UNIVERSITÄT DUISBURG ESSEN

Offen im Denken

PALUNO
The Ruhr Institute for Software Technology

paluno - The Ruhr Institute for Software Technology
Institut für Informatik und Wirtschaftsinformatik
Universität Duisburg-Essen

Software Systems Engineering
Prof. Dr. Klaus Pohl

Bachelorarbeit

Nutzung realer Mobilitätsdaten in der Simulation von Location-Privacy-Angriffen im Fog-Computing

Vorgelegt der Fakultät für Wirtschaftswissenschaften
der Universität Duisburg-Essen (Campus Essen) von

Lukas Spiekermann
Bülowstraße 51
45479 Mülheim an der Ruhr
Matrikelnummer: 3066731

Mülheim an der Ruhr, den 9. September 2021

Betreuung:	Theresa Wettig und Dr. Zoltán Mann
Erstgutachter:	Prof. Dr. Klaus Pohl
Zweitgutachter:	Prof. Dr. Michael Goedicke
Studiengang:	Wirtschaftsinformatik (B. Sc.)
Semester:	8

Abstract

In Fog systems, computing resources are offered between end devices and cloud servers in the form of Fog nodes. These Fog Nodes are characterised above all by their heterogeneity and geographical distribution. They offer a variety of possible services, such as storage close to end devices, and thus enable real-time applications that require low latency. However, the heterogeneity of Fog nodes could also give rise to risks. For example, Fog nodes whose computing power is not sufficient to ensure adequate security of the sensitive data they process could pose a threat. Since Fog nodes often process the data of nearby users, location-privacy risks may arise in Fog systems for these users. For instance, attackers controlling a Fog node could approximate the locations of users which are currently communicating with that Fog node. Attackers who control multiple Fog nodes could even trace the trajectories of users. The simulator LocPrivFogSim was developed in a previous work to simulate different scenarios that involve such location-privacy risks. The simulation results from that previous work show that even a small number of compromised Fog nodes can be sufficient to accurately determine both the location and trajectory of a user. However, one shortcoming of the test system used for these simulations is that randomly generated paths are used for the simulations instead of real user paths. But these randomly generated paths could have influenced the simulation in such a way that the location privacy risk was misjudged. Thus, a repetition of the simulation with real mobility data seems appropiate in order to be able to discuss this effect. The aim of this work is therefore to extend LocPrivFogSim in such a way that real mobility data can be used for simulation. For this purpose, the GeoLife dataset collected by Microsoft Research Asia is integrated into LocPrivFogSim. The simulation results conducted with this extension will then be evaluated and compared to the findings of *T. Wettig* and *Z.Á. Mann*.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

1.1 Motivation

In Zeiten des Internet of Things gilt es die stetig steigende Menge an, von Endgeräten generierten, Daten effizient zu verarbeiten und zu verwalten. Ein bewährtes Konzept, um mit derartigen Datenmengen umzugehen, ist das Cloud-Computing [12]. Im Cloud-Computing werden Ressourcen, wie Rechenkapazitäten und Speicherplatz, in einem zentralisiert Netzwerk angeboten.

Es zeichnet sich jedoch ab, dass aktuelle Cloud-Systeme möglicherweise bei der zukünftig zu verarbeiteten Datenmengen an ihre Grenzen kommen könnten [21]. Dies ist primär darin begründet, dass die Bandbreite, die für die Übermittlung dieser Datenmengen benötigt wird, begrenzt ist [39]. Eine zweite Einschränkung aktueller Cloud-Systeme besteht darin, dass die zentralisierte Architektur dieser Systeme zu hohen Latenzzeiten führen kann [16]. Aufgrund dieser hohen Latenzzeiten und weiterer Mängel (z.B. fehlendes Standortbewusstsein) könnten aktuelle Cloud-Systeme den Anforderungen von Echtzeitanwendungen nicht gerecht werden.

Eine Möglichkeit, um diese Einschränkungen aktueller Cloud-Systeme zu adressieren, stellt das Konzept des Fog-Computing [16] dar. In Fog-Systemen werden Ressourcen am Rand des Netzwerkes, und somit in der Nähe der Endgeräte mit geringer Latenzzeit bereitgestellt [16, 35]. Dies wird ermöglicht, indem verschiedenste geografisch verteilte Geräte, sogenannte Fog-Knoten, zwischen Cloud und Endgeräten in das Netzwerk eingebracht werden. Fog-Knoten bieten beispielsweise Dienste zur Datenspeicherung, Datenvorverarbeitung und zur Orchestrierung von Datenflüssen an.

In diesen heterogenen Fog-Knoten werden auch potenziell sensible Daten verarbeitet und gespeichert [19]. Zu diesen sensiblen Daten gehören auch standortbezogene Daten. Ist es Dritten möglich auf jene standortbezogenen Daten einer Person zuzugreifen, stellt dies ein Risiko für die Location-Privacy dar [14]. Die Adressierung derartiger Location-Privacy-Risiken stellt eine Herausforderung im Fog-Computing dar. Dritte Parteien könnten beispielsweise über den bekannten Standort eines kompromittierten Fog-Knoten darauf schließen, dass sich ein verbundenes Gerät in dessen Nähe befinden müsste.

Um eben solche Risiken für Location-Privacy im Fog-Computing zu untersuchen, wurde der Simulator LocPrivFogSim in einer vorherigen Arbeit entwickelt [33]. Bislang können mit LocPrivFogSim Angriffe auf Location-Privacy, auf Basis von eigens für die Simulation zufällig generierten Bewegungspfaden durch ein Fog-System, simuliert und ausgewertet werden. Zur Auswertung wurden hier von LocPrivFogSim berechnete Metriken genutzt, welche das Wissen eines Angreifers, und damit die Gefährdung der Location-Privacy quantifizieren. Für den Experimentaufbau aus [33] ergab sich, dass das Wissen eines Angreifers bereits bei einer geringen Anzahl an kompromittierten Fog-Knoten ausreichte, um präzise Angaben über den Bewegungspfad eines Endgerätes anzustellen.

Es ist jedoch fraglich, ob dieses düstere Bild tatsächlich realistisch ist oder ob es der Nutzung der zufällig generierten Pfade geschuldet ist. Daher erscheint es sinnvoll, den LocPrivFogSim Simulator derart zu erweitern, dass reale Mobilitätsdaten in den Simulationen genutzt werden können. Auf Basis der, mit diesen realen Mobilitätsdaten gewonnenen, Simulationsergebnissen, kann daraufhin diskutiert werden, inwiefern sich diese mit den Ergebnissen aus [33] decken oder unterscheiden. Falls sich die Ergebnisse klar unterscheiden, müsste diskutiert werden, auf welche Eigenschaften der realen Pfade sich diese Unterschiede zurückführen lassen könnten.

1.2 Zielsetzung und Vorgehensweise

Das Ziel dieser Arbeit ist es, mit dem Simulator LocPrivFogSim [33] Location-Privacy-Angriffe anhand realer Mobilitätsdaten zu simulieren. Die hierzu benötigten Mobilitätsdaten entstammen dem GeoLife-Datensatz [40, 41, 42], den es gilt aufzubereiten und daraufhin in LocPrivFogSim einzubinden. Zur Aufbereitung des Datensatzes gehört es, anomale Daten (z.B. syntaktische Fehler innerhalb der Dateien) herauszufiltern und die gefilterten Daten in eine für die weitere Nutzung sinnvolle Form (z.B. in

einer Datenbank) zu überführen. Um die Simulation mit diesem Datensatz durchführen zu können muss LocPrivFogsim erweitert und angepasst werden. So müssen beispielsweise Funktionen und Typen derart angepasst werden, dass sie zu dem Format der GPS-Koordinaten und den variierenden Pfadlängen passen. Nach erfolgreicher Implementierung der Erweiterungen und Anpassungen werden die Simulationen aus [33] mit den nun eingebundenen realen Mobilitätsdaten wiederholt. Die Simulationsergebnisse werden daraufhin mit denen aus [33] verglichen. Abschließend werden die Ergebnisse der Arbeit diskutiert. Hier steht primär der Erkenntnissgewinn, über den Einfluss realer Mobilitätsdaten auf die Simulation von Location-Privacy-Angriffen mit LocPrivFogSim, im Fokus.

1.3 Aufbau der Arbeit

In Kapitel 2 werden die, für das weitere Verständnis dieser Arbeit nötigen, grundlegenden Begriffe eingeführt. Hier wird das, dieser Arbeit zugrunde gelegte, Verständnis über das Konzept des Fog-Computing definiert. Ebenso werden die Begriffe Privacy und insbesondere Location-Privacy erläutert und in den Kontext des Fog-Computing gesetzt. Zusätzlich werden Angriffsmöglichkeiten auf Location-Privacy im Fog-Computing dargelegt. Darüber hinaus werden die aus der Vorarbeit von [33] stammenden betrachteten Angriffsszenarien, Annahmen und Metriken zur Quantifizierung der Gefährdung von Location-Privacy ausgeführt. Abschließend werden die zwei technischen Grundlagen dieser Arbeit, der Simulator LocPrivFogSim und der GeoLife-Datensatz, dargestellt.

Kapitel 3 beschreibt die durchzuführenden Erweiterungen und Anpassungen des Simulators LocPrivFogSim, sodass die Simulation mittels realer Mobilitätsdaten möglich ist. Hier wird der Fokus auf dem Software Design und der Implementierung der Erweiterungen und Änderungen liegen. In Kapitel 4 wird die Simulation beschrieben. So werden neben dem gewählten Testsystem, die Simulationsdurchführung, die Simulationsergebnisse beschrieben und daraufhin der Vergleich der gewonnen Simulationsergebnisse zu den Ergebnissen aus [33] angestellt. Daraufhin werden in Kapitel 5 die Ergebnisse der Arbeit diskutiert, wobei Validitätsrisiken erläutert werden. Abschließend wird ein Fazit über die gewonnenen Erkenntnisse gezogen.

2 Grundlagen

Folgend werden die, für das Verständnis der weiteren Arbeit nötigen, Grundlagen ausgeführt. Dazu wird erst das Konzept des Fog-Computing erläutert. Daraufhin werden, aufbauend auf dem Begriff der Privacy, Risiken für Privacy inbesondere für Location-Privacy in Fog-Computing-Systemen dargestellt. Dafür werden mögliche Angriffsarten, die dieser Arbeit zugrunde gelegten Angriffsszenarien und Metriken zur Messung von Location-Privacy Risiken erläutert. Darüber hinaus wird der Fog-Computing-Simulator LocPrivFogSim, der in dieser Arbeit erweitert wird, ausgeführt. Abschließend wird der GeoLife Datensatz dargestellt, welcher in LocPrivFogSim eingebunden wird, um so Location-Privacy Risiken auf Basis von realen Mobilitätsdaten zu simulieren.

2.1 Fog-Computing

Das Konzept des Fog-Computings wird in der Literatur nicht einheitlich definiert. Dies ist vor allem darin begründet, dass Fog-Computing in vielen Aspekten Schnittmengen zu einer Vielzahl ähnlicher Konzepte wie beispielsweise dem Edge-Computing hat, wodurch die Findung einer, zu diesen ähnlichen Konzepten trennscharfe, Definition des Fog-Computing erschwert ist [39]. Gemein ist den meisten Definitionen jedoch, dass das Ziel des Fog-Computing darin besteht, die Lücke zwischen der zentralen Cloud und Endgeräten im Layer des Internet of Things, folgend IoT-Layer genannt, zu schließen. Dieses Ziel wird primär durch zwei abzusehende Einschränkungen aktueller Cloud-Computing Systeme motiviert:

1. Die für Cloud-Computing Systemen relevante Bandbreite des Netzwerks ist begrenzt. Dadurch besteht die Gefahr, dass diese Systeme nicht ausreichend skalierbar sind, um zukünftige von Endgeräten und im IoT generierte Datenmengen adäquat verarbeiten zu können [39].

2. Für viele Echtzeitanwendungen im IoT bedarf es minimaler Latenzzeiten und Situations- bzw. Standortbewusstsein. Diesen Anforderungen werden aktuelle Cloud-Computing Systeme meist nicht gerecht [16].

Neben diesen technischen Einschränkungen des Cloud-Computing stellen auch immer häufiger auftretende Datenschutzbedenken seitens Nutzern, aufgrund der zentralen Speicherung ihrer Daten, eine Motivation dar [31].

Um die Lücke zwischen der Cloud und den Endgeräten zu schließen, werden die Ressourcen potenziell sämtlicher Geräte, die sich am Rande des Netzwerkes befinden genutzt, um unterschiedlichste Aufgaben, wie die Datenvorverarbeitung oder der Orchestrierung von Datenflüssen, durchzuführen. Diese Geräte, folgend Fog-Knoten genannt, können verschiedenster Natur sein (mobile Geräte, Router, Server etc.) und bilden nun wie in Abbildung 1 gezeigt den Fog-Layer. Da mithilfe dieses Fog-Layers Rechenleistung, Speicher und Netzwerkdienste außerhalb der Cloud angeboten werden, und so typische Aufgaben der Cloud ausgelagert werden können, liegt der Schluss nahe, Fog-Computing könne als reine Erweiterung des Cloud-Computing, mit dem Zweck Ressourcen der Cloud effizienter nutzen zu können,definiert werden. Eine solche Definition würde jedoch auch auf weitere, dem Fog-Computing ähnliche, Konzepte zutreffen. Daher wird dieser Arbeit die Definition nach Vaquero und Rodero-Merino zugrunde gelegt, welche den Fokus auf die Kommunikation und Kooperation zwischen verschiedenartigsten Fog-Knoten zur Verarbeitung und Speicherung von Daten legt:

> „Fog computing is a scenario where a huge number of heterogeneous (wireless and sometimes autonomous) ubiquitous and decentralised devices communicate and potentially cooperate among them and with the network to perform storage and processing tasks without the intervention of third-parties. These tasks can be for supporting basic network functions or new services and applications that run in a sandboxed environment. Users leasing part of their devices to host these services get incentives for doing so.“[31]

Wie in Abbildung 1 dargestellt, bestehen typische Fog-Systeme aus den drei Schichten: Cloud-Layer, Fog-Layer und dem IoT-Layer - wobei die Bezeichnungen dieser Schichten in der Literatur variieren. Der IoT-Layer beinhaltet Endnutzer und IoT-Geräte. Diese IoT-Geräte sind mitunter mobile Geräte, die

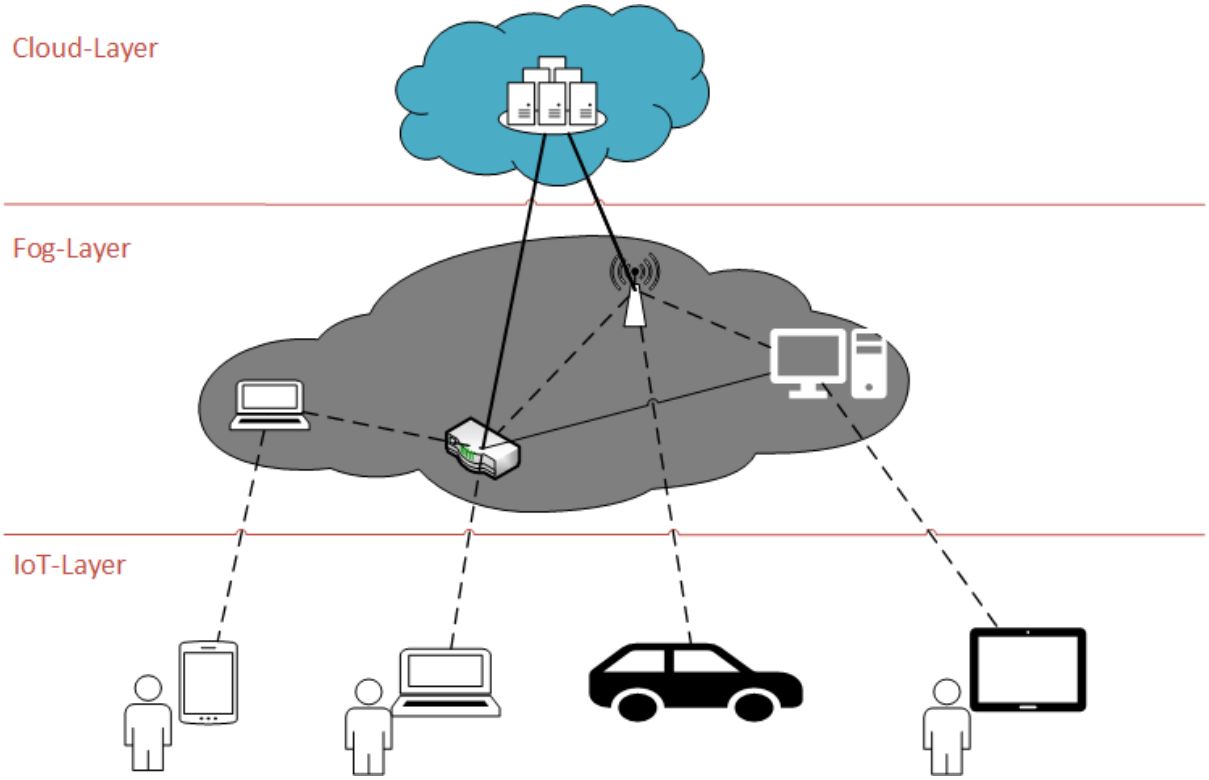


Abbildung 1: Beispielhafte Architektur eines Fog-Computing-Systems

mittels Sensoren Daten aufnehmen und teilweise bereits vorverarbeiten, um diese daraufhin vorzüglich über kabellose Netzwerkverbindungen an Fog-Knoten weiterzuleiten [27]. Im Fog-Layer befindet sich eine Vielzahl dezentral positionierter Fog-Knoten. Jene heterogenen Fog-Knoten bieten Ressourcen zur Datenverarbeitung, Datenspeicherung und Netzwerkaufgaben (z.B. Routing) und entlasten so die Cloud [27]. Insbesondere wird so die benötigte Bandbreite reduziert. Ebenfalls sind diese Ressourcen für Geräte des IoT-Layer mit geringen Latenzzeiten zugänglich. Geografische Verteilung der Fog-Knoten und die geringen Latenzzeiten ermöglichen so Echtzeit-Anwendungen. Der Cloud-Layer umfasst die zentralen Cloud-Rechenzentren, welche dem System weitere Rechenleistung für aufwendige Aufgaben und Speicherplatz, zur Persistierung großer Datenmengen, zur Verfügung stellen [12].

2.2 Privacy

Der englische Begriff Privacy vereint die Konzepte des Datenschutzes und der Privatsphäre, daher wird in dieser Arbeit kein Versuch einer Übersetzung angestellt. Privacy beinhaltet somit den Schutz aller personenbezogener Daten vor dem Zugriff unautorisierter Parteien. Personenbezogene Daten sind laut DSGVO alle Daten, die eine Person direkt (z.B. Name) oder indirekt (z.B. Arbeitszeiten, Standort) identifizierbar machen [3].

2.2.1 Privacy in Fog-Computing-Systemen

Yi et al. [36] geben drei Kategorien für Privacy in Fog-Computing-Systemen, folgend auch Fog-Systeme, an: Während Data Privacy den Schutz von persönlichen Daten bezeichnet, betrifft Usage Privacy den Schutz von Nutzungsdaten. Die Kategorie Location Privacy bezieht sich ferner auf den Schutz von Standortdaten eines Nutzers."

Während die Heterogenität der Geräte in IoT-Layer und Fog-Layer in einem Fog-System Chancen zur effizienten Nutzung der Cloud bieten, indem typische Aufgaben der Cloud ausgelagert werden [35], stellt eben diese Heterogenität, in Kombination mit den meist eingeschränkten Ressourcen dieser Geräte, eine Herausforderung für den Schutz von personenbezogenen Daten dar. Alrawais et al. [11] benennen verschiedene weitere Probleme, die es zur Gewährleistung von Privacy in Fog-Computing-Systeme, zu lösen gilt - die im Folgenden vereinfacht als Fragestellungen dargestellt werden:

- Wie können Kryptografietechniken, die für beispielsweise die Authentifizierung von berechtigten Parteien oder die Verschlüsselung personenbezogener Daten benötigt werden, auf diesen stark ressourcenbeschränkten Geräten durchgeführt werden?
- Wie kann bei der Vielzahl und Vielfältigkeit der Geräte Vertrauen zwischen diesen etabliert werden?
- Wie kann das Eindringen einer böswilligen dritten Partei in das System, sowohl durch die übernahme bestehender Fog-Knoten als auch durch das Einbringen eigener Fog-Knoten, verhindert werden?

Aufgrund dieser Probleme kategorisieren Guan et al. [19] IoT-Geräte und Fog-Knoten als potenziell böswillig, da stets die Möglichkeit von Angriffen ausgehend von diesen besteht.

