Informatics

Data Leak Detection

Towards privacy aware asset protection in data driven research

DISSERTATION

zur Erlangung des akademischen Grades

Doktor der Technischen Wissenschaften

eingereicht von

DI. Peter Kieseberg
Matrikelnummer 0125969

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Univ. Prof. DI. Mag. Dr.techn. Edgar Weippl
Zweitbetreuung: Prof. Ing. MMag. Dr.phil. Andreas Holzinger, MBCS, MAE

Diese Dissertation haben begutachtet:

Jungwoo Ryoo	Günther Pernul

Wien, 5. August 2024

Peter Kieseberg

Informatics

Data Leak Detection

Towards privacy aware asset protection in data driven research

DISSERTATION

submitted in partial fulfillment of the requirements for the degree of

Doktor der Technischen Wissenschaften

by

DI. Peter Kieseberg
Registration Number 0125969

to the Faculty of Informatics

at the TU Wien

Advisor: Univ. Prof. DI. Mag. Dr.techn. Edgar Weippl
Second advisor: Prof. Ing. MMag. Dr.phil. Andreas Holzinger, MBCS, MAE

The dissertation has been reviewed by:

Jungwoo Ryoo

Günther Pernul

Vienna, August 5, 2024

Peter Kieseberg

Erklärung zur Verfassung der Arbeit

DI. Peter Kieseberg

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.
Ich erkläre weiters, dass ich mich generativer KI-Tools lediglich als Hilfsmittel bedient habe und in der vorliegenden Arbeit mein gestalterischer Einfluss überwiegt. Im Anhang „Übersicht verwendeter Hilfsmittel“ habe ich alle generativen KI-Tools gelistet, die verwendet wurden, und angegeben, wo und wie sie verwendet wurden. Für Textpassagen, die ohne substantielle Änderungen übernommen wurden, habe ich jeweils die von mir formulierten Eingaben (Prompts) und die verwendete IT-Anwendung mit ihrem Produktnamen und Versionsnummer/Datum angegeben.

Wien, 5. August 2024

Peter Kieseberg

Danksagung

Diese Arbeit ist die Zusammenfassung verschiedener Forschungsarbeiten im Bereich der Erkennung von Datenlecks in den letzten zehn Jahren, die oft aus unterschiedlichen Bereichen stammen, vom medizinischen Sektor bis hin zu Open Source Intelligence (OSINT), wobei der Schwerpunkt auf dem erstgenannten Bereich liegt. So sind die meisten Ergebnisse bereits wissenschaftlich publiziert worden und wurden daher wörtlich aus den entsprechenden Publikationen übernommen, um die Qualitätskontrolle, die der wissenschaftliche Publikationsprozess bietet (peer review), nicht zu untergraben
Diese Arbeit wäre ohne die fachliche und persönliche Unterstützung einiger Personen durch Ideen, Geduld oder Hilfe, aber auch durch das Hinterfragen von Annahmen und Ansätzen nicht möglich gewesen. Meinem Doktorvater Prof. Edgar Weippl bin ich sehr dankbar, dass er mich in meiner Forschung stets unterstützt, aber auch kritisches Feedback und alternative Sichtweisen geliefert hat. Das Gleiche gilt für meinen Co-Betreuer Andreas Holzinger, der mich dabei unterstützt hat, verrückte Ideen zu verfolgen, die zu wertvollen Ergebnissen führten. Ich möchte mich auch bei all meinen Kollegen und Freunden bei SBA Research und der Fachhochschule St. Pölten bedanken - insbesondere bei Simon Tjoa, Sebastian Schrittwieser und Peter Frühwirt.
Weiters wäre diese Arbeit ohne die finanzielle Unterstützung des COMET K1-Zentrums SBA (Projektnummer 843274) durch die FFG Österreichische Forschungsgesellschaft, sowie der folgenden Projekte aus dem KIRAS-Programm (alle FFG) nicht möglich gewesen: DIANA (Projektnummer 832356), DIANGO (Projektnummer 840824), Darknet Analysis (Projektnummer 858591). Dies gilt auch für das COIN-Projekt Big Data Analytics (Projektnummer 866880). Darüber hinaus bot mir die Zusammenarbeit im Rahmen des COMET-Projekts DEXHELPP reichlich Gelegenheit, mich mit Medizinern auszutauschen, um von ihren realen Problemen zu lernen und meine Forschung entsprechend anzupassen.
Vor allem aber bin ich meinen Eltern Marianne und Erik Kieseberg dankbar, die mich in all den Jahren unterstützt und mir das Studium ermöglicht haben, ebenso wie meinem Bruder Klaus. Besonders danken möchte ich meiner Frau Christine für ihre unendliche Liebe, Geduld und Unterstützung während der Erstellung meiner Diplomarbeit sowie meinen Söhnen Viktor und Leopold, die mir die Kraft gaben, die Arbeit zu vollenden.

Acknowledgements

This thesis is the cumulation of different research endeavours in the area of data leak detection throughout the last ten years, often derived from different backgrounds, ranging from the medical sector to Open Source Intelligence (OSINT), with a strong focus on the former.Thus, most of the results have already been published academically and were therefore taken verbatim from the corresponding peer-reviewed publications in order not to undermine the quality control provided by the academic publication process .

This thesis would not have been possible without the support, professionally, as well as personally, of some individuals through the contribution of new ideas, patience or assistance, but also through challenging assumptions and approaches. I am very grateful to my supervisor Prof. Edgar Weippl, who always supported me in my research, as well as provided critical feedback and alternative views. The same holds true for my co-supervisor Andreas Holzinger, who supported me in pursuing crazy ideas that led to valuable results. I would also like to thank all my colleagues and friends at SBA Research and the University of Applied Sciences St. Pölten - in particular Simon Tjoa, Sebastian Schrittwieser and Peter Frühwirt.

Furthermore, this work would not have been possible without the financial support of the COMET K1 center SBA (grant 843274) by the FFG Austrian Research Agency, as well as the following projects from the KIRAS programme (all FFG): DIANA (grant: 832356), DIANGO (grant: 840824), Darknet Analysis (grant: 858591) and the COIN project Big Data Analytics (grant: 866880). In addition, the cooperation within the COMET-project DEXHELPP gave me ample opportunities to interface with medical scientists to learn from their real-life problems and to adjust my research accordingly.

Above all, I am grateful for support from my parents Marianne and Erik Kieseberg, who supported me throughout the years and made it possible for me to study, as well as my brother Klaus. I would like to especially thank my wife Christine for her endless love, patience and support during the creation of my thesis, as well as my sons Viktor and Leopold for giving me the strength to finish.

Kurzfassung

Da datengetriebene Technologien sowohl im akademischen Umfeld, als auch in der Geschäftswelt immer beliebter werden, sind Daten zu einer immer wichtigeren und schützenswerten Ressource geworden. Darüber hinaus erfordert die Explosion der verfügbaren Techniken in Data Mining und Machine Learning, dass viele DatenbesitzerInnen, sowie ExpertInnen in spezifischen Anwendungsbereichen, mit anderen ExpertInnen zusammenarbeiten, entweder aus dem Bereich der künstlichen Intelligenz, aber auch aus anderen Anwendungsbereichen, um einen Mehrwert für ihren Business Case oder Forschungsansatz zu schaffen. Mit dieser Notwendigkeit der Zusammenarbeit stoßen traditionelle Methoden zum Schutz von Daten, wie Verschlüsselung oder sichere Speicherung an entfernten und nicht mit dem Internet angebundenen Orten, an ihre Grenzen, da Daten aktiv an andere PartnerInnen verteilt werden müssen, um ihren Wert optimal zu nutzen. Daher ist es wichtig, eine gewisse Kontrolle gegen die unberechtigte Weitergabe der mitgeteilten Informationen an Dritte behalten zu können, die nicht dazu bestimmt waren, in Besitz dieser Informationen zu sein - die Erkennung sogenannter *data leaks* (Datenlecks). Darüber hinaus sind datengetriebene Systeme anfällig für Angriffe basierend auf Datenmanipulation, die entweder dazu genutzt werden können, die Ergebnisse zu verfälschen oder früher begangene Fehler zu vertuschen. In dieser Doktorarbeit werden wir mehrere Methoden zur Bewältigung dieser Probleme vorschlagen, die auf verschiedene Rahmenparameter und Kollaborationsszenarien abstellen: In einfachen Fällen werden Daten von einem/r EigentümerIn an eine begrenzte Anzahl von EmpfängernInnen gesendet, sodass sog. *data fingerprinting* die Methode der Wahl ist, in komplexeren Fällen können alle teilnehmenden PartnerInnen Informationen (oder Informationspartikel) in eine zentralisierte Datenbank eintragen, in der intransparente ML-Algorithmen an den Daten arbeiten und die Ergebnisse wieder in den Datenspeicher oder nachfolgende Workflows einbringen. Hier erfordert die Kontrolle über die Datenexfiltration andere Techniken, die auf komplexe Audit- und Control-Mechanismen zurückgehen. Darüber hinaus sind die Systeme, die diese Daten speichern, Ziele für fortgeschrittenere Angriffe, nicht nur auf der reinen BenutzerInnenebene, sondern auch auf der Ebene der DatenbankadministratorInnen, entweder durch Social Engineering, Korruption oder durch den Einsatz von ausgeklügelten sog. *targeted attacks* (gezielten Angriffen) mit Malware, die speziell für den Angriff auf ein spezifische System entwickelt wurde. Die Mechanismen, die in dieser Doktorarbeit vorgeschlagen werden zielen darauf ab, den Schutz vor möglichst mächtigen AngreiferInnen zu gewährleisten, oft unter Einbeziehung des/r Datenbank-Administrators oder Administratorin

als (primäre(n)) bösartige(n) BenutzerIn. Der erste Teil dieser Doktorarbeit konzentriert sich auf die Frage des Fingerprintings sensibler Informationen, die auch die Anwendung von Anonymisierungstechniken erfordern, durch die Kombination dieser beiden Aufgaben in einem Ansatz: k-Anonymity-basierte Fingerprints. Darüber hinaus ist dieser Ansatz in der Lage, ein Datenleck auf der Grundlage eines einzelnen gefundenen Datensatzes zu erkennen und damit einen erheblichen Vorteil gegenüber verwandten Ansätzen zu besitzen. Zusätzliche Erweiterungen des Ansatzes inkludieren Widerstandsfähigkeit gegen kolludierende AngreiferInnen, sowie die Einbeziehung komplexerer Definitionen von Anonymität, sowie Methoden zu deren Erreichung. Der zweite Teil der Doktorarbeit ist auf die Erkennung von Datenbankmanipulationen in Audittabellen ausgerichtet. Hierzu nutzen wir die intrinsische Struktur der Elemente einer Tabelle entlang des Index, die in Form eines $\mathcal{B}^+$-Baumes, einer speziellen Form des Suchbaums, strukturiert sind. Wir können nachweisen, dass bestimmte Einfügungsstrategien zu sehr spezifischen Bäumen führen, die auch vom Datenbankadministrator / von der Datenbankadministratorin nicht manipuliert werden können. Darüber hinaus stellen wir zwei Methoden zur Erkennung von Datenlecks für ein spezielles Sharing-Szenario vor: Die Verbreitung in Dateiform, was nützlich ist, wenn es darum geht, vollständige Kopien einer Datenbank zu teilen, Backups zu erstellen oder verschlüsselte Datenbanken zu verteilen. Schließlich können unsere Ergebnisse zur Struktur von $\mathcal{B}^+$-Bäumen auch bestimmte Aspekte der nachweisbaren Löschung in Datenbanken garantieren. Der dritte Teil dieser Doktorarbeit bietet eine manipulationssichere Architektur für sog. "expert in the loop"-Systeme, die typischerweise im Bereich des interaktiven maschinellen Lernens anzutreffen sind. Zusätzlich zur Gewährleistung der Manipulationssicherheit bietet dieser Abschnitt zwei verschiedene Ansätze zur Erkennung von Datenexfiltration: Da Fingerprints in solchen Szenarien schwierig einzuführen sind, da die betreffenden Systeme in der Regel nicht auf der Ebene ganzer Datensätze, sondern eher auf der Ebene des interaktiven Datenzugriffs funktionieren, schlagen wir Audit- und Control-Mechanismen vor, die in der Lage sind, alle Lesezugriffe über die SQL-Schnittstelle abzufangen und gleichzeitig widerstandsfähig gegenüber einem/r AngreiferIn mit Datenbankadministratorrechten sind.

Abstract

With data driven technologies becoming increasingly popular, both, in the academic, as well as the business world, the value of data has become an increasingly important resource that requires protection. In addition, the explosion in available data mining and machine learning techniques requires many data owners, as well as experts in application domains, to cooperate with other experts, either from the artificial intelligence world or from other domains, in order to add value to their business cases or research approaches. With this need to collaborate, the traditional methods for protecting data, like encryption or secure storage in remote and disconnected locations, reach their limits, as data has to be actively distributed to other partners to make the most of its value. Thus, it is important to be able to retain some form of control against unsolicited propagation of the shared information to third parties that were not meant to be in the possession of said information - the detection of so-called *data leaks*. Furthermore, data driven systems are prone to attacks through data manipulation, which could be used to either obstruct the results, or cover up previous errors.
In this thesis we will provide several methods for tackling this problem, reflecting towards different side parameters and collaboration scenarios: In simple cases, data is sent from one owner to a limited number of recipients, thus making fingerprinting the technique of choice, in more complex cases, all participating partners might introduce information (or information particles) to a centralized database, where nontransparent machine learning algorithms will work on the data and re-introduce the results to the data store or subsequent workflows. Here, control over data exfiltration requires different techniques harkening back to complex Audit & Control techniques.
In addition, valuable information makes the systems storing this data targets for increasingly advanced attacks, not only stopping at the pure user level, but possibly extending to the level of database administrators, either through social engineering, corruption, or through the use of elaborate targeted attacks utilizing specially designed malware. Thus, the mechanisms proposed in this thesis aim at providing protection against attackers as potent as possible, often incorporating the database administrator as (primary) malicious user.
The first part of this thesis focuses on the issue of fingerprinting sensitive information that also requires the application of anonymization techniques by concentrating these two tasks into one: k-anonymity based fingerprinting. Furthermore, this approach is capable of detecting a data leak on the basis of a single leaked record, thus greatly extending its

utility over related approaches. Additional extensions of the approach provide resilience against colluding attackers, as well as the incorporation of more advanced definitions of anonymity and methods for achieving it.

The second part of the thesis is focused towards the detection of database manipulation in audit tables. Here we use the intrinsic structure of the elements of a table along the index, which is structured in the form of a $\mathcal{B}^+$-Tree, a special form of search tree. We can proof that certain insertion strategies will result in very specific trees, which are secure against manipulation even by the database administrator. Furthermore, we provide methods for data leak detection for different sharing scenarios: The dissemination in file form, which is useful in case of sharing complete copies, providing backups, or in case of encrypted databases. Finally, our results regarding the structure of $\mathcal{B}^+$-Trees can also guarantee certain aspects of provable deletion in databases.

The third part of this thesis provides a manipulation secure architecture for so-called *expert in the loop* systems, typically found in the realm of interactive machine learning. In addition, to providing manipulation security, this section provides two different approaches for the detection of data exfiltration: As fingerprinting might be difficult to introduce, as such systems typically do not work on the level of whole data sets, but more on interactive data access, we propose audit & control mechanisms that are capable to catch all data exfiltration through the SQL interface, while being resilient against an attacker holding database administrator privileges.

Contents

Kurzfassung **xi**

Abstract **xiii**

Contents **xv**

1 Introduction **1**
1.1 Motivation 1
1.2 Problem Statement 1
1.3 Main Results 2
1.4 Structure of the Work 6

2 State of the Art **7**
2.1 Protecting Anonymity in Data-Driven Biomedical Science 8
2.2 Data Leak Detection 12

3 k-anonymity based fingerprinting **17**
3.1 Motivation & Prerequisites 17
3.2 Background 19
3.3 Basic k-anonymity based fingerprinting 25
3.4 Providing Resilience against Collusion Attacks 33
3.5 Generalization to advanced methods for achieving k-anonymity 47
3.6 Generalization to advanced anonymity definitions 60

4 Detecting manipulations in databases using $\mathcal{B}^+$-Trees **69**
4.1 Introduction 69
4.2 Background 70
4.3 Database forensics using the index tree 75
4.4 Limitations 80
4.5 Using the structure of index trees for logging 85
4.6 Fingerprinting databases using index trees 95

5 Building trust for the expert in the loop **105**
5.1 Introduction and Motivation 105

5.2 Background 106
5.3 The Approach 110
5.4 Evaluation 120

6 Conclusion **125**
6.1 Summary 125
6.2 Future Work and Open Issues 126

Academic & non-academic publication **129**
Published papers 129
Other dissemination activities 131

Overview of Generative AI Tools Used **133**

List of Figures **135**

List of Tables **137**

Bibliography **139**

CHAPTER 1

Introduction

1.1 Motivation

During the past decade, and especially within the last few years, data as become a vital resource for many applications, both, in the business, as well as in the academic world. Contrary to other, especially natural, resources, data, like know-how, is easily transferable. Contrary to pure know-how, where typically special expertise is required for actually using the transferred assets, the barriers for making use of transferred/copied data is rather low. This fact can be seen as one of the ultimate drivers for data driven sciences and business, as data which was collected can be re-used in various other forms and approaches (not counting legal barriers like imposed by the General Data Protection Regulation and other regulations). In addition, due to the simple shareability, collaboration with experts and other labs becomes relatively easy, thus enabling the further utilization of data sets. Still, data, like any other asset, possesses an intrinsic value, like e.g. illustrated by various approaches regarding data obfuscation [NC09], or the example of Skype [YH08], where protecting the network of clients is vital for the business case itself.

1.2 Problem Statement

The main problem when dealing with (sensitive or insensitive) data in real life environments is that data needs to be shared with other people, be it experts on certain technical aspects, legal personal, collaborating researchers or simply for business reasons. Thus, while data protection is of course of vital importance, solely providing sealed environments does not fulfill the requirements of modern data driven approaches in science and business. Therefore, data needs to be protected even after it was shared, as well as controlled within shared environments. Thus, the central research question of this thesis lies in finding methods to protect data after it was shared, i.e. we want to be able to detect so-called *data leaks* [RKS11], parties that legitimately received information, but then decided (willingly or unwillingly, e.g. due to being themselves infected by malware) to further distribute the data without consent of the original data owner. To be more specific, we look at different sorts of environments, mainly

- direct distribution of the data sets from the *data owner*, the entity who is in full control and rightful possession of the information, to the *data recipients*, parties that were selected by the data owner to make use of the data for clearly specified purposes (e.g. [GEK09, IDC15]), or
- shared collaboration environments with centralized data stores, like e.g. envisioned in expert in the loop systems (see e.g. [WKR95, BGD05, SWPB18]), as well as centralized data repositories [TGH+08, TCN+14].

While mainly dealing with the task of detecting information theft, data manipulation must also be taken into account, especially when considering shared collaborative environments. Here it is important not to reduce the attacker models to the standard models of malicious data owners or data recipients, but to also take more potent attackers into account, the file system and especially the database administrator, as especially the later role possesses a lot of privileges and can circumvent most security mechanisms in databases.

1.3 Main Results

In this section, we will outline the main results presented in this thesis. Most of the major results have already been published to the scientific community in the form of papers (see Appendix 6.2), while some smaller results will be published in the aftermath. Thus, these results were therefore taken verbatim from the corresponding peer-reviewed publications in order not to undermine the quality control provided by the academic publication process .

Methodological Approach: The aim of our research, as outlined in the introduction, was to protect different forms of data sharing in data driven environments, often resulting in (slightly) different requirements. Furthermore, there already existed an ample selection of methods for some of these tasks. Thus, in order to shape our endeavours into a form that yields practical and new approaches for open problems, we took the following steps in our methodological approach:

→ **Study on protecting anonymity in data-driven biomedical science:** In order to gather an overview on the pressing questions regarding data protection in real life research environments (apart from Privacy and GDPR related issues), we collected a selection of research questions which formed the background of our motivation [KHS+14]. While we did not follow this through right from the start (as our literature list indicates, several key approaches pre-date this study), it was important to challenge our research approach, as well as the requirements and side parameters that it was built upon. In addition to this literature study, we actively participated in several projects incorporating medical data and the notion of collaborative medical research, thus enabling us to engage with real-world tasks in realistic environments and witness the issues and problems with many model algorithms put forth in theory.

→ ***k*-anonymity based fingerprinting:** Anonymization of sensitive data was determined to be a strict requirement in many cases of collaboration, thus already incorporating a certain amount of distortion to the data and the results of data mining algorithms [MKH17]. In addition, often it is possible to use several different anonymization strategies for generating differently anonymized data sets of about the same value. Thus, we focussed our research on using these different anonymization strategies for fingerprinting [SKE+11b, SKE+11a]. We then extended our basic approach to settings with colluding attackers [KSM+14], as well as more complex notions of anonymity and anonymization.

→ **Utilizing database internals for information protection:** Another important issue derived from our studies was the issue of manipulation detection, especially in the case of either working in a shared environment sporting a centralized database or when sharing whole databases with other institutions. Here we provide an approach based on the intrinsic features of information structuring within modern database management systems (DBMSs) [KSM+11b, KSM+11a, KSM+13], thus even being able to withstand attackers holding administrator privileges on the database. In addition, the results of this research could also be utilized for data leak detection [KSW18].

→ **Designing models for real-life research environments:** In order to better understand the real challenges to data protection in data driven environments, as well as the protection of participants, we generated a model of a medical research platform incorporating all kinds of (benign and malicious) players, ranging from simple users over experts to the introduction of powerful (and intransparent) AI [KSF+15, KMF+16]. The platform was secured with a chaining approach directly inside the database internals (see [FKKW14]). Furthermore, we investigated the possibility to use these internals for data leak protection [KW19] and managed to design an alternative fingerprinting algorithm that works on complete databases [KSW19], sporting completely different requirements and side parameters when compared to the approaches based on anonymization.

1.3.1 Data fingerprinting

Based on various works in the literature available at the time of research, as well as practical experience in medical research projects, we concluded that often data needs to be given away as full set to other experts in order to generate practical cooperation. Thus, data fingerprinting seems a viable solution for protecting data against unsolicited distribution. Still, our main issue with the approaches at the time was that especially medical data is extremely sensitive with respect to privacy, thus requiring protection against theft at the record level, i.e. detection of the data leak should be made possible on a single record basis. This cannot be guaranteed by the typical approaches relying on the insertion (or deletion) of so-called *marker records* into the, often otherwise untainted, original data set. This background in medical research and collaboration also led to the assumption that in case of sensitive information sharing, anonymization techniques will routinely be applied. This assumption was further strengthened as the General Data Protection Regulation was entering into force, with a very strict definition of the term *anonymization*, as well as strong rules when anonymization needs to be applied. The major contribution behind our approach lies in the idea to combine anonymization and fingerprinting by fingerprinting the information through the selection of the underlying anonymization strategies. In addition, we refined this basic approach to cater for stronger attacker models, as well as different approaches for anonymization. More precisely, we provide the following main contributions in this area (see Chapter 3):

1. We present an approach that allows the detection of data leaks based on a single data record, while also guaranteeing a pre-defined anonymization level in order to protect privacy in Section 3.3.
2. In addition, we present a hardened version of the approach that is not susceptible to colluding attacking data recipients in Section 3.4.
3. Since the original approach is only incorporating the de-facto standard method for achieving k-anonymity, *full-domain generalization*, we extended it to work with more complex, but also more effective, generalization methods, as well as suppression and anatomization in Section 3.5.
4. Basic k-anonymity has several shortcomings, thus we studied the extension of our approach to more sophisticated anonymity definitions in Section 3.6.

1.3.2 Detection of Data Manipulation

Data driven technologies, be they part of the academic or the business world, are highly prone to manipulation through data fabrication [BCM+13]. Thus, in order to provide secure and practically useful data driven approaches, the data not only needs protection against simple leakage, but also against attackers targeting the accuracy and integrity of the underlying information. Again, the main fundamental idea lies in using intrinsic features of already utilized technology, this time not any form of anonymization, but the actual structure data is represented within most current database management systems. Since these internals are practically unmodifiably even for the database administrator, we can provide defensive mechanisms against a threat level that is typically not considered in other approaches. Furthermore, our research in this area leads to novel approaches regarding detection of theft (or unwarranted propagation) of whole databases. More precisely, we provide the following main contributions in this area (see Chapter 4):

1. The fundamental approach for detecting data manipulation in databases for revision-secure tables, considering certain side-parameters like an ”INSERT-only” strategy as typical for Audit & Control information, as well as strictly monotonous primary keys (see Section 4.3). This approach is extended to the dual case and an algorithm is provided.

2. A study on the possibilities of extending the algorithm to the detection of DELETE-statements. We can prove that such an extension is not possible, since all except one index tree structure can be generated by solely using insertion, still, this result has great value when considering the problem of provable deletion that is dual to the problem of manipulation detection in many aspects (see Section 4.4).

3. In Section 4.5 we provide a new forensic log that offers the possibility of storing the structural information of database tables for various applications like manipulation detection, forensic copies and so forth.

4. Furthermore, in Section 4.6, we provide two different approaches for data leak detection based on the index tree, one by structuring the information stored in the tables in different trees, the second by using slack space in the tree constituting the primary index of a table.

1.3.3 Securing expert-in-the-loop systems

While protecting the information stored in a data driven environment is of the utmost importance, one aspect of such a system is currently often neglected: The experts need to trust the system, i.e. they need to be assured that the system does not put blame for wrong decisions on their expertize. Thus, while of course in case of artificial intelligence the problem of explainability [Gun17] prevails, at least some level of transparency is required in order to mitigate manipulation of results in the aftermath of a wrong decision. This is especially important in the medical sector, where errors can easily lead to mistreatment with possibly terminal effects (e.g. cancer recognition [HDJ14]). On the other hand, also the expert could try to cover up his/her errors by blaming the intransparency of machine learning after manipulating his/her input information. Thus, in order to make such a system accountable and thus trustworthy for the patient, methods for thwarting data manipulation must be put in place. Furthermore, such systems are highly interactive, as expert information is used to train machine learning algorithms, as well as getting combined with the knowledge of other experts. Thus, information theft and data leakage is an enormous problem, especially since key know-how can be very valuable. Due to the high level of integration in such a system, the methods for detecting unsolicited data retrieval and propagation cannot be based on fingerprinting alone, but need to be included into the very extraction of the information by the expert. More precisely, we provide the following main contributions in this area (see Chapter 5):

1. Section 5.3 gives a detailed overview on our approach for providing a manipulation-secure expert-in-the-loop approach, especially focussing on the biomedical research sector.

2. Further on, in Section 5.3.3, this approach is extended to multiple decision makers, as well as closed source database management systems in Section 5.3.4.

3. The remainder of Section 5.3 is dedicated to providing two different approaches for data leak detection, one extending the original chained witnesses approach [FKKW14] that was also used for providing manipulation detection, the second using slack space inside the database (see Section 5.3.5).

4. Finally, in Section 5.4, we provide a thorough evaluation of the original approach with respect to selected attacker models.

1.4 Structure of the Work

As already outlined in Section 1.3, the thesis is structured along the technological research areas (i) k-anonymity based fingerprinting (Chapter 3), (ii) $\mathcal{B}^+$-Tree forensics (Chapter 4) and (iii) protection through chaining database internals (Chapter 5). While the first of these chapters is more focused towards fingerprinting in order to detect data leaks, the second leans more towards manipulation detection at the start and generates methods for data leak detection out of these results. The same holds true for the third chapter, which at the beginning focusses on providing trust through manipulation detection, but later on introduces novel techniques for data leak detection as well.

Regarding background and related work, we provide some general related work in Chapter 2, but decided to put the background required to understand our approaches, or that our approaches build upon, directly into the respective chapters. Still, some background is required for more than one of the technological chapters (Chapters 3 to 5), thus always being provided in the chapter it is required first.

This thesis is based on work already published in peer-reviewed journals and at academic conferences, thus, most of the results were therefore taken verbatim from the corresponding publications in order not to undermine the quality control provided by the academic publication process. See Appendix 6.2 for the list, also including other dissemination activities related to the content of this thesis.

CHAPTER 2

State of the Art

In this chapter we provide an overview on the state of the art regarding security issues surrounding the topic of data driven research with sensitive and/or valuable data. It must be kept in mind though that the publications this thesis is built on have been developed and published during the last ten years, thus at the time of the publication of our results, the state of the art was completely different from now. To reflect this, the related work is structured as follows:

1. In the first part of this section we will provide an initial discussion on *Protecting Anonymity in Data-Driven Biomedical Science*, where we highlighted several key research issues. This work was originally published as part of a larger state of the art analysis [KHS$^+$14] and gives an overview on important research aspects.

2. The second part of this section is dedicated to discussing alternative approaches to the topic of data leak detection. Here it must be specially kept in mind that this represents the state of the art as of finishing this thesis in early 2019, not the state of the art that was current when several of the approaches discussed in this thesis have been devised and published.

3. Furthermore, background knowledge required for achieving the results is moved to the respective technical chapters, e.g. we have moved the background regarding the definition and construction of $\mathcal{B}^+$-Trees to Chapter 4, which is dedicated to providing new techniques for data leak and manipulation detection based on the index trees of a database table.

2.1 Protecting Anonymity in Data-Driven Biomedical Science

New and advanced methods in statistical analysis and rapidly emerging technological improvements, e.g., in computation performance, data storage, cloud computing and technologies that support worldwide collaborative work, have laid the foundation for a new field of science that we call *data-driven science*. In particular biomedical informatics is becoming such a data-driven science due to the increasing trend towards personalized and precision medicine [CD13, Hol14, HDJ14]. Data-driven science uses these new resources to analyze enormous data sets, often using the term *big data* [Jac09], and reasons based on the empirical findings and evidence from these analyses. The sources of big data can vary extremely, ranging from data gathered online from open sources to data sets provided by research partners, companies or volunteers, or coming from the own laboratories or hospitals; in the medical domain data can come from clinical patient treatment and/or biomedical research, from hospital sources or bio-banks. The range in size and complexity of the data sets allows for a large variety of inferences to be made, which makes big data very useful for research but can, at the same time, potentially be exploited to extract information that could be used in malicious ways or that might violate the privacy of the data subjects [LC11]. This especially concerns data-driven science in the medical sector, since, as a principle, most data in this field is sensitive and issues of privacy, security, safety and data protection are always an issue [WHT06]. Even when access to the data is limited to a specific researcher or a research team, whose members might be from different organizations or universities, there is a high risk of disclosure. The more people have access to classified information, the higher the risk of it being exploited for malicious purposes. However, research, particularly non-commercial research, is usually intended - or should be intended - for public dissemination through conferences or journals [TH12]. The peer-review procedure normally ensures the quality of such research [Spi02], but without access to the underlying data, work in the field of data-driven medical science cannot be validated by reviewers [CMG+13]. The result is an extremely problematic situation where authors either include the data only in a condensed or abstracted form, which protects privacy but has the drawback that the reader cannot validate the results or evaluate the data for a personal learning effect, or publish the data, even if only for the duration of the review process and with restricted access. The former solution is problematic in that the research cannot be properly reviewed, which results in chances for fraud and poor research, especially in the "publish or perish" atmosphere of pressure to publish frequently to gain the recognition of the academic field or funding institutions. Furthermore, even in the absence of fraud, missing data can make perfectly valid results look debatable. The latter solution, while mitigating these problems, exposes data sets to potential misuse. Furthermore, especially regarding data-driven research in medical sciences, the publication of raw data will most certainly result in legal issues. Recently, there is a strong movement towards the promotion of open data sets in biomedical research [BHF09], but the question what to do in case the data cannot be made openly available is still largely unsolved. This problem is not a problem of the health sector alone, albeit the general public is far more sensitive concerning medical information than most other personal information with respect to data breaches. Still, the problems outlined above also concern data driven science in other areas of research, especially since data science is a major driver for research in many fields (see [Bon12, ZB11, CYA12, SCNP10, DJR12, WL12] for examples regarding the IT sector, with a focus on IT security, as well as general trend analysis). For example, Dey et al. [DJR12] analyzed approximately 1,400,000 Facebook account settings to infer privacy trends for several personal attributes. Although they used public accounts for their research, their results, combined with the data they measured and recorded, are highly sensitive and should not be published without applying appropriate anonymization or pseudonymization techniques [PK01]. We, as researchers, are responsible for protecting the data we use and for preserving the privacy of our subjects, who are often volunteers. This protection includes ensuring the unlinkability of sensitive data so that data sets can be published to allow the validation of research, collaboration between several research groups, and learning by enabling the reader to repeat the described data analysis.

Following, we have collected a selection of research questions we see as vital for enhancing the quality of data driven research, while maintaining protection of sensitive information. We structured these questions along their major assets, still, we do not consider this list to be comprehensive and especially the importance of solutions might change drastically in the future due to new regulations.

2.1.1 Questions Regarding Quasi Identifiers

The first group of concerns lies in the selection and treatment of the quasi identifiers (see Section 3.2 for a definition). While this is rather straight forward in the standard examples (e.g. sex, birthdate), there are some inherent questions that require discussion:

Definition of quasi identifiers: While sounding trivial and indeed very easy to decide in the standard examples, this is not so trivial when considering more complex information like medical data. A diagnose could, for example, be so rare that this field together with the ZIP-code results in de-anonymization of the respective person. The diagnose would need to be treated as a quasi identifier in this example. Further examples include rare blood types, parts of genome sequences or browser fingerprints [Eck10]. The determination of QIs is an important research area for guaranteeing the privacy of patients in data-driven research.

