Bundeskriminalamt

h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

Hochschule Darmstadt
–Fachbereich Informatik–

Entwicklung von Methoden zur Erkennung verschlüsselter Daten auf Datenträgern und ihre Anwendung auf die digitale Forensik

Abschlussarbeit zur Erlangung des akademischen Grades Master of Science (M.Sc.)

vorgelegt von
Simon Thurner

Referent:	Prof. Dr. Harald Baier
Korreferent:	M.Sc. Frank Breitinger
Betreuer des Bundeskriminalamts:	Dipl.-Wirtsch.-Inf. Sven Schmitt
Ausgabedatum:	13.08.2012
Abgabedatum:	13.02.2013

Kurzzusammenfassung

Diese Arbeit beschäftigt sich mit der Erkennung verschlüsselter Daten auf unterschiedlichen Abstraktionsebenen eines Datenträgers. Die Erkennung spielt im Bereich der Computerforensik eine wichtige Rolle, insbesondere bei der Live-Analyse von Computersystemen. Temporär entschlüsselte Festplattenbereiche müssen dabei identifiziert und vor Abschaltung des Rechners gesichert werden, da ein erneutes Entschlüsseln ohne Kenntnis des Schlüssels in den meisten Fällen nicht praktikabel ist. Die Erkennung kann signaturbasiert erfolgen, wobei verschlüsselte Daten anhand von spezifischen Bytemustern, Signaturen genannt, identifiziert werden. Diese Signaturen müssen jedoch nicht zwangsläufig vorhanden sein oder lassen sich oftmals problemlos manipulieren. Darüber hinaus existiert der weit verbreitete Entropietest, welcher auf eine statistische Eigenschaft verschlüsselter Daten, die Gleichverteilung, überprüft. Komprimierte sowie zufällige Daten besitzen diese Eigenschaft ebenfalls, sodass eine korrekte Klassifizierung dieser Daten mittels des Entropietests fehlschlägt.

Aus diesen Gründen wurde nach einem Ansatz gesucht, welcher unabhängig von der Signatur verschlüsselte Daten als solche identifiziert. Darüber hinaus sollen die durch den Entropietest fälschlicherweise als verschlüsselt eingestuften Daten reduziert werden. Dafür eignen sich erweiterte statistische Eigenschaften einer verschlüsselten Bytefolge, welche anhand unterschiedlicher Tests überprüft werden können.

Bei der Erkennung verschlüsselter Anwendungsdateien kann auf einen signaturbasierten Ansatz nicht verzichtet werden. Metadaten, welche das Dateiformat beschreiben und oftmals im Klartext vorliegen, können insbesondere bei kleinen Anwendungsdateien dominieren und die Testergebnisse verfälschen. Um eine Erkennung dieser Daten zu gewährleisten, wurden im Vorfeld für einige ausgewählte Dateiformate Signaturen, welche auf Verschlüsselung hinweisen, identifiziert.

Die Maßnahmen zur Erkennung von Verschlüsselung auf Datenträgern wurden im Zusammenhang mit dieser Arbeit in ein Tool umgesetzt. Während der Evaluierung dieses Tools dienten synthetisch generierte Datenträgerimages zur Überprüfung eines zuvor entwickelten Ablaufs einer Datenträgeranalyse. Dabei wurde gezeigt, dass sich statistische Tests zur Unterscheidung verschlüsselter und komprimierter Daten eignen. Jedoch kann mit diesen Tests nicht zwischen verschlüsselten und zufälligen Daten unterschieden werden, da diese Daten die gleichen statistischen Eigenschaften besitzen.

Abstract

This thesis deals with the identification of encrypted data on different abstraction layers of a data medium. The identification plays an important role in computer forensics, especially in live analysis of computer systems. Temporarily opened encrypted data areas have to be identified before shutting down the computer, because decrypting again without knowledge of the key is infeasible in practice. For identifying encrypted data, signature based approaches, which check for specific signatures, can be used. These signatures are not mandatory or can be manipulated. Moreover, the wide spread entropy test is used, which checks for a statistical property of encrypted data, which is uniform distribution. However, this property is also possessed by compressed and random data, so that a correct classification fails. For these reasons, an approach is wanted where encrypted data can be detected independently from the signature. Additionally, false positives produced by the entropy test should be reduced. For this, extended statistical properties of encrypted data were identified, which can be checked by applying different statistical tests.

For detecting encrypted application data the signature based approach cannot be dispensed. Metadata, which describe the file format, are still available in plain text. This data can be dominating especially in small files and lead to false positives. To ensure the identifiaction of such files, signatures for certain file formats, which indicate encryption, were identified previously.

In the context of this thesis, the measures for detecting encrypted data on data mediums were implemented in a tool. For evaluating this tool, synthetic images were generated for checking the workflow of a previously developed data medium analysis. It was shown, that statistical tests can be used to distinguish encrypted and compressed data. But with these tests, it isn't possible to differ enrypted from random data, because both kind of data have the same properties.

Inhaltsverzeichnis

Abkürzungsverzeichnis

AES Advanced Encryption Standard

BIOS Basic Input Output System

BKA Bundeskriminalamt

CBC Cipher Block Chaining

CSV Comma-seperated values

DES Data Encryption Standard

ECTCOE Electronic Crime Technology Center of Excellence

EFS Encrypting File System

FAT File Allocation Table

GPT GUID Partition Table

IPL Initial Program Loader

LBA Logical Block Address

LUKS Linux Unified Key Setup

MBR Master Boot Record

MFT Master File Table

NIJ National Institute of Justice

NIST National Institute of Standards and Technology

NTFS New Technology File System

OEM Original Equipment Manufacturer

OLECF Object Linking and Embedding Compound File

OPC Open Packaging Convention

PDF Portable Document Format

RSA Rivest, Shamir und Adleman

SSH Secure Shell

SSL Secure Sockets Layer

TLS Transport Layer Security

UEFI Unified Extensible Firmware Interface

UML Unified Modeling Language

VBR Volume Boot Record

XEX Xor-encrypt-xor

XOR Exklusiv Oder

XTS XEX-based tweaked-codebook mode with ciphertext stealing

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Die zunehmende Anwendung von Verschlüsselung stellt Computerforensiker vor neue Herausforderungen. Als Verschlüsselung bezeichnet man dabei den Prozess, Klartext in einen Geheimtext zu transformieren, um dessen Inhalt zu verschleiern und gegen unbefugten Zugriff zu schützen. Um dies zu erreichen, werden durch kryptographische Algorithmen komplexe Transformationen, bestehend aus Substitutionen und Permutationen, auf den Klartext angewendet [MH06].

Die Identifikation von Verschlüsselung spielt insbesondere auch aus Sicht der Live-Forensik, welche die Untersuchung laufender Systeme bezeichnet, eine wichtige Rolle. Sind auf eingeschalteten Rechnern verschlüsselte Dateien oder Festplattenbereiche entschlüsselt, geht dieser temporäre Zugriff beim Ausschalten des Computers verloren. Diese zur Laufzeit geöffneten Bereiche, welche möglicherweise ermittlungsrelevante Daten enthalten, sollten vor Abschaltung des Computers gesichert werden, da ein erneutes Entschlüsseln ohne Kenntnis des Schlüssels in den meisten Fällen nicht praktikabel ist.

Derzeit angewendete Maßnahmen zur Erkennung von Verschlüsselung benötigen datenspezifisches Wissen. Dieses Wissen umfasst bspw. charakteristische Bytemuster, nachfolgend Signaturen genannt, welche auf Verschlüsselung hinweisen. Diese Signaturen müssen bei Aktualisierungen des Dateiformats angepasst werden, außerdem lassen sich Manipulationen nicht ausschließen. Bei einem weiteren Ansatz erfolgt die Erkennung anhand von Prozesslisten. Dabei wird versucht, Verschlüsselungssoftware zu identifizieren, welche auf dem Rechner aktiv ist. Dieser Vorgang führt jedoch nicht zwangsläufig zu den verschlüsselten Daten.

Da die gegenwärtigen Erkennungsmaßnahmen nicht erfolgsversprechend sind wird nach Möglichkeiten gesucht, womit verschlüsselte Daten unabhängig von der verwendeten Verschlüsselungssoftware identifiziert werden können. Somit soll letztendlich eine Entscheidungshilfe über Abschaltung oder Durchführung einer Vor-Ort-Sicherung des eingeschalteten Computers geschaffen werden, um den Zugriff auf möglicherweise relevante Daten nicht zu verlieren.

Im Gegensatz zur Live-Forensik liegt der Rechner bei der Post-mortem-Analyse im ausgeschalteten Zustand vor. Neben dem Versuch, verschlüsselte Dateien oder Festplattenbereiche ohne Kenntnis des Schlüssels zu entschlüsseln, müssen diese zuerst als

solche identifiziert werden. Dies ist mit bisher angewandten Methoden nicht immer zweifelsfrei möglich. Diese Arbeit befasst sich mit der Entwicklung von Methoden, welche angewandte Verschlüsselung auf Live- als auch auf Offline-Systemen erkennen sollen.

Im Zusammenhang mit der Analyse von Offline-Systemen entwickelt das Bundeskriminalamt (BKA) im Projekt FILEX eine Software, mit welcher Dateien auf Datenträgern kategorisiert werden können. Ein weiterer Bestandteil dieser Software wird die Identifikation verschlüsselter Daten sein. Neben der Analyse von Live-Systemen bilden die in dieser Arbeit entwickelten Methoden die Grundlage für ein zu entwickelndes Modul, welches im Rahmen von FILEX Verschlüsselung auf unterschiedlichen Ebenen eines Datenträgers identifiziert.

1.1 Zielstellung

Bei der Untersuchung von Datenträgern im Bereich der Computerforensik spielt die Vorverarbeitung eine wichtige Rolle, bei welcher mittels Software jegliche auf dem Datenträger befindlichen Dateien identifiziert und kategorisiert werden. Werden übliche Dateitypen anhand eines fixen Bytemusters, einer Signatur, erkannt, führt dieses Verfahren nicht immer zur Identifikation von verschlüsselten Dateien. Dasselbe gilt für verschlüsselte Festplattenbereiche.

Ziel dieser Arbeit ist die Entwicklung eines signaturunabhängigen Ansatzes, mit welchem angewandte Verschlüsselung auf unterschiedlichen Ebenen eines Datenträgers identifiziert werden kann.

Erkennung von Verschlüsselung auf unterschiedlichen Ebenen

Angewandte Verschlüsselung auf einem Datenträger lässt sich auf verschiedenen Abstraktionsebenen gemäß Abbildung 1.1 feststellen. Die unterschiedlichen Abstraktionsebenen sowie deren Merkmale, welche gegenwärtig zur Erkennung von verschlüsselten Daten verwendet werden, werden im Nachfolgenden näher erläutert.

Abbildung 1.1: Unterschiedliche Abstraktionsebenen eines Datenträgers

Anwendungsebene
Dateisystemebene
Partitionsebene
Datenträgerebene

Verschlüsselter Inhalt auf der Anwendungsebene bezeichnet verschlüsselte Anwendungsdateien (z. B. Dokumente oder Archive), welche in logischer Form vorhanden sind. Die Identifikation von verschlüsselten Dateien auf dieser Abstraktionsebene setzt voraus, dass charakteristische Eigenschaften wie Header-Informationen oder fixe Bytemuster, welche auf Verschlüsselung schließen lassen, bekannt sind und nicht manipuliert wurden.

Verschlüsselung auf der Dateisystemebene umfasst Verschlüsselung, welche durch das Dateisystem durchgeführt wird. Das Dateisystem NTFS bietet diese Möglichkeit der Nutzdatenverschlüsselung. Damit erfolgt zwar eine vollständige Verschlüsselung der Datei, für die Organisation solcher Dateien benötigt das Dateisystem jedoch Informationen, welche weiterhin im Klartext vorliegen und sich zur Identifizierung eignen.

Die Partitionsebene behandelt verschlüsselte Partitionen, welche oftmals anhand von Signaturen im Volume Boot Record (VBR) identifiziert werden können. Analog dazu findet auf der Datenträgerebene Vollverschlüsselung statt, welche den gesamten Nutzdatenbereich des Datenträgers verschlüsselt. Hier liefert der Master Boot Record (MBR) gelegentlich Hinweise auf die eingesetzte Software.

Gelöschte Partitionen bzw. nicht-allozierte Datenträgerbereiche werden dem sogenannten File Carving unterzogen. Dabei erfolgt eine sequentielle Analyse eines Speicherbereichs ohne Metadaten des Dateisystems. Mit File Carving lassen sich typischerweise Fragmente von Dateien finden und wiederherstellen. Als Identifikationsmerkmal gelöschter Dateien nutzt der klassische File Carving-Prozess, analog zur Anwendungsebene, spezifische Bytemuster. Da die Identifizierung ohne Metadaten des Dateisystems geschieht hängt der Erfolg der Wiederherstellung vom Fragmentierungsgrad des Datenträgers ab.

Die aufgeführten Identifikationsmerkmale lassen sich nicht zuverlässig auf die Erkennung verschlüsselter Daten übertragen. Zum Einen können Manipulationen an Signaturen grundsätzlich nicht ausgeschlossen werden, des Weiteren existieren Datenformate, welche keinerlei Signaturen besitzen. Heutzutage dient neben den erläuterten Merkmalen die Entropie als Erkennungsmerkmal, welche die Verteilung der im Bytestrom auftretenden Bytes misst. Dabei werden jedoch komprimierte Datenformate häufig als verschlüsselt eingestuft, da bei diesen Formaten, analog zu verschlüsselten Daten, Redundanzen entfernt wurden, wodurch eine annähernde Gleichverteilung der im Bytestrom beobachteten Bytes resultiert.

In dieser Arbeit werden erweiterte Kriterien für eine zuverlässigere Erkennung von Verschlüsselung herangezogen. Neben der Betrachtung von Indikatoren wie der Dateigröße, welche bei Vielfachem der Blockgröße eines Blockverschlüsselungsalgorith-

mus ein Hinweis sein kann, sind es vor allem bestimmte statistische Eigenschaften, welche verschlüsselte Daten besitzen. Der Erfolg der Erkennung verschlüsselter Daten kann vom Dateiformat abhängen. Dabei spielt der Anteil an Metadaten, welche das Dateiformat beschreiben, eine wichtige Rolle. Metadaten erschweren die Erkennung, da diese nach der Verschlüsselung oftmals weiterhin im Klartext vorliegen und demnach nicht den Eigenschaften verschlüsselter Daten genügen.

1.2 Aufbau dieser Arbeit

Kapitel 2 liefert Grundlagen, welche für das Verständnis des in dieser Arbeit entwickelten Erkennungsansatzes notwendig sind. Dies betrifft insbesondere die Basiseigenschaften symmetrischer Kryptosysteme und die daraus resultierenden statistischen Eigenschaften verschlüsselter Datenströme. Zur Erkennung verschlüsselter Daten sieht der Erkennungsansatz die Überprüfung dieser Eigenschaften anhand der Durchführung statistischer Tests vor.

In Kapitel 3 werden statistische Tests, welche zur Identifizierung verschlüsselter Daten beitragen sollen, ausgewählt und erläutert. Die Tests werden auf die Datenträgerebenen aus Abbildung 1.1 angewendet und auf ihre Geeignetheit überprüft.

Nachdem statistische Tests als Grundlage für die Erkennung von verschlüsselten Daten identifiziert wurden, verdeutlicht Kapitel 4 einen Ablauf über eine vollständige Datenträgeranalyse. Die Analyse erfolgt dabei auf dem gesamten Datenträgerbereich. Abhängig von ersten Analyseergebnissen werden Datenträgerbereiche ggf. unterschiedlichen Tests unterzogen.

Kapitel 5 beschreibt die Umsetzung der Datenträgeranalyse in ein lauffähiges Tool. Wichtige Bestandteile der Implementierung werden aufgeführt und näher erläutert. Darüber hinaus erfolgt die Evaluierung des Tools, wobei synthetisch generierte Datenträgerimages als Testobjekte dienen.

Zum Abschluss der Arbeit erfolgt mit Kapitel 6 eine Zusammenfassung der erlangten Erkenntnisse sowie ein Ausblick auf mögliche Weiterentwicklungen.

1.3 Eigenbeitrag zur Lösung der Problemstellung

Für die Entwicklung eines möglichen Ansatzes zur Erkennung verschlüsselter Daten wurden zunächst die Vor- und Nachteile gegenwärtiger Maßnahmen identifiziert. Einer der Ansätze ist der vielfach eingesetzte Entropietest, welcher auf eine statistische Eigenschaft verschlüsselter Daten, die Gleichverteilung auftretender Bytes, überprüft.

Die Gleichverteilung wird jedoch auch von anderen, insbesondere komprimierten Dateiformaten erfüllt. Eine Einstufung durch den Entropietest erfolgt daher in vielen Fällen fälschlicherweise.

Als vielversprechend erwies sich die Idee, nach weiteren Eigenschaften zu suchen, welche verschlüsselte Daten besitzen. Da bei der Verschlüsselung von Daten durch moderne Kryptosysteme eine Bytefolge entsteht, welche zufälligen Daten gleicht, wurde auf Ansätze zur Überprüfung der Güte von Zufallszahlengeneratoren zurückgegriffen. Dazu existieren etablierte Tests, welche in dieser Arbeit auf die Erkennung von verschlüsselten bzw. zufälligen Daten übertragen wurden.

Die Anwendung der in dieser Arbeit aufgeführten Tests erfolgte bisher lediglich in Bezug auf die Messung der Güte von Pseudozufallszahlengeneratoren. Eine Überprüfung, ob sich die Tests zur Identifizierung verschlüsselter Daten eignen, erfolgte zunächst auf Clusterebene. Dafür wurden jeweils 10.000 Cluster ausgewählter verschlüsselter sowie unverschlüsselter Dateiformate den statistischen Tests unterzogen und die Ergebnisse notiert. Die erlangten Erkenntnisse wurden zur Identifizierung von Einstufungskriterien für zufällige bzw. verschlüsselte Daten verwendet.

Im weiteren Verlauf der Arbeit erfolgte die Entwicklung einer vollständigen Datenträgeranalyse. Neben der Identifizierung von Verschlüsselung auf Datenträger- bzw. Partitionsebene umfasst die Analyse die Identifizierung verschlüsselter Anwendungsdateien. Dafür wurden exemplarisch Methoden zur Interpretation des Dateisystems NTFS implementiert, um unabhängig vom Fragmentierungsgrad einer Partition Cluster Dateien zuordnen zu können. Auf allen sonstigen unverschlüsselten Datenträgerbereichen erfolgt die Anwendung einer im Rahmen dieser Arbeit entwickelten Dateiwiederherstellung, welche anhand der statistischen Tests verschlüsselte bzw. zufällige Daten identifiziert und im defragmentierten Zustand vollständig extrahiert.

Die Evaluierung des Tools erfolgte anhand eigens erstellter Datenträgerimages. Dafür wurden 5 Szenarien entwickelt, welche in unterschiedlichen Analyseabläufen resultieren. Die Einhaltung der Analyseabläufe sowie eine Klassifizierung der Daten erfolgte anhand des entwickelten Tools.

2 Grundlagen

Dieses Kapitel beschreibt Grundlagen, welche für das weitere Verständnis dieser Arbeit notwendig sind. Dies umfasst insbesondere Grundlagen aus dem Bereich der Kryptographie, welche für die Datei- bzw. Datenträgerverschlüsselung relevant sind, sowie Grundlagen aus dem Bereich der Statistik.

2.1 Kryptographische Techniken

Bei Anwendung von kryptographischen Techniken wird ein Klartext mit Hilfe eines Verschlüsselungsverfahrens in einen Geheimtext konvertiert, welcher ohne Kenntnis eines Schlüssels nur mit sehr aufwändigen Maßnahmen entschlüsselt werden kann. Die Durchführung von Verschlüsselung kann anhand von symmetrischen und asymmetrischen Verschlüsselungsverfahren erfolgen. Während symmetrische Verfahren zum Ver- und Entschlüsseln den gleichen Schlüssel verwenden, erfolgt die Ver- und Entschlüsselung bei asymmetrischen Verfahren mit unterschiedlichen Schlüsseln [RSA78]. Auf diese beiden Verfahren und deren Relevanz im Bereich der Datenträgerverschlüsselung wird im Nachfolgenden näher eingegangen.

2.1.1 Asymmetrische Chiffren

Asymmetrischen Chiffren, auch Public-Key-Verschlüsselungsverfahren genannt, finden heutzutage hauptsächlich Anwendung im Bereich der Kommunikation, wie im E-Mail-Verkehr (z. B. S/MIME) oder bei kryptographischen Protokollen wie Secure Shell (SSH) oder Secure Sockets Layer (SSL)/Transport Layer Security (TLS). Dabei existiert für jeden Kommunikationsteilnehmer ein Schlüsselpaar: ein öffentlicher Schlüssel (public key), mit welchem verschlüsselt wird, und ein geheimer Schlüssel (private key), mit welchem entschlüsselt wird. Der öffentliche Schlüssel kann dabei über einen unsicheren Kanal an den Kommunikationspartner übermittelt und zur Verschlüsselung verwendet werden, während der geheime Schlüssel den empfangenen Geheimtext entschlüsselt und somit geheimgehalten werden muss. Mit dem öffentlichen Schlüssel ist eine Entschlüsselung nicht möglich. Für die Sicherheit des asymmetrischen Verfahrens ist es notwendig, dass die zugrundeliegende Berechnung des

Schlüssels praktisch unumkehrbar ist, da ansonsten der geheime Schlüssel aus dem öffentlichen Schlüssel berechnet werden kann.

Ein weit verbreitetes Public-Key-Verschlüsselungsverfahren ist RSA, welches mathematisch betrachtet auf dem Faktorisierungsproblem basiert. Während das Multiplizieren von Primzahlen einfach ist, gestaltet sich die Zerlegung einer großen Zahl in ihre Primfaktoren als sehr aufwändig. Allerdings resultiert aus der mathematischen Abhängigkeit von öffentlichem und privatem Schlüssel eine größere Schlüssellänge, was im Vergleich zur Schlüsselberechnung bei symmetrischen Verschlüsselungsverfahren mit höherem Rechenaufwand verbunden ist. Aus diesem Grund werden Public-Key-Verschlüsselungsverfahren hauptsächlich eingesetzt, um symmetrische Schlüssel für den eigentlichen Datenaustausch zu übertragen.

Public-Key-Verfahren finden allerdings auch Anwendung bei der Verschlüsselung von Daten auf Datenträgern. Dabei wird der zur eigentlichen Datenverschlüsselung eingesetzte symmetrische Schlüssel mittels öffentlichen Schlüssels verschlüsselt. Das Schlüsselpaar resultiert dabei aus einem Benutzerpasswort, welches oftmals mittels Hashverfahren auf eine bestimmte Länge, z. B. 1024 oder 2048 Bit, transformiert wird. Ohne Anwendung dieses Verfahrens entspräche das Benutzerpasswort dem symmetrischen Schlüssel und müsste somit den gleichen Sicherheitseigenschaften genügen [CGM12].

2.1.2 Symmetrische Chiffren

Bei symmetrischen Verfahren wird zum Ver- und Entschlüsseln der gleiche Schlüssel verwendet. Neben der Anwendung im Bereich der Kommunikation eignen sich diese Verfahren ebenfalls, um Daten performant z. B. auf Datenträgern verschlüsselt abzuspeichern. Die Sicherheit der Verfahren basiert dabei auf Anwendung komplexer Transformationen auf den Klartext, sodass ohne Kenntnis des Schlüssels ein Rückschluss auf den Klartext nur schwer durchführbar ist. Eine Möglichkeit zur Erlangung des Klartextes ist die Anwendung von Brute-Force-Attacken, welche jedoch bei geeigneter Länge und Wahl des Schlüssels praktisch nicht durchführbar sind [DR02]. Die Basiseigenschaften sowie Konstruktionsmethoden von symmetrischen Chiffren werden im Nachfolgenden näher erläutert.

2.1.2.1 Basiseigenschaften symmetrischer Chiffren

Klassische Verschlüsselungsverfahren bezeichnen einfache auf Substitution basierende Algorithmen, welche jede Klartexteinheit durch eine eindeutige Zuordnung in eine

Geheimtexteinheit übersetzen. Durch diese eindeutige Zuordnung lässt sich mittels einer Häufigkeitsanalyse auf die ursprüngliche Klartexteinheit schließen. Dabei werden die einzelnen Geheimtexteinheiten gezählt und ihre Häufigkeit notiert.

Aufgrund der spezifischen Häufigkeit von Buchstaben in einer Sprache kann vom Geheimtext auf den ursprünglichen Klartext geschlossen werden. In der deutschen Sprache beispielsweise tritt der Buchstabe E mit rund 17% am häufigsten auf. Kommt in einer verschlüsselten Nachricht eine Geheimtexteinheit mit etwa 17% vor, lässt sich diese dem E zuordnen. Der Geheimtext lässt sich dabei leichter entschlüsseln, je länger eine Nachricht ist, da die Genauigkeit der Häufigkeit mit der Länge der Nachricht steigt [Bau00].

Claude Shannon identifizierte 1949 in seinem Paper „Communication Theory of Secrecy Systems“ [Sha49] zwei Eigenschaften eines sicheren Kryptosystems. Diese bezeichnet er *Confusion* und *Diffusion*. Nach seiner Definition bezeichnet *Confusion* das Herstellen eines sehr komplexen Zusammenhangs zwischen Klar- und Geheimtext. Mittels *Diffusion* werden Redundanzen des Klartextes bei Konvertierung in einen Geheimtext eliminiert. Bei dieser Eigenschaft ändert sich der Geheimtext bei Änderung eines Klartextbits vollständig in einer unvorhersehbaren und pseudozufälligen Art und Weise. Die in der natürlichen Sprache auftretende Ungleichverteilung individueller Buchstaben wird durch eine größere Struktur des Geheimtextes umverteilt und ist somit schwieriger zu erkennen.

Substitution bezeichnet eine Regel zum Ersetzen von Klartextzeichen durch andere Zeichen. Diese Technik wurde als Mechanismus für *Confusion* identifiziert. Die Permutation, welche das Vertauschen oder Neuordnen von Klartextzeichen bezeichnet, ist eine Technik für *Diffusion*.

Nach Shannon entstehen durch eine geschickte Anwendung von Mechanismen für *Confusion* und *Diffusion* Daten, welche zufällig erscheinen. Aus den im Klartext enthaltenen Informationen resultiert ein gleichverteilter und stochastisch unabhängiger Geheimtext, welcher zwar mit spektralen Parametern übereinstimmt, jedoch keinerlei Redundanzen enthält und keinen Zusammenhang zwischen Datenelementen aufweist, wodurch statistische Angriffe ihre Wirkung verlieren. Kryptosysteme sollten demnach so ausgelegt werden, dass sich resultierende Geheimtexte nicht von Zufallsdaten unterscheiden.

Klassische Verschlüsselungsverfahren wurden eingesetzt, als elektronische Rechner noch nicht oder nur geringfügig vorhanden waren. Während der ursprüngliche Anwendungszweck von Verschlüsselungsalgorithmen ganze Buchstaben oder Buchstabengruppen waren, arbeiten moderne Kryptosysteme auf einzelnen Bits der Daten.

Dadurch vergrößert sich die Anzahl der möglichen Transformationen erheblich und es können Daten verarbeitet werden, welche keinen Text repräsentieren.

Heutzutage eingesetzte Verschlüsselungsverfahren konvertieren einen Klartext in einen unvorhersehbaren Geheimtext, sodass keine eindeutige Zuordnung zwischen Klar- und Geheimtexteinheiten existiert. Dabei unterscheidet man zwischen Strom- und Blockchiffren.

Bei Stromchiffren erfolgt die Verschlüsselung eines Datenstroms bitweise. Im Vorfeld wird dazu ein Schlüsselstrom generiert, welcher im Anschluss mit dem Klartext XOR-verknüpft wird. Bei einer perfekt sicheren Stromchiffre muss der Schlüssel ebenso lange sein wie der Klartext und zufällig ausgewählt werden. In der Praxis dient jedoch häufig ein Schlüssel fixer Länge als Saat für einen Pseudozufallszahlengenerator, welcher den effektiven Schlüsselstrom berechnet. Vorteile ergeben sich insbesondere in der Echtzeitdatenübertragung, da jedes Klartextzeichen sofort in ein Geheimtextzeichen übersetzt und übertragen werden kann [MH06].

Beim Einsatz von Blockchiffren wird ein zufälliger Schlüssel generiert, welcher einen Klartext fester Länge in einen Geheimtext gleicher Länge konvertiert. Durch die Anwendung von Betriebsmodi können Klartexte über Blocklängen hinaus unter Vewendung des gleichen Schlüssels verschlüsselt werden. Je nach Betriebsmodus entsteht ein Geheimtext mit hohem Sicherheitsniveau, da identische Klartextblöcke in unterschiedlichen Geheimtextblöcken resultieren [DR02]. Aufgrund der höheren Relevanz im Bereich der Verschlüsselung auf Datenträgern wird im nachfolgenden Abschnitt lediglich auf Blockchiffren näher eingegangen.

