidden Volume. http://www.truecrypt.org/docs/?s=hidden-volume, 2009. [Online; Stand 27. September 2009].

[Fra08] Mark Frauenfelder. US Customs TSA confiscating laptops. http://www.boingboing.net/2008/02/07/tsa-confiscating-lap.html, Februar 2008. [Online; Stand 24. August 2009].

[Fru04a] Clemens Fruhwirth. EME32-AES for CryptoAPI. http://article.gmane.org/gmane.linux.kernel.device-mapper.dm-crypt/544, Oktober 2004. [Online; Stand 25. September 2009].

[Fru04b] Clemens Fruhwirth. Linux hard disk encryption settings. http://clemens.endorphin.org/LinuxHDEncSettings, 2004. [Online; Stand 20. September 2009].

[Fru04c] Clemens Fruhwirth. TKS1 - An anti-forensic, two level, and iterated key setup scheme. http://clemens.endorphin.org/TKS1-draft.pdf, Juli 2004. [Online; Stand 24. August 2009].

[Fru05] Clemens Fruhwirth. New Methods in Hard Disk Encryption. http://clemens.endorphin.org/nmihde/nmihde-A4-os.pdf, Juli 2005. [Online; Stand 24. August 2009].

[fSidI] Bundesamt für Sicherheit in der Informationstechnik. Sicherheit durch Verschlüsselung. https://www.bsi.bund.de/ContentBSI/Publikationen/Faltblaetter/F27Verschluesselung.html. [Online; Stand 17. September 2009].

[fSidI09] Bundesamt für Sicherheit in der Informationstechnik. BSI - Technische Richtlinie - Bürgerportale Funktionalitätsprüfspezifikation - Übersichtsdokument. https://www.bsi.bund.de/cae/servlet/contentblob/486046/publicationFile/31132/TR_BP_FU-PS_pdf.pdf, 2009. [Online; Stand 24. August 2009].

[Fut07] David Futcher. [needs-packaging] Truecrypt. https://bugs.edge.launchpad.net/ubuntu/+bug/109701, April 2007. [Online; Stand 16. September 2009].

[Gre09] Lucky Green. Encrypting Disk Partitions. http://www.freebsd.org/doc/en/books/handbook/disks-encrypting.html, 2009. [Online; Stand 26. August 2009].

[Gri09] Ryan Grim. WATCH: DoJ Official Blows Cover Off PATRIOT Act. http://www.huffingtonpost.com/2009/09/23/watch-doj-official-blows_n_296209.html, September 2009. [Online; Stand 27. September 2009].

[Gro08] GrouNation. How to use a TPM with Linux. https://www.grounation.org/index.php?post/2008/07/04/8-how-to-use-a-tpm-with-linux, Juli 2008. [Online; Stand 22. September 2009].

[HR03a] Shai Halevi and Phillip Rogaway. A Parallelizable Enciphering Mode. http://eprint.iacr.org/2003/147.pdf, Juli 2003. [Online; Stand 24. August 2009].

[HR03b] Shai Halevi and Phillip Rogaway. A Tweakable Enciphering Mode. http://eprint.iacr.org/2003/148.pdf, Juni 2003. [Online; Stand 24. August 2009].

[Hö04] Ralf Hölzer. Cryptoloop HOWTO. http://www.tldp.org/HOWTO/Cryptoloop-HOWTO/, Januar 2004. [Online; Stand 26. August 2009].

[Ini] Open Source Initiative. The Open Source Definition. http://www.opensource.org/docs/osd. [Online; Stand 25. August 2009].

[IR07] Wyllys Ingersoll and Scott A. Rotondo. OpenSolaris Project: Trusted Plat-

form Module support. http://www.opensolaris.org/os/project/tpm/, Februar 2007. [Online; Stand 26. September 2009].

[Jae08] Andreas Jaeger. [opensuse-buildservice] non-OSI compliant packages in the openSUSE Build Service. http://lists.opensuse.org/opensuse-buildservice/2008-10/msg00055.html, Oktober 2008. [Online; Stand 16. September 2009].

[Joh06] Steve Johnson. Trusted Boot Loader. http://elinux.org/images/2/28/Trusted_Boot_Loader.pdf, April 2006. [Online; Stand 24. August 2009].

[Kal92] B. Kaliski. The MD2 Message-Digest Algorithm. http://www.rfc-editor.org/rfc/rfc1319.txt, April 1992. [Online; Stand 24. August 2009].

