Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Bachelorarbeit

Entwicklung eines Visualisierungswerkzeuges zur Demonstration datenschutzfreundlicher Dokumentspeicherdienste

vorgelegt von

David Kirchhausen Monteiro

geb. am 24. Januar 1994 in Hildesheim

Matrikelnummer 6530927

Studiengang Software-System-Entwicklung

eingereicht am 13. August 2018

Betreuer: Maximilian Blochberger, M. Sc.

Erstgutachter: Prof. Dr.-Ing. Hannes Federrath

Zweitgutachter: Tilmann Stehle, M. Sc.

Aufgabenstellung

Im Zuge dieser Bachelorarbeit soll ein einfacher Dokumentenspeicher entwickelt werden, welcher möglichst viele Nutzerdaten erfasst und speichert. Die erfassten Daten sollen anschaulich grafisch dargestellt werden können. Weiter sollen verschiedene Szenarien entwickelt werden, welche aufzeigen wie eine mögliche Benutzung des Services mit und ohne der Verwendung von datenschutzfreundlichen Methoden zum Anonymisieren von Daten aussieht. Anhand der Szenarien soll eine grafische Auswertung Unterschiede zwischen anonymisierten Daten und nicht anonymisierten Daten visuell sichtbar machen und die Unterschiede somit leicht zugänglich sein.

Eidesstattliche Versicherung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel – insbesondere keine im Quellenverzeichnis nicht benannten Internet-Quellen – benutzt habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht. Ich versichere weiterhin, dass ich die Arbeit vorher nicht in einem anderen Prüfungsverfahren eingereicht habe und die eingereichte schriftliche Fassung der auf dem elektronischen Speichermedium entspricht.

Hamburg, den 13. August 2018

_____

David Kirchhausen Monteiro

Abstract

Um ein mögliches Missbrauchspotential und die generelle Informationsgewinnung aus gesammelten Verkehrs- und Metadaten durch Servicebetreiber deutlich zu machen, wird im Rahmen dieser Arbeit ein einfacher Dokumentenspeicher konzipiert und implementiert. Der Dokumentenspeicher sammelt Verkehrs- und Metadaten und besitzt ein Visualisierungstool um diese gesammelten Daten anschaulich grafisch darzustellen. Weiter werden zwei Szenarien definiert und umgesetzt, welche die Benutzung des Dokumentenspeichers exemplarisch aufzeigen und die Visualisierungsmöglichkeiten anhand der Beispieldaten der Szenarien vorführt. Anhand der verschiedenen Visualisierungen wird der potentielle Informationsverlust für einen Servicebetreiber, der durch die Nutzung verschiedener datenschutzfreundlicher Methoden zur Anonymisierung von Daten entsteht, sichtbar gemacht.

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

Dokumentenspeicherdienste sind nützliche Alltagsgegenstände, welche für private sowie kommerzielle Nutzer meist unverzichtbar sind. Sie bieten nicht nur den Speicherplatz für wichtige Dateien der Nutzer, sondern bieten auch Sicherheit für die Dateien und machen sie global jederzeit verfügbar. Durch die große Datensammlung dieser Dienstleister machen sie sich nicht nur selbst zu lukrativen Zielen von gezielten Angriffen. Dies zeigen mehrere Fälle, in denen z. B. von Yahoo Daten gestohlen wurden und das Ausmaß der gestohlenen Nutzerdaten [Sok17]. Auch die Dienstleister selber können die Daten über die Nutzer sammeln, auswerten und weiter verwenden, z. B. die Dateigröße oder den Autor der Datei aus den Metadaten oder die IP-Adresse oder die HTTP (Hypertext Transfer Protocol) Header Felder [RFC2616] aus den Verkehrsdaten. Vor allem private Nutzer sind meist gar nicht über die Risiken und das Missbrauchspotenzial aufgeklärt, welche die Verwendung solcher Dienstleistungen mit sich bringen können. Methoden zur Verschlüsselung oder das Anonymisieren von Daten sind Benutzern meist nicht bekannt, werden von den Dienstanbietern nicht angeboten oder sind schwer umzusetzen, da es einen meist erheblichen Aufwand für die Benutzer bedeutet und Kompetenzen erfordert, welche diese Benutzer oft nicht besitzen. Zudem ist Benutzern oft nicht klar, inwiefern sie Informationen über sich selbst durch die Nutzung eines Services preisgeben. Um genau die Risiken und Missbrauchspotenziale aufzuzeigen wird im Zuge dieser Arbeit ein Dokumentenspeicherdienst entwickelt, welcher Verkehrs- und Metadaten der Nutzer sammelt und diese in einer visuellen Darstellung zusammenfasst. Zur Implementation des Dokumentenspeichers wird dabei das Microsoft ASP.NET Core Framework verwendet. Das Framework wird benutzt, um die Webbenutzeroberfläche sowie die Web API (Application programming interface) des Dokumentenspeichers zu realisieren. Dazu wird das Data-Driven Documents Framework (d3.js[1]), zur Visualisierung der Daten verwendet. Der Dokumentenspeicher soll vor allem den Unterschied zwischen der Verwendung von Methoden zum Anonymisieren von Daten und ungeschützten Daten visualisieren. Dazu verwaltet der Dokumentenspeicher zwei verschiedene Datensätze, wobei eine Datenmenge ohne und eine Datenmenge mit der Verwendung von Methoden zum Anonymisieren von Daten erzeugt wird. Der entstehende Unterschied der gesammelten Daten durch die verschiedenen Methoden führt dann zu einer Veränderung in der Visualisierung, was den Effekt und Nutzen der Methoden deutlich sichtbar macht.

1. https://d3js.org/

2 Related Work

Die Ergebnisse und Erkenntnisse mehrerer Arbeiten wurden in dieser Arbeit verwendet, um die Aufgabenstellung zu bearbeiten. Dies ergab sich aus der Nähe ihrer Themen zur Aufgabenstellung und dem erarbeiteten Lösungsansatz. Diese werden nun kurz vorgestellt.

In den Ausarbeitungen von Eckersley [Eck10] und Boda u. a. [Bod+12] werden Methoden zum Erzeugen eines Fingerprints (Fingerabdruck) anhand gesammelter Daten eines Benutzers erarbeitet. Dieser Fingerprint soll verwendet werden, um einen Benutzer wiedererkennen zu können. In den Arbeiten [Eck10] und [Bod+12] wird dabei eine ähnliche Methodik zum Erzeugen des Fingerprints vorgestellt. Eckersley [Eck10] und Boda u. a. [Bod+12] verwenden beide HTTP Header und Javascript, zum Ermitteln von Informationen über einen Benutzer.

Eckersley [Eck10] betrachtet dabei die HTTP Felder *User-Agent* und *Accept* sowie Informationen zu Cookies (enabled/disabled). Zusätzlich wird mittels Javascript Funktionalitäten die Bildschirmgröße, Zeitzone, Informationen zu Browserplugins und Systemfonts ermittelt und ein Test auf Supercookies durchgeführt. [Eck10, p. 4-5]

Boda u. a. [Bod+12] betrachtet ebenfalls das HTTP Headerfeld *User-Agent* und sammelt mittels Javascript Informationen zu Bildschirmgröße, Zeitzone und Systemfonts. Zusätzlich werden Informationen zur IP-Adresse, dem Betriebssystem, des Standpunktes des Benutzers sowie eine Benutzer ID und der Zeitstempel wann, der Fingerprint erzeugt wurde, gesammelt. [Bod+12, p. 4-6]

Die so erzeugten Fingerprints werden von Eckersley [Eck10] und Boda u. a. [Bod+12] verwendet um anhand von Beispieldaten[1] diese auf Zuverlässigkeit zur Wiedererkennung von Benutzern zu testen.

Diese Methodik des Erzeugens eines Fingerprints für einen Benutzer wird für diese Arbeit aufgegriffen. Dabei wird die Verwendung von Javascript zum Erfassen von Systeminformationen der Benutzer jedoch vernachlässigt, sodass nur die Header der Benutzer zum Erzeugen eines Fingerprints verwendet werden. Dieser Headerfingerprint wird in dieser Arbeit dazu verwendet, Benutzerdateien zu gruppieren und diese Gruppierungen zu visualisieren.

Dingledine, Mathewson und Syverson [DMS04] stellen in ihrer Ausarbeitung Tor vor, welches ein Netzwerk für anonyme Kommunikation ist. In der vorliegenden Arbeit wird Tor als eine Schutzmethode verwendet, welche es den Benutzern ermöglichen soll Dateien anonym an den entwickelten Dokumentenspeicher zu senden. Diese Schutzmethode wird den Daten, welche ohne Schutzmethode (ungeschützt) hochgeladen wurden gegenübergestellt, indem beide Datenmenge visualisiert werden. Die Unterschiede in den beiden Visualisierungen sollen dann den Effekt der Verwendung des Tor-Netzwerks und -Browser sichtbar und einem Betrachter diese möglichst leicht zugänglich machen.

Graham und Kennedy [GK10] behandeln verschiedene Methoden zur Darstellung von hierarchischen Baumstrukturen. Für die vorliegende Arbeit relevant ist die Ausführung zur Darstellung

1. http://panopticlick.eff.org/

von einfachen Baumstrukturen. Als eine Möglichkeit der Darstellung einer Baumstruktur beschreiben Graham und Kennedy [GK10] die Baumkarte. Da in ihr die Eltern-Kind-Relationen einer Baumstruktur als ineinander geschachtelte Boxen dargestellt werden [GK10, p. 4-5] erschien sie mir als sehr anschaulich und somit als besonders geeignet für die hier vorliegende Arbeit. Eine solche Darstellung ist in Kapitel 3 exemplarisch aufgezeichnet.