2.1.2.2 Blockchiffren

Eine Blockchiffre ist ein Verfahren, welches einen Klartextblock fester Länge in einen Geheimtextblock gleicher Länge unter Einfluss eines Schlüssels transformiert. Weitverbreitete Blockchiffren wie AES arbeiten mit logischen Verknüpfungen wie XOR-Operationen, Substitutionen und Permutationen [Dwo01].

Advanced Encryption Standard Der Advanced Encryption Standard (AES) wurde im Jahr 2000 als Nachfolger von Data Encryption Standard (DES) bekanntgegeben. AES wird auch nach seinen Entwicklern Joan Daemen und Vincent Rijmen „Rijndael“ genannt. Rijndael [DR02] besitzt eine variable Blockgröße von 128, 192 oder 256 Bit und eine variable Schlüssellänge von ebenfalls 128, 192 oder 256 Bit. Der einzige Unterschied zwischen Rijndael und AES ist die variable Blockgröße, welche AES auf 128 Bit einschränkt. Aufgrund des größeren Schlüsselraums ergibt sich

im Vergleich zum Vorgänger DES eine wesentlich höhere Sicherheit bei Brute-Force-Angriffen. In Abhängigkeit von der Schlüssellänge steigt zudem die Anzahl der Iterationen. Ein Evaluationskriterium von AES war die Geeignetheit als Zufallszahlengenerator [Sot99].

2.1.2.3 Betriebsmodi

Um Klartexte zu verschlüsseln, welche über die Blocklänge einer Chiffre hinausgehen, existieren unterschiedliche Betriebsmodi. Diese spezifizieren, wie die Chiffre auf den Klartext angewendet wird. Das National Institute of Standards and Technology (NIST) empfiehlt Vertraulichkeitsmodi für symmetrische Blockchiffren [Dwo01]. Die für Datenverschlüsselung auf Datenträgern relevanten Betriebsmodi sind insbesondere Cipher Block Chaining (CBC) sowie XEX-based tweaked-codebook mode with ciphertext stealing (XTS), welche im Nachfolgenden erläutert werden.

Cipher Block Chaining Um Redundanzen in Geheimtextblöcken zu vermeiden ohne einen neuen Schlüssel verwenden zu müssen, wird jeder Block nach der Verschlüsselung permutiert. CBC erreicht dies, indem ein Geheimtextblock mit dessen Vorgänger XOR-verknüpft wird. Daraus resultiert, dass zwei identische Klartextblöcke niemals in den gleichen Geheimtextblock übersetzt werden. Zur Einleitung des Verschlüsselungsprozesses benötigt CBC einen Initialisierungsvektor. Nachfolgende Abbildung verdeutlicht den Ver- und Entschlüsselungsprozess [Dwo01].

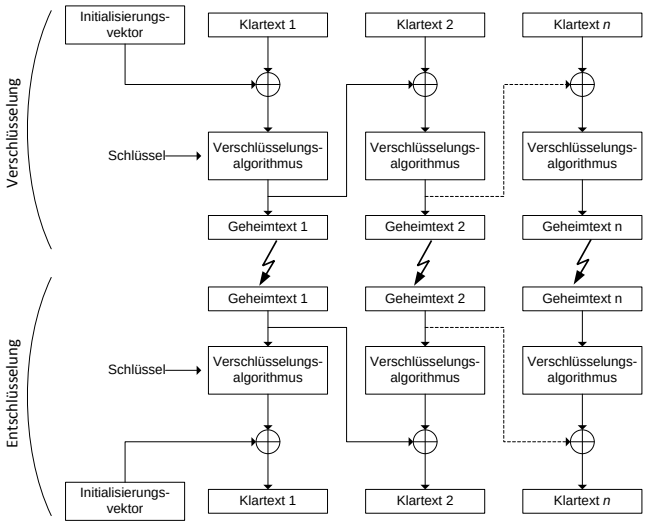

Abbildung 2.1: Cipher Block Chaining Ver- und Entschlüsselung

CBC eignet sich vorrangig zur Verschlüsselung von Dateien auf Datenträgern, wel-

che zum Lesezeitpunkt vollständig entschlüsselt vorliegen müssen. Der Nachteil von CBC zeigt sich bei Anwendung auf große Datenmengen, wie z. B. bei Verschlüsselung von Partionen oder Datenträgern. Aufgrund der Abhängigkeiten der Blöcke untereinander müsste bei Zugriff auf einen Datenträgerblock der Entschlüsselungprozess ab dem ersten Block initiiert werden, was zu großen Verzögerungen führt. Microsoft's Partitionsverschlüsselung Bitlocker verwendet AES-CBC mit einem *diffuser* [Fer06], welcher Datenträgereinheiten unabhängig voneinander verschlüsselt. Damit identische Klartexblöcke in unterschiedlichen Geheimtextblöcken resultieren sowie zur Gewährleistung der Authentizität, fließt die Sektornummer des entsprechenden Blocks mit in den jeweiligen Initialisierungsvektor ein.

XEX-based tweaked-codebook mode with ciphertext stealing Der XTS- Betriebsmodus [Dwo10] wird häufig in Verbindung mit AES zur Verschlüsselung ganzer Partitionen oder Datenträger eingesetzt. Bekannte Anwendungsbeispiele von XTS sind File Vault 2 (Mac OS X), Linux Unified Key Setup (LUKS) oder TrueCrypt.

AES-XTS wird als *tweakable blockcipher* bezeichnet und basiert auf der Konstruktion von Rogaway's Betriebsmodus Xor-encrypt-xor (XEX) [Rog04]. Die Verschlüsselung erfolgt dabei pro Datenträgerblock (beliebiger Größe, jedoch Vielfaches von 128 bit). Für jeden Block existieren dabei 2 Schlüssel, ein sogenannter *tweak key* zur Verschlüsselung eines *tweak value*, welcher sich aus dem Datenträgeroffset ableitet, und ein Schlüssel zur eigentlichen Datenverschlüsselung. Nachfolgende Abbildung verdeutlicht den Verschlüsselungsprozess [CGM12].

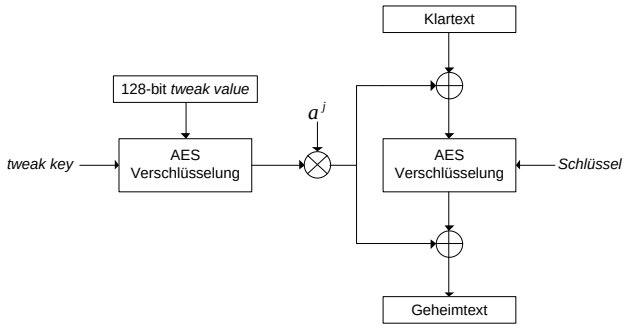

Abbildung 2.2: AES-XTS Datenträgerverschlüsselung

Zunächst wird der *tweak value* unter Verwendung des *tweak key* verschlüsselt. Dieser Vorgang erfolgt einmal pro Datenträgerblock. Anschließend wird der Datenträgerblock in 128 Bit Blöcke aufgeteilt und jedem dieser Blöcke eine sequentielle Nummer j (aufsteigend ab $j = 0$) zugewiesen. Der verschlüsselte *tweak value* wird nun um j Stellen nach links geshiftet und anschließend mit dem Klartext und dem resultierenden

Geheimtext XOR-verknüpft. Die Geheimtexte werden im Anschluss zusammenhängend auf dem Datenträger abgespeichert. Die Entschlüsselung erfolgt analog, lediglich die zweite AES-Verschlüsselung wird durch eine AES-Entschlüsselung ersetzt.

2.2 Güte von Zufallszahlen

Wie in Abschnitt 2.1.2.1 erwähnt, resultiert bei Anwendung moderner Verschlüsselungsverfahren ein Geheimtext, welcher einer Zufallsfolge gleicht. Betrachtet man das Erzeugen von Zufallszahlen durch einen Zufallszahlengenerator, so gelten folgende Anforderungen an die Zufallsfolge [Bau09]:

- Gleichverteilung: Die Zufallsfolge genügt der Gleichverteilung. Jeder Wert in einem bestimmten Intervall besitzt ungefähr die gleiche relative Häufigkeit.
- Unvorhersehbarkeit: Nach Erzeugung einer Zahl soll die nächste Zahl nicht vorhersehbar sein.
- Unabhängigkeit: Keine Zahl hängt von einer anderen ab.

Zur Unterscheidung von guten und weniger guten Zufallszahlengeneratoren existieren theoretische und empirische Tests. Während theoretische Tests am Generator selbst ansetzen, betrachten empirische Tests die erzeugten Zufallsdaten. Letzterer Ansatz lässt sich auf die Identifizierung verschlüsselter Daten übertragen, wobei vor allem Tests auf Gleichverteilung sowie Unabhängigkeit der Zahlenfolge von Interesse sind. Nachfolgend wird von zufälligen Daten oder Bytefolgen gesprochen, welche als Synonym für verschlüsselte oder zufällige Daten stehen. Wie der weitere Verlauf der Arbeit zeigt, ist keine Unterscheidung zwischen zufälligen und verschlüsselten Daten möglich, da diese dieselben statistischen Eigenschaften besitzen.

Nachfolgender Abschnitt beschäftigt sich zunächst mit einem historischen und anschließend mit einem heutzutage vielfach eingesetzten Erkennungsansatz.

2.2.1 Golomb's Zufallskriterien

Golomb's Zufallskriterien waren historisch betrachtet ein erster Versuch, notwendige Bedingungen aufzustellen, nach welchen eine Zufallszahlensequenz als pseudozufällig gilt [MOV97]. Diese Bedingungen sind allerdings nur hinreichend für diese Charakterisierung.

1. In jeder Periode gleicht die Anzahl aller Nullen annähernd der Anzahl aller Einsen (die Differenz beträgt maximal 1).

2. In jeder Periode besitzen mindestens die Hälfte aller Runs die Länge 1, mindestens ein Viertel aller Runs die Länge 2, mindestens ein Achtel aller Runs die Länge 4, usw. so lange die Anzahl der Runs 1 übersteigt. Als Run wird ein Abschnitt aufeinanderfolgender identischer Bits bezeichnet.

3. Die Autokorellationsfunktion besitzt genau zwei Werte. Sie ist definiert als $C(\tau) = \frac{A(\tau)-D(\tau)}{p}$, wobei $A(\tau)$ = Anzahl der übereinstimmenden Glieder pro Zyklus zwischen zwei Folgen, $D(\tau)$ = Anzahl der nicht übereinstimmenden Glieder pro Zyklus zwischen zwei Folgen und p die Anzahl der Stellen bezeichnet.

Es ist offensichtlich, dass sich diese Kriterien nur bedingt zur Charakterisierung von Zufallsfolgen eignen, da sie die Erfüllung genau vorgegebener Werte fordern, während man bei wirklich zufälligen Folgen lediglich die ungefähre Übereinstimmung mit diesen Werten beobachtet. Diese Kriterien reichen demnach nicht aus, um Zahlenfolgen zuverlässig als zufällig charakterisieren zu können, sie können jedoch als hinreichendes Kriterium dienen.

2.2.2 Entropietest

Eine heutzutage weit verbreitete Methode zur Erkennung von Verschlüsselung ist der Entropietest, welcher die Menge an Zufallsinformationen von Daten bestimmt.

Die Entropie nach Shannon [Sha48] definiert den mittleren Informationsgehalt eines Zeichensystems. Beobachtet man die möglichen Zeichen $x_i (i = 1, \ldots, n)$ einer Zufallsvariablen X, welche mit der Wahrscheinlichkeit p_i auftreten, so ist der Informationsgehalt I eines Zeichens x_i definiert als $I(x_i) = -log_2 p_i$. Die Zufallsvariable besitzt hierbei die Eigenschaft, dass die Wahrscheinlichkeit für das Auftreten eines bestimmten Zeichens zu einem Zeitpunkt immer gleich p_i und völlig unabhängig vom Zeitpunkt und zuvor gesendeten Zeichen ist. Liegt die Auftrittswahrscheinlichkeit eines Zeichens x_i nahe bei 0, resultiert ein hoher Informationsgehalt. Tritt häufig das selbe Zeichen x_i auf, geht der Informationsgehalt gegen 0, die Unsicherheit über das nächste zu lesende Zeichen sinkt somit. Die Entropie einer Zufallsvariablen X ist nun definiert als

$$H(X) = -\sum_{i=1}^{n} p_i \cdot log_2 p_i$$

Bei der Betrachtung der oberen und unteren Grenze von $H(X)$ stellt sich heraus, dass die Entropie maximal ist, wenn alle Zeichen mit gleicher Wahrscheinlichkeit auftreten. Die Entropie beträgt dann

$$H(X) = log_2 n$$

Treten die Zeichen x_i mit gleicher Wahrscheinlichkeit auf, spricht man von Gleichverteilung. Wie bereits zuvor erwähnt, ist dies eine Anforderung, welche von Zufallsfolgen erfüllt werden muss. Die Entropie eignet sich somit grundsätzlich zur Feststellung gleichverteilter Daten.

Der Begriff Entropie findet auch im Bereich der verlustlosen Kompression Anwendung und beschreibt, wie viele Bits an Information in einer Nachricht eigentlich vorhanden sind. Mittels der Entropiekodierung werden jedem einzelnen Zeichen x_i unterschiedlich lange Folgen von Bits zugeordnet, um die Gesamtzahl der benötigten Bits zu minimieren. Eine weitere Minimierung wird durch Betrachtung aufeinanderfolgender Zeichen erreicht. Durch die Umverteilung der Auftrittswahrscheinlichkeiten können somit Daten entstehen, welche ebenfalls eine erhöhte Entropie besitzen [GN96].

2.2.2.1 Entropie als Identifikationsmerkmal

Zur Überprüfung der Geeignetheit der Entropie als Identifikationsmerkmal von Verschlüsselung werden die Entropien verschlüsselter Nutzdaten und komprimierter Daten verglichen. Die Daten werden dabei byteweise betrachtet, wodurch eine maximale Entropie von $H(X) = log_2 2^8 = 8$ Bits/Byte resultiert.

Die Testergebnisse 100 verschlüsselter Dateien, repräsentiert durch TrueCrypt-Container, und 100 ZIP-komprimierten Dateien (Dateitypendefinition siehe S. 31) sind nahezu identisch: während Verschlüsselung eine durchschnittliche Entropie von 7,9998857 Bits/Byte besitzt, liegt die durchschnittliche Entropie bei komprimierten Daten bei 7,9961532 Bits/Byte. Mittels des Entropietests lässt sich somit keine eindeutige Aussage über Verschlüsselung oder Komprimierung treffen.

Etablierte Software wie EnCase nutzt diesen Test, um verschlüsselte Daten zu identifizieren. In dieser Arbeit wurde ein Ansatz entwickelt, welcher eine zuverlässigere Klassifizierung ermöglicht, indem neben der Gleichverteilung auf weitere Eigenschaften verschlüsselter Datenströme überprüft wird. Diese Eigenschaften lassen sich durch kombinatorische Überlegungen herleiten und als permutierte Zahlenfolgen in Daten wiederfinden. Als Grundlage der Untersuchungen dienen Häufigkeitsanalysen, welche

diese Permutationen zählen und anschließend überprüfen, ob die beobachtete Anzahl annähernd mit der hypothetischen Anzahl übereinstimmt. Die Durchführung solcher Vergleiche kann anhand von statistischen Hypothesentests erfolgen.

2.2.3 Statistische Tests

Es existieren verschiedene statistische Tests, welche eine vermeintliche Zufallsfolge mit einer wirklichen Zufallsfolge vergleichen und entsprechend bewerten. Dazu dienen die eingangs genannten Eigenschaften einer Zufallsfolge, welche sich in Wahrscheinlichkeiten ausdrücken lassen. Daraus lässt sich ein Referenzwert, welcher perfekte Zufallsfolgen repräsentiert, vor Durchführung des eigentlichen Tests bestimmen.

Im Zusammenhang mit dieser Arbeit bestehen Ansätze, welche die Erkennung einzelner verschlüsselter Dateiformate ermöglichen. Ein Ansatz behandelt die Identifizierung von TrueCrypt-Containern [SSY12]. Dabei wird mittels einer Häufigkeitsanalyse und der Chi-Quadrat-Verteilung überprüft, ob die Datenbytes der Verteilung verschlüsselter Daten genügen. Darüber hinaus werden bestimmte Dateiformate anhand ihrer Headersignatur aussortiert sowie die Dateigröße, welche bei TrueCrypt-Containern Vielfaches von 512 beträgt, als Kriterium herangezogen. Der Ansatz eignet sich zur Erkennung von TrueCrypt-Containern, wenn sich diese nicht in komprimierten Archiven befinden und die Dateigröße nicht manipuliert wurde.

Ein weiterer Ansatz untersucht mittels statistischer Tests Binärporgramme, um darin möglicherweise enthaltene Schadsoftware zu erkennen [GWH11].

Drüber hinaus existieren eine Reihe von Testsuiten, welche die Güte von Zufallszahlengeneratoren messen. Ein prominentes Beispiel ist die statistische Test-Suite für Zufalls- sowie Pseudozufallszahlengeneratoren für kryptographische Anwendungen des NIST [RSN10]. Da Pseudozufallszahlengeneratoren häufig in kryptographischen Anwendungen und somit sicherheitsrelevanten Bereichen eingesetzt werden, ist es sehr wichtig, den Anforderungen von Zufallszahlen zu genügen. NIST beschreibt statistische Tests, von denen es ausgeht, dass diese Abweichungen von Zufälligkeit in Zahlenfolgen erkennt. Der Schwerpunkt liegt auf der Betrachtung von Binärfolgen, da im Bereich der IT-Sicherheit oftmals nur kurze Bitfolgen von Interesse sind. Viele der Tests können nicht performant auf die Analyse von großen Datenmengen angewendet werden, sodass auf größere Einheiten zurückgegriffen werden muss.

Der amerikanische Informatikprofessor Donald E. Knuth beschreibt im zweiten Teil seiner Buchreihe *The Art of Computer Programming* [Knu81] theoretische als auch statistische Eigenschaften guter Pseudozufallszahlengeneratoren. Einige der von

Knuth entwickelten Tests werden in dieser Arbeit dazu verwendet, um verschlüsselte Daten als solche zu identifizieren. Die Tests basieren auf der Messung von Verteilungen von Bytes und Bytesequenzen und lassen sich effizient realisieren. Die Durchführung der Überprüfung erfolgt anhand statistischer Hypothesentests.

2.3 Durchführung statistischer Tests

Im Allgemeinen wird bei einem statistischen Test eine Nullhypothese (H_0) getestet. In dieser Arbeit beschreibt die Nullhypothese die Eigenschaft einer Bytefolge, zufällig und somit möglicherweise verschlüsselt zu sein. Im Zusammenhang mit der Nullhypothese steht die Gegenhypothese (H_1), welche die Eigenschaft beschreibt, nicht zufällig bzw. verschlüsselt zu sein. Nach jedem Test wird abschließend entschieden, ob die Nullhypothese angenommen oder abgelehnt wird, die Folge ist also höchstwahrscheinlich zufällig oder nicht.

Für jeden angewandten Test muss im Vorfeld eine Kenngröße ermittelt werden, welche die Verteilung möglicher Werte repräsentiert. Eine theoretische Verteilung wird durch mathematische Methoden bestimmt und dient als Referenz. Anhand dieser Referenzverteilung wird ein kritischer Wert bestimmt, welcher von der Teststatistik nicht überschritten werden darf.

Die Teststatistik ergibt sich aus der Durchführung des Tests und dient im Anschluss als Entscheidungskriterium über Annahme oder Ablehnung der Nullhypothese. Bei zufälligen Daten resultiert eine Teststatistik, welche mit einer sehr geringen Wahrscheinlichkeit (z. B. 0,1%) den kritischen Wert überschreitet. Sollte dieser überschritten werden, ist die Zufallsfolge aus Sicht des Hypothesentests nicht natürlich und wird demnach als falsch bzw. nicht-verschlüsselt eingestuft.

Zusammenfassend lässt sich anhand des statistischen Hypothesentests feststellen, ob eine Zahlenfolge höchstwahrscheinlich zufällig (H_0 wird angenommen) oder nichtzufällig (H_0 wird abgelehnt) ist.

2.3.1 Fehlergrößen

Die statistische Entscheidung beinhaltet 2 mögliche Fehlertypen, welche in Tabelle 2.1 aufgeführt sind. Der *Type-I-Error* bezeichnet den prozentualen Anteil des Ablehnens der Nullhypothese, obwohl diese wahr ist. Die Wahrscheinlichkeit des *Type-I-Error* wird als Signifikanzniveau α bezeichnet und liegt der Kryptographie im Intervall α = [0,01;0,1] [Dwo01]. Das Annehmen einer Nullhypothese, obwohl die Zufallsfolge

nicht-zufällig ist, wird als *Type-II-Error* β bezeichnet. Im Gegensatz zu α ist β keine fixe Größe sondern ergibt sich aus der jeweils berechneten Teststatistik.

Tabelle 2.1: Type-I- und Type-II-Error

Wahre Situation	Schlussfolgerung	
	H_0 wird angenommen	H_1 wird angenommen
Daten sind zufällig (H_0 ist wahr)	Kein Fehler	Type-I-Error (false positive)
Daten sind nicht zufällig (H_1 ist wahr)	Type-II-Error (false negative)	Kein Fehler

Besitzt die Teststatistik den Wert S und der kritische Wert den Wert t, dann beträgt die Wahrscheinlichkeit für einen *Type-I-Error* $P(S > t||H_0$ ist wahr$) = P(H_0$ wird abgelehnt $||H_0$ ist wahr$)$ und die Wahrscheinlichkeit für einen *Type-II-Error* $P(S \leq t||H_0$ ist nicht wahr$) = P(H_0$ wird angenommen $||H_0$ ist nicht wahr$)$. Die Teststatistik wird nun dazu verwendet, um einen p-Wert zu bestimmen, welcher als Überschreitungsmaß für die Nullhypothese gilt. Auf die Erzeugung von Zufallszahlen bezogen ist der p-Wert die Wahrscheinlichkeit, mit welcher der Zufallszahlengenerator eine Zufallsfolge generiert, die weniger zufällig ist als die getestete Zufallsfolge. Liegt der p-Wert nahe an 1, scheint die Zufallsfolge perfekt zufällig zu sein. Ein p-Wert gegen 0 deutet auf nicht-zufällig hin. Ist der p-Wert $< \alpha$, wird die Nullhypothese abgelehnt [RSN10].

Beispiel: Beträgt $\alpha = 0{,}01$, bedeutet dies, dass 1 Zufallsfolge aus 100 als nicht-zufällig eingestuft wird, obwohl diese zufällig ist. Bei einem p-Wert $\geq 0{,}01$ wird die Zufallsfolge mit einer Sicherheit von 99% als zufällig angesehen, bei einem p-Wert $< 0{,}01$ wird diese mit einer Sicherheit von 99% als nicht-zufällig angesehen.

In Bezug auf die Klassifizierung von Daten werden in dieser Arbeit anstelle von *Type-I-Error* und *Type-II-Error* die Bezeichnungen false positives bzw. false negatives verwendet. False positives kennzeichnen somit Daten, welche als nicht-verschlüsselt eingestuft wurden, obwohl diese verschlüsselt sind. False negatives kennzeichnen Daten, welche als verschlüsselt eingestuft wurden, obwohl diese nicht-verschlüsselt sind.

2.3.2 Chi-Quadrat-Test

Knuth bezeichnet den Chi-Quadrat-Test (χ^2-Test) als eine Basismethode und einen der bestbekanntesten statistischen Tests überhaupt [Knu81]. Der Test vergleicht beobachtete mit erwarteten Häufigkeiten und testet, ob eine Zahlenfolge einer bestimmten Wahrscheinlichkeitsverteilung genügt. Die Beobachtungen in der Bytefolge werden in n möglichen Kategorien mit jeweils h_k Elementen zusammengefasst. Nun gilt die Überlegung, wie viele Beobachtungen in der jeweiligen Kategorie k liegen müssten, damit die Bytefolge die hypothetische Verteilung besäße. Dies geschieht über die Berechnung der Wahrscheinlichkeit p_k, mit welcher eine Beobachtung in die jeweilige Kategorie k fällt, multipliziert mit der Anzahl aller Beobachtungen m. Die Prüfgröße für den Test ergibt sich nun aus

$$\chi^2 = \sum_{k=1}^{n} \frac{(h_k - mp_k)^2}{mp_k}$$

Hierbei lässt sich erkennen, dass Kategorien mit abweichenden Vorkommnissen eine hohe Gewichtung erhalten und den χ^2-Wert erhöhen. Durch das Quadrieren erhält man lediglich positive Terme, wodurch sich Diskrepanzen nicht ausgleichen können. Je geringer demnach die Abweichung zwischen der realen Verteilung über den Kategorien und der theoretischen Verteilung ist, desto geringer ist der Wert von χ^2 und desto eher ist die Bytefolge entsprechend verteilt.

Ziel ist es nun, mittels der Testgröße χ^2 eine Hypothese

$$H_0 : p_k = \frac{h_k}{m}$$

gegen eine andere Hypothese

$$H_1 : p_k \neq \frac{h_k}{m}$$

zu testen. Eine angenäherte Verteilung der Testgröße ergibt sich aus dem folgenden theoretischen Hilfsmittel: Wenn die Hypothese über die Wahrscheinlichkeitsverteilung zutrifft, strebt die Verteilung gegen eine χ^2-Verteilung. Je nach Abweichung wird H_0 angenommen oder abgelehnt, abhängig von der Schranke c. Diese ergibt sich aus der zugelassenen Wahrscheinlichkeit für einen *Type-I-Error*, dem Testniveau $\alpha > 0$ und der Anzahl der Freiheitsgrade υ. Diese bezeichnen die Anzahl der frei wählbaren, voneinander unabhängigen Ausprägungen einer Zufallsvariablen.

In Abhängigkeit von υ und der berechneten Testgröße χ^2 lässt sich nun die Über-

schreitungswahrscheinlichkeit $p = [0; 1]$ berechnen. Diese lässt sich dazu nutzen, um unabhängig von der Schranke c ein Intervall zu definieren, in welchem H_0 angenommen oder abgelehnt wird. Darüber hinaus ist mittels des p-Wertes ein Vergleich unterschiedlicher Testwerte möglich, selbst bei unterschiedlichem v.

Knuth schlägt vor, für alle Freiheitsgrade v folgende Werte anzunehmen:

- Die Hypothese H_0 wird akzeptiert, wenn anhand der berechneten χ^2-Statistik $p = (0,05; 0,95)$ resultiert.
- Die Hypothese H_0 wird verworfen, wenn anhand der berechneten χ^2-Statistik $p \leq 0,05$ bzw. $p \geq 0,95$ resultiert.

Bei Anwendung des χ^2-Test auf kurze Bytefolgen entstehen false positives. Knuth empfiehlt daher eine minimale Anzahl von $m \cdot p_k = 5$ erwarteten Elementen in jeder Kategorie. Je größer m ist, desto besser lässt sich globale Zufälligkeit feststellen, während lokale Abweichungen geglättet werden.

NIST hingegen schlägt in [Dwo01] vor, H_0 abzulehnen, wenn aus der berechneten χ^2-Statistik $p \leq 0,01$ resultiert.

2.4 Zusammenfassung

Dieses Kapitel führte die für das weitere Verständnis dieser Arbeit notwendigen Grundlagen auf. Dies sind zunächst die Basiseigenschaften symmetrischer Kryptosysteme, woraus hervorgeht, dass eine verschlüsselte Bytefolge zufälligen Daten gleicht. Zufällige Daten besitzen dabei offensichtliche Eigenschaften wie z. B. Gleichverteilung und Unabhängigkeit der beobachteten Bytes. Mit dem heute vielfach eingesetzten Entropietest lässt sich die Gleichverteilung von Daten feststellen. Dabei gilt je höher die Entropie, desto gleichverteilter die Daten.

Der Ansatz in dieser Arbeit sieht vor, erweiterte Eigenschaften zu überprüfen, um die Anzahl der durch den Entropietest fälschlicherweise als verschlüsselt bzw. zufällig eingestuften Daten zu reduzieren. Diese Eigenschaften ergeben sich aus kombinatorischen Überlegungen und können in hypothetische Häufigkeiten überführt werden. Diese Häufigkeiten lassen sich annähernd in wirklichen Zufallsfolgen wiederfinden. Zur Überprüfung dienen statistische Hypothesentests.

Vor Anwendung eines statistischen Hypothesentests wird eine Kenngröße ermittelt, welche die vorliegende Bytefolge repräsentiert. Der Chi-Quadrat-Test berechnet dabei die Abweichungen hypothetischer und beobachteter Häufigkeiten. Aus dem Test

resultiert das Überschreitungsmaß p, welches die Wahrscheinlichkeit angibt, dass eine zufällige Folge existiert, welche weniger zufällig als die derzeit beobachtete Folge ist. Umso größer p demnach ist, desto zufälliger ist die Folge. Der *Type-I-Error* α liefert eine Schranke nach unten. Ist $p < \alpha$, wird die Nullhypothese abgelehnt, die Folge ist demnach nicht zufällig.

