Die Anwendung des BagIt-Formats im Deutschen Literaturarchiv Marbach

Steffen Fritz

Das vorliegende Papier beschreibt die Verwendung des Archivierungsformats BagIt im Deutschen Literaturarchiv Marbach (DLA). Es liefert eine kurze, allgemeine Einführung in das Format, beschreibt die praktische, objektbezogene Anwendung am DLA und schlägt eine Änderung an der aktuellen Spezifikation vor. Dieser Aufsatz entstand im Rahmen eines DFG-Projektes, das sich mit der Archivierung von Netzliteratur beschäftigt, bezieht sich daher auf Publikationen dieser Literaturgattung.

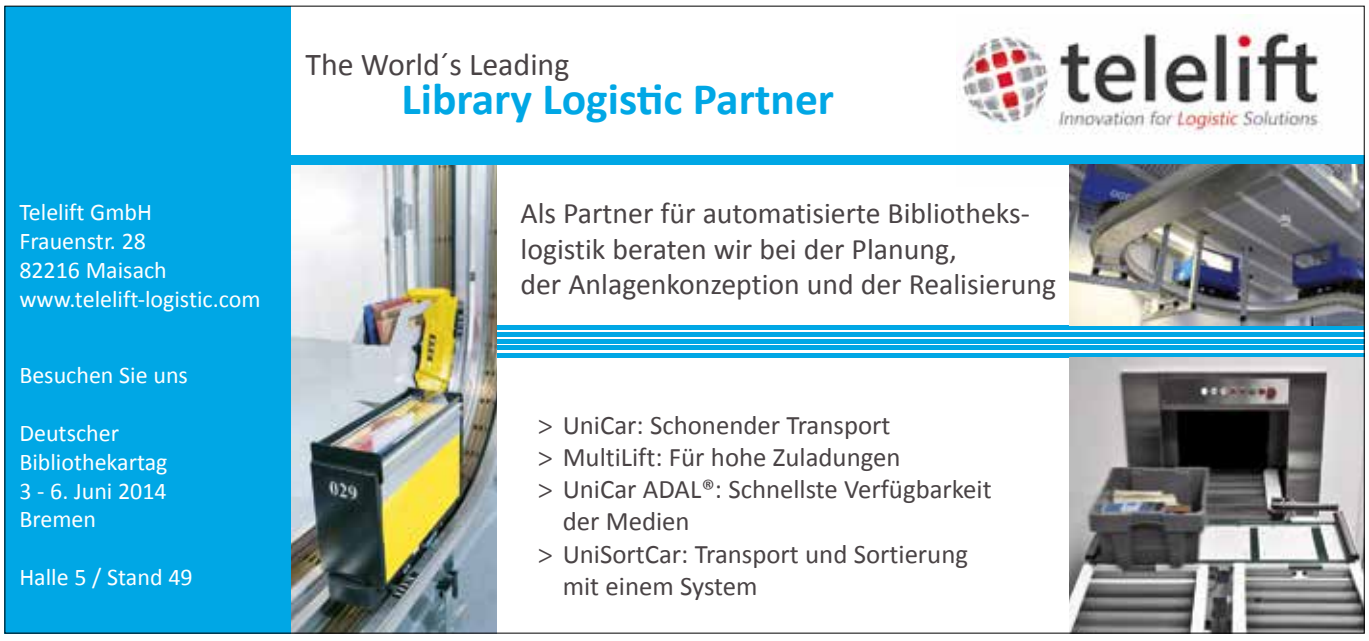
This paper describes the use of the archiving format BagIt at the German Literature Archive in Marbach (DLA). It provides a brief, general introduction to the format, describes a practical, object-oriented application and proposes a change to the current specification. This paper was produced as part of a DFG-project, which is concerned with the preservation of online literature, therefore, it relates to publications of this literary genre.

