TECHNISCHE UNIVERSITÄT MÜNCHEN
Fakultät für Informatik
Lehrstuhl für medizinische Informatik

Datenschutz und biomedizinische Forschung: Konzepte und Lösungen für Anonymität

Diplom-Informatiker Univ.
Florian M. Kohlmayer

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender: Univ.-Prof. Dr. Thomas Neumann

Prüfer der Dissertation:

1. Univ.-Prof. Dr. Klaus A. Kuhn
2. Univ.-Prof. Dr. Claudia Eckert

Die Dissertation wurde am 29.10.2015 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 17.03.2016 angenommen.

Zusammenfassung

Die Verarbeitung und Integration großer und heterogener personenbezogener Datenmengen nimmt stetig zu. Die Anonymisierung dieser Daten ist eine Maßnahme die, unter allen Datenschutzmaßnahmen, eine wesentliche und letztlich die zentrale Rolle einnimmt. Die in dieser Arbeit gewählte Anwendungsdomäne ist die Medizin. Auch hier haben aktuelle Entwicklungen völlig neue Herausforderungen für die IT-Sicherheit und den Datenschutz nach sich gezogen. Moderne Forschung ist multizentrisch und datenintensiv geworden. Monokausale oder auch nur oligokausale Erkrankungen sind zwar nicht selten, aber die großen Volkskrankheiten wie Krebs, Koronare Herzkrankheit und Diabetes mellitus sind multikausal. Um die komplexen Zusammenhänge zu erforschen, haben sich Forschungsprojekte entwickelt, die national oder sogar international Daten und Bioproben sammeln. Dabei können die Tiefe der Datenerfassung, das Volumen und die Variationsbreite sehr hoch sein (detaillierte Erfassung der Phänotypen, kombiniert mit genomischen Daten). Ein Beispiel hierfür sind die Datenvolumina, die moderne Labor- und Sequenziermethoden in immer kürzerer Zeit produzieren.

In dieser Arbeit werden neue Algorithmen für die Anonymisierung von großen medizinischen Datensätzen vorgestellt. Bei Anonymisierungsprozessen werden Daten transformiert, um vorgegebene Datenschutzgarantien zu erreichen. In dieser Arbeit werden zwei Transformationsmodelle betrachtet: (1) Generalisierung von Attributwerten gefolgt von (2) Unterdrückung von Datensätzen. Die vorgeschlagenen Algorithmen optimieren die Ausgabe des Anonymisierungsprozesses hinsichtlich der Datenqualität. Um dabei große Datenmengen effizient verarbeiten zu können, nutzen verwandte Algorithmen Optimierungstechniken, die Teile des Suchraumes ausschließen. In dieser Arbeit wird erstmalig bewiesen, dass diese Optimierungstechniken nicht genutzt werden können, wenn Generalisierung und Unterdrückung als kombiniertes Transformationsmodell eingesetzt werden. Es werden neue Optimierungsmethoden für diese Fälle vorgeschlagen. Die entwickelten Algorithmen berücksichtigen diese und passen sich spezifisch den Anforderungen an. Die anschließende Evaluation zeigt, dass der gewählte Ansatz signifikant effizienter ist als verwandte Methoden.

Das Anonymisierungsproblem wird zudem auf verteilt vorliegenden Daten erweitert. Diese Daten wurden unter verschiedener Hoheit gesammelt und sollen nun, unter Wahrung des Datenschutzes, integriert und anonymisiert werden. Verwandte Ansätze sind unflexibel und unterstützen nur bestimmte Datenschutzgarantien und Transformationsmodelle. Außerdem skalieren sie schlecht. Die Arbeit beschreibt effiziente und flexible Protokolle zur Anonymisierung verteilt vorliegender Daten, sodass keine Partei die nicht-anonymisierten Daten sieht. Der Ansatz unterstützt sowohl horizontal als auch vertikal verteilte Datenbestände und kann mit verschiedenen Datenschutzgarantien und Transformationsmodellen genutzt werden. Zudem garantieren die Protokolle, dass der Anonymisierungsprozess sich bzgl. der Qualität des Ergebnisses nicht von einer Anonymisierung nicht-verteilter Daten unterscheidet. Die vorgestellten Protokolle ermöglichen eine Abwägung von Performanz und Sicherheit.

Abstract

In a modern IT-driven world, large and heterogeneous person-related data sets are collected and analyzed. As a consequence, the protection of sensitive personal data is a highly relevant topic, which has led to completely new challenges for IT security and privacy. Data anonymization takes a central role among all privacy protection mechanisms. The focus of this thesis is the biomedical domain. Modern research has become multicentric and data intensive. Mono- or even oligo-causal diseases are frequent but the most common diseases such as cancer, coronary heart disease and diabetes mellitus are multi-causal. To investigate complex disease processes, research projects have been established to collect data and biological samples on national or even international scales. The depth of data collection, its volumes and variations are high (detailed collection of phenotypic data, combined with genomic data). For example, ever increasing data volumes are produced by modern laboratory and sequencing equipment.

In this thesis, new algorithms are presented for the anonymization of large biomedical data sets. Data needs to be transformed to ensure that it adheres to given privacy requirements. In this work, we focus on two transformation methods: (1) generalization of attribute values followed by (2) tuple suppression. The proposed methods optimize their output regarding data quality. To ensure scalability, current methods use pruning techniques to exclude parts of the search space. We first prove that these techniques cannot be used when data is transformed with attribute generalization and tuple suppression. Next, we propose alternative pruning methods to overcome this limitation. We then describe a newly developed family of algorithms that is able to adopt itself to a wide variety of requirements by using different pruning techniques for solving different anonymization problems. Our evaluation shows that our approach significantly outperforms previous methods.

We also extend the anonymization problem to a distributed setup, in which data have been collected by different authorities and should be integrated and anonymized in a privacy-preserving manner. Previous solutions to this problem are either inflexible or not scalable. This work describes flexible and efficient protocols in the semi-honest model. The protocols support both horizontally and vertically distributed data and they can handle a wide variety of privacy models. In addition, the protocols assure that the quality of the output data of the anonymization process is not different from the quality that could be obtained by using a centralized method. The presented protocols allow users to trade-off performance with privacy guarantees.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

KAPITEL 1

Einleitung

Die Anonymisierung personenbezogener Daten ist eine Maßnahme, die angesichts der ständig wachsenden Bedeutung von Verarbeitung und Integration großer und heterogener Datenmengen („Big Data“) unter allen Datenschutzmaßnahmen eine wesentliche und letztlich die zentrale Rolle einnimmt. Von herausragender Bedeutung ist dabei, dass der Personenbezug als solcher verändert wird: Gemäß Definition im Bundesdatenschutzgesetz [1] verändert Anonymisierung personenbezogene Daten derart, dass die Daten einer bestimmbaren natürlichen Person nicht mehr oder nur mit unverhältnismäßig großem Aufwand zugeordnet werden können. Eine Parallele findet sich in dem „Health Insurance Portability and Accountability Act“ (HIPAA) [2] unter dem Abschnitt der „de-identification of protected health information“: dort bedeutet „de-identification“ (mit ähnlicher Definition, s. u.), dass die Daten danach „without oversight“ [3] verarbeitet werden können.

Die in dieser Arbeit gewählte Anwendungsdomäne ist die Medizin. Auch hier haben aktuelle Entwicklungen völlig neue Herausforderungen für die IT-Sicherheit und den Datenschutz nach sich gezogen. Herausragend ist die Diskussion um „genomic privacy“ [4–7], die aus der rapide wachsenden Verfügbarkeit genetischer und genomischer Daten resultiert. Diese können nun erstmals mit klinischen Daten (sogenannte phänotypische Daten [8]) und klinischen Verläufen zusammengeführt werden. Eine der wesentlichen Forschungsideen ist es, den Informations- und Methodentransfer vom Labor (das hier sehr breit zu sehen ist und „Next Generation Sequencing“ [9] ebenso umfasst wie Tiermodelle), also der „bench“, hin zum Menschen und weiter zur klinischen Anwendung, der „bedside“, zu unterstützen. Geschlossen wird der Kreis („from bench to bedside and back“), wenn Daten aus der Krankenversorgung erhoben werden, um aus Verläufen zu lernen und auch neue Hypothesen zu generieren. Die Typen von Studien sind naturgemäß verschieden und reichen von „first in man“ bis hin zu klinischen und epidemiologischen Registern und Kohorten. Die Schlüsselrolle für die Verfügbarkeit genetischer Daten spielen Bioproben [10], die den Zugang zu molekularbiologischen Daten erschließen (sequencing, genomics, transcriptomics, proteomics, usw., oft nur als „omics“ zusammengefasst).

An dieser Stelle sei kurz erwähnt, dass die in dieser Arbeit genutzten klassischen Anonymisierungsansätze bei genomischen Daten rasch an Grenzen stoßen, da diese Daten hoch-dimensional sind. Diese Hochdimensionalität bringt oft einen großen Informationsverlust mit sich [11]. Die zugehörigen phänotypischen Daten können jedoch mit den hier vorgestellten Ansätzen anonymisiert werden. Dies erschwert, nach dem heutigen Stand der genetischen Analysemöglichkeiten, die Zuordnung

von genetischen Daten zu den jeweiligen Spendern.

Moderne Forschung ist multizentrisch und datenintensiv geworden. Dies liegt zum einen in den erheblichen Datenvolumina begründet, die moderne Labor- und Sequenziermethoden in immer kürzerer Zeit produzieren. Hinzu kommt, dass monokausale oder auch nur oligokausale Erkrankungen zwar nicht selten sind (es gibt derzeit 5.000 - 8.000 seltene Krankheiten [12], die alle auf geringen Änderungen im Genom basieren), dass aber die großen Volkskrankheiten wie Krebs, Koronare Herzkrankheit und Diabetes mellitus multikausal sind. Neben genetischer Disposition spielen Verhalten und Umwelt eine wesentliche Rolle. Um das komplexe Geschehen zu erforschen, haben sich Forschungsprojekte entwickelt, die international oder sogar weltweit Daten und Bioproben sammeln. Dabei können die Tiefe der Datenerfassung, das Volumen und die Variationsbreite sehr hoch sein (detaillierte Erfassung der Phänotypen, kombiniert mit genomischen Daten).

Prominente Beispiele sind das „International Cancer Genome Consortium“ (ICGC) [13] und „The Cancer Genome Atlas“ (TCGA) [14]. Auch europäische Projekte wie BioMedBridges [15] oder BBMRI-LPC [16] versuchen, (Daten-) Brücken über Forschungsprojekte oder über Forschungsinfrastrukturen zu schlagen. Unmittelbar assoziiert mit der Zusammenführung von Daten und Biomaterialien ist die umgekehrte Richtung, das „Data Sharing“ zum Nutzen der Forschung. Die Global Alliance for Genomics and Health [17], und die Research Data Alliance [18] befassen sich mit grundsätzlichen Fragen hierzu. Die Arbeit „Towards a data sharing Code of Conduct for international genomic research“ [19] ist ein gutes Beispiel für die Auseinandersetzung mit ethischen Fragen in diesem Umfeld.

Es liegt auf der Hand, dass diese Entwicklungen das Spannungsfeld zwischen den Anforderungen zur Erforschung häufiger und bedrohlicher Krankheiten einerseits und dem Schutz der Privatsphäre des Patienten andererseits grundsätzlich und erheblich verändert haben [20–22].

Einen Überblick über die Herausforderungen der verteilten Forschung sowie Vorschläge für Lösungen präsentieren Malin et al. [23]. El Emam et al. haben in [24] systematisch verschiedenste Reidentifikationsangriffe auf medizinische Daten untersucht und gezeigt, dass bei unzureichenden Schutzmaßnahmen ein erhebliches Risiko für Patienten/Probanden entstehen kann. Speziell für Biobanken haben Malin et al. in [25] untersucht, welche Eigenschaften der Daten Einfluss auf eine mögliche Reidentifikation haben, wie das Risiko modelliert und gemessen werden, und auf welche Arten das Risiko minimiert werden kann.

1.1 Datenschutz in medizinischen Forschungsprozessen

Den medizinischen Forschungsprozess kann man grob in drei Phasen einteilen. In der ersten Phase werden Daten und Bioproben von Patienten oder Probanden gesammelt und verwaltet. Zu Beginn der wissenschaftlichen Erhebungen werden Voten von Ethikkommissionen sowie Einwilligungserklärungen von Patienten oder Probanden eingeholt. Ein besonderes Problem, auf das hier nicht eingegangen werden kann, stellt der „secondary use“ von Daten (oder auch Bioproben) aus der Krankenversorgung für die Forschung dar. Multizentrische Ansätze spielen in dieser ersten Phase eine erhebliche Rolle. Die nächste wichtige Phase ist die gemeinsame Nutzung und

Analyse von Daten und Bioproben. Hier wird bereits innerhalb eines Forschungskonsortiums der angesprochene Fall eintreten, dass auf der Basis von Ethikvotum und Einwilligung Forscher Daten, Bioproben und Analyseergebnisse erhalten, ohne dass sie mit den Patienten je in Kontakt waren. Dies gilt natürlich auch, wenn Daten und Biomaterialien auch Dritten zur Verfügung gestellt werden. Insbesondere für diesen Fall (Data / biomaterial sharing) kann es von Bedeutung sein, eine Portallösung zwischenzuschalten. Beispiele sind „The database of Genotypes and Phenotypes“ (dbGaP) [26] und „The European Genome-phenome Archive“ (EGA) [27]. In jedem Fall, also innerhalb eines Konsortiums oder darüber hinaus, spielt die Integration heterogener und verteilter Daten eine wesentliche Rolle.

Grundlage der Datenverarbeitung sind nationale und internationale Gesetze und Regularien. In Deutschland sind an erster Stelle das Bundesdatenschutzgesetz [1] und die jeweiligen Landesdatenschutzgesetze (z. B. das bayerische Datenschutzgesetz [28]) zu nennen. Auf EU-Ebene wird die maßgebliche EU-Direktive 95/46/EC [29] derzeit überarbeitet, und es zeichnet sich ab, dass die neue Regulierung [30] den Charakter einer Verordnung der Europäischen Union einnehmen wird. Speziell für Bioproben gibt es die „Recommendation Rec(2006)4 of the Committee of Ministers to member states on research on biological materials of human origin“ [31]. In den USA spielt der „Health Insurance Portability and Accountability Act“ (HIPAA) [2] eine wichtige Rolle, insbesondere die HIPPAA Privacy und Security Rule. Alle maßgeblichen Gesetze und Regularien fordern den Schutz von personenbezogenen Daten und Bioproben sowohl auf organisatorischer als auch technischer Ebene.

Eine Übersicht über notwendige Maßnahmen wurde in BioMedBridges [32, 33] erstellt: Auf organisatorischer Ebene werden Kommissionen und Verträge benutzt, um die Probanden/Patienten zu schützen. Ethikkommissionen beurteilen, vor dem Start, das Forschungsvorhaben aus ethischer, rechtlicher und oft auch datenschutzrechtlicher Sicht. Basierend darauf erstellt sie ein Votum für oder gegen das Forschungsvorhaben. Neue Probanden/Patienten werden vor Einschluss über das Forschungsvorhaben und dessen Konsequenzen für sie aufgeklärt. Mit der Einwilligungserklärung wird dokumentiert, dass er basierend darauf, eine Entscheidung getroffen hat. Bevor Forscher Zugriff auf Daten oder Proben aus einem Forschungsvorhaben erhalten, überprüft ein Datenzugriffskomitee die Anfragen. Es stimmt Anfragen zu, wenn sie konform mit den (z. B. in der Einwilligungserklärung) festgelegten Regeln für den Zugriff auf die Daten sind (z. B. Zweckbestimmung, Einhaltung des Datenschutzes auf Empfängerseite), oder verweigert die Herausgabe von Daten. Vertraglich werden die Rechte und Pflichten der Forscher daraufhin mit Datennutzungsverträgen und Materialübertragungsvereinbarungen festgehalten. Mit diesen Verträgen verpflichtet er sich, unter anderem, keine Reidentifikation vorzunehmen, die Daten nur für den angegebenen Zweck zu verwenden und die Daten sicher zu verarbeiten.

Auf technischer Ebene werden zunächst klassische Sicherheitsmaßnahmen implementiert, um die Vertraulichkeit der Probandendaten zu gewährleisten. Unautorisierter Zugriff wird verhindert, indem sich Nutzer authentisieren müssen. Die Berechtigungen der Nutzer werden durch rollenbasierte Zugriffskontrolle und/oder verschiedene Zugriffsstufen geregelt. Die Kommunikation zwischen Systemen und dem Nutzer erfolgt in einer sicheren (d. h. verschlüsselten, authentisierten und integri-

tätsgeschützten) Weise mittels Standardprotokollen. Auch gespeicherte Daten sollten verschlüsselt abgelegt werden. Eine ausführliche Protokollierung der Zugriffe, Änderungen und der Herkunft der Daten (z. B. mittels audit trail, intrusion detection system) ermöglicht die Entdeckung von Unregelmäßigkeiten und das Nachvollziehen von (erfolgreichen) Angriffen. Eine Datensicherungsstrategie und ein Wiederanlaufplan erhöhen die Verfügbarkeit der Daten.

Bei der Pseudonymisierung wird die Assoziation zwischen dem Probanden und den Daten reversibel entfernt. Nur autorisierte Personen können die Assoziation wieder herstellen. Pseudonymisierung wird häufig während der Datenerhebung verwendet, um Folgeerhebungen zu ermöglichen. Eine Pseudonymisierungsarchitektur für ein Forschungsnetz ist z. B. in [34] beschrieben.

Der Fokus dieser Arbeit liegt auf Anonymisierungsmethoden. Laut Bundesdatenschutzgesetz (BDSG) § 40 sind „[...] personenbezogene[n] Daten [...] zu anonymisieren, sobald dies nach dem Forschungszweck möglich ist. [...]“ [1]. Anonym sind solche Daten nach dem BDSG, wenn „ [...] die Einzelangaben über persönliche oder sachliche Verhältnisse nicht mehr oder nur mit einem unverhältnismäßig großen Aufwand an Zeit, Kosten und Arbeitskraft einer bestimmten oder bestimmbaren natürlichen Person zugeordnet werden können. [...]“ [1]. Solche Daten werden häufig auch als faktisch anonym bezeichnet [35].

In der HIPAA Privacy Rule § 164.514 [2] wird der Begriff „de-identification“ verwendet. Hierbei müssen die Daten so verändert werden, dass es keinen berechtigten Grund zu der Annahme mehr gibt, dass die Daten für eine Reidentifikation genutzt werden könnten. Dies kann, laut HIPAA, durch zwei Methoden erfolgen. Die erste Methode erfordert das Entfernen oder Modifizieren einer definierten Menge von 18 Attributen. Diese Heuristik reduziert das Reidentifikationsrisiko signifikant [36]. Allerdings können hierdurch Attribute entfernt werden, welche für eine weitere Analyse der Daten benötigt werden [37] oder unter bestimmten Umständen eine Reidentifikation nicht verhindern [38]. Die zweite Methode der HIPAA Privacy Rule ist die sogenannte „expert determination“, hierbei muss ein Experte das Restrisiko für eine Reidentifikation eines anonymisierten Datensatzes, vor der Herausgabe, als sehr gering einschätzen [2].

Bei dem ISO Standard „Health informatics - Pseudonymization“ (ISO/TS25237) [39] ist Anonymisierung als der Prozess definiert, welcher die Assoziation zwischen den identifizierenden Daten und dem Datensubjekt permanent und irreversibel entfernt.

Absolute Anonymisierung ist in der Realität kaum zu erreichen und bei der „faktischen Anonymisierung“ ebenso wie bei der HIPAA Methode der „expert determination“ gibt es einen Spielraum, der auch Gegenstand der Forschung war und ist. Es ist rasch klar geworden, dass Anonymisierung nicht erreicht werden kann, indem nur stark kennzeichnenden Attributen wie Name, Adresse und anderen direkten Identifikatoren (z. B. Personalausweisnummer) entfernt werden. Latanya Sweeney hat bereits 1997 gezeigt, dass durch Verknüpfen von vermeintlich anonymen Daten mit einem Wählerverzeichnis eine Reidentifikation mittels der Kombination aus Postleitzahl, Geschlecht und Geburtsdatum möglich ist. In einer späteren Studie schätzt sie, dass mit diesen drei Attributen wohl 87 % aller Amerikaner eindeutig identifizierbar sind. Es liegt nahe, dass auch medizinische Parameter für eine Reidentifikation genutzt werden können. Beispielsweise wurde dies anhand von

Diagnosen gezeigt [38]. Auch weniger offensichtliche Angriffspunkte wurden identifiziert, wie Muster in Stammbäumen [40] oder Besuchsmuster von verschiedenen Krankenhäusern [41].

Man kann sehen, dass ein breites Spektrum von Angriffspunkten existiert, bei denen Datensätze mit identifizierenden Informationen mit anonymisierten Daten verknüpft werden oder sensible Attributwerte aus den anonymisierten Daten für angegriffene Personen abgeleitet werden können [42]. Letztendlich resultierte die Einsicht, dass es keine „one-size-fits-all“ Anonymisierungslösung gibt, welche die Daten unter allen Umständen schützt und gleichzeitig den notwendigen Informationserhalt gewährleistet [42].

1.2 Problemstellung

Anonymisierungskonzepte müssen flexibel gestaltet sein. Sie müssen vor verschiedenen Bedrohungen schützen und für unterschiedliche Einsatzbereiche nutzbar sein. Insbesondere bieten sie einen Trade-off zwischen Datenqualität und Datenschutz. So kann ein gegebener Datensatz beispielsweise auf viele mögliche Arten transformiert werden, um vorgegebene Datenschutzgarantien zu erreichen. Im Bereich der Medizin wird hierfür vor allem eine Generalisierung von Attributwerten gefolgt von einer Unterdrückung von Datensätzen eingesetzt. Daten, die mittels Generalisierung anonymisiert werden, sind besonders für die Analysemethoden der Forscher geeignet. Wird diese Transformationsart kombiniert mit Unterdrückung eingesetzt, kann die Datenqualität deutlich zunehmen. Datensätze, welche eine stärkere Generalisierung erfordern würden, können unterdrückt werden und müssen somit nicht mehr berücksichtigt werden. Hier können informationstheoretische Modelle genutzt werden, um die Qualität des Resultats von Anonymisierungsprozessen zu bewerten und deren Ausgabe entsprechend zu optimieren. Dennoch ist im Regelfall eine Benutzerinteraktion notwendig um Daten optimal auf einen Anwendungsfall hin anzupassen. Oftmals ist dabei eine wiederholte Anwendung von Anonymisierungsmethoden mit unterschiedlichen Parametern erforderlich.

Als Konsequenz müssen diese sehr effizient sein. Um Skalierbarkeit zu erreichen, nutzen existierende Ansätze Optimierungstechniken, die es erlauben große Teile des Suchraumes auszuschließen. Dies geschieht, indem Eigenschaften der Ergebnisse von Transformationen aus Eigenschaften von Ergebnissen anderer Transformationen abgeleitet werden. Dabei können Schranken, sowohl für Datenschutzgarantien sowie für Datenqualität, berechnet werden, ohne dass die Transformationen auf den Daten angewendet werden müssen. Bestehende Ansätze, die diese Optimierungen nutzen fokussieren auf eine Transformation von Daten mittels Generalisierung. Diese Techniken können nicht genutzt werden, wenn Daten mittels Generalisierung und Unterdrückung transformiert werden. Da sowohl die Datenqualität als auch die Skalierbarkeit eine wichtige Rolle spielen, müssen neue Methoden entwickelt werden. Da sie außerdem flexibel sein müssen, sollten sie unterschiedliche Anonymitätsmodelle und Methoden zur Messung der Datenqualität unterstützen.

In der Praxis werden Daten sowohl von einzelnen Institutionen, als auch von Verbünden freigegeben. Neben dem Anonymisierungsproblem für lokal vorliegende Daten ergibt sich also auch ein Anonymisierungsproblem für Datensätze, welche verteilt über verschiedene Parteien gespeichert werden. Im zweiten Fall bestehen

ebenfalls die eben genannten Anforderungen. Eine geeignete Methode muss flexibel und skalierbar sein. Zusätzlich erfordern die Autonomie der Parteien und gesetzliche Vorgaben, dass die Daten nicht von einer zentralen Partei anonymisiert werden können. Existierende Ansätze sind unflexibel und unterstützen nur bestimmte Datenschutzgarantien, Algorithmen sowie Transformationsmodelle. Darüber hinaus erreichen ihre Ausgaben oftmals nicht die bestmögliche Datenqualität. Daten können sowohl horizontal (d. h. alle Parteien haben die gleichen Attribute aber Datensätze von verschiedenen Individuen) als auch vertikal (d. h. die Parteien haben unterschiedliche Attribute von denselben Individuen) verteilt vorliegen. Existierende Ansätze sind oft nicht in der Lage beide Szenarien gleichzeitig zu unterstützen. Auch die Skalierbarkeit ist bei einigen Ansätzen ein Problem. Aus diesen Gründen müssen Methoden entwickelt werden, die eine effiziente und flexible Integration und Anonymisierung von verteilt vorliegenden Daten ermöglichen.

1.3 Eigener Beitrag

Generalisierung von Attributwerten und Unterdrückung von Datensätzen sind die am besten geeigneten Transformationsmethoden für die Anonymisierung medizinischer Daten. In dieser Arbeit wird zunächst eine effiziente Umsetzung dieser Methoden untersucht. Eine Analyse bestehender Ansätze zeigt, dass neue Methoden entwickelt werden müssen, um die oben genannten Anforderungen zu erfüllen. Diese bilden den eigenen Beitrag der Arbeit.

- Zuerst wurden existierende Ansätze untersucht. Diese fokussieren sich auf einfache Datenschutzgarantien und Generalisierung als Transformationsmodell. Dabei nutzen sie Optimierungstechniken, um große Teile des Suchraums auszuschließen. In dieser Arbeit wird erstmalig bewiesen, dass diese Optimierungstechniken bei den, in dieser Arbeit betrachteten, komplexeren Anonymitätsmodellen (d. h. ℓ-Diversität, t-Closeness und δ-Präsenz) und Transformationsmodellen (d. h. Generalisierung und Unterdrückung) nicht angewandt werden können.

- Um dennoch große Datenmengen verarbeiten zu können, werden in dieser Arbeit alternative Optimierungstechniken entwickelt. Diese erlauben es auch bei Einsatz der hier betrachteten Modelle Teile des Suchraumes auszuschließen. Im nächsten Schritt wurde eine Familie von Algorithmen entwickelt, welche in der Lage ist adaptiv verschiedene Optimierungstechniken zu kombinieren. Dadurch sind unterschiedliche Kombinationen von Datenschutz-, Transformationsmodellen und Methoden zur Messung des Informationsverlustes effizient handhabbar.

- Die genannten Methoden wurden implementiert und evaluiert. Das Ergebnis zeigt, dass der vorgeschlagene Ansatz signifikant effizienter ist als verwandte Arbeiten. Darüber hinaus ist nur der hier vorgestellte Ansatz in der Lage komplexere Anonymitäts- und Transformationsmodelle effizient zu lösen.

Der zweite Teil der Arbeit befasst sich mit dem Fall, dass Daten unter verschiedenen Hoheiten stehen und verteilt vorliegen. Ziel ist es, diese Daten in einem sicheren Prozess zu integrieren und zu anonymisieren.

- In der Arbeit wird zunächst experimentell nachgewiesen, dass die Ergebnisse von naiven Lösungsansätzen eine schlechte Datenqualität aufweisen. Darüber hinaus können naive Methoden sogar die angestrebten Anonymitätsgarantien verletzen.

- Deswegen werden neuartige Protokolle entwickelt. Diese stellen sicher, dass im Falle einer verteilten Datenhaltung dieselbe Lösung gefunden wird, wie wenn die Daten zentral gespeichert worden wären. Anders ausgedrückt, ist die Lösung äquivalent mit dem Ergebnis des nicht sicheren Prozesses, zuerst alle Daten an eine zentrale Partei zu übermitteln und diese anschließend zu anonymisieren. Die vorgestellten Protokolle ermöglichen eine Abwägung von Performanz und Sicherheitsgarantien. Sie wurden so flexibel gestaltet, dass sie verschiedene Algorithmen, Anonymitätsmodelle und Qualitätsmodelle unterstützen. Auch sind sie in der Lage sowohl horizontal als auch vertikal verteilte Daten zu anonymisieren. Sie unterstützen eine beliebige Anzahl von Parteien.

- Die Protokolle wurden effizient implementiert und ausgiebig evaluiert. Die Ergebnisse zeigen die Skalierbarkeit des Ansatzes. Sowohl die notwendige Bandbreite als auch die benötigte Rechenleistung kann von handelsüblicher Hardware bereitgestellt werden.

1.4 Gliederung

- Kapitel 2: In diesem Kapitel werden etablierte Methoden vorgestellt, die im Rahmen dieser Arbeit Verwendung finden. Das verwendete Bedrohungsmodell wird erläutert; es ist aus der Literatur zum Thema Anonymisierung bekannt. Hinsichtlich der möglichen Methoden der statistischen Offenlegungskontrolle wird eine eigene Klassifikation vorgestellt, die aus 2 bekannten Klassifikationen entwickelt wurde. Hierdurch wird ein Lösungsraum beschrieben, der zu den Anforderungen der Domäne in Beziehung gesetzt wird. Die am besten geeigneten Methoden und die darauf aufbauenden Anonymisierungskriterien werden im Detail vorgestellt.

- Kapitel 3: Dieses Kapitel stellt neu entwickelte Anonymisierungsalgorithmen für die Anonymisierung lokal vorliegender Daten vor, die im vorherigen Kapitel ausgewählten Methoden und Kriterien werden dabei effizient unterstützt. Zuerst wird die Monotonie verschiedener Anonymisierungskriterien untersucht, da diese einen großen Einfluss auf die Effizienz der Anonymisierung hat. Die neu entwickelte Familie von Algorithmen erlaubt erstmalig auch bei nicht-monotonen Kriterien eine optimale, bezüglich einer beliebigen, auch nicht-monotonen, Informationsverlustmetrik, eine Lösung zu finden. Die anschließende Evaluation zeigt, dass die Algorithmen sowohl effizienter sind als die bekannten verwandten Algorithmen für monotone Metriken und Kriterien als auch, dass im Falle der nun erstmalig unterstützten Nicht-Monotonie von Kriterien und Metriken es immer noch effizient ist, eine optimale Lösung zu finden.

- Kapitel 4: Dieses Kapitel erweitert das Anonymisierungsproblem auf verteilt vorliegende Daten, die unter verschiedener Hoheit gesammelt wurden und nun,

unter Wahrung des Datenschutzes, integriert werden sollen. Es beschreibt für diesen Fall effiziente Protokolle um die Daten zu anonymisieren, ohne dass es eine Partei gibt, welche die nicht-anonymisierten Daten sieht. Die entwickelten Protokolle sind so entworfen worden, dass verschiedene Algorithmen, Kriterien und Metriken genutzt werden können, um die Daten zu anonymisieren. Für die folgende Evaluation wurde der in Kapitel 3 vorgestellte Algorithmus verwendet.

- Kapitel 5: Das letzte Kapitel fasst die wichtigsten Ergebnisse zusammen und diskutiert die lokalen und verteilten Anonymisierungskonzepte. Ein Ausblick beschreibt daraufhin die noch offenen Fragen und präsentiert Vorschläge für weitere Forschungsarbeiten im Bereich der Anonymisierung medizinischer Daten.

KAPITEL 2

Methodische Grundlagen

In diesem Kapitel werden die verwendeten Methoden erklärt. Dazu wird zunächst das in der Literatur übliche Bedrohungsmodell eingeführt, gefolgt von einer Übersicht und Klassifikation der bekannten Anonymisierungsverfahren. Diese Anonymisierungsverfahren werden daraufhin mit den Anforderungen in der medizinischen Domäne in Einklang gebracht. Als Nächstes werden die für die medizinische Domäne am besten geeigneten Anonymisierungsverfahren mit ihren spezifischen Schwachstellen erklärt. Um diese Kriterien erfüllen zu können, müssen die Daten verändert werden. Die beiden in dieser Domäne besonders geeigneten Verfahren (und damit auch die in dieser Arbeit verwendeten), Generalisierung und Unterdrückung, werden daraufhin beschrieben. Das Methodenkapitel beschließt ein Überblick über gängige Metriken um den Informationsverlust zu messen, welcher durch die Anonymisierung verursacht wird.

2.1 Vorstellung des verwendeten Bedrohungsmodells

Sollen Daten oder Ergebnisse aus Analysen dieser Daten für die Nutzung verfügbar gemacht werden, können diese Daten von Angreifern auf verschiedene Weise genutzt werden. Im Folgenden wird das hier verwendete Bedrohungsmodell vorgestellt. Zur Vereinfachung der Betrachtung wird davon ausgegangen, dass die Daten in einem Rechteckschema vorliegen. Dies ist in Abbildung 2.1 dargestellt.

In der Tabelle enthält jede Zeile die Daten eines einzelnen Individuums. Die Spalten (Attr. A - Attr. H) stellen die gesammelten Attribute der Individuen dar. Die Annahme in dieser Arbeit ist, dass, wenn ein Angreifer diese Daten erhält, er versucht diese Daten bestimmten Individuen zuzuordnen. Hierbei kann man die Spalten in zwei Klassen einteilen. Die eine Klasse beinhaltet die Spalten, welche von einem Angreifer zur Reidentifizierung benutzt werden können. Diese werden als identifizierenden Attribute bezeichnet. Die zweite Klasse enthält die restlichen Attribute, die nicht-identifizierenden. Beide Klassen zerfallen in jeweils zwei Kategorien. Die erste Klasse der identifizierenden Attribute zerfällt in direkt identifizierende und quasi identifizierende Attribute [43]. Die direkt identifizierenden Attribute werden üblicherweise zur Identifikation verwendet und sind für jedes Individuum eindeutig. Solche Attribute sind z. B. Namen und Adressangaben oder eindeutige Identifikationsnummern (Personenkennzeichen). Bei der zweiten Kategorie wird vereinfachend davon ausgegangen, dass der Angreifer Hintergrundwissen über die

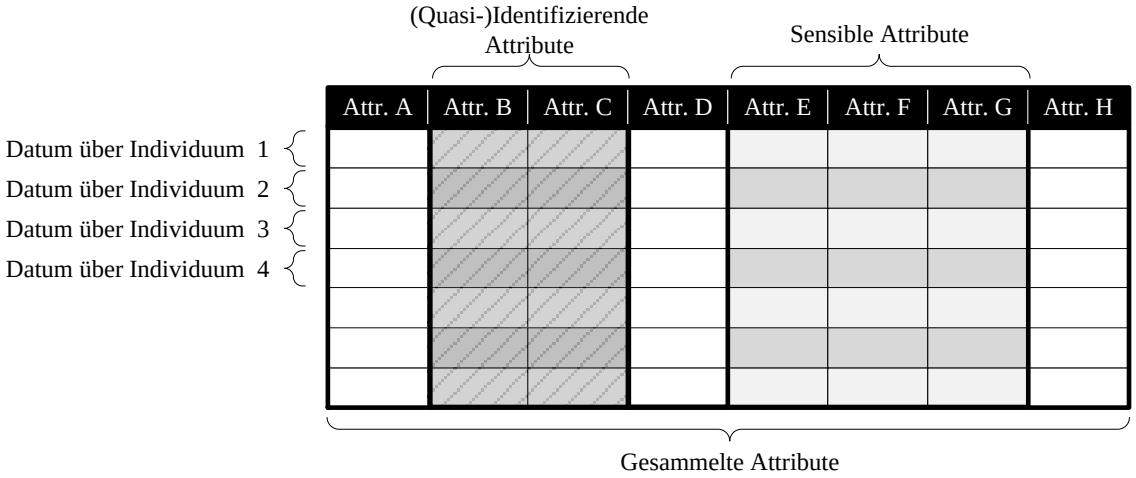


Abbildung 2.1: Datenmodell

anzugreifenden Individuen besitzt. Dieses Hintergrundwissen kann auch wieder als Tabelle modelliert werden. Diese beiden Tabellen können nun miteinander in Beziehung gesetzt werden (vgl. Join-Operation der relationalen Algebra). Die Attribute, die ein kombinieren ermöglichen, werden als Quasi-Identifikatoren bezeichnet. Diese Attribute ermöglichen einem Angreifer das Identifizieren eines Individuums. Die zweite Klasse der Attribute, die nicht-identifizierenden, zerfallen wiederum in zwei Kategorien. Die erste Kategorie sind die sensiblen Attribute, diese möchte der Angreifer einem Individuum zuordnen können [44]. Die zweite Kategorie sind die Attribute, die den Angreifer nicht interessieren und auch nicht für eine Reidentifikation genutzt werden können. Diese Einteilung der Attribute hängt somit vom konkreten Angreifer ab. Die Attribute können also in eine andere Kategorie oder Klasse fallen, je nachdem welcher Angreifer betrachtet wird. Eine mögliche alternative Vorgehensweise zur Bestimmung der Quasi-Identifikatoren ist die Bestimmung der Eindeutigkeit von Attributkombinationen im jeweiligen Datensatz. Die Obermenge aller Attributkombinationen, die einen Schwellwert überschreiten, werden zu Quasi-Identifikatoren [45]. Generell ist hier anzumerken, dass das Problem der Auswahl der Quasi-Identifikatoren und sensitiven Attribute Anwendungsfall spezifisch ist und in dieser Arbeit nicht weiter betrachtet wird. Es wird im Folgenden davon ausgegangen, dass eine Einteilung der Daten in die jeweiligen Klassen erfolgt ist.

Ausgehend von diesem Datenmodell und der Unterscheidung der Attribute in identifizierende und sensible Attribute wurden zwei verschiedene Typen von Informationsenthüllungen in der Literatur identifiziert [46, 47]. Identitätsaufdeckung (identity disclosure) und Attributsaufdeckung (attribute disclosure). Identitätsaufdeckung tritt auf, wenn ein Datensatz einer bestimmten Person zugeordnet werden kann. Attributsaufdeckung tritt auf, wenn neue Information über eine Person bekannt wird. Hierbei ist es also möglich, aus den veröffentlichten Daten, Charakteristika einer Person genauer oder überhaupt zu bestimmen. Ein spezieller Fall von Attributsaufdeckung ist die Mitgliedschaftsaufdeckung (membership disclosure) [48]. Hierbei ist die aufgedeckte Charakteristik die Mitgliedschaft der bedrohten Person zu einer Gruppe. Ist der Datensatz zum Beispiel von einem Krankheitsnetzwerk, leidet die betreffende Person mit hoher Wahrscheinlichkeit an dieser Krankheit. Im

Falle einer Identitätsaufdeckung folgt meistens auch eine Attributsaufdeckung, da die Attribute des Datensatzes der Person zugeordnet werden können. Auf der anderen Seite ist es möglich, dass eine Attributsaufdeckung ohne Identitätsaufdeckung erfolgt. Auch kann eine Mitgliedschaftsaufdeckung erfolgen, ohne dass weitere Attribute zu der Person bekannt werden. Dieser Sachverhalt ist in Abbildung 2.2 dargestellt.

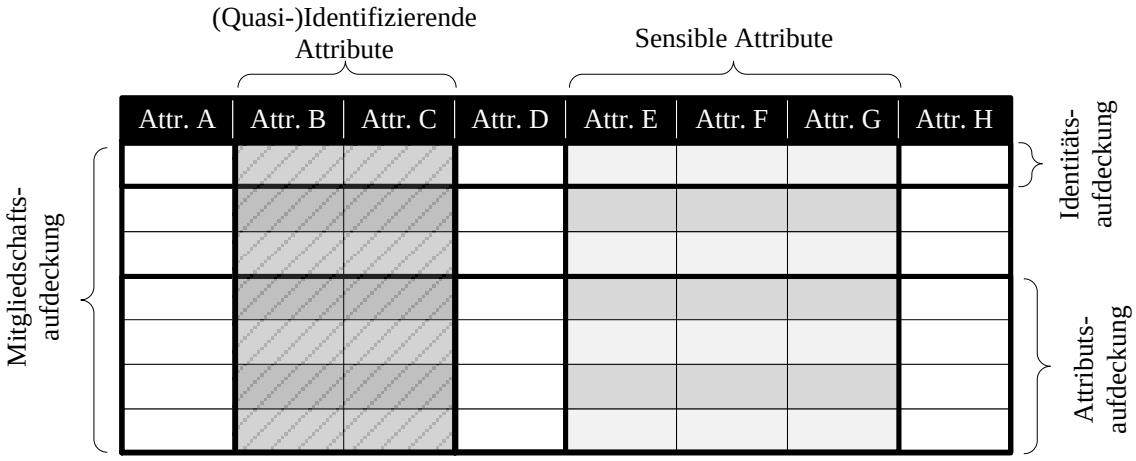


Abbildung 2.2: Bedrohungsmodell

Hierbei bedeutet die Identitätsaufdeckung, dass ein Angreifer genau ein Tupel der Relation einer Person zuordnen kann. Im Falle der Attributsaufdeckung ist es dem Angreifer nicht mehr möglich die Person genau einem Tupel zuzuordnen, sondern nur noch einer Menge von Tupeln. Die Attributsaufdeckung ist erfolgreich, sollten alle diese Tupel einen gemeinsamen Attributwert für mindestens ein sensibles Attribut aufweisen. Alternativ kann es auch möglich sein, dass aus der Verteilung der Attributwerte zusätzliche Information über die Menge ableitbar ist. Bei der Mitgliedschaftsaufdeckung hingegen ist es dem Angreifer nur noch möglich festzustellen, ob die angegriffene Person in dem Datensatz durch ein Tupel repräsentiert ist oder nicht. Generell ist es hierbei wichtig zu unterscheiden, dass es zwei Arten der Aufdeckung von Informationen gibt, negative Aufdeckung und positive Aufdeckung [44]. Beide Arten der Aufdeckung können Schaden für die betroffene Person verursachen. Positive Aufdeckung bedeutet, dass bestimmte Attributwerte (mit einer hohen Wahrscheinlichkeit) einer Person zugeordnet werden können. Negative Aufdeckung hingegen schließt die Zuordnung von bestimmten Attributwerten (mit einer hohen Wahrscheinlichkeit) aus. Außerdem wurde in [46] festgestellt, dass auch falsche Zuordnung von Informationen zu einer Person, Schaden verursachen kann. Diesen Bedrohungen kann man mit verschiedenen Methoden, welche in den folgenden Abschnitten beschrieben werden, begegnen.

2.2 Klassifikation der verschiedenen Anonymisierungsmethoden

Um den oben genannten Bedrohungen zu begegnen, kann man die Daten anonymisieren. Die Idee hierbei ist, dass dem Angreifer die Möglichkeit genommen wird, sein Hintergrundwissen mit den veröffentlichten Daten in Beziehung zu setzen. Die

hierzu verwendeten Anonymisierungsmethoden können grob in zwei Klassen eingeteilt werden, die interaktiven und nicht-interaktiven Methoden [49]. Das Prinzip der interaktiven Verfahren ist es, dem Nutzer eine Schnittstelle zur Verfügung zu stellen, die Anfragen auf den Daten erlaubt. Die jeweiligen Antworten werden, vor der Weitergabe an die Anfragenden, bereinigt. Bei einigen Verfahren können die Ergebnisse verfälscht werden (perturbativ), andere Verfahren reduzieren nur den Informationsgehalt der Daten (nicht-perturbativ). Anwendung finden interaktive Verfahren bei Systemen wie DataSHIELD [50] oder SHRINE [51]. Die Hauptanwendung von DataSHIELD ist es, Analysen bei verteilten Datenbeständen durchzuführen, ohne dass dabei die Mikrodaten an einer zentralen Stelle zusammengeführt werden müssten. Dabei müssen die Analysemethoden so erweitert werden, dass nur nicht-identifizierende Teilergebnisse an die zentrale Stelle übermittelt werden. Diese kombiniert daraufhin das Ergebnis. DataSHIELD wurde für den Einsatz von generalisierten linearen Modellen als Analysemethode entwickelt. SHRINE ist ein föderiertes Forschungssystem welches zum Ziel hat, Daten aus vielen Kliniken zu integrieren und dabei den Datenschutz der Patienten sicherzustellen. Anfragen an das System liefern nur die Größe der Ergebnismenge zurück. Jede Anfrage wird an die Kliniken verteilt und auf den lokalen Daten ausgeführt. Das Ergebnis wird lokal anonymisiert, bei der zentralen Stelle zusammengeführt und dem Nutzer präsentiert. Eine vielversprechendes Konzept in diesem Bereich ist „Differential Privacy“ [49]. Die Idee bei Differential Privacy ist, dass sich das Risiko für ein Individuum nicht stark verändern darf, ob das Individuum in dem Datensatz repräsentiert ist oder nicht. Das wird erreicht indem gefordert wird, dass eine Analyse auf einem Datenbestand mit dem Individuum und auf einem Datensatz ohne das Individuum ungefähr dasselbe Ergebnis liefert. Man kann also anhand des Ergebnisses nicht unterscheiden auf welcher der beiden Datensätze die Funktion evaluiert wurde. Damit wird sichergestellt, dass die Teilnahme an einer statistischen Datenbank kein Datenschutzproblem für einen Teilnehmer darstellt. Eine häufig eingesetzte Methode um Differential Privacy zu erreichen ist das Hinzufügen von Störungen zu den Abfrageergebnissen. Das Ergebnis kann somit verfälscht sein.

Basis der nicht-interaktiven Verfahren ist die Herausgabe eines bereinigten Datensatzes, der dann für Analysen zur Verfügung steht. Die übergebenen Daten werden geschützt, wobei die mögliche Breite, der gegen sie gestellten Anfragen, ein Kernproblem hinsichtlich der Datenqualität darstellt. Es können bereinigte Mikrodaten oder aggregierte Daten herausgegeben werden. Auch hier gibt es wieder Verfahren, welche die Daten verfälschen können (perturbativ) und solche, die nur den Informationsgehalt reduzieren (nicht-perturbativ). Bei den nicht-interaktiven Methoden, besonders bei Herausgabe von Mikrodaten, ist es wichtig, die Daten robust zu anonymisieren. Hierbei geht die unmittelbare Hoheit über die Mikrodaten verloren. Um dieses Problem zu lindern, werden diese Verfahren meist durch vertragliche und gesetzliche Regelungen begleitet, welche zumindest eine mittelbare Hoheit garantieren sollen. Der Hauptvorteil bei Herausgabe von anonymisierten Mikrodaten ist, dass der Empfänger der Daten beliebige Analysen auf diesen ausführen kann. Zudem muss kein Service zur Beantwortung der Anfragen zur Verfügung gestellt werden und die Analysemethoden müssen dem Herausgeber der Daten nicht offen gelegt werden. Übliche Kriterien für nicht-interaktive Methoden sind k-Anonymität, ℓ-Diversität, t-closeness und δ-Präsenz (siehe Abschnitt 2.4). Eine Übersicht über

die Klassifikation ist in Abbildung 2.3 dargestellt, sie kombiniert die Klassifikationen von [49, 52] und [53].

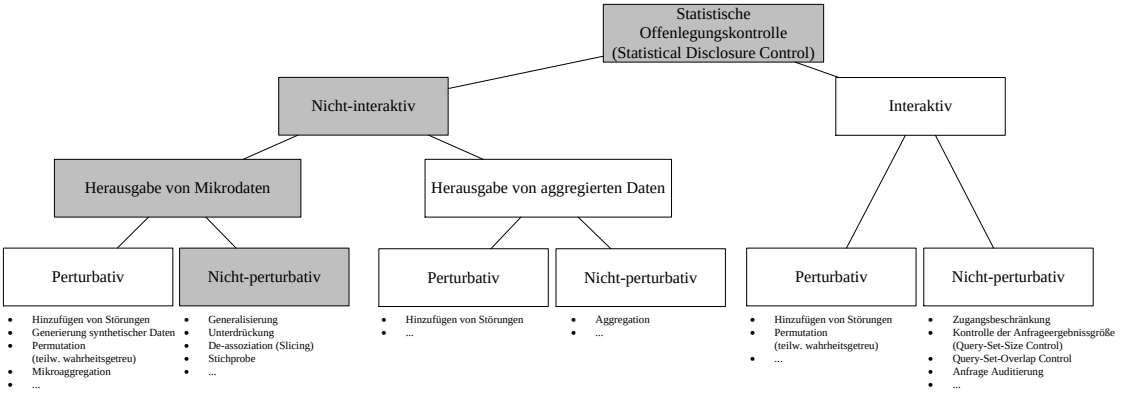


Abbildung 2.3: Klassifikation der Methoden der statistischen Offenlegungskontrolle (kombiniert aus [49, 52, 53])

In dieser Arbeit liegt der Fokus auf den nicht-perturbativen Verfahren zur Herausgabe von Mikrodaten, insbesondere mittels Generalisierung und Unterdrückung. Diese Art der Anonymisierung ist für die medizinische Domäne besonders geeignet (siehe auch Abschnitt 2.3). Der Pfad zu den in dieser Arbeit verwendeten Verfahren ist in der Abbildung 2.3 grau hinterlegt.

2.3 Methodische Anforderungen der Domäne in Relation zu bekannten Verfahren

Einige der oben genannten Anonymisierungsmethoden werden aktuell auf Eignung in der medizinischen Domäne geprüft [54]. Aktuell ist es in der medizinischen Forschung üblich, nicht-interaktive Anonymisierungsverfahren zu nutzen, um es den Forschern zu ermöglichen, die Daten zu analysieren. Die Daten werden dann in anonymisierter Form an die auswertende Stelle gegeben. Nach Dankar und El Emam [54] hat dies folgende Gründe:

- Die Daten sind oft fehlerhaft und weisen unerwartete Verteilungen auf. Hierbei ist es am einfachsten z. B. passende Kohorten auszuwählen und korrigierende Transformationen anzuwenden, wenn auf die Daten direkt zugegriffen werden kann.
- Die Forscher haben oft angepasste Software-Tools und Standardvorgehensweisen. Hierbei wäre es schwer, die Forscher davon zu überzeugen, andere Verfahren anzuwenden und die Auswertungen in fremde Hände zu geben.
- Die Rechtssicherheit ist größer, da die Verfahren zur Anonymisierung und Herausgabe seit mehreren Jahrzehnten praktiziert werden. Im Falle eines Rechtsstreits sind einige Präzedenzfälle vorhanden, um die Wahl der Datenschutzmethodik zu rechtfertigen.

Analysemethoden von Forschern können proprietär sein. Interaktive Verfahren haben hier den Nachteil, dass die Analysemethoden oft offengelegt werden müssen.

Aus diesen Gründen liegt der Fokus in den folgenden Kapiteln auf nicht-interaktiven Verfahren zur Anonymisierung der Daten zum Zwecke der Weitergabe. Da hierdurch einerseits eine erhebliche Relevanz und Verbreitung solcher Verfahren begründet wird, andererseits besteht aber auch noch nicht unerheblicher Optimierungs- und damit Forschungsbedarf.

Bei den abzuwendenden Bedrohungen liegt der aktuelle Fokus der Forschung auf dem Schutz vor Identitätsaufdeckung. Dies liegt zum größten Teil daran, dass die aktuellen Gesetze einzig diesen Schutz fordern (siehe z. B.[1, 2, 29]) und damit der Schutz vor Identitätsaufdeckung sehr praxisrelevant ist. Außerdem ist das Risiko der Identitätsaufdeckung höher einzuschätzen als das der Attributsaufdeckung [55]. Aus diesen Gründen wird in den folgenden Kapiteln meistens auf die Bedrohung der Identitätsaufdeckung eingegangen. Da allerdings viele der Verfahren durch leichte Anpassungen auch Schutz vor Attributsaufdeckung und Mitgliedschaftsaufdeckung bieten können, liegt es nahe, ebenfalls diese beiden erweiterten Bedrohungen zu untersuchen.

Hinsichtlich der Anonymisierungsverfahren wurden in dieser Arbeit Generalisierung und Unterdrückung gewählt, da diese für die medizinische Domäne gut geeignet sind. Die Generalisierung wird mittels globalem Recodieren und mit vom Nutzer definierten Generalisierungshierarchien durchgeführt. Ausreißer, welche eine unverhältnismäßige Generalisierung verursachen würden, werden auf Tupel Ebene unterdrückt. Eine genauere Beschreibung dieser beiden Methoden findet sich in Abschnitt 2.5 und Abschnitt 2.5.2. Außerdem sollte eine Anonymisierung in dem resultierenden Suchraum eine global optimale Lösung finden. Optimal bedeutet, dass, gegeben eine Metrik zum Messen des Informationsverlustes (siehe Abschnitt 2.6), die Transformation gefunden wird, welche den niedrigsten Informationsverlust hat. Im Einzelnen werden die Gründe für diese Wahl in der folgenden Liste dargelegt (siehe auch [56–58]):

- Durch die Verwendung von benutzerdefinierten Generalisierungshierarchien können diese den konkreten Anwendungsszenarien angepasst werden.
- Die Nutzung von „global-recoding“ macht die Verwendung von statistischen Standardverfahren zur Analyse der Daten einfacher, im Gegensatz zu „local-recoding“, da alle Werte innerhalb einer Spalte hierbei konsistent vergröbert werden.
- Global optimale Lösungen helfen den Informationsverlust bei der Anonymisierung gering zu halten.
- Dem Empfänger der Daten ist leicht zu erklären, was mit den Daten gemacht wurde. Dies erleichtert die Interpretation der Daten.
- Die Daten werden nicht verfälscht, sondern nur vergröbert („truthful“), was wiederum der Datenqualität zugutekommt.

Neben der Generalisierung und Unterdrückung als nicht-perturbative Verfahren wird auch das Hinzufügen von Störungen als Anonymisierungsverfahren empfohlen [54]. Der größte Nachteil bei diesen perturbativen Verfahren ist, dass diese die Daten verfälschen können. Analysen auf diesen Daten können somit zu

falschen Schlussfolgerungen führen. Das Ziehen von Stichproben, eine anderes nicht-perturbatives Verfahren, verkleinert den Stichprobenumfang, was die statistische Aussagekraft verringern kann. Generalisierung und Unterdrückung sind somit gut geeignete Anonymisierungsverfahren für die medizinische Domäne.

2.4 Überblick über geeignete Anonymisierungskriterien

Zur Illustration der verschiedenen Anonymisierungskriterien wird die in 2.4 dargestellte Relation mit 8 Zeilen und vier Spalten als Beispiel verwendet. Dieses Beispiel wird später auch zur Veranschaulichung der Funktionalitäten der neu entwickelten Algorithmen genutzt. Jede Zeile repräsentiert einen Patienten und für jeden Patienten wurde sein Alter, Geschlecht, Postleitzahl seines Wohnortes und seine Diagnose erhoben. Dieses Beispiel entspricht dem abstrakten Datenmodell der Abbildung 2.1. Die Spalte Alter, Geschlecht und Postleitzahl sind (quasi-)identifizierende Attribute und das Attribut Diagnose ist ein sensibles Attribut.

Quasi-identifizierend			Sensibel
Alter	Geschlecht	Postleitzahl	Diagnose
34	Männlich	82667	Lungenentzündung
45	Weiblich	81775	Lungenentzündung
66	Männlich	81925	Gastritis
70	Männlich	81931	Lungenentzündung
35	Weiblich	81951	Lungenentzündung
21	Männlich	82451	Gastritis
18	Weiblich	82931	Lungenentzündung
19	Weiblich	82004	Gastritis

Abbildung 2.4: Beispiel Relation

Um den Bedrohungen, die in Abschnitt 2.2 angesprochen worden sind, zu begegnen, stehen zahlreiche Anonymisierungskriterien zur Verfügung. Im hier betrachteten nicht-interaktivem Umfeld sind einige Kriterien bekannt und teilweise verbreitet. K-Anonymität schützt vor Identitätsaufdeckung, ℓ-Diversität und t-Closeness schützen vor Attributsaufdeckung und δ-Präsenz schützt vor Mitgliedschaftsaufdeckung. Die meisten weiteren und weniger bekannten Kriterien versuchen Attributsaufdeckung zu verhindern.

2.4.1 k-Anonymität

Die häufig zum Einsatz kommende k-Anonymität [59–61], welche vor Aufdeckung der Identität schützt, ist wie folgt definiert:

Definition 2.1 (*k-Anonymität*)

Sei $T(A_1, ..., A_n)$ eine Tabelle und $\{A_i, ..., A_q\} \subset \{A_1, ..., A_n\}$ die Menge der quasi-identifizierenden Attribute. Eine Tabelle ist k-anonym bzw. erfüllt die k-Anonymität, genau dann, wenn jedes Tupel $t = (a_i, ..., a_q)$ mindestens k mal in T auftritt.

Informal kann man sagen, dass eine Tabelle genau dann k-anonym ist, wenn jede Zeile, bei einer Projektion auf die Quasi-Identifikatoren, mindestens k-mal auftritt. Somit kann ein Angreifer eine Person nicht einem spezifischen Tupel zuordnen, sondern nur einer Äquivalenzklasse, welche aus mind. k Einträgen und somit anderen Personen besteht. Ein oft verwendeter Wert in der biomedizinischen Forschung ist $k = 5$, wobei sich hier das Spektrum zwischen $k = 3$ und $k = 25$ bewegt [62].

Eine 2-anonyme Version der Relation 2.4 ist in Abbildung 2.5 dargestellt.

Alter	Geschlecht	Postleitzahl	Diagnose
20-60	Männlich	82***	Lungenentzündung
20-60	Weiblich	81***	Lungenentzündung
≥ 61	Männlich	81***	Gastritis
≥ 61	Männlich	81***	Lungenentzündung
20-60	Weiblich	81***	Lungenentzündung
20-60	Männlich	82***	Gastritis
≤ 19	Weiblich	82***	Lungenentzündung
≤ 19	Weiblich	82***	Gastritis

Abbildung 2.5: 2-anonyme Relation

2.4.2 ℓ-Diversität

K-Anonymität bietet keinen Schutz vor Attributsaufdeckung. In der Arbeit von Machanavajjhala et al. [44] wurden diesbezüglich zwei Angriffe vorgestellt. Der Homogenitätsangriff (homogeneity attack) nutzt identische sensible Attribute in einer k-Gruppe. Angenommen alle Tupel in einer Äquivalenzklasse haben den gleichen Wert (z. B. Lungenentzündung). Kann nun ein Angreifer eine Person zu dieser Klasse zuordnen, dann kann dieser den wahrscheinlichen Wert für diese Person herleiten (d. h., diese Person ist höchstwahrscheinlich an Lungenentzündung erkrankt). Bei dem Angriff mit Hintergrundwissen (background-knowledge attack) kann der Angreifer unwahrscheinliche Werte durch sein Hintergrundwissen ausschließen und somit den wahrscheinlichen Wert für die angegriffene Person bestimmen. Zum Schutz vor diesen beiden Angriffen wurde deshalb von Machanavajjhala et al. [44] ein weiteres Anonymisierungskriterium namens ℓ-Diversität vorgestellt. Die Idee ist hierbei, dass es einem Angreifer nicht mehr möglich sein soll, sensible Attribute einer Person zuordnen zu können. Damit soll eine Attributsaufdeckung verhindert werden. Das ℓ-diversitäts-Prinzip besagt, dass es in jeder Äquivalenzklasse mindestens ℓ „gut repräsentierte“ Werte für die sensiblen Attribute geben soll. Machanavajjhala et al. [44] befassen sich mit Interpretationen von „gut repräsentiert“:

- Distinkte-ℓ-Diversität: Im einfachsten Fall muss jede Äquivalenzklasse mindestens ℓ unterschiedliche sensible Attributwerte beinhalten. Das Problem hierbei ist, dass diese Art der ℓ-Diversität nicht gegen Wahrscheinlichkeitsrückschlüsse schützt. Es kann zum Beispiel vorkommen, dass eines der Attributwerte sehr häufig in einer Äquivalenzklasse vorkommt. Dies ermöglicht einem Angreifer darauf zu schließen, dass eine Person, die in diese Äquivalenzklasse

fallen würde, mit einer sehr hohen Wahrscheinlichkeit diesen sensiblen Wert hat.

- Entropie-ℓ-Diversität: Die Entropie einer Äquivalenzklasse E ist definiert als

$$Entropie(E) = -\sum_{s \in S} p(E, s) \log p(E, s) \tag{2.1}$$

  Hierbei bezeichnet S die Domäne des sensiblen Attributes und $p(E, s)$ den Bruchteil der Einträge in E die s als sensiblen Wert haben. Eine Tabelle ist entropie-ℓ-divers, wenn für jede Äquivalenzklasse E gilt, dass $Entropie(E) \geq \log \ell$. Hierbei liegt die Idee zugrunde, dass je mehr Entropie in einer Äquivalenzklasse vorhanden ist, umso besser ist die Repräsentanz der Werte verteilt. Diese Definition kann allerdings zu restriktiv sein, da hierbei die gesamte Tabelle, für das jeweilige sensible Attribut, mindestens eine Entropie von $\log \ell$ haben muss.

- Rekursive-(c, ℓ)-Diversität: Diese Art der ℓ-Diversität ist die von den Autoren bevorzugte. Sie ist nicht so streng, wie die Entropie-Diversität. Die Idee hierbei ist, dass die am häufigsten vorkommenden Werte in einer Äquivalenzklasse nicht zu häufig vorkommen und dass die selten auftretenden Werte nicht zu selten vorkommen. Die Häufigkeit ist hierbei steuerbar anhand des Parameters c. Eine Äquivalenzklasse ist (c, ℓ)-divers, wenn gilt dass $r_1 < c(r_\ell + r_{\ell+1} + \cdots + r_m)$. M stellt die Anzahl der verschiedenen Werte dar und $r_i, 1 \leq i \leq m$ gibt jeweils die Häufigkeit des i-häufigsten Wertes in der Äquivalenzklasse an. Eine Tabelle ist somit wieder (c, ℓ)-divers, wenn alle Äquivalenzklassen (c, ℓ)-divers sind. Daraus folgt, dass je größer der Parameter c gewählt wird, umso einfacher ist es (c, ℓ)-Diversität zu erreichen.

ℓ-Diversität einer Tabelle ist wie folgt definiert:

Definition 2.2 (*ℓ-Diversität*)

Eine Äquivalenzklasse ist ℓ-divers, wenn sie mindestens ℓ „gut repräsentierte“ Werte für die sensiblen Attribute beinhaltet. Eine Tabelle ist ℓ-divers, wenn alle Äquivalenzklassen ℓ-divers sind.

Die in Abbildung 2.6 dargestellte Tabelle ist distinkt-2-divers und rekursiv-(4,2)-divers. Eine interessante Beobachtung ist, dass jede ℓ-diverse Relation automatisch auch ℓ-anonym ist. Es gibt mindestens ℓ unterschiedliche sensible Attributwerte in jeder Äquivalenzklasse, daraus folgt, dass die Klasse mindestens ℓ Einträge umfassen muss.

Alter	Geschlecht	Postleitzahl	Diagnose
20-60	*	8****	Lungenentzündung
20-60	*	8****	Lungenentzündung
≥ 61	*	8****	Gastritis
≥ 61	*	8****	Lungenentzündung
20-60	*	8****	Lungenentzündung
20-60	*	8****	Gastritis
≤ 19	*	8****	Lungenentzündung
≤ 19	*	8****	Gastritis

Abbildung 2.6: Distinkt-2-diverse und rekursiv-(4,2)-diverse Relation

2.4.3 t-Closeness

Während ℓ-Diversität eine Verbesserung hinsichtlich der Attributsaufdeckung ist, gibt es mind. zwei Angriffe, gegen welche ℓ-Diversität nicht schützt [63]. Zum einen ist dies der Asymmetrie-Angriff (skewness attack), der bei ungleichmäßiger Verteilung der sensiblen Werte in einer Äquivalenzklasse durchgeführt werden kann. Als Beispiel wird die Annahme betrachtet, dass 0,5 % der Weltbevölkerung an HIV erkrankt sind und dies auch in dem Forschungsdatensatz repräsentiert ist. Gibt es nun eine Äquivalenzklasse in dem anonymisierten Datensatz, welche eine gleiche Anzahl an positiven wie negativen Werten für das HI-Virus hat, so genügt diese Klasse der 2-Diversität (in allen Formen). Allerdings kann ein Angreifer nun mit 50 % Wahrscheinlichkeit davon ausgehen, dass eine angegriffene Person, welche in diese Klasse fällt, HIV positiv ist. Der zweite Angriff, vor welchem ℓ-Diversität nicht schützen kann, ist der Ähnlichkeitsangriff (similarity attack). Hierbei wird davon ausgegangen, dass die Werte ein einer Spalte zwar syntaktisch unterschiedlich, semantisch aber ähnlich sind. Gibt es in einer Äquivalenzklasse zum Beispiel die zwei unterschiedlichen Werte „akute Gastritis“ und „chronische Gastritis“, so kann diese Äquivalenzklasse der 2-Diversität genügen. Ein Angreifer kann aber nun daraus ableiten, dass eine Person, welche in diese Äquivalenzklasse fällt, an Gastritis erkrankt ist.

Aus diesen Gründen wurde von Li et al. [63] das Prinzip der t-Closeness vorgeschlagen und in [64] auf das flexiblere (n,t)-Closeness erweitert. Die Idee bei Closeness ist, dass die Verteilung in den einzelnen Äquivalenzklassen nicht sehr stark von der Verteilung der gesamten Tabelle abweicht. Vereinfacht gesagt soll die Verteilung der sensiblen Attribute, in jeder Äquivalenzklasse, ähnlich sein wie die Verteilung der Werte in der gesamten Tabelle.

Um den Abstand zwischen zwei Verteilungen zu bestimmen, schlagen die Autoren die Earth mover's distance (EMD) vor. Sie definieren drei verschiedene Arten um die EMD zwischen zwei Verteilungen zu berechnen. Eine Version beschreibt die Möglichkeit um die EMD in einer geschlossenen Form für numerische Werte zu berechnen, sie lautet:

$$D[P,Q] = \frac{1}{m-1} \sum_{i=1}^{m} \left| \sum_{j=1}^{i} (p_j - q_j) \right| \tag{2.2}$$

Dabei ist $D[P,Q]$ die EMD zwischen den Verteilungen P und Q und p_i und q_i sind jeweils Elemente der Verteilung P bzw. Q.

Für kategoriale (categorical) Attribute schlagen die Autoren zwei verschiedene Varianten vor. Erstens eine einfache, welche gleichen Abstand, nämlich 1, zwischen zwei beliebigen kategorischen Werten annimmt. Zweitens eine komplexere, welche eine Generalisierungshierarchie (siehe Abschnitt 2.5) zu Hilfe nimmt, um den Abstand zwischen zwei Werten zu berechnen. Die Formel für den gleichen Abstand lautet:

$$D[P,Q] = \frac{1}{2}\sum_{i=1}^{m}|(p_j - q_j)| = -\sum_{p_i<q_i}(p_i - q_i) \tag{2.3}$$

Bei der hierarchischen Version ist der Abstand zwischen zwei Werten definiert als der niedrigste Level, auf welchem es einen Wert in der Generalisierungshierarchie gibt, zu dem sich beide Werte generalisieren lassen. Um die EMD nun anhand dieser Abstandsmetrik zu berechnen, zeigen die Autoren, dass gilt:

$$D[P,Q] = \sum_{N} cost(N) \tag{2.4}$$

Hierbei bezeichnet N die Menge aller nicht Blatt Knoten und

$$cost(N) = \frac{height(N)}{H} min(pos\_extra(N), neg\_extra(N)) \tag{2.5}$$

H ist die Höhe der Generalisierungshierarchie. $Pos\_extra(N)$ und $neg\_extra(N)$ sind definiert als:

$$pos\_extra(N) = \sum_{C \in Child(N) \wedge extra(C) > 0} |extra(C)| \tag{2.6}$$

$$neg\_extra(N) = \sum_{C \in Child(N) \wedge extra(C) < 0} |extra(C)| \tag{2.7}$$

Die Berechnung des Wertes extra für den Knoten N ist rekursiv definiert als:

$$extra(N) = \begin{cases} p_i - q_i, & \text{wenn N ist ein Blattknoten} \\ \sum_{C \in Child(N)} extra(C), & \text{sonst.} \end{cases} \tag{2.8}$$

Damit ist eine Äquivalenzklasse t-close, wenn für sie gilt, dass der Abstand zwischen ihrer Verteilung und der Verteilung in der Tabelle, kleiner t ist, also $D[P,Q] \leq t$.

Definition 2.3 (*t-Closeness*)

Eine Äquivalenzklasse ist t-close, wenn der Abstand der Verteilung der sensiblen Werte nicht mehr als t von der Verteilung des Attributes, in der gesamten Tabelle abweicht. Eine Tabelle ist t-close, wenn alle Äquivalenzklassen t-close sind.

Das Beispiel aus Abbildung 2.6 ist 0,2-Close für EMD sowohl mit hierarchischem als auch mit gleichem Abstand.

2.4.4 δ-Präsenz

Ein weiteres bekanntes Anonymitätskriterium wurde von Nergiz et al. vorgeschlagen [48]. δ-Präsenz schützt gegen Mitgliedschaftsaufdeckung. Die Idee ist hierbei die Daten so weit zu anonymisieren, dass ein Angreifer nur mit einer Wahrscheinlichkeit von $\leq \delta$, Aussagen über die Mitgliedschaft einer Person in dem anonymisierten Datensatz treffen kann. Hierbei ist die ganze Tabelle δ-präsent, wenn jedes Tupel δ-präsent ist. Die Wahrscheinlichkeit für eine Mitgliedschaft wird in der 2007 Arbeit direkt aus vorhandenem Populationswissen und der zu veröffentlichenden Tabelle hergeleitet, indem die Häufigkeit einer Äquivalenzklasse im Populationsdatensatz mit der Häufigkeit des Auftretens der dazugehörigen Äquivalenzklasse, in der zu veröffentlichenden Tabelle, in Relation gesetzt wird.

Definition 2.4 (δ-*Präsenz*)

Eine Tabelle T ist δ-präsent, mit $\delta = (\delta_{min}, \delta_{max})$, wenn für eine gegebene Generalisierung $T*$ gilt, dass $\delta_{min} \leq Pr(t \in T|T*, P) \leq \delta_{max}$. Hierbei stellt T die zu veröffentlichende Tabelle und P die Populationstabelle dar.

Das Beispiel in Abbildung 2.7 ist eine (0,0.5)-präsente Anonymisierung des Datensatzes aus Abbildung 2.4 unter der Annahme, dass die Forschungsteilmenge aus allen männlichen Personen besteht.

Teilmenge	Alter	Geschlecht	Postleitzahl	Diagnose
j	*	*	82***	Lungenentzündung
n	*	*	81***	Lungenentzündung
j	*	*	81***	Gastritis
j	*	*	81***	Lungenentzündung
n	*	*	81***	Lungenentzündung
j	*	*	82***	Gastritis
n	*	*	82***	Lungenentzündung
n	*	*	82***	Gastritis

Abbildung 2.7: (0,0.5)-präsente Relation

Die hier beschriebenen Anonymisierungskriterien wurden so ausgewählt, dass jeweils mindestens ein Kriterium vorgestellt wurde, um den drei Bedrohungen der Identitätsaufdeckung, Attributsaufdeckung und Mitgliedschaftsaufdeckung, entgegen treten zu können. Die in dieser Arbeit entwickelten Algorithmen und Protokolle sollen diese Kriterien unterstützen. Auch die Evaluation wird mit diesen Kriterien durchgeführt.

2.5 Überblick über geeignete Anonymisierungsverfahren

Zur Umsetzung der soeben in Abschnitt 2.4 beschriebenen Anonymisierungskriterien können verschiedene Verfahren eingesetzt werden. Wie in Abschnitt 2.3 bereits dargelegt, sind Generalisierung und Unterdrückung in der biomedizinischen Forschung besonders gut geeignete Kandidaten. Ein zentrales Argument ist, dass

nur der Informationsgehalt reduziert wird. Andere Verfahren (z. B. Störungen hinzufügen, siehe auch Abschnitt 2.2) können die Daten verfälschen. Im Folgenden sollen nun die relevanten Grundlagen erläutert werden.

2.5.1 Generalisierung als Anonymisierungsverfahren

Um eine Generalisierung vorzunehmen, stehen verschiedene Ansätze zur Verfügung, die im Folgenden in Anlehnung an Fung et al. [53] charakterisiert werden.

2.5.1.1 Globales und lokales Recodieren

Eine erste Unterscheidung ist zwischen globalem recodieren und lokalem recodieren vorzunehmen. Hierbei bedeutet lokales recodieren, dass verschiedene Generalisierungen für gleiche Werte innerhalb eines Attributes angewandt werden können. Ein Beispiel ist in Abbildung 2.8 zu sehen, wo das Geschlecht des Beispieldatensatzes nur teilweise generalisiert wurde.

