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 27. 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

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 jederzeit 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 Metadaten wie z.B. die Dateigröße oder den Autor der Datei sowie Verkehrsdaten wie z.B. die IP-Adresse oder die HTTP Header Felder über die Nutzer sammeln und weiterverwenden. Vor allem private Nutzer sind meist gar nicht über die Risiken und das Missbrauchspotenzial aufgeklärt, welche die Verwendung solcher Dienstleistungen mit sich bringen. 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 nicht besitzen. Um genau die Risiken und Missbrauchspotenziale aufzuzeigen wird im Zuge dieser Arbeit ein Dokumentenspeicherdienst entwickelt, welcher Meta- und Verkehrsdaten 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 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 das 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 das anonymisieren von Daten erzeugt wird. Der entstehende Unterschied der gesammelten Metadaten durch die verschiedenen Methoden führt dann zu einer Veränderung in der Visualisierung, was dann den Effekt und Nutzen der Methoden deutlich sichtbar macht.

2 Grundlagen

2.1 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 vor allem zeigen, dass durch die gesammelten Verkehr- und Metadaten es möglich ist Dateien welche vom gleichen Benutzer stammen korrekt einander zuzuordnen und das die Verwendung von datenschutzfreundlichen Methoden zum Anonymisieren von Daten dies verhindern kann. Da die Verwendung von Authentifizierungen durch einen Benutzeraccount oder ähnliches diese Zuordnung trivialisiert, wird für diesen Dokumentenspeicher keine solche Authentifizierungen verwendet. Für den Dokumentenspeicher wird angenommen das alle Dateien von den Benutzern so Verschlüsselt wurden, das nur die Benutzer in der Lage sind sie wieder zu Entschlüsseln. Daher werden alle Dateien durch kryptografische Methoden logisch getrennt in einen gemeinsamen Speicher abgelegt.

mandantenspezifische Verschlüsselung, siehe ...

Um es zu ermöglichen die gesammelten Daten in die zu Untersuchenden Gruppen von mit und ohne Verwendung von datenschutzfreundlichen Methoden zum Anonymisieren von Daten einzuteilen. Bietet der Dokumentenspeicher zwei verschiedene API-Endpunkte an um Dateien hoch zu laden. Die vorgesehene Verwendung sieht also vor das zuerst eine Datenmenge an Testdaten erzeugt wird, anhand welchen ein Szenario mit verschiedenen Benutzern erstellt wird, welche verschiedene Dateien zu verschiedenen Zeitpunkte hochladen sollen. Im zweiten Schritt wird eine oder mehrere datenschutzfreundliche Methoden zum Anonymisieren von Daten gewählt, welche untersucht werden sollen. Im dritten Schritt wird das Szenario einmal mit und ohne die Verwendung der gewählten Methoden zum Anonymisieren ausgeführt. Dabei wird sichergestellt das das durch Methoden zu Anonymisieren geschützte Szenario und das ungeschützte Szenario jeweils einen anderen API-Endpunkt ansprechen. Der Dokumentenspeicher besitzt so zwei verschiedene Datenmengen welche sich durch das wählen oder nicht wählen einer Methoden zum Anonymisieren von Daten unterscheiden.

In dem Visualisierungstool des Dokumentenspeichers kann eine Visualiserungsoption 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.

Der Dokumentenspeicher kann aber auch zur Visualisierung von einer gemischten Datenmenge aus geschützten und ungeschützten Dateien verwendet werden. Dies wird erreicht, indem die ungeschützten und geschützten Dateien lediglich über einen API-Endpunkt hochgeladen werden.

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

ausschließlich diese Unterschiede zu Visualisieren, da die Zugrunde liegenden Dateien sich sonst nicht unterscheiden.

2.2 Implementation des Dokumentenspeichers

2.2.1 Verwendete Technologien

Der Dokumentenspeicher wurde mit Hilfe des ASP.NET Core Framework erstellt. Das Framework ist Microsofts aktuellste 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 Hostingplatformen wie, z.B. das IIS von Microsoft, integrieren oder können 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 Datenrepräsentation verstanden. Views sind die HTML-Seiten welche an die Klienten ausgegeben werden. Controller sind zentrale Elemente. Sie stellen die Funktionalität von Views auf der Serverseite dar. Sie sind zur Steuerung verschiedener Routen und die Implementation von API-endpunkten vorgesehen. Dabei verwalten die Controller ebenfalls die zu Grunde liegenden Datenbanken.

