



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 12. Juli 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.

## Zusammenfassung

1. Dokumentenspeicherdienste Vorteile (Problemstellung erläutern)
2. Mögliche Datenschutz unfreundliche Aspekte von gängigen Anbietern (Problemstellung erläutern)
3. Entwicklung des Dokumentenspeichers und der Visualisierung zur deutlich Veranschaulichung von Potentiellen Unterschieden zwischen der Verwendung von Datenschutz freundlichen Methoden zum Anonymisieren oder nicht. (Bearbeitung der Problemstellung)
  - a) Implementation des Dokumentenspeichers
  - b) Implementation der API zur Datenübergabe
  - c) Implementation des Visualisierungswerkzeug
  - d) Darstellung der Szenarien zur Benutzung des Visualisierungswerkzeug

# Inhaltsverzeichnis

<b>1</b>	<b>Einleitung</b>	<b>5</b>
<b>2</b>	<b>Grundlagen</b>	<b>6</b>
2.1	Verwendungszweck des Dokumentenspeichers . . . . .	6
2.2	Terminologie . . . . .	7
<b>3</b>	<b>Hauptteil</b>	<b>8</b>
3.1	Implementation des Dokumentenspeichers . . . . .	8
3.2	Darstellung: IP Tree Map und IP Google Map . . . . .	10
3.3	Darstellung: Headerfingerprinting . . . . .	15
3.4	Darstellung: Time Line . . . . .	16
<b>4</b>	<b>Schluss</b>	<b>17</b>
4.1	Zusammenfassung der Ergebnisse . . . . .	17
4.2	kritische Bewertung des Ergebnisse . . . . .	17
4.3	neue Problemstellungen, Möglichkeiten zur Weiterführung der Arbeit . . . . .	17

# 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 **stellen auch die Sicherheit der Dateien sicher** und machen sie global jeder zeit verfügbar. Durch die große Datensammlung dieser Dienstleister machen sie sich nicht nur selbst zu lukrativen Zielen von gezielten Angriffen (**Yahoo, UBER** etc.), jedoch auch die Dienstleister selber können die Daten auswerten und weitere **Meta-Daten** über die Nutzer sammeln und weiter verwenden. Vor allem private Nutzer sind meist gar nicht über die Risiken und das Missbrauchspotenziale aufgeklärt, welche die Verwendung solcher Dienstleistungen mit sich bringen. Methoden zur Verschlüsselung oder des Anonymisieren von Daten sind Benutzern meist nicht bekannt, werden von den Diensteanbietern nicht angeboten oder sind schwer umzusetzen da es einen meist erheblichen Aufwand für die Benutzer bedeutet und Kompetenzen erfordert welche diese Benutzer nicht besitzen. Um genau die Risiken und Missbrauchspotenziale aufzuzeigen wird im Zuge dieser Arbeit ein Dokumentenspeicherdienst entwickelt, welcher prinzipiell alle Meta-Daten der Nutzer sammelt und diese in einer visuellen Darstellung zusammen fasst. Zur Implementation des Dokumentenspeichers wird dabei das Microsoft ASP.NET Core Framework verwendet. Das Framework wird benutzt um die Webbenutzeroberfläche sowie der Web-APIs des Dokumentenspeichers zu realisieren. Dazu wird das Javascript Framework Data-Driven Documents, i.d.R. d3.js genannt, zur Visualisierung der Daten verwendet. Der Dokumentenspeicher soll vor allem den Unterschied zwischen der Verwendung von Methoden zur Verschlüsselung oder des Anonymisieren von Daten visualisieren und verwaltet dazu zwei verschiedene Datensätze, wobei eine Datenmenge ohne, und eine Datenmenge mit der Verwendung von Methoden zur Verschlüsselung oder des Anonymisieren von Daten erzeugt wird. Der entstehende Unterschied der gesammelten Meta-Daten durch die verschiedenen Methoden führt dann zu einer Veränderung in der Visualisierung was dan den Effekt und Nutzen der Methoden deutlich sichtbar macht.