Quasi-identifizierend			Sensibel
Alter	Geschlecht	Postleitzahl	Diagnose
34	*	82667	Lungenentzündung
45	*	81775	Lungenentzündung
66	Männlich	81925	Gastritis
70	Männlich	81931	Lungenentzündung
35	*	81951	Lungenentzündung
21	*	82451	Gastritis
18	Weiblich	82931	Lungenentzündung
19	Weiblich	82004	Gastritis

Abbildung 2.8: Lokales Recoding des Geschlechts

Lokales recodieren wird meist von Algorithmen genutzt, welche auf clustering Verfahren basieren z. B. [65]. Globales recodieren hingegen wird meistens zusammen mit Generalisierungshierarchien eingesetzt. Allerdings können sowohl lokale als auch globale Recodierungsverfahren jeweils auf beiden Verfahren aufbauen.

Globales recodieren bedeutet, dass die gleiche Generalisierung für alle Instanzen des gleichen Wertes angewandt wird. Durch dieses restriktivere Recodierungsmodell werden die Möglichkeiten der Generalisierung eingeschränkt, was oft zu größerem Informationsverlust führt, dafür aber den Suchraum besser beherrschbar werden lässt. Eine Ausnahme hinsichtlich der Datenqualität sind viele Data-Mining Algorithmen [53]. Beim lokalen recodieren kann es vorkommen, dass die gleichen Werte mit unterschiedlichen Bezeichnern versehen werden, z. B. bleibt einmal der Bezeichner „Lungenentzündung“ bestehen, und beim nächsten Auftreten wird der gleiche Wert zu „Atemwegserkrankung“ generalisiert. Manche Data-Mining Algorithmen können nun nicht erkennen, dass beide Werte semantisch nahe beisammen liegen.

2.5.1.2 Eindimensionale und mehrdimensionale Generalisierung

Eine weitere Unterscheidung existiert zwischen eindimensionaler und mehrdimensionaler Generalisierung. Bei der eindimensionalen Generalisierung wird jedes Attribut, unabhängig von anderen Attributen, generalisiert. Im Gegensatz dazu steht

die mehrdimensionale Generalisierung, hierbei nimmt die Eingabe für die Generalisierungsfunktion mehrere Attributwerte entgegen. Damit kann z. B. „82667“ und „Lungenentzündung“ zu „826***“ und „Lungenentzündung“ generalisiert werden; wohingegen „81775“ und „Lungenentzündung“ zu „81775“ und „Atemwegserkrankung“ generalisiert wird.

2.5.1.3 Full-domain und Subtree Generalisierung

Die eindimensionale Generalisierung lässt sich in full-domain und subtree Generalisierung weiter aufgliedern. Bei der full-domain Generalisierung werden alle Werte auf die gleiche Stufe des Generalisierungsbaumes (siehe folgender Abschnitt) generalisiert. Bei der subtree Generalisierung hingegen werden alle Kinder eines Subbaumes auf den gleichen Wert generalisiert. Der Fokus dieser Arbeit liegt im Folgenden auf der Klasse der globalen, eindimensionalen und full-domain Algorithmen.

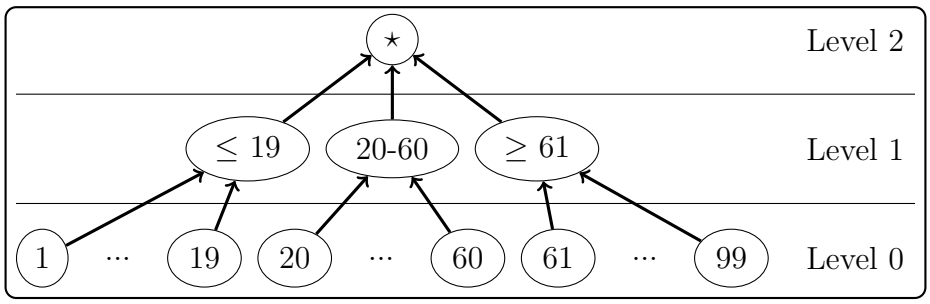


Abbildung 2.9: Generalisierungshierarchie *Alter*

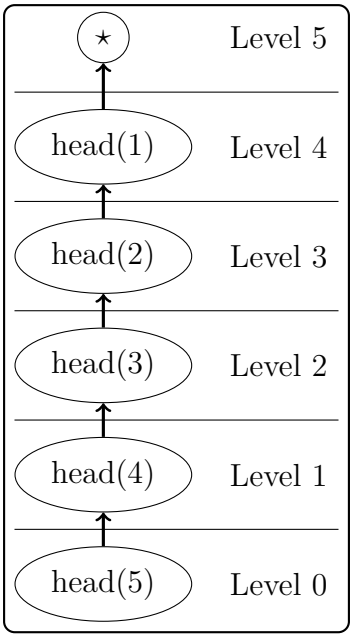


Abbildung 2.10: Generalisierungshierarchie *Postleitzahl*

2.5.1.4 Generalisierungshierarchien

Eine Möglichkeit, die Generalisierung einzelner Quasi-Identifikatoren zu steuern, ist die Definition von Generalisierungshierarchien. Sie definieren die schrittweise Ver-

gröberung der Attribute. Für das Beispiel in dieser Arbeit finden sich die Hierarchien in Abbildung 2.9, Abbildung 2.10 und Abbildung 2.11.

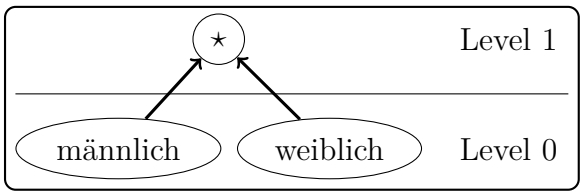


Abbildung 2.11: Generalisierungshierarchie *Geschlecht*

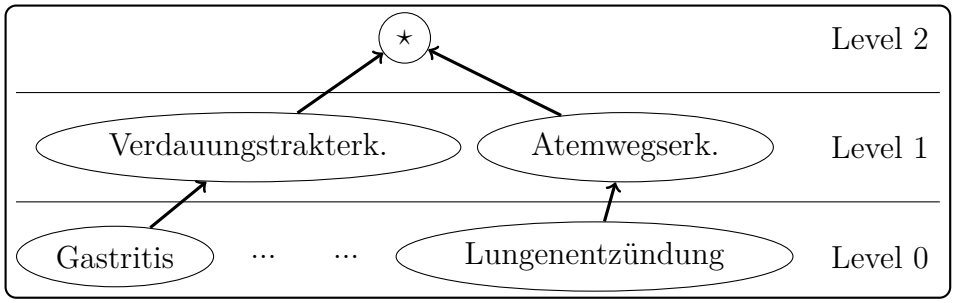


Abbildung 2.12: Generalisierungshierarchie *Diagnose*

Das Attribut Alter wird im ersten Level auf die Intervalle von 1-19, 20-60 und 61-99 generalisiert und im zweiten Schritt auf das Intervall 1-99 (∗). Das Geschlecht wird im ersten Schritt auf Person (∗) generalisiert. Die Postleitzahl wird schrittweise um die unwichtigste Stelle gekürzt und damit vergröbert. Die Diagnosen (siehe Abbildung 2.12 werden hinsichtlich ihrer Lokalisation generalisiert. Im Folgenden werden nur Generalisierungsbäume betrachtet, siehe dazu auch Abschnitt 3.2.

2.5.1.5 Monotonie der Hierarchien

Ein wichtiges Kriterium bei Generalisierungshierarchien ist die Monotonie, da sie Voraussetzung für Optimierungen ist (siehe Abschnitt 3.2). Monoton bedeutet hierbei, dass die Gruppen auf Level n sich nur aus Gruppen von Level $n-1$ zusammensetzen dürfen. Daraus resultieren Monohierarchien, also Bäume mit einer Halbordnung, bei der jeder Vorgänger eindeutig ist. Eine Verletzung der Monotonie wäre z. B. bei der Hierarchie Alter gegeben, wenn sich Intervalle auf verschiedenen Leveln überschneiden. Dies wäre der Fall, wenn auf Level 1 die Intervalle 1-15, 16-30, 31-45, 46-60, usw. (Intervallbreite 15) definiert wären, und auf Level 2 Intervalle mit der Breite 20 (d. h. 1-20, 21-40, 41-60, usw.).

2.5.1.6 Generalisierungsverband

Die Datenstruktur, welche alle möglichen Kombinationen von Generalisierungsmöglichkeiten in eine Ordnung bringt, ist ein Generalisierungsverband.

Auf dieser Datenstruktur operieren die meisten Algorithmen, welche globales Codieren nutzen. Die Menge der möglichen Kombinationen ist hierbei mit einer Halbordnungsrelation versehen, welche die Quersumme der einzelnen Generalisierungen als Ordnungskriterium nutzt. Diese Menge kann man als gerichteten Graph darstellen, der Hasse-Diagramm genannt wird. Ein Beispiel für den kompletten Verband,

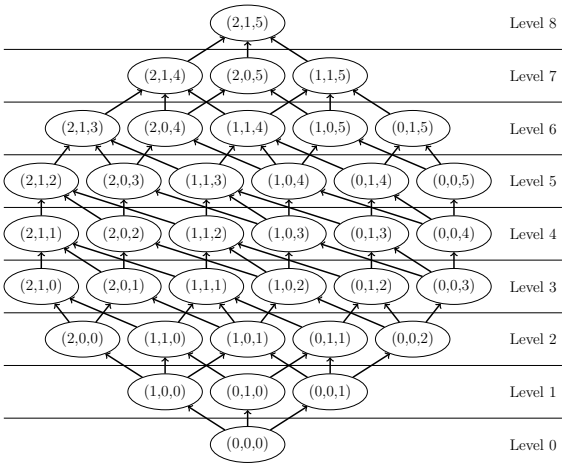


Abbildung 2.13: Generalisierungsverband

welcher aus den im vorherigen Abschnitt beschriebenen Generalisierungshierarchien hervorgeht, ist in Abbildung 2.13 zu sehen.

Jeder Knoten stellt eine gültige Kombination von Generalisierungsleveln der einzelnen Quasi-Identifikatoren dar. Der Knoten $(2, 0, 3)$ bedeutet zum Beispiel, dass Alter auf Level 2 generalisiert wurde, also alle Werte $\star$ entsprechen. Das Geschlecht ist unverändert (Level 0) und bei der Postleitzahl wurden die drei am wenigsten signifikanten Stellen abgeschnitten. Das Level eines Knoten in dem Verband ergibt sich aus der Quersumme der einzelnen Generalisierungslevel. Eine Kante zwischen zwei Knoten bedeutet, dass sich nur das Generalisierungslevel eines einzigen Quasi-Identifikators geändert hat. Der Knoten $(0, 0, 0)$, welcher den unveränderten Datensatz, also den am wenigsten generalisierten, repräsentiert, ist unten eingezeichnet und wird als Infimum bezeichnet. Der Knoten, welcher die am meisten generalisierte Transformation repräsentiert $(2, 1, 5)$, ist oben eingezeichnet und wird als Supremum bezeichnet. Die Transformation für das Beispiel in Abschnitt 2.4.1 ist $(1, 0, 3)$.

2.5.1.6.1 Eigenschaften eines Generalisierungsverbandes Es seien $q_0, ..., q_{n-1}$ die Quasi-Identifikatoren und somit n deren Anzahl. Jeder Quasi-Identifikator q_i hat eine zugeordnete Generalisierungshierarchie H_i, sie besteht aus den Leveln $h_{i_0}, ..., h_{i_{m-1}}$, und hat somit die Höhe m_i. Eine Strategie ist definiert als ein Pfad in dem Verband. Er startet beim Infimum und endet beim Supremum, hierbei stellt das Infimum (bottom) das kleinste Element und das Supremum (top) das größte Element der halb-geordneten Menge dar. Die Länge eines solchen Pfades ist immer v, was der Höhe des Verbandes entspricht. Ein Teilverband (Sublattice) ist definiert durch zwei Elemente b und t aus der gesamt Menge, hierbei gilt, dass die Quersumme der beiden Elemente ungleich ist und die Summe von b kleiner ist als die Summe von t. Damit ist b das Supremum und t das Infimum des Teilverbandes. Der Teilverband beinhaltet b, t und alle Spezialisierungen von t und alle Generalisierungen von b. Ein Knoten in dem Verband ist definiert als Tupel $k = (k_0, .., k_{n-1})$ wobei $0 \leq k_i \leq h_i$.

Die Anzahl der Knoten eines Verbandes mit n Quasi-Identifikatoren und ihren

maximalen Leveln $m_0, ..., m_{n-1}$ ist gegeben durch:

$$\#\text{Knoten}(m_0, ..., m_{n-1}) = \prod_{i=0}^{n-1}(m_i + 1) \tag{2.9}$$

Dies entspricht der Kombination aller möglichen Generalisierungen. Wird die Quersumme der Generalisierungslevel als Ordnungskriterium für die Knoten verwendet ergeben sich Level. Alle Knoten auf einem Level besitzen dieselbe Quersumme. Die Anzahl der dabei entstehenden Level ist gegeben durch:

$$\#\text{Level}(m_0, ..., m_{n-1}) = (\sum_{i=0}^{n-1} m_i) + 1 = \text{Höhe} = v \tag{2.10}$$

Der Eingangsgrad und Ausgangsgrad eines beliebigen Knoten in dem Verband ist gegeben als:

$$\text{Ausgangsgrad}(k_0, ..., k_{n-1}) = \sum_{i=0}^{n-1} I((m_i - k_i) > 0) \tag{2.11}$$

$$\text{Eingangsgrad}(k_0, ..., k_{n-1}) = \sum_{i=0}^{n-1} I(k_i > 0) \tag{2.12}$$

$I(x)$ stellt die Indikatorfunktion dar, die 1 zurückliefert, falls die Bedingung x wahr ist, sonst 0. Die Anzahl der erreichbaren Knoten, ausgehend von einem Knoten k, ist für die nach unten erreichbaren, also spezialisierten Knoten definiert als:

$$\#\text{ErreichbarAbwärts}(k_0, ..., k_{n-1}) = (\prod_{i=0}^{n-1}(k_i + 1)) - 1 \tag{2.13}$$

Die Anzahl der nach oben erreichbaren Knoten ist gegeben als:

$$\#\text{ErreichbarAufwärts}(k_0, ..., k_{n-1}) = (\prod_{i=0}^{n-1}(m_i - k_i + 1)) - 1 \tag{2.14}$$

Die zugrunde liegende Idee der Gleichungen 2.13 und 2.14 besteht darin, den Ausgangsknoten als Supremum oder Infimum eines Teilverbandes zu sehen. Ausgehend von diesem wird die Anzahl der Knoten in dem Teilverband berechnet und jeweils der Ausgangsknoten von dieser abgezogen.

Die Anzahl der verschiedenen Strategien, mit der Bedingung, dass sich je zwei Strategien um mindestens einen Knoten unterscheiden, ist gegeben durch:

$$\#\text{Strategien}(m_0, ..., m_{n-1}) = \frac{(\sum_{i=0}^{n-1} m_i)!}{\prod_{i=0}^{n-1}(m_i!)} \tag{2.15}$$

Die Anzahl an möglichen Teilverbänden, mit der Bedingung, dass sich die Menge der Knoten jedes Teilverbandes um jeweils mindestens einen Knoten unterscheidet, wird berechnet durch:

$$\#\text{Teilverbände}(m_0, ..., m_{n-1}) = \prod_{i=0}^{n-1}\binom{m_i + 2}{2} - \prod_{i=0}^{n-1}(m_i + 1) \tag{2.16}$$

Die Anzahl der Teilverbände mit nur einem Knoten entspricht der Anzahl der Knoten in dem Verband. Diesen Wert kann man zu oben angegebener Formel addieren, um die Anzahl der Teilverbände zu erhalten, in welcher auch die einelementigen Teilverbände eingerechnet wurden. Hierbei ist $\binom{a}{b}$ als 0 definiert, wenn $a < 0$. Die beiden obigen Formeln wurden durch empirische Tests ermittelt.

Eine weitere Eigenschaft des Verbandes ist die Anzahl der Knoten auf einem Level. Dieser Wert erlaubt es außerdem, eine Laufzeitabschätzung für einen optimalen Algorithmus, der eine optimale Lösung sucht, im worst-case anzugeben. Die Anzahl der Knoten auf einem Level l ist gegeben durch:

$$\#\text{KnotenLevel}(l) = \sum_{i=0}^{n} (-1)^i \sum_{j=1}^{\binom{n}{i}} \binom{n + l - s(i,j)}{n-1} \tag{2.17}$$

$s(t, j)$ ist hierbei definiert ist als die j-te Summe von t $(m_i + 1)$s [66]. Brown [66] berechnet mit dieser Formel die Anzahl der Möglichkeiten eine gegebene Anzahl von Bällen auf Urnen zu verteilen, die jeweils eine beschränkte Kapazität haben. Dieses Problem ist Äquivalent mit der hier betrachteten Anzahl von Knoten auf einem Level. Die Anzahl der Urnen entspricht der Anzahl der Quasi-Identifikatoren. Die Kapazität einer Urne entspricht dem maximalen Level eines Quasi-Identifikators und die Anzahl der gegebenen Bälle entspricht der Quersumme auf dem Level.

Brown [66] schätzt zudem die Anzahl der Möglichkeiten nach oben ab. Abgeleitet aus seiner Abschätzung kann hier die Anzahl der Knoten nach oben abgeschätzt werden durch:

$$\#\text{KnotenLevel}(l) \leq \sum_{i=0}^{n} (-1)^i \binom{n}{i} \binom{n + l - i(max(m_0, ..., m_{n-1}) + 1)}{n-1} \tag{2.18}$$

Um die optimale Lösung bei einem monotonen Kriterium mit monotonen Generalisierungshierarchien zu finden, muss ein Schnitt durch den Verband gefunden werden, welche die anonymen von den nicht-anonymen Knoten trennt. Da jede Strategie, bei einem monotonen Kriterium, genau einen Übergang von anonym zu nicht anonym hat, muss eine solche Trennung existieren. Bei einem optimalen Algorithmus müssen also im schlimmsten Fall somit mindestens die beiden benachbarten Knoten, wo anonym und nicht anonym beieinanderliegen, getestet werden. Der längste mögliche Schnitt würde also im schlechtesten Fall zwischen den beiden Leveln mit den meisten Knoten liegen. Ausgehend von Formel 2.17 kann man beobachten, dass die meisten Knoten auf dem mittleren Level, d. h. auf dem Level wo die Höhen der einzelnen Quasi-Identifikatoren auf die halbe Generalisierungshöhe generalisiert wurden, liegen. Damit ist die minimale Anzahl an Tests, die für die Optimale Lösung nötig sind, gegeben durch $\#KnotenLevel(v)+\#KontenLevel(w)$. V und w sind die beiden Level mit den meisten Knoten. V ist definiert als $v = \lfloor \frac{AnzahlLevel(m_0,...,m_{n-1})}{2} \rfloor$ und $w = v - 1$.

Für den Beispielverband aus Abbildung 2.13 gilt: $n = 3$, $m_0 = 2$,$m_1 = 1$ und $m_3 = 5$. Daraus folgt, dass bei dem Beispielverbund $\#KnotenLevel(4) = \#KnotenLevel(3) = 6$ dass 12 Überprüfungen auf Anonymität, im schlimmsten Fall eine untere Schranke darstellt, um eine optimale Lösung bei einem monotonen

Eigenschaft	Wert
Anzahl Knoten	36
Anzahl Level	9
Ausgangsgrad (1,0,3)	3
Eingangsgrad (1,0,3)	2
Anzahl Knoten erreichbar abwärts von (1,0,3)	7
Anzahl Knoten erreichbar aufwärts von (1,0,3)	11
Anzahl Strategien	168
Anzahl Teilverbände	342
Anzahl Knoten Level (4)	6

Abbildung 2.14: Eigenschaften des Beispielverbandes

Kriterium zu finden. Eine Übersicht über die Eigenschaften des Beispielverbandes ist in Abbildung 2.14 dargestellt.

Interessant anzumerken hierbei ist, dass die Anzahl der möglichen Strategien sehr schnell sehr viel größer wird als die Anzahl der Teilverbände. Diesen Umstand nutzt z. B. der OLA Algorithmus (siehe Abschnitt 3.5.3) aus. Nimmt man nur einen Quasi-Identifikator, z. B. die Diagnose aus dem Beispieldatensatz mit drei Leveln hinzu, entsteht ein Verband mit 108 Knoten, 7560 möglichen Strategien, aber nur 2268 Teilverbänden.

2.5.2 Unterdrückung als Anonymisierungsverfahren

Um gemäß der genannten Anonymisierungskriterien eine geforderte Gruppierung zu erreichen, lässt sich auch das Verfahren der Unterdrückung einsetzen, die Ausreißer entfernt. Dieses Verfahren bewirkt, dass die Generalisierung, vor allem bei multidimensionaler, globaler Recodierung, verringert werden kann und damit oft mehr Information des Ursprungsdatensatzes erhalten bleiben kann. Hierbei kann zwischen Tupel-Unterdrückung, Wert-Unterdrückung und Zellen-Unterdrückung unterschieden werden [53]. Zell-Unterdrückung entfernt nur beliebige einzelne Zellen und Wert-Unterdrückung entfernt alle Werte eines gegebenen Wertes. Tupel-Unterdrückung entfernt immer die gesamte Zeile. Hierzu wird normalerweise ein Parameter festgesetzt, welcher die Anzahl der maximal zulässigen unterdrückten Zeilen angibt (z. B. 2 % der Zeilen dürfen unterdrückt werden). Der Fokus liegt im Folgenden auf der Unterdrückung ganzer Zeilen (Tupel-Unterdrückung). Hierbei wird ein Parameter festgelegt $0 \leq s < 1$, damit wird ein Datensatz mit m Zeilen auch dann noch als anonym angesehen, wenn $\lfloor s * m \rfloor$ Zeilen das Kriterium nicht erfüllen. Diese Zeilen werden vor Herausgabe aus dem Gesamtdatensatz entfernt.

2.6 Methoden zur Messung des Informationsverlustes

Generalisierung und Unterdrückung führen in offensichtlicher Weise zu einem Informationsverlust. Da dieser unter Gesichtspunkten der Anwendung unerwünscht ist und sich zudem Anonymitätskriterien auf verschiedene Arten umsetzen lassen, besitzen Metriken eine hohe Bedeutung. Es wurden in der Literatur verschiede-

ne Metriken zum Messen des Informationsverlustes vorgeschlagen und diskutiert. Metriken können dabei entweder nur Metainformationen der Generalisierung verwenden, was eine von den eigentlichen Daten unabhängige Berechnung erlaubt oder aber Vergleiche zwischen den Originaldaten und den Ergebnisdaten erfordern. Auch Unterdrückung wird bei den Metriken unterschiedlich behandelt, einige „bestrafen“ unterdrückte Tupel und andere behandeln unterdrückte Tupel als wären sie maximal generalisiert.

Ähnlich wie bei den Generalisierungshierarchien können auch Metriken monoton oder nicht monoton sein. In dieser Arbeit werden solche Metriken als monoton bezeichnet, deren Wert mit zunehmender Generalisierung monoton zu- oder abnimmt. Zunehmende Generalisierung ist hierbei definiert als ein Pfad von der aktuellen Generalisierung zu der maximal möglichen. Die Unterscheidung zwischen monotonen und nicht-monotonen Metriken ist wichtig, da bei monotonen Metriken, wie bei monotonen Generalisierungshierarchien, Teile des Suchraumes ausgeschlossen werden können. Zudem ist eine monotone Metrik in vielen Fällen intuitiver, da diese bei stärker generalisierten Daten auch einen höheren Informationsverlust berechnet.

Definition 2.5 (*Monotone Metriken*)

Sei T eine Tabelle, $g(T)$ und $g'(T)$ zwei Generalisierungen von T. Eine Metrik $M(x)$ ist monoton steigend genau dann, wenn $M(g(T)) \geq M(g'(T))$ ist, wobei g' eine größere Generalisierung als g ist. M ist monoton fallend, wenn $M(g(T)) \leq M(g'(T))$ und g' eine größere Generalisierung als g ist.

Wie im Abschnitt 2.5.2 beschrieben, können oft Ausreißer aus dem Datensatz gefiltert werden, um die Generalisierung der restlichen Tupel geringer ausfallen zu lassen. Manche der Metriken berücksichtigen diese unterdrückten Tupel in der Berechnung, diese Metriken sind dann meist nicht-monoton. Metriken können also mittels drei Achsen kategorisiert werden: Abhängigkeit von den Daten, Monotonie und die Berücksichtigung der Unterdrückung. Eine Tabelle mit den benutzten Metriken und ihren Eigenschaften ist in Tabelle 2.15 dargestellt.

Metrik	Monoton	Abhängig	Unterdrückung
HM	X	-	-
PM*	X	-	-
PM	-	X	X
LM	X*	X	X
AECS	X*	X	-
DM	-	X	X
DM*	X	X	-
EM	-	X	X
NUEM	X	X	-

Abbildung 2.15: Implementierte Metriken

2.6.1 Höhe

Sehr einfach ist die Höhenmetrik (height metric, HM) [60]. Sie basiert auf der Beobachtung, dass niedrigere Level bei der Generalisierung durch einen geringeren

Informationsverlust charakterisiert sein sollten. Die Höhen Metrik berechnet hierzu die Summe der einzelnen Generalisierungshierarchien.

$$M_{\mathrm{HM}}(k_0, ..., k_{n-1}) = \sum_{i=0}^{n-1}(k_i) \tag{2.19}$$

Hierbei bezeichnet n die Anzahl der Spalten/Quasi-Identifikatoren und k_x bezeichnet den Level/die Höhe der Generalisierung bei Attribut x. Der Wertebereich der Metrik liegt somit zwischen $0 \leq M_{\mathrm{HM}}(k_0, ..., k_{n-1}) \leq \sum_{i=0}^{n-1} mh_i$, hierbei beschreibt mh_i die maximale Generalisierungshöhe des i-ten Quasi-Identifikators. Je höher der Wert der Metrik umso größer ist der Informationsverlust. Vorteil dieser Metrik ist die Einfachheit und Intuition. Ein Nachteil der Metrik ist, dass es im kompletten Suchraum meist viele Transformationen mit demselben Metrik Wert gibt, die unterschiedlich generalisiert wurden. Außerdem berücksichtigt diese Metrik nicht, die Höhe der einzelnen Hierarchien, da es oft der Fall ist, dass höhere Generalisierungshierarchien langsamer generalisieren.

2.6.2 Präzision

Eine weitere Metrik, welche die verschiedenen Höhen der Generalisierungshierarchien der Quasi-Identifikatoren mitberücksichtigt, ist die Präzision (Precision) Metrik (PM) [58]. Die Metrik normiert die jeweiligen Generalisierungslevel mit ihrer jeweiligen maximalen Höhe. Sie ist somit definiert als:

$$M_{\mathrm{PM}}(k_0, ..., k_{n-1}) = 1 - \frac{\sum_{i=0}^{n-1}\sum_{j=0}^{m-1}\frac{k_{ij}}{mh_i}}{m \cdot n} \tag{2.20}$$

Hierbei bezeichnet n wiederum die Anzahl der Spalten und m bezeichnet die Anzahl der Zeilen. K_{xy} bezeichnet den Level/die Höhe der Generalisierung bei Zelle (x, y). mh_i entspricht hierbei der maximalen Generalisierungshöhe. Der Wertebereich der Präzision Metrik liegt somit zwischen $0 \leq M_{\mathrm{PM}}(k_0, ..., k_{n-1}) \leq 1$, wobei die original Tabelle (nicht generalisiert) den Wert 1 erhält und die maximal generalisierte Tabelle den Wert 0. Je niedriger der Wert um so höher ist hierbei also der Informationsverlust.

Im Falle von globaler Recodierung, und ohne Berücksichtigung von unterdrückten Tupeln, kann diese Metrik vereinfacht werden zu:

$$M_{\mathrm{PM}^*}(k_0, ..., k_{n-1}) = 1 - \frac{\sum_{i=0}^{n-1}\frac{k_i}{mh_i}}{n} \tag{2.21}$$

Die beiden Metriken M_{HM} und M_{PM^*} nutzen nur die vom Nutzer definierten Generalisierungshierarchien und sind somit bei globalem recodieren unabhängig von den eigentlichen Daten. In [67] wird diese Metrik auch als „tree measure“ bezeichnet.

2.6.3 Verlust

Eine weitere Metrik, welche die Generalisierungshierarchien nutzt, um den Informationsverlust bei der Generalisierung zu messen, ist die von Iyengar et al. [68]

vorgeschlagene Verlust (Loss) Metrik (LM). Sie ist definiert als:

$$M_{\mathrm{LM}}(k_0, ..., k_{n-1}) = \frac{\sum_{i=0}^{n-1} \sum_{j=0}^{m-1} \frac{L_{R_{ij}} - 1}{L_{top_i} - 1}}{m \cdot n} \tag{2.22}$$

Hierbei bezeichnet L_x die Anzahl der Blätter des (unter-)Baumes mit der Wurzel x. L_{top_i} bezeichnet somit die Gesamtanzahl an Blättern in der Generalisierungshierarchie für die Spalte i. Diese Metrik kann Unterdrückung berücksichtigen, wenn man unterdrückte Zellen als maximal generalisiert annimmt, diese fließen also mit dem Wert 1 in die Berechnung mit ein. Damit ist der Informationsverlust die Summe über den normalisierten Wert und über alle Spalten. Er liegt in dem Bereich zwischen 0 und 1. Hierbei repräsentiert ein höherer Wert einen größeren Informationsverlust.

Neben der Messung des Informationsverlustes bei kategorischen Werten wird auch eine abgewandelte Form für die intervallbasierte Generalisierung von numerischen Werten vorgestellt:

$$M_{\mathrm{LM}_{\mathrm{numeric}}}(k_0, ..., k_{n-1}) = \frac{\sum_{i=0}^{n-1} \sum_{j=0}^{m-1} \frac{U'_{ij} - L'_{ij}}{U_i - L_i}}{m \cdot n} \tag{2.23}$$

Hierbei definiert U'_{ij} die obere Grenze des Intervalls, in welches der Wert ij generalisiert wurde und L_{ij} die untere Grenze des Intervalls. L_i und U_i sind dazu analog die Unter- und Obergrenze der Werte in der ungeneralisierten Tabelle.

2.6.4 Durchschnittliche Äquivalenzklassengröße

Informationsverlust kann auch durch die Größen der entstandenen Äquivalenzklassen gemessen werden. In LeFevre et al. [69] wurde eine Metrik vorgestellt welche über die normalisierte, durchschnittliche Äquivalenzgruppengröße (AECS) definiert ist. Diese Metrik hat außerdem die Besonderheit, dass indirekt die Qualität der Anonymisierung einfließen kann, da auch mit der geforderten Gruppengröße normalisiert wird:

$$M_{\mathrm{AECS}}(k_0, ..., k_{n-1}, p) = \frac{m}{EC(k_0, ..., k_{n-1}) \cdot p} \tag{2.24}$$

Hierbei bezeichnet m die Anzahl der Zeilen im Datensatz, $E(x)$ die Anzahl der Klassen im Zustand x und p ist die minimale Größe der Äquivalenzklasse für das jeweilige Anonymitätskriterium. Im Falle von k-Anonymität gilt $p = k$ und im Falle von ℓ-Diversität gilt $p = \ell$. Bei Kriterien, die keine Gruppengröße fordern, kann die Normalisierung entfallen. AECS bezeichnet somit die normalisierte, durchschnittliche Äquivalenzklassengröße nach der Transformation. Da es schon im Original Datensatz Äquivalenzklassen geben kann, gibt es keinen immer gültigen Minimalwert. Wenn im original Datensatz keine Tupel in eine Äquivalenzklasse fallen, kann die Metrik minimal den Wert $\frac{1}{p}$ und maximal den Wert $\frac{m}{p}$ annehmen.

2.6.5 Discernibility

Bayardo et al. schlägt in [70] die Discernibility Metrik (DM) vor. Die Metrik betrachtet wieder die Äquivalenzklassen nach der Transformation. Die Idee hierbei ist,

eine „Strafe“ für nicht unterscheidbare Tupel einzuführen. Sie ist definiert als:

$$M_{\mathrm{DM}}(k_0, ..., k_{n-1}) = \sum_{\forall E | E \notin O} (|E|^2) + \sum_{\forall E | E \in O} (m \cdot |E|) \tag{2.25}$$

Hierbei beschreibt $|E|$ jeweils die Größe der Äquivalenzklasse und m die Gesamtanzahl der Tupel und O definiert die Menge der unterdrückten Äquivalenzklassen. Der erste Term summiert somit die Quadrate der Größen der Äquivalenzklassen, die nicht unterdrückt sind. Durch den zweiten Term werden die unterdrückten Tupel stärker „bestraft“ als die nicht unterdrückten, da angenommen wird, dass unterdrückte Tupel von keinem anderen Tupel im Datensatz unterschieden werden können. Da diese Metrik nicht normiert ist, können nur worst- und best-case Schranken für den Minimal- und Maximalwert angegeben werden. Wenn im ungeneralisierten Fall keine Äquivalenzklasse größer als eins ist, beträgt der minimale Wert der Metrik m, hierbei ist m die Anzahl der vorhandenen Tupel im Datensatz. Der Maximalwert der Metrik beträgt $m * m$.

Der zweite Summand, welcher die unterdrückten Tupel berücksichtigt, zerstört die Monotonie der Metrik, weshalb in [56] eine monotone Version vorgestellt wurde, welche einfach den letzten Term weglässt. Damit ist die monotone Discernibility Metrik DM* definiert als:

$$M_{\mathrm{DM}^*}(k_0, ..., k_{n-1}) = \sum_{\forall E | E \notin O} (|E|^2) \tag{2.26}$$

DM* ist natürlich nur monoton, wenn auch die verwendeten Generalisierungshierarchien monoton sind, da sonst, mit steigender Generalisierung die Äquivalenzklassen kleiner werden könnten. Bei DM* gelten dieselben Minimal- und Maximalwerte wie bei DM.

2.6.6 Entropie

Das für den Informationsgehalt verbreitete Maß der Entropie wurde ebenfalls als Basis für eine Metrik vorgeschlagen [71]. Die Entropie Metrik (EM) bei globaler Recodierung ist dabei definiert als:

$$M_{\mathrm{EM}}(k_0, ..., k_{n-1}) = -\sum_{i=0}^{n-1} \sum_{j=0}^{m-1} X(i, j, k_i) \cdot log_2 X(i, j, k_i) \tag{2.27}$$

hierbei gilt, dass

$$X(i, j, k) = \frac{\sum_{l=0}^{m-1} I(R(i, l, 0) = R(i, j, 0))}{\sum_{l=0}^{m-1} I(R(i, l, k) = R(i, j, k))} \tag{2.28}$$

Zudem gilt, dass für einen Datensatz mit m Zeilen und n Spalten $R(i, j, k)$, mit $0 \geq i \geq n-1$ und $0 \geq j \geq m-1$, den Wert der i-ten Spalte in Zeile j generalisiert auf das Level k. $I(x)$ ist als Indikatorfunktion definiert und ist 1, wenn x wahr ist, 0 andernfalls.

In [67] zeigen Gionis et al., dass diese Metrik nicht monoton ist und definieren eine leicht abgewandelte monotone Metrik mit dem Namen non-uniform Entropie (NUEM) wo der erste Faktor weggelassen wird:

$$M_{\text{NUEM}}(k_0, ..., k_{n-1}) = -\sum_{i=0}^{n-1}\sum_{j=0}^{m-1} log_2 X(i, j, k_i) \tag{2.29}$$

Hierbei bezeichnet wieder n die Anzahl der Quasi-Identifikatoren und m die Anzahl der Zeilen. $I(x)$ ist wieder die Indikatorfunktion und ist 1, wenn x wahr ist, 0 andernfalls. Bei NUEM bedeutet also ein höherer Wert einen größeren Informationsverlust. Die NUE-Metrik ist auch nicht monoton, wenn Unterdrückung erlaubt ist [67]. Der Minimalwert sowohl von M_{EM} als auch von M_{NUEM} beträgt 0 und tritt auf, wenn die Identität des Datensatzes verglichen wird. Eine Obergrenze kann abgeschätzt werden für M_{NUEM} eines Datensatzes mit m Zeilen und n Spalten durch $-m \cdot n \cdot log_2 \frac{1}{m} = m \cdot n \cdot log_2 m$. Diese Abschätzung kommt dadurch zustande, dass $X(i, j, k_i)$ minimiert wird. Für M_{EM} beträgt die abgeschätzte Obergrenze analog $-m \cdot n \cdot \frac{1}{m} \cdot log_2 \frac{1}{m} = n \cdot log_2 m$.

2.7 Zusammenfassung

Dieses Kapitel gibt einen Überblick über die in dieser Arbeit verwendeten Methoden. Das betrachtete Bedrohungsmodell deckt die Bedrohungen der Identitätsaufdeckung, Attributsaufdeckung und Mitgliedschaftsaufdeckung ab. Die Anonymisierungsmethoden werden klassifiziert, um einen Überblick über die Möglichkeiten zu geben. Diese Möglichkeiten werden mit den Anforderungen in der medizinischen Domäne in Beziehung gesetzt. Daraufhin werden die, für die medizinische Domäne, am besten geeigneten Anonymisierungskriterien vorgestellt. Gegen jede der vorher beschriebenen Bedrohungen werden Kriterien vorgestellt, mit denen ein Datenschatz geschützt werden kann. Um die Datensätze entsprechend den Kriterien schützen zu können, müssen diese verändert werden. Für die medizinische Domäne am besten geeignet ist die Generalisierung und Unterdrückung von Tupeln. Daten können auf verschiedene Arten generalisiert werden. Die Eigenschaften des entstehenden Lösungsraumes und seine Größe sind beschrieben. Um den mit Generalisierung und Unterdrückung einhergehenden Informationsverlust messbar zu machen, werden gängige Metriken aus der Literatur vorgestellt. Die vorgestellten Kriterien, Verfahren und Metriken bilden bei den neu entwickelten Algorithmen und Protokollen, die im Folgenden vorgestellt werden, die Grundlage.

KAPITEL 3

Anonymisierung von lokal vorliegenden Daten

Bei dem heute typischen Data Sharing werden innerhalb oder außerhalb eines Forschungsverbunds Daten, oft Mikrodaten, zur Verfügung gestellt. Dabei ist die Anonymisierung vor Herausgabe eine wichtige Anforderung. Die Daten, die anonymisiert werden müssen, werden vor diesem Schritt von einer verantwortlichen Stelle verwaltet. Die Herausforderung liegt hierbei in dem Kompromiss zwischen ausreichender Anonymisierung einerseits und geringem Informationsverlust andererseits. Den Idealfall stellt eine im Sinne des BDSG faktische Anonymisierung, bei gleichzeitig im Sinne des Forschungszwecks nicht vorhandenen oder zumindest irrelevanten Informationsverlust, dar. Die Metriken für den Informationsverlust sollten an die inhaltlichen Anforderungen der Forscher anpassbar sein. Ein Anonymisierungstool sollte zudem eine Auswahl von Anonymisierungskriterien erlauben, um, je nach Anforderung, unterschiedliche Datenschutzgarantien zu ermöglichen. Ein iteratives Vorgehen kann angemessen sein, um das angestrebte Optimum zwischen Anonymisierungsparametern, z. B. wachsendes k bei k-Anonymität oder Anpassung des Unterdrückungsschwellwertes, und Anforderungen an die Datenqualität, z. B. Priorisierung eines Attributes, zu erreichen. Bei diesem iterativen Vorgehen spielt die Effizienz, insbesondere die benötigte Laufzeit für eine Iteration, eine große Rolle.

Im Rahmen dieser Arbeit wurde eine sehr effiziente Familie von neuen Algorithmen entwickelt. Die entwickelten Algorithmen sind die ersten ihrer Art, welche sowohl bei monotonen als auch nicht-monotonen Kriterien und Metriken sehr effizient eine global optimale Lösung des Anonymisierungsproblems finden. Dabei wurden die Algorithmen so generisch gestaltet, dass sie eine Vielzahl von Anonymisierungskriterien und Informationsverlustmetriken unterstützen. Außerdem wird in diesem Kapitel die Monotonie von Anonymisierungskriterien, bei Nutzung von Unterdrückung zusammen mit Generalisierung, erstmalig untersucht und bei den bekannten Kriterien bewiesen, dass diese nicht monoton sind, wenn Unterdrückung mit einer oberen Schranke verwendet wird. Um das Framework und den Algorithmus zu evaluieren, wurde eine Auswahl an Anonymisierungskriterien implementiert. Die Auswahl erfolgte dergestalt, dass es mindestens ein funktionierendes Kriterium für die in Abschnitt 2.1 beschriebenen Bedrohungen gibt. Als Informationsverlustmetriken wurden die bekanntesten Metriken implementiert (siehe Abschnitt 2.6). Es wurden Metriken berücksichtigt, die sowohl nur auf den Verwaltungsstrukturen der Kriterien arbeiten und daraus den Informationsverlust ableiten, als auch Metriken, die anhand der eigentlichen Daten versuchen den Informationsverlust zu berechnen. In dem Abschnitt 3.9 wird die Familie der in dieser Arbeit entwi-

ckelten Algorithmen mit den bekannten State-of-the-Art Algorithmen verglichen. Für den vollständig monotonen Fall (siehe Abschnitt 3.2) wird gezeigt, dass der neu entwickelte Algorithmus, bezüglich der Laufzeit und des Speicherverbrauches, in der hier genutzten Implementierung den State-of-the-Art Algorithmen überlegen ist. Außerdem ist der neue Algorithmus hinsichtlich der benötigten Anzahl an Anonymitätstests vergleichbar mit dem derzeit besten Algorithmus OLA. In allen anderen (nicht)-monotonen Fällen gab es, vor der hier vorgestellten Familie der Flash Algorithmen, keinen Algorithmus, der die optimale Lösung anders als mittels vollständigem Durchsuchen des Suchraumes fand. Somit kann auch hierbei die Familie der Flash Algorithmen als neuer Benchmark angesehen werden. Ergebnisse dieses Kapitels wurden publiziert in [72], [73], [74], [75] und [76].

3.1 Informationsverlust bei Einsatz von Unterdrückung

	Entropie					Precision				
Kriterium	ADULT	CUP	FARS	ATUS	IHIS	ADULT	CUP	FARS	ATUS	IHIS
(k)	55%	64%	47%	38%	61%	48%	39%	24%	23%	39%
(l)	64%	63%	50%	78%	62%	57%	38%	26%	54%	41%
(t)	94%	70%	67%	89%	100%	86%	50%	52%	85%	100%
(d)	69%	70%	66%	46%	69%	54%	49%	35%	31%	43%
(k,d)	78%	71%	72%	59%	74%	67%	53%	41%	45%	66%
(k,t)	94%	70%	67%	89%	100%	86%	50%	52%	85%	100%
(k,l)	64%	64%	51%	78%	68%	57%	39%	26%	54%	41%
(l,d)	74%	69%	61%	76%	77%	72%	52%	38%	59%	66%
(t,d)	95%	86%	81%	91%	93%	93%	67%	59%	85%	96%
(k,t,d)	95%	86%	81%	91%	93%	93%	67%	59%	85%	96%
(k,l,d)	74%	71%	64%	76%	77%	72%	53%	39%	59%	67%

	Discernibility					Average equivalence class size				
Criterion	ADULT	CUP	FARS	ATUS	IHIS	ADULT	CUP	FARS	ATUS	IHIS
(k)	16%	6%	27%	41%	79%	4%	1%	9%	10%	13%
(l)	22%	6%	27%	78%	100%	6%	1%	2%	4%	11%
(t)	82%	8%	22%	84%	100%	21%	2%	6%	12%	100%
(d)	40%	21%	37%	58%	99%	13%	3%	10%	10%	11%
(k,d)	43%	11%	17%	63%	65%	24%	2%	14%	23%	16%
(k,t)	82%	8%	22%	84%	100%	21%	2%	6%	12%	100%
(k,l)	22%	6%	28%	78%	100%	6%	1%	2%	4%	15%
(l,d)	28%	9%	17%	73%	66%	15%	1%	5%	9%	18%
(t,d)	79%	19%	46%	86%	100%	29%	11%	16%	23%	51%
(k,t,d)	61%	19%	46%	86%	100%	25%	11%	16%	23%	51%
(k,l,d)	29%	11%	17%	74%	66%	15%	2%	6%	10%	18%

Tabelle 3.1: Informationsverlust für s=5% als Prozentangabe des Informationsverlustes ohne Unterdrückung

In Abschnitt 2.5.2 wurde die häufig getroffene Annahme [56] beschrieben, dass durch Unterdrückung von Tupeln der Informationsverlust verringert werden kann. Dies lässt sich an einem Beispiel gut zeigen. In Tabelle 3.1 ist der Informationsgewinn für fünf Testdatensätze (siehe Abschnitt 3.9.1) dargestellt. Es wurden die vier oft in der Domäne eingesetzten Anonymisierungskriterien, k-Anonymität, ℓ-Diversität, t-Closeness und δ-Präsenz (siehe Abschnitt 2.4) angewendet. Es wurden 5-Anonymität (k), rekursive-(3,4)-Diversität (l), 0.2-Closeness (t) und (0.05,0.15)-Präsenz (d) als Kriterien gewählt. Die Auswahl der Parameter orientiert sich an den Empfehlungen aus der Literatur. Diese Kriterien wurden jeweils auch miteinander kombiniert. Der Informationsverlust wurde ohne Unterdrückung und mit 5 % Unterdrückung berechnet. Als Metriken kamen jeweils Entropie (EM), Precision (PM*), Discernibility (DM) und die durchschnittliche Äquivalenzklassengröße (AECS) zum Einsatz (siehe Abschnitt 2.6). Die Angaben sind als Prozent des Informationsverlustes ohne Unterdrückung angegeben.

Die Tabelle 3.1 zeigt deutlich, dass für moderate Unterdrückung von 5 % der Informationsverlust oft drastisch reduziert wird, was zu einer größeren Nützlichkeit des anonymisierten Datensatzes führt. Bei 5-Anonymität steigt der Informationsgehalt, je nach Metrik, um 36% (CUP, Entropie) bzw. 99% (CUP, AECS). Bei 0.2-Closeness ist der Anstieg des Informationsgewinns am geringsten, im Vergleich zu den einzelnen Kriterien 5-Anonymität, rekursive-(3,4)-Diversität und (0.05,0.15)-Präsenz. Selbst hier liegt der Informationsgewinn bei bis zu 98 % (CUP, AECS). Man kann weiterhin beobachten, dass der größte Datensatz (IHIS) im Schnitt am wenigsten von der Unterdrückung profitiert. Bedingt durch die Größe des Datensatzes lassen sich einfacher, d. h. mit geringerer Generalisierung und weniger Unterdrückung, Äquivalenzklassen finden. In vielen Fällen (d. h. bei 100%) werden, mit und ohne Unterdrückung, oft die gleichen Transformationen gefunden. Generell kann man allerdings bei über 88% der 220 durchgeführten Experimenten eine Verringerung des Informationsverlustes im zweistelligen Prozentbereich zu erkennen.

Die Möglichkeit, einzelne Ausreißer zu unterdrücken, hilft also den Informationsverlust bei dem hier zugrunde liegenden Codierungsmodell drastisch zu verringern. Allerdings hat die Unterdrückung von einzelnen Zeilen Einfluss auf die Monotonie einiger Anonymisierungskriterien, was sich negativ auf die Laufzeit vieler Algorithmen auswirkt.

3.2 Monotonie der Anonymisierungskriterien

Monotonie ist ein wichtiges Kriterium, um effizient eine optimale Lösung für ein gegebenes Anonymitätskriterium finden zu können. Damit können Algorithmen beim Suchen der optimalen Lösung viele potenzielle Transformationen von vornherein ausschließen. Dies ist besonders wichtig, da das Problem, die optimale Lösung für diese Kriterien, wenn Generalisierung verwendet wird, zu finden sehr schwer ist. Für k-Anonymität wurde bewiesen, dass das Problem NP-hart ist [77]. Deshalb versucht man den möglichen Suchraum (vgl. Abbildung 2.5.1.6) so weit wie möglich zu verkleinern.

Die Grundlage hierfür bilden monotone Generalisierungshierarchien (Monohierarchien, siehe auch Abschnitt 2.5.1.5). Viele der weitverbreiteten Kriterien (k-Anonymität, t-Closeness, ℓ-Diversität und δ-Präsenz) sind in ihrer einfachsten

Form, bei Nutzung von monotonen Generalisierungshierarchien, monoton. Ein monotones Kriterium ermöglicht nun vorausschauendes Markieren, d. h. bestimmen der Anonymität eines Zustandes, ohne explizites transformieren und testen, anzuwenden. Dies ist möglich, da durch die Monotonie jeder generalisierte Zustand eines anonymen Zustandes automatisch anonym ist. Folglich ist auch jeder spezialisierte Zustand eines nicht anonymen Zustandes nicht-anonym. Diese Optimierung erlaubt große Teile des gesamten Suchraumes, ohne direkte Prüfung auf Anonymität, der richtigen Klasse zuzuordnen. Ein Beispiel ist in Abbildung 3.1 dargestellt.

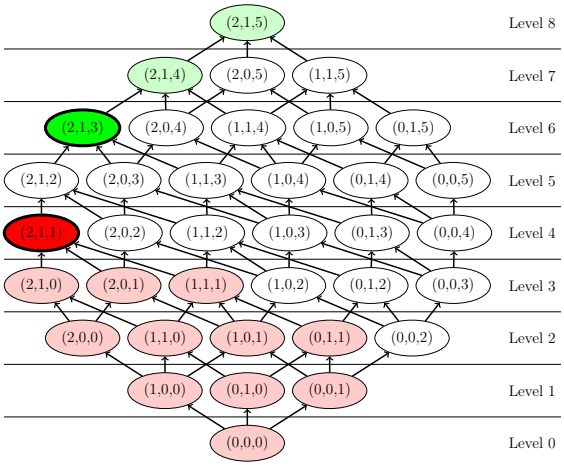


Abbildung 3.1: Beispiel für vorausschauendes markieren

Da der Zustand $(2, 1, 3)$ anonym (dunkelgrün) ist, folgt daraus automatisch, dass sowohl $(2, 1, 4)$ als auch $(2, 1, 5)$ (hellgrün) anonym sind. Analog: da der Zustand $(2, 1, 1)$ nicht anonym (dunkelrot) ist, sind alle Spezialisierungen von diesem auch nicht anonym (hellrot).

Wie bereits erwähnt, sind k-Anonymität, ℓ-Diversität, t-Closeness und δ-Präsenz monoton ohne Unterdrückung. Die dazugehörigen Beweise findet man in den jeweiligen Publikationen. Für k-Anonymität in [61], für Entropie-ℓ-Diversität und Rekursive-(c,l)-Diversität in [44] und in [63] findet sich der Beweis für t-Closeness. Auch das klassische δ-Präsenz [48] ist monoton. Die Modifikation von δ-Präsenz, welche ohne explizites Weltwissen auskommt, ist selbst ohne Unterdrückung nicht monoton [78].

Können nun allerdings, wie in Abschnitt 2.5.2 beschrieben, Tupel unterdrückt werden, verändert dies die Monotonieeigenschaften der Kriterien. Im Folgenden wird die Monotonie dieser Kriterien, bei Einsatz von Unterdrückung, untersucht und bewiesen oder mit Hilfe von Gegenbeispielen widerlegt.

3.2.1 k-Anonymität bei Einsatz von Unterdrückung

Im Falle von unterdrückten Tupeln gilt bei k-Anonymität dasselbe Monotoniekriterium wie im Falle ohne Unterdrückung. Ist die Generalisierung G^* anonym, mit der Bedingung, dass die Anzahl der unterdrückten Tupel kleiner oder gleich der erlaubten Maximalanzahl ist, dann ist auch die Generalisierung G^{**} anonym und es werden hierbei maximal so viele Tupel unterdrückt wie bei G^*. Sei S^{**} eine Spezialisierung von S^* und sei S^* nicht k-anonym, mit weniger als der erlaubten Anzahl

an unterdrückten Tupeln, dann ist auch S^{**} nicht-k-anonym mit weniger als der erlaubten Anzahl an Tupeln.

Der Beweis folgt aus der Monotonie der Generalisierungshierarchien. Durch diese Bedingung können die Äquivalenzklassen von G^{**} nur aus Äquivalenzklassen von G^{*} entstehen. Damit bleiben die Äquivalenzklassen entweder gleich groß oder wachsen, da bei der Generalisierung zwei oder mehrere Äquivalenzklassen zusammenfallen. Selbiges gilt für Spezialisierungen, dabei bleibt die Größe einer Äquivalenzklasse entweder gleich oder wird kleiner. Die Äquivalenzklassengröße der unterdrückten Tupel, bleibt analog dazu gleich oder wird kleiner im Falle einer Generalisierung, da evtl. unterdrückte Tupel weiterhin eine Klasse bilden oder nun in eine Klasse fallen, die nicht mehr unterdrückt wird. Auch hier gilt im umgekehrten Fall, dass die Anzahl der zu unterdrückenden Tupel gleich bleibt, oder zunimmt bei einer Spezialisierung.

3.2.2 ℓ-Diversität bei Einsatz von Unterdrückung

Wie bei k-Anonymität ist auch bei ℓ-Diversität die Monotonie im einfachsten Fall, wo pro Äquivalenzklasse mindestens ℓ verschieden sensible Attributwerte vorhanden sein müssen, gewährleistet. Der Beweis kann ähnlich wie bei der k-Anonymität geführt werden. Ist G^{*} ℓ-divers, dann ist auch jede Generalisierung G^{**} ℓ-divers, da, wegen der Monotonie der Generalisierungshierarchien, wieder nur vorhandene Äquivalenzklassen zusammenfallen können und hierbei die Anzahl verschiedener sensibler Attributwerte stetig zunimmt. Zusätzlich können unterdrückte Werte in die neue Äquivalenzklasse fallen, hierbei bliebt allerdings wieder die Anzahl an unterschiedlichen sensiblen Werten gleich oder nimmt zu und das Kriterium bleibt damit erfüllt. Im umgekehrten Fall, bei dem S^{**} eine Spezialisierung von einem nicht ℓ-diversen Zustand S^{*} ist, ist auch S^{**} nicht ℓ-divers. Auch hier werden durch die Spezialisierung die vorhandenen Äquivalenzklassen kleiner oder bleiben gleich, wodurch in jeder Äquivalenzklasse maximal so viele verschiedene Tupel sind, wie in der passenden Klasse in S^{*}. Damit müssen auch wieder mindestens so viele Tupel unterdrückt werden wie bei S^{*}, die Anzahl der zu unterdrückenden Tupel steigt also.

Weniger Generalisiert

	Alter	Diagnose	ℓ-Diversität	Unterdrückt
0	1-19	Lungenent.	R = $2 < 3 \cdot (1)$;	Nein
1	1-19	Gastritis	E = 1,88	Nein
2	1-19	Lungenent.		
3	20-60	Lungenent.	R = $1 < 3 \cdot (1)$;	Nein
4	20-60	Gastritis	E = 2,0	Nein
5	61-99	Gastritis	R = $10 > 3 \cdot (0)$;	Ja
6	61-99	Gastritis	E = 1,0	Ja
7	61-99	Gastritis		
8	61-99	Gastritis		
9	61-99	Gastritis		
10	61-99	Gastritis		
11	61-99	Gastritis		
12	61-99	Gastritis		
13	61-99	Gastritis		
14	61-99	Gastritis		

Mehr Generalisiert

Alter	Diagnose	ℓ-Diversität	Unterdrückt
⋆	Lungenent.	R = $12 > 3(3)$;	n. möglich
⋆	Gastritis	E = 1,65	n. möglich
⋆	Lungenent.		
⋆	Lungenent.		
⋆	Gastritis		
⋆	Gastritis		
⋆	Gastritis		
⋆	Gastritis		
⋆	Gastritis		
⋆	Gastritis		
⋆	Gastritis		
⋆	Gastritis		
⋆	Gastritis		
⋆	Gastritis		

Abbildung 3.2: Monotonie von ℓ-Diversität bei Einsatz von Unterdrückung, *R = rekursive-(3,2)-Diversität, E = Entropie-1.8-Diversität*

Im Falle von Entropie-ℓ-Diversität und Rekursiver-(c,l)-Diversität allerdings gilt das Monotoniekriterium nicht, wenn die Unterdrückung von Tupeln erlaubt ist. Es folgt der Beweis durch ein Gegenbeispiel. Die Annahme ist, dass Entropie- und Rekursiv-(c,l)-Diversität monoton ist. Dazu müsste jede Generalisierung G^{**} eines entropie- oder rekursiv-(c,l)-diversen Zustand G^{*} auch divers sein.

Gegeben ist der Ausschnitt aus einem Beispieldatensatz, dargestellt in Abbildung 3.2. Das Attribut „Alter“ ist der Quasi-Identifikator, „Diagnose“ bezeichnet das sensitive Attribut. Die rechte Seite ist eine weiter generalisierte Version der linken Hälfte. Die angenommenen Parameter sind $c = 3$ und $l = 2$ für Rekursive-(c,l)-Diversität und $\ell = 1,8$ für Entropie-ℓ-Diversität. Die maximal erlaubte Anzahl an unterdrückten Tupeln ist in beiden Fällen $u = 10$.

Damit ist die linke Seite der in Abbildung 3.2 dargestellten Tabelle sowohl rekursiv-(3,2)-divers als auch entropie-1,8-divers. Die konkreten Werte für die Äquivalenzklassen sind jeweils neben dem Kriterium angegeben (z. B. E = 1,88 bedeutet, dass diese Äquivalenzklasse einen Wert von 1,88 für die Entropie hat, $R = 2 < 3 \cdot (1)$ bedeutet, dass die rekursive c,l Ungleichung erfüllt ist, das häufigste Element kommt also zwei mal vor, was kleiner ist als 3 (=c) mal die Summe der Anzahl der restlichen Elemente (1)). Die beiden ersten Gruppen, bestehend aus den Tupeln $\{0, 1, 2\}$ und $\{3, 4\}$ erfüllen beide Diversitätsbedingungen. Die dritte Gruppe $\{5, 6, 7, 8, 9, 10, 11, 12, 13, 14\}$ erfüllt das Anonymitätskriterium nicht, sie kann aber unterdrückt werden, da sie genau zehn Elemente enthält. Somit ist die Tabelle, bestehend aus den Tupeln $0 - 4$, ℓ-divers.

Sei nun die rechte Seite der Tabelle 3.2 eine Generalisierung der linken Seite, wo der Quasi-Identifikator Alter zu einer einzigen Gruppe $\star$ generalisiert wird. Die so entstehende Tabelle ist weder rekursiv- noch entropie-ℓ-divers, da für die rekursive-(c,l)-Diversität gilt, dass $12 > 9$ ist. Außerdem ist die Entropie dieser Gruppe $1,65$, was kleiner als der geforderte Wert von $1,8$ ist. Um diese Tabelle noch ℓ-divers zu bekommen, müssten alle 15 Tupel unterdrückt werden, was über dem erlaubten Maximum von 10 Elementen liegt. Somit ist die rechte Seite der Tabelle 3.2 eine nicht ℓ-diverse Generalisierung der linken Seite. □

Weniger Generalisiert

	Alter	Diagnose	t-Closeness	Unterdrückt
0	1-19	Lungenentzündung	H=0,47;	Ja
1	1-19	Gastritis	G=0,47	Ja
2	1-19	Lungenentzündung		
3	20-60	Lungenentzündung	H=0,3;	Nein
4	20-60	Gastritis	G=0,3	Nein
5	61-99	Gastritis	H=0,2;	Nein
6	61-99	Gastritis	G=0,2	Nein
7	61-99	Gastritis		
8	61-99	Gastritis		
9	61-99	Gastritis		
10	61-99	Gastritis		
11	61-99	Gastritis		
12	61-99	Gastritis		
13	61-99	Gastritis		
14	61-99	Gastritis		

Mehr Generalisiert

Alter	Diagnose	t-Closeness	Unterdrückt
1-60	Lungenentzündung	H=0,4;	n. möglich
1-60	Gastritis	G=0,4	n. möglich
1-60	Lungenentzündung		
1-60	Lungenentzündung		
1-60	Gastritis		
61-99	Gastritis	H=0,2;	Nein
61-99	Gastritis	G=0,2	Nein
61-99	Gastritis		
61-99	Gastritis		
61-99	Gastritis		
61-99	Gastritis		
61-99	Gastritis		
61-99	Gastritis		
61-99	Gastritis		

Abbildung 3.3: Monotonie von 0.3-Closeness bei Einsatz von Unterdrückung, *H = Hierarchische-EMD, G = Gleich-EMD*

3.2.3 t-Closeness bei Einsatz von Unterdrückung

Für t-Closeness ohne Unterdrückung ist die Monotonie in [63] für die Earth-Mover's-Distance (EMD) bewiesen worden. Wenn die EMD als Ähnlichkeitsmaß zwischen zwei Verteilungen genutzt wird und wenn gleichzeitig eine beschränkte Anzahl von unterdrückten Tupeln erlaubt ist, ist das Kriterium nicht monoton. Im Folgenden der Beweis durch ein Gegenbeispiel. Ist das Kriterium monoton, gilt für jede Generalisierung von G^{**} eines anonymen, also t-closen, Zustandes G^*, dass dieser auch t-close ist. In Tabelle 3.3 ist ein Ausschnitt aus einem Beispieldatensatz dargestellt.

„Alter" ist wiederum der Quasi-Identifikator und „Diagnose" das sensible Attribut. Die linke Tabelle ist G^*, die rechte Tabelle ist die weiter generalisierte Version G^{**}. Gegeben sei der Parameter $t = 0,3$ und $u = 3$ als maximale Anzahl Tupel für die erlaubte Unterdrückung. Es wird auf 0.3-close getestet, sowohl mit der Gleich-EMD als auch mit der Hierarchischen-EMD. Für die Berechnung der hierarchischen EMD verwendete Hierarchie ist in Abbildung 2.12 dargestellt. Für die Gleich-EMD erfüllen die Tupel $\{0, 1, 2\}$ das Kriterium nicht, können aber unterdrückt werden, da die Anzahl der Tupel $x \leq 3$ ist. Diese drei Tupel sind 0.3-close hinsichtlich der Hierarchischen-EMD. Die Tupel $\{3, 4\}$ und $\{5, 6, 7, 8, 9, 10, 11, 12, 13, 14\}$ erfüllen jeweils beide t-closeness Kriterien. Die linke Seite ist somit 0.3-close für beide Kriterien.

Die rechte Seite sei nun eine Generalisierung der linken, bei der das Attribut „Alter" in zwei Gruppen $1-60$ und $61-99$ generalisiert wird. Diese Repräsentation des Datensatzes ist nun nicht mehr 0.3-close. Die Tupel $\{0, 1, 2, 3, 4\}$ sind weder für die Gleich-EMD noch für die hierarchische-Distanz 0.3-close und können auch nicht unterdrückt werden, da diese Gruppe mehr als die maximal erlaubte Anzahl an zu unterdrückenden Tupeln enthält. Damit ist auch der komplette Datensatz nicht 0.3-close. □

3.2.4 δ-Präsenz bei Einsatz von Unterdrückung

Für δ-Präsenz ohne Unterdrückung ist die Monotonie in [48] bewiesen worden. Das Kriterium der Nachfolgearbeit, welches ohne Weltwissen auskommt, ist selbst ohne Unterdrückung nicht monoton [78].

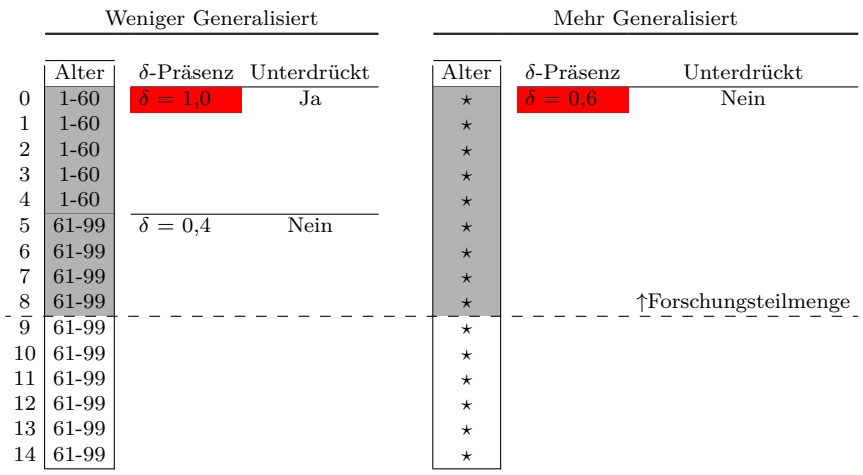


Abbildung 3.4: Monotonie von (0,0;0,5)-Präsenz bei Einsatz von Unterdrückung

Das δ-Präsenz Kriterium aus [48] ist allerdings nicht monoton mit Unterdrückung. Auch hier der Beweis durch ein Gegenbeispiel. Gegeben sei die Tabelle aus Abbildung 3.4 mit dem Quasi-Identifikator Alter, die Forschungsteilmenge besteht aus den Tupeln $\{0, 1, 2, 3, 4, 5, 6, 7, 8\}$ (grau markiert). Die Parameter seien $dmin = 0,0$, $dmax = 0,5$ und die maximal erlaubte Anzahl an erlaubter Unterdrückung $u = 5$. Das Weltwissen besteht aus den Tupeln $0 - 14$.

Die linke Seite der Abbildung ist (0,0;0,5)-präsent. Die Tupel der Forschungsteilmenge $\{0, 1, 2, 3, 4\}$ sind mit $\delta = 1,0$ zwar nicht δ-präsent, können aber unterdrückt werden. Die restlichen Tupel der Forschungsteilmenge $\{5, 6, 7, 8\}$ liegen mit $\delta = 0,4$ im geforderten Intervall und sind somit δ-präsent. Damit müsste jede Generalisierung dieser Daten auch δ-präsent sein, wenn das Kriterium monoton wäre. Sei nun die rechte Seite eine Generalisierung der linken, wo das Alter maximal generalisiert ist. Diese Transformation ist nicht (0,0;0,5)-präsent, da $\delta = 0,6$ was größer als $0,5$ ist. Diese Äquivalenzklasse kann auch nicht unterdrückt werden, da die maximal erlaubte Anzahl an unterdrückten Tupeln fünf ist. Somit ist die Generalisierung nicht δ-präsent obwohl eine Spezialisierung δ-präsent ist. □

Wie beschrieben sind also die Kriterien k-Anonymität, ℓ-Diversität, t-Closeness und δ-Präsenz ohne Unterdrückung monoton, k-Anonymität und distinkt-ℓ-Diversität sind auch mit Unterdrückung monoton. Bei diesen Konfigurationen ist es nun möglich, vorausschauendes Markieren des Suchraumes zur Effizienzsteigerung zu nutzen. Bei allen anderen genannten Kriterien mit Unterdrückung ist dies, wenn eine optimale Lösung gefunden werden soll, nicht möglich. In [78] wird vorgeschlagen, auch in diesen Fällen vorausschauendes Markieren zu nutzen, da trotzdem sehr oft die optimale Lösung gefunden wird (praktische Monotonie). In dieser Arbeit werden beide Vorgehensweisen berücksichtigt.

Wenn mehrere Kriterien miteinander kombiniert werden, z. B. k-Anonymität um gegen Identitätsaufdeckung zu schützen, plus t-Closeness um gegen Attributsaufdeckung zu schützen, und Unterdrückung erlaubt ist, tritt der Fall ein, dass die Kombination beider Kriterien nicht monoton ist, das Teilkriterium k-Anonymität hingegen ist monoton. Auch diese Teilmonotonie kann zur Optimierung genutzt werden. Wenn beim Durchsuchen des Suchraumes also eine Transformation nicht k-anonym ist, kann keine der Spezialisierungen k-anonym sein, und somit auch nicht die Kombination erfüllen. Ist eine Transformation allerdings k-anonym, so kann keine Aussage über Generalisierungen dieser Transformation hinsichtlich der Kombination der beiden Kriterien getroffen werden.

3.3 Monotonie von Methoden zur Messung des Informationsverlustes

Wie in Abschnitt 2.6 beschrieben, gibt es sowohl monotone als auch nicht-monotone Metriken. Auch bei monotonen Metriken kann eine Effizienzsteigerung gegenüber nicht-monotonen Metriken erreicht werden [56]. Im Falle von nicht-monotonen Metriken müsste der Informationsverlust von allen anonymen Transformationen bestimmt werden um das Minimum zu finden, bei monotonen Metriken ist dies nicht notwendig. Hierbei weiß man, dass alle Generalisierungen einer anonymen Transformation einen höheren Informationsverlust aufweisen. Dies führt dazu, dass, wenn man den Suchraum in verschiedene (möglicherweise überlappende) Strategien (Pfa-

de) aufteilt, das globale Optimum gefunden werden kann, wenn man das Minimum der lokalen Optima nutzt. Außerdem kann man, bei monotonen Kriterien, auch die Transformationen ausschließen, die vorausschauend markiert wurden, da diese weder ein lokales noch ein globales Optimum sein können.

3.4 Vorausschauendes Markieren bei nicht monotonen Kriterien und Metriken

Durch die Kombination von monotonen, teilweise monotonen und nicht-monotonen Kriterien sowie monotonen und nicht-monotonen Metriken, ergeben sich jeweils unterschiedliche Möglichkeiten vorausschauend zu markieren. In Abbildung 3.5 sind die verschiedenen Möglichkeiten aufgezeigt.

Vorausschauendes markieren		Kriterium		
		voll monoton	teilweise monoton	nicht monoton
Metrik	monoton	**(1)** ↓ ↑	**(3)** ↓ ↑	**(5)** ↑
	nicht monoton	**(2)** ↓	**(4)** ↓	**(6)** **X**

Bei (3) und (5) wird nicht der ganze Suchraum gemäß Anonymitätskriterium klassifiziert (=Unsicherheit)

Abbildung 3.5: Entscheidungstabelle: Vorausschauendes markieren

Im ersten Fall (1), wenn sowohl Metrik als auch Kriterium monoton sind, können die meisten Transformationen vorausschauend markiert werden. Durch das monotone Kriterium können bei einer nicht anonymen Transformation alle Spezialisierungen von vornherein ausgeschlossen werden. Bei einer anonymen Transformation weiß man, dass alle Generalisierungen auch anonym sind. In diesem Falle kann durch die monotone Metrik zusätzlich ausgeschlossen werden, dass irgendeine Generalisierung dieser anonymen Transformation einen besseren Informationsverlust hat (also ein Kandidat für das globale Optimum wäre). Diese Möglichkeit fällt im zweiten Fall (2) weg, damit muss nun der Informationsverlust für alle anonymen Transformationen berechnet werden. Dies bedingt in vielen Fällen, dass die Transformation angewendet werden muss, um den Informationsverlust bestimmen zu können.

Bei teilweise monotonem Kriterium und monotoner Metrik (Fall 3) kann nun als Erstes, nach dem monotonen Teilkriterium, vorausschauend markiert werden. Es kann hierbei der Umstand ausgenutzt werden, dass, wenn eine Transformation das monotone Teilkriterium nicht erfüllt, alle Spezialisierungen ausgeschlossen werden können, da diese auch das vollständige Kriterium nicht erfüllen. Da eine monotone Metrik verwendet wird, können außerdem alle Generalisierungen einer anonymen Transformation ausgeschlossen werden, da diese nie ein Optimum sein können, da alle einen schlechteren Informationsverlust haben als die aktuelle anonyme Transformation. Im vierten Fall (4) muss wieder bei allen anonymen Transformationen der Informationsverlust bestimmt werden. Das Ausschließen von Teilen des Suchraumes durch das Teilkriterium ist weiterhin möglich.

Bei nicht monotonen Kriterien ist kein markieren anhand des Kriteriums möglich. Dennoch können bei Nutzung einer monotonen Metrik (Fall 5) wieder alle Generalisierungen einer anonymen Transformation ausgeschlossen werden. Da die-

se Option bei Fall (6) wegfällt, ist hier kein verkleinern des Suchraumes anhand der Metrik möglich.

Interessant ist, dass bei Fall (3) und (5) der Suchraum nicht vollständig hinsichtlich des Anonymitätskriteriums klassifiziert sein muss. In diesen beiden Fällen kann es vorkommen, dass einige, anhand der Metrik ausgeschlossene, Transformationen nicht anonym sind, diese Information aber nicht bekannt ist, da diese Transformation nie getestet wurde. Auf das Finden des globalen Optimums hat dies selbstverständlich keine Auswirkung. Bei der Familie an Algorithmen, die in Abschnitt 3.7 vorgestellt wird, werden alle Markierungsmöglichkeiten beachtet.