2.2.2 Aufbau des Dokumentenspeichers

Der entwickelt Dokumentenspeicher besteht aus aus einer Controller-Klasse, einer Modell-Klasse und drei verschiedenen View-Klassen. Dazu eine Datenbankkontext-Klasse zur Verwaltung der Datenbank und Synchronisation zwischen Modell-Klasse und der verwendeten Datenbank. Die Controller-Klasse implementiert verschiedene API-Endpunkte, welche das das hochladen von Dateien und die Abfrage der gesammelten Daten ermöglichen.

/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 Dummy Daten

Beim hochladen der Dateien werden in der Kontroller-Klasse die Verkehrs und Metadaten der Dateien erfasst, sowie die hochgeladenen Dateien abgespeichert. Mit Hilfe der Modellklasse werden die gesammelten Daten sowie der Pfad zu der gespeicherten Datei in der Datenbank abgelegt.

Die Modellklasse hält für alle erfassten Verkehr- und Metadaten Eigenschaften, welche diese Repräsentieren.

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 welchem die Datei hochgeladen wurde

RegionName
Region (Bundesland) aus welchem die Datei hochgeladen wurde

City Stadt aus welchem 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

Der Dokumentenspeicher besitzt 3 Views, welche HTML-Seiten darstellen welche ein Benutzer über bestimme Routen aufrufen kann.

/FileEntry
Anzeige der Datenbank in tabellarischer Form

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

/FileEntryVisual
Visualisierung der gesammelten Daten

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

FileEntryPage beschreiben und Funktionen zeigen

3 Hauptteil

3.1 Beipiel: Alice und Bob

Um die Visualisierungen an einem Beispiel zu erläutern wird ein Szenario definiert welches die Benutzung des Dokumentenspeichers und die verwendeten Daten definiert. Dazu werden zwei Benutzer Alice und Bob definiert, welche sich beide in Hambrug befinden und sich in den Verkehrs- und Metadaten unterscheiden, sodass wir sie anhand dieser klar unterscheiden können. Alice und Bob laden jeweils drei hypothetische Dateien mit und ohne die Verwendung einer Methode zum Anonymisieren ihrer Daten hoch. Alice Dateien sind test0 bis test2 und Bobs Dateien sind test3 bis test5 und haben alle unterschiedliche Dateigrößen. Die Verkehrs- und Metadaten für diese Dateien sind so gewählt das sie nicht realitätsfern sind und sich eignen die verschiedenen Visualisierungen daran aufzuzeigen. Es wird angenommen das die Dateien einzeln in einem Abstand von ein paar Sekunden hochgeladen werden. Für die Verwendung einer Methode zum Anonymisieren ihrer Daten betrachten wir im 1. Fall die Verwendung eines Proxy welcher von Alice und Bob benutzt wird und im 2. Fall die Verwendung der Tor-Netzwerks. Bei der Verwendung des Tor-Netzwerks nehmen wir das für jede hochgeladene Datei eine andere Route durch das Tor-Netzwerk gewählt wird, sodass sich die Exit-Server für jede hochgeladenen Datei ändern.

Für die Benutzer nehmen wir folgende Verkehrsdaten an:

Alice

- IP:
  - IP-Adresse: 95.91.226.214
  - Lat: 53.5770988464355 Lon: 10.0190000534058
  - Land: Deutschland Region: Hamburg Stadt: Hamburg
- Headerfigerprint:
  - Accept: text/html, application/xhtml+xml, application/xml;q=0.9,*/*;q=0.8
  - Accept-Encoding: gzip, deflate
  - User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_6) AppleWebKit/601.7.7 (KHTML, like Gecko) Version/9.1.2 Safari/601.7.7

Bob

- IP:
  - IP-Adresse: 95.91.225.215
  - Lat: 53.5830001831055 Lon: 9.98130035400391
  - Land: Deutschland Region: Hamburg Stadt: Hamburg
- Headerfigerprint:
  - Accept: text/html, application/xhtml+xml, application/xml;q=0.9,*/*;q=0.8
  - Accept-Encoding: gzip, deflate
  - User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:60.0) Gecko/20100101 Firefox/60.0

3.1.1 IP-Adressen bezogene Visualisierung

Die IP-Adressen können mit Hilfe des Dokumentenspeichers auf zwei unterschiedliche weisen Visualisiert werden. Einmal die Darstellung durch eine sog. TreeMap, welche eine gegebene Baumstruktur visualisieren kann. Und die Darstellung auf einer Karte, indem die Geo-Position einer IP-Adresse gezeigt wird.