Die Anwendung statistischer Tests erfolgt heutzutage bereits zur Feststellung der Güte von Zufallszahlengeneratoren. Einige der von Knuth entwickelten Tests werden in den nachfolgenden Abschnitten auf die Erkennung verschlüsselter Daten übertragen.

3 Algorithmen zur Datenanalyse

Um verschlüsselte Daten möglichst genau als solche erkennen zu können, wird nun auf die im Grundlagenkapitel aufgeführten Eigenschaften von Block- sowie Stromchiffren zurückgegriffen.

3.1 Identifikation anhand der Blockgröße

Ein Identifikationsmerkmal von Blockchiffren stellt die Blockgröße des verwendeten Verschlüsselungsalgorithmus dar. Beträgt die Größe eines Datenfragmentes ein Vielfaches der Blockgröße eines bekannten Verschlüsselungsverfahrens, lässt sich dieses als verdächtig einstufen. Dementsprechend ist dieses Verfahren nicht auf Stromchiffren anwendbar, weil diese bit- oder byteweise Verschlüsseln [MH06]. Nachfolgende Tabelle führt weit verbreitete Blockchriffren auf [**?**].

Tabelle 3.1: Weit verbreitete Blockchiffren

Abk.	Bezeichnung	Blocklänge (Bit)
AES	Advanced Encryption Standard	128
DES	Data Encryption Standard	64
3DES	Triple DES	64
-	Blowfish	64
CAST	Auch CAST5 oder CAST-128	64
IDEA	International Data Encryption Algorithm	64
RC2	Rivest Cipher 2	64
RC5	Rivest Cipher 5	32, 64, 128

Nach Tabelle 3.1 sind die für die Identifizierung von Verschlüsselung relevanten Blockgrößen 32, 64, sowie 128 Bit.

Die Identifikation von Verschlüsselung anhand der Blockgröße scheint offensichtlich nicht erfolgsversprechend zu sein. Zum einen ist mit false negatives zu rechnen, da jede Dateigröße einer Datei Vielfaches einer Blockgröße sein kann. Zum anderen lässt sich aus antiforensischer Sicht die Identifizierung durch Hinzufügen eines einzigen

Bits umgehen. Der Einsatz einer solchen Methode ist aufgrund dieser Tatsache nicht geeignet und wird nicht weiter betrachtet.

3.2 Identifikation anhand der Datenstruktur

In diesem Abschnitt werden Tests beschrieben, welche zur Identifizierung von Zufallszahlen beitragen. Da für Zufälligkeit keine Definition existiert, lässt sich nicht endgültig klären, ob eine zufällige Folge tatsächlich vorliegt [VANK95]. So können durch das Testen bestimmter Eigenschaften Zahlenfolgen zwar als nicht zufällig eingestuft werden, das Bestehen mehrerer Tests jedoch ist keine hinreichende Bedingung dafür, dass die Zahlenfolge tatsächlich einer wahren Zufallsfolge entspricht.

Bei der Anwendung der Tests auf Daten lassen sich Zufallszahlen aus den Bytes einer Bytefolge erzeugen. Dadurch resultiert eine Zahlensequenz mit Zahlen im Intervall [0,255]. Die Anwendung der Tests soll zunächst auf Clusterebene erfolgen. Somit müssen die Tests auf kurze Bytefolgen anwendbar sein. Im Nachfolgenden wird eine Clustergröße von 4096 Bytes, einer heutzutage typischen Clustergröße, angenommen.

3.2.1 Tests auf Gleichverteilung

Eine Anforderung an Zufallsdaten ist eine Gleichverteilung der in der Bytefolge beobachteten Bytes. Dies lässt sich z. B. anhand einer Häufigkeitsanalyse überprüfen. Diese und weitere Tests werden im Nachfolgenden beschrieben und auf ihre Geeignetheit überprüft.

3.2.1.1 Frequenzanalyse

Die Grundvoraussetzung einer Zufallsfolge ist, dass alle möglichen Zeichen mit gleicher Wahrscheinlichkeit auftreten. Die Frequenzanalyse ermittelt die Verteilung der in den Daten beobachteten Bytes und überprüft, ob diese mit einer wahren Zufallsfolge übereinstimmt. Die Ausprägungen $x_i (i = 1, \ldots, n)$ der Zufallsvariable X entsprechen den theoretisch zu beobachtenden Zeichen. Die theoretische Auftrittswahrscheinlichkeit beträgt

$$P(X = x) = \frac{1}{n}$$

Der Test ordnet jedes beobachtete Zeichen einer Kategorie zu. Die Kategorie entspricht dabei dem Dezimalwert des Zeichens. Nach Abschluss der Zuordnungen wird die beobachtete Verteilung dem χ^2-Test mit $\upsilon = n - 1$ Freiheitsgraden unterzogen.

Die beobachtete Folge gilt als gleichverteilt, wenn $p > \alpha$.

Der Test lässt sich auf die Untersuchung von n-Grammen ausweiten. Dabei werden zwei (Bigramm, 2-Gramm), drei (Trigramm, 3-Gramm) oder n aufeinanderfolgende Zeichen des Alphabets Ω als jeweils eine Ausprägung x der Zufallsvariablen X betrachtet. Die Auftrittswahrscheinlichkeit beträgt dabei $P(X = x) = \frac{1}{|\Omega|^n}$. Um dem Durchführungskriterium gemäß Knuth zu genügen, welches mindestens 5 hypothetische Beobachtungen pro Kategorie fordert, muss die Bytesequenz mindestens $5 \cdot |\Omega|^n$ groß sein. Bei der Anwendung auf Daten ergibt sich somit bei der byteweisen Betrachtung eine Mindestgröße von 1280 Bytes, bei einer Betrachtung von 2-Grammen eine Mindestgröße von 655360 Bytes. Für eine Anwendung auf Clusterebene mit einer Clustergröße von 4096 Bytes eignet sich somit nur die byteweise Betrachtung.

3.2.1.2 Arithmetisches Mittel

Das Arithmetische Mittel $\overline{x}$ beschreibt den Mittelwert, welcher als Quotient aus der Summe aller Werte einer Folge geteilt durch die Anzahl der Werte resultiert.

$$\overline{x}_{arith} = \frac{1}{n} \sum_{i=1}^{n} x_i$$

Bei einer Gleichverteilung aller beobachteten Werte resultiert bei der byteweisen Betrachtung ein empirischer Erwartungswert von 127,5.

3.2.1.3 Anwendung des Arithmetischen Mittel

Nachfolgendes Beispiel mit $\Omega = \{1, 2, \ldots, 10\}$ verdeutlichen die Geeignetheit vorangehender Methoden zur Feststellung der Gleichverteiltheit:

$$1, 2, 3, 4, 5, 6, 7, 8, 9, 10$$

$$5, 5, 5, 5, 5, 5, 2, 3, 10, 10$$

Die erste Folge genügt einer gleichverteilten Folge mit $\overline{x}_{arith} = 55$, die zweite Folge ist offensichtlich nicht gleichverteilt mit $\overline{x}_{arith} = 55$. Demnach existiert für ein gegebenes arithmetisches Mittel keine eindeutige Verteilung. Ausserdem schwankt das arithmetische Mittel bei annähernder Gleichverteilung. Dabei gilt ein größeres Mittel bei Abweichungen größerer Dezimalwerte und kleineres Mittel bei Abweichungen kleinerer Dezimalwerte. Für die Feststellung der Gleichverteilung wird demnach die klassische Frequenzanalyse bevorzugt, welche die Abweichung von Vorkommnissen

und nicht absolute Werte betrachtet. Das Beispiel zeigt allerdings auch, dass eine gleichverteilte Folge nicht zwangsläufig zufällig sein muss, sodass dieser Test als Einstufungskriterium nicht ausreicht.

3.2.2 Tests auf stochastische Unabhängigkeit

Nachfolgende Tests überprüfen weitere statistische Eigenschaften, welche neben der Gleichverteilung von zufälligen Daten erfüllt werden müssen. Dies sind zunächst offensichtliche Eigenschaften, wie z. B. eine starke Streuung aufeinanderfolgender Zahlen. Darüber hinaus gibt es weitere Eigenschaften, welche sich durch kombinatorische Überlegungen als Urnenexperimente (Ziehen mit Zurücklegen) herleiten und in Wahrscheinlichkeiten ausdrücken lassen. Aus den Urnenexperimenten resultieren bestimmte Bytesequenzen, welche es gilt, in der Bytefolge zu beobachten und zu zählen. Der Chi-Quadrat-Test berechnet im Anschluss die Abweichung beobachteter und hypothetischer Häufigkeiten. Als Grundannahme für jeden Test gilt die Gleichverteilung, womit jedes Zeichen mit gleicher Wahrscheinlichkeit auftritt. Es ergibt sich eine große Anzahl an Tests, welche jedoch unterschiedlich große Datenmengen erfordern, um Knuth's Kriterium für die Durchführung des Chi-Quadrat-Tests zu genügen. Darüber hinaus sind die Tests unterschiedlich aufwändig in ihrer Realisierung. Die nachfolgend betrachteten Tests sind etablierte Tests und lassen sich effizient auch auf kleine Stichproben anwenden.

3.2.2.1 Lückentest

Der Lückentest überprüft, ob aufeinanderfolgende Zahlen einer bestimmten Streuung unterliegen. Dabei erwartet man bei kleinen Auftrittswahrscheinlichkeiten eine starke Streuung. Die Überprüfung dieser Eigenschaft lässt sich als Bernoulli-Experiment formulieren. Diese stochastische Modellierung betrachtet ein Zufallsexperiment mit genau zwei Ereignissen, welche beim Lückentest 2 Intervalle darstellen. Beobachtet werden nun die Anzahl aufeinanderfolgender Zeichen, welche im gleichen Interval liegen. Ein solches Ereignis wird als Run bezeichnet und die entsprechende Länge notiert.

Weisen die Zahlen untereinander keine Abhängigkeiten auf, dann unterliegen die Run-Längen einer geometrischen Verteilung. Der Einsatz der geometrischen Verteilung erfolgt z. B. bei der Analyse von Wartezeiten bis zum Eintreffen eines bestimmten Ereignisses. In diesem Fall entspricht die Wartezeit der Länge eines Runs. Abbildung 3.1 zeigt die Einteilung des Wertebereichs in zwei gleich große Bernoulli-

Ereignisräume. Eine sequentielle Betrachtung der Bytes liefert unterschiedlich lange Runs, wobei ein Run unterbrochen wird, wenn ein Byte im jeweils anderen Intervall liegt. Die Längen der Runs entsprechen den Ausprägungen $x_i (i = 1, \ldots, n)$ der Zufallsvariablen X.

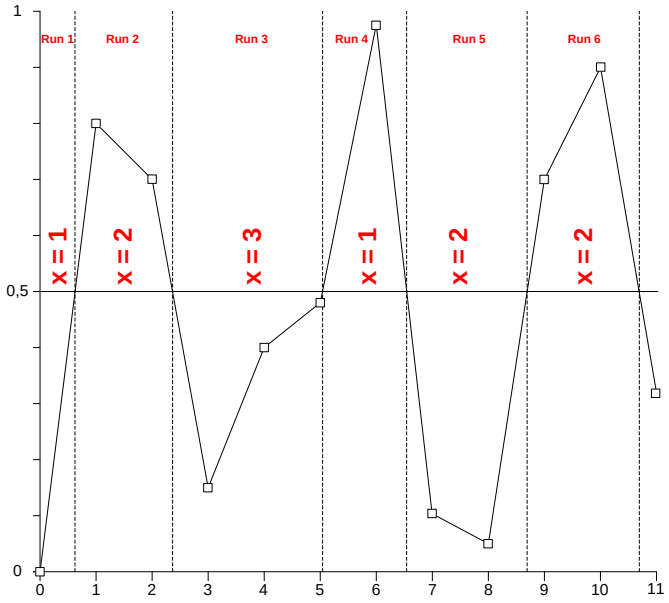

Abbildung 3.1: Beobachtung der Streuung aufeinanderfolgender Zahlen

Durchführung des Tests Die Durchführung des Lückentests erfordert nachfolgende Schritte.

- Vor Durchführung des Tests wird der gesamte Zahlenraum in zwei Intervalle eingeteilt, welche den zwei Elementarereignissen eines Bernoulli-Experiments entsprechen. In Abbildung 3.1 erfolgte die Einteilung in zwei gleich große Intervalle. Die Wahrscheinlichkeit, dass ein Byte in einem dieser Bereiche liegt, beträgt somit $p = 0,5$.
- Die Wahrscheinlichkeit für einen Run beträgt $P(X = x) = p(1 - p)^x$.
- Die Bytes der Zahlenfolge werden sequentiell betrachtet. Die Länge eines Runs ergibt sich aus der Anzahl aufeinanderfolgender Zahlen, welche im gleichen Interval liegen.
- Die beobachteten Run-Längen werden in entsprechende Kategorien eingeordnet. Nach Abschluss der Zuordnungen wird die beobachtete Verteilung einem χ^2-

Test unterzogen mit $v = n-1$. Die beobachtete Folge genügt bzgl. der Streuung einer echten Zufallsfolge, wenn $p > \alpha$.

Die Aussagekraft dieses Tests kann durch Ausführung mit unterschiedlichen Intervallen erhöht werden. Bei der Wahl der Anzahl der zu betrachtenden Kategorien sowie der Auswahl der Intervalle muss darauf geachtet werden, dass das Kriterium von Knuth, mindestens 5 hyptothetischen Beobachtungen pro Kategorie zu genügen, eingehalten wird. Nachfolgend wird die Anzahl benötigter Bytes in Abhängigkeit von der Run-Länge x aufgeführt, um 5 hypothetische Beobachtungen pro Kategorie zu erhalten. Der Zahlenbereich wurde in zwei gleich große Intervalle unterteilt, sodass $p = 0,5$ gilt.

x	1	2	3	4	5	6	>7
$P(X = x)$	1/4	1/8	1/16	1/32	1/64	1/128	130/256
# Ereignisse	20	40	80	160	320	640	10
# Bytes	20	80	240	640	1600	3840	>80

Die Durchführung des Lückentests mit $p = 0,5$ und 7 Kategorien erfordert somit eine Mindestgröße von 3840 Bytes, um in jeder der 7 Kategorien mindestens 5 hypothetische Beobachtungen zu erhalten. Mit diesen Parametern kann der Lückentest auf Clusterebene angewendet werden.

3.2.2.2 Poker-Test

Der Poker-Test untersucht kurze Folgen von Zufallszahlen k (Blatt) auf dessen Anzahl x verschiedener Werte d (Karte). Mithilfe dieses Verfahrens lassen sich Korrelationen zwischen jeweils k aufeinanderfolgenden Werten aufdecken. Dabei können u.a. folgende Muster beobachtet werden:

Alle verschieden:	abcde	Full House:	aaabb
Ein Paar:	aabcd	Vier einer Sorte:	aaaab
Zwei Paare:	aabbc	Fünf einer Sorte:	aaaaa
Drei einer Sorte:	aaabc		

Knuth [Knu81] beschreibt eine vereinfachte Version des Poker-Tests, welche lediglich die Anzahl unterschiedlicher Werte in einer Untersequenz notiert. Somit ergeben sich fünf verschiedene Kategorien:

5 verschieden	=	alle unterschiedlich
4 verschieden	=	ein Paar
3 verschieden	=	zwei Paare oder ein Drilling
2 verschieden	=	Full House oder ein Vierling
1 verschieden	=	ein Fünfling (im echten Poker nicht möglich)

Die Ausprägungen $x_i (i = 1, \ldots, 5)$ der Zufallsvariablen X entsprechen den unterschiedlichen Pokerkategorien.

Durchführung des Tests Die Durchführung des Poker-Tests erfordert nachfolgende Schritte.

- Unterteilung der Zahlenfolge in Untersequenzen der Länge k.
- Ermittlung der Anzahl unterschiedlicher Werte einer Untersequenz und Einordnung in entsprechende Kategorie.
- Anwendung des χ^2-Tests mit $\upsilon = k - 1$ und folgender Auftrittswahrscheinlichkeit:

$$P(X = x) = \frac{d(d-1)\ldots(d-x+1)}{d^k} \left\{ {k \atop x} \right\}$$

$\left\{ {k \atop x} \right\}$ bezeichnet die Stirling-Funktion zweiter Art und somit die Anzahl der Möglichkeiten, k Elemente in x nichtleere Teilmengen zu partitionieren [Car03].
Für den Poker-Test wird $k = 5$ verwendet, sodass nachfolgende Stirling-Zahlen resultieren.

x	1	2	3	4	5
$\left\{ {5 \atop x} \right\}$	1	15	25	10	1

- Die beobachtete Folge genügt einer Zufallsfolge, wenn $p > \alpha$.

Die Durchführung des Tests mit $d = 256$ und den 5 aufgeführten Kategorien erfordert eine sehr große Datenmenge von mindestens $1,07 \cdot 10^{11}$ Bytes, um mindestens 5 hypothetische Beobachtungen in jeder Kategorie zu erhalten. Aus diesem Grund wird hier auf kleinere Einheiten, 4 Bits, zurückgegriffen. Daraus resultiert der Zahlenraum [0,15] und somit $d = 16$. Nachfolgend wird die Anzahl benötigter Bytes in Abhängigkeit von der Poker-Kategorie x aufgeführt, um 5 hypothetische Beobachtungen pro Kategorie zu erhalten.

x	5	4	3	2	1
$P(X = x)$	4095/8192	6825/16384	2625/32768	225/65536	1/65536
# Ereignisse	11	13	63	1457	327680
# Bytes	55	65	315	7285	1638400
# Bytes	55	65	315	7250	
# Bytes	55	65	300		

Eine Betrachtung aller Poker-Kategorien erfordert selbst bei einem kleineren Zahlenraum eine große Datenmenge von mindestens 1638400 Bytes. Deshalb werden für die praktische Anwendung des Tests in dieser Arbeit die Kategorien 3, 2 und 1 zusammengefasst, sodass sich die Minimalgröße der zu untersuchenden Daten auf 300 Bytes reduziert. Demnach werden mindestens 300 zu untersuchende Bytes benötigt, um 5 Beobachtungen zu erhalten, welche entweder 3 verschiedene, 2 verschiedene oder 1 verschiedenen (= 5 unterschiedliche) Werte in Untersequenzen der Länge 5 enthalten sind. Der Test kann somit auf Clusterebene angewendet werden.

3.2.2.3 Runs-Test

Der Runs-Test betrachtet die Längen monoton wachsender bzw. fallender Teilsequenzen, Runs genannt. Abbildung 3.2 verdeutlicht diesen Vorgang. Von einer Zufallszahlenfolge erwartet man vor allem Runs kurzer Länge. Nach Aussage von Knuth [Knu81] sind die Verteilungen streng monoton fallender und steigender Untersequenzen identisch.

Ein Run wird beendet, wenn das Monotonieverhalten aufeinanderfolgender Zufallszahlen erstmals durch eine Stoppzahl Y_j unterbrochen wird. Ein aufsteigender Run ist demnach definiert als

$$Y_1 < Y_2 < Y_3 < \ldots < Y_{x+1} > Y_j$$

Die Ausprägungen $x_i (i = 0, \ldots, n)$ der Zufallsvariablen X entsprechen unterschiedlichen Run-Längen.

Durchführung des Tests Die Durchführung des Runs-Tests erfordert nachfolgende Schritte.

- Es erfolgt eine sequenzielle Analyse der Bytes. Der aktuelle Run wird dabei für jedes $Y_i > Y_{i-1}$ inkrementiert, andernfalls wird der Run unterbrochen und die Run-Länge notiert.

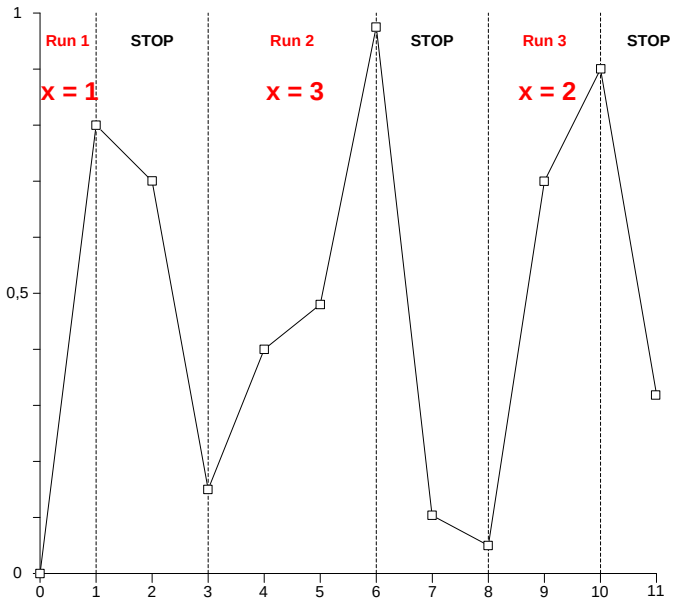

Abbildung 3.2: Beobachtung monoton steigender Teilsequenzen

- Anwendung des χ^2-Tests mit $\upsilon = n - 1$ und folgender Auftrittswahrscheinlichkeit:

$$P(X = x) = \frac{x}{(x+1)!}$$

- Die beobachtete Folge genügt einer wahren Zufallsfolge, wenn $p > \alpha$.

Die Anwendung des Tests in dieser Arbeit erfolgt mit 5 Kategorien. Somit werden die Run-Längen 1, 2, 3, 4 und alle Run-Längen >4 betrachtet.

x	1	2	3	4	>4
$P(X = x)$	1/2	1/3	1/8	1/30	1/120
# Ereignisse	10	15	40	150	600
# Bytes	10	30	120	600	>3000

Um erneut mindestens 5 hypothetische Beobachtungen in jeder Kategorie zu erhalten, müssen die zu überprüfenden Daten mindestens 3000 Bytes groß sein. Der Test kann somit auf Clusterebene angewendet werden.

3.2.2.4 Coupon-collector's-Test

In der Wahrscheinlichkeitstheorie beschreibt das Coupon-collector's-Problem, ob das Sammeln von Coupons (z. B. Aufkleber zum Vervollständigen von Sammelalben) mit

vertretbarem Aufwand realisierbar ist. Die Fragestellung dazu lautet: Es existieren d unterschiedliche Coupons, jedes tritt mit gleicher Wahrscheinlichkeit auf. Wie hoch ist die Wahrscheinlichkeit, dass mehr als x Versuche unternommen werden müssen, um alle d Coupons zu erhalten [MR95]?

Bezogen auf das Testen von Zufallszahlen beobachtet der Coupon-collector's-Test eine Zahlensequenz und notiert die Länge x der Untersequenz, welche den gesamten Zahlenbereich von 0 bis $d-1$ beinhaltet. Offensichtlich ist die minimale Länge $x = d$, eine Grenze nach oben existiert nicht, sodass die maximale Länge durch t beschränkt wird. Somit gilt $d \leq x < t$. Die Ausprägungen x_i der Zufallsvariablen X entsprechen unterschiedlichen Längen der Untersequenzen.

Durchführung des Tests Die Durchführung des Coupon-collector's-Tests erfordert nachfolgende Schritte.

- Mit jeder beobachteten Zahl wird x inkrementiert. Der Test läuft, bis alle Zahlen im Interval $[0, d-1]$ einmal beobachtet wurden. Die Sequenzlänge wird nach Beobachtung aller Werte notiert und zurückgesetzt.

- Anwendung des χ^2-Tests mit $\upsilon = t - d + 1$ und folgender Auftrittswahrscheinlichkeit:

$$\begin{aligned} P(X = x) &= \frac{d!}{d^x} \left\{ {x-1 \atop d-1} \right\} \qquad \text{für } d \leq x < t \\ P(X = t) &= 1 - \frac{d!}{d^{t-1}} \left\{ {t-1 \atop d} \right\} \end{aligned}$$

- Die beobachtete Folge genügt einer Zufallsfolge, wenn $p > \alpha$.

Eine Anwendung des Tests zeigt, dass es eine große Anzahl an möglichen Kategorien gibt. Um den Test auch auf kleinere Datenmengen anwenden zu können, werden die Daten analog zum Poker-Test nicht byteweise, sondern als 4 Bit-Wert betrachtet. Somit ergibt sich der Zahlenbereich [0,15]. Darüber hinaus werden die Längen von Untersequenzen gebündelt, um mindestens 5 hypothetische Beobachtungen pro Kategorie zu erhalten. x bezeichnet in nachfolgender Tabelle ein Intervall, in welchem Untersequenzen unterschiedlicher Längen liegen können (Intervallgröße 10).

x	16-25	26-35	36-45	46-55	56-65	66-75	76-85	>85
$P(X = x)$	0,009	0,118	0,246	0,239	0,167	0,100	0,056	0,65
# Ereignisse	575	43	21	21	31	50	89	77
# Bytes	14375	1505	945	1155	2015	3750	7565	6622
# Bytes	1450		945	1155	2015	3750	3192	

Wie die Tabelle zeigt, reicht die Einteilung in Intervalle der Größe 10 nicht aus, um den Test auch auf kleinere Datenmengen anwenden zu können. Dabei treten insbesondere sehr kurze sowie sehr lange Untersequenzlängen sehr selten auf, sodass hier eine weitere Bündelung notwendig ist, um den Test auf Clusterebene anwenden zu können.

3.3 Güte und Evaluation statistischer Tests

Zur Feststellung von Abhängigkeiten sowie einer möglichen Güte der Tests wurden Zahlenfolgen generiert, welche die im vorherigen Kapitel aufgeführten hypothetischen Verteilungen besitzen. Diese stellen für jeweils einen Test perfekte Zufallszahlen dar und absolvieren diesen mit $p = 1$. Anschließend wurden die Testdaten allen anderen Tests unterzogen und die resultierenden p-Werte in Tabelle 3.2 notiert.

Tabelle 3.2: Ausführungsergebnis der Tests zur Feststellung von Unabhängigkeiten

	Freq.	Lücken	Poker	Runs	Coupon
Frequenzeanalyse	1	0	0	0	0
Lückentest	0	1	0	0	0
Poker-Test	0	0	1	0	0
Runs-Test	0	0	0	1	0
Coupon collector's Test	0	0	0	0	1

Die Gegenüberstellung der Tests mit den für jeweils einen Test ausgelegten Testdaten in Tabelle 3.2 zeigt, dass sich die zu prüfenden Eigenschaften nicht überschneiden. Die Tests können unabhängig voneinander auf die Daten angewendet und betrachtet werden. Sie zeigen außerdem, dass sich Daten durch Bestehen eines einzigen Tests nur als vermutlich zufällig einstufen lassen, durch Kombination mehrerer Tests lässt sich die Aussagekraft erhöhen. Da sich Zufallszahlen jedoch nicht abschließend definieren lassen, kann auch eine Steigerung der Anzahl der Tests nicht endgültig klären, ob eine Zufallsfolge tatsächlich vorliegt. Durch die Tests können jedoch Daten, welche

mindestens eine der dargestellten Eigenschaften nicht besitzen, als nicht-verschlüsselt eingestuft werden.

Im Nachfolgenden werden die aufgeführten Tests auf unterschiedliche Dateiformate angewendet. Hauptaugenmerk liegt dabei auf komprimierten Dateiformaten, welche wie verschlüsselte oder zufällige Daten eine hohe Entropie besitzen. Ziel ist der Ausschluss eines möglichst hohen Anteils dieser Cluster. In einem ersten Schritt erfolgte die Anwendung jedes Tests auf Clusterebene, in einem weiteren auf unterschiedlichen Abstraktionseben eines Datenträgers.

3.3.1 Anwendung der Tests auf Cluster

Die statistischen Tests wurden im ersten Schritt auf jeweils mindestens 10.000 Cluster[1] der Dateiformate aus Tabelle 3.3 mit unterschiedlichem α angewendet. Die Clustergröße wurde dabei weiterhin mit 4096 Bytes angenommen. Die Dateiformate JPG und ZIP repräsentieren bei diesem Test komprimierte, PDF und DOCX vielfach verwendete Dateiformate, welche in der digitalen Forensik häufig von Interesse sind. TrueCrypt-Container (TC) stehen stellvertretend für verschlüsselte Nutzdaten, welche bei allen verschlüsselten Datenformaten dieselben statistischen Eigenschaften aufweisen. Praktischerweise enthalten mit TC verschlüsselte Dateien keinerlei unverschlüsselte Metainformationen und eignen sich deshalb besonders als Repräsentant. RND steht stellvertretend für Zufallsdaten, welche vom UNIX-Zufallszahlengenerator /dev/urandom stammen. RND wurde ausgewählt, um eventuelle Unterschiede zwischen verschlüsselten und zufälligen Daten zu beobachten.