Motivation und Ausgangslage

Die Bibliothek des DLA Marbach untersucht seit Januar 2013 in dem von der Deutschen Forschungsgemeinschaft geförderten und auf drei Jahre angelegten Projekt *Aufbau eines Quellencorpus für die seit den 1990er Jahren entstehende Literaturgattung Netzliteratur* Möglichkeiten, elektronische Publikationen authentisch zu archivieren und verfügbar zu machen. Netzliteratur zeichnet sich dadurch aus, dass sie genuin in einer Verbindung aus Text und Technik entsteht und dabei äußerst mannigfaltig in der Verwendung der technischen Möglichkeiten ist, die das Internet seit den frühen 1990er Jahren bietet. Netzliteratur ist allerdings vergänglich, da sie nur im Internet besteht. Wird ein Webserver abgeschaltet und wurde das Werk zuvor nicht archiviert, so ist es mit großer Wahrscheinlichkeit verloren.

Netzliteratur entsteht stets in einem historisch-technischen Kontext, der für die Gattung jeweils konstitutiv ist. Die Autoren, die zumeist computertechnisch bewandert sind, schreiben nicht allein Texte; sie entwickeln auf Grundlage der Möglichkeiten des Internets ihre Werke und verweben Wort und Computercode. Daraus folgt, dass sich die Archivierung von Netzliteratur nicht auf den textuellen Inhalt beschränken kann. Sowohl die technische Konstituente als auch die Präsentationsebene[1] müssen gesichert werden, um die Intention der Autoren in ihrer Authentizität zu erhalten. Dies kann nach derzeitigem

1 Die technische Konstituente stellt zumeist auch die Präsentationsebene dar oder beeinflusst sie, wenngleich nicht zwingend. Als Beispiele seien hier Hyperlinks und Cascading Style Sheets (CSS) genannt. Während Hyperlinks Multilinearität ermöglichen, also wesentlich für den Inhalt sind, formatieren CSS die visuelle Darstellung von Inhalten, lassen diese aber unberührt.



Stand im Projekt durch drei Vorgehensweisen erfolgen, die sich aus verwendeten Techniken und dem Zustand des Werks zum Zeitpunkt der Archivierung ergeben. Die einzelnen Verfahren sind nicht Gegenstand dieser Ausführungen, jedoch deren Produkte. Eine Webpublikation kann sich in einem der drei folgenden Zustände befinden:

1. online verfügbar: Online-Status
2. nicht online verfügbar: Offline-Status
3. nicht online verfügbar, aber Quelle ist verfügbar: Offline-Quellen-Status

Ist ein Werk online verfügbar, so besteht grundsätzlich die Möglichkeit, es mit gängigen Werkzeugen wie Heritrix[2] oder wget[3] zu spiegeln und eine warc-Datei[4] zu erstellen. Einschränkungen ergeben sich durch die unterschiedlich eingesetzten Techniken. Werden Inhalte dynamisch generiert oder werden Deep Web-Komponenten wie Datenbanken eingebunden, so können Crawler das gesamte Werk nicht erfassen. Im besten Fall liegt eine Momentaufnahme des Werks vor, die zumindest die Präsentationsebene und einen Teil des textuellen Inhalts widerspiegelt. Weitere Probleme stellen extern eingebundene Skripte und Medien dar, ebenso die Nichteinhaltung von Programmierstandards. Das Paper *CLEAR - a credible method to evaluate website archivability*[5] beschreibt weitere Probleme und Lösungsansätze, hier sei darauf lediglich verwiesen.

Ist aus technischen Gründen eine Spiegelung eines online verfügbaren Werkes nicht möglich, so wird ein Screencast der Webseite angefertigt, der einen Eindruck der literarischen Quelle vermitteln kann. Das Ergebnis dieses sicherlich als Notlösung zu bezeichnenden Vorgehens ist eine Videodatei. Ist ein Werk nicht online verfügbar, können jedoch alle benötigten Dateien beschafft werden, so lässt sich das Werk reaktivieren und in einen Online-Status überführen, wobei dann im Einzelfall entschieden werden muss, ob eine Spiegelung zur Erstellung einer warc-Datei sinnvoll ist. In dieser Variante liegen stets Programmdateien und eventuell Datenbanken vor, die dann den zu archivierenden Datenbestand bilden.