Die Baumstruktur besteht aus einem Wurzelknoten, den davon abgehenden Kindknoten und den Blattknoten, welche dadurch ausgezeichnet sind das sie keine Kindknoten besitzen. Der Wurzelknoten wird künstliche erzeugt und als Überschrift für die Visualisierung verwendet und Zeigt die Schlüsseleigenschaft über dem die Baumstruktur erzeugt wurde, in diesem Fall die IP-Adresse. Die restlichen Knoten sind aus den gegebenen Daten erzeugt worden, sodass alle Dateien mit der gleichen IP-Adresse unter einem Knoten zusammen gefasst werden. Die erzeugte Baumstruktur hat somit 3 Ebenen. Auf der 1. Ebene den Wurzelknoten, welche zum Visualisieren der Schlüsseleigenschaft benutzt wird. Auf der 2. Ebene die Kindknoten, welche die IP-Adressen darstellen, welche im Datensatz vorhanden sind und auf der 3. Ebene die Blattknoten, welche die Dateien selbst darstellen.

In Abbildung 3.1 ist die erzeugte Baumstruktur für die Beispieldaten von Alice und Bob zu sehen. Als Wurzelknoten wird die Schlüsseleigenschaft der IP-Adresse gewählt. Auf der darunter liegenden Ebene sind die IP-Adresse und Alice und Bob aufgeführt. Zu jedem dieser Knoten sind die jeweiligen Dateien der beiden Benutzer als Blattknoten angehängt.

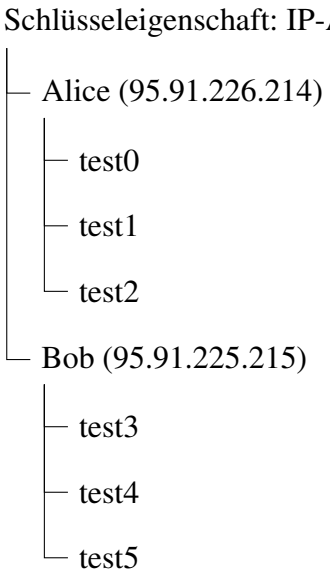

Abbildung 3.1: Baumstruktur erzeugt aus dem ungeschützten Beispieldaten von Alice und Bob

Aus dieser Struktur wird nun die TreeMap erzeugt. Für jeden Knoten im Baum wird ein Rechteck/Box erzeugt, dabei wird jeder Kindknoten in das Rechteck/Box des Elternknoten eingebettet. Durch farbliche Unterschiede sollen dann die verschiedenen Ebenen und Relationen deutlich werden. Eine Schematische Darstellung des Beispiels ist in 3.2 zu sehen.

