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 18. 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 Dokumentenspeicher dient vor allem dazu den Vergleich zwischen den unterschiedlichen gesammelten Metadaten mit und ohne die Verwendung von datenschutzfreundlichen Methoden zum Anonymisieren von Daten. Diese entstehenden 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 als geschützte Datenmenge beschrieben und nimmt an, dass 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 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, dass ein geschützter und ein ungeschützter Datensatz angelegt werden kann. Dafür wurden verschiedene API-Endpunkt 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 Anonymisierung der Daten und deren Effekt unterscheiden. Dies stellt sicher, dass 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, dass diese Dateien z.B. von dem gleichen Benutzer stammen. Die Betrachtung der geschützten Datenmenge sollte dann im ideal Fall zeigen, dass durch die Verwendung von einer oder mehreren Methoden zum Anonymisieren von Daten diese Relation nicht mehr sichtbar ist und die Methode somit einen 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 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 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 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-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 Controllerklasse ist auch für das Sammeln der Meta-Daten für die hochgeladenen Dateien zuständig und speichert diese mit Hilfe der Modellklasse in einem Datenbankschema ab. Die Modellklasse besitzt verschiedene Eigenschaften, welche die gesammelten Metadaten widerspiegeln. Die Eigenschaften 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 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

Diese Eigenschaften enthalten alle gesammelten Metadaten und können zur Visualisierung verwendet werden. Mit Hilfe des Entity Framework Core von Microsoft wird ein Datenbankschema erzeugt, was dieser Modellklasse entspricht. Dabei wird eine SQLite Datenbankdatei 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 Datenbankdateien mit der Anwendung bereit zu stellen, um so verschiedene Szenarien in verschiedenen Dateien bereitzustellen.

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

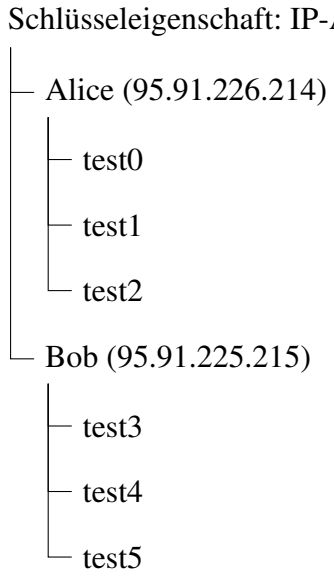
/FileEntryVisual Visualisierung der gesammelten Daten

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.

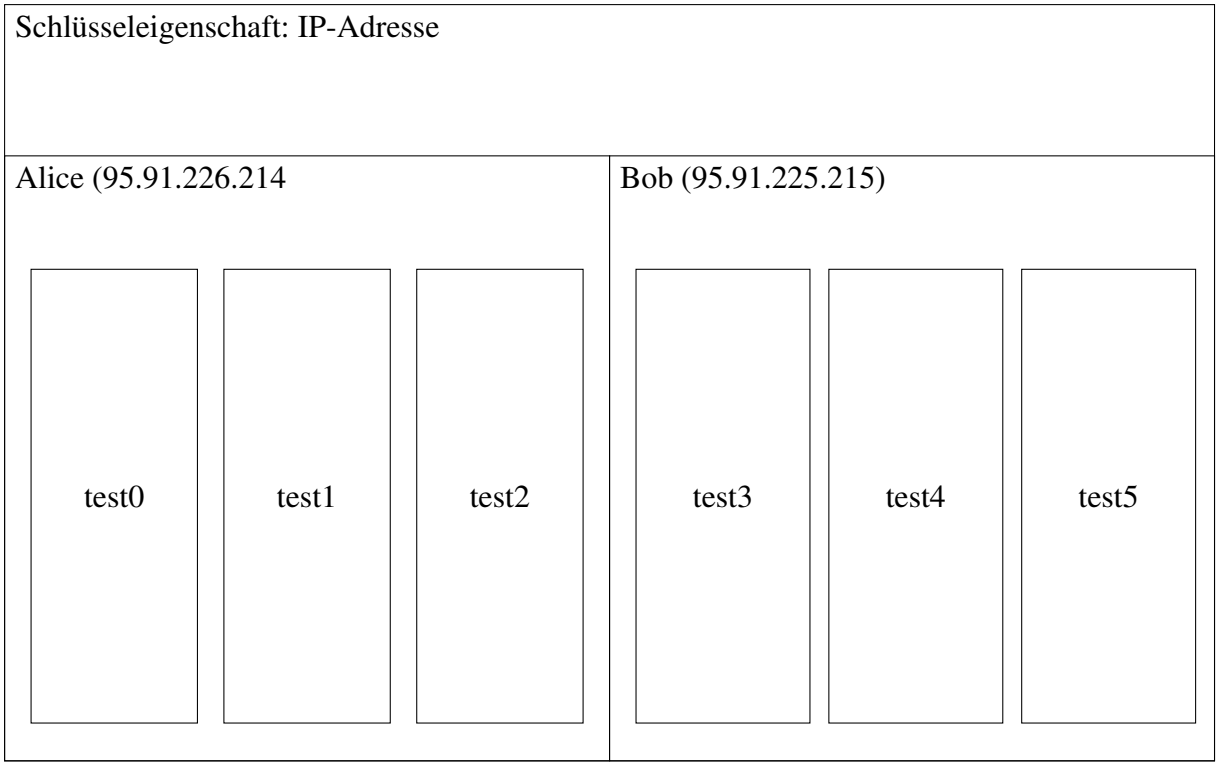
3.2 Darstellung: IP Tree Map und IP Google Map