2.2.2 Location-Privacy

Der Standort (engl. location) einer Person kann für verschiedenste Applikationen von Interesse sein. So gibt es eine Vielzahl von standortbezogenen Diensten, wie Dienste zur Findung von nahegelegenen Points of Interest (Restaurants, Sehenswürdigkeiten, etc.) oder auch Augmented Reality Games, die ein potenziell legitimes Interesse an dem (aktuellen) Standort eines Nutzers haben [32]. Ebenfalls teilen Nutzer immer häufiger nicht nur einen einzigen Standort, sondern auch Bewegungspfade (engl. movement trajectories), wenn beispielsweise Fitness-Applikationen den gelaufenen Weg aufnehmen [32]. Sowohl Standorte (aktuell oder vergangene) als auch Bewegungspfade eines Nutzers, können einer dritten Partei Rückschlüsse auf (sensitive) personenbezogene Informationen geben [32], weshalb die Möglichkeit besteht, dass durch standortbezogene Dienste die Location-Privacy derer Nutzer gefährdet wird. Hierbei kann die Gefährdung aktiv von Dienstleistern verschuldet sein [17], indem personenbezogene Nutzerdaten unbefugt weitergegeben/-verkauft werden oder passiv verschuldet sein, wenn Nutzerdaten durch einen Angreifer geleakt/gehackt werden [32]. Derartige Nutzerdaten erlauben potenziell böswilligen Parteien Rückschlüsse auf Aufenthaltsorte (z.B. Wohnort, Arbeitsplatz) oder Gewohnheiten (z.B. der Arbeitsweg) [17]. Alastair R. Beresford und Frank Stajano definieren Location-Privacy als:

> „[Location privacy is] the ability to prevent other parties from learning one's current or past location"[14].

Diese Arbeit stützt sich auf diese Definition mit der Ergänzung um den Aspekt der Bewegungspfade eines Nutzers, die es ebenfalls gilt, vor dem Zugriff durch unbefugte Parteien zu schützen. Der Grad, zu dem die Location-Privacy einer Person gefährdet ist, wird durch die Genauigkeit, mit welcher ein Angreifer den Standort oder den Bewegungspfad jener Person bestimmen kann, quantifiziert [18].

2.2.3 Location-Privacy im Fog-Computing

In Fog-Computing-Systemen werden Rechenleistung und Speicherkapazitäten in geografischer Nähe zu Endgeräten angeboten [38]. Werden von einem Endgerät eines solchen Systems, die Ressourcen eines Fog-Knotens bezogen, lässt sich darauf schließen, dass sich jenes Endgerät in dem Abdeckungsbereich dieses Fog-Knoten befindet. Unter Hinzunahme der Annahme, dass sich Endgeräte stets mit dem geografisch nächsten Fog-Knoten verbinden, ließe sich der Standort des Endgerätes weiter präzisieren. Hat ein Angreifer Zugriff auf dieses Wissen mehrerer Fog-Knoten, kann dieser möglicherweise die Bewegung eines Endgerätes in Form von Bewegungspfaden angenähern. Hierzu könnte auf Basis der Abfolge

von Fog-Knoten, mit denen sich das Endgerät auf seinem Weg verbunden hat, der Pfad des Endgerätes annäherungsweise rekonstruiert werden [13].

Das Wissen eines Angreifers über Standort und Bewegungspfade eines Endgerätes betrifft erst dann die Location-Privacy einer Person, wenn es dem Angreifer möglich ist, das Endgerät über den Beobachtungszeitraum einer Person zuzuordnen [18]. Die Zuordnung von Endgeräten zu Personen kann dem Angreifer über Beobachtung in Verbindung mit Hintergrundwissen gelingen. Ist dem Angreifer beispielsweise sowohl Wohnort als auch Arbeitsplatz einer Person bekannt und es wird ein Endgerät beobachtet, welches mehrfach pro Woche einen Pfad von Wohnort bis Arbeitsplatz zurücklegt, kann der Angreifer das Endgerät mit großer Wahrscheinlichkeit eben dieser zuordnen. In dieser Arbeit wird angenommen, dass es Angreifern möglich ist Endgeräte zu Nutzern zuzuordnen. Wie dem Angreifer hierbei diese Zuordnung gelingt, wird jedoch kein Bestandteil dieser Arbeit sein. Hat ein Angreifer nun Wissen über die Bewegungspfade von Personen, könnte dieses Wissen mit weiteren Informationen zu Bewegungsprofilen aggregiert werden [18]. Neben der Aufenthaltsdauer von Personen, in dem Abdeckungsbereich eines Fog-Knoten wären auch Ortskenntnisse Beispiele für relevante Informationen, zur Erstellung von Bewegungsprofilen.

2.2.4 Angriffe auf Location-Privacy im Fog-Computing

Asuquo et al. [13] kategorisieren die Location-Privacy im Fog-Computing betreffenden Angriffstypen wie folgt:

1. Global/Lokal: Es gibt globale Angreifer, die Zugriff auf das gesamte Netzwerk, und damit auf alle Fog-Knoten des Systems, haben und lokale Angreifer, die nur zu einer begrenzten Anzahl an Fog-Knoten Zugang haben.

2. Aktiv/Passiv: Während ein aktiver Angreifer ein Netzwerk beeinträchtigen kann, indem beispielsweise Nachrichten innerhalb des Netzwerks verfälscht werden, können passive Angreifer lediglich als Beobachter agieren und so Nachrichten mitlesen.

3. Statisch/Adaptiv: Ein Angreifer wird dann als statisch kategorisiert, wenn die Angriffsstrategie während des Angriffs unveränderlich ist. Adaptive Angreifer passen ihre Angriffsstrategie dynamisch an die Gegebenheiten des Netzwerks an.

4. Intern/Extern: Interne Angreifer, wie kompromittierte Fog-Knoten, die versuchen unbefugt auf Nutzerdaten zuzugreifen, sind bereits Bestandteil des Netzwerks. Externe Angreifer hingegen müssen erst Zugang zu dem Netzwerk erhalten.

Der in dieser Arbeit betrachtete Angreifertyp kontrolliert eine begrenzte Anzahl an Fog-Knoten und liest die Kommunikation zwischen ihnen und den Endgeräten mit. Es handelt sich somit um einen passiven Angreifer. Das Ziel des Angreifers ist es anhand dieser Kommunikation möglichst präzise Angaben zu dem Standort, bzw. falls möglich zu dem Bewegungspfad, eines Endgerätes und somit eines Nutzers treffen zu können. Wie in Abschnitt 2.2.3 beschrieben kann ein Angreifer, der nur einen Fog-Knoten kontrolliert so nur den aktuellen oder vergangenen Standort eines Nutzers bestimmen, während es einem Angreifer, der mehrere Fog-Knoten kontrolliert, möglich sein könnte, sogar den Bewegungspfad eines Endgerätes anzunähern. Die Präzision der Bestimmung des Standorts und des Bewegungspfads eines Nutzers, und damit der Grad der Gefährdung der Location-Privacy dieses Nutzers, ist von dem zusätzlichen Hintergrundwissen des Angreifers abhängig.

2.2.5 Betrachtete Angriffszenarien

Die von einem Angreifer gesammelten Informationen, welche die Location-Privacy einer Person betreffen, können mit Hintergrundwissen angereichert werden. Beispielsweise kontextualisiert so das Wissen (oder Unwissen) über die geografischen Positionen der Fog-Knoten (kompromittiert und nicht-kompromittierten) die, durch Mitlesen der Kommunikation erlangten, Informationen über Standorte und

	Angreifer kennt nur die geografischen Positionen der kontrollierten Fog-Knoten	**Angreifer kennt die geografischen Positionen aller Fog-Knoten**
Endgeräte sind immer mit dem Fog-System verbunden	Szenario 1.0	Szenario 2.0
Verbindung zwischen Endgerät und Fog-System kann unterbrochen werden	Szenario 1.1	Szenario 2.1

Tabelle 1: Betrachtete Szenarien mit variierendem Hintergrundwissen des Angreifers [33]

Bewegungspfade der Nutzer. Gleiches gilt für das Wissen über die möglichen Verbindungszustände (immer verbunden, Verbindung kann getrennt werden) der Endgeräte innerhalb des Fog-Systems.

Dieser Arbeit werden die Szenarien aus [33] zugrunde gelegt, welche daher folgend nur kurz ausgeführt werden. Da es der Fall, dass nicht einmal die Positionen der kompromittierten Fog-Knoten bekannt sind, einem Angreifer unmöglich machen würde, Aussagen über Standorte oder gar Bewegunspfade einer Person zu treffen, wird stets davon ausgegangen, dass die Positionen aller kompromittierten Fog-Knoten bekannt sind. Für das Spektrum des Wissens über die Positionen der nicht-kompromittierten Fog-Knoten, werden nur die Extremum (keine oder alle Positionen bekannt) betrachtet.

Bezüglich der möglichen Verbindungszustände wird einerseits der Fall betrachtet, dass Endgeräte zu jedem Zeitpunkt mit dem Fog-System verbunden sind. Obwohl dieser Fall ein Extrema darstellt, ist er keinesfalls unrealistisch, da eine permanente Verbindung zu einem Fog-System bereits mit heutiger Netzinfrastruktur vielerorts möglich ist. Andererseits wird der Fall betrachtet, dass die Verbindung zwischen Endgerät und Fog-System unterbrochen werden kann. Der Extremfall, dass ein Endgerät zu keinem Zeitpunkt eine Verbindung zu dem Fog-System eingeht, stellt keinen relevanten Fall dar.

Entsprechend zu [33] ergeben sich so die vier in Tabelle 1 dargestellten Szenarien mit variierendem Hintergrundwissen des Angreifers.

Bewegt sich ein Nutzer mit seinem Endgerät durch ein Fog-System, in welchem einige Fog-Knoten durch einen Angreifer kontrolliert sind, kann dieser Angreifer möglicherweise den Bewegungspfad als Reihenfolge von Standorten des Endgerätes nachverfolgen.

Je präziser der Angreifer hierbei die einzelnen Standorte bestimmen kann, desto präziser kann auch der Bewegungspfad nachvollzogen werden. Hierbei kann der Angreifer einen einzelnen Standort stets nur auf einen möglichen Aufenthaltsregion des Endgerätes, folgend $R(E)$ genannt, eingrenzen. Je kleiner $R(E)$ ist, desto genauer lässt sich der Standort des Endgerätes bestimmen. Folgend wird dargestellt, wie sich die in Tabelle 1 genannten Szenarien auf $R(E)$ auswirken. Es wird für jedes Szenario betrachtet, wie groß $R(E)$ zu zwei, sich möglicherweise wiederholenden, Zeitpunkten ist. Der erste Zeitpunkt $T^{connect}(E,F)$ ist hier der, zu dem eine Verbindung zwischen einem kompromittierten Fog-Knoten und einem Endgerät aufgebaut wird. Der zweite Zeitpunkt $T^{disconnect}(E,\{F_1^{comp},...,F_n^{comp}\})$ ist der, zu dem die Verbindung zwischen Endgerät und dem kompromittierten Teil des Fog-System, $\{F_1,...,F_n\}$, verloren wird. $T^{disconnect}(E,\{F_1^{comp},...,F_n^{comp}\})$ tritt also dann ein, wenn das Endgerät entweder die Verbindung beendet hat (z.B. ausgeschaltet wurde), sich mit einem nicht kompromittierter Fog-Knoten verbunden hat oder das Fog-System vollständig verlassen hat. Betrachtet wird ein kompromittierten Fog-Knoten $F_{betrachtet}^{comp}$. Weitere kompromittierte Fog-Knoten werden folgend mit F^{comp} notiert.

Für Szenario 1.0 und 1.1 gilt zu $T^{connect}(E,F)$: Gibt es keine Überschneidung zwischen dem maximal möglichen Abdeckungsbereich von $F_{betrachtet}^{comp}$, also $A^{max}(F^{comp})$, und dem Empfangsbereich eines anderen kompromittierten Fog-Knoten, gilt $R(E) = A^{max}(F_{betrachtet}^{comp})$. Wird $A^{max}(F_{betrachtet}^{comp})$ mindestens einmal geschnitten, ist $R(E)$ ein Teilbereich von $A^{max}(F^{comp})$. Für diesen Teilbereich gilt, dass $A^{max}(F^{comp}) > Teilbereich \geq A^{real}(F^{comp})$ ist, wobei der tatsächliche Abdeckungsbereich eines Fog-Knoten $A^{real}(F)$, den Bereich darstellt, in welchem sich Endgeräte tatsächlich mit dem Fog-Knoten F (hier F^{comp}) verbinden. Zu $T^{disconnect}(E,\{F_1^{comp},...,F_n^{comp}\})$ gilt hier, dass $R(E)$ dem Bereich des gesam-

Notation	Bedeutung
F	nicht-kompromittierter Fog-Knoten
$\{F_1,...,F_n\}$	Menge aller nicht kompromittierter Fog-Knoten
F^{comp}	kompromittierter Fog-Knoten
$\{F_1^{comp},...,F_n^{comp}\}$	Menge aller kompromittierter Fog-Knoten
E	(mobiles) Endgerät
$R(E)$	mögliche Aufenthaltsregion eines Endgerätes
$A^{max}(F)\|\|A^{max}(F^{comp})\|\|A^{max}(\{F_1,...,F_n\})$	maximal möglicher Abdeckungsbereich (entspricht der Reichweite) eines oder einer Menge von Fog-Knoten
$A^{real}(F)\|\|A(^{real}F^{comp})\|\|A^{real}(\{F_1,...,F_n\})$	tatsächlicher Abdeckungsbereich eines oder einer Menge von Fog-Knoten
$T^{connect}(E,F)$	Zeitpunkt des Verbindungsaufbaus zwischen Endgerät und Fog-Knoten
$T^{disconnect}(E,\{F_1^{comp},...,F_n^{comp}\})$	Zeitpunkt zu dem die Verbindung zwischen Endgerät und dem kompromittierten Teil des Fog-System getrennt wird

Tabelle 2: Notation

ten Fog-System entspricht.

Da in Szenario 2.0 und Szenario 2.1 $A^{real}(F^{comp})$ bekannt ist, gilt zu $T^{connect}(E,F)$, dass $R(E) = A^{real}(F^{comp})$ ist. Zu $T^{disconnect}(E,\{F_1^{comp},...,F_n^{comp}\})$ muss zwischen jedoch Szenario 2.0 und Szenario 2.1 differenziert werden. Für Szenario 2.0 muss das observierte Endgerät $T^{disconnect}(E,\{F_1^{comp},...,F_n^{comp}\})$ den kompromittierten Teil des Fog-System verlassen haben, daher gilt $R(E) = R(\{F_1,...,F_n\})$. In Szenario 2.1 könnte sich das observierte Endgerät auch nach dem Verlust der Verbindung (z.B. nach dem es ausgeschaltet wurde) weiterhin in $\{F_1^{comp},...,F_n^{comp}\}$ befinden, daher lässt sich der Standort nicht einschränken und es gilt wie für Szenario 1.0 und 1.1, dass $R(E)$ dem Bereich des gesamten Fog-System entspricht.

2.2.6 Annahmen

Da es das Ziel dieser Arbeit ist, den in [33] beschriebenen Simulator LocPrivFogSim zur Simulation von Location-Privacy Risiken in Fog-Systemen zu erweitern, werden ihr Annahmen aus [33] zugrunde gelegt:

- Die Engeräte der Nutzer verbinden sich stets mit den geografisch nächsten Fog-Knoten. Weitere Verbindungskriterien, wie aktuelle Auslastung der Fog-Knoten, werden demnach nicht betrachtet.
- Dem Angreifer sind die Standorte der von ihm kompromittierten Fog-Knoten bekannt.
- Der betrachtete Raum, durch den sich ein Endgerät bewegt, wird vollständig durch das Fog-System abgedeckt.
- Alle Fog-Knoten des Systems sind stets eingeschaltet und bereit dafür eine Verbindung zu einem Endgerät aufzunehmen.
- Jedes Endgerät hat eine eindeutige ID, mit welcher der Angreifer ein Endgerät identifizieren/wiedererkennen kann.
- Es ist dem Angreifer möglich Endgeräte zu Personen zuzuordnen.
- Der Angreifer kennt alle möglichen Bewegungspfade zu einer Spur eines Endgerätes (vgl. Abschnitt 2.2.7

2.2.7 Quantifizierung der Gefährdung für Location-Privacy

Um die Gefährdung der Location-Privacy innerhalb eines Fog-Systems zu quantifizieren werden in [33] zwei Metriken herangezogen. Diese wurden in dem Simulator LocPrivFogSim implementiert und stehen somit für diese Arbeit zur Verfügung.

Der **Trace Comparison Value** gibt die Anzahl der möglichen Bewegungspfade zu einer Spur eines Endgerätes an. Die Spur eines Endgerätes ergibt sich hier aus der Reihenfolge der Fog-Knoten, mit denen sich das Endgerät auf dem Weg verbunden hat. Es wird in dieser Arbeit angenommen, dass der Angreifer alle möglichen Bewegungspfade zu einer Spur kennt. Zusätzlich trifft der Angreifer jeweils eine Annahme darüber, ob sich ein mobiles Endgerät auf dem Weg von dem System trennen kann oder nicht. Je größer der Trace Comparison Value ist, desto unwahrscheinlicher ist es, dass einer der möglichen Bewegungspfade auch dem tatsächlich zurückgelegten Bewegungspfad des Endgerätes entspricht. Der Trace Comparison Value ist nicht davon abhängig, ob der Angreifer die Positionen aller Fog-Knoten kennt, sondern lediglich von der Bedingung, ob sich ein mobiles Gerät von dem Fog-System trennen darf [33]. Die Location-Privacy einer Person ist dann besonders gefährdet, wenn Trace Comparison Value 1 ist, da der Angreifer in diesem Fall den Bewegungspfad der Person eindeutig nachvollziehen kann.

Die **Size of Uncertainty Region** gibt den Durchschnitt der Genauigkeit an, mit welcher der Angreifer den Standort einer Person, über einen gewissen Zeitraum, bestimmen kann [33]. Die **Size of Uncertainty Region** ist direkt von dem gewählten Szenario abhängig, da der Angreifer abhängig von diesen die Region, in welcher sich das mobile Endgerät befinden kann, unterschiedlich eingrenzen kann (vgl. 2.2.5). Der Wertebereich der **Size of Uncertainty Region** liegt zwischen 0 und 1, wobei das Wissen des Angreifers mit sinkender Size of Uncertainty Region zunimmt.

2.3 Der Fog-Computing-Simulator LocPrivFogSim

Mit LocPrivFogSim [33] lassen sich Risiken für Location-Privacy in Fog-Systemen simulieren. Mit LocPrivFogSim lassen sich verschiedene Fog-Geräte, wie mobile Endgeräte und Fog-Knoten darstellen. Des Weiteren lässt sich die Bewegung eines mobilen Endgerätes durch ein Fog-System abbilden. Auf dem Pfad des Endgerätes, kann sich dieses mit verschiedenen Fog-Knoten verbinden. Welche Verbindungen hier zustande kommen, ist von der gewählte Verbindungsstrategie abhängig. Sowohl in [33] als auch in dieser Arbeit verbindet sich ein mobiles Endgerät stets mit dem nächstgelegenen Fog-Knoten. Zusätzlich lassen sich mit LocPrivFogSim verschiedene Angriffsszenarien simulieren. Dafür existiert die Rolle des Angreifers (Attacker), welcher Fog-Knoten kontrollieren und deren Datenverkehr beobachten kann. Anhand des Datenverkehrs kann der Angreifer Wissen über Standort und Bewegungspfad von Endgeräten erlangen. Um dieses Wissen zu quantifizieren, berechnet LocPrivFogSim die in Abschnitt 2.2.7 beschriebenen Metriken **Trace Comparison Value** und **Size of Uncertainty Region**.

In [33] wird folgender Simulationsaufbau gewählt: Fog-Systeme werden durch ein $50 * 50$ großes Quadrat dargestellt, auf welchem 20 Fog-Knoten plaziert sind. Jeder dieser Fog-Knoten deckt einen Kreis mit Radius 15 ab. Entsprechend des ausgewählten Verhältnis von kompromittierten Fog-Knoten zu nicht-kompromittierten Fog-Knoten, kontrolliert der Angreifer eine Anzahl an zufällig gewählten Fog-Knoten. Die Simulation wurde für jede Anzahl von kompromittierten Fog-Knoten von 1 bis 20 durchgeführt. Der simulierte Bewegungspfad eines Endgerätes wird zufällig aus einer Menge von 50 vordefinierten Pfaden ausgewählt. In [33] wurden für jede der, in Abschnitt 2.2.5 genannten Szenarien, jeweils 100 Simulationen durchgeführt. Die Simulationsergebnisse aus [33] zeigen, dass es für den gewählten Simulationsaufbau, einem Angreifer bereits bei einer geringen Anzahl an kompromittierten Fog-Knoten möglich wäre präzise Aussagen bezüglich des Bewegungspfades eines Endgerätes zu tätigen.