Tabelle 3.3: Dateiformate, auf welche statistische Tests angewendet werden

Dateityp	Beschreibung
JPG	komprimierte Grafik (unverschlüsselt)
ZIP	komprimiertes Archiv (unverschlüsselt)
PDF	Dokument (unverschlüsselt)
DOCX	MS Office-Dokument (unverschlüsselt)
TC	Datencontainer (verschlüsselt)
RND	Zufallsdaten (unverschlüsselt)

Hauptaugenmerk liegt auf der Erkennung verschlüsselter Cluster und dem Aus-

[1] Ein Cluster bezeichnet eine Speichereinheit des Dateisystems und wird im Zusammenhang mit der Analyse von Daten mittels Metainformationen des Dateisystems auf Seite 52 näher erläutert.

schluss von Clustern, welche nicht verschlüsselt sind. Die verwendeten Cluster enthalten weitestgehend Nutzdaten, d.h. Metainformationen wurden, soweit möglich, ausgeschlossen. Da diese Informationen meistens im Klartext vorliegen und somit nicht den statistischen Eigenschaften verschlüsselter Daten genügen, würden sie das Testergebnis verfälschen. Welche Probleme Metadaten jedoch im Praxiseinsatz verursachen, wird im nächsten Abschnitt deutlich.

Zunächst wurden in den nachfolgenden Tabellen die durchschnittlichen p-Werte sowie prozentualen Anteile positiver Cluster notiert. Die Bezeichnung „positives Cluster“ kennzeichnet ein Cluster, welches einen Test besteht. Aus den Tests resultierend wurde ein α für den weiteren Testverlauf gewählt, ab welchem untersuchte Daten als zufällig und somit möglicherweise verschlüsselt eingestuft werden. Das ausgewählte α stellt einen Kompromiss zwischen false positives und false negatives dar.

3.3.1.1 Frequenzanalyse

Die Testergebnisse der Frequenzanalyse in Tabelle 3.4 zeigen, dass sich nahezu alle verschlüsselten Cluster mittels dieses Tests erkennen lassen. Die Erkennungsrate sinkt leicht bei steigendem Signifikanzniveau α, wohingegen sich die durch die Formate ZIP und DOCX erzeugten false negatives stark reduzieren. Für JPG stellt dieser Test das zentrale Ausschlusskriterium dar; knapp 99% aller getesteten Cluster bestehen unerwarteterweise die Frequenzanalyse nicht. Für eine derartige Ungleichverteilung der Bytes in JPG-Dateien konnten im Rahmen dieser Arbeit keine stichhaltigen Gründe identifiziert werden. Mittels der Frequenzanalyse lassen sich TC und RND nicht unterscheiden.

Tabelle 3.4: Testergebnisse der Frequenzanalyse

	DOCX	JPG	PDF	ZIP	TC	RND
Ø p-Wert	0,08966	4,66e-5	0,02645	0,18570	0,50040	0,50423
Anteil positiver Cluster in %						
$\alpha = 0,001$	30,4324	0,57098	14,7776	77,2912	99,9010	99,9216
$\alpha = 0,005$	27,2060	0,18577	12,4943	71,4570	99,4114	99,6429
$\alpha = 0,010$	25,6532	0,10928	11,3070	67,8662	98,9478	99,2423
$\alpha = 0,015$	24,5374	0,06288	10,5672	65,0946	98,4165	98,6326
$\alpha = 0,020$	23,7657	0,04644	9,93698	62,6527	97,9685	98,1623

Zusammenfassend lassen sich mittels der Frequenzalayse verschlüsselte Cluster

zwar erkennen, aufgrund der hohen false negative-Rate unverschlüsselter Dateiformate hat eine praktische Anwendung dieses Tests auf Clusterebene jedoch wenig Aussagekraft.

3.3.1.2 Lückentest

Die Testergebnisse des Lückentests in Tabelle 3.5 zeigen, dass verschlüsselte Cluster einer starken Streuung genügen. Die Erkennungsrate bei steigendem α fällt langsamer als bei anderen Dateiformaten. Die false negative-Raten der Formate ZIP, DOCX und PDF liegen hoch. Eine Unterscheidung von TC und RND ist erneut nicht möglich.

Tabelle 3.5: Testergebnisse des Lückentest

	DOCX	JPG	PDF	ZIP	TC	RND
Ø p-Wert	0,26184	0,37938	0,27626	0,45382	0,58927	0,59047
Anteil positiver Cluster in %						
$\alpha = 0,001$	67,4012	95,2273	65,5768	96,9834	99,9479	99,9565
$\alpha = 0,005$	63,0683	91,9681	62,4441	95,1152	99,7500	99,6952
$\alpha = 0,010$	60,5207	89,6268	60,8092	93,7218	99,4687	99,4252
$\alpha = 0,015$	58,7634	87,8210	59,8228	92,2413	99,1770	99,0681
$\alpha = 0,020$	57,7220	86,3703	58,8638	91,5091	98,9218	98,8330

Zusammenfassend lassen sich mittels des Lückentests verschlüsselte Cluster zwar erkennen, aufgrund der hohen false negative-Raten unverschlüsselter Dateiformate hat eine praktische Anwendung dieses Tests auf Clusterebene, analog zur Frequenzanalyse, wenig Aussagekraft.

3.3.1.3 Runs-Test

Die Testergebnisse des Runs-Tests in Tabelle 3.6 zeigen erneut eine sehr hohe Erkennungsrate verschlüsselter Cluster. Bei steigendem α fällt diese, analog zu vorherigen Tests, leichter als bei anderen Dateiformaten. Viele false negatives werden durch die Formate JPG und ZIP erzeugt, auch bei den Formaten DOCX und PDF sind erhöhte Raten festzustellen. TC und RND sind abermals nicht zu unterscheiden.

Tabelle 3.6: Testergebnisse des Runs-Test

	DOCX	JPG	PDF	ZIP	TC	RND
Ø p-Wert	0,31840	0,47292	0,25423	0,47039	0,49976	0,50243
Anteil positiver Cluster in %						
$\alpha = 0,001$	75,0163	99,2733	69,8055	99,1119	99,8906	99,8345
$\alpha = 0,005$	73,4821	98,6941	66,1978	98,3998	99,5364	99,5384
$\alpha = 0,010$	72,1153	98,0084	64,0789	97,6571	99,0832	99,1465
$\alpha = 0,015$	71,0832	97,3992	62,7089	97,1289	98,5676	98,7284
$\alpha = 0,020$	70,3115	96,8036	61,6038	96,4245	98,0467	98,2146

Zusammenfassend lassen sich mittels des Runs-Tests verschlüsselte Cluster zwar erkennen, aufgrund der hohen false negative-Raten unverschlüsselter Dateiformate ist eine praktische Anwendung auf Clusterebene, analog zu vorherigen Tests, fragwürdig.

3.3.1.4 Poker-Test

Die Testergebnisse des Poker-Tests in Tabelle 3.7 zeigen, dass eine Erkennung verschlüsselter Cluster mittels dieses Tests möglich ist. Die Erkennungsrate bei steigendem α fällt, analog zu allen bisherigen Tests, leichter als bei anderen Dateiformaten. Durch unverschlüsselte Dateiformate werden jedoch weniger false negatives als beim Runs- oder Lückentest generiert. Erwartungsgemäß lassen sich auch hiermit TC und RND nicht unterscheiden.

Tabelle 3.7: Testergebnisse des Poker-Test

	DOC	JPG	PDF	ZIP	TC	RND
Ø p-Wert	0,29323	0,51505	0,13222	0,55475	0,77484	0,77239
Anteil positiver Cluster in %						
$\alpha = 0,001$	49,2143	92,9243	25,8928	89,3500	99,6979	99,6778
$\alpha = 0,005$	47,4942	89,9246	24,1301	87,3899	99,4947	99,4687
$\alpha = 0,010$	46,0438	87,5341	23,1254	85,7821	99,0728	99,0768
$\alpha = 0,015$	44,6304	85,2912	22,1664	83,7991	98,4321	98,4149
$\alpha = 0,020$	43,1706	82,5921	21,0978	80,6217	97,1143	97,0476

Zusammenfassend lassen sich mittels des Poker-Tests verschlüsselte Cluster zwar erkennen, aufgrund der hohen false negative-Raten unverschlüsselter Dateiformate ist

eine praktische Anwendung auf Clusterebene ebenfalls fragwürdig.

3.3.1.5 Coupon-collector's-Test

Auch mit dem Coupon-collector's Test lassen sich nahezu alle verschlüsselten Cluster erkennen, wie die Testergebnisse in Tabelle 3.8 zeigen. Die Erkennungsrate bei steigendem α sinkt, analog zu allen bisherigen Tests, leichter als bei anderen Dateiformaten. Die durch die Formate PDF und DOCX entstehenden false negatives reduzieren sich stark bei steigendem α. Eine Unterscheidung zwischen TC und RND ist, erwartungsgemäß wie bei allen vorherigen Tests, nicht durchführbar.

Tabelle 3.8: Testergebnisse des Coupon-collector's-Test

	DOCX	JPG	PDF	ZIP	TC	RND
Ø p-Wert	0,18807	0,37832	0,11255	0,39552	0,50051	0,50406
Anteil positiver Cluster in %						
$\alpha = 0,001$	59,1260	94,2738	47,8126	93,4615	99,8021	99,8084
$\alpha = 0,005$	54,9605	91,4299	41,6750	90,8277	99,3281	99,2597
$\alpha = 0,010$	52,2455	89,4520	38,6334	88,9289	98,8176	98,6501
$\alpha = 0,015$	50,3487	87,9221	36,4965	87,3899	98,3175	98,1362
$\alpha = 0,020$	48,8889	86,5834	34,7520	86,2798	97,7810	97,7007

Zusammenfassend lassen sich auch mittels des Coupon-collector's-Tests verschlüsselte Cluster zwar erkennen, aufgrund der hohen false negative-Raten unverschlüsselter Dateiformate hat eine praktische Anwendung des Tests auf Clusterebene wenig Aussagekraft.

3.3.1.6 Anwendung aller Tests

Während sich die Erkennung verschlüsselter Cluster mit jedem Test durchführen lässt, kann nur durch eine Kombination aller Tests die Anzahl der fälschlicherweise als verschlüsselt eingestuften Formate gesenkt werden. Tabelle 3.9 dokumentiert die Ausführung aller Tests auf alle Dateiformate. Durch ein steigendes Signifikanzniveau α sinkt die Anzahl der false negatives, während die false positive-Rate ansteigt. Als Kompromiss zwischen den beiden Raten wird für den weiteren Testverlauf $\alpha = 0,01$ angenommen. Während somit knapp 5 aus 100 verschlüsselten Clustern nicht erkannt werden, fällt die Erkennungsrate z. B. bei ZIP um 12% von 69,2% auf 57,2%. Der trotzdem noch hohe Anteil positiver Cluster beim Datenformat ZIP resultiert

aus der Kompression. Durch die Reduktion von Redundanzen können Bytesequenzen entstehen, welche, abhängig von den komprimierten Daten, zufälligen und somit verschlüsselten Daten ähneln. Die Anzahl der generierten false negatives lassen sich durch Entwicklung und Anwendung weiterer Tests vermutlich reduzieren.

Tabelle 3.9: Prozentualer Anteil positiver Cluster bei Kombination aller Tests

Test	DOCX	JPG	PDF	ZIP	TC	RND
$\alpha = 0,001$	25,4021	0,56005	10,7772	69,2137	99,2398	99,1987
$\alpha = 0,005$	22,2129	0,17758	8,4934	61,9861	97,5570	97,6485
$\alpha = 0,010$	20,4556	0,10655	7,5075	57,1702	95,4683	95,6279
$\alpha = 0,015$	18,9865	0,06010	6,6855	52,9822	93,1399	93,1197
$\alpha = 0,020$	17,8615	0,04371	5,9640	48,4879	90,2750	90,2891

3.3.1.7 Anwendung aller Tests auf einen TrueCrypt-Container

Vorheriger Abschnitt behandelte u. a. TrueCrypt-Container, welche aus rein verschlüsselten Nutzdaten bestehen. Tabelle 3.9 beinhaltet den prozentualen Anteil von Clustern, welche alle statistischen Eigenschaften besitzen. Nachfolgender Ausschnitt aus der Logdatei zeigt die Ergebnisse der einzelnen Tests. Das Ergebnis jedes Tests, der p-Wert, wird zeilenweise für jedes Segment (oder Cluster) aufgeführt.

Tabelle 3.10: Testergebnisse eines TrueCrypt-Containers (Ausschnitt aus Logdatei)

Seg.	Freq.	Lückent.	Runs-T.	Poker-T.	Coupon	Result
...	...	...	...	...	...	...
135	0,9449	0,5906	0,2547	0,9603	0,2718	OK
136	0,8915	0,9265	0,7900	0,7086	0,8992	OK
137	0,8613	0,2467	0,6468	0,5936	0,3421	OK
138	0,2829	0,1967	0,8676	0,9619	0,3279	OK
139	0,6870	0,4876	0,1891	0,4821	0,4948	OK
140	0,6666	0,4545	0,4501	0,9901	0,5236	OK
141	0,4684	0,7603	0,3173	0,4654	0,2834	OK
142	0,2250	0,2369	0,0922	0,7887	0,2211	OK
143	0,1917	0,6547	0,5627	0,7887	0,9181	OK
...	...	...	...	...	...	...

Die Testergebnisse verdeutlichen die Erkenntnisse aus Abschnitt 3.3, S. 31. Demnach ist in der Tabelle zu beobachten, dass das Testergebnis eines Tests keine Auswirkungen auf einen anderen Test hat. Segment Nr. 135 erweist eine gute Gleichverteilung, das Ergebnis des Runs-Test fällt jedoch schlechter aus. Segment Nr. 138 hingegen erzielt ein gutes Ergebnis beim Runs-Test, ist jedoch weniger gleichverteilt als vorheriges Cluster. Die überprüften Eigenschaften überschneiden sich also nicht.

Unterschiede in den Testergebnissen können vorkommen, da die überprüften Daten keine perfekten Zufallsdaten sind und somit nur annähernd identitische Merkmale aufweisen. Dadurch kann es zu Schwankungen der einzelnen Testergebnisse eines Segments kommen. Die obigen Ergebnisse erfüllen allesamt das Kriterium über Annahme der Nullhypothese, welches zuvor mit $p > 0,01$ festgelegt wurde. Die einzelnen Segmente wurden demnach alle als vermutlich verschlüsselt eingestuft.

3.3.2 Anwendung der Tests auf unterschiedlichen Abstraktionsebenen eines Datenträgers

In einem weiteren Schritt zur Überprüfung der Geeignetheit der Tests zur Erkennung verschlüsselter Daten erfolgte eine Anwendung auf den unterschiedlichen Verschlüsselungsebenen aus Abbildung 1.1, S. 2.

3.3.2.1 Anwendung auf vollverschlüsselte Datenträger

Die Tests zur Erkennung von Datenträgervollverschlüsselung wurden auf einem 10GB Datenträgerimage durchgeführt, welches zuvor mittels TrueCrypt vollverschlüsselt wurde. Ziel war die Ermittlung eines Anteils positiv getesteter Cluster, mit welchem ein Datenträger als vollverschlüsselt eingestuft werden kann.

Tabelle 3.11: Anwendung der Tests auf vollverschlüsselte Datenträger

Anteil analysierter Cluster	Anteil positiv ermittelter Cluster
100%	95,4212%
10%	95,4258%
5%	95,4387%
1%	95,4212%

Die Testergebnisse in Tabelle 3.11 zeigen, dass der Anteil positiv ermittelter Cluster aufgrund der Erkenntnisse aus Abschnitt 3.3.1.6, S. 36, erwartungsgemäß bei über

95% liegt. Einen vollverschlüsselten Datenträger kann man bei einer stichprobenartigen Analyse trotzdem nicht zwangsläufig als einen solchen identifizieren, da sich unverschlüsselte Festplattenbereiche nicht auf das Testergebnis auswirken müssen. Die Erkennung hängt dabei vom relativen Anteil nichtverschlüsselter Bereiche ab. Somit erfolgt stattdessen eine Überprüfung auf der nächsten Ebene, der Partitionsebene. Anhand verschlüsselter Partitionen kann auf Datenträgervollverschlüsselung geschlossen werden.

3.3.2.2 Anwendung auf verschlüsselte Partitionen

Die Anwendung der Tests auf der Partitionsebene erfolgt analog zur Erkennung von Datenträgervollverschlüsselung, allerdings nur auf dem von der Partition allozierten Speicherbereich. Die Testergebnisse gleichen der in Tabelle 3.11 aufgeführten Ergebnisse.

Im weiteren Verlauf der Arbeit wird als Einstufungskriterium verschlüsselter Partitionen ein Testergebnis von 95% bei einer stichprobenartigen Untersuchung von 1% aller Cluster vorausgesetzt. Somit können selbst kleine Partitionen entsprechend eingestuft werden.

3.3.2.3 Anwendung auf Anwendungsebene

Die Tests auf der Anwendungsebene beziehen sich auf die in Tabelle 3.3, S. 32, aufgeführten Dateiformate, welche sowohl unverschlüsselt als auch verschlüsselt abgespeichert werden können. In den nachfolgenden Tabellen wurde jeweils der Anteil verdächtiger Cluster pro Datei notiert. Als verdächtiges Cluster gilt, welches alle statistischen Tests besteht.

Ließen die Tests keine zweifelsfreie Zuordnung zu, wurde nach erweiterten Maßnahmen gesucht, welche die Aussagekraft des Testergebnisses steigern. Dabei wurde insbesondere auf den klassischen Ansatz, die Signaturerkennung, zurückgegriffen.

Portable Document Format PDF ist ein geräte-, plattform- und softwareunabhängiges Dateiformat und heute globaler Standard für den elektronischen Dokumentenaustausch [Ado12].

Ein PDF-Dokument [Ado06] besteht aus einem Dateiheader, welcher die PDF-Version enthält, einem Body, dem eigentlichen Dateiinhalt, einer *cross reference table* zur Verwaltung von Objekten und einem Trailer, welcher die Datei abschließt. Seit

Version 1.1 besteht die Möglichkeit, Dokumente verschlüsselt abzuspeichern. Informationen in Bezug auf Verschlüsselung werden im *encryption dictionary* gespeichert.

Einfach betrachtet besteht ein PDF-Dokument aus einer Sammlung von nummerierten Objekten. Ein Objekt stellt einen Datenstrom (z. B. Textabschnitt oder Grafik) oder sogenannte *dictionaries* (Strukturen, welche das Dokument selbst beschreiben) dar. Diese Objekte und derer Offsets werden in der *cross reference table* aufgelistet. Das *trailer dictionary* am Ende einer PDF listet Objektnummern wichtiger Objekte auf.

Testergebnisse Nach Tabelle 3.12 lassen sich unverschlüsselte PDF-Dokumente nicht immer als solche identifizieren. Demnach lag der Anteil verdächtiger Cluster bei 4 Dateien zwischen 26% und 50%. Eine genaue Untersuchung dieser Dateien ergab eine hohe Kompressionsrate. PDF stellt Kompressionsmöglichkeiten bereit.

PDF verschlüsselt lediglich die Nutzdaten, Metainformationen liegen weiterhin im Klartext vor. Zu den Metainformationen zählen Daten, welche die Dokumentstruktur beschreiben und sich beim Dateiformat PDF nicht zwangsläufig am Dateianfang oder -ende, sondern auch zwischen verschlüsselten Datenströmen befinden. Cluster, welche ausschließlich verschlüsselte Nutzdaten besitzen, werden als solche erkannt. Beinhalten diese jedoch zusätzlich Metadaten, schlägt eine Erkennung fehl. Darüber hinaus kann es bei der Betrachtung aller Cluster einer Datei sein, dass der Anteil negativer Cluster wegen der Metadaten dominiert und die Datei deshalb nicht als verschlüsselt erkannt wird, obwohl die eigentlichen Nutzdaten verschlüsselt sind. Dies hängt von der Dateigröße ab.

Tabelle 3.12: Anwendung der Tests auf PDF-Dokumente

Anteil verdächtiger Cluster	unverschlüsselt	verschlüsselt
0%	87	3
1 - 25%	9	9
26 - 50%	4	11
51% - 75%	0	68
76% - 100%	0	9

Keines der überprüften unverschlüsselten PDF-Dokumente erreichte einen höheren Anteil verdächtiger Cluster als 50%. Jedoch enthielten 23% der verschlüsselten Dateien einen positiven Clusteranteil unterhalb von 50%. In einem erweiterten Ansatz werden deshalb Metadaten herangezogen, welche auf Verschlüsselung hinweisen, um

auch verschlüsselte PDF-Dokumente mit einem geringen Anteil positiver Cluster zu identifizieren. Zur Identifizierung von verschlüsselten PDF-Dokumenten eignet sich das *encryption dictionary*, welches nur existiert, wenn das PDF-Dokument verschlüsselt ist. Erkennen lässt sich dieses anhand des ASCII-Strings `/Encrypt` im Trailer. Tabelle 3.13 fasst Bytemuster für eine signaturbasierte Identifizierung zusammen.

Tabelle 3.13: Übersicht identifizierter PDF-Signaturen

	Hex-Signatur	ASCII
Header	`25 50 44 46 2D`	`%PDF-`
Erweitert	`2F 45 6E 63 72 79 70 74`	`/Encrypt`
Footer	`0A 25 25 45 4F 46 0A`	`.%%EOF.`
	`0D 0A 25 25 45 4F 46 0D 0A`	`..%%EOF..`
	`0D 25 25 45 4F 46 0D`	`.%%EOF.`

ZIP-Archive Das ZIP-Dateiformat ist eines der bekanntesten Formate zur verlustlosen Komprimierung und Archivierung von Dateien. Dateien werden dabei einzeln komprimiert, d.h. es erfolgt im Vergleich zu anderen Kompressionsverfahren keine progressive Kompression. Vorteile sind ein schnelles Hinzufügen oder Löschen von einzelnen Dateien aus dem Archiv, da hierfür nicht alle Dateien neu komprimiert werden müssen. ZIP bietet die Möglichkeit, Archive verschlüsselt abzuspeichern [PKW12].

Testergebnisse Die Testergebnisse in Tabelle 3.14 zeigen das gleiche Problem wie zuvor bei PDF-Dokumenten. Demnach entstehen false negatives bei unverschlüsselten Archiven, welche auf die Kompression zurückzuführen sind. Ein Großteil der Cluster verschüsselter ZIP-Archive werden als solche klassifiziert. Probleme stellen kleine ZIP-Archive dar, bei welchen der Metadatenanteil dominiert.

Tabelle 3.14: Anwendung der Tests auf ZIP-Archive

Anteil verdächtiger Cluster	unverschlüsselt	verschlüsselt
0%	7	0
1 - 25%	57	1
26 - 50%	34	8
51% - 75%	2	25
76% - 100%	0	66

Lediglich 2 unverschlüsselte ZIP-Archive enthielten einen Anteil positiver Cluster von über 50%. Umgekehrt lag bei 9% der getesteten verschlüsselten ZIP-Archive der positive Clusteranteil unter 50%. Zur Erfassung dieser Dateien lassen sich Headerinformationen heranziehen, welche auf Verschlüsselung hindeuten. Diese Informationen liefern die *general purpose bits* (1 Byte) ab Offset `0x06`. Bit 0 ist bei verschlüsselten Archiven gesetzt. Tabelle 3.15 fasst Bytemuster für eine signaturbasierte Identifizierung zusammen.

Tabelle 3.15: Übersicht identifizierter ZIP-Signaturen

	Hex-Signatur	ASCII
Header	`50 4B 03 04`	`PK..`
Erweitert	Bit 0 ab Offset `0x06`	
Footer	`50 4B 05 06` + 18 Bytes	`PK..`

TrueCrypt TrueCrypt [Tru12] ist eine OpenSource-Verschlüsselungssoftware, welche on-the-fly-Verschlüsselung ermöglicht. On-the-fly-Verschlüsselung bedeutet dabei, dass Dateien beim Speichervorgang automatisch verschlüsselt und beim Ladevorgang automatisch entschlüsselt werden.

Zu verschlüsselnde Dateien werden in einem Container abgespeichert, welcher ein eigenes Dateisystem beinhaltet. Die Containerdatei kann beliebig groß sein und mit unterschiedlichen Algorithmen verschlüsselt werden. Die Besonderheit an diesen Containern ist, dass TrueCrypt jegliche Metainformationen verschlüsselt. Lediglich ein Salt, welcher in Kombination mit einem Passwort zum Entschlüsseln des symmetrischen Schlüssels genutzt wird, liegt unverschlüsselt vor. Da dieser zufällig generiert wurde, lässt er sich nicht von den restlichen Daten unterscheiden. TrueCrypt-Container lassen sich mittels der statistischen Tests zweifelsfrei identifizieren, wie Tabelle 3.16 zeigt.

Tabelle 3.16: Anwendung der Tests auf TrueCrypt-Container

Anteil verdächtiger Cluster	verschlüsselt
0%	0
1 - 25%	0
26 - 50%	0
51% - 75%	0
76% - 100%	100

Microsoft Office-Dokumente Microsoft Office mit seinen Komponenten Word, Excel, PowerPoint, Access und Outlook gilt nach wie vor als Marktführer unter den Office-Suiten mit einem deutlichen Abstand zur Konkurrenz. In Deutschland beträgt der Marktanteil rund 72%, gefolgt von OpenOffice mit 22%. Microsoft Office gibt es sowohl für Windows-Betriebssysteme als auch für Mac [Oms10]. Aufgrund der weiten Verbreitung ist dieses Dateiformat bei forensischen Untersuchungen sehr interessant.

Bei Microsoft Office-Dokumenten ist zwischen zwei verschiedenen Dateiformaten zu unterscheiden. Zum einen existiert weiterhin das ältere binäre Dateiformat (DOC, XLS, PPT,...) und das in Microsoft Office 2007 eingeführte neuere XML-Format (DOCX, XLSX, PPTX,...).

Microsoft Office 97-2003 verwendet das Object Linking and Embedding Compound File (OLECF)-Format zur Abspeicherung von Dateien. Eine OLECF-Datei stellt einen Container dar, welcher wie ein Dateisystem verwaltet wird. Der Container beinhaltet unabhängige Datenströme (vergleichbar mit Dateien in einem Dateisystem), welche in einer Hierarchie von *storages* organisiert werden (vergleichbar mit Unterordnern in einem Dateisystem) [Ren07].

Ab Microsoft Office 2007 werden Dokumente standardmäßig als Open Packaging Convention (OPC) abgespeichert. OPC definiert eine strukturierte Methode zum Speichern von Anwendungsdaten und zugehörigen Ressourcen in einem standardmäßigen ZIP-Archiv. Innerhalb dieser ZIP-Datei sind in vordefinierten Pfaden XML-Dateien abgespeichert, welche Metadaten sowie den eigentlichen Inhalt beschreiben [Ngo07].

Tabelle 3.17: Anwendung der Tests auf Microsoft Office-Dokumente

Anteil verdächtiger Cluster	unverschlüsselt	verschlüsselt
0%	57	5
1 - 25%	40	2
26 - 50%	2	6
51% - 75%	1	14
76% - 100%	0	58

Tabelle 3.17 zeigt die Testergebnisse bei Anwendung der statistischen Tests auf jeweils 100 unverschlüsselte sowie verschlüsselte DOCX-Dateien, welche stellvertretend für alle Office-Dateiformate untersucht wurden. Es lassen sich die gleichen Schwierigkeiten wie bei vorherigen Anwendungsdateien feststellen. Die Erkennungsrate verschlüsselter Dateien hängt dabei hauptsächlich von der Dateigröße ab. Bei besonders

kleinen Dateien kann der Anteil an Metainformationen über 80% betragen. Bei 13% der überprüften verschlüsselten DOCX-Dateien lag der Anteil positiver Cluster bei unter 50%, sodass zur Erfassung dieser Dateien ebenfalls erweiterte Erkennungsmaßnahmen herangezogen werden müssen. Tabelle 3.18 fasst die notwendigen Signaturen zusammen, wobei zwischen dem binären Dateiformat (verschlüsseltes DOC) und dem XML-Dateiformat (verschlüsseltes DOCX) in der erweiterten Erkennung unterschieden werden muss.

Tabelle 3.18: Übersicht identifizierter MS Office-Signaturen

	Hex-Signatur	ASCII
Header	`D0 CF 11 E0 A1 B1 1A E1`	
Erweitert (DOC)	`00 01 00 01`	

Erweitert (DOCX)	`45 00 6E 00 63 00 72 00 79`
	`00 70 00 74 00 69 00 6F 00`
	`6E 00 49 00 6E 00 66 00 6F`
		`E n c r y`
		`p t i o n`
		`I n f o`

3.4 Zusammenfassung

Dieses Kapitel betrachtete zunächst die Blockgröße verschiedener Blockverschlüsselungsalgorithmen als Identifikationsmerkmal für verschlüsselte Daten. Nach Überlegungen erwies sich dieses Merkmal als ungeeignet, da mit vielen false negatives zu rechnen ist. Darüber hinaus ließe sich die Erkennung einfach umgehen, indem die Dateigröße manipuliert wird.