3.5 Existierende Algorithmen zur Anonymisierung lokaler Datenbestände

In diesem Abschnitt werden kurz die bekanntesten Algorithmen vorgestellt, die eine global optimale Lösung finden. Die meisten dieser Algorithmen wurden speziell für den Fall 1 entwickelt (siehe Abschnitt 3.4). Samarati nutzt eine Binärsuche über die Level des Verbandes, findet allerdings nur mit der „Height Metrik“ (siehe Abschnitt 2.6) eine global optimale Lösung. Incognito hingegen implementiert eine horizontale Traversierungsstrategie (breadth-first). OLA, als aktuell schnellster Algorithmus, springt zwischen den Leveln des Generalisierungsverbandes um möglichst große Teile des Suchraums, durch vorausschauendes Markieren, von der weiteren Betrachtung auszuschließen.

Zwei weitere einfache Algorithmen sind die Breitensuche (BFS) und die Tiefensuche (DFS), die Standard Graph Traversierungsstrategien nutzen. Diese beiden Algorithmen können global optimale Lösungen für alle der 6 möglichen Kombinationen von monotonen/nicht-monotonen Kriterien und Metriken finden, da diese standardmäßig den kompletten Suchraum durchsuchen.

3.5.1 Samarati

Der bei Samarati [61] vorgeschlagene Algorithmus durchläuft den Generalisierungsverband binär über die Level.

Der Pseudocode des Algorithmus ist in Algorithmus 1 dargestellt.

Ein Level des Verbandes wird sequenziell durchlaufen, wenn ein anonymer Knoten auf dem aktuellen Level gefunden wird, fährt der Algorithmus auf der unteren Hälfte des Verbandes fort. Wird ein Level hingegen komplett durchlaufen, ohne dass ein anonymer Knoten gefunden wurde, fährt der Algorithmus in der oberen Hälfte des Verbandes fort. Der Algorithmus startet auf dem mittleren Level ($\lfloor maxLevel/2 \rfloor$) des Verbandes. Der Knoten auf dem niedrigsten Level, der anonym getestet wurde, wird als optimaler Knoten zurückgeliefert. Hierbei sieht man auch den größten Nachteil des Algorithmus, er kann nur mit der Height Metrik verwendet werden. Bei anderen Metriken liefert er nicht unbedingt die global optimale Lösung.

Ein Beispiel des Algorithmus ist in Abbildung 3.6 dargestellt. Samarati startet auf Level 4 ($\lfloor \frac{8}{2} \rfloor$) und prüft als Erstes den Knoten $(2, 1, 1)$. Dieser Knoten ist nicht anonym, folglich prüft der Algorithmus nun den nächsten Knoten auf dem Level $(2, 0, 2)$. Der Level wird solange weiter bearbeitet, bis der Zustand $(1, 0, 3)$ getestet wird. Dieser ist anonym, er wird als potenzielles Ergebnis festgehalten und der

Algorithm 1: Samarati angepasst von [61]

```
Input: Verband lattice
Result: Anonymer Knoten result auf dem niedrigsten Level
1  begin
2      low ← 0
3      high ← lattice.height
4      anonymousNodeOnLevel ← false
5      while low <= high do
6          mid ← ⌊(low + high)/2⌋
7          level ← lattice.level[mid]
8          foreach node ∈ level do
9              checkNode(node)
10             if (node.anonymous) then
11                 anonymousNodeOnLevel ← true
12                 result ← node
13                 break
14         if (anonymousNodeOnLevel) then
15             high = mid − 1
16         else
17             low = mid + 1
18     return result
```

Algorithmus fährt auf Level 1 fort ($\lfloor \frac{3}{2} \rfloor$). Der Zustand $(1, 0, 0)$ ist nicht anonym, damit fährt der Algorithmus mit dem Knoten $(0, 1, 0)$ fort. Der Algorithmus stoppt, wenn es keinen niedrigeren Level gibt, welcher noch anonyme Zustände enthält. Der Knoten $(1, 0, 3)$ ist am Ende das optimale Ergebnis.

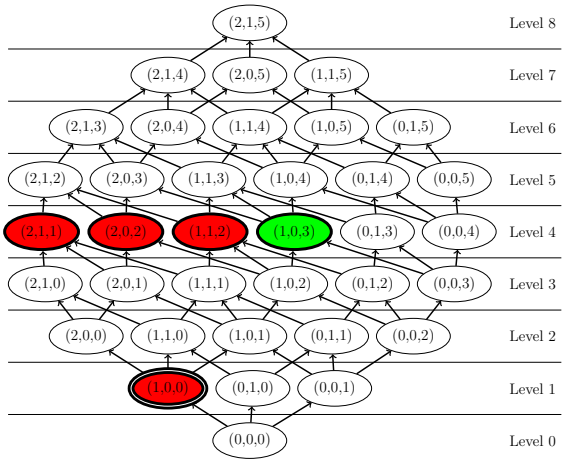


Abbildung 3.6: Beispiel für Samarati

3.5.2 Incognito

Incognito [79] ist ein Algorithmus, der sich an das Prinzip der dynamischen Programmierung anlehnt. Die wichtigste Eigenschaft hierbei ist, dass, wenn eine Teil-

menge der Quasi-Identifikatoren nicht anonym ist, dann auch keine Obermenge anonym ist. Aufbauend auf dieser Eigenschaft generiert der Algorithmus die Potenzmenge der Quasi-Identifikatoren, sortiert diese aufsteigend nach der Anzahl der Elemente und beginnt diese sortierte Menge der Reihenfolge nach zu bearbeiten. Passend zu jeder Teilmenge wird der dazugehörige Generalisierungsverband aufgebaut und mittels Breitensuche getestet. Dabei wird auch von der Möglichkeit des vorausschauenden Markierens Gebrauch gemacht, was natürlich nur bei anonymen Knoten Arbeit einspart, da hierbei alle Generalisierungen von der weiteren Suche ausgeschlossen werden können. Bei nicht-anonymen Knoten könnten die Spezialisierungen ausgeschlossen werden, diese wurden aber wegen der von unten beginnenden Breitensuche schon getestet. Nach Abschluss der Tests der Teilmengen von n Quasi-Identifikatoren kommt das Prinzip der dynamischen Programmierung ins Spiel. Alle Generalisierungen, die bei n Identifikatoren nicht anonym sind, sind bei $n+1$ Identifikatoren auch nicht anonym. Somit können vor dem Start der Breitensuche auf dem Verband der $n+1$ Identifikatoren die Knoten als nicht anonym markiert werden, die eine Teilmenge der nicht-anonymen Generalisierungen des Verbandes mit n Quasi-Identifikatoren beinhalten. Damit werden Teilergebnisse zwischen Elementen der Potenzmenge propagiert. Der Algorithmus ist beendet, wenn der Verband mit allen Quasi-Identifikatoren bearbeitet wurde. Der Pseudocode des Algorithmus ist in Algorithmus 2 dargestellt.

Algorithm 2: Incognito angepasst von [79]

```
   Input: Quasi Identifikator qis
   Result: Menge aller anonymen Knoten result
1  begin
2      powerSet ← orderedPowerSet(qis)
3      falseNodes ← emptyList
4      foreach qiSet ∈ powerSet do
5          lattice ← buildLattice(qiSet)
           /* Tag lattice using previous iterations */
6          foreach falseNode ∈ falseNodes do
7              if falseNode.qis ∈ qiSet then
8                  lattice.tag(falseNode, false)
           /* Traverse lattice in BFS */
9          foreach level ∈ lattice do
10             foreach node ∈ level do
11                 if !node.tagged then
12                     checkNode(node)
13                     if (node.anonymous) then
14                         lattice.tag(node, true)
15                     else
16                         lattice.tag(node, false)
17                         falseNodes ← node
```

Ein Beispiel ist in Abbildung 3.7 dargestellt. Es bezieht sich auf den Beispiel-

datensatz Abbildung 2.4, die dazugehörigen Generalisierungshierarchien und zeigt die Verbände, welche in den verschiedenen Iterationen erzeugt und getestet werden. Incognito beginnt mit dem testen von jeweils drei Quasi-Identifikatoren, „Alter“, „Geschlecht“ und „PLZ“. Beginnend mit Alter testet der Algorithmus zuerst den Zustand (0), befindet diesen für das QI-Attribut Alter alleine schon als nicht anonym, und fährt dann mit Zustand (1) fort, welcher anonym ist, was erlaubt Zustand (2) vorausschauend, ohne zu testen, zu markieren. Analog bearbeitet Incognito die Verbände für Geschlecht und PLZ.

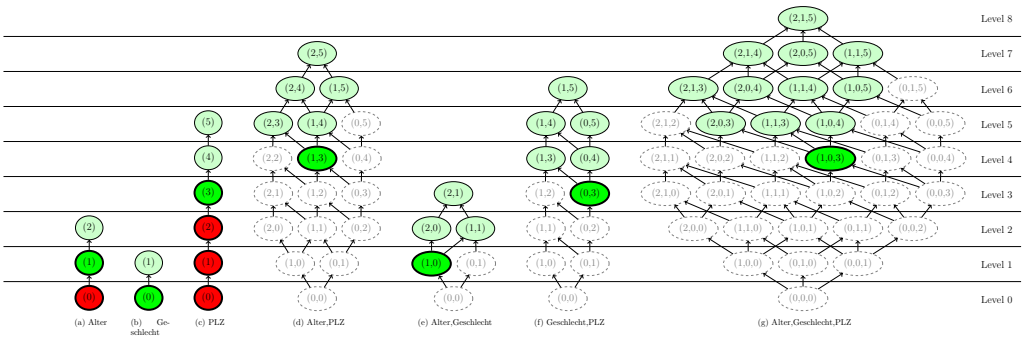


Abbildung 3.7: Beispiel für Incognito

Nachdem alle einelementigen Verbände bearbeitet sind, werden nun alle Kombinationen von zweielementigen Verbänden bearbeitet. Im Beispiel wäre das die Kombination von „Alter“ und „PLZ“. Hier können nun schon Teile des Suchraumes ausgeschlossen werden, da „Alter“ erst ab dem Zustand (1) und „PLZ“ erst ab dem Zustand (3) anonym sind, somit können alle Zustände der kombinierten Verbände als nicht-anonym gekennzeichnet werden, welche Zustände enthalten, die kleiner sind als (1) für „Alter“ und kleiner sind als (3) für „PLZ“. Hier startet der Algorithmus erst bei Zustand $(1, 3)$. Dieser ist anonym und somit können alle anderen, noch nicht markierten Zustände, in diesem Verband als anonym gekennzeichnet werden. Auch hier wird analog für die beiden noch verbleibenden Verbände [Alter, Geschlecht] und [Geschlecht, PLZ] vorgegangen. Nachdem nun alle zweielementigen Verbände bearbeitet sind, kommt der gesamte Verband an die Reihe, welche aus allen drei Quasi-Identifikatoren [Alter, Geschlecht, PLZ] besteht. Da Incognito auch hier wieder die Information aus den vorhergehenden Teilschritten verwenden kann, ist der erste zu testende Knoten $(1, 0, 3)$. Dieser ist anonym und es werden alle Nachfolger als anonym markiert. Da nun der Zustand aller Knoten in dem Gesamtverband determiniert ist, stoppt der Algorithmus und der anonyme Knoten mit der kleinsten Metrik $(1, 0, 3)$ ist die optimale Lösung.

3.5.3 Optimal Lattice Anonymization

Ein weiterer Algorithmus, Optimal Lattice Anonymization (OLA), welcher die optimale Lösung bei globalem Codieren, monotonen Anonymisierungskriterien und monotonen Metriken findet, wurde in [56] vorgeschlagen. Die Autoren zeigten, dass dieser Algorithmus sowohl den Algorithmus von Samarati (siehe Abschnitt 1) als auch den von Incognito (siehe Abschnitt 2) hinsichtlich Anzahl der zu testenden Zustände und der Laufzeit schlägt. Der Algorithmus baut auf dem Prinzip des Teile-und-Herrsche (divide-and-conquer) auf und teilt deshalb den großen Verband

in Unterverbände auf. Er nutzt, wie Incognito, das vorausschauende Markieren um Teile des Suchraumes, ohne zu testen, auszuschließen. Ein Unterverband ist durch einen Top-Zustand t, einem Grund-Zustand b und allen Zuständen, die sich von t und b aus, also allen Generalisierungen von b und allen Spezialisierungen von t, erreichen lassen. OLA startet mit dem Gesamtverband. Als nächsten Schritt werden alle Zustände M in der Mitte des Verbandes ($level = \frac{1}{2}\lfloor b.level + t.level \rfloor$) der Reihe nach bearbeitet. Wenn ein Knoten $m \in M$ noch nicht bearbeitet wurde, wird dieser Zustand auf Anonymität geprüft. Darauffolgend wird je nach Anonymität vorausschauend markiert. Ist m nun als anonym gekennzeichnet worden, fährt der Algorithmus mit der unteren Hälfte des Verbandes fort; er konstruiert also als Nächstes den Unterverband mit m als Top-Zustand und b als Grund-Zustand. Sollte der Knoten als nicht-anonym gekennzeichnet worden sein, würde der Algorithmus mit dem oberen Unterverband (t, m) fortfahren. Dieser rekursive Prozess stoppt, wenn alle Unterverbände enumeriert wurden. Der Pseudocode des Algorithmus ist in Algorithmus 3 zu sehen.

Algorithm 3: KMIN($bottom, top$) angepasst von [56]

```
Input: Grund-Knoten bottom und Top-Knoten top
Result: Menge aller anonymen Knoten result des Verbandes
1  begin
2      if top.level − bottom.level > 1 then
3          foreach n ∈ getMidNodes(bottom, top) do
4              if n.tagged ∧ n.anonymous then
5                  kmin(bottom, n)
6              else if n.tagged ∧ ¬n.anonymous then
7                  kmin(n, top)
8              else if checkNode(n) then
9                  tag(n, true), kmin(bottom, n)
10             else
11                 tag(n, false), kmin(n, top)
12     else
13         if ¬bottom.tagged then
14             if checkNode(bottom) then
15                 tag(bottom, true)
16             else
17                 tag(bottom, false)
```

Die ersten Iterationen des Algorithmus für den Beispieldatensatz sind in Abbildung 3.8 zu sehen. OLA beginnt hierbei alle Knoten auf Level 4 durchzugehen. Im Beispiel wird angenommen, dass der erste Knoten auf Level 4 $(2, 1, 1)$ ist. Dieser Knoten wurde noch nicht bearbeitet, also wird dieser auf Anonymität getestet. Dieser Knoten stellt eine nicht anonyme Transformation dar, deshalb wird er und all seine Vorgänger als nicht-anonym gekennzeichnet (hellrot). Daraufhin wird der Algorithmus mit dem Unterverband $((2, 1, 1), (2, 1, 5))$ fortfahren. Als

nächsten Schritt werden wieder die Knoten auf dem mittleren Level dieses Unterverbandes bearbeitet, im Beispiel entspricht dies dem Knoten $(2, 1, 3)$, da dieser Knoten nun einen anonymen Zustand widerspiegelt, wird als Nächstes der Unterverband $((2, 1, 1), (2, 1, 3))$ generiert. Im nächsten Schritt folgt dann der Unterverband $((2, 1, 1), (2, 1, 2))$. Nun hat der Algorithmus also die drei Knoten $(2, 1, 1)$, $(2, 1, 2)$ und $(2, 1, 3)$ getestet, um das lokale 2-anonyme Optimum $(2, 1, 3)$ zu finden. Nun würde der Algorithmus mit dem nächsten Knoten auf Level 4 $(2, 0, 2)$ weitermachen. Dieser Knoten ist nicht-anonym, weshalb der nächste Unterverband $((2, 0, 2), (2, 1, 5))$ generiert wird. Wenn alle Unterverbände enumeriert sind, stoppt der Algorithmus.

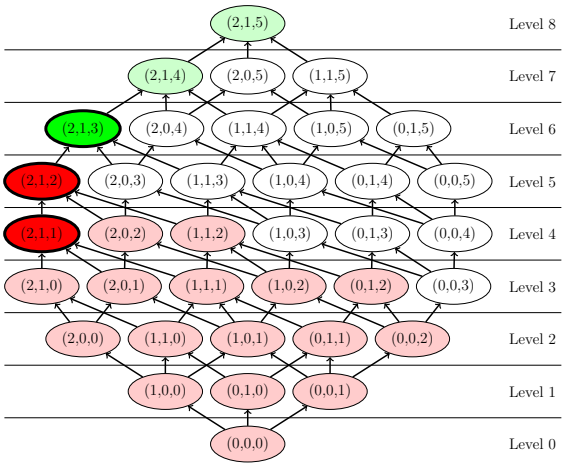


Abbildung 3.8: Beispiel für OLA

3.5.4 Breitensuche

Neben den speziell für das Problem konstruierten Algorithmen kann man auch klassische Graph-Traversierungsstrategien verwenden. Im einfachsten Fall kann eine Breitensuche (breadth-first) Strategie verwendet werden. Start hierbei ist der unterste Knoten des Verbandes, danach wird, Level für Level, der Verband vollständig bearbeitet. Hierbei kann auch, wie bei den anderen Algorithmen, im monotonen Fall vorausschauend markiert werden. Da aber von unten gestartet wird, können, wie bei Incognito, nur anonyme Transformationen ausgeschlossen werden.

Algorithm 4: Breadth-First Strategie

```
   Input: Verband lattice
1  begin
2      foreach level ∈ lattice do
3          foreach node ∈ level do
4              if !node.tagged then
5                  CheckAndTag(node)
6              Store(node)
```

Der Pseudocode in Algorithmus 4 zeigt BFS. Hierbei ist anzumerken, dass CheckAndTag, je nach der Monotonie des Kriteriums, vorausschauend markiert.

3.5.5 Tiefensuche

Eine weitere allgemeine Traversierungsstrategie für Graphen ist Tiefensuche (depth-first). Hierbei wird, anstatt über ein Level nach dem anderen, direkt versucht von dem untersten Knoten zu dem obersten Knoten zu gelangen, nachdem man am höchsten Knoten angelangt ist, wird dieser getestet, danach der nächste usw. Dabei werden im Standardfall alle möglichen Generalisierungsstrategien enumeriert. Da die Anzahl der Strategien aber sehr schnell sehr groß wird (siehe Abschnitt 2.5.1.6.1) ist dieser Grundalgorithmus nicht praktisch einsetzbar. Mit zwei kleinen Modifikationen kann er allerdings als Vergleich herangezogen werden. Zum einen wird der Test vorgezogen, d. h., dass zuerst der Knoten getestet und danach erst in die Tiefe gegangen wird. Dieses Vorgehen ist effizienter, da hierbei Optimierungen angewendet werden können (siehe Abschnitt 3.6). Außerdem wird bei jedem Test, wenn möglich, vorausschauendes Markieren verwendet. Der Pseudocode der äußeren Schleife ist in Algorithmus 5 dargestellt. Die rekursive Methode, welche den Pfad sucht und testet, ist in Pseudocode 6 zu sehen.

Algorithm 5: Depth-First Strategie

```
   Input: Verband lattice
1  begin
2      foreach level ∈ lattice do
3          foreach node ∈ level do
4              if !node.tagged then
5                  DFS(node)
```

Algorithm 6: DFS

```
   Input: Knoten node
1  begin
2      CheckAndTag(node)
3      Store(node)
4      foreach up ∈ node.successors do
5          if !up.tagged then
6              DFS(up)
```

3.6 Implementierungsdetails

In Abschnitt 2 wurde dargelegt, dass die beste Art der Anonymisierung, in der medizinischen Domäne, auf globalem recoding mit definierten Generalisierungshierarchien aufbaut. Um diese Art der Anonymisierung effizient anwenden zu können, wird hier das in [72] und [80] beschriebene Framework verwendet. Dieses Framework

erlaubt effizient Transformationen anzuwenden und verschiedene Anonymitätskriterien zu testen. Viele Algorithmen, die eine anonyme Lösung suchen, nutzen das gleiche Vorgehen. Zuerst wird eine (von vielen möglichen) Generalisierungsvorschrift ausgewählt, daraufhin werden die Daten passend zu dieser Vorschrift transformiert. Die so generalisierten Daten werden nun auf die Einhaltung des jeweiligen Anonymitätskriteriums geprüft. Ist die passende Lösung gefunden (z. B. die optimale Lösung mit dem geringsten Informationsverlust), wird die Schleife unterbrochen. Andernfalls wird eine neue Generalisierungsvorschrift ausgesucht und der Zyklus beginnt von Neuem. Bei Algorithmen die eine, hinsichtlich des Informationsverlustes, optimale Lösung suchen, müssen die Daten oftmals viele Male transformiert und auf Anonymität geprüft werden.

Das Framework nutzt den Umstand aus, dass durch Wörterbuchkomprimierung die zu anonymisierenden Daten in den Hauptspeicher passen. Da die Daten in Tabellenform vorliegen, ist ein zwei-dimensionales Integer Array die passendste Datenstruktur, um die codierten Daten effizient zu speichern. Auch die Generalisierungshierarchien werden in Tabellenform gespeichert und mithilfe der Wörterbücher komprimiert. Durch dieses Datenlayout und die Wörterbuchkomprimierung kann die Transformation als einfache Zuweisung ausgeführt werden [72]. Nachdem die Daten transformiert wurden, können mittels einer Hashtabelle die Äquivalenzklassen berechnet werden. Die Äquivalenzklassen können dann durchlaufen und auf Einhaltung des geforderten Anonymitätskriterium getestet werden. Die Laufzeit für die Transformation, das Formen der Äquivalenzklassen und testen auf Anonymität kann damit in amortisiert linearer Laufzeit bezüglich der Anzahl der Zeilen des Datensatzes erfolgen [72].

Das wiederholte Anwenden von Transformationen, die sich teilweise nur sehr wenig unterscheiden, erlaubt weitere Optimierungstechniken. Zum einen müssen nur die Spalten transformiert werden, die sich gegenüber der letzten Transformation geändert haben (Projektion). Zum anderen kann man, da die Generalisierungshierarchien monoton sind, Zwischenergebnisse wiederverwenden. Durch die Monotonie der Hierarchien ist sichergestellt, dass Äquivalenzklassen einer stärker generalisierten Transformation nur aus den Klassen der schwächer generalisierten Transformation entstehen können (Roll-up). Diese Zwischenergebnisse können auch gespeichert werden um sie später wiederzuverwenden (Historie). Durch geschickte Speicherung ist der benötigte Speicheroverhead gering [72].

3.6.1 Parallelisierung

Um die Möglichkeiten moderner Multi-Core Rechner auszunutzen, wurde die Implementierung des Frameworks parallelisiert (intra-operator Parallelisierung). Um einen möglichst großen Speedup zu erreichen, wird sowohl die Transformation als auch das Gruppieren parallelisiert. Hierbei werden die zu prüfenden Zeilen gleichmäßig auf alle verfügbaren Threads verteilt. Diese transformieren parallel die Daten und gruppieren diese in einem thread lokalen Gruppierer. Sobald ein Thread mit seiner Arbeit fertig ist, wird dessen Gruppierer der globale Gruppierer, und die anderen Gruppierer werden in diesen integriert. Dieser Schritt verursacht, im Gegensatz zur seriellen Version, zusätzliche Arbeit; allerdings können für dieses Gruppieren der Gruppierer die bereits berechneten Hashwerte wieder verwendet werden. Es wurden auch Versuche mit einer nicht-blockierenden Hashtabelle durchgeführt, welche

aber eine schlechtere Performanz zeigte. Zusätzlich wurde ein Schwellwert definiert, welcher die minimale Anzahl an Zeilen pro Thread festlegt.

Wie auch im Abschnitt 3.9 zu sehen, steigt der Gewinn dieser Optimierung im Verhältnis zur Größe der Eingabedaten. Ein Problem dieser Optimierung ist, dass die Datenstrukturen (Datenarray und Pufferarray) für alle Threads gleich sind. Dies ist besonders bei NUMA (Non-uniform memory access) Architekturen von Relevanz, da hier, je nachdem auf welchem Knoten der Thread läuft, die Zugriffszeiten auf die Daten und den Puffer länger sein können. Hierzu wurden auch Experimente durchgeführt. Es wurden die Daten und der Puffer durch die Anzahl der Threads geteilt und jeder Thread erhielt somit seinen eigenen Daten- und Pufferteilbereich. Allerdings wird bei dieser Methode die Last nicht gleichmäßig auf alle Threads verteilt, da die Gruppen in den Daten nicht gleichmäßig verteilt sind. Wie in den Ergebnissen im Abschnitt zu sehen, ist hier der Speedup Faktor geringer, obwohl diese Implementierung die besser parallelisierbare ist, da auch z. B. die Historie für jeden Thread unabhängig ist.

3.6.2 Effiziente Berechnung der Entropie Metrik

Das im vorhergehenden Abschnitt beschriebene Framework und die Optimierungstechniken erlauben das effiziente Transformieren und Testen eines Zustandes auf Anonymität. Sollte dabei ein anonymer Zustand gefunden werden, muss, um die optimale Lösung zu finden, der Informationsverlust berechnet werden. Das Framework implementiert eine Menge an monotonen und nicht monotonen Metriken. Die meisten, im Abschnitt 2.6 beschriebenen, Metriken lassen sich einfach effizient implementieren. Ausnahmen sind die Mitglieder der Entropie Metrik Familie, welche allerdings eine informationstheoretische Grundlage haben und damit für viele Anwendungsszenarien relevant sind. Deshalb wird hier eine sehr effiziente Implementierung der Entropie Metrik vorgeschlagen. Sie basiert auf der Beobachtung, dass, bei globalem recodieren, nur die Kardinalitäten der distinkten Werte (vor der Generalisierung und nach der Generalisierung) miteinander in Beziehung gesetzt werden müssen.

Wenn der zu anonymisierende Datensatz eingelesen ist, wird ein dreidimensionales Array $card[i, v, k]$ aufgebaut, welches die Anzahl der Werte mit der ID v auf dem Level k des i-ten Quasi-Identifikators speichert. Dieses Array kann effizient beim Einlesen der Daten generiert werden. Mithilfe der Generalisierungshierarchien in Tabellenform kann man nun die ID eines beliebigen Wertes v, des i-ten Quasi-Identifikators auf Level k bestimmen ($generalised[i, v, k]$). Die IDs der distinkten Werte des i-ten Quasi-Identifikators sind in einem zusätzlichen Array gespeichert ($distinct[i]$).

Damit kann nun die non-uniform Entropie Metrik aus Abschnitt 2.6 für die Transformation $(k_0, ..., k_{n-1})$ geschrieben werden als:

$$M_{\text{NUEM}}(k_0, ..., k_{n-1}) = -\sum_{i=0}^{m-1}\sum_{j=0}^{n-1} log_2 X(i, j, k_j) = \tag{3.1}$$

$$= -\sum_{i=0}^{n-1}\sum_{v \in distinct[i]} card[i, v, 0] \cdot log_2 \frac{card[i, v, 0]}{card[i, generalised[i, v, k_i], k_i]} \tag{3.2}$$

Dabei ist die Komplexität der Berechnung nur noch abhängig von der Summe der distinkten Werte der Quasi-Identifikatoren. Bei der Original Definition ist die Komplexität hingegen abhängig von der Anzahl der Zeilen des Datensatzes. Bei realweltlichen Daten kann davon ausgegangen werden, dass die Anzahl der distinkten Werte deutlich kleiner ist als die Anzahl der Zeilen, somit kann die Metrik effizient berechnet werden. Zusätzlich wurde bei der Implementierung ein Cache eingeführt, der pro Spalte und Generalisierungslevel den Wert der Metrik zwischenspeichert und somit die Berechnungen weiter vereinfacht. Damit ergibt sich ein, trotz der komplexen Metrik, nur sehr geringer Overhead für die Berechnung der Entropie Metrik.

Diese Implementierung hat allerdings den Nachteil, dass der Informationsverlust durch Unterdrückung nicht berechnet werden kann. Grund hierfür ist, dass durch die Unterdrückung eine Art lokales Recodieren vorgenommen wird, und damit die benötigten Kardinalitäten nicht mehr beim Einlesen des Datensatzes statisch berechnet werden können. Um die non-uniform Entropie Metrik zu implementieren, müsste das Framework wieder auf zellbasierte Vergleiche zurückgreifen, und es müsste die Beziehung zwischen allen transformierten Zellen/Zeilen kennen. Dies ist nicht effizient umzusetzen, sobald Optimierungen, die auf dem Roll-up Prinzip beruhen, umgesetzt werden (neben dem hier verwendeten Framework setzen auch andere Arbeiten, z. B. [79] und [56], diese Optimierungen ein).

Aufgrund dessen wird eine abgeänderte Version der non-uniform Entropie Metrik vorgeschlagen, die sich effizient in dem Framework implementieren lässt, und dennoch den Informationsverlust bei Unterdrückung berücksichtigt. Die Idee hierbei ist, dass die non-uniform Entropie Metrik für alle Tupel berechnet und dazu der Informationsverlust gezählt wird, der entsteht, wenn einzelne Tupel unterdrückt werden. Dazu wird der aktuelle Zustand des Teildatensatzes, bestehend aus den unterdrückten Tupeln, als Basis angenommen und der total generalisierte Zustand als die erfolgte Unterdrückung.

Analog zu der Funktion $X(i, j, k)$, die den original Datensatz mit dem transformierten Datensatz vergleicht, wird die Funktion $Y(i, j, k, S(k_0, ..., k_{n-1}))$ definiert, die den transformierten Teildatensatz mit dem total generalisierten Teildatensatz vergleicht. Die Funktion $S(k_0, ..., k_{n-1})$ definiert alle Indizes der Tupel, welche unterdrückt werden müssen für die Transformation gegeben als $(k_0, ..., k_{n-1})$. Angenommen der i-te Quasi-Identifikator wird komplett unterdrückt, dann gibt Y die bedingte Wahrscheinlichkeit zurück, dass ein zufällig ausgewählter Wert in Spalte j des i-ten Quasi-Identifikators in dem Teildatensatz generalisiert auf Level k $R(i, j, k)$ ist.

$$Y(i, j, k, S) = \frac{\sum_{l \in S} I(R(i, l, k) = R(i, j, k))}{|S|} \tag{3.3}$$

Somit ist die nicht-monotone nicht-uniforme Entropie Metrik definiert als:

$$M_{\text{NMNUEM}}(k_0, ..., k_{n-1}) = M_{\text{NUEM}}(k_0, ..., k_{n-1}) \tag{3.4}$$

$$-\sum_{i=0}^{n-1} \sum_{j \in S(k_0,...,k_{n-1})} log_2 \cdot Y(i, j, k, S(k_0, ..., k_{n-1})) \tag{3.5}$$

Diese Metrik kann nun, analog zu der nicht-uniformen Entropie Metrik, sehr effizient implementiert werden. Dazu müssen die beiden Arrays *distinct* und *card* für alle unterdrückten Äquivalenzklassen berechnet werden, bevor die Metrik berechnet werden kann. Die Komplexität des zweiten Summanden ist damit beschränkt durch die Anzahl der unterdrückten Klassen, welche generell kleiner sind als die Anzahl aller Tupel im Datensatz. Außerdem können die distinkten Werte und deren Kardinalitäten effizient, aus den repräsentativen Tupeln und der Größe der Äquivalenzklassen, berechnet werden. Damit können nun alle Metriken aus dem Abschnitt 2.6 effizient implementiert werden.

3.7 Die Familie der Flash Algorithmen

Durch das effiziente Framework erreicht OLA [56], der bisher schnellste Algorithmus, schon eine respektable Performanz (siehe Abschnitt 3.9). Dennoch gibt es Verbesserungspotenzial:

- Durch den großen Overhead den OLA beim durchlaufen des Suchraumes verursacht ist eine effiziente Implementierung schwierig.
- OLA nutzt nicht das gesamte Potenzial der Optimierungstechniken des Frameworks, da er beim durchlaufen des Suchraumes über verschiedene Level des Verbandes springt.
- OLA liefert nur mit monotonen Kriterien und Metriken ein optimales Ergebnis.

Aus diesen Gründen wurde eine Familie neuer Algorithmen entwickelt, welche die oben genannten Punkte adressieren. Die Grundideen aller neuen Algorithmen sind gleich. Genau wie OLA traversieren die Algorithmen den Verband vertikal, damit wird die Möglichkeit des vorausschauenden Markierens besser genutzt als beim horizontalen (Level für Level) traversieren. Hierbei werden möglichst lange Pfade von Knoten in dem Verband gesucht und diese dann mittels Binärsuche, wenn möglich, d. h. im monotonen Fall, geprüft. Dabei wird vorausschauend markiert. Die Pfadsuche ist eine gierige Tiefensuche (greedy depth-first).

3.7.1 Basic-Flash ($Flash_B$)

Der Basic-Flash Algorithmus wurde für den Fall 1, d. h. monotones Kriterium und monotone Metrik (siehe Abschnitt 3.4) entwickelt. Diese Basis Variante wurde auch in [73] und [80] beschrieben. Hierbei werden beim Testen des Pfades die getesteten und nicht-anonymen Knoten in eine Vorrangwarteschlange gelegt. Die Vorrangwarteschlange ist nach der im Abschnitt 3.8 beschriebenen Strategie sortiert. Vereinfacht kann man sagen, dass diese Strategie die am geringsten generalisierten Knoten bevorzugt. Solange die Vorrangwarteschlange nicht leer ist, wird der erste Knoten aus der Vorrangwarteschlange als Startpunkt für den nächsten Pfad verwendet. Damit generiert der Algorithmus, ausgehend von einem Hauptpfad, mehrere Nebenpfade, die sich wie Blitze vom Hauptpfad aus verästeln. Dieser Algorithmus nutzt die möglichen Optimierungstechniken des Frameworks sehr gut aus und liefert somit sehr gute Performanz hinsichtlich Laufzeit und Speicherverbrauch.

Algorithm 7: Hauptschleife des $Flash_B$ Algorithmus

```
   Input: Verband lattice
1  begin
2      pqueue ← empty min-pqueue
3      foreach level ∈ lattice do
4          foreach node ∈ level do
5              if !node.tagged then
6                  path ← FindPath(node)
7                  CheckPath(path, pqueue)
8                  while !pqueue.isEmpty do
9                      node ← pqueue.extractMin
10                     foreach up ∈ node.successors do
11                         if !up.tagged then
12                             path ← FindPath(up)
13                             CheckPath(path, pqueue)
```

Algorithmus 7 beschreibt die äußere Schleife des Algorithmus. Die äußere Schleife (Zeile 3) iteriert über alle Level des Verbandes, von unten nach oben. Die zweite Schleife (Zeile 4) iteriert über alle Knoten auf dem Level. Beim jeweils aktuellen Knoten wird überprüft, ob dieser bereits markiert ist. Im negativen Fall wird dieser Knoten als Ausgangspunkt für einen neuen Pfad gewählt (Zeile 7). Algorithmus 8 beschreibt den Prozess der Pfadsuche. Diese Pfadsuche ist gierig und sucht sich solange einen Pfad, bis sie entweder beim obersten Knoten in dem Verband angelangt ist, oder bis sie auf dem Weg dort hin einen Knoten findet, der keine nicht-markierten Nachfolger als Knoten mehr hat. Dieser Pfad wird dann, wie in Algorithmus 9 beschrieben, in binärer Weise geprüft. CheckPath beginnt beim mittleren Knoten, wenn dieser anonym ist, fährt der Algorithmus mit dem unteren Pfadteil fort und der anonyme Knoten wird als möglicher, optimaler Kandidat gespeichert. Im Falle eines nicht-anonymen Knotens wird der obere Teil des Pfades in binärer Weise getestet, und der Knoten wird in der Vorrangwarteschlange gespeichert, und dient im weiteren Verlauf als möglicher Startkandidat für einen neuen Pfad. Nachdem der Pfad vollständig getestet wurde, wird das erste Element der Vorrangwarteschlange als Startknoten für den nächsten Pfad verwendet. Die Vorrangwarteschlange ist sortiert, wie in Abschnitt 3.8 beschrieben. Die Idee hierbei ist, dass der Knoten, welcher den niedrigsten Level besitzt, den längsten Pfad bilden kann; was wiederum die Wahrscheinlichkeit erhöht, dass mehr Knoten vorausschauend markiert werden können. Da dieser Knoten außerdem schon getestet wurde, ist die Wahrscheinlichkeit, dass für diesen ein Schnappschuss (siehe 3.6) existiert und dadurch die weiteren Tests der Knoten auf diesem Pfad billiger werden, sehr hoch. Im Falle einer leeren Vorrangwarteschlange fährt der Algorithmus mit der äußeren Schleife fort bis entweder ein Knoten iteriert wird, der nicht markiert ist, oder die äußere Schleife terminiert. Durch die beiden Schleifen (Zeilen 3 und 4) ist sichergestellt, dass jeder Knoten des Verbandes einmal bearbeitet wird. Der Al-

gorithmus terminiert, wenn alle Knoten des Verbandes einmal durchlaufen wurden. Damit ist die Laufzeit des Algorithmus determiniert bei der Anzahl der Knoten in dem Verband.

Algorithm 8: FindPath(node)

```
   Input: Start node node
   Result: Path of untagged nodes path
1  begin
2      path ← new list
3      while path.head() ≠ node do
4          path.add(node)
5          foreach up ∈ node.successors do
6              if !up.tagged then
7                  node ← up
8                  break
9      return path
```

Algorithmus 8 beschreibt die Pfadsuche. Diese Methode startet beim Knoten *node* und sucht einen Pfad von nicht bearbeiteten Knoten bis zum letzten Knoten des Verbandes. Die Suche endet, wenn entweder die Spitze des Verbandes erreicht ist, oder der Algorithmus an einem Punkt angekommen ist, wo kein unbearbeiteter Knoten mehr erreichbar ist. Die Knoten, die den Pfad beschreiben, werden daraufhin als geordnete Liste an den Aufrufer zurückgegeben.

Algorithm 9: CheckPath(path, pqueue)

```
   Input: Path path, priority queue pqueue
1  begin
2      low ← 0
3      high ← path.size − 1
4      pathOptimum ← null
5      while low ≤ high do
6          mid ← ⌊½(low + high)⌋
7          node ← path.get(mid)
8          if CheckAndTag(node) then
9              pathOptimum ← node
10             high = mid − 1
11         else
12             pqueue.add(node)
13             low = mid + 1
14     Store(pathOptimum)
```

Algorithmus 9 beschreibt die Pfadbearbeitung. Diese Methode beschreibt eine klassische Binärsuche. Sie startet bei Knoten $node = path.get(\lfloor \frac{1}{2}(path.size - 1) \rfloor)$

und fährt in der oberen Hälfte fort, wenn der Knoten nicht-anonym ist. Zugleich wird dieser Knoten in die Vorrangwarteschlange gelegt als möglicher, neuer Startknoten für folgende Pfade. Ist *node* hingegen anonym, wird eine Referenz auf diesen Knoten als potenzielle pfad-optimale Lösung gespeichert, und die Suche geht in der unteren Hälfte des Pfades weiter. Da durch die Binärsuche sichergestellt ist, dass bei jedem Pfad genau die Grenze zwischen anonymen und nicht anonymen Knoten gefunden wird, und auch dass diese beiden Knoten getestet werden, ist sichergestellt, dass in diesem Fall *localOptimum* den am wenigsten generalisierten Knoten dieses Pfades enthält, der anonym ist. Da für den Grundalgorithmus gefordert wird, dass die Metriken monoton sind, also der Informationsverlust immer größer wird je stärker die Generalisierung ist (siehe 2.6), ist somit sichergestellt, dass dies der Knoten mit der besten Metrik auf dem Pfad ist. In Zeile 14 schließlich wird der lokal optimale Knoten an die Methode STORE übergeben, die den Knoten mit dem aktuellen globalen Optimum vergleicht und bei einem geringeren Informationsverlust diesen als neues globales Optimum speichert. Bei einem größeren Informationsverlust bleibt die Referenz auf das aktuelle, globale Optimum erhalten.

Wie im nächsten Abschnitt 3.8 beschrieben, wird auf alle Knoten des Verbandes eine totale Ordnung definiert. Dieser Ordnung wird immer dann gefolgt, wenn über mehrere Knoten iteriert wird. Dies trifft besonders bei den grau hinterlegten Zeilen in den Algorithmen 7 und 8 zu.

3.7.1.1 Beispiel für den Basic-Flash Algorithmus

Das Verhalten des Algorithmus bzgl. des Beispieldatensatzes mit 2-Anonymität als Kriterium ist in Abbildung 3.9 dargestellt. Die Bilder zeigen jeweils den Zustand des Verbandes nach dem Testen eines Knotens auf Anonymität. Der Knoten der getestet wurde, ist mittels einer doppelten Umrandung gekennzeichnet. Ein rot ausgefüllter Knoten bedeutet, dass dieser eine nicht anonyme Transformation der Daten repräsentiert, ein grüner Knoten hingegen repräsentiert eine anonyme Transformation. Dunkelrot und dunkelgrün bedeuten, dass der Knoten auf Anonymität getestet wurde, hellrot und hellgrün sind Resultate des vorausschauenden Markierens. Die roten Linien repräsentieren den jeweils aktuellen Pfad, der mittels Binärsuche getestet wurde.

Der Algorithmus startet, indem er einen Pfad vom Knoten $(0, 0, 0)$ bis zur Spitze des Verbandes $(2, 1, 5)$ aufbaut (rote Linie in 3.9a). Dieser Pfad wird nun binär bearbeitet und damit wird Knoten $(0, 0, 4)$ als Erstes auf Anonymität getestet. Da dieser Knoten nicht-anonym ist, wird dieser und alle seine Vorgänger als nicht-anonym markiert (rot). Danach folgen die Knoten $(1, 0, 5)$ (anonym) und $(2, 0, 5)$ (nicht-anonym), dargestellt in 3.9b und 3.9c. Damit ist der erste Pfad komplett bearbeitet und die Vorrangwarteschlange beinhaltet die Knoten $(0, 0, 4)$ und $(0, 0, 5)$. Da der Knoten $(0, 0, 4)$ bzgl. der hier vorgeschlagenen Strategie (siehe Abschnitt 3.8) bevorzugt wird, beginnt der Algorithmus einen neuen Pfad zu bauen, ausgehend von diesem Knoten. Dieser Pfad startet bei Knoten $(0, 0, 4)$ und endet bei Knoten $(2, 1, 4)$, siehe rote Linie in 3.9d. Nachdem die Vorrangwarteschlange leer ist, fährt der Algorithmus mit Knoten $(1, 0, 0)$ fort, siehe 3.9h. Der Algorithmus terminiert, nachdem er den Knoten $(2, 1, 2)$ getestet hat, und in der äußeren Schleife die letzten drei Level traversiert hat. Die gefundene optimale Lösung, bei Nutzung der non-uniform Entropie Metrik (siehe Abschnitt 2.6.6), ist der Knoten $(1, 0, 3)$.

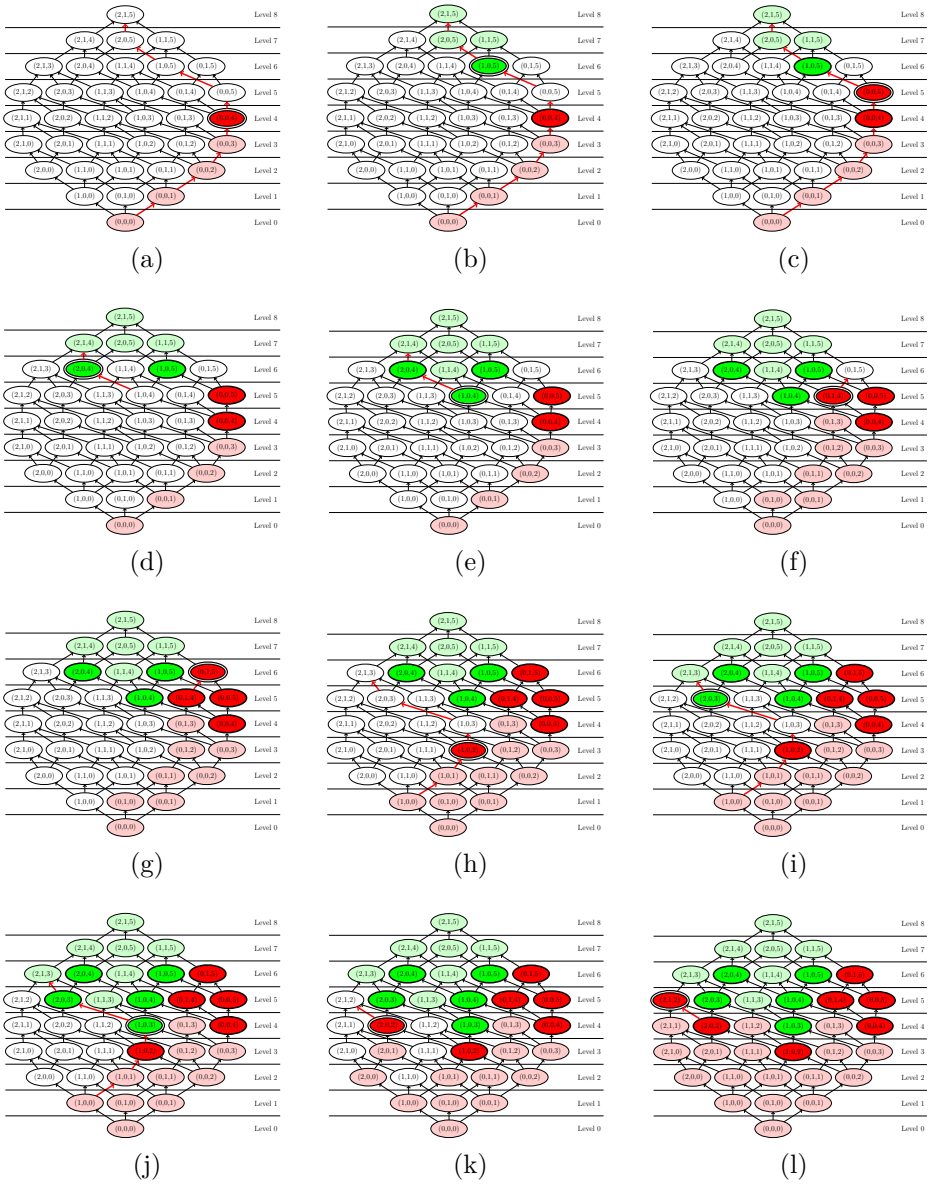


Abbildung 3.9: $Flash_B$: 2-Anonymität, 0% Unterdrückung und NUE-Metrik für den Beispieldatensatz

3.7.2 Parallel-Flash ($Flash_{Par}$)

Neben der inter-operator Parallelisierung (siehe Abschnitt 3.6.1) besteht die Möglichkeit, mehrere Knoten des Verbandes parallel zu testen. Den größten Laufzeitgewinn kann man erreichen, wenn jeder Thread möglichst den Knoten testet, der am meisten Knoten vorausschauend markieren kann, und nicht von anderen Threads markiert oder bearbeitet wird. Diesen optimalen Zustand kann man leider nicht erreichen, da man a priori wissen müsste, welche Knoten anonym oder nicht anonym sind, um die optimale Verteilung auf Threads zu erreichen. In der implementierten Parallelisierung wurde versucht, mit möglichst wenigen Synchronisierungen zwischen den Threads auszukommen. Als Proof-of-Concept wurde $Flash_B$ parallelisiert, die anderen Flash Versionen können nach demselben Prinzip parallelisiert

werden. Insbesondere die Teile der anderen Algorithmen, welche den Suchraum vollständig klassifizieren sollten, von der inter-operator Parallelisierung profitieren.

Algorithm 10: ParallelFlash(Verband)

```
   Input: Verband lattice
1  begin
2      foreach level ∈ lattice do
3          foreach node ∈ level do
4              if !node.tagged and !node.isInPath then
5                  new Thread(pathChecker(node))
```

Algorithm 11: PathChecker(node)

```
   Input: Start node node
1  begin
2      pqueue ← empty min-pqueue
3      path ← FindPathParallel(node)
4      CheckPath(path, pqueue)
5      while !pqueue.isEmpty do
6          node ← pqueue.extractMin
7          foreach up ∈ node.successors do
8              if !up.tagged then
9                  path ← FindPathParallel(up)
10                 CheckPath(path, pqueue)
```

Für die Parallelisierung wird der Flash Algorithmus in zwei Teile aufgeteilt. Der Hauptthread durchläuft die beiden äußeren Schleifen wie in Algorithmus 7 dargestellt. Wenn nun der Algorithmus bei einem nicht bearbeiteten Knoten anlangt, wird dieser einem Thread übergeben, welcher nun unabhängig von anderen Threads FindPathParallel und CheckPath bearbeitet. Die einzige Modifikation von FindPathParallel im Gegensatz zu FindPath besteht darin, dass nur Knoten in Pfade aufgenommen werden, die in keinem anderen aktuellen Pfad enthalten sind. Außerdem haben alle Threads eine gemeinsame Vorrangwarteschlange. Alle Threads haben eine gemeinsame Schnappschuss Verwaltung (Historie) und nur beim Zugriff auf diese müssen sich die Threads synchronisieren. Die zweite Synchronisierung stellt sicher, dass immer nur ein Thread die Nachfolger-Knoten, gemäß der in Abschnitt 3.8 beschriebenen Strategie, sortiert. Abgesehen davon können die Threads unabhängig voneinander ihre Berechnungen durchführen. Der Pseudocode

ist in Algorithmus 10, Algorithmus 11 und Algorithmus 12 dargestellt.

Algorithm 12: FindPathParallel(node)

```
   Input: Start node node
   Result: Path of untagged nodes path
1  begin
2      path ← new list
3      while path.head() ≠ node do
4          path.add(node)
5          node.isInPath = true
6          foreach up ∈ node.successors do
7              if !up.tagged and !node.isInPath then
8                  node ← up
9                  break
10     return path
```

3.7.3 DFS-Flash ($Flash_{DFS}$)

Für den Fall, dass eine vollständige Klassifizierung des Suchraumes notwendig ist (siehe z. B. Fall 6 in Abschnitt 3.4), wird hier vorgeschlagen, eine DFS Strategie zu nutzen (siehe Abschnitt 3.5.5). Diese klassische Suchstrategie kann, durch die Modifikation „teste-dann-traversiere“, die Optimierungstechniken des Frameworks sehr gut ausnutzen, da besonders häufig roll-up Operationen genutzt werden können. Im Prinzip wird hierbei ein Pfad sequenziell von unten nach oben durchlaufen und dann ein Backtracking initiiert. Um den Speicherverbrauch und die Laufzeit zu reduzieren, werden für die Speicherung von Schnappschüssen und auch deren Entfernung aus dem Cache, andere Regeln als beim $Flash_B$ genutzt. Zum Ersten wird die Beschränkung aufgehoben, nur nicht-anonyme Knoten in die Historie zu legen, da bei einem vollständigen Durchlaufen des Verbandes auch Nachfolger von anonymen Knoten getestet werden können, somit werden potenziell alle Knoten in die Historie gelegt. Auch die Regel für das Verdrängen aus dem Cache wird geändert. Es werden nur solche Schnappschüsse aus dem Cache verdrängt, bei denen alle Nachfolger bearbeitet wurden. Auch hier wird wieder, sollte die maximale Anzahl an Schnappschüssen erreicht sein, eine LRU-Strategie als letzte Möglichkeit verwendet.

3.7.4 Zwei-Phasen-Flash ($Flash_{2P}$)

In diesem Abschnitt wird eine Kombination von $Flash_B$ (siehe 3.7) und $Flash_{DFS}$ (siehe 3.7.3) beschrieben. Dieser Algorithmus kann nun für die Fälle genutzt werden, in denen Teile des Kriteriums oder der Metrik monoton sind. Im Speziellen sind dies die Fälle 2-5 aus Abschnitt 3.4.

Die Hauptidee hierbei ist, Teile des Suchraumes durch ein monotones (Teil-)Kriterium auszuschließen, und die restlichen Teile systematisch mittels $Flash_{DFS}$ abzusuchen. Dieser neue Algorithmus wird als zwei Phasen Flash $Flash_{2P}$ bezeichnet. Die Anwendung des Algorithmus mit diesen zwei Phasen ist bei vielen Anonymitätskriterien möglich, da diese meistens das k-Anonymitätskriterium beinhalten. Zum Beispiel bei der im Allgemeinen nicht monotonen ℓ-Diversität muss

jede Gruppe auch immer ℓ-anonym sein, diese ℓ-Anonymität ist wieder monoton und kann genutzt werden, um Teile des Suchraumes auszuschließen. Im Falle der t-Closeness wird standardmäßig sogar explizit zusätzlich auch eine k-Anonymität der Gruppen gefordert. Dies ermöglicht zumindest alle Zustände auszuschließen, welche nicht k/ℓ-anonym sind und automatisch alle ihre Spezialisierungen. Dieser Schritt kann mit dem effizienten $Flash_B$ durchgeführt werden. Die verbleibenden Knoten sind somit auf alle Fälle k/ℓ-anonym und müssen jetzt noch vollständig auf das eigentlich zu prüfende Kriterium mit $Flash_{DFS}$, getestet werden. Dieser zwei Phasen-Algorithmus kann auch bei k-Anonymität alleine von Nutzen sein, wenn eine nicht monotone Metrik verwendet wird (siehe Fall 2). Somit wird im ersten Schritt der Suchraum vollständig und korrekt klassifiziert und es muss in der zweiten Phase nur noch die Metrik für alle k-anonymen Knoten berechnet werden. Hängt die Metrik nicht direkt von den Eingangsdaten ab (z. B. Height, Precision) wird in der zweiten Phase keine Transformation der Daten durchgeführt. Im Falle einer von den Daten abhängenden Metrik (z. B. AECS), muss allerdings jeder anonyme Zustand auf die Daten angewendet werden und die Metrik anhand der Ergebnisse berechnet werden.

$Flash_{2P}$ nutzt also in der ersten Phase $Flash_B$, um einen Pfad zu bilden. Dieser wird dann in binärer Weise getestet und dabei vorausschauend die Knoten, hinsichtlich des monotonen Teilkriteriums oder der Metrik, markiert. Nachdem der Pfad bearbeitet wurde, werden die nach dem Teilkriterium anonymen Knoten und alle ihre Generalisierungen mittels $Flash_{DFS}$, bearbeitet. Das bedeutet, dass, nachdem ein Pfad mit beiden Phasen bearbeitet wurde, alle von diesem Pfad aus erreichbaren Knoten klassifiziert sind.

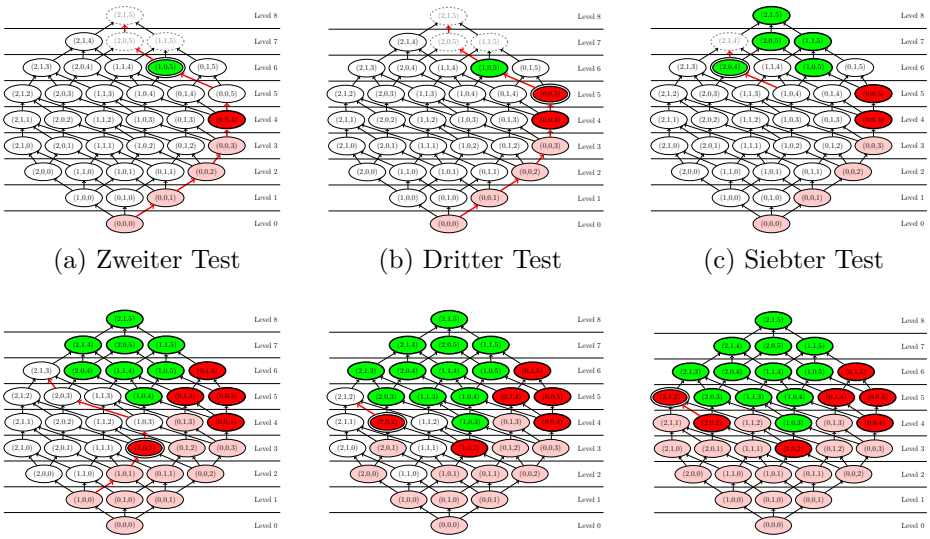
Algorithm 13: Äußere Schleife des zwei-Phasen-Flash Algorithmus

```
   Input: Verband lattice
1  begin
2      pqueue ← empty min-pqueue
3      foreach level ∈ lattice do
4          foreach node ∈ level do
           /* First Phase */
5              if !node.tagged then
6                  pqueue.add(node)
7                  while !pqueue.isEmpty do
8                      node ← pqueue.extractMin
9                      foreach up ∈ node.successors do
10                         if !up.tagged then
11                             path ← FindPath(up)
12                             head ← CheckPath(path, pqueue)
                           /* Second Phase */
13                             DFS(head)
```

Der Pseudocode der äußeren Schleife ist in Algorithmus 13 dargestellt. $Flash_{2P}$ iteriert über alle Knoten des Verbandes und startet mit dem untersten Level. Für alle Knoten wird nun FindPath($node$, 1) aufgerufen, wenn der Zustand eines Knotens

hinsichtlich des monotonen Teilkriteriums oder der Metrik unbekannt ist. Der Pfad wird, wie bei $Flash_B$, mittels einer gierigen Tiefensuche erstellt (siehe Algorithmus 8). Dieser Pfad wird nun in einer binären Weise getestet (siehe Algorithmus 9). Zusätzlich wird hierbei im Falle einer monotonen Metrik, wenn ein anonymer Knoten gefunden wird, alle Generalisierungen ausgeschlossen. Danach startet die zweite Phase des Algorithmus, hierbei wird FindPath($node, 2$) mit dem niedrigsten anonymen Knoten *node* des Pfades aufgerufen. Im Falle einer nicht monotonen Metrik werden alle erreichbaren Knoten nun komplett gegen das gesamte Kriterium getestet und deren Metrik berechnet (siehe Abschnitt 3.7.3). Im Falle einer monotonen Metrik werden nur die Nachfolger betrachtet, die nicht bereits durch die Metrik ausgeschlossen werden konnten. Wenn alle erreichbaren Knoten mittels $Flash_{DFS}$ abgearbeitet wurden, fällt der Algorithmus zurück in die erste Phase und bearbeitet nun den nächsten Knoten in der Vorrangwarteschlange. Ist die Vorrangwarteschlange leer, fährt der Algorithmus mit der äußersten Schleife fort.

Durch diesen Zwei-Phasen-Ansatz kann der Algorithmus beliebige Kombinationen von monotonen und nicht-monotonen Kriterien und Metriken bearbeiten, indem entweder nur die erste Phase, beide Phasen, oder nur die zweite Phase ausgeführt werden. Wenn nur die erste Phase ausgeführt wird, verhält der Algorithmus sich wie $Flash_B$, und nur monotone Kriterien und Metriken können benutzt werden. Wird hingegen nur die zweite Phase ausgeführt, wird $Flash_{DFS}$ nachgeahmt, und es muss kein monotones Teilkriterium existieren. Werden beide Phasen ausgeführt, wird die Möglichkeit des vorausschauenden Markierens benutzt, und dennoch wird effizient die optimale Lösung bei nicht monotonen Kriterien oder Metriken gefunden.



(a) Zweiter Test (b) Dritter Test (c) Siebter Test

(d) Dreizehnter Test (e) Achtzehnter Test (f) Letzter Test

Abbildung 3.10: Beispiel für $Flash_{2P}$: Rekursive-(4,2)-Diversität, 25 % Unterdrückung und AECS-Metrik für den Beispieldatensatz (Fall 4)

3.7.4.1 Beispiel für den zwei-Phasen-Flash Algorithmus

Um eine optimale Lösung für das nicht monotone rekursive-(4,2)-Diversitätskriterium mit 25 % Unterdrückung (≙ 2 Tupel) auf dem Beispieldaten-

satz von Abbildung 2.4 mit nicht-monotoner AECS Metrik (siehe Abschnitt 3.6.2) zu finden, durchläuft $Flash_{2P}$ unter anderem, folgende in Abbildung 3.10 dargestellten Schritte. Alter, Postleitzahl und Geschlecht sind die Quasi-Identifikatoren, Diagnose ist das sensible Attribut. Der erste Schritt des Algorithmus ist identisch mit dem in Abbildung 3.9a dargestellten. Als Erstes wird der rechte, rote Pfad erstellt und binär getestet. Damit ist der erste Knoten, der getestet wird $(0, 0, 4)$. Bei diesem Test wird nun die rekursiv-(4,2)-Diversität getestet und gleichzeitig auf die implizierte 2-Anonymität, da sie ein monotones Subkriterium ist. Der Knoten ist weder 2-anonym, noch rekursiv-(4,2)-divers, damit können auch alle seine Vorgänger als nicht anonym gekennzeichnet werden (hellrot). Der nächste Knoten der getestet wird, ist $(1, 0, 5)$. Da dieser anonym ist, sind alle Nachfolger dieses Knotens nun potenziell anonym (gestrichelt). Diese Knoten sind auf jeden Fall 2-anonym, aber potenziell nicht rekursiv-(4,2)-divers, da das Kriterium mit Unterdrückung nicht monoton ist. Als Nächstes wird der Knoten $(0, 0, 5)$, der binären Strategie folgend, getestet. Dieser ist nicht anonym. Nun setzt die zweite Phase des Algorithmus ein, der alle Nachfolger von $(1, 0, 5)$ mittels $Flash_{DFS}$ testet. Es werden dabei die Zustände $(2, 0, 5)$,$(2, 1, 5)$ und $(1, 1, 5)$ getestet (der Zustand des Verbandes nach diesen Tests ist in Abbildung 3.10c dargestellt). Nun fällt der Algorithmus wieder zurück in die erste Phase und erstellt den nächsten Pfad, der wieder binär durchsucht wird.

Die Knoten $(0, 0, 4)$ und $(0, 0, 5)$ liegen in der Vorrangwarteschlange. Gemäß der Strategie (siehe Abschnitt 3.8) wird, ausgehend vom Knoten $(0, 0, 4)$, der nächste Pfad aufgebaut (siehe Abbildung 3.10c). Wenn die Vorrangwarteschlange abgearbeitet ist, fällt der Algorithmus zurück in die äußere Schleife, und als Nächstes wird der in Abbildung 3.10d dargestellte Pfad getestet. Der vorletzte Test (siehe Abbildung 3.10e) wird auf der Transformation $(2, 0, 2)$ durchgeführt, die letzte getestete Transformation ist $(2, 1, 2)$. Der Zustand des Verbandes nach dem letzten Test ist in Abbildung 3.10f dargestellt. Die optimale Lösung ist in diesem Fall $(1, 0, 3)$.

3.8 Stabilität der Algorithmen: Vorschlag einer Ordnung

Eine besonders interessante Eigenschaft bei Algorithmen die, wie die Flash Algorithmen, den Verband vertikal traversieren, ist die Stabilität hinsichtlich der Laufzeit des Algorithmus. Das liegt daran, dass die Laufzeit, bei Anwendung der Optimierungen, wie in Abschnitt 3.6 beschrieben, von der Anzahl und der Abfolge der zu testenden Knoten abhängt. Je nachdem welche Knoten in dem Verband zuerst getestet werden, kann unterschiedlich vorausschauend markiert werden, was wiederum Einfluss auf die Auswahl des nächsten Knoten hat. Diese Unterschiede in der Laufzeit sind vor allem bei OLA signifikant [73]. Die Lösung für dieses Problem liegt darin, eine totale Ordnung über alle Knoten des Verbandes zu definieren, und im Falle einer Iteration über mehrere Knoten, dieser zu folgen. Die Hauptidee bei dieser Ordnung ist, die Knoten hinsichtlich ihrer Generalisierung zu ordnen, und die Knoten mit geringerer Generalisierung zu bevorzugen. Dafür werden alle Informationen, die ein Knoten in dem Framework besitzt, solange er noch nicht geprüft/generalisiert wurde, verwendet. Hierzu werden drei Kriterien benutzt. Das Level des Knotens, die mittlere Generalisierung des Knotens und das Verhältnis

an unterschiedlichen Werten auf dem aktuellen Level und dem Grundlevel. Eine detailliertere Beschreibung und Evaluation ist in [73] und [80] veröffentlicht.

Diese Kriterien werden genutzt beim Iterieren über die Knoten auf einem Level (z. B. Zeile 4 in Algorithmus 7), beim Iterieren über die Nachfolger eines Knotens (z. B. Zeile 10 in Algorithmus 7 und Zeile 5 in Algorithmus 8). Zuletzt dient diese Ordnung die Elemente in der Vorrangwarteschlange zu sortieren (Zeile 9 in Algorithmus 7). Trotz dieser drei Kriterien ist die definierte Ordnung keine Striktordnung, d. h., es gibt Knoten, die gleich sind. Damit ist theoretisch keine hundertprozentige Stabilität erreicht, die Abweichungen hinsichtlich der Laufzeit in den Experimenten waren aber vernachlässigbar gering. Außerdem ist zu beachten, dass die Sortierung aller Level des Verbandes und der Nachfolger aller Knoten, einen hohen Kostenfaktor darstellt. Deshalb wurde hierzu eine „lazy“ Strategie angewendet, und zum einen nur die Nachfolger der Knoten sortiert, die im Verlauf gebraucht werden. Zum anderen werden nicht die kompletten Level mit allen Knoten sortiert, sondern nur die Knoten, welche noch nicht markiert wurden. Diese Ordnung kommt bei allen drei Algorithmen der Flash Familie zum Einsatz.

3.9 Evaluation der Algorithmen

In diesem Abschnitt werden die Ergebnisse der Untersuchung der verschiedenen Algorithmen und Optimierungen anhand von verschiedenen Echtdatensätzen, hinsichtlich ihrer Laufzeiten und ihres Speicherverbrauches, dargestellt. Außerdem findet ein Vergleich mit den bekannten State-of-the-Art Algorithmen (siehe Abschnitt 3.5) statt.

3.9.1 Datensätze

Für die Evaluation wurden fünf Datensätze verwendet. Der erste Datensatz ist der „Adult Data Set“ (ADULT) des UCI Machine Learning Repository. Dieser besteht aus einem Auszug des 1994 US Zensus Datensatzes, welcher als de-facto Standard Benchmark in nahezu allen Publikationen zum Thema Anonymisierung verwendet wird. Für die Tests wurde der Datensatz von Einträgen mit leeren Feldern bereinigt. Es wurden dieselben Quasi-Identifikatoren wie bei El Emam [56] verwendet. Daraus resultierte ein Datensatz mit 30.162 Einträgen. Zusätzlich wurde der KDD Cup 1998 Datensatz (CUP) verwendet, welcher für die „Second International Knowledge Discovery and Data Mining Tools Competition“ erstellt wurde. Auch dieser wurde bereinigt, es bleiben 63.441 Datensätze übrig. Als dritter Datensatz werden die Daten über Autounfälle des „NTHSA Fatality Analysis Reporting System (FARS)“ mit 100.937 Einträgen genutzt. Des Weiteren wurden 539.253 Datensätze aus der „American Time Use Survey (ATUS)“ exportiert. Den größten Datensatz mit 1.193.504 stellt der Export des „Integrated Health Interview Series (IHIS)“ dar.

Die genaue Spezifikation der Datensätze ist in Tabelle 3.2 zu sehen. Die Anzahl der Zeilen liegt zwischen 30 K und 1 M, der Speicherplatzbedarf liegt zwischen 2,52 MB und 107,56 MB im unkomprimierten Zustand. Bei der Evaluation von Fall 1 wurde mit 8 bzw. 9 Quasi-Identifikatoren getestet. Die Generalisierungshierarchien haben eine Höhe zwischen zwei und sechs Leveln, was zu Verbandsgrößen zwischen 12.960 für ADULT und 45.000 für CUP führte.

Name	Quasi-Identifikator	Distinkte Werte	Höhe der Hierarchie	Anzahl der Datensätze	Anzahl Generalisierungen	Größe [MB]
ADULT[1]	sex	2	2	30.162	12.960 (4.320)	2,52
	age	72	5			
	race	5	2			
	marital-status	7	3			
	education	16	4			
	native-country	41	3			
	workclass	7	3			
	occupation*	14	3			
	salary-class	2	2			
CUP[2]	zip	13294	6	63.441	45.000 (9.000)	7,11
	age	94	5			
	gender	6	2			
	income	7	3			
	state	53	2			
	ramntall*	814	5			
	ngiftall	81	5			
	minramnt	58	5			
FARS[3]	iage	99	6	100.937	20.736 (5.184)	7,19
	irace	20	3			
	ideathmon	14	4			
	ideathday	33	4			
	isex	3	2			
	ihispanic	10	3			
	istatenum*	51	4			
	iinjury	8	3			
ATUS[4]	region	4	3	539.253	34.992 (8.748)	84,03
	age	83	6			
	sex	3	2			
	race	23	3			
	marital status	7	3			
	citizenship status	6	3			
	birthplace	155	3			
	highest level of school completed*	18	4			
	labor force status	6	3			
IHIS[5]	year	13	6	1.193.504	25.920 (12.960)	107,56
	quarter	4	3			
	region	4	3			
	pernum	25	4			
	age	86	5			
	marstat	10	3			
	sex	2	2			
	racea	16	2			
	educ*	26	2			

Tabelle 3.2: Testdatensätze und ihre Parameter

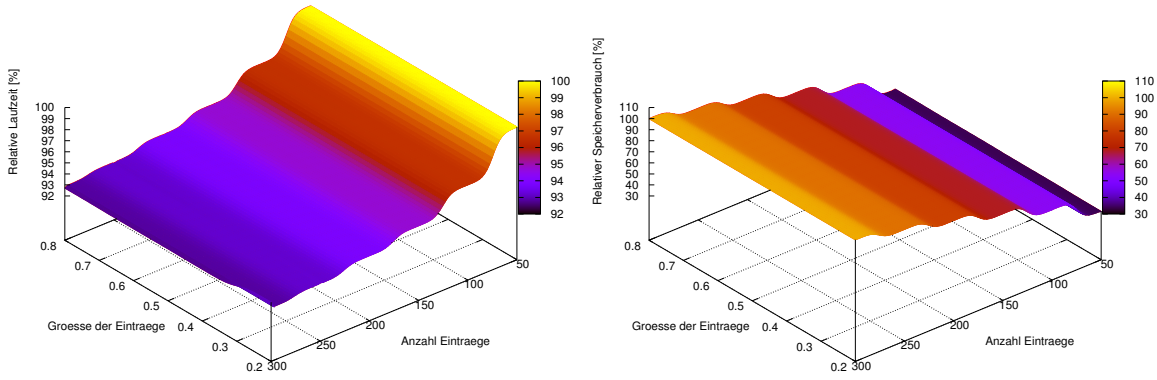
[1] http://archive.ics.uci.edu/ml/datasets/adult
[2] http://kdd.ics.uci.edu/databases/kddcup98/kddcup98.html
[3] http://www-fars.nhtsa.dot.gov/main/index.aspx
[4] http://atusdata.org/index.shtml
[5] http://www.ihis.us/

Bei den weiteren Tests (Fall 2-6) wurde jeweils eines der Attribute jedes Datensatzes als sensitives Attribut gekennzeichnet. Dieses ist mit * in der Tabelle 3.2 gekennzeichnet. Damit verringerte sich die Größe der Verbände auf 4.320 bis zu 12.960 Transformationen (angegeben in Klammern in Tabelle 3.2). Die unterschiedliche Anzahl an verschiedenen Werten pro Spalte liegt zwischen zwei für „salary-class“ und 13.294 für „zip“. CUP ist der Datensatz mit den meisten unterschiedlichen Werten. Insbesondere das Attribut „zip“ sticht dabei heraus.

3.9.2 Versuchsaufbau

Die Tests wurden auf einem normalen Desktop Rechner mit einer vier-Kern Intel CPU i5 mit 3.1 GHz und 8 GB RAM durchgeführt. Als Betriebssystem wurde ein 64-bit Linux mit 3.2 Kernel genutzt (Ubuntu 12.04 LTS). Java wurde als Implementierungssprache verwendet. Für die Evaluation kam eine Oracle 64-bit JVM zum Einsatz (1.7.0_51). Die maximale Heapsize wurde auf 2 GB festgelegt (-Xmx2G). Um die Ergebnisse verlässlicher zu machen, wurde vor dem Lauf einer Konfiguration ein Warm-up durchgeführt, welcher nicht in die Messungen einging. Die dargestellten Zeiten sind das arithmetische Mittel aus fünf aufeinanderfolgenden Messungen. Die dargestellten Zeiten enthalten nicht die Zeit für das Lesen der Daten von der Festplatte und das Wörterbuch codieren, dadurch sind diese Zeiten unabhängig von der Festplattengeschwindigkeit. In einer separaten Tabelle sind diese Daten dennoch gelistet. Die Transferrate des Testsystems beträgt ca. 80 MB/s.