Generalization of non-trivial QIs: Following the example in the previous paragraph, the field ”diagnose”, too, is a quasi identifier that a generalization strategy is needed for. While generalization is rather trivial for standard attributes like dates or numbers, it is rather difficult for free text fields. In the case of a diagnose, ontologies could help generating a generalization tree, still, when considering free text like it is found in notes from general practitioners, a lot of further research is needed for purely qualitative information.

Independence of QIs: Sometimes, these quasi identifiers may not be as independent as they seem, e.g., considering the QI ”sex”, an entry in the field ”diagnose” containing ”prostate cancer” leads to a 100% chance of this record belonging to a male person, thus rendering the generalization of ”sex” practically useless without the generalization of the field ”diagnose”. The research in this sector also includes the identification of such QIs, preferably without too much knowledge required on the respective field. One fruitful approach could be the utilization of rule mining [ZZ02] in order to derive such dependencies.

2.1.2 Questions Regarding Collaboration

Collaborating with other research institutes again opens up several interesting research questions that need to be tackled in the close future. Collaboration is often interesting when tackling a wide range of problems together, requiring interdisciplinary approaches, or simply external expertise in order to facilitate better results. While collaboration is thus a valuable tool, it faces multiple problems in real-world environments.

Data Precision Metrics: Most metrics for measuring the information loss currently in use are rather trivially depending on the selected generalization strategies. While there exist examples for metrics depending on the actual data distribution, these are rather inefficient. Finding efficient and expressive metrics seems to be a valuable research question to us. This also includes the question of fairness when using different strategies for different data recipients on the same source data.

Data Leak Detection: Even in case of perfect privacy protection, the leaking of research data may result in severe damage to the data owner, e.g. due to premature publication of results and the need for subsequent revision, or simply because of the value of the data set itself. While techniques for watermarking databases (e.g. see [AK02, DG09]) can be utilized, these face a multitude of different problems regarding applicability and stability, many of them e.g. being independent from the data (not the data storage, though) itself, thus making them removable without reducing the overall quality of the data sets. Thus, research on how to combine leak detection with privacy protection could enhance the willingness of data owners to share their data with other researchers.

Manipulation Detection: One major issues is the problem of data manipulation, i.e. the unsolicited change of information. Actually, the nature of the problem is twofold: Manipulation of data can be used to manipulate the results of algorithms, thus targeting the analysis itself, e.g. in order to influence decisions or even destroy machinery. On the other hand, legitimate users of a data driven system must be able to trust that data that they fed into the system is not corrupted afterwards, e.g. in order to cover up fatal decisions or bad engineering. This trust issue is especially problematic in the area of interactive machine learning and expert in the loop systems.

Information Security: Of course, another major issue with data driven business and science lies in the problem of proper data protection. Again, this is also a requirement of the GDPR, where data owners are required to provide proper protection of their data stores with state of the art technology. Still, while this is a major problem sporting a multitude of different facettes, ranging from pure technological issues to psychological research, e.g. in the realm of social engineering, it is not a problem specific to data driven business and research models, thus being omitted in the remainder of this thesis.

2.1.3 General Research Questions

This section contains some other research questions related to the topic of protecting privacy in data-driven science, which did not fit into the above categories.

Structuring unstructured data: While a lot of data used in medical research naturally possesses the form of structured data (e.g. derived from machines), there is also a wide variety of unstructured data found, e.g. notes and receipts from general practitioners, as well as simply older data. While considerable efforts have been spent on the topic of structuring this semi- and unstructured data vaults during the last years (e.g. by Heurix in [HRFN12] and [HRFN13]), a comparison of this research with a subsequent identification of the major weaknesses and research gaps is needed. Following this basic analysis, research into constructing a working mechanism needs to be conducted.

Practical Considerations: In order to popularize anonymization techniques, efficient implementations are needed, preferably open source in order to enable the researchers to optimize the algorithms with respect to their data. An example for a framework can be found in [KPE+12a], with an outline for an optimization for biomedical data sets in [KPE+12b]. This also includes research on the optimization of the algorithms, which e.g. has been conducted by El Emam et. al. for basic k-anonymity [EEDI+09]. Furthermore, for review processes, an interface allowing the anonymous exchange of data sets in the course of the peer-review process would be a valuable addition.

2.1.4 Influencing other research areas

In other research areas, the problem of privacy in data-driven science is rather new which results in a striking absence of specific laws or regulations. In classical IT-security we are under the impression that currently research data is usually held back instead of released in an anonymized state. In our opinion this is largely due to a lack of rules for anonymous data publishing which pushes responsibility for privacy protection onto individual researchers, thus resulting in uncertainty. This poses a major problem for the reproducibility of results, which is one of the main pillars of modern science and it's underlying review paradigm. Furthermore, todays anonymization concepts mostly come from medical research and are therefore designed for highly structured data, thus often cannot be used for other types of information. This again opens up a new area for research. In view of the rapidly growing trend towards more data-driven research, these new methods will be needed rather sooner than later. Specific policies and guidelines governing the public availability of data in (data-driven) science would also be helpful in order to guarantee the validity of published research.

2.2 Data Leak Detection

In this section, we discuss related approaches from the literature, especially focussing on the issues of database watermarking and fingerprinting, as well as other forms of data leak detection in shared data environments. We do not concern ourselves with issues of "secret" or hidden data extraction, as it is often applied by malware or industrial espionage. Neither do we consider general technologies of IT-Security like system hardening or attack detection or topics of organizational security, as these would be out of scope of this thesis, even though these topics are often in the focus of academic work on data leak detection (see e.g. [BK17]).

2.2.1 Fingerprinting and Watermarking

In a world driven by data analysis, stowing data away in separate data silos becomes a disadvantage in many application fields. Of course, a whole set of measures to provide data protection exists, especially for safeguarding data against unsolicited access, still, often the real value behind data sets lies in (a) combination with other data sources, either more of the same type of data or data that can be used to enrich the information available, and (b) cooperation with (external) experts, either domain-specific specialists, or data scientists. To this end, data needs to be shared somehow, opening up a plethora of problems, ranging from privacy over data quality to protection of valuable assets (see Section 2.1). Thus, the ex ante protection mechanisms targeting information (access) control and exfiltration detection are not sufficient anymore, as they do not cover the attack surface opened by explicitly intended data propagation. Thus, ex post protection mechanisms need to be put into place. These mechanisms will not stop the unsolicited propagation of information at all, but they will provide a technological lever for pursuing legal actions in case data was found to be shared illegally. Such techniques thus require to enable the information owner to pinpoint, who was leaking the information, preferably in a provable manner in order to take the issue to court. Thus, in this section we will provide an outline on these, so-called *data leak detection* mechanisms, notably we will discuss watermarking and fingerprinting techniques.

In terms of usage in both, the academic world, as well as the application domain, the terms watermarking and fingerprinting are a bit ambiguously defined and often used interchangeably. Typically, the term *watermarking* defines techniques that add visible or hidden information (e.g. a copyright notice) to the target data. An important characteristic of watermarking is that adding this information modifies the data, either visible or invisible to users. This digital watermark, in analogy to the original techniques in the physical world like used in bank notes, is primarily used in order to either prove authenticity, or ownership of the original user, but it is not personalized, i.e. it is not encoded, to whom the watermarked information was distributed to.

In contrast to watermarking, a consistent definition of fingerprinting does not exist among the research community. A common definition describes fingerprinting as a subtype of watermarking where a unique mark (i.e. the fingerprint) is added to each copy of the data. A second definition distinguishes fingerprinting from watermarking by the source of the fingerprint: While in watermarking information is added, fingerprinting uses intrinsic properties to uniquely differentiate the copies. In both definitions, however, the uniqueness of the fingerprint for each copy of the data is the key concept that enables a data owner to uniquely link a (intended) data recipient to a specific copy. It must be noted though that some authors use slightly different interpretations of the word *fingerprinting*, e.g. by differentiating methods that utilize information added to the data set for detection purposes (watermarks) versus the utilization of information intrinsic to the data without additional changes on the data [Mil12]. Interestingly, our approach outlined in Chapter 3 can also be viewed in a similar way, as, while the anonymization is definitively changed by the target of providing fingerprinted data sets, the anonymization itself is done for independent reasons.

In the past, especially in the first decade of this century, research on watermarking and fingerprinting techniques was primarily focused on multimedia data such as images or video files [LSL00, FS03, SJL$^+$05, HK99], which is not within the scope of this thesis. Marking of non-multimedia files such as database tables, is far less explored, but of great concern in our opinion, as especially considering the rise of data driven research and business proposals, data analysis is becoming one of the decisive factors in many industries. This also possesses great impact on the value associated with data. In [SAP04], the authors presented a watermarking technique for relational databases based on watermarking a numeric collection, which is robust against several attacks such as data resorting, subset selection and linear data changes. Another approach [Gro11] discussed the problem of watermarking databases, while preserving a set of parametric queries in a specified language. In [AO08], the authors introduced a watermarking concept for databases based on the insertion of binary image watermarks in non-numeric multi-word attributes of selected tuples. The idea of embedding a watermarked image into a database was put forth in [ZJWL04]. Other authors introduced a block oriented fingerprinting approach for relational databases [LWDS04]. In [Laf07] a security analysis concerning watermarking schemas was performed. The authors analyzed the concepts in terms of uncertainty on the location of watermarked parts of the database, i.e. the difficulty for an attacker to identify the watermark. Other approaches, like [WdW96] built fingerprints from combinations of identifying variables in the data records and used them for identification of specific records. The authors of [BOYD05] introduced a privacy and ownership preserving outsourcing concept for medical data based on binning and watermarking techniques. Another watermarking-based protection scheme for outsourced databases was introduced by [QYJjYs06]. In [AK02], the authors proposed a watermarking technique for relational data which uses a private key that is known only to the data owner. In recent research, the authors incorporated biometric information on the data recipients inside the fingerprints [ASKSA+20] through slight alteration of the original data. None of these schemes focuses on the detection of a data leaking party based on single records, like our approaches outlined in Section 3 do. Furthermore, none of the approaches use database steganography in order to mark full database implementations like our approaches in Section 4 and Section 5 do.

Collusion-resistance, the inability of collaborating data receivers to remove the mark by combining their versions, is an important feature of watermarking and fingerprinting techniques. In recent literature, several collusion-resistant approaches were introduced. In [SKH02], the authors presented a collusion-resistant watermark for video data, furthermore, they introduced an approach against linear frame collusions against watermarks in videos based on the implementation of a statistically invisible video watermark [SKH05]. Collusion-resistant fingerprinting was the target of research during the last decade as well. Trappe et. al. [TWWL03] investigated the problem of collusion-attacks and introduced a tree-structured detection algorithm for identifying colluders of fingerprinted multimedia files. Celik et. al. [CST] proposed a fingerprinting schema for multimedia files that renders collusions ineffective by reducing the result of a collusion to low quality and a related technique considering a combination of digital watermarks and collusion secure fingerprints for digital images in the medical sector was proposed by Dittmann et. al. [DSS+00]. The feature of collusion resistance is very important for our work and was one of our key features when developing the solutions outlined in this thesis. In recent work, the usage of fingerprinting has been extended to other complex types of structured information, like deep neural networks [CJG21], which is especially challenging to attack given issues related to explainability [ADRDS+20].

A recent comparison of techniques for fingerprinting structured data can be found in [ŠM19], where the author also included the approaches porposed in this thesis (see Chapter 3). The major difference between our approaches and the other ones discussed in this work lies in capabilities regarding the identification of data leaks based on a single record, no matter what record (some approaches can detect on a single record basis in case the right record is leaked, our approach is capable of identification based on any given record).

2.2.2 Other approaches for detecting data leaks

Since data protection is a major driver for the entire field of IT Security, many security measures can be used in order to gather intelligence on unsolicited data extraction. Still, in this thesis we focus on the use case that data was willingly shared with another party, e.g. in order to carry out experiments or to gather expertise, and only further, unsolicited, sharing with third parties should be detected and (at best) made counteracted.

This is of course rather different from the scenarios in ”classical data protection”, like e.g. detecting unsolicited exfiltration of information [GBC06] by malware and thus also requires completely different approaches.

In [PS17], the authors discussed the detection of data leaks through a Data Leak Detection / Data Leak Prevention System (DLD/DLPS), where the focus lies on different techniques like access control models like RBAC [SCFY96] and user rights management. This could especially be combined with technologies like *research servers* as facilitated by projects like DEXHELPP [ZPB14], where data is not provided to the recipients as data set to be analyzed at their own premises, but where a complete virtualized setting is set up, with all the tools required for the analysis already pre-installed, and no direct access to the underlying data. While data theft is still possible in most of these scenarios, it is made much harder, especially when factoring in unintentional loss as a major security issue.

Data Leak Detection has also been considered in terms of data exfiltration, i.e. data is sent (unencrypted) over the network, without there being any consent to this transfer [SY12a]. The techniques utilized for detection are quite similar to intrusion detection systems [SY12b], still, the whole issue of data exfiltration is outside the scope of this thesis.

Other related approaches that aim at detecting data leaks during transmission are based on analyzing n-grams [ASM13], again, while being developed into very interesting technologies with a broad range of applications and impressive developments [LHMM18], target data transmissions that should not have happened at all. This is very different from the background for our approaches, where the initial transmission to the data recipient is legit and only subsequent illegitimate data propagation needs to be detected. Related to these approaches and their targets, in their work [ÁKKP21], the authors provide a study on different data leaks regarding GDPR related information, especially focussing on detection through logs.

CHAPTER 3

k-anonymity based fingerprinting

3.1 Motivation & Prerequisites

When dealing with sensitive personal data and using machine learning algorithms to find correlations hinting at causalities, anonymization is a major tool for ensuring the rights of the persons in question with regards to privacy. This has become especially important since the implementation of the general Data Protection Regulation (GDPR) [Reg16], but has already been a standard issue before in selected industries, most notably the medical sector where regulations like HIPAA [fDCP+03] have long been in place. Anonymization is an expensive task when looking at the amount of data typically found in big data applications, as are most (and especially the more advanced) fingerprinting algorithms, thus the major idea of our research in this area lies in combining anonymization with fingerprinting by generating the fingerprint as an intrinsic attribute of the anonymization step, thus adding only little additional cost to the whole process.

Furthermore, we propose the following prerequisites that we require our fingerprinting algorithm to be capable of following:

1. **Stability:** Even when the data is utilized and changed, the fingerprint shall still stay detectible to a certain point. Especially, it must not be possible to easily remove the fingerprint.

2. **Variability:** The number of possible fingerprints needs to be high enough to be able to construct enough different data sets for the purpose in question. This criterion may vary heavily depending on the actual use case with a handful of fingerprints being enough for many scenarios in distributed academic research, but rising up to requiring many million different fingerprints when distributing digital media in a DRM scenario [PAB06].

3. **Little additional overhead:** Eventually added information should not make the data set inconveniently large, e.g. by blowing up the size of the set by adding huge amounts of marker records or by adding large and/or complicated attributes to the set. This, of course, is also depending on the actual size and structure of the original data set.

4. **Little additional effort:** Application of the fingerprint should not require expensive operations. Still, again, this criterion is heavily depending on the actual use case in question, especially considering the value of the information, as well as the number of records and the actual overall performance of other processes involved in the data analysis.

5. **Efficient detection:** The fingerprint should be detectible with a small number of records only. While of course there exist legitimate reasons for reducing this capability to only thwarting the unsolicited distribution of large parts of the data (e.g. in a DRM-case, where detection of single leaked video frames is typically not interesting for the copyright holders), in our research we set the goal to make the data leak detectible by a single leaked record. This is especially important, as even a single record can cause major problems in the area of medical research.

6. **Simple and performant detectability by the owner:** There needs to be a simple way to not only detect the fingerprint, but to also attribute it to a leaking data consumer. The efforts for detection and attribution need to be low enough for the owner to actually undertake this endeavour for any suspicious record. Of course, this can be split by making the detection of ownership especially easy and only undergo the efforts for attribution at the point of confirmation of ownership.

7. **Hard to remove:** For any data consumer it must be hard to remove the fingerprint. This can either be done by making the characteristics constituting the fingerprint hard to detect, or by making the removal require drastic methods that would e.g. render the data useless for most purposes.

8. **Stability against colluding attackers:** Fingerprints, especially when dealing with valuable information, not only need to be resilient against single attackers, but also against a collusion between multiple data recipients, be it with respect to detection, removal or trying to attribute the leakage to another (not colluding) data recipient.

The basic algorithm we will provide in Section 3.3 will be capable of providing the capability of detecting the leaking data recipient based on a single leaked data record. The stability gainst colluding attackers is discussed later on in Section 3.4.

3.2 Background

In this section, we will provide a short overview on the basic concepts of *quasi-identifiers* and k-anonymity, as well as a short overview on the issue of information loss, as these concepts build the fundamentals of our approach. While the major outlined algorithms are based on the "simple" notion of "pure" k-anonymity, and especially focus on the typically employed technique of *full-domain generalization* in order to achieve it, Sections 3.5 and 3.6 will show possibilities for generalization of the approach to other definitions of anonymity, as well as more advanced techniques for achieving k-anonymity based approaches.

3.2.1 Anonymization and k-anonymity

Anonymization has been the subject of lot of research work in the past decade, mainly due to the rising attention to threats and dangers to privacy and informational self-determination provided by new capabilities in machine learning together with the vast collection of private and personal information through major companies like Google and insurance companies (see [RBR+11] for an example using Google data). Depending on the use case, several strategies on dealing with personal and sensitive information have been put forward, often harkening back at the principles of privacy by design [Cav11], and especially to the principle of data minimization fundamental to it [Sch10].

Traditionally, anonymization was mainly concerned with the identification of directly identifying information in databases, like e.g. in [RBR+00], even though the problem of re-identification through statistical methods was already part of the state of the art as outlined in [Klo95] for example. Still, in her canonical paper [Swe02b] Sweeney brought this issue to prominence by showing that it is actually practically feasible to identify 87 percent of all American citizens when given the three attributes "birthdate", "zip code" and "sex". The major problem leading to this identification is the fact that data does not exist on its own in empty space, but that inferences with other, publicly available, data sets can be used for re-identification. This form of attack is called *record linkage* (see Figure 3.1) and despite its simplicity it opens up many legal questions, especially around the topic of what external data has to be expected to exist for such attacks and whether attackers would have access to it [MX07]. Based on these results, she introduced the concept of k-anonymity in [S+02], which is one of the most well-researched protection models for structured data and has spawned numerous variations like (X, Y)-anonymity [WF06] or l-diversity [MKGV07], thus extending the original scope and reacting to different attacks and requirements. Figure 3.1 shows a typical example for such an attack: The identifiers might have been removed from the "anonymized" set, but the attacker is in possession of additional information that allows to identify the person in the data sample and thus enables the attacker to gather information on the diagnosis. This example shows that, as Sweeney has pointed out, simple removal of identifiers is not enough in order to protect sensitive information.

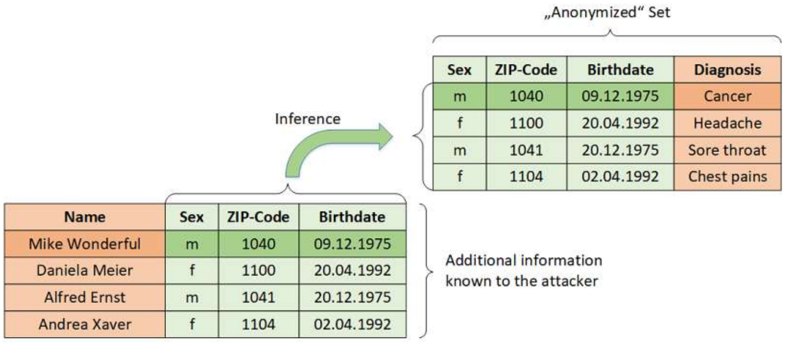

Figure 3.1: Record linkage attack

In general, anonymization techniques can be split into two main categories from a practical point of view: Strategies that take a data set, apply changes to it in order to make it (practically) impossible to re-identify and/or retrieve information on persons and then distribute the anonymized version of the set to data recipients on the one hand, and oracle-style approaches on the other hand, where the full data set is never distributed to anyone else but the original data owner who then answers statistical inquiries based on that data, but adds additional distortion in order to protect against (repeated) statements that could potentially harm the privacy of persons included in the data set. Prime examples for strategies of the first category are k-anonymity and perturbation-based approaches. In the latter, the actual data is changed by adding various forms of perturbation information, like random numbers, often based on models that to some extent guarantee the preservation of key statistical features or key relations between data attributes. The prime example for the second category is differential privacy [Dwo11] in its various variants (see e.g. [BO13]).

A very direct approach for defining anonymized data sets has been put forward by the General Data Protection Regulation (GDPR) of the European Commission [Dir95]. It states explicitly that any function where there exists an inverse de-anonymization function cannot be called *anonymization*, effectively removing pseudonymization (see e.g.[NH11] for an approach for protecting medical data) from the list of technical and organisational methods (TOMs, see e.g. [RLW18]) suitable for privacy protection [RP17].

Taking a step back, when proposing anonymization of information, it is of the utmost importance to determine what data actually poses a threat to the privacy of people and thus requires treatment through anonymization, leading us to the following definitions:

Definition 3.2.1 (Quasi-identifier)**.** *Let $\mathcal{A}$ be a set of attributes of a structured data set, e.g. columns in a table. Then we call the set of all attributes Q' that either identify a person directly or in combination with each other,* quasi-identifiers *of $\mathcal{A}$.*

While the above definition is the original one as given by Dalenius in [Dal86], in current literature (e.g. El Emam [EEDI+09]) $\mathcal{Q}$ is divided into the set of *identifiers* $\mathcal{I}$ that contains all attributes directly identifying a person and the set of *quasi identifiers* $\mathcal{QI}$. Throughout the rest of this thesis, we will stick with the latter notion, i.e. attributes that directly identify persons will be treated as identifiers, thus requiring removal from the data set before distribution (in fact, identifiers can sometimes be made into quasi-identifiers by distorting their contents, which is, for example, done for genomes e.g. by using an approach similar to k-anonymity [Mal05] or by noise introduction [WMC14]), and attributes that can only identify people in conjunction with other fields and external (additional) information will be treated as quasi-identifiers.

Definition 3.2.2 (k-anonymity)**.** *Following the above definitions, a set of data records is said to obey* k-anonymity *for a given* $k \in N$ *if and only if each record is indistinguishable from at least* $k-1$ *other records with respect to* $\mathcal{QI}$ *[S+02].*

It follows trivially that the records are divided into equivalency classes with respect to the values of the quasi identifiers, the k-anonymity principle can thus be seen as a partitioning of all records into equivalency classes with at least k members in every non-empty class. The parameter k is usually referred to as the *level of anonymity* or *privacy factor*, which can, in theory, be raised by increasing k. However, for practical considerations, it is often required to choose a rather low value for k in order not to reduce the overall value of the data too much and to still be able to generate meaningful results when analyzing it [MKWH16]. Still, recent experiments have concluded that the amount of data loss does not always grow proportional to data loss, but also depends on various other factors, especially the actual algorithm in use. Especially considering heuristic algorithms like Mondrian [LDR06], the selection of the data loss metric [GKKM07] has a high impact, among other issues. This can lead to behaviour, where an increase in the k factor yields more accurate results when compared to the execution of the same machine learning algorithms on the original, unanonymized data [SHK+21]. Still, while this effect exists, it can be attributed to fundamental re-distribution of the data records into new equivalency classes and other effects.

Example 3.2.3. *Table 3.1 contains a simple data set that obeys the k-anonymity criterion for $k = 2$. Please note that the attribute "diagnosis" was not classified as a quasi identifier, which, of course, can be discussed, especially with respect to very rare diseases, or simply in case the diagnosis gives away on other quasi identifiers, as e.g. prostate cancer would be directly connected to the sex of the patient. Figure 3.2 shows the effect during the record linkage attack, the re-identification of a single person is reduced to the identification of the respective equivalency class.*

The most common technique for achieving k-anonymity is *generalization*, notably *full-domain generalization*: By reducing the granularity of the quasi identifiers, the records

sex	year of birth	diagnosis
M	1980	Obesity
F	1976	Cancer
F	1976	Obesity
M	1980	Chest Pain

Table 3.1: Anonymized data ($k = 2$)

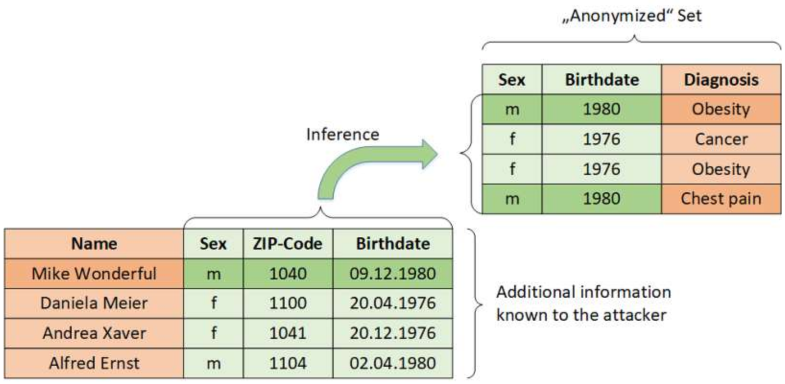

Figure 3.2: Record linkage attack thwarted by k-anonymity

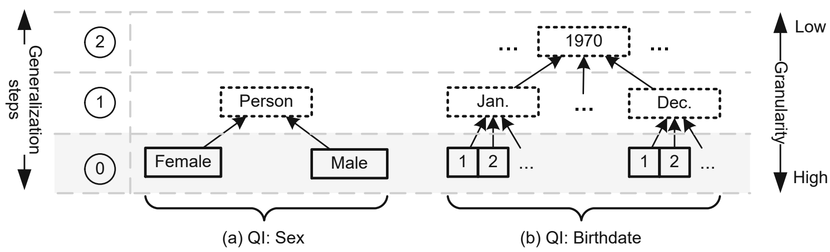

Figure 3.3: Generalization of quasi-identifiers.

get increasingly similar until the k-anonymity criterion is fulfilled. An example for generalizing the quasi identifiers "sex" and "birthdate" is outlined in Figure 3.3.

Definition 3.2.4 (Generalization Level)**.** *Let q be a quasi identifier of a data set. Then we call different levels of granularity that can be constructed from the original data* generalization levels*.*

Figure 3.3 shows an arbitrary solution for defining the generalization levels for the quasi identifiers "birth date" and "sex". While the proposed levels might be logical for the QI "sex" when excluding non-binary definitions (see [Bil05]), it is rather arbitrary for the QI "birth date", as there are many different ways for generalization, e.g. using seasons or quarters, etc.

Definition 3.2.5 (Generalization Pattern)**.** *Let n denote the number of quasi identifiers. A tuple $q := (q_1, \ldots, q_n)$ consisting of exactly one generalization level q_i for each quasi identifier from $\mathcal{QI}, i = 1, \ldots, n$ is called a* generalization pattern*, or simply* pattern*, of the data set. Each of these patterns resembles exactly one way to generalize the records in the data set with respect to the pre-defined generalization levels.*

Given a definition of the generalization levels as e.g. outlined in Figure 3.3, all possible generalization patterns form a lattice diagram, where directly adjacent patterns are constructed by further generalization (or vice versa specialization) of one quasi-identifier by one generalization level. The lattice diagram formed by the generalization levels defined in Figure 3.3 is shown in Figure 3.4. It must be noted that in case a pattern obeys the k-anonymity criterion for a given k, all other patterns above in the lattice obey the criterion for at least the same k and in case it does not, all data sets generalized with patterns below the given one cannot fulfill the criterion either.

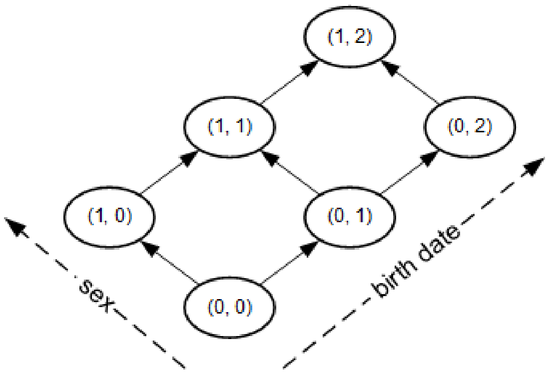

Figure 3.4: Lattice diagram for generalizations.

3.2.2 Measuring Data Loss

Throughout the description of our approach, we will use the notion of *data quality*, as we want to provide a kind of *fairness* amongst data recipients: Since anonymization is typically introducing distortion to the data set, the resulting loss in quality should be roughly equal amongst the recipients. Thus, measures for data quality, or dually for data loss, need to be established. While it was out of scope of this thesis to dwell on this subject in-depth, the following so-called *Data Precision Metrics* (DPMs) [BNSE15] (also called *data loss metrics* in the literature, see e.g. [ZO11]) were used throughout our work. It must be noted though that the problem of giving general definitions for data quality is open and most likely there is no general solution to the problem, as different tasks and data sets will lead to different interpretations of quality, leading to the problem of *explainable AI* [Gun17]. In addition it must be noted that these metrics only work when generating the anonymized data sets using (full domain) generalization.

The *Samarati metric* [Sam01] is very simple. It counts the height of a node in the lattice diagram used to visualize all possible combinations of generalization operations. Although it is intuitive and easy to calculate, this metric takes into account the actual level of generalization only, but not the total number of generalization levels. For example, generalizing {`female, male`} to {`person`} results in much greater information loss than generalizing a date (e.g. birthdate) from day granularity to month granularity. Although both operations are one-level generalizations, the perceived information loss is drastically different. Additionally, this means that an identifier possessing more levels of granularity will have a greater impact on the data precision metric. Furthermore, the metric is directly depending on the pre-defined generalization levels, thus, adding of additional levels into an already existing generalization strategy can obviously have great impact on this metric.

The *Precision metric* [Swe02a] was introduced by Sweeney in 2002. The main difference from the Samarati metric is that the generalizations are not weighted equally. Instead, they are weighted with respect to the number of generalization levels possible for each QI. If a QI has more than one level of generalization (i.e., two levels of granularity), the weighting is given by

$$\frac{\text{Number of levels generalized}}{\text{Number of possible generalization levels}}.$$

For example, if one QI can be granulated into five different levels (the lowest being the exact data), generalization of one level yields a weight of $\frac{1}{4}$, whereas the generalization from {`female, male`} to {`person`} would have a weight of 1. The drawback of the precision metric is that it does not factor in the size of the levels, only the number of generalization levels.

The *Discernability metric* [BA05] takes into account the number of elements distributed to each equivalency class; i.e., the equality of the distribution of the elements to the respective classes is taken into consideration. This stands in direct contrast to the two metrics above, which consider only the classification itself, apart from the actual data and can thus be calculated independently beforehand and for every data set using the same QIs and generalization strategies.

Let N be the number of classes, n_i the number of elements of the i-th class and $n = \sum_N n_i$ the number of all elements. Then

$$DM = \sum_{n_i \geq k} n_i^2 + \sum_{n_i < k} (n \cdot n_i).$$

Despite its name, the Discernability metric is not an actual metric since the criterion of monotonicity is not fulfilled [EEDI+09].

The *Modified Discernability metric* (DM*), a slightly modified version of the Discernability metric, was proposed by El Emam et al. [EEDI+09]. Contrary to the original discernability metric it is monotonic and thus a real metric with respect to the mathematical definition.

Again, N denotes the number of classes and n_i the number of elements in the i-th class.

$$DM^* = \sum_N n_i^2$$

Still, while all these metrics are certainly functional for selected applications in data science, other metrics might be more realistically portraying the effects of anonymization on the results of machine learning algorithms. One straight-forward measure can be derived by comparing the results of the algorithms executed on the un-anonymized, and thus undistorted, data set to the anonymized versions and calculating measures like the F1-score [S+07], like it was done in [MKH17]. Still, this requires the anonymizing party to carry out each analysis step before providing the data to the data recipients, which is completely unrealistic in any bigger and more sophisticated scenario and often voids the reason for using anonymized data or sharing data in the first place (see [KKH19] for a experimental setup regarding distortion analysis).

3.3 Basic k-anonymity based fingerprinting

In this section we provide an overview on the fundamental idea of k-anonymity based fingerprinting and a first algorithm. While the basic approach focusses on classical k-anonymity derived by full-domain generalization, it is extended in Section 3.4 in order to provide resilience against colluding attackers. In Sections 3.5 and 3.6 we will furthermore discuss some extensions to the algorithm based on refining either the anonymity definition it is based on, or by utilizing more complex (and more effective) anonymization methods. Most parts of this section are taken verbatim from the original papers.

3.3.1 General Approach

Imagine microdata containing sensitive medical health information that is to be released to three different data analysts for further investigation. To protect privacy and avoid linking persons to sensitive attributes, the records are anonymized before release. As anonymization would likely reduce data precision and thus diminish the usefulness of this data, a trade-off between the anonymity level and data precision must be made. The result will likely be partially anonymized microdata sets that still contain sensitive information. They must therefore be protected against unauthorized disclosure. Fingerprinting, a passive form of security, can be used to help identify the source of an unauthorized disclosure. In contrast to other approaches, where typically hidden data for identification is embedded into the host data, our fingerprinting method uses existing properties of the host data to identify it.

Our approach is to uniquely identify the sets of anonymized microdata, or even subsets thereof, solely based on the generalization patterns of the quasi-identifiers. We hypothesize that different generalization operations can be used to achieve the same levels of anonymity and similar levels of data precision while the resulting microdata sets have unique characteristics that enable their robust identification.