## 2 Grundlagen

### 2.1 Verwendungszweck des Dokumentenspeichers

Der Dokumentenspeicher dient vor allem dazu den Vergleich zwischen den unterschiedlichen gesammelten Meta-Daten mit und ohne die Verwendung von datenschutzfreundlichen Methoden zum Anonymisieren von Daten. Diese Entstehende Unterschiede werden in der Visualisierung der Daten deutlich und veranschaulichen so den Effekt der Verwendung von datenschutzfreundlichen Methoden zum Anonymisieren von Daten. Die durch datenschutzfreundlichen Methoden zum Anonymisieren von Daten erzeugte Datenmenge wird folgend ~~nur noch~~ als geschützte Datenmenge beschreiben und nimmt an ~~das~~ mindestens eine Methoden zum Anonymisieren von Daten benutzt wurde um diese Datenmenge zu generieren. Die erzeugte Datenmenge ohne die Verwendung von datenschutzfreundlichen Methoden zum Anonymisieren wird folglich ~~nur noch~~ als ungeschützte Datenmenge beschreiben und nimmt an ~~das~~ keine Methoden zum Anonymisieren von Daten benutzt wurde oder der zumindest keine Methoden zum Anonymisieren von Daten verwendet wurde im Aspekt auf die Eigenschaften die in einem gegebenen Szenario untersucht wurden. Der Dokumentenspeicher ist somit so konzipiert ~~das~~ ein geschützter und ein ungeschützter Datensatz angelegt werden kann. Dafür wurden verschiedene API-Punkte implementiert, welche die Trennung der beiden Datensätze ermöglicht. Es können jedoch auch geschützte und ungeschützte Daten in die gleiche Datenmenge eingespeist werden, falls dies für das zu betrachtende Szenario sinnvoll ist. Die folgenden vorgestellten Szenarien trennen jedoch die geschützte Datenmenge und die ungeschützte Datenmenge und verwenden die dafür vorgesehenen API-Punkte. Da die geschützte und ungeschützte Datenmenge verglichen werden soll um den Effekt der zu Untersuchenden Methode zum Anonymisieren von Daten sichtbar zu machen, ist es von Vorteil wenn die verwendeten Daten zum erzeugen der geschützten und ungeschützten Datenmenge dieselben sind und sich lediglich durch die zum Anonymisieren von Daten und deren Effekt unterscheiden. Dies stellt sicher ~~das~~ lediglich der Effekt der Methode zum Anonymisieren einen Unterschied in der Visualisierung erzeugt und keine anderen Faktoren die Visualisierung beeinflussen. Im Ideal Fall können so durch die Betrachtung der Visualisierung der ungeschützten Datenmenge Relationen zwischen Dateien deutlich werden, welche darauf schließen lassen ~~das~~ diese Dateien z.B. von dem gleichen Benutzer stammen. Die Betrachtung der geschützten Datenmenge sollte dann im Ideal Fall zeigen ~~das~~ durch die Verwendung von Methode zum Anonymisieren von Daten diese Relation nicht mehr sichtbar ist und die Methode somit einen ~~für uns~~ sichtbaren Effekt hat.

## 2.2 Terminologie

### 1. HTTP-Header

- a) Teil des Hypertext Transfer Protocol (HTTP)
- b) Headerblock und Headerfeld nach RFC 2616(<https://tools.ietf.org/html/rfc2616>)

### 2. Geo lookup

- a) eine Methodik zur Bestimmung des geografischen Standort einer IP-Adresse
- b) keine eigenen Implementation der Standortbestimmung, sondern Verwendung eines öffentlich zugänglichen Service, aus Zeit und komplexitätsgründen
- c) Patent US7752210B2

### 3. Header Fingerprint

- a) eine Methodik zur Identifikation eines Benutzers anhand der von ihm Verwendeten HTTP Header
- b) Aggregation über allen Http-Header oder einem ausgewählten Set an Headerfeldern
- c) erzeugt Fingerprint wird gehasht und gespeichert
- d) bei übereinstimmenden Fingerprints wird angenommen das diese vom gleichen Benutzer erzeugt wurden

## 3 Hauptteil

### 3.1 Implementation des Dokumentenspeichers

Der Dokumentenspeicher wurde mit Hilfe des **ASP .Net** Core Framework erstellt. Das Framework ist Microsofts aktuellste cross-platform Framework zur Realisierung von **Web-Anwendungen**. Das Framework unterstützt alle gängigen Betriebssysteme wie Windows, Mac OS und Linux. Mit dem ASP .Net Core entwickelte Web-Anwendungen **lassen sich in gängige Hostingplattformen wie, z.B. das IIS von Microsoft, integrieren oder auch in einem eigenen Prozess selbst gehostet werden**. Das ASP .Net Framework sieht dabei eine MVC-Architektur, der Projekte vor und verwenden diese Architektur intern zum Realisieren der Anwendungen. Modelle werden in diesem Kontext als Objekte zur Daten Repräsentation verstanden. Views sind die **HTML Pages** welche die Klienten ausgegeben werden. Controller sind Zentrale Elemente. Sie stellen die Funktionalität von Views auf der Server Seite dar. Sie sind zur Steuerung verschiedener Routen und die Implementation von API-Punkten vorgesehen. Dabei Verwalten die Controller ebenfalls die zu Grunde liegenden Datenbanken.