[Kal00] B. Kaliski. PKCS #5: Password-Based Cryptography Specification Version 2.0. http://www.rfc-editor.org/rfc/rfc2898.txt, September 2000. [Online; Stand 24. August 2009].

[Kam03] Poul-Henning Kamp. GBDE - GEOM Based Disk Encryption. http://phk.freebsd.dk/pubs/bsdcon-03.gbde.paper.pdf, September 2003. [Online; Stand 24. August 2009].

[Kam06] Poul-Henning Kamp. FreeBSD Kernel Interfaces Manual - GEOM. http://www.freebsd.org/cgi/man.cgi?query=geom&apropos=0&sektion=4&manpath=FreeBSD+7.2-RELEASE&format=html, Mai 2006. [Online; Stand 26. August 2009].

[Kar07] Nathan L. Karstens. Deniable Encryption. http://iasg.iac.iastate.edu/library/0607/20070213_nate_presentation.pdf, Februar 2007. [Online; Stand 24. August 2009].

[Kau07a] Bernhard Kauer. OSLO Changelog. http://os.inf.tu-dresden.de/~kauer/oslo/CHANGELOG, Juni 2007. [Online; Stand 26. August 2009].

[Kau07b] Bernhard Kauer. OSLO: Improving the security of Trusted Computing. http://os.inf.tu-dresden.de/papers_ps/kauer07-oslo.pdf, August 2007. [Online; Stand 18. September 2009].

[Kra09] David Krause. OpenBSD System Manager's Manual - mount_vnd, vnconfig - configure vnode disks. http://www.openbsd.org/cgi-bin/man.cgi?query=vnconfig&apropos=0&sektion=0&manpath=OpenBSD+Current&arch=i386&format=html, Februar 2009. [Online; Stand 26. August 2009].

[Kub09] Kubieziel. Data Encryption Standard. http://de.wikipedia.org/w/index.php?title=Data_Encryption_Standard&oldid=63676098#Anwendungen, August 2009. [Online; Stand 25. September 2009].

[Lab99] RSA Laboratories. PKCS #5 v2.0: Password-Based Cryptography Standard. ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-5v2/pkcs5v2-0.pdf, März 1999. [Online; Stand 25. August 2009].

[Lab02] RSA Laboratories. PKCS #1 v2.1: RSA Cryptography Standard. ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-1/pkcs-1v2-1.pdf, Juni 2002. [Online; Stand 25. August 2009].

[Liv08] Randy Livingston. Stanford alerts employees that stolen laptop had personal data. http://news-service.stanford.edu/news/2008/june11/laprelease-061108.html, Juni 2008. [Online; Stand 24. August 2009].

[LRW02] Moses Liskov, Ronald L. Rivest, and David Wagner. Tweakable Block Ciphers. http://www.eecs.berkeley.edu/~daw/papers/tweak-crypto02.pdf, August 2002. [Online; Stand 24. August 2009].

[MHP09] Cameron McDonald, Philip Hawkes, and Josef Pieprzyk. Differential Path for SHA-1 with complexity $O(2^{52})$. http://eprint.iacr.org/2009/259, Juni 2009. [Online; Stand 24. August 2009].

[Min08] Kazuhiko Minematsu. A Study of Block Cipher Modes for Encryption and Authentication. http://dspace.wul.waseda.ac.jp/dspace/bitstream/2065/28755/3/Honbun-4809.pdf, März 2008. [Online; Stand 14. September 2009].

[MNM04] M. Matsui, J. Nakajima, and S. Moriai. A Description of the Camellia Encryption Algorithm. http://www.rfc-editor.org/rfc/rfc3713.txt, April 2004. [Online; Stand 24. August 2009].

[Mof07] Darren J Moffat. Data at rest: ZFS & lofi crypto. http://opensolaris.org/os/project/zfs-crypto/files/opensolaris-zfs-crypto.pdf, Januar 2007. [Online; Stand 26. August 2009].

[Mof08] Darren Moffat. ccm.c. http://cr.opensolaris.org/~darrenm/zfs-crypto-gate/usr/src/common/crypto/modes/ccm.c.html, 2008. [Online; Stand 25. September 2009].

[Mof09] Darren Moffat. OpenSolaris Project: ZFS on disk encryption support. http://www.opensolaris.org/os/project/zfs-crypto/, September 2009. [Online; Stand 26. September 2009].