Für die Tests bezüglich der Parallelisierung wurde ein 2x6-Core Intel Xeon X5690 Server mit 3.46 GHz verwendet. Der Rechner hat einen Hauptspeicher von 100 GB. Als Betriebssystem wurde ein 64-bit Linux mit 3.2.0 Kernel genutzt (Ubuntu 12.04 LTS). Als JVM wurde die 64-bit Version (1.7.0_09) von Oracle eingesetzt. Da es sich bei dieser Maschine um eine NUMA basierte Architektur (2 NUMA Knoten) handelt, wurden die Parameter für NUMA (-XX:+UseNUMA) und der dazu passende parallele Garbage-Collector (-XX:+UseParallelGC) gewählt. Auch in diesem Setup wurde ein Warm-up durchgeführt, der nicht in die Berechnungen einging. Die dargestellten Werte sind jeweils der Speedup Faktor über das arithmetische Mittel aus 10 aufeinanderfolgenden Messungen; auch hierbei sind die Zeiten für das Lesen der Daten nicht enthalten.



(a) Laufzeit (%)

(b) Speicherverbrauch (%)

Abbildung 3.11: Vergleich der Laufzeiten und des Speicherverbrauchs bei variierender Anzahl und Größe der Schnappschüsse

3.9.3 Wahl der Historie Parameter

Bei der genutzten Implementierung (siehe Abschnitt 3.6) wurden die Parameter für die Historie auf 20 % und 200 Einträge begrenzt und die folgenden Tests mit diesen Parametern durchgeführt. Der in Abbildung 3.11 dargestellte Plot zeigt, dass 200 Einträge und die maximale Größe von 20 % des original Datensatzes, einen guten Kompromiss zwischen Laufzeit und Speicherverbrauch darstellen.

Als Konfigurationen wurde der $Flash_B$ Algorithmus, mit allen fünf Datensätzen, mit k-Anonymität, jeweils 0 %, 2 % und 4 % Unterdrückung und Discernibility Metrik (DM^*) genutzt. Die beiden Plots zeigen jeweils das relative geometrische Mittel über alle Läufe hinsichtlich der Laufzeit und des Speicherverbrauchs. Die Parameter variieren bei der Größe zwischen 20 % und 80 %. Die Anzahl der Einträge wurde gemessen zwischen 50 und 300. Wie auf dem Laufzeitplot zu erkennen, flacht die Kurve bei ca. 200 Einträgen relativ stark ab, wohingegen der Speicherverbrauch, kontinuierlich mit der Anzahl der zu speichernden Schnappschüsse, steigt. Die Größe der zu speichernden Schnappschüsse hat im Mittel so gut wie keinen Einfluss auf die Laufzeit und den Speicherverbrauch.

3.9.4 Getestete Kriterien

Um ein breites Spektrum von Bedrohungen abzudecken, wurde die Evaluation auf eine breite Auswahl von Anonymitätskriterien und deren Kombinationen gestützt. Es wurden k-Anonymität, ℓ-Diversität, t-Closeness und δ-Präsenz alleine und für alle sinnvollen Kombinationen evaluiert. Diese Kombinationen umfassen die Kriterien alleine, also nur k-Anonymität (bezeichnet mit k), ℓ-Diversität (bezeichnet mit l), t-Closeness (bezeichnet mit t) und δ-Präsenz (bezeichnet mit δ). Als Nächstes wurden sinnvolle Kombinationen aus zwei Kriterien gebildet, diese sind: k-Anonymität mit ℓ-Diversität (bezeichnet mit (k, l)), mit t-Closeness (bezeichnet mit (k, t)) und mit δ-Präsenz (bezeichnet mit (k, δ)) um die Datensätze, jeweils gegen Identitätsaufdeckung und Attributsaufdeckung bzw. Mitgliedschaftsaufdeckung, zu schützen. Außerdem wurde δ-Präsenz mit ℓ-Diversität kombiniert (bezeichnet mit (δ, l)) und mit t-Closeness (bezeichnet mit (δ, t)) um die Datensätze vor Attributsaufdeckung und Mitgliedschaftsaufdeckung zu schützen. Als Letztes wurde k-Anonymität mit δ-Präsenz und ℓ-Diversität kombiniert (bezeichnet mit (k, δ, l)) und mit t-Closeness (bezeichnet mit (k, δ, t)) um die Datensätze vor allen drei Bedrohungen gleichzeitig zu schützen. Dies resultiert in 11 Kombinationen von Kriterien. Für k-Anonymität wurde immer $k = 5$ gesetzt, da dies ein typischer Wert für die biomedizinische Forschung ist. Von den verschiedenen Varianten von ℓ-Diversität und t-Closeness wurden die geläufigsten Varianten genutzt, d. h. Rekursive-(c, ℓ)-Diversität und t-Closeness mit der Earth-Mover's-Distanz (EMD) basierend auf Generalisierungshierarchien. Es wurden $c = 3$, $\ell = 4$ und $t = 0.2$ als Parameter gewählt, da diese Parameter häufig genutzt werden. Für δ-Präsenz wurde $\delta_{min} = 0.0$ und $\delta_{max} = 0.2$ mit einem 10 % zufällig generiertem Subset als Daten und dem Gesamtdatensatz als Weltwissen genutzt. Diese Wahl an Parametern resultiert in einer Menge von schweren Anonymisierungsproblemen, da nur bis zu 7 % der möglichen Transformationen jedes Suchraums, das jeweilige Kriterium erfüllen. Die Anzahl der anonymen Transformationen variiert pro Datensatz um bis zu einem Faktor von 42. Zusätzlich erlaubt die Festlegung auf fixe Parameter, die Algorithmen für Datensätze mit

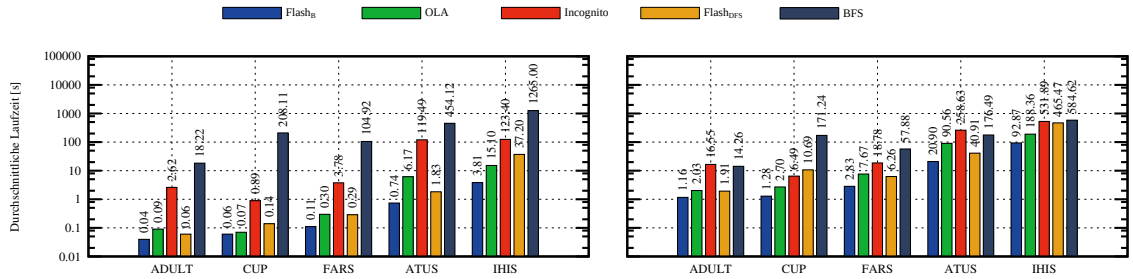
unterschiedlichen Eigenschaften zu vergleichen, und die Unterschiede zwischen den Kriterien herauszustreichen. Da k-Anonymität das am häufigsten verwendete Kriterium ist, wurde bei Fall 1 auch über verschiedene Werte von k variiert.

3.9.5 Fall 1: Monotone Kriterien und monotone Metriken

Fall 1 stellt den besten Fall dar, da hier die meisten Zustände aus dem Suchraum vorausschauend markiert werden können. Zuerst werden die Laufzeiten, dann der Speicherverbrauch untersucht. Um außerdem eine Framework unabhängige Bewertung vornehmen zu können, werden als Letztes die Anzahl der benötigten Anonymitätstests miteinander verglichen. Als Erstes wird k-Anonymität untersucht. Als zweiter Schritt werden die wichtigsten anderen monotonen (d. h. ohne Unterdrückung) Kriterien untersucht. Außerdem wird als Letztes eine parallele Implementierung von Flash$_B$ untersucht.

3.9.5.1 Laufzeiten bei k-Anonymität

Für den vollständig monotonen Fall 1 kam für die erste Evaluation k-Anonymität zum Einsatz, da dieses Kriterium auch mit Unterdrückung monoton ist, und in vielen anderen Arbeiten als Referenz genutzt wird. Damit sind die beiden speziellen Anonymisierungsalgorithmen OLA und Incognito hierbei besonders gut vergleichbar, da auch die Autoren der beiden Algorithmen dieses Kriterium nutzen. Der Parameter k wurde variiert zwischen 2 und 100, $k = \{2, 3, 4, 5, 6, 7, 8, 9, 10, 20, 50, 100\}$ und es wurde jeweils mit 0 % und 5 % Unterdrückung evaluiert. Als Metrik wurde die monotone non-uniform Entropie Metrik (NUEM) genutzt. Als Algorithmen wurden für diesen Fall Flash$_{DFS}$, BFS, Incognito, OLA und Flash$_B$ miteinander verglichen. Alle diese Algorithmen nutzen vorausschauendes Markieren. Flash$_{DFS}$ ist eine stabile Version von DFS (siehe Abschnitt 3.8 und Abschnitt 3.7.3). OLA wird mit natürlicher Ordnung getestet [73]. Die Datensätze wurden mit 8/9 Quasi-Identifikatoren konfiguriert und ohne ein sensitives Attribut (siehe Tabelle 3.2). Damit ergeben sich Verbandsgrößen zwischen 12.960 und 45.000. Das arithmetische Mittel, über jeweils fünf Läufe und alle 12 k-Parameter, ist in Abbildung 3.12 dargestellt. Die y-Achse zeigt die benötigte Ausführungszeit in Sekunden und ist logarithmisch skaliert.

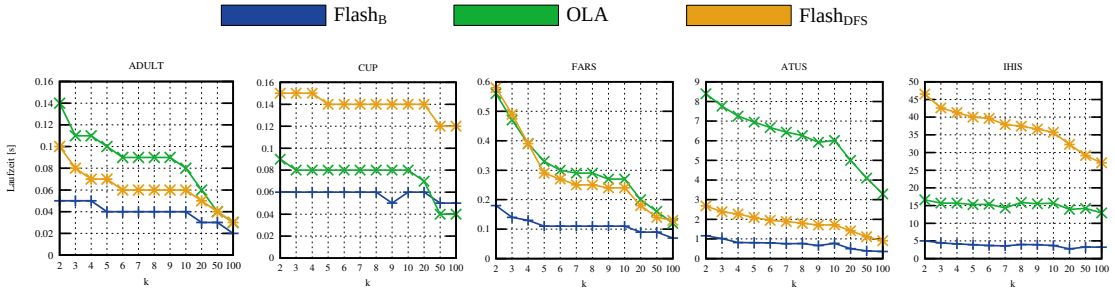


(a) 0 % Unterdrückung (b) 5 % Unterdrückung

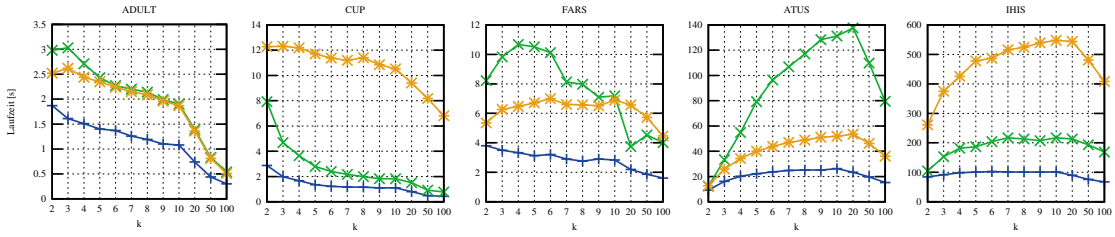
Abbildung 3.12: Gemittelte Laufzeiten für k-Anonymität

Man kann sehen, dass Flash$_B$, bei allen fünf Datensätzen und den beiden Unterdrückungsparametern, im Mittel immer der schnellste Algorithmus ist. Da alle Algorithmen mit demselben Framework implementiert wurden und damit alle die gleichen, oben beschriebenen Optimierungen nutzen können, ist dieser Vergleich fair.

Hinsichtlich der Laufzeiten bei 0 % und 5 % Unterdrückung ist die Rangfolge der Algorithmen, Flash$_B$, OLA, Flash$_{DFS}$, Incognito und BFS. Da BFS den Verband horizontal, von unten nach oben traversiert, kann er so gut wie kein vorausschauendes Markieren im 0 % Fall nutzen, da viele der Transformationen nicht k-anonym sind und BFS nur anonyme Transformationen (da „nach oben“ markiert werden kann) gut vorausschauend markieren kann. Man kann dies gut bei BFS im 5 % Fall sehen, wo zum Beispiel bei IHIS nur noch eine Laufzeit von ungefähr 580 Sek. im Gegensatz zu über 1200 Sek. bei 0 % Unterdrückung. Somit nähert sich die Laufzeit bei BFS im 5 % Fall den anderen Algorithmen an. Hierbei ist anzumerken, dass BFS selbst hierbei noch um mind. den Faktor 6 langsamer ist als Flash$_B$. Bei CUP, bedingt durch den großen Verband, ist BFS bei 5 % Unterdrückung sogar um mehr als den Faktor 133 schneller. Incognito verhält sich, da er auch eine horizontale Strategie verfolgt, ähnlich wie BFS und ist der zweit-langsamste Algorithmus. In den meisten Konfigurationen ist Incognito schneller als BFS, der Faktor liegt hierbei zwischen knapp 4 und über 200. Da aber Incognito durch sein Teilmengen Vorgehen unter Umständen mehr Arbeit macht als BFS, gibt es auch Konfigurationen (z. B. ADULT 5 % und ATUS 5 %) in welchen BFS um ca. 30 % schneller ist als Incognito. Die drei schnellsten Algorithmen Flash$_B$, OLA und Flash$_{DFS}$ verhalten sich, durch ihre vertikale Strategie den Verband zu durchlaufen, ähnlich. Interessant zu sehen ist, dass Flash$_{DFS}$, als „Standard-“Algorithmus in einigen Fällen, bedingt durch das hier genutzte Framework, schneller ist als OLA.



(a) 0 % Unterdrückung



(b) 5 % Unterdrückung

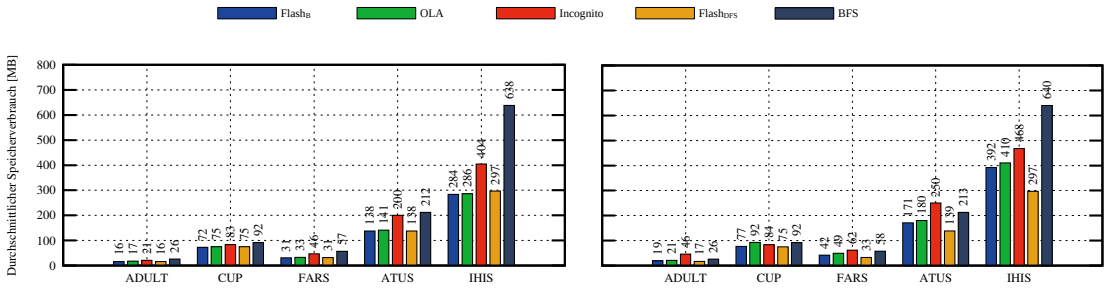
Abbildung 3.13: Laufzeiten für k-Anonymität, $2 \leq k \leq 100$ und Unterdrückung 0 %, 5 %

Dies gilt für fünf Konfigurationen (ADULT 0 %, ATUS 0 %, ADULT 5 %, FARS 5 % und ATUS 5 %), hierbei ist Flash$_{DFS}$ bis zu 70 % schneller. Flash$_B$ ist in diesen Tests, zwischen einem Faktor von 1.2 und mehr als 8, schneller als der derzeit beste Algorithmus OLA. Dieser Vorsprung ist besonders groß bei den beiden größeren Datensätzen ATUS und IHIS. Hierbei ist noch anzumerken, dass dieser Vorsprung

nicht nur im Mittelwert, sondern auch bei jeder (bis auf zwei Ausnahmen) einzelnen Konfiguration von k vorhanden ist. Für die beiden Unterdrückungsparameter 0 % und 5 % sind die absoluten Laufzeiten, gemittelt über jeweils fünf Läufe, in Abbildung 3.13 dargestellt. Zur besseren Übersicht wurden bei diesen Diagrammen die beiden langsamsten Algorithmen, BFS und Incognito, nicht gezeichnet. Man kann sehen, dass Flash$_B$ in jeder dieser Konfigurationen der schnellste Algorithmus ist. Die beiden einzigen Ausnahmen sind der CUP Datensatz mit 0 % Unterdrückung und 50- bzw. 100-Anonymität. Hierbei liegen die absoluten Laufzeiten allerdings im Millisekundenbereich und es sind nur drei von 45.000 Transformationen anonym. Bei allen drei Algorithmen kann man den Trend erkennen, dass bei größerem k die Laufzeiten abnehmen. Da, wenn Flash$_B$ zum Anonymisieren verwendet wird, die absoluten Laufzeiten, bis auf ATUS und IHIS mit jeweils 5 % Unterdrückung, unter 5 Sekunden liegen, stellt der Anonymisierungsprozess nun keinen Engpass bei einem etwaigen ETL-Prozess dar. Selbst bei den Konfigurationen, die über 5 Sekunden benötigen um eine optimale Lösung zu finden, liegen die Laufzeiten mit bis zu 102 Sekunden im vertretbaren Rahmen. Außerdem kann bei solch geringen Laufzeiten der Nutzer bei der Anonymisierung beinahe in Echtzeit Feinabstimmungen vornehmen.

3.9.5.2 Speicherverbrauch bei k-Anonymität

Das hier genutzte Framework benötigt nicht sehr viel Speicher, wie Abbildung 3.14 zeigt. Bei k-Anonymität liegt der Speicherverbrauch für Flash$_B$ zwischen 16 MB für ADULT mit 0 % und 392 MB für IHIS mit 5 % Unterdrückung. Der Speicherverbrauch verhält sich analog zu den Laufzeiten, auch hier benötigt Flash$_B$ am wenigsten Speicher, gefolgt von Flash$_{DFS}$, OLA, Incognito und BFS. Wie man aus diesen Zahlen entnehmen kann, ist das Framework auch bei anderen implementierten Algorithmen effizient. Der maximale Speicherverbrauch liegt bei 640 MB bei BFS und IHIS (≈ 1 Mio. Zeilen) mit 5 % Unterdrückung.



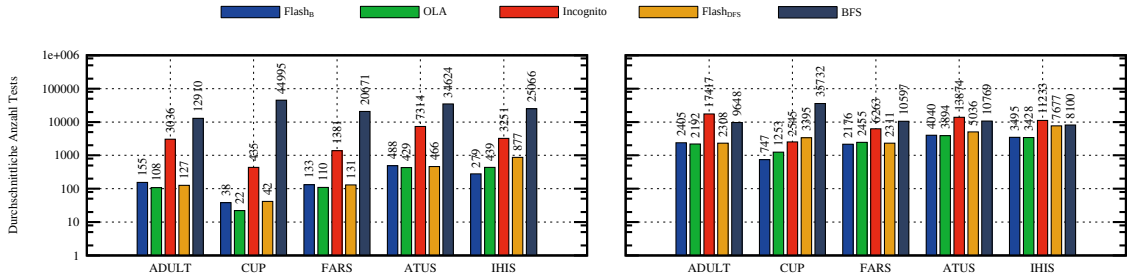
(a) 0 % Unterdrückung (b) 5 % Unterdrückung

Abbildung 3.14: Gemittelter Speicherverbrauch für k-Anonymität

3.9.5.3 Anonymitätstests und mögliche Roll-ups bei k-Anonymität

Die vorhergehenden Vergleiche sind sehr von dem verwendeten Framework abhängig. In diesem Abschnitt werden nun zwei Framework unabhängige Kennzahlen zum Vergleich herangezogen. Zum einen die Anzahl benötigter Tests gesamt, zum anderen die Anzahl an möglichen Roll-up Optimierungen. Diese Roll-up Optimierung kann leicht in jeder Implementierung verwendet werden. Die gemittelte Anzahl an

benötigten Anonymitätstests, d. h. die Anzahl der Transformationen, welche berechnet und auf Anonymität getestet werden müssen, sind in Abbildung 3.15 dargestellt.

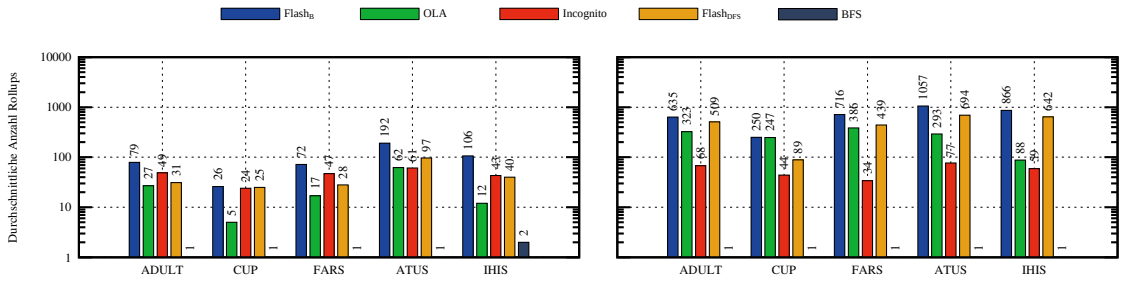


(a) 0 % Unterdrückung (b) 5 % Unterdrückung

Abbildung 3.15: Gemittelte Anzahl Anonymitätstests für k-Anonymität

Die Anzahl der Tests korreliert nur bedingt mit den oben gemessenen Ausführungszeiten, da nicht jeder Test, bedingt durch mögliche Optimierungen, gleich teuer ist. Insbesondere trifft dies für Incognito zu, da dieser Algorithmus ja Teilmengen der Quasi-Identifikatoren pro Transformation generalisiert. Generell kann man sagen, dass OLA und Flash$_B$ die wenigsten Tests benötigen. Bei den zehn aufgetragenen Messpunkten benötigt allerdings in sieben Fällen OLA weniger Tests. Die Differenz liegt bei bis zu 40 % bei CUP mit 0 % Unterdrückung. Allerdings gewinnt auch Flash in drei anderen Fällen mit bis zu wieder 40 % bei CUP mit 5 % Unterdrückung. Gemittelt über alle Konfigurationen gewinnt dennoch Flash knapp mit ca. 3 % weniger an benötigten Tests. BFS und Incognito benötigten bei diesen Konfigurationen die meisten Tests, um die optimale Lösung zu finden. Flash$_{DFS}$ ist mit Flash$_B$ und OLA vergleichbar.

Bei den möglichen Roll-ups ist Flash$_B$ der klare Sieger. Er kann in allen Fällen die meisten Roll-ups durchführen, darauf folgt Flash$_{DFS}$, OLA und Incognito. BFS kann so gut wie keine Roll-ups, wegen der horizontalen Strategie, durchführen. Dass Incognito, trotz horizontaler Strategie, so viele Roll-ups durchführen kann, liegt an den kleinen (d. h. mit wenigen Quasi-Identifikatoren) Verbänden, die durchlaufen werden. Dies ist auch bei 5 % Unterdrückung zu sehen, da hier trotz der größeren Anzahl an Tests, die Anzahl an möglichen Roll-up Operationen so gut wie nicht zunimmt. Die Zahlen sind in Abbildung 3.16 dargestellt.

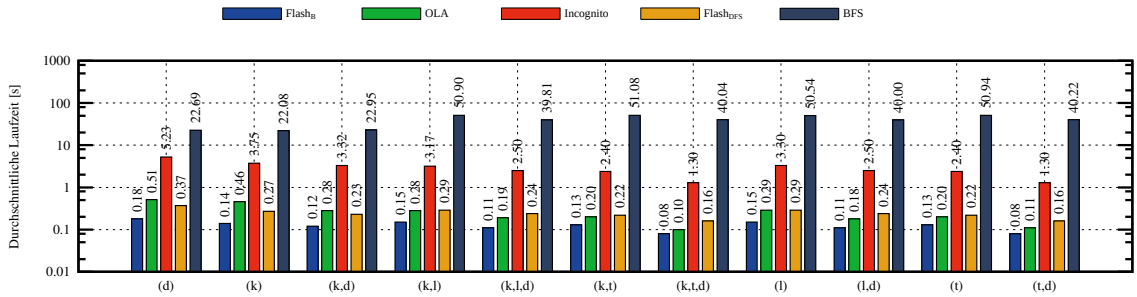


(a) 0 % Unterdrückung (b) 5 % Unterdrückung

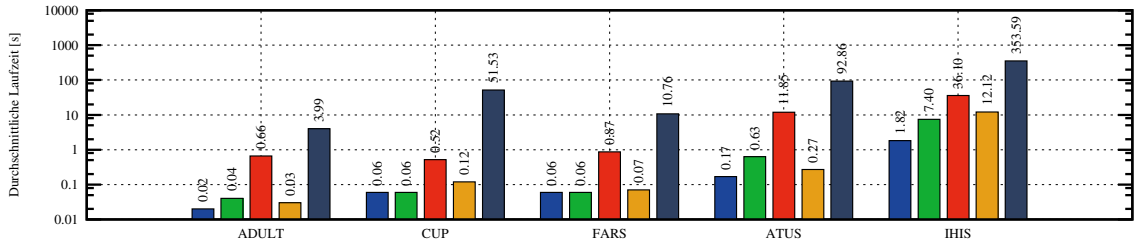
Abbildung 3.16: Gemittelte Anzahl Roll-up für k-Anonymität

3.9.5.4 Arbeitslasten für weitere monotone Kriterien

Als Nächstes werden die in Abschnitt 3.9.4 Kriterien und deren Kombinationen ohne Unterdrückung untersucht, da diese sonst nicht monoton sind. Als Metrik kam die monotone NUE-Metrik zum Einsatz. Zudem sind die hier dargestellten Verbände nun kleiner, da viele Konfigurationen ein sensitives Attribut benötigen. Die genaue Konfiguration ist in den Abschnitten 3.9.1 und 3.9.4 beschrieben. Die Laufzeiten und Arbeitsspeichermessungen wurden zuerst wieder über 5 Läufe arithmetisch gemittelt. In den (a) Abbildungen sind diese Mittelwerte über alle fünf Datensätze geometrisch gemittelt. Die Abbildungen (b) sind hingegen über alle 11 Kriterien geometrisch gemittelt. Durch diese Mittelung kann man die Performance der Algorithmen, bei verschiedenen Arbeitsbelastungen, untersuchen. Die Laufzeiten sind in Sekunden und der Speicherverbrauch in MB angegeben.



(a) Gemittelt über Datensätze

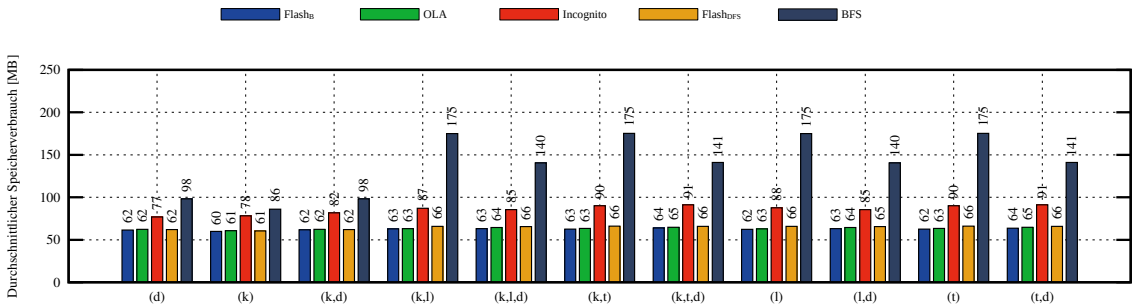


(b) Gemittelt über Kriterien

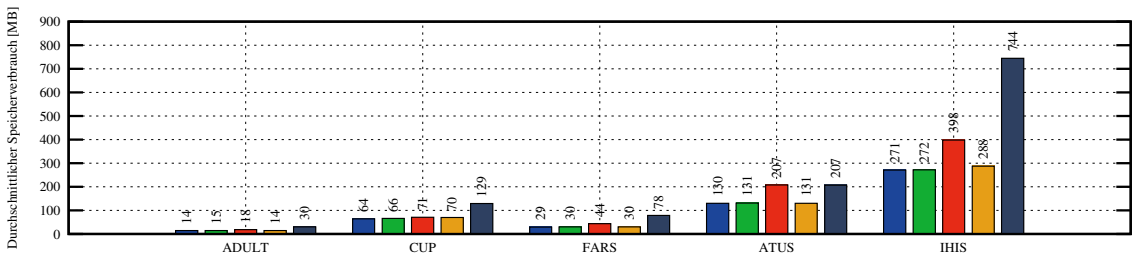
Abbildung 3.17: Gemittelte Laufzeiten für Fall 1

In Abbildung 3.17 sind die Mittelwerte der Laufzeiten über die Datensätze und die Kriterien dargestellt. Bei den getesteten Arbeitslasten ist Flash$_B$ immer der schnellste oder gleich schnell wie OLA. OLA und Flash$_{DFS}$ folgen. Auf Platz vier liegt Incognito, danach mit weitem Abstand BFS. In drei von 11 Fällen ist Flash$_{DFS}$ sogar schneller als OLA, bei der Mittelung über die Kriterien ist Flash$_{DFS}$ einmal schneller (ATUS) und einmal gleich schnell (ADULT). Flash$_B$ ist bei k-Anonymität bis zu einem Faktor von 3.3 schneller als OLA. Auch bei δ-Präsenz ist Flash$_B$ um 2.8x schneller als OLA. Bei der Kombination (k,t,d) ist Flash$_B$ immer noch ca. 25 % schneller als OLA. Gemittelt über die Kriterien ist OLA in zwei Fällen gleich schnell wie Flash$_B$, in den anderen drei Fällen ist Flash$_B$ wieder schneller. Besonders bei dem größten Testdatensatz IHIS ist Flash$_B$ im Mittel um den Faktor 4 schneller.

Beim Speicherverbrauch, dargestellt in Abbildung 3.18, zeigt sich ein ähnliches Bild wie bei den Laufzeiten.



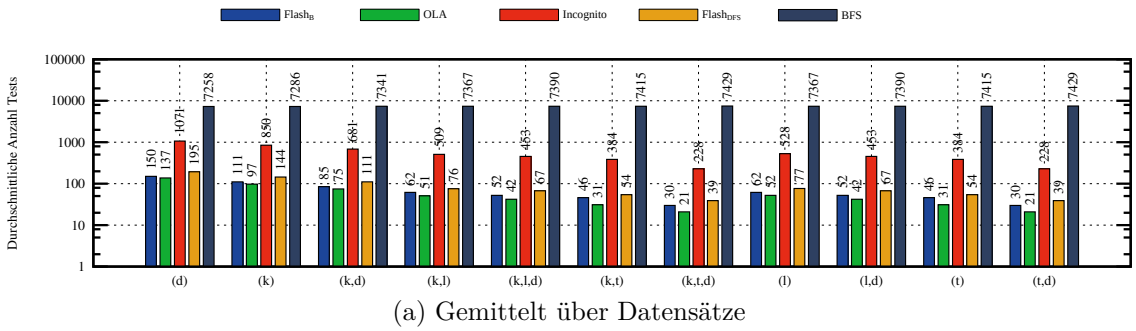
(a) Gemittelt über Datensätze



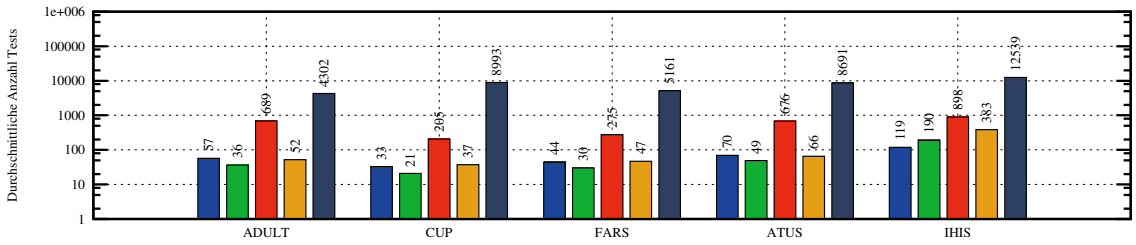
(b) Gemittelt über Kriterien

Abbildung 3.18: Gemittelter Speicherverbrauch für Fall 1

Flash$_B$ benötigt am wenigsten Speicher in dem hier genutzten Framework. OLA und Flash$_{DFS}$ liegen gemeinsam auf Platz zwei. Den dritten Platz belegt Incognito, gefolgt von BFS, welcher den meisten Speicherplatz benötigt. Dennoch liegen alle der getesteten Algorithmen (Ausnahme ist BFS) sehr dicht beieinander. Außerdem kann der Speicherverbrauch des Frameworks konfiguriert werden, was zu einem Space-Time Trade-off führt (siehe Abschnitt 3.9.3). Man kann dennoch sehen, dass die Strategie von Flash$_B$ bei IHIS über 2.7x weniger Arbeitsspeicher benötigt.



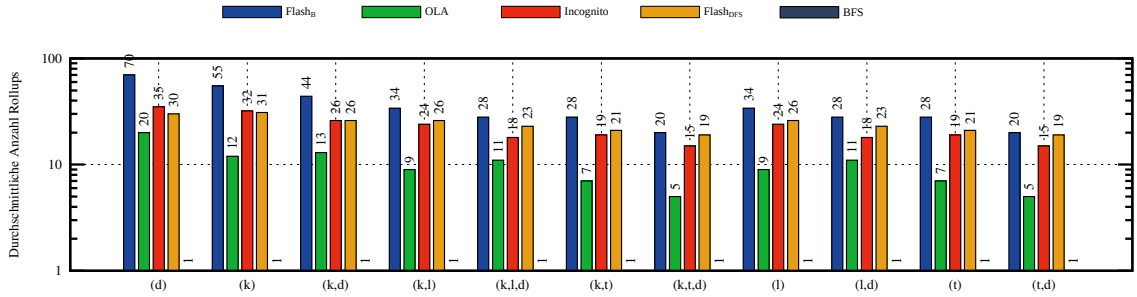
(a) Gemittelt über Datensätze



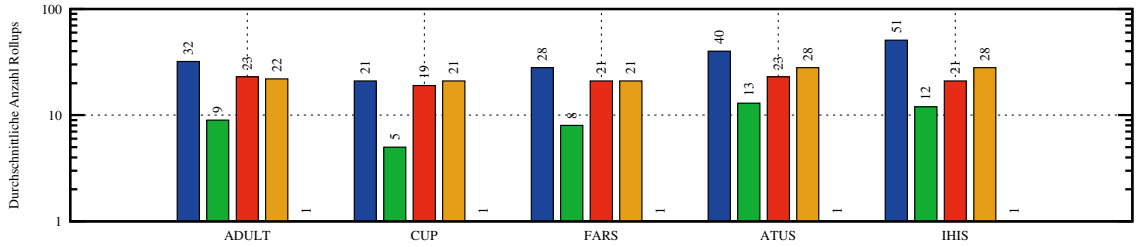
(b) Gemittelt über Kriterien

Abbildung 3.19: Gemittelte Anzahl Anonymitätstests für Fall 1

Wie bereits in Abschnitt 3.9.5.3 angesprochen, ist ein weiteres Vergleichskriterium die Anzahl der benötigten Anonymitätstests. Die Auswertung über diesen Parameter ist in Abbildung 3.19 zu sehen. Bei diesem Parameter schneidet OLA ein wenig besser ab als Flash$_B$. Die benötigten Anonymitätstests liegen aber maximal 33 % auseinander bei der Mittelung über die Datensätze. Bei der Mittelung über die Kriterien kann Flash$_B$ die optimale Lösung im Schnitt sogar mit weniger Tests als OLA finden. Hierbei ist wieder anzumerken, dass OLA instabil ist [73], d. h. sich die benötigten Tests stark ändern können, sollten die Spalten der Daten in einer anderen Reihenfolge vorliegen. Außerdem interessant zu sehen ist, dass in zwei Fällen (ADULT, ATUS) sogar Flash$_{DFS}$ weniger Tests benötigt als Flash$_B$. Incognito und DFS benötigen die meisten Tests, hierbei ist dieser Vergleich bei Incognito ein wenig unfair. Bei Incognito sind einzelne Tests oft billiger, da nur Teile der Quasi-Identifikatoren getestet werden.



(a) Gemittelt über Datensätze



(b) Gemittelt über Kriterien

Abbildung 3.20: Gemittelte Anzahl Roll-ups für Fall 1

Die Auswertung über die möglichen Roll-up Optimierungen sind in Abbildung 3.20 dargestellt. Wie schon bei k-Anonymität und den großen Verbänden (siehe Abschnitt 3.9.5.3) kann Flash$_B$ auch bei den anderen Kriterien die meisten potenziellen Roll-ups nutzen. Dadurch, dass OLA im Verband springt, muss dieser Algorithmus sich hierbei sogar auch noch Incognito und Flash$_{DFS}$ geschlagen geben.

Man kann aus den vorherigen Auswertungen erkennen, dass für Fall 1, also monotone Metriken und monotone Kriterien, Flash$_B$ der beste Algorithmus ist. Er ist in dem hier genutzten Framework der schnellste und benötigt am wenigsten Speicher. Sogar außerhalb des Frameworks kann Flash$_B$ mit OLA konkurrieren, insbesondere wenn die einfache Roll-Up Optimierung genutzt wird.

3.9.5.5 Speedup bei der Parallelisierung

Den Speedup der intra-operator Parallelisierung von Flash$_B$ bei k-Anonymität ist in Abbildung 3.21 dargestellt. Für jeden Messpunkt wurde das arithmetische Mittel

über $k = \{2, 3, 4, 5, 6, 7, 8, 9, 10, 20, 50, 100\}$ und fünf Läufe berechnet. Das Ergebnis wurde mit der nicht parallelisierten gemittelten Laufzeit (1 Kern) ins Verhältnis gesetzt. Die y-Achse zeigt den Speedup Faktor, die x-Achse zeigt die Anzahl der verwendeten Prozessorkerne.

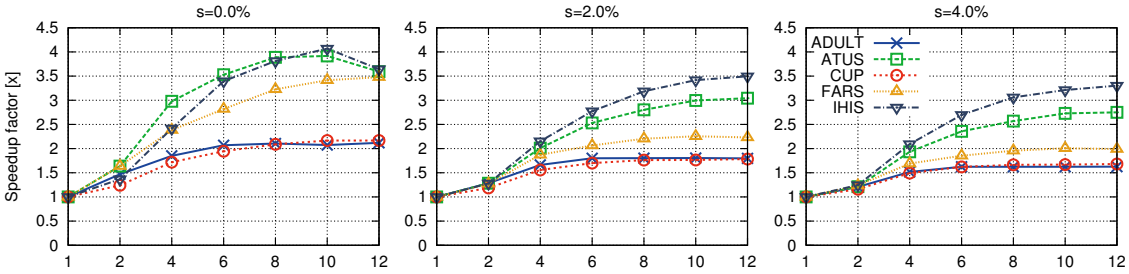


Abbildung 3.21: Speedup Faktor für intra-operator Parallelisierung für $1 \leq$ Kerne ≤ 12

Man kann hierbei deutlich sehen, dass die intra-operator Parallelisierung besonders bei den großen Datensätzen IHIS und ATUS einen Geschwindigkeitsvorteil bringt. Bei den beiden kleineren Datensätzen ADULT und CUP bringt der Einsatz von mehr als vier Kernen nahezu keinen Geschwindigkeitsvorteil. Der Speedup ist also mit der Größe des Datensatzes korreliert. Generell kann man feststellen, dass die aktuelle Implementierung nicht in dem wünschenswerten Speedup (4Kerne, 4x; 8Kerne 8x usw.) resultiert, der maximal Speedup von ca. 4x ist bei 10 Kernen und ATUS und IHIS Datensatz mit 0 % Unterdrückung erreicht. Andererseits gibt es keinen Speedup kleiner eins, d. h. der zusätzliche Overhead durch das zusätzliche Zusammenführen der Gruppierer der einzelnen Threads (siehe Abschnitt 3.6.1) überwiegt nicht den Geschwindigkeitsvorteil der parallelen Bearbeitung.

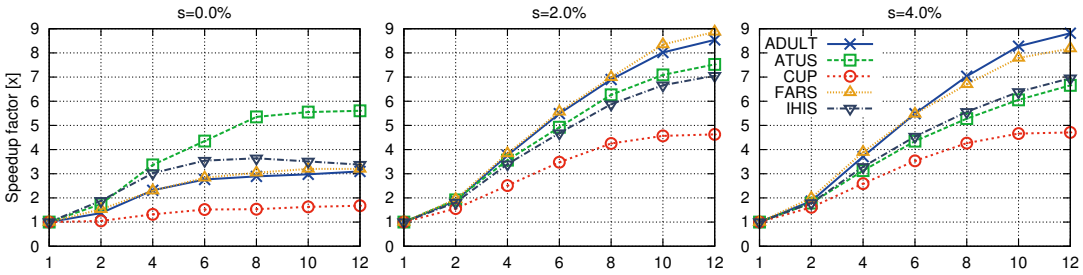


Abbildung 3.22: Speedup Faktor für inter-operator Parallelisierung für $1 \leq$ Kerne ≤ 12

Wird hingegen Flash$_B$ selbst parallelisiert, wie in Abschnitt 3.7.2 beschrieben, ist der Geschwindigkeitsvorteil deutlich zu sehen. In Abbildung 3.22 ist der Speedup bei inter-operator Parallelisierung dargestellt. Auch hier wurde wieder für jeden Messpunkt das arithmetische Mittel über die k's, wie oben angegeben, fünf Läufe berechnet, und mit der nicht parallelisierten Laufzeit (1 Kern) ins Verhältnis gesetzt.

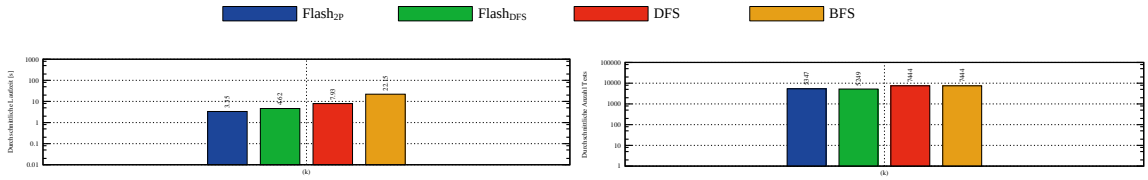
Bei 0 % Unterdrückung ist der Geschwindigkeitsvorteil eher gering. Dies liegt in der geringen Anzahl an notwendigen Anonymitätstests begründet. Bei größeren Unterdrückungsparametern (2 %, 4 %) ist der Speedup Faktor sehr gut. Interessant zu beobachten ist, dass CUP, welcher den größten Verband hat, am wenigsten von

der Parallelisierung profitiert. Dieser Effekt ist vermutlich darin begründet, dass die verwendete Parallelisierungsstrategie relativ lokal arbeitet und so wenig von dem vorausschauenden Markieren profitiert. Sehr gut funktioniert die Parallelisierung bei ADULT und FARS, hier ist der Speedup Faktor bei vier Kernen nahezu vier und liegt bei acht Kernen immerhin noch bei sieben. Bei der inter-operator Parallelisierung hängt der Speedup somit nicht primär von der Größe des Datensatzes ab, vielmehr von den Eigenschaften des Suchraumes. Trotz dieser vielversprechenden ersten Ergebnisse wird der Fokus in den folgenden Abschnitten wieder auf die Single-threaded Variante von Flash gelegt. Dabei werden weitere Kombinationen von Kriterien und Metriken untersucht.

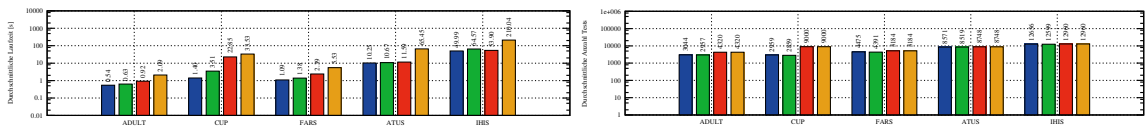
3.9.6 Fall 2-4: Teilweise monotone Kriterien und Metriken

Für die anderen Fälle 2-4, welche, zumindest teilweise nicht-monotone Kriterien beinhalten, wurden DFS, BFS, Flash$_{DFS}$ und Flash$_{2P}$ als Algorithmen genutzt, da nur diese auch mit Nicht-Monotonie umgehen können. OLA und Incognito können nur mit monotonen Metriken und Kriterien eine optimale Lösung finden. Im Prinzip muss bei Nicht-Monotonie der gesamte Suchraum durchsucht werden, deshalb sind DFS und BFS als worst-case Algorithmen angegeben. Weder BFS noch DFS verwenden vorausschauendes Markieren und durchsuchen somit immer den gesamten Suchraum. Wie in Abschnitt 3.4 beschrieben, können aber immer die monotonen Teile für das vorausschauende Markieren genutzt werden. In diesen Fällen wurden, jeweils zum Fall passend, verschiedenen Kriterien und Kombinationen genutzt. Es wurde immer 5 % Unterdrückung erlaubt.

Für Fall zwei wurde 5-Anonymität mit nicht monotoner Precision Metrik verwendet. Fall drei wird mit monotoner Entropie Metrik und den Kriterienkombinationen (l), (k, d), (k, t), (l, d), (k, t, d) und (k, l, d) getestet. Dieselben Kriterienkombinationen, aber mit nicht monotoner Precision Metrik, sind die Konfigurationen für Fall vier. Alle diese Kriterienkombinationen haben mindestens ein monotones Subkriterium.



(a) Gemittelt über Datensätze

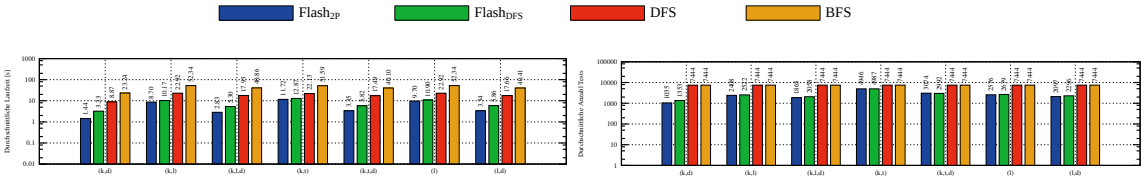


(b) Gemittelt über Kriterien

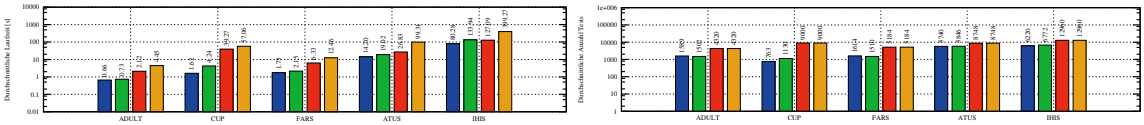
Abbildung 3.23: Gemittelte Laufzeiten und benötigte Anonymitätstests für Fall 2

Die Auswertungen für Fall zwei sind in Abbildung 3.23 dargestellt. Wieder wurden die Arbeitslasten berechnet, einmal über die Datensätze gemittelt und einmal über die Kriterien. Um die Effektivität des vorausschauenden Markierens aufzuzeigen, wurde jeweils den Laufzeiten die benötigte Anzahl an Anonymitätstests

gegenübergestellt. Interessant zu sehen ist, dass der zwei Phasen Flash weniger Zeit benötigt um die optimale Lösung zu finden, allerdings ein DFS mit stabiler Strategie und vorausschauendem Markieren, anhand des k-Anonymisierungskriteriums, weniger Tests benötigt. Nachdem allerdings bei 5-Anonymität viele Transformationen auch anonym sind, ist die Einsparung an Tests durch das markieren nicht sonderlich groß, da alle anonymen Transformationen auch, wegen der nicht-monotonen Metrik, getestet werden müssen.



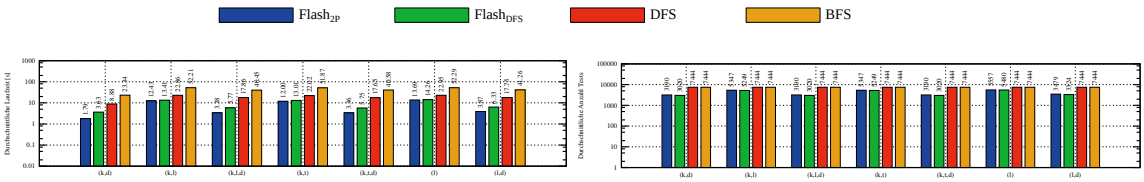
(a) Gemittelt über Datensätze



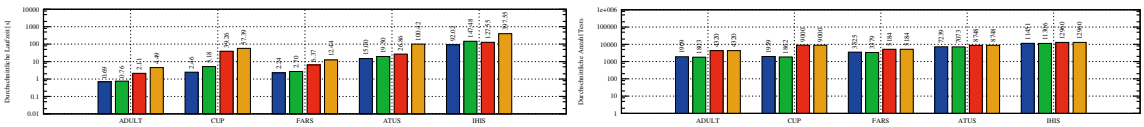
(b) Gemittelt über Kriterien

Abbildung 3.24: Gemittelte Laufzeiten und benötigte Anonymitätstests für Fall 3

Beim dritten Fall, siehe Abbildung 3.24, können mehr Transformationen ohne Anwendung ausgeschlossen werden. Bei der Kriterienkombination (k, d) zum Beispiel müssen im Schnitt nur ein Siebtel des Suchraumes von Flash_{2P} durchsucht werden. Bei dem Mittel über die Kriterien muss Flash_{2P} bei CUP sogar nur ein Elftel des Suchraumes testen. In allen Fällen ist der zwei Phasen Flash wieder der schnellste Algorithmus, hierbei benötigt er im Schnitt auch weniger Tests als Flash_{DFS}. Ausnahmen sind die Kriterienkombination (k, t, d) und der Datensatz FARS, wo Flash_{DFS} weniger Tests benötigt, als Flash_{2P}.



(a) Gemittelt über Datensätze



(b) Gemittelt über Kriterien

Abbildung 3.25: Gemittelte Laufzeiten und benötigte Anonymitätstests für Fall 4

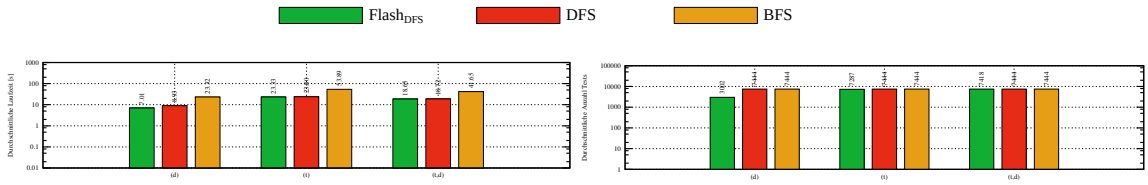
Selbst wenn das vorausschauende Markieren anhand der Metrik, wie in Fall 4, wegfällt, können immer noch Teile des Suchraumes durch das monotone Subkriterium ausgeschlossen werden. Die Auswertung ist in Abbildung 3.25 zu sehen. Im Falle

von (k, d), gemittelt über die fünf Testdatensätze, kann jetzt allerdings nur noch ca. die Hälfte des Suchraumes ausgeschlossen werden. Die nun noch zu testenden 3.190 Transformationen kann Flash_{2P} in nur 1,79 Sekunden testen, was einen Laufzeitvorteil gegenüber DFS (8.88 Sekunden) von fast einem Faktor 5 entspricht. Ähnlich verhält es sich auch bei (k, l, d), (k, t, d) und (l, d), also allen Kombinationen mit δ-Präsenz, was darauf hindeutet, dass ca. die Hälfte der möglichen Transformationen nicht 5-anonym bzw. bei (l, d) nicht 4-anonym ist. Bei der Mittelung über die Kriterien sticht CUP heraus, bei welchem Flash_{2P} weniger als ein Viertel des Suchraumes betrachten muss, was bei der Laufzeit in einem Speedup Faktor von fast 16x resultiert. Man kann sehen, dass hinsichtlich der Anzahl der benötigten Anonymitätstests Flash_{2P} und Flash_{DFS} sehr nahe beieinanderliegen, hierbei benötigt Flash_{DFS} fast immer weniger Tests, um die optimale Lösung zu finden. Bei der Laufzeit allerdings ist Flash_{2P} immer der schnellste der vier getesteten Algorithmen.

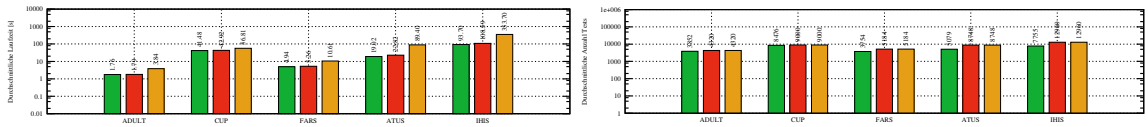
Abschließend kann man feststellen, dass bei teilweise monotonen Kriterien und monotonen und nicht monotonen Metriken, Flash_{2P} der schnellste Algorithmus ist, um eine optimale Lösung zu finden.

3.9.7 Fall 5-6: Nicht monotone Kriterien und Metriken

Die in diesem Abschnitt untersuchten Kriterien um Metrik Kombinationen unterscheiden sich von den Fällen 1-4, da bei diesen Fällen nicht nach dem Kriterium vorausschauend der Suchraum verkleinert werden kann. Damit gibt es hier nur noch drei Kandidaten für einen Vergleich. Flash_{DFS} welcher bei der Tiefensuche, zumindest im Fall 4, nach der Metrik vorausschauend markieren kann, DFS und BFS, welche beide alle möglichen Transformationen testen müssen. In Fall 6 muss nun der gesamte Suchraum durchsucht werden, damit können nur zwei Kandidaten DFS und BFS miteinander verglichen werden, da DFS und Flash_{DFS} sich hierbei identisch verhalten.



(a) Gemittelt über Datensätze

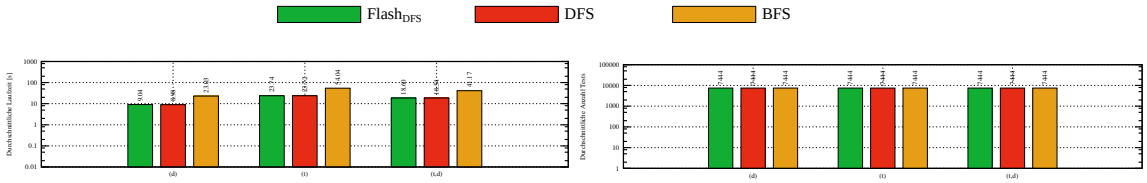


(b) Gemittelt über Kriterien

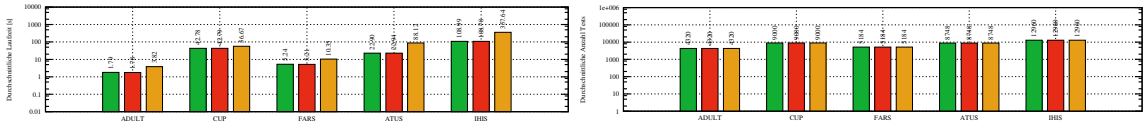
Abbildung 3.26: Gemittelte Laufzeiten und benötigte Anonymitätstests für Fall 5

Die Auswertungen für Fall 5, wo anhand der Metrik Teile des Suchraumes ausgeschlossen werden können, ist in Abbildung 3.26 dargestellt. Bei δ-Präsenz kann Flash_{DFS} noch, gemittelt über die Datensätze, mehr als die Hälfte aller Zustände ausschließen, bei t-closeness sind es nur noch knapp 2 % und bei der Kombination aus t-closeness und δ-Präsenz (t, d) sind es nur noch knapp 0,5 %. Bei der Mittelung über die Kriterien kann man sehen, dass die beiden großen Datensätze ATUS und

IHIS jeweils ca. 40 % des Suchraumes ausgeschlossen werden können. Die reduzierte Anzahl an Tests findet sich nicht im gleichen Verhältnis bei der Laufzeit wieder. Flash_{DFS} ist zwar immer der Schnellste, der Laufzeitgewinn gegenüber DFS beträgt aber nur zwischen ca. 0,5 % und 20 % bzw. ca. 2 % und 20 %. Hierbei können Differenzen unter 1 % als Messungenauigkeiten angesehen werden (vgl. nächster Fall 6).



(a) Gemittelt über Datensätze



(b) Gemittelt über Kriterien

Abbildung 3.27: Gemittelte Laufzeiten und benötigte Anonymitätstests für Fall 6

Zahlen für Fall 6 sind in Abbildung 3.27 dargestellt. Wie erwartet sind die Anzahl der benötigten Tests bei allen drei Algorithmen identisch, da aktuell keinerlei vorausschauendes Markieren verwendet wird. Auch in diesem Fall ist DFS schneller als BFS, obwohl beide Algorithmen den gesamten Suchraum durchsuchen müssen. Der Grund hierfür liegt wieder bei dem hier genutzten Framework, welches durch die Roll-up und Historie Optimierungen die vertikale Traversierung des Suchraumes begünstigt. Die Laufzeitunterschiede zwischen Flash_{DFS} und DFS sind der Messungenauigkeit geschuldet, da sich beide Algorithmen in diesem Fall identisch verhalten und liegen in allen Fällen unter 1 %. Interessant anzumerken ist, dass sich trotz Durchlaufen des gesamten Suchraumes eine optimale Lösung, in vertretbarer Zeit, finden lässt. Im Falle des ADULT Datensatzes kann man eine Lösung in weniger als 2 Sekunden finden. Selbst bei dem größten Datensatz IHIS liegt die benötigte Zeit bei unter zwei Minuten.

3.10 Diskussion der Algorithmen

Die Evaluation hat belegt, dass das hier genutzte Framework zusammen mit den entwickelten Algorithmen sehr effizient eine global optimale Lösung bei full-domain Generalisierung findet. Es wurden die neu entwickelten Algorithmen implementiert und evaluiert. Zum Vergleich mit den vorgestellten Algorithmen wurden zudem die in Abschnitt 3.5 beschriebenen Algorithmen implementiert. Das genutzte Framework unterstützt alle Algorithmen, die iterativ verschiedene Transformationen anwenden und auf Anonymität testen (bspw. [56, 61, 79]). Als mögliches Transformationsmodell kommt full-domain Generalisierung, gefolgt von Tupel-Unterdrückung, infrage. Auch verschiedene Metriken zum Messen des Informationsverlustes wurden implementiert. Insbesondere wegen der effizienten Implementierung der Historie und Roll-up Optimierung (siehe Abschnitt 3.6) gibt es aber eine Einschränkung. Da für

jede Gruppe immer nur ein Repräsentant gespeichert wird, ist es nicht einfach möglich herauszufinden, welche Zeilen des Originaldatensatzes in eine Gruppe fallen. Diese Information ist zum Beispiel normalerweise für die Berechnung der Entropie Metrik notwendig. In Abschnitt 3.6.2 wurde daher eine alternative Berechnung der Metrik vorgeschlagen, die mit dem hier vorgestellten Framework harmoniert. Die hier vorgestellte Algorithmenfamilie findet eine global optimale Lösung des Anonymisierungsproblems. Die Algorithmen wurden zwar speziell für das hier vorgestellte Framework entwickelt, können aber auch in anderen Frameworks noch effizient eine optimale Lösung finden. Da diese, genau wie OLA [56], den Generalisierungsverband vertikal traversieren und somit die Möglichkeit des vorausschauenden Markierens sehr gut nutzen können, um Teile des Suchraumes auszuschließen. Sollte es kein monotones (Teil-)Kriterium geben und damit der Suchraum vollständig durchsucht werden, kann die Algorithmenfamilie anonyme Transformationen für alle möglichen Kombinationen von Kriterien und Metriken finden. Der Algorithmus von Samarati [61] war einer der Ersten, welche eine global optimale Lösung sehr effizient berechnen konnte. Die Qualitätsmetrik ist hierbei allerdings auf die Höhenmetrik (siehe Abschnitt 2.6.1) eingeschränkt. Er nutzt dazu eine Binärsuche im Lösungsraum und benötigt, wegen der Beschränkung auf die Höhenmetrik, nicht einmal vorausschauendes Markieren, um Teile des Lösungsraumes auszuschließen. Durch die Beschränkung auf die spezielle Metrik ist der Algorithmus allerdings nicht direkt mit dem hier vorgestellten Algorithmus vergleichbar. Incognito [79] ist einer der ersten Algorithmen der eine beliebige, monotone Metrik unterstützt. Dabei benutzt er das Prinzip der dynamischen Programmierung, indem er die Tatsache ausnutzt, dass, wenn eine Teilmenge der Quasi-Identifikatoren nicht anonym ist, dann auch keine Obermenge anonym ist. Dazu baut Incognito die Potenzmenge aller Kombinationen der Quasi-Identifikatoren, was den Suchraum wesentlich vergrößert. Dieser Vergrößerung steht auf der anderen Seite die geringere Arbeit beim Testen auf Anonymität entgegen, da bei jedem Test einer Teilmenge von Quasi-Identifikatoren weniger Arbeit gemacht werden muss, als wenn alle Quasi-Identifikatoren transformiert und auf Anonymität getestet werden müssten. Zusätzlich kann er Teile des Suchraums, eben durch die Teilmengenbeziehung der Quasi-Identifikatoren, ausschließen. Dennoch zeigte sich in der Evaluation, dass sehr viele Tests auf Anonymität durchgeführt werden müssen und dafür viel Zeit benötigt wird. Der Optimal Lattice Anonymization (OLA) Algorithmus [56] war der bisher effizienteste Algorithmus. Er benötigt, durch seine „divide-and-conquer“ Strategie, sehr wenige Anonymitätstests. Er durchsucht den Suchraum in binär und kann dabei, durch die Beschränkung auf monotone Metriken und Kriterien, vorausschauendes Markieren effektiv einsetzen. Wie in [73] gezeigt ist er allerdings nicht stabil, d. h., seine Laufzeit hängt z. B. von der Reihenfolge der Spalten des Eingabedatensatzes ab. Da der Algorithmus durch seine Strategie im Suchraum umherspringt, können auch die Optimierungen des Frameworks nicht ihr volles Potenzial zeigen. Auch die effiziente Implementierung der Traversierung des Lösungsraumes ist nicht trivial, siehe [72]. Die hier vorgestellte Algorithmenfamilie hat diese Einschränkungen nicht und ist dabei effizienter und flexibler. Zwei klassische Traversierungsstrategien bei Graphen sind die Tiefensuche (DFS) und die Breitensuche (BFS). Diese sind nicht an das konkrete Problem angepasst, sind allerdings sehr einfach zu implementieren. Beide Strategien können so implementiert werden, dass sie vorausschauendes Markieren verwenden, um Teile des Suchraumes

auszuschließen. Der Nachteil bei BFS liegt, wegen der horizontalen Komponente bei der Breitensuche darin, dass zum einen die Optimierungen des Frameworks nicht optimal funktionieren und zum anderen, dass nur in eine Richtung (anonyme Transformationen) vorausschauend markiert werden kann. Bei der Tiefensuche hingegen können sowohl die Optimierungen effektiv genutzt werden als auch das potenziell mögliche vorausschauende Markieren. Aus diesem Grund wurde bei der hier vorgestellten Algorithmenfamilie DFS genutzt, wenn eine vollständige Durchsuchung des Suchraumes notwendig ist.

3.11 Zwischenfazit und Perspektiven

Im Rahmen dieser Arbeit wurde eine neue Familie von Algorithmen entwickelt. Diese ist sowohl bei der Laufzeit als auch beim Speicherverbrauch in dem hier verwendeten Framework effizient. Die Algorithmen durchsuchen den Suchraum effizient und die Optimierungstechniken, die das Framework anbietet, werden sehr gut genutzt. Damit stellen sie erstmalig eine umfassende, effiziente und flexible Möglichkeit dar, optimale Lösungen von verschiedenen Kriterien und Metriken, mit globaler full-domain Recodierung, zu finden. Es wurde außerdem mit Flash_{2P} der erste Algorithmus vorgestellt, der auch nicht monotone Kriterien unterstützt und dabei trotzdem alle Möglichkeiten nutzt, nach Teilkriterien vorausschauend zu markieren. Dadurch ist es nun möglich, effizient einen Datensatz vor Identitäts-, Attributs- und Mitgliedschaftsaufdeckung zu schützen, bei akzeptablem Informationsverlust. Dies wurde möglich, da auch bei erweiterten, nicht-monotonen Kriterien, nun Ausreißer automatisch ausgeschlossen werden können und trotzdem die optimale Lösung gefunden wird. Aktuelle Algorithmen können nicht garantieren in diesem Szenario eine optimale Lösung zu finden, da die hier untersuchten Kriterien mit Unterdrückung nicht monoton sind. Der Nachweis wurde in Abschnitt 3.2 geführt.

Die Algorithmen wurden umfassend mit k-Anonymität, ℓ-Diversität, t-Closeness und δ-Präsenz getestet. Dabei wurde festgestellt, dass der hier vorgestellte Algorithmus und das genutzte Framework alle anderen State-of-the-Art Algorithmen bezüglich Laufzeit, Speicherverbrauch und Flexibilität schlägt. Generell wurde dabei festgestellt, dass ℓ-Diversität und t-Closeness mehr Speicher benötigen als k-Anonymität, da für jede Gruppe auch das Frequenzset der sensiblen Attribute mitgeführt werden muss.

Die hier entwickelten Verfahren wurden für den Einsatz in der medizinischen Forschung entwickelt, lassen sich aber einfach auf andere Domänen übertragen, die Daten anonymisieren wollen. Die jeweilige Auswahl des richtigen Verfahrens ist Anwendungsfall spezifisch und wird auch in der Community diskutiert [42]. Heutzutage üblich ist k-Anonymität mit $3 \leq k \leq 25$ [62]. Ein ungelöstes Problem hierbei ist die konkrete Auswahl der Quasi-Identifikatoren. Es gibt Ansätze in der Literatur diese automatisch zu identifizieren [45], wobei der manuelle Ansatz die Quasi-Identifikatoren aus der Literatur (z. B. HIPAA [3] oder Review von Reidentifikationsangriffen [24]) zu nehmen, in der Praxis überwiegt.

Erweiterungen dieser Arbeit können in verschiedenen Richtungen erfolgen. Zum einen können weitere Ansätze untersucht werden, weitere Teile des Suchraumes z. B. durch monotone Untermetriken auszuschließen. Zum anderen könnte auch das genutzte Framework erweitert werden, hierbei wäre z. B. eine Festplatten basierte

Anonymisierung denkbar, um auch riesige Datenmengen, die trotz Wörterbuchkomprimierung nicht in den Hauptspeicher passen, zu anonymisieren. Auch könnte die proof-of-concept Implementierung der Parallelisierung verbessert, sowie auf die anderen Mitglieder der Flash Familie erweitert werden. Als interessanteste, weitere Arbeit könnte das Recodierungsmodell geändert werden, um noch mehr Informationen bei der Anonymisierung zu erhalten. Beispiele wären Slicing [81] und Subtree Generalisierung, wie in [82] beschrieben. Slicing kommt ohne Generalisierung aus, da es die Verbindung zwischen Quasi-Identifikatoren und den anderen Attributen löst und so eine höhere Datenqualität ermöglicht. Die Integration von Slicing ist einfach möglich, da nur die Ausgabe der anonymisierten Daten verändert werden müsste. Die Unterstützung von Subtree Generalisierung hingegen würde eine starke Überarbeitung der dargestellten Optimierungen des Frameworks erfordern. Dabei müsste wohl auch von der Forderung nach der optimalen Lösung abgerückt werden, da es hier zu einer kombinatorischen Explosion möglicher Lösungen kommen kann. Hierbei wäre eine weitere interessante Idee die beiden Ansätze, globales Recodieren und Subtree Generalisierung, zu kombinieren und z. B. zuerst mittels globaler Recodierung eine optimale Lösung zu finden und diese dann rekursiv mittels Subtree Generalisierung zu verfeinern, solange das Anonymitätskriterium erhalten bleibt. Eine andere Klasse von Algorithmen nutzt heuristische Verfahren, um eine Lösung des Anonymisierungsproblems zu finden. Die Top-Down-Specialization [82] nutzt zwar ein anderes Recodierungsmodell, könnte aber auch genutzt werden, um bei der hier verwendeten full-domain Generalisierung eine Lösung zu finden. Heuristische Verfahren werden normalerweise dann genutzt, wenn eine optimale Lösung nicht oder nur nach inakzeptabel langer Zeit gefunden werden kann. Bei den hier untersuchten Fällen ist dies allerdings nicht der Fall, darum erscheint ein heuristischer Ansatz nicht zwingend. Sollte es allerdings Fälle geben in denen die Dimensionalität höher ist, wären heuristische Verfahren eine gute Alternative. Generalisierungshierarchien, welche die Regeln der Vergröberung vorgeben, sind aufwendig zu generieren und erfordern oft Domänenwissen. Dabei können Experten maximal durch z. B. Wizards unterstützt werden, müssen aber selbst viel Arbeit investieren. Ansätze, die auf clustering Verfahren beruhen (z. B. [83]), benötigen keine benutzerdefinierten Hierarchien, können deshalb allerdings nur schwer auf einen speziellen Anwendungsfall angepasst werden. Eine einfache Möglichkeit die Vorteile von clustering Verfahren und Hierarchien zu kombinieren ist es, zur Erstellung der Hierarchien clustering Verfahren zu nutzen. Das generische Framework und die Implementierungen der hier vorgestellten Algorithmen sind als open-source Software unter http://arx.deidentifier.org/ verfügbar.

KAPITEL 4

Anonymisierung von verteilt vorliegenden Daten

In Kapitel 3 wurde der häufige Fall betrachtet, dass Daten, die anonymisiert werden müssen, unter einer einheitlichen Hoheit stehen und lokal vorliegen. In diesem Kapitel wird nun der Fall betrachtet, dass Daten, die anonymisiert werden müssen, unter verschiedenen Hoheiten stehen und verteilt vorliegen. Oft ist allerdings ein physisches zusammenführen der personenbezogenen Mikrodaten an einer lokalen Stelle verboten [84]. Zur besseren Generierung und Verifizierung von Forschungshypothesen besteht aber oftmals der Wunsch oder sogar die Notwendigkeit, diese verteilten Daten Forschern verfügbar zu machen. Beispiele für dieses Szenario finden sich bei DataSHIELD [50], SHRINE [51] und Mohammed et al. [85]. DataSHIELD beschreibt den Fall, dass mehreren Studien in einer gemeinsamen Analyse bearbeitet werden sollen, ohne dass personenbezogene Daten die jeweiligen Zentren verlassen. SHRINE und Mohammed et al. beschreiben den Fall, dass Daten über verschiedene Krankenhäuser verteilt vorliegen. Eine einfache Lösung wäre es, die Daten jeweils direkt bei den Zentren zu anonymisieren und erst die anonymisierten Daten zusammenzuführen. Es liegt nahe, dass integrierte Ansätze der Anonymisierung gegenüber der Anonymisierung der Einzelpools von Vorteil sind. Zum einen kann die Datenqualität bei dem integrierten Ansatz höher ausfallen, zum anderen kann eine nachträgliche Zusammenführung anonymisierter Daten sogar zu einer Verletzung der angestrebten Datenschutzgarantien führen. Diese beiden Vorteile werden in Abschnitt 4.1.2 genauer beleuchtet. Ziel ist es ein nicht-interaktives (d. h. Weitergabe von anonymisierten Mikrodaten) Verfahren zu entwickeln, welches eine höhere Datenqualität ermöglicht als der einfache Ansatz, und dabei dennoch den Datenschutz der Probanden sicherstellt. Wie bereits in Kapitel 3 angesprochen, müssen Lösungskonzepte flexibel und effizient gestaltet werden. Es müssen verschiedenste Anonymisierungskriterien, -metriken und -algorithmen unterstützt werden. In diesem Kapitel werden neue, flexible und effiziente Methoden vorgestellt, um unter unterschiedlicher Hoheit verteilt vorliegende Daten, so zu anonymisieren, dass die Nutzbarkeit der Daten gewährleistet bleibt und keine der Parteien nicht-anonymisierte Daten sieht. Teilergebnisse dieses Kapitels wurden publiziert in [86]. Außerdem wurde zu diesem Thema eine Diplomarbeit betreut [87].

4.1 Methodische Grundlagen

4.1.1 Verteilung der Daten

Im Allgemeinen können die Daten auf zwei verschiedene Weisen verteilt vorliegen. Zum einen horizontal verteilt, d. h. die Parteien haben das gleiche Schema, die Einträge allerdings stammen von unterschiedlichen Patienten. Zum anderen vertikal verteilt, hierbei sind die Daten von denselben Patienten, allerdings werden hier, bei den Parteien, jeweils unterschiedliche Attribute gespeichert. Das Schema ist also unterschiedlich (vergleiche horizontale und vertikale Fragmentierung bei verteilten Datenbanken [88]). Etwas formaler ausgedrückt sind die beiden Verteilungen in folgenden Definitionen beschrieben:

Definition 4.1 (*Horizontal verteilte Daten*)

Sei $A = \{a_1, ..., a_k\}$ eine Menge von k Attributen und $T = \{t_1, ..., t_n\}$ eine Menge von n Tupeln über diesen Attributen. Sei weiterhin $P = \{p_1, ..., p_l\}$ eine Menge von l Teilnehmern, dann heißt die Relation horizontal verteilt, wenn p_i die Menge der Tupel $T_{p_i} = \{t_x | 1 \leq x \leq n\}$ gespeichert hat, wobei $T_{p_y} \cap T_{p_x} = \emptyset$ für $1 \leq y, x \leq l$ und $y \neq x$.