Figure 3.5 illustrates our idea of clustering datasets with similar anonymity and data precision values. The black dots refer to the different generalization patterns that are arranged along the axis of a coordinate system based on their anonymity and utility values. Similar values above the minimum level of anonymity can be clustered and used for the generation of comparable sets in terms of data quality (this is denoted by the two circles). In our example, the three data analysts would receive three different sets of partially anonymized microdata. All sets, however, would have the same or very similar levels of anonymity and data precision. If a subset of one of these microdata sets is disclosed to an unauthorized party, the source can easily be identified on the basis of the unique generalization pattern that was used for anonymization. As discussed before, the quality of a generalization (and thus its affiliation to a quality class) is largely defined by the data precision metric in use.

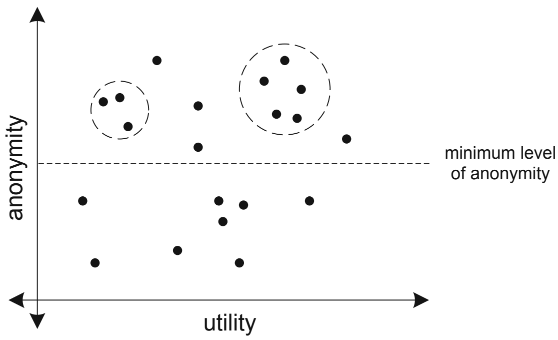

Figure 3.5: Performing different generalization/suppression operations to achieve the same level of anonymity.

As shown in the example datasets in Table 3.2, the same level of anonymity can be achieved with different generalization strategies. While the first set in the tables was won by generalizing the birthdate to year granularity, the second data set was won by suppressing the gender and by generalizing the birthdate to a monthly granularity. Both data sets obey the k-anonymity criterion for $k = 2$. The attribute "diagnosis" as well as any direct identifiers were removed from the example for the sake of brevity.

These unique generalization patters can be used as a fingerprint. Consider a result cluster C_1 containing two anonymized data sets with the generalization patterns (a_1, b_1) and (a_0, b_2) for the QIs gender (a) and birthdate (b). The first one is given to user U_1, and the second one to user U_2. If, for example, data record $(m, 1970)$ is disclosed, it is easy to identify user U_2 as the data source since user U_1 is not in the possession of gender data with this granularity. Figure 3.6 illustrates the approach of determining the generalization levels of the QIs and comparison to stored generalization patterns of released data sets.

Original data set			Anonymized set 1		Anonymized set 2	
Gender	Birthdate	Diagnosis	Gender	Birthdate	Gender	Birthdate
m	19.03.1970	diabetic	F	1970	P	03.1970
m	20.03.1970	disability	M	1970	P	03.1970
f	18.04.1970	blind	F	1970	P	04.1970
f	21.04.1970	disability	M	1970	P	04.1970

Table 3.2: Original microdata and two anonymized versions with $k = 2$

3.3.2 Algorithms for generation and detection

In this section we propose an algorithm for the generation of unique fingerprints for microdata sets, as well as one for the identification of the source of a data leak.

Algorithm for Generating Fingerprints

Our approach is based on El Emam's algorithm for calculating the optimal solution for k-anonymization in case of full-domain generalization and even under the utilization of (small amounts) of suppression [EEDI+09]. Since we need all k-anonymous solutions instead of just the optimal one (i.e. the one with minimum information loss), we cannot rule out the less granular solutions of a known solution, but still have to calculate the data precision for each of these nodes.

1. Define a minimum k for the k-anonymity criterion, the minimum and maximum levels of data loss l_{min} and l_{max} and the data precision metric to be used.
2. Define the generalization strategies for each identifier.
3. Calculate the lattice diagram derived from all possible generalizations.
4. Choose a node at middle height and decide whether it is at least k-anonymous.
   a) In case it is not, rule out all nodes below in the lattice diagram.
   b) In case it is, mark all nodes above the chosen one as possible solutions.
5. Start with step four for the remaining subgraph, similar to the original algorithm.
6. In case no subgraph is left, start by choosing another initial node at middle height and proceed with step four until all nodes are evaluated.
7. For each at least k-anonymous solution, calculate data precision and the actual k. Remove all solutions with data precision outside the bounds of l_{min} and l_{max}.
8. Classify and cluster the solutions by their data precision.
9. Create "similar" microdata sets based on results in one cluster and distribute them to the recipients.

Node	Sam	Prec	DM*	k
(a_0, b_0)	0	0	4	1
(a_1, b_0)	1	1	4	1
(a_0, b_1)	1	0.5	8	2
(a_0, b_2)	2	1	8	2
(a_1, b_1)	2	1.5	8	2
(a_1, b_2)	3	2	16	4

Table 3.3: Impact of different metrics

Following we give further details on some of the steps:

Data precision metrics and maximum levels (Step 1) All recipients shall be provided with (roughly) the same level of data precision. Thus, the method of measuring data precision can have a great impact on the definition of the classes of equivalent anonymization strategies (with respect to data quality). Table 3.3 gives an overview on the perceived data precision of the data from Table 3.2 based on all identified generalization techniques (see Figure 3.3), with respect to the three metrics discussed in Section 3.2.2. Additionally, it must be decided beforehand, how big the maximum tolerable data loss is.

Eliminating nodes (Step 4) One major idea behind the original algorithm by El Emam [EEDI+09] lies in the construction of the lattice diagram of all possible generalization patterns (as outlined in Figure 3.4), together with the observation that in case the generalization strategy provided by a node in the lattice does not fulfill the k-anonymity criterion for a predefined k, none of the (more granular) patterns in the lattice below can fulfill it and therefore need not be tested. On the other hand, in case the pattern does fulfill the criterion, so does every pattern above, but with less quality due to lower granularity, thus also not requiring any further calculations. Contrary to this algorithm for finding the optimal solution, the nodes above the chosen one cannot be ruled out in case of 4.b (i.e. the node represents a k-anonymous generalization), since we need all solutions, not only the optimal one. Still, there is some room for performance optimization, in case minimal quality (i.e. maximal data loss) thresholds have been devised.

Clustering the solutions (Step 8) In step eight the solutions are clustered by data loss. As discussed in Section 3.2.2, the data precision metric used can have a great impact on the measured data loss and thus on the resulting clustering. Some metrics lead to finer grained results, thus no real clustering can be achieved by using the equality-function (see the case using the precision metric in Table 3.3). Here, a reasonable form of discretization (e.g. rounding or defining intervals) needs to be applied in order to generate usable classes.

Final distribution (Step 9) Before distribution, a list containing all assignments from generalization patterns to recipients is generated. We will later refer to this list as the *pattern-list*.

Principle of the identification of data leaks

In this section we introduce the algorithm for detecting data leaks (see Figure 3.6 for an illustration). The scenario is based on a data holder encountering data samples in the wild that he gave to other organizations in an anonymized (and fingerprinted) form.

1. The original data holder encounters some leaked data samples.
2. The underlying generalization pattern of this data is extracted.
3. The pattern is compared to the pattern list.
    a) In case the pattern directly matches the pattern of the data given to a recipient, this recipient is identified as the source of disclosure.
    b) Otherwise, the minimum of recipients that possess the knowledge (i.e. data) to be able to generate the encountered data set structure is calculated.

Figure 3.6 gives a more graphic explanation of the approach. It must be assured though, as for any other fingerprinting algorithm, that every fingerprinted set is unique to its recipient.

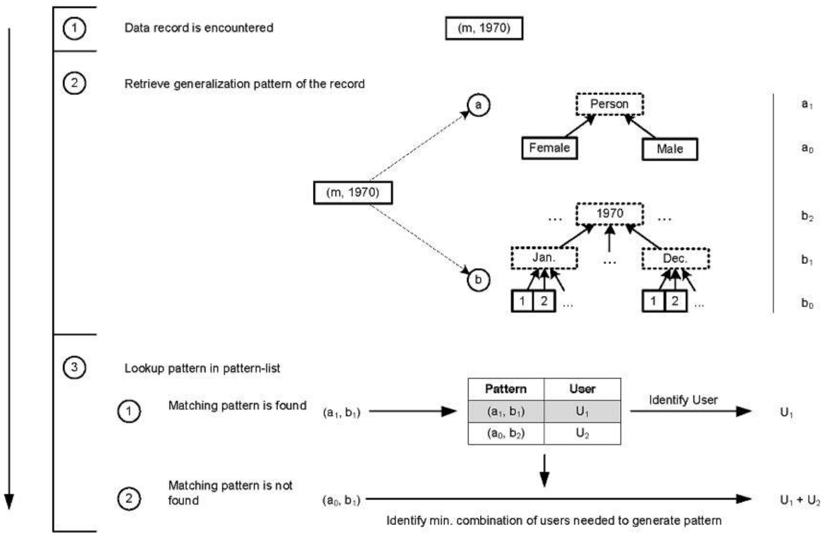

Figure 3.6: Identifying the source of data leakage based on generalization patterns.

Example 3.3.1. *Original Data D_0 is anonymized using two different patterns (a_1, b_1) and (a_0, b_2), thus generating the anonymized data sets D_1 and D_2 respectively. Recipient R_1 is provided with D_1 and R_2 with D_2. If, for example, a data record of the form (a_0, b_2) is disclosed (i.e. in our previous examples this would be a data record where "sex" is provided at original granularity and "birthdate" at year-granularity), it is easy to identify user U_2 as source of the leaked data, since user U_1 would not be able to provide this much detail on the QI "sex".*

Step-by-step description

For this example, we use the microdata from Table 3.2. In step one we define the side constraints $k = 2$, $l_{min} \geq 1$, and $l_{max} \leq 2$ (minimal and maximal level of data loss) and choose the Samarati metric for measuring data precision. In step two and three we use the generalization strategy from Figure 3.3, yielding the lattice diagram shown in Figure 3.7.

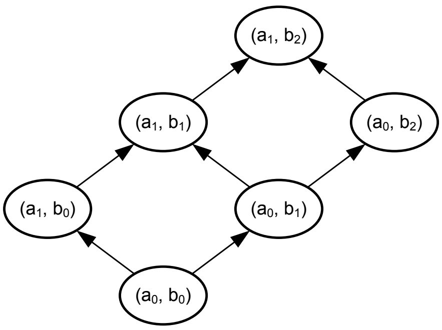

Figure 3.7: Lattice diagram after step three.

Analogous to El Emam's algorithm, we then choose a node of middle height as starting node for step four, e.g. (a_1, b_0). This generalization does not fulfill the $k - anonymity$

Sex (a)	Birthdate (b)
p	19.03.1970
p	20.03.1970
p	18.04.1970
p	21.04.1970

Table 3.4: Generalization (a_1, b_0).

criterion for $k = 2$, thus we can rule out all nodes below (a_1, b_0) (in our example, this applies to one node only: (a_0, b_0)), because they would not fulfill the criterion as well (step 4a). We now take the resulting subgraph of the chosen generalization path (step 5) and apply step 4 to node (a_1, b_1), which fulfills $k = 2$, and thus (by applying step 4b) the node (a_1, b_2) fulfills the 2-anonymity criterion as well. This leads to the intermediate lattice diagram shown in Figure 3.8.

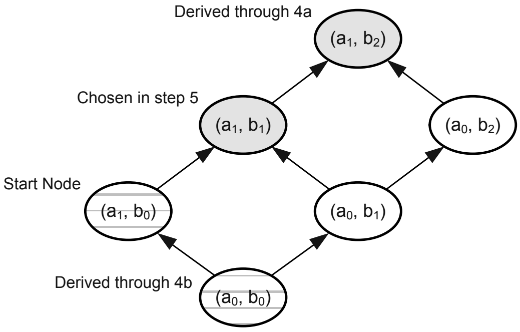

Figure 3.8: Lattice diagram after the first generalization path.

As no subgraph is left, we arrive at step 6 and again choose a starting node (a_0, b_1). It does provide 2-anonymity, so we can continue with step 4b and and mark node (a_0, b_2) as a possible solution. The resulting lattice diagram is shown in Figure 3.9.

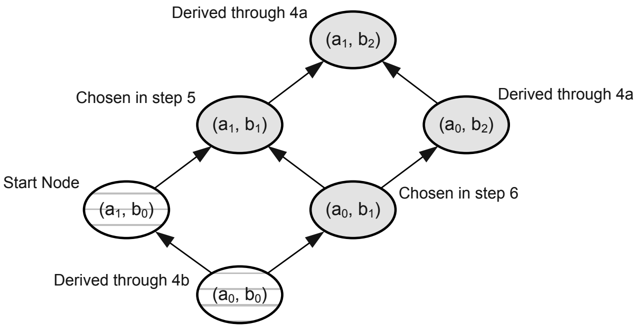

Figure 3.9: Lattice diagram after the second generalization path.

Since we now have traversed through the entire lattice diagram, we can proceed with step 7 and calculate the data precision and the actual k for each node that fulfills the k-anonymity criterion for $k = 2$ (see Table 3.5).

Node	Data Loss	k
(a_0, b_1)	1	2
(a_0, b_2)	2	2
(a_1, b_1)	2	2
(a_1, b_2)	3	4

Table 3.5: Data loss and k for solutions.

We remove node (a_1, b_2) from the solution set, since the data precision is out of our defined bounds for the data loss l_{min} and l_{max}. The remaining candidate microdata sets are shown in Table 3.6.

(a_0, b_1)		(a_1, b_1)		(a_0, b_2)	
Sex	Birthdate	Sex	Birthdate	Sex	Birthdate
m	03.1970	p	03.1970	m	1970
m	03.1970	p	03.1970	m	1970
f	04.1970	p	04.1970	f	1970
f	04.1970	p	04.1970	f	1970

Table 3.6: Generalization results.

Finally, by clustering the remaining data sets by data precision, we derive *generalization clusters*. All solutions in such a cluster will be treated as being equivalent with respect to the data quality provided. In some cases (e.g. when the classes are too small), we have to use ranges instead of exact values as classification criteria. In our example we derived two clusters: $C_1 = \{(a_0, b_1)\}$ with data precision of 1 and $C_2 = \{(a_1, b_1), (a_0, b_2)\}$ with data precision of 2.

In case the data needs to be sent to two recipients, we choose cluster C_2 and send data anonymized with strategy (a_1, b_1) to the first recipient and data anonymized with strategy (a_0, b_2) to the second one. Additionally, a list containing tuples of data recipients and generalization patterns used is stored by the provider. When data is disclosed, the provider can determine the generalization levels of the quasi-identifier attributes and then compare them to the list of stored generalization patterns. Thus, it is possible to identify the original recipient of the data based on the unique generalization variant.

3.3.3 Evaluation

In this chapter we focus on the evaluation of the following aspects:

1. Number of possible fingerprints
2. Robustness of the fingerprints
3. Removing the fingerprint by utilization of complementary releases

Number of fingerprints When utilizing this approach for fingerprinting, it is very important that the owner of the original data is able to generate enough fingerprints, i.e. there must exist enough suitable anonymization patterns so that each recipient can be given a unique data set. Unfortunately, this number depends on various parameters like number of quasi-identifiers and generalization levels, the anonymization level k, the data precision metric in use, and, last but not least, the actual data itself. Therefore, an exact number cannot be given without analyzing the source data. Still, an upper bound can be retrieved by plainly looking at the quasi-identifiers and their respective generalization strategies.

Be N the number of quasi-identifiers and n_i the number of generalization levels of the i-th QI. Then F is an upper bound for the number of different fingerprints with

$$F = \prod_{i=1}^{N} n_i$$

Robustness of a fingerprint In this paragraph we discuss how fingerprints devised by our method can be removed or changed in a way to avoid detection of the disclosing participant. Since the actual granularity of the data is used for fingerprinting, every removal must incorporate changing the generalization patterns. Since the attacker is not in the possession of finer grained data, the only reasonable way would lie in further generalization of one or more quasi-identifiers (actually the attacker could also "invent" finer granulations than he possesses, thus faking the data. Still, this would result in wrong, thus worthless data). This approach results in two major drawbacks:

1. In order to become undetectable, the disclosing party must be in the possession of the generalization strategy that was used for the other recipients, i.e. if U_1 received data with the pattern (a_1, b_2) and U_2 received (a_2, b_1), reducing U_2's granularity to (a_3, b_1) does not avoid U_2's detection.

2. Even if detection is avoided by the disclosing user, the data quality is reduced significantly. In our example, U_1 would at least need to generalize the data to the form (a_2, b_2) which is much more general than the data U_1 would be able to disclose.

Removing fingerprints through complementary releases Collusion attacks utilizing complementary releases can pose a severe threat to the anonymization of the data sets. Still, our fingerprint can lead to detection of the leaking parties, although it can not be guaranteed anymore. The following example shows detection of two collaborating leaking parties. The quasi-identifiers "birthdate" and "sex" are generalized like in the previous examples and "zip code" is granulated on two levels.

Revelation of the record {`p, 03.1970, 1015, short breath`} reveals data sets one and two as sources: The month of birth could only be extracted from data set one, the zip code in this granulation only from data set two. Still, our basic approach does not guarantee these detection capabilities, thus we will go into a deeper discussion on this issue in Section 3.4.

3.4 Providing Resilience against Collusion Attacks

The combination of two or more data sets can be used to subvert both, the anonymization, as well as the fingerprinting efforts (collusion attack, e.g. [ZL14]), which was not taken into account in the original approach proposed in Section 3.3 and the original papers. Therefore, in this extended approach, we analyze this threat and show which anonymization strategies

data set	name	sex	birthdate	zip code	disease
original data set	Bob	m	18.03.1970	1004	chest pain
	Dave	m	19.03.1970	1015	short breath
	Alice	f	20.04.1970	1004	obesity
	Eve	f	21.04.1970	1015	short breath
anonymized set 1	-	p	1970	1004	chest pain
	-	p	1970	1015	short breath
	-	p	1970	1004	obesity
	-	p	1970	1015	short breath
anonymized set 2	-	p	03.1970	100X	chest pain
	-	p	03.1970	101X	short breath
	-	p	04.1970	100X	obesity
	-	p	04.1970	101X	short breath
anonymized set 3	-	m	1970	100X	chest pain
	-	m	1970	101X	short breath
	-	f	1970	100X	obesity
	-	f	1970	101X	short breath

Table 3.7: Original data and three generalizations.

must be used to mitigate the resulting effects. Since these two aspects result in different requirements for the fingerprint, we will discuss them separately: Section 3.4.1 is concerned with the analysis of collusions against k-anonymity, whereas Section 3.4.2 proposes a strategy for identifying leaks. The reduction of the anonymization level may not only result in reputation loss of the data owner, it might even lead to legal actions or reveal critical business information to competitors. On the other hand, the subversion of the fingerprinting property of the scheme might lead to severe financial loss for companies that base part of their business on the trading of information, by making it publicly available without the possibility for the original data owner to hold the leaking party responsible for the disclosure. Most parts of this section are taken verbatim from [KSM+14].

Prerequisites 3.4.1. *As a prerequisite we assume that two different generalizations of the same record set can be identified and matched with each other, e.g. by using non-QIs like medical diagnosis or primary keys. This prerequisite constitutes a kind of worst-case scenario as we assume that the resilience of the anonymization solely relies on proper application of k-anonymity, still, in the case of large amounts of structured data, the ability to match records from differently anonymized sets must be taken into account to effectively protect sensitive data.*

Example 3.4.2. *Table 3.8 (a modification of Table 3.2 from Section 3.3) shows the original data together with two different anonymizations, the fields "sex" and "birthday" were identified as QIs. Both sets obey k-anonymity for $k = 2$, still the records can be matched by using the field "disease" which was not identified as QI (and will not be generalized in order to retain the data value).*

Original data set			Anonymized set 1			Anonymized set 2		
Sex	Birthdate	Diagn.	Sex	Birthd.	Diagn.	Sex	Birthd.	Diagn.
m	19.03.1970	diabetic	F	1970	diabetic	P	03.1970	diabetic
m	20.03.1970	disability	M	1970	disability	P	03.1970	disability
f	18.04.1970	blind	F	1970	blind	P	04.1970	blind
f	21.04.1970	disability	M	1970	disability	P	04.1970	disability

Table 3.8: Original microdata and two anonymized versions with $k = 2$

In this section we show how a systematic selection of anonymization strategies prevents collusion attacks targeting the anonymization level as well as the robustness of the fingerprint. In particular, the core contributions presented in this section are:

1. Description and analysis of collusions against our concept of k-anonymity based fingerprinting.
2. Protection of the anonymity level against collusions.
3. Achieving detectability of colluding attackers.
4. Presentation of an algorithm for constructing k-anonymity based fingerprints incorporating these results.

It must be noted that the approach proposed in this section does not constitute a strategy for defending the underlying anonymization technique against inference attacks utilizing arbitrary external data, thus providing a flexible approach that can be adapted to be used with a wide variety of other methods for anonymization.

3.4.1 Analysis of collusion attacks against the k-anonymity criterion

As a starting point for our analysis we formalize the information gain inherent to knowing a specific pattern by defining its *hull*. Using this, we formalize the information gain through collusions and the possible consequences for anonymization.

Definition 3.4.3 (Hull)**.** *Let A be a set of generalization patterns. The hull $H(a_0)$ of a pattern $a_0 \in A$ is the set of all patterns that can be constructed by lowering the granularity of QIs:*

$$H(a_0) := \{a_i \in A | a_{i_j} \geq a_{0_j}, \forall j = 1, \ldots, n\},$$

where a_{i_j}, a_{0_j} denote the generalization levels of the j-th QI of the patterns a_i and a_0 respectively. Furthermore, $a_{i_j} > a_{0_j}$ denotes that the generalization level of the j-th QI is higher in the pattern a_i than in pattern a_0, i.e. that the granularity of pattern a_0 is higher than of a_i with respect to the j-th QI. This slight change in notation compared to the previous sections is needed since we will have to compare several patterns and wanted to avoid double indices before.

In other words, the hull of a pattern contains all generalizations that can be constructed by an attacker without additional information. Figure 3.10 shows an example lattice diagram based on two QIs together with the hull of a pattern.

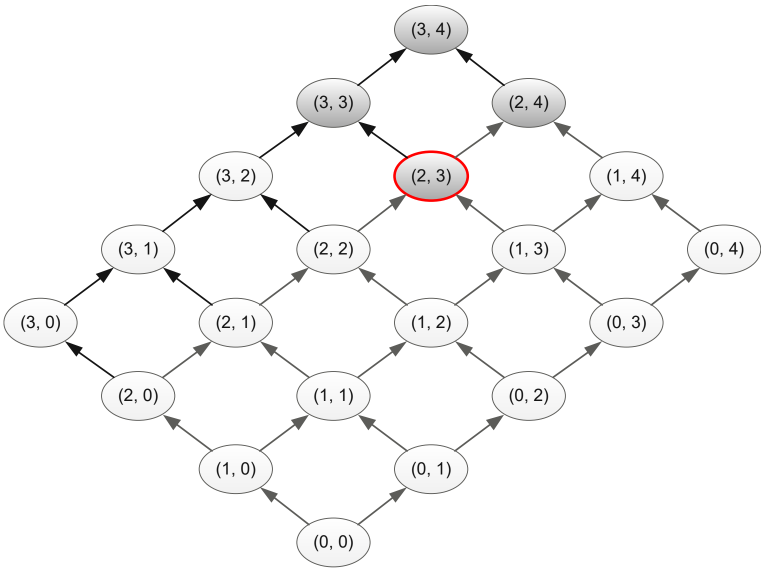

Figure 3.10: The hull of a generalization pattern

Corollary 3.4.4. *If a given pattern a_0 obeys k-anonymity for a given k, then all elements of the hull $H(a_0)$ obey k-anonymity for at least the same k.*

Proof. The proof for this corollary is rather trivial: Following the definition of the hull it holds true that $\forall a_i \neq a_0 \in H(a_0) : (\exists j : a_{i_j} > a_{0_j}) \wedge (\nexists j : a_{i_j} < a_{0_j})$, i.e. for all patterns in the hull $H(a_0)$ except for a_0 itself at least one QI has a higher generalization level, while none has a lower one (this is derived directly from the definition of the hull). □

Following we want to analyze, what patterns can be constructed by applying collusions against two different known patterns $a = (a_i)_{i=1,\ldots,n}$ and $b = (b_i)_{i=1,\ldots,n}$. By using the definition of the hull, all patterns within the union of the two hulls can be constructed trivially (Figure 3.11 shows an example with two QIs).

Furthermore, with the prerequisite given at the start of the section, the possibility of matching the same record in differently anonymized sets, it is possible to construct the pattern $c = (min(a_1, b_1), \ldots, min(a_n, b_n))$ and thus the hull $H(c) \supseteq H(a) \cup H(b)$ (Figure 3.12 shows an example involving two QIs). We call c the *minimal generalization pattern* with respect to the patterns a and b. The minimal generalization pattern always exists because either, without loss of generality, $a \in H(b)$ trivially yielding $c = b$ (and vice versa for $b \in H(a)$), or not. In the latter case it is always possible to use the construction by calculating the minimal generalization levels with respect to each QI. Since we

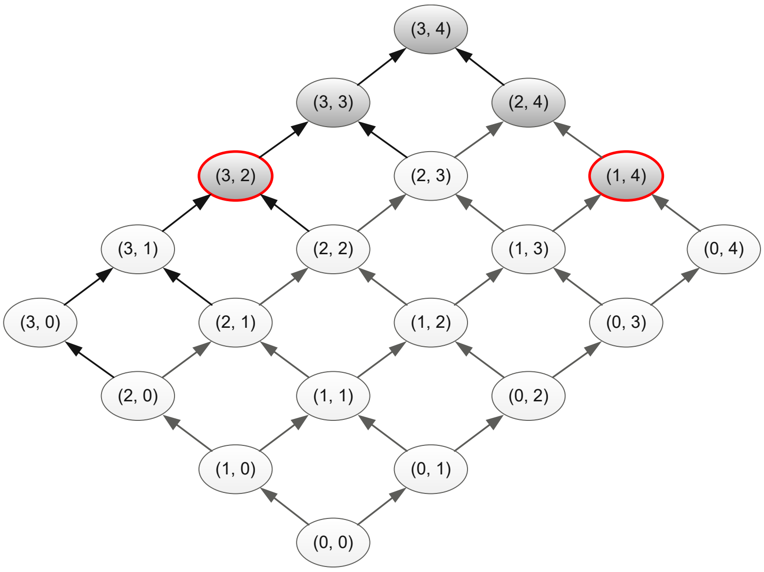

Figure 3.11: The union of the hulls of two patterns

implicitly assume a finite number of QIs and the ">"-relation forms a well-ordering on the generalization levels for each QI, the minimum can always be calculated.

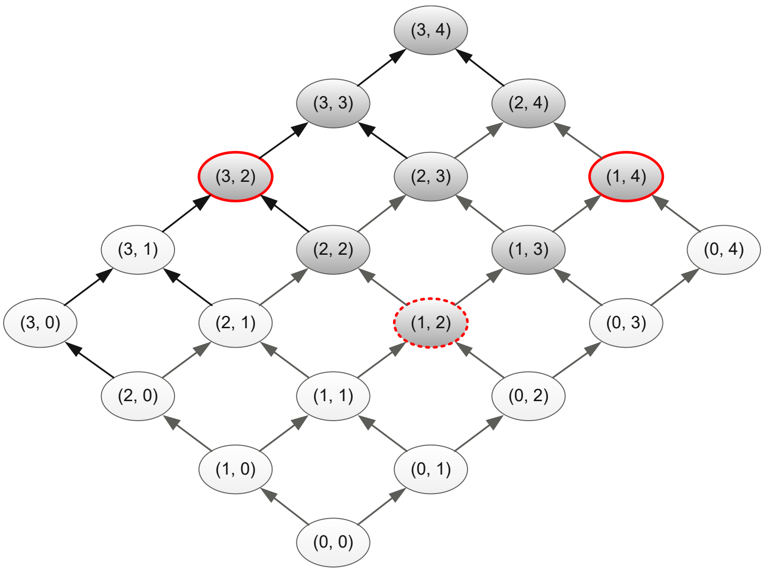

Figure 3.12: The hull generated by colluding two patterns

This can be further extended to the general case of a finite number of generalization patterns:

Definition 3.4.5. *Let $A = (a_i)_{i=1,\ldots,r}$ be a set of r generalization patterns with respect to*

n QIs. We define the hull $H(A)$ as $H(\overline{a})$, where $\overline{a}$ is the minimal generalization pattern with respect to all $a_i \in A$.

$$H(A) := H(\overline{a}) = H(min(a_{1_1}, \ldots, a_{r_1}), \ldots, min(a_{1_n}, \ldots, a_{r_n}))$$

Since this is just an extension of the two-dimensional to the general finite-dimensional case, the existence of the minimal generalization pattern can be derived easily by using complete induction (to be precise, complete induction proofs this for the case of countably infinite quasi identifiers).

Theorem 3.4.6. *Let A be a set of generalization patterns distributed to consumers, each obeying the k-anonymity condition for k. Then the level of anonymity of the data cannot be reduced beyond k by any selection of colluding consumers if and only if the minimal generalization pattern of A at least obeys the same k-anonymity condition.*

Proof. In case the minimal generalization pattern does not obey k-anonymity for k, a collusion of all consumers breaks the anonymization. On the other hand, the minimal generalization pattern contains the most granular generalization of each QI with respect to the information given to all consumers, i.e. it is not possible to generate another pattern from the patterns owned by the consumers that has a higher granulated QI. Thus, in case the minimal generalization pattern obeys k-anonymity, all other patterns derived by colluding the consumers data sets obey at least the same criterion. □

3.4.2 Identification of colluding attackers

Following we discuss how to choose generalization patterns to identify the colluding attackers. For this, we need more prerequisites regarding our attacker model:

Prerequisites 3.4.7. *Like in the original fingerprinting approach, we assume that attackers always try to generate the best possible data set, i.e. they are not willing to reduce the granularity of a QI in case a higher one is available to them. One reason for this lies in the fact that every fingerprint (even without considering collusions) can be removed trivially by generalizing all QIs to the maximum generalization level (of course this includes losing virtually everything in terms of quality). Furthermore, we assume that consumers want to generate as good data as possible through collusions. As an additional prerequisite, we assume that attackers do not know the patterns distributed to non-colluding consumers. Please note that these prerequisites are not needed in Section 3.4.1.*

Thus, the following targets need to be achieved by such a solution in order to be defined as resistant to collusion attacks with respect to our prerequisites:

1. All colluding partners shall be identified.

2. No innocent partner shall be suspected wrongfully.

The first observation is that any pattern a_i within the hull of a pattern a_0 does not fulfill above requirements since a collusion would go undetected. Generalization of this statement leads us to the following theorem:

Theorem 3.4.8. *Let $A = \{a_i | i = 1, \dots, r\}$ be a set of patterns. Then A constitutes a set of patterns resistant to collusions, if and only if $a_i \notin H(A \backslash a_i), \forall i = 1, \dots, r$.*

Proof. (1) Let $a_0 \in A$ be a pattern for which the precondition holds true, i.e. $a_0 \notin H(A \backslash a_0) =: H(\overline{a_0})$, where $\overline{a_0}$ is the minimal generalization pattern of $A \backslash a_0$. Thus follows that a_0 has at least one QI a_{0_j} of finer granularity than $\overline{a_0}$, thus is not constructible by using solely the patterns in $A \backslash a_0$. If a set anonymized with a pattern containing a_{0_j} with the granularity found in a_0, it can be guaranteed that the user holding pattern a_0 participated in the collusion. If generalized for all $i = 1, \dots, r$, the first implication follows trivially. (2) On the other hand, if all colluding partners shall be identifiable, it must not be possible to construct a pattern from the set containing the other patterns $A' = A \backslash a_0$, i.e. $a_0 \notin H(A') \Rightarrow \exists a_{0_j} : a_{0_j} < a_{i_j}, \forall i \neq 0$. Since this must hold true for all patterns a_i, we can generalize that $\forall l = 1, \dots, r : \exists a_{l_j} : a_{l_j} < a_{i_j}, \forall i \neq l$ trivially leading to $a_i \notin H(A \backslash a_i), \forall i = 1, \dots, r$. $\square$

Example 3.4.9. *Figure 3.13 shows three patterns $a_1 = (3, 2), a_2 = (0, 2)$ and $a_3 = (0, 4)$. In case of collusion of the partners holding data generalized with a_1 and a_3, the partner having data of the form a_2 could be accused innocently. A collusion of partner a_2 and a_3 would not be detected, partner a_3 would still be deemed trustworthy.*

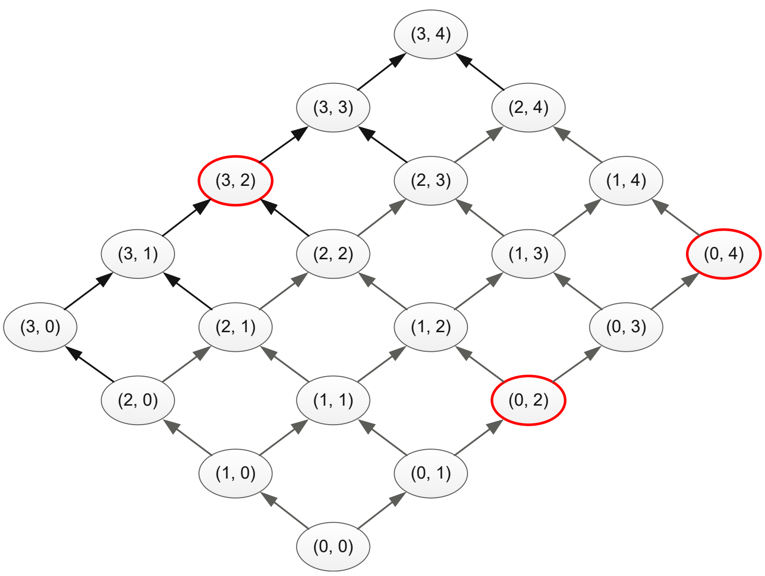

Figure 3.13: Example for a collusion

The above theorem states that every pattern in A has to be more detailed than all the others with respect to at least one QI and the attacker model outlined at the start of this section.