Die Darstellung von IP bezogenen Metadaten 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 bis test5 hoch. Alice und Bob laden ihre Dateien jeweils einzeln mit einer Verzögerung von ein paar Sekunden einzeln hoch.

Die Darstellung erfolgt durch eine sog. TreeMap, welche eine gegebene Datenstruktur, welche als Baum dargestellt werden kann, visualisiert. 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 „IpAddress“. Die restlichen Knoten sind aus den gegebenen Daten erzeugt worden, wobei der Schlüssel: „IpAddress“, die Ausschlag gebende Eigenschaft ist, über welcher die Baumstruktur erzeugt wird. 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. Für das Beispiel von Alice und Bob sieht die Baumstruktur also wie folgt aus:



schreiben das dies die zu erwartenden Daten sind



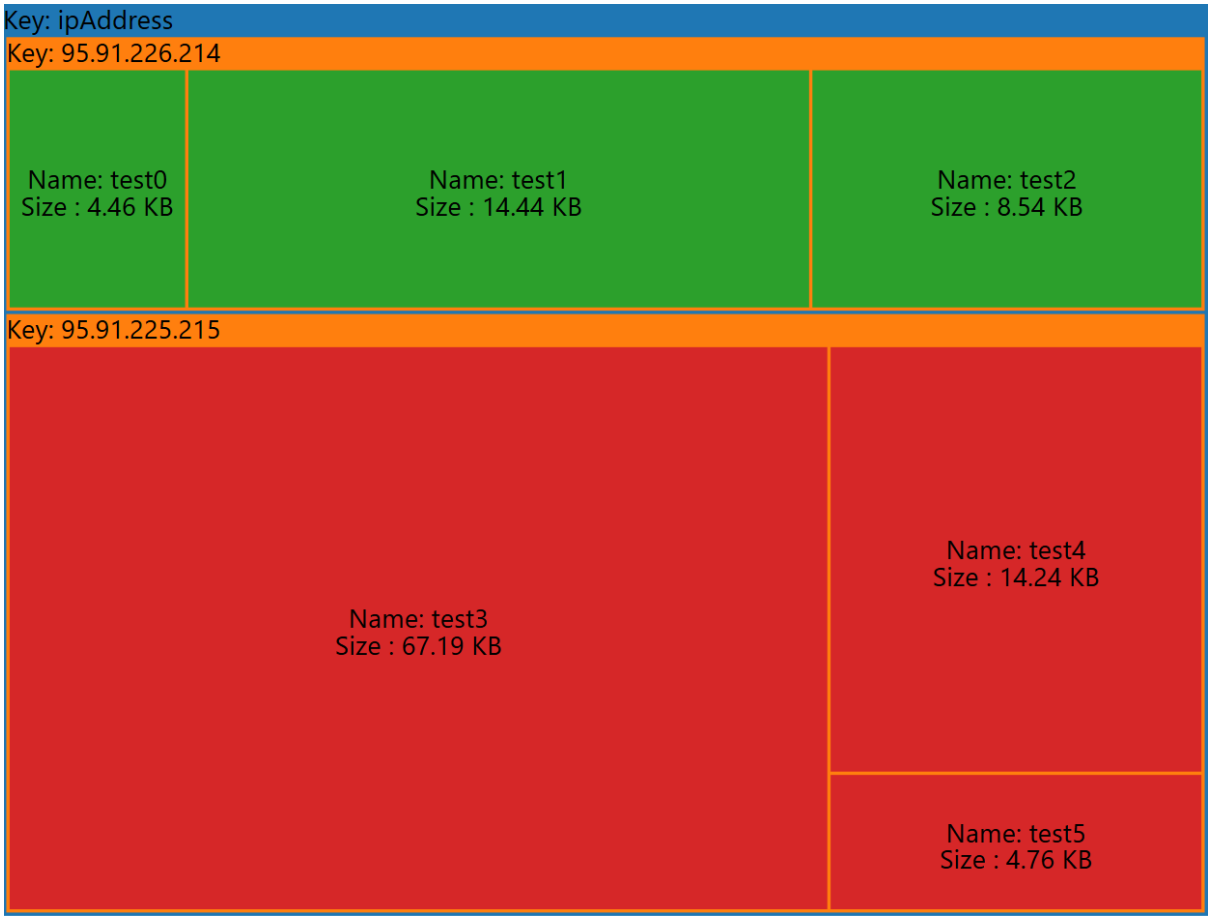


Abbildung 3.1: Darstellung der Daten von Alice und Bob

In Abbildung 3.1 werden die Daten von Alice und Bob dargestellt, welche ohne die Verwendung eines Proxys hochgeladen worden sind.

Move this to the end of the text block

In Abbildung 3.1 sind die IP-Adressen von Alice(95.91.226.214) und Bob(95.91.225.215) 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 Geoposition einer IP-Adresse anzeigt