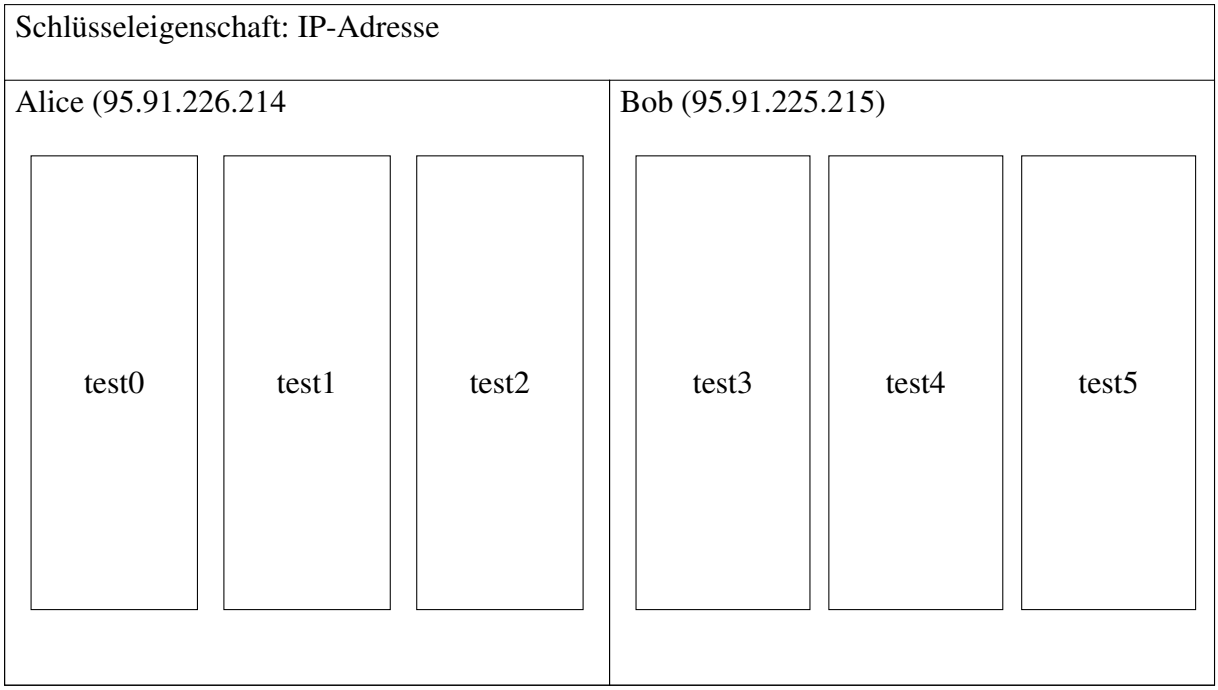

Abbildung 3.2: Schematische Darstellung einer Visualitionmöglichkeit der Baumstruktur

In Abbildung 3.3 und der Abbildung 3.4 werden die Beispieldaten von Alice und Bob mit dem Visualisierungstool des Dokumentenspeichers dargestellt. In Abbildung 3.3 sind die zwei IP-Adressen von Alice(95.91.226.214) und Bob(95.91.225.215) dargestellt. Alice sind die Dateien testo, test1 und test2 zugeordnet. Bob sind die Dateien test3, test4 und test5 zugeordnet. In Abbildung 3.4 sind zwei Standpunkte im Hamburg zu erkennen. Den Standpunkt von Alice (Lat: Lon:) und dem Standpunkt von Bob (Lat: Lon:)

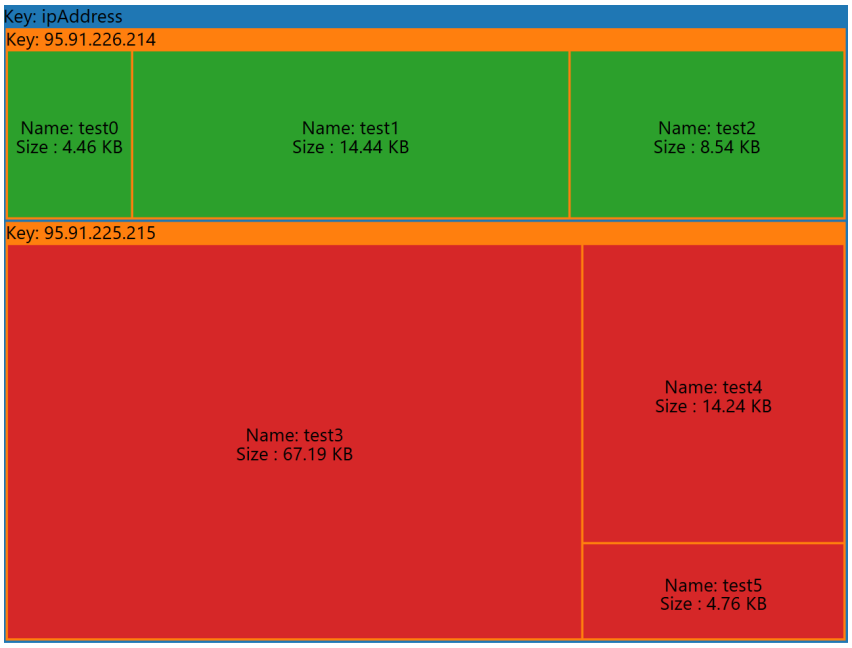

Abbildung 3.3: Darstellung der ungeschützten Beispieldaten von Alice und Bob

Abbildung 3.4: Darstellung der 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 Benutzer zu zuordnen.