Der Entwickelte Dokumentenspeicher besteht aus einer Controller-Klasse, welche verschiedene API-Punkte implementiert. Die API-Punkte werden durch Routen angesprochen, welche vorher definiert worden. Die Implementierten Routen sind:

**/api/GetA** HTTP Get Methode welche die Daten des Set A, dem ungeschützten Datensatz zurückgibt

**/api/GetB** HTTP Get Methode welche die Daten des Set B, dem geschützten Datensatz zurückgibt

**/api/uploadA** HTTP Post Methode zum hochladen von ungeschützten Dateien ins Set A

**/api/uploadB** HTTP Post Methode zum hochladen von geschützten Dateien ins Set B

**api/uploadEmu** HTTP Post Methode zum erzeugen von Dummy Daten

In der Controller Klasse ist auch für das Sammeln der Meta-Daten für die hoch geladenen Dateien zuständig und speichert diese mit Hilfe der Modell Klasse in einem Datenbankschema ab. Die Modell Klasse besitzt verschiedene **Properties** welche die gesammelten Meta-Daten widerspiegeln. Die Properties sind :

**ID** Datenbank Index

**Set** Das Set bezeichnet die Gruppe welcher die Datei zugeordnet wurde

**Filename** Der Dateiname

**Filepath** Der Pfad zur gespeicherten temporären Datei

**Size** Die Dateigröße in Byte

**IPAddress** Die IP-Adresse von der die Datei hochgeladen wurde



**Headers** Ein String bestehend aus den Headern der Datei

**HeaderFingerprint** Ein Zusammenschluss aus ausgewählten Headern um eine möglichst eindeutige Signatur zu erzeugen

**DateTime** Als Zeitstempel für das Hochladen der Datei

**Country** Land aus welches die Datei hochgeladen wurde

**RegionName** Region (Bundesland) aus welches die Datei hochgeladen wurde

**City** Stadt aus welches die Datei hochgeladen wurde

**Lat** Breitengrad welcher mit den bekannten IP-Adresse assoziiert wird

**Lon** Längengrad welcher mit den bekannten IP-Adresse assoziiert wird

**Isp** Der Internetanbieter welcher der IP zugeordnet ist

Diese Properties enthalten alle gesammelten Meta-Daten und können zur Visualisierung verwendet werden. Mit Hilfe des Entity Framework Core von Microsoft wird ein Datenbankschema erzeugt was dieser Modell Klasse entspricht. Dabei wird eine SQLite Datenbank Datei erzeugt die dem Datenbankschema entspricht und im Verzeichnis der Anwendung liegt. Im Gegensatz zur Verwendung eines festen SQL-Server der die Datenbank verwaltet, bietet die Anwendung so mehr Flexibilität und erlaubt es mehrere Datenbank Dateien mit der Anwendung bereit zu stellen, um so verschiedene Szenarien in verschiedenen Dateien bereitzustellen.

Der Dokumentenspeicher besitzt 3 Views, welche HTML Pages darstellen welche ein Benutzer über bestimmte Routen aufrufen kann.

**/FileEntry** Anzeige der Datenbank in Tabellarischer Form

**/FileEntryCreate** Bietet Möglichkeiten zum hochladen von Dateien oder dem erzeugen von Dummy Daten

**/FileEntryVisual** Visualisierung der gesammelten Daten

Die FileEntry-Page ist die StartPage der Web-Anwendung und verweist zu der FileEntry-Create und FileEntryVisual Page. Die Page zeigt lediglich die Datenbank in tabellarischer Form und ist hauptsächlich für die Entwicklung gebraucht worden. Dies gilt auch für die FileEntryCreate-Page. Die FileEntryVisual-Page ist der Hauptbestandteil dieser Arbeit und stellt die verschiedenen Visualisierungsmöglichkeiten dar.