3 Dokumentenspeicher

3.1 Verwendungszweck des Dokumentenspeichers

Der Dokumenspeicher dient vor allem dazu, die gesammelten Verkehrs- und Metadaten mit und ohne die Verwendung von datenschutzfreundlichen Methoden zum Anonymisieren von Daten, zu vergleichen und deren Unterschiede grafisch möglichst aussagekräftig darzustellen. Die Unterschiede in den gesammelten Daten sollen zeigen, dass die Verwendung von Methoden zum Anonymisieren von Daten eine Auswirkung darauf hat, ob anhand der gesammelten Verkehrs- und Metadaten, Dateien so einander zugeordnet werden können, dass die entstehenden Gruppen nur aus den Dateien eines Benutzers bestehen. In den Visualisierungen werden deshalb die Dateien anhand verschiedener Eigenschaften ihrer Verkehrs- und Metadaten gruppiert dargestellt. Die dabei erzeugten Gruppen sind ohne die Verwendung von datenschutzfreundlichen Methoden zum Anonymisieren von Daten bei z. B. der Eigenschaft der IP-Adresse, so eindeutig, dass die dargestellten Gruppen den Dateien eines Benutzers entsprechen. Die Verwendung von datenschutzfreundlichen Methoden zum Anonymisieren von Daten kann jedoch die Zuordnung des Benutzers anhand der Verkehrs- und Metadaten verhindern, sodass die erzeugten Gruppen in der Visualisierung nicht mehr den tatsächlichen Relationen von Benutzern und Daten entsprechen.

Da für einen Dokumentenspeicher die Verwendung von Authentifizierungen durch einen Benutzeraccount oder ähnliches diese Zuordnung trivialisiert, wird für diesen Dokumentenspeicher eine solche Authentifizierung nicht verwendet. Für diesen Dokumentenspeicher wird zudem angenommen, dass alle Dateien von den Benutzern so verschlüsselt wurden, dass nur die Benutzer in der Lage sind, sie wieder zu entschlüsseln. Daher werden alle Dateien in einem gemeinsamen Speicher abgelegt.

Um die Effekte einer datenschutzfreundlichen Methode zur Anonymisierung von Daten aufzeigen zu können, müssen die gesammelten Dateien im Dokumentenspeicher unterscheidbar sein in Daten mit und ohne Anonymisierung. Dies wird durch die Verwendung zweier API-Endpunkte erreicht, die beim Hochladen der Dateien durch die Benutzer zu wählen sind. Im Dokumentenspeicher gibt es also eine Datenmenge von Dateien, welche durch die Nutzung von datenschutzfreundlichen Methoden zum Anonymisieren geschützt sind und eine ungeschützte Datenmenge. Damit diese Datenmengen vergleichbar sind, sieht die Verwendung des Dokumentenspeichers vor, dass für den Vergleich von geschützten und ungeschützten Datenmengen die zu untersuchenden Daten einmal mit und einmal ohne die Verwendung von Methoden zum Anonymisieren von Daten hochgeladen werden. Dabei sollte möglichst darauf geachtet werden, dass die so entstehenden Datenmengen sich nur durch die verwendeten Methoden zum Anonymisieren der Daten unterscheiden. Die daraus resultierenden Datenmengen eignen sich dazu, die Visualisierung der ungeschützten Datenmenge zu zeigen und die daraus ableitbaren Informationen visuell sichtbar zu machen. Die Visualisierung der geschützten Menge sollte dann im Kontrast dazu sichtbar machen, dass durch die verwendete Methode zum Anonymisieren von Daten die abgeleiteten Informationen nicht mehr in allen Aspekten denen der ungeschützten Datenmenge entsprechen. Entsprechende Szenarien, welche diese Verwendung exemplarisch darstellen, sind in Kapitel 4 zu sehen. Alternativ ist es jedoch auch möglich Datenmengen

aus geschützten und ungeschützten Daten zu betrachten. Dafür sollten alle Dateien an einen API-Endpunkt gesendet werden, sodass alle Dateien in der selben Datenmenge vorhanden sind. Die Betrachtung von einer solchen gemischten Datenmenge aus geschützten und ungeschützten Daten kann ebenfalls benutzt werden, um den Effekt von verschieden Methoden zum Anonymisieren von Daten aufzuzeigen. Sie kann aber nicht aufzeigen, ob und welche Methoden zum Anonymisieren verwendet wurden.

In dem Visualisierungstool des Dokumentenspeichers kann eine Visualiserung ausgewählt werden und beliebig zwischen der geschützten Datenmenge und der ungeschützten Datenmenge gewechselt werden, sodass die Unterschiede anhand der Visualisierung der beiden Datenmengen leicht erkennbar sind.

Die Verwendung reiner Datenmengen (aus nur ungeschützten und nur geschützten Dateien) eignet sich somit, die entstehenden Unterschiede durch die Verwendung von datenschutzfreundlichen Methoden zum Anonymisieren von Daten und ohne diese zu visualisieren, und somit ausschließlich diese Unterschiede zu visualisieren, da die zugrunde liegenden Dateien sich sonst nicht unterscheiden.

3.2 Implementation des Dokumentenspeichers

3.2.1 Verwendete Technologien

Der Dokumentenspeicher wurde mit Hilfe des ASP.NET Core Framework implementiert. Das Framework ist Microsofts aktuellstes plattformübergreifendes Framework zur Realisierung von Webanwendungen. Das Framework unterstützt alle gängigen Betriebssysteme wie Windows, Mac OS und Linux. Mit dem ASP.NET Core entwickelte Webanwendungen lassen sich in gängige Hostingplattformen, wie z. B. der IIS (Internet Information Services) von Microsoft, integrieren oder können auch in einem eigenen Prozess selbst gehostet werden. Für mehr Informationen zu dem Framework siehe [Lut18]. Das ASP.NET Core Framework sieht dabei eine MVC (Model-view-controll) Architektur der Projekte vor. Als Model wird in diesem Kontext ein Objekt zur Datenrepräsentation verstanden. Ein View ist eine HTML-Seite, die an den Klienten ausgegeben wird. Controller sind die zentralen Elemente mit verschiedenen Funktionen. Sie stellen die Funktionalität von Views auf der Serverseite sicher. Sie sind zur Steuerung verschiedener Routen und die Implementation von API-Endpunkten vorgesehen und verwalten dabei die zugrunde liegenden Datenbanken.

3.2.2 Aufbau des Dokumentenspeichers

Der entwickelte Dokumentenspeicher besteht aus

- einer Model-Klasse,
- drei View-Klassen,
- einer Controller-Klasse, sowie
- einer Datenbankkontext-Klasse.

Die Model-Klasse hält für alle erfassten Verkehrs- und Metadaten Eigenschaften bereit, welche diese repräsentieren.

ID Datenbank Index

Set Menge, welcher die Datei zugeordnet wurde (geschützte oder ungeschützte Menge)

Filename
Dateiname

Filepath
Pfad zur gespeicherten temporären Datei

Size Dateigröße in Byte

IPAddress
IP-Adresse, von der die Datei hochgeladen wurde

Headers
String bestehend aus den Headern der Datei

HeaderFingerprint
Hash erzeugt aus dem Zusammenschluss von ausgewählten Headerfeldern

DateTime
Zeitstempel für das Hochladen der Datei

Country
Land, aus welchem die Datei hochgeladen wurde

RegionName
Region (Bundesland), aus welchem die Datei hochgeladen wurde

City Stadt, aus welcher die Datei hochgeladen wurde

Lat Breitengrad, welcher mit der bekannten IP-Adresse assoziiert wird

Lon Längengrad, welcher mit der bekannten IP-Adresse assoziiert wird

ISP Der Internetanbieter, welcher der IP zugeordnet ist

Mit Hilfe der Model-Klasse werden die gesammelten Daten sowie der Pfad zu der gespeicherten Datei in der Datenbank abgelegt.

Die drei View-Klassen repräsentieren die HTML-Seiten, welche von einem Benutzer über bestimmte Routen aufgerufen werden können:

/FileEntry
Anzeige der Datenbank in tabellarischer Form

/FileEntryCreate
Bietet Möglichkeiten zum Hochladen von Dateien oder zum Erzeugen von Pseudodaten

/FileEntryVisual
Visualisierung der gesammelten Daten

Die *FileEntry* HTML-Seite ist die Startseite der Webanwendung und verweist zu der *FileEntryCreate* HTML-Seite und *FileEntryVisual* HTML-Seite. Zudem zeigt sie die Datenbank in tabellarischer Form getrennt in die beiden möglichen Mengen (geschützt und ungeschützte Dateien). Diese Seite ist hauptsächlich zur Entwicklung gebraucht worden. Die *FileEntryCreat* HTML-Seite enthält mehrere Funktionen, um Dateien hochzuladen. Dabei können die jeweils verschiedenen API-Endpunkte ausgewählt oder Beispieldaten über die Eingabe in verschiedene Textfelder erzeugt werden. Diese Seite ist ebenfalls hauptsächlich zur Entwicklung und zum Testen des Dokumentenspeichers verwendet worden. Die *FileEntryVisual* HTML-Seite bietet verschiedene Visualisierungsmögichkeiten an und ist der Hauptbestandteil des Dokumentenspeichers.

Die Controller-Klasse implementiert verschiedene API-Endpunkte, welche das Hochladen von Dateien und die Abfrage der gesammelten Daten ermöglicht.

/api/GetA
HTTP Get Methode, welche die gesammelten Verkehrs- und Metadaten der ungeschützten Daten ausgiebt

/api/GetB
HTTP Get Methode, welche die gesammelten Verkehrs- und Metadaten der geschützten Datensatz ausgiebt

/api/uploadA
HTTP Post Methode zum Hochladen von ungeschützten Dateien

/api/uploadB
HTTP Post Methode zum Hochladen von geschützten Dateien

api/uploadEmu
HTTP Post Methode zum Erzeugen von Dummydaten

api/getFile
HTTP Get Methode zum Herunterladen einer Datei

Beim Hochladen der Dateien werden in der Controller-Klasse die Verkehrs- und Metadaten der Dateien erfasst, sowie die hochgeladenen Dateien abgespeichert. Während einige Eigenschaften direkt aus den Verkehrs- und Metadaten entnommen werden können, werden andere Eigenschaften durch die Auswertung oder Weiterverwendung der Verkehrs- und Metadaten erzeugt. Der Headerfingerprint wird so aus dem Zusammenschluss aus den Header-Feldern

- *User-Agent* [RFC7231, sec. 5.5.3]
- *Accept* [RFC7231, sec. 5.3.2]
- *Accept-Encoding* [RFC7231, sec. 5.3.4]

erzeugt [GK10, p. 4-5]. Die Kombination dieser Werte wird mit Hilfe des MD5 [RFC1321] Algorithmus gehasht und als Headerfingerprint gespeichert.