Beim Betrachten des ersten Falls, der Verwendung eines Proxys erhalten wir die Visualisierung wie in Abbildung 3.5 und Abbildung 3.6. In Abbildung 3.5 ist nur die IP-Adresse des Proxys dargestellt. Alle definierten Dateien sind dieser IP-Adresse zugeordnet. In Abbildung 3.6 ist ein Standpunkt in Berlin markiert.

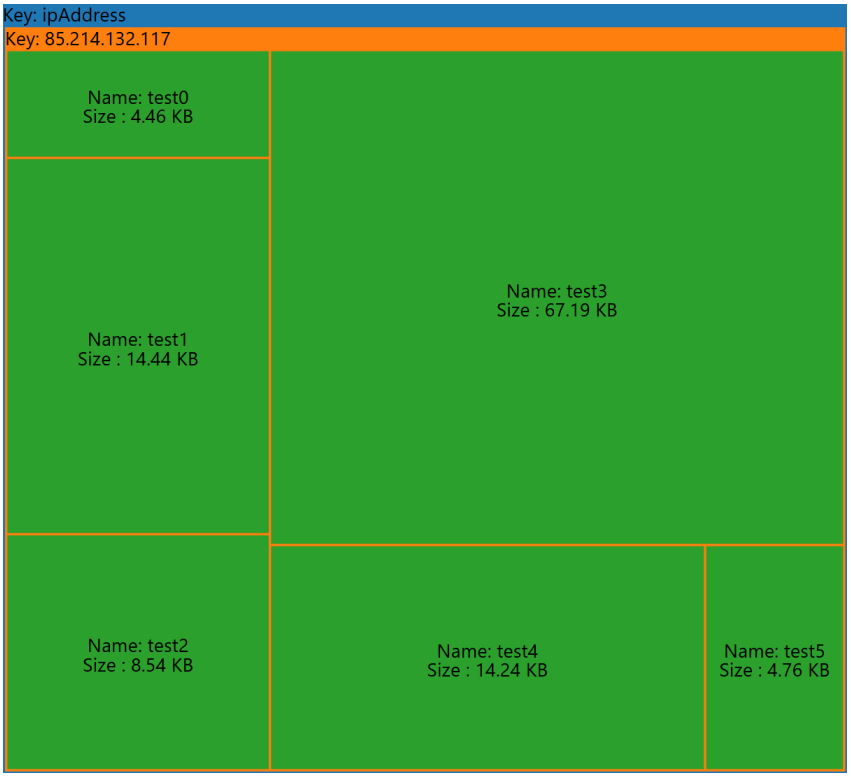

Abbildung 3.5: Darstellung der Beispieldaten von Alice und Bob bei der Verwendung eines Proxys

Abbildung 3.6: Darstellung der Beispieldaten von Alice und Bob bei der Verwendung eines Proxys

Der Effekt der Proxys, welche die IP-Adressen von Alice und Bob maskiert ist in Abbildung 3.5 klar erkennbar. Die Ip-Adressen von Alice und Bob sind nicht erkennbar und alle Dateien der beiden Benutzer der Ip-Adresse des Proxys zugeordnet. Die Dateien von Alice und Bob sind somit durch die entstandene Anonymitätsmenge geschützt, sodass wir annehmen können das die Anonymitätsmenge die Daten der beiden Benutzer enthält, die Daten aber innerhalb der Menge nicht unterschiden werden können und somit Anonymität erreicht haben. Im Vergleich zu den ungeschützten Daten aus der Abbildung 3.3, ist die Fähgkeit anhand der IP-Adresse die Dateien genau den jeweiligen Benutzern zu zuordnen verloren gegangen, dadurch das die Benutzer einen Proxy verwendet haben.

Die entstehender Visualisierungen beim betrachten des zweiten Falls sind in Abbildung 3.7 und Abbildung 3.8 abgebildet.