Definition 4.2 (*Vertikal verteilte Daten*)

Sei, wieder, $A = \{a_1, ..., a_k\}$ eine Menge von k Attributen und $T = \{t_1, ..., t_n\}$ eine Menge von n Tupeln. Sei weiterhin $P = \{p_1, ..., p_l\}$ eine Menge von l Teilnehmern, dann heißt die Relation vertikal verteilt, wenn p_i die Menge der Attribute $A_{p_i} = \{a_x | 1 \leq x \leq k\}$ gespeichert hat, wobei $A_{p_y} \cap A_{p_x} = \emptyset$, für $1 \leq y, x \leq l$ und $y \neq x$.

Neben diesen zwei Extremen sind natürlicherweise auch Mischformen denkbar, welche als *hybrid* bezeichnet werden. Auch kann die Schnittmenge der Attribute/Tupel in manchen Fällen nicht leer sein, d. h., die Disjunktheit muss nicht gegeben sein. In diesem Fall können allerdings Konsistenzprobleme bei den Daten auftreten.

Im Weiteren gilt die Beschränkung auf die beiden, oben genannten, Definitionen und es wird davon ausgegangen, dass die Integrationsprobleme bereits gelöst wurden. Zudem wird gefordert, dass es im vertikal verteilten Fall einen gemeinsamen *Tupel Identifikator (TID)* gibt. Dieser repräsentiert die Beziehungen zwischen den Tupeln der verschiedenen Parteien.

4.1.2 Anonymisierungsmethoden für verteilte Daten

Um verteilt vorliegende Daten sicher zu anonymisieren, haben Jurczyk und Xiong in [89], drei mögliche Herangehensweisen beschrieben und stellen die daraus resultierenden Architekturen dar.

Das einfachste Verfahren kann man als *integriere-und-anonymisiere* bezeichnen [85]. Hierbei werden die Daten zunächst bei einer vertrauenswürdigen Stelle zentral integriert und danach anonymisiert (z. B. mittels k-Anonymität). Die zentrale Stelle muss hier vertrauenswürdig sein, da sie alle Daten in nicht anonymisierter Form, wenn auch nur kurzzeitig, bearbeitet. In vielen Szenarien gibt es entweder keine vertrauenswürdige Partei oder es ist, wie in dem hier angenommen Szenario, eine Integration aber, je nach Einverständniserklärung, nur im anonymisierten Zustand zulässig und/oder erfordert zusätzliche Verträge (z. B. data-use-agreements).

Aus diesem Grund wird dieser Lösungsansatz hier nicht weiter verfolgt.

Das zweite offensichtliche Vorgehen beschreitet den entgegengesetzten Weg und kann als *anonymisiere-und-integriere* bezeichnet werden [85]. Die Idee ist, die lokalen Datenbestände zuerst zu anonymisieren und im nächsten Schritt diese anonymen Daten zu integrieren. Doch gibt es bei diesem Vorgehen mindestens zwei Probleme. Zum Ersten kann es, da die Daten lokal anonymisiert werden, zu einem unnötigen Informationsverlust bei der Anonymisierung kommen. Wenn die Daten z. B. k-anonymisiert werden und horizontal verteilt vorliegen, kann es passieren, dass die Daten stärker generalisiert werden, als dies bei einem „integriere-und-anonymisiere“ Vorgehen notwendig wäre (für eine experimentelle Evaluation siehe Abschnitt 4.3). Im vertikal verteilten Szenario hingegen, wo die Quasi-Identifikatoren über mehrere Datenbestände verteilt sein können, kann die Anonymisierung durch die Integration im Nachhinein wieder zerstört werden. Betrachtet man den Fall, dass PLZ, Geschlecht und Alter einer Person einem Angreifer bekannt sind. Im Falle der k-Anonymität müsste jede Kombination dieser drei Quasi-Identifikatoren mindestens k-mal in der Tabelle vorhanden sein. Wenn nun aber z. B. die PLZ in einer anderen Datenbank gespeichert ist als Geschlecht, und Alter und beide Datenbanken unabhängig voneinander anonymisiert werden, d. h. für die erste Datenbank gibt es nur den Quasi-Identifikator PLZ und für die zweite Datenbank nur die Quasi-Identifikatoren Alter und Geschlecht, kann es wie in Tabelle 4.1 dargestellt nach der Integration zu einer Verletzung der K-Anonymität kommen (der Zeilenindex in den Tabellen ist der gemeinsame Identifikator). Eine naheliegende Vorgehensweise ist nun, diese integrierte Datenbank erneut zu anonymisieren. Diese Vorgehensweise wirft aber zwei Probleme auf. Zum einen werden die Daten hierbei in vielen Fällen mehr als nötig generalisiert, zum anderen gibt es wieder mindestens eine Instanz, die nicht anonyme Daten bearbeitet.

Postleitzahl
819**
817**
819**
826**
817**
826**
824**
824**

(a) P_1: 2-anonym

Alter	Geschlecht
20-60	männlich
20-60	weiblich
≥ 61	männlich
≥ 61	männlich
20-60	weiblich
20-60	männlich
≤ 19	weiblich
≤ 19	weiblich

(b) P_2: 2-anonym

Alter	Geschlecht	Postleitzahl
20-60	männlich	819**
20-60	weiblich	817**
≥ 61	männlich	819**
≥ 61	männlich	826**
20-60	weiblich	817**
20-60	männlich	826**
≤ 19	weiblich	824**
≤ 19	weiblich	824**

(c) Integration, nicht 2-anonym

Abbildung 4.1: Vertikale Integration von zwei Datenbeständen

Eine zweite, mögliche Lösung besteht darin, dass eine Partei lokal ihren eigenen Datenbestand anonymisiert und Informationen bzgl. der entstehenden Äquivalenzklassen (z. B. durch Austauschen der Tupel-Identifikatoren) an die nächste Partei weitergibt. Diese muss nun ihre eigenen Daten so weit generalisieren, dass die integrierte Version das Anonymitätskriterium beibehält. Im Falle von z. B. k-Anonymität muss hierbei sichergestellt werden, dass keine Klassen entstehen, die

weniger als k-Elemente enthalten. Diese (neuen) Äquivalenzklassen werden an die nächste Partei weitergeben, die genauso verfährt. Am Ende kann dann der so generalisierte Datensatz veröffentlicht werden. Dieses Vorgehen hat allerdings einen hohen Informationsverlust zur Folge (für eine experimentelle Evaluation siehe wieder Abschnitt 4.3). Zum anderen werden hierbei Informationen in Form des TIDs, der an die Teilnehmer versendet wird, über den Datensatz ausgetauscht.

Die dritte Herangehensweise an das Problem der verteilten Anonymisierung, diese wird *virtuell anonymisiert* genannt, ist im Allgemeinen mit einem höheren Berechnungsaufwand verbunden, weißt aber bei korrekter Spezifikation und Implementierung, im Gegensatz zu den ersten beiden o. g. Herangehensweisen, keine ihrer Probleme auf. Hierbei wird ein sicheres Protokoll zum Anonymisieren verwendet, welches den Teilnehmern möglichst nur das anonymisierte Ergebnis zur Kenntnis bringt, d. h., es gibt keinen/nur geringen zusätzlichen Informationsgewinn bei der Ausführung des Protokolls. Das anonyme Endergebnis wird hierbei nicht als zusätzlicher Informationsgewinn betrachtet. Die Idee der sicheren Mehrparteienberechnung (SMC) bei l Parteien, jeweils mit deren Input x_i, mit $1 \leq i \leq l$, besteht darin, dass die Parteien eine Funktion $f(x_1, x_2, ..., x_l)$ berechnen und sie jeweils nur ihren eigenen Input und das Ergebnis der Funktion kennen.

Die Konstruktion eines solchen sicheren Protokolls, zur verteilten Berechnung einer Anonymisierung von verteilten Datenbeständen, ist der Fokus in diesem Kapitel.

4.1.3 Angreifermodell

In [90] definiert Goldreich drei verschiedene Szenarien um solch sichere Mehrparteienberechnungen durchzuführen. Das Szenario mit einer vertrauenswürdigen dritten Partei wurde im obigen Abschnitt kurz angesprochen und soll hier nicht weiter behandelt werden. Das *semi-honest* und *malicious* Szenario hingegen ist hier von Bedeutung. Besonders das semi-honest Sicherheitsmodell soll im Folgenden die Grundlage für die vorgestellten Protokolle sein.

4.1.3.1 Semi-Honest

Das halb-ehrliche oder semi-honest Modell wird oft auch als ehrlich-aber-neugierig (honest-but-curious) bezeichnet. Bei diesem Modell halten sich alle Teilnehmer an das spezifizierte Protokoll, versuchen aber aus den Zwischenergebnissen und ihrem eigenen Input zu dem Protokoll, zusätzliche Informationen zu extrahieren. Die zugrunde liegende Idee hierbei geht davon aus, dass ein Angreifer vielleicht nicht in den Programmablauf eingreift, aber die Ein- und Ausgabe des Programms aufzeichnet und versucht daraus Rückschlüsse auf Eingabewerte anderer Parteien zu ziehen. Eine formalere Definition ist in [90] zu finden. In diesem Modell werden nur Angreifer betrachtet, die an der Berechnung teilnehmen (Insider). Um auch externe Angreifer abzuwehren, wird im Folgenden davon ausgegangen, dass sichere Kommunikationskanäle vorhanden sind (d. h. mittels verschlüsselter und integritätsgeschützter Verbindungen zwischen authentisierten Teilnehmern, realisiert z. B. mit SSL/TLS).

4.1.3.2 Malicious

Im malicious oder unehrlichem Modell gelten keine Einschränkungen wie im semihonest Modell. Bei diesem Modell kann sich jeder Teilnehmer verhalten wie er

will. Jeder kann auf beliebige Weise den Protokollablauf stören oder versuchen mittels falscher Eingaben, zusätzliches Wissen über Eingaben von anderen Parteien zu erlangen, welches nicht sowieso aus dem Ergebnis der Berechnung herleitbar wäre. Da dieses Szenario in dem hier untersuchten Umfeld, selten bis gar nicht, auftreten wird und im Zweifelsfall z. B. durch Verträge oder Gesetze sanktioniert werden kann, wird der Fokus auf das semi-honest Modell gelegt.

4.1.4 Kommutative Verschlüsselung

Ein wichtiges, kryptographisches Hilfsmittel in einem der hier vorgestellten Protokolle stellt die kommutative Verschlüsselung dar. Die Idee der kommutativen Verschlüsselung ist es ordnungsunabhängig zu sein, d. h. die Reihenfolge aufeinanderfolgender Verschlüsselungen und Entschlüsselungen, mit unterschiedlichen Schlüsseln, ist irrelevant. Formaler ausgedrückt: Sei $E_{k_e}(x)$ eine Verschlüsselungsfunktion und $D_{k_d}(x)$ die dazugehörige Entschlüsselungsfunktion, sodass gilt $D_{k_d}(E_{k_e}(x)) = x$. Damit muss bei einem kommutativen Verfahren gelten, dass für alle Paare von Schlüsseln (k_{ei}, k_{di}) und beliebigen Permutationen

$$E_{k_{e1}}(E_{k_{e2}}(x)) = E_{k_{e2}}(E_{k_{e1}}(x)) \tag{4.1}$$

sowie

$$D_{k_{d1}}(D_{k_{d2}}(x)) = D_{k_{d2}}(D_{k_{d1}}(x)) \tag{4.2}$$

gilt.

In dem ersten hier vorgestellten Protokoll wird eine kommutative und deterministische Verschlüsselung, d. h., dass ein gleicher Klartext immer in denselben Chiffretext transformiert wird, benötigt.

4.1.5 Additiv homomorphe Verschlüsselung

Bei additiv homomorphen Verschlüsselungsverfahren kann eine Rechenoperation auf verschlüsselte Klartexte angewendet werden, die als Ergebnis die Addition auf den Klartext ergibt. Damit ist es möglich, dass im verschlüsselten Raum Additionen berechnet werden können, die der Addition im Klartextraum entsprechen. Sei $E_h(x)$ eine Verschlüsselungsfunktion und $D_h(x)$ die dazugehörige Entschlüsselungsfunktion, dann muss gelten dass $D_h(E_h(x)) = x$ ist. E_h ist nun additiv homomorph, falls es eine mathematische Operation $+_g$ gibt, sodass $E_h(x_1) +_g E_h(x_2)$ eine gültige Verschlüsselung von $x_1 + x_2$ liefert.

Ein additiv homomorphes Verschlüsselungsverfahren wird im dritten Protokoll eingesetzt. Für dieses Verschlüsselungsverfahren wird zusätzlich, bei dem hier vorgestellten Protokoll, die rechnerische Ununterscheidbarkeit (IND-CPA) gefordert (für eine Definition siehe [91]). Dies bedeutet, vereinfacht ausgedrückt, dass derselbe Klartext bei jeder Verschlüsselung in einem anderen Chiffretext resultiert.

4.1.6 Anforderungen

Ein für die Domäne geeignetes Protokoll zur Anonymisierung von verteilten Daten sollte beliebig (horizontal, vertikal oder hybrid) verteilte Daten anonymisieren können. Da es in der medizinischen Domäne üblich ist, die Daten mittels Generalisierung und Unterdrückung (siehe Abschnitt 2.3) zu anonymisieren, muss ein Protokoll für den verteilten Fall dies unterstützen. Um außerdem an die spezifischen Anforderungen beim Datenschutz angepasst werden zu können, sollte ein

Protokoll, welches verteilte Daten anonymisiert, verschiedenste Anonymisierungskriterien unterstützen. Da zudem oft unterschiedliche Anforderungen an die Qualität und Eigenschaften der anonymisierten Daten gestellt werden, sollten auch unterschiedliche Recodierungsverfahren (z. B. lokal recoding, global recoding usw., siehe Abschnitt 2.5) unterstützt werden. Da auch gerne full-domain Generalisierung als Generalisierungsmethode genutzt wird und diese Art der Recodierung optimale Lösungen hinsichtlich des Informationsverlustes erlaubt (siehe Abschnitt 2.3), sollte ein Protokoll dies unterstützen. Wünschenswert wäre zudem, dass beliebige Anonymisierungsalgorithmen für den zentralisierten Fall (d. h. Algorithmen für den Fall, dass die Daten nicht verteilt vorliegen) genutzt werden könnten. Außerdem soll sowohl die benötigte Laufzeit des Protokolls als auch die zu übertragende Datenmenge praktikabel sein (d. h. z. B. Laufzeit im Minutenbereich inklusive der Zeit für die Übertragung der Datenmenge).

4.2 Existierende Algorithmen zur Anonymisierung verteilter Datenbestände

Neben der Unterteilung des Forschungsfeldes, nach der Verteilung der Daten in horizontale und vertikale Verteilung, kann man es auch, wie oben angesprochen, anhand der für die Lösung verwendeten Architektur unterteilen. In diesem Abschnitt werden die existierenden Lösungen kurz vorgestellt. Der Fokus liegt auf den Lösungen, welche in die gleiche Klasse, wie die hier vorgestellten Protokolle fallen, nämlich virtuelle Anonymisierungslösungen, die das semi-honest Angreifermodell annehmen.

Schon 1986 hat Yao in [92] gezeigt, dass jede (zentrale) Berechnung auch als Mehrparteienberechnung durchgeführt werden kann. Der Nachteil dieser Lösung, die mittels (simulierten) kombinatorischen Schaltkreisen (combinatorical circuits) arbeitet, ist der große notwendige Overhead. Aufgrund dieser Beobachtung wurden einige, speziell für die Anonymisierung von verteilten Daten angepasste, Lösungen entwickelt.

Die meisten entwickelten Lösungen sind speziell für einen bestimmten Anonymisierungsalgorithmus entwickelt worden. Ein Ansatz der Algorithmen unabhängig ist wurde von Jiang et al. [93] beschrieben. Er löst das Problem der k-Anonymisierung, bei vertikal verteilten Daten, für zwei Parteien. Die Idee ist, dass man durch die sichere Berechnung des Durchschnitts zweier Mengen, den Test auf k-Anonymität realisieren kann. Vorbedingung ist, dass es einen globalen, eindeutigen Identifikator gibt, welcher für den Join zwischen den beiden Parteien benutzt werden kann. Beide Parteien besitzen zudem jeweils ihren eigenen Schlüssel für die genutzte kommutative Verschlüsselung. Das Protokoll startet damit, dass beide Parteien diesen gemeinsamen Identifikator verschlüsseln. Sie tauschen die verschlüsselten Identifikatoren aus und verschlüsseln diese erneut mit ihrem Schlüssel. Jede der beiden Parteien besitzt nun eine zweimal kommutativ verschlüsselte Repräsentation der Tupel-Identifikatoren (d. h. gleiche Tupel-Identifikatoren sind bei beiden Parteien identisch). Nun anonymisieren die beiden Parteien iterativ ihre eigenen Daten und bilden Gruppen aus den Tupel-Identifikatoren, die in eine Äquivalenzklasse fallen. Als nächsten Schritt bildet jede Partei die Frequenzmengen für gleiche Werte ihrer Quasi-Identifikatoren (Äquiva-

lenzklassen), welche jeweils aus den globalen, verschlüsselten Identifikatoren bestehen. Diese Frequenzmengen können nun miteinander geschnitten werden. Ein Zustand ist global k-anonym, wenn die Anzahl keiner Schnittmenge kleiner als k ist. Bezogen auf die Tabelle 4.1 würde es folgende zwei Frequenzmengen geben, hierbei stellen die Indexpositionen der Tupel jeweils ihren Tupel-Identifikator dar: Partei A $\{\{1,3\},\{2,5\},\{4,6\},\{7,8\}\}$ und Partei B $\{\{1,6\},\{2,5\},\{3,4\},\{7,8\}\}$. Die integrierte Tabelle ist also nicht k-anonym, da zum Beispiel $|\{1,3\} \cap \{1,6\}| = 1 < k$ ist. Für diese Berechnung der Größen der Schnittmengen schlagen die Autoren ein neues, sicheres Durchschnittsprotokoll vor. Dafür wird ein probabilistisches, additiv homomorphes Verschlüsselungsverfahren verwendet. Damit kann nun, wie im vertikal verteilten Fall, ein bestimmter Zustand auf k-Anonymität geprüft werden. Die Autoren nutzen für die Anonymisierung den Datafly Algorithmus, welcher keine optimale Lösung berechnet, dafür aber mit sehr wenigen Prüfungen „greedy“ einen k-anonymen Zustand finden kann.

Ein anderer Ansatz, welcher das Problem der Anonymisierung bei vertikal verteilten Daten löst, ist in [94, 95] beschrieben. Es beruht auf dem Ansatz der *top-down specialization* [82]. Hierbei wird vom maximal generalisierten Datensatz ausgegangen und iterativ immer mehr spezialisiert, solange bis das Anonymitätskriterium verletzt wird. Bei jeder Spezialisierung wird „greedy“ die Generalisierung mit dem geringsten „Score“ ausgewählt. Der Score basiert auf der Entropie und der Anzahl der verschiedenen Attribute; dies ist eine spezielle Art zur Bestimmung des Informationsverlustes. Es wird darauf aufbauend ein zwei Parteien Protokoll beschrieben, welches sich aber auf n Parteien verallgemeinern lässt. Es wird auch hierbei mit der maximalen Generalisierung angefangen. Diese und weniger generalisierte, aber anonyme, Zwischenergebnisse werden an alle Parteien verteilt, sodass am Ende eine Lösung, unter Einhaltung des Datenschutzes, generiert werden kann. Hierbei wird in Kauf genommen, dass Zwischenergebnisse den Teilnehmern bekannt werden, die aber nur direkte k-anonyme Nachfolger der optimalen Lösung sind. Außerdem erfahren die Teilnehmer den „Score“, hierbei schlagen die Autoren vor, dass ein sicheres Protokoll für die gemeinsame Maximumberechnung verwendet werden könnte. Die Rationale ist, dass diese Zwischenergebnisse auch aus dem veröffentlichten Ergebnis berechnet werden können und somit die beiden Parteien keine zusätzliche Information bei der Protokollausführung erhalten. Bei dem vorgestellten Protokoll wird von einem semi-honest Angreifer ausgegangen. In einem Folgepapier [96] wird der Algorithmus erweitert, um auch in einem abgeschwächten malicious Szenario zu funktionieren. Abgeschwächt, da angenommen wird, dass die bösartige Partei zwar falsche Werte während des Protokollablaufes angibt, ihren Eingabedatensatz aber nicht verändert. Die Begründung der Autoren, dass diese abgeschwächte Form eine valide Annahme ist, besteht darin, dass es schwierig ist, eine bösartige Partei am Manipulieren ihrer Eingabedaten zu hindern, da die Daten geheim und daher nicht überprüfbar seien. Ihre Hauptidee, um den Algorithmus im malicious Modell sicher zu machen, ist die Anwendung von spieltheoretischen Konzepten. Die Grundlage ist, dass die bösartige Partei versucht ihre eigenen Daten so wenig wie möglich zu spezialisieren und gleichzeitig versucht die anderen Parteien dazu zu bringen, ihre Daten so weit wie möglich zu spezialisieren. Dabei kann sich ein globales Optimum erreichen lassen, wo die beiden Ziele der Parteien im Gleichgewicht sind. Darauf aufbauend wird ein Protokoll entworfen, welches, wenn sich die Parteien daran halten,

das Optimum erreicht und bei einer Abweichung einer Partei, die anderen Parteien nicht zu einer Spezialisierung ihrer Daten verleitet. Die Idee hierbei ist, dass jede Partei über den Beitrag der anderen Parteien Buch führt und anhand dieser Buchführung bei jeder Runde des Protokolls entscheidet, ob sie weiterhin an dem Protokoll teilnimmt.

In [85] erweitern die Autoren ihr Protokoll auf horizontal verteilte medizinische Daten. Der untersuchte Anwendungsfall betrifft den Hong Kong Blutspendedienst. Das gespendete Blut wird an verschiedene öffentliche Krankenhäuser verteilt, die Daten über die Patienten sammeln, die eine Bluttransfusion erhalten haben. Eine Auswahl aus diesen Daten (z. B. Art der Operation, Namen der Ärzte, Grund für die Transfusion) müssen in periodischen Abständen an den Blutspendedienst übermittelt werden. Die zu übermittelnden Daten sollen nun anonymisiert werden. Bei der vorgestellten Lösung wird wieder ein halb-ehrliches Angreifermodell zugrunde gelegt. Es basiert wiederum auf der *top-down specialization* [82]. Bei diesem Protokoll wird als Erstes ein Protokollleiter bestimmt, welcher die Synchronisation übernimmt. Dieser Leiter sammelt nun in jeder Runde die Größen der Äquivalenzklassen aller Teilnehmer, um die validen Kandidaten zu bestimmen. Dabei wird vorgeschlagen, das in [97] beschriebene, sichere Summenprotokoll zu verwenden. Die restlichen Protokollschritte folgen demselben Ablauf wie im vertikalen Fall.

Eine weitere Möglichkeit horizontal verteilte Daten, unter Annahme eines halbehrlichen Angreifermodells, zu anonymisieren ist in [89, 98] und ihrem technischen Report [99] beschrieben. Sie schlagen ein Verfahren vor, mit welchem Daten, die über n Parteien verteilt sind, anonymisiert werden können. Die Autoren nutzen dazu den Mondrian [100] Algorithmus und spezielle multi-party Primitive um die Berechnung sicher zu realisieren. Die benötigten Primitive bei einer Ausführung von Mondrian, im zentralen Fall, sind hierbei die Berechnung der Summe, des Minimums/Maximums und des Medians. Für diese Primitive werden in der Arbeit nun sichere Varianten über n Teilnehmer vorgeschlagen. Damit kann nun der Algorithmus, wie im zentralen Fall, ausgeführt werden. Hierbei ist anzumerken, dass Mondrian versucht, durch Auftrennen des mehrdimensionalen Raumes, anhand von Medianen Gruppen zu bilden, welche mindestens die Größe k haben. Als erster Schritt wird hierbei das Attribut bestimmt, welches den größten Wertebereich besitzt. Dazu wird das sichere min/max Protokoll benutzt. Danach wird der Median der Werte bestimmt, mittels des sicheren Median Berechnungsprotokolls. Anhand dieses Medians wird nun der Datensatz aufgeteilt. Die Größen der jeweiligen Hälften werden mittels des sicheren Summenprotokolls, über alle Parteien hinweg, berechnet und damit wird ggf. eine neue Iteration des Algorithmus gestartet. Der verwendete Algorithmus findet also, wie das zugrunde liegende Mondrianverfahren auch, keine optimale Lösung, nutzt allerdings multidimensional global recoding. Diese Art der Codierung kann Ergebnisse mit weniger Informationsverlust liefern, als der in Kapitel 3 vorgeschlagene und hier eingesetzte Algorithmus, welcher full-domain Generalisierung nutzt und dabei eine optimale Lösung findet [100].

Einen anderen Ansatz verfolgen die Autoren von [101]. Anstatt eine k-anonyme Lösung zu berechnen, definieren sie das Problem so um, dass aus einem verteilten Datenbestand eine k-anonyme Teilmenge extrahiert werden soll. Die Hauptidee hierbei ist die Verschlüsselung der sensiblen Werte mit einem Schlüssel, welcher aus den Quasi-Identifikatoren selbst abgeleitet ist und nur berechnet werden kann, wenn

das k-Anonymitätskriterium erfüllt ist (d. h. es mindestens k Zeilen gibt, welche bzgl. der Quasi-Identifikatoren gleich sind). Die Annahme ist auch hier die horizontale Verteilung der Daten mit einem halb-ehrlichen Angreifermodell. Der größte Unterschied zu den anderen Ansätzen ist hierbei, dass nur die sensiblen Attribute geschützt werden, die Quasi-Identifikatoren werden öffentlich allen Teilnehmern an dem Protokoll bekannt. Das vorgestellte Protokoll nutzt das (n,k)-secret sharing von Shamir [102], hierbei ist k die Anzahl der benötigen Teile, um das Geheimnis zu berechnen, darstellt und mit dem k-Anonymitäts k übereinstimmt, n ist die Gesamtmenge von Teilen. Nachdem die Quasi-Identifikatoren und die verschlüsselten, sensiblen Attribute an eine zentrale Stelle gesendet wurden, kann diese die sensitiven Attribute entschlüsseln, wenn es mindestens k Quasi-Identifikatoren gibt die übereinstimmen, da der Schlüssel aus dem Hashwert der Quasi-Identifikatoren berechnet wurde, welcher zuvor mit einem Teil des geteilten Geheimnisses „verschlüsselt“ wurde. Als Zweites präsentieren die Autoren einen verteilten Algorithmus für die k-Anonymisierung mittels Unterdrückung von Einträgen, basierend auf dem Algorithmus von [77]. Bei diesem Protokoll werden neben dem k-anonymen Endergebnis auch die Entfernungen zwischen allen Paaren von Zeilen bekannt, hierbei ist die Entfernung zwischen zwei Zeilen definiert als die Anzahl der Quasi-Identifikatoren, die sich jeweils unterscheiden. Die, für die sichere Implementierung, des in [77] beschrieben Algorithmus, benötigten kryptographischen Verfahren sind zum einen eine multiplikativ homomorphe Verschlüsselung und zum anderen ein Schwellwert Verschlüsselungsverfahren. Da dieses Verfahren keine Generalisierung für die Anonymisierung unterstützt, wird es hier nicht weiter betrachtet.

Tassa et al. [103] stellen eine verteilte Implementierung des sequenziellen Clustering Algorithmus [83] vor. Dieser Ansatz benutzt zwei sichere Primitive. Zum einen ein sicheres Summenprotokoll [97] und ein eigenes, sicheres Verfahren das logische „UND“ zu implementieren. Um die Anzahl der Elemente und den entstandenen Informationsverlust in einem Cluster zu berechnen, wird das sichere Summenprotokoll verwendet. Um das gefundene Cluster zu generalisieren, wird der sichere logische „UND“ Operator verwendet.

Die neuen, hier vorgestellten Protokolle basieren auf Ideen aus [93]. Wie bei dieser Arbeit werden kommutative und additiv homomorphe Verschlüsselungen eingesetzt, um das Problem der Anonymisierung von verteilt vorliegenden Daten zu lösen. In Abschnitt 4.7.3 werden die hier vorgestellten Arbeiten mit den eigenen Ansätzen verglichen und diskutiert.

4.3 Informationsverlust bei Basisverfahren

In den obigen Abschnitten wurde die These aufgestellt, dass ein virtuelles Verfahren gegenüber einem „anonymisiere-und-integriere“ Ansatz, einen geringeren Informationsverlust haben kann. In diesem Abschnitt wird deshalb der Informationsverlust zwischen zwei Basisvorgehensweisen für den horizontalen, vertikalen und einem zentralen Fall (dieser hat den gleichen Informationsverlust wie die später vorgestellten Protokolle, da diese den zentralen Fall nachahmen) verglichen. Hierbei wird ein optimaler Algorithmus verwendet (siehe Kapitel 3) mit den Parametern $k = 5$ für k-Anonymität und $c = 4, l = 3$ für rekursive-(c,l)-Diversität. Die maximal erlaubte Unterdrückung beträgt 3%. Bei dem Vergleich wird t-Closeness und δ-Präsenz aus-

geschlossen, da diese beiden Verfahren schwieriger im anonymisiere-und-integriere Ansatz zu implementieren sind. Es wird sowohl der horizontale als auch der vertikale Fall verglichen. Im horizontalen Fall werden einfach alle Teildatensätze nach den beiden Kriterien anonymisiert und danach integriert. Dieses Ergebnis wird verglichen mit dem Fall, dass die Daten vorher integriert und dann anonymisiert werden. Im vertikalen Fall wird ein Teildatensatz anonymisiert und die anderen Teildatensätze werden nun so weit generalisiert, dass die integrierte Version die ursprünglichen Klassen beibehält. Auch hier wird mit dem Fall verglichen, als ob die Daten vorher integriert worden wären. Diese beiden „anonymisiere-und-integriere“ Varianten stellen den einfachsten Fall dar, im verteilten Fall ohne zentrale, vertrauenswürdige dritte Partei, die Daten unter Wahrung des Datenschutzes zu integrieren und können somit als Basis für weitere Vergleiche genutzt werden. Die Ergebnisse sind in Tabelle 4.1 dargestellt.

	k-Anonymität				ℓ-Diversität			
Dataset	2P - vertikal	3P - vertikal	2P - horizontal	3P - horizontal	2P - vertikal	3P - vertikal	2P - horizontal	3P - horizontal
ADULT	50	45	92	97	76	65	70	68
CUP	15	13	43	32	87	70	65	55
FARS	62	60	89	98	79	79	89	85
ATUS	40	39	79	85	64	65	82	92
IHIS	25	22	67	56	42	35	69	63

Tabelle 4.1: Informationsverlust eines global optimalen Algorithmus, im Vergleich zu Basisverfahren, für zwei und drei Parteien, im horizontalen und vertikalen Fall, in [%].

Die Tabelle zeigt den Informationsverlust des zentralen Falles (und somit den gleichen Informationsverlust, der entstehen würde, wenn eines der neuen, hier vorgestellten, Protokolle ausgeführt werden würde), im Vergleich zu den Basismethoden. Der Informationsverlust wurde mittels der non-uniform Entropie Metrik gemessen (siehe Abschnitt 2.6). Wie man aus der Tabelle entnehmen kann, hat der zentrale (und somit auch der hier vorgestellte virtuelle) Ansatz durchgehend einen höheren Informationsgehalt der anonymisierten Daten. Im Falle von k-Anonymität und dem CUP Datensatz, mit drei Parteien im vertikalen Szenario, hat, zum Beispiel, der zentrale Fall einen um 87 % geringeren Informationsverlust als der „anonymisiere-und-integriere“ Ansatz. Aus der Tabelle lassen sich zwei Trends herauslesen. Zum einen, dass der Informationsgewinn, in 85 % der Fälle, im vertikalen Szenario größer ist als im horizontalen. Zum anderen, dass der Informationsgewinn, in 75 % der Fälle, mit drei Parteien größer ist im Vergleich mit zwei Parteien. Das lässt vermuten, dass die Datenqualität, bei den Basisverfahren, mit zunehmender Anzahl von Teilnehmern abnimmt. Eine interessante Beobachtung hierbei ist, dass im horizontalen Szenario das Basisverfahren lokale Recodierung implementiert, da jede Partei

unterschiedliche Generalisierungen auf ihre Teildaten anwenden kann.

4.4 Protokolle für die sichere Anonymisierung bei verteilten Daten

Im Folgenden werden Protokolle zur Berechnung einer anonymisierten, integrierten Datenmenge vorgestellt. Die Daten können vertikal oder horizontal verteilt vorliegen und mittels einer Vielzahl von Anonymisierungsalgorithmen anonymisiert werden. Auch die Anwendung von verschiedenen Anonymisierungskriterien ist möglich, bspw. k-Anonymität, ℓ-Diversität und δ-Präsenz.

4.4.1 Vorbedingungen und Annahmen

Im Weiteren wird von n Parteien $P = \{P_1, ..., P_n\}$ ausgegangen. Um die Protokolle besser beschreiben zu können, gehen wir davon aus, dass sie einen geschlossenen Kreis formen, d. h., der rechte Nachbar von P_i ist definiert als $p_{1+(i \bmod n)}$. Im Allgemeinen kann die Vernetzungsinfrastruktur beliebig gestaltet sein, solange die Nachbarparteien miteinander kommunizieren können. Die Daten der jeweiligen Partei werden mit d_i bezeichnet. Eine der Parteien agiert als Master, welcher, je nach konkretem Protokoll, unterschiedliche Aufgaben übernimmt; es wird angenommen, dass diese Partei P_1 ist. Im ersten Protokoll erhält z. B. diese Partei die integrierten Daten und anonymisiert diese (siehe Abschnitt 4.4.3). Ein Beispiel dieses Aufbaus mit drei Parteien und ihren Daten ist in Abbildung 4.2 dargestellt.

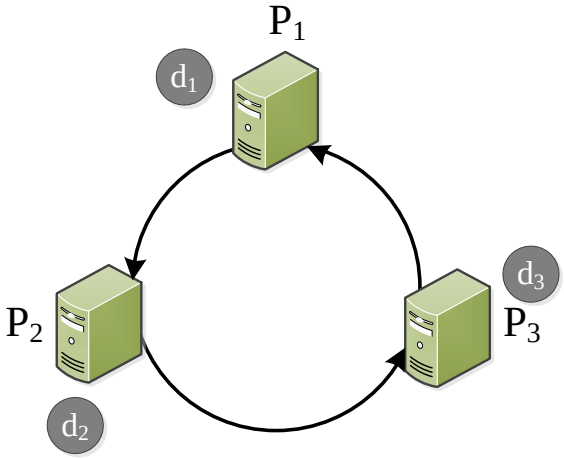


Abbildung 4.2: Drei Parteien P_1, P_2, P_3, angeordnet als geschlossener Kreis mit ihren Daten d_1, d_2, d_3

Generell ist es üblich, bei der Anonymisierung eine relationale Tabelle A_i als Schema zu verwenden. Die Spalten lassen sich hierbei in vier verschiedene Kategorien einordnen (siehe Abschnitt 2.1), nämlich direkt-identifizierende I_i, quasi-identifizierende Q_i, sensible S_i und nicht-sensible M_i. Jede Partei P_i klassifiziert für sich alleine oder in Absprache mit den anderen Parteien (insbesondere bei Quasi-Identifikatoren) ihre Attribute in direkt, quasi, sensibel und nicht-sensibel. Es wird angenommen, dass die direkt-identifizierenden Attribute vor dem Protokolllauf, von jeder Partei selbst, entfernt werden. Im Folgenden werden deshalb nur die letzten

drei Genannten betrachtet und einfachheitshalber die nicht-sensiblen Attribute genauso wie die sensiblen behandelt. Damit müssen bei den folgenden Protokollen nur mehr zwei Attributklassen (Quasi-Identifikatoren und sensible Attribute) unterschieden werden. Da die hier vorgestellten Protokolle Generalisierung und Unterdrückung für die Anonymisierung benutzen, wird davon ausgegangen, dass die benötigten Generalisierungsvorschriften allen Parteien bekannt sind. Diese Hierarchien müssen vor der Ausführung der Protokolle erstellt worden sein. Sollte die zu generalisierende Domäne kontinuierlich sein, können die Hierarchien auch Abbildungsfunktionen enthalten, die auf dem lokalen Datensatz ausgeführt werden können. Die Generalisierungsregeln und Funktionen müssen so beschaffen sein, dass gleiche Eingabewerte unabhängig konsistente Ausgabewerte liefern, für alle Teildatensätze der Teilnehmer. Ein Beispiel für eine Generalisierungsfunktion für kontinuierliche Werte könnte sein, dass ein Gleiszahlwert inkrementell bei jeder Generalisierung um eine Stelle verkürzt wird. Um die Beispiele möglichst verständlich zu halten, wird im Folgenden von diskreten Domänen ausgegangen. Außerdem wurden die Hierarchien materialisiert. Außerdem wird davon ausgegangen, dass alle Hierarchien allen Teilnehmern bekannt sind. Außerdem extrahiert jede Partei aus den vorhandenen Generalisierungshierarchien nur den Teil, welchen sie braucht, um ihre Daten generalisieren zu können. Die Integration von verteilten Daten wird in der Praxis mit Sicherheit zu Problemen (z. B. Inkonsistenzen) führen. Es wird hier jedoch angenommen, dass die Integrationsprobleme vor der eigentlichen Protokollausführung gelöst wurden und keine Konsistenzprobleme verbleiben. Eine weitere Annahme im vertikal verteilten Szenario ist, dass es gemeinsame Tupel-Identifikatoren (TID) gibt, welche die Zusammengehörigkeit zwischen den Teiltupeln der einzelnen Teilnehmer repräsentieren. Im horizontalen Fall wird weiterhin angenommen, dass sich die Teildatensätze nicht überlappen. Wie in vielen anderen Ansätzen auch, wird hier für die Protokolle das semi-honest Szenario angenommen und gefordert, dass die Kommunikation zwischen den Teilnehmern mittels sicherer Kanäle stattfindet (z. B. TLS Verbindungen, verschlüsselt, integritätsgeschützt und authentisiert) und damit vor externen Angreifern geschützt ist. Außerdem wird angenommen, dass vor dem Protokolllauf, eine Aushandlung der zu benutzenden deterministischen und im erweiterten Protokoll probabilistischen kryptographischen Algorithmen und deren Parametern (z. B. Schlüssellänge oder Blocklänge) stattfindet. Zuletzt wird angenommen, dass anhand dieser ausgetauschten Parameter, jede Partei zufällig die jeweils benötigten Schlüssel generiert. Welche Schlüssel genau benötigt werden, ist vom Protokoll abhängig, und wird nochmals im jeweiligen Abschnitt beschrieben.

4.4.2 In den Protokollen verwendete Kommunikationspattern

Die vorgestellten Protokolle tauschen, in verschiedenen Schritten, Daten zwischen den Teilnehmern aus. Dazu nutzen die Protokolle zwei unterschiedliche Kommunikationspattern. Protokoll A nutzt ein Round-Robin-Verfahren, wohingegen Protokoll B und C ein Sternschema nutzt.

4.4.2.1 Round-Robin

Protokoll A nutzt zwei verschiedene Pattern. Zum einen ein sequenzielles, hierbei wird ein Subset von Daten einmal im Kreis gesendet. Dieses Pattern wird sequen-

zielles Round-Robin (SRR) genannt. In Abbildung 4.3 ist es dargestellt.

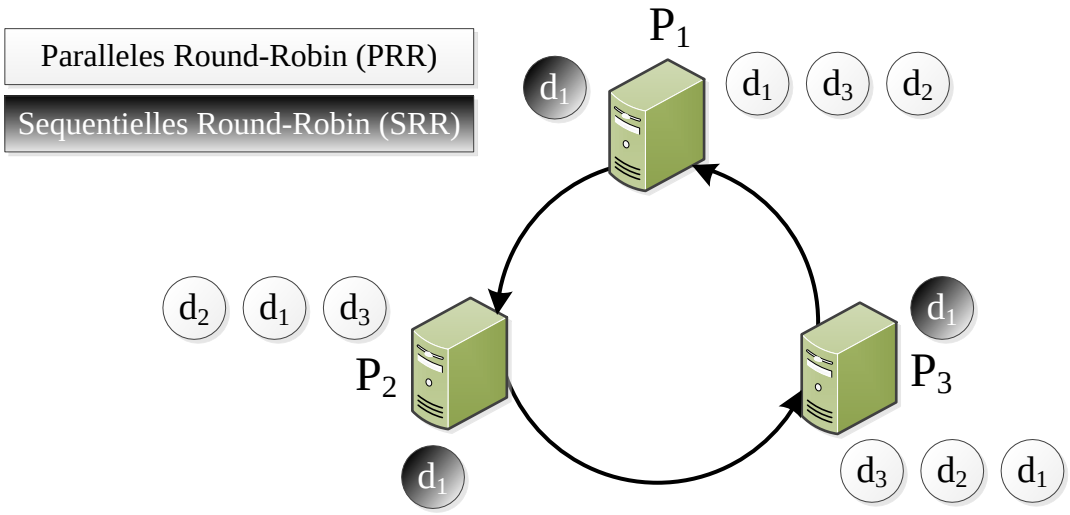


Abbildung 4.3: SRR mit den Daten d_1 und PRR mit den Daten d_1, d_2, d_3

Hier wird die Datenteilmenge d_1 (dunkelgrau hinterlegt) von P_1 nach P_2 und danach zu P_3 gesendet. Bevor die Daten zum nächsten Teilnehmer gesendet werden, müssen die Daten bearbeitet werden. Der Prozess stoppt, wenn der Nachbar der initiierenden Partei die Daten bearbeitet hat. Das zweite Kommunikationspattern wird parallel Round-Robin (PRR) genannt. Hier wird die Kommunikation von allen Teilnehmern gleichzeitig initiiert und im Kreis gesendet. Auch dieser Prozess hält, wenn der Nachbar der jeweils initiierenden Partei, die Daten bearbeitet hat. Wie im Beispiel in Abbildung 4.3 dargestellt, bearbeitet P_3 zuerst die eigenen Daten, erhält und bearbeitet als Nächstes die Daten von P_2 und als Letztes von P_3.

Sowohl SRR als auch PRR kann in beiden Richtungen, d. h. im Uhrzeigersinn (R-xRR) und gegen den Uhrzeigersinn (L-xRR), stattfinden. Diese Unterscheidung ist wichtig, da im Weiteren zwei aufeinanderfolgende Schritte, in unterschiedlichen Richtungen, ausgeführt werden müssen. Im Vergleich, bedingt durch die Parallelität, ist PRR effizienter als SRR. Synchronisationspunkte sind hierbei nur notwendig am Ende einer Phase des Protokolls, hierbei müssen alle Teilnehmer warten, bis alle Nachrichten verarbeitet wurden.

4.4.2.2 Stern

Protokoll B und C nutzen ein einfacheres Kommunikationsschema. Alle Primärparteien senden die Daten an die zentrale Hilfspartei (schwarz). Nach der Anonymisierung sendet die Hilfspartei die Daten wieder an eine Primärpartei (hellgrau). Das Schema ist in Abbildung 4.4 dargestellt.

4.4.3 Protokoll A

Im Folgenden wird ein Basisprotokoll zur Berechnung einer anonymisierten, integrierten Datenmenge vorgestellt. Dieses Protokoll kann sowohl bei vertikal als auch bei horizontal verteilt vorliegenden Daten verwendet werden. Zuerst wird ein Überblick über den Protokollablauf gegeben, gefolgt von einer detaillierten Beschreibung des Protokolls. Hierbei wird auf die Unterschiede bei den beiden Verteilungen ein-

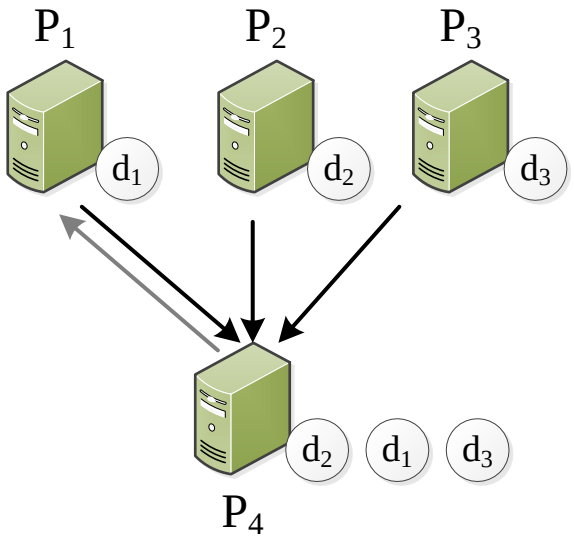


Abbildung 4.4: Kommunikationspattern bei Protokoll B und C

gegangen. Folgend werden zwei Beispiele dargestellt, je eines für den horizontal und vertikal verteilten Fall. Abschließend wird die Komplexität und Sicherheit des Protokolls analysiert.

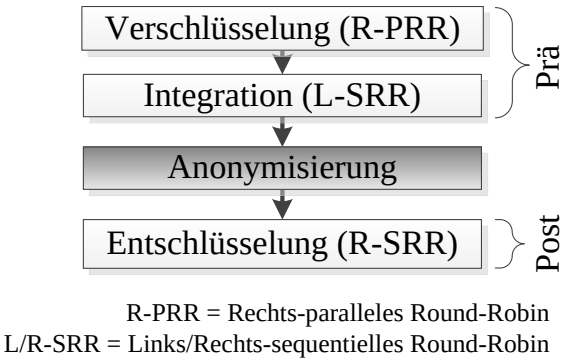


Abbildung 4.5: Die vier Phasen des Protokolls

4.4.3.1 Protokoll

Die Grundidee des Basisprotokolls ist, dass die Quasi-Identifikatoren und sensiblen Attribute so verschlüsselt werden, dass es möglich ist, die verschlüsselte Repräsentation der Daten zu anonymisieren. Darum wird als erster Schritt eine verschlüsselte und integrierte Version der Daten und der Generalisierungshierarchien erstellt. Dieser Schritt wird als Vorverarbeitung bezeichnet, er ist untergliedert in die zwei Schritte Verschlüsselung und Integration. Als zweiter Schritt wird nun ein Anonymisierungsalgorithmus auf diese Version der Daten und Hierarchien angewendet. Dieser Schritt wird als Anonymisierungsschritt bezeichnet. Dieser Schritt ist möglich, da die Daten und Generalisierungshierarchien im vorhergehenden Schritt auf die gleiche Weise mittels eines deterministischen Verfahrens auf Tabellenzellenebene verschlüsselt wurden. Als letzter Schritt wird dieser anonymisierte und verschlüsselte Datensatz nun entschlüsselt, was zu einem integrierten und anonymisierten Gesamtdatensatz führt (Entschlüsselungsschritt). Diese Schritte sind in Abbildung 4.5

dargestellt.

Einen Unterschied zwischen vertikalem und horizontalem Szenario gibt es vor allem in der Integrationsphase. Dieser wird im Abschnitt 4.4.3.1.2 genauer erläutert.

4.4.3.1.1 Verschlüsselung Der Verschlüsselungsschritt läuft bei vertikaler und horizontaler Verteilung der Daten identisch ab. Zuerst werden in einem parallelen Round-Robin-Verfahren, als Richtung wird rechtsherum (R-PRR) angenommen, die Daten und die Hierarchien kommutativ verschlüsselt. Jeder der Teilnehmer generiert zufällig, für alle seine Spalten, einen Schlüssel für das kommutative Verschlüsselungsverfahren. Jede Spalte wird nun, unabhängig von jedem Teilnehmer, mit dem generierten Schlüssel verschlüsselt. Die Generalisierungshierarchien werden auch mit dem jeweiligen, zur Hierarchie gehörenden Spaltenschlüssel, verschlüsselt. Somit sind die Daten jeder Spalte und die dazugehörige Hierarchie mit demselben Schlüssel verschlüsselt. Nach der Verschlüsselung werden nun die Daten und die Generalisierungshierarchien an den rechten Nachbar weitergereicht, der seinerseits die Daten und die Hierarchien mit seinen Schlüsseln verschlüsselt. Dabei muss sichergestellt werden, dass immer der gleiche Schlüssel für gleiche Spalten/Attribute verwendet wird. Diese Phase ist abgeschlossen, wenn jeder Teilnehmer die Daten seines linken Nachbarn hat und diese von jedem Teilnehmer verschlüsselt wurde. Da der Verschlüsselungsmechanismus deterministisch, also gleicher Klartext immer in denselben Ciphertext verschlüsselt wird, und kommutativ ist, wird am Ende des Round-Robin-Verfahrens, gleicher Klartext bei verschiedenen Teilnehmern, in gleichem Ciphertext resultieren.

4.4.3.1.2 Integration Während der Integrationsphase werden die Daten der einzelnen Teilnehmer kombiniert. Diese Phase läuft bei horizontal und vertikal verteilten Daten jeweils leicht unterschiedlich ab. Im vertikalen Fall sortiert (z. B. lexikographisch, aufsteigend) jeder Teilnehmer als Erstes die Daten anhand des Tupel-Identifikators. Da dieser kommutativ von jedem Teilnehmer verschlüsselt wurde, besitzt nun jeder Teilnehmer die Daten in der gleichen Reihenfolge. Nach dem Sortieren kann jede Partei den verschlüsselten Tupel-Identifikator entfernen, da für die weiteren Integrationsschritte die gleiche Reihenfolge der Tupel als Identifikator ausreicht. Diese Integrationsschritte werden von der Partei, links neben der Master Partei, welche zufällig gewählt ist und allen Teilnehmern bekannt, initiiert, im Beispiel P_3. Hierbei werden die n-mal verschlüsselten Daten schrittweise integriert, indem sie mittels eines sequenziellen Round-Robin (SRR), in der entgegengesetzten Richtung des Verschlüsselungsschritts, gesendet werden. Im Beispiel ergibt sich hierbei ein L-SRR Kommunikationsmuster. Somit sendet nun, im Beispiel P_3, die n-mal kommutativ verschlüsselten Daten von ihrem rechten Nachbarn P_1 an ihren linken Nachbarn P_2. P_2 integriert die erhaltenen Daten nun mit denen von P_3 und sendet die so integrierten Daten an P_1, welche diese wiederum mit den Daten von P_2 integriert. Im horizontal verteilten Fall besteht die Integration im Zusammenfügen der Tupel der Parteien. Zusätzlich werden die Generalisierungshierarchien von gleichen Attributen integriert, hierbei werden eventuell auftretende Duplikate entfernt. Diese Duplikate können auftreten, wenn zwei Parteien dieselben Datenelemente haben. Dabei werden von beiden Parteien zweimal dieselben Generalisierungsregeln in beiden Generalisierungshierarchien eingefügt. Hierbei werden keine Konflikte auftreten, da angenommen wird, dass diese Regeln global definiert wurden (vgl. Ab-

schnitt 4.4.1). Im vertikal verteilten Fall werden bei dem Integrationsschritt die Spalten der Parteien konkateniert. Hierbei muss die Sortierung erhalten bleiben, da sonst falsche Dateneinträge integriert würden. Die Generalisierungshierarchien müssen in diesem Falle nicht integriert werden, da davon ausgegangen wird, dass die Spalten disjunkt sind und somit jede Partei Generalisierungshierarchien für verschiedene Attribute beiträgt. Dieser Integrationsschritt endet, wenn die Master Partei den verschlüsselten globalen, integrierten Datensatz, zusammen mit den verschlüsselten Generalisierungshierarchien erhalten hat.

4.4.3.1.3 Anonymisierung Da nun die Master Partei die integrierten und verschlüsselten Daten und Generalisierungshierarchien erhalten hat, kann diese die Anonymisierung vornehmen. Hierzu kann ein zentralisierter Algorithmus (z. B. Flash, siehe 3) verwendet werden. Dies ist möglich, da der Generalisierungsprozess nur ein Ersetzen von Werten mit anderen (stärker generalisierten) Werten ist. Der Algorithmus benötigt also beim Generalisieren keine Information über die Werte. Die Information über die (semantische) Generalisierung ist in den verschlüsselten Generalisierungshierarchien enthalten und somit ist es nicht nötig, Klartext Werte zu kennen. Die Generalisierung läuft nun genauso ab, wie im Kapitel 3 beschrieben. Dadurch sind viele aktuelle Algorithmen (z. B. OLA [56], Incognito [79]) und Anonymisierungskriterien (k-Anonymität, ℓ-Diversität usw.) verwendbar. Weitere Details werden in Abschnitt 4.7 dargelegt. Das Ergebnis dieses Schritts ist ein verschlüsselter anonymisierter Datensatz.

4.4.3.1.4 Entschlüsselung Als letzter Schritt in diesem Protokoll werden die anonymisierten, aber verschlüsselten Daten, entschlüsselt. Während dieser Nachbearbeitungsphase (Postprocessing) des Protokolls wird ein sequenzieller round-robin (R-SRR) Kommunikationsschritt durchgeführt. Jede Partei entschlüsselt hierbei die anonymisierten Daten mit ihrem eigenen Schlüssel und reicht die Daten danach an ihren Nachbarn weiter, der wiederum die Daten entschlüsselt. Dieser Prozess wird so lange durchgeführt, bis der linke Nachbar der Master Partei eine komplett entschlüsselte Version des anonymisierten Datensatzes hält.

4.4.3.1.5 Zufälliges Permutieren Ähnlich wie bei anderen Ansätzen (z. B. [98, 103]) werden hier die Datenschutzgarantien der traditionellen, sicheren Mehrparteienberechnung aufgeweicht. Ursprünglicherweise wird keine andere Information außer dem Ergebnis der Berechnung von den Parteien gelernt; in den hier vorgestellten Protokollen können zusätzliche Informationen bekannt werden. Dennoch ist jede zusätzliche Information nur den Parteien bekannt, die beim Protokollaustausch mitwirken, da davon ausgegangen wird (siehe Abschnitt 4.4.1), dass die Kommunikation und die Parteien selbst, vor externen Angreifern geschützt sind. In der grundlegenden Version des Protokolls erhalten und bearbeiten die Parteien nicht-anonymisierte Teildatensätze von anderen Parteien, die von einigen oder allen Parteien verschlüsselt wurden. Da die Daten in dem vorgestellten Protokoll deterministisch auf Zellenebene verschlüsselt werden, können die Teilnehmer statistische Informationen aus den verschlüsselten Daten lernen. Diese Informationen können benutzt werden, um Teildatensätze aus dem integrierten Datensatz herauszulösen, was wiederum potenziell erlaubt, Informationen über die Daten anderer Teilnehmer zu lernen. Um diesen Angriff zu verhindern, werden die Daten während der Verschlüsselungsphase zufällig permutiert, bevor sie an die nächste Partei gesendet

werden. Diese Permutation bewirkt, dass die Reihenfolge Information entfernt wird und somit die Muster schwerer zu erkennen sind. Weiterhin ist es aber möglich Verteilungen der Daten zu berechnen, die wiederum für Frequenzangriffe genutzt werden können. Frequenzangriffe wurden unter anderem auch im Zusammenhang mit ordnungserhaltender Verschlüsselung in [104, 105] untersucht. In Abschnitt 4.4.3.5 werden weitere mögliche Gegenmaßnahmen diskutiert.

Neben der Permutation während der Verschlüsselungsphase werden die Teildatensätze auch während der Integrationsphase, im horizontalen Falle, dadurch geschützt, dass die Daten iterativ integriert werden. Somit ist die Master Partei die einzige Partei, die den kompletten, integrierten, permutierten und verschlüsselten Datensatz erhält. Damit ist sie auch die einzige Partei, die ihre eigenen Daten komplett (also n-mal) verschlüsselt sieht, allerdings sind diese Daten mit den n-2 anderen Teildatensätzen zufällig vermischt, dass es sehr schwer wird, die einzelnen Tupel der Master Partei herauszurechnen.

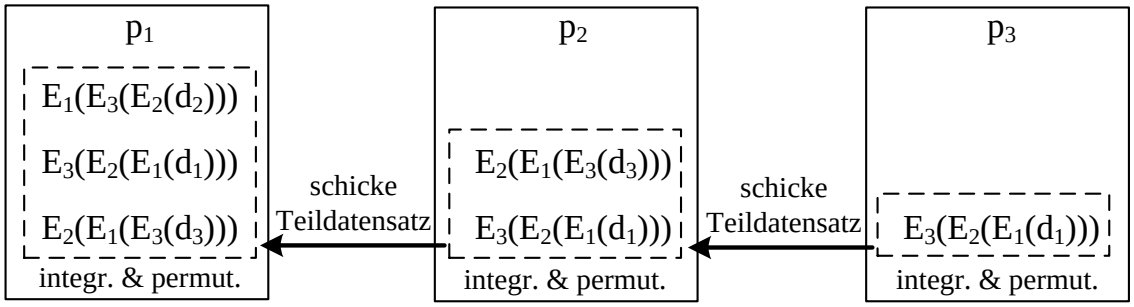


Abbildung 4.6: Beispiel für die Vorgehensweise bei der Integration von horizontal verteilten Daten

Ein Beispiel ist in Abbildung 4.6 dargestellt. Hierbei sendet P_3 die n-mal verschlüsselten Daten von P_1 an die Partei P_2. P_2 integriert und permutiert zufällig die Daten mit den n-mal verschlüsselten Daten von P_3. Diese permutierten und integrierten Daten werden nun von P_2 an P_1 gesendet. Somit sind die Daten von P_1, wenn diese Partei ihre Daten wieder erhält, mit den Daten von P_3 zufällig permutiert. Analog werden die Daten beim Entschlüsseln nach jeder Teilentschlüsselung zufällig permutiert um eine Abbildung der eigenen Teildatensätze, mit der integrierten und anonymisierten Version des gesamt Datensatzes, zu verhindern.

4.4.3.2 Beispiel: vertikal verteilt

In diesem Beispiel wird von drei Parteien P_1, P_2 und P_3 ausgegangen. P_1 ist die Master Partei und nimmt somit auch die Anonymisierung vor. Jede Partei besitzt den Tupel-Identifikator (TID) und ein weiteres Attribut, wie in Abbildung 4.7 zu sehen ist. Somit hat P_1 TID und die Postleitzahl (PLZ) als Attribute, P_2 besitzt den TID und das Alter und P_3 hat neben dem TID das Geschlecht als Attribut. Jede Partei hat für jedes Attribut einen eigenen Schlüssel für die kommutative Verschlüsselung. Um den jeweiligen Schlüssel einfach referenzieren zu können, wird jedem Schlüssel einen Index wie folgt gegeben: TID=1, Geschlecht=2, Alter=3 und Postleitzahl=4. Damit bedeutet $E_{23}(45)$, dass Partei P_2 das Alter 45 mit dem Schlüssel für Attribut 3 verschlüsselt.

TID	PLZ
1	82676
2	81775
3	81925
4	81825
5	81774
6	82669
7	82453
8	82451

(a) P_1

TID	Alter
1	34
2	45
3	66
4	70
5	35
6	21
7	18
8	17

(b) P_2

TID	Geschlecht
1	m
2	w
3	m
4	m
5	w
6	m
7	w
8	w

(c) P_3

Abbildung 4.7: Verteilung der Beispieldaten auf die drei Parteien

Die Hierarchien für die Attribute sind in den Abbildungen 2.10, 2.9 und 2.11 dargestellt. Jede Partei hält die respektive Generalisierungshierarchie. P_1 hält die Hierarchie für die Postleitzahl, P_2 die Hierarchie für das Alter usw. Im Folgenden liegt der Fokus auf Partei P_2, die anderen beiden Parteien verhalten sich analog.

TID	Alter
$E_{21}(1)$	$E_{23}(34)$
$E_{21}(2)$	$E_{23}(45)$
$E_{21}(3)$	$E_{23}(66)$
$E_{21}(4)$	$E_{23}(70)$
$E_{21}(5)$	$E_{23}(35)$
$E_{21}(6)$	$E_{23}(21)$
$E_{21}(7)$	$E_{23}(18)$
$E_{21}(8)$	$E_{23}(17)$

(a) Datensubset

Lvl-0	Lvl-1	Lvl-2
$E_{23}(34)$	$E_{23}(20\text{-}60)$	$E_{23}(*)$
$E_{23}(45)$	$E_{23}(20\text{-}60)$	$E_{23}(*)$
$E_{23}(66)$	$E_{23}(\geq 61)$	$E_{23}(*)$
$E_{23}(70)$	$E_{23}(\geq 61)$	$E_{23}(*)$
$E_{23}(35)$	$E_{23}(20\text{-}60)$	$E_{23}(*)$
$E_{23}(21)$	$E_{23}(20\text{-}60)$	$E_{23}(*)$
$E_{23}(18)$	$E_{23}(\leq 19)$	$E_{23}(*)$
$E_{23}(17)$	$E_{23}(\leq 19)$	$E_{23}(*)$

(b) Hierarchie

Abbildung 4.8: Verschlüsselte Daten von P_2

Als Erstes generiert P_2 zufällige Schlüssel für alle Attribute und verschlüsselt TID und Alter kommutativ. Auch die Generalisierungshierarchie für das Attribut Alter wird auf dieselbe weise verschlüsselt. Das Ergebnis ist in Abbildung 4.8 dargestellt. Als nächstes permutiert P_2 die verschlüsselten Daten zufällig und sendet diese an P_3. Analog erhält P_2 die verschlüsselten und permutierten Daten und Hierarchien von P_1. Sie verschlüsselt diese erneut, permutiert diese und sendet sie an P_3.

Zweitens, wenn die Verschlüsselungsphase abgeschlossen ist, startet der linke Nachbar der Master Partei, in diesem Beispiel P_3, die Integrationsphase. Sie sortiert die verschlüsselten Daten von P_1 anhand des TID Attributes, entfernt dieses und sendet die Daten an zu P_2. P_2 verfährt mit den Daten von P_3 genauso, sortiert diese also anhand des TIDs und entfernt diesen. Danach werden die verschlüsselten Daten von P_1 und P_3 aneinandergefügt. Hierbei wird nicht permutiert, um die Ordnung zu erhalten. Dieser Schritt ist in Abbildung 4.9 skizziert. Als letzten Schritt erhält nun P_1 die Daten und integriert diese wieder, somit hat P_1 nun den integrierten Gesamtdatensatz und alle verschlüsselten Generalisierungshierarchien.

In Abbildung 4.9 ist zu sehen, dass alle Zellen innerhalb einer Spalte in derselben Reihenfolge verschlüsselt wurden, die unterschiedlichen Spalten aber verschiedene Reihenfolgen aufweisen. Da die Verschlüsselung aber kommutativ ist und somit der Reihenfolge im Weiteren keinerlei Bedeutung beigemessen wird, dient dies hier nur der Veranschaulichung.

Geschlecht		Alter		PLZ		TID (von P_1)	
$E_{22}(E_{12}(E_{32}(\text{m})))$	+	$E_{13}(E_{33}(E_{23}(34)))$	+	$E_{34}(E_{24}(E_{14}(82676)))$		$E_{34}(E_{24}(E_{14}(1)))$	← Lexikogr. sortiert
$E_{22}(E_{12}(E_{32}(\text{w})))$		$E_{13}(E_{33}(E_{23}(35)))$		$E_{34}(E_{24}(E_{14}(81774)))$		$E_{34}(E_{24}(E_{14}(5)))$	
$E_{22}(E_{12}(E_{32}(\text{m})))$		$E_{13}(E_{33}(E_{23}(66)))$		$E_{34}(E_{24}(E_{14}(81925)))$		$E_{34}(E_{24}(E_{14}(3)))$	
$E_{22}(E_{12}(E_{32}(\text{m})))$		$E_{13}(E_{33}(E_{23}(21)))$		$E_{34}(E_{24}(E_{14}(82669)))$		$E_{34}(E_{24}(E_{14}(6)))$	
$E_{22}(E_{12}(E_{32}(\text{w})))$		$E_{13}(E_{33}(E_{23}(45)))$		$E_{34}(E_{24}(E_{14}(81775)))$		$E_{34}(E_{24}(E_{14}(2)))$	
$E_{22}(E_{12}(E_{32}(\text{w})))$		$E_{13}(E_{33}(E_{23}(18)))$		$E_{34}(E_{24}(E_{14}(82453)))$		$E_{34}(E_{24}(E_{14}(7)))$	
$E_{22}(E_{12}(E_{32}(\text{m})))$		$E_{13}(E_{33}(E_{23}(70)))$		$E_{34}(E_{24}(E_{14}(81825)))$		$E_{34}(E_{24}(E_{14}(4)))$	
$E_{22}(E_{12}(E_{32}(\text{w})))$		$E_{13}(E_{33}(E_{23}(17)))$		$E_{34}(E_{24}(E_{14}(82451)))$		$E_{34}(E_{24}(E_{14}(8)))$	

Abbildung 4.9: Integrationsschritte zum integrierten Gesamtdatensatz

Drittens kann nun P_1 die integrierten Daten mithilfe der Generalisierungshierarchien anonymisieren. In diesem Beispiel wird von 2-Anonymität ausgegangen, siehe Abbildung 4.10. Während des Anonymisierungsprozesses wird die optimale Lösung von P_1 gefunden, indem die verschlüsselten Werte des Attributes Alter, mit den Werten des Levels 1 der verschlüsselten Generalisierungshierarchie, ersetzt werden. Außerdem werden die Werte für die PLZ mit den Werten des Levels 3 ersetzt. Das Attribut Geschlecht bleibt unverändert.

Als vierter und letzter Schritt wird der anonymisierte und verschlüsselte Datensatz (siehe Abbildung 4.10) entschlüsselt. Dazu wird dieser als Erstes von P_1 entschlüsselt, permutiert und daraufhin an P_2 gesendet. P_2 entschlüsselt wiederum den Datensatz mit ihren Schlüsseln, permutiert ihn und sendet diesen nun an P_3. Als letztes entschlüsselt nun auch P_3 die Daten und hält somit den unverschlüsselten und anonymisierten Datensatz. Dieser anonymisierte Datensatz entspricht nun genau jenem, der entstanden wäre, wenn die Teildatensätze von einer vertrauenswürdigen dritten Partei (vgl. Abschnitt 4.1.2) erzeugt worden wäre.

Geschlecht	Alter	PLZ
$E_{22}(E_{12}(E_{32}(\text{m})))$	$E_{13}(E_{33}(E_{23}(20-60)))$	$E_{34}(E_{24}(E_{14}(82***)))$
$E_{22}(E_{12}(E_{32}(\text{w})))$	$E_{13}(E_{33}(E_{23}(20-60)))$	$E_{34}(E_{24}(E_{14}(81***)))$
$E_{22}(E_{12}(E_{32}(\text{m})))$	$E_{13}(E_{33}(E_{23}(\geq 61)))$	$E_{34}(E_{24}(E_{14}(81***)))$
$E_{22}(E_{12}(E_{32}(\text{m})))$	$E_{13}(E_{33}(E_{23}(\geq 61)))$	$E_{34}(E_{24}(E_{14}(81***)))$
$E_{22}(E_{12}(E_{32}(\text{w})))$	$E_{13}(E_{33}(E_{23}(20-60)))$	$E_{34}(E_{24}(E_{14}(81***)))$
$E_{22}(E_{12}(E_{32}(\text{m})))$	$E_{13}(E_{33}(E_{23}(20-60)))$	$E_{34}(E_{24}(E_{14}(82***)))$
$E_{22}(E_{12}(E_{32}(\text{w})))$	$E_{13}(E_{33}(E_{23}(\leq 19)))$	$E_{34}(E_{24}(E_{14}(82***)))$
$E_{22}(E_{12}(E_{32}(\text{w})))$	$E_{13}(E_{33}(E_{23}(\leq 19)))$	$E_{34}(E_{24}(E_{14}(82***)))$

Abbildung 4.10: 2-anonymisierter und verschlüsselter Datensatz

4.4.3.3 Beispiel: horizontal verteilt

Der horizontale Fall ist dem vertikalen Fall sehr ähnlich. Auch in diesem Beispiel wird von drei Parteien P_1, P_2 und P_3 ausgegangen. P_1 ist wiederum die Master

Partei und nimmt somit auch die Anonymisierung vor. Jede Partei hat nun ein Teildatensatz mit den drei Attributen Geschlecht, Alter und PLZ. Diese Verteilung ist in Abbildung 4.11 dargestellt. P_1 und P_2 haben jeweils drei Tupel, wohingegen P_3 nur zwei Tupel besitzt. Jede Partei hat wieder für jedes Attribut einen eigenen Schlüssel für die kommutative Verschlüsselung. Um den jeweiligen Schlüssel einfach referenzieren zu können, wird auch hier wieder jedem Schlüssel der gleiche Index wie im vertikalen Fall gegeben: Geschlecht=2, Alter=3 und Postleitzahl=4. Damit bedeutet $E_{23}(45)$, dass Partei P_2 das Alter 45 mit dem Schlüssel für Attribut 3 verschlüsselt. Im folgenden Abschnitt wird auf die Partei P_1 fokussiert. Die relevante Teilmenge der Generalisierungshierarchien für P_1 ist in Abbildung 4.12 dargestellt.

Alle Parteien generieren nun zufällig die Schlüssel für alle Spalten und verschlüsseln die Daten und Generalisierungshierarchien. Der resultierende Teildatensatz und die Hierarchie für das Attribut Alter ist in Abbildung 4.13 dargestellt. Die beiden anderen Generalisierungshierarchien (Geschlecht, PLZ) sind analog zu Abbildung 4.13b verschlüsselt.

Geschl.	Alter	PLZ
m	34	82667
w	45	81775
m	66	81925

(a) P_1

Geschl.	Alter	PLZ
m	70	81825
w	35	81774
m	21	82669

(b) P_2

Geschl.	Alter	PLZ
w	18	82453
w	17	82451

(c) P_3

Abbildung 4.11: Verteilung der Daten auf die Parteien im horizontalen Fall

Lvl-0	Lvl-1
m	*
w	*

(a) Geschlecht

Lvl-0	Lvl-1	Lvl-2
34	20-60	*
45	20-60	*
66	≥ 61	*

(b) Alter

Lvl-0	Lvl-1	...	Lvl-5
82667	8192*	...	*
81775	8177*	...	*
81925	8192*	...	*

(c) PLZ

Abbildung 4.12: Hierarchien für das horizontale Subset von P_1

Geschl.	Alter	PLZ
E_{12}(m)	E_{13}(34)	E_{14}(82667)
E_{12}(w)	E_{13}(45)	E_{14}(81775)
E_{12}(m)	E_{13}(66)	E_{14}(81925)

(a) Verschlüsselte Daten

Lvl-0	Lvl-1	Lvl-2
E_{13}(34)	E_{13}(20-60)	E_{13}(*)
E_{13}(45)	E_{13}(20-60)	E_{13}(*)
E_{13}(66)	E_{13}($\geq$61)	E_{13}(*)

(b) Verschlüsselte Gen. Hier. Alter

Abbildung 4.13: Verschlüsselter Teildatensatz von P_1

Die Daten werden nun von P_1 permutiert und zum rechten Nachbarn P_2 gesendet. P_2 verschlüsselt die Daten und Hierarchien mit ihren eigenen Schlüsseln und sendet diese an P_3, welche analog verfährt.

Während der Integrationsphase werden die Daten nun von P_1, welche P_3 im vorherigen Schritt erhalten hatte, permutiert und an P_2 gesendet. P_2 kombiniert die erhaltenen Daten mit den Daten von P_3 und permutiert diese zufällig. Danach werden die kombinierten Daten und Hierarchien an P_1 gesendet, welche die fehlenden Daten und Hierarchien integriert. P_1 hält daraufhin wieder einen integrierten Gesamtdatensatz und die dazugehörigen Generalisierungshierarchien und kann nun die verschlüsselten Daten anonymisieren. Der verschlüsselte Datensatz ist in Abbildung 4.14 dargestellt.