Die digitalen Archivalien, die das DLA innerhalb des Projekts zu betrachten hat, sind also warc-, Video-, Programm- und Datenbankdateien. Der ungenaue Begriff ‘Programmdateien’ ist dem Umstand geschuldet, dass es sich dabei um Skriptdateien, aber auch um Bytecode handeln kann. Ein in diesem Zusammenhang häufig verwendeter Begriff ist der des Quellcodes, im Hinblick auf Bytecode ist er allerdings unzutreffend.

Unabhängig von den zu archivierenden Datentypen muss es möglich sein, eine Metadatendatei beizufügen, die die Quelldaten nach bibliographischen und technischen Gesichtspunkten beschreibt. Es muss also ein Format gefunden werden, in dem Dateien unterschiedlichster Beschaffenheit zusammengefasst und als Archival Information Package behandelt werden können. Diese Anforderungen werden im BagIt-Format erfüllt.

Das BagIt-Format

Diese kurze Einführung in das BagIt File Packaging Format, nachfolgend BagIt genannt, bezieht sich auf den IETF[6]-Entwurf in Version 0.97, Draftversion 10[7]. BagIt definiert eine hierarchische Verzeichnisstruktur, die obligatorische und fakultative Dateien enthält. Ein Ordner, der nach diesem Format aufgebaut ist, wird Bag genannt, die Benennung ist frei wählbar. In diesem Verzeichnis befindet sich das Payload-Verzeichnis, mit den zu archivierenden Dateien sowie mit Metadaten-Dateien, die als Tags bezeichnet werden.

Das Payload-Verzeichnis muss *data* heißen und kann arbiträre Dateien enthalten. Auf derselben Ebene wie das data-Verzeichnis müssen sich die Tag-Dateien *bagit.txt* sowie *manifest-ALG.txt* befinden. bagit.txt benennt die BagIt-Version und gibt das Encoding der Tag-Datei(-en) an. Die Manifest-Datei listet alle Dateien im Payload-Verzeichnis mit zugehöriger Hashsumme auf. Der Platzhalter *ALG* im Dateinamen muss den Algorithmus benennen, mit dem die Hashwerte berechnet werden.

Abbildung 1 zeigt eine minimale Bag mit Payload-Verzeichnis, bagit.txt und einer Manifest-Datei, die SHA-1-Hashes enthält. Da das Payload-Verzeichnis nicht leer sein darf, enthält es die leere Datei .keep.

2 https://webarchive.jira.com/wiki/display/Heritrix/Heritrix [12.03.2014].

3 https://www.gnu.org/software/wget/ [12.03.2014].

4 The WARC File Format (ISO 28500) - Information, Maintenance, Drafts, http://bibnum.bnf.fr/warc/ [12.03.2014].

5 Banos, Vangelis/ Kim, Yunhyong/ Ross, Seamus/ Manolopoulos, Yannis: CLEAR - a credible method to evaluate website archivability, 2013.

6 Internet Engineering Task Force: http://www.ietf.org/ [13.03.2014].

7 Kunze, John/ Littman, Justin/ Madden, Liz/ Vargas, Brian/Boyko, Andy: The BagIt File Packaging Format (V0.97), https://tools.ietf.org/html/draft-kunze-bagit-10 [13.03.2014].

```
BagIt-Version: 0.97
Tag-File-Character-Encoding: UTF-8
```

```
7ff3bbd18f40b307ea820a78acafa274  data/example.txt
d41d8cd98f50b209e43013941c163119  data/needs.gif
aaad1c3b51f4b10fe610091fec18427a  data/doggy.jpg
```

```
Bag-Software-Agent: bagit.py <http://github.com/edsu/bagit>
Bagging-Date: 2014-03-13
Payload-Oxum: 3987.3
Contact-Name: fritz@dla-marbach.de
Source-Organization: Deutsches Literaturarchiv Marbach
```