Die Datenbankkontext-Klasse dient der Verwaltung und dem Zugriff auf die Datenbank.

Für die Visualisierung ist es wesentlich, zwischen den einzelnen Visualisierungen wechseln zu können. Daher besteht die *FileEntryVisual* HTML-Seite neben einem Bereich für die Darstellung der Visualisierung auch aus einem Navigationsbereich. Die Navigation bietet die Möglichkeit zwischen den ungeschützten und geschützten Datensatz zu wechseln, verschiedene Visualisierungsmöglichkeiten auszuwählen und hat einen Bereich, wo die gesammelten Verkehrs- und Metadaten einer ausgewählten Datei aufgelistet werden. Implementiert sind drei Visualisierungsmöglichkeiten:

- eine Baumkarte (Treemap),
- eine geografische Karte mit der Anzeige von Standpunkten und
- einem Zeitstrahl.

Baumkarte: Eine Baumkarte ist eine Möglichkeit eine Baumstruktur darzustellen, wobei Boxen entsprechend der Baumstruktur ineinander verschachtelt werden [Rib]. Bei der Verwendung der Baumkarte muss eine Eigenschaft der Model-Klasse (ausgenommen *Filepath*) ausgewählt werden, um eine Baumstruktur zu bilden. Diese ausgewählte Eigenschaft wird als Schlüsseleigenschaft für die Baumkarte bezeichnet und ist die Eigenschaft der gesammelten Verkehrs- und Metadaten, nach der gruppiert werden soll. Die Baumstruktur besteht aus einem Wurzelknoten, den davon abgehenden Kindknoren und den von diesen abgehenden Blattknoten, welche sich dadurch auszeichnen, dass von ihnen keine weiteren Knoten abgehen. Der Wurzelknoten wird über die Schlüsseleigenschaft erzeugt. Die restlichen Knoten werden aus den gegebenen Daten erzeugt, sodass alle Dateien mit der gleichen Eigenschaft für die gewählte Schlüsseleigenschaft unter einem Knoten zusammengefasst werden. Die Knoten dieser Eigenschaften werden unter dem Wurzelknoten angeordnet. Die erzeugte Baumstruktur hat somit drei Ebenen. Auf der 1. Ebene den Wurzelknoten mit der gewählten Schlüsseleigenschaft. Auf der 2. Ebene die Knoten, welche die gesammelten Werte für diese Schlüsseleigenschaft darstellen. Auf der 3. Ebene die Dateien, welche entsprechend ihres Wertes im Bezug auf die Schlüsseleigenschaft dem entsprechenden Knoten der 2. Ebene untergeordnet wurden.

Abbildung 3.1 zeigt eine beispielhafte Baumstruktur für sechs Dateien, die über zwei verschiedene IP-Adressen hochgeladen wurden. Der Wurzelknoten zeigt die Schlüsseleigenschaft: *IP-Adresse*. Auf der 2. Ebene werden die gesammelten IP-Adressen angezeigt. Auf der 3. Ebene werden die Dateien den jeweiligen IP-Adressen zugeordet - die Dateien data0, data1 und data2 zu der IP-Adresse 192.0.2.1 und data3, data4 und data5 zu der IP-Adresse 192.0.2.2.

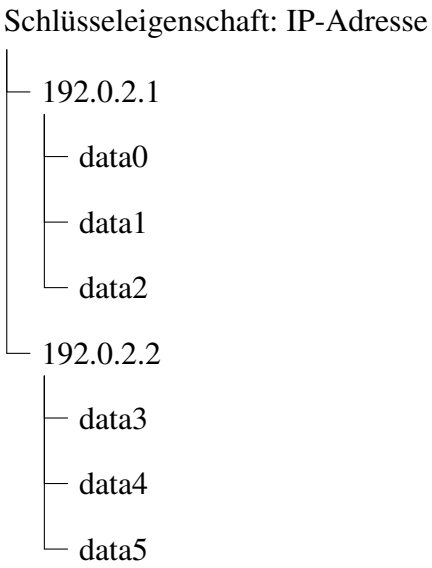

Abbildung 3.1: Beispielhafte Baumstruktur für das erstellen einer Baumkarte

Aus dieser Struktur wird nun die Baumkarte erzeugt. Für jeden Knoten im Baum wird eine Box erzeugt, dabei wird jeder untergeordnete Knoten in die Box des übergeordneten Knotens eingebettet und in Abhängigkeit zu diesem eingefärbt wird [GK10, p. 4-5]. Durch diese farblichen Unterschiede sollen dann die verschiedenen Ebenen und Relationen deutlich werden. Die Größe der jeweiligen Boxen ist dabei abhängig von der Größe der einzelnen Dateien, bzw. der Größe aller Dateien, die unter einer Eigenschaft gruppiert sind. Eine schematische Darstellung des Beispiels ist in Abbildung 3.2 zu sehen.

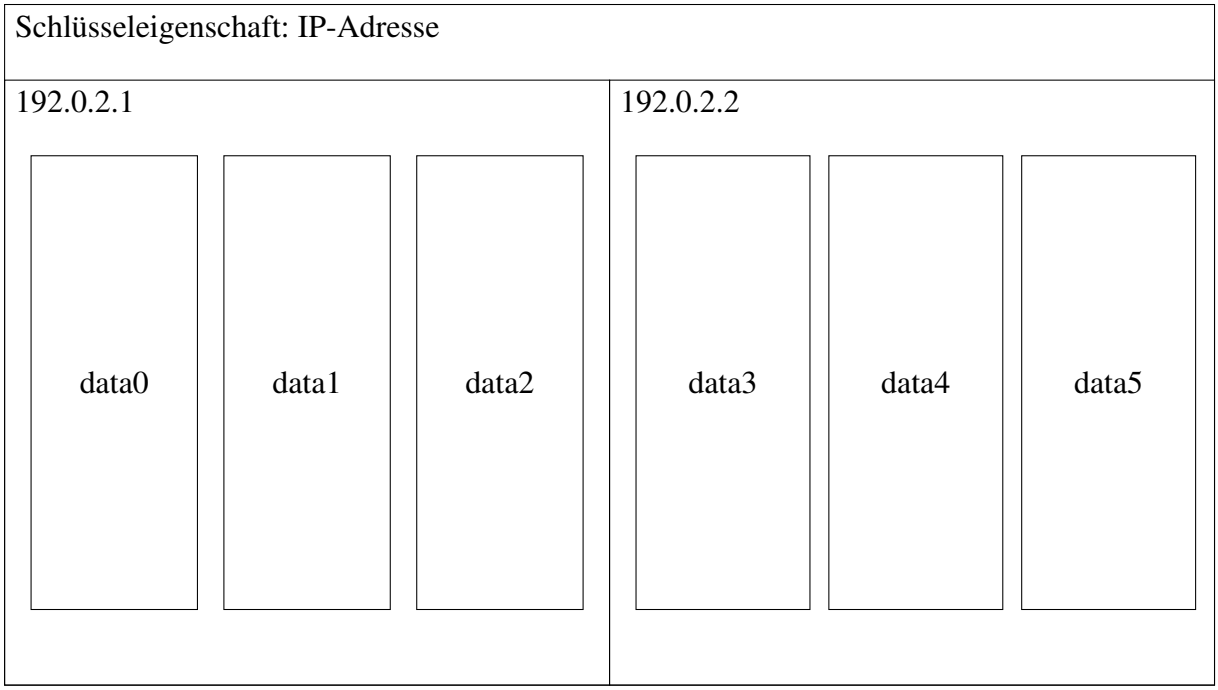

Abbildung 3.2: Schematische Darstellung der Visualisierungsmöglichkeiten der Baumstruktur

geografische Karte: Die Verwendung der Darstellung der Dateien als Standpunkte auf einer Karte basiert auf den Eigenschaften Lat (Latitude - Breitengrad) und Lon (Longitude - Längengrad)

der Model-Klasse. Für die Bestimmung eines Standpunktes anhand der IP-Adresse wird die API von http://ip-api.com verwendet. Von der API werden die Längen- und Breitengrade, sowie die Informationen zu Land, Region, Stadt und des Internetanbieters abgefragt.

Zeitstrahl: Die Darstellung als Zeitstrahl zeigt, wie viele Dateien zu einen Zeitpunkt hochgeladen wurden. Die X-Achse repräsentiert die Zeit und die Y-Achse die Anzahl der Dateien.

In Abbildung 3.3 ist eine beispielhafte *FileEntryVisual* HTML-Seite zu sehen. Die HTML-Seite bildet im rechten Teil des Fensters eine Baumkarte ab. Die repräsentierten Daten beziehen sich auf die Beispieldaten im Szenario des Kapitel 4.1 und werden dort näher beschrieben. Im linken Bereich des Fensters ist der Navigationsbereich zu erkennen. Es wird das ausgewählte *Set A* angezeigt, welches die ungeschützten Daten repräsentiert. Für das Wechseln zwischen den beiden Datenmengen *Set A* und *Set B* sind zwei Schaltflächen mit den Beschreibungen A und B zu erkennen. Unter den beiden Schaltflächen sind zwei Auswahlmenüs platziert. Das Obere zur Auswahl der Visualisierung, das Untere zur Auswahl der Schlüsseleigenschaft, welche zum Erstellen der Baumkarte benötigt wird. Darunter sind die verschiedenen Eigenschaften für eine ausgewählte Datei aufgelistet. Die ausgewählte Datei in der Abbildung ist dataAlice1, welche in der Visualisierung im rechten Teil des Fensters als grüne Box zu erkennen ist. Durch das Anklicken einer Box werden die Eigenschaften dieser Datei angezeigt.