2.4 Der GeoLife Datensatz

In dieser Arbeit wird eine Erweiterung des Fog-Computing-Simulator LocPrivFogSim vorgestellt, die es ermöglicht reale Mobilitätsdaten zur Simulation von Location-Privacy-Angriffen zu nutzen. Die hierfür verwendeten Mobilitätsdaten entstammen der Version 1.3 des GeoLife Datensatzes [8, 40, 41, 42]. Der Datensatz entstand im Verlauf des von Microsoft Research Asia durchgeführten GeoLife Projektes. In Tabelle 3 werden die relevanten Eckdaten bezüglich des GeoLife Datensatzes dargestellt. Die Bewegungspfade wurden von den Teilnehmern des GeoLife Projektes während der Ausübung verschiedenster

Dateigröße	ca. 1.6GB
Anzahl Pfade	18.670
Anzahl Punkte	24.876.978
Gesamtlänge der Pfade	1.292.951 Kilometer
Anzahl der Teilnehmer	182
Zeitraum	April 2007 - August 2012
Koordinatendichte auf Pfaden	Für 91,5% aller Pfade wurden im Abstand von ca 1-5 Sekunden Koordinaten aufgenommen
Primäres Aufnahmegebiet	Peking

Tabelle 3: Übersicht über den GeoLife Datensatz

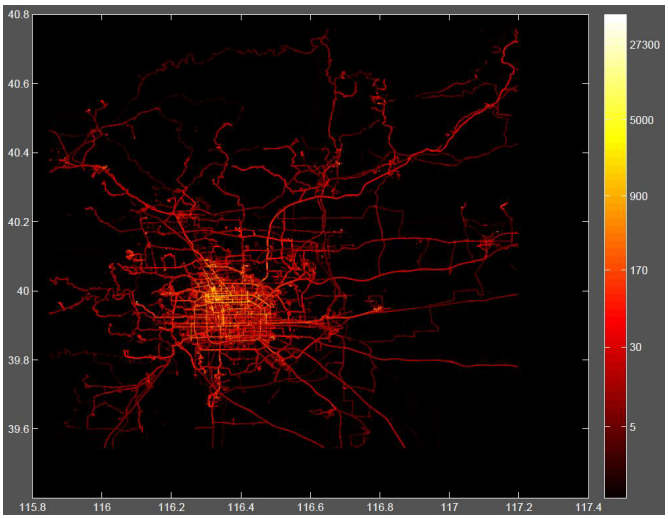


Abbildung 2: Heatmap: Verteilung der Mobilitätsdaten des GeoLife Datensatzes in Peking [43]

Aktivitäten aufgenommen.

In Abbildung 2 wird die Verteilung der Mobilitätsdaten des GeoLife Datensatzes in Peking in Form einer Heatmap angegeben. Auf der X-Achse werden Längengrade und auf der Y-Achse Breitengrade dargestellt. Die Zahlen rechts von der Heatbar (rechts in Abbildung 2) entsprechen der Anzahl an Punkten, die in einem bestimmten Bereich aufgenommen wurden.

In der Vergangenheit wurden unterschiedliche Versionen des GeoLife Datensatz für verschiedene Anwendungen genutzt. So nutzen Xiao et al. [34] den Datensatz, um ähnliche Nutzer auf Basis derer Bewegungspfade zu finden. In einer weiteren Arbeit stellten Yoon et al. [37] einen Systemansatz vor, der auf Basis des GeoLife Datensatzes Reiserouten erstellt.

In dem Datensatz, ist für jeden Nutzer ein Ordner angelegt, in welchem für jeden Bewegungspfad eine PLT-Datei enthalten ist. Innerhalb der PLT-Dateien wird ein Bewegungspfad (ab Zeile 7) als Reihe von Punkten dargestellt. Ein Punkt entspricht einer Zeile, die jeweils GPS-Koordinaten mit zugehörigen Datum und Zeit enthält. Das Format dieser Zeilen ist:

<Breitengrad>,<Längengrad>,0,<Höhe>,<Tage seit 12/30/1899>,<Datum>,<Zeit>

Ein Beispiel einer validen Zeile wäre entsprechend:

39.975171,116.330073,0,404.562135826772,40602.7834259259,2011-02-28,18:48:08

3 Erweiterung und Anpassung von LocPrivFogSim um Nutzung von realen Pfaden in der Simulation

In diesem Kapitel werden die Erweiterungen und Anpassungen an dem Simulator LocPrivFogSim beschrieben. Die benötigten Erweiterungen/Anpassungen ergeben sich dabei aus dem in Unterkapitel 3.1 beschriebenen, für diese Arbeit gewählten, Simulationsaufbau. Daraufhin wird in Unterkapitel 3.2 die Integration der Pfade des GeoLife-Datensatzes in den Simulator beschrieben. Dabei wird insbesondere auf die Aufbereitung der Pfade eingegangen. Daraufhin werden in Unterkapitel 3.3 die tatsächliche Implementierung und damit einhergehende Designentscheidungen der Erweiterungen/Änderungen dargestellt.

3.1 Darstellung des gewählten Simulationsaufbau

In diesem Unterkapitel werden die Entscheidungen bezüglich des gewählten Simulationsaufbaus ausgeführt. Darunter fällt die Wahl des Simulationsfeldes, die Wahl der Parameter bezüglich der simulierten Fog-Knoten und letztlich die Anforderungen, die Pfade des GeoLife-Datensatzes erfüllen müssen, sodass diese als in die Simulation aufgenommen werden. Dieses Kapitel stellt die Grundlage der weiteren benötigten Implementierungsarbeiten, hinsichtlich der Aufbereitung der Pfade und der Erweiterung/Anpassung des Simulators, dar.

3.1.1 Simulationsfeld

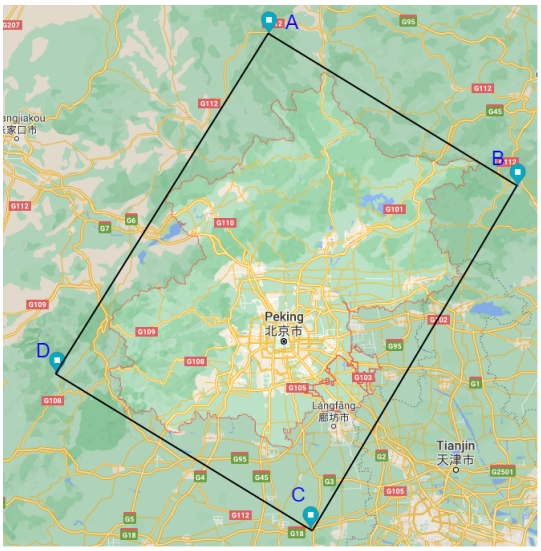


Abbildung 3: geschätztes Peking umfassendes Feld mit den Eckpunkten A,B,C,D

A	41.229556359747455, 116.32212585037877
B	40.56688298442188, 117.73143927643063
C	39.099855, 116.519898
D	39.75667, 115.13559

Tabelle 4: Koordinaten der Eckpunkte des Feldes

In dieser Arbeit wird simuliert, wie sich ein mobiles Endgerät durch ein Feld bewegt, welches vollständig von einem Fog-Netzwerk abgedeckt wird. Der Pfad des Endgerätes soll überall auf dem Feld starten und enden können, um so möglichst verschiedene Situationen abzubilden. Dies wird durch die Bestimmung eines festgelegten Feldes, welches bereits möglichst viele Pfade enthält gewährleistet. Die in dem GeoLife-Datensatz enthaltenden Pfade befinden sich größtenteils in Peking, daher ist es naheliegend ein Feld zu konstruieren, welches Peking umfasst. Die Konstruktion eines minimal großen Feldes, welches eine Stadt (hier Peking) umfasst, stellt keinen Betrachtungsgegenstand dieser Arbeit dar. Daher wird das Feld lediglich grafisch so geschätzt, dass die Stadtgrenzen Pekings möglichst gut eingegrenzt

werden. In Abbildung 3 wird das entstandene Rechteck ABCD mit den Seitenlängen von ca. 140 km und 194 km dargestellt. Tabelle 4 enthält die den Eckpunkten zugehörigen GPS-Koordinaten.

3.1.2 Fog-Knoten

Anzahl	27.160
angenommener Abdeckungsbereich	optimal kreisförmig mit Radius = 2 Km
Verteilung	Randomisiert, 300 Meter Mindestabstand
Weitere Eigenschaften	stationär, stets Verfügbar, decken gesamtes System ab

Tabelle 5: Angenommene Parameter bezüglich der Fog-Knoten

Fog-Knoten können, wie in Abschnitt 2.1 beschrieben, verschiedenster Ausprägung sein. Im Kontext dieser Arbeit werden Eigenschaften von Fog-Knoten wie Speicher- und Rechenkapazitäten nicht weiter betrachtet, da diese keine Auswirkungen auf das in dieser Arbeit simulierte System haben. In Tabelle 5 werden die in dieser Arbeit angenommenen und für die Simulation relevanten Parameter für Fog-Knoten dargestellt. Neben den aus [33] übernommenen Parametern wurden hier Werte für die Gesamtanzahl, den Radius des Abdeckungsbereichs und den Mindestabstand zwischen den Fog-Knoten bestimmt:

- **Radius des Abdeckungsbereichs:** Der Radius des angenommen Abdeckungsbereichs eines Fog-Knoten entspricht maximal der Reichweite des entsprechenden Fog-Knoten. Als Anhaltspunkt zur Bestimmung eines Wertes dieser Reichweite wird sich an der Reichweite von 5G-Basisstationen orientiert, da es sich für ein flächendeckendes Fog-System anbieten könnte, die Fog-Knoten über eben diese zu vernetzen. Der größte chinesische Mobilfunkanbieter China Mobile verfügt über das Frequenzband n41 bei welchem sich die Frequenzen im Bereich von 2515 MHz und 2675 MHz befinden[22]. Die Reichweite von Basisstationen die auf derartigen Frequenzen kommunizieren liegt im Bereich von ungefähr 1 km bis 3 km [10]. In dieser Arbeit wird eine, der Mitte dieser Spanne entsprechende, Reichweite von ungefähr 2 km angenommen. Der Abdeckungsbereich der Fog-Knoten wird somit als kreisförmig angenommen und besitzt einen Radius von 2 km.

- **Verteilung:** Da Fog-Knoten potenziell an beliebigen Stellen platziert werden können, werden die Standorte der Fog-Knoten in dem simulierten System randomisiert festgelegt. Es gilt trotz Randomisierung sicherzustellen, dass stets das gesamte Feld durch das Fog-System abgedeckt ist (vgl. Annahme 2.2.6). Dazu werden die Fog-Knoten zuerst mit einem Abstand von jeweils 1 km in einem 139 km x 193 km großen Gitter, das jeweils 0.5 km Abstand zu den Kanten des 140 km x 194 km Feldes hat, angeordnet. Daraufhin wird jeder Fog-Knoten jeweils um einen zufälligen Wert zwischen -0.5 km und 0.5 km in der Vertikalen und der Horizontalen verschoben. Auf diese Weise können einerseits sämtliche Koordinaten des Feldes als Standort von Fog-Knoten auftreten und andererseits ist es ausgeschlossen, dass durch die Randomisierung Lücken im Fog-System entstehen. Da von einer Mindestreichweite der 5G-Masten von 300 m ausgegangen wird, wird angenommen, dass Fog-Knoten mindestens 300 m voneinander entfernt sind.

- **Anzahl**: Entspricht der für die beschriebene Verteilung benötigten Anzahl an Fog-Knoten.

3.1.3 Pfade

Der simulierte Pfad eines mobilen Endgerätes wird zufällig aus den sich in Peking befindenden Pfaden des GeoLife-Datensatzes (vgl. Abschnitt 2.4) ausgewählt. Die Pfade werden jeweils über eine Abfolge an Standorten (als GPS-Koordinaten) und zugehörige Zeitstempel dargestellt. Wie präzise diese aufeinander folgenden Standorte tatsächlich den aufgenommenen Pfad darstellen, ist einerseits von der Genauigkeit

der einzelnen Standortbestimmungen und andererseits von den zeitlichen Abstände, zwischen den Standorten abhängig. Die Genauigkeit der Standortbestimmung ist primär von dem gewählten Gerät abhängig, das zur Aufnahme des Pfades gewählt wurde. Bei der Erhebung des GeoLife-Datensatzes wurden hauptsächlich GPS Logger und Smartphones genutzt [43]. Insbesondere können dabei die durch Smartphones aufgenommen Standorte eine geringere Präzision aufweisen. Dies ist hauptsächlich darin begründet, dass die Geo-Lokalisierung eines Smartphones auf verschiedenen Technologien (Zell-ID, WiFi, GPS) basieren kann und daher die Genauigkeit auch über einen Pfad hinweg variieren kann. Daher gilt es mögliche unpräzise Standortbestimmungen zu berücksichtigen.

Für den gewählten Systemaufbau werden nur Pfade genutzt, die bestimmten Anforderungen genügen:

1. **Mindestlänge:** Pfade sollen aus mindestens 40 Koordinaten bestehen und mindestens eine Koordinate des Pfades soll 200 Meter von der Startposition entfernt sein. Diese Werte wurden gewählt, sodass Pfade, die aus Versehen entstanden sind, beispielsweise durch versehentlichen Start des GPS-Trackings, nicht in der Menge der möglicherweise simulierten Pfade auftreten.

2. **Abstände zwischen Standorten:** Aufeinander folgenden Standorte eines Pfades sollen maximal 20 m und/oder maximal 10 s auseinander liegen. Diese Werte stellen jeweils das Doppelte der laut [43] geltenden gewöhnlichen Koordinatendichte des GeoLife-Datensatzes (maximal 10 m und 5 s Differenz) dar. Auf diese Weise wird auch verhindert, dass Pfade mit grob unpräzisen Standortbestimmungen als simulierte Pfade auftreten.

3. **Innerhalb Pekings:** Pfade sollen sich in den Grenzen des in Abschnitt 3.1.1 beschriebenen Feldes und somit in der angenäherten Stadtgrenze Pekings befinden.

Erfüllt ein Pfad im gesamten diese Anforderungen nicht, wird geprüft ob ein oder mehrere Teilpfade diese Anforderungen erfüllen. Existieren derartige Teilpfade, werden sie in die Menge der möglicherweise simulierten Pfade aufgenommen.

Duplikate

Da die 17.621 Pfade des GeoLife-Datensatzes von lediglich 181 Teilnehmern des Projektes aufgezeichnet wurden, lässt sich vermuten, dass einige Pfade mehrfach aufgezeichnet wurden. Dies könnte beispielsweise eintreten, wenn eine Person mehrfach den gleichen Arbeitsweg aufgezeichnet hat. Tritt ein Pfad mehrfach auf, beeinflusst dies die Wahrscheinlichkeit mit welcher dieser Pfad - als simulierter Pfad des mobilen Endgerätes - gewählt wird. Dies stellt ein gewünschtes Verhalten dar, da ein Angreifer diese Pfade auch in der Realität vermutlich häufiger verfolgen würde. Jedoch beeinflussen mehrfach auftretende Pfade auch den berechneten Trace Comparison Value. So würde zu einer durch einen Angreifer verfolgten Spur nicht nur der tatsächlich gewählte Pfad und alternative Pfade zugeordnet, sondern es würden auch Replikationen des tatsächlichen Pfades und der alternativen Pfade zugeordnet werden. Dadurch würde der Trace Comparison Value zu einer Spur verfälscht werden. Um dem entgegen zu wirken, werden Dopplungen von Pfaden aus dem Datensatz entfernt. Um das gewünschte Verhalten, dass häufig aufgezeichnete Pfade auch häufiger zur Simulation gewählt werden, zu erhalten, werden die Pfade, entsprechend der Anzahl mit der sie im GeoLife-Datensatz vertreten sind, gewichtet. Folglich wird die Gewichtung bei der Auswahl eines simulierten Pfades eines Endgerätes einbezogen.

3.2 Integration des GeoLife-Datensatzes in LocPrivFogSim

In diesem Kapitel wird beschrieben, wie die Pfade des GeoLife-Datensatzes aufbereitet und in eine der Simulation dienliche Form überführt werden. Ein wichtiger Aspekt hierbei ist die Entfernung der Pfad-Duplikate, um so gewährleisten zu können, dass Duplikate nicht den Trace Comparison Value im Endergebnis verfälschen. In Abschnitt 3.2.3 werden die Ergebnisse der Vorverarbeitung der GeoLife-Pfade dargestellt.

3.2.1 Aufbereitung und Bereinigung des GeoLife Datensatzes

Für die initiale Filterung und Aufbereitung der relevanten Pfade des GeoLife-Datensatzes werden diese einmalig in den Hauptspeicher geladen. Hierzu werden die, wie in Abschnitt 2.4 beschrieben formatierten, PLT-Dateien zeilenweise durchlaufen. Zeilen, die hier relevante Informationen, also Standorte in Form von GPS-Koordinate und Zeitstempel, enthalten sind stets gleich strukturiert (vgl. Abschnitt 2.4). So lassen sich diese über einen regulären Ausdruck identifizieren und die Standorte in eine sortierte Liste überführen. Jede so entstandene Liste entspricht einem unverarbeitetem und somit potentiell unbrauchbarem Pfad, welchen es gilt aufzubereiten oder auszusortieren. Bei der Aufbereitung der Pfade des GeoLife-Datensatzes stehen die Anforderungen aus Abschnitt 3.1.3 im Vordergrund:

1. Pfade bestehen aus mindestens 40 Koordinaten.
2. Mindestens eine Koordinate auf dem Pfad soll über 200 m von der Startposition entfernt sein
3. Aufeinander folgende Koordinaten dürfen maximal 20 m auseinander liegen und/oder maximal in einem Abstand von 10 s aufgenommen worden sein.
4. Alle Koordinaten eines Pfades befinden sich innerhalb des in Abschnitt 3.1.1 bestimmten Feldes.
5. Jeder Pfad ist einmalig.

Abbildung 4 Pseudocode: Anforderungen 1-4

Input List of locations L ▷ location = [lat,lon, timestamp]
Output List of valid paths P

while $L.length() > 40$ **And** Distance($L[0]$, $L[L.length() - 1] > 200$) **do**
  Set loc_{start} to $L[0]$ ▷ startposition
  Set loc_{curr} to $L[0]$ ▷ currently viewed location
  Init loc_{prev} with null ▷ previously viewed location
  Init loc_{curr} $tmpPath$ with null
  Init $distanceValid$ with True ▷ distance between adjacent coordinates
  Init $max\_dist\_to\_start = 0$
  Init $timeDIffValid$ with True
  Check If loc_{curr} isInPeking()
  while ($distanceValid$ **Or** $timeDIffValid$) **And** $inPeking$ **do**
    Add loc_{curr} to $tmpPath$
    Set $max\_dist\_to\_start$ to maxOf($max\_dist\_to\_start$, Distance(loc_{curr}, loc_{start}))
    Set loc_{prev} to loc_{curr}
    Set loc_{curr} to $L[tmpPath.length()]$
    Update $distanceValid$, $timeDIffValid$, $inPeking$ for loc_{curr}
  end while
  Delete all $tmpPath$ Entries from L
  if $tmpPath.length() > 40$ **And** $max\_dist\_to\_start > 200$ **then**
    Add $tmpPath$ to P
  end if
end while

Genügt ein Pfad im gesamten den Anforderungen nicht, werden mögliche valide Teilpfade ermittelt. So können aus einem ursprünglich ungültigen Pfad auch mehrere gültige Pfade extrahiert werden. Auf diese Weise wird ermöglicht, dass eine möglichst große Vielfältigkeit der Pfade erhalten bleibt. Offensichtlicher Weise werden doppelte Pfade erst nach Vollendung der übrigen Überarbeitung gefiltert. Pfade die den Anforderungen 1 bis 4 genügen lassen sich in einem Iterativen Prozess erhalten. Ausgang ist die sortierte Liste L der Standorte eines unverarbeiteten Pfades. Produkt des Prozesses ist eine Liste alle validen Pfade P. Es existiert eine Variable $max\_dist\_to\_start$, welche die aktuell maximale

Distanz berechnete Distanz zwischen einer Koordinate und der Startkoordinate der aktuellen Iteration hält (entsprechend der Anforderung 2). Solange L die Anforderung 1 erfüllen kann wird in jeder Iteration zuerst geprüft, ob der erste Standort aus L sich innerhalb des Peking begrenzenden Feldes befindet. Ist dies nicht der Fall, wird dieser Standort aus L entfernt und es wird fortgefahren. Ist der Standort jedoch innerhalb des Feldes, wird der Standort einer weiteren Liste *tmpPath* hinzugefügt und daraufhin als vorheriger Standort loc_{prev} zwischengespeichert. Zusätzlich wird für jeden Standort geprüft, ob die Distanz zur Startposition größer als das aktuelle Maximum (*max_dist_to_start*) ist und falls ja wird *max_dist_to_start* ersetzt. Analog wird für alle folgenden Standorte aus L verfahren, bis eine der Anforderungen 1,3 oder 4 nicht mehr erfüllt ist. Ist Anforderung 3 und/oder Anforderung 4 nicht erfüllt, wird geprüft ob der in *tmpPath* zwischengespeicherte Pfad Anforderungen 1 und 2 genügt, ist dies der Fall wird der (Teil-)Pfad den validen Pfaden P hinzugefügt. In jedem Fall werden die Standorte die in *tmpPath* zwischengespeichert sind aus L gelöscht und daraufhin *tmpPath* und loc_{prev} zurückgesetzt, bevor die nächste Iteration beginnt.