Als geeigneter stellte sich die Überprüfung der aufgeführten statistischen Eigenschaften verschlüsselter Daten heraus. Die Anwendung der Tests zur Überprüfung dieser Eigenschaften erfolgte dabei zunächst auf Clusterebene. Verschlüsselte Cluster ließen sich dabei mit jedem Test zu einem sehr hohen Anteil (>99%) als solche identifizieren. Einen ähnlich hohen Anteil besitzten Zufallsdaten, welche identische Eigenschaften aufweisen und somit nicht von verschlüsselten Daten unterschieden werden können. Abhängig vom Test resultierten durch unverschlüsselte Datenformate unterschiedlich hohe false negative-Raten. Diese konnten durch eine Kombination aller Tests gesenkt werden. Ein einzelnes Cluster definitiv als verschlüsselt einstufen zu können, wird auch durch Überprüfung weiterer statistischer Eigenschaften nicht möglich sein.

In der Praxis bestehen verschlüsselte Daten häufig aus mehreren Clustern. Der Erkennungsprozess erfolgt somit auf Betrachtung mehrerer aufeinanderfolgender Cluster. Verschlüsselte Cluster liegen mit einer Erkennungsrate von 95% weit über den anderen (fälschlicherweise als verschlüsselt eingestuften) Erkennungsraten. Somit ist eine Unterscheidung verschlüsselter bzw. zufälliger Daten und unverschlüsselter Daten bei Betrachtung mehrerer aufeinanderfolgender Cluster möglich.

Die Anwendung der Tests auf unterschiedlichen Abstraktionsebenen eines Datenträgers zeigte, dass sich die präparierten verschlüsselten Daten erkennen ließen. Die Erkennung gestaltet sich dabei zuverlässiger, je höher der Anteil rein verschlüsselter Nutzdaten ist. Während eine korrekte Klassifizierung verschlüsselter Datenträger bzw. Partitionen möglich ist, ergeben sich aufgrund der Konstuktionseigenschaften unterschiedlicher Dateiformate Probleme.

Durch die Dateiformate PDF, ZIP und DOCX entstehen false negatives, welche auf die Kompression zurückzuführen sind. Viele der überprüften unverschlüsselten Dateien enthielten dabei einen positiven Clusteranteil von über 50%. Umgekehrt existierten verschlüsselte Dateien mit einem Anteil positiver Cluster von unter 50%. Dies resultiert aus dem Anteil der Metadaten, der insbesondere bei kleinen Dateien dominiert. Metadaten können zudem zwischen verschlüsselten Datenströmen vorhanden sein und eine Identifizierung verschlüsselter Cluster verhindern. Nachfolgend wird als Einstufungskriterium für mögliche Verschlüsselung auf der Anwendungsebene ein positiver Clusteranteil von 50% angenommen. Dies stellt einen Kompromiss zwischen false positives und false negatives dar. Um die false positive-Rate zu reduzieren, wird unterstützend zu den statistischen Tests die klassische Signaturerkennung, erweitert um die Überprüfung auf spezifische, bei der Verschlüsselung gesetzte Bits bzw. eingefügte Signaturen, auf die Daten angewendet. Dafür wurden für einige vorgestellte Dateiformate charakteristische Bytemuster identifiziert. Nachteil dieser Methode ist die manuelle Anpassung zur Erkennung weiterer Dateiformate sowie ggf. Anpassung bestehender Dateiformate, wenn sich diese durch Programmaktualisierungen ändern.

Die Überprüfung der Signatur hat zwar Auswirkungen auf die Performanz der Analyse, jedoch ergibt sich durch die Kombination von Signaturerkennung und statistischer Tests auch ein Vorteil. Veränderungen an der Signatur haben keine Auswirkungen auf das Ergebnis der statistischen Analyse. So könnten Manipulationen aufgedeckt werden, insofern der Metadatenanteil der Datei eine Identifizierung nicht verhindert.

4 Abfolge der Datenträgeranalyse

Dieses Kapitel umfasst die Abfolge der Analyseschritte bei der Untersuchung von Datenträgern, sowohl bei Live-Systemen als auch bei Offline-Systemen. Die Grundlage aller Analyseschritte bilden die im vorherigen Kapitel aufgeführten Tests. Abbildung 4.1 zeigt den Ablauf der Untersuchung. Die einzelnen Schritte werden im Verlauf dieses Kapitels erläutert.

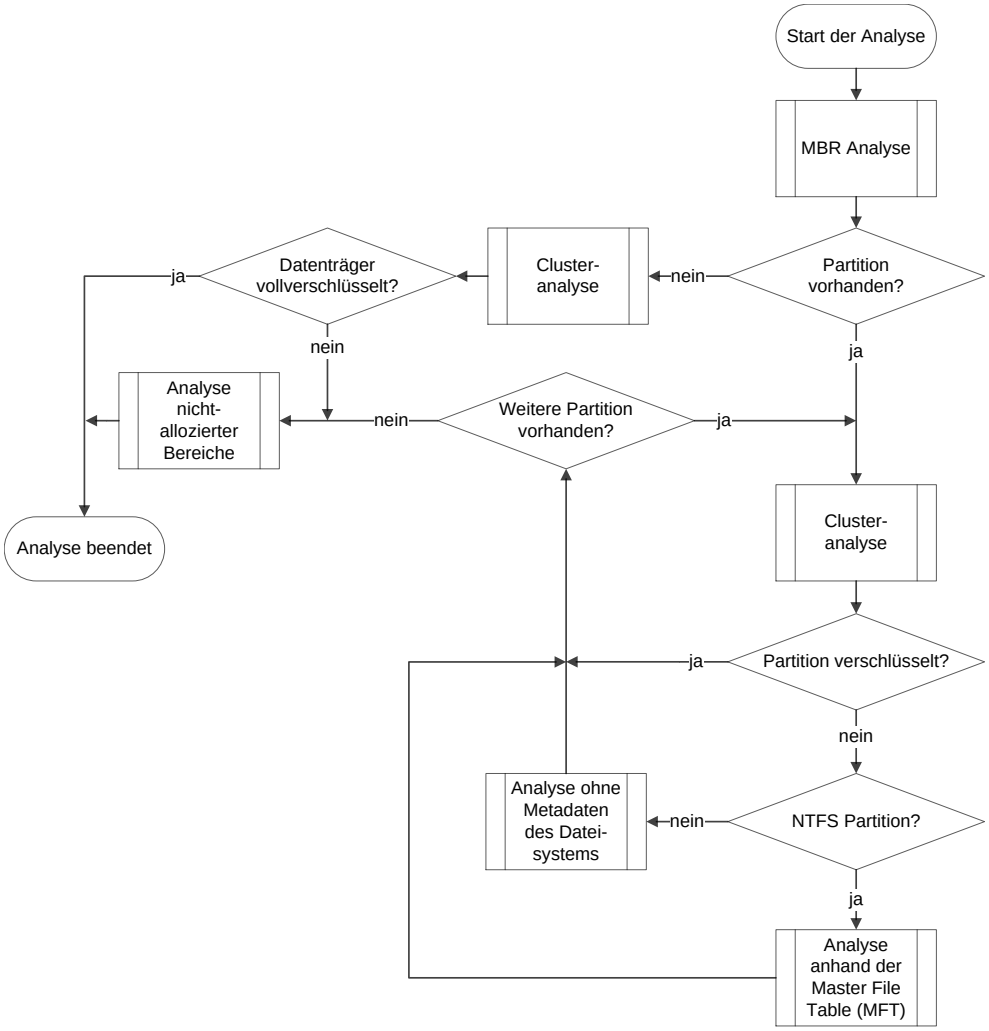

Abbildung 4.1: Abfolge der Datenträgeranalyse

4.1 Analyse des Master Boot Record

Der Master Boot Record (MBR) ist der erste Datenblock eines in Partitionen aufgeteilten Speichermediums. Das Auslesen des MBR dient vorrangig zur Bestimmung des Datenträgerlayouts. Darüber hinaus können Original Equipment Manufacturer (OEM)-Strings enthalten sein, welche auf Partitions- bzw. Datenträgervollverschlüsselung hinweisen und somit als Voreinstufung dienen können.

Bei BIOS-basierten Computern sind die Layouts einzelner Partitionen direkt im MBR, bei UEFI-basierten Computern hingegen in der GUID Partition Table (GPT) abgespeichert. Für die Analyse im Rahmen dieser Arbeit wurden exemplarisch Methoden für die Analyse des MBR implementiert.

Nachfolgend wird zunächst die Bestimmung des Datenträgerlayouts näher erläutert, im Anschluss werden OEM-Strings identifiziert, welche auf mögliche Verschlüsselung durch weit verbreitete Verschlüsselungssoftware hinweisen können.

4.1.1 Bestimmung des Datenträgerlayouts

Abbildung 4.2 zeigt die Struktur des MBR. Die Partitionstabelle umfasst dabei maximal vier primäre Partitionen. Darüber hinausgehende Partitionen werden in einer sogenannten erweiterten Partition aufgeführt [Car10]. Die Analyse des MBR im Rahmen dieser Arbeit beinhaltet das Auslesen der Layouts der vier primären Partitionen.

Abbildung 4.3 zeigt die Struktur der GPT. Bei UEFI-basierten Rechnersystemen befindet sich diese ab dem zweiten Sektor. Die Signatur `EFI PART` ab Offset `0x00` deutet auf GPT hin. Weiterhin wird der Datenträger in der klassischen Partitionstabelle des MBR als vollständig alloziert markiert, um vor einem ungewollten Zugriff älterer Partitionierungsprogramme zu schützen, welche GPT möglicherweise nicht interpretieren können. Ab Logical Block Address (LBA)[1] 3 bis bis einschließlich 33 können Partitionen aufgelistet sein. Ein Eintrag umfasst dabei 128 Byted, sodass pro Datenträgerblock mehrere Einträge gespeichert werden können. Somit besteht bei einer Sektorgröße von 512 Bytes Kapazität für bis zu 128 Partitionen [Uni12].

Aus den Partitionslayouts heraus lassen sich außerdem Datenträgerbereiche ableiten, welche nicht von Partitionen alloziert wurden und gesondert überprüft werden müssen.

[1] LBA bezeichnet eine Methode zur Adressierung von Sektoren auf Datenträgern und wird im Zusammenhang mit der Analyse von Daten mittels Metainformationen des Dateisystems auf Seite 52 näher erläutert.

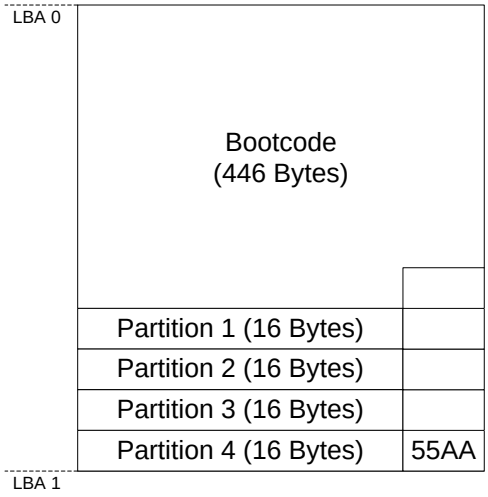

Abbildung 4.2: Struktur MBR

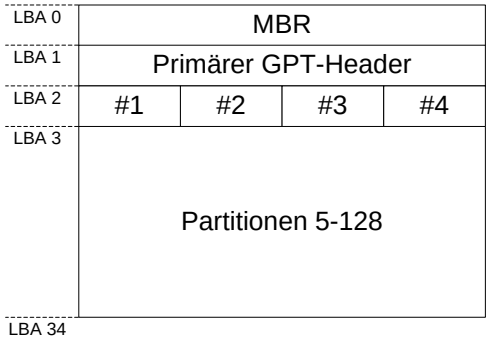

Abbildung 4.3: Struktur UEFI

4.1.2 Untersuchung auf bekannte OEM-Strings

Der im MBR befindliche Bootcode kann auf Verschlüsselung des Datenträgers oder von Partitionen hinweisen. Die für den Bootvorgang notwendigen Daten zur Entschlüsselung des Datenträgers bzw. der Betriebssystempartition müssen durch einen speziellen Bootmanager, welcher im Klartext vorliegen muss, entschlüsselt werden. Die Existenz solcher Informationen lässt auf Verschlüsselung sowie oftmals auf die verwendete Verschlüsselungssoftware schließen. Die im Nachfolgenden analysierten Verschlüsselungstools sind Bitlocker, TrueCrypt, LUKS sowie FileVault. Die Identifizierung dieser Tools soll zunächst als Voreinschätzung über den Einsatz von Verschlüsselungssoftware dienen. Diese Voreinschätzung muss in einem weiteren Schritt anhand der Durchführung statistischer Analysen bestätigt werden.

4.1.2.1 Bitlocker

Bitlocker ist in den Versionen Ultimate und Enterprise der Windows-Versionen Vista und Windows 7, der Professional- und Enterprise-Version von Windows 8 sowie einigen Windows Server-Versionen integriert und ermöglicht die Verschlüsselung der Betriebssystempartition. Ein verschlüsseltes Betriebssystem wird durch Bitlocker erst gestartet, nachdem die Integrität von Startkonfigurationsdaten sichergestellt wurde. Bitlocker sperrt das System bei Manipulation (z. B. durch Offline-Attacken oder Bootsektor-Malware) der überwachten Daten. Microsoft gibt nur wenige Details der Bitlockerspezifikation bekannt [Mic12].

Bitlocker benötigt zwei NTFS-Partitionen. Die Systempartition enthält einen Bootmanager und weitere Boot Utilities, welche die Authentizität des Benutzers sowie die Integrität des Systems überprüfen. Die Betriebssystempartition enthält das Betriebssystem. Während die Betriebssystempartition bei Anwendung von Bitlocker verschlüsselt wird, liegt die Systempartition weiterhin unverschlüsselt vor.

NVLabs, betrieben von IT-Sicherheitsforschern, hat einen unvollständigen Entwurf über die Bitlocker-Datenstrukturen unter Windows Vista veröffentlicht [KK12]. Nach ihren Analysen befinden sich Hinweise auf die Verwendung von Bitlocker nicht im MBR, sondern im ersten Cluster der verschlüsselten Partition, dem Volume Boot Record (VBR). Nach den Analysen stimmt der VBR der Bitlocker-Partition zu 96% mit dem VBR einer unverschlüsselten NTFS-Partition überein. Es ergaben sich lediglich 2 Veränderungen:

- Der OEM-String ab dem Partitionsoffset `0x03` ist gegenüber unverschlüsselter NTFS-Partitionen von `NTFS....` auf `-FVE-FS-` abgeändert.
- Die Angabe der Clusternummer der Datei `$MFT_Mirror` ab dem Partitionsoffset `0x38` wird dazu genutzt, um die Lage von `FVE_META_DATA` anzugeben. `FVE_META_DATA` ist eine Datenstruktur, welche Verschlüsselungsparameter enthält.

Eine Analyse von Bitlocker-Partitionen im Rahmen dieser Arbeit hat gezeigt, dass die Angabe `-FVE-FS-` im VBR nicht zwingend erforderlich ist bzw. abgeändert werden kann, ohne dass die Funktionalität der Bitlocker-Partition beeinflusst wird. Der o.g. OEM-String eignet sich somit nicht zur endgültigen Einstufung von mit Bitlocker verschlüsselten Partitionen.

4.1.2.2 TrueCrypt

TrueCrypt [Tru12] ist eine OpenSource-Verschlüsselungssoftware, mit welcher einzelne Partitionen bzw. ganze Datenträger verschlüsselt werden können. Vor dem Entschlüsseln wird mittels einer Pre-Boot-Authentifizierung ein Passwort abgefragt. Dieser Vorgang wird vom TrueCrypt-Bootloader durchgeführt, welcher sich im VBR der Systempartition befindet.

Der TrueCrypt-Bootloader lädt im Anschluss den Standard-Volume-Header, welcher sich im VBR der verschlüsselten Systempartition befindet. Im nächsten Schritt versucht der Bootloader, diesen zu entschlüsseln. TrueCrypt speichert keinerlei Informationen über den verwendeten Verschlüsselungsalgorithmus oder die Schlüssellänge. Diese und weitere Parameter werden mit einem Trial-and-Error-Verfahren bestimmt.

Die Parameter stimmen überein, wenn die ersten 4 Bytes des Headers entschlüsselt den ASCII-String `TRUE`. Bei erfolgreicher Übereinstimmung können die Master Keys entschlüsselt werden, welche für die Entschlüsselung der Systempartition (ausgenommen der Volume Header) notwendig sind.

Das Electronic Crime Technology Center of Excellence (ECTCOE), eine Unterorganisation des National Institute of Justice (NIJ), führte eine Evaluierung von TrueCrypt durch [NIJ11]. Dabei wurden alle von TrueCrypt angebotenen Konfigurationen angewendet und im Anschluss mit den offiziellen Angaben von TrueCrypt verglichen. Die Analyse von NIJ ergab, dass sich ein mit TrueCrypt verschlüsseltes Systemlaufwerk durch 2 Merkmale identifizieren lässt:

- Falls der Rechner mit dem TrueCrypt-Bootloader gebootet wurde, lassen sich mittels eines Memory Dumps aktive TrueCrypt-Dienste und möglicherweise das TrueCrypt-Passwort auslesen.
- Im ausgeschalteten Zustand lässt sich im VBR der Systempartition der String `TrueCrypt boot loader` feststellen.

Eine Untersuchung im Rahmen dieser Arbeit hat gezeigt, dass der OEM-String `TrueCrypt boot loader` für die Authentifikation nicht zwingend erforderlich ist und problemlos überschrieben bzw. manipuliert werden kann. Der o.g. OEM-String eignet sich somit nicht zur endgültigen Einstufung von mit TrueCrypt verschlüsselten Partitionen.

4.1.2.3 LUKS

Linux Unified Key Setup (LUKS) ist eine Spezifikation für Datenträgerverschlüsselung, welche ursprünglich nur für Linux gedacht war. LUKS spezifiziert heute ein platformunabhängiges Format, welches von unterschiedlichen Tools verwendet wird und neben der Kompatibilität auch Interoperabilität zwischen unterschiedlichen Programmen herstellt [Fru08].

Ab dem Partitionsoffset `0x00` enthalten LUKS-Partitionen die Signatur `4C 55 4B 53 BA BE` (`LUKS..`). Diese Signatur dient als Voreinstufungskriterium für LUKS-Partitionen.

4.1.2.4 FileVault 2

Omar Choudary, Joachim Metz und Felix Gröbert haben die mit Mac OS X Lion eingeführte Full Disk Encryption FileVault 2 analysiert und die Architektur von Apples

proprietärer Komplettverschlüsselung erstmals dokumentiert [CGM12].

Macintosh-Rechner nutzen ausschließlich UEFI als Firmware. Mit FileVault 2 verschlüsselte Datenträger, auch CoreStorage Volumes genannt, besitzen den GPT- Partitionstypen `53746F72-6167-11AA-AA11-00306543ECAC`. Dieser Partitionstyp lässt sich jedoch nicht nur durch FileVault 2 verschlüsselten Datenträgern zuordnen, sondern noch weiteren Partitionstypen, welche durch Mac OS X verwaltet werden können. Eine zuverlässige Klassifizierung ist somit ebenfalls nicht möglich.

4.2 Identifikation verschlüsselter Datenträger und Partitionen

Die im vorherigen Abschnitt aufgeführten Verschlüsselungstools lassen sich anhand charakteristischer Strings im MBR oder in Bootsektoren identifizieren. Die Angaben sind in vielen Fällen nicht zwingend erforderlich oder können problemlos manipuliert werden, was eine zuverlässige Erkennung ausschließt. Außerdem lässt sich anhand der Signaturen nicht immer feststellen, ob der gesamte Datenträger oder lediglich einzelne Partitionen verschlüsselt sind. Zur Bestätigung bzw. Ablehnung der im ersten Schritt erlangten Erkenntnisse wird der Datenträger in einem weiteren Analyseschritt den statistischen Tests unterzogen. Wie in Abschnitt 3.3.2.1, S. 38, beschrieben, genügt dabei eine stichprobenartige Clusteranalyse, wobei ein Datenträger bzw. eine Partition als vollverschlüsselt eingestuft wird, wenn mindestens 95% der überprüften Cluster alle Tests bestehen. Abbildung 4.4 schildert den Ablauf der Analyse.

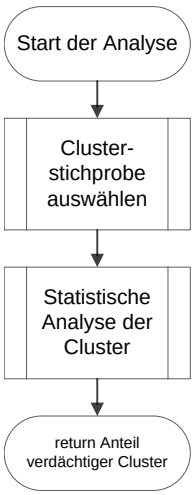

Abbildung 4.4: Abfolge der Datenanalyse zur Feststellung von Vollverschlüsselung

Die Auswahl der Cluster erfolgt stichprobenartig zur Vorbeugung antiforensischer Maßnahmen. Verwendet wurde dafür der Mersenne Twister-Zufallszahlengenerator [MN98]. Dieser wurde 1997 entwickelt und hat gegenüber anderen Pseudozufallszahlengeneratoren einige Vorzüge. Zum einen besitzt dieser mit 64-Bit einen sehr großen und für diesen Anwendungszweck benötigten Wertebereich, darüber hinaus ist er nach eigenen Angaben nicht langsamer als der Pseudozufallszahlengenerator der Standard ANSI-C-Bibliothek. Eine Überprüfung der Güte des Generators im Rahmen dieser Arbeit hat gezeigt, dass der Mersenne Twister-Pseudozufallszahlengenerator qualitativ hochwertige Zufallszahlen generiert, welche nicht von verschlüsselten Daten unterschieden werden können. Zur Überprüfung wurden 100.000.000 64-Bit Zufallszahlen generiert und den statistischen Tests unterzogen. 95,69% der von den Zufallszahlen allozierten Clustern bestanden die Tests. Die Analyse verschlüsselter Cluster in Abschnitt 3.3.1.6, S. 36, ergab einen ähnlich hohen Anteil. Als Saat für den Zufallszahlengenerator dient die Anzahl der Milisekunden seit 01.01.1970 00:00:00.000 UTC.

Die Erkennung von Partitionsverschlüsselung findet auf gleiche Art und Weise statt, jedoch nur auf dem von der Partition allozierten Speicherbereich. Auf unverschlüsselten NTFS-Partitionen findet im Anschluss eine Analyse der Daten mittels Metainformationen des Dateisystems statt, alle anderen Partitionen sowie Datenträgerbereiche werden einer gesonderten Analyse unterzogen.

4.3 Analyse von Daten mittels Metainformationen des Dateisystems

Dieser Abschnitt beschreibt die Analyse von Daten mittels Metainformationen des Dateisystems. Für das bessere Verständnis von Dateisystemen werden im Nachfolgenden zunächst strukturelle Eigenschaften von Datenträgern erläutert. Die Details stammen von Brian Carrier [Car10]. Ergänzende Informationen stammen von Microsoft [Mic12].

4.3.1 Struktur von Datenträgern

Datenträger verwalten ihre Daten in Sektoren. Ein Sektor bezeichnet dabei die kleinste zu adressierende Einheit mit einer heutzutage typischen Größe von 512 Bytes. Diese Einheiten werden über das sogenannte *Logical Block Addressing*- Verfahren ab Datenträgerbeginn mit LBA = 0 adressiert.

4.3.2 Dateisysteme

Dateisysteme sind die Schnittstelle zwischen Betriebssystem und Partitionen auf Datenträgern. Sie regeln, wie Dateien auf Datenträgern abgespeichert, organisiert und verwaltet werden und abstrahieren dabei von der auf Datenträgern vorhandenen Sektorstruktur. Dabei werden mehrere Sektoren zu einem Cluster zusammengefasst. Je größer diese Cluster sind, desto niedriger liegt der Verwaltungsaufwand bei der Speicherung großer Dateien. Außerdem ergibt sich ein geringerer Fragmentierungsgrad. Bei kleineren Dateien kann es jedoch sein, dass Cluster nicht vollständig belegt werden und somit Speicherplatz verschwendet wird. Die Metainformationen von Dateisystemen lassen sich nun dazu nutzen, um unabhängig vom Fragmentierungsrad des Datenträgers Cluster Dateien zuordnen zu können.

Es existieren unterschiedliche Dateisysteme, welche die Speicherung von Dateien unterschiedlich realisieren. Microsoft's NTFS ist proprietäres Dateisystem des Betriebssystems Windows und daher das mit Abstand bekannteste und weit verbreitetste Dateisystem. NTFS wurde 1993 veröffentlicht und hat seinen Vorgänger File Allocation Table (FAT) weitestgehend abgelöst. Aufgrund der hohen Relevanz dieses Dateisystems erfolgt die Analyse von Daten mittels Metainformationen in dieser Arbeit exemplarisch für NTFS.

4.3.2.1 Organisation des Dateisystems NTFS

Das Dateisystem NTFS besteht aus in Abbildung 4.5 dargestellten Komponenten, welche im Nachfolgenden näher beschrieben werden.

Abbildung 4.5: Komponenten des Dateisystems NTFS

NTFS-Bootsektor	Master File Table (MFT)	Daten	Kopie Master File Table

NTFS-Bootsektor Der NTFS-Bootsektor kennzeichnet den ersten Cluster einer NTFS-Partition. Die ersten 512 Byte enthalten dabei den Basic Input Output System (BIOS) Parameter Block, welcher den physikalischen Aufbau und einige dateisystemrelevante Strukturen der Partition beschreibt sowie den Initial Program Loader (IPL), welcher den in den nachfolgenden Sektoren enthaltenen Bootcode aufruft. Zu den dateisystemrelevanten Strukturen gehören u. a. Angaben über die Lokation sowie Größe der Master File Table (MFT) und Größe eines MFT-Eintrags.

Master File Table NTFS betrachtet jede Datei und jedes Verzeichnis wie in Abbildung 4.6 als eine Sammlung von Attributen. Diese sind lose definiert und werden für jede Datei zusammengefasst in einem Eintrag gespeichert. Die Einträge werden in der MFT, dem zentralen Metadatenverzeichnis, verwaltet. Die ersten 16 Einträge beinhalten dabei Metadaten von Dateien, welche für die Organisation des Dateisystems notwendig sind. Dazu zählen u. a. die `$MFT`-Datei selbst oder die `$BITMAP`-Datei, welche den Belegungsstatus jedes Clusters des Dateisystems enthält.

Am Ende einer NTFS-Partition befindet sich eine Kopie der ersten 4 MFT-Einträge, welche im Fehlerfall für die Wiederherstellung des Dateisystems notwendig sind.

MFT-Eintrag Header Ein MFT-Eintrag lässt sich mittels der Signatur `FILE` identifizieren. Durch die fixe Struktur des Headers lassen sich Informationen wie das Byteoffset zum ersten Attribut herauslesen. Über den Belegungsstatus lässt sich feststellen, ob eine Datei logisch vorhanden ist oder zum Überschreiben freigegeben (= gelöscht) wurde.

MFT-Dateiattribute NTFS erstellt für jede Datei und jedes Verzeichnis auf dem Datenträger einen MFT-Eintrag. Dieser Eintrag beinhaltet eine Folge von Attributen, welche die auf dem Datenträger abgespeicherte Datei beschreiben. Dies sind Informationen wie der Dateiname, welcher im `$FILE_NAME`-Attribut abgespeichert wird, oder Zeitstempel, welche im `$STANDARD_INFORMATION`-Attribut enthalten sind. Bei einer Dateigröße bis zu 700 Bytes wird der Dateiinhalt im `$DATA`-Attribut abgespeichert. In diesem Fall wird das `$DATA`-Attribut als *resident* bezeichnet, weil die Daten innerhalb der MFT angesiedelt sind. Dateien größer 700 Bytes werden ausgelagert, das `$DATA`-Attribut enthält stattdessen Clusterruns. Anhand der Clusterruns ergeben sich diejenigen Cluster, welche von der Datei alloziert wurden. In diesem Fall wird das `$DATA`-Attribut als *non-resident* bezeichnet, da es lediglich einen Verweis auf die eigentlichen Daten enthält. Abbildung 4.6 verdeutlicht den Aufbau eines MFT-Eintrages. Insbesondere das `$DATA`-Attribut wird im späteren Verlauf der Arbeit dazu verwendet, um Cluster Dateien zuordnen zu können. Die Attribute werden anhand eines *Attribute Type Identifiers*, einer eindeutigen numerischen Id, identifiziert.

Abbildung 4.6: Aufbau eines MFT-Eintrags (vereinfacht)

4.3.3 Verschlüsselung durch das Dateisystem

Auf der Abstraktionsebene unterhalb der Anwendungsebene findet Verschlüsselung durch das Dateisystem statt. Das Dateisystem NTFS bietet als einziges Dateisystem diese Möglichkeit durch dessen Erweiterung Encrypting File System (EFS) [Bra12], mit welcher einzelne Dateien oder Verzeichnisse unter Microsoft Windows Betriebssystemen verschlüsselt werden können. EFS agiert dabei als transparente Schicht zwischen den verschlüsselten Daten auf dem Datenträger und dem Dateisystem.

Bei Anwendung von EFS werden die im `$DATA`-Attribut abgelegten Cluster in einen Geheimtext transformiert und an der Stelle der ursprünglichen Originaldatei abgespeichert. Falls Nutzdaten innerhalb der MFT abgelegt wurden, werden diese nun ausgegliedert. Die Speicherung der für die Entschlüsselung benötigten Parameter erfolgt in dem für EFS vorgesehenen `$LOGGED_UTILITY_STREAM`-Attribut. Abbildung 4.7 zeigt einen MFT-Eintrag vor und nach der Anwendung von EFS.