Corollary 3.4.10. *The maximum number of independent patterns is bound by the number of QIs, i.e. the number of partners must be smaller or equal than the number of QIs.*

It must be noted that detecting colluding attackers is completely independent from providing sets that obey k-anonymity. Thus in real-life scenarios the number of possible colluding consumers that can be detected while still obeying k-anonymity may be much smaller.

3.4.3 Modifying the algorithm for a privacy preserving generalization strategy

Based on the last section and the algorithm proposed in Section 3.3.2, we propose an algorithm for generating a privacy preserving solution. This solution is needed to achieve both features mandatory in a fingerprinting scheme targeting information business: Providing an irreducible anonymity level while maintaining enough stability in order to identify colluding data recipients.

Privacy preserving generalization strategies

Definition 3.4.11. *We define a set A consisting of r generalization patterns as privacy preserving, when the solution is conforming to the following properties:*

1. *Each pattern at least obeys the chosen anonymization level k.*
2. *The pattern generated by colluding all r patterns of A at least obeys k-anonymity for the same k.*
3. *In case of collusions, it is possible to identify all participants.*
4. *All patterns $a_i \in A$ have the same quality with respect to the chosen DPM and tolerance.*

If several privacy preserving generalization strategies are found, it is necessary to choose a "best" solution. Several measures can be used for defining this solution, e.g. lowest average data loss, lowest difference in quality between the patterns or highest number of patterns.

As discussed in Section 3.4.1, a privacy preserving solution as defined above can only be found in case the number of consumers r is not greater than n, the number of QIs. Depending on the actual sets and the anonymization level demanded, r may even be significantly smaller than n to be able to provide a generalization strategy that obeys the criteria mentioned above. Thus, it may be useful to define a weaker definition for privacy preserving generalization strategies that can be obeyed more easily while still providing reasonable security. However, we believe that typical databases in many application domains like medical health records or omics-data contain enough QIs for our approach to be practical.

The algorithm

The algorithm proposed in this section, like the algorithm proposed in Section 3.3.2, is based on El Emam's algorithm for calculating an optimal solution for the k-anonymity problem of single sets [EEDI+09] (i.e. finding the pattern with minimum information loss), combined with the results from the previous sections concerning resilience against collusions.

1. All side parameters are defined: The anonymization level k, minimum/maximum bounds for data loss l_{min} and l_{max}, the DPM and the tolerance t. Furthermore, the algorithm terminates in case $n < r$.
2. For each QI, a generalization strategy including the different levels of granularity is defined.
3. The lattice diagram holding all possible patterns is calculated.
4. A node at middle height is chosen and tested for k-anonymity.
   a) If it does not fulfill the selected k-anonymity criterion, all nodes below in the lattice diagram (including the current one) are ruled out as possible solutions.
   b) In case it does, all nodes above (including the current one) are marked as possible solutions.
5. Step four is repeated for the remaining subgraphs. If all nodes below were ruled out (4.a), one QI of the node used in step 4 is increased by one and used as input in step 4. In case of 4.b, the same approach is followed with one QI decreased. This step is repeated for all remaining subgraphs until all nodes in the original subgraph are evaluated.
6. In case no subgraph is left, another still unevaluated node at middle height is chosen and the algorithm proceeds with step four until all nodes are evaluated.
7. For each node marked as possible pattern, the DPM and the actual k is calculated. All patterns with a precision outside l_{min} and l_{max} are removed.
8. The remaining patterns are clustered into equivalency classes according to their quality and the tolerance t.

9. For each resulting class, the minimal generalization pattern is generated. Classes resulting in a pattern not obeying k-anonymity are removed from the possible solutions.

10. For each remaining class, the ability of detecting colluding attackers is tested according to Section 3.4.2 and eventually removed from the set of possible solutions.

11. If the set of possible solutions is empty, t, l_{min} and l_{max} are relaxed with respect to a predefined strategy. The algorithm then proceeds with step seven in case $t \leq l_{max}$ (in case $t \geq l_{max}$, t can obviously be set to l_{max} without losing any solutions. The algorithm then proceeds with step seven too).

12. If more than one class remains, the best solution is chosen (see Section 3.4.3).

13. The data sets for the data recipients are generated.

Following we will provide some annotations on the most important steps:

DPMs and maximum levels (Step 1): The algorithm should provide approximately the same level of precision to all consumers. Still, it may not be possible to find a generalization strategy with r patterns possessing exactly the same quality. Thus, the tolerance parameter t is defined as the maximal tolerated difference in quality inside an equivalency class, i.e. $t \geq (max_i(M(a_i)) - min_j(M(a_j)))$, where M is the DPM used. The parameters l_{min} and l_{max} hold the bounds for the minimal and maximal data quality of a valid solution.

Eliminating nodes (Step 4): Since we need the set of all possible solutions to construct the equivalency classes (this is a major deviation from the algorithm outlined in [EEDI+09]), the nodes holding less granular patterns cannot be removed from the set of possible solutions in case of (4.b).

Clustering the solutions (Step 8): Step eight clusters the patterns into classes by their quality and with respect to the tolerance t. Patterns may (and with a higher t will most likely be) inside several such equivalency classes, especially since all subsets of clusters are valid classes too. Classes holding less than r patterns are removed from the list of solutions.

Finding a privacy preserving solution (Steps 9 and 10): In these steps, the previous results are incorporated to find a privacy preserving solution as defined in Section 3.4.3. Step nine guarantees that colluding all patterns of the resulting generalization strategy does not result in a pattern that fails the k-anonymity criterion, while step ten guarantees the identification of each colluding data recipient.

Relaxing the side parameters (Step 11): In case no solution for a privacy preserving generalization strategy with respect to the parameters t, l_{min} and l_{max} is found, the algorithm tries to relax these parameters and proceeds with step seven to find a privacy preserving generalization strategy with lower requirements. The strategy for relaxing these parameters must be defined by the user beforehand. Choosing suitable strategies depends very much on the actual data and requirements and is not part of this work.

Choosing the best solution (Step 12): Several strategies may be chosen for selecting the best privacy preserving strategy: Section 3.4.3 gives some ideas on reasonable choices for comparing the solutions.

In the course of this chapter we always demonstrated our approach with respect to the common concept of k-anonymity. In case other, related, concepts like l-diversity or t-completeness are used, the check for compliance of the pattern in step 4 has to be adapted to resemble the new criteria put forth in Section 3.6.

Example

To demonstrate the applicability of the approach, we provide a small example. Due to better readability we will omit unnecessary intermediate results.

Step 1: Choosing the side parameters: We choose an anonymization level $k = 2$, $l_{min} = 0$, $l_{max} = 4$ and $t = 0$ using the Samarati Metric (SM). Furthermore we want to distribute the fingerprinted data to three recipients. The original data can be found in Table 3.9 (the attributes ID, birthday, ZIP-Code and sex, with generalization level 0, i.e. the columns with “0” in the second row).

	Birthdate				ZIP code				Sex	
ID	0	1	2	3	0	1	2	3	0	1
1	31.05.1970	05.1970	1970	70s	1042	104	10	1	F	P
2	25.04.1970	04.1970	1970	70s	1062	106	10	1	M	P
3	16.05.1970	05.1970	1970	70s	1041	104	10	1	F	P
4	08.04.1970	04.1970	1970	70s	1062	106	10	1	M	P

Table 3.9: All generalization levels for the chosen data set

Step 2: For the QIs birthday, ZIP-Code and sex the following generalization levels are defined (see Table 3.9 for all generalization levels):

- Birthdate: Exact to (0) day, (1) month, (2) year, (3) decade.
- ZIP-Code: Exact to (0) sub district (4 digits), (1) district (3 digits), (2) inner/outer city (2 digits), (3) city (1 digit).

- Sex: Either (0) female (F)/male (M) or (1) person (P).

Step 3: The lattice diagram holding all possible generalization patterns is calculated.

Step 4: Since birthdate and ZIP-Code both possess 4 generalization levels $(0,\ldots,3)$ and sex possesses 2 levels $(0,1)$, the maximum height of the lattice diagram is $7 \quad (3+3+1)$, the minimal $0 \quad (0+0+0)$, thus all levels with a combined sum of generalization levels of 4 are at middle height (since we use the Samarati Metric for measuring the data precision, the definition of the height in the lattice diagram and the data loss is equal. This does not hold for other DPMs though). In our example seven nodes with height 4 exist. A node at middle height is chosen at random and checked for compliance with the k-anonymity criterion for $k = 2$. For example we choose the node $a_0 = (0,3,1)$ which does not obey the criterion (see Table 3.9). Thus all nodes (x,y,z) with $x \leq 0$, $y \leq 3$ and $z \leq 1$ are removed from the set of possible generalization patterns (see Step 4.a in the algorithm).

Step 5: Step four is repeated for the remaining subgraphs, i.e. a node that can be derived by lowering the granularity of one QI in a_0 is constructed and used as input for step 4, e.g. $(1,3,1)$. This node fulfils k-anonymity, thus all nodes (x,y,z) with $x \geq 1$, $y \geq 3$ and $z \geq 1$ are added to the list of possible generalization patterns. This is done for all remaining subgraphs.

Step 6: The steps four and five are repeated until all nodes at middle height are evaluated. Table 3.10 shows the set of possible generalization patterns obeying the k-anonymity criterion for $k \geq 2$ before applying the bounds for the data precision.

Pattern	SM	k	Pattern	SM	k	Pattern	SM	k	Pattern	SM	k
(1,1,0)	2	2	(1,3,0)	4	2	(2,2,0)	4	2	(3,1,0)	4	2
(1,1,1)	3	2	(1,3,1)	5	2	(2,2,1)	5	4	(3,1,1)	5	2
(1,2,0)	3	2	(2,1,0)	3	2	(2,3,0)	5	2	(3,2,0)	5	2
(1,2,1)	4	2	(2,1,1)	4	2	(2,3,1)	6	4	(3,2,1)	6	4
(3,3,0)	6	2	(3,3,1)	7	4						

Table 3.10: Possible generalization patterns for $k \geq 2$

Step 7: All patterns with data loss higher than 4 are removed, for all remaining nodes the actual k is calculated.

Step 8: The remaining generalization patterns are clustered with respect to the tolerance level $t = 0$. Table 3.11 gives an overview on the resulting clusters (in order to keep this example well-arranged, it already contains the results from steps nine (the minimal generation pattern $\bar{a}$) and the clusters are grouped by their data loss and the number of patterns respectively).

ID	SM	# Patterns	Patterns	$\bar{a}$	k for $\bar{a}$
1	3	3	(1,1,1),(1,2,0),(2,1,0)	(1,1,0)	2
2	4	5	(1,2,1),(1,3,0),(2,1,1),(2,2,0),(3,1,0)	(1,1,0)	2
3	4	4	(1,3,0),(2,1,1),(2,2,0),(3,1,0)	(1,1,0)	2
4	4	4	(1,2,1),(2,1,1),(2,2,0),(3,1,0)	(1,1,0)	2
5	4	4	(1,2,1),(1,3,0),(2,2,0),(3,1,0)	(1,1,0)	2
6	4	4	(1,2,1),(1,3,0),(2,1,1),(3,1,0)	(1,1,0)	2
7	4	4	(1,2,1),(1,3,0),(2,1,1),(2,2,0)	(1,1,0)	2
8	4	3	(1,2,1),(1,3,0),(2,1,1)	(1,1,0)	2
9	4	3	(1,2,1),(1,3,0),(2,2,0)	(1,1,0)	2
10	4	3	(1,2,1),(1,3,0),(3,1,0)	(1,1,0)	2
11	4	3	(1,2,1),(2,1,1),(2,2,0)	(1,1,0)	2
12	4	3	(1,2,1),(2,1,1),(3,1,0)	(1,1,0)	2
13	4	3	(1,2,1),(2,2,0),(3,1,0)	(1,1,0)	2
14	4	3	(1,3,0),(2,1,1),(2,2,0)	(1,1,0)	2
15	4	3	(1,3,0),(2,1,1),(3,1,0)	(1,1,0)	2
16	4	3	(1,3,0),(2,2,0),(3,1,0)	(1,1,0)	2
17	4	3	(2,1,1),(2,2,0),(3,1,0)	(1,1,0)	2

Table 3.11: The constructed clusters

Step 9: For each cluster, the minimal generalization pattern is constructed and tested whether it obeys k-anonymity. Due to the simple metric, small tolerance and small data set, the minimal generalization pattern for each cluster is the same in this example.

Step 10: All clusters that do not provide the ability to detect colluding attackers are removed from the sample, reducing the set of possible results to the pattern with ID 11 (see Table 3.11).

Steps 10 and 12: The side parameters t, l_{min} and l_{max} could be lowered to generate more solutions, which we omit in this example. The data is anonymized with the resulting patterns from the cluster with ID 11 and distributed (see Table 3.12).

Data set 1			Data set 2			Data set 3		
Birthdate	ZIP code	sex	Birthdate	ZIP code	sex	Birthdate	ZIP code	sex
05.1970	10	P	1970	104	P	1970	10	F
04.1970	10	P	1970	106	P	1970	10	M
05.1970	10	P	1970	104	P	1970	10	F
04.1970	10	P	1970	106	P	1970	10	M

Table 3.12: The distributed data sets

Evaluation

In this section, we evaluate the presented approach with respect to certain key features for fingerprinting algorithms. It must be noted that some criteria are heavily dependent on the actual use case, e.g. the number of achievable fingerprints.

Number of possible data consumers: For our approach it is vital that every consumer receives the data anonymized with a unique, non-reusable pattern. In general, the number of possible fingerprints is heavily depending on side-parameters like the anonymization level, the minimal and maximal allowed data quality and clustering tolerance, as well as on the data itself. Furthermore, since the approach requires a privacy preserving generalization strategy, the number of possible patterns is reduced drastically and can be limited to $F_{opt} \leq n$. Still, e.g. considering call-detail-records (CDRs) in interconnection-billing or anti-fraud systems for telecommunication providers [RR98], the typical data set possesses much more QIs (exceeding 50 for a typical ICB-system) than possible recipients (usually less than 10), thus rendering this approach perfectly feasible. Other application fields where we see no problem with the number of possible data recipients are medical research, especially considering large omics-databases (e.g. [DWY$^+$10]), as well as PSE process implementations (e.g. [WRK19]), where few partners meet a large amount of quasi identifiers.

Robustness of the fingerprints and the anonymization: Another important evaluation criterion lies in the robustness of the fingerprint against tampering and removal. Since the fingerprint is constituted by the intrinsic structure of the data, the only way to remove such fingerprints lies in changing the data structure, i.e. changing the underlying anonymization pattern. In order to hide, the consumer would need to find a pattern that is either distributed to another consumer (which is impossible when using a privacy preserving generalization strategy), or that at least lies in the hull of another distributed pattern. To achieve this, the attacker must (i) know the pattern used for another consumer and (ii) know the generalization levels of the quasi identifiers. At any rate, the value of the leaked data will be drastically reduced, since the attacker is only able to reduce the granularity of known data in order to hide the pattern. Concerning the robustness of the anonymization, the proposed approach ensures that the k-anonymity criterion cannot be broken by inferencing an arbitrary selection of the fingerprinted data sets. Still, attacks against the concept of k-anonymity itself, especially considering inference attacks involving external data, may lead to unwanted disclosure.

Example 3.4.12. *Let A and B be two consumers, where A gets data anonymized with pattern* $(3, 1)$ *and B with pattern* $(1, 3)$. *Assuming A knows the pattern for B and the generalization levels for each QI, the best unidentifiable pattern that A can generate would be pattern* $(3, 3)$.

Validity of the algorithm: For the practical applicability of the approach, an algorithm that is able to construct the fingerprint is needed. Thus we will show that the proposed algorithm terminates and that all possible privacy preserving solutions are generated: Steps 1 to 3 in the algorithm set the side parameters and generate the generalization levels and the lattice diagram which terminates trivially in case of a finite number of QIs and possible generalizations. Steps 4 to 6 iterate through all nodes in order to decide if they obey k-anonymity. In every invocation of step 4, nodes are either marked as possible patterns or removed from the evaluation. In both cases, these nodes will not be chosen by steps 5 or 6 as input node for step 4, thus guaranteeing that each node is evaluated at most once. Thus this cycle terminates after a finite number of iterations. Step 7 terminates trivially, step 8 generates all possible clusters from the set of possible patterns which is again finite. Since the minimal generalization pattern always exists (see Section 3.4.1), step 9 and 10 terminate too. Furthermore, the algorithm needs to generate all solutions. Lets assume one solution is missing: Since the solutions are generated by clustering all patterns obeying k-anonymity into equivalency classes, it follows trivially that at least one pattern a_0 must be missing. Since the lattice diagram calculated in step 3 holds all possible patterns, a_0 must have been lost in the loops implemented in steps 4 to 6. This part is functionally equivalent to the algorithm proposed in[EEDI+09] and proven to be complete: For each middle node, either the upper subgraph (including the middle node) can be marked as possible pattern, or can be ruled out. For each middle node the algorithm cycles through the remaining subgraph and the process is repeated until no unevaluated nodes are left.

Detection capabilities: Another highly important evaluation criterion is the detection capability of a fingerprinting algorithm. In Section 3.1, we already put forth the design criterion that, contrary to other fingerprinting mechanisms, detection should be made possible by a single leaked data record, as this is a major issue in many sensitive application areas like the medical domain. This can be achieved by design, furthermore, the loss of all columns containing non-QIs and even selected columns containing QIs can be tolerated in certain practical examples.

3.5 Generalization to advanced methods for achieving k-anonymity

One major aspect that is often neglected in the description of k-anonymity and related approaches for practical usage, as well as in academic papers, covers alternative methods for achieving said criteria. Typically, authors describe a form of generalization that is called *full-domain generalization*, where all records are treated similar. Some authors also add simple forms of *suppression* in order to cater for the removal of outliers that can severely impact the anonymization process, especially in case of full-domain generalization. The approaches outlined so far in this chapter have also been developed with full-domain generalization in mind. Still, while full-domain generalization is simple and straightforward, it faces major problems when dealing with unequally distributed data, and especially in case of hierarchical qualitative concepts which only allow for an ”unevenly” distributed generalization into more general concepts. Thus, within this section, we will cover the following alternative methods for generating k-anonymous data sets. The selection has been quite broad by not only including prominent examples typically encountered in practical implementations, but also more complex and rarely encountered methods:

- Several forms of **generalization** and
- some types of **suppression**, as well as
- **anatomization**.

Typically, these extensions will have an impact on the number of fingerprints available, but also on the performance of the fingerprint generation. This section, as well as Section 3.6 are under submission at the time of submitting this thesis.

3.5.1 Advanced generalization methods

We will start the extension to advanced methods aiming at different forms of generalization.

Full-domain Generalization: This is the most prominent form of generalization and the one that has been used to develop the original approaches in this chapter. The main characteristic of this method lies in the fact that a generalization strategy is chosen for each quasi-identifier as a whole, i.e. the selected QI is generalized in the same way for all records, independent of it's actual value in the respective record.

Subtree Generalization: While in the full-domain generalization approach, the same generalization level is chosen for all values of a given quasi-identifier, *subtree generalization* [Iye02] allows for choosing different levels in each subtree of the generalization tree. Figure 3.14 together with Table **??** illustrate the difference for the quasi-identifier "age". It must be noted though that one additional important prerequisite for choosing the patterns exists: For a non-leaf node, either all child nodes are generalized, or none, i.e. the generalization takes place on the subtree level (hence the name).

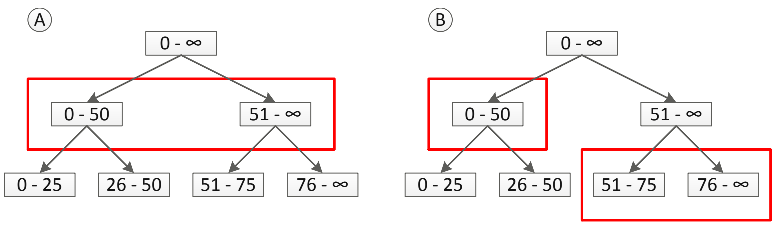

Figure 3.14: Examples for full-domain (A) and subtree (B) generalization.

Subtree generalization allows for a broader range of generalization patterns and finer levels of granularity may be chosen for selected values or intervals of a quasi-identifier, especially when the pattern needed in order to achieve k-anonymity in the full-domain case is influenced by one sparsely populated value range (see Table **??** with respect to Figure 3.14).

Original data			Full-domain		Subtree	
ID	sex	age	sex	age	sex	age
1	m	26	m	[0 - 50]	m	[0 - 50]
2	m	44	m	[0 - 50]	m	[0 - 50]
3	f	52	f	[51 - ∞]	f	[51 - 75]
4	f	62	f	[51 - ∞]	f	[51 - 75]

Table 3.13: Generalizing QI "age" for achieving k-anonymity ($k = 2$)

Sibling Generalization: Sibling generalization is a method very similar to subtree generalization with one difference: Instead of the basic requirement that all nodes of a subtree need to be generalized to the same level, it is possible that some siblings remain ungeneralized. The parent node is then chosen as representative of all missing child nodes. Figure 3.15 (B) shows an example: The intervals [19-30] and [31-50] are generalized to the parent interval [0-50], while the child interval [0-18] of [0-50] remains ungeneralized. In general, sibling generalization produces less data loss compared to subtree generalization, since only child nodes violating the k-anonymity criterion need to be generalized, still, this is very much depending on the actual data and precision metric in use.

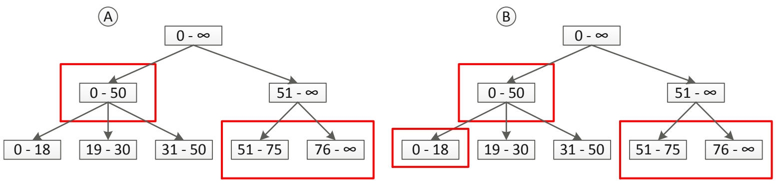

Figure 3.15: Examples for subtree (A) and sibling (B) generalization.

Fingerprinting with subtree generalization

In this section, we will discuss the adaption of the basic fingerprinting approach to subtree generalization.

Fingerprint generation & identification: Since the original approach uses full-domain generalization, the generation of the fingerprints in the subtree generalization case works very similar. The only difference lies in the selection process of the patterns, where more possibilities for each QI exist. Furthermore, in order to guarantee stability, some prerequisites for the selection exist even for the non-collusion case (see paragraph *Stability* below). The identification procedure works exactly like in the original approach, the generalization pattern of the encountered data record is calculated and compared to the list matching the data recipients with the patterns.

Diversity: The number of fingerprints that can be generated by this method is theoretically higher than in the original approach, since the number of different generalizations may be higher for each QI. Still, in order to achieve stability, the criteria outlined in paragraph *Stability* must be obeyed, again reducing the number of possible fingerprints depending on the actual data. Still, as a lower bound, the number of fingerprints when using subtree generalization is at least the number of fingerprints in the full-domain case. This is easy to proof, since every pattern in the full-domain case forms a pattern in the subtree case that trivially follows the stability criteria outlined below. On the other hand, all other valid patterns in the subtree case are not valid when using full-domain generalization.

Stability: Since this adaption of the fingerprinting approach relies on the uniqueness of the generalization pattern for each data recipient, it must be ensured that this prerequisite is kept for all possible values inside the anonymized data sets, in order to be able to identify the fingerprint based on each single record. While this is trivial for the full-domain case, it adds some difficulty when using subtree generalization: For example, let us assume that two data sets D_1 and D_2 are anonymized with two patterns p_1 and p_2 respectively and the only difference between these patterns lies in the quasi-identifier "age": p_1 is anonymized using pattern (A) from Figure 3.14 (which is also a subtree in addition to a full-domain generalization), p_2 is using pattern (B). Even though the anonymization patterns differ in general, there may exist records that are similar in both patterns, e.g. any record with an age between 0 and 50. The reason behind this is that even though the patterns are different *in general*, they are not different in *each record*. There are actually two simple solutions for mitigation: (i) Reducing the requirement of being able to identify the data leak in every single record or (ii) providing uniqueness of the pattern in each record. While the first mitigation strategy is trivial, it reduces the usability of the overall approach drastically, thus leaving the second one as the better option: As an additional prerequisite for the selection of the patterns it must hold true that each pattern is unique for every record encountered in the original set. This is a relaxation of the more strict requirement of being unique for every possible record, since only the data actually fingerprinted needs to be taken into account: The above mentioned examples for p_1 and p_2 could be a perfectly fine selection, in case no records with values between 0 and 50 for the QI "age" exist in the original set. Furthermore, in the general case, the uniqueness of a pattern does not solely depend on a single QI, but on the combination of several ones, so problems in the uniqueness with respect to one single QI may even themselves out when regarding the whole pattern. Thus we arrive with the following additional selection criteria during the generation phase:

1. In case of the relaxed requirement that each pattern must be unique for every encountered record, the patterns need to be checked for uniqueness at generation time.

2. In case of the more strict requirement that each pattern must be unique for every possible valid record (within the expected value ranges), it is possible to rule out

all patterns p_j where it holds true that there exists an already selected pattern $p_i, i \neq j$ and a QI l with $p_{j_l}(V) = p_{i_l}(V)$ for any possible legitimate value V.

Resilience against inference attacks: In general, the results for the approach using full-domain generalization given in Section 3.4 can be adapted, with only requiring the incorporation of different granularization levels per pattern, depending on the actual value.
Let A be the set of all possible patterns and X_i be the set of all possible values for the i^{th} quasi identifier. The *hull* of pattern a_0 is defined as the set of all patterns that can be constructed by lowering the granularity of quasi identifiers for a set of values:

$$H(a_0) := \{a_i \in A | a_{i_j}(x) \geq a_{0_j}(x), \forall j = 1, \dots, n, \forall x \in X_j\}$$

where $a_{i_j}(x)$ and $a_{0_j}(x)$ denote the the generalization levels of the j^{th} quasi-identifier with respect to the value $x \in X_j$. Furthermore, the definition of the $\geq$-relation denotes that the granularity of the pattern on the left is lower than on the right side.
Based on this adaption of the hull, we can define the *minimal generalization pattern* with respect to subtree generalization as

$$\bar{a} := (\min(a_{1_1}(x), \dots, a_{r_1}(x)), \dots, \min(a_{1_n}(x), \dots, a_{r_n}(x)),$$

where n is the number of quasi-identifiers and r the number of patterns. Let U be the set of patterns obeying the k-anonymity criterion that is distributed to the data recipients. Following the proof given in Section 3.4 and exchanging the definitions of the hull and the minimal generalization pattern for the extended versions given above, it follows trivially that the level of anonymity cannot be reduced below k by any given selection of colluding data recipients, if and only if the minimal generalization pattern of U obeys the criterion for at least the same k.
Regarding the identification of attackers, we assume the strict requirement that each pattern must be unique for every possible valid record within the expected value ranges, in order to provide stability in the non-colluding case. Then the set of patterns U is resistant to collusion, if and only if $a_i(x) \notin H(U \backslash \{a_i(x)\}), \forall i = 1, \dots, r, \forall x \in X_i$. The proof of this theorem is a slight adaption to the one outlined in Section 3.4.

Advantages: In general, this modification has two major advantages: (i) The number of possible fingerprints is typically higher than in the general approach, especially when looking at the relaxed version without incorporating inference attacks, and (ii) the impact on the overall data quality may be reduced by reducing the amount of unneeded generalization of records possessing frequent values in the quasi-identifiers.

Limitations: The main limitation of this approach lies in the additional performance costs, especially when considering the relaxed version: In this case it is not sufficient to check for obeying the k-anonymity criterion for each pattern, additionally it has to be checked for uniqueness. Furthermore, with the number of possible fingerprints increasing, also the performance requirements of the algorithm for providing "fair" fingerprints increases. This can be mitigated by only searching for a pre-defined number m of valid fingerprints with a higher threshold in difference regarding data quality, instead of searching for the best solution, i.e. the algorithm finishes when a suitable set of m fingerprints is found and does not care for finding a better solution.

Example: Table **??** shows an example of an original data set together with two fingerprinted samples. Any attribute containing actual payload was removed from the example in order to arrange it more clearly.

Original data			First Set		Second Set	
ID	sex	age	sex	age	sex	age
1	m	26	p	[26-50]	m	[26-75]
2	m	27	p	[26-50]	m	[26-75]
3	m	34	p	[26-50]	m	[26-75]
4	m	60	p	[51-75]	m	[26-75]
5	f	62	p	[51-75]	f	[26-75]
6	f	62	p	[51-75]	f	[26-75]

Table 3.14: Subtree generalization: Original data and two anonymized sets ($k = 2$)

Fingerprinting with sibling generalization

In order to evaluate the usability of sibling generalization for fingerprinting and to be able to compare it to the approach based on subtree generalization, we propose the following identification: For every parent node used as representative of all missing child nodes, we identify this node with the combination of all of it's missing child nodes, e.g. in Figure 3.15 (B), the node $[0-50]$ is identified with $[19-30] \cup [31-50]$, thus forming a kind of subtree generalization. When applying this mapping, it has to be kept in mind though that in the identification routine, the parent node and not an element of this combination is encountered.

For the resulting fingerprint, based on the identification with the subtree generalization, the same characteristics hold true with one exception: In case of two patterns a_1 and a_2, where w.l.o.g. a_1 is a parent node as representative for missing child nodes, while a_2 always uses the same node for generalization, it may not be possible to distinguish the fingerprints on the record level for every record. Thus it must always be taken into account that the selected patterns do not overlap, like the selection done in the subtree generalization case.

3.5.2 Suppression

In general, suppression can be seen as the most simple form of generating uniform records: Depending on the original data set and the parameter k, data that violates the

k-anonymity criterion is removed from the set until every equivalency class contains at least k records (or, of course, the data set is empty). Still, different definitions for data suppression exist, depending on the actual data removal process.

Row/Record Suppression: This form of suppression is the one that is dominant in the literature, usually in combination with generalization: Through the deletion of whole data records from the set, either k-anonymity is achieved directly, or more common, the amount of generalization required to achieve k-anonymity is reduced in order to provide more granularity in the remaining records.

Column/Attribute Suppression: In the case of attribute suppression, attributes are deleted as a whole from the recipients data sets in order to achieve anonymity, as well as the fingerprinting ability (or, of course, no more quasi-identifiers are left). While this approach is rather theoretical for most applications, we will nevertheless discuss its advantages and disadvantages.

Cell Suppression: Especially in cases where only a single attribute in one record might thwart fulfilling the k-anonymity criterion for a data set, only changing the content of this attribute in this specific record might be a valid solution, further on referred to as *cell suppression*. In the case of cell suppression, the content of an attribute is selected to a default value (e.g. "NA") in the selected records.

Suppression and Generalization: Of course, one vital strategy for generating k-anonymous datasets lies in applying suppression and generalization at once, i.e. by generating the equivalency classes through mild penalization, keeping almost all records and then treating the outliers using various forms of suppression. While this is a good strategy for generating k-anonymous data sets, the problem of good fingerprints is heavily depending on the actual form of the data, thus requiring detailed analysis of the data set and thus cannot be solved in this abstract form. Still, the approach will still be applicable, as it is solely a combination of the two singular approaches of generalization and suppression.

Fingerprinting with record suppression

In this section, we will discuss the adaption of the basic fingerprinting approach to record suppression.

Fingerprint generation: The generation of the fingerprints is done in two steps: (i) removal of all records that violate the k-anonymity criterion and (ii) generation of different data sets for distribution.
Since no further operation is following the suppression, it follows that all records that do not fall into a class possessing more than $k-1$ records need to be deleted and cannot be used for establishing the fingerprints. This is trivial, since contrary to generalization, the

further removal of records in order to generate the fingerprint does not generate new class-affiliations where they previously did not exist, i.e. classes cannot merge due to suppression. Let $T = \{t_i, i = 1, \dots m\}$ be the table holding the data and t_i denote the records. W.l.o.g. we only consider the quasi-identifiers when comparing records, i.e. two records are in the same equivalency class if and only if they have the same values in all quasi-identifiers. Then, in order to achieve anonymization, $\forall t_i \in T : |\{t_j | t_i = t_j, j = 1, \dots, n\}| \geq k$ must hold true.
The second step contains the generation of the actual fingerprints: For each data recipient, (at least) one equivalency class is defined as *marker class* that is removed entirely from the set for all other recipients, i.e. only data recipient U_i possess data of class m_i. This class forms the fingerprint inside the distributed data set.

Fingerprint identification: When distributing the fingerprinted data sets to the recipients, the data owner stores a list containing the assignment of marker classes to users. When encountering a leaked data set, the data leak is determined by searching for the records constituting the marker classes and thus identifying the defaulting user. The capability to detect leaks based on single records is thus not fulfilled anymore.

Diversity: The number of fingerprints that can be generated highly depends on the number of records in the table and the size of k: For each existing k-anonymity class after the first generation step one fingerprint can be generated. Furthermore, also combinations of single marker classes could be used for identifying one data recipient, e.g. user U_1 possessing the marker classes m_2 and m_3.

Stability: The stability of a single fingerprint largely depends on the completeness of the set: In case more than one marker class is missing, it is not possible to identify the defaulting recipient. This is especially critical when considering leakage of single records or small subsets.