In der Abbilung 3.7 sind 6 verschiedene IP-Adressen erkennbar. Jeder IP-Adresse sind eine Datei zugeordnet.

81.7.10.29 Datei: test0

129.13.131.140 Datei: test1

2.202.33.28 Datei: test2

81.169.133.228 Datei: test3

85.197.58.203 Datei: test4

178.63.80.54 Datei: test5

Für jede dieser IP-Adressen ist in der Karte in Abbildung 3.8 ein Standort abgebildet, welche über Deutschland verteilt sind. s

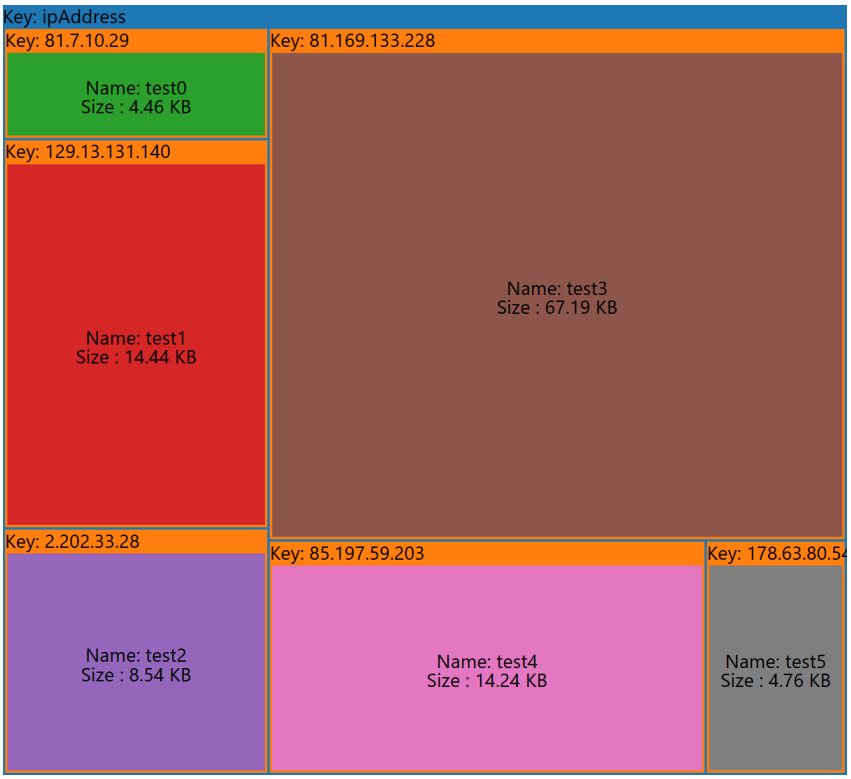

Abbildung 3.7: Darstellung der Beispieldaten von Alice und Bob bei der Verwendung des Tor-Netzwerks

Abbildung 3.8: Darstellung der Beispieldaten von Alice und Bob bei der Verwendung des Tor-Netzwerks

Die Visualisierung des zweiten Falls unterscheidet sich klar von den ungeschützten Beispieldaten sowohl auch von den Beispieldaten mit Verwendung des Proxys als Methode zum Anonymisieren der Daten. Da jeder Datei eine anderen IP-Adresse zugeordnet ist, ist es nicht möglich anhand der IP-Adresse die Dateien den Benutzern Alice und Bob zu zuordnen. Beide Methoden zum Anonymisieren der Daten haben somit ihre Aufgabe erfüllt und lassen eine Zuordnung der Dateien eines Benutzer zu diesem Benutzer nicht mehr zu.

3.1.2 Darstellung: Headerfingerprinting

Der Headerfingerprint wird wie im Kapitel 3.1.1 beschrieben mit einer sog. TreeMap und einer dieser TreeMap zugrundeliegender Baumstruktur. Nur die Schlüsseleigenschaft wird für diese Visualisierung von der IP-Adresse auf den Headerfingerprint geändert. Für die ungeschützten Beispieldaten von Alice und Bob erhalten wir die Visualisierung nach Abbildung 3.9. Zusehen ist der Headerfingerprint von Alice mit zugeordneten Dateien test0, test1 und test2, sowie den Headerfingerprint von Bob mit den zugeordneten Dateien test3, test4 und test5.