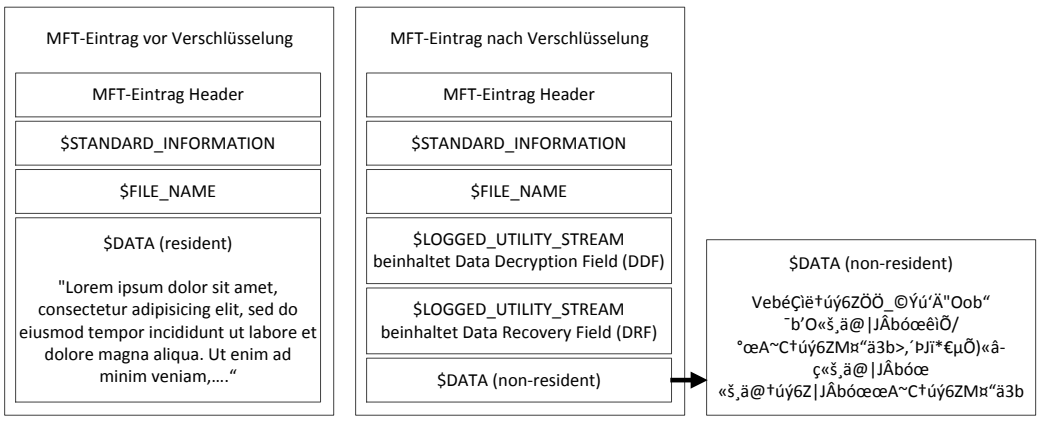

Abbildung 4.7: MFT-Eintrag vor und nach EFS-Verschlüsselung

Die Identifikation von Verschlüsselung durch das Dateisystem lässt sich anhand des `$LOGGED_UTILITY_STREAM`-Attributs, welches den Attribute-Type-Identifier `0x0100` und darüber hinaus die Namensbezeichnung `$ E F S` besitzt, durchführen. Durch EFS verschlüsselte Anwendungsdateien haben jedoch den Vorteil, dass die gesamte Datei inklusive Metainformationen verschlüsselt wird, wodurch sich eine Erkennung anhand der statistischen Tests durchführen lässt. Deshalb wird auf eine signaturbasierte Erkennung verzichtet, wodurch die Erkennung EFS-verschlüsselter Dateien möglichen antiforensischen Maßnahmen, wie z. B. einer Manipulation des *Attribute Type Identifiers* oder der Namensbezeichnung `$ E F S`, stand hält.

Eine Untersuchung im Rahmen dieser Arbeit hat gezeigt, dass gelöschte EFS-Dateien ausgenullt und der entsprechende MFT-Eintrag überschrieben wird. Dabei wurde eine Datei mittels EFS verschlüsselt und der entsprechende MFT-Eintrag gesichert. Im Anschluss wurde die verschlüsselte Datei gelöscht und der daraus resultierende MFT-Eintrag mit dem zuvor gesicherten MFT-Eintrag verglichen. Der geänderte MFT-Eintrag ließ keinen Rückschluss auf eine zuvor mittels EFS verschlüsselte Datei zu, sodass gelöschte EFS-Dateien weder mit einem signaturbasierten Ansatz noch mit statistischen Tests identifiziert werden können.

4.3.4 Datenanalyse anhand der Master File Table

Abbildung 4.8 zeigt den Ablauf der Analyse von NTFS-Partitionen. Durch Auslesen des NTFS-Bootsektors wird zunächst die Lage und das Layout der MFT bestimmt. Die enthaltenen Einträge mit der Signatur `FILE` werden sequentiell geparst. Dabei sind vor allem Cluster-Runs im `$DATA`- sowie das `$FILE_NAME`-Attribut von Interesse. Im MFT-Eintrag-Header wird ebenfalls überprüft, ob der Eintrag zum Überschreiben freigegeben wurde. In diesem Fall wird anhand der `$BITMAP`-Datei überprüft, ob die freigegebenen Cluster bereits wiederbelegt wurden. Jedes Bit der `$BITMAP`-Datei repräsentiert dabei einen Cluster des Dateisystems. Bei gegebener Clusternummer wird das entsprechende Bit gemäß nachfolgendem Beispiel ausfindig gemacht:

```
Cluster 13:
Byte: 13 / 8 = 1
Offset: 13 - 8 x 1 = 5
```

Somit wird Byte 1 der `$BITMAP`-Datei ausgelesen und das Bit 5 überprüft. Ein gesetztes Bit bedeutet dabei, dass Cluster 13 belegt ist. Ist dies der Fall, wird das Cluster im Zusammenhang mit der entsprechenden Datei nicht weiter untersucht und auch nicht extrahiert. Die Datei enthält somit Lücken an der Stelle des jeweiligen Clusters.

Selbst wenn alle Cluster immer noch als unbelegt markiert sind ist es möglich, dass diese zwischendurch belegt waren. Ein Zusammenhang zwischen Clustern und nicht-allozierten Datein ist somit nicht zweifelsfrei herstellbar, sodass möglicherweise Daten untersucht und extrahiert werden, welche nicht von der eigentlich zu untersuchenden Datei stammen.

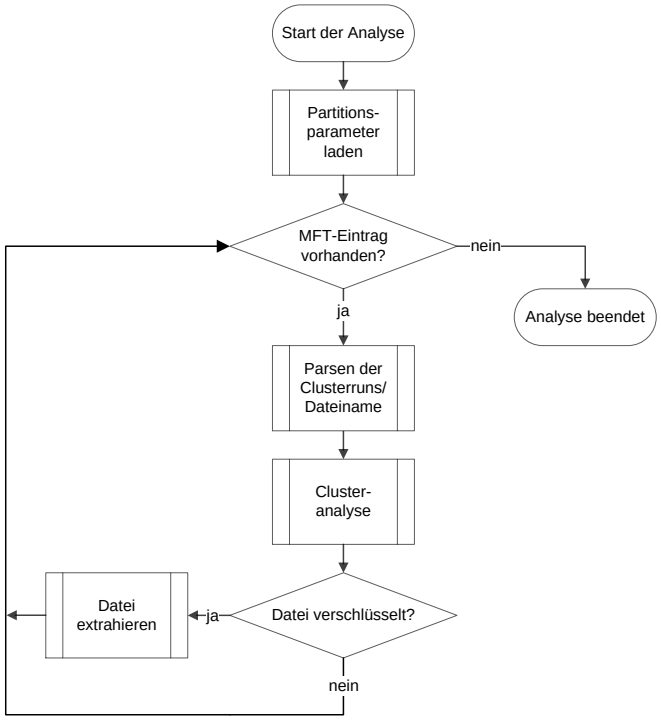

Abbildung 4.8: Ablauf der Analyse von NTFS-Partitionen

Die Cluster werden im Anschluss den statistischen Tests unterzogen. Wie in Abschnitt 3.3.2.3, S. 39, festgelegt, erfolgt eine Extraktion der überprüften Datei bei einem Anteil positiver Cluster von über 50% oder Feststellung einer der zuvor identifizierten Signaturen.

4.4 Analyse nicht-allozierter Datenträgerbereiche

Dieser Analyseschritt betrifft in erster Linie Datenträgerbereiche, welche nicht durch Partitionen alloziert wurden. Diese Bereiche lassen sich aus dem im MBR hinterlegten Partitionslayout ableiten. Darüber hinaus werden Partitionen mit Dateisystemen, welche nicht interpretiert werden können, dieser Analyse unterzogen. Zur Erkennung gelöschter Dateien in diesen Bereichen findet in der Praxis das Prinzip File Carving

seine Anwendung. Damit lassen sich Dateien unabhängig vom verwendeten Dateisystem erkennen und möglicherweise vollständig wiederherstellen.

4.4.1 Einsatz von File Carving

Je nach Dateisystem werden Dateien bzw. Partitionen beim Löschen nicht physisch überschrieben, sondern die allozierten Bereiche lediglich als frei verfügbar markiert. Ist dieser Bereich zum Zeitpunkt der Datenwiederherstellung noch nicht überschrieben worden, lassen sich diese Dateien möglicherweise wiederherstellen. Der Wiederherstellungserfolgt hängt dabei vom Fragmentierungsgrad der Datei ab. File Carving nutzt dabei keine Metainformationen des Dateisystems und hat darüber hinaus den Vorteil, dass es über Partitionsgrenzen hinausgeht. Dazu zählen die Bereiche

- nicht-allozierter Speicherbereich eines Datenträgers (Partition Slack),
- Cluster, welche vom Dateisystem als defekt markiert wurden,
- Cluster, welche vom Dateisystem nicht alloziert wurden,
- Host Protected Area (HPA) oder Device Configuration Overlay (DCO), welche normalerweise für die Systemwiederherstellung und Sicherung von Konfigurationsdaten vorhanden sind.

In dieser Arbeit erfolgt eine Erkennung verschlüsselter Daten anhand der statistischen Tests. Im Vergleich zu File Carving erfolgt somit eine Identifizierung unabhängig von der Signatur. Aufgrund von Metadaten, welche die Erkennung verschlüsselter Anwendungsdateien erschweren, beschränkt sich der nachfolgend entwickelte Ansatz auf die Erkennung von TrueCrypt-Containern, welche ausschließlich aus verschlüsselten Nutzdaten bestehen. Eine Erkennung solcher Daten ist mittels des klassischen File Carvings nicht möglich, da diese keine Signatur besitzen. Die Anwendung der nachfolgend beschriebenen Dateiwiederherstellung erfolgt auf allen Datenträgerbereichen, welche nicht durch Partitionen alloziert wurden. Diese Bereiche ergeben sich aus den im MBR hinterlegten Partitionen. Darüber hinaus findet die Dateiwiederherstellung auf Partitionen statt, welche nicht durch das Dateisystem NTFS alloziert wurden.

4.4.2 Herausforderungen der Dateiwiederherstellung

Der hier entwickelte Ansatz zur Erkennung verschlüsselter Daten auf nicht-allozierten Bereichen bringt, analog zum klassischen File Carving, einige Herausforderungen mit

sich mit. Zuerst muss eine Datei als solche auf dem Datenträger identifiziert und überprüft werden, ob diese noch intakt ist. Schließlich wird die Datei aus dem Image extrahiert. Während für diesen Vorgang beim klassischen File Carving dateitypenspezifisches Wissen notwendig ist, erfolgt eine Identifikation in dieser Arbeit mit Hilfe der aufgeführten statistischen Tests. Bekannte Carving-Tools wie Foremost bringen in Bezug auf die Fragmentierung einige Einschränkungen mit sich. Diese Tools carven Dateien, indem sequentielle Bytebereiche zwischen Header und Footer extrahiert und zusammengehängt werden. Das Ergebnis der Extraktion hängt davon ab, ob und wie stark eine Datei fragmentiert ist.

Da die Analyse ohne Metadaten des Dateisystems erfolgt, lässt sich dieses Problem auch mit dem hier entwickelten Ansatz nicht lösen. So lassen sich verschlüsselte Datenfragmente zwar identifizieren, auf einen möglichen Zusammenhang bzw. die Reihenfolge von Fragmenten kann jedoch nicht geschlossen werden.

4.4.3 Der Wiederherstellungsprozess

Der Wiederherstellungsprozess beginnt zunächst mit einer Clusteranalyse des gesamten nicht-allozierten Bereichs. Da einzelne Cluster keinen Aufschluss über Verschlüsselung geben können, erfolgt die Identifikation durch Betrachtung verdächtiger Cluster in einem bestimmten Intervall, nachfolgend Fenster oder *sliding window* genannt. Die vorherigen Kapitel haben gezeigt, dass ein sehr hoher Anteil verschlüsselter Nutzdaten den statistischen Eigenschaften genügt. Von diesen Daten wird erwartet, dass das *sliding window* eine hohe Auslastung besitzt. Von nicht verschlüsselten bzw. komprimierten Daten wird eine geringere Auslastung erwartet.

Der Erkennungsansatz sieht nun vor, dass die Auslastung des *sliding window* als Entscheidungskriterium über verschlüsselte Daten dient.

4.4.3.1 Vorverarbeitung

Da bei nicht-allozierten Datenträgerbereichen keine Metainformationen des Dateisystems vorliegen bzw. bei unbekannten Dateisystemen diese Metadaten nicht interpretiert werden können, wird die statistische Analyse zunächst auf dem gesamten nicht-allozierten bzw. durch unbekannte Dateisysteme allozierten Speicherbereich durchgeführt. Diese Analyse erfolgt auf Clusterebene; die Clustergröße wird standardmäßig mit 4096 Bytes, einer derzeit typischen Clustergröße, angenommen. Die aufgeführten statistischen Tests sind dafür geeignet und wurden entsprechend parametrisiert. Eine Anwendung auf kleineren Clustergrößen hat Auswirkungen auf den

Chi-Quadrat-Test, da in diesem Fall die erforderlichen 5 hypothetischen Beobachtungen pro Kategorie nicht erreicht werden können. Eine Anwendung der Tests auf kleineren Einheiten führt somit möglicherweise zu false positives.

In der Praxis geht der Trend zu größeren Datenträgern mit Clustergrößen eines Vielfachem von 4096 Bytes, sodass der Dateibeginn korrekt identifizieren werden kann. Bei kleineren Speichereinheiten muss die Analyse mit unterschiedlichen Offsets ab Partitionsbeginn erfolgen, um den Dateibeginn korrekt zu erfassen.

Die Testergebnisse der überprüften Cluster werden sequentiell abgespeichert, wobei ein verdächtiges Cluster durch den Wert 1, alle anderen Cluster durch den Wert 0 repräsentiert werden.

4.4.3.2 Sliding Window-Prinzip

Nach der Analyse des gesamten Speicherbereichs wird die Anzahl verdächtiger Cluster im *sliding window* ermittelt. Das Fenster bewegt sich bei diesem Schritt sequentiell von Beginn an über die Testergebnisse. Bei Überschreitung einer bestimmten Auslastung des Fensters wird der Beginn, bei erstmaliger Unterschreitung das Ende verschlüsselter Nutzdaten vermutet. Abbildung 4.9 verdeutlicht diesen Vorgang.

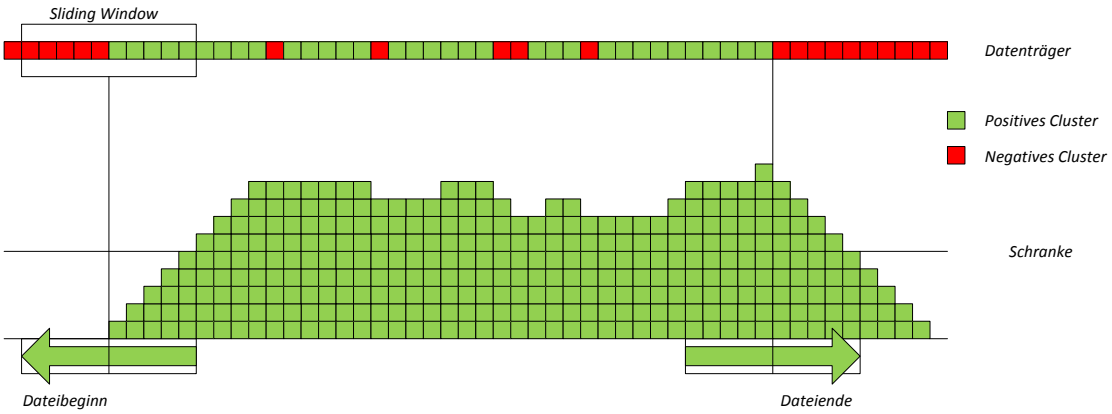

Abbildung 4.9: File Carving mittels Sliding Window

Zur Durchführung des Sliding Window-Prinzips müssen zunächst mögliche Fenstergrößen sowie Übergänge, welche auf ein mögliches Dateibeginn bzw. -ende hinweisen, ermittelt werden.

Ermittlung einer Fenstergröße Nachfolgende Tests ermitteln die Auslastungen des *sliding window* bei Anwendung unterschiedlicher Fenstergrößen w. Die Durchführung erfolgt auf den bisher betrachteten Dateiformaten aus Tabelle 3.3, S. 32 -

sowohl verschlüsselt als unverschlüsselt. Die Tabellen notieren den relativen Anteil an Fenstern, welche jeweils die Auslastung Σ besitzen.

Tabelle 4.1 enthält die Testergebnisse bei Anwendung der Fenstergröße $w = 4$. Somit werden 4 aufeinanderfolgende Cluster betrachtet. Die Ergebnisse zeigen, dass die verschlüsselten Formate DOCX, PDF und ZIP Untersequenzen der Länge 4 besitzen, welche kein positives Cluster enthalten. Diese Dateibereiche können bei dieser Fenstergröße durch das *sliding window* nicht erfasst werden.

Bei Betrachtung der unverschlüsselten Formate zeigt sich, dass ein nicht unerheblicher Anteil der Untersequenzen 4 positive Cluster enhält. Unabhängig von einer Mindestauslastung, welche als Schranke für verschlüsselte Datenbereiche festgelegt werden muss, führen die unverschlüsselten Daten zu false negatives. Um diese false negatives zu senken, wurde die Fenstergröße angehoben um Entscheidungskriterien zu finden, welche die false positive-Rate nur gering steigern. Dazu wurde mit weiteren Fenstergrößen versucht, die Granularität zu verfeinern, um einen geeigneten Kompromiss zwischen false negatives und false positives zu erhalten. Dabei muss beachtet werden, dass nur verschlüsselte Daten mit einer Größe (in Clustern) erkannt werden, welche die Mindestauslastung (in Clustern) übersteigen. Durch Anhebung der Schranke können außerdem nur verschlüsselte Dateien mit einem bestimmten Anteil positiver Cluster erkannt werden. Zur Erkennung verschlüsselter Anwendungsdateien eignet sich diese Methode daher nicht.

Tabelle 4.1: Identifizierung einer Mindestauslastung bei einer Fenstergröße $w = 4$

	verschlüsselte Formate				unverschlüsselte Formate		
Σ	DOC	PDF	ZIP	TC	DOC	PDF	ZIP
0	1,05322	1,55766	1,21827	0	80,4709	67,8871	21,1220
1	0,87768	3,66450	0,98018	0,031263	7,26262	12,9038	9,58768
2	2,02328	8,51722	2,68699	1,229680	5,48171	8,71884	15,5809
3	16,9346	25,6615	18,2031	15,57940	3,63131	5,60732	26,8547
4	79,1112	60,5991	76,9055	83,15960	3,15351	4,88295	26,8547

Tabelle 4.2 zeigt die Testergebnisse bei Durchführung der Tests mit einer Fenstergröße $w = 6$. Somit werden 6 aufeinanderfolgende Cluster betrachtet. Anhand der Ergebnisse lässt sich eine geeignetere Schranke finden, welche bei Überschreitung bzw. Unterschreitung auf mögliche verschlüsselte Nutzdaten hinweisen kann. TC-Container lassen sich im defragmentierten Zustand vollständig wiederherstellen, wenn als Kriterium ein Übergang der Auslastung von $\Sigma = 2$ auf $\Sigma = 3$ und umgekehrt angenommen

wird. Bei dieser Annahme lassen sich bei den untersuchten nichtverschlüsselten Formaten viele (kurze) false negatives feststellen; ein Großteil aller ZIP-Untersequenzen besitzt eine ähnlich hohe Auslastung, sodass weitere Fenstergrößen und erweiterte Maßnahmen, wie ein signaturbasierter Ansatz, in Betracht gezogen werden können. Weiterhin muss beachtet werden, dass mit steigender Mindestauslastung Dateien, dessen Größe in Clustern unterhalb dieser Grenze liegt, nicht betrachtet werden.

Tabelle 4.2: Identifizierung einer Mindestauslastung bei einer Fenstergröße $w = 6$

	verschlüsselte Formate				unverschlüsselte Formate		
Σ	DOC	PDF	ZIP	TC	DOC	PDF	ZIP
0	0,72075	0,80895	0,94004	0	77,4351	63,6471	17,7756
1	0,44354	1,34825	0,53496	0	6,64697	10,4383	7,00598
2	0,70227	2,95616	0,69885	0	5,26544	8,14976	6,58439
3	1,04417	4,86368	1,21525	0,17197	3,71014	6,78653	10,2177
4	3,38200	11,3053	4,38480	2,74101	2,51977	4,62972	17,4843
5	22,4543	29,2020	23,5629	21,2038	2,31123	3,65834	23,2485
6	71,2530	49,5156	68,6632	75,8833	2,11139	2,69024	17,6836

Tabelle 4.3 zeigt die Auslastungen des *sliding window* bei einer Fenstergröße $w = 8$.

Tabelle 4.3: Identifizierung einer Mindestauslastung bei einer Fenstergröße $w = 8$

	verschlüsselte Formate				unverschlüsselte Formate		
Σ	DOC	PDF	ZIP	TC	DOC	PDF	ZIP
0	0,56377	0,52942	0,74218	0	74,9804	60,5427	15,6854
1	0,19409	0,62931	0,44840	0	6,36135	9,93183	6,02576
2	0,36969	1,29857	0,42985	0	5,44017	6,69477	4,63048
3	0,62847	2,21756	0,58562	0	3,50222	5,72990	5,20546
4	0,89649	3,82579	0,76383	0,01042	2,80699	5,13716	7,64336
5	1,30314	5,99341	1,59260	0,44299	1,90319	5,01202	12,3275
6	5,10166	14,0745	6,38587	4,60183	1,52950	3,03619	17,1190
7	26,6451	30,7662	27,5845	25,6984	1,98140	2,25245	19,2656
8	64,2976	40,6653	61,4652	69,2464	1,49474	1,66299	12,0975

Die Auslastungen des *sliding window* zeigen selbst bei einer Fenstergröße $w = 8$ einen Anteil unverschlüsselter Formate bei $\Sigma = 8$, d.h. mindestens 8 aufeinanderfol-

gende (fälschlicherweise) positiv eingestufte Cluster. Umgekehrt existieren weiterhin Anteile verschlüsselter Formate bei $\Sigma = 0$, d.h. mindestens 8 aufeinanderfolgende negative Cluster. Für TrueCrypt-Container (TC), welche keinerlei unverschlüsselte Metadaten enthalten, lassen sich zuverlässige Schranken definieren. Dabei wird der Beginn eines TrueCrypt-Containers vermutet, wenn die Auslastung von $\Sigma = 3$ auf $\Sigma = 4$ steigt, umgekehrt wird das Ende vermutet, wenn die Auslastung von $\Sigma = 4$ auf $\Sigma = 3$ sinkt. Im defragmentierten Zustand ließen sich TrueCrypt-Container somit vollständig extrahieren. Da jedoch keine Unterscheidung von verschlüsselten und zufälligen Daten möglich ist, lassen sich TrueCrypt-Container lediglich vermuten.

Aufgrund der erlangten Erkenntnisse beschränkt sich der *sliding window*-Ansatz in dieser Arbeit auf die Erkennung und Wiederherstellung gelöschter TrueCrypt-Container, welche sich mittels des klassischen File Carving nicht erkennen lassen.

4.5 Zusammenfassung

Dieses Kapitel zeigte den Ablauf der Verschlüsselungserkennung auf allen Bereichen eines Datenträgers. Dabei dient zunächst eine Signaturüberprüfung zur Voreinstufung von Datenträgervoll- bzw. Partitionsverschlüsselung. OEM-Strings im MBR bzw. VBR eignen sich dabei als Einstufungskriterium für angewandte Verschlüsselung. Jedoch ist oftmals keine eindeutige Unterscheidung zwischen Datenträgervoll- bzw. Partitionsverschlüsselung möglich. Darüber hinaus lässt sich eine Manipulation der Angaben ohne Auswirkung auf die Funktionsfähigkeit der Partition in vielen Fällen durchführen. Aufgrund dieser Tatsache wird, wie in Abbildung 4.1 dargestellt, unabhängig vom MBR-Analyseergebnis eine Clusteranalyse durchgeführt. Die Clusteranalyse umfasst eine stichprobenartige Analyse aller Clustern, wobei ein Datenträgerbereich als verschlüsselt eingestuft wird, wenn mindestens 95% aller analysierten Cluster die Tests bestehen.

Bei der Analyse unverschlüsselter NTFS-Partitionen wird auf Metainformationen des Dateisystems zurückgegriffen, um unabhängig vom Fragmentierungsgrad sowie Belegungsstatus Dateien analysieren zu können. Als Grundlage dienen wiederum die statistischen Tests erweitert um eine Signaturerkennung, welche auf zuvor identifizierte Signaturen verschlüsselter Anwendungsdateien überprüft. Die Erweiterung um die Signaturerkennung hat zwar Auswirkungen auf die Performanz der Analyse, jedoch ergibt sich durch die Kombination von Signaturerkennung und statistischer Tests auch ein Vorteil. Veränderungen an der Signatur haben keine Auswirkungen auf das Ergebnis der statistischen Analyse. So könnten Manipulationen aufgedeckt werden,

insofern der Metadatenanteil der Datei nicht überwiegt und eine Erkennung mittels statistischer Tests zulässt.

Bei der Betrachtung nicht-allozierter Datenträgerbereiche sowie bisher nicht interpretierbaren Dateisystemen kommt ein entwickelter Ansatz zur Identifizierung verschlüsselter Nutzdaten zum Einsatz. Während hier die klassische Signaturerkennung aufgrund der Konstruktion vieler Dateiformate nicht wegzudenken ist, bleibt sie bei der Erkennung rein verschlüsselter Nutzdaten, wie im Beispiel TrueCrypt, wirkungslos. Es hat sich gezeigt, dass das *sliding window*-Prinzip aufgrund der Konstruktionseigenschaften vieler Dateiformate viele false negatives erzeugt bzw. verschlüsselte Datenströme aufspaltet. Aus diesem Grund wird der Wiederherstellungsprozess in dieser Arbeit dazu verwendet, um TrueCrypt-Container bzw. Zufallsdaten zu identifizieren und zu extrahieren.

5 Implementierung und Evaluierung der Datenträgeranalyse

Dieses Kapitel beschreibt die Umsetzung der Datenträgeranalyse aus Kapitel 4 in ein kommandozeilenbasiertes Tool sowie dessen Evaluierung. Für die Entwicklung wurde das Framework QT herangezogen, welches zunächst erläutert wird.

Im Anschluss an die Umsetzung erfolgt die Evaluierung. Dafür dienen synthetisch generierte Datenträgerimages welche es gilt, ggf. auf unterschiedlichen Abstraktionsebenen korrekt zu klassifizieren.

5.1 Implementierung eines Tools zur Datenträgeranalyse in QT

QT (lies englisch *cute*) ist ein weitverbreitetes Framework, welches zur plattformunabgängigen Entwicklung von Programmem mit grafischen Benutzeroberflächen (GUI) als auch zur Enticklung von kommandozeilenbasierten Applikationen verwendet wird.

QT ist eine C++-Klassenbibliothek und wurde designed, um Applikationen und Benutzerschnittstellen für unterschiedliche Zielplattformen ohne Anpassung übersetzen zu können. Die Entwicklungsumgebung, der QT creator, stellt Werkzeuge für den gesamten Applikations-Entwicklungszyklus bereit, von der Erstellung von Projekten bis zur Anpassung an die Zielplattform [Qt 12].

5.2 Struktur und Aufbau des Quellcodes

Die in dieser Arbeit entwickelten Methoden zur Erkennung verschlüsselter Daten auf unterschiedlichen Abstraktionsebenen eines Datenträgers wurden als kommandozeilenbasiertes Tool umgesetzt. Die Analyse durch das entwickelte Tool beschränkt sich dabei auf die Analyse von Datenträgerimages. Der Quellcode ist dabei sehr einfach strukturiert. Abbildung 5.1, S. 67, gibt einen Überblick über alle erstellten Klassen in einem Unified Modeling Language (UML)-Diagramm, außerdem erfolgt nachfolgend

eine kurze Beschreibung aller erstellten Klassen. Der Quellcode befindet sich auf der CD-ROM im Anhang dieser Arbeit.

Beschreibung der Klasse manager

Die Klasse `manager` regelt den Abauf der Datenträgeranalyse nach Abbildung 4.1, S. 46. Bei Ausführung werden zunächst die im MBR hinterlegten Partitionen ausgelesen. Im Anschluss werden Methoden der Klasse `media` aufgerufen, welche auf Datenträgervoll- bzw. Partitionsverschlüsselung überprüfen. Abhängig vom Testergebnis und Partitionstypen werden mit der Klasse `ntfs` NTFS-Partitionen überprüft oder die Datenanalyse ohne Metadaten des Dateisystems durchgeführt. Das Partitionslayout sowie alle Testergebnisse werden der Klasse `logger` übergeben, welche die Ereignisse in einer Comma-seperated values (CSV)-Datei persistent protokolliert.