Zur Berechnung der Distanz zwischen zwei Koordinaten wird hier die Haversine-Formel [26] verwendet, mit welcher die kürzeste Strecke zwischen zwei Punkten auf einer Kugeloberfläche berechnet werden kann:

$$d = 2r\sin^{-1}\left(\sqrt{sin^2\left(\frac{\phi_2-\phi_1}{2}\right)+\cos(\phi_1)\cos(\phi_2)\sin^2\left(\frac{\psi_2-\psi_1}{2}\right)}\right) \tag{3.1}$$

wo d die Haversine Distanz zwischen zwei Punkten mit den Breitengraden ϕ_1, ϕ_2 und Längengraden ψ_1, ψ_2 für einen Radius r, hier der Erdradius, ist.

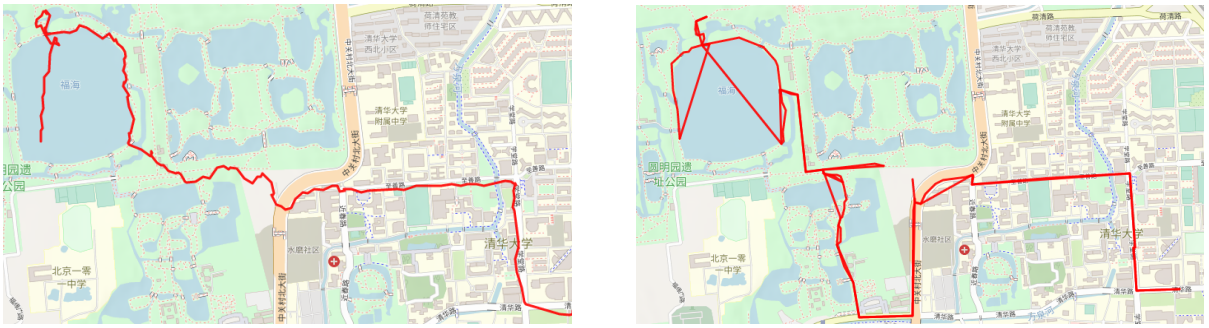
Das Bestimmen, ob sich ein Standort Z innerhalb des Feldes befindet gelingt unter Nutzung des Kreuzproduktes. Die Kanten des Feldes werden durch die vier Vektoren $\vec{AB}$, $\vec{BC}$, $\vec{CD}$, $\vec{DA}$ beschrieben. Z befindet sich genau dann in dem Feld, wenn Z sich jeweils rechts von diesen Vektoren liegt. Für $\vec{AB}$ würde Z dann rechts liegen, wenn auch $\vec{AZ}$ rechts (oder im Uhrzeigersinn gedreht) liegt. Dies ist dann gegeben, wenn das Kreuzprodukt (in R^2 gleich der Determinante) $\vec{AB}$ x $\vec{AZ} > 0$ ist [2]. Insgesamt gilt demnach, Z innerhalb des Feldes liegt, wenn $\vec{AB}$ x $\vec{AZ} > 0$ und $\vec{BC}$ x $\vec{BZ} > 0$ und $\vec{CD}$ x $\vec{CZ} > 0$ und $\vec{DA}$ x $\vec{DZ} > 0$ sind.

3.2.2 Entfernung von Pfad-Duplikaten

Wie in Abschnitt 3.1.3 ausgeführt, sollen die in der Simulation genutzten Pfade duplikatfrei sein. In diesem Abschnitt soll daher erläutert werden, wie Duplikate zu Pfaden identifiziert werden können. Hierbei ist zu beachten, dass die Gleichheit zweier Pfade nicht auf der Gleichheit der Abfolge/Sequenz aller Koordinaten beruht, sondern zwei Pfade als gleich gewertet werden, wenn die Koordinaten-Sequenzen der Pfade eine ausreichende Ähnlichkeit aufweisen. Pfade die im Rahmen dieser Arbeit als gleich gewertet werden, müssen dabei jedoch nicht auch von tatsächlich gleichen Pfaden ausgehen. Ebenso können ursprünglich gleiche Pfade im Rahmen dieser Arbeit möglicherweise nicht als gleich gewertet werden. Dies könnte beispielsweise auf Ungenauigkeiten bei der Aufnahme der Pfade zurückzuführen sein, welche in unähnlichen Koordinaten-Sequenzen resultieren könnten. Auf Grund der in Abschnitt 3.2.1 beschriebenen Aufbereitung des Datensatzes sollten diese Fälle jedoch unwahrscheinliche Ausnahmen darstellen. Folgend werden verschiedene Ansätze ausgeführt, mit welchen eine (synthetische) ausreichende Ähnlichkeit zweier Pfade bestimmt werden kann.

Transformationen:

Eine Möglichkeit hier die Ähnlichkeit zweier Sequenzen zu erkennen, besteht darin, dass die zu vergleichenden Sequenzen auf die gleiche Art transformiert werden, mit dem Ziel, dass die Ergebnisse der Transformationen einfacher, bzw. Optimalerweise direkt vergleichbar sind [24]. Exemplarisch könnte für zwei Sequenzen $A = 0,2,4,2,2$ und $B = 0,0,0,2,4,4,2$ die Transformation darin bestehen, dass Elemente die gleich dem vorherigen Element sind (falls vorhanden) entfernt werden. Da diese Transformation jeweils zu der Sequenz $X = 0,2,4,2$ führen würde, könnte so geschlossen werden, dass die



(a) ursprünglicher Pfad **(b)** generierte Route

Abbildung 5: Mittels GraphHopper [4] generierte Routen, repräsentieren komplexere Pfade des GeoLife-Datensatzes häufig nur ungenügend.

Sequenzen A und B eine gewisse Ähnlichkeit aufweisen müssen oder unter entsprechenden Bedindungen sogar als gleich angesehen werden können. Diesem Prinzip folgend könnten Pfade beispielsweise auf ihre Form reduziert werden und die erhaltenen Formen daraufhin verglichen werden [24]. Es würde dabei jedoch von Informationen wie realen Koordinaten und tatsächlichen Abständen zwischen diesen Koordinaten abstrahiert werden. Es würde somit von Informationen, welche für die möglichst realitätsnahe Simulation der Benutzerpfade des GeoLife-Datensatzes relevant sind, abstrahiert werden müssen, weshalb dieses Verfahren im Kontext dieser Arbeit ungeeignet ist.

Map Matching:

Wie in 3.1.3 dargelegt, ist das Aufkommen von ungenauen Standortmessungen nicht unwahrscheinlich. Daher stellen vor allem Transformationsmöglichkeiten, welche robust gegenüber derartigen Ungenauigkeiten scheinen, aussichtsreiche Lösungen dar. Die Nutzung von Map Matching Algorithmen stellen eine eben solche Lösung dar. Mit diesen ist es möglich ein Mapping von im Voraus aufgenommene Pfaden, jeweils aus einer Reihenfolge von GPS-Koordinaten bestehend, auf reale Routen in einem Straßen-/Wegsystem zu erstellen. Passen laut diesem erstellten Mapping zwei oder mehrere Pfade zu derselben Route, würde sich annehmen lassen, dass auch die Pfade ausreichend ähnlich sind, um sie als gleich bewerten zu können. GraphHopper [4] bietet einen Open-Source-Routing-Server an, mit welchem eine Map Matching Funktionalität über eine Web Schnittstelle angeboten wird. So lassen sich für Pfade, die vorher in das GPX-Format überführt wurden, passende Routen erstellen. Es lassen sich Routen für verschiedene Bewegungsmodi (zu Fuß, Auto, etc.) erstellen, wobei zu beachten ist, dass der Modus über eine Route hinweg unveränderlich ist. Die Routenerstellung durch GraphHopper wurde stichprobenartig für verschiedene Pfade des GeoLife-Datensatzes erprobt. Hierzu wurden die ursprünglichen Pfade und die mittels GraphHopper generierten Routen graphisch dargestellt (z.B. via GPS Visualizer [1]). Es zeigte sich, dass Routen, die für kurze, einfache Pfade erstellt wurden, diese Pfade präzise abdeckten. Wie in 5 dargestellt, scheitert dieser Ansatz jedoch dabei für komplexere, längere Pfade adäquate Routen zu generieren. Dies scheint in den Einschränkungen der vorkonfigurierten Bewegungsmodi begründet zu sein. Dieser Lösungsansatz ist somit für diese Arbeit nicht zielführend.

Euklidische Distanz:

Neben Lösungen, die darauf abzielen die zu vergleichenden Sequenzen zu transformieren, um so Ähnlichkeiten/Gleichheiten direkt zu erkennen, besteht die Möglichkeit die Unterschiedlichkeit zwischen zwei Sequenzen zu quantifizieren [30]. Die Unterschiedlichkeit zweier Sequenzen lässt sich als Distanz zwischen diesen Sequenzen betrachten. So lässt sich für zwei Sequenzen der Länge n F_1 mit den Elementen $e_{1,1}, e_{1,2}, \ldots, e_{1,i}$ und F_2 mit Elementen $e_{2,1}, e_{2,2}, \ldots, e_{2,i}$ die euklidische Distanz:

$$d_{Euclidean}(F_1, F_2) = \frac{\sum_{n=1}^{N} d(e_{1,i}, e_{2,i})}{n} \tag{3.2}$$

berechnen [30]. Hat eine Sequenz F_1 die Länge n und eine weitere Sequenz F_2 der Länge m mit $n > m$, lässt sich die euklidische Distanz zwischen F_2 und einer Teilsequenz von F_1 der Länge m berechnen. Da

(a) Zwei ähnliche Pfade

(b) Paarweise Zuordnung der Koordinaten beider Pfade

Abbildung 6: Sind Pfade unterschiedlich skaliert und/oder die Koordinaten unterschiedlich dicht verteilt, lässt sich kaum aus der euklidische Distanz auf die Ähnlichkeit dieser Pfade schließen.

1 | 2	∞				
1 | 1	∞				
1 | 0.5	∞				
0 | 0.5	∞				
0 | 0	∞				
	0	∞	∞	∞	∞
		0 | 0	0 | 1	1 | 1	1 | 2

(a) Initialisierung DTW

1 | 2	∞	5.26	4.03	2	1
1 | 1	∞	3.03	2.62	1	2
1 | 0.5	∞	1.62	1.62	1	2.5
0 | 0.5	∞	0.5	0.5	1.62	3.42
0 | 0	∞	0	1	2.41	4.65
	0	∞	∞	∞	∞
		0 | 0	0 | 1	1 | 1	1 | 2

(b) DTW-Ergebnismatrix mit grün eingefärbten warping path

Abbildung 8: Initialisierung und Ergebnismatrix des DTW Algorithmus. Die DTW Distanz beträgt 1 Längeneinheiten im kartesischen Koordinatensystem. Links und unter den Matrizen sind jeweils die Koordinaten der Pfade zu sehen. [29]

in diesem Fall nicht alle Elemente der Sequenz F_1 zur Berechnung der Unterschiedlichkeit gleichzeitig betrachtet werden können, ist ein Informationsverlust nicht auszuschließen. Des weiteren ist die euklidische Distanz nur dann als Maß der Ähnlichkeit herausziehbar, wenn die Elemente der Sequenzen gleich verteilt sind. Im Kontext von Pfaden bedeutet dies, dass stets gelten muss, dass der Abstand zwischen den Koordinaten $k_{1,i}$ und $k_{1,i+1}$ des Pfades 1, gleich dem Abstand zwischen $k_{2,i}$ und $k_{2,i+1}$ des Pfades 2 sein muss. Wie in Abbildung 6 zu sehen, würde im gegenteiligen Fall eine suboptimale Zuordnung die Folge sein. Offensichtlich zueinander gehörende Koordinaten, wie beispielsweise die Spitzen der Pfade in Abbildung 6, würden möglicherweise nicht einander zugeordnet werden. Daher hätte eine auf Basis dieser suboptimalen Zuordnung berechnete euklidische Distanz nur eine sehr stark eingeschränkte Aussagekraft über die Ähnlichkeit der zugrundeliegenden Pfade.

Dynamic time warping:

Der dynamic time warping (DTW) Algorithmus [15] bietet Lösungen zu den genannten Einschränkungen der Nutzung der euklidischen Distanz zur Quantifizierung der Ähnlichkeit zweier Pfade an. So werden im DTW Algorithmus einerseits stets alle Elemente der zu vergleichenden Sequenzen, hier also alle Koordinaten der Pfade, betrachtet und andererseits wird eine optimale Zuordnung der Koordinaten der zu vergleichenden Pfade, bestimmt. Bei dem Vergleich zweier Pfade ist hier die Zuordnung optimal, bei welcher die Summe aller Distanzen zwischen den zugeordneten Koordinatenpaaren minimal ist. Die optimale Zuordnung ist als Pfad, auch warping path [29], durch die Ergebnismatrix des DTW Algorithmus gegeben (s. rechts in Abbildung 8). Folgend wird der DTW Algorithmus anhand eines willkürlich gewählten Beispiels zweier kurzer Pfade in einem kartesischen Koordinatensystem vereinfacht dargestellt. Es sind $\text{Pfad}_a = [[0|0], [0|1], [1|1], [1|2]]$ der Länge $n = 4$ und

Abbildung 7 Pseudocode: Berechnung der DTW-Distanz [15] zwischen zwei Pfaden

Input $Path_a = \{a_1, a_2, ..., a_n\}$ and $Path_b = \{b_1, b_2, ..., b_m\}$
Output DTW-distance between $Path_a$ and $Path_b$

Store $Path_a.length()$ as N and store $Path_b.length()$ as M
Init cost matrix as $C \in^{(N+1)x(M+1)}$
$C[0,0] = 0$
for i=1 to N **do**
    set $C[i,1] = \text{inf}$
end for
for i=1 to M **do**
    set $C[1,i] = \text{inf}$
end for
for i=1 to N **do** ▷ now fill leftover cells
    for j=1 to M **do**
        distance = calcHaversineDistance($Path_a[i], Path_b[j]$)
        $C[i,j] = distance + min(C[i-1,j-1], C[i,j-1], C[i-1,j])$
    end for
end for
return $C[A,B]$ ▷ C[A,B] contains final DTW distance

$\text{Pfad}_b = [[0|0],[0|0.5],[1|0.5],[1|1],[1|2]]$ der Länge $m = 5$ gegeben. Zu Beginn wird eine Matrix C der Größe (n+1)x(m+1) (hier also 5x6) erstellt. Nun wird die Zelle [0,0] der Matrix und die restlichen Felder der ersten Spalte und der ersten Zeile mit ∞ gefüllt, vgl. links in Abbildung 8. Daraufhin beginnt das eigentliche Befüllen der Matrix ausgehend von Zelle $C[1,1]$ und endend in $C[4,5]$. Für jede Zelle wird die lokale DTW_Distanz eingetragen. Für eine beliebige Zelle [i,j] berechnen sich diese DTW_Distanz entsprechend der Formel:

$$\begin{aligned} DTW\_Distanz(i,j) = Distanz(Pfad_a[i], Pfad_b[j]) + min(&DTW\_Distanz(i-1,j-1),\\ &DTW\_Distanz(i,j-1),\\ &DTW\_Distanz(i-1,j)) \end{aligned} \tag{3.3}$$

Für die erste betrachtete Zelle $C[1,1]$ wird die Distanz zwischen den ersten Koordinaten beider Pfade (jeweils [0|0]) berechnet, die 0 beträgt. Zusätzlich sind wie oben beschrieben die ersten Spalte und die ersten Zeile bereits ausgefüllt, und so ist bekannt, dass $DTW\_Distanz(i-1,j-1) = 0$, $DTW\_Distanz(i,j-1) = \infty$ und $DTW\_Distanz(i-1,j) = \infty$ sind, womit 0 das Minimum darstellt. Daher wird in die Zelle $C[1,1] = 0$ eingetragen. Analog dazu werden in die restlichen Zellen iterativ die Kosten eingetragen. Da für jede Zelle die geringste $DTW\_Distanz$ der vorherigen Zellen ([i-1,j], [i,j-1], [i-1,j-1]) addiert wird, liefert die berechnete $DTW\_Distanz$ für $C[n,m]$ (die "letzte Zelle", hier $C[4,5]$) die gesamt $DTW\_Distanz$ zwischen den verglichenen Pfaden.

DTW ist anfällig für verrauschte Daten [30]. Die in Abschnitt 3.2.1 beschriebene Aufbereitung der Pfade wirkt dieser Schwäche des DTW Algorithmus jedoch entgegen, indem offensichtlich fehlerhafte Koordinaten-Sequenzen beseitigt werden. Eine weitere Schwäche des DTW Algorithmus besteht darin, dass lediglich die Koordinaten, getrennt von den zugehörigen Zeitstempeln, betrachtet werden, womit die Dimension der Zeit, welche bei der Bewertung von Pfaden eine zentrale Rolle spielen kann, nicht in der DTW Distanz berücksichtigt wird. Da so auch die Geschwindigkeit, die sich unter Berücksichtigung der Dimension der Zeit nachvollziehen lässt, nicht betrachtet wird, werden zwei ähnliche Pfade, bei denen sich die Transportationsart jedoch unterscheidet, als gleich angenommen. Diese Nachteile sind im Rahmen dieser Arbeit jedoch vernachlässigbar. Einerseits wird angenommen, dass die zeitliche Dimension der sich in Peking befindenden Pfade des GeoLife-Datensatzes, nur eine geringe Aussagekraft über deren Charakter besitzt. Dies wird angenommen, da die Pfade im Verkehr der Großstadt Peking aufgenommen wurden und daher die zeitliche Dimension stark durch Verkehrsaufkommen o.ä. beein-

flusst wird. So ließen sich beispielsweise die Pfade eines sich im Stau befindenden PKWs und einer parallel dazu auf einem Bürgersteig gehenden Person aufgrund der ähnlichen zeitlichen Dimensionen nicht unterscheiden. Daher werden in dieser Arbeit zur Idenfikation gleicher Pfade lediglich die räumlichen Differenzen betrachtet. Für diese wird, aufgrund der oben beschriebenen Eigenschaften des DTW Algorithmus, angenommen, dass die DTW Distanz ein geeignetes Maß darstellt. Zwei Pfade gelten somit dann als gleich, wenn die berechnete DTW Distanz einen Schwellenwert nicht überschreitet. Dieser Schwellenwert soll so gesetzt sein, dass sowohl tatsächliche räumliche Unterschiede, die über einen Pfad hinweg vorkommen können (z.B. die Breite einer Fußgängerstraße), aber auch Ungenauigkeiten in der GPS Standortbestimmung betrachtet werden. Für ersteres wird ein Wert von 10 Metern geschätzt. Für zweiteres wird sich an einer Studie von Krista Merry und Pete Bettinger [23] orientiert, welche eine durchschnittliche Genauigkeit der GPS-Standortbestimmung in einer Stadt von 9.9 Metern errechneten. Der daraus resultierende lokale Schwellenwert von 19.9 Meter bezieht sich jeweils auf ein verglichenes Koordinatenpaar aus den verglichenen Pfaden. Die Anzahl der Koordinatenpaare entspricht der Anzahl der Koordinaten des Pfades, welcher aus der größeren Anzahl an Koordinaten besteht. Insgesamt soll für gleiche Pfade also gelten:

$$\frac{DTW\_Distanz}{Anzahl\_Koordinatenpaare} \leqq 19.9m \tag{3.4}$$

Umsetzung:

Sowohl die intiale Aufbereitung der Pfade (vgl. Abschnitt 3.2.1) als auch der Umgang mit Pfad Duplikaten wurde in Form eines Python Skripts implementiert. In 9 wird als Pseudocode dargestellt, wie Duplikate gehandhabt werden. Begonnen wird dabei mit einer, entsprechend des Pseudocodes für Al-