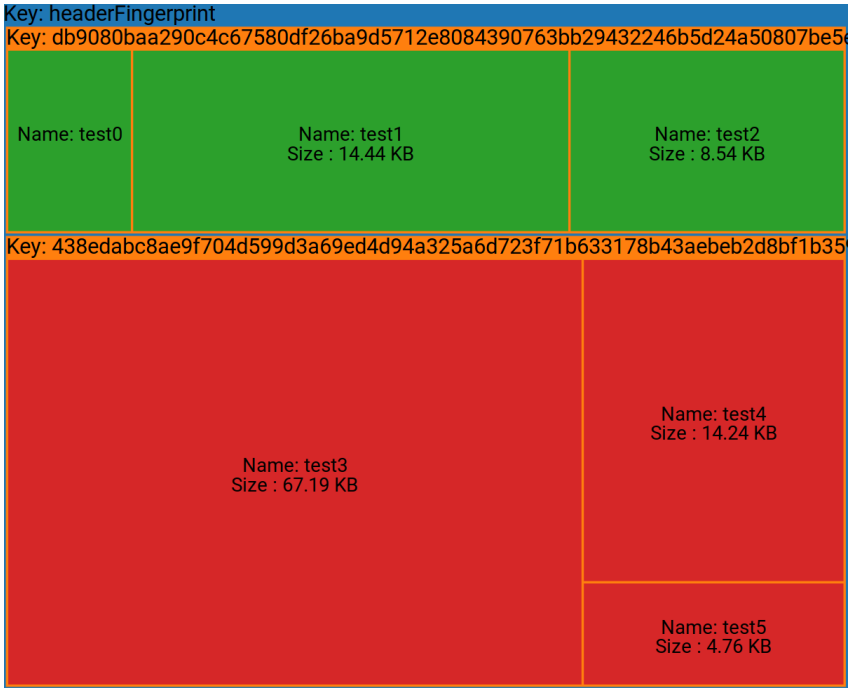

Abbildung 3.9: Darstellung der Beispieldaten von Alice und Bob bei der Verwendung eines Proxy

Wie in Abbildung 3.3 stellt Abbildung 3.9 die Relationen der Beispieldaten und Benutzer richtig dar.

Bei Betrachtung der Visualisierung der Beispieldaten bei der Verwendung eines Proxys erhalten wir die Visualisierung nach Abbildung 3.10. Diese Visualisierung ist mit der Visualisierung der ungeschützten Beispieldaten in Abbildung 3.9 identisch. Der Headerfingerprint von Alice ist dargestellt sowie die Dateien von Alice test0, test1 und test2 sind diesem Headerfingerprint zugeordnet. Der Headerfingerprint von Bob ist ebenfalls dargestellt und die Beispieldateien von Bob test3, test4 und test5 sind diesem zugeordnet.