Listing 1: Beispieldatei: bagit.txt
Listing 2: Beispieldatei: manifest-md5.txt
Listing 3: bag-info.txt

```
<BASIS_VERZEICHNIS> /
    data/
        .keep
    bagit.txt
    manifest-sha1.txt
```

Abbildung 1: Struktur einer minimalen Bag ohne Payload

Diese zwei Tag-Dateien sowie das Dateiverzeichnis stellen obligatorische Elemente einer Bag dar. Zusätzlich kann die Bag noch weitere Tag-Dateien enthalten, die ausführlich im Entwurf erläutert werden. Eine der optionalen Tag-Dateien wird im Folgenden vorgestellt: Die Datei *bag-info.txt* beinhaltet Informationen zur Bag selbst und zu den Daten im Payload-Verzeichnis. Diese Datei kann arbiträre Einträge enthalten, die einer bestimmten Syntax folgen müssen. Bei den Einträgen handelt es sich um einfache Parameter-Wert-Paare, die durch einen Doppelpunkt getrennt sind. Pro Zeile darf es nur einen Eintrag geben. Der IETF Entwurf definiert außerdem 14 reservierte Metadatenelemente, wovon fünf beispielhaft in Listing 3 aufgeführt sind.

Die Listings 1 bis 3 zeigen Beispiele für diese Tag-Dateien. Die meisten Einträge sind selbsterklärend. *Payload-Oxum* in Listing 3 bedarf jedoch einer näheren Betrachtung, da der Begriff ungewöhnlich ist. Oxum leitet sich von octetstream sum ab und gibt in zwei Zahlen die Summe aller 8-bit Bytes und die Anzahl aller Dateien im Payload-Verzeichnis an.
Das Beispiel in Listing 3 zeigt also, dass sich im Ordner data/ drei Dateien befinden, die zusammen eine Größe von 3987 Byte haben. Diese Angabe dient Programmen, die BagIt implementieren, als ein möglicher Prüfwert.

Die Anwendung des BagIt-Formats im DLA

Im Projekt Netzliteratur des DLA wird als Grundlage eine minimale Bag wie oben beschrieben verwendet, diese wird um die Tag-Datei bag-info.txt erweitert, welche damit obligatorisch wird. Die im vorangegangenen Abschnitt vorgestellten fünf Elemente müssen dabei enthalten sein. Das Payload-Verzeichnis muss eine Metadatendatei namens *metadata.xml* umfassen. Außerdem sind mindestens zwei Screenshots beizule-

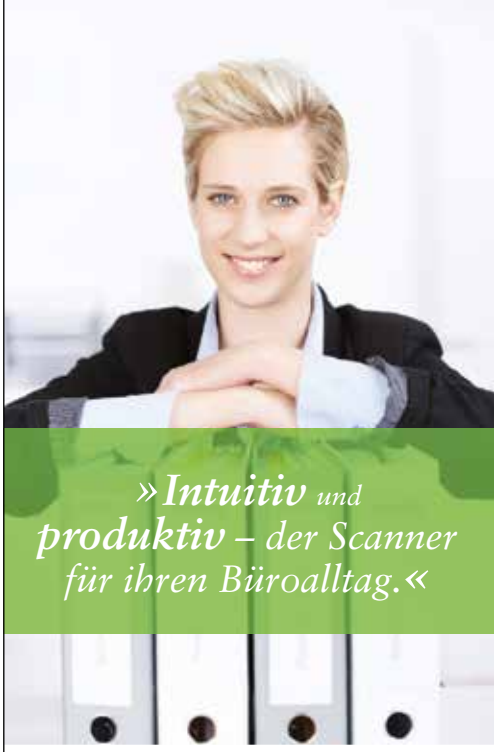
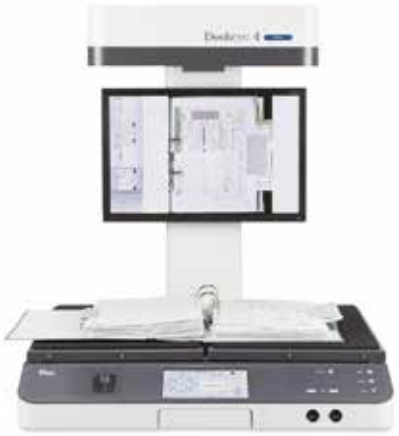