Geschlecht	Alter	PLZ
$E_{22}(E_{12}(E_{32}(\text{w})))$	$E_{23}(E_{13}(E_{33}(18)))$	$E_{24}(E_{14}(E_{34}(82453)))$
$E_{12}(E_{32}(E_{22}(\text{w})))$	$E_{13}(E_{33}(E_{23}(35)))$	$E_{14}(E_{34}(E_{24}(81774)))$
$E_{12}(E_{32}(E_{22}(\text{m})))$	$E_{13}(E_{33}(E_{23}(70)))$	$E_{14}(E_{34}(E_{24}(81825)))$
$E_{32}(E_{22}(E_{12}(\text{m})))$	$E_{33}(E_{23}(E_{13}(66)))$	$E_{34}(E_{24}(E_{14}(81925)))$
$E_{22}(E_{12}(E_{32}(\text{w})))$	$E_{23}(E_{13}(E_{33}(17)))$	$E_{24}(E_{14}(E_{34}(82451)))$
$E_{32}(E_{22}(E_{12}(\text{m})))$	$E_{33}(E_{23}(E_{13}(34)))$	$E_{34}(E_{24}(E_{14}(82676)))$
$E_{12}(E_{32}(E_{22}(\text{m})))$	$E_{13}(E_{33}(E_{23}(21)))$	$E_{14}(E_{34}(E_{24}(82669)))$
$E_{32}(E_{22}(E_{12}(\text{w})))$	$E_{33}(E_{23}(E_{13}(45)))$	$E_{34}(E_{24}(E_{14}(81775)))$

Abbildung 4.14: Verschlüsselter Gesamtdatensatz im horizontalen Fall

Auch hier ist die Reihenfolge der Verschlüsselung nur der Nachvollziehbarkeit wegen eingezeichnet und hat keinerlei Auswirkungen auf das Protokoll. Die verschlüsselten, globalen Generalisierungshierarchien sind eine Integration der lokalen Teilhierarchien. Ein Beispiel für eine globale Generalisierungshierarchie für das Attribut Geschlecht ist in Abbildung 4.15 dargestellt. Die erste Zeile der Hierarchie stammt von einer anderen Teilhierarchie (P_1) als die zweite (P_2). Dies ist erkennbar gemacht an der unterschiedlichen Reihenfolge der Verschlüsselung, bei der Ausführung ist dies durch die Eigenschaft der Kommutativität nicht erkennbar. Die Anonymisierung und Entschlüsselung wird nun genauso durchgeführt wie im vertikalen Fall. Auch hierbei entsteht dieselbe Anonymisierung wie im Fall einer vertrauenswürdigen dritten Partei.

Lvl-0	Lvl-1
$E_{32}(E_{22}(E_{12}(\text{m})))$	$E_{32}(E_{22}(E_{12}(*)))$
$E_{12}(E_{32}(E_{22}(\text{w})))$	$E_{12}(E_{32}(E_{22}(*)))$

Abbildung 4.15: Globale, verschlüsselte Hierarchie für das Attribut Geschlecht

4.4.3.4 Analyse der Komplexität

In diesem Abschnitt wird das vorgestellte Protokoll analytisch, hinsichtlich Laufzeitperformanz und ausgetauschtem Datenvolumen, untersucht. Die meiste Zeit des Protokolls nehmen die kryptographischen Operationen und die Übertragung der ausgetauschten Daten ein. Damit sind diese beiden die einzigen berücksichtigten Operationen in dem Modell. Die Anzahl der teilnehmenden Parteien wird mit n bezeichnet, die Anzahl der Zeilen im Gesamtdatensatz mit r und die Anzahl der Spalten mit c. Die Anzahl der distinkten, also unterschiedlichen Werte, wird mit

d bezeichnet. Zur Vereinfachung der Analyse wird ein homogenes Setup angenommen, hierbei kann jede teilnehmende Maschine M_c kommutative Operationen pro Sekunde ausführen. Die Bandbreite für die Kommunikation wird mit M_b in Bits/-Sekunde angegeben. L_c ist die Länge eines kommutativ verschlüsselten Zellwertes. Wie in Abschnitt 4.5.4 beschrieben, kann der globale Datensatz repräsentiert werden als ein Array der Größe $c \cdot r$ für die Strukturinformationen und ein Wörterbuch der Größe d für die Datenelemente. Jedes Element (Zelle) im Strukturarray hat in dem Modell die Länge L_a. Als weitere Vereinfachung der Modellierung wird eine gleichmäßige Datenverteilung angenommen. Das bedeutet, dass im horizontalen Fall jede Partei $\frac{r}{n}$ Zeilen, c Spalten und d distinkte Werte aus dem globalen Datensatz besitzt. Im vertikalen Fall wird folglich angenommen, dass jede Partei dieselbe Anzahl an Spalten besitzt, $\frac{c}{n}$, die gleiche Anzahl an Zeilen r und dass die distinkten Werte auch gleichmäßig über alle Parteien sind, also jede Partei $\frac{d}{n}$ unterschiedliche Werte besitzt. Die Analyse kann ausgeweitet werden, um heterogene Setups und ungleichmäßige Datenverteilungen zu modellieren.

4.4.3.4.1 Kryptographische Operationen Während der Verschlüsselungsphase ist die Arbeitslast parallelisiert über alle Parteien, damit ist die Gesamtzeit vorgegeben durch die Arbeitslast einer einzelnen Partei. Im Falle einer vertikalen Verteilung der Daten besitzt jede Partei r distinkte Tupel-Identifikatoren und $\frac{d}{n}$ distinkte Werte. Da jede Partei die Daten von allen Parteien verschlüsseln muss, ist die Gesamtzahl der kommutativen Verschlüsselungsoperationen gegeben durch $e_v = n \cdot (r + \frac{d}{n})$. Im horizontalen Fall hingegen muss jede Partei $e_h = n \cdot d$ kommutative Operationen durchführen.

Während der Entschlüsselungsphase sind die Daten bereits integriert und somit ist hierbei kein Unterschied zwischen dem horizontalen und vertikalen Fall. Die Entschlüsselung geschieht sequenziell, hierbei muss jede Partei alle Daten entschlüsseln, damit ergeben sich $d_{vh} = n \cdot d$ kommutative Entschlüsselungsoperationen. Damit kann die Gesamtanzahl an notwendigen, kommutativen Operationen abgeschätzt werden mit $t_v^c = (e_v + d_{vh}) \cdot M_c$ im vertikalen Fall und $t_h^c = (e_h + d_{vh}) \cdot M_c$ im horizontalen Setup.

4.4.3.4.2 Ausgetauschtes Datenvolumen Die Anzahl der ausgetauschten Nachrichten in jeder der vier Phasen des Protokolls kann, anhand des Kommunikationspatterns (siehe Abschnitt 4.4.2) berechnet werden, und ist unabhängig von der Verteilung der Daten. Während der ersten Phase wird jedes Datensubset zu $n-1$ Parteien gesendet. Insgesamt werden also $f_v^1 = f_h^1 = n \cdot (n-1)$ Nachrichten versendet. Während der Integrationsphase (zweite Phase) werden die Daten iterativ integriert. Dieser Umstand wird so modelliert, dass ein Subset $n-1$ mal versendet wird, das zweite $n-2$ mal usw. Insgesamt resultiert dies in $f_v^2 = f_h^2 = n\frac{(n-1)}{2}$ versendeten Nachrichten. In den letzten beiden Phasen (drei und vier) werden $f_v^{34} = f_h^{34} = n-1$ Nachrichten versendet, da jede Nachricht an alle Teilnehmer, außer zum Initiator, verschickt wird.

Im nächsten Schritt wird die Größe einer Nachricht in den jeweiligen Phasen abgeschätzt. In der ersten Phase und im vertikalen Fall wird von jeder Partei ihr eigenes Datensubset versendet. Die Größe kann abgeschätzt werden durch $s_v^1 = (\frac{c}{n}+1) \cdot r \cdot L_a + (\frac{d}{n}+r) \cdot L_c$, da jede Partei $\frac{c}{n}$ Spalten hat und zusätzlich eine Spalte mit Tupel-Identifikatoren. Die zweite Klammer ergibt sich aus den Wörterbüchern,

hierbei hat jede Partei $\frac{d}{n}$ distinkte Werte und r distinkte Tupel-Identifikatoren. Die Datenmenge in der zweiten Phase, pro Partei, entspricht derjenigen in der ersten Phase, ohne die Tupel-Identifikatoren, also $s_v^2 = \frac{c}{n} \cdot r \cdot L_a + \frac{d}{n} \cdot L_c$. Im horizontalen Fall und der zweiten Phase hat jede Partei ein Datensubset der Größe $s_h^{12} = \frac{r}{n} \cdot c \cdot L_a + d \cdot L_c$, da jede Partei d distinkte Werte hat. In der dritten und vierten Phase enthält jede Nachricht eine Repräsentation des globalen Wörterbuch komprimierten Datensatzes, sowohl im horizontalen als auch vertikalen Fall. Die Größe kann abgeschätzt werden mit $s_v^{34} = s_h^{34} = c \cdot r \cdot L_a + d \cdot L_c$.

Zusammenfassend kann das Datenvolumen im vertikalen Fall abgeschätzt werden durch $d_v = f_v^1 \cdot s_v^1 + f_v^2 \cdot s_v^2 + 2 \cdot f_v^{34} \cdot s_v^{34}$. Im horizontalen Szenario kann es abgeschätzt werden mit $d_h = f_h^1 \cdot s_h^{12} + f_h^2 \cdot s_h^{12} + 2 \cdot f_h^{34} \cdot s_h^{34}$. Dadurch kann die benötigte Zeit für den Datentransfer abgeschätzt werden mit $t_h^d = d_h \cdot M_b$ bzw. $t_v^d = d_v \cdot M_b$.

4.4.3.5 Sicherheit

Das vorgestellte Protokoll hat etwas schwächere Sicherheitsgarantien, als traditionelle secure multi-party computation (SMC) Protokolle. Traditionell wird gefordert, dass alle teilnehmenden Parteien nur ihre eigenen Eingabedaten und das Gesamtergebnis lernen können. In dem hier vorgestellten Ansatz lernen die teilnehmenden Parteien allerdings auch die Verteilung der Werte von anderen Parteien. Im Folgenden wird erläutert, welche zusätzliche Information den einzelnen Teilnehmern wirklich bekannt wird, und welche Gegenmaßnahmen getroffen werden können. Da die Verschlüsselung deterministisch ist, kann ein Teilnehmer versuchen sein eigenes, verschlüsseltes Datensubset zu erkennen und somit Rückschlüsse auf die Werte anderer Parteien ziehen. Dies wäre möglich, da er seine eigenen Klartextwerte den verschlüsselten eigenen Geheimtextwerten zuordnen könnte und somit dieselben Werte anderer Parteien entschlüsseln könnte. Dies könnte bei horizontaler Datenverteilung während der Integrationsphase geschehen, wenn andere Parteien dieselben Werte teilen würden. Bei vertikaler Datenverteilung könnte das Wissen um die eigenen Geheimtextwerte dazu beitragen, dass man zu eigenen, speziellen Tupeln die Verteilung der anderen Attribute von anderen Teilnehmern lernt, die diesen speziellen Tupeln zugeordnet sind. Zwei Gegenmaßnahmen zu dieser Bedrohung existieren im Protokoll. Die Erste ist die Permutation der Tupel während der Verschlüsselungsphase (siehe Abschnitt 4.4.3.1.5). Diese soll es dem Angreifer erschweren, seine eigenen Daten wiederzuerkennen. Die zweite Maßnahme, das sukzessive Integrieren während der Integrationsphase, bewirkt, dass nur die Master Partei den Gesamtdatensatz bekommt, welcher aber vorher mit anderen Teildatensätzen vermischt wurde, um es wiederum dieser Partei schwerer zu machen, ihren eigenen Anteil der Daten zu erkennen. Sollten diese zwei Maßnahmen als nicht ausreichend erachtet werden, kann man eine weitere Partei in das Protokoll aufnehmen, die selbst keine Daten beiträgt, sondern nur als zentrale Partei fungiert, welche die verschlüsselten Daten der anderen erhält und anonymisiert. Somit gibt es keine Partei, die ihre eigenen Daten mit den komplett verschlüsselten vergleichen könnte. Diese Erweiterung wird im nächsten Protokoll (Protokoll B) genutzt.

Eine weitere verbleibende Bedrohung ist die Frequenzanalyse. Diese liegt in der Natur der deterministischen (nicht semantisch sicheren) Verschlüsselung, die für den vorgestellten Ansatz notwendig ist. Bei Ausnutzung dieser Bedrohung könnte ein Angreifer die Werte und die Verteilung eines Attributes raten und diese mit den

verschlüsselten Werten übereinanderlegen. Dies würde potenziell eine Entschlüsselung von Attributwerten erlauben. Die Anfälligkeit eines Attributes für einen solchen Angriff liegt in der Beschaffenheit des Attributes, d. h. dessen Verteilung und die Verfügbarkeit von Wissen über die Verteilung dieses Attributes in spezifischen Populationen. Für weitere Informationen siehe z. B. [104, 105]. Auch gegen diese Bedrohung existieren zwei Gegenmaßnahmen. Als Erstes kann die Bedrohung minimiert werden, indem vorgeneralisiert wird. Eine zweite Gegenmaßnahme existiert auf organisatorischer, rechtlicher Ebene mit den in der medizinischen Domäne üblichen Data-use-agreements (DUA). Dies sind Verträge, die verhindern sollen, dass Angriffe auf die Daten stattfinden. DUAs werden häufig eingesetzt, wenn komplette Anonymisierung die Datenqualität zu stark beeinflussen würde, um für die Forschung noch von Nutzen zu sein. Die technische Gegenmaßnahme der Vorgeneralisierung kann bei den Attributen angewendet werden, bei welchen eine starke Anfälligkeit für Frequenzangriffe angenommen wird. Hierbei wird der lokale Datensatz leicht generalisiert, bevor das Protokoll startet, damit wird die Verteilung etwas vergröbert und der Angriff erschwert. Wird das oben beschriebene Protokoll z. B. mit einem Anonymisierungsalgorithmus, welcher eine optimale Lösung findet, kombiniert, kann es, siehe Abschnitt 4.3, immer noch eine bessere Datenqualität liefern als die einfachen Varianten der verteilten Anonymisierung in Abschnitt 4.3, da hier oft große Informationsgewinnmargen vorhanden sind. Diese Margen können für die Vorgeneralisierung genutzt werden. Man kann sich die Vorgeneralisierung einfach so vorstellen, dass der Suchraum verkleinert wird und die Attribute mindestens auf die vorgeneralisierte Stufe generalisiert werden müssen. Außerdem haben die hier durchgeführten Versuche gezeigt, dass eine Reduktion der Entropie des Datensatzes, in einer signifikanten Beschleunigung des Protokolls, resultieren kann. Somit ist diese Gegenmaßnahme eine klassische Abwägung zwischen Sicherheit, Datenqualität und Effizienz.

Diese zwei Bedrohungen werden im malicious Angreifermodell schlimmer. In diesem Fall kann z. B. die Master Partei speziell manipulierte Eingabedaten erstellen, die besonders anfällig sind für Frequenzanalysen. Die Nutzung von manipulierten Daten kann allerdings nicht generell verhindert werden und ist auch bei anderen Arbeiten, die das malicious Angreifermodell annehmen, nicht berücksichtigt worden (z. B. [96]). Zusätzlich könnte die Master Partei als Angriff einfach die Anonymisierung nicht vornehmen und somit würde der nicht-anonymisierte Gesamtdatensatz in der Entschlüsselungsphase entschlüsselt. Dieser Bedrohung könnte entgegengewirkt werden, indem der verschlüsselte Gesamtdatensatz an alle Parteien verteilt würde. Dabei könnte jede Partei, unabhängig voneinander, anonymisieren. Diese unabhängigen Ergebnisse könnten dann verglichen werden und der Datensatz würde nur entschlüsselt werden, wenn alle Ergebnisse übereinstimmen. Allerdings wird in dieser Arbeit nur das semi-honest Modell berücksichtigt. Die Erweiterung auf das malicious Modell kann in zukünftigen Arbeiten erfolgen.

4.4.4 Protokoll B

Im Folgenden wird ein weiteres Protokoll zur Berechnung einer anonymisierten, integrierten Datenmenge vorgestellt. Wie auch Protokoll A kann dieses Protokoll sowohl bei vertikal als auch bei horizontal verteilt vorliegenden Daten verwendet werden. Zuerst wird ein Überblick über den Protokollablauf gegeben, gefolgt von

einer detaillierten Beschreibung des Protokolls. Abschließend wird die Komplexität und Sicherheit des Protokolls analysiert.

4.4.4.1 Protokoll

Dem Protokoll B liegt die gleiche Idee zugrunde, dass die Quasi-Identifikatoren und sensiblen Attribute so verschlüsselt werden, dass es möglich ist, die verschlüsselte Repräsentation der Daten zu anonymisieren; es werden also auch die dazugehörigen Generalisierungshierarchien verschlüsselt. Der Unterschied zu Protokoll A liegt in der Anforderung, dass es bei diesem Protokoll eine zusätzliche Partei (Hilfspartei) gibt, die nicht mit den anderen Parteien (Primärparteien) kooperiert.

Diese Hilfspartei übernimmt die Anonymisierung der verschlüsselten Daten, damit fällt zum einen die Gefahr weg, dass teilnehmende Primärparteien die verschlüsselten Daten und Generalisierungshierarchien auf ihre eigenen Teildaten abbilden, um die Verschlüsselung zu brechen, da keine der Primärparteien die verschlüsselten, integrierten und nicht anonymisierten Daten bzw. Generalisierungshierarchien sieht. Zum anderen entfällt bei diesem Protokoll die Notwendigkeit der homomorphen Verschlüsselung, was sowohl der Laufzeit als auch der zu übertragenden Datenmenge zugutekommt.

Das Protokoll startet indem sich die Primärparteien, welche die Daten halten, auf einen gemeinsamen Schlüssel einigen, den sie vor der zusätzlichen Hilfspartei geheim halten. Dazu können die Primärparteien z. B. ein beliebiges Schlüsselvereinbarungsprotokoll für Gruppen verwenden (z. B. das Octopus Protokoll, ein iterierter Diffie-Hellman, hierbei wird der per Diffie-Hellman generierte Schlüssel als Eingabe für den Zufallswert des Diffie-Hellman in der nächsten Runde verwendet [106]). Auch dieses Protokoll lässt sich in dieselben vier Phasen gliedern wie Protokoll A, siehe auch Abbildung 4.5. In der ersten Phase werden die Daten nun mittels einfacher symmetrischer Verschlüsselung, zellenweise verschlüsselt. Wichtig ist auch hierbei wieder, dass die Verschlüsselung deterministisch ist, d. h. gleiche Zellwerte in gleiche Kryptotexte transformiert werden. Die so verschlüsselten Daten und Generalisierungshierarchien müssen nun vor den anderen Parteien geheim gehalten werden und werden nur der Hilfspartei, die den Schlüssel nicht kennt, zur Kenntnis gebracht. Damit kann die Hilfspartei aus allen Teildaten der Primärparteien einen verschlüsselten, integrierten Datensatz und dazugehörige, verschlüsselte und integrierte Generalisierungshierarchien, erstellen. Nun können die integrierten Daten von der Hilfspartei anonymisiert werden. Dieser Schritt ist wiederum möglich, da die Daten und Generalisierungshierarchien im Verschlüsselungsschritt auf die gleiche Weise, mittels eines deterministischen Verfahrens, dem gleichen Schlüssel und auf Tabellenzellenebene, verschlüsselt wurden. Hat die Hilfspartei nun eine anonyme Repräsentation der Daten errechnet, kann sie diese einer beliebigen Primärpartei zur Entschlüsselung schicken.

4.4.4.1.1 Verschlüsselung Der Verschlüsselungsschritt läuft bei vertikaler und horizontaler Verteilung der Daten nahezu identisch ab. Zuerst einigen sich alle Primärparteien auf einen gemeinsamen Schlüssel je Spalte. Im vertikalen Fall kann jede Partei sich die Schüssel für ihre eigenen Spalten selbst generieren und diese geheim halten. Dann muss allerdings im Entschlüsselungsschritt der Datensatz an alle Parteien versendet werden (siehe Abschnitt über Entschlüsselung). Jede Spalte jeder Primärpartei wird nun, mittels dem dazugehörigen Schlüssel, von ihr verschlüsselt.

Mit den dazugehörigen Generalisierungshierarchien wird genauso verfahren. Diese Daten und Generalisierungshierarchien können nun permutiert werden. Somit besitzt jede Primärpartei eine verschlüsselte Repräsentation ihrer eigenen Daten und Generalisierungshierarchien. Nun sendet jede Primärpartei die verschlüsselten Daten und Generalisierungshierarchien an die Hilfspartei. Damit ist die Verschlüsselungsphase abgeschlossen.

4.4.4.1.2 Integration Daraufhin kann die Hilfspartei den integrierten Datensatz und die integrierten Generalisierungshierarchien generieren. Hier gibt es leichte Unterschiede zwischen horizontal und vertikal verteilten Daten.

Im horizontalen Fall müssen die Teildaten nur konkateniert werden, um diese zu integrieren. Auch die (Teil-)Hierarchien werden nur konkateniert und evtl. vorhandene, doppelte Einträge eliminiert. Duplikate können auftreten, wenn Parteien dieselben Datenelemente haben, da dadurch zweimal dieselben Generalisierungsregeln in beiden Generalisierungshierarchien eingefügt worden sind. Hierbei können keine Konflikte auftreten, da diese Regeln global definiert wurden (vgl. Abschnitt 4.4.1).

Im vertikalen Fall werden die Teildaten, anhand des gemeinsamen Tupel-Identifikators, zusammengefügt. Sollten nicht alle Parteien Daten zu allen Einträgen haben, können fehlende Werte z. B. mit Null Werten oder Platzhaltern aufgefüllt werden. Die Generalisierungshierarchien müssen im vertikalen Fall nicht integriert werden, da (siehe Abschnitt 4.4.1) jede Partei unterschiedliche Spalten und somit unterschiedliche Generalisierungshierarchien beiträgt.

4.4.4.1.3 Anonymisierung Die Hilfspartei hält, nach der vorherigen Integration, die integrierten und verschlüsselten Daten und Generalisierungshierarchien und kann nun die Anonymisierung vornehmen. Der Anonymisierungsschritt ist identisch mit dem in Protokoll A beschriebenen Anonymisierungsschritt (vgl. Abschnitt 4.4.3.1.3). Das Ergebnis dieses Schritts ist ein verschlüsselter, anonymisierter Datensatz.

4.4.4.1.4 Entschlüsselung Der letzte Schritt des Protokolls ist die Entschlüsselung der anonymisierten Daten. Hierbei wird der anonymisierte und verschlüsselte Datensatz der Hilfspartei an eine beliebige Primärpartei gesendet. Diese kann die Daten, mithilfe der jeweiligen symmetrischen Schlüssel, entschlüsseln und erhält so die anonymisierten Daten. Im vertikalen Fall, wenn die Parteien ihre eigenen Schlüssel geheim halten wollen, muss nun der Datensatz noch an die anderen Primärparteien gesendet werden, welche jeweils ihre eigenen Spalten entschlüsseln.

4.4.4.2 Beispiel: vertikal verteilt

Für das folgende Beispiel wird dieselbe Aufteilung der Daten auf die drei Primärparteien P_1, P_2 und P_3, wie schon in Abschnitt 4.4.3.2 Abbildung 4.7 beschrieben angenommen. Zusätzlich gibt es eine vierte Partei P_4, die für die Anonymisierung zuständig ist und selbst keine Daten beiträgt. Die Parteien P_1, P_2 und P_3 einigen sich nun auf einen symmetrischen Schlüssel pro Spalte. Die Schlüssel werden mit einem Index, wie folgt, bezeichnet: TID=1, Geschlecht=2, Alter=3 und Postleitzahl=4. $E_3(45)$ bedeutet nun, dass das Alter 45 mit dem Schlüssel für Attribut 3 symmetrisch verschlüsselt wird. Im Folgenden wird davon ausgegangen, dass die Schlüssel der jeweiligen Spalten, allen Primärparteien bekannt sind. Der Schlüssel der TID-Spalte muss in jedem Fall allen Parteien bekannt sein. Als Generalisie-

rungshierarchien werden wiederum die Bäume aus den Abbildungen 2.10, 2.9 und 2.11 genutzt. Im Folgenden wird wieder auf Partei P_2 fokussiert, die anderen Parteien verhalten sich analog.

TID	Alter
$E_1(1)$	$E_3(34)$
$E_1(2)$	$E_3(45)$
$E_1(3)$	$E_3(66)$
$E_1(4)$	$E_3(70)$
$E_1(5)$	$E_3(35)$
$E_1(6)$	$E_3(21)$
$E_1(7)$	$E_3(18)$
$E_1(8)$	$E_3(17)$

(a) Datensubset

Lvl-0	Lvl-1	Lvl-2
$E_3(34)$	E_3(20-60)	$E_3(*)$
$E_3(45)$	E_3(20-60)	$E_3(*)$
$E_3(66)$	$E_3(\geq 61)$	$E_3(*)$
$E_3(70)$	$E_3(\geq 61)$	$E_3(*)$
$E_3(35)$	E_3(20-60)	$E_3(*)$
$E_3(21)$	E_3(20-60)	$E_3(*)$
$E_3(18)$	$E_3(\leq 19)$	$E_3(*)$
$E_3(17)$	$E_3(\leq 19)$	$E_3(*)$

(b) Hierarchie

Abbildung 4.16: Verschlüsselte Daten von P_2

Als Erstes verschlüsselt P_2 ihre eigenen Daten symmetrisch mit dem Schlüssel der jeweiligen Spalte, im Beispiel TID und Alter. Auch die dazugehörige Generalisierungshierarchie für das Attribut Alter wird mit dem Schlüssel für die Spalte Alter verschlüsselt. Das Ergebnis ist in Abbildung 4.16 dargestellt. Als nächsten Schritt permutiert nun P_2 ihre Daten und Hierarchien zeilenweise und sendet diese an die Hilfspartei P_4. Nachdem die anderen Primärparteien ihrerseits die verschlüsselten und permutierten Daten an die Hilfspartei gesendet haben, hält die Hilfspartei nun alle Teildaten und Generalisierungshierarchien der Primärparteien.

Geschlecht	Alter	PLZ
E_2(m)	E_3(20-60)	$E_4(82^{***})$
E_2(w)	E_3(20-60)	$E_4(81^{***})$
E_2(m)	$E_3(\geq 61)$	$E_4(81^{***})$
E_2(m)	$E_3(\geq 61)$	$E_4(81^{***})$
E_2(w)	E_3(20-60)	$E_4(81^{***})$
E_2(m)	E_3(20-60)	$E_4(82^{***})$
E_2(w)	$E_3(\leq 19)$	$E_4(82^{***})$
E_2(w)	$E_3(\leq 19)$	$E_4(82^{***})$

Abbildung 4.17: 2-anonymisierter und verschlüsselter Datensatz

Die Hilfspartei kann nun mit der Integration beginnen, indem sie die Teildatensätze aller Primärparteien, anhand der verschlüsselten TID Spalte, sortiert und zusammenfügt. Somit hält die Hilfspartei P_4 nun einen integrierten und verschlüsselten Gesamtdatensatz, mit den dazugehörigen verschlüsselten Hierarchien.

Nun kann die Hilfspartei die Daten mithilfe der Hierarchien anonymisieren. Wie auch im letzten Beispiel wird wieder angenommen, dass 2-Anonymität gefordert ist. Abbildung 4.17 zeigt die, während des Anonymisierungsprozesses berechnete, optimale und verschlüsselte Lösung.

Nun wird, als letzter Schritt, dieser verschlüsselte und anonymisierte Datensatz an eine der drei Primärparteien gesendet (die genaue Partei ist wahlfrei, da die Annahme besteht, dass alle Parteien alle Schlüssel kennen). Im Beispiel erhält nun P_1 den Datensatz und entschlüsselt diesen. Dieser Datensatz entspricht nun wieder genau dem, welcher auch entstanden wäre, wenn eine vertrauenswürdige, dritte Partei die integrierten Daten anonymisiert hätte.

4.4.4.3 Beispiel: horizontal verteilt

Auch im horizontalen Fall wird das Beispiel wieder mit drei Primärparteien P_1, P_2 und P_3 durchgespielt. Außerdem gibt es wieder eine vierte Partei (Hilfspartei) P_4, welche die Anonymisierung übernimmt und keine eigenen Daten zum Protokoll beiträgt. Die Verteilung der Daten auf die Primärparteien ist dieselbe wie in Abschnitt 4.4.3.3 Abbildung 4.11. Die Generalisierungshierarchien für das Datensubset von Partei P_1 sind auch dieselben wie in Abbildung 4.12. Außerdem hat nun jede der Primärparteien, für jede Spalte, einen gemeinsamen Schlüssel. Auch hier wird die Subscript Notation für die Verschlüsselung, wie im vorherigen Abschnitt, verwendet. Die Schlüssel werden mit einem Index, wie folgt bezeichnet: Geschlecht=2, Alter=3 und Postleitzahl=4 (die Spalte mit dem TID ist im horizontalen Fall nicht notwendig und wird daher weggelassen). $E_3(45)$ bedeutet nun, dass das Alter 45 mit dem Schlüssel für Attribut 3 symmetrisch verschlüsselt wird, da der Schlüssel für alle Primärparteien derselbe ist, benötigt man keinen separaten Index für die Partei. Um konsistent mit dem Beispiel für den horizontalen Fall, bei Protokoll A, zu sein, wird im ersten Schritt wieder auf die Partei P_1 fokussiert.

Als Erstes kann P_1 ihre eigenen Teildaten und die dazugehörigen Teilhierarchien, mit dem jeweiligen, gemeinsamen Schlüssel je Spalte verschlüsseln. Der verschlüsselte Teildatensatz und die Teilhierarchie für das Attribut Alter ist in Abbildung 4.18 dargestellt. Die beiden anderen Teilgeneralisierungshierarchien (Geschlecht, PLZ) sind analog zu der Hierarchie Alter, dargestellt in Abbildung 4.18b, verschlüsselt.

Geschl.	Alter	PLZ
E_2(m)	E_3(34)	E_4(82667)
E_2(w)	E_3(45)	E_4(81775)
E_2(m)	E_3(66)	E_4(81925)

(a) Verschlüsselte Daten

Lvl-0	Lvl-1	Lvl-2
E_3(34)	E_3(20-60)	E_3(*)
E_3(45)	E_3(20-60)	E_3(*)
E_3(66)	E_3(≥61)	E_3(*)

(b) Verschlüsselte Gen. Hier. Alter

Abbildung 4.18: Verschlüsselter Teildatensatz von P_1

Nun werden die Teildaten und Teilhierarchien von P_1 permutiert und zur Hilfspartei P_4 gesendet. Die anderen Primärparteien verfahren analog.

Nachdem nun P_4 alle verschlüsselten Teildatensätze und Teilhierarchien erhalten hat, kann diese die Daten integrieren. Dazu konkateniert P_4 nun einfach die Daten und die Hierarchien. Bei den Hierarchien werden dabei Duplikate entfernt. Ein Beispiel für eine integrierte Generalisierungshierarchie für das Attribut Geschlecht wo alle Duplikate bereits entfernt wurden ist in Abbildung 4.19 dargestellt.

Lvl-0	Lvl-1
$E_2(E_2(E_2(\text{m})))$	$E_2(E_2(E_2(*)))$
$E_2(E_2(E_2(\text{w})))$	$E_2(E_2(E_2(*)))$

Abbildung 4.19: Globale, verschlüsselte Hierarchie für das Attribut Geschlecht

Auf den integrierten und verschlüsselten Daten kann nun, mithilfe der verschlüsselten und integrierten Hierarchien, die Anonymisierung durchgeführt werden. Auch hierbei entsteht dieselbe Anonymisierung, als wenn die Anonymisierung auf unverschlüsselten und integrierten Daten, die eine vertrauenswürdige dritte Partei erstellt, durchgeführt worden wäre. Das Ergebnis ist in Abbildung 4.20 dargestellt.

Geschlecht	Alter	PLZ
$E_2(\text{m})$	$E_3(20\text{-}60)$	$E_4(82^{***})$
$E_2(\text{w})$	$E_3(20\text{-}60)$	$E_4(81^{***})$
$E_2(\text{m})$	$E_3(\geq 61)$	$E_4(81^{***})$
$E_2(\text{m})$	$E_3(\geq 61)$	$E_4(81^{***})$
$E_2(\text{w})$	$E_3(20\text{-}60)$	$E_4(81^{***})$
$E_2(\text{m})$	$E_3(20\text{-}60)$	$E_4(82^{***})$
$E_2(\text{w})$	$E_3(\leq 19)$	$E_4(82^{***})$
$E_2(\text{w})$	$E_3(\leq 19)$	$E_4(82^{***})$

Abbildung 4.20: 2-anonymiserter und verschlüsselter Datensatz im horizontalen Fall

4.4.4.4 Analyse der Komplexität

Die Analyse der Komplexität ist einfacher als bei Protokoll A. Um einen Vergleich zu ermöglichen, wird dasselbe Vorgehen wie bei Protokoll A gewählt. Wieder wird die Laufzeit und das benötige Datenvolumen anhand von Basisparametern des Datensatzes und der Plattform geschätzt. Die Anzahl der teilnehmenden Parteien wird wieder mit n bezeichnet, die Anzahl der Zeilen im Gesamtdatensatz mit r und die Anzahl der Spalten mit c. Die Anzahl der distinkten, also unterschiedlichen Werte, wird mit d bezeichnet. Die Anzahl der symmetrischen Ver-/Entschlüsselungsoperationen pro Sekunde wird mit M_s bezeichnet. Es wird angenommen, dass alle teilnehmenden Maschinen die gleiche Rechenleistung haben. Die Bandbreite für die Kommunikation wird mit M_b in Bits/Sekunde angenommen. L_s ist die Länge eines symmetrisch verschlüsselten Zellwertes. Für die restlichen, benötigten Parameter werden dieselben Werte wie in Abschnitt 4.4.3.4 angenommen, $c \cdot r$ für die Strukturinformationen, ein Wörterbuch der Größe d und jedes Element im Strukturarray hat die Länge L_a. Die Daten sind gleichmäßig verteilt. Es hält also jede Primärpartei im horizontalen Fall $\frac{r}{n}$ Zeilen, c Spalten und d distinkte Werte. Im vertikalen Fall hat jede Primärpartei die gleiche Anzahl an Spalten, $\frac{c}{n}$, die gleiche Anzahl an Zeilen r und $\frac{d}{n}$ unterschiedliche Werte.

4.4.4.4.1 Kryptographische Operationen Während der Verschlüsselungsphase ist die Arbeitslast parallelisiert über alle Primärparteien, damit ist wieder die Gesamtzeit vorgegeben, durch die Arbeitslast einer einzelnen Partei. Im vertikalen Fall also, wenn jede Partei eine zusätzliche Spalte mit r Tupel-Identifikatoren

besitzt, muss jede Primärpartei $e_v = r + \frac{d}{n}$ distinkte Zellenwerte symmetrisch verschlüsseln. Im horizontalen Fall hingegen muss jede Partei nur $e_h = d$ Verschlüsselungsoperationen durchführen. Auch bei Protokoll B sind während der Entschlüsselungsphase die Daten bereits integriert und somit ist hierbei kein Unterschied zwischen dem horizontalen und vertikalen Fall. Die Entschlüsselung geschieht im einfachsten Fall durch eine Primärpartei, welche somit $d_{vh} = d$ Entschlüsselungsoperationen durchführen muss. Damit kann die Gesamtanzahl an notwendigen, symmetrischen Verschlüsselungsoperationen abgeschätzt werden mit $t_v^c = (e_v + d_{vh}) \cdot M_s$ im vertikalen Fall und $t_h^c = (e_h + d_{vh}) \cdot M_s$ im horizontalen Setup.

4.4.4.4.2 Ausgetauschtes Datenvolumen Die Nachrichten werden nach einem Sternmuster ausgetauscht. Während der Verschlüsselungsphase senden alle Primärparteien ihre verschlüsselten Teildaten an die Hilfspartei. Insgesamt werden also $f_v^1 = f_h^1 = n$ Nachrichten versendet. Während der Integrations- und Anonymisierungsphase werden keine Nachrichten versendet. Bei der Entschlüsselungsphase wird nur eine Nachricht mit dem anonymisierten und verschlüsselten Gesamtdatensatz versendet.

Der Einfachheit halber, und weil sie nur einen kleinen Teil des Datenvolumens ausmachen, werden die Generalisierungshierarchien bei der Berechnung der Größe der einzelnen Nachrichten weggelassen und nur die Daten betrachtet. Die Größe jeder der n Nachrichten in der ersten Phase, im vertikalen Fall, kann berechnet werden durch $s_v^1 = (\frac{c}{n} + 1) \cdot r \cdot L_a + (\frac{d}{n} + r) \cdot L_s$, da jede Partei $\frac{c}{n}$ Spalten hat und zusätzlich eine Spalte mit Tupel-Identifikatoren. Die zweite Klammer ergibt sich aus den Wörterbüchern, da jede Partei $\frac{d}{n}$ distinkte Werte und r distinkte Tupel-Identifikatoren hat. Im horizontalen Fall und in der Verschlüsselungsphase hat jede Partei ein Datensubset der Größe $s_h^1 = \frac{r}{n} \cdot c \cdot L_a + d \cdot L_s$, da jede Partei d distinkte Werte hat. Bei der Entschlüsselungsphase werden der gesamte Datensatz und die Wörterbücher übertragen. Die Größe kann abgeschätzt werden mit $s_{vh}^4 = c \cdot r \cdot L_a + d \cdot L_s$.

Zusammenfassend kann das Datenvolumen, im vertikalen Fall, abgeschätzt werden durch $d_v = n \cdot s_v^1 + s_{vh}^4$. Im horizontalen Szenario kann es abgeschätzt werden mit $d_h = n \cdot s_h^1 + s_{vh}^4$. Dadurch kann die benötigte Zeit für den Datentransfer abgeschätzt werden mit $t_h^d = d_h \cdot M_b$ bzw. $t_v^d = d_v \cdot M_b$.

4.4.4.5 Sicherheit

Es gelten bei Protokoll B zunächst die gleichen Sicherheitsgarantien wie bei Protokoll A. Auch bei Protokoll B lernt eine Partei, in diesem Fall die Hilfspartei, die Verteilung der Daten. Diese hat allerdings, anders als bei Protokoll A, keine Möglichkeit, mittels Abgleich mit ihren eigenen unverschlüsselten Daten, Rückschlüsse auf die unverschlüsselten Daten zu ziehen. Außerdem kann die Hilfspartei, bei wiederholter Anwendung des Protokolls, Unterschiede in der Verteilung der verschlüsselten Daten feststellen und somit evtl. Rückschlüsse auf neu hinzugekommene Tupel herstellen.

Der größte Unterschied bei Protokoll B zu Protokoll A liegt darin, dass jeder einzelne Teilnehmer die Daten der anderen entschlüsseln kann. Es muss also bei Protokoll B sichergestellt werden, dass die verschlüsselten Daten der Primärparteien nur der Hilfspartei bekannt sind, und dass diese nicht mit einer Primärpartei kollaboriert. Der ersten Bedrohung kann begegnet werden, indem die Primärpartei-

en einen sicheren (und damit auch authentisierten) Kommunikationskanal mit der Hilfspartei aufbauen. Die zweite Bedrohung kann durch rechtliche Rahmenbedingungen gemindert werden.

4.4.5 Protokoll C

Protokoll A und Protokoll B haben gegenüber anderen, bekannten Ansätzen (siehe Abschnitt 4.2) den Vorteil, dass sie keine Kommunikation zwischen den Parteien während der Anonymisierungsphase benötigen. Allerdings werden, bei den beiden oben vorgestellten Protokollen, Informationen über die Verteilung der Daten mindestens einem Teilnehmer bekannt. Das folgende Protokoll benötigt nun eine Kommunikation während der Anonymisierung, es kann aber hierbei sichergestellt werden, dass die Information über die Verteilung so verschleiert wird, dass keinerlei nützliche Information nicht berechtigten Parteien bekannt wird. Dazu wird eine additiv homomorphe Verschlüsselung während der Anonymisierungsphase benötigt. Im Folgenden wird dieses Protokoll vorgestellt und anschließend, anhand eines Beispiels, durchgespielt. Abschließend wird wieder die Komplexität und Sicherheit des Protokolls analysiert.

4.4.5.1 Protokoll

An dem Protokoll nehmen wieder die Primärparteien teil, die ihre Daten gemeinsam anonymisieren wollen und eine Hilfspartei, welche die Anonymisierung vornimmt.

Wie bei Protokoll A und B gibt es die vier Phasen Verschlüsselung, Integration, Anonymisierung und Entschlüsselung. Es werden wieder die Werte auf Zellebene deterministisch verschlüsselt. Allerdings werden bei diesem Protokoll zusätzlich Pseudotupel eingefügt, welche die wahre Verteilung verschleiern. Um diese Pseudotupel bei der Anonymisierung herausrechnen zu können, wird bei jedem Tupel zusätzlich ein Zähler mitgeführt, welcher die Anzahl des wirklichen Vorkommens enthält. Im Falle eines Pseudotupels also 0. Dieser Tupel Zähler wird additiv homomorph und semantisch sicher verschlüsselt. Der Anonymitätstest muss dann so ausgeführt werden, dass diese Pseudotupel nicht berücksichtigt werden. Außerdem dürfen keiner Partei gleichzeitig die Daten, die zu einer Gruppe gehören und die Anzahl der Pseudotupel in einer Gruppe, bekannt sein.

Auch in diesem Protokoll gibt es wieder eine Hilfspartei, welche die eigentliche Transformation und Gruppierung auf die verschlüsselten Daten vornimmt. Die Transformation läuft wie gewohnt ab; die Gruppierung, also das Zählen, wie viele Tupel in eine Gruppe fallen, geschieht durch Addition der homomorph verschlüsselten Tupel-Zähler. Die Information über die Gruppengrößen (ohne die Information über die Daten, welche zu dieser Gruppe gehören) wird für jede Transformation verschlüsselt an eine Primärpartei gesendet, welche die Größe der jeweiligen Gruppe entschlüsselt und auf Anonymität prüft. Diese meldet sodann der Hilfspartei zurück, ob die aktuell zu testende Transformation anonym ist. Anhand dieser Information kann die Hilfspartei mit dem Protokoll fortfahren. Im Folgenden wird von k-Anonymität als Kriterium ausgegangen. Das genannte Protokoll ist aber auch auf andere Kriterien erweiterbar, solange wieder keine Partei während des Anonymisierungsprozesses Zugriff auf die Klartextwerte benötigt.

4.4.5.1.1 Verschlüsselung Wie bei Protokoll B startet dieses Protokoll damit, dass die Primärparteien sich im horizontalen Fall auf einen gemeinsamen symme-

trischen Schlüssel, je Spalte, einigen. Im vertikalen Fall kann jede Partei wiederum ihren Schlüssel selbst generieren und geheim halten. Zusätzlich wird ein asymmetrisches, additiv homomorphes Schlüsselpaar generiert. Den öffentlichen Schlüssel erhalten alle Parteien, der private Schlüssel verbleibt bei einer ausgewählten Partei. Diese ist später für die Auswertung der transformierten Daten zuständig. Jede der teilnehmenden Parteien, außer der Hilfspartei, welche die Anonymisierung vornimmt, kann diese Rolle übernehmen. Im einfachsten Fall wird davon ausgegangen, dass die erste Primärpartei das Schlüsselpaar generiert und auch den geheimen Schlüssel hält. Nachdem alle Schlüssel verteilt wurden, kann jede Partei ihre Pseudotupel generieren. Die Beschaffenheit und die Anzahl der Tupel kann, frei von der jeweiligen Partei, gewählt werden. Die Verteilung der Pseudotupel sollte so gewählt werden, dass die Verteilung der echten Tupel maskiert wird. Jedes Tupel, echt oder pseudo, bekommt nun einen Zähler zugeordnet. Der Zähler ist null für die Pseudotupel und eins für die echten Tupel. Daraufhin wird dieser Zähler, mit dem öffentlichen Schlüssel für das additiv homomorph und semantisch sichere Verschlüsselungsverfahren, verschlüsselt. Das semantisch sichere Verschlüsselungsverfahren bewirkt, dass, obwohl nur zwei verschiedene Werte (0 und 1) in der Zählerspalte auftreten, es nicht möglich ist, dem verschlüsselten Wert anzusehen, zu welcher der beiden Gruppen er gehört. Die Werte der neu hinzugefügten Tupel müssen nun auch in die jeweilige Generalisierungshierarchie aufgenommen werden. Die Generalisierungshierarchien werden dann, wie in Protokoll B, symmetrisch verschlüsselt. Nun werden die Tupel zufällig permutiert, zusammen mit den verschlüsselten Zählern und den verschlüsselten Generalisierungshierarchien an die Hilfspartei gesendet.

4.4.5.1.2 Integration Die Integrationsphase läuft nun, wie in Protokoll B, ab. Da die Hilfspartei alle Teildatensätze und Hierarchien erhalten hat, kann sie nun die Daten integrieren. Im horizontalen Fall werden die Tupel konkateniert und die Teilhierarchien integriert, was bedeutet, dass gleiche Regeln gelöscht und nur einmal in die Gesamthierarchie aufgenommen werden. Im vertikalen Fall werden die Tupel wieder sortiert und anhand der verschlüsselten TID werden die Spalten integriert (vgl. auch Protokoll B). Hierbei müssen die Generalisierungshierarchien nicht integriert werden, da jede Partei disjunkte Spalten beisteuert. Nun hält die Hilfspartei die verschlüsselten Tupel, jeweils mit den verschlüsselten Zählern und den verschlüsselten Generalisierungshierarchien.

4.4.5.1.3 Anonymisierung Der Anonymisierungsschritt unterscheidet sich nun von Protokoll B. Die Hilfspartei kann nun den Suchraum möglicher Generalisierungen durchlaufen. Soll nun ein Kandidat des Suchraumes auf Anonymität getestet werden, wird die Hilfspartei als Erstes die Transformation vornehmen, sodass die Daten der gewünschten Transformation entsprechen. Dies ist mit den verschlüsselten Daten und Generalisierungshierarchien, wie in den vorherigen Protokollen möglich, da dies nur ein Ersetzen von verschlüsselten Werten ist. Nach der Transformation werden die Daten gruppiert, um im einfachsten Fall festzustellen, ob alle Gruppen mindestens die Größe k haben (k-Anonymität). Da die Hilfspartei nicht weiß, welche der Tupel echt sind und welche Pseudotupel sind, nutzt nun die Hilfspartei beim Gruppieren die additiv homomorphe Eigenschaft der verschlüsselten Zähler aus. Die Hilfspartei kann nun einfach die verschlüsselten Zähler so kombinieren (im Falle des Paillier-Kryptosystems ist das eine Modulomultiplikation),

dass die verschlüsselte Summe der beiden zu gruppierenden Zähler entsteht. Da die Pseudotupel verschlüsselte null Werte enthalten, tragen diese nicht zu der Summe bei. Nach diesem Schritt hat nun die Hilfspartei Gruppen, mit einem jeweils dazugehörigen, verschlüsselten Zähler, gebildet. Sie kann nun die Gruppenidentifikatoren entfernen und an die Primärpartei eine Liste mit verschlüsselten und permutierten Zählern senden. Die Primärpartei entschlüsselt die Zähler der Gruppen und kann nun testen, ob alle Gruppen die geforderte Größe haben. Bei diesem Schritt kann auch die Unterdrückung von einzelnen Tupeln berücksichtigt werden. Das Ergebnis des Tests, also ob die Transformation anonym ist oder nicht, sendet die Primärpartei nun wieder an die Hilfspartei zurück. Die Hilfspartei kann dann mit dem Ergebnis weiterarbeiten und ggf. die nächste, mögliche Transformation generieren, und mithilfe der Primärpartei, auf Anonymität testen. Hat die Hilfspartei den gewünschten, anonymen Zustand gefunden, hat diese nun wieder einen verschlüsselten und anonymen Datensatz, der allerdings noch die unechten Tupel enthält.

4.4.5.1.4 Entschlüsselung Um den anonymisierten und unverschlüsselten Datensatz zu erhalten, sendet nun die Hilfspartei die transformierten und erneut permutierten Tupel, zusammen mit den ursprünglich verschlüsselten Zählern an die Primärpartei. Diese kann nun die Daten entschlüsseln und anhand der entschlüsselten Zähler, die Pseudotupel entfernen. Damit erhält sie den anonymisierten und unverschlüsselten Gesamtdatensatz.

4.4.5.2 Beispiel: vertikal verteilt

Für das Beispiel wird wieder dieselbe Aufteilung der Daten auf die drei Primärparteien P_1, P_2 und P_3, wie in Abschnitt 4.4.3.2 Abbildung 4.7 dargestellt, angenommen. Zusätzlich gibt es, wie im Protokoll B (siehe Abschnitt 4.4.4) eine vierte Partei P_4, die für die Anonymisierung zuständig ist und selbst keine Daten beiträgt. Der Beginn von Protokoll C läuft wie der Beginn von Protokoll B ab. Die drei Primärparteien P_1, P_2 und P_3 einigen sich auf einen symmetrischen Schlüssel pro Spalte. Außerdem generiert eine Primärpartei ein Schlüsselpaar für das additiv homomorphe Verschlüsselungsverfahren. Den privaten Schlüssel behält die Partei und der öffentliche Schlüssel wird an die restlichen Primärparteien und die Hilfspartei verteilt. Wie bei den anderen Beispielen, werden die Schlüssel mit einem Index, wie folgt, bezeichnet: TID=1, Geschlecht=2, Alter=3 und Postleitzahl=4. $E_3(45)$ bedeutet nun, dass das Alter 45 mit dem Schlüssel für Attribut 3, symmetrisch verschlüsselt wird. Die Verschlüsselung mit dem additiv homomorphen Verfahren wird mit $E_h(x)$ bezeichnet, also bedeutet $E_h(1)$, dass der Wert 1 mit dem additiv homomorphen Verfahren verschlüsselt wird. Als Nächstes einigen sich die Primärparteien auf die Anzahl der hinzuzufügenden Pseudotupel, im Beispiel 25 %. Damit werden im Beispiel, bei jeder Primärpartei, zwei zusätzliche Pseudotupel generiert. Außerdem muss im vertikalen Fall noch sichergestellt werden, dass die TIDs der Pseudotupel bei allen Parteien gleich sind. Da im Beispiel davon ausgegangen wird, dass alle Parteien die gleiche Anzahl an Pseudotupel besitzen, werden die Pseudotupel einfach sequenziell aufsteigend durchnummeriert und dieser Zähler wird als TID genutzt. Im Beispiel wird die TID eines Pseudotupels zusätzlich mit P, als Präfix kenntlich gemacht. Außerdem wird jedem Tupel der Primärparteien ein Zähler Feld hinzugefügt, dass bei einem echten Tupel den Wert 1 erhält und bei einem Pseudotupel den Wert 0. Diese Unterscheidung macht nur eine Partei, die anderen

fügen an ihre Tupel immer den Wert 0 an. Dies ist notwendig, da sonst im Integrationsschritt, bei der Addition der Zählerwerte, die echten Tupel den Wert 3 hätten. Es werden auch in diesem Beispiel die Hierarchien aus den Abbildungen 2.10, 2.9 und 2.11 verwendet. Wenn nun die Pseudotupel in den Teildatensatz der Partei eingefügt werden, müssen auch passende Generalisierungsregeln eingefügt werden. Auch bei diesem Beispiel liegt der Fokus am Anfang des Protokolls wieder auf der Partei P_2.

In Abbildung 4.21 ist der verschlüsselte Teildatensatz von P_2 dargestellt, nachdem die zwei Pseudotupel (P1 und P2) generiert und die Zählerspalte hinzugefügt wurde.

TID	Alter	Zähler
$E_1(1)$	$E_3(34)$	$E_h(1)$
$E_1(2)$	$E_3(45)$	$E_h(1)$
$E_1(3)$	$E_3(66)$	$E_h(1)$
$E_1(4)$	$E_3(70)$	$E_h(1)$
$E_1(5)$	$E_3(35)$	$E_h(1)$
$E_1(6)$	$E_3(21)$	$E_h(1)$
$E_1(7)$	$E_3(18)$	$E_h(1)$
$E_1(8)$	$E_3(17)$	$E_h(1)$
$E_1(\text{P1})$	$E_3(65)$	$E_h(0)$
$E_1(\text{P2})$	$E_3(20)$	$E_h(0)$

(a) Datensubset

Lvl-0	Lvl-1	Lvl-2
$E_3(34)$	$E_3(20\text{-}60)$	$E_3(*)$
$E_3(45)$	$E_3(20\text{-}60)$	$E_3(*)$
$E_3(66)$	$E_3(\geq 61)$	$E_3(*)$
$E_3(70)$	$E_3(\geq 61)$	$E_3(*)$
$E_3(35)$	$E_3(20\text{-}60)$	$E_3(*)$
$E_3(21)$	$E_3(20\text{-}60)$	$E_3(*)$
$E_3(18)$	$E_3(\leq 19)$	$E_3(*)$
$E_3(17)$	$E_3(\leq 19)$	$E_3(*)$
$E_3(65)$	$E_3(\geq 61)$	$E_3(*)$
$E_3(20)$	$E_3(20\text{-}60)$	$E_3(*)$

(b) Hierarchie

Abbildung 4.21: Verschlüsselte Daten von P_2

Außerdem wurde die Generalisierungshierarchie um die Werte der Pseudotupel erweitert. Nun permutiert P_2 diese Daten, z. B. indem sie die Daten anhand der verschlüsselten TIDs sortiert, und sendet sie an die Hilfspartei. Die anderen Primärparteien verfahren analog. Nun hat die Hilfspartei alle Teildaten erhalten und kann mit der Integration beginnen. Nachdem alle Teildatensätze anhand der TID Spalte integriert wurden, werden die Zählerspalten der jeweiligen Tupel in verschlüsselter Form summiert. Für das entstehende Tupel mit der ID 1 $(E_3(34), E_4(82676), E_2(m))$ wäre das also $E_h(1) \circ_+ E_h(0) \circ_+ E_h(0) = E_h(1)$, dabei stellt $\circ_+$ die Rechenoperationen für die Addition im verschlüsselten Raum dar. Damit haben nun die echten Tupel den verschlüsselten Zählerwert von 1, wohingegen die Pseudotupel den Zählerwert 0 haben.

Mit diesem integrierten Datensatz startet nun die Hilfspartei das Anonymisierungsverfahren. Für das Beispiel wird angenommen, dass die erste Transformation die ausgeführt wird, die Generalisierung $(0, 1, 3)$ ist.

Geschlecht	Alter	PLZ	Zähler
E_2(m)	E_3(20-60)	E_4(82***)	E_h(1)
E_2(w)	E_3(20-60)	E_4(81***)	E_h(1)
E_2(m)	$E_3(\geq 61)$	E_4(81***)	E_h(1)
E_2(m)	$E_3(\geq 61)$	E_4(81***)	E_h(1)
E_2(w)	E_3(20-60)	E_4(81***)	E_h(1)
E_2(m)	E_3(20-60)	E_4(82***)	E_h(1)
E_2(w)	$E_3(\leq 19)$	E_4(82***)	E_h(1)
E_2(w)	$E_3(\leq 19)$	E_4(82***)	E_h(1)
E_2(w)	$E_3(\geq 61)$	E_4(82***)	E_h(0)
E_2(m)	E_3(20-60)	E_4(82***)	E_h(0)

Abbildung 4.22: Verschlüsselter und in den Zustand $(0, 1, 3)$ transformierter Datensatz

Das Ergebnis dieser Transformation ist in Abbildung 4.22 dargestellt. Der nächste Schritt, nach der Transformation, ist die Gruppierung der Tupel, dabei wird normalerweise das Vorkommen der Tupel gezählt. In diesem speziellen Fall werden die Zähler der einzelnen Tupel einer Gruppe addiert. Als Ergebnis erhält die Hilfspartei die in Abbildung 4.23 dargestellten Gruppen. Das verschlüsselte Zähler Attribut enthält nun die wahre Anzahl der in der Gruppe enthaltenen, echten Tupel.

Gruppe	Geschlecht	Alter	PLZ	Zähler
1	E_2(m)	E_3(20-60)	E_4(82***)	E_h(2)
2	E_2(w)	E_3(20-60)	E_4(81***)	E_h(2)
3	E_2(m)	$E_3(\geq 61)$	E_4(81***)	E_h(2)
4	E_2(w)	$E_3(\leq 19)$	E_4(82***)	E_h(2)
5	E_2(w)	$E_3(\geq 61)$	E_4(82***)	E_h(0)

Abbildung 4.23: Gruppen der Transformation $(0, 1, 3)$

Im Beispiel kann man sehen, dass, aus der Sicht der Hilfspartei, in der ersten Gruppe drei Tupel sind, obwohl nur zwei echte Tupel enthalten sind. Somit kann die Hilfspartei nicht feststellen, wie viele Tupel wirklich in eine Gruppe fallen. Nach der Gruppierung sendet nun die Hilfspartei das Zählerattribut, nachdem sie es permutiert hat, an die Primärpartei, die den privaten Schlüssel hält. Im Beispiel also wird das Tupel ($E_h(2)$,$E_h(2)$,$E_h(0)$,$E_h(2)$,$E_h(2)$) an die Primärpartei gesendet. Sie kann nun die Werte entschlüsseln und prüfen, ob alle Gruppen das Anonymitätskriterium erfüllen (im Beispiel k=2) und das Ergebnis an die Hilfspartei zurücksenden. In diesem Fall würde die Hilfspartei zurückgemeldet bekommen, dass die Transformation anonym ist, da alle Gruppen entweder leer sind und somit nur unechte Tupel enthalten, oder mindestens zwei Tupel enthalten. Außerdem ist im Beispiel diese Transformation auch das Endergebnis. Somit kann nun die Hilfspartei die Daten, zusammen mit den ursprünglichen Zählern (siehe Abbildung 4.22), an die Primärpartei senden, welche die Daten entschlüsselt, die unechten Tupel entfernen kann und danach eine anonyme Transformation der Daten hält.

4.4.5.3 Beispiel: horizontal verteilt

Das Setup im horizontalen Fall besteht auch wieder aus den drei Primärparteien P_1, P_2 und P_3 und einer Hilfspartei P_4, die keine eigenen Daten beiträgt. Die Verteilung der Daten auf die Primärparteien ist wieder dieselbe wie in Abschnitt 4.4.3.3 Abbildung 4.11. Auch die Generalisierungshierarchien bleiben dieselben wie beim horizontalen Beispiel von Protokoll A. Als Erstes einigen sich wieder die drei Primärparteien auf einen gemeinsamen Schlüssel je Spalte. Eine Primärpartei (im Beispiel P_1) generiert das Schlüsselpaar für das additiv homomorphe Verschlüsselungsverfahren. Den privaten Schlüssel behält P_1 und der öffentliche Schlüssel wird an die anderen beiden Primärparteien und die Hilfspartei verteilt. Die Schlüssel werden wieder mit einem Index, wie folgt, bezeichnet: Geschlecht=2, Alter=3 und Postleitzahl=4. $E_3(45)$ bedeutet, dass das Alter 45 mit dem Schlüssel für Attribut 3 symmetrisch verschlüsselt wird. Fokus im ersten Schritt des Protokolls ist wieder die Partei P_1.

Als Erstes wird nun P_1 beliebige Pseudotupel, im Beispiel ein einziges, erzeugen, um die echte Verteilung zu maskieren. Das Schema der Pseudotupel folgt dem Schema der echten. Als nächsten Schritt erhalten die echten und die Pseudotupel wiederum einen Zähler. Jedes echte Tupel erhält den Zählerwert 1, die Pseudotupel erhalten den Zählerwert 0. Nun werden die Tupel zusammengeführt und permutiert. Als Nächstes werden die Tupel mit den jeweiligen Schlüsseln für die Spalten verschlüsselt. Die Zählerspalte wird mit dem additiv homomorphen Verfahren und dem dazugehörigen, öffentlichen Schlüssel verschlüsselt. Die Teilgeneralisierungshierarchien werden mit dem zur jeweiligen Spalte gehörenden Schlüssel verschlüsselt, nachdem zuvor alle notwendigen, neuen Regeln für die Generalisierung der Pseudotupel, eingefügt wurden. Das Ergebnis dieser Schritte ist in Abbildung 4.24 dargestellt.

Geschl.	Alter	PLZ	Zähler
E_2(m)	E_3(34)	E_4(82667)	E_h(1)
E_2(w)	E_3(45)	E_4(81775)	E_h(1)
E_2(m)	E_3(66)	E_4(81925)	E_h(1)
E_2(m)	E_3(20)	E_4(82543)	E_h(0)

(a) Verschlüsselte Daten

Lvl-0	Lvl-1	Lvl-2
E_3(20)	E_3(20-60)	E_3(*)
E_3(34)	E_3(20-60)	E_3(*)
E_3(45)	E_3(20-60)	E_3(*)
E_3(66)	E_3(≥61)	E_3(*)

(b) Verschl. Gen. Hier. Alter

Abbildung 4.24: Verschlüsselter Teildatensatz von P_1

Die anderen Teilgeneralisierungshierarchien der Partei P_1 werden analog erstellt und zusammen mit den Teildaten an die Hilfspartei gesendet. Die anderen Parteien verhalten sich genau so und senden ihre verschlüsselten Daten und Hierarchien an die Hilfspartei. Diese kann nun, wie in Protokoll B, die Daten und die Teilhierarchien integrieren. Als Nächstes wird die Hilfspartei die integrierten Daten transformieren. Dies läuft nun genauso wie im vertikalen Fall ab. Als Erstes transformiert die Hilfspartei die Daten und gruppiert diese anschließend nach den Quasi-Identifikatoren. Dabei werden wieder die verschlüsselten Zählerwerte aller Tupel, die in dieselbe Gruppe fallen, im verschlüsselten Raum addiert. Diese Zählerwerte werden nun wieder an P_1 gesendet. P_1 kann die Werte entschlüsseln und prüfen, ob das Anony-

mitätskriterium bei dieser Transformation erfüllt ist. Wenn die gesuchte anonyme Transformation gefunden wurde, sendet wiederum P_4 die transformierten Daten, zusammen mit der verschlüsselten Zählerspalte, an die Partei P_1. Diese kann nun die Werte und die Zählerspalte entschlüsseln und die Pseudotupel (alle Tupel mit dem Wert 0) entfernen. Danach hält sie das 2-anonyme Endergebnis.

4.4.5.4 Analyse der Komplexität

Da bei Protokoll C eine Kommunikation und kryptographische Operationen während der Anonymisierungsphase notwendig sind, ist die Analyse von mehr Parametern abhängig wie bei Protokoll A und B. Als Erstes wird die Menge an Tupeln im Protokoll, durch das Einfügen von Pseudotupeln, erhöht. Die Gesamtzahl der eingefügten Pseudotupel wird mit r_p bezeichnet. Auch kann sich, beim Einfügen, die Anzahl der distinkten Werte ändern. Vereinfachend wird aber im Folgenden davon ausgegangen, dass diese gleich bleiben. Weitere Parameter, um die Komplexität abschätzen zu können, sind bei Protokoll C, neben denen, die auch bei Protokoll B Verwendung finden, die Anzahl der benötigten Tests, um eine anonyme Lösung zu finden und die Anzahl der Gruppen für jeden dieser Tests. Diese Werte sind nur schwer abschätzbar, da sie, sowohl von den Daten als auch von dem Anonymitätskriterium abhängen. Für die folgende Analyse wird davon ausgegangen, dass die Anzahl der benötigten Tests mit t bezeichnet wird. Die durchschnittliche Größe der Gruppen, je Test, wird mit g bezeichnet. Die Anzahl der additiv homomorphen kryptographischen Operationen pro Sekunde wird mit M_h bezeichnet und ist der Durchschnitt aus Entschlüsselung, Verschlüsselung und homomorphen Rechenoperationen. Mit L_h wird die Länge eines Zellwertes bezeichnet, welcher additiv homomorph verschlüsselt wurde. Die restlichen Parameter folgen dem Benennungsschema von Protokoll B (siehe Abschnitt 4.4.4.4).

4.4.5.4.1 Kryptographische Operationen Grundlage für die Arbeitslast bei der Verschlüsselung ist die benötigte Zeit für die additiv homomorphe Verschlüsselung der Zählerspalte. Diese ist wieder parallelisiert über alle Primärparteien. Da diese Verschlüsselung wesentlich länger dauert als die benötigten symmetrischen Operationen, liegt der Fokus im Folgenden nur auf dieser. Im vertikalen Fall muss jede Partei, neben der symmetrischen Verschlüsselung der distinkten Werte und der TID-Spalte, auch noch $e_v = r + r_p$ mal die Zählerspalte verschlüsselt werden. Im horizontalen Fall muss jede Partei nur $e_h = (r + r_p)/n$ Zellwerte additiv homomorph verschlüsseln. Die Entschlüsselung ist bei beiden Fällen gleich und kann mit $d_{vh} = r + r_p$ abgeschätzt werden, da eine Primärpartei die Zählerspalten entschlüsseln muss, um die Pseudotupel herauszufiltern. In der Anonymisierungsphase kann man die Anzahl der kryptographischen Operationen abschätzen mit $a_{vh} = t * 2 * g$. Einmal muss die Hilfspartei, bei der Gruppierung die Zählerspalte, im verschlüsselten Raum addieren und einmal muss eine Primärpartei die Zähler der einzelnen Gruppen entschlüsseln und beides muss „Anzahl-Test“ mal ausgeführt werden. Somit kann die Arbeitslast im vertikalen Fall wieder abgeschätzt werden mit $t_v^c = (e_v + d_{vh} + a_{vh}) \cdot M_h$ und im horizontalen mit $t_h^c = (e_h + d_{vh} + a_{vh}) \cdot M_h$

4.4.5.4.2 Ausgetauschtes Datenvolumen Auch bei Protokoll C folgt der Nachrichtenaustausch einem Sternmuster. Es werden also während der Verschlüsselungsphase n Nachrichten ausgetauscht, da alle Parteien ihre Teildaten an die Hilfs-

partei senden. Während der Entschlüsselungsphase wird wieder nur eine Nachricht an eine Primärpartei gesendet. Während der Anonymisierungsphase werden bei jedem Test die Informationen über die Gruppen versendet, es werden also t Nachrichten versandt. Auch hier werden wieder, zur Vereinfachung der Analyse, die Generalisierungshierarchien vernachlässigt. Während der Integrationsphase werden keine Daten versendet. Die Größe einer Nachricht im vertikalen Fall und in der Verschlüsselungsphase kann nun abschätzt werden mit $s_v^1 = (\frac{c}{n}) \cdot L_a + (\frac{d}{n}) \cdot L_s + (r + r_p) \cdot L_h$, da jede Partei $\frac{c}{n}$ Spalten hat und zusätzlich eine Spalte mit Tupel-Identifikatoren. Der zweite Term ergibt sich aus den Wörterbüchern, da jede Partei $\frac{d}{n}$ distinkte Werte hat. Der letzte Term bezeichnet die Zählerspalte, die jedem Tupel hinzugefügt wird und die Länge L_h besitzt. Im horizontalen Fall kann die Größe einer Nachricht während der ersten Phase mit $s_h^1 = \frac{r+r_p}{n} \cdot L_a + d \cdot L_s + \frac{r+r_p}{n} \cdot L_h$ abgeschätzt werden. Die Größe der Nachricht, während der Entschlüsselungsphase, kann mit $s_{vh}^4 = c \cdot L_a + d \cdot L_s + (r + r_p) \cdot L_h$ abgeschätzt werden. Die Größe einer Nachricht, während der Anonymisierungsphase, kann abgeschätzt werden mit $s_{vh}^3 = g \cdot L_h$, da jede Gruppe, im einfachsten Fall, als verschlüsselter Zähler dargestellt werden kann. Damit kann das Datenvolumen im vertikalen Fall abgeschätzt werden mit $d_v = n \cdot s_v^1 + t \cdot s_{vh}^3 + s_{vh}^4$ und im horizontalen mit $d_h = n \cdot s_h^1 + t \cdot s_{vh}^3 + s_{vh}^4$. Die Laufzeit kann damit abschätzt werden als $t_h^d = d_h \cdot M_b$ und als $t_v^d = d_v \cdot M_b$.

4.4.5.5 Sicherheit

Bei Protokoll A und B kann jeweils eine Partei die Verteilung der Daten, wenn auch nur im verschlüsselten Zustand, lernen. Diese Schwachstelle wird beim Protokoll C geschlossen. Da die Parteien hierbei beliebige Pseudotupel einfügen können, kann die Verteilung maskiert werden. Hier besteht im Allgemeinen ein Performanz-, Sicherheits-Trade-off. Je mehr Pseudotupel hinzugefügt werden, umso besser wird die echte Verteilung maskiert, allerdings erhöht dies die Datenmenge und die benötigte Zeit für das Protokoll. Da es nun mehr Tupel pro Partei sind, dauert die Verschlüsselung länger, die Dauer der Transformationen und des Gruppierens der Hilfspartei nimmt durch die erhöhte Anzahl der Tupel zu. Da die Überprüfung auf Anonymität nur anhand der Zählerwerte von einer Primärpartei vorgenommen wird, lernt einzig die Hilfspartei, ob eine gewisse Transformation anonym ist oder nicht. Allerdings kennt die Hilfspartei, wegen der Pseudotupel, nicht die genaue Verteilung der Tupel, welche in eine Gruppe fallen. Allerdings kann etwas mehr Information bekannt werden, da die Hilfspartei weiß, wie viele Tupel (echte und Pseudotupel) in eine Gruppe fallen. z. B., wenn auf 2-Anonymität ohne Unterdrückung getestet wird, in eine Gruppe nur ein Tupel fällt und die Primärpartei die Transformation dennoch als anonym bezeichnet, kann daraus die Hilfspartei lernen, dass dieses Tupel ein Pseudotupel sein muss. Dies kann, durch Einfügen von passenden Pseudotupeln, verhindert werden.

Ein weiterer Nachteil des Protokolls ist, dass für jeden Test auf Anonymität eine Kommunikation notwendig ist, welche Zeit kostet. Somit ist das Protokoll generell langsamer als z. B. Protokoll B. Man sollte hier allerdings anmerken, dass die Anzahl der benötigten Anonymitätstests unabhängig von der Anzahl der eingefügten Pseudotupel sind.

4.5 Implementierungsdetails

Bei der Implementierung der Protokolle wurden einige Optimierungen vorgenommen, um die Laufzeit und die zu übertragende Datenmenge zu reduzieren. Hierzu wurden hoch-optimierte Bibliotheken, Komprimierungstechniken, Parallelisierung und State-of-the-Art kryptographische Verfahren eingesetzt. Der Prototyp wurde mit Java implementiert.

4.5.1 Verschlüsselungsalgorithmen

In den Protokollen werden dieselben Schlüssel verwendet, um verschiedene Klartexte zu verschlüsseln. Bei der Wahl eines passenden Verschlüsselungsalgorithmus ist es deshalb notwendig, dass in so einem Szenario das Verfahren nicht anfällig gegenüber Angriffen ist. Würde beispielsweise eine einfache XOR Verschlüsselung genutzt, könnten Teile des Schlüssels rekonstruiert werden, bei einem bekannten Klartext (known-plaintext-attack). Für die in den Protokollen verwendeten Verfahren gelten dieselben Anforderungen, wie in [107] beschrieben. Die Wahl für den Prototyp fiel, bei der kommutativen Verschlüsselung, auf das Pohlig-Hellman Verfahren (siehe Abschnitt 16.2 in [108]) und bei der additiv homomorphen Verschlüsselung auf das Paillier Verfahren [109]. Als symmetrisches Verschlüsselungsverfahren kam AES mit 128-Bit im CBC Mode mit PKCS7 Padding zum Einsatz (AES/CBC/PKCS7Padding).

4.5.2 Hashing der Attributwerte

Im Protokoll A wird kommutative Verschlüsselung eingesetzt. Das bekannteste Verfahren ist das Pohlig-Hellman Verfahren, welches auf Exponentiation modulo einer Primzahl basiert. Bei diesem Verfahren sollte darauf geachtet werden, dass nicht mit dem neutralen Element Eins und der Null gerechnet wird. Um dieses Problem zu umgehen, wird ein Vorverarbeitungsschritt umgesetzt. Bei diesem wird von jeder Partei ein geheimes Hashwörterbuch aufgebaut, d. h. auf jeden Zellwert, sowohl der zu generalisierenden Daten als auch den Werten der Generalisierungshierarchien, wird die Hashfunktion angewendet und in einer Zuordnungstabelle die Zuordnung zwischen Hash und Originalwert gehalten, um diesen Prozess, am Ende des Protokollablaufes, wieder revidieren zu können. Dies entspricht zwar schon einer simplen Substitutionsverschlüsselung, diese ist aber anfällig für brute-force- oder Ausprobier- Angriffe und stellt somit keinen genügenden Schutz dar. Weiterhin berechnet jede der teilnehmenden Parteien denselben Hashwert für identische Werte und kann somit, wenn sie dieselben Werte wie eine andere Partei besitzt, mit ihrem Wörterbuch die Substitutionsverschlüsselung wieder rückgängig machen. Zusätzlich wird durch das Anwenden der Hashfunktion auch sichergestellt, dass die Plaintextlänge bei allen Feldern konstant ist und somit die Verschlüsselung mit nur einer Anwendung des Verfahrens möglich ist. Das Wörterbuch bleibt geheim und ist nur der generierenden Partei bekannt. In der Implementierung wird hierbei als Hashverfahren SHA-1 mit 160Bit eingesetzt. Die Codierung wird transparent bei der Verschlüsselungsphase durchgeführt. Die Decodierung wird mittels einer sequenziellen, zusätzlichen Phase am Ende des Protokolls durchgeführt, hierbei ersetzt jede Partei die Hashwerte, die sie kennt, mit den Plaintextwerten aus ihrem Wörterbuch. Der daraus resultierende, leicht erweiterte Protokollfluss für Protokoll A ist

in Abbildung 4.25 dargestellt.