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

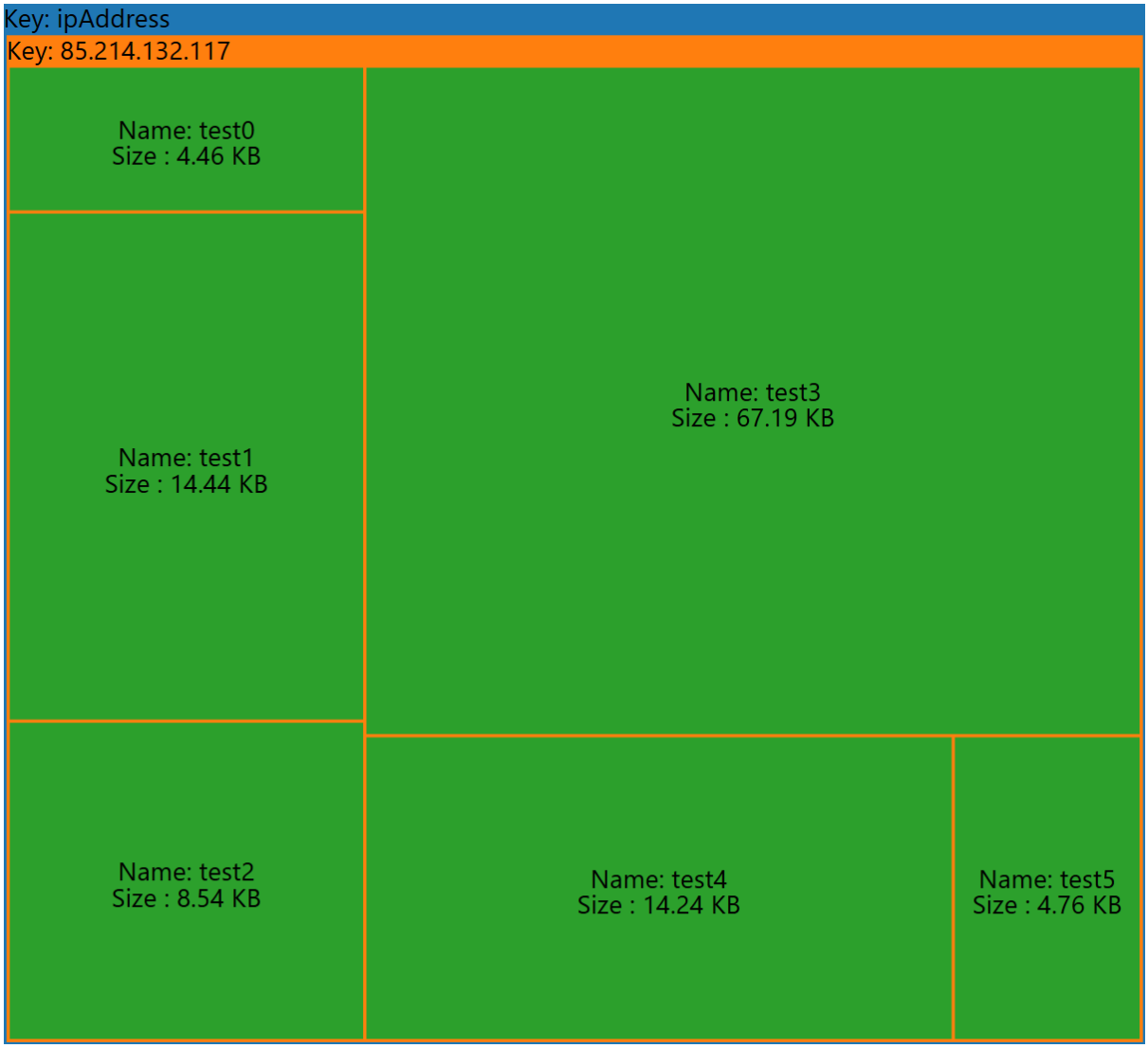


Abbildung 3.3: Visualisierung von Dateien mittels einer TreeMap, welche nach der IP-Adresse gruppiert sind

In Abbildung 3.3 sind die Daten visualisiert, welche mittels eines proxy 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.

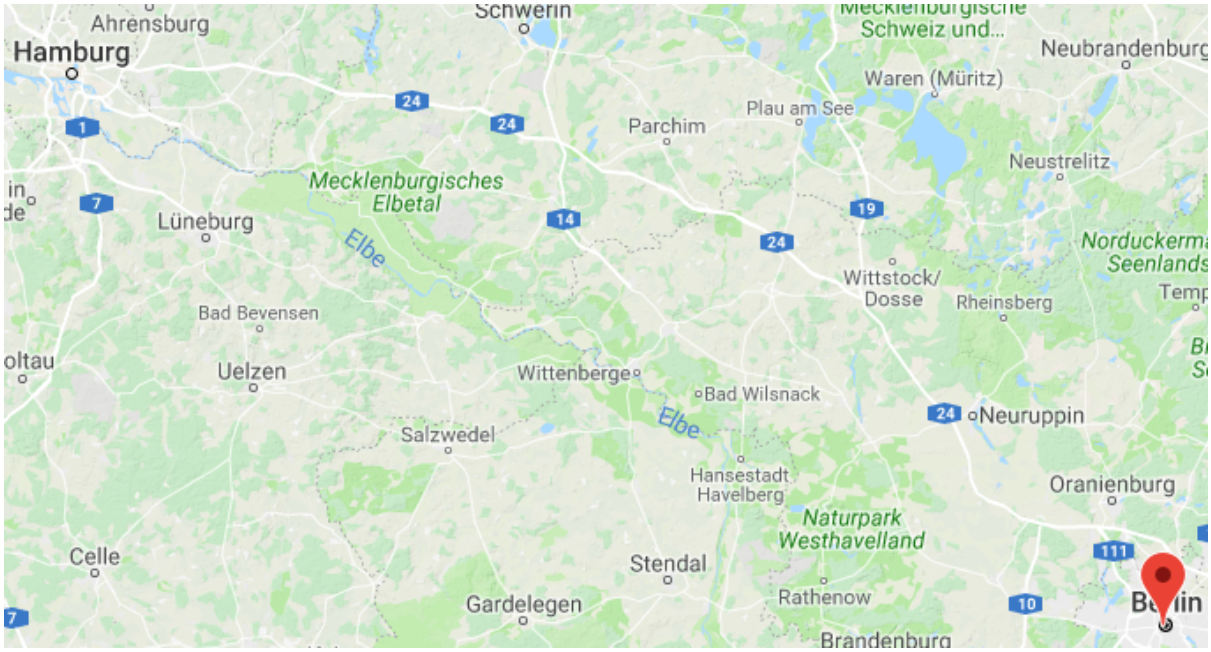


Abbildung 3.4: Visualisierung von Dateien mittels einer Karte, welche die Geoposition einer IP-Adresse anzeigt

Abbildung 3.4 zeigt und nun die Geoposition des proxys welches in Berlin steht. Damit sind auch die eigentlichen Geoposition 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 Dateien einzeln hochgeladen wurden und jeder hochgeladenen Datei eine andere Route durch das Tor-Netzwerk zugewiesen wurde, sodass jede hochgeladene Dateie einen anderen Exit-Server des Tor-Netzwerkes durchläuft.