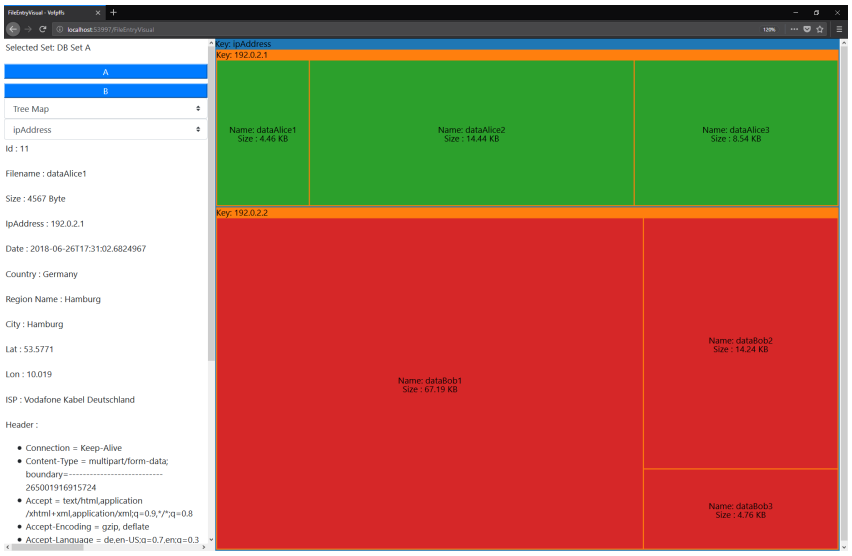

Abbildung 3.3: FileEntryVisual HTML-Seite mit Beispieldaten aus Kapitel 4

4 Szenarien

Um die Visualisierungen erzeugen zu können werden Daten benötigt, die verschiedenen Benutzern gehören. Diese Daten sollen wie bereits ausgeführt als geschützte Daten und ungeschützte Daten vorhanden sein. Dafür werden im folgenden zwei Szenarien definiert, anhand welcher die Visualisierungen beispielhaft dargestellt werden. Diese Beispiele basieren nicht auf Echtdaten. Die Daten wurden so gewählt, dass der Effekt durch die Visualisierung präsentiert werden kann. Die Verkehrs- und Metadaten werden für jeden Benutzer so gewählt, dass diese sich unterscheiden. Für die genauen Daten werden Standartwerte und Beispieldaten aus anderen Quellen verwendet. Für die benutzten IP-Adressen wurden die nach [RFC5737] gewählten IP-Adressen für Dokumentationszwecke gewählt. Die Werte für die Header *Accept* und *Accept-Encoding* entsprechen den Beispielen aus [RFC7231, sec. 5.3.2] und [RFC7231, sec. 5.3.4]. Der Header *User-Agent* für die Benutzer wurden aus [Sch17] übernommen.

4.1 Beispiel: Alice und Bob

Um die Visualisierungen an einem Beispiel zu erläutern wird ein Szenario definiert, welches die Benutzung des Dokumentenspeichers und die verwendeten Daten beschreibt. Dazu werden zwei Benutzer Alice und Bob definiert, welche sich beide in Hamburg befinden und sich in den Verkehrs- und Metadaten unterscheiden, sodass sie anhand dieser klar unterschieden werden können. Alice und Bob laden jeweils drei Dateien mit und ohne die Verwendung einer Methode zum Anonymisieren ihrer Daten hoch. Alice Dateien sind dataAlice1, dataAlice2 und dataAlice3 und Bobs Dateien sind dataBob1, dataBob2 und dataBob3 und haben alle unterschiedliche Dateigrößen. Die Verkehrs- und Metadaten für diese Dateien sind so gewählt, dass sich die Verkehrsdaten von Alice und Bob im *User-Agent* Headerfeld unterscheiden. Es wird angenommen, dass die Dateien von Alice einzeln in einem Abstand von mindestens zehn Minuten hochgeladen werden. Die Dateien von Bob werden alle zusammen in einem Dateiupload hochgeladen. Für die Verwendung einer Methode zum Anonymisieren ihrer Daten werden zwei Fälle betrachtet:

1. Die Verwendung eines Proxys, welcher von Alice und Bob verwendet wird.
2. Die Verwendung des Tor-Netzwerks von Alice und Bob.

Für den Proxy nehmen wir an, dass die End-to-End Header [RFC2616, sec. 13.5.1] unverändert übermittelt werden. Bei der Verwendung des Tor-Netzwerks nehmen wir an, dass Alice und Bob den Tor-Browser[1] verwenden und die Standarteinstellungen des Browsers nicht verändert haben. Das Tor-Netzwerk ist einfach beschrieben ein Netzwerk aus Rechnern, welches dazu da ist den Datenverkehr der Benutzer zu anonymisieren. Auf die genaue Funktionsweise des Tor-Netzwerks und des Tor-Browsers wird hier nicht eingegangen, sondern auf [DMS04] verwiesen.

Die zu visualisierenden Daten bestehen demnach aus drei Datenmengen:

1. https://www.torproject.org/projects/torbrowser.html.en

- der ungeschützten Datenmenge
- der durch einen Proxy geschützten Datenmenge
- und der durch die Verwendung der Tor-Browsers geschützten Datenmenge.

Die Visualisierungen dieser drei Datenmengen werden in den folgenden Abschnitten vorgestellt und ausgewertet.

Für die Benutzer nehmen wir folgende Verkehrs- und Metadaten an:

Alice

- IP:
  - IP-Adresse: 192.0.2.1
  - Lat: 53.5770988464355 Lon: 10.0190000534058
  - Land: Deutschland Region: Hamburg Stadt: Hamburg
- Headerfingerprint: *764529ca99fefafd6f805bca7f2e5194* aus :
  - *Accept*: text/*, text/plain, text/plain;format=flowed, */*
  - *Accept-Encoding*: compress, gzip
  - *User-Agent*: Mozilla/5.0 (Macintosh; Intel Mac OS X x.y; rv:42.0) Gecko/20100101 Firefox/42.0

Bob

- IP:
  - IP-Adresse: 192.0.2.2
  - Lat: 53.5830001831055 Lon: 9.98130035400391
  - Land: Deutschland Region: Hamburg Stadt: Hamburg
- Headerfingerprint: *f0ed0280bb701436185829473d55a185* aus:
  - *Accept*: text/*, text/plain, text/plain;format=flowed, */*
  - *Accept-Encoding*: compress, gzip
  - *User-Agent*: Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:47.0) Gecko/20100101 Firefox/47.0

4.1.1 IP-Adressen bezogene Visualisierung

In den weiteren Abbildungen dieses Kapitels werden immer die Beispieldaten von Alice und Bob mit dem Visualisierungstool des Dokumentenspeichers dargestellt.

Die Abbildung 4.1 zeigt die Visualisierung der Baumkarte für die Datenmenge der ungeschützten Dateien nach dem schematischem Beispiel aus Abbildung 3.2. In Abbildung 4.1 ist eine blaue Box als Wurzelknoten zu erkennen, welche den Titel: „*Key: ipAddress*“ trägt und damit die gewählte Schlüsseleigenschaft benennt. In diese Box sind zwei orange Boxen eingebettet. Beide Boxen tragen einen Titel, der obere „*Key: 192.0.2.1*“, die IP-Adresse von Alice, und der untere „*Key: 192.0.2.2*“, die IP-Adresse von Bob. Die obere Box (mit Alices IP-Adresse betitelt) sind drei weitere grüne Boxen eingebettet, welche für die Dateien dataAlice1, dataAlice2 und dataAlice3 stehen. In der unteren Box (mit Bobs IP-Adresse betitelt) sind drei rote Boxen eingebettet, welche die Dateien dataBob1, dataBob2 und dataBob3 darstellen. Die Farbe der einzelnen Boxen ist abhängig von dem jeweiligen Elternknoten in der Baumstruktur, sodass Kindknoten eines Knoten immer gleich eingefärbt sind. Somit wird die Zusammengehörigkeit von Dateien anhand der farblichen Markierung sofort sichtbar. Die blaue Box stellt immer den Wurzelknoten dar, die orangenen dessen Kindknoten, welche die Einträge für die gewählte Schlüsseleigenschaft der gesammelten Verkehrs- und Metadaten sind. Hier in diesem Beispiel die IP-Adressen von Alice und Bob.

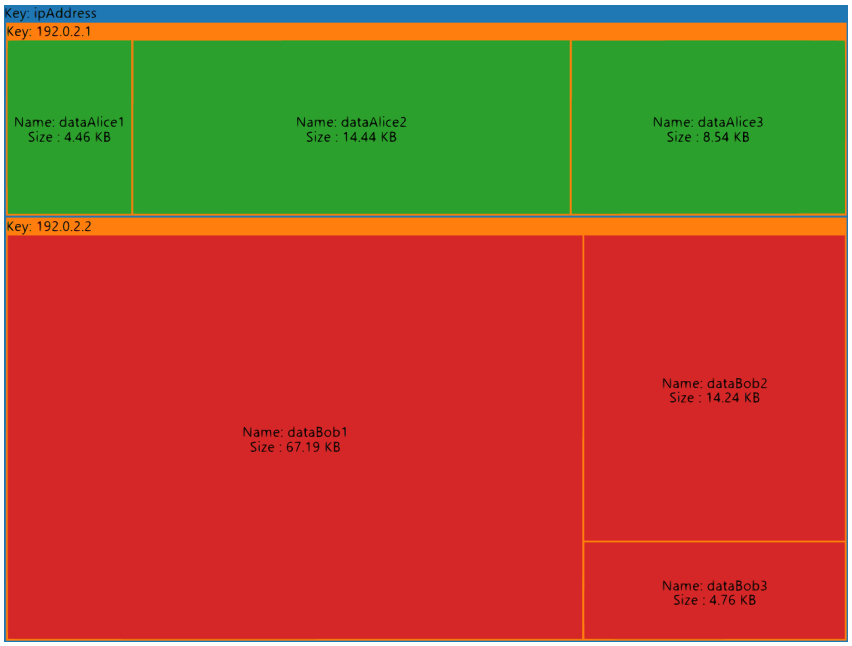

Abbildung 4.1: Baumkarte der IP-Adressen der ungeschützten Beispieldaten von Alice und Bob

In Abbildung 4.2 ist eine Karte von Hamburg Zentrum zu erkennen. Auf dieser Karte sind 2 Standpunkte markiert. Ein Punkt liegt bei Lat: 53.5770988464355 Lon: 10.0190000534058, welcher aus der IP-Adresse von Alice ermittelt wurde und dem definierten Standpunkt von Alice entspricht. Der zweite Punkt liegt bei Lat: 53.5830001831055 Lon: 9.98130035400391, welcher aus der IP-Adresse von Bob ermittelt wurde und dem im Szenario definierten Standpunkt von Bob entspricht. Beim Überfliegen eines Punktes mit der Maus wird im Szenario für jeden Punkt

die zugehörige IP-Adresse als Tooltip[2] angezeigt und die Anzahl von Dateien, welche von dieser IP-Adresse hochgeladen wurden.