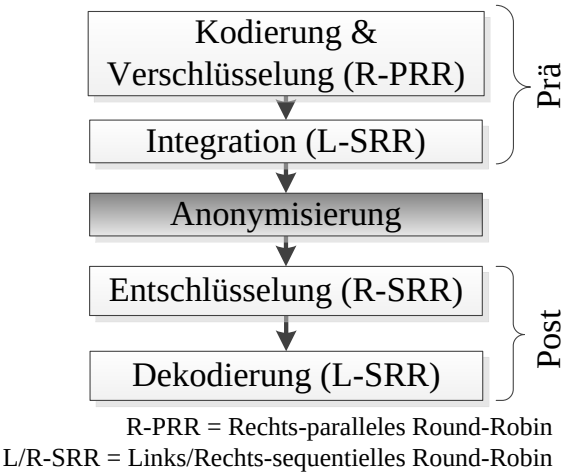


Abbildung 4.25: Die Phasen des implementierten Prototyps für Protokoll A

4.5.3 Implementierung mittels GMP

Da die kryptographischen Operationen bei den Protokollen viel Zeit beanspruchen, wurde ein starker Fokus auf effiziente Implementierung gelegt. Insbesondere der kommutative Verschlüsselungsalgorithmus bei Protokoll A wurde optimiert, da die Laufzeit der JAVA Implementierung der BigInteger Arithmetik, vor allem die modulare Exponentiation, in ersten Tests sehr langsam war. Um dieses Problem zu reduzieren, wurde auf Funktionen der *GNU Multiple Precision Arithmetic Library* (GMP) [110] zurückgegriffen. Es wurde mittels Java Native Interface (JNI) [111] eine Verbindung zwischen der Java Implementierung und der GMP Library geschaffen. Um dennoch ein gewisses Maß an Plattformunabhängigkeit zu erreichen, wurde auf die im I2P Projekt [112] genutzte Idee zurückgegriffen, die für verschiedene Plattformen speziell vorkompilierte Bibliotheken mitbringt, die je nach System automatisch geladen werden. Sollte keine passende Bibliothek gefunden werden, wird automatisch auf die native JAVA Funktionen zurückgegriffen. Sowohl für die Implementierung des klassischen Pohlig-Hellman Verfahrens, als auch für die ECC-Version wurde eine effiziente Implementierung mittels der GMP Bibliothek realisiert. Dazu wurde die modulare Exponentiation an die GMP-Bibliothek übergeben. Außerdem wurde auch die skalare Multiplikation von elliptischen Kurvenpunkten komplett mittels GMP implementiert. Dazu wurde die *Non-Adjacent Form* Methode verwendet, da die GMP keine native Unterstützung von „skalarer Multiplikation“ für elliptische Kurven mitbringt. Auch das additiv homomorphe Paillier Verfahren für Protokoll C wurde mittels GMP beschleunigt. Für die symmetrische AES Verschlüsselung wurde die BouncyCastle Bibliothek [113] genutzt.

4.5.4 Komprimierung der zu übertragenden Daten

Für den Datenaustausch wird in dem Prototyp eine Wörterbuchkomprimierung verwendet. Die Daten werden als zwei-dimensionales Array, von Strukturinformationen und dazugehörigen Wörterbüchern, repräsentiert. Ein Beispiel ist in Abbildung 4.26 gezeigt, hierbei wird jeder Eintrag im Wörterbuch mit seinem Zeilenindex referenziert.

Geschl.	Alter	PLZ
m	34	82676
w	45	81775
m	66	81925
...	...	...

(a) Daten

⇒

Geschl.	Alter	PLZ
0	0	0
1	1	1
0	2	2
...	...	...

(b) Struktur

Geschl.
m
w

(c) Dict_1

Alter
34
45
66
...

(d) Dict_2

PLZ
82676
81775
81925
...

(e) Dict_3

Abbildung 4.26: Wörterbuchkomprimierung

Ein weiterer Vorteil der Wörterbuchcodierung ist, dass nur die Wörterbucheinträge verschlüsselt werden müssen und somit pro Spalte, jeder unterschiedliche Wert, nur einmal verschlüsselt werden muss. Für eine weitere Komprimierung, insbesondere der Strukturinformationen, wurde außerdem ein schneller Kompressor, Snappy [114] (ursprünglich von Google entwickelt), eingesetzt. Alle Nachrichten werden, vor dem versenden, mit diesem Kompressor komprimiert.

4.5.5 Parallelisierung

Neben der Implementierung mittels der schnellen GMP Bibliothek kann die Implementierung von der Parallelität moderner Prozessoren profitieren. Da die Verschlüsselungen der einzelnen Zellen vollständig unabhängig voneinander sind, kann dabei die gesamte Arbeitslast gleichmäßig auf die vorhandenen Prozessoren aufgeteilt werden. Damit kann fast ein idealer Speedup erreicht werden und die Implementierung skaliert sehr gut mit der Anzahl vorhandener Prozessoren. Für die parallelisierte Version wurde die Anzahl der Threads mit der Anzahl der Kerne initialisiert, also waren pro Rechner in dem hier verwendeten Testbed vier Threads gleichzeitig beschäftigt, die Daten zu ver-/entschlüsseln.

Bei Protokoll C wurde außerdem der Test auf Anonymität zwischen den Primärparteien parallelisiert. Damit muss für jeden Test jede Primärpartei nur noch einen Teil der Gruppen auf Anonymität prüfen, was bedeutet, dass die Entschlüsselung der Gruppenzähler auch parallel vorgenommen werden kann.

4.5.6 Elliptische Kurven

Bei den vorgestellten Protokollen sind die teuersten Operationen die Verschlüsselungen. Insbesondere ist dies die kommutative Verschlüsselung (Pohlig-Hellman), da sie auf modular-exponentiation mit großen Zahlen beruhen. Die, für eine sichere Ausführung benötigten, Bitlängen sind vergleichbar mit RSA. Daher sollten sie sich zwischen 1024 und 4096-Bit bewegen, hierbei sollte die untere Grenze heutzutage nicht mehr unterschritten werden (siehe z. B. [115]), die obere Grenze aber auf Kosten der Rechenzeit natürlich erhöht werden kann. Eine Verbesserung ist die Anwendung von elliptischen Kurven. Elliptische-Kurven-Kryptographie ist ein asymmetrisches Kryptoverfahren, basierend auf elliptischen Kurven über einem endlichen Körper. Kryptoverfahren, die auf dem diskreten Logarithmus-Problem basieren, können auf elliptische Kurven übertragen werden. Der Vorteil der elliptischen Kurven, im Gegensatz zu den Standardverfahren, ist die kürzere Schlüssellänge, bei vergleichbarem Sicherheitsniveau. Damit werden die Berechnungen schneller, obwohl die Anzahl der Operationen, im Vergleich zu Standardverfahren, zunimmt. In 4.27 ist ein Vergleich der vom NIST empfohlenen Schlüssellängen zwischen RSA

und elliptischen Kurven dargestellt.

RSA (Bits)	Elliptische Kurven (Bits)
1024	160
2048	224
3072	256
7680	384
15360	521

Abbildung 4.27: NIST: Vergleichbare Schlüssellängen [116]

Wegen des Geschwindigkeitsvorteils und der (durch die geringere Schlüssellänge) verringerten Nachrichtenlänge, wurde auch eine Version des Pohlig-Hellman Verfahrens mittel elliptischen Kurven implementiert. Die Implementierung beruht auf der in [108] dargestellten Version des EC-Pohlig-Hellman Verfahrens. Um die zu verschlüsselnden Daten auf Punkte der elliptischen Kurve abzubilden, wurde das probabilistische Verfahren wie in [117] beschrieben, verwendet. Für die Wahl der Bitlänge wurde in den Tests die aktuell minimale, als sicher geltende Länge gewählt. Als elliptische Kurve wurde die kleinste von BouncyCastle[113] unterstütze Kurve mit dem Namen prime192v1, standardisiert in ANSI X9.62 [118], mit 192-Bits genutzt. Diese Kurve ist über F_p definiert. Die minimal, aktuell sichere Schlüssellänge nach NIST Empfehlungen ist, für auf diskreten Logarithmus beruhende Verfahren, aktuell 1024-Bit. Um die Vergleichbarkeit hinsichtlich des Sicherheitsniveaus mit exponential Pohlig-Hellman zu erhalten, wurde für die Tests die Bitlänge auf 1536 Bits erhöht.

Das Paillier Verfahren kann theoretisch auch mit elliptischen Kurven realisiert werden, was allerdings laut [119] aktuell keinen Geschwindigkeitsvorteil bringt. Aus diesem Grund wurden die Experimente für Protokoll C mit dem klassischen Paillier Verfahren durchgeführt.

4.6 Evaluation der Protokolle für verteilte Datenbestände

In diesem Abschnitt werden, mittels der schon in 3.9 verwendeten Datensätze, die Wirksamkeit der verschiedenen Optimierungen, die Laufzeit und der Speicherverbrauch der Protokolle, getestet. Die Evaluation wird analytisch und experimentell vorgenommen. Es wird angenommen, dass drei Parteien an den Protokollen teilnehmen, im Falle von Protokoll B und C gibt es eine zusätzliche, externe Partei, welche die Anonymisierung übernimmt.

4.6.1 Verteilung der Datensätze

Für die Evaluation werden die gleichen Datensätze wie in Abschnitt 3.9 verwendet, dort sind auch weitere Details zu den Datensätzen zu finden. Es sind fünf Datensätze mit 8 bzw. 9 Spalten und 30.162-1.193.504 Zeilen. Die Daten werden beim vertikalen Fall, wie in Tabelle 4.2 dargestellt, auf die drei Parteien aufgeteilt. Bei den Datensätzen mit neun Spalten bekommt jede Partei drei Spalten, bei den Datensätzen mit nur acht Spalten, bekommt die letzte Partei nur zwei Spalten. Im

Horizontalen Fall werden die Daten gleichmäßig, zeilenweise auf die drei Parteien verteilt, auch hier bekommt die letzte Partei die überschüssigen Zeilen.

Datensatz	P_1	P_2	P_3
ADULT	age race sex	education martial-status native-country	occupation salary-class workclass
CUP	age gender zip	income ramntall state	minramnt ngiftall
FARS	iage ideathmon irace	ideathday ihispanic isex	iinjury istatenum
ATUS	age region sex	citizenship status martial status race	birthplace highest level of school completed labor force status
IHIS	quarter region year	age marstat pernum	educ racea sex

Tabelle 4.2: Vertikale Verteilung der Testdaten auf die drei Parteien

4.6.2 Analytische Evaluation

In diesem Abschnitt werden die Protokolle analytisch evaluiert. Dazu werden Schätzwerte für die Schlüsselvariablen angenommen und mit den oben vorgestellten Komplexitätsabschätzungen Gesamtlaufzeiten berechnet. Für die Evaluation wird angenommen, dass die Maschinen ca. $M_c = 3000$ EC-Pohlig-Hellman Operationen pro Sekunde ausführen können. Die Ausführungsgeschwindigkeit für symmetrische AES Verschlüsselung liegt bei $M_s = 3000000$ Operationen pro Sekunde und für das additiv homomorphe Paillier Verfahren bei $M_s = 3240$ Operationen pro Sekunde.

Die Bandbreite des Netzwerkes beträgt ca. $M_b = 88000000$ Bits/s (Fast Ethernet). Jeder Eintrag des strukturellen Arrays hat $L_a = 32$ Bit (Java Integer). Jeder kommutativ verschlüsselte Eintrag hat eine Größe von $L_c = 192$ Bit. Symmetrisch verschlüsselte Einträge haben die Länge $L_c = 128$ Bit und jeder additiv homomorph verschlüsselte Eintrag hat eine Größe von $L_c = 768$ Bit. Für Protokoll A werden die analytisch berechneten Zeiten mit den Zeiten verglichen, welche die in Abschnitt 3.7 vorgestellten Algorithmen benötigen, um die Daten zu anonymisieren. Als Kriterien wurden 5-Anonymität, rekursive-(4,3)-Diversität, 0.2-Closeness mit hierarchischer Earth-Mover's-Distance und (0,05;0,15)-Präsenz mit einer 10 % Teilmenge als Forschungsteilmenge angenommen. t-Closeness und δ-Präsenz wurden mit 5-Anonymität kombiniert. Die angegebenen Zeiten stammen von dem in Kapitel 3 vorgestellten Flash Algorithmus, der die optimale Anonymisierung findet, da auch solch ein Algorithmus in den hier vorgestellten Protokollen anwendbar ist und weiter unten angewendet wurde. Die große Anzahl an Privacy Kriterien soll die Flexibilität des Ansatzes zeigen. Die Wahl $k = 5$ wurde getroffen, da dies die

ungefähre, obere Grenze der Ausführungszeit darstellt, d. h., mit größerem k sinkt die Ausführungszeit, und weil dieser Wert häufig in der biomedizinischen Forschung verwendet wird. Alle Anonymisierungen wurden außerdem mit 3 % Unterdrückung durchgeführt.

4.6.2.1 Protokoll A

Die Tabelle 4.3 stellt die Ausführungszeiten für das Protokoll für zwei Parteien (P2) und für drei Parteien (P3) dar. Wie in Abschnitt 4.4.3.4 dargestellt, hängen diese Zeiten nur von der Anzahl der teilnehmenden Parteien und grundlegenden, statistischen Eckwerten der Eingabedaten ab. Aus der Tabelle wird ersichtlich, dass, im Falle der vertikalen Verteilung, die Ausführungszeiten für Prä- und Postprocessing Phase ungefähr um eine Größenordnung höher sind, als die Zeiten der Anonymisierungsphase. Außerdem ist ersichtlich, dass die Zeiten für zwei Parteien niedriger sind als für drei Parteien. Dies resultiert aus der größeren Anzahl von Tupel-Identifikatoren (jede Partei hat einen eigenen Satz an Identifikatoren), die verschlüsselt werden müssen. Die Laufzeit der verschiedenen Datensätze wächst linear mit der Anzahl der Zeilen. Im horizontalen Fall ist die Ausführungszeit für Prä- und Postprocessing nahezu identisch mit den Zeiten, die für die Anonymisierung benötigt werden. Hierbei wird die Ausführungszeit dominiert von der Anzahl der unterschiedlichen (distinkten) Werte im Datensatz. Dies lässt sich schön beim CUP Datensatz sehen, der, durch seine große Anzahl an unterschiedlichen Werten, bei der Prä- und Postprocessing Phase, ungefähr eine Größenordnung langsamer ist als die Anonymisierung. Generell führt das Protokoll A zu einem merklichen Overhead im vertikalen Fall und zu einem geringen im horizontalen Fall.

Datensatz	P2 - vertikal	P3 - vertikal	P2 - horizontal	P3 - horizontal	k-Anonymität	ℓ-Diversität	t-Closeness	δ-Präsenz
ADULT	21	32	1	1	1	1	1	1
CUP	58	85	20	31	1	3	6	1
FARS	69	105	1	3	3	3	3	1
ATUS	369	560	7	13	41	25	18	3
IHIS	816	1239	14	28	92	119	101	39

Tabelle 4.3: Geschätzte Zeiten für Prä- und Postprocessing für Protokoll A [s] vs. gemessene Zeiten für die Anonymisierung [s]

4.6.2.2 Protokoll B

Die Tabelle 4.4 stellt die Ausführungszeiten für das Protokoll B für zwei Primärparteien (P2) und für drei Primärparteien (P3) dar. Wie in Abschnitt 4.4.4.4 dargestellt, hängen auch bei Protokoll B diese Zeiten nur von der Anzahl der teilnehmenden Parteien und grundlegenden, statistischen Eckwerten der Eingabedaten ab. Man kann im Vergleich zu den Schätzungen bei Protokoll A sehen, dass die Zeiten wesentlich geringer ausfallen. Auch hier ist wieder ersichtlich, dass die Zeiten für zwei Primärparteien niedriger sind als für drei Primärparteien, was wieder an den Tupel-Identifikatoren liegt. Im Vergleich mit den Zeiten, die für die Anonymisierung benötigt werden (siehe Tabelle 4.3) kann man sehen, dass Protokoll B fast immer

weniger Zeit braucht, als die Anonymisierung. Ausnahme ist δ-Präsenz, wo das Protokoll B für einige Datensätze etwas langsamer ist. Generell kann man sagen, dass Protokoll B fast keinen Overhead hat, im Vergleich zur Anonymisierung.

Datensatz	P2 - vertikal	P3 - vertikal	P2 - horizontal	P3 - horizontal
ADULT	0,32	0,37	0,20	0,20
CUP	0,66	0,77	0,47	0,49
FARS	0,99	1,17	0,59	0,59
ATUS	5,67	6,65	3,53	3,53
IHIS	12,55	14,72	7,81	7,81

Tabelle 4.4: Geschätzte Zeiten für Prä- und Postprocessing [s] für Protokoll B

4.6.2.3 Protokoll C

Die Tabelle 4.5 stellt wieder die Ausführungszeiten für das Protokoll C, für zwei Primärparteien (P2) und für drei Primärparteien (P3) dar. Die Zahlen wurden mittels der in Abschnitt 4.4.5.4 vorgestellten Formeln berechnet. Da bei Protokoll C eine Kommunikation für jeden Test auf Anonymität notwendig ist, sind die Ausführungszeiten deutlich länger als bei den anderen beiden Protokollen. Dennoch liegen diese Zeiten im praktikablen Rahmen von maximal ca. 30 Minuten. Für diese Schätzwerte wurden, die in Abschnitt 4.6.4.3 experimentell ermittelten Werte für die Anzahl von Anonymitätstest, verwendet (ADULT 169, CUP 40, FARS 135, ATUS 557, IHIS 295). Die Menge der Pseudotupel wurde auf 25 % festgelegt. In diesem Experiment wurde auch die durchschnittliche Gruppengröße je Test berechnet; es ergaben sich folgende Gruppengrößen: ADULT 353 Tupel, CUP 4808, FARS 1464, ATUS 1170 und IHIS 6376.

Datensatz	P2 - vertikal	P3 - vertikal	P2 - horizontal	P3 - horizontal
ADULT	43	37	37	29
CUP	112	93	99	76
FARS	143	124	123	97
ATUS	647	584	539	435
IHIS	1557	1376	1318	1048

Tabelle 4.5: Geschätzte Zeiten für Prä- und Postprocessing [s] für Protokoll C

4.6.3 Versuchsaufbau

Die Tests werden auf vier normalen Desktop Rechnern mit jeweils einer vier-Kern Intel CPU i5 mit 3.1 GHz und 8 GB RAM durchgeführt. Als Betriebssystem wurde ein 64-Bit Linux mit 3.2.0 Kernel genutzt (Ubuntu 12.04 LTS). Java wurde als Implementierungssprache verwendet. Für die Evaluation kam eine Oracle 64-Bit JVM zum Einsatz (1.7.0_51). Die maximale Heapsize wurde auf 6 GB festgelegt (-Xmx6G), um keine Probleme mit zu geringer Heapsi-

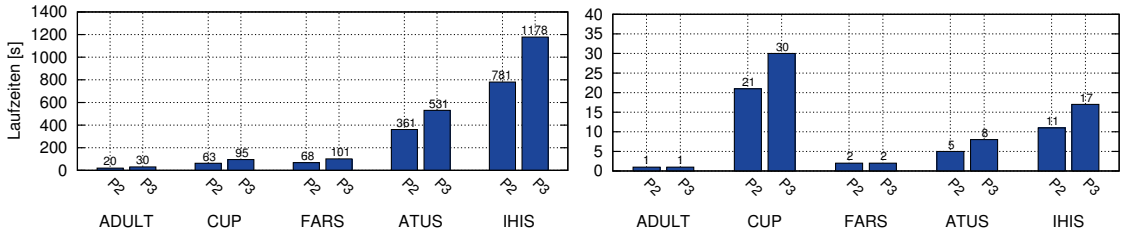
ze zu bekommen. Des Weiteren wurde der parallele Garbage-Collector aktiviert (-XX:+UseConcMarkSweepGC), da sich in den Experimenten zeigte, dass somit weniger Schwankungen in den Laufzeitmessungen auftraten. Um die Ergebnisse noch verlässlicher zu machen, wurde vor dem Lauf einer Konfiguration, ein Warm-up durchgeführt, welcher nicht in die Messungen einging. Die dargestellten Zeiten sind das arithmetische Mittel aus fünf aufeinanderfolgenden Messungen. Sie beinhalten die komplette Laufzeit, exklusive der Zeit für das Einlesen der Daten von Festplatte. Die GMP Bibliothek wurde speziell für die Testrechner kompiliert. Hierzu wurde der gcc in Version 4.6.1 eingesetzt. Die Rechner sind mittels Fast-Ethernet (100 MBit) über einen Switch miteinander verbunden.

4.6.4 Vergleich der Laufzeiten

In diesem Abschnitt werden die absoluten Laufzeiten der beiden Datenverteilungen und jeweils zwei Parteien (Protokoll A: P2, Protokoll B/C: P3) bzw. drei Parteien (Protokoll A: P3, Protokoll B/C: 4P) gegenübergestellt. Diese Zahlen enthalten alle notwendigen Operationen des Prä- und Postprocessing. Die benötigten Zeiten für die Anonymisierung sind, bei den Protokollen A und B, nicht enthalten, da diese exakt den Zeiten im Abschnitt 3.9 entsprechen und für die Gesamtlaufzeit einfach hinzuaddiert werden können. Die Laufzeiten für Protokoll C enthalten die benötigte Zeit für die Anonymisierung. Es werden die gleichen fünf Datensätze, wie im vorherigen Kapitel, für die Evaluation verwendet.

4.6.4.1 Protokoll A

In Abbildung 4.28 sind die Laufzeiten für Protokoll A über alle fünf Datensätze und die beiden Verteilungen (horizontal und vertikal) dargestellt. Man kann sehen, dass die gemessenen Zeiten nur geringe Abweichungen zu den analytisch ermittelten Zeiten aus Abschnitt 4.6.2.1 aufweisen. In 13 von 20 Fällen liegt die Abweichung nur bei 1 bis 2 Sekunden. In vier Fällen liegen die geschätzten Laufzeiten ca. 5 bis 10 Sekunden von den gemessenen entfernt. In drei Fällen liegen die Laufzeiten um 20 bis 50 Sekunden auseinander, allerdings bei Laufzeiten von 10 bis 20 Minuten.



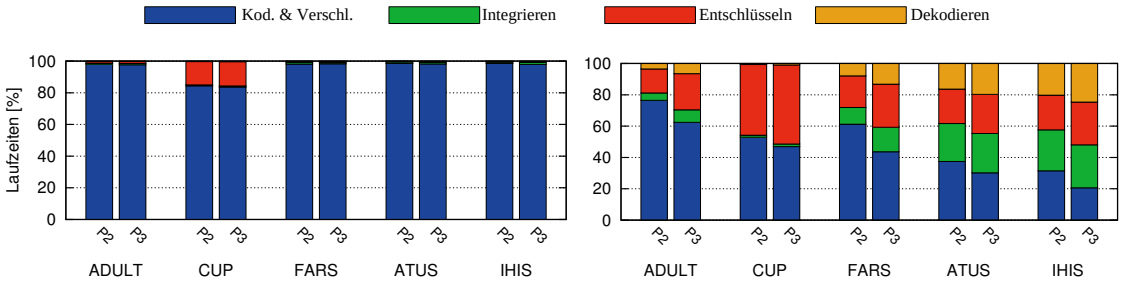
(a) Vertikale Verteilung (b) Horizontale Verteilung

Abbildung 4.28: Protokoll A: Laufzeiten für vertikale und horizontale Verteilungen

Im vertikalen Fall (vgl. Abbildung 4.28a) liegen die Laufzeiten bei zwei Parteien zwischen 20 Sekunden (ADULT) und 13 Minuten (IHIS). Bei drei Parteien nimmt die Laufzeit ca. um den Faktor 1,5 zu. Damit benötigt die Anonymisierung des ADULT Datensatzes 30 Sekunden und für IHIS 19,6 Minuten. Wenn die Datensätze eine konstante Anzahl an distinkten Werten haben, dann nimmt die Laufzeit

konstant mit der Anzahl der Zeilen im Datensatz zu, da in diesem Fall die Verschlüsselung der Tupel-Identifikatoren die Laufzeit bestimmt. Im Vergleich dazu sei auf CUP hingewiesen, der erheblich mehr distinkte Werte aufweist und damit eine unverhältnismäßig längere Laufzeit besitzt (z. B. im Vergleich zu FARS, der fast 2x so viele Zeilen hat, aber ca. genau solange wie CUP für das Protokoll benötigt).

Im horizontalen Fall (vgl. Abbildung 4.28b) sind die Laufzeiten signifikant geringer. Bei dieser Verteilung hängt die Laufzeit maßgeblich von der Anzahl der distinkten Werte im Datensatz ab. Als Ergebnis hat der CUP Datensatz die längste Laufzeit, da dieser die meisten distinkten Werte besitzt. Wenn die Datensätze relativ wenige distinkte Werte haben (z. B. alle anderen Datensätze außer CUP), dann wird die Laufzeit dominiert von der benötigten Zeit, die Daten über das Netzwerk zu übertragen. Die Laufzeiten reichen hierbei von ca. 1 Sekunde für ADULT bei zwei Parteien bis zu 30 Sekunden für CUP bei drei Parteien. Die Laufzeitunterschiede für zwei bzw. drei Parteien liegen zwischen einem Faktor von 1 für ADULT und 0,63 für ATUS.



(a) Vertikale Verteilung (b) Horizontale Verteilung

Abbildung 4.29: Protokoll A: Relative Laufzeiten für verschiedene Verteilungen

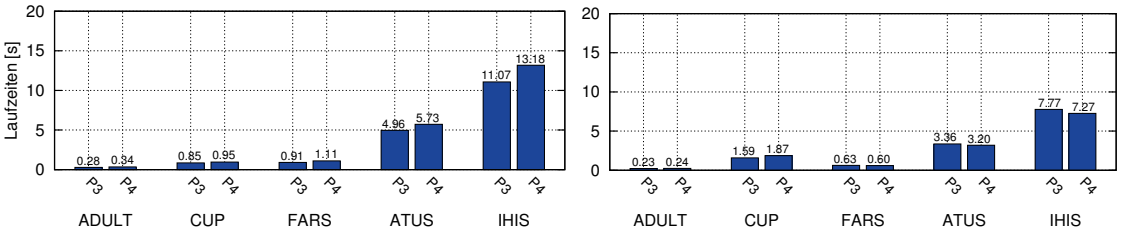
In Abbildung 4.29 sind die Laufzeiten der verschiedenen Phasen des Protokolls relativ zueinander dargestellt. Im vertikalen Fall sind die Laufzeiten dominiert von der Verschlüsselungsphase. Hierbei müssen n Spalten mit eindeutigen Tupel-Identifikatoren verschlüsselt werden. Dies benötigt zwischen 84 % der Laufzeit bei CUP und ca. 98 % bei den anderen Datensätzen. Im horizontalen Fall hingegen hat die benötigte Zeit für die Datenübertragung den größten Einfluss. Dies ist in der Abbildung dadurch reflektiert, dass die Integrations-, Entschlüsselungs- und Decodierungsphasen den größten Anteil am Protokolllauf haben. Auch hier ist der CUP Datensatz die Ausnahme, da dieser, durch seine große Anzahl an unterschiedlichen Werten, mehr Zeit für die Ver- und Entschlüsselung benötigt. Im Vergleich der Ausführungszeiten zwischen zwei und drei Parteien kann man eine kleine Abnahme der relativen Zeit für die Verschlüsselung beobachten, was aus der Zunahme der zu übertragenen Daten bei drei Parteien herrührt (vgl. Abschnitt 4.6.5.1).

4.6.4.2 Protokoll B

In Abbildung 4.30 sind die Laufzeiten für Protokoll B über alle Datensätze und die beiden Verteilungen dargestellt. Die angegebenen Zeiten sind wiederum die Laufzeiten ohne Anonymisierung. Man kann sehen, dass die Laufzeiten für Protokoll B im vertikalen Fall um ca. den Faktor 70 bis 100 geringer sind, als bei Protokoll A. Die meiste Zeit wird durch den Wechsel zur symmetrischen Verschlüsselung gespart, da

diese um mehr als den Faktor 1000 schneller sind. Im horizontalen Fall, da dort weniger Werte verschlüsselt werden müssen, ist das Protokoll B nur noch ca. zwischen 1,4x und 16x schneller als Protokoll A. Man kann auch hier wieder sehen, dass die experimentell gemessenen Zeiten nur geringfügig von den geschätzten Zeiten aus Abschnitt 4.6.2.2 abweichen, der Fehler liegt zwischen 1 % und 16 %. Nur CUP bildet hier eine Ausnahme, hierbei reicht der Fehler bis zu einem Faktor von drei im horizontalen Fall. Dieser Fehler wird zum einen durch die, in den Schätzungen nicht berücksichtigte, Datenkompression verursacht, zum anderen durch die nicht eingerechnete Zeit, die für die Übertragung und Integration der Generalisierungshierarchien benötigt wird. Diese sind im Falle von CUP verhältnismäßig groß, da der Datensatz viele distinkte Werte enthält.

Die absoluten Zeiten im vertikalen Fall liegen somit zwischen 0,28 Sekunden für den ADULT Datensatz und zwei Primärparteien plus Hilfspartei (P3) und ca. 13 Sekunden für IHIS bei drei Primärparteien plus Hilfspartei (P4). Auch bei Protokoll B sticht wieder CUP als Ausnahme heraus. Hier ist die Laufzeit bei CUP, im vertikalen Fall, fast genauso groß wie für den Datensatz FARS, obwohl dieser fast doppelt so viele Zeilen hat wie CUP. Der Grund liegt auch hierbei wieder an der größeren Anzahl von distinkten Werten des CUP Datensatzes, wodurch mehr Verschlüsselungsoperationen nötig sind. Zudem sind, wie oben bereits erwähnt, die Generalisierungshierarchien größer. Der Unterschied zwischen zwei und drei Primärparteien liegt bei ungefähr 12 %-22 % Prozent mehr Laufzeit.



(a) Vertikale Verteilung (b) Horizontale Verteilung

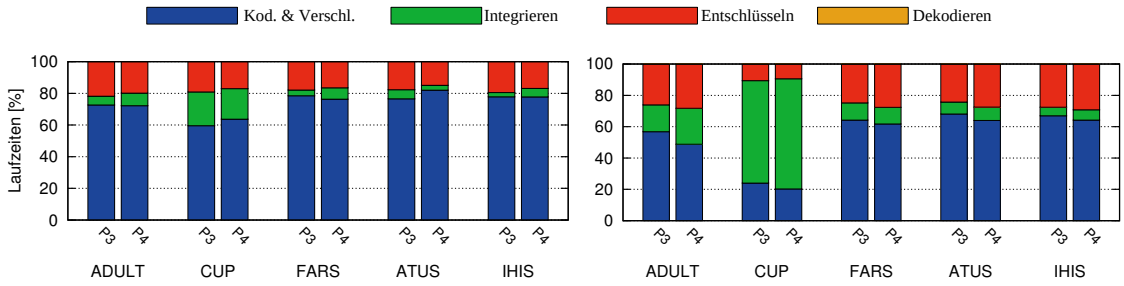
Abbildung 4.30: Protokoll B: Laufzeiten für verschiedene Verteilungen

Die Laufzeiten im horizontalen Fall sind, wie bei Protokoll A, geringer als im vertikalen, da auch hier die zusätzliche Tupel-Identifikator-Spalte wegfällt und damit nicht mehr verschlüsselt werden muss. Hier liegen die Laufzeiten somit zwischen 0,23 Sekunden für ADULT und ca. 8 Sekunden für IHIS mit zwei Primärparteien. Interessant ist hierbei, dass die Laufzeiten für FARS, ATUS und IHIS bei zwei Primärparteien ca. 5 %-7 % höher sind als für drei Primärparteien. Diese Besonderheit kann damit erklärt werden, dass die Verschlüsselung im Falle von zwei Primärparteien nur auf zwei Parteien parallelisiert wird, hingegen im drei Parteien Fall, die gleiche Menge an Daten von drei Parteien verschlüsselt wird. Eine weitere Ausnahme kann wieder beim CUP Datensatz beobachtet werden, wo die Laufzeit im horizontalen Falle höher ausfällt, als im vertikalen. Das kann erklärt werden, wenn man Abbildung 4.31 betrachtet.

Man kann hierbei feststellen, dass die meiste Zeit für die Integrationsphase aufgewendet werden muss. Dies liegt daran, dass hierbei die Generalisierungshierarchien an die Hilfspartei geschickt werden. Da viele Regeln im horizontalen Fall, bei den

verschiedenen Parteien identisch sind, wird hierbei eine größere Datenmenge an die Hilfspartei übertragen als im vertikalen Fall. Diese gleichen Generalisierungsregeln müssen danach noch in der Integrationsphase eliminiert werden. Im vertikalen Fall ist die Integration von Teilgeneralisierungshierarchien nicht notwendig.

In Abbildung 4.31 sind die Laufzeiten der verschiedenen Phasen des Protokolls relativ zueinander dargestellt. Man kann sehen, dass die meiste Zeit (ca. 60 %-80 %) im vertikalen Fall für die Verschlüsselungsphase verwendet werden muss. Hierbei sind in der Verschlüsselungsphase auch das für die Integration im vertikalen Falle notwendige Sortieren der Daten und das Übertragen an die Hilfspartei berücksichtigt. Im horizontalen Fall müssen die Daten zum einen nicht sortiert werden und zum anderen werden keine Tupel-Identifikatoren verschlüsselt, somit ist die Verschlüsselungsphase kürzer.



(a) Vertikale Verteilung (b) Horizontale Verteilung

Abbildung 4.31: Protokoll B: Relative Laufzeiten für verschiedene Verteilungen

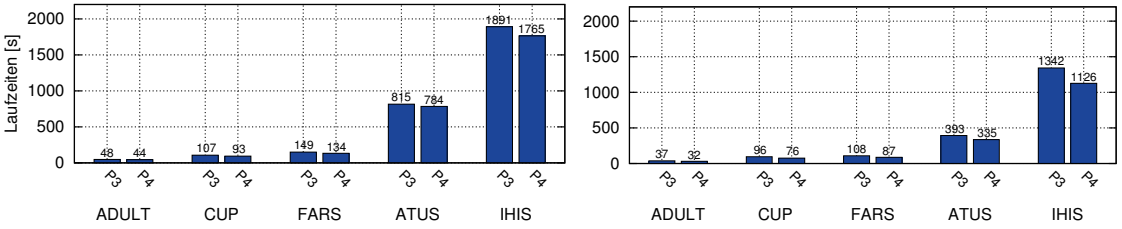
4.6.4.3 Protokoll C

In Abbildung 4.32 sind die Laufzeiten für Protokoll C, über alle fünf Datensätze und die beiden Verteilungen dargestellt. Da im Gegensatz zu den beiden Protokollen A und B bei diesem Protokoll die Anonymisierung eine Rolle für die Performanz des Protokolls spielt, werden hier noch die Parameter für die Anonymisierung angegeben. Es wurden die Daten 5-anonymisiert und 100-anonymisiert, jeweils mit 25 % Pseudotupeln um die Verteilungen zu verstecken. Die 25 % werden berechnet, basierend auf der Gesamtanzahl der Tupel, d. h., im vertikalen und im horizontalen Fall werden, absolut gesehen, gleich viele Tupel hinzugefügt. Es ist keine Unterdrückung erlaubt (s=0 %) und es wurden die in Tabelle 3.2 definierten 8 bzw. 9 Quasi-Identifikatoren genutzt. Der Flash Algorithmus benötigt bei 5-Anonymität beim ADULT Datensatz und dieser Konfiguration 169 Tests, um die optimale Lösung zu finden. Bei CUP sind es 40, bei FARS sind es 135, bei ATUS 557 und beim IHIS Datensatz 295 Tests auf Anonymität. Bei 100-Anonymität sind es beim ADULT Datensatz nur 62 Tests, bei CUP 31, bei FARS 73, bei ATUS 278 und beim IHIS Datensatz 217. Das eingesetzte additiv homomorphe Paillier-Verfahren nutzt eine Bitlänge von 768-Bit.

Die Zeiten, um die fünf Testdatensätze mit Protokoll C bei vertikaler Verteilung zu 5-anonymisieren, liegen zwischen 44 Sekunden für ADULT bei vier Parteien (P4, drei Primärparteien plus Hilfspartei) und knapp 32 Minuten für den IHIS Datensatz bei drei Parteien (P3, zwei Primärparteien plus Hilfspartei). Interessant ist, dass bei drei Primärparteien die Laufzeiten geringer sind als bei zwei Primärparteien. Ein Grund hierfür liegt an der Implementierung der verteilten Anonymitätstests. Im

Prototyp wurde der Test auf Anonymität so implementiert, dass jede Primärpartei nur einen Teil der Gruppen entschlüsseln und testen muss. Somit kann die Arbeitslast, die bei zwei und drei Primärparteien gleich bleibt, bei drei Primärparteien auf eine Partei mehr verteilt werden. So sind die Laufzeiten zwischen 4 %-13 % geringer. Dies wird auch durch Abbildung 4.34 erhärtet, wo zu sehen ist, dass die relativen Zeiten für die Anonymisierung bei vier Parteien, geringer sind als bei drei.

Auch bei Protokoll C sind die horizontalen Laufzeiten geringer als die vertikalen. Grund hierfür sind die im horizontalen Fall nicht benötigten zusätzlichen Zählerspalten. Im vertikalen Fall muss für jedes Tupel des globalen Datensatzes, bei jeder Primärpartei, eine Zählerspalte hinzugefügt werden, im horizontalen Fall nur eine Spalte für alle Parteien. Dadurch wird im vertikalen Fall für jedes Tupel die Zählerspalte n-mal erzeugt, verschlüsselt, übertragen und beim Integrieren im verschlüsselten Raum addiert. Es werden zwar keine Tupel-Identifikatoren benötigt, da diese aber symmetrisch verschlüsselt werden, hat dies so gut wie keinen Einfluss auf die Laufzeit.

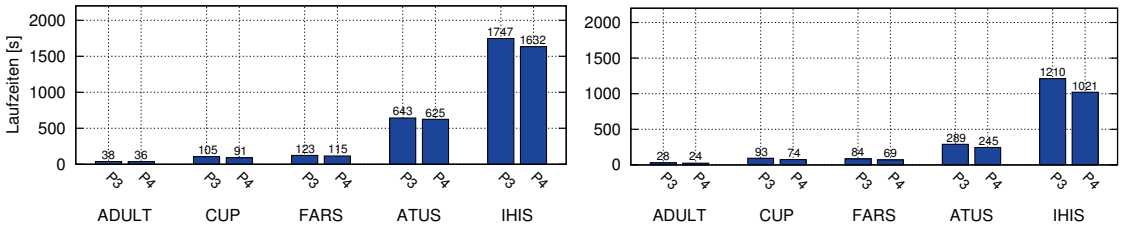


(a) Vertikale Verteilung (b) Horizontale Verteilung

Abbildung 4.32: Protokoll C: Laufzeiten für verschiedene Verteilungen und 5-Anonymität

Die Laufzeiten bewegen sich zwischen 32 Sekunden für den ADULT Datensatz bei drei Primärparteien (P4) und ungefähr 22 Minuten für IHIS bei zwei Primärparteien (P3). Auch im horizontalen Fall kann man sehen, dass die Parallelisierung auf eine Partei mehr, einen Laufzeitgewinn mit sich bringt. Im Gegensatz zu Protokoll B bildet der CUP Datensatz hierbei keine so große Ausnahme mehr, da die benötigte Zeit für die kryptographischen Operationen auf der Zählerspalte, die Zeit für das symmetrische Verschlüsseln der distinkten Werte, um ein vielfaches übersteigt. Nur die Übertragung der größeren Generalisierungshierarchien benötigt etwas mehr Zeit, was dazu führt, dass sich die Zeiten von CUP denen von FARS annähern. Auch im horizontalen Fall sind die Laufzeiten für drei Primärparteien geringer, als die Laufzeiten bei zwei Primärparteien, nämlich zwischen 14 % und 19 %. Der Grund lässt sich auch hier wieder in den verteilten Anonymitätstests finden.

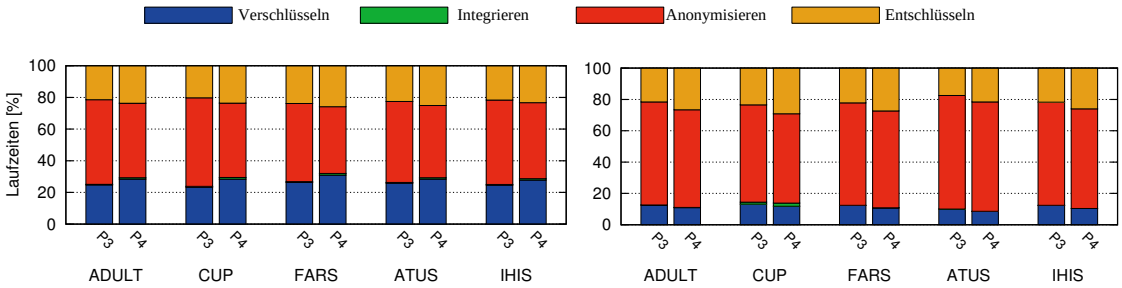
Um später (siehe Abschnitt 4.7.3) einen besseren Vergleich mit verwandten Arbeiten zu ermöglichen, wurden die fünf Datensätze auch noch 100-anonymisiert. Die Zeiten sind in Abbildung 4.33 dargestellt. Wie durch die geringere Anzahl an notwendigen Tests zu vermuten, sind die Laufzeiten, um 100-anonyme Lösungen zu finden, bis auf CUP, ca. 8 % bis 37 % geringer als für 5-anonyme. Bei CUP sind die Laufzeitgewinne geringer und liegen zwischen 2 % und 3 %.



(a) Vertikale Verteilung (b) Horizontale Verteilung

Abbildung 4.33: Protokoll C: Laufzeiten für verschiedene Verteilungen und 100-Anonymität

In Abbildung 4.34 sind wieder die Laufzeiten der verschiedenen Phasen des Protokolls relativ zueinander dargestellt. Man kann sehen, dass im vertikalen Fall die meiste Zeit für das Anonymisieren aufgebracht werden muss (zwischen ca. 40 % und 56 %). Ungefähr die gleiche Zeit wird noch mal benötigt, um die Daten zu verschlüsseln und zu entschlüsseln. Die benötigte Zeit für die Integration ist vernachlässigbar. Im horizontalen Fall muss, relativ gesehen, noch mehr Zeit für die Anonymisierung aufgewendet werden (zwischen ca. 57 % und 72 %). Auch hierbei ist die benötigte Zeit für das Integrieren zu vernachlässigen.



(a) Vertikale Verteilung (b) Horizontale Verteilung

Abbildung 4.34: Protokoll C: Relative Laufzeiten für verschiedene Verteilungen und 5-Anonymität

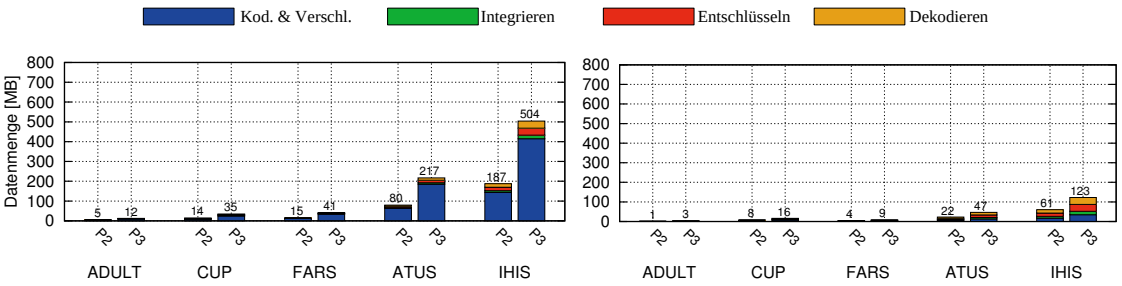
4.6.5 Vergleich der übertragenen Datenmengen

In diesem Abschnitt werden die zu übertragenden Datenmengen der beiden Datenverteilungen und jeweils zwei Primärparteien (P2) bzw. drei Primärparteien (P3) gegenübergestellt. Bei Protokoll B und C wird der Fall mit drei Primärparteien als P4 und mit zwei Primärparteien als P3 bezeichnet, da jeweils eine Hilfspartei dazu kommt.

4.6.5.1 Protokoll A

Abbildung 4.35 zeigt die Menge an übertragenen Daten mit Komprimierung (siehe Abschnitt 4.5.4) für alle fünf Datensätze. Die Datenmenge reicht von 1 MB für ADULT mit zwei Parteien im horizontalen Fall bis zu 504 MB für den IHIS Datensatz bei drei Parteien im vertikalen Fall. Die Datenmenge im vertikalen Szenario ist ca. viermal so groß wie im horizontalen Fall, da die Tupel-Identifikatoren nicht komprimierbar sind. Wie im Abschnitt 4.6.4.1 gezeigt, korreliert dies nicht mit den

benötigten Laufzeiten, da im vertikalen Fall die Laufzeiten von den kryptographischen Operationen dominiert werden und im horizontalen Fall die Laufzeiten von der zu übertragenden Datenmenge. Im vertikalen Szenario werden die meisten Daten während der Verschlüsselungsphase übertragen, da hierbei die Tupel-Identifikatoren mit übertragen werden. Diese werden vor der Integrationsphase verworfen und haben somit ab diesem Zeitpunkt keinen Einfluss mehr auf die Datenmenge. Im horizontalen Fall sind die übertragenen Datenmengen während der Verschlüsselungs-, Integrations- und Entschlüsselungsphase nahezu gleich. Dies ergibt sich daraus, da die Datenmenge hierbei von den Strukturarrays (siehe Abschnitt 4.5.4) dominiert wird.



(a) Vertikale Verteilung (b) Horizontale Verteilung

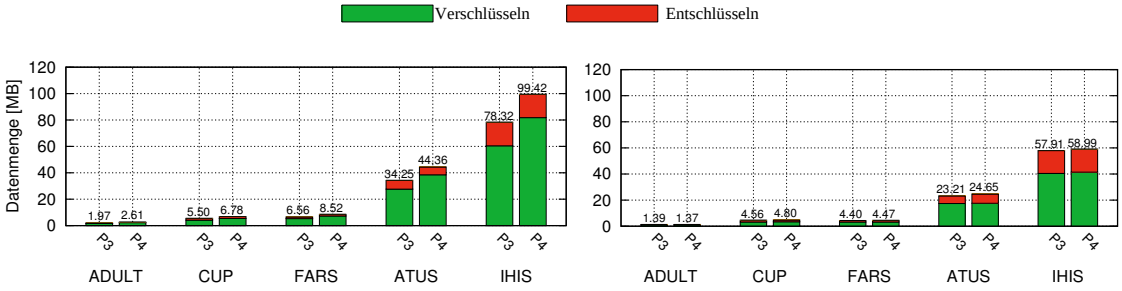
Abbildung 4.35: Protokoll A: Ausgetauschte Datenmenge für verschiedene Verteilungen

4.6.5.2 Protokoll B

Abbildung 4.36 zeigt die Menge an übertragenen Daten mit Komprimierung für alle Konfigurationen. Die Datenmenge reicht von ca. 2 MB für ADULT im vertikalen Fall mit zwei Primärparteien plus Hilfspartei (P3) bis zu ca. 100 MB für IHIS mit drei Primärparteien (4P). Im horizontalen Fall ist die Datenmenge geringer, da auch hier die Tupel-Identifikatoren nicht mit übertragen werden. Die Datenmenge reicht in diesem Fall von ca. 1,4 MB für ADULT (4P) bis zu ca. 59 MB für IHIS (4P) und ist somit ungefähr halb so groß wie im vertikalen Fall. Der kleinere Unterschied, im Gegensatz zu Protokoll A, kann dadurch erklärt werden, dass dort die Größe eines verschlüsselten Tupel-Identifikators bei 192 Bits liegt, wohingegen bei Protokoll B diese Länge nur 128 Bit beträgt. Außerdem muss der Tupel-Identifikator, bedingt durch das Round-Robin-Verfahren, mehrmals über das Netzwerk übertragen werden. Außerdem kann man sehen, dass während der Verschlüsselungsphase mehr Daten übertragen werden als während der Entschlüsselungsphase. Zum einen werden bei der Entschlüsselung die Generalisierungshierarchien nicht mitübertragen, zum anderen ist durch die Integration der Daten eine bessere Komprimierung möglich.

Da bei diesen Messungen keine Anonymisierung vorgenommen wurde, können diese Werte als obere Grenze gesehen werden, da sich durch die Anonymisierung die Entropie verringert und die Daten somit auch besser, während der Entschlüsselungsphase, komprimiert werden können. Die nahezu gleichgroße Datenmenge bei zwei und drei Parteien im horizontalen Fall resultiert daraus, dass in beiden Fällen die gleichen Daten über das Netzwerk übertragen werden, nur anders verteilt. Der erhöhte Kommunikationsoverhead für die größere Anzahl an Nachrichten, im Falle

von drei Primärparteien, ist zu vernachlässigen.

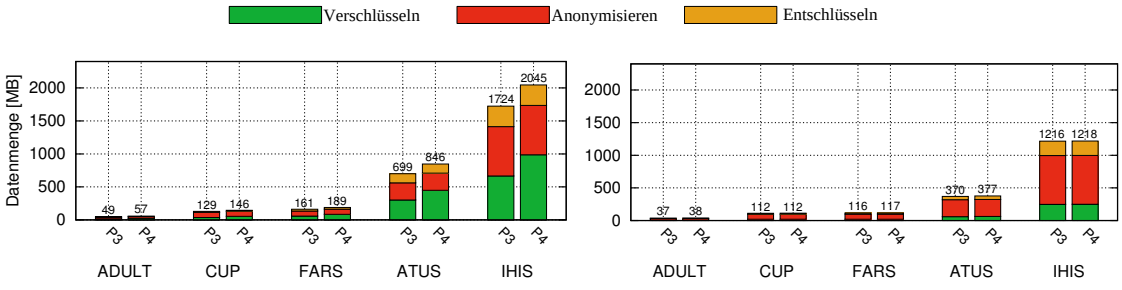


(a) Vertikale Verteilung (b) Horizontale Verteilung

Abbildung 4.36: Protokoll B: Ausgetauschte Datenmenge für verschiedene Verteilungen

4.6.5.3 Protokoll C

Abbildung 4.37 zeigt die Menge an übertragenen Daten für Protokoll C, im Falle von 5-Anonymisierung. Für die Anonymisierung von ADULT im vertikalen Fall, bei zwei Primärparteien (P3), werden 49 MB übertragen, die zu übertragende Datenmenge bei IHIS und drei Primärparteien (P4) beträgt etwas über 2 GB. Damit wird eine wesentlich höhere Datenmenge als bei Protokoll A und B übertragen. Im Vergleich zu Protokoll B ist es ca. die 19- bis 25-fache Datenmenge, im Vergleich zu Protokoll A immerhin noch die fünf- bis elffache. Die größere Datenmenge, beim vertikalen Fall, im Vergleich zum horizontalen Fall, kann dadurch erklärt werden, dass weniger Zählerspalten während der Verschlüsselungsphase übertragen werden müssen. Für jede Partei muss eine Zählerspalte, pro Tupel, hinzugefügt werden. Die benötigten Datenmengen für das Anonymisieren und Entschlüsseln sind in beiden Fällen nahezu identisch. Die minimalen Unterschiede bei der Entschlüsselungsphase können dadurch erklärt werden, dass die Verschlüsselung der Zählerspalte semantisch sicher ist und somit sich die Effizienz der Komprimierung ändern kann.



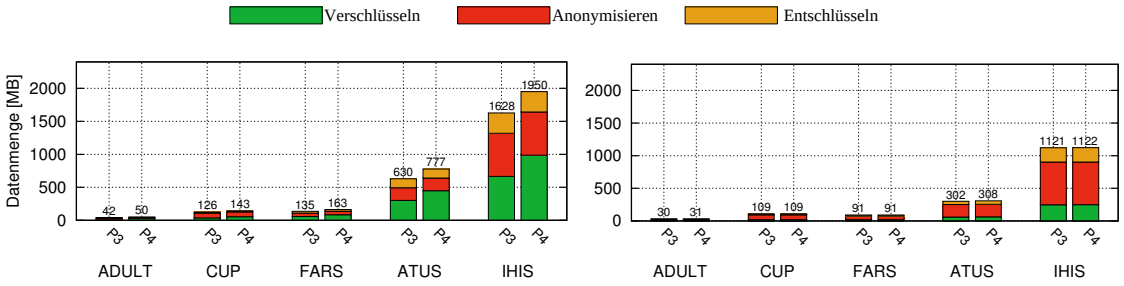
(a) Vertikale Verteilung (b) Horizontale Verteilung

Abbildung 4.37: Protokoll C: Ausgetauschte Datenmenge für verschiedene Verteilungen und 5-Anonymität

Im horizontalen Fall reicht die Datenmenge von 37 MB für die Anonymisierung des ADULT Datensatzes, bei zwei Primärparteien (P3) bis zu ca. 1,2 GB für IHIS bei zwei und drei Primärparteien. Wie bei Protokoll B kann hierbei beobachtet werden, dass die Datenmenge bei zwei oder drei Primärparteien nahezu identisch ist. Der

höhere Kommunikationsoverhead bei drei Primärparteien, im Gegensatz zu zwei Primärparteien ist gering, z. B. max. 7 MB bei ATUS, was ca. 2 % entspricht. Die geringere Datenmenge bei der Verschlüsselungsphase, im Vergleich zum vertikalen Fall, kann damit erklärt werden, dass, bevor die Daten an die Hilfspartei gesendet werden, jede Primärpartei schon Gruppen bilden kann. Dadurch fallen gleichartige Tupel weg und nur der Zählerwert jeder Spalte erhöht sich.

In Abbildung 4.38 sind die zu übertragenden Datenmengen für den Fall $k = 100$ dargestellt. Da sich hierbei hauptsächlich die Anzahl der Tests auf Anonymität ändert, kann man die größte Reduktion der Datenmenge in der Anonymisierungsphase erkennen. Die Datenmenge ist im Vergleich zum 5-anonymisierten Fall um ca. 2 %-29 % geringer und liegt somit zwischen 30 MB im horizontalen Fall, bei zwei Primärparteien und knapp unter 2 GB bei drei Primärparteien und dem IHIS Datensatz.



(a) Vertikale Verteilung (b) Horizontale Verteilung

Abbildung 4.38: Protokoll C: Ausgetauschte Datenmenge für verschiedene Verteilungen und 100-Anonymität

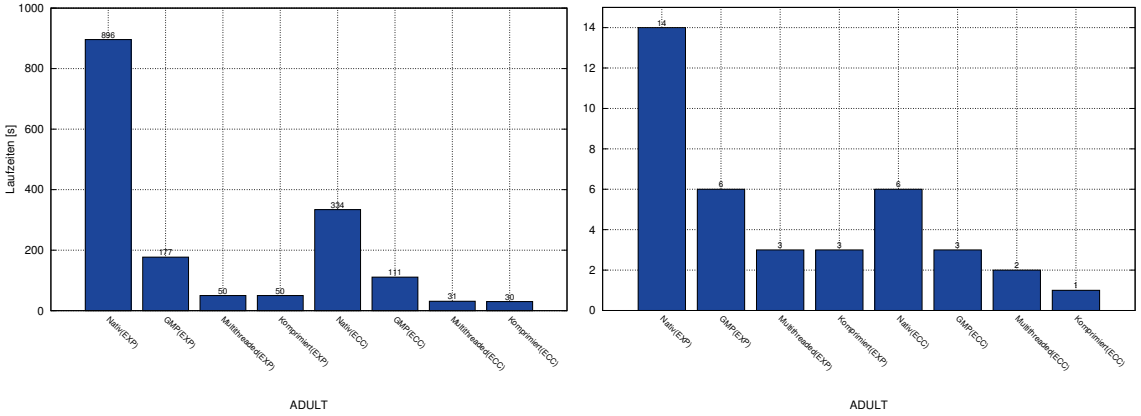
4.6.6 Vergleich der Optimierungen

In diesem Abschnitt werden die Gewinne, durch die verschiedenen in Abschnitt 4.5 diskutierten Optimierungen hinsichtlich der Laufzeiten, dargestellt. Da das Protokoll A am meisten Optimierungen beinhaltet (z. B. der Wechsel auf Elliptische-Kurven, welcher nur in Protokoll A genutzt wird) werden die Zahlen bei diesem Protokoll und mit dem ADULT Datensatz aufgeführt. Die anderen Datensätze verhalten sich hierbei sehr ähnlich und werden deshalb nicht gesondert diskutiert. Es werden die absoluten Laufzeiten der beiden Datenverteilungen und jeweils zwei Parteien (P2) bzw. drei Parteien (P3) gegenübergestellt. Diese Zahlen enthalten alle notwendigen Operationen des Prä- und Postprocessing von Protokoll A.

4.6.6.1 Laufzeiten

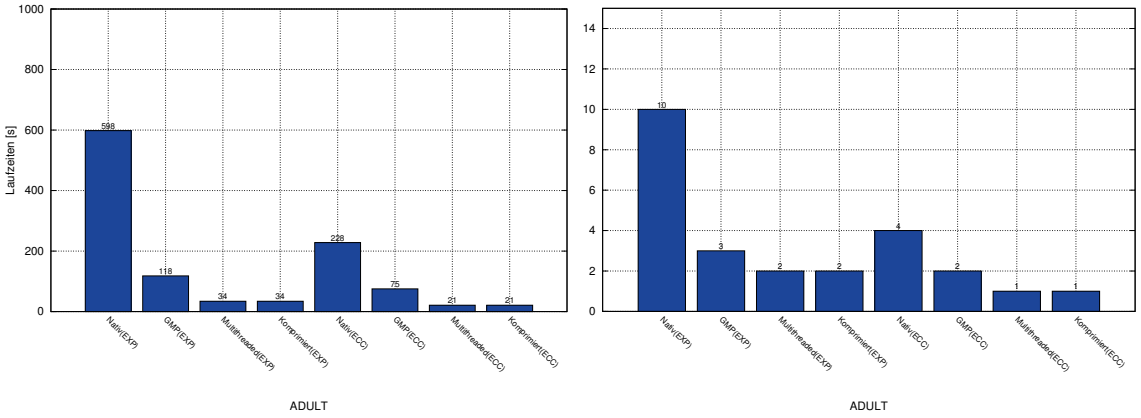
In Abbildung 4.39 sind die Laufzeiten für die verschiedenen Optimierungen, die in Abschnitt 4.5 beschrieben sind, gegenübergestellt. Im vertikalen Fall, bei drei Parteien (Abbildung 4.39a) und ohne Optimierungen, also mittels Standard-Pohlig-Hellman direkt mit Java BigInteger implementiert (Nativ(EXP)) benötigt das Protokoll ca. 15 Minuten. Greift man stattdessen auf die GMP Bibliothek mittels JNI für die Exponentiation (GMP(EXP)) zu, wird die Ausführungszeit auf ca. 3 Minuten reduziert. Mit vier Threads auf einer Quad-core Maschine kann die Ausführungszeit um fast den Faktor 3,5 auf nur noch ca. 50 Sekunden gedrückt werden (Multithreaded(EXP)). Die zusätzlich durchgeführte Komprimierung (Komprimiert(EXP)) hat

so gut wie keinen Einfluss auf die Laufzeit. Bei Nutzung von elliptischen Kurven als Grundlage für das Pohlig-Hellman Verfahren, bewegt sich die Ausführungszeit zwischen knapp 5,5 Minuten im unoptimierten Fall (Nativ (ECC)) und ca. 31 Sekunden im parallelisierten Fall. Im, mittels GMP implementierten und komprimierten Fall, beträgt die Laufzeit noch 30 Sekunden (Komprimiert(ECC)). Die vorgeschlagenen und umgesetzten Optimierungen erzielen somit einen Speedup um ca. den Faktor 30 im Vergleich zur naiven Implementierung des Algorithmus. Bei zwei Parteien (siehe Abbildung 4.39c) sind die absoluten Laufzeiten geringer, der Speedup ist aber bei einem Faktor von ca. 28 vergleichbar mit dem bei drei Parteien.



(a) Vertikale Verteilung - 3 Parteien

(b) Horizontale Verteilung - 3 Parteien



(c) Vertikale Verteilung - 2 Parteien

(d) Horizontale Verteilung - 2 Parteien

Abbildung 4.39: Laufzeiten für verschiedene Optimierungen und Verteilungen am Beispiel des ADULT Datensatzes

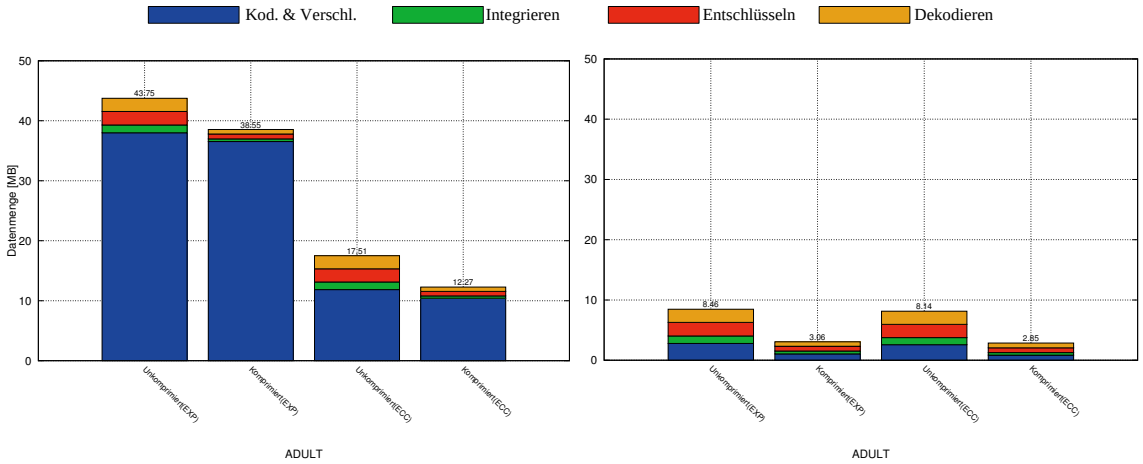
Im horizontalen Fall sind die Laufzeiten, im unoptimierten Fall, um den Faktor 64 schneller als im vertikalen Fall (siehe Abbildung 4.39b). Bei maximaler Optimierung (Komprimiert(ECC)) liegt der Faktor bei 30. Damit wurde die Ausführungszeit von ca. 14 Sekunden für den ADULT Datensatz auf nur noch ca. 1 Sekunde gesenkt. Der gesamt Speedup liegt somit bei ungefähr 14x. Auch im horizontalen Fall ist die Laufzeit, bei drei Parteien, wieder niedriger und der Speedup beträgt hierbei ca. Faktor 10 zwischen unoptimierten und optimiertem Protokoll.

Den größten Performanzvorteil bringt die Implementierung der kryptographischen Operationen mittels der GMP Bibliothek (bis zu 5x beim klassischen Pohlig-

Hellman und Faktor 2-3 bei ECC). Hierbei halbiert die Nutzung von elliptischen Kurven, anstelle des Standard-Pohlig-Hellman Verfahrens, die Laufzeit ungefähr. Der Einfluss auf die zu übertragenden Datenmengen ist in Abschnitt 4.6.6.2 beschrieben.

4.6.6.2 Datenmengen

In Abbildung 4.40 werden die zu übertragenden Datenmengen, für die verschiedenen Optimierungen, gegenübergestellt. Die hier dargestellten Zahlen wurden mit 1536-Bit klassischem Pohlig-Hellman und 192-Bit ECC Pohlig-Hellman gemessen. Die Komprimierung wurde mittels des Snappy Kompressors [114] durchgeführt. Die Datenmenge kann natürlich auch durch Reduzierung der jeweiligen Bitlängen erreicht werden, was aber eine Verminderung der Sicherheit zur Folge hat. Wie oben bereits erwähnt, wurden diese Bitlängen gewählt, da sie den aktuellen Empfehlungen folgen. 1536-Bit (anstatt nur 1024-Bit) wurden gewählt, um einen fairen Vergleich zwischen ECC und klassischem Pohlig-Hellman, also bei gleichem Sicherheitsniveau, zu erzielen.



(a) Vertikale Verteilung - 3 Parteien

(b) Horizontale Verteilung - 3 Parteien

(c) Vertikale Verteilung - 2 Parteien

(d) Horizontale Verteilung - 2 Parteien

Abbildung 4.40: Protokoll A: Übertragene Datenmengen für verschiedene Optimierungen und Verteilungen für den ADULT Datensatz

Im vertikal verteilten Szenario und bei drei Parteien werden pro Partei, beim ADULT Datensatz mit etwas über 30.000 Datensätzen und ca. 2,52 MB Speicher-

verbrauch auf der Festplatte, mit EXP-Pohlig-Hellman ca. 44 MB an Daten über das Netzwerk übertragen. Die Nutzung des Snappy Komprimierungsalgorithmus reduziert die zu übertragende Datenmenge um ca. 11 %. Bei Einsatz von elliptischen Kurven reduziert sich die Datenmenge, im unkomprimierten Fall, auf 17,51 MB. Wird vor der Übertragung eine Komprimierung durchgeführt, reduziert sich die Datenmenge nochmals auf ca. 12,3 MB. Dies entspricht insgesamt einer Reduzierung der Datenmenge um ca. den Faktor 3,5 bei gleich gebliebenen Sicherheitsgarantien. Bei zwei Parteien ist die Datenmenge ungefähr um den Faktor 3 geringer, die Ersparnis zwischen unkomprimierten EXP-Pohlig-Hellman und der auf elliptischen Kurven basierten und komprimierten Datenmenge, liegt auch bei einem Faktor von ca. 3,5.

Im horizontal verteilten Fall ist die zu übertragende Datenmenge, bei drei Parteien, ungefähr um den Faktor 5 geringer als im vertikalen Fall, da die Tupel-Identifikatoren nicht mehr benötigt werden. Beim ADULT Datensatz müssen im horizontal verteilten Fall, nun unkomprimiert, 8,46 MB an Daten über das Netzwerk übertragen werden. Die Komprimierung reduziert die zu übertragene Datenmenge um den Faktor 2,7 auf 3,06 MB. Wird nun ECC verwendet, müssen im unkomprimierten Fall 8,14 MB übertragen werden. Wird wieder komprimiert, sinkt die Datenmenge auf nun mehr 2,85 MB um ca. 65 %. Auch im horizontal verteilten Fall kann somit insgesamt eine Reduktion der Datenmenge, sowohl bei zwei als auch drei Parteien, um insgesamt ca. den Faktor 3 erreicht werden.

4.7 Diskussion der Protokolle für die Anonymisierung verteilter Datenbestände

Wie aus den vorherigen Abschnitten ersichtlich, ermöglichen es die hier vorgestellten Protokolle, erstmals in praktikabler Zeit und mit vertretbarer zu übertragender Datenmenge, eine Anonymisierung von horizontal und vertikal verteilten Daten durchzuführen und dabei eine global optimale Lösung hinsichtlich des Informationsverlustes, zu finden. Zuerst werden die unterstützten Algorithmen und Anonymisierungskriterien diskutiert, danach werden die hier vorgestellten Lösungen mit vorhergehenden Ansätzen verglichen. Als Letztes werden die Vorbedingungen und Annahmen, die diesem Protokoll zugrunde liegen, diskutiert.

4.7.1 Unterstützte Algorithmen

Die hier vorgestellten Protokolle können auch mit einer Vielzahl anderer Algorithmen benutzt werden, wenn diese den folgenden Anforderungen genügen. Algorithmen müssen entweder direkt hierarchiebasiert sein, oder es muss möglich sein, solche Generalisierungshierarchien aus verschlüsselten Datenelementen zu generieren. Diese Anforderung erfüllen alle Algorithmen aus Abschnitt 3.5. Ein nicht unterstützter Algorithmus zum Beispiel ist Mondrian. Dieser Algorithmus arbeitet partitionsbasiert und benötigt eine totale Ordnung auf allen Elementen. Diese totale Ordnung kann nicht für die verschlüsselten Datenelemente bereitgestellt werden. Das automatische Erstellen von Generalisierungsgraphen, vorgestellt in [120], kann hingegen bei den hier vorgestellten Protokollen genutzt werden. Die Protokolle können auch bei den meisten vorgestellten Clustering Algorithmen verwendet werden, da diese Distanzmetriken mittels Generalisierungshierarchien realisieren.

Beim Generalisierungs- oder Transformationsschritt implementieren manche Algorithmen generalisierungsbasierte Verfahren und andere partitionierungsbasierte Verfahren für kontinuierliche Werte, z. B. [83] oder [82]. Nur die generalisierungsbasierten Verfahren können mit den Protokollen genutzt werden. Im Falle, dass kontinuierliche Attribute als Quasi-Identifikatoren betrachtet werden, müssen für diese Generalisierungshierarchien zur Verfügung gestellt werden (vgl. auch Abschnitt 4.7.4).

Die hier vorgestellten Protokolle ermöglichen eine effiziente Implementierung von einer Vielzahl von Algorithmen. Dazu gehören heuristische Algorithmen, z. B. [82], clustering Algorithmen, z. B. [83], Algorithmen, die eine global optimale Lösung finden, z. B. [73] und Domain spezifische, z. B. [121].

4.7.2 Unterstützte Anonymisierungskriterien

Neben den o. g. Algorithmen werden auch eine Vielzahl von Anonymisierungskriterien unterstützt. Zum Beispiel können alle, in Abschnitt 2.4 angesprochenen Kriterien, mit den hier vorgestellten Protokollen implementiert werden. Protokoll C kann aktuell nur k-anonyme Lösungen erstellen, eine Erweiterung auf andere Kriterien ist aber möglich. Die einzige Limitierung bei den Protokollen A und B ist, dass die Kriterien keine Berechnungen auf den verschlüsselten Daten ausführen können. Ein Beispiel für so eine Berechnung wäre die Differenz zwischen zwei numerischen Werten. Ein semantischer Vergleich ist jedoch, zumindest durch die Generalisierungshierarchien, möglich.

Aus diesen Gründen können, mit den hier vorgestellten Protokollen, keine Anonymisierungskriterien implementiert werden, die speziell für numerische Attribute entwickelt wurden. Damit kann z. B. nicht t-closeness für numerische Werte implementiert werden, alle anderen Varianten von t-closeness allerdings schon. Auch wird (k, e)-Anonymität [122] und (ϵ, m)-Anonymität [123] nicht unterstützt.

In der hier vorgestellten Version unterstützen die Protokolle auch nicht ℓ-Site-Diversität, da sie nicht die Informationen über die Site/das Zentrum mitliefert, von welcher die Datenelemente stammen. Diese Information würde den Bemühungen des Protokolls zuwiderlaufen, was genau diese Information vor den Teilnehmern verbirgt (siehe auch Abschnitt 4.4.3.5).

$(\epsilon, \delta)^k$-Dissimilarity [124] hingegen ist ein generisches Anonymisierungskriterium, welches verschiedene Distanzmaße zulässt. Die *variational distance*, welche die Autoren in ihrer Arbeit als Beispiel nutzen, kann mit der hier vorgestellten Methode genutzt werden.

Unseres Wissens sind somit die hier vorgestellten Protokolle die Ersten, die ein so breites Spektrum an Anonymisierungskriterien, in einer verteilten Umgebung, unterstützen. Die Protokolle A und B unterstützen unter anderem: LKC-privacy, (α, k)-Anonymität, p-sensitive k-Anonymität, $(\epsilon, \delta)^k$-dissimilarity und m-Invarianz.

Protokoll C muss für andere Kriterien angepasst werden. Im Falle von z. B. ℓ-Diversität muss die Verteilung der sensitiven Attribute, von der Hilfspartei an die Primärpartei in einer Weise gesendet werden, dass die Pseudotupel wieder herausgerechnet werden können. Im einfachsten Fall kann dies erreicht werden, indem für jedes sensitive Attribut die Zählerspalte an die Primärpartei gesendet wird, die damit wiederum die Pseudotupel entfernen kann. Dadurch lernt die Primärpartei zwar die Verteilung der sensitiven Attribute, allerdings nicht die Verteilung der

Quasi-Identifikatoren.