[MP93] G. Malkin and T. LaQuey Parker. Internet Users' Glossary. http://www.rfc-editor.org/rfc/rfc1392.txt, Januar 1993. [Online; Stand 24. August 2009].

[MS08] Felix Teerkorn Marcel Selhorst, Christian Stüble. TSS-Studie - Einführung und Analyse des quelloffenen TCG Software Stacks TrouSerS und Werkzeuge

in dessen Umfeld. http://www.sirrix.de/media/downloads/53985.pdf,download, Mai 2008. [Online; Stand 23. September 2009].

[Mul04] Frédéric Muller. The MD2 Hash Function is Not One-Way. http://www.iacr.org/archive/asiacrypt2004/33290211/33290211.ps, Dezember 2004. [Online; Stand 24. August 2009].

[Nas09] Nasa-verve et. al. IEEE P1619 - LRW issue. http://en.wikipedia.org/w/index.php?title=IEEE_P1619&oldid=308420013#LRW_issue, August 2009. [Online; Stand 3. September 2009].

[otUK00] Parliament of the United Kingdom. Regulation of Investigatory Powers Act 2000. http://www.opsi.gov.uk/acts/acts2000/ukpga_20000023_en_8#pt3-pb1, July 2000. [Online; Stand 24. August 2009].

[Pee09] Marco Peereboom. OpenBSD Programmer's Manual - softraid - Software RAID. http://www.openbsd.org/cgi-bin/man.cgi?query=softraid&apropos=0&sektion=0&manpath=OpenBSD+Current&arch=i386&format=html, Juli 2009. [Online; Stand 26. August 2009].

[Pro06] Graeme Proudler. First European Summer School on Trusted Infrastructure Technologies. smb://samba.inform.fh-hannover.de/skripte/skripte/master/spez_it_sicherheit/ws0809_sicherheit_in_lokalen_netzen/Graeme_Proudler_first_euro_summer_school_oxford_200609.pdf, 2006.

[Qua06] Thomas Quas. RFP: truecrypt – Free open-source disk encryption software for Windows XP/2000/2003 and Linux. http://bugs.debian.org/cgi-bin/bugreport.cgi?bug=364034, April 2006. [Online; Stand 16. September 2009].

[Que98] Philippe Queau. Eine ethische Vision der Informationsgesellschaft. http://www.heise.de/tp/r4/artikel/2/2539/1.html, November 1998. [Online; Stand 26. August 2009].

[Rei04] Markus Reichelt. Why Mainline Cryptoloop Should Not Be Used. http://mareichelt.de/pub/texts.cryptoloop.php/, Juni 2004. [Online; Stand 26. August 2009].

[Riv92a] R. Rivest. The MD4 Message-Digest Algorithm. http://www.rfc-editor.org/rfc/rfc1320.txt, April 1992. [Online; Stand 24. August 2009].

[Riv92b] R. Rivest. The MD5 Message-Digest Algorithm. http://www.rfc-editor.org/rfc/rfc1321.txt, April 1992. [Online; Stand 24. August 2009].

[RRSY98] Ronald L. Rivest, M. J. B. Robshaw, R. Sidney, and Y. L. Yin. The

RC6 Block Cipher. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.35.7355, August 1998. [Online; Stand 25. August 2009].

[Ruu09] Jari Ruusu. loop-AES.README. http://loop-aes.sourceforge.net/loop-AES.README, Juni 2009. [Online; Stand 26. August 2009].

[RW03] Phillip Rogaway and David Wagner. A Critique of CCM. http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/comments/800-38_Series-Drafts/CCM/RW_CCM_comments.pdf, Februar 2003. [Online; Stand 27. September 2009].

[ryu08] ryukent. TruecryptHiddenVolume. https://help.ubuntu.com/community/TruecryptHiddenVolume, November 2008. [Online; Stand 24. September 2009].

[Saa04a] Markku-Juhani Olavi Saarinen. Encrypted Watermarks and Linux Laptop Security. http://www.tcs.hut.fi/~mjos/doc/wisa2004.pdf, August 2004. [Online; Stand 24. August 2009].

[Saa04b] Markku-Juhani Olavi Saarinen. Linux for the Information Smuggler. http://mareichelt.de/pub/notmine/diskenc.pdf, Februar 2004. [Online; Stand 24. August 2009].

[Sch93] Bruce Schneier. Description of a New Variable-Length Key, 64-Bit Block Cipher (Blowfish). http://www.schneier.com/paper-blowfish-fse.html, Dezember 1993. [Online; Stand 25. August 2009].