Resilience against inference attacks: In general, colluding attackers may want to achieve two (separate) targets: (i) Reducing the anonymity level and (ii) disguising the partners participating in the collusion. Due to the construction of the fingerprints, it is not possible for any selection of colluding partners to reduce the anonymity-level: Even a collusion of all recipients would only result in the intermediate set after generation step one, which already obeys the k-anonymity criterion.
Regarding removal of the fingerprint, collusion of only two recipients allows them to single out their respective marker classes by calculating the symmetric difference between the two data sets (in case of using combinations this may not be 100 percent effective though): $m_1 = (D_1 \triangle D_2) \cap D_1$ and $m_2 = (D_1 \triangle D_2) \cap D_2$, where m_i denotes the marker class for the i-th data recipient, holding data set D_i. By singling out the marker classes, for the resulting data set $D := D_1 \cap D_2$ it cannot be decided, which partners participated in the collusion, since $D \cap D_i = D, \forall D_i$.

Advantages: One of the main advantages of this approach compared to those based on generalization lies in it's high performance in the generation part: The data owner only needs to calculate k-anonymity once, the subsequent fingerprinting is solely done by deleting classes. Actually achieving k-anonymity is much faster when using suppression compared to generalization: The data owner solely needs to group all records with respect to the QIs and delete any group (class) with less than k entries. Furthermore, the number of available fingerprints solely relies on the number of equivalency classes, not on the number of quasi-identifiers. Unknown dependencies between QIs also do not pose any threat to the fingerprint. Furthermore, data sets fingerprinted based on suppression can be easily extended with additional markers: The only prerequisite of the added data is that (i) it does not involve the marker classes and (ii) k-anonymity is preserved after addition to the original set.

Limitations: The main disadvantage of this approach lies in the fact that the amount of records that need to be removed in order to achieve k-anonymity solely by suppression can be very high. This is especially true for very detailed records or those consisting of many quasi-identifiers. Another main disadvantage of this approach, compared to those relying on generalization, is that the whole data set is needed for identifying the fingerprint, a single record is usually not sufficient (except the rare case of the record belonging to the marker class). Furthermore, the removal of the classes may add even more unwanted distortion to the data, which can have a severe impact on the distribution of key factors for the subsequent analysis. While generalization may reduce the data value by reducing the granularity of the information, the errors introduced by suppression can be far more subtle and simulate an accuracy that is not present.

Example: Table 3.15 shows an example of an original data set together with the basic anonymized set and two fingerprinted samples.

Original data			Anonymized Set		First Set		Second Set	
ID	sex	ZIP code	sex	ZIP code	sex	ZIP code	sex	ZIP code
1	m	1040	m	1040	m	1040	m	1040
2	f	1070	f	1070			f	1070
3	f	3022						
4	m	1040	m	1040	m	1040	m	1040
5	f	1070	f	1070			f	1070
6	m	1080						
7	f	1120	f	1120	f	1120		
8	f	1120	f	1120	f	1120		
9	f	1080						

Table 3.15: Original data and two anonymized sets ($k = 2$)

Fingerprinting with attribute suppression

Fingerprint generation & identification: Attribute Suppression can be modelled as a special form of the full-domain generalization scheme that was used in the original approach (see also [Iye02]): Instead of assigning different generalization levels for each quasi-identifier, only two exist: Full granularity, i.e. the granularity that was obtained in the original data set and no granularity at all, i.e. the level of granularity where all records possess the same value. While the latter one does not seem obvious, the quasi-identifier ”sex” often possesses exactly these two levels of granularity: High granularity (”female”, ”male”) and no granularity (”person”), see also Figure 3.3. The fingerprint can thus be obtained by the same methods as in the original approach. Since quasi-identifiers are either left in original granularity, or set to a default value (this is equivalent to removal), the only possible solutions lie in the corners of the lattice diagram shown in Figure 3.4. Due to the identification of attribute suppression with some instances of full-domain generalization, the validation procedure of the original approach can be adopted.

Diversity: The number of fingerprints that can be generated by this method is bounded by the number of quasi-identifier, still in most cases it will be considerably lower and for many applications this technique will not suffice in finding anonymized versions of the set that are still usable for further analysis.

Stability: Since, contrary to fingerprinting based on row suppression, the fingerprint is assigned on the structural level of the data, only a single record is needed in order to detect the leaking recipient. Due to the identification with a generalization based approach, the same conclusions concerning stability can be drawn.

Resilience against inference attacks: Let $q_i, i = 1, \ldots, k \leq n$ be the set of the k quasi-identifiers out of n in the original set that are not removed in all anonymized sets, i.e. for each $q_i, i = 1, \ldots, k$ there exists at least one anonymized set $D_j, j = 1, \ldots r$ where the attribute is not deleted. Then k-anonymity with respect to general collusion can be provided if and only if $D := \bigcup_{j=1}^{r} D_j$ obeys the k-anonymity criterion for the same k, which is a far stronger prerequisite than in the original approach. In order to provide stability of the fingerprint against collusions, every recipient must at least possess one attribute, the other recipients do not possess (see Section 3.4).
A more elegant proof for this corollary can be constructed by using the definition of the hull and the minimum generalization pattern in Section 3.4.

Advantages: The generation of this fingerprint is very fast and efficient, the detection of leaking recipients can be done on the record level, thus the validating party need not have access to the whole leaked set.

Limitations: The main limitation of this approach lies in the low number of possible anonymized data sets, still this could be of no concern in selected application domains.

Another major limitation lies in the resulting data loss due to the removal of whole attributes, thus adding the possibility of rendering the data set useless for further investigation.

Example: Table **??** shows an example of an original data set together with two fingerprinted samples. Instead of deleting the suppressed attributes, the generalization for the attribute "sex" with the values "m" and "f" to "p' and for "adult" with values "yes" and "no" to "x" is shown in order to demonstrate the identification of attribute suppression with generalization. Any attribute containing actual payload was removed from the example in order to arrange it more clearly.

Original data			First Set		Second Set	
ID	sex	adult	sex	adult	sex	adult
1	m	yes	p	yes	m	x
2	m	no	p	no	m	x
3	f	yes	p	yes	f	x
4	f	no	p	no	f	x

Table 3.16: Attribute Suppression: Original data and two anonymized sets ($k = 2$)

3.5.3 Fingerprinting with anatomization

The concept of *anatomization* as proposed by Xia and Tao in [XT06] does not modify the quasi-identifiers themselves, but tries to disassociate the relationship between the set QI of quasi-identifiers and the set ST of sensitive attributes. More precisely, QI and ST are stored in separate tables while sharing an attribute *group-ID* that links the records in an ambiguous way, thus distorting the direct connection. The subsequently released fingerprinted data sets consist of two separate tables QIT, holding the quasi-identifiers of the original table in an undistortioned way together with a group-ID, and ST, holding the sum of encountered values for the sensitive attributes per group.

Fingerprint generation: Anatomization based fingerprinting of data works as follows: For a given k and m, where k denotes the anonymity level and m the number of fingerprints, the original data is grouped into m different tables $QID_j, j = 1, \ldots, m$: In each table the data is partitioned and/or generalized into groups in a way that at most $\frac{1}{k}$ records contain the same values in the sensitive attributes per group. Furthermore, the m tables need to possess different groups in order to generate the fingerprint. The group-ID is added to each record in QID_j. Next, for each of the QID-tables, a copy of the original table is generated, where the columns holding the sensitive attributes are removed and a new attribute "group-ID" holding the respective group-ID derived from the table QID_j is added. These tables $QIT_j, j = 1, \ldots, m$ form the first part of each fingerprinted data set. The second part $ST_j, j = 1, \ldots, m$ is formed by selecting the count of each representation of the sensitive attributes in the respective QID-table and adding the sensitive attributes as well as the "group-ID". The data owner needs to store both tables for each data recipient in order to unambiguously identify the defaulting data recipient in the approach outlined here.

Fingerprint identification: In general, both tables QIT and ST are needed as a whole in order to identify the data leak: The two tables are compared to the ones stored in the owner's database and the defaulting recipient is identified subsequently. It is possible to construct data sets, where only one table, or even parts of one table are sufficient for identification. Since the exact form of the tables highly depends on the actual data and it's distribution with respect to the sensitive attributes, as well as the chosen grouping strategy, it is not possible to further generalize these examples.

Diversity: The number of fingerprints that can be generated by this method is bounded by the amount of different intermediate tables QID_j that can be constructed. These groupings can be done rather arbitrarily, the only additional prerequisite is that the sizes of the groups need to be different with respect to the values of the sensitive attributes.

Stability: Compared to the fingerprints based on either generalization or suppression, this approach is neither working on the purely structural, nor solely on the record level - while in general both tables are needed as a whole in order to guarantee the identification of the data leak, in reality, assuming a notable difference in the selection and construction of the groups, enough distortion will be introduced to alow the detection based on much smaller excerpts. Also the diversity of the tables involved is largely depending on the selection of the groups.

Advantages: The major advantage of using anatomization for fingerprinting lies in the fact that the data in both, QIT and ST remains largely unchanged, only the links between these tables are distorted. This is especially important for medical investigations that involve aggregations or need an undistorted distribution of certain attributes in the data sample.

Limitations: The main drawback of this approach lies in the complexity of finding suitable groupings, especially for large factors of k, since each group is only allowed to hold $\frac{1}{k}$ records with the same values for the sensitive attributes. Furthermore, identification is rather complicated and the stability of the resulting fingerprint largely depends on the original data.

Example: Following we show a very reduced example as a demonstration of the anatomization based approach for $k = 2$. Table **??** shows an example of an original data set, while Table **??** contains two different groupings. In this example, the attributes "age" and "sex" form the set of quasi-identifiers, while "disease" is the sensitive attribute.

Similar to the previous examples, the identifer "ID" is removed from every anonymized set in the first step. Afterwards, two different groupings QID are generated, both

ID	Age	Sex	Disease
1	42	m	Obesity
2	43	m	Cancer
3	44	m	Obesity
4	44	m	Cancer
5	50	m	Flu
6	55	m	Flu
7	55	m	Cancer
8	28	f	Flu
9	33	f	HIV
10	34	f	HIV
11	37	f	Flu

Table 3.17: Anatomization based fingerprinting: Original data

obeying the prerequisite that for each value for the sensitive attribute only $\frac{1}{k}$ records are encountered. The groups are numbered and the respective group-ID ("gID") is added to each record in the $QIDs$.

First QID				Second QID			
Age	Sex	Disease	gID	Age	Sex	Disease	gID
[40 – 55]	m	Obesity	1	[40 – 50]	m	Obesity	1
[40 – 55]	m	Cancer	1	[40 – 50]	m	Cancer	1
[40 – 55]	m	Obesity	1	[40 – 50]	m	Obesity	1
[40 – 55]	m	Cancer	1	[40 – 50]	m	Cancer	1
[40 – 55]	m	Flu	1	[40 – 50]	m	Flu	1
[40 – 55]	m	Flu	1	[51 – 60]	m	Flu	2
[40 – 55]	m	Cancer	1	[51 – 60]	m	Cancer	2
[25 – 40]	f	Flu	2	[25 – 40]	f	Flu	3
[25 – 40]	f	HIV	2	[25 – 40]	f	HIV	3
[25 – 40]	f	HIV	2	[25 – 40]	f	HIV	3
[25 – 40]	f	Flu	2	[25 – 40]	f	Flu	3

Table 3.18: Anatomization based fingerprinting: Two QIDs

Based on the two QID-tables, two anatomized and fingerprinted data sets are created: The column "disease" in the original data tables is therefore replaced with the new attribute "gID" and the corresponding aggregation tables ST containing the sum of each sensitive attribute grouped by "gID" and "disease" are created. The two fingerprinted data sets are then distributed to the respective data recipients. They are shown in Table **??**, "Ct" denotes the attribute holding the count.

First data set						Second data set					
QIT			ST			QIT			ST		
Age	Sex	gID	gID	Disease	Ct	Age	Sex	gID	gID	Disease	Ct
42	m	1	1	Obesity	2	42	m	1	1	Obesity	2
43	m	1	1	Cancer	3	43	m	1	1	Cancer	2
44	m	1	1	Flu	2	44	m	1	1	Flu	1
44	m	1	2	Flu	2	44	m	1	2	Flu	1
50	m	1	2	HIV	2	51	m	1	2	Cancer	1
55	m	1				55	m	2	3	Flu	2
55	m	1				55	m	2	3	HIV	2
28	f	2				28	f	3			
33	f	2				33	f	3			
34	f	2				34	f	3			
37	f	2				37	f	3			

Table 3.19: Anatomization based fingerprinting: Two sets (QIT, ST)

3.6 Generalization to advanced anonymity definitions

In this section, we will present the most important anonymity models extending the basic k-anonymity concept. While our basic approach is based on the original k-anonymity concept, it can be easily extended to the following more advanced definitions for anonymity. Thus, we will provide an overview on the extensions of the original concept that we deem most important, either based on the impact their respective publications had in the scientific community with respect to citations, or due to their practical applicability in real life scenarios. This section, as well as Section 3.5 are under submission at the time of submitting this thesis.

3.6.1 Stronger notions of anonymity

While a plethora of different modifications of the original k-anonymity criterion exist, we will provide some background on the most popular or (in our opinion) useful approaches.

(X, Y)-anonymity: Many papers describing the concept of k-anonymity miss out on an important assumption that was described in the original work [Sam01]: It is assumed that each physical person is only present once in the whole data set, i.e. there is only one record for each person. Obviously this is rather untypical for many areas of application domains like billing or medical data recording, as there are several records holding information on the same person. Since the quasi identifiers might not change for this person, this may result in equivalency classes holding less than k records from different persons, when applying the standard version of k-anonymity. This relation between identifiers and quasi identifiers can be generalized a bit more, leading us to the following definition:

Definition 3.6.1 ((X,Y)-anonymity)**.** *Let X and Y be sets of attributes with $X \cap Y = \emptyset$ and $a_Y(x), x \in X$ the number of co-occuring values of x in Y. Then a table T obeys the (X,Y)-citerion for a specified k if $A_Y(X) := min\{a_Y(x) | \forall x \in X\} \geq k$.*

(X,Y)-anonymity can be used to define *k-anonymity with respect to persons* by identifying X with the set QI of the quasi-identifiers and Y with I, the set of direct identifiers.

l-diversity: The original k-anonymity criterion, even when considering the extension to (X,Y)-anonymity, possesses another major limitation, which can be best outlined by the example given in Table 3.20: The original data set on the left side is anonymized with $k = 3$, still the distribution of the sensitive attribute "disease" over the equivalency classes is not optimal: While the standard approach for achieving k-anonymity makes the data sets of the three male patients indistinguishable with respect to the quasi-identifiers, they all fall into the same equivalency class. Since the value for the sensitive attribute is the same for all three of them, the anonymization effort is negated regarding protecting them against retrieval of their medical condition.

Original data				k-anonymity		
ID	sex	birthdate	disease	sex	birthdate	disease
1	m	01.09.1974	cancer	M	1974	cancer
2	m	20.05.1974	cancer	M	1974	cancer
3	f	10.10.1972	anorexia	F	1972	anorexia
4	f	31.03.1972	typhlitis	F	1972	typhlitis
5	m	10.12.1974	cancer	M	1974	cancer
6	f	12.12.1973	flue	F	1972	flu

Table 3.20: Original data and an anonymized set ($k = 3$)

The main purpose of l-diversity as proposed in [MKGV07] lies in the generation of equivalency classes, where the sensitive attributes are well-represented in each class. This opens up new questions on the definition of the term *well-represented*, which has been discussed extensively in the literature, a good overview can be found in [FWCY10]. The most basic definition of the term is the *distinct l-diversity model* proposed in [TC10]:

Definition 3.6.2 (Distinct l-diversity)**.** *Let a be a generalization pattern. Then a obeys the* distinct l-diversity *criterion for a given l if and only if each equivalency class defined by the pattern contains at least l different values for any sensitive attribute.*

Since distinct l-diversity is susceptible to some more advanced probabilistic inference attacks, other definitions for well-represented sensitive attributes have been proposed like *entropy l-diversity* [MKGV07] or recursive (c,l)-diversity [SLW11] being the most prominent amongst others.

Definition 3.6.3 (Recursive (c, l)-diversity)**.** *Let a be a generalization pattern. Then a obeys the* recursive (c, l)-diversity *criterion for given parameters c and l if and only if each equivalency class defined by the pattern contains at least $r \geq l$ different values for any sensitive attribute and the following additional criterion holds true for each equivalency class:*

$$f_1 < c \cdot \sum_{i=l}^{r}$$

where the r values are sorted in descending order with respect to their occurrence, i.e. f_1 is the value that occurs the most.

t-closeness: The concept of t-closeness was developed as an improvement to k-anonymity and l-diversity in order to mitigate so-called *skewness attacks*. When the distribution of sensitive values in an equivalency class differs from the distribution in the overall table, this again yields information on a specific person attributed to this equivalency class. The basic principle behind t-closeness lies in choosing the equivalence classes in a way that the distribution of any sensitive attribute in any class is similar to its distribution in the original table [LLV07]. More precisely, let D_{all} be the distribution of a sensitive attribute in the original table holding all data records and D_i be the distribution of that same attribute in the $i-$th equivalence class, for all classes $i = 1, \ldots, n$ as defined in the k-anonymity paradigm. Then these equivalence classes are obeying the t-closeness criteria for a given value t if and only if the distance between D_{all} and D_i is at most $t, \forall i = 1, \ldots, n$. However, the main questions is, how to measure this distance between equivalence classes, while including the semantic distance between values. One solution is the so-called *Earth Mover Distance* (EMD) as defined in [LLV07]. The t-closeness paradigm has some drawbacks though: (i) The first and most important drawback considers the impact of enforcing t-closeness on the dataset: When assuming t-closeness, the sensitive values will have the same distribution in all equivalence classes with respect to the quasi identifiers, thus having a significant impact on the correlation between these attributes and the QIs. Since a lot of research e.g. in medical sciences is actually targeting at such correlations, t-closeness remains unusable in these cases. (ii) Another drawback is that t-closeness lacks the ability to specify separate protection levels for each quasi identifier. Furthermore, (iii) there still exist special attribute linkage attacks on t-closeness when utilizing it on sensitive numerical attributes as shown in [LTX08]. (iv) Last, but not least, it is often impossible to enforce t-closeness without reducing the value of the data too much for any meaningful analysis due to requiring too large classes and too much generalization. This is especially the case when correlations (or causalities) between QIs and payload data exist.

3.6.2 Extending the fingerprinting approach

In this section we will discuss the implications of selecting a more advanced anonymity definition on the fingerprinting approach outlined in Sections 3.3 and 3.4.

Fingerprinting based on (X, Y)-anonymity

Fingerprint generation & identification: Like in the original approach, the generation of the (X, Y)-anonymity based fingerprints is based on the selection of patterns in the lattice diagram. In the original algorithm, as outline in Section 3.3, only the calculation of the k-anonymity criterion needs to be changed to calculating the (X, Y)-criterion for the same value of k. Furthermore, the records in the original data table need to possess an identifier denoting the user the record belongs to, further on referred to as "ID" (of course this can also be a set of identifiers as noted in the original definition of (X, Y)-anonymity). Since, like in the case of k-anonymity, for every pattern following the (X, Y)-anonymity criterion, all patterns in the hull (see Section 3.4) of the pattern follow the criterion for at least the same k, the original algorithm can be used with respect to this prerequisite.

Diversity: The number of possible fingerprints can be smaller than in the original approach, depending on the actual original data: Since a data set anonymized with a pattern obeying (X, Y)-anonymity for a given value k is also obeying the k-anonymity criterion for at least the same k, the latter forms an upper bound for the number of possible fingerprints. Furthermore, as outlined in the example above, not every pattern obeying k-anonymity is necessarily obeying (X, Y)-anonymity. For the single pattern case, the number of possible fingerprints is bound by the number of patterns obeying the (X, Y)-criterion, still, in the case of collusion between data recipients, the theoretical upper bound for the number of fingerprints providing the possibility for identification of the defaulting parties (with respect to the prerequisites given in Section 3.4) is equal to the number of quasi-identifiers (see below).

Stability: Let $A = \{a_1, \ldots, a_r\}$ be a set of patterns obeying the (X, Y)-anonymity criterion and $\bar{a}$ be the *minimal generalization pattern* of A (see Section 3.4 for the definition of the minimal generalization pattern). Then, following the original approach, the anonymity of the data is secure against every possible selection of colluding recipients, if and only if $\bar{a}$ obeys the (X, Y)-anonymity criterion for the same k.

Resilience against inference attacks: For the identification of colluding attackers, nothing changes compared to the original approach outlined in Section 3.4, since the stricter criteria on the selection of the patterns just reduce the number of patterns allowed in the generation process. Furthermore, every pattern inside the hull also follows the same criteria. The proof for this claim is trivial: Due to the criteria, every class in the set with finer granularity contains at least records of k different users. In case of further generalization of a pattern, classes are merged, but never split, thus every pattern resulting from generalization again possesses at least records of k different users. Following the argument given in Section 3.4, the number of generalization patterns that can be used for fingerprinting while providing identification of the colluding attackers is bound by the number of quasi-identifiers. Still, depending on the actual data, the number of identifiable patterns that thwart de-anonymization in the case of collusion may be much smaller.

Fingerprinting based on l-diversity & t-closeness

Generalization is the common tool for achieving l-diversity, thus this concept can be incorporated into the fingerprinting approach. Since there is no general rule for defining the l-diversity criterion, it is not possible to give a general quantification and qualification of the resulting fingerprints. In order to be able to give more details, at least one definition of the term *well-represented* for the sensitive values has to be selected. Following we will discuss an approach based on the classical concept of distinct l-diversity.

Fingerprint generation & identification: Similar to the original approach, the generalization of a pattern obeying the distinct l-diversity criterion again yields patterns that obey the same criterion for at least the same number l. This is trivial, since any generalization may only merge classes obeying the criterion that were separated before, i.e. if two classes with each possessing l different values for a sensitive attribute are merged, the result will again possess at least l different values. Furthermore, it must be noted that per definition any pattern obeying the distinct l-diversity criterion automatically obeys the k-anonymity criterion for $k = l$, since in order to contain at least l different values for the sensitive attributes per class, it is mandatory that each class contains at least l records (the issue of several records belonging to the same person as outlined in Section 3.6.1 must still be taken into account though). Thus it can be noted that the set of generalization patterns obeying the distinct l-diversity criterion forms a subset of the set of patterns providing k-anonymity with $k = l$. Based on this result, the only part that needs to be changed in order to provide a distinct l-diversity based fingerprint lies in the selection of the valid patterns: In addition to checking for the size of each resulting equivalency class, the classes need to be checked for the number of different values in the sensitive attributes. Of course, k-anonymity and l-diversity can be combined for any $k > l$ (for $k \leq l$ this follows trivially) by checking the sizes of the classes with respect to at least k records with at least l distinct values in the sensitive attributes.

Diversity & stability: Similar to (X, Y)-anonymity based fingerprinting, the number of possible fingerprints can be bound by the number of patterns obeying the k-anonymity criterion for $k = l$. Furthermore, since not every pattern obeying the latter criterion also obeys l-diversity, the number of fingerprints can be considerably smaller, mainly depending on the actual data and is given by the number of patterns obeying the l-diversity criterion. In the case of collusion attacks, the theoretical upper bound for the number of fingerprints providing the possibility for identification of the defaulting parties is equal to the number of quasi-identifiers. Regarding stability, since the patterns obeying l-diversity form a subset of the patterns obeying k-anonymity for $k = l$, the results of the original approach stay valid.

Resilience against inference attacks: Due to the characteristic of patterns obeying l-diversity being a subset of the ones providing k-anonymity ($k = l$), the results from the original approach regarding collusion of data recipients can be directly translated to distinct l-diversity:

1. Let $A = \{a_1, \ldots, a_r\}$ be a set of generalization patterns obeying the distinct l-diversity criterion. Then the collusion of an arbitrary selection of data recipients still obeys the criterion if and only if this holds true for the minimal generalization pattern $\bar{a}$ of A.

2. For the identification of the colluding recipients nothing changes compared to the original approach. This can be easily derived from the results regarding stability in Section 3.4.

Advantages: The major advantage of fingerprinting with l-diversity does not lie in the fingerprint per se, but in the better protection of end user privacy, thus this extension does have its advantages.

Limitations: The major issue, besides the problem of reduced diversity and thus a limited number of data recipients, lies in the typical problems surrounding l-diversity, more precisely the practical applicability of the concept with respect to real data and the resulting reduction in data quality: In addition to fulfilling the k-anonymity criterion, which aims at equalizing enough records with respect to the QIs to generate equivalency classes holding at least k records each, l-diversity has the opposite target with respect to non-QI payload attributes by introducing additional diversity into these classes. This is especially problematic in cases, where the analysis question in place tries to find correlations or even causalities between QIs and payload attributes. To even further add to the problem, this is typically always the case when using a proper privacy by design [Lan01] approach, as data minimization is a key concept [C$^+$09] and data (e.g. a QI) that is not required in the analysis must not be included into the data set at all, i.e. all quasi identifiers present in the data set need to be interesting for the analysis in a privacy-aware approach.

Using recursive (c, l)-diversity: Since recursive (c, l)-diversity is also monotonous in nature, i.e. generalization of a pattern obeying the criterion cannot result in a pattern not obeying it anymore, the notion of distinctive l-diversity can be exchanged through (c, l)-diversity in the paragraphs above by simply changing the function calculating the l-diversity of equivalency classes defined by the generalization strategies

Non-monotonous definitions of l-diversity: While no non-monotonous definition for the l-diversity criterion is known to us, this can be done theoretically, as the criterion of "well-distribution" of the sensitive values does not indicate monotonicity. Our fingerprinting approach could still be used for such definitions, with one exception in the algorithm: It would no longer be possible to skip the calculation of nodes below a node not fulfilling the criterion, as this less general node with higher granularity could suddenly obey the criterion and thus form a valid fingerprint. A non-monotonous definition would also result in a much more problematic issue: Any attacker holding a data set obeying the criterion that loses l-diversity through simple generalization would be able to do so and thus remove the criterion from his/her data set. Still, as we were not able to find a case where non-monotonicity has any advantages and also did not find any examples for such definitions in the literature, we did not put much focus on this issue.

Fingerprinting with t-closeness: Fingerprinting with t-closeness is extremely similar to fingerprinting based on a l-diversity paradigm, the major difference is that the requirements regarding the equivalency classes generated through the application of the k-anonymity criterion are even more strict with respect to the diversity required in said classes, as they not only need to fulfill requirements regarding their internal distribution, but this internal distribution itself needs to follow the overall distribution. To the best of our knowledge we are not aware of any practical implementation for t-closeness due to this point, especially since the limitations of l-diversity are typically multiplied by the added requirement regarding the distribution. Still, with respect to the fingerprinting approach, the same can be said for t-closeness based fingerprinting as for the approach adapted to l-diversity, including the simple adaptation of the algorithm by simply exchanging the check routines.

Using other algorithms

We built the algorithm in our approach based on the algorithm proposed by El Emam, since it is an optimal and deterministic one, i.e. it not only constructs the optimal solution, but also all possible solutions with respect to the predefined generalization levels in case of full-domain generalization. It must be noted though that the algorithm requires adaption for subtree generalization and other anonymity criteria besides simple k-anonymity.

Still, many other algorithms like e.g. Mondrian [LDR06], Incognito [LDR05] or greedy approaches [MT11, CT08] have been proposed for calculating k-anonymity and derived anonymization models, especially focussing on efficiency. Furthermore, adopting the notions of more complex generalization methods typically also requires other algorithms in order to be practically feasible, as the lattice of all possible solutions not only becomes much more fine-grained, ist also gains a lot more dimensions in order to cater for different subtree selections. Many of these algorithms are not deterministic, but heuristic approaches that only aim at delivering a valid, but not the best or all solutions.

In order to make such algorithms useful for our approach, they need to be run often enough in order to generate the required amount of solutions that are then within clustering range with respect to the clustering threshold. If these cannot be constructed, it must be decided, whether a cluster simply hasn't been found due to the heuristic nature of the algorithm, or if there simply does not exist a suitable cluster with the given data and side parameters, and a relaxation of these side parameters is required. This is a problem for all heuristic algorithms or algorithms that do not construct all possible solutions. Still, it might be useful in some instances to use other algorithms for the determination, whether the anonymity criterion was fulfilled, especially for more complicated strategies like sibling generalization, where constructing the lattice diagram for all solutions is typically infeasible.

CHAPTER 4

Detecting manipulations in databases using $\mathcal{B}^+$-Trees

4.1 Introduction

Modern database systems have a very high traceability of modifications because of intense logging mechanisms [Fin08]. Database storage engines such as InnoDB for MySQL store every single manipulation statement in their log files, thus making hidden manipulation of database records a difficult task [FHMW10b]. However, if traces of manipulations are removed from log files, traditional database forensic techniques are ineffective, because they are limited to mainly data file and log file analysis. A malicious database administrator, who has full access to the database and the log files, can effectively cover tracks of manipulations.

In this chapter, we introduce a novel forensic technique on the $\mathcal{B}^+$-Tree level of the database storage engine based on the ideas described in [MW]. The concept of our approach is a sanity check for database content by comparing a stated insertion sequence to the data stored in the tablespace, the transaction logs and the $\mathcal{B}^+$-Tree structure of the indexes.

While a forensic analysis of a $\mathcal{B}^+$-Tree is not likely to contain precise information on which data was modified at what exact time, it can be used very effectively to prove the violation of security and compliance policies. Compliance in a database is usually ensured by access control, stored procedures, triggers, and audit logs. A database user might only have permissions to insert into a table and not delete. In addition, the user might not have permissions to insert directly into the table but only to execute a stored procedure. This procedure would create additional audit logs, ensure that e.g. only increasing time stamps are used and enforce other integrity constraints.

If the current $\mathcal{B}^+$-Tree cannot be derived from the insertion sequence record in the log files or does not match the insertion policy (e.g. strict ordered insert only), a manipulation of the database is very likely. $\mathcal{B}^+$-Tree forensic cannot replace traditional database forensic methods, because evidential quality varies greatly on a case-by-case basis. However, our approach can substantiate evidence collected by other methods.

A second area for its application could be file system forensics. Many of today's files systems such as NTFS [RF04], ReiserFS [Rei04], and BtrFS [RBM13] use $\mathcal{B}^+$-Trees for their internal organization of stored files. Previous forensic methods on the file system layer [Car05, SPS07, BF08, Eck05, EJ05] did not analyze the structure of the underlying $\mathcal{B}^+$-Tree, still Koruga et al. [KB10] reconstructed the $\mathcal{B}$-Trees of file systems to recover deleted files.

Our main contributions are to show that the structure of a database's $\mathcal{B}^+$-Tree can be a very important characteristic of the insert order of the elements inside a database table and to show that the analysis of a $\mathcal{B}^+$-Tree allows to draw forensic conclusions to disprove an input order extracted from log files. Still, as we will further discuss in the section on limitations (Section 4.4), the promise of providing evidence in case of manipulations can only be delivered in certain circumstances. While this poses a serious limitation with respect to detection capabilities, it does have benefits for data protection: As outlined by the GDPR, data deletion is a requirement for privacy protection, thus, the limitations proven in this chapter also reduce the problem introduced by already deleted information being leaked by the structure of a typical database table and thus reduce the problem of unintentional data recovery, which has been pointed out to be a major problem from a legal perspective [VKL18].

4.2 Background

4.2.1 $\mathcal{B}$-Trees and $\mathcal{B}^+$-Trees

In the following subsection we provide the definitions of $\mathcal{B}$-Trees [BM72] and $\mathcal{B}^+$-Trees [Com79]. While it must be noted that implementations of these data structures often deviate a bit from the original definitions, these deviations are typically (and with respect to InnoDB where we evaluated our approaches most definitely) not big enough to thwart our basic concept, but merely require some additional changes in the implementation. The most prominent difference typically is the introduction of additional pointers between adjacent leaves in $\mathcal{B}^+$-Trees, thus allowing them to be used as a sorted list.

Definition 4.2.1. *We define a $\mathcal{B}$-Tree of order b as a balanced tree with the following properties:*

- *Every node except the root node contains at least $\frac{b}{2}$ and at most b elements.*
- *The root node contains b elements at most.*
- *An inner node with x elements has got $x+1$ child nodes.*

- *All leaf nodes lie on the same level.*
- *All elements inside a leaf are sorted.*
- *Adding of elements:*
  1. *The tree is searched for the leaf the element should be placed in (search works like in a binary tree).*
  2. *In case this leaf node contains less than b elements, the new element is simply added.*
  3. *In case this leaf node contains b elements, the leaf is split into two leaves, the middle element is inserted as a new key into the parent node. In case the parent node contains $b+1$ keys after this operation, the parent node is split too. This is done iteratively until either a parent node contains less than $b+1$ elements, or a new root is formed. In case $b/2$ is an even number, we define that the element with number $b/2+1$ is propagated to the parent node.*

Example 4.2.2. *Be B a $\mathcal{B}$-Tree containing the elements $1, 2, 3, 4, 5$, inserted in ascending order. See Figure 4.1 for the resulting $\mathcal{B}$-Tree.*

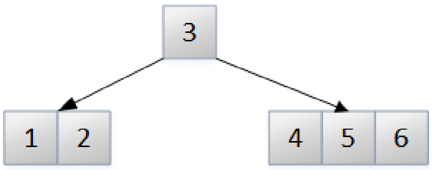

Figure 4.1: Structure of the resulting $\mathcal{B}$-Tree

Definition 4.2.3. *A $\mathcal{B}^+$-Tree of order b is defined like a $\mathcal{B}$-Tree of order b, but all keys reside in the leaf nodes, i.e. all elements in inner nodes are only for referencing purposes, the pointers to the actual data are all stored in the leaves.*

It has to be defined from the start, if the parent node contains the topping elements of the lower child nodes, or the bottoming elements of the higher child nodes, i.e. it has to be defined in case of splitting a node, whether a copy of the highest element of the child node containing the lower elements is propagated to become a key in the parent node, or the lowest element of the node containing the higher elements.