4.7.3 Vergleich mit existierenden Arbeiten

In diesem Abschnitt werden die in Abschnitt 4.2 vorgestellten, verwandten Arbeiten mit den hier vorgestellten Ansätzen verglichen. Im Folgenden wird der Ansatz aus [93] DkA_v genannt, die verteilte Implementierung von Mondrian von [98] wird mit $Mondrian_h$ bezeichnet. Der Ansatz der Top-down-Spezialisierung wird für die vertikale Variante [95] mit TDS_v bezeichnet und die horizontale[85] Variante mit TDS_h. Der verteilte clustering Ansatz von Tassa et al. [103] wird im Folgenden mit SCA_{hv} bezeichnet.

Die drei oben vorgestellten Protokolle implementieren einen Mix aus den Methoden des Designraumes. Es wird die integriere-und-anonymisiere Methode benutzt, allerdings mit verschlüsselten Daten. Die integrierte, verschlüsselte Sicht auf die Daten wird mit SMC Methoden gebildet (zumindest bei Protokoll A und C) und repräsentiert damit die Methode der virtuellen Anonymisierung. Wird das Präprocessing verwendet um die Verteilungen in den Daten zu verschleiern (siehe Abschnitt 4.4.3.5, dann kann man das als die Methode anonymisiere-und-integriere, ansehen. In dem Evaluationsabschnitt wurde schon gezeigt, dass die drei Protokolle die naive Variante, anonymisiere-und-integriere, hinsichtlich Datenqualität und Flexibilität, in den Schatten stellen.

Die anderen verteilten Ansätze bringen Support für verschiedene Anonymisierungsmethoden, Anonymisierungskriterien und Verteilungen der Daten mit:

- Die hier vorgestellten Protokolle unterstützen eine beliebige Anzahl von Parteien. Protokoll B und Protokoll C brauchen aber, neben den Primärparteien mit Daten, noch eine weitere Hilfspartei für die Ausführung. Die anderen verwandten Ansätze unterstützen bis auf DkA_v auch eine beliebige Anzahl von Parteien.

- Ähnlich wie die Ansätze DkA_v und SCA_{hv} unterstützen die hier vorgestellten Ansätze globales und lokales recoding. $Mondrian_h$, TDS_h und TDS_v unterstützen hingegen nur globales recoding.

- DkA_v, TDS_h und TDS_v bieten hinsichtlich ihrer Annahmen perfekten Datenschutz. Im Gegensatz dazu wird bei den hier vorgestellten Protokollen A und B sowie bei den Ansätzen $Mondrian_h$ und SCA_{hv} etwas Information während der Protokollausführung bekannt. Protokoll C kann, wenn die Pseudotupel richtig eingesetzt werden, auch komplett ohne Informationsleck betrieben werden.

- Ähnlich wie SCA_{hv} unterstützen die hier vorgestellten Protokolle sowohl horizontal als auch vertikal verteilte Daten. TDS_h und $Mondrian_h$ können nur mit horizontal verteilten Daten umgehen, wohingegen DkA_v und TDS_v nur das vertikale Szenario unterstützen.

- Die Ansätze unterscheiden sich außerdem in den implementierten Anonymisierungskriterien. DkA_v ist der einzige vergleichbare Kriterium und Algorithmus unabhängige Ansatz. Dennoch untersuchen die Autoren in ihrer Veröffentlichung nur k-Anonymität und, da der Ansatz darauf basiert, dass

Äquivalenzklassen mit Tupel-Identifikatoren geschnitten werden, ist DkA_v z. B. nicht verwendbar für clustering Algorithmen. Die Veröffentlichung über TDS_v und SCA_{hv} beschreiben, dass die Ansätze mit k-Anonymität und ℓ-Diversität funktionieren. $Mondrian_h$ implementiert k-Anonymität und ℓ-site-Diversität. TDS_h implementiert LKC-privacy, was Support für k-Anonymität und ℓ-Diversität impliziert. Welche anderen Anonymisierungskriterien die Ansätze unterstützen, müsste gesondert untersucht werden. Die hier vorgestellten Protokolle unterstützen ein weit größeres Spektrum an Kriterien. Implementiert wurde für Protokoll A und B k-Anonymität, ℓ-Diversität, t-Closeness und δ-Präsenz. Bei Protokoll C wurde nur k-Anonymität implementiert, die anderen drei vorgenannten Kriterien werden aber prinzipiell auch bei Protokoll C unterstützt.

Im restlichen Abschnitt wird nun die Performanz (und damit auch, zumindest teilweise, die Praxisrelevanz) der Protokolle A-C mit den verwandten Arbeiten verglichen. Diese Protokolle sind, bis auf Protokoll C, unabhängig von den eingesetzten Algorithmen zur Anonymisierung, da sie eine Prä- und Postbearbeitungsphase haben. Damit kann die eigentliche Anonymisierung auch mit zentralen Varianten der Algorithmen eingesetzt werden, was aber nicht impliziert, dass diese schneller sind, als dedizierte, verteilte Anonymisierungsalgorithmen. Bei Protokoll C muss der eigentliche Anonymisierungsalgorithmus angepasst werden, da es einen eigenen Test auf Anonymität benötigt (Näheres ist in Abschnitt 4.4.5.1.3 zu finden).

Da keine Implementierungen von DkA_v, $Mondrian_h$, TDS_v, TDS_h oder SCA_{hv} verfügbar sind, werden die Vergleiche auf hardwareunabhängige Metriken oder analytische Modelle, so weit wie möglich, basieren. Wenn beides nicht möglich ist, werden auch Performanzmessungen von anderer Hardware einbezogen, aber nur in einer sehr konservativen Weise, welche die zwischenzeitlich größer gewordene Performanz moderner Hardware einbezieht. Da viele verwandte Arbeiten eine systematische, experimentelle oder analytische Evaluation missen lassen, haben andere verwandte Arbeiten (z. B. [95], [85], [103]) denselben Weg hinsichtlich des Vergleiches eingeschlagen. $Mondrian_h$ wird aus den Vergleichen ausgeschlossen, da dieser Ansatz den Mondrian Algorithmus von [100] implementiert, welcher von den hier vorgestellten Protokollen nicht unterstützt wird.

Wie oben beschrieben, ist DkA_v unabhängig von dem eingesetzten Algorithmus. Als ein Beispiel nutzen die Autoren in [93] den Datafly Algorithmus von Sweeney [58], um ihre vertikal verteilte Version des ADULT Datensatzes mit zwei Parteien zu k-anonymisieren. In der Arbeit wird ein hardwareunabhängiges Modell vorgeschlagen, welches erlaubt, die Ausführungszeit abzuschätzen. Das Modell hängt von der Anzahl der auszuführenden homomorphen kryptographischen Operationen pro Sekunde ab (COps/s). Das verwendete Testbed kann ca. 3000 kommutative kryptographische Operationen ausführen. Diese 3000 Ops/s werden als konservativer Wert für die weiteren Berechnungen angenommen, obwohl homomorphe Operationen wesentlich langsamer sind (auf dem in dieser Arbeit verwendeten Testbed ca. 800). Mit diesen Annahmen benötigt der in [93] vorgestellte Ansatz 5 Tage für k=20, 6 Tage für k=50 und 8 Tage für k=100. Verglichen mit der Prä- und Postbearbeitung bei Protokoll A von 20 Sekunden und bei Protokoll B von nur 0,28 Sekunden. Zum Vergleich wurde Datafly auf der hier benutzten Hardware ausgeführt, um den ADULT

Datensatz mit k=20, 50 und 100 zu anonymisieren. Für die Anonymisierung wurden dabei nie mehr als 1 Sekunde und 14 Tests auf Anonymität benötigt. Dies zeigt, dass die Ausführung des zentralisierten Datafly Algorithmus mit einem der hier vorgestellten Protokolle, signifikant schneller ist als DkA_v. Protokoll C benötigt für die gesamte Anonymisierung bei k=5 und 169 Tests auf Anonymität 48 Sekunden.

TDS_v wurde evaluiert indem eine auf zwei Parteien vertikal verteilte Version des ADULT Datensatzes k-anonymisiert wurde [95]. Die Autoren evaluierten ihren Ansatz auf einem Testbed mit Computern mit Intel Pentium IV 2.6GHz CPUs und einem FastEthernet LAN. Die Autoren geben an, dass ihr Ansatz nie mehr als 20 Sekunden benötigt für $20 \leq$ k ≤ 50. Die neu entwickelten Ansätze brauchen zwischen 0,28 Sekunden (Protokoll B) und 20 Sekunden (Protokoll A) für das Prä- und Postprocessing. Die 5-Anonymisierung mittels Protokoll C benötigt 48 Sekunden, bei 100-Anonymisierung nur noch 38 Sekunden. Diese Ergebnisse wurden auf modernerer Hardware gemessen. Diese Zahlen legen nahe, dass die Ausführung eines zentralen Algorithmus bei Protokoll A und C nicht schneller ist als TDS_v für $20 \leq$ k ≤ 50. Bei Protokoll B muss noch die Zeit für die Anonymisierung hinzugerechnet werden, welche ca. 1 Sekunde benötigt. Damit ist Protokoll B schneller als TDS_v für diese Parameter. Wie der Top-down Ansatz sich verhält, wenn k kleiner wird (z. B. $k = 5$ als Parameter typischerweise in der medizinischen Domäne genutzt), benötigt weitere Untersuchungen.

Im horizontalen Fall, TDS_h, wurde zur Evaluierung von den Autoren das LKC-Kriterium auf den ADULT Datensatz angewandt, dieser wurde horizontal auf drei Parteien verteilt [85]. Das Testbed bestand aus Computern mit Intel Core2 Quad Q6600 2.4GHz CPUs, verbunden mittels FastEthernet LAN. Die Autoren geben Ausführungszeiten von 30 Sekunden für $L = 4$, $20 \leq K \leq 100$ und $C = 0.2$ an. Das Protokoll A benötigt für das Prä- und Postprocessing ca. 1 Sekunde, Protokoll B hingegen 0,23 Sekunden. Dies ist ein Hinweis, dass es keinen Vorteil bringt, die zentrale Version des Algorithmus mit dem Protokoll A zu nutzen. Bei Protokoll B könnte es einen kleinen Vorteil bringen. Protokoll C kann hier nicht direkt verglichen werden, da nur k-Anonymität als Kriterium implementiert wurde.

SCA_{hv} unterstützt sowohl vertikal als auch horizontal verteilte Daten. Für den Vergleich werden Hardware unabhängige Zahlen für den Kommunikationsoverhead genutzt. Die zusätzlich notwendigen Rechenoperationen werden ignoriert. In [103] geben die Autoren Zeiten für die k-Anonymisierung eines vertikal verteilten ADULT Datensatzes auf zwei Parteien an. Für k=100 benötigt ihr Ansatz 2880 Sekunden für den Datentransfer in einem FastEthernet LAN. Diese Zahl ist nahezu unabhängig von k, sie nimmt nur marginal ab, wenn k zunimmt. Zum Vergleich benötigt das hier vorgestellte Protokoll A für das Prä- und Postprocessing ungefähr 20 Sekunden, Protokoll B benötigt 0,28 Sekunden und Protokoll C 48 Sekunden (diese Werte inkludieren die Zeiten für die Berechnungen und Datenübertragung). Die Zeiten für Protokoll A und B sind unabhängig von k. Die Zeiten von Protokoll C nehmen ab, wenn k größer wird. Somit benötigt Protokoll C nur noch 38 Sekunden für k=100. Die Autoren berichten auch Zeiten einer zentralisierten Implementierung ihres Algorithmus. Für denselben Datensatz benötigt ihre Implementierung 150 Sekunden für k=100 auf älterer Hardware (Intel Core Duo T2350 CPU 1.86 GHz). Damit ist deutlich, dass die hier vorgestellten Protokolle A und B, zusammen mit der zentralen Version des Algorithmus, schneller sind als der dediziert entwickelte

dezentrale SCA_{hv}.

Die Autoren geben außerdem Zeiten für die k-Anonymisierung eines horizontal verteilten ADULT Datensatzes auf zwei Parteien an. Für k=100 benötigt ihr dezentraler Ansatz 800 Sekunden allein für den Datentransfer in einem FastEthernet LAN. Diese Zeiten werden signifikant größer mit abnehmendem k. Bei k=25 benötigt der Ansatz bereits 4400 Sekunden. Ob mit dieser Zunahme der Ansatz überhaupt in der medizinischen Domäne benutzte Parameter (z. B. k=5) unterstützt, benötigt weitere Untersuchungen. Verglichen mit den Laufzeiten für Protokoll A mit 1 Sekunde, Protokoll B mit 0,23 Sekunden und Protokoll C mit 37 Sekunden für $k = 5$ in einer vergleichbaren Netzwerkumgebung ist es offensichtlich, dass die Implementierung der zentralen Version des Algorithmus, in Kombination mit einem der drei Protokolle, schneller ist als SCA_{hv}.

4.7.4 Vorbedingungen und Annahmen

In diesem Abschnitt werden die Vorbedingungen und Annahmen für die Protokolle A-C, die in Abschnitt 4.4.1 beschrieben sind, diskutiert.

Wie die meisten anderen Ansätze, z. B. [85, 93, 95, 98, 103], wird hier davon ausgegangen, dass die Datenintegration bereits vor dem Start des Protokolls erfolgt ist. Wenn Daten in der biomedizinischen Forschung in einer verteilten Umgebung gesammelt werden, z. B. bei einem Forschungsnetz und im Falle von Pseudonymisierung, ist die Verteilung der Daten vordefiniert. Im horizontal verteilten Fall werden die Daten nach demselben Protokoll gesammelt, welches das Schema und oft auch die Attributsausprägungen vorgibt. Wenn eine Pseudonymisierungsarchitektur die Daten vertikal verteilt, ist die Aufteilung fest vorgegeben und es existiert in den meisten Fällen auch ein gemeinsamer Tupel-Identifikator. Aus diesen Gründen sind Inkonsistenzen selten. Sollten die Daten doch inkonsistent sein, hat dies keine Auswirkung auf die Protokollabläufe und es ist unwahrscheinlich, dass diese Inkonsistenzen dann zu Datenschutzproblemen führen. Wenn solche Inkonsistenzen aufgelöst werden sollen, kann man Arbeiten im Bereich des Datenschutz erhaltenden Data cleansing zu Hilfe nehmen, z. B.[125, 126].

Den vorgestellten Protokollen liegt das halb-ehrliche Sicherheitsmodell zugrunde. Auch diese Annahme ist bei verwandten Arbeiten oft anzutreffen, z. B. [85, 93, 95, 98, 103]. Außerdem ist diese Annahme in der biomedizinischen Domäne auch realistisch, siehe auch Abschnitt 4.4.3.5. Die sicheren Kommunikationskanäle sind auch eine oft geforderte Vorbedingung. Hierbei kann auf eine Vielzahl von Standardprotokollen zurückgegriffen werden, z. B. SSL/TLS [127]. Die sichere Aushandlung von benötigten kryptographischen Parametern für die Protokolle kann auch auf verschiedene Weise ausgeführt werden, z. B. mittels iteriertem Diffie-Hellman (siehe auch Abschnitt 4.4.4).

Eine weitere Annahme ist, dass gemeinsame, globale Generalisierungshierarchien für alle Quasi-Identifikatoren vorhanden sind. Auch diese Annahme ist vielen verwandten Arbeiten gemeinsam. Aus diesem Grund kann es keine Konsistenzprobleme geben, wenn die Teilgeneralisierungshierarchien, während der Protokollausführung, zusammengeführt werden. Alle Generalisierungsregeln entstammen der globalen Generalisierungshierarchie, deshalb können nur Duplikate auftreten, die einfach während des Integrationsschritts entfernt werden können. Auch bei eventuell angewendeter Prägeneralisierung, z. B. als Gegenmaßnahme gegen Frequenzanalysen (siehe

Abschnitt 4.4.3.5), gibt es hierbei keine Konsistenzprobleme. Die benötigten Regeln für die prägeneralisierten Daten können entweder in den globalen Hierarchien vordefiniert sein oder aus diesen, von jeder Partei für ihre eigenen Daten, abgeleitet werden.

Diese globalen Generalisierungshierarchien sind im vertikalen Fall einfach zu erstellen. Da jede Partei eine distinkte Menge an Attributen hält, kann diese ihre eigenen Hierarchien definieren. Deshalb können auch komplexe Funktionen, z. B. Clustering, verwendet werden, um Generalisierungshierarchien für kontinuierliche Variablen dynamisch zu erstellen. Im horizontalen Fall ist das Erstellen der Hierarchien etwas komplizierter. Für diskrete Variablen besteht kein Problem, auch nicht, wenn diese eine sehr große Domäne haben. Erstens werden nur relevante Regeln, also diejenigen, zu welchen es mindestens einen Wert im Datensatz gibt, übertragen. Zweitens können Hierarchien auch als Funktionen ausgedrückt werden, welche zwischen den Parteien ausgetauscht und dazu benutzt werden, dynamisch Hierarchien für vorhandene Werte zu generieren. Es muss hierbei nur sichergestellt werden, dass die generierten Regeln für gleiche Werte konsistent für alle Parteien sind. Außerdem werden die Regeln lokal generiert, ohne dass eine globale Sicht auf die Daten vorhanden ist. Damit ist es nicht ohne Weiteres möglich, z. B. clustering Algorithmen für die Erstellung der Hierarchien zu benutzen. Allerdings können einfachere Funktionen, z. B. das inkrementelle Reduzieren der Präzision einer Gleitkommazahl, benutzt werden.

Generalisierungshierarchien sind nur erforderlich für Quasi-Identifikatoren und manchmal für sensitive Attribute. Typische Quasi-Identifikatoren sind kategorischer Natur [24]. Außerdem wird von diesen angenommen, dass sie ein hohes Reidentifikationsrisiko haben [128]. Damit müssen diese Attribute reproduzierbar sein, d. h. eine große Wahrscheinlichkeit besitzen, dass sie wiederholt für ein Individuum auftreten. Zusätzlich müssen diese mit einer hohen Wahrscheinlichkeit auch dem Angreifer bekannt sein [25]. Sollten kontinuierliche Variablen diese Eigenschaft haben, dann kann es sein, dass im horizontalen Fall, einfache Funktionen Repräsentationen nicht ausreichen, um eine globale Hierarchie zu erstellen. In so einem Fall könnte man untersuchen, inwieweit Kategorisierungsmethoden wie [129] mit sicheren Mengenvereinigungsprotokollen wie [130] kombiniert werden können.

4.8 Zwischenfazit und Perspektiven

In diesem Kapitel wurden drei neue Ansätze präsentiert, die es in einer datenschutzkonformen Weise ermöglichen, verteilt vorliegende Daten zu anonymisieren. Vereinfacht kann man sagen, dass die neu entwickelten Protokolle eine globale, verschlüsselte Sicht auf die Daten generieren, die im nächsten Schritt anonymisiert wird. Bei Protokoll C sind, aufgrund der höheren Sicherheitsgarantien, noch zusätzliche Schritte bei der eigentlichen Anonymisierung notwendig. Dennoch kann man feststellen, dass durch diesen Ansatz alle drei hier neu entwickelten Protokolle sehr flexibel sind und ein breites Spektrum an Anonymisierungskriterien und Algorithmen unterstützen. Damit können verschiedene Bedrohungen angegangen werden, da sowohl Kriterien gegen Identitätsaufdeckung, Attributsaufdeckung als auch Mitgliedsaufdeckung unterstützt werden. Außerdem sind heuristische als auch optimale Methoden implementierbar. In den Beispielen wurde auf k-Anonymität,

ℓ-Diversität, t-Closeness und δ-Präsenz, welche alle mittels des Flash Algorithmus umgesetzt wurden, fokussiert. Bei Protokoll C wurde nur k-Anonymität betrachtet.

Die Einsetzbarkeit in der Praxis wurde mittels experimenteller Evaluation gezeigt. Sowohl die zu übertragende Datenmenge als auch die benötigte Rechenzeit, für alle drei Protokolle bewegt sich in einem Rahmen, um es mit handelsüblicher Desktop Hardware zu bewältigen. Die vorgestellten Protokolle bieten, im Vergleich mit anderen Ansätzen, eine sehr gute Performanz. Außerdem ist die in dieser Arbeit erstellte Implementierung die erste effiziente, welche eine global optimale Lösung finden kann. Diese Klasse von Algorithmen ist besonders gut für die biomedizinische Forschung geeignet (siehe auch El Emam et al. [56]).

Die Konzepte können erweitert werden, sodass auch hybrid verteilte Daten anonymisiert werden können. In diesem Fall müssten die Tupel-Identifikatoren während der Integrationsphase beibehalten werden, da die unterschiedlichen Teilmengen dadurch korreliert werden können. Danach könnten die Protokolle analog, wie im horizontalen Fall, weiter ausgeführt werden.

Interessant wäre auch die Kombination der hier vorgestellten Protokolle mit Privatsphäre erhaltender Duplikaterkennung. Damit könnte die Anforderung an einen gemeinsamen Identifikator pro Datensatz, der für zusammengehörige Datensätze bei allen Parteien identisch ist, aufgegeben werden (siehe z. B. [131]). Eine weitere Möglichkeit, das hier vorgestellte Protokoll zu erweitern, ist die Erweiterung auf das Malicious-Modell (siehe Abschnitt 4.1.3.2). Ein erster Ansatzpunkt hierfür könnte die Arbeit [96] sein, welche spieltheoretische Ansätze nutzt, damit alle Parteien immer die korrekte Berechnung durchführen, da diese, aus ihrer Sicht, der beste nächste Schritt für sie wäre.

In der Zukunft wäre es auch denkbar, Protokoll C um andere Anonymisierungskriterien zu erweitern. Außerdem könnte man die drei Protokolle miteinander kombinieren. Interessant ist hierbei vor allem die Kombination von Protokoll A mit Protokoll C. Damit wäre es möglich, durch die kommutative Verschlüsselung, den Fall von zusammenarbeitenden Parteien abzumildern und außerdem durch das Einfügen von Pseudotupeln, Frequenzanalysen zu erschweren. Nachteil ist hierbei natürlich die längere Laufzeit und das erhöhte Datenvolumen.

KAPITEL 5

Diskussion und Ausblick

In dieser Arbeit wurde zunächst dargelegt, dass in der medizinischen Domäne zwei anwendungsseitig verwandte Probleme bearbeitet werden müssen: die Anonymisierung von Datenbeständen, die lokal oder verteilt vorliegen können. Die Zielsetzung ist in beiden Fällen gleich: die Daten sollen datenschutzkonform weiterverwendet werden.

Für beide Probleme wurden hier erstmals praktikable Algorithmen und Protokolle beschrieben, die sich umfangreich konfigurieren lassen und somit sehr flexibel sind. Sie erlauben das Anonymisieren mit einer Vielzahl von Anonymisierungskriterien auf üblicher Hardware und ermöglichen somit das iterative Anpassen der Anonymisierung an den jeweiligen Anwendungsfall. Es wurde gezeigt, dass Unterdrückung den Informationsverlust bei den hier getesteten Metriken signifikant reduzieren kann, aber einige Anonymisierungskriterien dann nicht mehr monoton sind und somit aktuelle Algorithmen nicht mehr effizient eine optimale Lösung finden können. Die Optimalität der gefundenen Lösung garantiert in dem hier betrachteten Recodierungsmodell minimalen Informationsverlust nach einer gewählten Metrik. Auf dieser Basis wurden für diese Problemstellung eine Familie von Algorithmen entwickelt. Der Anfang wurde mit einem sehr effizienten Algorithmus für den monotonen Fall gemacht. Dieser kann den Suchraum vorausschauend sehr stark verkleinern. Zusammen mit dem genutzten, hoch-skalierbaren Framework schlägt dieser die aktuellen Algorithmen hinsichtlich der Laufzeit. Dieser Algorithmus wurde so erweitert, dass auch bei nicht-monotonen Fällen erstmalig und immer noch sehr effizient, eine optimale Lösung gefunden werden kann. Dazu wurde die Tatsache genutzt, dass es auch bei diesen Fällen oft monotone Teilkriterien gibt, die es ermöglichen, Teile des Suchraumes auszuschließen. Zudem können die Algorithmen und das Framework erstmalig bei verschiedene Kombinationen von Anonymisierungskriterien angewendet werden, um den Schutzbedarf gemäß den Anforderungen zu justieren. Es ist nun zum Beispiel möglich, den Datensatz vor Identitätsaufdeckung und Attributsaufdeckung zu schützen. Bei der Entwicklung des Frameworks und der Algorithmen wurde zudem sehr großer Wert auf Benutzerfreundlichkeit gelegt. Dies führte dazu, dass es sowohl eine API für Entwickler gibt, die das Framework in ihre Projekte einbauen wollen, als auch ein umfangreiches graphisches Nutzerinterface. Beides ist open-source und verfügbar unter [132]. Das Framework und die Algorithmen wurden ausführlich evaluiert. Hierbei wurde festgestellt, dass die Anonymisierung, auch von großen Datensätzen, effizient möglich ist. Es können somit Nutzer mittels eines effizienten Prozesses ihre Daten anonymisieren. Zuerst konfigurieren sie

die gewünschten Datenschutzkriterien. Danach können sie den resultierenden Lösungsraum automatisch und/oder manuell durchsuchen. Als Letztes können sie den anonymisierten Datensatz auf die Nützlichkeit hinsichtlich des Anwendungsfalles untersuchen. All diese Schritte können durch die effizienten Implementierungen und entwickelten Algorithmen nahezu in Echtzeit erfolgen. Um den Anwender bei der Konfiguration zu unterstützen, werden für viele Parameter sinnvolle Standardwerte genutzt.

Für das verwandte Problem, dass die Daten verteilt vorliegen, wurde in dieser Arbeit ein Lösungskonzept erarbeitet. Bei diesem Konzept werden die Daten vor der Anonymisierung nicht bei einer zentralen Partei integriert. Zuerst wurde gezeigt, dass die naiven Ansätze entweder einen großen Informationsverlust in Kauf nehmen oder die gewünschten Datenschutzgarantien nicht einhalten können. Basierend auf dieser Grundlage wurde eine neue, praktikable, da effizient und einfach zu implementierende, Protokollfamilie vorgestellt. Die vorgestellten Protokolle sind hoch flexibel hinsichtlich der unterstützten Anonymisierungskriterien. Die Implementierung der Protokolle ist einfach, und die erzielten Laufzeiten sind bei der Flexibilität der Protokolle gering. Es wurden mehrere Protokollversionen vorgestellt, die einen trade-off zwischen Laufzeit, Kosten und Sicherheitsgarantien ermöglichen. Selbst das komplexeste der Protokolle erlaubt eine verteilte Anonymisierung bei drei Parteien und großen Datensätzen in praktikabler Zeit. Die zu übertragenden Datenmengen während der Protokollausführung lassen sich leicht mit Standardnetzwerkhardware bewältigen. Die neben der praktischen Evaluation durchgeführte analytische Evaluation erlaubt es, die hier vorgestellten Protokolle hardwareunabhängig mit anderen Protokollen zu vergleichen.

Die in dieser Arbeit vorgestellten Lösungskonzepte nutzen nicht-interaktive Anonymisierungsverfahren mittels Generalisierung und Unterdrückung, da diese Methoden für die medizinische Domäne gut geeignet sind (siehe Abschnitt 2.3). Daneben gibt es auch Anwendungsfälle bei denen interaktive Anonymisierungsverfahren sehr nützlich sein können. Hier sind allerdings methodische Einschränkungen zu erwarten (bspw. auf generalisierte lineare Modelle [84]), da jede Methode speziell angepasst werden muss. Im Gegensatz zu den in dieser Arbeit verwendeten „syntaktischen“ Verfahren, bei welchen die anonymisierten Daten gewissen Regeln genügen müssen (z. B. Gruppengröße), bietet Differential Privacy [49] (siehe auch Abschnitt 2.2) eine semantische Grundlage für die Anonymisierung. Bei den „syntaktischen“ Verfahren müssen umfangreiche Annahmen über das Hintergrundwissen der betrachteten Angreifer getroffen werden. „Semantische“ Verfahren, wie Differential Privacy, benötigen weniger Annahmen, gehen aber oft mit schlechterer Datenqualität und Einschränkungen bei den unterstützten Workflows einher [133]. Auch nicht-interaktive Ansätze können, in gewissen Grenzen, mit diesem Konzept realisiert werden [134]. Es sollte hervorgehoben werden, dass durch Ziehen einer Stichprobe, in Kombination mit Generalisierung und Unterdrückung, Daten dem Kriterium der (ϵ, δ)-Differential Privacy genügen [57] können.

Oft enthalten medizinische Forschungsdaten verschiedene Werte eines Attributs, die im zeitlichen Verlauf erhoben worden sind (sog. transactional data). In dem hier verwendeten Datenmodell müssten diese vor der Anonymisierung in ein Rechteckschema transformiert werden. Dabei kann bei der danach folgenden Anonymisierung ein unnötiger Informationsverlust auftreten. Aus diesem Grund sind angepasste Kri-

terien und Algorithmen in Zukunft von Interesse, siehe z. B. [135–137].

Zudem werden auch immer häufiger genetische Daten erhoben. Durch das schrittweise besser werdende Verständnis der resultierenden Bedrohungen werden immer neue Angriffsvektoren identifiziert [4, 5, 138, 139]. Verfahren zur Anonymisierung genetischer Daten, wie sie beispielsweise in [140–142] vorgestellt werden, sind noch nicht in der Praxis einsetzbar. Daraus resultiert ein zunehmender Bedarf an Weiterentwicklungen [143, 144]. Von zentraler Bedeutung ist, dass genomische Daten sehr stark singularisieren, da sie hoch-dimensional sind. Deshalb lassen sich die hier vorgestellten Verfahren nicht sinnvoll auf genomische Daten anwenden. Lediglich ein einziger Versuch wurde beschrieben [145].

Zudem ist das entstehende Risiko bei der gemeinsamen Herausgabe von phänotypischen und genotypischen Daten noch nicht vollständig verstanden. Auch hier gibt es einige Ansätze, das Risiko zu bestimmen [146–152]. Sie müssen weiterentwickelt und an die sich ändernden Umstände angepasst werden.

Um dennoch auch bei genetischen Daten den Datenschutz zu gewährleisten, müssen aktuell andere Techniken zum Einsatz kommen. Erlich und Narayanan [138] haben neben dem hier vorgestellten Ansatz der Anonymisierung Zugriffskontrollmechanismen und kryptographische Methoden vorgeschlagen. Im Gegensatz zu der Idee, Daten so zu verändern, dass keine Reidentifikation mehr möglich ist (Anonymisierung), wird der Zugriff auf die Daten eingeschränkt und protokolliert (Zugriffskontrolle). Auch vertragliche und gesetzliche Bedingungen hinsichtlich der Verwendung der Daten regulieren Zugriff und Verwendung. Sie spielen hier eine wichtige Rolle. Ein Data Use Agreement wird typischerweise unter anderem fordern, dass keine Reidentifikationsversuche unternommen werden, und die Daten auf Empfängerseite sicher verwahrt werden.

Secure multiparting Computing und homomorphe Verschlüsselung werden insbesondere bei den schwer zu anonymisierenden und gleichzeitig hoch sensiblen genetischen Daten eine zunehmende Rolle spielen [138]. Sie lassen Auswertungen auf verschlüsselten Daten zu, ohne dass die Originaldaten bekannt werden. Zu den Problemen zählt derzeit die Effizienz der Verfahren, z. B. die notwendigen Schlüsselgrößen im Gigabyte Bereich.

Aber auch bereits bei der Anonymisierung von phänotypischen Daten gibt es Diskussionen um die generelle Wirksamkeit verschiedener Techniken, siehe [153–156]. Generell ist anzumerken, dass es keine „one-size-fits-all“ Lösung gibt und wohl auch nicht geben wird. Die Bedrohungen ändern sich ständig, da neue Angriffsvektoren gefunden werden, welche ein Datenschutzproblem verursachen können. Dennoch ist eine „better-than-nothing“ Strategie in den meisten Fällen ausreichend, da die technischen Methoden auch immer flankiert werden sollten von Verträgen und Datenschutzgesetzen. Diese schützen die Probanden zusätzlich indem sie die Weitergabe, Speicherung, Verarbeitung und Nutzung der Daten regeln. Allerdings sollte bei allen technischen Methoden immer genau dokumentiert werden, gegen welche Bedrohungen, unter welchen Voraussetzungen, geschützt werden kann.

Je besser die Bedrohungen, Risiken und Chancen verstanden werden, umso zielgerichteter können die vorhandenen Techniken eingesetzt werden, um die Daten der Probanden zu schützen. Die hier vorgestellten Protokolle und Algorithmen, die es erlauben den Anonymisierungsprozess iterativ zu verfeinern und zu verbessern, sind ein wichtiger Schritt um Daten in der Praxis praktikabel und effektiv zu schützen.

Literaturverzeichnis

[1] Bundestag. „Bundesdatenschutzgesetz in der Fassung der Bekanntmachung vom 14. Januar 2003 (BGBl. I S. 66), das zuletzt durch Artikel 1 des Gesetzes vom 14. August 2009 (BGBl. I S. 2814) geändert worden ist“. In: *Bundesgesetzblatt* (2009).

[2] U.S. Department of Health and Human Services Office for Civil Rights. *HIPAA Administrative Simplification - Regulation Text (45 CFR Parts 160, 162, and 164)*. 2013.

[3] U.S. Department of Health & Human Services. *Guidance Regarding Methods for De-identification of Protected Health Information in Accordance with the Health Insurance Portability and Accountability Act (HIPAA) Privacy Rule*. Accessed: 2014-10-27. URL: `http://www.hhs.gov/ocr/privacy/hipaa/understanding/coveredentities/De-identification/guidance.html`.

[4] Amy L. McGuire and Richard A. Gibbs. „No Longer De-Identified“. In: *Science* 312.5772 (2006), pp. 370–371. DOI: `10.1126/science.1125339`.

[5] Jennifer Couzin. „Genetic privacy. Whole-genome data not anonymous, challenging assumptions“. In: *Science* 321.5894 (Sept. 2008), p. 1278. DOI: `10.1126/science.321.5894.1278`.

[6] Dov Greenbaum, Jiang Du, and Mark Gerstein. „Genomic anonymity: have we already lost it?“ In: *The American Journal of Bioethics* 8.10 (2008), pp. 71–74. DOI: `10.1080/15265160802478560`.

[7] C. Heeney, N. Hawkins, J. de Vries, P. Boddington, and J. Kaye. „Assessing the privacy risks of data sharing in genomics“. In: *Public Health Genomics* 14.1 (2011), pp. 17–25. DOI: `10.1159/000294150`.

[8] Martin Mahnera and Michael Karyb. „What Exactly Are Genomes, Genotypes and Phenotypes? And What About Phenomes?“ In: *Journal of Theoretical Biology* 186.1 (1997), pp. 55–63. DOI: `10.1006/jtbi.1996.0335`.

[9] Michael L Metzker. „Sequencing technologies - the next generation“. In: *Nature Reviews Genetics* 11.1 (2010), pp. 31–46. DOI: `10.1038/nrg2626`.

[10] Monya Baker. „Biorepositories: Building better biobanks“. In: *Nature* 486 (2012), pp. 141–146. DOI: `10.1038/486141a`.

[11] Charu C. Aggarwal. „On k-anonymity and the Curse of Dimensionality“. In: *Proceedings of the 31st International Conference on Very Large Data Bases*. 2005, pp. 901–909.

[12] C. Rodwell and S. Aymé. *2014 Report on the State of the Art of Rare Disease Activities in Europe*. 2014. URL: http://www.eucerd.eu/upload/file/Reports/2014ReportStateofArtRDActivities.pdf.

[13] International Cancer Genome Consortium. *B. Consortium Goals*. Accessed: 2014-10-27. URL: https://icgc.org/icgc/goals-structure-policies-guidelines/b-consortium-goals.

[14] National Cancer Institute at NIH. *Home - The Cancer Genome Atlas - Cancer Genome - TCGA*. Accessed: 2014-10-27. URL: http://cancergenome.nih.gov/.

[15] BioMedBridges. *Home | BioMedBridges*. Accessed: 2014-10-27. URL: http://www.biomedbridges.eu/.

[16] Markus Perola and Gert-Jan van Ommen. *BBMRI-LPC - A four-year project to help scientists to have better access to large European studies on health*. Accessed: 2014-10-27. 2013. URL: http://bbmri.se/en/About-us/News-archive/BBMRI-LPC-an-EU-project-to-help-scientists-to-have-better-access-to-large-European-studies-on-Health/.

[17] Global Alliance. *Home | Global Alliance for Genomics and Health*. Accessed: 2014-10-27. URL: http://genomicsandhealth.org/.

[18] Research Data Alliance. *RDA | Research Data Sharing without barriers*. Accessed: 2014-10-27. URL: https://rd-alliance.org/.

[19] Bartha Knoppers, Jennifer Harris, Anne Tasse, Isabelle Budin-Ljosne, Jane Kaye, Mylene Deschenes, and Ma'n Zawati. „Towards a data sharing Code of Conduct for international genomic research". In: *Genome Medicine* 3.7 (2011), p. 46. DOI: 10.1186/gm262.

[20] Laura J. Damschroder, Joy L. Pritts, Michael A. Neblo, Rosemarie J. Kalarickal, John W. Creswell, and Rodney A. Hayward. „Patients, privacy and trust: Patients' willingness to allow researchers to access their medical records". In: *Social Science & Medicine* 64.1 (2007), pp. 223 –235. DOI: 10.1016/j.socscimed.2006.08.045.

[21] Unabhängiges Landeszentrum für Datenschutz Schleswig-Holstein. *Datenschutzrechtliches Gutachten - Datentreuhänderschaft in der Biobank-Forschung*. Tech. rep. April. ULD, 2009, pp. 1–93.

[22] Roberta B. Ness. „Influence of the HIPAA Privacy Rule on Health Research". In: *JAMA* 298.18 (2007), pp. 2164–2170. DOI: 10.1001/jama.298.18.2164.

[23] Bradley Malin, David Karp, and Richard H. Scheuermann. „Technical and Policy Approaches to Balancing Patient Privacy and Data Sharing in Clinical and Translational Research". In: *Journal of investigative medicine: the official publication of the American Federation for Clinical Research* 58.1 (2010), pp. 11–18. DOI: 10.231/JIM.0b013e3181c9b2ea.

[24] Khaled El Emam, Elizabeth Jonker, Luk Arbuckle, and Bradley Malin. „A systematic review of re-identification attacks on health data". In: *PloS one* 6.12 (Jan. 2011). DOI: 10.1371/journal.pone.0028071.

[25] Bradley Malin, Grigorios Loukides, Kathleen Benitez, and EllenWright Clayton. „Identifiability in biobanks: models, measures, and mitigation strategies“. In: *Human Genetics* 130.3 (2011), pp. 383–392. DOI: 10.1007/s00439-011-1042-5.

[26] National Center for Biotechnology Information. *Home - dbGaP - NCBI*. Accessed: 2014-10-27. URL: http://www.ncbi.nlm.nih.gov/gap/.

[27] EMBL-EBI. *European Genome-phenome Archive*. Accessed: 2014-10-27. URL: https://www.ebi.ac.uk/ega/home.

[28] Landtag des Freistaates Bayern. „Bayerisches Datenschutzgesetz (BayDSG) vom 23. Juli 1993 zuletzt geändert am 22. Juli 2014 (GVBl 2014, S. 286)“. In: *Gesetz- und Verordnungsblatt* (2014).

[29] European Parliament. *Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data*. Official Journal of the European Communities. Nov. 1995.

[30] Europäische Kommission. *Verordnung des Europäischen Parlaments und des Rates zum Schutz natürlicher Personen bei der Verarbeitung personenbezogener Daten und zum freien Datenverkehr (Datenschutz-Grundverordnung)*. Jan. 2012.

[31] Committee of Ministers. *Recommendation Rec(2006)4 of the Committee of Ministers to member states on research on biological materials of human origin*. Mar. 2006.

[32] Raffael Bild, Florian Kohlmayer, Sabine Brunner, Klaus Kuhn, Benedicto Rodriguez, Ashish Lamichhane, Stefan Klein, Michael Raess, Christoph Lengger, Philipp Gormanns, Christian Ohmann, Wolfgang Kuchinke, Ugis Sarkans, Murat Sariyar, Chris Morris, Tommi Nyrönen, and Jaakko Leinonen. *Report describing the security architecture and framework (D5.3)*. Tech. rep. Accessed: 2014-10-27. BioMedBridges WP5, 2014. URL: http://www.biomedbridges.eu/deliverables/53.

[33] Florian Kohlmayer, Raffael Bild, Imre Västrik, Klaus Kuhn, Benedicto Rodriguez, Sabine Brunner, Ashish Lamichhane, and Christian Ohmann. *Process specification for secure sharing of and access to PM data (D8.1)*. Tech. rep. Accessed: 2014-10-27. BioMedBridges WP8, 2014. URL: http://www.biomedbridges.eu/deliverables/81.

[34] Florian Kohlmayer, Ronald R. Lautenschläger, Sebastian H. R. Wurst, Thomas Klopstock, Holger Prokisch, Thomas Meitinger, Claudia Eckert, and Klaus A. Kuhn. „Konzept für ein deutschlandweites Krankheitsnetz am Beispiel von mitoREGISTER“. In: *Informatik 2010: Service Science - Neue Perspektiven für die Informatik, Beiträge der 40. Jahrestagung der Gesellschaft für Informatik e.V.* Vol. P-176. 2010, pp. 746–751.

[35] Statistische Ämter des Bundes und der Länder. *Datenzugang | Anonymität von Mikrodaten*. Accessed: 2014-10-27. URL: http://www.forschungsdatenzentrum.de/anonymisierung.asp.

[36] Kathleen Benitez and Bradley Malin. „Evaluating re-identification risks with respect to the HIPAA privacy rule“. In: *Journal of the American Medical Informatics Association* 17.2 (2010), pp. 169–177. DOI: 10.1136/jamia.2009.000026.

[37] Bradley Malin, Kathleen Benitez, and Daniel R. Masys. „Never too old for anonymity: a statistical standard for demographic data sharing via the HIPAA Privacy Rule“. In: *Journal of the American Medical Informatics Association* 18.1 (2011), pp. 3–10. DOI: 10.1136/jamia.2010.004622.

[38] Grigorios Loukides, Joshua C. Denny, and Bradley Malin. „The disclosure of diagnosis codes can breach research participants' privacy“. In: *Journal of the American Medical Informatics Association* 17.3 (2010), pp. 322–327. DOI: 10.1136/jamia.2009.002725.

[39] International Organization for Standardization. *Health informatics - Pseudonymization (ISO/TS 25237:2008)*. 2008.

[40] Bradley A. Malin. „Technical Evaluation: An Evaluation of the Current State of Genomic Data Privacy Protection Technology and a Roadmap for the Future“. In: *Journal of the American Medical Informatics Association* 12.1 (2005), pp. 28–34. DOI: 10.1197/jamia.M1603.

[41] Bradley Malin and Latanya Sweeney. „How (not) to protect genomic data privacy in a distributed network: using trail re-identification to evaluate and design anonymity protection systems“. In: *Journal of Biomedical Informatics* 37.3 (2004), pp. 179–192. DOI: 10.1016/j.jbi.2004.04.005.

[42] Article 29 Data Protection Working Party. *Opinion 05/2014 on Anonymisation Techniques*. Accessed: 2014-10-27. Apr. 2014. URL: http://ec.europa.eu/justice/data-protection/article-29/documentation/opinion-recommendation/files/2014/wp216_en.pdf.

[43] Latanya Sweeney. *Simple demographics often identify people uniquely*. Tech. rep. Carnegie Mellon University, 2000, pp. 1–34. URL: http://dataprivacylab.org/projects/identifiability/paper1.pdf.

[44] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. „L-diversity: Privacy beyond k-anonymity“. In: *Transactions on Knowledge Discovery from Data* 1.1 (Mar. 2007). DOI: 10.1145/1217299.1217302.

[45] Rajeev Motwani and Ying Xu. „Efficient Algorithms for Masking and Finding Quasi-Identifiers“. In: *Proceedings of the Conference on Very Large Data Bases (VLDB)*. 2007, pp. 83–93.

[46] George T Duncan and Diane Lambert. „Disclosure-limited data dissemination“. In: *Journal of The American Statistical Association* 81.393 (1986), pp. 10–18. DOI: 10.1080/01621459.1986.10478229.

[47] Diane Lambert. „Measures of Disclosure Risk and Harm“. In: *Journal of Official Statistics* 9.2 (1993), pp. 313 –331.

[48] Mehmet Ercan Nergiz, Maurizio Atzori, and Chris Clifton. „Hiding the presence of individuals from shared databases“. In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 2007, pp. 665–676. DOI: 10.1145/1247480.1247554.

[49] Cynthia Dwork. „Differential privacy“. In: *Proceedings of the 33rd International Colloquium on Automata, Languages and Programming*. Vol. 4052. Springer LNCS, 2006, pp. 1–12. DOI: 10.1007/11787006_1.

[50] Michael Wolfson, Susan E Wallace, Nicholas Masca, Geoff Rowe, Nuala A Sheehan, Vincent Ferretti, Philippe LaFlamme, Martin D Tobin, John Macleod, Julian Little, Isabel Fortier, Bartha M Knoppers, and Paul R Burton. „DataSHIELD: resolving a conflict in contemporary bioscience - performing a pooled analysis of individual-level data without sharing the data“. In: *International Journal of Epidemiology* 39.5 (2010), pp. 1372–1382. DOI: 10.1093/ije/dyq111.

[51] Andrew J. McMurry, Shawn N. Murphy, Douglas MacFadden, Griffin Weber, William W. Simons, John Orechia, Jonathan Bickel, Nich Wattanasin, Clint Gilbert, Philip Trevvett, Susanne Churchill, and Isaac S. Kohane. „SHRINE: Enabling Nationally Scalable Multi-Site Disease Studies“. In: *PLoS ONE* 8.3 (2013). DOI: 10.1371/journal.pone.0055811.

[52] Charu C. Aggarwal and Philip S. Yu. *Privacy-Preserving Data Mining: Models and Algorithms*. Springer, 2008, p. 514. ISBN: 9780387709918.

[53] Benjamin C.M. Fung, Ke Wang, Ada Wai-Chee Fu, and Philip S. Yu. *Introduction to Privacy-Preserving Data Publishing: Concepts and Techniques*. Chapman and Hall/CRC, 2011, p. 376. ISBN: 1420091484.

[54] Fida Kamal Dankar and Khaled El Emam. „The application of differential privacy to health data“. In: *Proceedings of the 2012 Joint EDBT/ICDT Workshops*. 2012, pp. 158–166. DOI: 10.1145/2320765.2320816.

[55] Jiuyong Li, Jixue Liu, Muzammil M. Baig, and Raymond Chi-Wing Wong. „Information based data anonymization for classification utility“. In: *Data & Knowledge Engineering* 70.12 (2011), pp. 1030–1045. DOI: 10.1016/j.datak.2011.07.001.

[56] Khaled El Emam, Fida Kamal Dankar, Romeo Issa, Elizabeth Jonker, Daniel Amyot, Elise Cogo, Jean-Pierre Corriveau, Mark Walker, Sadrul Chowdhury, Regis Vaillancourt, Tyson Roffey, and Jim Bottomley. „A Globally Optimal k-Anonymity Method for the De-Identification of Health Data“. In: *Journal of the American Medical Informatics Association* 16.5 (2009), pp. 670–682. DOI: 10.1197/jamia.M3144.

[57] Ninghui Li, Wahbeh H. Qardaji, and Dong Su. „Provably Private Data Anonymization: Or, k-Anonymity Meets Differential Privacy“. In: *CoRR* abs/1101.2604 (2011). URL: http://arxiv.org/abs/1101.2604.

[58] Latanya Sweeney. „Achieving k-Anonymity Privacy Protection Using Generalization and Suppression“. In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 10.5 (2002), pp. 571–588. DOI: 10.1142/S021848850200165X.

[59] Latanya Sweeney. „k-Anonymity: A Model for Protecting Privacy“. In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 10.5 (2002), pp. 557–570. DOI: 10.1142/S0218488502001648.

[60] Pierangela Samarati and Latanya Sweeney. *Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression*. Tech. rep. Technical report, SRI International, 1998. URL: http://epic.org/privacy/reidentification/Samarati_Sweeney_paper.pdf.

[61] Pierangela Samarati. „Protecting Respondents' Identities in Microdata Release“. In: *IEEE Transactions on Knowledge and Data Engineering* 13.6 (2001), pp. 1010–1027. DOI: 10.1109/69.971193.

[62] Committee on Strategies for Responsible Sharing of Clinical Trial Data; Board on Health Sciences Policy; Institute of Medicine. *Sharing Clinical Trial Data: Maximizing Benefits, Minimizing Risk*. The National Academies Press, 2015, p. 280. ISBN: 978-0-309-36732-5.

[63] Ninghui Li, Tiancheng Li, and Suresh Venkatasubramanian. „t-Closeness: Privacy Beyond k-Anonymity and l-Diversity“. In: *Proceedings of the 23rd International Conference on Data Engineering*. 2007, pp. 106–115. DOI: 10.1109/ICDE.2007.367856.

[64] Ninghui Li, Tiancheng Li, and Suresh Venkatasubramanian. „Closeness: A New Privacy Measure for Data Publishing“. In: *IEEE Transactions on Knowledge and Data Engineering* 22.7 (2010), pp. 943–956. DOI: 10.1109/TKDE.2009.139.

[65] Gagan Aggarwal, Samir Khuller, and Tomás Feder. „Achieving Anonymity via Clustering“. In: *Symposium on Principles of Database Systems*. 2006, pp. 153–162. DOI: 10.1145/1798596.1798602.

[66] Kevin Brown. *Balls In Bins With Limited Capacity*. Accessed: 2014-10-27. URL: http://www.mathpages.com/home/kmath337.htm.

[67] Aristides Gionis and Tamir Tassa. „k-Anonymization with Minimal Loss of Information“. In: *IEEE Transactions on Knowledge and Data Engineering* 21.2 (Feb. 2009), pp. 206–219. DOI: 10.1109/TKDE.2008.129.

[68] Vijay S. Iyengar. „Transforming data to satisfy privacy constraints“. In: *Proceedings of the 8th International Conference on Knowledge Discovery and Data Mining*. 2002, pp. 279–288. DOI: 10.1145/775047.775089.

[69] Kristen LeFevre, David J. DeWitt, and Raghu Ramakrishnan. *Multidimensional K-Anonymity (TR-1521)*. Tech. rep. University of Wisconsin, 2005. URL: http://research.cs.wisc.edu/techreports/2005/TR1521.pdf.

[70] Roberto J. Bayardo and Rakesh Agrawal. „Data Privacy through Optimal k-Anonymization“. In: *Proceedings of the 21st International Conference on Data Engineering*. 2005, pp. 217–228. DOI: 10.1109/ICDE.2005.42.

[71] Ton de Waal and Leon Willenborg. „Information loss through global recoding and local suppression“. In: *Netherlands Official Statistics* 14 (1999), pp. 17–20.

[72] Florian Kohlmayer, Fabian Prasser, Claudia Eckert, Alfons Kemper, and Klaus A. Kuhn. „Highly efficient optimal k-anonymity for biomedical datasets“. In: *Proceedings of the 25th International Symposium on Computer-Based Medical Systems*. June 2012, pp. 1–6. DOI: 10.1109/CBMS.2012.6266366.

[73] Florian Kohlmayer, Fabian Prasser, Claudia Eckert, Alfons Kemper, and Klaus A. Kuhn. „Flash: Efficient, Stable and Optimal K-Anonymity“. In: *Proceedings of the 2012 International Conference on Privacy, Security, Risk and Trust, and 2012 International Confernece on Social Computing*. Sept. 2012, pp. 708–717. DOI: 10.1109/SocialCom-PASSAT.2012.52.

[74] Fabian Prasser, Florian Kohlmayer, and Klaus A. Kuhn. „A Benchmark of Globally-Optimal Anonymization Methods for Biomedical Data“. In: *Proceedings of the 27th International Symposium on Computer-Based Medical Systems*. 2014, pp. 66–71. DOI: 10.1109/CBMS.2014.85.

[75] Fabian Prasser, Florian Kohlmayer, and Klaus A. Kuhn. „ARX - A Comprehensive Tool for Anonymizing Biomedical Data“. In: *Proceedings of the AMIA 2014 Annual Symposium*. Nov. 2014, pp. 984–993.

[76] Florian Kohlmayer, Fabian Prasser, and Klaus A. Kuhn. „The cost of quality: implementing generalization and suppression for anonymizing biomedical data with minimal information loss“. In: *Journal of Biomedical Informatics* 58 (2015), pp. 37–48. DOI: 10.1016/j.jbi.2015.09.007.

[77] Adam Meyerson and Ryan Williams. „On the Complexity of Optimal K-Anonymity“. In: *Proceedings of the 23rd Symposium on Principles of Database Systems*. 2004, pp. 223–228. DOI: 10.1145/1055558.1055591.

[78] Mehmet Ercan Nergiz and Christopher W. Clifton. „d-Presence without Complete World Knowledge“. In: *IEEE Transactions on Knowledge and Data Engineering* 22.6 (2010), pp. 868–883. DOI: 10.1109/TKDE.2009.125.

[79] Kristen LeFevre, David J. DeWitt, and Raghu Ramakrishnan. „Incognito: Efficient Full-Domain K-Anonymity“. In: *Proceedings of the 2005 International Conference on Management of Data*. 2005, pp. 49–60. DOI: 10.1145/1066157.1066164.

[80] Fabian Prasser. „Incremental Ontology-Based Integration for Translational Medical Research“. PhD thesis. TU München, 2013.

[81] Tiancheng Li, Ninghui Li, Jian Zhang, and Ian Molloy. „Slicing: A New Approach for Privacy Preserving Data Publishing“. In: *Transactions on Knowledge and Data Engineering* 24.3 (2012), pp. 561–574. DOI: 10.1109/TKDE.2010.236.

[82] Benjamin C. M. Fung, Ke Wang, and Philip S. Yu. „Top-Down Specialization for Information and Privacy Preservation“. In: *Proceedings of the 21st International Conference on Data Engineering*. 2005, pp. 205–216. DOI: 10.1109/ICDE.2005.143.

[83] Jacob Goldberger and Tamir Tassa. „Efficient Anonymizations with Enhanced Utility“. In: *Transactions on Data Privacy* 3.2 (2010), pp. 149–175.

[84] E.M. Jones, N.A. Sheehan, N. Masca, S.E. Wallace, M.J. Murtagh, and P.R. Burton. „DataSHIELD - shared individual-level analysis without sharing the data: a biostatistical perspective“. In: *Norsk epidemiologi* 21.2 (2012). URL: http://www.ntnu.no/ojs/index.php/norepid/article/view/1499.

[85] Noman Mohammed, Benjamin C. M. Fung, Patrick C. K. Hung, and Cheuk kwong Lee. „Centralized and Distributed Anonymization for High-Dimensional Healthcare Data“. In: *Transactions on Knowledge Discovery from Data* 4.4 (2010), p. 18. DOI: 10.1145/1857947.1857950.

[86] Florian Kohlmayer, Fabian Prasser, Claudia Eckert, and Klaus A. Kuhn. „A flexible approach to distributed data anonymization“. In: *Journal of Biomedical Informatics* 50 (2014), pp. 62–76. DOI: 10.1016/j.jbi.2013.12.002.

[87] Philipp Zieris. „Konzeption und prototypische Umsetzung eines k-Anonymity Algorithmus bei verteilten Daten“. MA thesis. TU München, 2011.

[88] Ramez Elmasri and Shamkrant Navathe. *Fundamentals of database systems*. Addison-Wesley, 2011, p. 1172. ISBN: 0136086209.

[89] Pawel Jurczyk and Li Xiong. „Towards privacy-preserving integration of distributed heterogeneous data“. In: *Proceedings of the Second Ph.D. Workshop on Information and Knowledge Management*. 2008, pp. 65–72. DOI: 10.1145/1458550.1458562.

[90] Oded Goldreich. *Foundations of Cryptography: Volume 2, Basic Applications*. Cambridge University Press, 2009, p. 452. ISBN: 052111991X.

[91] Mihir Bellare, Anand Desai, David Pointcheval, and Phillip Rogaway. „Relations Among Notions of Security for Public-Key Encryption Schemes“. In: *Proceedings of the 18th Annual International Cryptology Conference*. 1998, pp. 26–45. DOI: 10.1007/BFb0055718.

[92] Andrew Chi-Chih Yao. „How to Generate and Exchange Secrets (Extended Abstract)“. In: *Proccedings of the 27th Annual Symposium on Foundations of Computer Science*. 1986, pp. 162–167. DOI: 10.1109/SFCS.1986.25.

[93] Wei Jiang and Chris Clifton. „A secure distributed framework for achieving k-anonymity“. In: *The International Journal on Very Large Databases* 15.4 (2006), pp. 316–333. DOI: 10.1007/s00778-006-0008-z.

[94] Ke Wang, Benjamin C. M. Fung, and Guozhu Dong. „Integrating Private Databases for Data Analysis“. In: *Proceedings of the 2005 International Conference on Intelligence and Security Informatics*. 2005, pp. 171–182. DOI: 10.1007/11427995_14.

[95] Noman Mohammed, Benjamin C. M. Fung, Ke Wang, and Patrick C. K. Hung. „Privacy-preserving data mashup“. In: *Proceedings of the 12th International Conference on Extending Database Technology*. 2009, pp. 228–239. DOI: 10.1145/1516360.1516388.

[96] Noman Mohammed, Benjamin C. M. Fung, and Mourad Debbabi. „Anonymity meets game theory: secure data integration with malicious participants“. In: *The International Journal on Very Large Databases* 20.4 (2011), pp. 567–588. DOI: 10.1007/s00778-010-0214-6.

[97] Bruce Schneier. *Applied Cryptography: Protocols, Algorithms, and Source Code in C, Second Edition*. John Wiley & Sons, 1995, p. 784. ISBN: 0471117099.

[98] Pawel Jurczyk and Li Xiong. „Distributed Anonymization: Achieving Privacy for Both Data Subjects and Data Providers“. In: *Proceedings of the 23rd Working Conference on Data and Applications Security*. 2009, pp. 191–207. DOI: 10.1007/978-3-642-03007-9_13.

[99] Pawel Jurczyk and Li Xiong. *Privacy-Preserving Data Publishing for Horizontally Partitioned Databases (TR-2008-013)*. Tech. rep. Accessed: 2014-10-27. Mathematics and Computer Science Emory University, 2008. URL: http://www.mathcs.emory.edu/technical-reports/techrep-00138.pdf.

[100] Kristen LeFevre, David J. DeWitt, and Raghu Ramakrishnan. „Mondrian Multidimensional K-Anonymity“. In: *Proceedings of the 22nd International Conference on Data Engineering*. 2006, p. 25. DOI: 10.1109/ICDE.2006.101.

[101] Sheng Zhong, Zhiqiang Yang, and Rebecca N. Wright. „Privacy-enhancing k-anonymization of customer data“. In: *Proceedings of the 24th Symposium on Principles of Database Systems*. 2005, pp. 139–147. DOI: 10.1145/1065167.1065185.

[102] Adi Shamir. „How to Share a Secret“. In: *Communications of the ACM* 22.11 (1979), pp. 612–613. DOI: 10.1145/359168.359176.

[103] Tamir Tassa and Ehud Gudes. „Secure distributed computation of anonymized views of shared databases“. In: *ACM Transactions on Database Systems* 37.2 (2012), p. 11. DOI: 10.1145/2188349.2188353.

[104] T. Suga, T. Nishide, and K. Sakurai. „Weakness of provably secure searchable encryption against frequency analysis“. In: *Proceedings of the 2012 International Conference on Communications, Computers and Applications*. Oct. 2012, pp. 142–147.

[105] Mihir Bellare, Alexandra Boldyreva, and Adam O'Neill. „Deterministic and Efficiently Searchable Encryption“. In: *Proceedings of the 27th Annual International Cryptology Conference*. 2007, pp. 535–552. DOI: 10.1007/978-3-540-74143-5_30.

[106] Klaus Becker and Uta Wille. „Communication Complexity of Group Key Distribution“. In: *Proceedings of the 5th Conference on Computer and Communications Security*. 1998, pp. 1–6. DOI: 10.1145/288090.288094.

[107] Murat Kantarcioglu and Chris Clifton. „Privacy-Preserving Distributed Mining of Association Rules on Horizontally Partitioned Data“. In: *IEEE Transactions on Knowledge and Data Engineering* 16.9 (2004), pp. 1026–1037. DOI: 10.1109/TKDE.2004.45.

[108] Sam Wagstaff and Samuel S. Wagstaff Jr. *Cryptanalysis of Number Theoretic Ciphers*. Chapman and Hall/CRC, 2002. ISBN: 1584881534.

[109] Pascal Paillier. „Public-Key Cryptosystems Based on Composite Degree Residuosity Classes“. In: *Proceedings of the 17th International Conference on the Theory and Application of Cryptographic Techniques*. 1999, pp. 223–238. DOI: 10.1007/3-540-48910-X_16.

[110] Free Software Foundation. *The GNU Multiple Precision Arithmetic Library*. Accessed: 2014-10-27. URL: http://gmplib.org/.

[111] Oracle. *Java SE 7 Java Native Interface-related APIs and Developer Guides*. Accessed: 2014-10-27. URL: http://docs.oracle.com/javase/7/docs/technotes/guides/jni/index.html.

[112] I2P. *JBIGI*. Accessed: 2014-10-27. 2011. URL: https://geti2p.net/en/misc/jbigi.

[113] The Legion of the Bouncy Castle. *Bouncy Castle Java Crypto API*. Accessed: 2014-10-27. 2012. URL: https://www.bouncycastle.org/.

[114] Dain Sundstrom. *Snappy in Java*. Accessed: 2014-10-27. 2011. URL: https://github.com/dain/snappy.

[115] RSA Laboratories. *The RSA Factoring Challenge*. 1991. URL: https://www.rsa.com/rsalabs/node.asp?id=2092.

[116] Elaine Barker, William Barker, William Burr, William Polk, and Miles Smid. *Recommendation for Key Management - Part 1: General (Revision 3)*. July 2012.

[117] Wade Trappe and Lawrence C. Washington. *Introduction to Cryptography with Coding Theory (2nd Edition)*. Pearson, 2006. ISBN: 0130618144.

[118] ANSI. *Public Key Cryptography for the Financial Services Industry, The Elliptic Curve Digital Signature Algorithm (ECDSA) (X9.62:2005)*. American National Standards Institute, 2005.

[119] Nitin Jain, Saibal K. Pal, and Dhananjay K. Upadhyay. „Implementation and Analysis of Homomorphic Encryption Schemes“. In: *International Journal on Cryptography and Information Security* 2.2 (June 2012), pp. 27 –44. DOI: 10.5121/ijcis.2012.2203.

[120] Mehmet Ercan Nergiz and Chris Clifton. „Thoughts on k-anonymization“. In: *Data & Knowledge Engineering* 63.3 (2007), pp. 622–645. DOI: 10.1016/j.datak.2007.03.009.

[121] H. Ye and E. S. Chen. „Attribute Utility Motivated k-anonymization of datasets to support the heterogeneous needs of biomedical researchers“. In: *Proceedings of the 2011 AMIA Annual Symposium* 2011 (2011), pp. 1573–1582.

[122] Qing Zhang, Nick Koudas, Divesh Srivastava, and Ting Yu. „Aggregate Query Answering on Anonymized Tables“. In: *Proceedings of the 23rd International Conference on Data Engineering*. 2007, pp. 116–125. DOI: 10.1109/ICDE.2007.367857.

[123] Tiancheng Li and Ninghui Li. „Injector: Mining Background Knowledge for Data Anonymization“. In: *Proceedings of the 24th International Conference on Data Engineering*. 2008, pp. 446–455. DOI: 10.1109/ICDE.2008.4497453.

[124] Ting Wang, Shicong Meng, Bhuvan Bamba, Ling Liu, and Calton Pu. „A General Proximity Privacy Principle“. In: *Proceedings of the 25th International Conference on Data Engineering*. 2009, pp. 1279–1282. DOI: 10.1109/ICDE.2009.220.

[125] Nilothpal Talukder, Mourad Ouzzani, Ahmed K Elmagarmid, and Mohamed Yakout. *Detecting inconsistencies in private data with secure function evaluation (TR-11-006)*. Tech. rep. Department of Computer Science Purdue University, 2011. URL: http://docs.lib.purdue.edu/cgi/viewcontent.cgi?article=2753&context=cstech.

[126] Geetha Jagannathan and Rebecca N. Wright. „Privacy-preserving imputation of missing data“. In: *Data & Knowledge Engineering* 65.1 (2008), pp. 40–56. DOI: 10.1016/j.datak.2007.06.013.

[127] T. Dierks and E. Rescorla. *The Transport Layer Security (TLS) Protocol (RFC5246)*. Internet Engineering Task Force, 2008. URL: http://tools.ietf.org/rfc/rfc5246.txt.

[128] Bradley Malin. „Secure construction of k-unlinkable patient records from distributed providers“. In: *Artificial Intelligence in Medicine* 48.1 (2010), pp. 29–41. DOI: 10.1016/j.artmed.2009.09.002.

[129] Irantzu Barrio, Inmaculada Arostegui, Jose Quintana, and IRYSS-COPD Group. „Use of generalised additive models to categorise continuous variables in clinical prediction“. In: *BMC Medical Research Methodology* 13.1 (June 2013), p. 83. DOI: 10.1186/1471-2288-13-83.

[130] Lea Kissner and Dawn Xiaodong Song. „Privacy-Preserving Set Operations“. In: *Proceedings of the 25th Annual International Cryptology Conference*. 2005, pp. 241–257. DOI: 10.1007/11535218_15.

[131] Mehmet Kuzu, Murat Kantarcioglu, Ali Inan, Elisa Bertino, Elizabeth Durham, and Bradley Malin. „Efficient privacy-aware record integration“. In: *Proceedings of the 16th International Conference on Extending Database Technology*. 2013, pp. 167–178. DOI: 10.1145/2452376.2452398.

[132] Florian Kohlmayer and Fabian Prasser. *arx-deidentifier/arx*. Accessed: 2014-10-27. URL: https://github.com/arx-deidentifier/arx.

[133] Fida Kamal Dankar and Khaled El Emam. „Practicing Differential Privacy in Health Care: A Review“. In: *Transactions on Data Privacy* 6.1 (2013), pp. 35–67. URL: http://www.tdp.cat/issues11/abs.a129a13.php.

[134] David Leoni. „Non-interactive differential privacy: a survey“. In: *Proceedings of the First International Workshop on Open Data*. 2012, pp. 40–52. DOI: 10.1145/2422604.2422611.

[135] Manolis Terrovitis, Nikos Mamoulis, and Panos Kalnis. „Local and global recoding methods for anonymizing set-valued data“. In: *The International Journal on Very Large Databases* 20.1 (2011), pp. 83–106. DOI: 10.1007/s00778-010-0192-8.

[136] Aris Gkoulalas-Divanis and Grigorios Loukides. „PCTA: privacy-constrained clustering-based transaction data anonymization“. In: *Proceedings of the 2011 International Workshop on Privacy and Anonymity in Information Society*. 2011, p. 5. DOI: 10.1145/1971690.1971695.

[137] Aris Gkoulalas-Divanis, Grigorios Loukides, and Jimeng Sun. „Publishing data from electronic health records while preserving privacy: A survey of algorithms“. In: *Journal of Biomedical Informatics* 50 (2014), pp. 4–19. DOI: 10.1016/j.jbi.2014.06.002.

[138] Yaniv Erlich and Arvind Narayanan. „Routes for breaching and protecting genetic privacy“. In: *CoRR* abs/1310.3197 (2013). URL: http://arxiv.org/abs/1310.3197.

[139] Melissa Gymrek, Amy L. McGuire, David Golan, Eran Halperin, and Yaniv Erlich. „Identifying Personal Genomes by Surname Inference“. In: *Science* 339.6117 (2013), pp. 321–324. DOI: 10.1126/science.1229566.

[140] Mikhail J. Atallah, Florian Kerschbaum, and Wenliang Du. „Secure and Private Sequence Comparisons“. In: *Proceedings of the 2003 Workshop on Privacy in the Electronic Society*. 2003, pp. 39–44. DOI: 10.1145/1005140.1005147.

[141] Emiliano De Cristofaro, Sky Faber, and Gene Tsudik. „Secure genomic testing with size- and position-hiding private substring matching“. In: *Proceedings of the 12th annual Workshop on Privacy in the Electronic Society*. Nov. 2013, pp. 107–118. DOI: 10.1145/2517840.2517849.

[142] Erman Ayday, Jean Louis Raisaro, Urs Hengartner, Adam Molyneaux, and Jean-Pierre Hubaux. „Privacy-Preserving Processing of Raw Genomic Data“. In: *Proceedings of the 8th International Workshop on Data Privacy Management and Autonomous Spontaneous Security*. 2013, pp. 133–147. DOI: 10.1007/978-3-642-54568-9_9.

[143] Muhammad Naveed, Erman Ayday, Ellen W. Clayton, Jacques Fellay, Carl A. Gunter, Jean-Pierre Hubaux, Bradley A. Malin, and XiaoFeng Wang. „Privacy and Security in the Genomic Era“. In: *CoRR* abs/1405.1891 (2014). URL: http://arxiv.org/abs/1405.1891.

[144] Erman Ayday, Emiliano De Cristofaro, Jean-Pierre Hubaux, and Gene Tsudik. „The Chills and Thrills of Whole Genome Sequencing“. In: *CoRR* abs/1306.1264 (2013). URL: http://arxiv.org/abs/1306.1264.

[145] Bradley A. Malin. „Protecting Genomic Sequence Anonymity with Generalization Lattices“. In: *Methods of Information in Medicine* 44.5 (2005), pp. 687–692. ISSN: 0026-1270.

[146] Traian Marius Truta, Farshad Fotouhi, and Daniel C. Barth-Jones. „Disclosure Risk Measures for Microdata“. In: *Proceedings of the 15th International Conference on Scientific and Statistical Database Management*. 2003, pp. 15–22. DOI: 10.1109/SSDM.2003.1214948.

[147] Fida Kamal Dankar and Khaled El Emam. „A method for evaluating marketer re-identification risk“. In: *Proceedings of the 3rd International Workshop on Privacy and Anonymity in the Information Society*. 2010. DOI: 10.1145/1754239.1754271.

[148] Jerome P Reiter. „Estimating Risks of Identification Disclosure in Microdata“. In: *Journal of the American Statistical Association* 100.472 (2005), pp. 1103–1112. DOI: 10.1198/016214505000000619.

[149] David McClure and Jerome P. Reiter. „Differential Privacy and Statistical Disclosure Risk Measures: An Investigation with Binary Synthetic Data“. In: *Transactions on Data Privacy* 5.3 (2012), pp. 535–552.

[150] Traian Marius Truta, Farshad Fotouhi, and Daniel C. Barth-Jones. „Assessing global disclosure risk in masked microdata“. In: *Proceedings of the 2004 Workshop on Privacy in the Electronic Society*. 2004, pp. 85–93. DOI: 10.1145/1029179.1029202.

[151] Guy Lebanon, Monica Scannapieco, Mohamed R. Fouad, and Elisa Bertino. „Beyond k-Anonymity: A Decision Theoretic Framework for Assessing Privacy Risk“. In: *Transactions on Data Privacy* 2.3 (2009), pp. 153–183.

[152] Fida Kamal Dankar, Khaled El Emam, Angelica Neisa, and Tyson Roffey. „Estimating the re-identification risk of clinical data sets“. In: *BMC Medical Informatics and Decision Making* 12 (2012), p. 66. DOI: 10.1186/1472-6947-12-66.

[153] Paul Ohm. „Broken Promises of Privacy: Responding to the Surprising Failure of Anonymization“. In: *UCLA Law Review* 57 (2010), pp. 1701–1777. URL: http://ssrn.com/abstract=1450006.

[154] Ann Cavoukian and Daniel Castro. *Big Data and Innovation, Setting the Record Straight: De-identification Does Work*. Tech. rep. Accessed: 2014-10-27. Information and Privacy Commissioner Ontario, Canada, 2014. URL: http://www.privacybydesign.ca/index.php/paper/big-data-innovation-setting-record-straight-de-identification-work/.

[155] Arvind Narayanan and Edward W. Felten. *No silver bullet: De-identification still doesn't work*. Tech. rep. Accessed: 2014-10-27. Princeton, 2014. URL: http://randomwalker.info/publications/no-silver-bullet-de-identification.pdf.

[156] Khaled El Emam and Luk Arbuckle. *Why de-identification is a key solution for sharing data responsibly*. Accessed: 2014-10-27. 2014. URL: http://www.futureofprivacy.org/2014/07/24/de-identification-a-critical-debate/.