Abbildung 4.2: Standorte ermittelt aus den ungeschützten Beispieldaten von Alice und Bob

Die Darstellung der Daten spiegelt somit die definierten Beispieldaten korrekt wieder und erlaubt es die jeweiligen Daten den richtigen Benutzern zuzuordnen. Der Servicebetreiber kennt somit die Position von Alice und Bob und kann die von ihnen hochgeladenen Dateien durch die IP-Adresse identifizieren.

Für die Verwendung von datenschutzfreundlichen Methoden zum Anonymisieren von Daten ergeben sich de folgenden Abbildungen.

Beim Betrachten des ersten Falls, der Verwendung eines Proxys, ergeben sich die Visualisierungen wie in Abbildung 4.3 und Abbildung 4.4. In Abbildung 4.3 ist nur eine orangene Box für die IP-Adresse des Proxys dargestellt. Alle sechs definierten Dateien, die Dateien von Alice dataAlice1, dataAlice2 und dataAlice3 und die Dateien von Bob dataBob1, dataBob2 und dataBob3, sind dieser IP-Adresse zugeordnet und als grüne Boxen unter der IP-Adresse des Proxys angeordnet. In Abbildung 4.4 ist ein Standpunkt in Berlin markiert. Der Standpunkt liegt bei Lat: 52.5167 Lon: 13.4 und ist durch die IP-Adresse des Proxys ermittelt worden.

2. siehe https://en.wikipedia.org/wiki/Tooltip

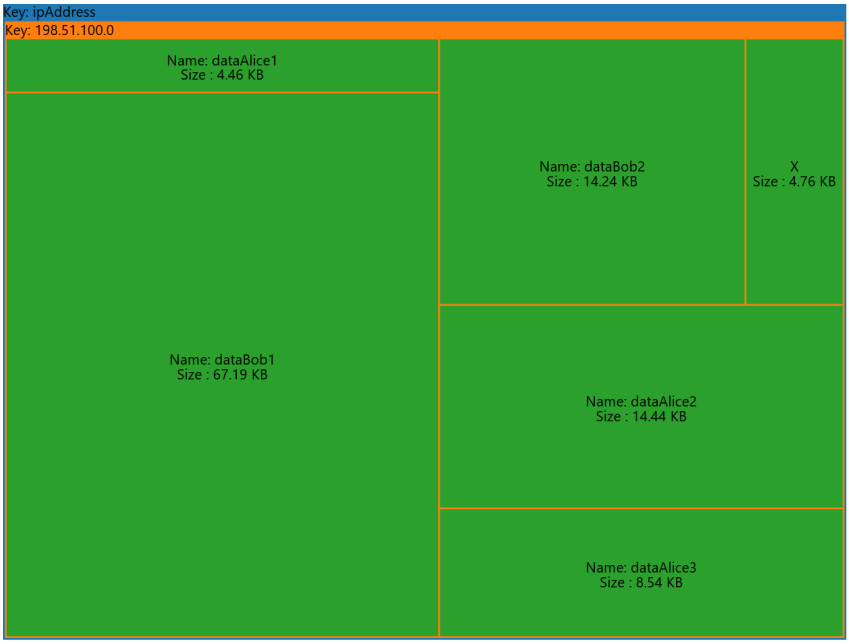

Abbildung 4.3: Baumkarte der IP-Adresse der Beispieldaten von Alice und Bob bei der Verwendung eines Proxys

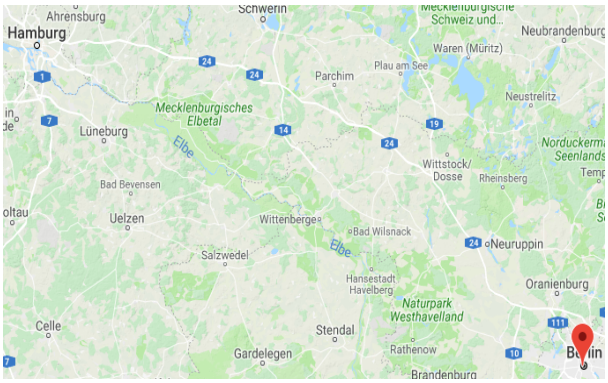

Abbildung 4.4: Standorte von Alice und Bob bei der Verwendung eines Proxys

Beim Vergleich der Abbildungen 4.1 und 4.3 sowie 4.2 und 4.4 ist klar zu erkennen, dass durch die Verwendung eines Proxys, der Servicebetreiber die einzelnen Dateien nicht mehr anhand der IP-Adresse deren tatsächlichen Besitzern zuordnen kann. Die Zuordnung der Dateien zu einer einzelnen IP-Adresse lässt dem Servicebetreiber bei der alleinigen Betrachtung der IP-Adresse nur darauf schließen, dass alle Dateien von demselben Benutzer stammen. Die Dateien von Alice und Bob sind somit durch die Verwendung des Proxys durch eine Anonymitätsmenge geschützt, sodass die Dateien in der durch die IP-Adresse erzeugte Gruppierung anonym werden. Da die Beispieldaten zwei Benutzer definieren und der Servicebetreiber anhand des verwendeten Proxys nur auf einen Benutzer schließen kann, ist der Effekt durch die Verwendung des Proxys klar anschaulich geworden.

Abbildung 4.5 und 4.6 zeigen die Visualisierungen für die Verwendung des Tor-Netzwerks als Methode zum Anonymisieren der Daten. In der Abbilung 4.5 sind vier verschiedene orange Boxen für vier verschiedene IP-Adressen abgebildet.

Die Zuordnung der Dateien zu den IP-Adressen entspricht:

203.0.113.1
Datei: dataAlice1, Farbe: grün

203.113.5
Datei: dataAlice2, Farbe: lila

203.0.113.13
Datei: dataAlice3, Farbe: braun

203.0.113.2
Farbe: rot

1. dataBob1
2. dataBob2
3. dataBob3

Für jede dieser IP-Adressen ist in der Karte in Abbildung 4.6 ein Standort abgebildet, welche über Deutschland verteilt sind.

1 Hermsdorf (Thüringen) - Lat: 50.8919 Lon: 11.8682

2 Karlsruhe (Innenstadt-Ost) - Lat: 49.0096 Lon: 8.41217

3 Frankfurt am Main - Lat: 50.1109 Lon: 8.68213

4 Berlin (Charlottenburg-Wilmersdorf) - Lat: 52.5239 Lon: 13.3214

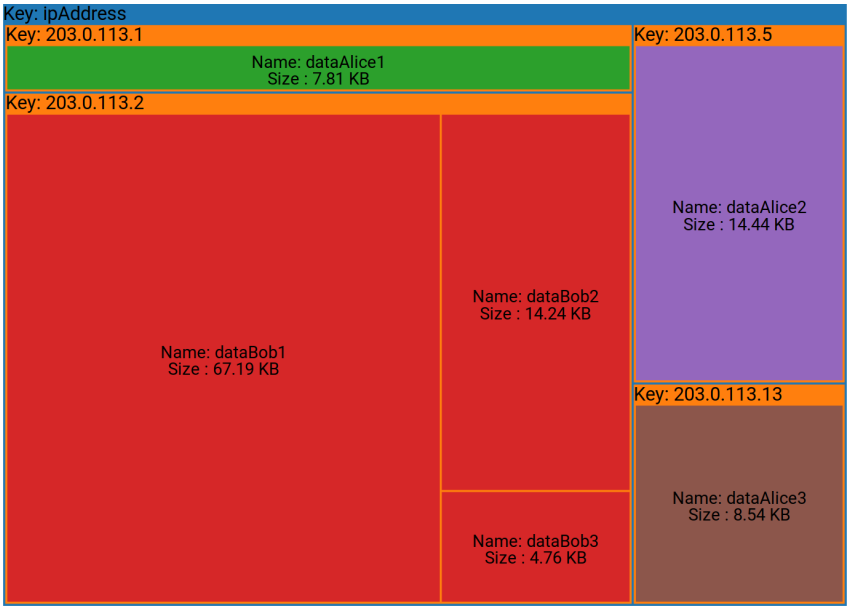

Abbildung 4.5: Baumkarte der IP-Adresse der Beispieldaten von Alice und Bob bei der Verwendung des Tor-Netzwerks

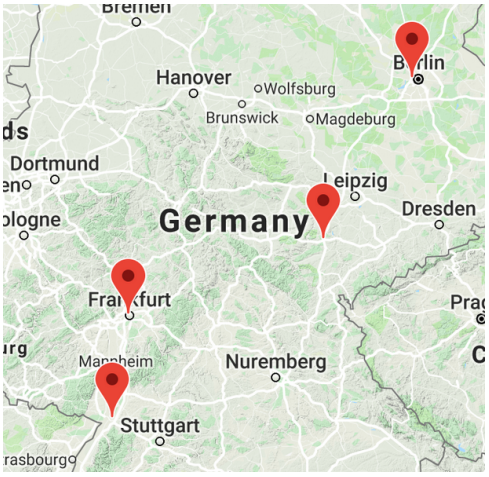

Abbildung 4.6: Standorte der Dateien von Alice und Bob bei der Verwendung des Tor-Netzwerks

Vergleicht man wiederum Abbildung 4.1 mit 4.5 und 4.2 mit 4.6 fallen deutliche Unterschiede in den Visualisierungen auf. Während, wie bereits (oben) ausgeführt, in Abbildung 4.1 die Beispieldaten korrekt gruppiert visuell dargestellt werden, lässt die Darstellung aus Abbildung 4.5 und Abbildung 4.6 dies nicht mehr zu. Die Dateien von Bob können zwar zusammen gruppiert werden, jedoch sind die Dateien von Alice auf drei verschiedene IP-Adressen verteilt. Die Gruppierung der Dateien von Bob ist zwar korrekt jedoch entspricht die IP-Adresse und somit auch der Standort, der daraus abgeleitet ist, nicht mehr dem definierten Standort was darauf zurückzuführen ist, dass die Dateien alle zusammen hochgeladen wurden. Ein Servicebetreiber muss so anhand der IP-Adresse annehmen, dass die Dateien von vier verschiedenen Benutzern stammen. Dies widerspricht klar den definierten Daten und zeigt die Anonymisierung der Daten durch die Verwendung des Tor-Netzwerks.