## 3.2 Darstellung: IP Tree Map und IP Google Map

Die Darstellung von IP bezogenen Meta-Daten wird an einem einfachen Beispiel vorgeführt. Gegeben sind Benutzer Alice und Bob. Alice und Bob befinden sich beide in Hamburg. Sie laden jeweils drei Dateien hoch, einmal mit und einmal ohne die Verwendung eines Proxys zum Maskieren ihrer IP-Adresse. Der verwendete Proxy ist bei Alice und Bob der selbe. Alice lädt die Dateien test0 bis test2 und Bob die Dateien test3-test5 hoch. Alice lädt ihre drei Dateien in **einem Post** hoch und Bob lädt die drei Dateien jeweils einzeln mit einem Abstand von ein paar Sekunden hoch.



**Abbildung 3.1:** Visualisierung von Dateien mittels einer TreeMap, welche nach der IP-Adresse gruppiert sind

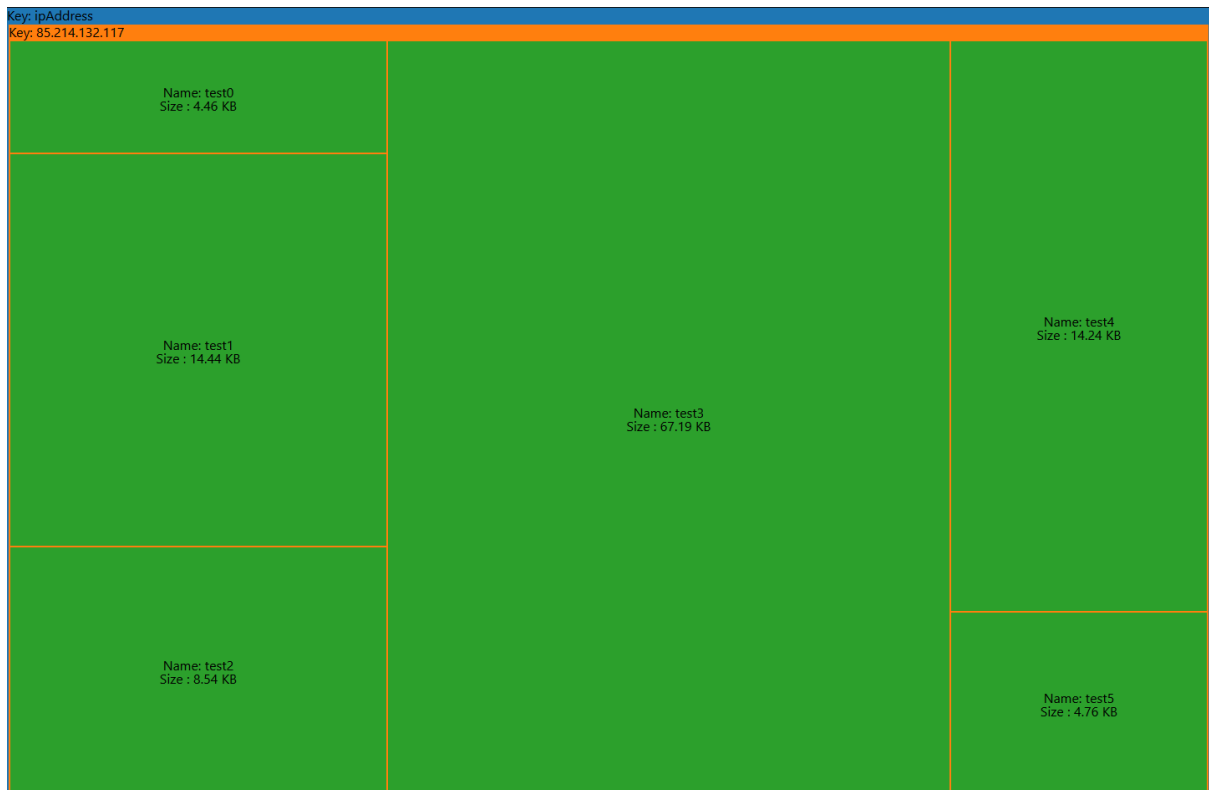
In Abbildung 3.1 werden die Daten von Alice und Bob dargestellt, welche ohne die Verwendung eines Proxys hochgeladen worden sind. Die Darstellung erfolgt durch eine sog. TreeMap, welche eine gegebene Datenstruktur, welche als Baum dargestellt werden kann, visualisiert. Die Baumstruktur besteht aus einer Wurzel-Knoten, sog. Root-Node und den davon abgehenden Child-Nodes, Nodes in der Baumstruktur, welche keine weiteren Child-Nodes besitzen, werden Leaf-Nodes genannt. Die Root-Node wird künstlich erzeugt und als Überschrift für die Visualisierung verwendet und zeigt den Key, über dem die Baumstruktur erzeugt wurde, in diesem Fall die **IP-Adresse**. Die restlichen Nodes sind aus dem gegebenen Daten erzeugt worden, wobei der Key: `ipAddress`, die ausschlaggebende Property ist, über welcher die Baumstruktur erzeugt wird. Die erzeugte Baumstruktur hat somit 3 Ebenen. Auf der 1. Ebene die Root-Node, welche zur Visualisierung der Key-Property benutzt wird. Auf der 2. Ebene die Nodes, welche die IP-Adressen darstellen, welche im Datensatz vorhanden sind. Und auf der 3. Ebene die Leaf-Nodes, welche die Dateien selbst darstellen. In Abbildung 3.1 sind die IP-Adressen **95.91.226.214** von Alice und