Beschreibung der Klasse media

Die Klasse `media` beinhaltet Funktionalitäten, welche unabhängig vom Dateisystem auf den Datenträger angewendet werden können. Dies sind zum einen Funktionen zur Überprüfung auf Datenträgervoll- bzw. Partitionsverschlüsselung. Dabei werden statistische Tests der Klasse `statistics` verwendet, um einen Anteil verdächtiger Cluster eines übergebenen Datenträgerbereichs zu erhalten. Die Analyse findet dabei auf einer Stichprobe aller Cluster statt; wie in Abschnitt 3.3.2.1, S. 38, festgelegt beträgt die Stichprobe 1% aller Cluster, welche mittels des Mersenne Twister-Pseudozufallszahlengenerators [NM13] zufällig ausgewählt werden. Die Klasse enthält darüber hinaus Funktionen für die Durchführung von Datenanalysen in unbekannten Dateisystemen oder nicht-allozierten Datenträgerbereichen. Dabei wird der gesamte Datenträgerbereich statistischen Tests unterzogen und im Anschluss mittels des *sliding window*-Ansatzes aus Abschnitt 4.4, S. 57, mögliche TrueCrypt-Container identifiziert und extrahiert.

Beschreibung der Klasse statistics

Die Klasse `statistics` stellt das Herzstück des Programmes dar. Es beinhaltet die vorgestellten statistischen Tests und liefert Aussagen über den Anteil verschlüsselter Einheiten eines übergebenen Datenträgerbereichs. Bei erstmaligem Aufruf werden die hypothetischen Auftrittswahrscheinlichkeiten berechnet, welche für die Ausführung des Chi-Quadrat-Tests notwendig sind. Die Funktion `performAnalysis()` beinhaltet die eigentlichen Tests. Bei Ausführung wird der Datenträgerbereich übergeben, auf

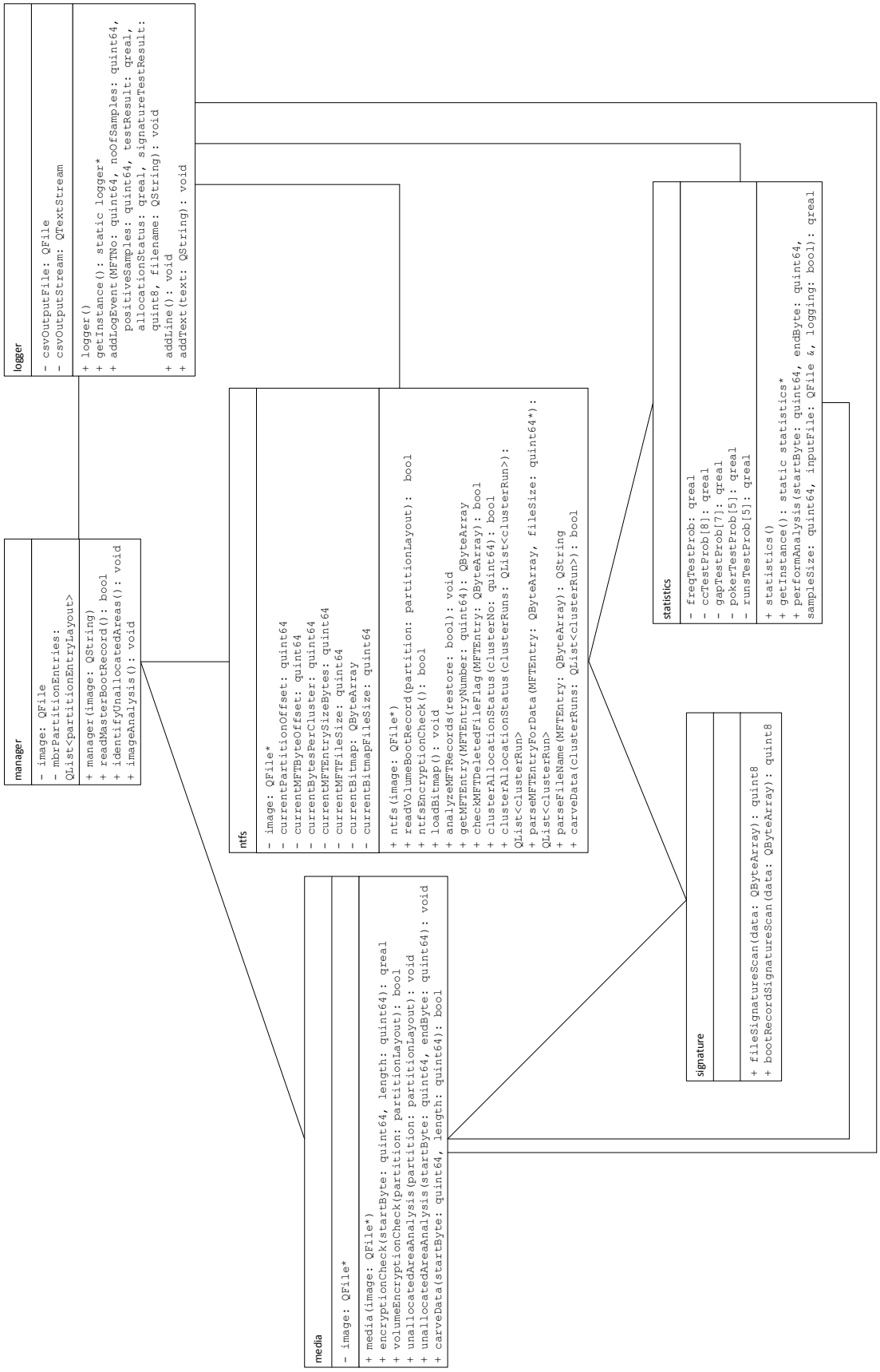

Abbildung 5.1: Klassendiagramm des Tools zur Datenträgeranalyse in UML

welchem die Tests durchgeführt werden sollen. Dieser wird in Einheiten der Größe `sampleSize` unterteilt. Diese Einheiten werden unabhängig voneinander den Tests unterzogen. Der Rückgabewert der Funktion ist der relative Anteil positiv getesteter Einheiten. Zur Klassifizierung einer Einheit ist die Berechnung des p-Wertes notwendig. Die Berechnung des p-Wertes erfolgt anhand der berechneten χ^2-Teststatstik und der Anzahl der Freiheitsgrade v. Das Modul für die Berechnung des p-Wertes stammt von Gary Perlman vom Wang Institut [Per12].

Beschreibung der Klasse ntfs

Die Klasse `ntfs` stellt Methoden bereit, welche für Analysen von Partitionen mit dem Dateisystem NTFS notwendig sind. Bei Initialisierung werden zunächst die übergebenen Partitionsparameter in Byteoffsets umgerechnet. Nach der Lokalisierung der `$MFT`-Datei werden die enthaltenen MFT-Einträge sequentiell geparst. Die in jedem Eintrag hinterlegten Cluster-Runs liefern dabei eine Zuordnung von Clustern und Dateien. Diese Runs werden den statistischen Tests der Klasse `statistics` unterzogen. Das Testergebnis ist der relative Anteil positiv getesteter Cluster. Wie in Abschnitt 3.3.2.3, S. 39, festgelegt, wird die Datei ab einem positiven Clusteranteil von 50% aus dem Image extrahiert.

Beschreibung der Klasse signature

Die Klasse `signature` beinhaltet Funktionalitäten zur Überprüfung auf Signaturen in übergebenen Datenträgerbereichen. Diese Überprüfung umfasst zum einen Signaturen auf der Anwendungsebene, welche auf verschlüsselte Anwendungsdateien der Formate Office (DOCX, XLSX,...), PDF und ZIP hinweisen. Zum andern wird auf Signaturen geprüft, welche auf der Datenträger- bzw. Partitionsebene auf Datenträgervoll- bzw. Partitionsverschlüsselung hinweisen.

Beschreibung der Klasse logger

Die Klasse `logger` nimmt jegliche Testergebnisse aller Klassen entgegen und speichert diese persistent in einer CSV-Datei, um den Ablauf der Datenträgeranalyse nachträglich nachvollziehen zu können.

5.3 Evaluierung des Tools zur Datenträgeranalyse

Ziel der Evaluierung ist es, den Ablauf der Datenträgeranalyse nach Abbildung 4.1, S. 46, einzuhalten, um eine Analyse des gesamten Datenträgerbereichs zu gewährleisten. Dafür ist es notwendig, mögliche Anwendungsfälle zu generieren und diese mittels des implementierten Tools richtig zu klassifizieren. Darüber hinaus wird die Ausführungsdauer der Analyse festgehalten.

5.3.1 Synthetisch generierte Datenträgerimages

Die Evaluierung erfolgt anhand synthetisch generierter Datenträgerimages. Die darauf enthaltenen verschlüsselten Daten, welche es gilt, im Rahmen der Evaluierung zu erkennen, sind somit bekannt und dienen als Vergleich mit den erzielten Ergebnissen. Darüber hinaus werden mögliche false negatives dokumentiert. Die generierten Datenträgerimages sollen dabei viele unterschiedliche Anwendungsfälle abdecken, welche sich aus einer Kombination unterschiedlicher Abstraktionsebenen ergeben. Die generierten Datenträgerimages werden in Tabelle 5.1 vorgestellt.

Die Generierung der Images erfolgte mittels handelsüblicher Rechner und den Microsoft Betriebssystemen Windows 7 Professional und Windows 8 Professional, da der Schwerpunkt der Analyse auf NTFS-Dateisystemen liegt. Die Installation eines Betriebssystems erfolgte über das Standard-Setup auf einem ausgenullten Datenträger. Im Rahmen des Setups erfolgte die Partitionierung des Datenträgers. Die Betriebssystempartition wurde bei allen Datenträgerimages mit einer Größe von 10GB, welche für die Installation sowie Abspeicherung verschlüsselter Daten ausreicht, erstellt. Bei der Installation von Windows 8 entstand automatisiert eine weitere NTFS-Partition, welche einen Bootmanager sowie Tools zur Systemwiederherstellung enthält. Auf allen Partitionen wurde eine Clustergröße von 4096 Bytes festgelegt. Im Anschluss an die Installation erfolgte die Erstellung einer weiteren Partitionen, entweder mit dem Dateisystem NTFS oder FAT, manuell über die Datenträgerverwaltung des Betriebssystems. Um Datenträgervollverschlüsselung zu erreichen, erfolgte die Installation und Ausführung von TrueCrypt. Bitlocker, welches in Windows 8 enthalten ist, wurde bei einem Image zur Partitionsverschlüsselung verwendet. Auf allen unverschlüsselten NTFS-Partitionen erfolgte die Abspeicherung verschlüsselter Anwendungsdaten der Formate DOCX, PDF und ZIP sowie TrueCrypt-Container. Auf FAT-Partitionen sowie nicht-allozierten Datenträgerbereichen erfolgte die Abspeicherung von TrueCrypt-Containern. Mit dem Unix-Befehl `dd` wurden Images aus den Datenträgern erstellt, welche im Anschluss mit dem Tool untersucht werden konnten.

Die einzelnen Images sowie die Testergebnisse werden im Nachfolgenden vorgestellt. Ziel der Analyse durch das Tool ist nun die korrekte Interpretation des Datenträgerlayouts und das Ausführen der korrekten Analysemethoden auf den entsprechenden Datenträgerbereichen.

Tabelle 5.1: Übersicht synthetisch generierter Datenträgerimages

Image	Datenträgeraufteilung
#1	Partition #0: NTFS, 350MB, verschlüsselt (TrueCrypt) Partition #1: NTFS, 10.000MB verschlüsselt (TrueCrypt)
#2	Partition #0: NTFS, 10.000MB verschlüsselt (Bitlocker) Partition #1: NTFS, 5.000MB unverschlüsselt Partition #2: NTFS, 350MB unverschlüsselt
#3	Partition #0: NTFS, 350MB unverschlüsselt Partition #1: NTFS, 10.000MB unverschlüsselt Partition #2: FAT32, 5.000MB unverschlüsselt
#4	Nicht-allozierter Datenträgerbereich: 10.000MB, unverschlüsselt
#5	Partition #0: FAT16, 968MB, unverschlüsselt

Anhand der Images wird zunächst überprüft, ob das Datenträgerlayout richtig interpretiert und der Ablauf der Datenträgeranalyse nach Kapitel 4 eingehalten wird. Darüber hinaus gilt die richtige Klassifizierung der Datenträgerbereiche sowie die Identifizierung und Extraktion verschlüsselter Dateien. Dafür werden nachfolgend Szenarien detailliert erläutert.

Zunächst erfolgt mit Datenträgerimage #1 die Evaluierung auf der Datenträgerebene, welche vollständig verschlüsselte Datenträger betrachtet. Beobachtet wird hierbei das Ergebnis der Clusteranalyse auf Datenträgerebene sowie die vorgenommene Klassifizierung der Partitionen.

Datenträgerimage #2 zielt auf die Erkennung von Verschlüsselung auf Partitionsebene ab. Dabei liegt eine Partition verschlüsselt, zwei weitere Partitionen unverschlüsselt vor. Die unverschlüsselten Partitionen sind mit dem Dateisystem NTFS formatiert. Partition #1 beinhaltet verschlüsselte Anwendungsdaten. Ziel ist die korrekte Klassifizierung der verschlüsselten Partition sowie die Identifizierung und Extraktion verschlüsselter Anwendungsdateien.

Bei einem weiteren Image, Datenträgerimage #3, erfolgt die Untersuchung auf unterschiedlichen Abstraktionsebenen. Zunächst existiert eine NTFS-Partition mit Windows-Betriebssystem und verschlüsselten Anwendungsdateien. Auf einer weite-

ren Partition mit dem Dateisystem FAT befinden sich TrueCrypt-Container. Ziel ist zum einen die Identifizierung und Extraktion verschlüsselter Anwendungsdateien mittels der MFT-Analyse auf NTFS-Partitionen, zum anderen die Identifizierung und Extraktion von TrueCrypt-Containern auf der FAT-Partition mittels der Wiederherstellung verschlüsselter Nutzdaten.

Das Datenträgerimage #4 besteht aus gelöschten Partitionen, sodass hier ebenfalls die Analyse zur Wiederherstellung verschlüsselter Nutzdaten angewendet wird. Ziel ist die Identifizierung und Extraktion von TrueCrypt-Containern.

Datenträgerimage #5 ist das Speicherabbild einer SD-Karte aus einer Digitalkamera. Schwerpunkt liegt, ähnlich wie bei vorherigen Images, auf der Extraktion von TrueCrypt-Containern mittels des Wiederherstellungsansatzes. Insbesondere bei diesem Datenträgerimage gilt eine Beobachtung von false negatives, welche durch das Dateiformat JPG möglicherweise entstehen.

5.3.2 Testergebnisse

Die Datenträgerimages aus Tabelle 5.1 wurden im Rahmen der Evaluation automatisiert analysiert. Als Ziel gilt zunächst die richtige Interpretation des Datenträgerlayouts, im Anschluss erfolgen Tests auf Verschlüsselung einzelner Partitionen bzw. Datenträgervollverschlüsselung. Je nach Ausgang der Tests werden Daten entweder mithilfe von Metainformationen (im Falle von NTFS-Partitionen) oder ohne Metainformationen des Dateisystems analysiert. Durch Letzteres erfolgt eine Analyse des Speicherbereichs zur Wiederherstellung verschlüsselter Nutzdaten.

5.3.2.1 Testimage #1

Das Testimage #1 repräsentiert einen vollverschlüsselten Datenträger, welcher mit TrueCrypt verschlüsselt wurde. Dazu wurde zunächst das Betriebssystem Windows 8 Professional installiert, welches im Rahmen der Installation automatisiert eine Systempartition mit Bootmanager und Notfalltools erstellt hat. Im Anschluss an die Installation erfolgte die Datenträgervollverschlüsselung mittels TrueCrypt. Dabei wurden neben den Partitionen auch nicht-allozierte Bereiche verschlüsselt. Lediglich der MBR liegt weiterhin unverschlüsselt vor, sodass das Datenträgerlayout ausgelesen werden kann. Nachfolgende Abbildung zeigt das durch die Implementierung interpretierte Datenträgerlayout sowie die Testergebnisse der Clusteranalyse.

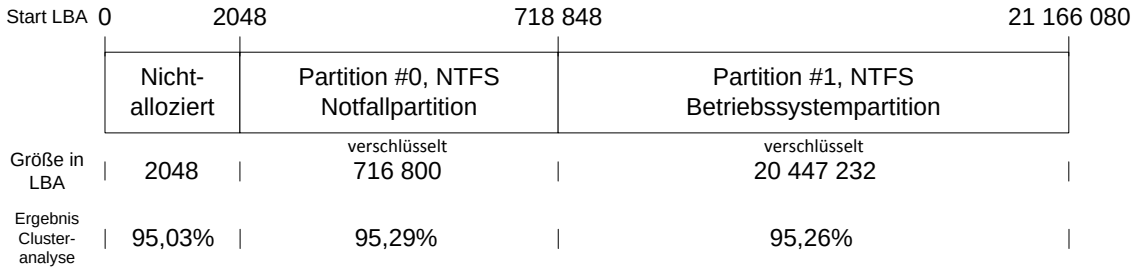

Abbildung 5.2: Datenträgerlayout Testimage #1

Das Layout des Datenträgers wurde richtig interpretiert, die Analyse auf Datenträgervollverschlüsselung ergab einen Anteil verschlüsselter Cluster von 95,27%. Auf Partitionsebene erfolgten die Einstufungen als verschlüsselte Partitionen bei Anteilen verschlüsselter Cluster von 95,29% sowie 95,26%. Der nicht-allozierte Bereich enthielt einen verschlüsselten Clusteranteil von 95,03%. Darüber hinaus wurde im MBR der OEM-String `TrueCrypt boot loader` identifiziert. Das Datenträgerimage wurde folglich als vollverschlüsselt eingestuft.

Zusammenfassend erfolgte eine korrekte und vollständige Analyse des Testimages #1 bei einer Dauer von ca. 6 Minuten.

Tabelle 5.2: Zusammenfassung der Testergebnisse des Datenträgerimages #1

	Partitionen		nicht-allozierte Bereiche
	#0	#1	#0
Layoutbestimmung	OK	OK	OK
Ergebnis Clusteranalyse	95,29%	95,26%	95,03%

5.3.2.2 Testimage #2

Das Testimage #2 repräsentiert einen teilverschlüsselten Datenträger, auf welchem zunächst Windows 8 Professional installiert wurde. Im Rahmen der Installation entstand eine Systempartition, welche den Bootmanager sowie Notfalltools enthält. Im Anschluss wurde die Betriebssystempartition mittels Bitlocker, welche in Windows 8 Professional integriert ist, verschlüsselt. Auf einer weiteren NTFS-Partition, welche im Anschluss an die Windows-Installation über die Datenträgerverwaltung des Betriebssystems manuell erstellt wurde, wurden verschlüsselte Anwendungsdateien abgespeichert.

Die Überprüfung der Datenträgervollverschlüsselung des Testimages #2 lieferte einen Anteil verschlüsselter Cluster von 64,23%. Abbildung 5.3 zeigt das Datenträgerlayout, wie es mittels des Tools bestimmt wurde, sowie die Ergebnisse der Partitionsverschlüsselung.

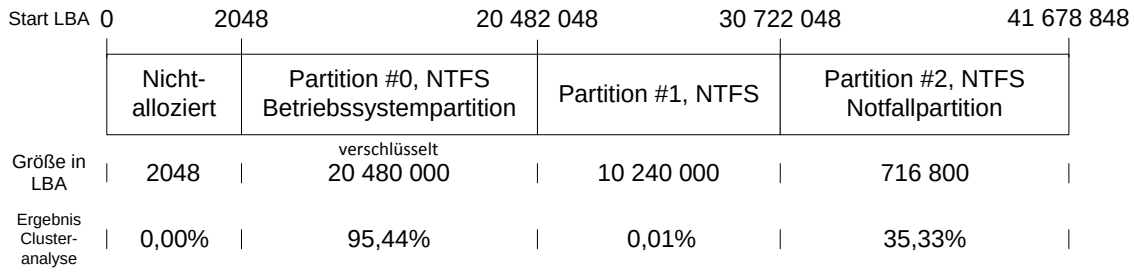

Abbildung 5.3: Datenträgerlayout Testimage #2

Das Datenträgerlayout wurde richtig interpretiert. Dabei entsprechen die Partitionen #0 und #1 den manuell erstellten NTFS-Partitionen, Partition #2 einer bei Installation des Betriebssystem Windows 8 automatisiert erstellten Systempartition. Partition #0 wurde im Anschluss an die Installation mittels Bitlocker verschlüsselt.

Dabei bestanden 95,44% aller getesteten Cluster die Tests der mit Bitlocker verschlüsselten Partition #0. Darüber hinaus wurde die Bitlocker-Signatur `-FVE-FS-` im VBR der Partition identifiziert. 0,01% aller Cluster der unverschlüsselten Partition #1 und ein Anteil von 35,33% aller Cluster der Partition #2 bestanden die Tests. Aufgrund der Testergebnisse wurde Partition #0 nicht weiter betrachtet und als verschlüsselt eingestuft, auf die Partitionen #1 und #2 eine Analyse anhand der MFT und auf den nicht-allozierten Datenträgerbereich korrekterweise die Analyse zur Wiederherstellung gelöschter TrueCrypt-Container angewendet.

Die MFT-Analyse der Partition #1 identifizierte alle beinhalteten verschlüsselten Dateien. Dabei wurden von den Dateiformaten DOCX und ZIP jeweils 4 Dateien mittels Signaturerkennung und jeweils 6 Dateien mittels statistischer Tests identifiziert. Die Identifizierung von 6 PDF-Dokumenten erfolgte durch die Signaturerkennung, 4 Dateien durch die statistischen Tests. Alle 10 TrueCrypt-Container wurden durch die statistischen Tests erkannt und extrahiert. Somit wurden alle verschlüsselten Dateien identifiziert.

Eine Analyse der NTFS-Partition #2 ergab keine weiteren Auffälligkeiten. Der hohe Anteil positiver Cluster der Clusteranalyse resultierte aus komprimierten Daten wie z. B. Bestandteile der Datei Winre.wim. Diese mit 28216 positiven von insgesamt 49026 Clustern große Datei dient zur Wiederherstellung der Bootumgebung. Da der

Anteil positiver Cluster 50% überschreitet, wurde diese Datei extrahiert.

Zusammenfassend erfolgte eine korrekte und vollständige Analyse des Testimages #2 bei einer Dauer von ca. 19 Minuten.

Tabelle 5.3: Zusammenfassung der Testergebnisse des Datenträgerimages #2

		Partitionen			nicht-allozierte Bereiche
		#0 (NTFS)	#1 (NTFS)	#2 (NTFS)	#0
Layoutbestimmung		OK	OK	OK	OK
Ergebnis Clusteranalyse		95,44%	0,01%	35,33%	0,00%
gefundene Dateien	DOCX	–	10/10	0/0	0/0
	PDF	–	10/10	0/0	0/0
	ZIP	–	10/10	0/0	0/0
	TC	–	10/10	0/0	0/0
# Fale negatives		–	0	2	0

5.3.2.3 Testimage #3

Das Testimage #3 beinhaltet erstmals einen durch das Tool nicht-interpretierbaren Partitionstypen, sodass hier die Datenanalyse ohne Metadaten des Dateisystems zur Wiederherstellung verschlüsselter Nutzdaten erfolgt.

Zunächst wurde das Betriebssystem Windows 8 Professional installiert, welches analog zu vorherigen Images, eine Systempartition erstellt. Sowohl auf der System- als auch auf der Betriebssystempartition wurden verschlüsselte Anwendungsdateien abgespeichert. Im Anschluss wurde eine weitere Partition vom Typ FAT manuell mittels der Datenträgerverwaltung des Betriebssystems erstellt. Während der Fragmentierungsgrad bei der Abspeicherung von Dateien auf NTFS-Partitionen keine Rolle spielt, können TrueCrypt-Container auf anderen Partition nur dann erfolgreich wiederhergestellt werden, wenn sich diese im defragmentieren Zustand befinden. Aus diesem Grund wurde sichergestellt, dass die Abspeicherung von TrueCrypt-Containern auf der FAT-Partition im defragmentiertem Zustand erfolgte. Abbildung 5.4 zeigt das interpretierte Datenträgerlayout.

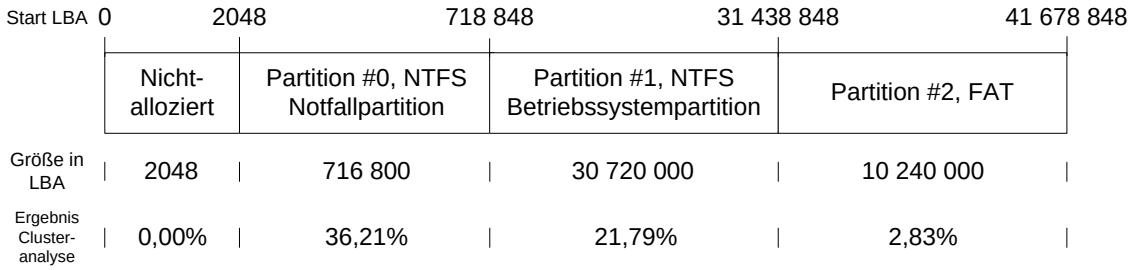

Abbildung 5.4: Datenträgerlayout Testimage #3

Das Datenträgerlayout wurde richtig interpretiert. Dabei entspricht die Partition #0 der Systempartition, welche wie im vorherigen Fall automatisiert durch das Betriebssystem Windows 8 zur Systemwiederherstellung erstellt wurde. Partition #1 beinhaltet das Betriebssystem, Partition #2 wurde im Anschluss an die Betriebssysteminstallation manuell als FAT-Partition mit einer Clustergröße von 4096 Bytes erstellt.

Die Ergebnisse der Clusteranalyse entsprechen weitestgehend den Erwartungen unverschlüsselter Datenträger sowie Partitionen. Dabei ergab eine Überprüfung der Clusterstichprobe des gesamten Images einen Anteil positiver Cluster von 17,38%. Auf Partitionsebene ergab die Analyse der Partition #0 einen Anteil verschlüsselter Cluster von 36,21%, die Überprüfung der Betriebssystempartition, Partition #1, einen Anteil verschlüsselter Cluster von 21,79%. Die FAT-Partition kam schließlich auf einen Anteil positiver Cluster von 2,83%. Aufgrund der Testergebnisse wurden die Partitionen #0 und #1 der MFT-Analyse, Partition #2 der Wiederherstellung von TrueCrypt-Containern unterzogen.

Die MFT-Analysen der Partitionen #0 und #1 lieferten alle verschlüsselten Dateien. Dabei wurden auf Partition #0 1/5 der DOCX-Dateien sowie 3/5 der PDF-Dokumente ausschließlich anhand der Signatur, alle restlichen Dateien sowie 5/5 der ZIP-Dateien und 5/5 der TrueCrypt-Container mittels statistischer Tests erkannt. Anhand der Analyse der Partition #1 wurden 5/5 der ZIP-Archive, 4/5 der DOCX-Dateien, 5/5 der PDF-Dokumente sowie 5/5 der TrueCrypt-Container mittels statistischer Tests, 1/4 der DOCX-Dateien mittels der Signaturüberprüfung identifiziert. Darüber hinaus wurden insgesamt 116 Dateien fälscherlicherweise als verschlüsselt eingestuft, davon 30 anhand des Signaturenscans (ausschließlich DOCX-Signatur) und 86 anhand statistischer Tests (komprimierte Daten). Die Untersuchung umfasste insgesamt 80.816 MFT-Einträge. Die Analyse der FAT-Partition identifizierte beide zurvor abgespeicherte TrueCrypt-Container, welche erfolgreich extrahiert wurden.

Die Datenträgeranalyse dauerte 1 Stunden und 25 Minuten.

Tabelle 5.4: Zusammenfassung der Testergebnisse des Datenträgerimages #3

		Partitionen			nicht-allozierte Bereiche
		#0 (NTFS)	#1 (NTFS)	#2 (FAT)	#0
Layoutbestimmung		OK	OK	OK	OK
Ergebnis Clusteranalyse		36,21%	21,79%	2,83%	0,00%
gefundene Dateien	DOCX	5/5	5/5	0/0	0/0
	PDF	5/5	5/5	0/0	0/0
	ZIP	5/5	5/5	0/0	0/0
	TC	5/5	5/5	2/2	0/0
# False negatives		2	114	0	0

5.3.2.4 Testimage #4

Testimage #4 repräsentiert einen Datenträger, bei welchem der gesamte Speicherbereich als nicht-alloziert interpretiert wird. Hier findet initial ein Test auf Datenträgervollverschlüsselung statt, abhängig vom Ergebnis wird der Datenträger als vollverschlüsselt eingestuft oder mit der Datenanalyse ohne Metainformationen des Dateisystems fortgefahren.

Das Image besteht aus einer Windows 7 Professional-Installation, für welche lediglich eine Partition erforderlich ist. So entstand im Vergleich zur Installation von Windows 8 keine zusätzliche Systempartition. Nach der Installation erfolgte das Kopieren mehrerer TrueCrypt-Container auf den Datenträger, welche es gilt, im Rahmen der Datenanalyse zu identifizieren und extrahieren. Insgesamt wurden 4/5 der TrueCrypt-Container im defragmentierten Zustand, 1/5 der TrueCrypt-Container im fragmentierten Zustand abgespeichert. Um das Image letztendlich als nicht-allozierten Datenträger zu interpretieren, wurden die in der Partitionstabelle hinterlegten Partitionen ausgenullt.