Die Betrachtung beider Fälle von Anonymisierung (Verwendung eines Proxys oder des Tor-Netzwerks) ergibt, das beide Schutzmethoden im Hinblick auf die IP-Adresse einen derart ausreichenden Effekt gehabt haben, dass eine korrekte Zuordnung von Datei und Benutzer bezüglich der IP-Adresse nicht mehr möglich waren.

4.1.2 Headerfingerprint bezogene Visualisierung

Nach der Auswertung der Informationen, welche aus der IP-Adressbezogenen Visualisierung abgeleitet werden können, kann ein Servicebetreiber weitere der gesammelten Eigenschaften untersuchen um festzustellen, ob andere Eigenschaften die gleichen Schlussfolgerungen erlauben wie die IP-Adresse und diese somit erhärten oder ob andere Eigenschaften zu anderen Schlussfolgerungen führen.

Um die Veränderungen bezüglich des Headerfngerprints visualisieren zu können, muss in der *FileEntryVisual* HTML-Seite im Navigationsbereich die Schlüsseleigenschaft auf den *Headerfingerprint* umgestellt werden.

Für die ungeschützten Beispieldaten von Alice und Bob erhalten wir die Visualisierung als Baumkarte, wie sie in Abbildung 4.7 gezeigt wird. Zu erkennen sind zwei orange Boxen für die beiden verschiedenen Headerfingerprints. Der oberen orangen Box mit dem Headerfingerprint

764529ca99fefafd6f805bca7f2e5194 sind drei Dateien dataAlice1, dataAlice2 und dataAlice3 zugeordnet, welche alle farblich grün markiert sind. Der unteren orangen Box mit dem Headerfingerprint *f0ed0280bb701436185829473d55a185* sind die drei Dateien dataBob1, dataBob2 und dataBob3 zugeordnet, welche farblich rot markiert sind.

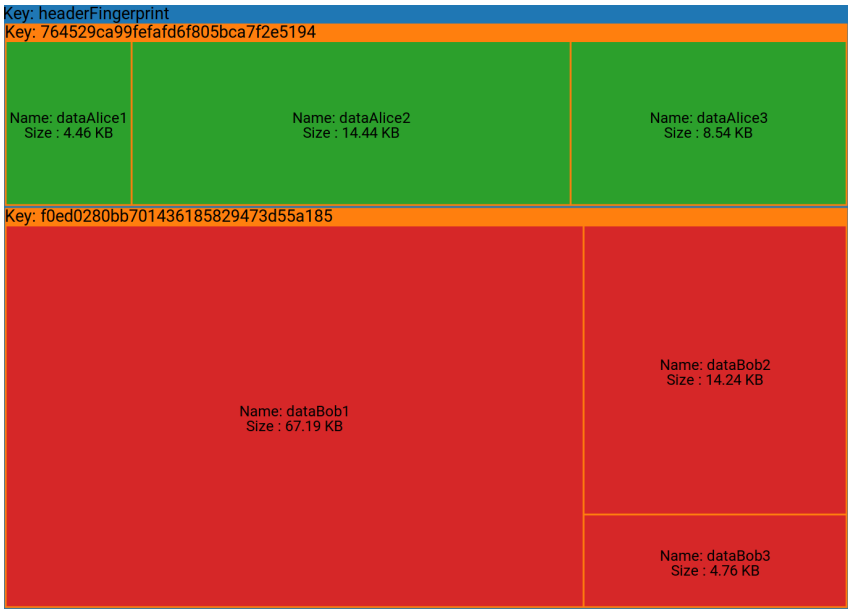

Abbildung 4.7: Baumkarte der Beispieldaten von Alice und Bob bei Betrachtung der Schlüsseleigenschaft: Headerfingerprint

Wie in Abbildung 4.1 stellt Abbildung 4.7 die Zusammengehörigkeit der Dateien dataAlice1, dataAlice2 und dataAlice3 sowie den Dateien dataBob1, dataBob2 und dataBob3 richtig dar.

Die Verwendung des Proxys hat keinen Effekt bezüglich der Visualisierung des Headerfingerprints und entspricht somit der Abbildung 4.7. Dies ist darauf zurückzuführen, dass die Verwendung des Proxys die Header-Felder *User-Agent*, *Accept* und *Accept-Encoding* nicht verändert und die eigentlichen Werte von Alice und Bob weitergegeben werden.

Bei Betrachtung des zweiten Falls, der Verwendung des Tor-Netzwerkes, entspricht die Visualisierung der Abbildung 4.8. Zu erkennen ist, dass ähnlich der Abbildung 4.3 alle Dateien unter einem Headerfingerprint *274e7dc5014d882a1cf8756d0600a52a* angeordnet sind (die Datei dataBob3 ist mit einem X dargestellt, da nicht genung Platz ist, um den Dateinamen anzuzeigen). Dies ist darauf zurückzuführen, dass durch die Verwendung des Tor-Browsers Alice und Bob standardisierte Werte für die Header-Felder *User-Agent*, *Accept* und *Accept-Encoding* verwenden. Die standartisierten Felder erzeugen einen identischen Headerfingerprint für Alice und Bob.

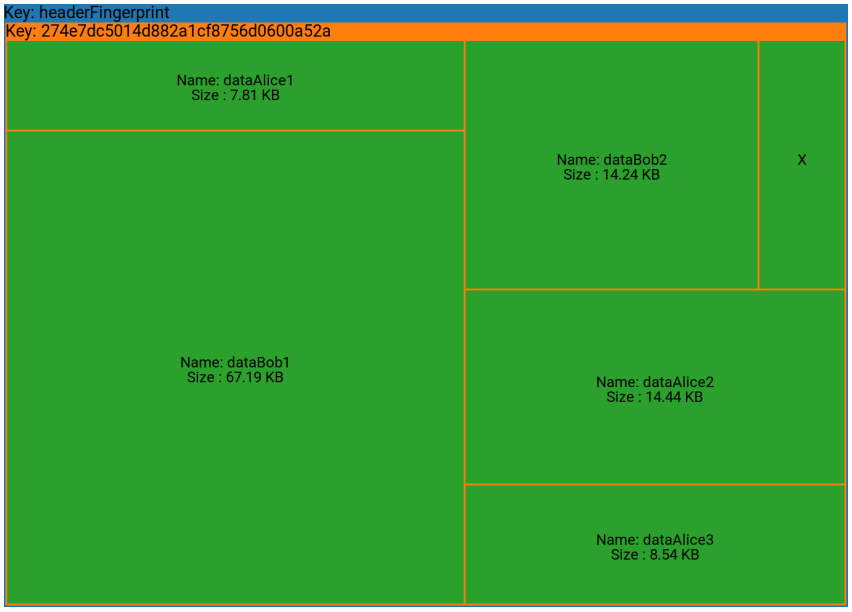

Abbildung 4.8: Baumkarte des Headerfingerabdrcuks der Beispieldaten von Alice und Bob geschützt druch die Verwendung des Tor-Netzwerks

Die Verwendung des Tor-Browsers hat somit zur Folge, dass die Dateien und Alice und Bob alle unter einem einzelnen Headefingerprint gruppiert werden. Für den Servicebetreiber bedeutet dies, dass anhand des Headerfingerprints anzunehmen ist, dass die Dateien alle von einem einzelnen Benutzer stammen. Der Effekt der Verwendung des Tor-Netzwerks und -Browsers ist so sichtbar, da die Informationen, die ein Servicebetreiber ableiten kann, sich von denen der ungeschützten Daten unterscheiden.

Da die Verwendung des Proxys keinen Effekt auf die für den Headerfingerprint ausschlaggebenden Header hatte, ist der Headerfingerprint der Benutzer trotz Benutzung des Proxys gleich geblieben. Die Verwendung des Tor-Netzwerks hat hingegen dazu geführt, dass die Headerfingerprints von Alice und Bob, aus dem ungeschützten Beispiel, beide durch einen standardisierten Headerfingerprint ersetzt wurden und somit die korrekte Zuordnung der Dateien zu den jeweiligen Benutzern nicht mehr möglich war.

Die Verwendung des Tor-Netzwerks hat somit eine Anonymitätsmenge für die Eigenschaft des Headerfingerprints erzeugt, in welcher die Dateien anonym sind und den Benutzern Alice und Bob nicht zugeordnet werden können.

4.1.3 Vergleich der Schutzmethoden

Die Visualisierungen haben eindeutig und anschaulich gezeigt, dass die Verwendung eines Proxys im Hinblick auf die IP-Adresse (Abbildung 4.3) nicht aber im Hinblick auf den Headerfingerprint (Abbildung 4.7) einen anonymisierenden Schutz bewirkt. So wird durch die Verwendung eines Proxys die eigentliche IP-Adresse des Benutzers durch die des Proxys maskiert, der Headerfingerprint des Benutzer jedoch nicht verändert. Sollten mehrere Benutzer den gleichen Proxy verwenden (wie im Beispiel von Alice und Bob) können die Dateien dieser Nutzer anhand der IP-Adresse nicht mehr korrekt zugeordnet werden. Sollte jedoch ein Benutzer