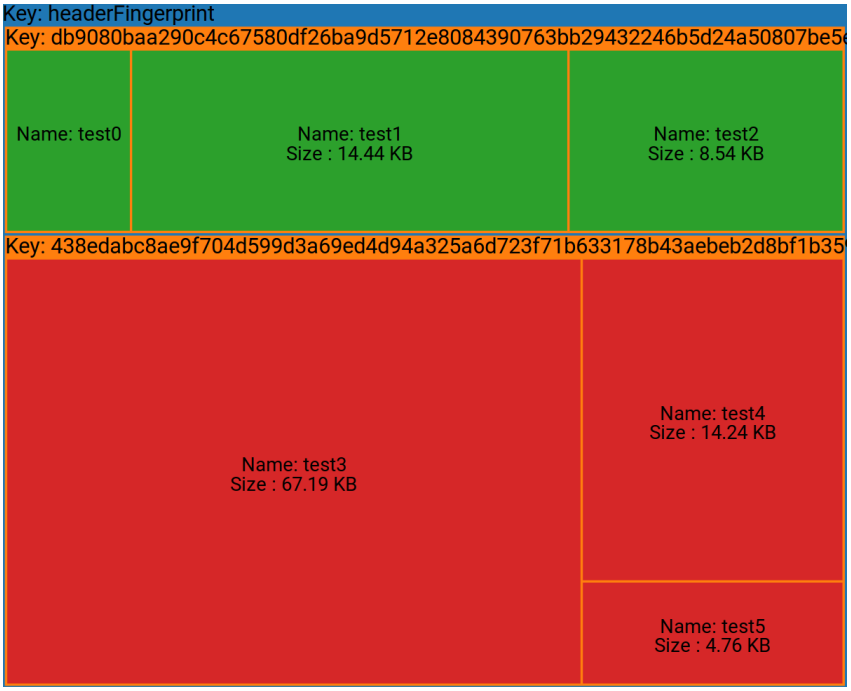

Abbildung 3.10: Darstellung der Beispieldaten von Alice und Bob bei der Verwendung eines Proxy

Bei Betrachtung des zweiten Falls der Verwendung des Tor-Netzwerkes entspricht die Visualisierung der Abbildung 3.11. Zu erkennen ist das ähnlich der Abbildung 3.5 alle Dateien unter einem Headerfingerprint angeordnet sind. Header-Tor-SetB.png

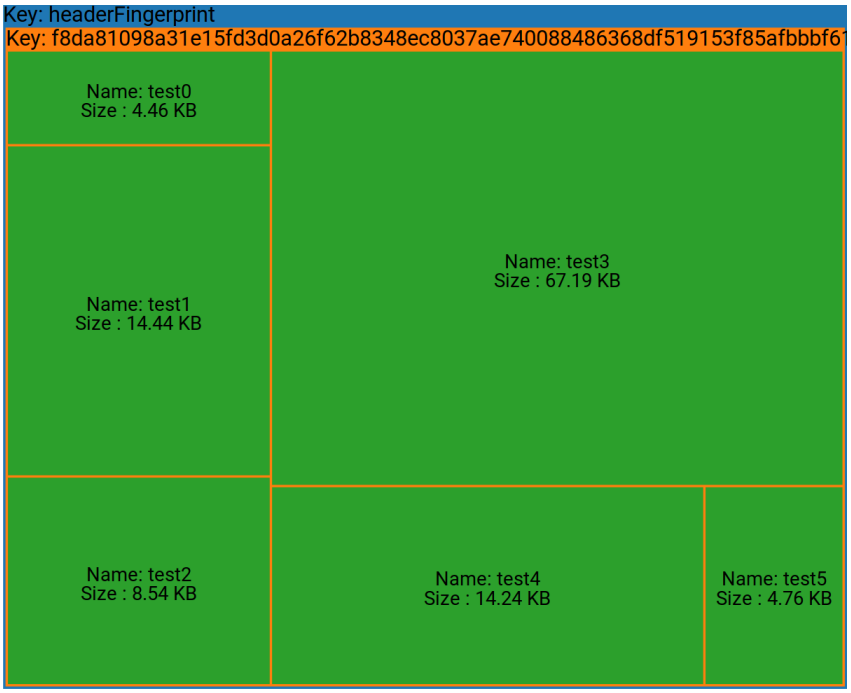

Abbildung 3.11: Darstellung der Beispieldaten von Alice und Bob bei der Verwendung des Tor-Netzwerkes

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

Somit ist klar erkennbar das die Verwendung eins Proxys keine Auswirkung auf die übermittelten Header hatte und Alice und Bob trotz der Verwendung eines Proxys an ihren spezifischen Headern in diesem Beispiel identifiziert werden können. Die Verwendung des Tor-Netzwerks, jedoch erzeugt eine Anonymitätsmenge über dem Headerfingerprint, durch welche die Dateien Anonymisiert werden. Die Verwendung des Proxys zeigt hier somit bereits eine Schwäche gegenüber der Verwendung der Tor-Netzwerks auf, sie nur Schutz vor der Zuordnung der Dateien im Hinblick auf die IP-Adresse gibt. Nicht jedoch im Hinblick auf den Headerfingerprint. Die Verwendung des Tor-Netzwerks schützt sowohl vor der Erkennnung der Relationen von Benutzern und Dateien im Bezug auf die IP-Adresse sowohl auch der des Headerfingerprint.

4 Schluss

4.1 Zusammenfassung der Ergebnisse

4.2 Limitierungen

4.3 Ausblick