Example 4.2.4. *Figure 4.2 shows the $\mathcal{B}^+$-Tree for the set from the previous example with element number 6 added.*

4.2.2 Compositions of $n \in \mathbb{N}$

Some of the problems tackled in Section 4.6.1 are solved by reducing them to an equivalent problem concerning the composition of a number $n \in \mathbb{N}^*$, thus we will provide some

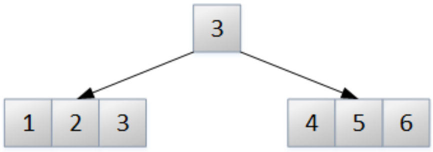

Figure 4.2: Structure of the resulting $\mathcal{B}^+$-Tree

background in this section. For further information on the proofs we would like to refer the reader to [Opd10].

Definition 4.2.5. *A tuple* $(a_1, \ldots, a_k), a_i \in \mathbb{N}^*$ *is called a* composition *of* $n \in \mathbb{N}^*$, *when* $a_1 + \cdots + a_k = n$. *Contrary to the partitions of a number* n, *two compositions* (a_i) *and* (b_i) *are only equivalent if* $a_i = b_i, \forall i = 1, \ldots, k$ *holds true (i.e. order is important).*

Example 4.2.6. *The compositions of the number 3 are:* $(1, 1, 1), (1, 2), (2, 1)$ *and* (3).

Theorem 4.2.7. *The number* $c(n)$ *of all compositions of a number* $n \in \mathbb{N}^*$ *is calculated by* $c(n) = 2^{n-1}$ *[Opd10].*

Proof. We write n as a sum of ones and exchange the "+"-signs by $\circ$:

$$n = \overbrace{1 + \ldots + 1}^{n} = \overbrace{1 \circ 1 \circ 1 \ldots 1 \circ 1 \circ 1}^{n}$$

Now we have $2^{(n-1)}$ different possibilities of replacing $\circ$ by +-signs, thus resulting in $2^{(n-1)}$ possible compositions.

□

Another case is when setting boundaries on the elements of the composition. According to Opdyke the closed form for calculating the bounded compositions is denoted by the following recursion. The proof of this equation can be found in [Opd10]:

Theorem 4.2.8.

$$c(n, a, b) = I(a \leq n \leq b) + \sum_{i=max[1,(n-b)]}^{(n-a)} c(i, a, b) \tag{4.1}$$

with the indicator function I *being defined as* $I(a \leq n \leq b) = 1$ *for* $a \leq n \leq b$ *and* $I(a \leq n \leq b) = 0$ *for all other values of* n.

The next theorem (also see [Opd10]) gives us a valuable result on compositions that limit the size of one particle. We will use this result in our proof that we are able to construct a tree for every given valid partitioning of our tree leaves in Section 4.6.1.

Theorem 4.2.9. *Be $b, n \in \mathbb{N}, b/2 < n$. $(n-1)$ can be written as a composition of numbers $a_i \in [\lfloor \frac{b}{2} \rfloor; b], i = 1 \dots k$. Then it holds true that there are numbers $a'_i \in [\lfloor \frac{b}{2} \rfloor; b], i = 1 \dots j$ with $\sum_i a'_i = n$, i.e. n can be written as composition of numbers in $[\lfloor \frac{b}{2} \rfloor; b]$ as well.*

Proof. $n - 1 = \sum_{i=1}^{k} a_i, a_i \in [\lfloor \frac{b}{2} \rfloor; b]$. Now there are two cases:

1. $\exists a_i : a_i < b \Rightarrow$ With $a'_i := a_i + 1 : \sum_{j \neq i} a_j + a'_i = n$
2. $\nexists a_i : a_i < b \Rightarrow (n-1) = \sum_{i=1}^{k} a_i = k \cdot b = (k-1) \cdot b + \lfloor \frac{b}{2} \rfloor + \lceil \frac{b}{2} \rceil \Rightarrow n = (k-1) \cdot b + \lfloor \frac{b}{2} \rfloor + \lceil \frac{b}{2} \rceil + 1 = \underbrace{b + \ldots + b}_{k-1 \text{ times}} + (\lceil \frac{b}{2} \rceil) + (\lfloor \frac{b}{2} \rfloor + 1) = \sum_{i=1}^{k+1} b_i$ with $b_1 \dots b_{k-1} = b, b_k = \lceil \frac{b}{2} \rceil$ and $b_{k+1} = \lfloor \frac{b}{2} \rfloor + 1$.

The starting point for our induction is $n = \frac{b}{2}$ for which this is trivial.

□

This result can be further generalized into the following theorem:

Corollary 4.2.10. *Be $a, b, n \in \mathbb{N}, a < b, a < n$ and b even. $(n-1)$ can be written as composition of numbers $a_i \in [a; b]$. Then it holds true that*

1. *if $a \leq \frac{b}{2}$, n can be written as composition of numbers in $[a; b]$.*
2. *if $a > \frac{b}{2}$ there always exists a $n \in \mathbb{N}$ so that n cannot be written as a composition of numbers between $[\frac{b}{2}; b]$.*

Proof. We split this proof into two parts and utilize the results on compositions from [Opd10]:

1. The first part trivially follows from the theorem before.
2. For the second part we simply look at $n := b+1$: Since $n > b$ it cannot be composed of one element, so at least two a_1 and a_2 are needed. The lowest value allowed for $a_1 = a$, the lowest possible a in our theorem is $\frac{b}{2} + 1$. But if $a_1 = \frac{b}{2} + 1$ (or higher), then $a_2 = \frac{b}{2}$ (or lower) which results in a contradiction to our assumption, thus leading us to the conclusion drawn in the theorem.

□

4.2.3 Index tree steganography

Steganography (see e.g. [JJ98]), compared to cryptography, tries to hide the existence of information altogether, i.e. where secrecy, especially regarding the algorithm (see Kerckhoffs principle, e.g. in [Pet11]) is a non-issue in modern cryptography and the existence of information exchange is typically not considered to be an issue, steganography aims at hiding the existence of information (exchange) altogether. In historic times this has been facilitated by a number of techniques like invisible inks, microdots [JDJ01] (even by introducing these markers into DNA [CRB99]), and so on. Steganography with file systems and other large structured data vaults often use the term “slack space”, which typically defines the problem that most file systems need to reserve whole clusters when writing files to the disc [EJ05]. In case the data stored inside the file does not fill up the whole allocated space, the remaining allocated space is lost. Still, direct addressing can be used to either retrieve non-overwritten old data from this slack space within a forensic investigation [KCGD06], or even for using it to store data [Gar09]. In our work [FKW15], we have extended the notion of slack space to MySQL, more precisely, using the InnoDB storage engine in order to hide data. While other authors [PO12] had already provided ideas for hiding data inside databases, as they offer multiple advantages like high volatility and typically large size, we used the index tree in order to generate this free space instead of hiding data inside other records. The basic idea of the concept is that all data inside InnoDB is structured along the primary key (the $\mathcal{B}^+$-Tree of a table) which is therefore always present and constitutes the primary index of the table. Furthermore, the leaf nodes are, as in the original works on $\mathcal{B}^+$-Trees, structured as lists, while the pages inside the tree leaves are interconnected with each other. On deletion, the database, like many file systems, does not overwrite the data, but simply removes the key and the address from the primary index, as well as updates all secondary indices accordingly [FHMW10b]. The free page address is then added to the garbage collection for future re-use (see Figure 4.3).

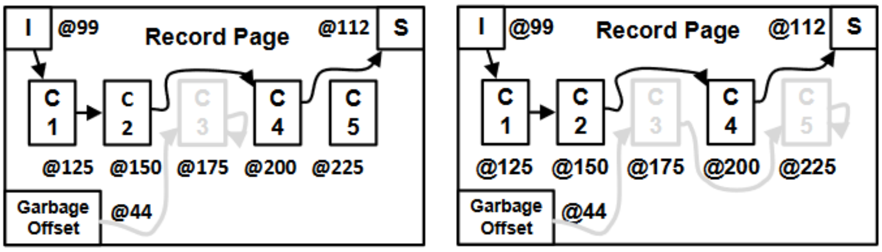

Figure 4.3: One and two deleted records in a $\mathcal{B}^+$-Tree leaf (modified and taken from [FKW15])

Thus, the only modification to the database source required to generate large amounts of slack space, is hindering the garbage collection from accessing the space we want to utilize for data storage. This can be done by simple modification of the deletion function [FKW15], which will result in no space being added to the garbage collection with the drawback of potentially opening up a problematic data leak. The major advantage of slack space generated through this method over other methods of database steganography is the complete loss of any control of the database and especially the database administrator over this space. Neither can the database access this space, nor can it be overwritten by database reorganization methods, no board tools allow access short of using file carving [PM09] to retrieve and change the information in the underlying database storage files, which is a tedious, dangerous and noticeable task. Thus, in order to actually use this space, either the user him/herself needs to use file carving, or, specific additional read and write functions need to be implemented into the database. More details on how to exactly construct such a slack space, especially considering also retrieval through secondary indexes, as well as page dictionaries, can be found in [FKW15].

4.3 Database forensics using the index tree

4.3.1 Notation and general assumptions

The following notations and general assumptions are used throughout this chapter:

- $\mathbb{N}$ denotes the natural numbers *including* zero, $\mathbb{N}^*$ the natural numbers *excluding* zero.
- b denotes the maximum number of keys in a node of a given $\mathcal{B}^+$-Tree. We call b the *order* of the tree.
- All nodes of a $\mathcal{B}^+$-Tree except for the root node have a keys with $\lfloor \frac{b}{2} \rfloor \leq a \leq b$ and (in case they are not leaf nodes) $a+1$ child nodes.
- We always propagate the highest element of the lower child to the parent node when splitting a node, i.e. when splitting the root tree $(e_1, \ldots, e_b)$ because of adding the element e_{b+1}, a copy of the element with number $\lfloor \frac{b}{2} \rfloor + 1$ is propagated to become the new root node, thus resulting in the two child nodes $(e_1, \ldots, e_{\lfloor \frac{b}{2} \rfloor + 1})$ and $(e_{\lfloor \frac{b}{2} \rfloor + 2}, \ldots, e_b)$.

4.3.2 Forensics on revision secure tables

The need for regulatory compliance (e.g. the US Sarbanes-Oxley Act [Sar02], as well as the European Data Directive on Privacy [Fro00] force companies to effectively protect the access to sensitive business data and enable traceability of business processes) drives the demand for databases that conform to strict limitations on the kinds of operations that are allowed. So, many companies limit access to certain databases to a strict insert-only policy, thus prohibiting deletion and updates on a technical basis. This scenario is the starting point for the forensic approach specified in this section.

The following prerequisites were taken into account:

- Only *INSERT*, but no *UPDATE* and *DELETE* statements are allowed. This mechanism is introduced at the software-level, i.e. not even the DB-administrator possesses the ability to change this without breaking the system.
- The considered table has a primary key that is constantly incrementing (e.g. timestamp in milliseconds). This key is also used for structuring the $\mathcal{B}^+$-Tree as it creates the primary index inside the non-leaf nodes.

The following theorem gives us a statement on the structure of the emerging trees:

Theorem 4.3.1. *Let B be a $\mathcal{B}^+$-Tree with $n > b$ elements which are added in ascending order. Then it holds true that the partition of the leaves of B has the following structure:*

$$n = \sum_{i=1}^{k} a_i, \ \textit{with } a_i = \frac{b}{2} + 1, \forall i \neq k \textit{ and } a_k \geq \frac{b}{2}.$$

Figure 4.4 provides an illustration of this specific structure.

Proof. We start by inserting b elements into an empty root, thus when inserting the next element we have to split the root and generate a new one with two leaves. The only possibility is to propagate the middle element to the root and split the leaves in one containing $\frac{b}{2} + 1$, the other containing $\frac{b}{2}$ elements (else one of the prerequisites for $\mathcal{B}^+$-Trees would be violated. Iteratively when we add elements, it is always added to the rightmost leaf, thus resulting in three cases:

1. The leaf contains less than b elements $\Rightarrow$ the new element is added to the rightmost leaf.
2. The leaf contains b elements. The leaf is split into two leaves, the first (according to the assumption made above that we always propagate the highest element of the lower leaves into the parent node) containing $\frac{b}{2} + 1$, the second $b/2$ elements. This can result in two cases:
    a) The parent node is full $\Rightarrow$ it is split itself into two parent nodes containing $b/2$ elements and $b/2 + 1$ child-nodes each and propagating the middle element into the next level. Eventually this could result in the generation of a new root containing only one single element. Still, no element will be put into leaves that are left of the current leaf.
    b) The parent node has enough space, in this case nothing else happens.

Thus, it is impossible for the tree to add further elements into leaves except the rightmost. Again, from this the proposed partitioning follows quite easily. $\square$

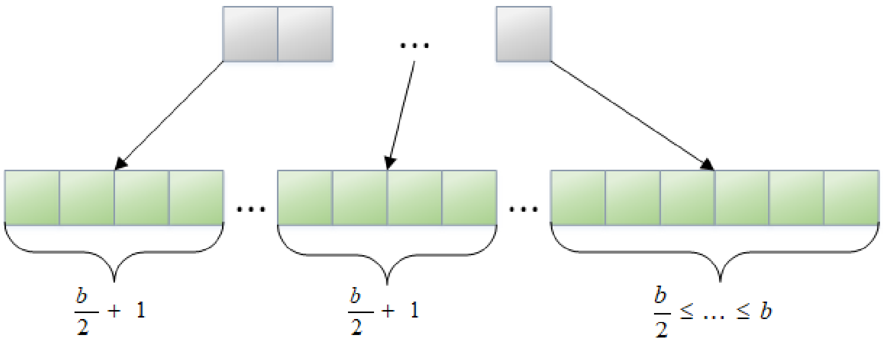

Figure 4.4: A $\mathcal{B}^+$-Tree resulting from ascending ordered inserts only.

The main argument for our forensic approach lies in the fact that even with full insert rights, normally, in a database application, a database administrator is not able to change the data at $\mathcal{B}^+$-Tree-level, since this is solely managed by the DBMS itself. Thus at this level, a malicious administrator will not be able to fake evidence.

We will assume that the database does not rearrange the tree elements (e.g. for reasons of performance) and we are able to directly read the structure of the $\mathcal{B}^+$-Tree (e.g. in InnoDB [Zai09]).

In a $\mathcal{B}^+$-Tree that was built by sorted inserts, non-sorted inserts can be detected by analyzing the fill rate of the leaf nodes. If data is inserted in a strictly sorted order (e.g. a logging table with a timestamp as primary key), the fill rate of all leaves except the rightmost one is, according to Theorem 4.3.2 exactly $\lfloor \frac{b}{2} \rfloor + 1$ elements.

In case a malicious administrator inserts an additional record with a forged, namely pre-dated timestamp, this record may be added to a leaf node that is not the rightmost one. This happens, if there are already at least $b + 1$ elements with a higher key (i.e. timestamps) in the table (in case there are less than $b + 1$ higher keys in the table, the forged record would be added to the rightmost leaf. On an eventual split, this record would then reside in the correct leaf and additionally this leaf would only contain $\lfloor \frac{b}{2} \rfloor + 1$ elements).

When analyzing the fill rate of leaf nodes in this tree, the pre-dated record is located in a node that has a too high fill rate for strictly sorted inserts (i.e. $> \lfloor \frac{b}{2} \rfloor + 1$ elements). This hidden modification can be detected, because the resulting $\mathcal{B}^+$-Tree does not correspond to the insert policy of the database (see Figure 4.5). Since the elements inside a leaf node are sorted by default, only the leaf containing the forged record can be detected, the identification of the element itself is not possible.

Be aware that not all inserts of this form can be detected by this approach, since with a combination of different forged records, the structure can be fixed again by adding enough (i.e. $\frac{b}{2} + 1$) records (see Figure 4.6) Actually, in real life tree-sizes, this would result in the insertion of many forged records, which should be detectable by other means

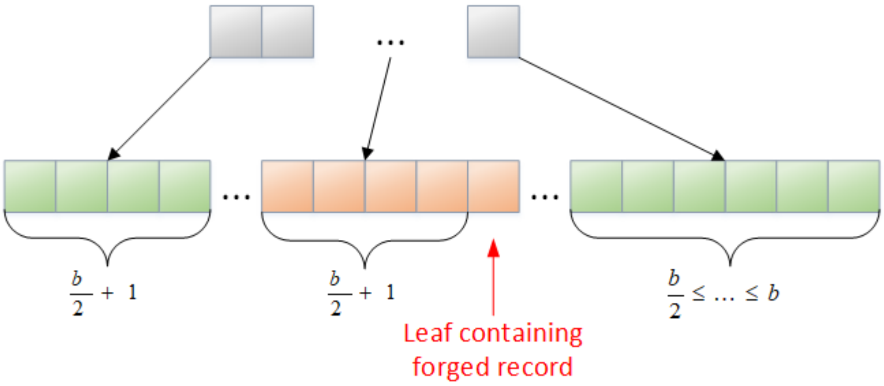

Figure 4.5: Forged record in $\mathcal{B}^+$-Tree with sorted inserts only.

(comparisons, sanity checks). Furthermore, these insertions also affect at least the parent node, since a new leaf is generated.

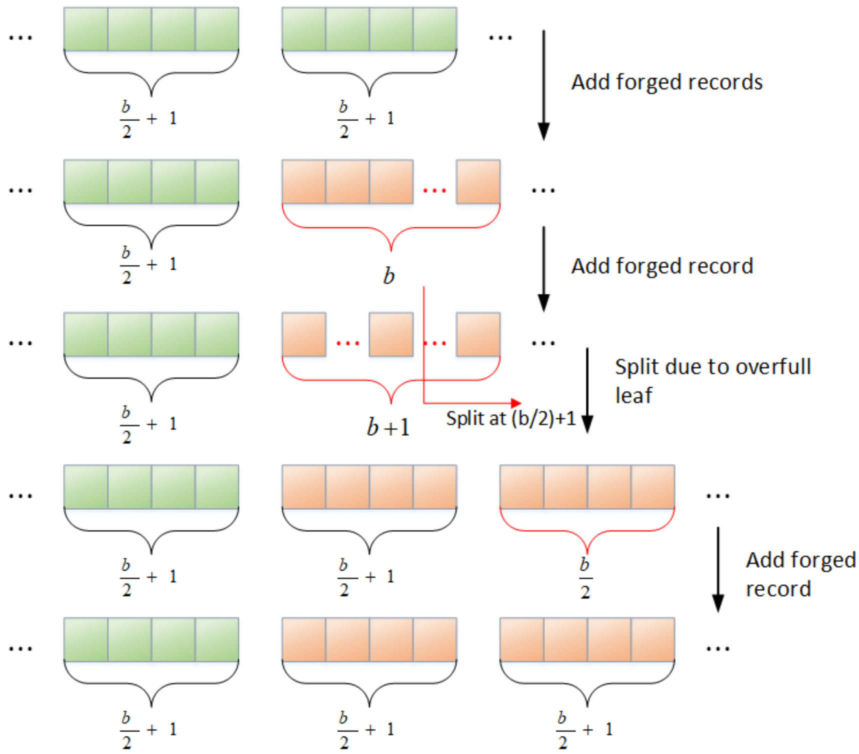

Figure 4.6: Forging many records to thwart forensic analysis.

With this approach, deleted records can be detected in a very similar way. When a malicious administrator deletes one record from a leaf that is not the rightmost, the fill rate of this leaf node falls to $\lfloor \frac{b}{2} \rfloor$ elements, which again is detectable. In case more than one element is removed from the same leaf, the number of elements falls below $\lfloor \frac{b}{2} \rfloor$, thus violating the lower $\mathcal{B}^+$-Tree-boundary for the leaf-size. The following re-balancing will again result in a $\mathcal{B}^+$-Tree-structure, that identifies the manipulation (the merged leaves will result in a leaf of size $> \frac{b}{2} + 1$).

4.3.3 Extension to monotonic descending indices

While we have discussed the approach with respect to strictly monotonic ascending indices, a similar result can be found for strictly monotonic descending indices. On a side note, since we always speak of the primary index, which must be unique, strict monotonicity always follows from monotonicity by default.

Theorem 4.3.2. *Let B be a $\mathcal{B}^+$-Tree with $n > b$ elements which are added in descending order. Then it holds true that the partition of the leaves of B has the following structure:*

$$n = \sum_{i=1}^{k} a_i, \ \textit{with } a_i = \frac{b}{2}, \forall i \neq 1 \textit{ and } a_1 \geq \frac{b}{2}.$$

The proof for this theorem is analogous to the proof given for the strictly monotonic ascending index, the major issue lies in providing a meaningful descending series for indexing purposes in real-life environments. We start with an empty root and add elements (now to the start of the sorted list holding all elements inside the root node) until we reach b elements. When adding the next element, the node must be split. According to the strategy used above, the left leaf will hold $\frac{b}{2} + 1$ and the right leaf $\frac{b}{2}$ elements (as outlined in the background section regarding $\mathcal{B}^+$-Trees 4.2.1, we use the convention that splitting is always done on the $\frac{b}{2} + 1$-th element). This continues analogue to the proof outlined in Section 4.3.2, with the difference that addition is always done to the leftmost instead of the rightmost leaf.

4.3.4 Adoption & Evaluation

This section evaluates how today's database management systems have to be modified to utilize this approach. For performance reasons, database management systems tend to generate wide trees with a small number of levels. Our evaluation showed that even with 450,000 records in an InnoDB table, the tree's height did not exceed a value of 3. Thus, the $\mathcal{B}^+$-Tree does not have much structure to analyze and the extraction of forensic evidence is difficult. We evaluated, based on a modified version of InnoDB, how a more structured tree can be generated and what implications these modifications have on performance.

In InnoDB, both leaf and non-leaf nodes have a fixed size of 16 kilobytes. As non-leaf nodes of $\mathcal{B}^+$-Trees only store keys that have a typical length of a few bytes, there is enough space for adding thousands of keys to a node before it has to be split into two nodes. Thus, a typical non-leaf node in InnoDB has a huge number of children and the resulting tree mostly grows in width but not in height. To increase forensic value of a database tree, we considered two approaches that add structure to an InnoDB tree.

We first modified the fixed size of nodes in the source code of InnoDB. While node sizes other than 16 kilobytes are not officially supported, it is still possible to decrease it to a minimum value of 4 kilobytes. If we assume an integer primary key of 4 bytes, a non-leaf node in the original InnoDB storage engine can store up to 4,000 keys. We can reduce the value to 1,000 by defining a fixed node size of 4 kilobytes in the source code of InnoDB. This value, however, is still far too large for the generation of forensic-aware database trees. In a second evaluation, we increased the size of the primary keys in order to limit the number of keys that can be stored in one node. We defined a `varchar` primary key with a length of 767, which represents the maximum length of a key in InnoDB. Using the 450,000 test records from the previous evaluation, the resulting tree now has a branching level of 21, i.e., each node has up to 21 child nodes and the height of the tree rises to 7, which makes it more valuable for forensic investigations.

We compared performance of two InnoDB tables, one with a tree height of 3 (small primary key) and one with a height of 7 (large primary key). All experiments were performed with MySQL 5.1.44 on a machine equipped with an Intel Core i7 2.66 GHz CPU and 8GB of available system memory. The results (Table 4.1) show that queries to the forensic-aware database are about twice as slow as to the database with the wide tree. While the performance losses are quite big, we argue that there exist cases (e.g. SOX compliant accounting) where security outranks performance.

Statement	tree height = 3	tree height = 7
Simple full table scan	0.0003s	0.0007s
Full table scan for single value	0.0055s	0.0115s
Update one column in entire table	0.1066s	0.2387s

Table 4.1: Practical evaluation of runtime impact.

4.4 Limitations

In this section we discuss the limitations of our approach with respect to (i) stability, (ii) explanatory power, as well as (iii) possibilities for generalization.

4.4.1 Limitations regarding stability

One major prerequisite in order for this approach to work is the requirement that the database does not reorder the elements in the $\mathcal{B}^+$-Tree, e.g. for the sake of performance optimization. While this might hold true for specialized revision-secure databases or file systems, it often does not hold true for real-life implementations of large-scale databases, where performance is of the utmost importance and thus constant re-organization of the primary indexes is taking place. In order to use our approach as outlined in this chapter, these mechanisms must be turned off. While this might not be suitable for high-volume data as typically processed in data warehouses, this can be possible for the Audit & Control [Web98] data typically collected in said systems, i.e. while this approach might not be feasible for protecting the data in the data warehouse, it can be useful for protecting the audit information. This focus on audit information was also a starting point for the design of the detection mechanism, as this information must be insert-only (maybe with small exceptions on some timestamps, depending on the exact implementation of the mechanism, but not relevant for our approach) and can often be structured in a sorted manner, typically using timestamps and IDs of workflows, data sources and incidents.

4.4.2 Limitations regarding bijectivity

The most interesting idea regarding this approach towards database forensics would lie in its generalization to be suitable for all kinds of operations, no matter what order they are applied. Unfortunately we can prove that this is impossible since the $\mathcal{B}^+$-Tree does not give us enough information for this in general. More precisely, the main limitation of our approach lies in the fact that in general the operation of inserting an element into a $\mathcal{B}^+$-Tree is not bijective since the inverse operation is not injective, i.e. even with knowledge on the inserted elements, it can be impossible to recalculate the original tree (or an intermediate state), even though the resulting tree and a log of all operations is available.

Example 4.4.1. *The example given in Figure 4.7 illustrates how adding the same element to two $\mathcal{B}^+$-Trees A and A' with different structures generates the same tree B.*

Thus the order of insertion (like specified in the revision secure case) is very important for our approach.

4.4.3 Adoption for arbitrary INSERT-only tables

The approach outlined in Section 4.3, while providing valuable insights for certain types of tables typically encountered in working environments like data warehouses and workflow engines, has a major drawback: The insertion strategy for the data is severely limited to (strictly) monotonous INSERT orders. Still, while this is enough for Audit & Control databases, most data owners want to have more flexibility, thus it would be interesting to extend the approach towards more general INSERT-orders. Thus, we will have a short glance at the additional research question, whether this approach is able to detect the deletion of elements in an INSERT-only database, where the order of the INSERT-operations is arbitrary. If this is possible, then we would next look into the detection of general database operations, i.e. extending the approach towards general forms of DELETE-operations.
In order to decide the applicability of this approach, we will discuss, what forms of finite $\mathcal{B}^+$-Trees (i.e. the number of elements in the tree is finite) can be constructed by INSERTS only and where DELETES are needed. For the sake of simplicity, we identify

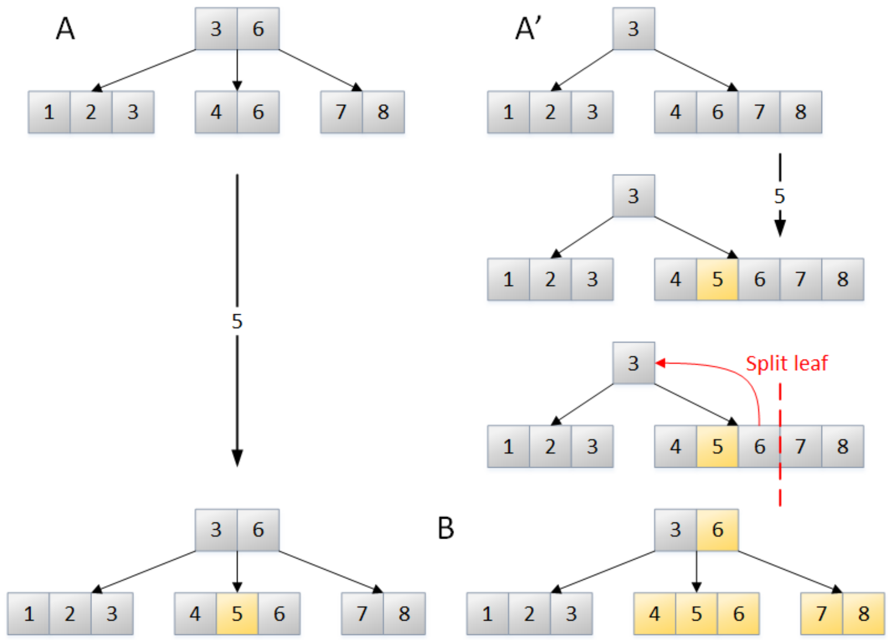

Figure 4.7: $\mathcal{B}^+$-Tree resulting from two different $\mathcal{B}^+$-Trees.

each element with its number when sorted in ascending order, i.e. 4 denotes the 4^{th} smallest element of the set. Additionally, we identify each leaf node with its position in the tree, i.e. a_i is the length of the ith leaf node.
As already outlined in Section 4.3, when doing inserts of n keys sorted in ascending order, we get the following resulting leaf forms:

$$n = \sum_{i=1}^{k} a_i, \text{ with } a_i = \frac{b}{2} + 1, \forall i \neq k \text{ and } a_k \geq \frac{b}{2}.$$

with k being the number of leaves and a_i the number of elements in the i-th leaf.

Our first observation is that we can construct a $\mathcal{B}^+$-Tree where the j-th leaf contains $\frac{b}{2} + 2$ elements (see Figure 4.8).

$$n = \sum_{i=1}^{k} a_i, \text{ with } a_j = \frac{b}{2} + 2 \text{ and } a_i = \frac{b}{2} + 1, \forall i \neq j, k \text{ and } a_k \geq \frac{b}{2}.$$

Proof. For constructing this tree, we just have to add the elements (which are sorted by size) in the following order:

$$1, \ldots, (j \cdot (\lfloor \frac{b}{2} \rfloor + 1) - 1), (j \cdot (\lfloor \frac{b}{2} \rfloor + 1) + 1), \ldots, n, j \cdot (\lfloor \frac{b}{2} \rfloor + 1)$$

The first $n-1$ INSERTS will construct a tree like the above structure, where the j-th leaf node has got $\lfloor \frac{b}{2} \rfloor + 1$ elements. If we now add element $j \cdot (\lfloor \frac{b}{2} \rfloor + 1)$, it won't be inserted at the rightmost node, but in the j-th node (see Figure 4.8). $\square$

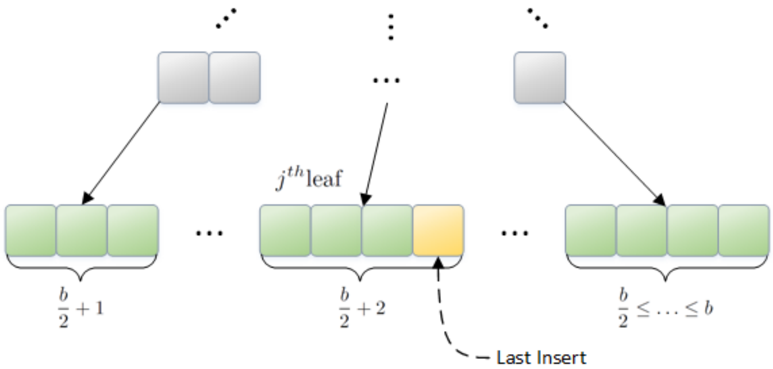

Figure 4.8: $\mathcal{B}^{+}$-Tree with one node with $\frac{b}{2}+2$ elements

The above approach can be easily expanded to insert up to b elements in one node by leaving out the suitable elements.

The next question is, if this can be done with an arbitrary number of leaf-nodes.

$$n = \sum_{i=1}^{k} a_i, \text{ with } a_i \geq \frac{b}{2}+1, \forall i \neq k \text{ and } a_k \geq \frac{b}{2}.$$

Actually, this question is equivalent to the question of building any composition of the number n with particles between $\lfloor \frac{b}{2} \rfloor + 1$ and b. For solving this, we adapt the algorithm presented in the last proof with a slight difference (see also Figure 4.9): The adding of the elements that were left out does not happen in the very end for all nodes at the same time, but in order to guarantee that there are no side-effects, we add these elements soon after the leaf node right to the node they belong to was started.