[Sch00] Bruce Schneier. Crypto-Gram: March 15, 2000. http://www.schneier.com/crypto-gram-0003.html#8, März 2000. [Online; Stand 26. September 2009].

[Sch02] Bruce Schneier. Crypto-Gram Newsletter - One-Time Pads. http://www.schneier.com/crypto-gram-0210.html#7, Oktober 2002. [Online; Stand 3. September 2009].

[Sch07] Peter Schaar. *Das Ende der Privatsphäre - Der Weg in die Überwachungsgesellschaft*. C. Bertelsmann, 2007.

[Sch08] Bruce Schneier. Taking your laptop into the US? Be sure to hide all your data first. http://www.guardian.co.uk/technology/2008/may/15/computing.security, Mai 2008. [Online; Stand 24. August 2009].

[Sch09] Bruce Schneier. Schneier on Security. http://www.schneier.com/blog/archives/2009/08/password_advice.html, August 2009. [Online; Stand 26. September 2009].

[SCS+08] Boaz Shahar, David Clunie, Rich Shroeppel, Phillip Rogaway, Vijay Bharadwaj, and Neils Ferguson et al. Public Comments on the XTS-

AES Mode. http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/comments/XTS/collected_XTS_comments.pdf, September 2008. [Online; Stand 15. September 2009].

[Sec09] RSA Security. Public-Key Cryptography Standards (PKCS). http://www.rsa.com/rsalabs/node.asp?id=2124, 2009. [Online; Stand 25. September 2009].

[SKW+98] Bruce Schneier, John Kelsey, Doug Whiting, David Wagner, Chris Hall, and Niels Ferguson. Twofish: A 128-Bit Block Cipher. http://www.schneier.com/paper-twofish-paper.pdf, Juni 1998. [Online; Stand 25. August 2009].

[Sou09] Soulskill. Encryption? What Encryption? http://it.slashdot.org/story/09/08/12/1255241/Encryption-What-Encryption, August 2009. [Online; Stand 25. September 2009].

[SPT+08] Jan Steffan, Andreas Poller, Jan Trukenmüller, Jan-Peter Stotz, and Sven Türpe. BitLocker Drive Encryption im mobilen und stationären Unternehmenseinsatz - Ein Leitfaden für Anwender. http://testlab.sit.fraunhofer.de/content/output/project_results/bitlocker/BitLocker-Leitfaden.pdf, März 2008. [Online; Stand 26. August 2009].

[SSA+08] Alexander Sotirov, Marc Stevens, Jacob Appelbaum, Arjen Lenstra, David Molnar, Dag Arne Osvik, and Benne de Weger. MD5 considered harmful today. http://www.win.tue.nl/hashclash/rogue-ca/, Dezember 2008. [Online; Stand 25. August 2009].

[Sta99] Federal Information Processing Standards. DATA ENCRYPTION STANDARD. http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf, Oktober 1999. [Online; Stand 25. August 2009].

[Sta01] Federal Information Processing Standards. ADVANCED ENCRYPTION STANDARD. http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf, November 2001. [Online; Stand 25. August 2009].

[Sta02] Federal Information Processing Standards. The Keyed-Hash Message Authentication Code. http://csrc.nist.gov/publications/fips/fips198/fips-198a.pdf, März 2002. [Online; Stand 25. August 2009].

[TB09] A. Tschentscher and Sven Broichhagen. BVerfGE 65, 1 - Volkszählung. http://www.servat.unibe.ch/verfassungsrecht/bv065001.html#Rn158, April 2009. [Online; Stand 26. September 2009].

[vH08] Josef von Helden. Spezialthema IT-Sicherheit: Sicherheit in lokalen Netzen. smb://samba.inform.fh-hannover.de/skripte/skripte/master/spez_

it_sicherheit/ws0809_sicherheit_in_lokalen_netzen/sicherheit_lokale_netze_vl_kap8_trusted_computing_ws0809_v41.pdf, 2008.

[War07] Mark Ward. Campaigners hit by decryption law. http://news.bbc.co.uk/1/hi/technology/7102180.stm, November 2007. [Online; Stand 24. August 2009].

[WFLY04] Xiaoyun Wang, Dengguo Feng, Xuejia Lai, and Hongbo Yu. Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD. http://eprint.iacr.org/2004/199.pdf, August 2004. [Online; Stand 24. August 2009].