Image Access
/// made in germany

Deutscher Bibliothekartag
3.–6. Juni 2014
Messe & Congress Centrum Bremen
Halle 5, Stand 118



bereit für & BOOKEYE® 4 V2 OFFICE

Bookeye® 4

Der Scanner für Ihren Büroalltag

Lästiges Ein- und Ausheften aus Aktenordnern, dreckige Glasscheiben, Papierstau: Ihr Multifunktionskopierer.

Mit dem ***Bookeye® 4 V2 Office*** gehören diese Szenarien der Vergangenheit an. Der Aufsichtsscanner vereint intuitive Bedienbarkeit mit hoher Produktivität. Benutzerprofile über den großen Touchscreen aktivieren und einfach per Fingerabdruck komplette Arbeitsabläufe steuern. Das Scannen von Ordnern und gebundenen Dokumenten erledigt der Benutzer komfortabel mit dem ***Bookeye® 4 V2 Office.*** Als DMS System zurzeit verfügbar mit: Zertifizierter EMC2 Schnittstelle und Saperion Interface.

Image Access GmbH
Hatzfelder Straße 161–163, 42281 Wuppertal
+49 (0)202 270580, info@imageaccess.de
www.imageaccess.de

gen, einer im .jpeg-, der andere im .tif-Format. Sollte es im Falle eines Werkes im Offline-Quellen Status beispielsweise nicht möglich sein, einen Screenshot anzufertigen, so sind zwei Dummydateien beizufügen. Die Benennung der Screenshots erfolgt mittels Präfix *screenshot_*, einem Infix, der aus einer laufenden Nummer beginnend mit '0' gebildet wird, gefolgt von einem Suffix, der entsprechend dem Format zu wählen ist. Die Benennung der Wurzel der Bag erfolgt nach dem Muster *Name Des Werkes*, Trennung mittels *Unterstrich*, gefolgt von der *Datumsangabe der Bagerstellung* in der Form Jahr Monat Tag, Trennung *Unterstrich* und einer *zweistelligen laufenden Nummer*, beginnend mit '00'. Besteht der Name des Werkes aus mehreren Teilen, so wird die Benennung unter Verwendung von Binnenmajuskeln gebildet. Die laufende Nummer dient der Versionierung möglicher Mehrfacharchivierungen. Die erste Bag des Werkes 'Die Aaleskorte der Ölig', die am 23. Mai 2013 erstellt wurde, würde also die Benennung *DieAaleskorte_20130523_00* erhalten. Abbildung 2 zeigt eine generische minimale DLA-Bag.

```
└── <NameDesWerkes_JJJJMMTT_00> /
    ├── data/
    │   ├── screenshot_0.jpg
    │   ├── screenshot_0.tiff
    │   └── metadata.xml
    ├── bagit.txt
    ├── bag-info.txt
    └── manifest-sha1.txt
```

Abbildung 2: Struktur einer minimalen DLA-Bag ohne Payload

```
└── <NameDesWerks_JJJJMMTT_00> /
    ├── data/ (obl)
    │   ├── webarchiv.warc.gz
    │   ├── screencast.videoformat
    │   ├── Quellcode/Datenbanken/Medien
    │   ├── screenshot_0.jpg (obl)
    │   ├── screenshot_0.tiff (obl)
    │   └── metadata.xml (obl)
    ├── bagit.txt (obl)
    ├── bag-info.txt (obl)
    └── manifest-sha1.txt (obl)
```

Abbildung 3: Struktur einer DLA-Bag

Ergänzt man die Bag aus Abbildung 2 um die Datentypen, die im Payload-Verzeichnis abgelegt werden können, so ergibt sich eine Struktur wie in Abbildung 3, die eine komplette DLA-Bag zeigt. Die obl-Markierungen kennzeichnen Dateien, die in jedem Fall vorhanden sein müssen.