This can be extended a bit further, in order to construct trees that have leaves with only $\frac{b}{2}$ instead of $\frac{b}{2}+1$ elements in most leaf nodes (except the first and the last leaf), i.e. we can build all leaf-structures of the form

$$n = \sum_{i=1}^{k} a_i, \text{ with } a_1 \geq \frac{b}{2}+1, a_k \geq \frac{b}{2}, a_i = \frac{b}{2}, \forall i \neq 1, k.$$

For proofing this theorem, we just have to proof it is possible for us to generate a leaf holding $\frac{b}{2}$ elements only at each leaf-position except the first and the last (Note: The last

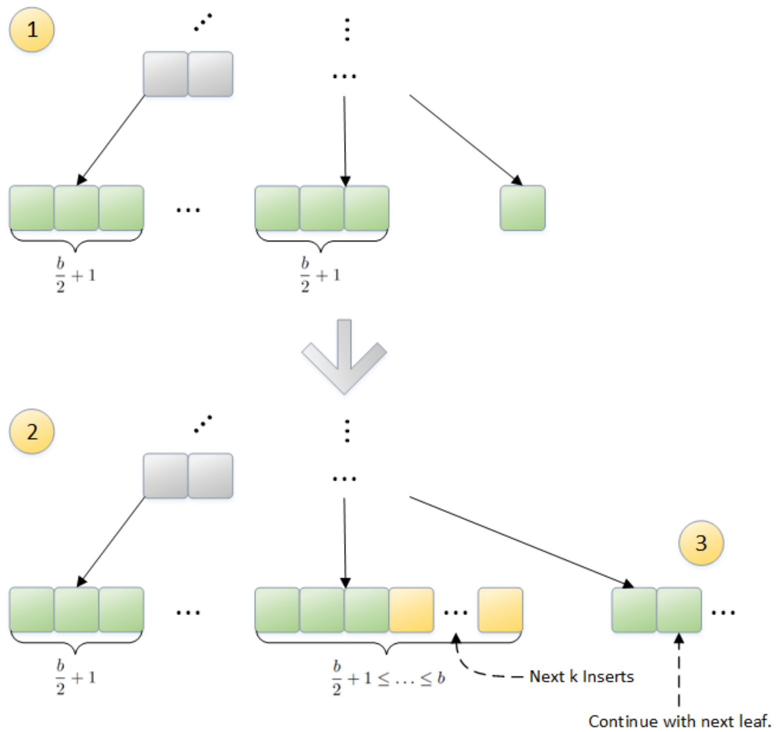

Figure 4.9: Constructing a $\mathcal{B}^+$-Tree with nodes holding more than $\frac{b}{2}+1$ elements

node always contains the remaining items, its size actually depends on the total number of elements).

Proof. If we have a leaf node i with $a_i = \frac{b}{2}+1$ elements, it is possible to insert a node k with $a_k \geq \frac{b}{2}$ elements left to it by adding these a_k to node i. Since i originally possesses $\frac{b}{2}+1$ elements this sums up to $\frac{b}{2}+1+\frac{b}{2} = b+1$ elements, thus forcing a split of the node. The first $\frac{b}{2}+1 = a_i$ elements will stay in node i, thus changing nothing there. The other $\frac{b}{2}$ elements will form the new leaf node (see Figure 4.10). Thus for constructing a leaf node with only $\frac{b}{2}$ elements, we just need to omit $\frac{b}{2}$ elements in the insertion order and add them right after the next leaf node is started. Actually this is not possible for the first node. The proof is rather trivial: According to our prerequisites, in the case of splitting, there are always $\frac{b}{2}+1$ elements left on the right node. So the only possibility of having less elements in the rightmost node at leaf-level is having a tree only consisting of the root node (or doing DELETES).

Additional INSERTS for generating leaf nodes with more than $\frac{b}{2}+1$ elements have to be done after all nodes with $\frac{b}{2}$ elements have been generated, or this will thwart the generation process of the nodes containing only $\frac{b}{2}$ elements. □

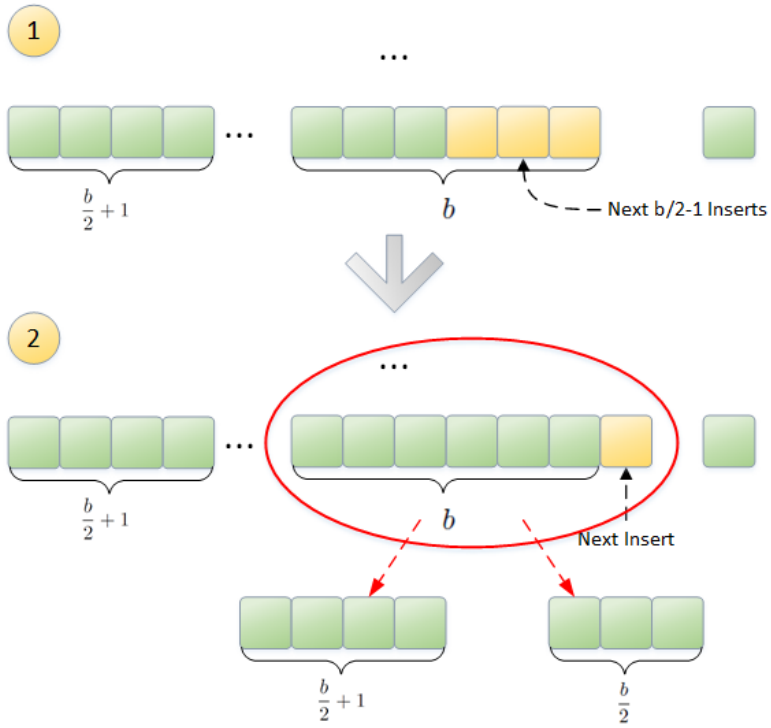

Figure 4.10: Constructing a $\mathcal{B}^+$-Tree with nodes holding only $\frac{b}{2}$ elements

The major result of this proof is that there is only one leaf structure of a $\mathcal{B}^+$-Tree where a DELETE-operation is required for construction: When the first leaf node only contains $\frac{b}{2}$ elements (of course it must be kept in mind that the length of the last node solely relies on the number of remaining attributes).
Thus, in this section we were able to show that the approach for $\mathcal{B}^+$-Tree-forensics for databases cannot be generalized for the detection of data deletion in case of a table that allows general, non-monotonous, data insertion. Of course, the approach can still be used for the task it was originally intended, indicating manipulations in Audit & Control tables. It must be noted though that, similar to the original approach, in this section we solely concentrated on the structure of the leaf nodes and did not consider the whole structure of the tree including internal nodes. Still, in essence the techniques work similar for changing the inner node structure, just the amount of elements required to be inserted later gets rather large. Furthermore, these proofs can be trivially expanded from $\mathcal{B}^+$-Trees to $\mathcal{B}^*$-Trees, as the only difference between the two is the minimal number of elements inside the leaf nodes, the other requirements remain the same, which also means that the proofs given above work similar.

4.5 Using the structure of index trees for logging

As we have shown in the previous sections, the structure of a $\mathcal{B}^+$-Tree heavily relies on the insertion order of the elements into the tree, especially in cases, where there is no optimization.

4.5.1 Approach for Logging

We start by defining the *signature* for $\mathcal{B}^+$-Trees, since this is the prevalent data structure used in most modern relational database systems such as MySQL with the InnoDB storage engine as already outlined in the previous sections. Since the signature is only concerned about structure, this definition can easily be adapted to be suitable for $\mathcal{B}$-Trees and, as a starting point for future research, other balanced data structures.

Definition 4.5.1. *Let $L_i, i = 1 \dots k$ be all leaf nodes of a $\mathcal{B}^+$-Tree B. Then we define the k-tuple $(|L_1|, \dots, |L_k|)$, where $|L_i|$ is the number of elements in node L_i, as the* leaf signature *of B.*

The leaf signature already provides some valuable insights on the distribution of the data and could be used for digital forensics. Still, it is not sufficient for a unique description of the tree, since even with the (needed) knowledge of the data, several different trees possessing the same leaf signature (and thus the same partitioning of the data into leaf nodes) can be constructed (Figure 4.11 shows an example).

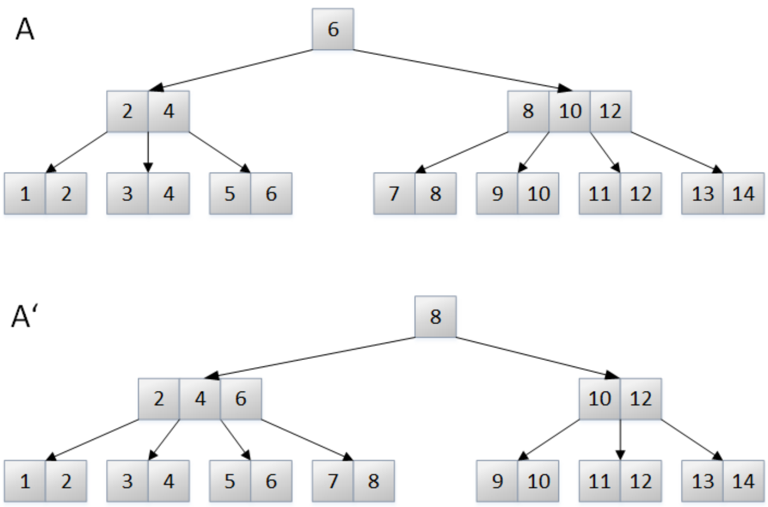

Figure 4.11: Two different $\mathcal{B}^+$-Trees with the same leaf signature

Definition 4.5.2. *Let $C_{i,j}$ be the jth node on the ith level of the tree, where nodes are counted in ascending order of the elements contained, and where $i = 1$ denotes the level of the root. Let h be the height of the tree. Let D_i be the* level signature, *such that $D_i = (|C_{i,1}|, |C_{i,2}|, \dots)$ for some i. Thus, D_i is the tuple containing the size of the nodes on the i-th level. Then we define S to be the* tree signature, *such that $S = (D_1, D_2, \dots, D_h)$. Thus, the tree signature is the tuple containing the level signatures of each level of the tree.*

Using the definition above, we will now proof that the information contained in the signature is sufficient for a comprehensive description of a $\mathcal{B}^+$-Tree.

Corollary 4.5.3. *The structure of a $\mathcal{B}^+$-Tree is well-defined by its signature.*

Proof. For our proof, we assume having a non-empty tree with a valid tree signature. Let us consider D_1, the root level signature, which will always have the form (x). We know that the root of any valid tree will have $x + 1$ child nodes and, since we have assumed a valid tree signature, D_2 must therefore represent these $x+1$ children. Since all elements inside a node are sorted in ascending order (by the definition of a tree), there is exactly one valid way of matching the elements of D_2 onto the children of the root of the tree. Each child node can then be considered as the root of a subtree. By applying the same principle, we can conclude that for each unique tree signature, there is exactly one corresponding structure. Thus, the structure of a tree is well-defined by its signature. □

Example 4.5.4. *Let B be a $\mathcal{B}^+$-Tree of order 4 containing the numbers $1 \ldots 16$ with the following signature: $S = ((1), (2, 2), (3, 3, 2, 3, 3, 2))$. Then the resulting $\mathcal{B}^+$-Tree can be seen in Figure 4.12.*

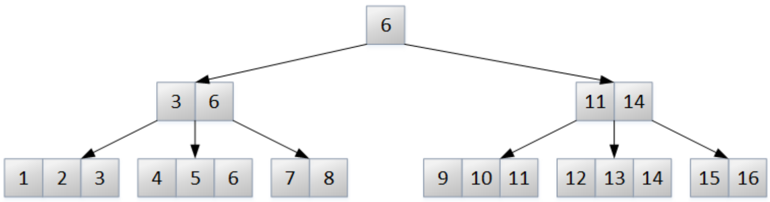

Figure 4.12: $\mathcal{B}^+$-Tree of order 4 with $S = ((1), (2, 2), (3, 3, 2, 3, 3, 2))$

4.5.2 Constructing trees from signatures

In this section we outline an algorithm for constructing a $\mathcal{B}^+$-Tree from the corresponding set of (sorted) index data together with the signature.

1. The last tuple of the signature is retrieved and the index data is partitioned accordingly, i.e. it is divided into blocks with the length derived from the last signature-tuple respectively. The resulting groups form the leaves of the $\mathcal{B}^+$-Tree.

2. The leaves are grouped according to the signature of the level above, i.e. by the second to last tuple of the signature.

3. Step 2 is repeated for each newly constructed level until we reach the root node (the first entry in the signature).

4. The final tree is constructed by promoting the correct elements into the inner nodes and the root node. This step is well defined by the $\mathcal{B}^+$-Tree-definition.

Following, we give a short example in order to illustrate the algorithm.

Example 4.5.5. *Let B be the $\mathcal{B}^+$-Tree of order $b = 4$ as shown in Figure 4.12 together with the signature $S = \{(1), (2, 2), (3, 3, 2, 3, 3, 2)\}$.*
In the first step of the algorithm we partition the data according to the last tuple of the signature in order to generate the leaves of the tree. The elements $1 \ldots 16$ partitioned by $(3, 3, 2, 3, 3, 2)$ thus form the leaves $(1, 2, 3), (4, 5, 6), (7, 8), (9, 10, 11), (12, 13, 14), (15, 16)$. Now, each leaf node is an element in the next partitioning, i.e. the leaf nodes are grouped according to the signature of the next level $(2, 2)$ thus resulting in two inner nodes with three child nodes each: $((1, 2, 3), (4, 5, 6), (7, 8))$ and $((9, 10, 11), (12, 13, 14), (15, 16))$. Combining these two subtrees with the root node yields the final $\mathcal{B}^+$-Tree.

The complexity of the algorithm can be calculated in the following way: Since we assume that the n leaf-elements are already sorted in ascending order, partitioning can be done very fast (n operations). This has to be repeated for each level, where the maximum number of elements on each level can be estimated with $n_i \leq \log_{\frac{b}{2}+1} n_{i+1}$ (upper bound), thus resulting in the following formula for the whole tree (l denotes the number of levels of the $\mathcal{B}^+$-Tree.):

$$\sum_{1}^{l} A_i, \text{ with } A_i = \log_{\frac{b}{2}+1} A_{i+1}, A_l = n$$

4.5.3 Application

The signature can be used as an additional log that is only concerned with the structure of the tree, not its content. Establishing such a log helps tracing reorganizations, i.e. rebalancing or reindexing and all other operations that do not change the content of the tree, but its internal structure.

In this section we discuss the following applications for the signature log:

1. Providing an additional log file.
2. Forensics using reconstruction of old versions.
3. Forensics by simulating tree development.
4. Providing exact backup copies.

Additional log file

The signature log provides an additional log that can be used in order to detect manipulations of a database table. Since many operations result in changes of the underlying $\mathcal{B}^+$-Tree-structure, these operations will also affect the signature of the $\mathcal{B}^+$-Tree. Since in our forensic analysis we also need to be aware of a malicious administrator who is able to manipulate this log, we want to analyze the significance and robustness of this additional log in more detail.

In case the adversary is not able to tamper with the signature log (i.e. she does not hold the privileges or the ability to rewrite log files), the following information can be derived:

1. The structure of the current tree must correspond to the signature in the signature log.
2. Additionally, the number of logged signatures must be the same as operations in the transaction log, so we can use the signature log to cross-check for any obvious manipulation in these transaction-logs.

Tampering with the signature log is far from trivial, since:

- The adversary must be able to read out the structure of the $\mathcal{B}^+$-Tree by herself.
- Insert very subtle changes, since even small changes in the insert order of elements can have a great impact on the structure of the resulting trees - especially when there are more complex transactions following the one that got changed.
- In the end, all these changes must lead to a tree that has the same structure like the one that is observed at the moment.

As was already discussed in Section 4.3, the structure of the underlying $\mathcal{B}^+$-Tree can be used for forensic analysis, especially ordered insert statements can be revealing.

Reconstructing old $\mathcal{B}^+$-Tree-versions

As outlined in the limitations (Section 4.4), just logging the transactions issued against a database is not enough to preserve a log of all previous structures of the underlying $\mathcal{B}^+$-Tree, since in general not even the operation of inserting an element into a $\mathcal{B}^+$-Tree is bijective (the operation is not injective). Moreover, when dealing with multiple INSERT- and DELETE-operations, it is usually impossible to reconstruct the former tree (See Figure 4.7).

Since the signature of the tree (together with its characterization by its order) identifies the structure, the signature log together with the tree and a log of all operations on the tree can be used to retrieve any intermediate state, since

1. the partitioning of the nodes (and thus the structure) of the $\mathcal{B}^+$-Tree is denoted in the signature,
2. the data inside the $\mathcal{B}^+$-Tree can be retrieved by combining the current tree with the information written in the change logs for every intermediate state and
3. the combination of this information (together with the basic characterization of the tree) defines the whole tree.

Thus, the original tree can be restored with a combination of the data (taken from other log files) and the signature log. Again, we can apply the algorithm described in Section 4.5.2.

1. Be E the set of all elements in the current $\mathcal{B}^+$-Tree.
2. Choose a point in time t, for which the tree should be reconstructed.
3. Extract $D(t)$ and $I(t)$ from the transaction logs, where $D(t)$ is the set of all deleted elements and $I(t)$ the set of inserted elements since t.
4. Join E with D and subtract I in order to calculate E(t), the set of all elements at point in time t: $E(t) = (E \cup D(t)) \setminus I(t)$.
5. Use the tree signature $S(t)$ at point in time t and the algorithm proposed in Section 4.5.2 to reconstruct the old state of the $\mathcal{B}^+$-Tree.

Obviously this approach is also invulnerable against reorganizations of the $\mathcal{B}^+$-Tree due to performance optimization strategies, since these operations only change the underlying structure of the $\mathcal{B}^+$-Tree, which is already stored in the signature log.

One additional prerequisite must be fulfilled though: UPDATE-statements must be logged as a combination of the respective INSERT- and DELETE-statements.

Example 4.5.6. *Let $E = \{1, 2, 3, 4, 5, 6, 7, 8\}$ be the elements of a $\mathcal{B}^+$-Tree of order 4 that was constructed from a previous $\mathcal{B}^+$-Tree with signature $S(t) = ((2), (2, 2, 3))$ by inserting the elements 2 and 4, as well as deleting element 9.*
Thus we can construct the old tree with $E(t) = ((E \cup D(t)) \setminus I(t) = \{1, 3, 5, 6, 7, 8, 9\}$ where $D(t) = \{9\}$ and $I(t) = \{2, 4\}$ and reconstruct the $\mathcal{B}^+$-Tree with respect to the signature $S(t)$. The result can be seen in Figure 4.13.

In case of differences between a reconstructed tree and an old backup, manipulations can be detected.

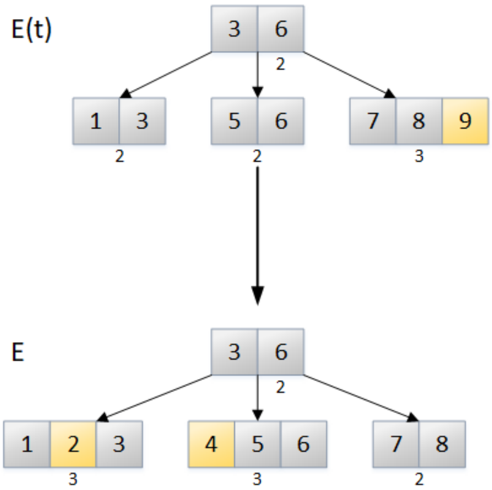

Figure 4.13: Reconstructed $\mathcal{B}^+$-Tree

Forensics by simulation

A straight forward approach for manipulation detection is to find inconsistencies by starting with an old version of the tree and iteratively applying the transactions step-by-step. In case some data was simply erased from the tree and all transactions concerning this record were simply deleted from the log, there is still a good possibility of revealing this manipulation by differences in the structure of the stepwise generated $\mathcal{B}^+$-Tree compared to the one found in the live system.

The main problem with this approach lies in the generation of an old version, since this would either need a complete and exact backup, even covering the structure of the tree, or a complete remake of the table, starting with the very first transaction.

In order to overcome this weakness of the approach, we propose the following strategy for simulating the $\mathcal{B}^+$-Tree with the help of our signature approach:

1. Identify a point in time t, where the database still seemed trustworthy.
2. Reconstruct the tree at this point according to Section 4.5.3.
3. Extract all transactions from the transaction log that are needed to construct the current tree.
4. Apply the transactions from the transaction log step-by-step and compare the structure of the resulting trees with the structures saved in the signature log.

For very big tables, this approach can be rather costly, especially in case a long period of time needs to be covered, since all transactions starting with the beginning of the simulation period must be executed. Still, this technique can be highly parallelized by dividing the simulation interval into several subintervals and calculating them separately:

1. Identify a point in time t_0, where the database still seemed trustworthy.
2. Divide the interval $[t_0; \text{now}]$ into several subintervals $\{[t_0; t_1], [t_1; t_2], \ldots, [t_{n-1}; \text{now}]\}$.
3. Reconstruct the historic $\mathcal{B}^{+}$-Tree at each $t_i, i = 0, \ldots n-1$ and carry out the simulation according to the algorithm given before.
4. Abort in case inconsistencies were found in one interval and conduct further analysis on this interval.

Providing exact copies of $\mathcal{B}^{+}$-Trees

Another application for the signature log lies in providing the possibility to generate exact copies for migration or backup, i.e. copies that not only hold the information on the data inside the tables and indexes, but also consider the structural aspects of the indexes. This can be especially useful, if the database needs to be backed up very often and the ability to do forensic techniques such like the techniques outlined in 4.5.3 should still be possible. Traditionally, this can be done by creating backups of the underlying DBMS-files holding the indexes, which is very costly if done very frequently. Utilizing incremental transaction logs for simulating the tree starting from a full backup is not applicable for this purpose, since any reorganization would thwart the simulation. The tree signature thus provides a very straightforward additional mechanism for full or incremental structural backups:

1. Generate a full backup and backup the signature.
2. Incrementally backup all data changes (e.g. via the transaction log) and changes in the signature.
3. Additionally backup the signature log before and after each reorganization.

We also consider migration of revision secure databases as an area of interest for this application. Especially in combination with the simulation of the tree structure, or some other forensic techniques concerning $\mathcal{B}^{+}$-Trees, this application can be of great value in order to generate SOX-conforming backup strategies.

4.5.4 Limitations

One limitation of this approach lies in the fact that the technique must be implemented inside the DBMS, i.e. it is possible to do this via external tools, but to us this seems a rather costly alternative concerning performance. So the vendor of the DBMS must be convinced to change the software, which is rather unlikely concerning the big players in this marked.

Another limitation is based on the problem that the bijectivity of the transaction can only be assured when *every* change in the underlying structure is logged. This will result in some additional space needed, as well as an additional logging operation after each transaction. Since every log entry covers the structure of the whole tree, the following mitigation strategies should be considered for this issue:

- In case a transaction only affects a sub-tree, the logging-logic could reuse the last entry of the signature log and directly alter the part of the signature that is concerned with this sub tree.
- The logging could rely on some external tool that calculates intermediate steps out of two tree signatures (extracted the usual way) and the transaction log.

The most important drawback, however, lies in the fact that an almighty administrator is, at least in theory, able to completely fake the log. Even though, faking the signature log should be drastically more complicated compared to a typical database audit log, this possible threat has to be taken into consideration. In order to thwart this attack, the signature log could be combined with other forensic techniques that protect logging information, like for example the chained witnesses approach on transaction logs (see [FKKW14]).

4.5.5 Size of the signature log

In this section we discuss some boundaries on the expected size of the proposed signature log with respect to the number n of elements stored in the $\mathcal{B}^+$-Tree (the actual size in bytes thus depends on the implementation of the DBMS) and give some boundaries on the size needed. It is assumed that the same data type is used for storing numbers, i.e. that storing small numbers is as expensive as storing longer ones (this is usually the case in real-life applications).

Lower bound: The best case is that as many leaves as possible are filled with b elements. Note that we do not need to concern ourselves with the possibility that, by applying this concept, we fail to satisfy the constraint that all non-root nodes must contain at least $\frac{b}{2}$ elements, since we are only trying to find a valid lower bound. We then construct the inner nodes of the tree iteratively using the same argument

$$N_L(n) = \left\lceil \frac{n}{b} \right\rceil + N_L\left(\left\lceil \frac{n}{b} \right\rceil - 1\right), N_L(n \leq b+1) = 1$$

Upper bound: In order to calculate an upper bound we construct a tree that has as many nodes as possible. This can be achieved by constructing a $\mathcal{B}^+$-Tree containing as many leaf nodes as possible (it is obvious that less leaf nodes can not result in more inner nodes), and we therefore want each leaf node to contain the fewest allowable number of elements. The n elements of the $\mathcal{B}^+$-Tree are divided into leaf nodes L_i such that $|L_i| = \lceil \frac{b}{2} \rceil, \forall i$ where $|L_i|$ is the number of elements in the i-th leaf. As with the construction of the lower bound, we are less concerned about breaking the capacity constraints on nodes, so long as we find a valid and useful upper bound. Once again, we continue iteratively until we reach less than or equal to $b+1$ elements left on the last level. The only valid way of adding a node at this level is by adding a root node. This results in the following recursion:

$$N_U(n) = \left\lceil \frac{n}{\lceil \frac{b}{2} \rceil} \right\rceil + N_U\left(\left\lceil \frac{n}{\lceil \frac{b}{2} \rceil} \right\rceil - 1 \right), N_U(n \leq b+1) = 1$$

Example 4.5.7. *Consider $n = 50$ elements in a $\mathcal{B}^+$-Tree of order 8. By applying our bound calculations, we see that for $b = 8$, $N_L(50) = 8$ and $N_U(50) = 15$. Thus, we would need a signature log of between 8 and 15 in length, for any possible tree structure fitting these criteria. Figure 4.14 shows a $\mathcal{B}^+$-Tree of this form that uses the least possible number of nodes, which is 8. This tree would therefore require a signature log of length 8. Similarly, Figure 4.15 satisfies the same properties of our example but uses the highest possible number of nodes, which is 15, requiring a signature log of length 15. This should provide some indications on the accuracy of our bounds, although admittedly this example is impractically small. Note that, in general, the number of elements and order of a $\mathcal{B}^+$-Tree are both much higher.*

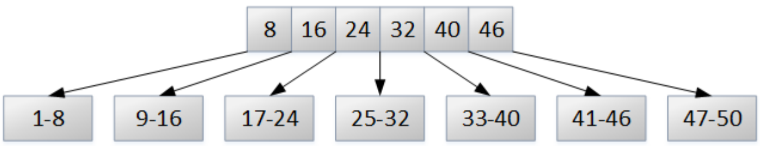

Figure 4.14: An example of a minimal $\mathcal{B}^+$-Tree, $n = 50, b = 8$

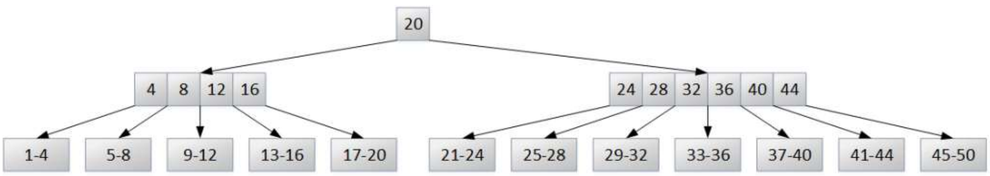

Figure 4.15: An example of a maximal $\mathcal{B}^+$-Tree, $n = 50, b = 8$

Example 4.5.8. *Consider $n = 20,000$ elements in a $\mathcal{B}^+$-Tree with order 1024, we need between 21 and 41 nodes, thus resulting in a signature log of 21-41 integers, needing a most 2 bytes each, thus resulting in a total size of 42-82 bytes per entry (not including delimiters).*

4.6 Fingerprinting databases using index trees

In this section we will discuss, how the index tree could be used for fingerprinting whole databases, i.e. we assume that the data is not exchanged based on extraction files (e.g. .csv [Sha05], JSON [Bra17] or other structured formats), but by copying the database file(s). There are quite some scenarios, where this would be a sensible choice: For example, a file system admin could copy the files of a (probably encrypted) database without requiring access to the data itself. Also, when migrating the data to a mirror machine, it can be handy to simply move the whole database installation to a second server, without the risk of forgetting information, stored procedures, secondary indices and other assets [LX09]. In addition, this is especially interesting in cases, where the database is the target of a forensic investigation and a bit-by-bit forensic copy (a so-called *forensic image* [Joh12]) is required in order to provide material that will hold as a prove in court [CS04]. Thus, within this section, we will only look at the case that the whole database containing all data files, as well as structural information, is copied as it is, without further changes.

4.6.1 Fingerprinting using the $\mathcal{B}^+$-Tree signature

The previous sections, and especially the definition of the $\mathcal{B}^+$-Tree signature in Section 4.5.1, provide us with a straightforward instrument for fingerprinting databases. The basic idea is to provide every data recipient with the same data, but to have the primary index (which in many databases and especially in MySQL using InnoDB is structured in the form of a $\mathcal{B}^+$-Tree) being structured differently in order to generate a different mark for each recipient respectively.

Fingerprint generation and verification

In order to generate these fingerprints, the data owner is required to generate several fundamentally different $\mathcal{B}^+$-Tree structures for the same data. For the sake of simplicity, we assume that the database only holds a single table containing data and leave all the internal database tables and views aside. This does not pose a problem, as the approach can be easily extended to databases holding multiple tables by either only using a single one as carrier for the fingerprint, or by attaching a fingerprint to each table separately.

For fingerprint generation, the data owner can use the technique outlined in Section 4.4.3, where we were able to generate all but one very specific $\mathcal{B}^+$-Tree signature solely by data insertion. More specific, we propose the following routine:

1. Generate a valid signature for each data recipient that is constructible by INSERT operations only. To simplify this task, it is sufficient to generate a valid leaf signature and iteratively generate a valid $\mathcal{B}^+$-Tree from there.

2. Rename the table and generate an empty one with the same attributes under the old name.

3. Generate the fingerprinted table by using the insert strategy outlined in Section 4.4.3.

4. Delete the renamed table.

When selecting the fingerprints in the first step it is of course useful to have them differ in large portions in order to make the fingerprint a bit more stable against changes due to working with the database.

Evaluation and limitations

In this section we will give a short discussion of this approach with respect to (amongst others) key issues like variability, stability and detectability of the signature based fingerprinting approach outlined above.

Performance: The generation of the fingerprint is quite costly in nature, as n INSERT-operations need to be executed in the right order to generate each fingerprint. Normally, databases offer methods for bulk insertion for transferring large amounts of structured information (e.g. [SCS+08]), thus arriving at $m \cdot n$ INSERT operations for m data recipients. Furthermore, the calculation of this insertion order requires additional efforts. This effort can be drastically reduced when limiting the fingerprint to a small amount of elements, i.e. instead of spreading the fingerprint across the whole set of elements, only a short sub-interval is used. This reduces the effort for tree construction for all subsequent $m-1$ fingerprints after the first one to $(m-1) \cdot n_i$, where n_i is the length of the fingerprinted interval. It must be kept in mind though that this reduction also reduces the variability of the fingerprints, as well as the stability against accidental and/or intended forging of fingerprints of other users.

Variability: The variability of this fingerprinting approach, and thus the number of possible fingerprints and data recipients, is directly depending on the number of constructible $\mathcal{B}^+$-Trees. Thus, in order to provide a lower bound for the number of fingerprints that can be generated with this approach, we will show that the problem of the number of $\mathcal{B}^+$-Trees is isomorphic to the problem regarding numbering compositions discussed in Section 4.2.2:

Lemma 4.6.1. *$\mathcal{B}^+$-Trees store all information in the leaf nodes, thus the leaves are forming a partition of the elements stored in the tree.*

Proof. The proof is rather trivial. Following the definition of a $\mathcal{B}^+$-Tree, there is no data stored in the inner nodes, thus all elements appear in the leaves. On the other hand, no element is stored in two different leaves (also by definition). Thus $\bigcup L_i$, where L_i denotes all leaves of the tree, contains all elements of the tree and $L_i \cap L_j = \emptyset, \forall i \neq j$. $\square$

Theorem 4.6.2. *The next observation is that for every partitioning of our data into particles with length between $\lfloor \frac{b}{2} \rfloor$ and b there is at least one $\mathcal{B}^+$-Tree.*

Proof. We divide the proof corresponding to the number of leaves:

1. In case we have only one leaf, this is trivial: This leaf becomes the root and our tree is the trivial one only consisting of a root.

2. In case we have at most $b+1$ leaves. Then we can attach these leaves to one root (by promoting the highest element of each leaf except the last one into the root node). There is no problem with attaching less than $\frac{b}{2}+1$ leaves, since the root node is the only node without a lower restriction on the number of elements.

3. The last case considers more than $b+1$ leaves. In this case we will have to add inner nodes, since it is not possible to attach more than $b+1$ leaves to the root. Be $m_0 > b+1$ the number of leaves in our tree. By using Theorem 4.2.9, we just calculate the number of compositions m_1 of m_0. Given that $m_0 > b+1$, it holds that $m_1 > 0$ [1]. We now apply this trick recursively until we reach case one or two.

□

Based on the last theorem, we are now able to generate a lower boundary for the number of possible $\mathcal{B}^+$-Trees for a given tree-order b and number of elements n that are stored in the tree.

Theorem 4.6.3. ***Lower bound*** *The former result directly leads us to a lower bound for the number of possible $\mathcal{B}^+$-Trees for n elements:*

$$\mathcal{C}(b,n) \geq I(\lfloor b/2 \rfloor \leq n \leq b) + \sum_{i=max[1,(n-b)]}^{(n-\lfloor b/2 \rfloor)} c(i, \lfloor b/2 \rfloor, b),$$

where $c(i,a,b)$ is the number of compositions of the number i with particles between a and b.

Proof. We have shown that for every partitioning of the elements into chunks with sizes between $\lfloor \frac{b}{2} \rfloor$ and b, there exists at least one corresponding $\mathcal{B}^+$-Tree. Furthermore, we have shown that the number of possibilities of generating such partitions of n elements can be related to the number of compositions of n when applying these boundaries for each particle of the composition, i.e. $n = \sum_{i=1}^{k} a_i$, with $\frac{b}{2} \leq a_i \leq b$. For this, there already exists a formula which leads us to the result given in Theorem 4.6.3. □

[1] See Theorem 4.2.9

Stability: Stability is quite a large problem for this fingerprint, as many DMBSs employ internal routines for performance optimization through index re-organization, thus potentially removing the fingerprint throughout normal operation, undetected and undisposed by the user. Furthermore, attackers could try to remove the fingerprint by issuing large amounts of random operations against the database. While the first problem can be mitigated by using sealed environments, e.g. research servers, where such changes are made impossible, the later one is far harder, if not impossible, to combat. Audit & Control features could be used to detect the heavy use of these random changes, but (i) this can still be a problem when selecting fingerprints that are too close and (ii) this needs to be done at the data recipient side, which is rather unlikely in many real-life scenarios.