[WHF03] D. Whiting, R. Housley, and N. Ferguson. Counter with CBC-MAC (CCM). http://www.rfc-editor.org/rfc/rfc3610.txt, September 2003. [Online; Stand 24. August 2009].

[Wil06] Andreas Wilkens. Boeing kommt Laptop mit tausenden Mitarbeiterdaten abhanden. http://www.heise.de/newsticker/Boeing-kommt-Laptop-mit-tausenden-Mitarbeiterdaten-abhanden--/meldung/82523, Dezember 2006. [Online; Stand 24. August 2009].

[Wil09] Andreas Wilkens. Erste Passwort-Erzwingungshaft in Großbritannien. http://www.heise.de/newsticker/Erste-Passwort-Erzwingungshaft-in-Grossbritannien--/meldung/143462, August 2009. [Online; Stand 25. September 2009].

[Zie08] Peter-Michael Ziegler. Sprunghafter Anstieg der Zahl von Datenklau-Opfern in den USA. http://www.heise.de/security/Sprunghafter-Anstieg-der-Zahl-von-Datenklau-Opfern-in-den-USA--/news/meldung/101222, Januar 2008. [Online; Stand 24. August 2009].

[ZP90] J. Zweig and C. Partridge. TCP Alternate Checksum Options. http://www.rfc-editor.org/rfc/rfc1146.txt, März 1990. [Online; Stand 24. August 2009].

11 Anhang

11.1 GRUB

Grand Unified Bootloader (GRUB) ist ein unter GPL stehendes Urladeprogramm, welches meist zum Starten von Betriebssystemen verwendet wird. GRUB ist inzwischen in der Lage zahlreiche Dateisysteme zu lesen und von unterschiedlichen Medien als auch per TFTP vom Netzwerk zu booten. GRUB ist vor allem innerhalb der Unix-Welt und Linux-Distributionen weit verbreitet. Konzeptionell ist GRUB Legacy[1] in drei Teile gegliedert:

Stage 1 wird in den Master Boot Record[2] (MBR) geschrieben und dient dazu, Stage1.5 zu laden.

Stage 1.5 liegt zwischen MBR respektive Stage 1 und dem ersten Block der ersten Partition und dient dazu, Stage 2 zu laden. Stage 1.5 kann genau das Dateisystem lesen, auf dem Stage 2 liegt.

Stage 2 kann sich in einer beliebigen Partition befinden und dient dazu, den Betriebssystemkern oder ein anderes MBR zu laden und zu starten. Hierzu enthält es Dateisystemtreiber, um das Dateisystem in dem bspw. der Betriebsystemkern vor-

[1] Als GRUB Legacy werden alle Versionen bis 0.97 bezeichnet

[2] Die Größe des MBR ist auf 512 Byte beschränkt. Das MBR enthält die 64 Byte Partitionstabelle, so dass höchstens 448 Byte für die Installation von Stage 1 übrig bleiben.

liegt, lesen zu können. Zusätzlich enthält es Programmcode zum Darstellen eines Auswahlmenüs und einer interaktiver Kommandozeile.

GRUB2 stellt einen völligen Neuentwurf gegenüber GRUB Legacy dar. GRUB2 verfolgt nach Aussage der Projektseite[3] Ziele wie Unterstützung für Skriptsprachen, GUI, dynamisch nachladbare Module, Sprachanpassung, modularer und hierarchischer sowie objektorientierter Aufbau etc. Im Unterschied zu GRUB Legacy entfällt Stage 1.5. Es bleiben:

Stage 1 wird in den Master Boot Record (MBR) geschrieben und dient dazu, Stage 2 zu laden.

Stage 2 ist in einen Kernel (kernel.img) und diverse Module (.mod) aufgeteilt. Der Kernel enthält ausschließlich essentiellen Code (Dekompression, Lader für Module, Rettungs-Shell), der Rest liegt in den Modulen. Die Verteilung der Komponenten erfolgt an unterschiedliche Orte:

- Der Kernel und die Module, die zum Lesen des Dateisystems, auf dem die restlichen Module sich befinden, werden zu einer komprimierten Datei names `core.img` zusammengeführt.
- Die restlichen Module liegen in einem Dateisystem und können vom Kernel nachgeladen werden.

Durch dieses Vorgehen erreicht man, dass `core.img` meist relativ klein ist und somit zwischen MBR und dem ersten Block der ersten Partition liegen kann.

[3] http://www.gnu.org/software/grub/grub-2.en.html