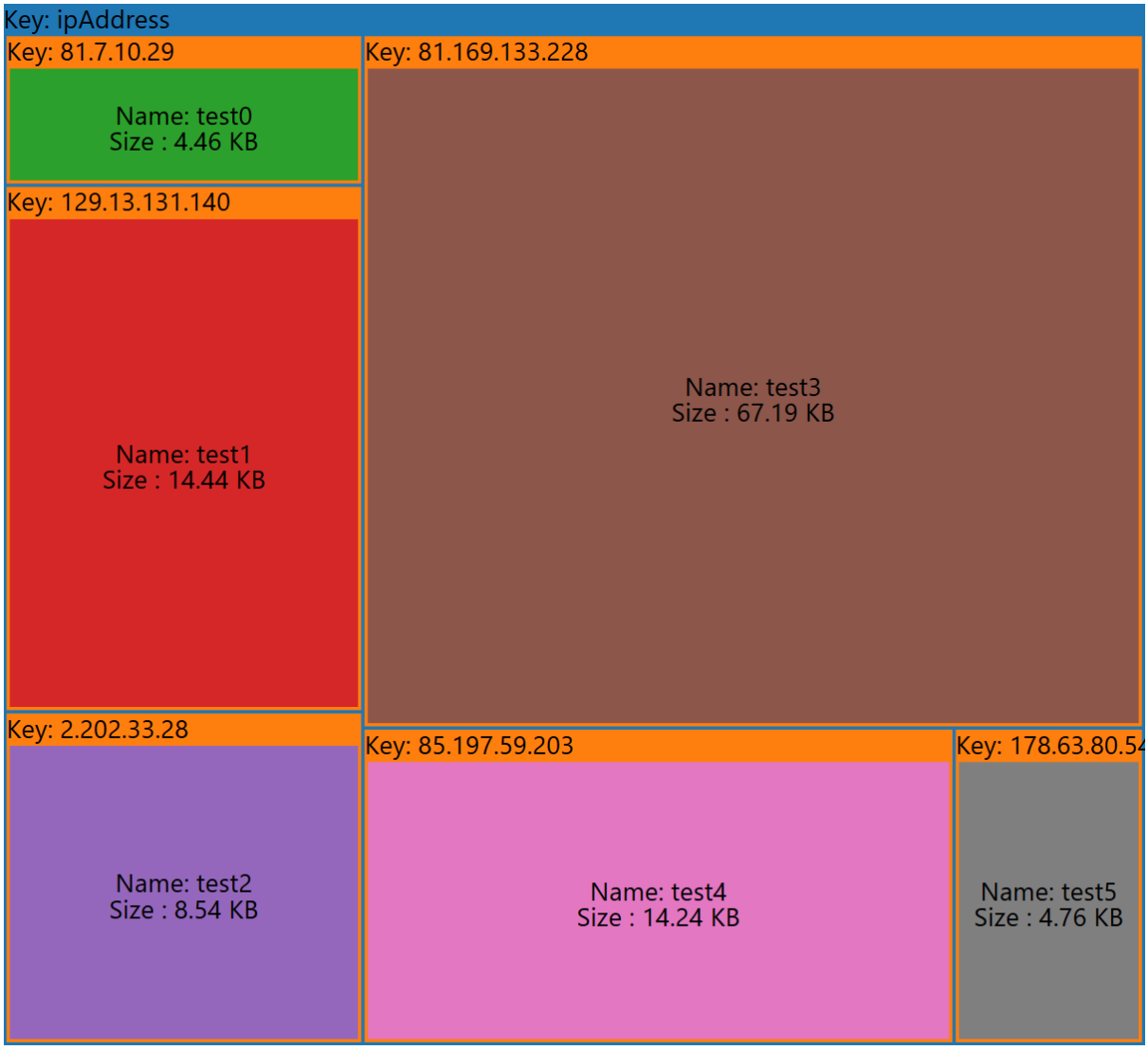


Abbildung 3.5: Visualisierung von Dateien mittels einer TreeMap, welche nach der IP-Adresse gruppiert sind

In Abbildung 3.5 sehen wir nun die IP-Adressen der verschiedenen Exit-Server des Tor-Netzwerks. Da jeder hochgeladener Datei ein neuer Exit-Server zugewiesen wurde, können wir über die IP-Adresse die Dateien nicht mehr entsprechend der Benutzer gruppieren.

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

Bei Betrachtung der Abbildung 3.6 ist auch erkennbar, das die verschiedenen Positionen der Exit-Server des Tor-Netzwerks variieren und es keinen Anhaltspunkt auf die eigentliche Postion 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 Darstelung: Time Line

4 Schluss

4.1 Zusammenfassung der Ergebnisse

4.2 kritische Bewertung des Ergebnisse

4.3 neue Problemstellungen, Möglichkeiten zur Weiterführung der Arbeit