einen Proxy alleine benutzen, so kann der Proxy lediglich die IP-Adresse des Benutzer maskieren, erlaubt jedoch immer noch die Zuordnung der Dateien über z. B. den Headerfingerprint. Da der Headerfingerprint nicht verändert wird, kann ein Benutzer trotz der Verwendung eines Proxys über seinen Headerfingerprint identifiziert werden, falls dieser einzigartig genug sein sollte. Nach dem Beispiel der Visualisierungen für die Verwendung des Proxys hat ein Servicebetreiber verschiedene sich widersprechende Auslegungsmöglichkeiten für die visualisierten Daten. Aus der IP-Adresse kann ein Servicebetreiber nur auf einen Benutzer schließen, der sechs Dateien hochgeladen hat. Aus dem Headerfingerprint kann jedoch auf zwei verschiedene Benutzer geschlossen werden. Ein Servicebetreiber muss anhand dieser Informationen nun abwägen, welche der Schlussfolgerungen wahrscheinlicher ist. Ein Servicebetreiber kann annehmen, dass alle Dateien von einem Benutzer stammen und z. B. durch die Verwendung eines anderen Browser oder durch ein Update des verwendeten Browsers sich der Headerfingeprint des Benutzers geändert hat. Oder es wird angenommen, dass es zwei Benutzer gibt, welche einen verschiedenen Headerfingerprint besitzen und z. B. durch die Verwendung eines Proxys, wie in dem Beispiel angenommen, ihre IP-Adresse maskieren. Sollte ein Servicebetreiber die Gänze der gesammelten Eigenschaften der Dateien von Bob betrachten wäre so auffällig, dass die drei Dateien alle identische Verkehrsdaten haben. Ob die restlichen Dateien ebenfalls zu diesem Benutzer oder anderen Benutzern gehören ist anhand der betrachteten Informationen nicht eindeutig festzustellen. Die Verwendung des Tor-Netzwerks dagegen kann, wie in Abbildung 4.5 und Abbildung 4.8 gezeigt, Schutz vor der Identifizierung durch die IP-Adresse und den Headerfingerprint bieten. Dabei ist zu beachten, dass die während durch die Visualisierung aus Abbildung 4.5 ein Servicebetreiber von vier Benutzern ausgehen muss. Die Visualisierung in Abbildung 4.8 nur auf einen Benutzer schließe lässt. Dass Bob in diesem Beispiel die Dateien alle als ein Dateiupload hochlud, macht seine Dateien leicht eindeutig zuordbar, da das Hochladen der Dateien in einer Anfrage an den Dokumentenspeicher dieses trivialisiert.

Aus dem definierten Szenario lässt sich schließen, dass während die Verwendung eines Proxys und des Tor-Netzwerks im Hinblick auf die Anonymisierung der IP-Adresse gleich erfolgreich seien können, die Verwendung eines Proxys gegenüber dem Tor-Netzwerk jedoch einen deutlichen Nachteil hat, bezogen auf die Möglichkeit durch seinen eigenen individuellen Headerfingerprint identifiziert zu werden. Auch dass Dateien einzeln hochgeladen werden sollten, ist anhand der Identifikation der Zusammengehörigkeit der Dateien von Bob sichtbar geworden.

4.2 Beispiel: Adam, Beth und Chris

Wie in Kapitel 4.1 wird ein Beispielszenario für die Vorstellung einer Visualisierungsmöglichkeit definiert. Die zu betrachtende Visualisierung ist der Zeitstrahl. Dafür werden drei Benutzer definiert, sowie ein Proxy, welcher als Schutzmethode verwendet wird. Die drei Benutzer laden jeweils mehrere Dateien hoch:

- Adam: drei Dateien
- Beth: drei Dateien
- Chris: zwei Dateien

Die Dateien von Adam werden täglich um 05:30 Uhr hochgeladen. Die Dateien von Beth werden am 26.6. um :

- 17:31 Uhr,

- 18:10 Uhr und
- 19:54 Uhr

hochgeladen. Die Dateien von Chris werden zusammen am 28.6. um 12.21 Uhr hochgeladen. Da die Visualisierung von IP-Adresse und Headerfingerprint bereits in Kapitel 4.1 behandelt wurden, werde diese Verkehrsdaten der Benutzer für dieses Beispiel vernachlässigt. Der Proxy, welcher als Schutzmethode verwendet wird, funktioniert wie folgt. Jede hochgeladene Datei wird im Proxy in einem Puffer gespeichert und dieser Puffer wird täglich um 00:00 Uhr einmal geleert, indem die gesammelten Dateien an den Dokumentenspeicher geschickt werden.

4.2.1 Zeitbezogene Visualisierung

In der zeitbezogenen Visualisierung werden die Zeitstempel verwendet, wann eine Datei hochgeladen wurde, um eine Visualisierung zu erzeugen. Diese Visualisierung zeigt wie viele Dateien zu welchem Zeitpunkt hochgeladen wurden.

In Abbildung 4.9 sind die ungeschützten Daten von Adam, Beth und Chris dargestellt. Die Dateien von Adam sind jeweils einzeln am 26.6., 27.6. und 28.6. um 5:30 Uhr zu erkennen. Die Dateien von Beth sind am 26.6. zwischen 17 und 20 Uhr abgebildet. Die Dateien von Chris sind am 28.6. zu erkennen und sind in einem Zeitpunkt mit der Mächtigkeit zwei zusammengefasst, da diese Dateien zum gleichen Zeitpunkt hochgeladen wurden.

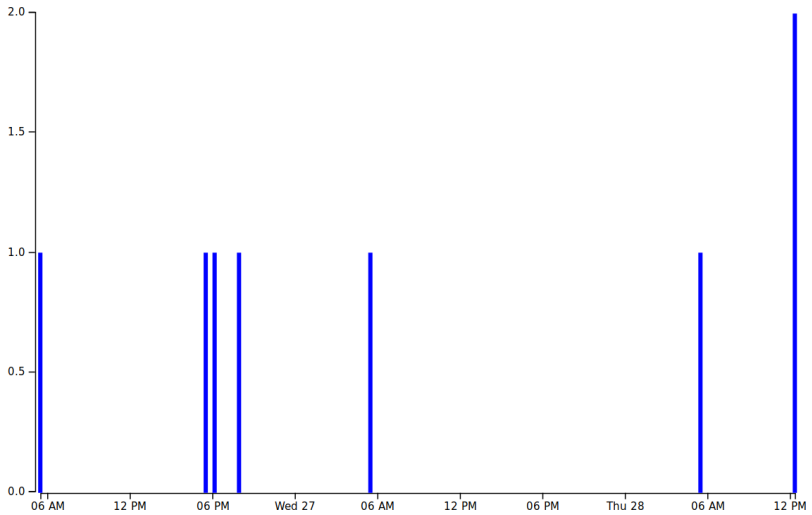

Abbildung 4.9: Zeitstrahl der ungeschützten Beispieldaten von Adam, Beth und Chris

Das regelmäßige Hochladen der Dateien von Adam ist in der Abbildung zu erkennen. Die immer gleich bleibenden Abstände zwischen den blauen Balken kann als ein Indiz dafür genommen werden, dass diese von demselben Benutzer stammen. Dies ist auch der Fall für die Dateien von Chris, da sie in einem Upload hochgeladen wurden und somit einen identischen Zeitstempel haben. Bei den Dateien von Beth ist eine Zuordnung nicht möglich, da die Zeitpunkte des Hochladens unregelmäßige Abstände haben.

Um diese möglichen Gruppierungen der Dateien von Adam und Chris aufzulösen, verwenden nun alle drei Benutzer den in Kapitel 4.2 definierten Proxy.

In der Abbildungs 4.10 ist nun die Visualisierung der gesammelten Daten bei der Verwendung des Proxys zu sehen. Wie nach Definition des Proxys gibt es nur drei Punkte im betrachteten Zeitraum, jeweils um 00:00 Uhr:

- 27.6.: 4 Dateien,
- 28.6.: 1 Datei,
- 29:6.: 3 Dateien.

Die Dateien vom 27.6. bestehen aus einer Datei von Adam und den drei Dateien von Beth. Die Dateien vom 29.6. bestehen aus einer Datei von Adam und den beiden Dateien von Chris. Die Datei vom 28.6 ist die Datei von Adam.

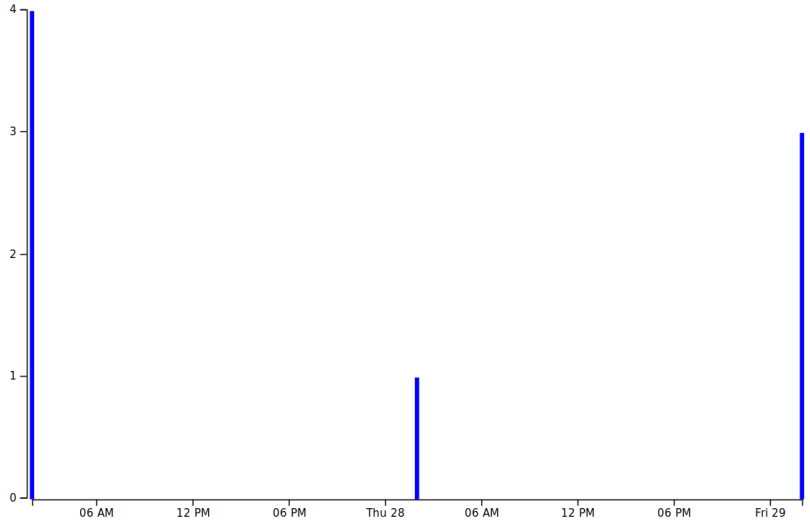

Abbildung 4.10: Zeitstrahl der durch einen Proxy geschützten Beispieldaten von Adam, Beth und Chris

Da für alle Dateien, die einen identischen Zeitstempel haben, angenommen werden kann, dass sie von demselben Benutzer stammen, ergibt sich bereits eine Veränderung bei der Zuordbarkeit der Daten von Adam und Chris. Da die zeitlichen Abstände zwischen den Uploads auch gleich sind, lässt sich weiter schließen, dass alle Dateien zusammengehörig sind. Ein Servicebetreiber kann so anhand der Visualisierung auf einen Benutzer, was dem Proxy entsprechen würde, schließen, jedoch nicht auf die Benutzer des Proxys.

Wie aus der Abbildung 4.9 abgeleitet werden konnte, gelingt die Gruppierung der Dateien von Adam und Chris, wenn diese ihre Dateien ohne die Verwendung einer Schutzmethode hochladen. Diese Gruppierung kann bei Nutzung des Proxys, wie sie in Kapitel 4.2 festgelegt wurde, nicht mehr erfolgen. Dabei sind jedoch zwei Dinge zu beachten. Die Benutzung des Dokumentenspeichers ist durch den Proxy erheblich verzögert, da die Dateien im Puffer des Proxys auf die Weiterleitung an den Dokumentenspeicher warten müssen. Während der Wartezeit können die Dateien z. B. nicht durch den Dokumentenspeicher zugänglich gemacht werden. Genauso ist zu beachten, dass die Verwendung des Proxys zwar den Servicebetreiber des Dokumentenspeichers daran hindert die originalen Zeitstempel der Dateien zu erfassen, der Servicebetreiber des Proxys jedoch immer noch dazu in der Lage ist.