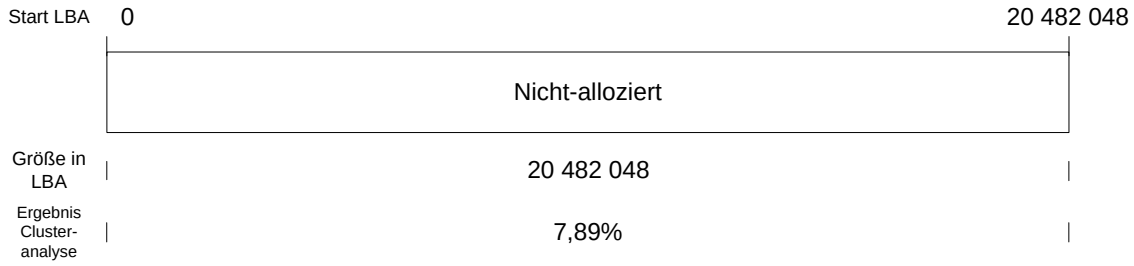

Abbildung 5.5: Datenträgerlayout Testimage #4

Der Datenträger wurde als nicht-alloziert eingestuft, da die Clusteranalyse auf Datenträgerebene lediglich einen verschlüsselter Clusteranteil von 7,89% ergab. Somit erfolgte eine Datenanalyse ohne Metadaten des Dateisystems zur Wiederherstellung verschlüsselter Nutzdaten.

Aus der Datenanalyse resultierten viele, insbesondere kurze false negatives. Insgesamt wurden 687 verdächtige Datenfragmente extrahiert, welche größtenteils eine Größe zwischen 40 - 300 KBytes besaßen. Unter den gefundenen Fragmenten befanden sich die 5 zuvor auf dem Image abgespeicherten TrueCrypt-Container, welche aufgrund ihrer Dateigröße aus den gefundenen Fragmenten herausstechen. So stellten die TrueCrypt-Container die mit Abstand größten extrahierten Fragment dar mit einer Größe von 20.000 - 240.000 KBytes. Einer der TrueCrypt-Container ließ sich aufgrund seines Fragmentierungsgrades nicht an einem Stück, sondern in 3 Teilstücken extrahieren. Eine Zusammenführung der Fragmente lässt sich im Allgemeinen nicht druchführen, da kein Rückschluss auf die Reihenfolge der Zusammensetzung existiert.

Zusammenfassend wurden durch das Testimage #4 die Erkenntnisse der vorherigen Kapitel bestätigt. TrueCrypt-Container lassen sich demnach nicht eindeutig klassifizieren, da keine Unterscheidung zwischen verschlüsselten und zufälligen Datenfragmenten möglich ist. Außerdem entstehen kurze false negatives durch komprimierte Datenformate. TrueCrypt-Container stechen jedoch möglicherweise aufgrund ihrer Dateigröße aus den extrahierten Fragmenten heraus. Die Validierung eines TrueCrypt-Containers kann jedoch nur mit dem dazugehörigen Schlüssel erfolgen. Die Analyse dauerte 31 Minuten.

Tabelle 5.5: Zusammenfassung der Testergebnisse des Datenträgerimages #4

		nicht-allozierte Bereiche
		#0
Layoutbestimmung		OK
Ergebnis Clusteranalyse		7,89%
gefundene Dateien	TC	5/5
# False negatives		688

5.3.2.5 Testimage #5

Das letzte Testimage stellt das Abbild einer 1-GB-SD-Speicherkarte dar. Die Speicherkarte enthält 128 Dateien des Typs JPG und 2 TrueCrypt-Container. Die TrueCrypt-Container gilt es, im Rahmen der Untersuchung zu identifizieren und extrahieren. Da das Dateisystem FAT von dem Tool nicht interpretiert werden kann, wird hier die Datenanalyse zur Identifikation und Wiederherstellung verschlüsselter Nutzdaten angewandt. Entsprechend der Erkenntnisse aus Abschnitt 3.3.1.6, S. 36, welche die Anwendung statitischer Tests auf das Dateiformat JPG beschreibt, werden wenige false negatives durch das Dateiformat JPG erwartet.

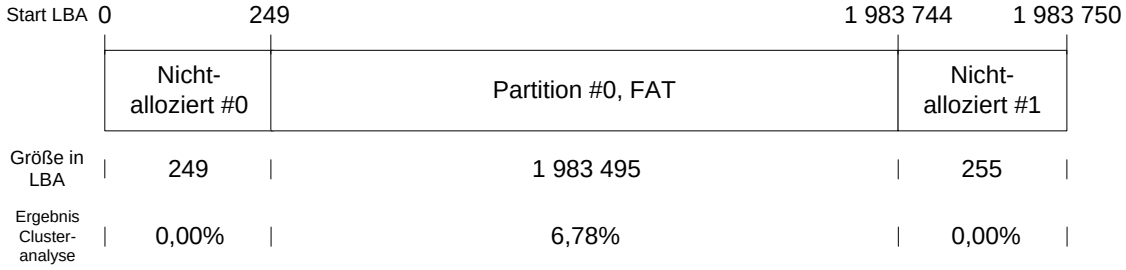

Abbildung 5.6: Datenträgerlayout Testimage #5

Abbildung 5.6 zeigt das korrekt interpretierte Datenträgerlayout durch das Tool. Die Clusteranalysen auf die einzelnen Bereiche entsprechen den Erwartungen: das Image enthält 2 TrueCrypt-Container, welche knapp 10% des gesamten Speicherbereichs einnehmen und sich entsprechend auf die Clusteranalyse auswirken.

Die Datenanalyse verlief erfolgreich. So wurden die TrueCrypt-Container aus dem Image extrahiert, darüber hinaus entstanden keine false positives durch das Dateiformat JPG. Die Analyse des Images erfolgte in 7 Minuten.

Dieses Datenträgerimage wurde ebenfalls einem Entropietest unterzogen. Die Cluster wurden dabei ebenfalls unabhängig voneinander untersucht und die Entropien notiert. Die durchschnittliche Entropie aller Cluster betrug dabei 7,996 Bits/Byte. Mit dem Entropietest lässt sich somit nicht zwischen dem Format JPG und verschlüsselten Daten unterscheiden.

Tabelle 5.6: Zusammenfassung der Testergebnisse des Datenträgerimages #5

		Partitionen	nicht-allozierte Bereiche	
		#0	#0	#1
Layoutbestimmung		OK	OK	OK
Ergebnis Clusteranalyse		0,00%	8,78%	0,00%
gefundene Dateien	TC	–	2/2	–
# False negatives		0	0	0

5.4 Zusammenfassung

Dieses Kapitel beschreibt die Implementierung der zuvor aufgeführten statistischen Tests sowie die Anwendung auf synthetisch generierte Datenträgerimages zur Überprüfung der Funktionalität. Die Umsetzung der Implementierung erfolgte dabei anhand des Entwicklungsframeworks QT, mit welchem der Sourcecode für unterschiedliche Zielplattformen in ausführbaren Code übersetzt werden kann. Die Implementierung umfasst dabei Funktionalitäten zur Durchführung einer Datenträgeranalyse auf unterschiedlichen Abstraktionsebenen eines Datenträgers. Diese Funktionalitäten identifizieren verschlüsselte Daten auf der Datenträgerebene, Partitionsebene, Dateisystem- sowie Anwendungsebene. Auf der Dateisystemebene wurden dabei exemplarisch Methoden zur Analyse von NTFS-Dateisystemmetadaten realisiert. Dadurch ist es möglich, unabhängig vom Fragmentierungsgrad des Datenträgers eine Zuordnung von Clustern und Dateien zu erhalten. Der Erkennungserfolg hängt somit von den statistschen Tests sowie den zuvor identifizierten Signaturen ab.

Die generierten Datenträgerimages zur Evaluierung der Implementierung decken alle Anwendungszenarien ab. Das vollverschlüsselte TrueCrypt-Image wurde dabei gemäß den zuvor erlangten Erkenntnissen aufgrund des sehr hohen positiven Clusteranteils als vermutlich verschlüsselt eingestuft. Im Falle eines mit Zufallsdaten überschriebenen Datenträgers ließe sich jedoch anhand der statistischen Tests keine ein-

deutige Aussage treffen. Aus diesem Grund erfolgt zusätzlich eine Überprüfung des MBR auf bekannte OEM-Strings, außerdem können im MBR hinterlegte Partitionen einen Kontext liefern.

Verschlüsselte Anwendungsdaten, welche auf unverschlüsselten NTFS-Partitionen vorhanden waren, wurden alle extrahiert. Der Erkennungserfolg durch die Anwendung statistischer Tests hing dabei, wie erwartet, von der Dateigröße ab. So wurden besonders kleine Dateien ausschließlich durch die Signaturüberprüfung erkannt. Auf der Anwendungsebene entstand ein geringer Anteil an false positives. Die fälscherlicheweise als verschlüsselt eingestuften Dateiformate enthielten komprimierte Daten, welche ab einer bestimmten Menge zur Einstufung als vermutlich verschlüsselt führen können.

Der zuvor entwickelte File Carving-Ansatz eignete sich zur Erkennung verschlüsselter Daten auf nicht-allozierten Bereichen sowie unbekannten Dateisystemen. So wurden alle TrueCrypt-Container aus diesen Bereichen extrahiert, da diese aus rein verschlüsselten Nutzdaten bestehen und das eingesetzte *sliding window* eine sehr hohe und konstante Auslastung besitzt. Trotzdem hat der praktische Einsatz gezeigt, dass hauptsächlich durch komprimierte Datenfragmente false positives entstehen. Diese Fragmente besitzen jedoch sehr häufig eine kleine Dateigröße, TrueCrypt-Container ließen sich anhand der Dateigröße vermuten. Eine zuverlässige Aussage ist hier jedoch nicht möglich. Ein auf einem Image vorhandener TrueCrypt-Container ließe sich nicht erfolgreich extrahieren, da dieser fragmentiert abgespeichert war. Ein Rückschluss auf die Reihenfolge der Fragmente zur Zusammenführung ist nicht möglich.

Zusammenfassend wurde gezeigt, dass sich die statistischen Tests zur Erkennung verschlüsselter Daten auf Datenträgern eignen. Dabei kann auf einen signaturbasierten Ansatz zur Erkennung kleiner verschlüsselter Daten nicht verzichtet werden, der Erkennungserfolg mittels File Carving hängt dabei zusätzlich vom Fragmentierungsgrad des Datenträgers ab. Somit empfiehlt sich, die durch das Dateisystem bereitgestellten Metainformationen zu nutzen, welches die Implementierung von Methoden zur Interpretation weiterer Dateisysteme erfordert.

Die Ausführungsdauer der Analysen hängt, wie die einzelnen Datenträgerimages zeigten, von den verwendeten Analysemethoden ab. Die Analyse des annähernd vollbelegten 15GB-Datenträgerimage #3 nahm dabei mit 1 Stunde 25 Minuten am meisten Zeit in Anspruch. Die Evaluierung zeigte somit, dass eine Anwendung auf Live-Systeme unpraktikabel ist. Je nach Rechenleistung und Datenträgergröße lässt sich eine Vor-Ort-Analyse mit dem implementierten Tool bisher praktisch nicht durchführen.

6 Zusammenfassung und Ausblick

Diese Arbeit beschäftigte sich mit der Entwicklung eines Ansatzes zur Erkennung verschlüsselter Daten auf allen Abstraktionsebenen eines Datenträgern. Die Erkennung verschlüsselter Daten kann auf allen Ebenen signaturbasiert erfolgen, wobei Verschlüsselung anhand von spezifischen Bytemustern identifiziert wird. Diese Signaturen müssen jedoch nicht zwangsläufig vorhanden sein oder lassen sich oftmals problemlos manipulieren. Darüber hinaus existiert der weit verbreitete Entropietest, welcher auf eine statistische Eigenschaft verschlüsselter Daten, die Gleichverteilung, überprüft. Komprimierte sowie zufällige Daten besitzen ebenfalls diese Eigenschaft, sodass mittels des Entropietests eine korrekte Klassifizierung dieser Daten fehlschlägt. Im Vergleich zu gegenwärtigen Erkennungsmaßnahmen ist es mit dem in dieser Arbeit entwickeltem Ansatz möglich, zwischen verschlüsselten und komprimierten Daten zu unterscheiden.

Die Messung der Zufälligkeit der Daten, die Entropie, betrachtet Auftrittswahrscheinlichkeiten, mit welcher Bytes in einer Bytefolge auftreten. Die Entropie ist dabei maximal, wenn alle beobachteten Zeichen mit gleicher Wahrscheinlichkeit erscheinen. Neben verschlüsselten Daten besitzen komprimierte Daten sowie Zufallsdaten ebenfalls die Eigenschaft der Gleichverteilung, sodass mittels des Entropietests keine Unterscheidung möglich ist.

Da bei der Verschlüsselung von Daten durch moderne Kryptosysteme eine Bytefolge entsteht, welche Zufallsdaten gleicht, wurde in dieser Arbeit auf Ansätze zur Überprüfung der Güte von Zufallszahlengeneratoren zurückgegriffen. Dazu existieren etablierte Tests, welche neben der Gleichverteilung die stochastische Unabhängigkeit einer Folge prüfen. 5 dieser Tests wurden in dieser Arbeit auf die Erkennung von verschlüsselten bzw. zufälligen Daten übertragen. Da Zufallsdaten jedoch die gleichen statistischen Eigenschaften besitzten, ist keine Unterscheidung zwischen diesen und verschlüsselten Daten möglich.

Die aufgeführten Tests überprüfen statistische Eigenschaften von Zufallszahlen, welche sich anhand von Zufallsexperimenten herleiten lassen. Daraus ergeben sich Permutationen von Bytefolgen, welche es gilt, im Datenstrom zu beobachten. Der Chi-Quadrat-Test ist dabei das zentrale Element zur Berechnung der Abweichung beobachteter und hypothetischer Häufigkeiten. Anhand der Teststatistik wird die

Überschreitungswahrscheinlichkeit p berechnet, welche die Zufälligkeit einer Folge repräsentiert. Wird das Signifikanzniveau, der *Type-I-Error* $\alpha = 0,01$, bei einem der Tests überschritten, gelten die untersuchten Daten aus Sicht dieses Tests als zufällig und sind somit möglicherweise verschlüsselt. Dies bedeutet zugleich, dass 1% der Daten durch diesen Test als nicht-zufällig eingestuft werden, obwohl diese verschlüsselt sind. Mit diesem Ansatz gelten Daten als möglicherweise verschlüsselt, wenn alle der 5 aufgeführten statistischen Tests bestanden werden.

Der hier entwickelte Ansatz basiert auf der Messung absoluter Häufigkeiten. Somit lassen sich Abweichungen feststellen, welche sich im Vergleich zum Entropietest auf den gesamten Zahlenbereich beziehen. Durch die Überprüfung erweiterter Eigenschaften verschlüsselter Daten kann mit den verwendeten Tests zwischen komprimierten und verschlüsselten Daten unterschieden werden. Eine Klassifizierung auf Clusterebene lässt sich mittels dieser Tests jedoch nicht durchführen, da durch komprimierte Daten weiterhin false negatives erzeugt werden. Die gesenkte false negative-Rate komprimierter Daten eignet sich jedoch zur Klassifizierung aufeinanderfolgender Cluster. Während von 10.000 überprüften verschlüsselten Clustern über 95% den statistischen Eigenschaften genügen, lässt sich bei einem verschlüsselten Clusteranteil von 57% im Falle 10.000 überprüfter ZIP-komprimierter Cluster Verschlüsselung ausschließen.

Die in dieser Arbeit aufgeführten Tests zur Überprüfung statistischer Eigenschaften von Daten wurden im Rahmen dieser Arbeit in ein Tool implementiert. Dieses Tool umfasst die Analyse von Datenträgerimages auf allen Abstraktionsebenen. Die Überprüfung auf Datenträger- bzw. Partitionsebene beginnt zunächst mit einer Analyse des MBR bzw. VBR. Dabei möglicherweise identifizierte Signaturen können bereits auf Datenträgervoll- bzw. Partitionsverschlüsselung hinweisen, welche jedoch aufgrund möglicher Manipulationen durch einen erweiterten Analyseschritt bestätigt werden müssen. Dieser erweiterte Schritt umfasst die Untersuchung statistischer Eigenschaften einer Clusterstichprobe, welche zufällig ausgewählt wurde. Der Datenträger bzw. die Partition werden dabei als verschlüsselt eingestuft, wenn mindestens 95% aller analysierten Cluster die Tests bestehen. Da die untersuchten Daten jedoch auch Zufallsdaten sein könnten, eignet sich die zuvor durchgeführte OEM-Analyse, einen Kontext zur Verschlüsselung herzustellen. Außerdem lassen in diesem Fall vorhandene Partitionseinträge im MBR auf Verschlüsselung vermuten.

Aufgrund unterschiedlicher Konstruktionseigenschaften von Dateiformaten wird eine Erkennung verschlüsselter Daten auf der Anwendungsebene erschwert. TrueCrypt-Container, welche aus rein verschlüsselten Nutzdaten bestehen, erreichen den größtmöglichen Anteil positiver Cluster von 95%. Andere Dateiformate, wie die in dieser

Arbeit untersuchten Formate PDF, DOCX und ZIP, besitzen jedoch einen hohen Anteil an Metadaten, welche die Dokumentenstruktur beschreiben. Diese Informationen liegen im Anschluss an die Verschlüsselung weiterhin im Klartext vor und genügen somit nicht den statistischen Eigenschaften verschlüsselter Daten. Dieser meist fixe Anteil an Klartext dominiert vor allem bei kleineren Dateien, was zu false positives führt. Um dies zu verhindern, wurden die statistischen Tests um den signaturbasierten Erkennungsansatz ergänzt. Dazu wurden für ausgewählte Dateiformate Signaturen identifiziert, welche Verschlüsselung vermuten lassen. Die Erweiterung um die Signaturerkennung hat zwar Auswirkungen auf die Performanz der Analyse, jedoch haben Manipulationen der Signatur keine Auswirkungen auf das Ergebnis der statistischen Analyse. Somit können Manipulationen aufgedeckt werden, insofern der Metadatenanteil der Datei nicht überwiegt.

Die Analyse von Dateien in NTFS-Partitionen erfolgt anhand von Metainformationen des Dateisystems. Dafür wurden exemplarisch Methoden zur Interpretation des Dateisystems NTFS implementiert, um unabhängig vom Fragmentierungsgrad sowie Belegungsstatus Dateien analysieren zu können.

Für die Analyse nicht-allozierter Speicherbereiche sowie nicht-interpretierbarer Dateisysteme wurde ein Ansatz entwickelt, welcher Datenträgerbereiche ohne Metadaten des Dateisystems analysiert. Während auf eine Signaturerkennung zur Identifizierung verschlüsselter Anwendungsdaten nicht verzichtet werden kann, bleibt sie bei der Erkennung rein verschlüsselter Nutzdaten, wie im Beispiel TrueCrypt, wirkungslos. TrueCrypt-Container lassen sich mit dem entwickelten *sliding window*-Prinzip erkennen und möglicherweise vollständig wiederherstellen. Der Wiederherstellungserfolg hängt dabei vom Fragmentierungsgrad des Datenträgers ab.

Eine mögliche Weiterentwicklung stellt die Implementierung von Methoden zur Interpretation weiterer Dateisysteme dar. Dadurch ergibt sich eine Zuordnung von Clustern und Dateien, welche eine Datenanalyse unabhängig vom Fragmentierungsgrad ermöglicht. Während bei der Datenanalyse ohne Metainformationen des Dateisystems auf eine Reihenfolge bzw. einen Zusammenhang verschlüsselter Datenfragmente nicht geschlossen werden kann, ist eine vollständige Wiederherstellung solcher Daten bei Nutzung der Metainformationen möglich.

Die Qualität der Testergebnisse lässt sich durch Überprüfung weiterer statistischer Eigenschaften steigern. Weitere Eigenschaften können durch kombinatorische Überlegungen hergeleitet und mittels des Chi-Quadrat-Test überprüft werden. Dadurch lassen sich false negatives vermutlich weiter reduzieren.

Literaturverzeichnis

[Ado06] Adobe Systems Incorporated: PDF Reference. http://www.adobe.com/content/dam/Adobe/en/devnet/acrobat/pdfs/pdf_reference.pdf. November 2006. – zuletzt abgerufen am 15.01.2013

[Ado12] Adobe Systems Incorporated: Adobe PDF. http://www.adobe.com/de/products/acrobat/adobepdf.html. 2012. – zuletzt abgerufen am 20.11.2012

[Bau00] Bauer, F. L.: *Entzifferte Geheimnisse: Methoden und Maximen der Kryptologie*. 3. Auflage. Springer, 2000

[Bau09] Baumeister, J.: Numerische Methoden der Finanzmathematik. http://www.math.uni-frankfurt.de/~numerik/lehre/Vorlesungen/Comp_Fin09/skript.shtml. 2009. – Kapitel 3: Erzeugung von Zufallszahlen, zuletzt abgerufen am 28.01.2013

[Bra12] Bragg, R.: The Encrypting File System. http://technet.microsoft.com/en-us/library/cc700811.aspx. 2012. – zuletzt abgerufen am 22.11.2012

[Car03] Cargal, J. Discrete Mathematics for Neophytes: Number Theory, Probability, Algorithms, and Other Stuff. http://www.cargalmathbooks.com/. 2003

[Car10] Carrier, B.: *File system forensic analysis*. Upper Saddle River, NJ [u.a.]: Addison-Wesley, 2010

[CGM12] Choudary, O.; Groebert, F.; Metz, J. Infiltrate the Vault - Security Analysis and Decryption of Lion Full Disk Encryption. Cryptology ePrint Archive 2012/375. 2012

[DR02] Daemen, J.; Rijmen, V.: *The design of Rijndael: AES - the advanced encryption standard*. Berlin [u.a.], Springer, 2002

[Dwo01] Dworkin, M.: Recommendation for Block Cipher Modes of Operation / National Institute of Standards and Technology. 2001. – Forschungsbericht Special Publication 800-38A 2001 Edition

[Dwo10] Dworkin, M.: Recommendation for Block Cipher Modes of Operation: The XTS-AES Mode for Confidentiality on Storage Devices / National Institute of Standards and Technology. Januar 2010. – Forschungsbericht Special Publication 800-38E

[Fer06] Ferguson, N.: AES-CBC + Elephant diffuser: A disk encryption algorithm for Windows Vista. `http://go.microsoft.com/fwlink/?LinkId=82824`. 2006. – zuletzt abgerufen am 15.01.2013

[Fru08] Fruhwirth, C.: LUKS On-Disk Format Specification. `http://wiki.cryptsetup.googlecode.com/git/LUKS-standard/on-disk-format.pdf`. Dezember 2008. – zuletzt abgerufen am 15.01.2013

[GN96] Gailly, J.-L.; Nelson, M.: *The Data Compression Book*. 2. Ausgabe. M & T Books, New York, 1996

[GWH11] Groebert, F.; Willems, C.; Holz, T.: Automated Identification of Cryptographic Primitives in Binary Programs. **In:** *14th International Symposium on Recent Advances in Intrusion Detection, RAID* (2011)

[KK12] Kumar, N.; Kumar, V.: Bitlocker and Windows Vista. `http://www.nvlabs.in/nvbit_bitlocker_white_paper.pdf`. 2012. – zuletzt abgerufen am 02.11.2012

[Knu81] Knuth, D. E.: *Seminumerical algorithms*. 2. Ausgabe. Reading, Mass. [u.a.], Addison-Wesley, 1981

[MH06] Mousa, A.; Hamad, A.: Evaluation of the RC4 Algorithm for Data Encryption. **In:** *International Journal of Computer Science & Applications* Vol. 3 (Juni 2006), Nr. 2

[Mic12] Microsoft TechNet Library: BitLocker Drive Encryption in Windows 7. `http://technet.microsoft.com/de-de/library/ee449438(v=ws.10).aspx`. 2012. – zuletzt abgerufen am 15.11.2012

[MN98] Matsumoto, M.; Nishimura, T.: Mersenne Twister: A 623-dimensionally equidistributed uniform pseudorandom number generator. **In:** *ACM Transactions on Modeling and Computer Simulation* Vol. 8 (Januar 1998), Nr. 1, S. 3–30

[MOV97] Menezes, A. J.; Oorschot, P. C.; Vanstone, S. A.: *Handbook of Applied Cryptography*. Boca Raton [u.a.]: CRC Pr., 1997

[MR95] Motwani, R.; Raghaven, P.: *Randomized Algorithms*. Cambridge University Press, 1995

[Ngo07] Ngo, T.: OFFICE OPEN XML OVERVIEW / ECMA - TC45. `http://www.ecma-international.org/news/TC45_current_work/OpenXML%20White%20Paper.pdf`. Juli 2007. – zuletzt abgerufen am 15.01.2013

[NIJ11] NIJ Criminal Justice Electronic Crime Technology Center of Excellence: TrueCrypt Version 7.0a - Evaluation Report. Januar 2011. – NCJ 235736

[NM13] Nishimura, T.; Matsumoto, M.: Software module (mt19937-64.c). `http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/VERSIONS/C-LANG/mt19937-64.c`. 2013. – A C-program for MT19937-64 - zuletzt abgerufen am 28.01.2013

[Oms10] International OpenOffice market shares. `http://www.webmasterpro.de/portal/news/2010/02/05/international-openoffice-market-shares.html`. Februar 2010. – zuletzt abgerufen am 20.11.2012

[Per12] Perlman, G.: Software module (chisq.c). `ftp://hazards.cr.usgs.gov/hazmaps/sr3mods/chisq.c`. 2012. – compute approximations to chisquare distribution probabilities - zuletzt abgerufen am 25.11.2012

[PKW12] PKWARE Inc.: .ZIP File Format Specification. `http://www.pkware.com/documents/casestudies/APPNOTE.TXT`. September 2012. – zuletzt abgerufen am 15.01.2013

[Qt 12] Qt Project: Qt Reference Documentation. `http://qt-project.org/`. 2012. – zuletzt abgerufen am 12.12.2012

[Ren07] Rentz, D.: OpenOffice.org's Documentation of the Microsoft Compound Document File Format. `http://www.openoffice.org/sc/compdocfileformat.pdf`. August 2007. – zuletzt abgerufen am 20.11.2012

[Rog04] Rogaway, P.: Efficient Instantiations of Tweakable Blockciphers and Refinements to Modes OCB and PMAC / University of California, Davis. Januar 2004. – Forschungsbericht

[RSA78] Rivest, R.; Shamir, A.; Adleman, L.: A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. **In:** *Communications of the ACM* Vol. 21 (1978), S. 120–126

[RSN10] Rukhin, A.; Soto, J.; Nechvatal, J. e. a.: A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications / National Institute of Standards and Technology. April 2010. – Forschungsbericht Special Publication 800-22

[Sha48] Shannon, C.: A Mathematical Theory of Communication. **In:** *Bell System Technical Journal* Vol. 27 (Oktober 1948), S. 379–423, 623–656

[Sha49] Shannon, C.: Communication Theory of Secrecy Systems. **In:** *Bell System Technical Journal* Vol. 28 (1949), S. 656–715

[Sot99] Soto, J.: Randomness Testing of the Advanced Encryption Standard Candidate Algorithms / National Institute of Standards and Technology. 1999. – Forschungsbericht NISTIR 6390

[SSY12] 16 Systems (Hrsg.): TCHunt - Finding TrueCrypt Volumes Since 2007. `http://16s.us/TCHunt/index.php`. 2012. – zuletzt abgerufen am 22.11.2012

[Tru12] TrueCrypt Foundation: TrueCrypt Online-Documentation. `http://www.truecrypt.org/docs/`. 2012. – zuletzt abgerufen am 15.11.2012

[Uni12] Unified EFI: Unified Extensible Firmware Interface Specification. `http://www.uefi.org/specs/download/UEFI_2_3_1_Errata_C_final`. Juni 2012. – Version 2.3.1, Errata C, zuletzt abgerufen am 28.01.2013

[VANK95] Vattulainen, I.; Ala-Nissila, T.; Kankaala, K.: Physical models as tests of randomness. **In:** *Physical Review E* Vol. 52 (1995), Nr. No. 3: 3205–3214

A Erklärung

Ich versichere hiermit, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die im Literaturverzeichnis angegebenen Quellen benutzt habe. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder noch nicht veröffentlichten Quellen entnommen sind, sind als solche kenntlich gemacht. Die Zeichnungen oder Abbildungen in dieser Arbeit sind von mir selbst erstellt worden oder mit einem entsprechenden Quellennachweis versehen. Diese Arbeit ist in gleicher oder ähnlicher Form noch bei keiner anderen Prüfungsbehörde eingereicht worden.

Darmstadt, den 12. Februar 2013,

Simon Thurner