Abbildung 9 Pseudocode: Filterung der Duplikat

Input List of valid paths $P[path_1[coord_1, ..., coord_n], ...]$
Output Set <$path_a$, $numberOfDuplicates$>

sort P by $Distance(coord_1, [41.229556359747455, 116.32212585037877])$ ▷ $coord_1$ of each path
init X [[$path_1$, $isHandled$], ...] with all paths from P and $false$ for each
init $resultList$ [[$path_1$, $numberOfDuplicates$], ...]
for $i < X.length()$ **do**
    $j = 1$
    if $X[i][1] == true$ **then**
        $continue$
    end if
    append [$X[i][1]$, 0] to $resultlist$
    if $!(i + j < X.length())$ **then**
        $break$
    end if
    while $Distance(X[i][0][0], X[i+j][0][0]) < 19.9$ **do** ▷ X[i][0][0] = first coordinate of $path_i$
        if $X[i+j][1] == false$ **then**
            if $pathsAreSimilar(X[i][0], X[i+j][0])$ **then**
                increment $numberOfDuplicates$ for corresponding path in $resultList$
                set $X[i+j][1] = true$
            end if
        end if
        if $i + j + 1 < X.length()$ **then**
            $increment\, j$
        else
            $break$
        end if
    end while
end for

gorithmus 4 erstellten, Liste aller validen Pfade. Diese wird daraufhin anhand der Distanz zwischen der ersten Koordinate (dem Startpunkt) jedes Pfades zu dem Eckpunkt A (s. Tabelle 4) aufsteigend sortiert. Jedem Pfad der nun sortierten Liste $\text{Pfade}_{sortiert}$ wird ein Boolean *behandelt* (initial false) zugeordnet, der angibt, ob ein Pfad bereits behandelt wurde, also entweder bereits dem Resultatset hinzugefügt wurde oder bereits als Duplikat eines anderen Pfades erkannt wurde. Die Anzahl der erkannten Duplikate eines Pfades Pfad_a wird mit $Anz\_Duplikate(Pfad_a)$ bezeichnet und für jeden Pfad mit 0 initialisiert. Die Resultatliste $Resultat$[[Pfad_1, $Anzahl\_Duplikate$], ...] beinhaltet alle Pfade einfach und zusätzlich die Anzahl der zuvor entfernten Duplikate zu jedem Pfad. Es wird über $\text{Pfade}_{sortiert}$ iteriert und für jeden Pfad Pfad_a, für den *behandelt* false ist, werden alle darauf folgenden Pfade [Pfad_{a+1}, Pfad_{a+2}, ..., Pfad_{a+n}] als mögliche Duplikate betrachtet, für die gilt, dass die Distanz der ersten Koordinate des Pfades Pfad_a und des Pfades Pfad_{a+n} geringer gleich 19.9 Meter ist. Für jedes mögliche Duplikat wird daraufhin die $DTW\_Distanz$ bestimmt (s. Pseudocode Algorithmus 7) und geprüft ob Bedingung 3.4 gilt. Trifft dies zu, wird das mögliche Duplikat auch als tatsächliches Duplikat bewertet, weshalb für den dazugehörige Pfad Pfad_{a+n} *behandelt* auf true gesetzt werden kann und die Anzahl der erkannten Duplikate $Anz\_Duplikate(Pfad_a)$ inkrementiert wird. Mögliche Duplikate, die die Bedingung 3.4 nicht erfüllen, sind als keine tatsächlichen Duplikate von Pfad_a zu bewerten, weshalb hier *behandelt* weiterhin als false gesetzt bleibt. Sind alle Duplikate zu Pfad_a gefunden, wird zu diesem *behandelt* auf true gesetzt. Pfad_a wird daraufhin zusammen mit $Anz\_Duplikate(Pfad_a)$ in $Resultat$ gespeichert.

Abbildung 10: Zwei in dieser Arbeit als gleich zu bewertende Pfade (rot blau)

3.2.3 Resultat der Vorverarbeitung des GeoLife Datensatzes

Bezeichnung	Datentyp	Beschreibung
path_id	INTEGER	Eindeutiger Kennzeichner eines Pfades
path	BLOB	Abfolge der Koordinaten und zugehörigen Zeitstempel des Pfades
duplicates	INTEGER	Anzahl der Duplikate eines Pfades
distance	REAL	Gesamtlänge des Pfades
min_lat	REAL	Grenzt den Pfad in Richtung Süden ein
max_lat	REAL	Grenzt den Pfad in Richtung Norden ein
min_lon	REAL	Grenzt den Pfad in Richtung Westen ein
max_lon	REAL	Grenzt den Pfad in Richtung Osten ein
fog_nodes_trace	Text	Spur des Pfades - z.B. '1093,3510,21498' wobei jeder durch Komma separiert, Eintrag die ID eines Fog-Knoten referenziert

Tabelle 6: Schema der Pfad-Tabelle der Datenbank

Aus den ursprünglich 18.670 Pfaden des GeoLife Datensatzes wurden 61610 Pfade extrahiert, welche den Anforderungen aus Abschnitt 3.2.1 genügen. Unter diesen befinden 3663 Duplikate (z.B. Abbildung 10), womit insgesamt 57.961 verschiedene Pfade in die Simulation eingehen. Diese Pfade werden in einer SQLite Datenbank [7] persistiert. Die Datenbank besteht aus einer Pfad-Tabelle und einer Tabelle, in welcher die Positionen aller 27.160 Fog-Knoten als Breiten- und Längengrad gespeichert werden können. Einträge in der Pfad-Tabelle beinhalten neben den Koordinatensequenzen auch die Anzahl der Duplikate und die Länge der Pfade. Zusätzlich werden bei der Vorverarbeitung zu jedem Pfad der minimale Breitengrad, der maximale Breitengrad, der minimale Längengrad und der maximale Längengrad bestimmt, welche zu jedem Pfad in der Pfad-Tabelle gespeichert werden werden. Diese Extrema grenzen den Pfad in Form einer Bounding Box ein. So können später gezielt Pfade, deren Grenzen sich innerhalb eines bestimmten Bereichs befinden, mittels einer SQL-Abfrage abgefragt werden. Letztlich beinhaltet die Pfad-Tabelle eine Spalte *Fog_Nodes_Trace*, in welcher entsprechend des Namens die Reihenfolge der Fog-Knoten, mit welchen sich ein mobiles Endgeräte über den Pfad hinweg verbinden würde, persistiert werden. Die Einträge der Spalte *Fog_Nodes_Trace* sind abhängig von den Positionen der Fog-Knoten und werden somit stets nur gesetzt/aktualisiert, wenn auch diese Positionen initial gesetzt oder geändert werden.

Während der Vorverarbeitung der Pfade wurden Log-Dateien angelegt, deren Nutzen primär in der Behebung von Bugs und der Laufzeitoptimierung lagen. Hier zeigte sich, dass insbesondere die Nutzung des Numba JIT-Compilers [6] die Laufzeit deutlich verringern konnte, indem Funktionen die repetitive Rechenaufgaben durchführen, wie z.B. die Berechnung der *DTW_Distanz* oder der Haversine-Distanz, im nopython mode kompiliert werden, sodass diese daraufhin unabhängig vom Python-Interpreter ausgeführt werden können. Des weiteren wurde ein Verzeichnis generiert, in welchem zu jedem mehrfach auftretende Pfad und der zugehörigen Duplikate GPX-Dateien erstellt wurden. Auf Basis dieses Verzeichnis, konnten doppelte Pfade wie in Abbildung 10 dargestellt, visualisiert werden. So konnten die oben genannten Bedingungen für die Gleichheit von Pfaden und die Umsetzung unter Nutzung der *DTW_Distanz* stichprobenartig visuell verifiziert werden.

3.3 Implementation der Erweiterungen und Anpassungen an LocPrivFogSim

Der Fog-Simulator LocPrivFogSim [33] wird in einer weiteren Arbeit [28] zur Simulation von Risiken für Location-Privacy bei der Nutzung verschiedener Computation Offloading Strategien erweitert. Im Rahmen dieser Arbeit wurden Verbesserungen des Quellcodes und Bugfixes implementiert. Da die Verbesserungen und Bugfixes die weitere Implementierung begünstigen, stellt die entsprechende Codebasis [5] die Grundlage der weiteren Implementierung dar. Folgend wird grob skizziert, wie es einer Angreiferentität in LocPrivFogSim gelingen kann, den Pfad eines mobilen Endgerätes nachzuvollziehen. Daraufhin wird die Implementierung des in [33] beschriebenen Simulationsaufbaus beschrieben.

Nachfolgend werden Entscheidungen bezüglich des Software Designs dargelegt, woraufhin der Implementierungsprozess der nötigen Änderungen und Erweiterungen dargestellt wird.

3.3.1 Funktionsweise der Verfolgung der Pfade von mobilen Endgeräten

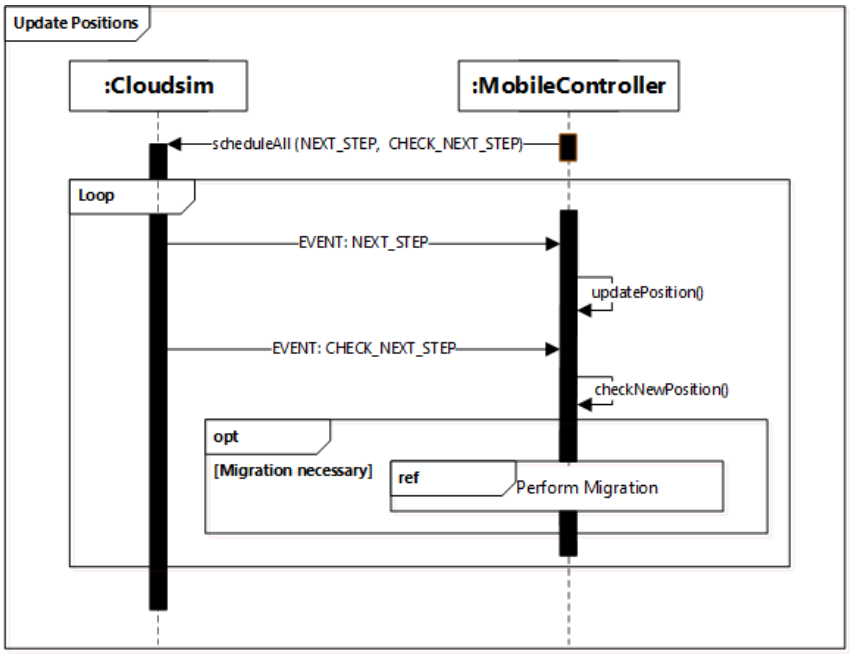


Abbildung 11: Sequenzdiagramm der Simulation der Fortbewegung eines mobilen Gerätes.

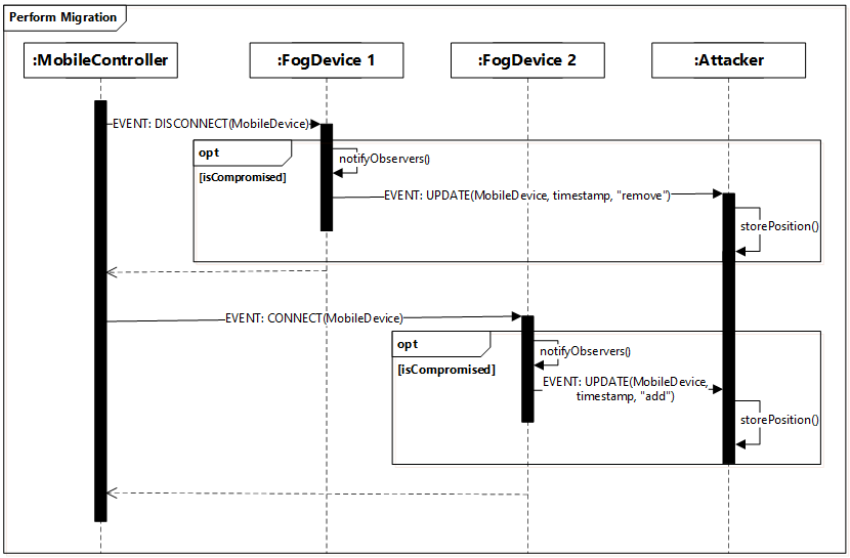


Abbildung 12: Sequenzdiagramm: Vereinfachte Darstellung der Migration und des Observer Patterns [33]

LocPrivFogSim basiert auf dem Simulator MobFogSim [25], welcher auf iFogSim[20] aufbaut. iFogSim ermöglicht die Simulation von Fog-Systemen. MobFogSim erweitert iFogSim um Funktionalitäten zur Simulation von mobilen Geräten innerhalb eines Fog-Systems. Dies beinhaltet die Migration der mobilen Geräte von Fog-Knoten zu Fog-Knoten entsprechend verschiedener Migrations-Strategien. In dieser Arbeit wird, wie in [33] angenommen, dass sich mobile Endgeräte stets mit dem nächstgelegenen Fog-Knoten verbinden, weshalb stets von der korrespondierenden Migrations-Strategie (niedrigste

Distanz) ausgegangen wird. Die Kommunikation zwischen den verschiedenen simulierten Geräten/Entitäten (Fog-Knoten, Endgerät, Cloudzentrum) wird mittels Events dargestellt. Diese Events können von Geräten selbst ausgelöst/konsumiert werden, aber auch in verschiedenen Controllern ausgelöst/konsumiert werden.

In Diagramm 11 wird hier die Abfolge der für die Simulation der Bewegung eines *MobileDevice* benötigten Events NEXT_STEP und CHECK_NEXT_STEP. Diese werden zu Beginn der Simulation von *MobileController* in einem definierten Zeitintervall gescheduled. Daraufhin werden diese entsprechend des Zeitintervalls von *CloudSim* ausgelöst und führen jeweils dazu, dass in *MobileController* die nächste Position des mobilen Endgerätes gesetzt wird und daraufhin in *checkNewPosition()* evaluiert wird. Resultiert hier, dass ein *MobileDevice* migriert werden muss, tritt die in Diagramm 12 dargestellte Sequenz ein. Dabei trennt *FogDevice 1* die Verbindung und *FogDevice 2* nimmt die Verbindung zu dem entsprechenden Gerät auf. Falls ein *FogDevice* kompromittiert ist, wird der *Attacker* entsprechend des Observer Patterns darüber informiert und worauf die so aufgenommenen Positionen der entsprechenden *MobileDevice* in der Methode *update()* verarbeitet und gespeichert werden.

3.3.2 Design und Implementierung der Änderungen und Erweiterungen

In diesem Unterkapitel wird beschrieben, welche Änderungen und Erweiterung an der vorherigen Version [9] des LocPrivFogSim-Simulators vorgenommen werden, um die Simulation der Pfade des GeoLife-Datensatzes zu ermöglichen. Dabei werden getroffene Designentscheidungen erklärt. Die in dem Klassendiagramm 13 dargestellten Änderungen und Erweiterungen dienen hierbei nicht nur der reinen Ermöglichung der Simulation der GeoLife-Pfade, sondern es gilt auch die Laufzeiten der einzelnen Simulationsdurchläufe soweit zu reduzieren, dass sich das gesamte Experiment in einer annehmbaren Zeit durchführen lässt.

Ein Simulationsdurchgang mit LocPrivFogSim lässt sich in Initialisierungsphase, Simulationsphase und Auswertungsphase aufteilen, welche folgend getrennt betrachtet werden.

Initialisierungsphase

Während der Initialisierungsphase werden alle für die Simulation benötigten Entitäten instanziiert und den korrespondierenden Controllern zugewiesen. Daraufhin werden die simulierten Verbindungen zwischen den Entitäten, bspw. zwischen mobilen Endgeräten und Fog-Knoten, aufgebaut. Im Kontext dieser Arbeit sind in dieser Phase insbesondere das **Laden der Pfade**, welche daraufhin den mobilen Endgeräten zugewiesen werden und das **Instanziieren und Positionieren der Fog-Knoten** von Interesse. Folgend werden die Initialisierungsphase betreffende Erweiterungen von bestehenden Klassen und Eigenschaften/Funktionalitäten von neu hinzugefügten Klassen beschrieben:

- **Neue Klasse *DBConnector*:** Die aufbereiteten Pfade des GeoLife-Datensatzes liegen in einer SQLite Datenbank vor (vgl. Abschnitt 3.2.3). Es gilt daher eine Verbindung zu dieser Datenbank zu ermöglichen und zu verwalten. Hierfür wird die Klasse *DBConnector* implementiert. Für das Laden der Pfade wird hier ermöglicht, dass einzelne Pfade anhand ihrer ID abgefragt werden können, alle Pfade als Liste abgefragt werden können oder alle Pfade, deren Bounding Box sich in einem bestimmten Bereich befinden als Liste abgefragt werden können. Abgefragte Einträge der *path*-Tabelle werden hier auf Instanzen der neu hinzugefügten Klasse *Path* übertragen. Da die Positionen aller Fog-Knoten in der Datenbank gespeichert werden, werden in *DBConnector* ebenfalls Methoden zur (initialen) Generierung und Speicherung der Positionen alle Fog-Knoten sowie der Abfrage dieser Positionen angeboten. Die Speicherung und Wiederverwendung der Positionen der Fog-Knoten stellt einen Unterschied zu der Funktionsweise der vorherigen Version von LocPrivFogSim dar, in welcher Positionen der Fog-Knoten stets in der Initialisierungsphase des Simulators zufällig bestimmt werden. Durch die Änderung entspricht das Verhalten des Simulator der Annahme aus Abschnitt 3.1.2, dass Fog-Knoten stationär sind. Zusätzlich kann der Zeitaufwand, der für die Initialisierung der Simulation nötig ist, reduziert werden.
- **Neue Klasse *Path*:** Instanzen der Klasse *Path* verfügen Attribute, welche den Spalten der Path-Datenbanktabelle 6 entsprechen. Hierbei wird die in der Datenbank als BLOB gepeicherte Se-

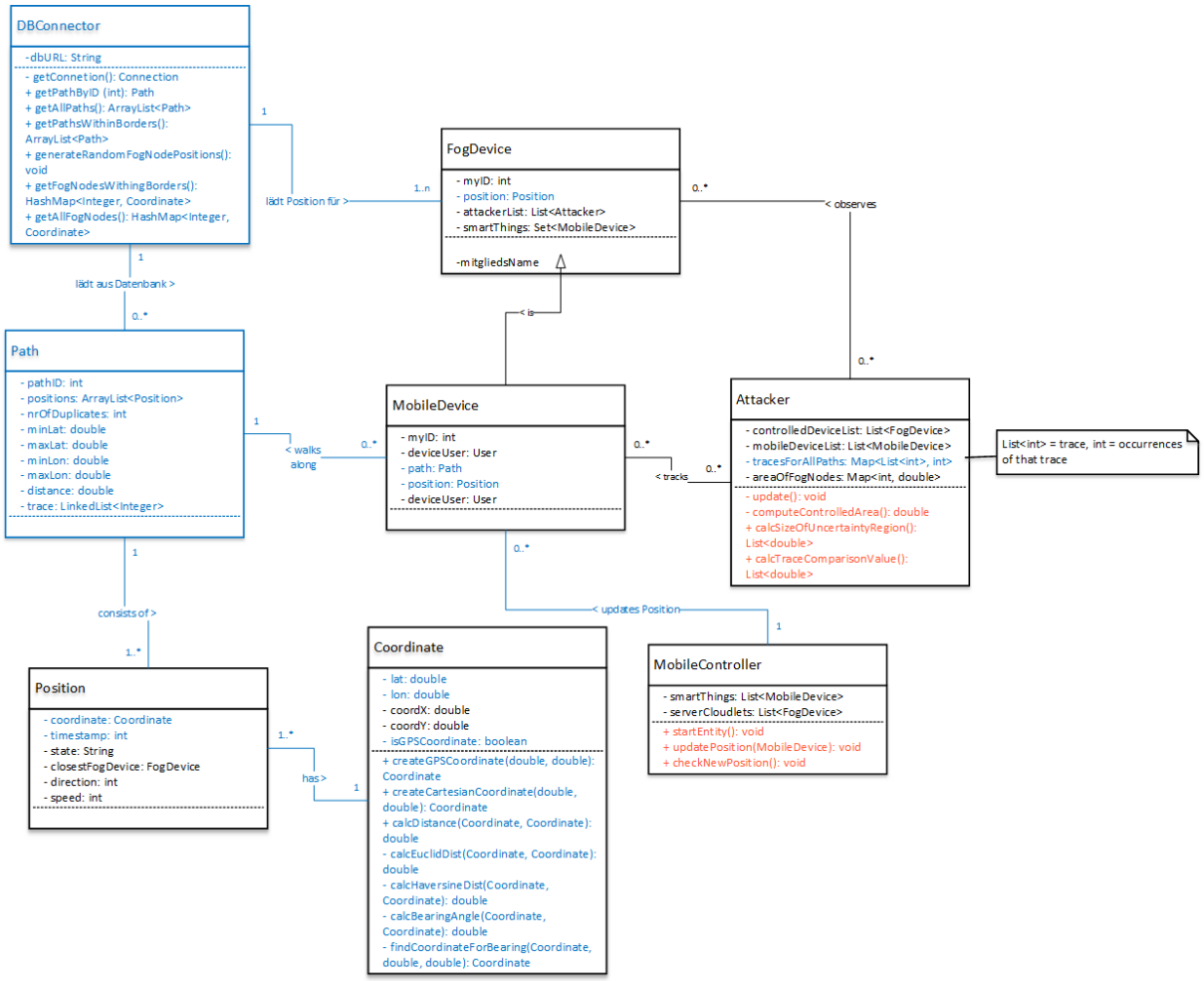


Abbildung 13: Ausschnitt des Klassendiagramms, wobei Änderungen Orange und neu Hinzugefügtes Blau markiert sind.

quenz der mit Zeitstempel versehenen Koordinaten auf eine generische Liste übertragen, welche Instanzen des Typ *Position* hält.