Eine solche Bag wird anschließend serialisiert und zu einer tar-Datei zusammengefasst. Die Benennung dieser Archivdatei ist dabei identisch mit dem Namen der Wurzel, erweitert um den Suffix tar und würde dem entsprechend *NameDesWerkes_JJJJMMTT_00*.tar heißen. Dateien, die im tar- Format vorliegen, lassen sich als abgeschlossene Einheit archivieren, ermöglichen aber die Extraktion einzelner Bestandteile.

Zusammenfassung und Änderungsvorschlag

Das BagIt-Format bietet eine einfache, klare Struktur, ist robust und stellt sicher, dass Bags selbstbeschreibend sind. Eine Bag kann mit Systemmitteln gängiger Betriebssysteme wie Linux, Unix und Microsoft Windows erstellt und auf Datenintegrität überprüft werden, wenngleich es bereits Implementierungen gibt, die den Prozess automatisieren[8] und auch für technisch weniger versierte Personen verwendbar machen.

Die Verlässlichkeit von Hash-Verfahren liegt außerhalb der Kontrolle von BagIt, jedoch nicht die Auswahl selber. Der Entwurf, auf dem diese Ausführungen beruhen, und der in naher Zukunft ein RFC (Request for Comments)-Standard werden wird, verwendet den Begriff 'Algorithmus' im Abschnitt 2.1.3. *Payload Manifest: manifest-<ALG>.txt*. Der Begriff Algorithmus wird in diesem Kontext als problematisch erachtet. Dies begründet sich in den Eigenschaften möglicher Verfahren, namentlich MD5, SHA-1, SHA-2 und SHA-3. Denn die Algorithmen MD5, SHA-1 sowie SHA-3 bezeichnen auch die zu Grunde liegende Hashfunktion. SHA-2 denotiert jedoch eine Gruppe von Funktionen. MD5 und SHA-1 sind beide als gebrochen zu betrachten[9] [10].

8 Am DLA Marbach wird bagit.py verwendet, das sich unter https://github.com/libraryofcongress/bagit-python findet [14.03.2014].

9 Turner, Sean/ Chen, Lei: Updated Security Considerations for the MD5 Message-Digest and the HMAC-MD5 Algorithms, https://tools.ietf.org/rfc/rfc6151.txt [14.03.201].