95.91.225.215 Bob erkennbar. Zu jeder IP-Adresse können jeweils die drei Dateien der Benutzer zugeordnet werden. Die Dateien test0 bis test2 sind der IP-Adresse 95.91.226.214 zugeordnet und test3 bis test5 sind der IP-Adresse 95.91.225.215 zugeordnet. Damit ist das gegebene Beispiel richtig abgebildet und die echten Relationen zwischen Dateien und IP-Adressen sind richtig modelliert und nachvollziehbar.



Abbildung 3.2: Visualisierung von Dateien mittels einer Karte, welche die Geo-Position einer IP-Adresse anzeigt

Die Abbildung 3.2 zeigt uns die Geo-Position der IP-Adressen von Alice und Bob und zeigt deren ungefähre Position in Hamburg, wobei die Position nur einen ungefähren Standpunkt darstellt und die eigentliche Position bis zu einem Kilometer abweichen kann.



**Abbildung 3.3:** Visualisierung von Dateien mittels einer TreeMap, welche nach der IP-Adresse gruppiert sind

In Abbildung 3.3 sind die Daten visualisiert, welche mittels eines Proxys hochgeladen wurden. Alle Dateien wurden über die IP-Adresse des Proxys 85.214.132.117 gruppiert und somit eine Anonymitätsmenge erzeugt. Die Daten von Alice und Bob sind nun in einer Anonymitätsmenge zusammengefasst und lassen sich anhand der IP-Adresse nicht mehr eindeutig den beiden Benutzern zuordnen.



Abbildung 3.4: Visualisierung von Dateien mittels einer Karte, welche die Geo-Position einer IP-Adresse anzeigt

Abbildung 3.4 zeigt und nun die Geo-Position des Proxys welches in Berlin steht. Damit sind auch die eigentlichen Geo-Position von Alice und Bob durch die Verwendung des Proxys maskiert worden.

Anstelle eines Proxys könnten Alice und Bob das Tor-Netzwerk benutzen um ihre IP-Adressen zu maskieren. Alice und Bob laden drei Dateien, einmal ungeschützt und einmal geschützt durch das Tor-Netzwerk hoch. Dabei nehmen wir an das alle Datei-Uploads einzeln passieren und jedem Dabei-Upload eine andere Route durch das Tor-Netzwerk zugewiesen wird, sodass jeder Upload einen anderen Exit-Server des Tor-Netzwerkes durchläuft.



Abbildung 3.5: Visualisierung von Dateien mittels einer TreeMap, welche nach der IP-Adresse gruppiert sind

In Abbildung 3.5 sehen wir nun die IP-Adressen der verschiedenen Exit-Server des Tor-Netzwerks. Da jedem Datei-Upload ein neues Exit-Server zugewiesen wurde, können wir über die IP-Adresse die Dateien nicht mehr zuordnen.



Abbildung 3.6: Visualisierung von Dateien mittels einer Karte, welche die Geo-Position einer IP-Adresse anzeigt

Bei Betrachtung der Abbildung 3.6 ist auch erkennbar, dass die verschiedenen Positionen der Exit-Server des Tor-Netzwerks variieren und es keinen Anhaltspunkt auf die eigentliche Position von Alice und Bob gibt.

### 3.3 Darstellung: Headerfingerprinting

Der Headerfingerprint welcher beim ~~hochladen~~ der Datei erzeugt wird, wird nun verwendet um die Dateien jeweils

### 3.4 Darstellung: Time Line



## **4 Schluss**

### **4.1 Zusammenfassung der Ergebnisse**

### **4.2 kritische Bewertung des Ergebnisse**

### **4.3 neue Problemstellungen, Möglichkeiten zur Weiterführung der Arbeit**