- **Erweiterung von *Position*:** Die Klasse *Position* wurde um das Attribut *timestamp* erweitert, sodass eine Position auf einem Pfad neben Eigenschaften wie Geschwindigkeit, Richtung und einer Koordinate auch eine zeitliche Dimension besitzt. Jede im Laufe der Datenbankabfragen erstellte *Position* Instanz hält initial nur eine Koordinate, repräsentiert durch eine *Coordinate* Instanz, und einen Zeitstempel (Integer). Während der tatsächlichen Simulation werden für *Position* Instanzen auch weitere Attribute wie bspw. Geschwindigkeit oder Richtung gesetzt.

- **Erweiterung von *Coordinate*:** Instanzen der Klasse *Coordinate* stellen in der vorherigen Version des Simulators stets ganzzahlige kartesische Koordinaten dar. In dieser Form genügt *Coordinate* daher nicht, um die aus Breiten- und Längengrad bestehenden GPS-Koordinaten des GeoLife-Datensatzes abzubilden. Daher wird *Coordinate* um die Attribute Breitengrad (als *lat*) und Längengrad (als *lon*) Erweitert. Zusätzlich wird das boolean Attribute *isGPSCoordinate* hinzugefügt, mit welchem *Coordinate* Instanzen kategorisiert werden. Es wurde davon abgesehen, von GPS-Koordinaten und kartesischen Koordinaten zu abstrahieren und so eine Vererbungshierarchie (Oberklasse *Coordinate*, Unterklassen *cartesianCoordinate*, *GPSCoordinate*) zu Implementieren. Dies ist darin Begründet, dass GPS-Koordinaten und kartesischen Koordinaten im Kontext des Simulators die einzigen relevanten Koordinatenarten sind und somit die Einführung von spezifischen *cartesianCoordinate*/*GPSCoordinate* Klassen lediglich dem Nutzen der besseren Codeverständnis dienen würde. Dieser Nutzen wird jedoch dadurch relativiert, dass an verschiedensten Stellen Fall-

unterscheidungen (GPS vs. kartesisch) vorgenommen werden müssten, welches einen gegenteiligen Effekt mit sich führen würde. Koordinaten Instanzen der jeweiligen Kategorie werden, angelehnt an das Factory Method Pattern, über die Methoden *Coordinate.createCartesianCoordinate()* und *Coordinate.createGPSCoordinate()* erhalten. Sämtliche Funktionalitäten, welche einer Fallunterscheidung der Koordinaten bedürfen, werden zentral von *Coordinate* angeboten. Insbesondere wird eine statische Methode *calcDistance(Coordinate c1, Coordinate c2)* angeboten, welche für kartesische Koordinaten die euklidische Distanz zurückgibt und für GPS-Koordinaten die Distanz über die Haversine-Formel 3.1 berechnet und zurückgibt. Damit die in *DBConnector* angebotene Generierung der Positionen der Fog-Knoten, welche entsprechend zu 3.1.2 in einem Gitter angeordnet sind, gelingen kann, bedarf es der Funktionalität zur Berechnung des Peilwinkels zwischen zwei Koordinaten. Hierfür wird in *Coordinate* die Methode *calcBearingAngle()* implementiert. Ebenso bedarf es der Funktionalität, der Peilung einer Zielkoordinate ausgehend von einer Startkoordinate, eines Peilwinkels und einer Distanz, welche in *calcCoordinateForBearingAndDistance()* implementiert wird.

Sind die Klassen *DBConnector*, *Path*, *Position* und *Coordinate* wie beschrieben hinzugefügt bzw. bearbeitet und davon betroffene Code-Segmente aus abhängigen Klassen angepasst, kann die Simulation bereits mit Pfaden des GeoLife-Datensatzes und den in der Datenbank hinterlegten Positionen der Fog-Knoten initialisiert werden.

Tests zeigten jedoch, dass die Initialisierungsphase für den Fall, dass alle 27.160 Fog-Knoten vollständig initialisiert werden, auch nach 15 Minuten nicht beendet wird. Da für das gesamte Experiment die Simulation für verschiedenste Konfigurationen initialisiert werden muss, gilt es daher die Initialisierung anzupassen, sodass die benötigte Zeit pro Initialisierung gering genug ist, um das Experiment in einem akzeptablen (/realistischen) Zeitraum durchzuführen. Weiteres Testen und die Nutzung des Debuggers zeigten, dass insbesondere das Instanziieren aller 27.160 Fog-Knoten und die Erstellung des Netzwerks zwischen allen Fog-Knoten viel Zeit benötigten. Bei der Dauer der Instanziierung der Fog-Knoten als *FogDevice* fällt vor allem die Registrierung jedes Fog-Knoten als *SimEntity* Instanz ins Gewicht. Nach der Registrierung als *SimEntity* können die Fog-Knoten Events erhalten. Im Kontext dieser Arbeit und des gewählten Simulationsaufbau sind nur die Events MobileEvent CONNECT und MobileEvent DISCONNECT relevant, da diese von dem Angreifer mitverfolgt werden können, um so den Pfad eines Gerätes nachzuvollziehen. Diese Events können nur von Fog-Knoten erhalten werden, welche sich in der Nähe des simulierten Pfades eines mobilen Endgerätes befinden. Daher ist es für den gewählten Simulationsaufbau genügend, wenn lediglich diese, sich in der Nähe eines simulierten Pfades befindenden und somit relevanten Fog-Knoten auch als *SimEntity* instanziiert werden. Da auch die übrigen irrelevanten Fog-Knoten durch den Angreifer kompromittiert sein könnten, dürfen diese jedoch nicht vollständig ignoriert werden. Insbesondere die Positionen dieser irrelevanter Fog-Knoten sind für die Berechnung des Trace Comparison Value und der Size of Uncertainty Region von Bedeutung. Daher sollen diese als „dummy“ *FogDevice* instanziiert werden und lediglich eine *Position* und eine ID als Attribute gesetzt bekommen. Die Zeit zur Erstellung des Netzwerks kann nun reduziert werden, indem nur Verbindungen zwischen den relevanten Fog-Knoten aufgebaut werden. Ein Fog-Knoten gilt dann als relevant, wenn die Distanz zwischen diesem und einer Position eines simulierten Pfades unter einem Schwellenwert von 1.5km liegt. Der Schwellenwert kann mit diesem Wert festgelegt werden, da aufgrund der Eigenschaften der Verteilung der Fog-Knoten zu jedem beliebigen Punkt innerhalb des Simulationsfeldes ein Fog-Knoten existiert, der maximal $\sqrt{2km}$ weit entfernt ist. Testdurchläufe zeigten, dass mit dieser Methode die benötigte Zeit zur Instanziierung alle Fog-Knoten für die längsten Pfade auf unter 10 Sekunden, und für durchschnittlich lange Pfade auf unter 2 Sekunden reduziert werden konnte.

Simulationsphase

Die oben beschriebenen Anpassungen an dem Simulator ermöglichen bereits, dass die Fortbewegung des mobilen Endgerätes durch das Fog-System simuliert werden kann.

- **Anpassungen in *MobileController*:** Dies wird ermöglicht indem in einem Intervall, welches in der vorherigen Version des Simulators auf 1 Sekunde gesetzt ist, die nächste Position des Endge-

rätes aus dessen Pfad in der Methode *updatePosition()* gesetzt wird. Die neue Position wird daraufhin in *checkNewPosition()* evaluiert, wobei insbesondere geprüft wird, ob eine Migration von einem Fog-Knoten zu einem anderen nötig ist. Die Intervalldauer wird in der Methode *startEntity* auf 1 Millisekunde reduziert. Die Notwendigkeit dieser Reduktion ist primär darin begründet, dass die benötigte Dauer eines Simulationsdurchlaufs möglichst minimal sein soll. So würde ohne diese Reduktion ein Simulationsdurchgang für einen Pfad mit 1000 Positionen, was ab einer Pfad-Gesamtlänge von ca. 5 km zu erwarten ist (da Koordinaten ca. alle 5 m, vgl. 2.4), zu einer Dauer von über 16 Minuten führen würde.

- **Anpassungen in *Attacker*:** In der Klasse *Attacker* gilt es im Kontext der Simulationsphase, die Methode *update()* anzupassen, sodass hier verfolgte Positionen von mobilen Endgeräte korrekt verarbeitet werden. Die Methode *update()* wird, entsprechend des implementierten Observer Patterns [33], ausgelöst wenn der *Attacker* von einem *FogDevice* über eine neue Verbindung zu einem mobilen Endgerät oder der Trennung einer solchen benachrichtigt (Methode *notifyObservers*) wird. Damit in *update()* zu der erstellen *Position* der korrekte Zeitpunkt gespeichert werden kann, wird *update()* um den Parameter *timestamp* erweitert. In vorheriger Version wurde hier stets die aktuelle Systemzeit als Zeitpunkt der verfolgten Position gespeichert, was nicht den tatsächlichen Zeitpunkten der GeoLife-Pfade entsprechen würde.

Auswertungsphase

Nachdem die Simulationsphase beendet ist und der *Attacker* die Informationen über die Pfade der simulierten mobilen Endgeräte gesammelt hat, werden Trace Comparison Value und Size Of Uncertainty Region berechnet, um den Erfolg des *Attackers* zu quantIfizieren. Die folgend beschriebenen Anpassungen werden in der Klasse *Attacker* umgesetzt und stellen jeweils Adaptionen der Verfahrensweisen der vorherigen Version dar.

Abbildung 14 Pseudocode: Prüfen ob ein Pfad zu einer Spur passt

Input recorded trace $t_{recorded}$ as $List < Integer >$
actual trace of some path t_{actual} as $List < Integer >$ ▷ where each Integer is ID of a fog-node
Output boolean *pathHasSameTrace*

```
1:  if mobile cannot be turned off then
2:      remove all elements from t_a that do not refer to a compromised fog-node
3:      if t_a == t_r then
4:          return true
5:      end if
6:  else
7:      int i = 0 , j = 0
8:      while i < t_recorded.length() and j < t_actual.length() do
9:          if t_recorded[i] equals t_actual[j] then
10:             i++
11:         end if
12:         j++
13:     end while
14:     if i == recorded.length() then
15:         return true
16:     end if
17: end if
18: return false
```

- **Der *Attacker* kennt alle möglichen Pfade:** Während in vorheriger Version alle 50 möglichen Pfade über die Methode *addPrecomputedPaths* geladen werden, werden diese nun über den Konstruktor in Form einer Liste weitergegeben. Da die Pfade des GeoLife-Datensatzes bereits in der

TestKlasse *TestExample4* vor der Instanziierung des *Attacker* geladen werden, wäre ein erneutes Laden der Pfade durch eine Funktion innerhalb von *Attacker* redundant.

- **Berechnung des Trace Comparison Values:** Die Berechnung des Trace Comparison Values in der Methode *calcTraceCompVal* entspricht dem in 2.2.7 beschriebenen Konzept. Für jedes simulierte mobile Endgerät, nimmt der *Attacker* die Spur als eine Liste *trace* auf. Die Annahme, ob sich mobile Endgeräte trennen dürfen, ist über den Wert von *MOBILE_CAN_BE_TURNED_OFF* gegeben. Ebenfalls existiert eine Map *actualTracesForPathIds*, welche für jeden Pfad (bzw. dessen ID) den in der Datenbank gespeicherten *actualTrace* (die tatsächliche Spur) hält. Dass ein *Attacker* zu jedem Pfad die tatsächliche Spur kennt, ist dabei eine bewusste Vereinfachung zur Reduktion der Laufzeit. Für jeden *actualTrace* wird entsprechend Algorithmus 14 geprüft, ob dieser zu *trace* passt. Passt ein *actualTrace* wird der Trace Comparison Value für *trace* inkrementiert.

- **Berechnung der Size Of Uncertainty Region:** Zur Berechnung der Size Of Uncertainty Region werden in *Attacker* entsprechend des simulierten Szenarios die Abdeckungsbereiche der kontrollierten Fog-Knoten bestimmt. Hierzu werden die Abdeckungsbereiche in der Methode *calcControlledArea* in der Klasse *Attacker* angenähert. Dabei werden, analog zu dem in [33] gewählten Verfahren, im Abstand von 100 Metern, Punkte über das Peking eingrenzende Feld in Form eines zu den Kanten des Feldes parallel orientierten Gitters verteilt. Zu jedem dieser insgesamt 26.951.40 Punkte wird der nächstgelegene kompromittierte Fog-Knoten bestimmt. Ist die Distanz zwischen Punkt und Fog-Knoten geringer als die maximale Reichweite des Fog-Knoten, wird der Punkt und damit die 100 qm dem Fog-Knoten zugeordnet. Nachdem alle Punkte betrachtet wurden, berechnet sich die angenommene Abdeckungsfläche eines Fog-Knoten aus der Anzahl der zugeordneten Punkte multipliziert mit 100 qm. Würde hier tatsächlich für jeden der 26 Millionen Punkten die Distanzen zu jedem der bis zu 27.160 Fog-Knoten (bei Kompromittierungsrate von 100%) berechnet werden, würde dies zu einer langwierigen Berechnungszeit führen. Daher werden sowohl die Gitterpunkte als auch die Fog-Knoten zuerst aufsteigend nach Distanz zu einem Eckpunkt des Simulationsfeldes (Peking) sortiert. Daraufhin wird über die Liste der Fog-Knoten, angelehnt an einen Sliding Window Algorithmus, mit zwei Indizes (index_links, index_rechts) iteriert. Die Indizes werden dabei stets so weit erhöht, dass nur Fog-Knoten betrachtet werden, für die gilt, dass sich die Distanz zwischen ihnen und dem Eckpunkt nicht mehr als 750 Metern von der Distanz zwischen Eckpunkt und betrachteten Gitterpunkt unterscheidet. Die Berechnung der Abdeckungsflächen kann auch nach dieser Optimierung noch mehrere Minuten benötigen. Daher soll diese Berechnung lediglich einmal bei Instanziierung des *Attacker* ausgeführt werden und nicht, wie in vorherhiger Version, bei jeder Berechnung der Size Of Uncertainty Region für die simulierten Pfade. Folgend wird der mit der Berechnung einhergehende Zeitaufwand daher zu der Initialisierungsphase gezählt. Die Berechnung der tatsächlichen Size Of Uncertainty Regions wird in der Methode *calcSizeOfUncertaintyRegion()* unter Berücksichtigung der berechneten Abdeckungsbereiche der kontrollierten Fog-Knoten vorgenommen. Die Funktionsweise von *calcSizeOfUncertaintyRegion()* unterscheidet sich nur geringfügig zur vorherigen Version - gibt nun jedoch eine Liste der Size Of Uncertainty Regions aller simulierter Pfade zurück, anstatt der der Size Of Uncertainty Region eines einzigen Pfades.

4 Simulation

In diesem Kapitel wird die im Kontext dieser Arbeit durchgeführte Simulation im Gesamten beschrieben. In Unterkapitel 4.1 werden das gewählte Testsystem beschrieben und die Durchführung der Simulation dargelegt, mit welchen das Experiment aus [33] für reale Mobilitätsdaten wiederholt wird. Daraufhin werden die mit diesem Testsystem erhaltenen Ergebnisse mit den Befunden aus [33] verglichen und Unterschiede und Gemeinsamkeiten diskutiert.

4.1 Testsystem und Simulationsdurchführung

Zur Durchführung des Experimentes werden verschiedene Konfigurationen für den in Abschnitt 3.1 beschriebenen Simulationsaufbau simuliert. Konfigurationen beinhalten dabei:

1. das gewählte Szenario (vgl. 2.2.5)
2. die Kompromittierungsrate der Fog-Knoten
3. die Anzahl der simulierten mobilen Endgeräte

Zur Simulation der verschiedenen Konfigurationen wird folgendes, in der Main-Klasse *TestExample4* definiertes, Testsystem genutzt:

Das Simulationsfeld wird entsprechend der Koordinaten aus Abschnitt 3.1.1 initialisiert. Über dieses werden die 27.160 Fog-Knoten , entsprechend ihrer in der Datenbank gespeicherten Koordinaten, verteilt. Die Positionen der Fog-Knoten sind so gleichbleibend, soweit nicht explizit eine Umverteilung der Positionen innerhalb der Datenbank vorgenommen wird. Dies könnte bei Bedarf beispielsweise über die Methode *setFogNodePositions()* in der Testklasse *TestPathLoading* geschehen. Daraufhin wird entsprechend des Simulationsparameters *Anzahl mobile Endgeräte* eine Anzahl mobiler Endgeräte erstellt. Jedem dieser mobilen Endgeräte wird ein zufälliger Pfad aus der Datenbank zugewiesen. Die Wahrscheinlichkeit, dass ein Pfad dabei gewählt wird, ist von der Anzahl der Duplikate des Pfades abhängig. So ist die Wahrscheinlichkeit zur Auswahl eines Pfades mit vielen Duplikaten entsprechend größer, als die Auswahl eines Pfades ohne Duplikate. Weitere Eigenschaften der Simulationsentitäten (Fog-Knoten, mobile Endgeräte etc.), wie CPU Ressourcen o.ä. haben keinen Einfluss auf die Ergebnisse des in dieser Arbeit durchgeführten Experimentes und werden so aus dem Testsystem von [33] übernommen. Die neben *Anzahl mobile Endgeräte* benötigten Simulationsparameter werden in Tabelle 7 in Kurzform beschrieben.

Parameter	Einheit	Beschreibung
Szenario	int	Bestimmt Szenario: 1 = Szenario 1.1: kennt Lage kontrollierter Fog-Knoten, Endgerät immer Verbunden 2 = Szenario 2.1: kennt Lage aller Fog-Knoten, Endgerät immer Verbunden 3 = Szenario 1.2: kennt Lage kontrollierter Fog-Knoten, Endgerät kann Verbindung trennen 4 = Szenario 2.2: kennt Lage aller Fog-Knoten, Endgerät kann Verbindung trennen
Kompromittierungsrate	double	Angegeben in % Bestimmt die Anzahl der kompromittierten Fog-Knoten
Seed 2	int	Seed zur Generierung zufälliger Zahlen zur Initialisierung der Simulation. Benötigt z.B. bei der Erstellung des mobilen Endgerätes
Seed 3	int	Seed zur Generierung zufälliger Zahlen für Attacker.
Anzahl mobile Endgeräte	int	Bestimmt die Anzahl der simulierten mobilen Geräte (und damit die Anzahl der ausgewählten Pfade). Geringste Laufzeiten bei einem Wert von ca. 70

Tabelle 7: Beschreibung der Simulationsparameter

Das Projekt wird als jar verpackt. Damit diese fehlerfrei ausführbar ist, muss sich die gefüllte Datenbank Datei (*geoLifesPaths.db*) in dem gleichen Verzeichnis befinden. Während der tatsächlichen Experimentdurchführung, wird die jar über ein Shell Script (*run_geo_life_paths*) für jede Konfiguration und für zufällige Seeds (vgl. 7) ausgeführt. Die Kompromittierungsrate wird dabei ausgehend von 5 % stets um 5 % bis 100 % erhöht. Für die Gesamtanzahl der für ein Szenario und eine Kompromittierungsrate simulierten mobilen Endgeräte (und Pfade) wurde 1000 bestimmt. Hierzu wurden Durchschnittwerte für Trace Comparison Value und Size of Uncertainty Region für die Anzahlen 10, 100, 1000 bestimmt und verglichen. Es zeigte sich, dass die Durchschnittswerte bereits bei 100 simulierten mobilen Endgeräten weitestgehend konstant blieben und bei 1000 keine Ausreißer mehr ersichtlich waren.

Um eine Durchführung des gesamten Experiments in akzeptabler Zeit zu ermöglichen wird die Anzahl der simulierten Pfade pro Simulationsdurchgang auf 70 bestimmt. Für jede Konstellation (Kompromittierungsrate, Szenario) werden daher 15 Simulationsdurchläufe durchgeführt, womit jeweils 1050 Pfade simuliert werden. Die Anzahl 70 wurde hier bestimmt, indem die durchschnittliche Simulationsdauer pro Pfad (Gesamtdauer/Anzahl) für verschiedene Pfadanzahlen bestimmt wurde. Bei 70 Pfaden war diese minimal und mit einem Wert von 2.7 Sekunden pro Pfad und stelle gegenüber des Ausgangswertes (bei Pfadanzahl = 1) von 105 Sekunden pro Pfad eine deutliche Reduktion der Laufzeit dar. Die Durchführung des gesamten Experiments bedarf bei ca 2.7 Sekunden pro Pfad ungefähr 32 Stunden. Durch Aufteilung des gesamten Experiments, beispielsweise sodass die 4 Szenarien getrennt in 4 Shell Scripts betrachtet werden, welche parallel und auf mehreren Maschinen ausgeführt werden können, kann die benötigte Zeit weiter reduziert werden. Die Ergebnisse der einzelnen Simulationsergebnisse werden in eine CSV-Datei geschrieben (*results.csv*) und können nach Vollendung des Experimentes ausgewertet werden.