10 Schneier, Bruce: Cryptanalysis of SHA-1, https://www.schneier.

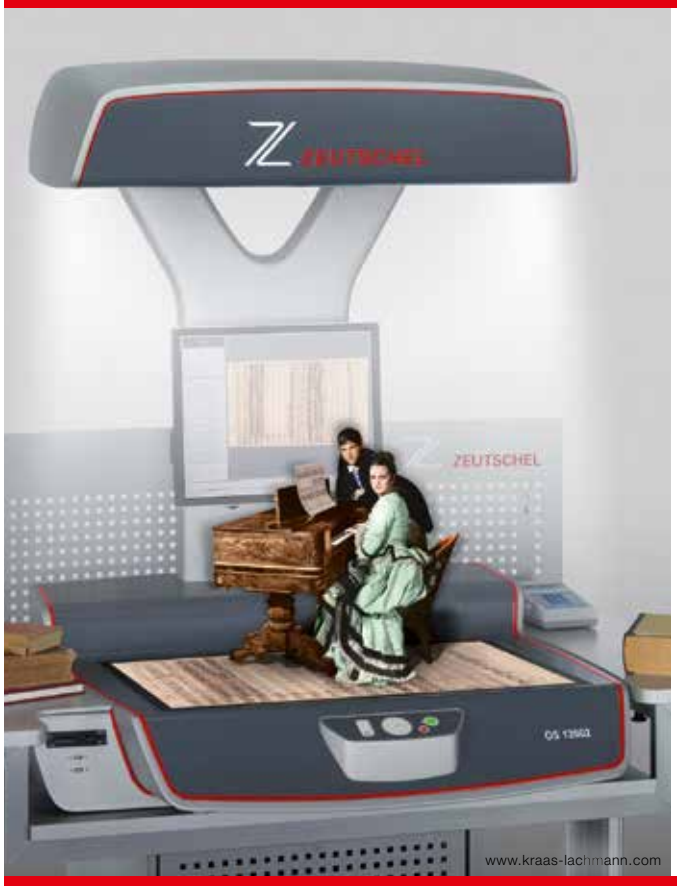
Selbst wenn keine sicherheitsrelevanten Bedenken gegen die Verwendung eines gebrochenen kryptographischen Verfahrens bestehen, so sollten diese nicht verwendet werden. SHA-3 ist kryptographisch sicher, bei einer Softwareimplementierung und abhängig von der jeweiligen Hardware jedoch mäßig performant[11]. SHA-2, das kryptographisch ebenfalls als sicher gilt und performant ist, lässt sich auf Grund der Wortwahl im aktuellen Entwurf von John Kunze et al nicht nutzen. Daher wird vorgeschlagen, den Algorithmus SHA-2 dennoch zu wählen und je nach Funktion die Datei manifest-FUNC.txt zu benennen, wobei als Werte für FUNC *sha224, sha256, sha384, sha512* möglich sind.

Das BagIt File Packaging Format hat sich als sinnvoller Nukleus einer Archivierungsstrategie für digitale Inhalte erwiesen. Im Projekt *Netzliteratur authentisch archivieren und verfügbar machen* wird die weitere Entwicklung des Formats aufmerksam verfolgt. Es besteht die Hoffnung, dass die vorgeschlagene Modifikation übernommen wird, bevor der Entwurf in den RFC-Prozess überführt wird. ❙

com/blog/archives/2005/02/cryptanalysis_o.html [Zugegriffen am 14.03.2014].

11 Bernstein, Daniel J./ Lange, Tanja: eBACS: ECRYPT Benchmarking of Cryptographic Systems. Measurements of SHA-3 finalists, indexed by machine, http://bench.cr.yp.to/results-sha3.html [14.03.2014].

Steffen Fritz
Diplom-Linguist und IT-Experte im DFG-Projekt „Netzliteraur authentisch archivieren und verfügbar machen“ am DLA Marbach
Deutsches Literaturarchiv Marbach
Bibliothek
Schillerhöhe 8-10
71672 Marbach am Neckar
fritz@dla-marbach.de



Die Vergangenheit lebendig halten.

In Bibliotheken und Archiven auf der ganzen Welt werden wertvolle Bücher, Zeitungen, Verträge und Manuskripte mit unseren Hightech-Scannern und Softwarelösungen erfasst, aufbereitet und der Forschung und Wissenschaft digital zur Verfügung gestellt. Seit 1961 halten digitale und analoge Speichersysteme von Zeutschel so die Vergangenheit lebendig. Mit einem OS 12000 Bookcopy lassen sich zum Beispiel sehr empfindliche Kulturgüter und Dokumente wie die Handschriften von Johann Sebastian Bach schonend und in bester Qualität digitalisieren. Schreiben Sie uns, wenn Sie mehr darüber wissen möchten: info@zeutschel.de

Besuchen Sie uns auf dem Bibliothekartag 2014 in Bremen
3. - 6.6.2014, Stand Nr. 81

ZEUTSCHEL

ZEUTSCHEL – die Zukunft der Vergangenheit. www.zeutschel.de