4.3 Zusmamenfassung der Visualisierungen

In Kapitel 4.1 wurde anhand des Beispiels von Alice und Bob die Darstellungsmethode der Baumkarte und der Darstellung von Standpunkten auf einer Karte vorgestellt. Dabei wurde die Verwendung eines Proxys und des Tor-Netzwerks und Tor-Browsers als Schutzmethoden exemplarisch an Beispieldaten visualisiert und ausgewertet. Beide Methoden wurden hinsichtlich der Eigenschaft der IP-Adresse und des Headerfingerprints untersucht und eine Aussage darüber getroffen, inwiefern diese sich im gegebenen Beispiel eignen, um eine der beiden Eigenschaften zu anonymisieren.

In Kapitel 4.2 wurde anhand eines Beispiels von drei Benutzern die Darstellungsmethode des Zeitstrahls vorgestellt. Dabei wurde ein Proxy mit einem bestimmten definierten Verhalten als Schutzmethode eingesetzt. Es wurde gezeigt, dass durch die Betrachtung von regelmäßig auftretenden Zeitstempeln oder bei identischen Zeitstempeln Gruppierungen erzeugt werden können und wie ein speziell definierter Proxy dies verhindern kann.

5 Schluss

5.1 Zusammenfassung der Ergebnisse

Im Rahmen dieser Arbeit wurde ein einfacher Dokumentenspeicher sowohl konzipiert als auch implementiert. Entsprechend der Aufgabenstellung ist das Verhalten des Dokumentenspeichers so gewählt, dass die Verkehrs- und Metadaten der hochgeladenen Dateien gesammelt und anschließend visualisiert werden. Dazu wurden verschiedene Verwendungsmöglichkeiten im Hinblick auf das Hochladen von Dateien geschaffen. Zur Darstellung der gesammelten Daten und der angewendeten Schutzfunktion wurden verschiedene Visualisierungsmöglichkeiten implementiert. Diese Visualisierungsmöglichkeiten werden anhand zweier Beispielszenarien vorgestellt. Durch die Szenarien wird die Benutzung des Dokumentenspeichers mit und ohne die Verwendung von datenschutzfreundlichen Methoden zum Anonymisieren von Daten vorgestellt. Es wurden verschiedene Methoden zum Anonymisieren von Daten betrachtet. Die Unterschiede zwischen anonymisierten Daten und nicht anonymisierten Daten sowie den gewählten Methoden werden anhand von Abbildungen gezeigt und ausgewertet. Die Auswertung stellt vor allem geschützte und ungeschützte Datenmengen gegenüber. Anhand der Visualisierungen und der Auswertung kann eingeschätzt werden, welchen Effekt die Verwendung einer datenschutzfreundlichen Methode zum Anonymisieren von Daten auf die Möglichkeiten hat, wie ein Servicebetreiber aus den Verkehrs- und Metadaten Informationen über den Besitzer der gespeicherten Daten ableiten kann.

5.2 Limitierungen & Ausblick

Der Dokumentenspeicher erfüllt seine Funktion gemessen an seinem Verwendungszweck, jedoch gibt es einige Aspekte, anhand welcher der Dokumentenspeicher verbessert werden kann. Momentan ist der Dokumentenspeicher sehr einfach aufgebaut und hat nur eine kleine Menge an Features. Es können Dateien hoch- und runtergeladen und die gesammelten Verkehrs- und Metadaten anhand weniger einfacher Visualisierungen dargestellt werden.

Erweiterungsmöglichkeiten ergeben sich im Bereich des Funktionalität des Dokumentenspeichers. Es könnten Features, wie das Ersetzten von bereits hochgeladenen Dateien durch eine neue Version dieser Datei, implementiert werden. Genauso könnten die HTML-Seiten hinsichtlich der Benutzerfreundlichkeit und ihres Aussehens verbessert werden. Weiter wäre es sinnvoll, die verwendeten Verfahren zum Identifizieren von Benutzern, bisher hauptsächlich IP-Adresse und HTTP Fingerabdruck, durch weitere Methoden zur Identifikation eines Benutzers zu erweitern. Die bisherigen Verfahren können so verbessert werden, dass die recht einfach gehaltene Implementation durch eine fundiertere abgelöst wird. Neben der Untersuchung der Verkettbarkeit der Daten können auch Ansätze erschlossen werden, welche die Metadaten der Dateien genauer betrachten und versuchen aus den Metadaten Informationen abzuleiten, die für einen Servicebetreiber von Interesse sein könnten. Dabei könnten z. B. untersucht werden inwiefern die Größe einer Datei auf den Dateityp oder Dateiinhalt schließen lässt.

Die Visualisierungen selbst basieren momentan auf den einzelnen Eigenschaften, die im Dokumentenspeicher gesammelten werden. Diese Eigenschaften können momentan hauptsächlich durch die Baumkarte dargestellt werden. Während dieser Ansatz hinreichend ist, um in einem einfachen dafür konstruierten Fall diese Zuordnungen zu zeigen, ist es fraglich, ob dieser Ansatz bei Echtdaten zuverlässige Ergebnisse liefert. Neben dem Prüfen, ob die Baumkarte sich als Visualisierungsmethode eignet, sollte in weiteren Arbeiten nach anderen Visualisierungsmethoden gesucht werden, welche sich eignen, die Verkettbarkeit der Daten zu repräsentieren. Die bisherigen Visualisierungen könnten verbessert und neue Visualisierungsmöglichkeiten geschafft werden. Dazu könnte die Arbeit von Chi [Chi00] verwendet werden, um eine Struktur in die Implementation der Visualisierungen einfließen zu lassen und die Implementation so zu vereinfachen.

In weiteren Arbeiten kann untersucht werden, welche Effekte die Kombination von verschiedenen Eigenschaften auf die Zuverlässigkeit der Zuordnung und die daraus resultierenden Visualisierungen hat. Die Ansätze aus [Eck10] können dabei weiter verfolgt werden.

Zusätzlich kann in weiteren Arbeiten ein Klient für den Dokumentenspeicher entwickelt werden, welcher dem Verwendungszweck aus Kapitel 3 entspricht. Der Klient sollte so eine Menge an verschiedenen Schutzfunktionen anbieten, die beim Hochladen einer Datei verwendet werden können. Zusätzlich zum Hochladen der geschützten Datei müsste dann zum Zweck der Vergleichbarkeit auch die ungeschützte Datei hochgeladen werden. Durch die beiden unterschiedlichen API-Endpunkte für die geschützten und ungeschützten Daten können so die für die Visualisierung verwendeten Datenmengen erzeugt werden. Durch die Verwendung dieses Klienten würde so das Erzeugen von Beispielszenarien oder das Sammeln von Echtdaten vereinfacht werden.

Literatur

[Bod+12] Károly Boda u. a. *User Tracking on the Web via Cross-browser Fingerprinting*. In: *Proceedings of the 16th Nordic Conference on Information Security Technology for Applications*. NordSec'11. Tallinn, Estonia: Springer-Verlag, 2012, S. 31–46. ISBN: 978-3-642-29614-7. DOI: 10.1007/978-3-642-29615-4_4. URL: http://dx.doi.org/10.1007/978-3-642-29615-4_4.

[Chi00] Ed H. Chi. *A Taxonomy of Visualization Techniques Using the Data State Reference Model*. In: *Proceedings of the IEEE Symposium on Information Vizualization 2000*. INFOVIS '00. Washington, DC, USA: IEEE Computer Society, 2000, S. 69–. ISBN: 0-7695-0804-9. URL: http://dl.acm.org/citation.cfm?id=857190.857691.

[DMS04] Roger Dingledine, Nick Mathewson und Paul Syverson. *Tor: The Second-generation Onion Router*. In: *Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13*. SSYM'04. San Diego, CA: USENIX Association, 2004, S. 21–21. URL: http://dl.acm.org/citation.cfm?id=1251375.1251396.

[Eck10] Peter Eckersley. *How Unique is Your Web Browser?* In: *Proceedings of the 10th International Conference on Privacy Enhancing Technologies*. PETS'10. Berlin, Germany: Springer-Verlag, 2010, S. 1–18. ISBN: 3-642-14526-4, 978-3-642-14526-1. URL: http://dl.acm.org/citation.cfm?id=1881151.1881152.

[GK10] Martin Graham und Jessie Kennedy. *A Survey of Multiple Tree Visualisation*. In: *Information Visualization* 9.4 (Dez. 2010), S. 235–252. ISSN: 1473-8716. DOI: 10.1057/ivs.2009.29. URL: http://dx.doi.org/10.1057/ivs.2009.29.

[Lut18] D. Roth Rick A. Shaun Luttin. *Introduction to ASP.NET Core*. 2018. URL: https://docs.microsoft.com/en-us/aspnet/core/?view=aspnetcore-2.1.

[RFC1321] R. Rivest. *The MD5 Message-Digest Algorithm*. April 1992. URL: https://tools.ietf.org/html/rfc1321.

[RFC2616] R. Fielding J.Gettys J. Mogul H. Frystyk L. Masinter P. Leach T. Berners-Lee. *Hypertext Transfer Protocol – HTTP/1.1*. June 1999. URL: https://tools.ietf.org/html/rfc2616.

[RFC5737] J. Arkko M. Cotton L. Vegoda. *IPv4 Address Blocks Reserved for Documentation*. January 2010. URL: https://tools.ietf.org/html/rfc5737.

[RFC7231] R. Fielding J. Reschke. *Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content*. June 2014. URL: https://tools.ietf.org/html/rfc7231.

[Rib] S. Ribecca. *The Data Visualisation Catalogue*. URL: https://datavizcatalogue.com/methods/treemap.html.

[Sch17] F. Scholz. *User-Agent*. 2017. URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/User-Agent.

[Sok17] Daniel AJ Sokolov. *Rekordhack bei Yahoo war drei Mal so groß*. 2017. URL: https://www.heise.de/security/meldung/Rekordhack-bei-Yahoo-war-drei-Mal-so-gross-3849303.html.