4.2 Vergleich der Simulationsergebnisse und Diskussion

In diesem Kapitel werden die mit dem in Unterkapitel 4.1 beschrieben Testsystem erhaltenen Ergebnisse dargelegt und mit den Ergebnissen aus [33] verglichen. Dazu werden die Ergebnisse für Trace Comparison Value und Size Of Uncertainty Region getrennt betrachtet.

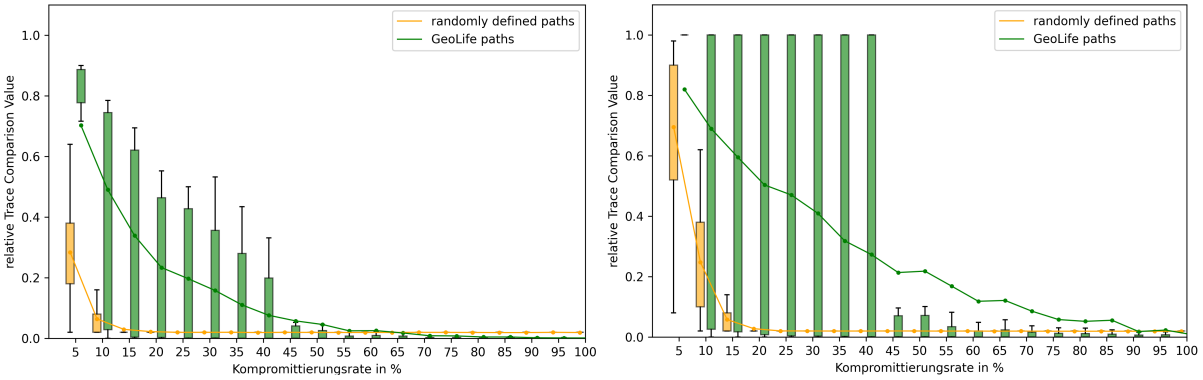
4.2.1 Trace Comparison Value

Rate in %	5	10	15	20	25	30	35	40	45	50	55	60	65	70	75	80	85	90	95	100
arith. Mittel[1]	40723	28410	19641	13543	11400	9150	6400	4404	3337	2651	1436	1473	1025	496	462	256	254	107	77	70
arith. Mittel[2]	47537	39982	34487	29174	27255	23716	18455	15815	12373	12633	9763	6856	7018	4966	3350	3009	3219	1048	1335	454

Tabelle 8: Wertetabelle der Ergebnisse (gerundet auf Ganzzahlen). arith.
Mittel[1] = *stets verbunden*, arith.
Mittel[2] = *Verbindung darf getrennt werden*

Die Berechnung des Trace Comparison Values ist unabhängig von der Größe der angenommenen Abdeckungsbereiche der kompromittierten Fog-Knoten (vgl. 2.2.7). Lediglich das Verbindungsverhalten, ob sich ein mobiles Endgerät von dem Fog-System trennen darf (oder auch nicht) spielt bei der Berechnung des Trace Comparison Value eine Rolle. In Abbildung 15 werden die im Kontext dieser Arbeit erhaltenen Ergebnisse bezüglich des Trace Comparison Value neben den Ergebnissen aus [33] dargestellt. Dazu werden die totalen Werte des Trace Comparison Values (s. Tabelle 8) jeweils relativ zu der Gesamtanzahl der Pfade betrachtet. Folgend werden die Ergebnisse anhand einiger Aspekte verglichen:

- **Auswirkung des angenommenen Verbindungsverhaltens:** Gemein haben die Ergebnisse aus [33] mit denen aus dieser Arbeit, dass ein Angreifer für eine Spur stets mehr Pfade ausschließen kann, wenn er annimmt, dass das mobile Endgerät stets verbunden sein muss. Die Ergebnisse [33] deuten darauf, dass für den Fall *Verbindung darf getrennt werden* der Trace Comparison Value



(a) Mobile Engeräte stets verbunden

(b) Mobile Engeräte können Verbindung trennen

Abbildung 15: Vergleich des relativen Trace Comparison Value zu den Ergebnissen aus [33]

lediglich um 5% verschoben ist. Der Angreifer muss in diesem Fall somit 5% mehr Fog-Knoten kontrollieren, um gleich erfolgreich zu sein. Dieser Schluss deckt sich nicht mit den Ergebnissen dieser Arbeit (vgl. grüne Kurven in 15. Hier zeigt sich, dass die Kurve des Trace Comparison Values für den Fall *Verbindung darf getrennt werden* deutlich langsamer abfällt als für den Fall *stets verbunden*.

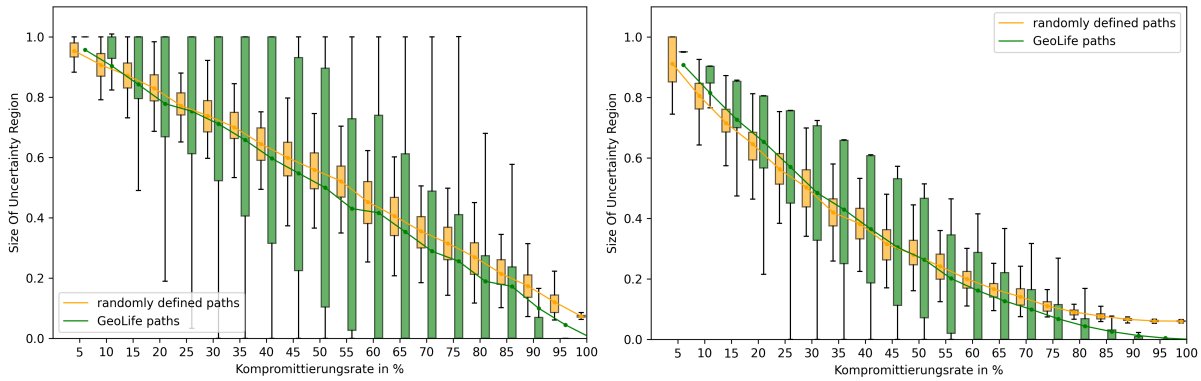
- **Vergleich der relativen Werte:** Unabhängig des angenommenen Verbindungsverhaltens, liegen die relativen Trace Comparison Values für die GeoLife-Pfade über weite Strecken deutlich über den Werten aus [33]. Bereits bei einer Kompromittierungsrate von 15% bzw. 20% werden für den Experimentaufbau aus [33] das Minimalniveau von 0.02 erreicht (entspricht $\frac{1}{50}$), welches für die Pfade des GeoLife-Datensatzes erst bei ca. 60% bzw. 90% erreicht wird. Dabei ist zu beachten, dass bei diesem Niveau im Mittel immer noch ca. 1200 Pfade des GeoLife-Datensatzes in Frage kommen würden. Selbst bei 100% kontrollierten Fog-Knoten ist es dem Angreifer im Mittel nicht möglich die Spur auf einem Pfad eindeutig zuzuordnen. Für diesen deutlichen Widerspruch scheinen insbesondere zwei Erklärungen schlüssig:
  1. **Hohe Komplexität der Pfade in [33]:** Eine Erklärung dieses deutlichen Widerspruchs zu den Ergebnissen aus [33] könnte durch die unterschiedlichen Beschaffenheiten der gewählten Pfade in Kombination mit der Dichte der Fog-Knoten gegeben sein. In jedem der in [33] für die Simulation definierten Pfad wird sich mindestens mit 11 und im Mittel mit 15 unterschiedliche Fog-Knoten verbunden. Aus den in der Datenbank gespeicherten Längen der GeoLife-Pfade lässt sich ablesen, dass der Mittelwert der Pfadlängen des GeoLife-Datensatzes bei ca. 1km liegt. Es lässt sich demnach vermuten, dass sich im Mittel mit nur einem oder wenigen Fog-Knoten über einen Pfad hinweg verbunden wird. Pfade aus [33] haben somit im Mittel deutlich komplexere Spuren, welche diese leichter voneinander abgrenzen.
  2. **Cluster der GeoLife-Pfade in Pekings Stadtzentrum:** Wie in der Heatmap in Abbildung 2 ersichtlich wurden die meisten Pfade des GeoLife-Datensatzes im Zentrum Pekings aufgenommen. Es lässt sich annehmen, dass hierbei verschiedenste Pfade zu ähnlichen oder sogar gleichen Spuren führen. So ist es dem Angreifer nur schwer möglich, Spuren auf einzelne Pfade zurückzuführen.
- **Unterschiede in der Streuung:** Insbesondere für den Fall *Verbindung darf getrennt werden* (rechts in Abbildung 15) zeigt sich, dass hier 3 Bereiche ausgemacht werden können:
  - **5% Kompromittierungsrate:** Ausschließlich für eine Kompromittierungsrate von 5% ist die Streuung des Trace Comparison Values für die Pfade des GeoLife-Datensatzes kleiner als die Streuung der Ergebnisse aus [33]. Erklärt könnte dies damit werden, dass der Angreifer bei einer solch geringen Anzahl an kontrollierten Fog-Knoten nur in absoluten Ausnahmefällen überhaupt eine Spur zu einem mobilen Endgerät erhält. So fallen diese Ausnahmefälle

im Vergleich zu den übrigen Kompromittierungsraten kaum ins Gewicht.

- **10-40% Kompromittierungsrate:** Die Streuung des Trace Comparison Values fällt für die GeoLife-Pfade stets deutlich größer aus als für die Ergebnisse aus [33]. Dies lässt sich darauf zurückführen, dass die Spuren zu den Pfaden aus [33] deutlich komplexer sind und somit auch bei geringen Kompromittierungsraten schon viele Pfade ausgeschlossen werden können. Eine weitere Auffälligkeit besteht darin, dass während für *stets verbunden* die Streuung mit ansteigender Kompromittierungsrate sinkt, sie für *Verbindung darf getrennt werden* weitestgehend konstant bleibt. Eine Erklärung dafür könnte wie folgt aussehen: Wie oben beschrieben sind die Spuren zu den GeoLife-Pfaden meist von geringer Komplexität (da im Mittel sich mit nur einem bis wenigen Fog-Knoten verbunden wird). Für solche Spuren ist der Fall, dass keiner der Fog-Knoten kompromittiert ist, daher auch bei steigender Kompromittierungsrate (bis ca. 40%) noch so wahrscheinlich (1 Fog-Knoten 60%, 2 Fog-Knoten 36%, ...), dass er deutlich ins Gewicht fällt. In diesem Fall könnte der Angreifer für *stets verbunden* alle Pfade ausschließen, auf denen sich das mobile Endgerät mit einem kontrollierten Fog-Knoten verbunden hätte. Die Anzahl der Pfade die so ausgeschlossen werden können, steigt mit der Kompromittierungsrate. Währenddessen kann der Angreifer für *Verbindung darf getrennt werden* in diesem Fall gar keine Pfade ausschließen, weshalb hier ein relativer Trace Comparison Value von 1.0 noch häufig auftritt.
- **45-100% Kompromittierungsrate:** Bei 45% fällt die Streuung sowohl für *stets verbunden* als auch *Verbindung darf getrennt werden* stark ab. Dies könnte darin begründet sein, dass ab ca. 45% der Fall, dass sich auf einem Pfad mit gar keinem kompromittierten Fog-Knoten mehr verbunden wird, zu unwahrscheinlich ist um deutlich ins Gewicht zu fallen. Mit ansteigender Kompromittierungsrate sinkt die Streuung nun, da der Angreifer stets mehr Pfade ausschließen kann und es somit immer seltener wird, dass deutlich nach oben abweichende Trace Comparison Values berechnet werden.

Insgesamt lässt sich festhalten, dass die Ergebnisse dieser Arbeit bezüglich des Trace Comparison Values ein deutlich anderes Bild als die Ergebnisse aus [33] darstellen. Insbesondere die geringe Komplexität der realen Pfade, scheint es dem Angreifer zu erschweren die Pfade voneinander unterscheiden zu können.

4.2.2 Size Of Uncertainty Region



(a) Szenario 1.0 ebenfalls repräsentativ für Szenarien 1.0 und 1.1

(b) Szenario 2.0

Abbildung 16: Vergleich der Size Of Uncertainty Region zu den Ergebnissen aus [33]

- **Vergleich der mittleren Werte:** Wie in Abbildung 16 zu sehen, unterscheiden sich der Verläufe der mittleren Höhen der Size Of Uncertainty Region zwischen den in dieser Arbeit erhaltenen Ergebnissen und den Befunden aus [33] nur geringfügig. Diese geringen Differenzen lassen sich vermutlich primär statistischen Schwankungen zuschreiben. Es ist zu beachten, dass die Size Of Uncertainty Region für den Experimentaufbau aus [33] mit 20 Fog-Knoten ein Minimum von ca.

0.05 (also $\frac{1}{20}$) nicht unterschreiten kann, während das Minimum für die GeoLife-Pfade (mit 27.160 Fog-Knoten) erst bei ca. 0.000037 (ungefähr $\frac{1}{27.160}$) erreicht wird. Daher sinkt die Size Of Uncertainty Region für die GeoLife-Pfade sichtbar weiter ab, wobei daraus keine Schlussfolgerungen gezogen werden können. Da sich die Kurven der Werte für Size Of Uncertainty Region derart ähnlich sind, liegt die Vermutung nahe, dass die Auswahl der Pfade hier keinen Einfluss auf die mittleren Werte hat.

- **Vergleich der Streuung:** Ähnlich wie für den Trace Comparison Value unterscheiden sich die Ergebnisse jedoch auch hier deutlich in der Streuung der Size Of Uncertainty Region, wobei erneut die Streuung der GeoLife-Pfade deutlich über der Streuung der zufällig gewählten Pfade liegt. Auch hier lässt sich vermuten, dass dies damit zu begründen ist, dass sich auf vielen Pfaden des GeoLife-Datensatzes mit nur einer geringen Anzahl an Fog-Knoten verbunden wird. Analog zur Begründung bezüglich der Streuung in Abschnitt 4.2.1 könnte auch hier die Begründung darin liegen, dass die Extrema deutlich wahrscheinlicher für die GeoLife-Pfade eintreten als für die Pfade aus [33]. Auch zeigt sich, dass im Vergleich zu Szenarien 1.0, 1.1 und 2.1 die Streuung für Szenario 2.0 insbesondere in Richtung des oberen Endes deutlich geringer ausfällt. Dies lässt sich damit begründen, dass es dem Angreifer für Szenario 2.0 auch dann möglich ist die Position des mobilen Endgerätes einzugrenzen, wenn das mobile Endgerät nicht mit einem kompromittierten Fog-Knoten verbunden ist. In diesem Fall, kann der Angreifer die Position des Endgerätes auf die Fläche aller nicht kompromittierter Fog-Knoten eingrenzen. Da diese Fläche mit steigender Kompromittierungsrate sinkt, sinkt auch der maximal mögliche Wert der Size Of Uncertainty. Der maximal mögliche Wert entspricht dabei erwartungsgemäß, wie die oberen Antennen der Box-Plots vermuten lassen, stets $max \approx 1 - Kompromittierungsrate$.

- **Vergleich der unteren Box-Plot Antennen:** Wie bereits beschrieben ist die Streuung für die Pfade des GeoLife-Datensatzes deutlich größer als die zufälligen Pfade aus [33]. Hier ist insbesondere die Streuung in Richtung der unteren Kante interessant, da diese Rückschlüsse darauf zulässt, ab welcher Kompromittierungsrate sehr niedrige Werte für die Size Of Uncertainty Region vermehrt auftreten können. In Abbildung 16 lässt sich an den unteren Antennen der Box-Plots ablesen, dass dies für die GeoLife-Pfade bereits ab ca. 25% vermehrt der Fall ist. Im Vergleich dazu treten derartige Erfolgsfälle für den Simulationsaufbau aus [33] erst bei ca. 70% ein. Als Begründung lässt sich hier weiterhin die geringe mittlere Anzahl der Fog-Knoten auf den Pfaden des Geo-Life Datensatzes anbringen (vgl. Ausführungen in Abschnitt 4.2.1).

Zusammenfassend lässt sich sagen, dass ein Angreifer zwar unabhängig der Wahl der Pfade (ob zufällige [33] oder reale Pfade) einen gleichbleibenden mittleren Erfolg erwarten kann, er jedoch bei realen Pfaden bereits bei niedrigen Kompromittierungsraten vereinzelte große Erfolge vermerken könnte.

5 Diskussion

Die geschilderten Ergebnisse bezüglich des Einfluss der realen Mobilitätsdaten unterliegen dabei jedoch verschiedenen Validitätsrisiken, die folgend ausgeführt werden. Validitätsrisiken, welche den aus [33] übernommenen Annahmen entspringen (s. 2.2.6), werden in dieser Arbeit dabei nicht erneut diskutiert.

- **Einfluss der realen Pfade kann nicht isoliert betrachtet werden:** Neben der Integration der Pfade des GeoLife-Datensatzes, wurden weitere Änderungen an dem Experimentaufbau vorgenommen. So umfasst das Simulationsfeld nun die Stadt Peking (Abbildung 3) und unterscheidet sich so drastisch von dem in [33] gewählten 50 x 50 großen Simulationsfeld. Desweiteren wurde mit 27.160 eine Anzahl an Fog-Knoten geschätzt, die weit über den in [33] simulierten 20 Fog-Knoten liegt. Aufgrund dieser zusätzlichen Änderungen lassen sich die Simulationsergebnisse nicht auf die Integration der realen Pfade selbst zurückführen. Insbesondere lässt sich vermuten, dass die Anzahl der Fog-Knoten einen ähnlich starken Einfluss auf die Werte des Trace Comparison Values hat wie die Pfade. Dies ist darin begründet, dass wie in Abschnitt 4.2.1 beschrieben, die Dichte der Fog-Knoten die Komplexität der Spuren (zu wie vielen Fog-Knoten wird sich über einen Pfad hinweg verbunden?) beeinflusst. Um hier den Einfluss der Pfade präziser herausstellen zu können, könnte ein Experiment konzipiert werden, welches dem aus [33] entspricht, mit der Ausnahme, dass die Pfade des GeoLife-Datensatzes auf das 50 x 50 große Simulationsfeld gemappt werden.
- **Falsch bestimmte Paramter des Experimentaufbaus:** Der gewählte Experimentaufbau wird in Kapitel 3.1 ausgeführt und begründet. Hierbei mussten verschiedene Parameter bestimmt werden, welche bei fehlerhafter Wahl die Ergebnisse verfälschen könnten:
  1. **Anzahl der Fog-Knoten:** Wie bereits beschrieben, lässt sich vermuten, dass die Anzahl der simulierten Fog-Knoten insbesondere den Trace Comparison Value beeinflusst. Die Anzahl der Fog-Knoten musste in dieser Arbeit genähert werden, da in der Literatur keine ähnlichen Experimente gefunden wurden. Dazu wurde sich an der Anzahl an 5G-Sendemasten orientiert, die für eine vollständige Abdeckung des Feldes benötigt werden würde. In der Realität trifft die damit verbundene Vereinfachung, dass Fog-Knoten immer direkt an 5G-Sendemasten gebunden sind, nicht zu. Da es hier an ähnlichen Experimenten in der Literatur mangelt, die die bestimmte Anzahl (27.160) widerlegen oder decken könnten, könnte der Einfluss der Anzahl der Fog-Knoten experimentell erprobt werden. Dazu könnte eine geringe Anzahl (möglichst repräsentativer) Pfade des GeoLife-Datensatzes ausgewählt und für diese die Simulation für verschiedene Anzahlen von Fog-Knoten durchgeführt werden.
  2. **Verteilung der Fog-Knoten:** Die Fog-Knoten wurden in dieser Arbeit gitterförmig über das Simualtionsfeld verteilt (vgl. Abschnitt 3.1.2). Die Verteilung wurde derart gewählt, um der Annahme *Fog-Knoten decken das gesamte System ab* (vgl. Annahmen 2.2.6) zu entsprechen. Dabei wurde von in der Realität für die Verteilung von Fog-Knoten relevanten Aspekten (z.B. Auslastung) abstrahiert, da die explizite Betrachtung dieser über den Betrachtungsgegenstand dieser Arbeit hinausreichen würde. Die Betrachtung einer oder mehrerer realistischerer Verteilungen der Fog-Knoten könnte die Grundlage einer weiterführenden Arbeit darstellen.
  3. **Qualitätsanforderungen an Pfade:** In Abschnitt 3.1.3 werden Anforderungen an die für die Simulation brauchbaren Pfade gestellt. Die hier bestimmten Anforderungen beeinflussen die Menge der möglicherweise simulierten Pfade. Falls die Anforderungen dabei zu unpräzise gewählt werden, könnte dies zur Folge haben, dass fehlerhafte Pfade nicht entdeckt wurden und so die Ergebnisse beeinflussen. Werden die Anforderungen hingegen zu präzise/restriktiv gewählt, könnte dies dazu führen, dass nur ähnliche Pfade diesen Anforderungen genügen. Die Vielfältigkeit der möglicherweise simulierten Pfade würde so eingeschränkt werden. Auf Grund der hohen Anzahl der erhaltenen Pfade (ca. 58.000) konnte die Wahl der Anforderungen nicht ausführlich getestet werden. Jedoch wurden einerseits Pfade stichprobenartig

grafisch verglichen (z.B. Abbildung 10) und zusätzliche Informationen, wie Gesamtlänge, erhoben und in der Datenbank gespeichert. Sowohl die stichprobenartigen Vergleiche als auch die Betrachtung der in der Datenbank gespeicherten Informationen ließen nicht vermuten, dass die Anforderungen hier drastisch fehlerhaft gewählt wurden.

4. **Anzahl der simulierten Pfade pro Simulationsdurchgang:** Die Anzahl der simulierten Pfade pro Simulationsdurchgang konnte nicht statistisch hergeleitet werden, da die Anzahl der möglichen Kombination bei
   - 27.160 Fog-Knoten
   - ca. 58.000 Pfade

   zu hoch ist um hier beispielsweise eine statistisch relevante Stichprobengröße zu bestimmen. Daher konnte hier lediglich eine Zahl erprobt werden, wodurch eine fehlerhafte Bestimmung dieser nicht auszuschließen ist. Bei einer zu gering gewählten Anzahl könnten Ausreißer die Ergebnisse verfälschen. Obwohl die gewählte Anzahl (mindestens 1000 pro Simulationsdurchgang) nicht statistisch bestätigt werden kann, könnte der Verlauf der Size Of Uncertainty Region (s. Abbildung 16), welcher sehr ähnlich zu den Befunden aus [33] ist, ein Indiz dafür sein, dass die Anzahl auch hier nicht entscheidend fehlerhaft gewählt wurde.

- **Implementierungsfehler:** Dies beinhaltet einerseits mögliche Implementierungsfehler, die bereits vor Anpassung/Erweiterung von LocPrivFogSim enthalten waren und mögliche Fehler, die während der Implementierung der Anpassungen/Erweiterungen hinzugekommen sein könnten. Die Korrektheit der Experimentergebnisse ist dabei insbesondere davon abhängig, dass:
  1. die Fortbewegung des mobilen Endgerätes korrekt simuliert wird und Migrationen entsprechend durchgeführt werden, und
  2. die Berechnungen für Trace Comparison Value und Size Of Uncertainty Region korrekt sind.

  Um Ersteres zu testen, wurden Konsolenausgaben an den Programmstellen, welche zu Fortbewegung und Migration von mobilen Endgeräte gehören, hinzugefügt. Die Konsolenausgaben für verschiedene Pfade konnten daraufhin mit den entsprechenden Einträgen (Abfolge der Koordinaten, Spur) innerhalb der Datenbank verglichen werden. Es wurden für keinen der so getesteten Pfade Fehler beobachtet. Um die Fehlerwahrscheinlichkeit für Zweiteres zu reduzieren, wurden genutzte Algorithmen (z.B. Pseudocode 14) in entsprechenden Testklassen erprobt. Auch in den Algorithmen verwendete Hilfsmethoden, wie die Berechnung der Haversine-Distanz (vgl. 3.1), wurden hier mit Modultests getestet.

6 Fazit

Das Ziel dieser Arbeit besteht darin den Einfluss realer Mobilitätsdaten auf die Simulation von Location-Privacy-Angriffen herauszustellen. Dazu wurde der Fog-Simulator LocPrivFogSim erweitert und angepasst, sodass es mit diesem möglich war ein an [33] angelehntes Experiment unter Nutzung der realen Pfade des GeoLife-Datensatzes durchzuführen. Hierzu wurden aus dem GeoLife-Datensatz Pfade heraus gefiltert, welche verschiedenen Qualitätsansprüchen genügen (s. Abschnitt 3.2.1) und Duplikate entfernt. Daraufhin wurden die so erhaltenen Pfade in eine Datenbank überführt, auf welche LocPrivFogSim Zugriff hat (vgl. Abschnitt 3.2.3). Damit die Simulation für die aus der Datenbank geladenen Pfade möglich war, musste die Daten-Repräsentation der Pfade innerhalb des Simulators erweitert und angepasst werden (s. Abschnitt 3.3.2). Neben diesen Änderungen, die auf das grundsätzliche Ermöglichen der Simulation abzielten, waren weitere Anpassungen nötig, um die Laufzeit der einzelnen Simulationsdurchgänge zu reduzieren, sodass eine Experimentdurchführung in einem akzeptablen Zeitraum möglich war. Die Auswertung der Ergebnisse (s. 4.2) zeigte, dass sich die erhaltenen mittlere Werte der Size Of Uncertainty Region mit den Ergebnissen aus [33] deckten. Jedoch war es dem Angreifer für die Pfade des GeoLife-Datensatzes vereinzelt bei niedrigen Kompromittierungsraten möglich, die Position eines mobilen Endgerätes präzise zu bestimmen, während solche Fälle nicht in dem Experiment aus [33] auftraten. Insgesamt scheint es jedoch, als hätte die Auswahl der simulierten Pfade (ob zufällig oder reale Pfade) nur einen geringfügigen Einfluss auf den Gesamterfolg des Angreifers. Die Ergebnisse bezüglich des Trace Comparison Values unterscheiden sich hingegen drastisch von den Ergebnissen aus [33]. Vergleicht man die relativen Werte des Trace Comparison Values, zeigt sich, dass dieser bei der Simulation der GeoLife-Pfade auf einem deutlich höheren Niveau startet und dieses Niveau deutlich langsamer sinkt. Das Zielniveau aus [33] von 0.02% (entspricht $\frac{1}{50}$ Pfaden) wird daher weitaus später erreicht (s. 15). Es lässt sich demnach Vermuten, dass es einem Angreifer beträchtlich schwerer fallen würde, aufgenommen Spuren zu realen Pfaden zuzuordnen, als es die Ergebnisse aus [33] suggerieren.

Die Ergebnisse dieser Arbeit stellen die Befunde aus [33] insbesondere hinsichtlich des Trace Comparison Values in Frage. Wie beschrieben, lassen sich diese Differenzen jedoch nicht eindeutig auf die Nutzung der Pfade des GeoLife-Datensatzes zurückführen. Daher könnten weitere ähnliche Experimente, mit variierender Auswahl der Pfade und variierender Anzahl und Verteilung der Fog-Knoten, unternommen werden. Auf diese Weise könnte der Einfluss der Pfade bzw. der Verteilung der Fog-Knoten differenzierter betrachtet werden. Weiterhin könnten der in dieser Arbeit genutzte Simulationsaufbau variiert werden, sodass ein Angreifer bewusst die kontrollierten Fog-Knoten auswählen kann. So könnte beispielsweise verglichen werden, wie erfolgreich ein Angreifer abhängig davon sein kann, ob er entweder Fog-Knoten zentriert im Stadtzentrum Pekings kontrolliert oder dezentral in den weniger frequentierten Randbereichen Pekings kontrolliert.

Abschließend ist zu bemerken, dass die Folgerung aus [33], Angreifer könnten mit geringem Aufwand die Bewegung eines mobilen Endgerätes verfolgen, in dieser Arbeit nicht bestätigt wurde. Dies kann jedoch nicht Anlass dazu sein die Notwendigkeit von Sicherheitsmechanismen zum Schutz der Location-Privacy im Fog-Computing-Systemen zu relativieren. Vielmehr zeigt sich der Bedarf weiterer Experimente, welche die in dieser Arbeit erhaltenen Ergebnisse und die aus [33] kontextualisieren.

Literaturverzeichnis

[1] GPS Visualizer. accessed 2021-07-24. URL https://www.gpsvisualizer.com/.

[2] Cross product - Computational geometry. accessed 2021-07-28. URL https://en.wikipedia.org/wiki/Cross_product#Geometric_meaning.

[3] Europäische Union (2016) VERORDNUNG (EU) 2016/ 679 DES EUROPÄISCHEN PARLAMENTS UND DES RATES - vom 27. April 2016 - zum Schutz natürlicher Personen bei der Verarbeitung personenbezogener Daten, zum freien Datenverkehr und zur Aufhebung der Richtlinie 95/ 46/ EG (Datenschutz-Grundverordnung): RL 95/46/EG; Amtsblatt der Europäischen Union(L 119): S. 1–88. https://eur-lex.europa.eu/legal-content/DE/TXT/PDF/?uri=CELEX:32016R0679.

[4] GraphHopper Directions API. accessed 2021-07-28. URL https://www.graphhopper.com/.

[5] locprivfogsim. accessed 2021-08-04. URL https://git.uni-due.de/sjmsschl/locprivfogsim.

[6] Numba: A high performance python compiler. accessed 2021-08-04. URL http://numba.pydata.org/.

[7] SQLITE. accessed 2021-08-15. URL https://www.sqlite.org/.

[8] GeoLife: Building Social Networks Using Human Location History, 2009 accessed 2021-05-22). URL http://research.microsoft.com/en-us/projects/geolife/default.aspx.

[9] LocPrivFogSim, accessed 2021-05-24. URL https://git.uni-due.de/snthwett/locprivfogsim.

[10] 5G-Sender Reichweite mit verschiedenen Frequenzen, accessed 2021-06-24. URL https://ltemobile.de/5g-sendereichweiten-mit-unterschiedlichen-frequenzen/.

[11] Arwa Alrawais, Abdulrahman Alhothaily, Chunqiang Hu, and Xiuzhen Cheng. Fog Computing for the Internet of Things: Security and Privacy Issues. *IEEE Internet Computing*, 21(2):34–42, 2017. doi: 10.1109/MIC.2017.37.

[12] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. A View of Cloud Computing. *Communications of the ACM*, 53(4):50–58, April 2010. ISSN 0001-0782. doi: 10.1145/1721654.1721672.

[13] Philip Asuquo, Haitham Cruickshank, Jeremy Morley, Chibueze P. Anyigor Ogah, Ao Lei, Waleed Hathal, Shihan Bao, and Zhili Sun. Security and Privacy in Location-Based Services for Vehicular and Mobile Communications: An Overview, Challenges, and Countermeasures. *IEEE Internet of Things Journal*, 5(6):4778–4802, 2018. doi: 10.1109/JIOT.2018.2820039.

[14] A.R. Beresford and F. Stajano. Location privacy in pervasive computing. *IEEE Pervasive Computing*, 2(1):46–55, 2003. doi: 10.1109/MPRV.2003.1186725.

[15] Donald J. Berndt and James Clifford. Using dynamic time warping to find patterns in time series. In *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining*, AAAIWS'94, page 359–370. AAAI Press, 1994.

[16] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. Fog Computing and Its Role in the Internet of Things. In *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, MCC '12, page 13–16, New York, NY, USA, 2012. Association for Computing Machinery. ISBN 9781450315197. doi: 10.1145/2342509.2342513.

[17] Reynold Cheng, Yu Zhang, Elisa Bertino, and Sunil Prabhakar. Preserving User Location Privacy in Mobile Data Management Infrastructures. In *Privacy Enhancing Technologies*, pages 393–412, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. ISBN 978-3-540-68793-1.

[18] Marco Gruteser and Baik Hoh. On the Anonymity of Periodic Location Samples. In *International Conference on Security in Pervasive Computing*, pages 179–192. Springer, 2005. ISBN 978-3-540-32004-3.

[19] Yunguo Guan, Jun Shao, Guiyi Wei, and Mande Xie. Data Security and Privacy in Fog Computing. *IEEE Network*, 32(5):106–111, 2018. doi: 10.1109/MNET.2018.1700250.

[20] Harshit Gupta, Amir Vahid Dastjerdi, Soumya K. Ghosh, and Rajkumar Buyya. ifogsim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments. *Software: Practice and Experience*, 47(9):1275–1296, 2017. doi: https://doi.org/10.1002/spe.2509.

[21] Fatima Haouari, Ranim Faraj, and Jihad M. AlJa'am. Fog Computing Potentials, Applications, and Challenges. In *2018 International Conference on Computer and Applications (ICCA)*. IEEE, 2018.

[22] Guoqin Kang, Shuiqiao Jiang, and Zhiyuan Zhao. Research on millimeter wave spectrum planning for 5g. *Journal of Physics: Conference Series*, 1345:032040, 11 2019. doi: 10.1088/1742-6596/1345/3/032040.

[23] Krista Merry and Pete Bettinger. Smartphone gps accuracy study in an urban environment. *PLoS ONE*, 14:e0219890, 07 2019. doi: 10.1371/journal.pone.0219890.

[24] Shane T. Mueller, Brandon S. Perelman, and Elizabeth S. Veinott. An optimization approach for mapping and measuring the divergence and correspondence between paths. *Behavior Research Methods*, 48(1):53–71, March 2015. doi: 10.3758/s13428-015-0562-7.

[25] Carlo Puliafito, Diogo M. Gonçalves, Márcio M. Lopes, Leonardo L. Martins, Edmundo Madeira, Enzo Mingozzi, Omer Rana, and Luiz F. Bittencourt. MobFogSim: Simulation of mobility and migration for fog computing. *Simulation Modelling Practice and Theory*, 101:102062, 2020. ISSN 1569-190X. doi: https://doi.org/10.1016/j.simpat.2019.102062. Modeling and Simulation of Fog Computing.

[26] Nitin R.Chopde and Mangesh K. Nichat. Landmark based shortest path detection byusing a* and haversine formula. *International Journal of Innovative Research in Computer and Communication Engineering*, 1(2):298–302, 2013.

[27] Subhadeep Sarkar, Subarna Chatterjee, and Sudip Misra. Assessment of the Suitability of Fog Computing in the Context of Internet of Things. *IEEE Transactions on Cloud Computing*, 6(1): 46–59, 2018. doi: 10.1109/TCC.2015.2485206.

[28] Markus Schlotbohm. Consideration of computation offloading strategies in a simulation-based analysis of threats to location privacy in fog computing, Universität Duisburg-Essen, 2021.

[29] Pavel Senin. Dynamic time warping algorithm review. *Information and Computer Science Department University of Hawaii at Manoa Honolulu, USA*, 855(1-23):40, 2008.

[30] Han Su, Shuncheng Liu, Bolong Zheng, Xiaofang Zhou, and Kai Zheng. A survey of trajectory distance measures and performance evaluation. *The VLDB Journal*, 29(1):3–32, Jan 2020. ISSN 0949-877X. doi: 10.1007/s00778-019-00574-9.

[31] Luis M. Vaquero and Luis Rodero-Merino. Finding Your Way in the Fog: Towards a Comprehensive Definition of Fog Computing. *SIGCOMM Computer Communication Review*, 2014.

[32] Marius Wernke, Pavel Skvortsov, Frank Dürr, and Kurt Rothermel. A Classification of Location Privacy Attacks and Approaches. *Personal and Ubiquitous Computing*, 18(1):163–175, January 2014. ISSN 1617-4909. doi: 10.1007/s00779-012-0633-z.

[33] Theresa Wettig and Zoltán Mann. Simulation-based analysis of threats to location privacy in fog computing. In *2021 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*, pages 736–741, 03 2021. doi: 10.1109/PerComWorkshops51409.2021.9431073.

[34] Xiangye Xiao, Yu Zheng, Qiong Luo, and Xing Xie. Finding Similar Users Using Category-Based Location History. In *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*, GIS '10, page 442–445, New York, NY, USA, 2010. Association for Computing Machinery. ISBN 9781450304283. doi: 10.1145/1869790.1869857.

[35] Shanhe Yi, Cheng Li, and Qun Li. A Survey of Fog Computing: Concepts, Applications and Issues. In *Proceedings of the 2015 Workshop on Mobile Big Data*, Mobidata '15, page 37–42, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450335249. doi: 10.1145/2757384.2757397.

[36] Shanhe Yi, Zhengrui Qin, and Qun Li. Security and Privacy Issues of Fog Computing: A Survey. In *Wireless Algorithms, Systems, and Applications*, pages 685–695, Cham, 2015. Springer International Publishing. ISBN 978-3-319-21837-3.

[37] Hyoseok Yoon, Yu Zheng, Xing Xie, and Woontack Woo. Smart Itinerary Recommendation based on User-Generated GPS Trajectories. In *Proceedings of the International Conference on Ubiquitous Intelligence and Computing*. UIC 2010, October 2010. URL `https://www.microsoft.com/en-us/research/publication/smart-itinerary-recommendation-based-on-user-generated-gps-trajectories/`. Best Paper Award.

[38] Ashkan Yousefpour, Genya Ishigaki, Riti Gour, and Jason P. Jue. On Reducing IoT Service Delay via Fog Offloading. *IEEE Internet of Things Journal*, 5(2):998–1010, 2018. doi: 10.1109/JIOT.2017.2788802.

[39] Ashkan Yousefpour, Caleb Fung, Tam Nguyen, Krishna Kadiyala, Fatemeh Jalali, Amirreza Niakanlahiji, Jian Kong, and Jason P. Jue. All one needs to know about fog computing and related edge computing paradigms: A complete survey. *Journal of Systems Architecture*, 98:289–330, 2019. ISSN 1383-7621. doi: https://doi.org/10.1016/j.sysarc.2019.02.009.

[40] Yu Zheng, Xing Xie, and Wei-Ying Ma. Understanding Mobility Based on GPS Data. In *Proceedings of the 10th ACM conference on Ubiquitous Computing (Ubicomp 2008)*, September 2008. URL `https://www.microsoft.com/en-us/research/publication/understanding-mobility-based-on-gps-data/`.

[41] Yu Zheng, Xing Xie, and Wei-Ying Ma. Mining Interesting Locations and Travel Sequences From GPS Trajectories. In *Proceedings of International conference on World Wide Web 2009*, New York, NY, USA, April 2009. Association for Computing Machinery. URL `https://www.microsoft.com/en-us/research/publication/mining-interesting-locations-and-travel-sequences-from-gps-trajectories`. WWW 2009.

[42] Yu Zheng, Xing Xie, and Wei-Ying Ma. GeoLife: A Collaborative Social Networking Service among User, location and trajectory. June 2010. URL `https://www.microsoft.com/en-us/research/publication/geolife-a-collaborative-social-networking-service-among-user-location-and-trajectory`

[43] Yu Zheng, Hao Fu, Xing Xie, Wei-Ying Ma, and Quannan Li. *Geolife GPS trajectory dataset - User Guide*, geolife gps trajectories 1.1 edition, July 2011. URL `https://www.microsoft.com/en-us/research/publication/geolife-gps-trajectory-dataset-user-guide/`. Geolife GPS trajectories 1.1.

Anhang

Dateiverzeichnis

Die im Kontext dieser Arbeit erstellten Artefakte sind in dem Ordner *locPrivFogSim* beigefügt:

- Digitale Version der Arbeit: locPrivFogSim/docs
- Python-Script zur Vorverarbeitung der Pfade: locPrivFogSim/python_scripts/process_geolife/main.py gefunden werden
- Testsystem: locPrivFogSim/src/org/fog/vmmobile/TestExample4.java
- Wichtigsten Änderungen/Erweiterungen von Klassen:
  - DBConnector: locPrivFogSim/src/org/fog/utils/DbConnector.java
  - Attacker: locPrivFogSim/src/org/fog/privacy/Attacker.java
  - Position: locPrivFogSim/src/org/fog/privacy/Position.java
  - Path: locPrivFogSim/src/org/fog/localization/Path.java
  - Coordinate: locPrivFogSim/src/org/fog/localization/Coordinate.java
  - MobileController: locPrivFogSim/src/org/fog/placement/MobileController.java
- Experimentdurchführung:
  - Datenbank-Datei: locPrivFogSim/experiment_script/geoLifePaths.db
  - Experiment-Script: locPrivFogSim/experiment_script/run_geo_life_paths.sh
  - Ergebnisse: locPrivFogSim/experiment_script/results/result.csv
  - jar: locPrivFogSim/experiment_script/LocPrivFogSim.jar

Eidesstattliche Erklärung

Ich versichere an Eides statt durch meine Unterschrift, dass ich die vorstehende Arbeit selbständig und ohne fremde Hilfe angefertigt und alle Stellen, die ich wörtlich oder annähernd wörtlich aus Veröffentlichungen entnommen haben, als solche kenntlich gemacht haben, mich auch keiner anderen als der angegebenen Literatur oder sonstiger Hilfsmittel bedient haben. Die Arbeit hat in dieser oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen.

Mülheim an der Ruhr, den 9. September 2021