Detectability: In order to validate a fingerprint, the validator requires access to the signature of the $\mathcal{B}^+$-Tree, or at least the signature of the fingerprinted tables. This, of course, involves access to more or less the whole database for detection. Again, this effort can be reduced by hiding the fingerprint in smaller portions of the table, thus greatly reducing variability and, even more problematic, stability, as colluding attackers could use their common knowledge in order to detect these parts.

Susceptibility to collusion attacks: In order to construct fingerprints that cannot be detected by a collusion of an arbitrary selection of $m-1$ out of m total data recipients, each possible (sub-)set of recipients must have at least one feature they all share (and thus do not recognize as part of the fingerprint) that differs from the remaining recipients not colluding in the attack. This is equivalent to the notion that every subset of the set of recipients possesses one feature specific to this subset, thus the number of features required is equivalent to the size of the power set of the set of recipients minus one (there is no feature specific to the empty set of colluding recipients), i.e. $2^m - 1$ for m recipients.

While the approach outlined above is feasible in general and simple in nature, it faces some serious limitations when used in real life environments:

- In order to verify it, access to the database needs to be provided, i.e. it is not enough to just see the data, but the very intrinsic structure of the $\mathcal{B}^+$-Tree is required.
- More precisely, it is not possible to detect the data leak based on a single record, or a small subset.
- Any change to the database might result in changes to the fingerprint. This is of course no issue in scenarios, where changes to the database are made impossible, e.g. by only allowing changes in copies of the data on a so-called *research server*, instead of the database itself (see e.g. the concept of the DEXHELPP research server [ZPB14]).

- The generation of a different fingerprint requires the restructuring of the data in the table index, which in many database management systems conforms to re-arranging the whole internal data store.

- Many databases use index re-organization in order to boost performance - the fingerprinted version of the database needs to suppress this behavior in order to thwart the negative effects on the fingerprint.

- In theory, every fingerprint can also be constructed by accident by taking a database copy incorporating another fingerprint and executing insertions and deletions against it. Thus, in order to make this unlikely, the tables in the database need to be large and the distribution of the fingerprints across the fingerprinting space, i.e. the space of possible $\mathcal{B}^{+}$-Trees, needs to be wide.

While this technique seems rather limited in use when considering the imitations outlined above, it does have its uses when considering specific scenarios like sealed analysis environments, where the source database is copied to a virtualized environment holding all analysis tools and intermediate information, where full control over database meta-routines like index re-organization can be enforced.

4.6.2 Fingerprinting using database steganography

While fingerprinting using the $\mathcal{B}^{+}$-Tree signature might be a viable solution in some cases, the stability is lacking in case of subsequent heavy usage of the database, i.e. when the fingerprinted database is afterwards changed heavily through addition and removal of data records from the fingerprinted tables. It must be outlined though that the signature based fingerprint can be applied to all of the tables holding data in the database, thus significantly improving the resilience of the fingerprints. In Section 4.2.3 we already referenced some of our earlier work regarding information hiding in index trees, especially considering manipulating the primary index in order to generate highly stable slack space that cannot be accessed by the database anymore. The approach required the deletion function of the database management system to be changed in order to not add deleted records to the garbage collection for further re-use. In case the data was to be read again, special ”hidden” functions were required to be implemented into the DBMS as well. While surely feasible, this opens the problem that the system needs to run a special version of the DBMS in question, which might make users (or at least administrators) suspicious and introduces additional overhead and programming effort. In our fingerprinting approach the scenario is far more simple: While the original owner of the DBMS will need to have his/her own version of the DBMS running, the data recipient is not required to be in the possession of anything unorthodox, i.e. he/she can run the ”normal” version of the DBMS.

DBMS modifications

The data owner is required to run a DBMS that he/she has full control of at the source code level in order to introduce some changes, as two new functions *fpgenerate (table, fingerprint)* and *fpextract (address, length)* (see below) need to be introduced.

Function fpgenerate is a duplication of the normal deletion routine with the exception that it inserts the parameter *fingerprint* into the table provided in parameter *table* using the normal insertion routine provided by the database and then executes an incomplete deletion that does not invoke the addition of the record to the garbage collection, thus effectively removing the information from the reach of normal database functions. Since the transaction log is stored in external files in MySQL, we can simply use the original INSERT routines, as we won't copy these files to the recipient, thus not leaking information through them and still maintaining transaction safety even in the fingerprinting process. Of course, in order to guarantee this (and maybe in case of other DBMSs), also the INSERT routine might be changed in order to not leave any traces.

1. Insert a new record into the table that contains the fingerprint specifically constructed for the data recipient.
2. Unlink the record from the neighbouring records in the $\mathcal{B}^+$-Tree leaf node, but do not add it to the garbage collection pointer.
3. Check secondary indices for the record - in case of positive retrieval, unlink it from the respective secondary indices.
4. Check the dictionary of the table - in case the record can be retrieved from it, unlink it.
5. Depending on the actual DBMS in use, return the absolute address or the offset in the DBMS file of the fingerprint location

Function fpextract takes an absolute address or an offset in the DBMS file (depending on the actual implementation and the DBMS in question), as well as the length of the information requiring extraction and returns the data at the given address/offset.

Fingerprint generation and verification

One major benefit of this technique is the possibility to introduce an arbitrary amount of fingerprints to the database extract, i.e. the data owner does not need to resort to a single fingerprint, but can spread multiple instances across the database files.

For the construction of a fingerprint, any reasonable technique might be used. In our basic approach we assume that the fingerprint is a salted hash of data identifying the respective data recipient together with the table name in order to thwart detection through duplicate entries. This fingerprint will be stored in the primary key of the table, if suitable, the other columns will be filled with default values, depending on the actual structure of the table. Thus, in real-life implementations, it might be practical to provide standardized functions for fingerprint generation for various types of data. We further assume to use a different salt for each fingerprint in the same table:

$$f_{i,j,k}(u_i, t_j, s_k) := \mathfrak{H}(u_i \oplus t_j \oplus salt_k),$$

with u_i denoting the i-th data recipient, t_j the j-th table in the database and s_k being the k-th entry of the set of salt values. $\mathfrak{H}$ denotes a cryptographic hash function.

For adding the fingerprints, the data owner simply selects the table in the database that should be used as carrier for the fingerprint and generates the fingerprinting information according to the table structure as outlined above. Then the fingerprint is added by invoking the *fpgenerate* function. The resulting address needs to be stored in a table outside the database, together with the recipient ID and the fingerprint.

Detection is straightforward: In case an illegal copy of the database is found, the data owner needs to obtain a full version of it and check all addresses or offsets (again depending on the DBMS) using the function fpextract. The results of fpextract are then checked against the respective fingerprint values in the table holding all fingerprints for all copies of the database in question. Figure 4.16 gives an overview on the proposed approach.

Evaluation and limitations

In this section, we will discuss the benefits and limitations of the oulined approach.

Performance: Generation of the fingerprints is rather simple and fast as we require the calculation of one hash and two xors, as well as one database insert and one deletion per fingerprint, which is negligible compared to the computational effort introduced by requiring to copy all database files. For verification, the worst case scenario can be constructed as follows: Let n be the number of data recipients and $m_i, i = 1, \ldots, n$ the number of fingerprints for the i-th recipient. Then, in the worst case scenario, $\sum_{i=1}^{n} m_i$ fingerprints need to be checked. Given that this only constitutes the extraction of information at a single address/offset and comparison with pre-calculated hashes, this is feasible even for higher numbers of n and $m_i, i = 1, \ldots, n$ respectively.

Variability: The construction of the fingerprint allows for quasi endless variations, depending on the actual size of the carrier database.

Stability: Within the defined side-parameters, i.e. the copying of whole databases on a file basis, the fingerprint is extraordinarily stable: As it is not encoded within the actual data in the database, but in meta-structure, it is possible to change every single data record completely and still being able to tell, who was the original data recipient of the database instance in question. This notion is so strong that it must be discussed though, whether this still constitutes a desirable amount of stability, or is simply too

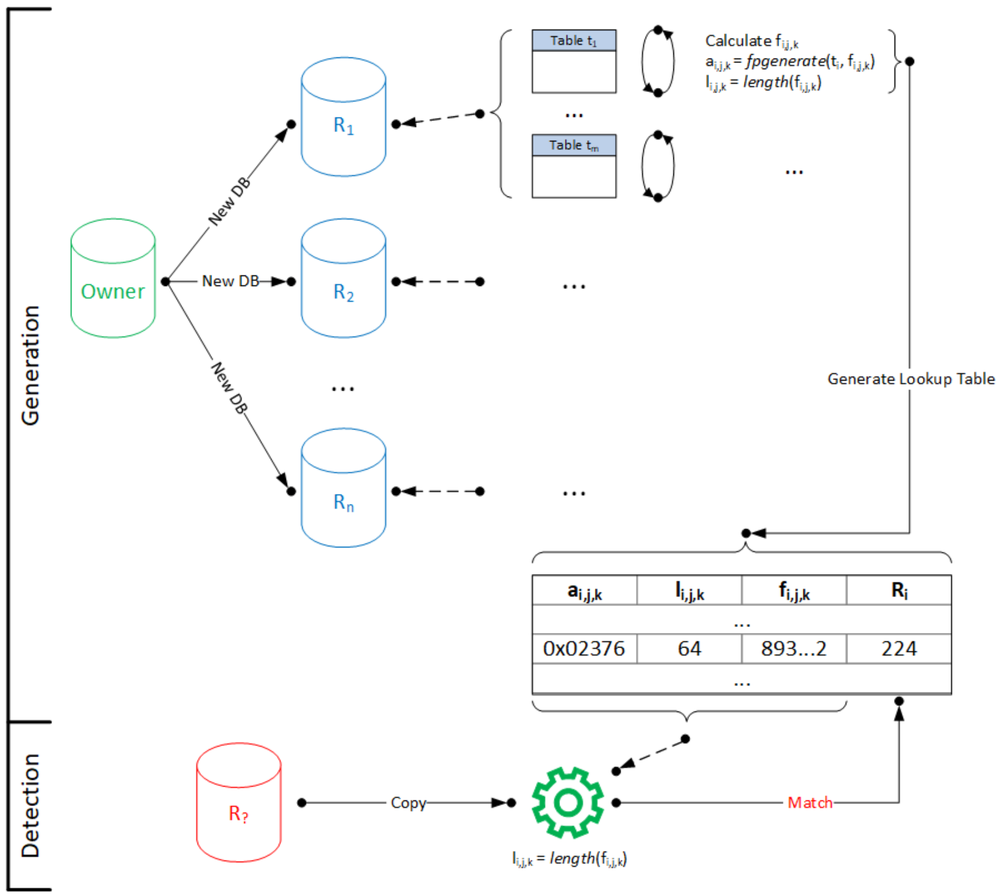

Figure 4.16: Overview on the proposed approach

strong, as the fingerprint will also "detect" databases that, due to not possessing any original information any more, most surely do not constitute a case of data leakage.

Detectability: While the approach outlined in Chapter 3, allows for detection of a data leak by only using one single leaked record, this is completely different for this approach. While it is stronger in the sense that it does not require any leaked record to be present for detecting a copied database, it does require the presence of the original files with their meta-information, most precisely the database slack space.

Limitations: While this technique is practical in order to detect copies of whole databases, even in case all the data in the tables have changed, it does have drawbacks: First, it can easily be circumvented by not copying the database files, but by inserting full table scans into another freshly set up database, or by extracting (part of) the data to files. This major drawback is due to the fact that the fingerprint is not provided within the information that we want to protect itself, but within the meta-structure of the database. Furthermore, attackers could still try to remove the fingerprint by (i) finding all addresses in the database files that are not referenced by any primary index (and are not used by other database assets like procedures, views, secondary indices ...) and (ii) overwriting the information at said addresses by using file carving techniques. While this is feasible from a theoretical point of view, depending on the size of the database, it does involve a lot of effort, know-how on the very internals of the DBMS in question and any error might result in total corruption of the database in question. Furthermore, as a database is an immensely complex system of high dynamics, this threat can only be seen as very theoretical in real life applications.

Susceptibility to collusion attacks: In case two data recipients work together for breaking the fingerprint in two fresh copies of the shared database, they could calculate diffs for the received files and thus extract the data holding the fingerprints. This can be circumvented by selecting the fingerprints in a way that each subset of the m recipients R_i have at least one fingerprint that they all share, but none of the other recipients possesses. This leads to the introduction of at least $2^m - 1$ fingerprints (the empty subset does, of course, not require a fingerprint, thus the "-1", see 4.6.1). While this seems to be infeasible for large numbers of data recipients, again, for attackers working at this level, regarding removal of the fingerprint it would far more easy to simply extract all the information from the tables to a file and load these files into a fresh database instance, thus removing the fingerprint entirely at the cost of not being in the possession of a forensic copy anymore.

CHAPTER 5

Building trust for the expert in the loop

5.1 Introduction and Motivation

While the concept of the ”expert in the loop” [FOJ03], and especially the ”doctor in the loop” [GKH15] seems to be a logical consequence of the application of machine learning technologies and derived knowledge into (medical) science and engineering, one major problem arises: The expert in question is forced to trust the results derived from algorithms based on the authenticity of stored data to a large extent, while being seen as the primary responsible party during information provisioning [WGWW+16]. In case of a medical use-case, during treatment, i.e. the doctor retains responsibility or, in case he/she is involved in the selection of the source data, even gains more, while losing control over the process. With the technology available to tackle large amounts of complicated data in real time through big data techniques, results derived from such processes may even become more uncontrollable (see also the problem of ”explainable AI” e.g. in [DSB17]). This opens up the problem of acceptance of the ”expert in the loop” approach as has been pointed out to us by medical personal [HMK+17]. The question is the trustworthiness of the underlying data and execution chains, especially considering manipulation, e.g. in the aftermath of a wrong treatment, as well as the leakage and unsolicited re-distribution of the gathered information. Besides privacy related issues, data leakage is a serious problem in collaborative data driven research when considering the value of research data. Thus, the issue of unsolicited data extraction needs to be tackled too:

- In case of errors on the expert side, he/she could try to cover the tracks by changing the wrongful data that led to the treatment.
- On the other hand, as the expert is seen as the responsible person that is going to be blamed in case of errors, he/she needs to be ensured that in case the

wrong suggestions came from the system, he/she is protected against legal actions. Safeguarding the doctors is especially important, since the whole concept relies on their participation [KWH16].

- Securing the system against manipulation is especially important in case external users like patients are involved in order to generate trust in the system on their side. Furthermore, as an expert in the loop system in a critical field like medical support would logically constitute an ICT-critical infrastructure, data manipulation could be a possible threat in scenarios of cyber-crime [Fue17] (e.g. illegal drug acquisition) or cyber-terrorism (e.g. by seeding distrust towards entities in the national health system).
- In addition, medical data is often of high value to the collecting agency/hospital/lab, thus read access needs to be monitored in order to thwart data leakage.

Thus, in order to mitigate these risks for the overall concept, manipulations in the underlying database need to be detected, as well as control over the information entered by the expert needs to be safeguarded against subsequent manipulation and unsolicited propagation. This also includes the manipulation-secure logging of execution chains of enrichments, as well as analytics algorithms and workflows. The contributions in this chapter can be summarized as follows:

- We provide a model of the ”expert in the loop concept” including an abstract architecture of its entities with respect to security. Hereby we exemplarily focus on the medical sector with the ”doctor in the loop” concept, as this is a field with a large amount of sensitive information, quick decision processes and high political and societal relevance.
- Attack scenarios and attacker models against this approach are devised.
- Based on these models, strategies for mitigation are defined.
- Finally, we propose two methods for thwarting unsolicited data leakage, one based on strategies outlined in Section 5.2.2, the other based on principles of database steganography as outlined in Section 4.2.3.

It must be noted that the results of this chapter have been published in the form of papers. In order not to thwart quality control introduced by the peer review process they were taken verbatim from the corresponding peer-reviewed publications.

5.2 Background

5.2.1 Experts in the loop and medical databases

Interactive machine learning has been a very popular topic in research throughout recent years, especially considering the medical domain with its vast amount of applications in the sectors of diagnosis, as well as treatment. In [BG12] the authors provide a comprehensive comparison on different training algorithms for supervised machine learning, mainly focusing on the aspects of speed, accuracy and scalability. Focusing on the pure algorithmic level, matters of security and especially data protection are left out. Alongside this work, [WZA$^+$16] discusses how computer-assisted presentation of case data can help experts to infer machine-implementable rules for case definitions in electronic health records, which are valuable resources for analyzing issues and shortcomings of health systems [NHW$^+$10, ZPB14], as well as co-morbidities [ARG$^+$10]. Here, the authors apply an expert in the loop approach and demonstrate the usefulness of their techniques with a practical medical example on acute liver dysfunction (ALD). Furthermore, a multitude of applied work that utilizes data mining and (interactive) machine learning in medical research has been proposed in the recent past like [SASS11] for the prediction of heart diseases, or [WS12] discussing possible applications in radiology. In [JJB12], the authors consider the benefit, but also the challenges when mining electronic health records (EHRs).

Regarding sharing of information in medical databases, in [KLOT12] the authors discuss the effects of the American Recovery and Reinvestment Act of 2009 on medical research [Ste09], namely the production of large-scale databases of patient data allowing better and larger studies. While acknowledging the need for removing protected patient information, the paper focusses more on the benefits of data sharing without acknowledging malicious intent. Various work exists on the problem of de-anonymization of health records (e.g. [KNJ$^+$12]), which is thus not in the focus of this work, since the architecture proposed in this work allows the application of various kinds of additional countermeasures in order to protect patient privacy. In their work [WSB$^+$12] the authors propose a practical framework under development that makes recent developments in the realm of machine learning accessible to practitioners. This is based on their observation that despite the fundamental developments on the theoretical and conceptual side, adoption of machine learning techniques by practitioners has been low [CNCC15]. Still, while providing many practical considerations, the topic of security and protection of sensitive data was only touched slightly in this concept.

The problem of securing infrastructures relying on human behaviour has been discussed throughout the last decade and more, being one of the very fundamental problems for computer security [Smi03]. The problem is often related to the issues of awareness [Lup08] or missing usability in security [CGM$^+$11], as well as other subtopics, also including the sharing of data between different entities [KHS$^+$14] where each participant has his/her own agenda in dealing with the supplied data. It has to be kept in mind that research data forms a very valuable resource for many research laboratories. This is also often related to the issues of providing health related information to other clinicians [Ran05] or to automated systems [WJA06], where the original owner of the data loses control over the further dissemination. In [LSC$^+$12], the authors identify security and privacy issues as one of the major open research issues in the development of medical cyber-physical systems (MCPS), especially considering that interoperability capabilities will open up new attack surfaces that can be misused to harm patients by e.g. manipulation of data or direct access to critical system components.

Another topic that frequently comes up when discussing privacy and (medical) data analysis is the notion of *privacy by design*, a term describing a fundamental basic principle in constructing software and systems: Privacy is not an add-on, but a fundamental architectural principle. First used by Langheinrich [Lan01], it is based on ideas harkening back to the 1960s and especially to notions regarding fair information practices from the 1974 US Privacy Act which were later on codified in OECD Guidelines [fECoC80] and, in a more concrete form, in the European Union's Directive 95/46/EC on the protection of individuals with regard to the processing of personal data and on the free movement of such data [Dir95]. The framework in [Lan01] was later on refined into 7 foundational principles by Ann Cavoukian [C+09]. These still rather abstract design principles have been refined in subsequent years in order to provide more practical guidelines for actually delivering on the promises of privacy by design: In [Cav12] the authors provide some practical examples on how they would introduce Privacy by Design in certain environments, still, the approaches are rather high-level, while in [Hoe14] the authors provide a selection of generalized strategies. Cavoukian, amongst others, has also provided some more insight on the issues of Privacy by Design with respect to data driven environments in [CJ12], still, as recent research has pointed out, especially the most important principle of "data minimization" can reduce the quality of machine learning algorithms [MKWH16]. Most prominently, ENISA has put forward some implementation strategies for Privacy by Design based on [DDFH+15], which, while still rather general, can be used as starting points for our architecture. Summarized, while literature regarding general principles and even technological fundamentals for privacy by design is plentiful, there is still some research to be done in order to adopt the principle in practical data driven environments.

5.2.2 Chained Witnesses

The term "chained witnesses" was coined in [FKKW14], where the authors proposed a technique for securing internal mechanisms of databases against manipulation. The main advantage of this approach over the multitude of approaches described in the literature was resilience against an attacker model that included the database administrator as possible adversary. While this requires discussion in most real-life systems where the database administrator is seen as a trusted entity, this is especially interesting in the "doctor in the loop" and other highly sensitive expert in the loop systems.

The main principle of this approach lies in appending a so-called *witness* for each transaction that is issued against the database to the internal logging mechanisms: The database storing the information is considered as untrusted, furthermore, even file system administrator rights are assumed for the attacker.

Definition 5.2.1. *Let D_i be the i^{th} data record written to the database at time t_i.*

Furthermore, we assume that $\mathcal{H}$ is a cryptographically secure one-way hash function, $\mathcal{T}$ is a trusted third party and $\mathcal{R}$ is a secure pseudo random number generator (PRNG) and r_i is the result of its i^{th} iteration. The witness for transaction D_i is calculated as

$$w_i = \mathcal{H}(w_{i-1}||D_i||t_i||r_i) = \mathcal{H}(w_{i-1}||D_i||t_i||\mathcal{R}(r_{i-1}))$$

with $||$ denoting string concatenation. The tuple (t_i, w_i) is then called the signature *of the record D_i. In order to start the hash-chain, an initialization phase is required: A trusted third party $\mathcal{T}$ selects a random number s as seed for the PRNG and thus generates r_0 by using the PRNG on s. Furthermore, the initial witness is defined as $w_0 := \mathcal{H}(r_0)$.*

Due to the definition of the witnesses as chained hashes, any changes in older data sets lead to cascading changes in all subsequent witnesses (Figure 5.1 shows the chaining). For the verification, the data of the protected internal logging mechanisms is executed against an old trusted backup under the premise of $\mathcal{T}$ and compared to the investigated database instance. This also works, when reverting the whole database to an old state: As the PRNG is seeded with a seed unknown to the attacker, every iteration changes the state of the PRNG, which cannot be calculated backwards (this is ensured by the criteria for the hash function and the PRNG in [FKKW14]). Thus, in the case of reverting, the states of the missing entries can be detected easily. In [FKKW14] the authors propose several mechanisms for achieving this kind of manipulation security in real-life environments, especially targeting internal database mechanisms for providing rollbacks (so-called *transaction logs*[1]). The transaction log is stored directly by the database environment as an internal component. Figure 5.1 also demonstrates, how the log entry is extended in order to store the witness, more details on the exact specification of the log entries can be found in [FKKW14]. Furthermore, the database management system (DBMS) must be modified in a way to provide the calculation of the respective witness as an atomic action, invisible to and un-interruptable by the database administrator, i.e. the mechanism for writing the transaction log needs to be modified directly in the DBMS-code in order to fetch the random numbers r_i and calculate the witness immediately, without leaking r_i to the administrator. As shown in [FKKW14] the implementation of such a process can be done for MySQL, furthermore, the authors pointed out solutions for closed-source DBMSs based on the *database replication logs*.

5.2.3 The doctor in the loop

The concept of the "doctor in the loop" is an extension of the increasingly frequent use of knowledge discovery for the enhancement of medical treatments together with the "human in the loop" concept: The expert knowledge of the doctor is incorporated into "intelligent" systems (e.g. using interactive machine learning) and enriched with additional information and expertise. Using machine-learning algorithms, medical knowledge and

[1] It must be noted that the term "logs" is slightly misleading, since these are not human readable log files, but internal mechanisms for ensuring transaction safety.

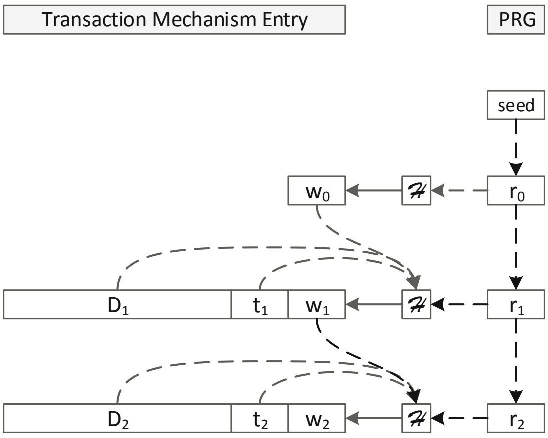

Figure 5.1: Construction and storage of the chained witnesses.

optimal treatments are identified. This knowledge is then fed back to the doctor to assist him/her (see Figure 5.2).

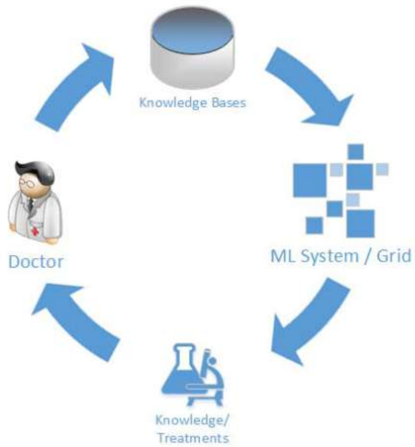

Figure 5.2: The doctor in the loop.

While general techniques regarding data driven research have their own problems with respect to privacy protection (see e.g. [HSKW13]), an additional major problem for the doctor in the loop lies in guaranteeing the trustworthiness of the data provided by other entities and by analysis workflows.

5.3 The Approach

The approach outlined in this section is based on the generic concept of the "doctor in the loop" as described in Section 5.2.3. In order to motivate the chaining approach, we will define the entities and their relations, including the chaining mechanism.

5.3.1 Entities and Relations

For our analysis, we define a more specific model for the doctor in the loop. Figure 5.3 gives an overview on the components:

- The **Doctor**, who is the main expert in the cycle, collects data from patients, including their reactions to individual treatments and potential other effects. Furthermore, he/she provides additional knowledge from his/her experience and sanity-checks results. All data he/she provides to the system is sent to the Knowledge Base, which also provides him/her with the relevant feedback. In this basic approach we resort to a single doctor entity, which is rather uncommon in real-life environments, the required extensions to the chaining for including multiple doctor entities can be found in Section 5.3.3.
- The **Knowledge Base** provides the store for the data and all results of workflows and external resources, as well as the only means for communication between the doctor and the other entities. This entity is the primary target for our chained witnesses approach, since all data that is transferred between the relevant entities for the ”doctor in the loop” approach utilizes it. The knowledge base may also host stored procedures for the analysis of the data, i.e. parts of the ML-grid are implemented as stored procedures inside the knowledge base.
- The **Grid** serves as a generic model for a machine learning / reasoning structure that takes input data and returns results using analytics algorithms. The grid may be implemented as external mediation tool, as well as in the form of internal stored procedures inside the knowledge base. In our concept, the exact definition of the grid will be kept on an abstract level, since providing manipulation security will be done on the side of the underlying database of the knowledge base.
- **Interfaces** from other entities to the knowledge base are logged by the underlying DBMS. This includes all transactions changing data or structures in the database, as well as the change and invocation of stored procedures that may implement part of the grid.
- The entity **Medical Research** denotes external knowledge bases that serve as external data input to the grid, or to the knowledge base.
- **ML Research** provides the grid with new algorithms for the analysis of the data stored in the knowledge base.

5.3.2 Interaction and Chaining

For the abstract approach we only consider a general scenario where a generic data receiving decision maker $\mathfrak{M}$ (e.g. the doctor) sends data to a generic data store $\mathfrak{S}$ (e.g. the knowledge base). Furthermore, an entity $\mathfrak{P}$, the data provider, operates on the same

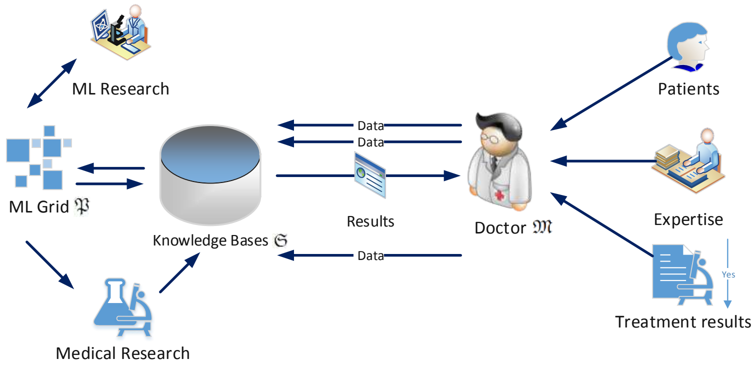

Figure 5.3: Entities and Relations.

database and delivers a result to $\mathfrak{S}$. $\mathfrak{M}$ takes a result (e.g. a treatment) based on the data in the database and returns additional information, especially on the reaction of the patient and other (side-)effects. Furthermore, $\mathfrak{M}$ controls the results stored in $\mathfrak{S}$ with respect to sanity-checks based on his/her background knowledge and issues respective corrections to $\mathfrak{S}$ that are subsequently used by $\mathfrak{P}$. From a security point of view, this especially implies that the exact order of the transactions with respect to the knowledge base is of vital importance in order to guarantee authenticity.

Data Provider: The model of $\mathfrak{P}$ is selected to be as generic as possible and covers all single data providing entities except the decision maker. This especially includes all parts of the grid, as well as additional data sources with respect to Section 5.3.1. Due to the assumption that $\mathfrak{P}$ might be some proprietary entity, incorporating additional mechanisms for controlling the decision provider(s) is not reasonable. Furthermore, $\mathfrak{P}$ might in reality consist of several different entities (internal stored procedures and external workflow engines), i.e. $\mathfrak{M}$ might provide data to and receive information from several different data providers $\mathfrak{P}_i, i \in \mathbb{N}$. Thus, $\mathfrak{P}$ only needs to fulfill the following prerequisites:

1. All results are written to $\mathfrak{S}$, there is no additional side channel to $\mathfrak{M}$, i.e. $\mathfrak{M}$ and $\mathfrak{P}$ are independent.
2. Everything sent to $\mathfrak{S}$ by $\mathfrak{P}$ is signed using state of the art cryptographic technologies and is therefore assumed to be unforgeable.

Especially requirement two seems to be strong, still this is a standard requirement in many current communication protocols.

Data Store: The data store possesses an internal table structure for storing all collected data, invoked enrichment algorithms, as well as the received data, protected with the chained witnesses approach: For each entry in the transaction log D_i, the respective signature (t_i, w_i) is stored (see [FKKW14]). It must be kept in mind that the only connection between two entries D_i and D_j lies in their timely succession, i.e. all changes in all tables are stored in the same transaction mechanism, ordered by the time of entering t_i. This also holds true in case of several decision makers $\mathfrak{M}_i$. In the **setup phase**, the initialization is done by a trusted third party $\mathfrak{T}$ (see below). We furthermore assume that the data store is run independently from the underlying physical server, i.e. $\mathfrak{S}$ possess administrator privileges over all tables, as well as full access to the file system for enrichment and processing of incoming and outgoing data, as well as for restructuring the database layout (tables, views ...), including full control over log settings. Still, it does not possess root privileges on the underlying machine, which is run by $\mathfrak{T}$ or another trusted entity. In addition, the data store frequently sends a backup to $\mathfrak{T}$, which is validated as shown below. The newly validated database image iteratively serves as the new base point for the next validation cycle.

Decision Maker: The decision maker $\mathfrak{M}$ is independent from the data store, i.e. it does not have any control over $\mathfrak{S}$. Furthermore, it is also independent from all data providers (see there). In this approach, we assume that the decision maker is honest in general (see data insertion).

Trusted Third Party: The trusted third party $\mathfrak{T}$ controls and manages the random values needed in the chained witnesses approach for the data store. During the setup phase, a new random seed s is selected and the first random value r_0 is generated using the cryptographically secure pseudo random number generator (PRNG). Furthermore, the first witness $w_0 = \mathfrak{H}(r_0)$ is sent to $\mathfrak{M}$. Additionally, $\mathfrak{T}$ can be the entity responsible for running the physical server for $\mathfrak{S}$, including root privileges. While $\mathfrak{T}$ is thus in a very powerful position, it must be independent from all other entities, especially from all data providing parties, thus possessing no interest in data manipulation. Furthermore, interaction between $\mathfrak{S}$ and $\mathfrak{T}$ is limited to the setup phase and during the verification of authenticity.

Network Providers: The network provider is responsible for enabling the communication between the data provider and the decision maker. We assume that all traffic is protected by end-to-end encryption against eavesdropping, other attacks by a malicious network provider, e.g. denial of service, are not inside the scope of this paper. This also holds true for the underlying public key infrastructure that is needed in order to facilitate the encrypted communication.

Data Insertion: The decision maker is modeled to receive data from outside machine based systems, especially by the patients during personal consultation. As outlined later in the attacker model in Section 5.4.1, we assume that the decision maker is in principle honest, i.e. at the time of consultation, no harm towards the patient is intended from his/her side. This also means that the data entered into the database is honest at the time of insertion. All data received by the patients is immediately stored to $\mathfrak{S}$, without stalling for later malicious behaviour.

Verification of Authenticity: In the verification step, $\mathfrak{T}$ extracts the internal transaction logs (this is possible using a method provided in [FKS+12]) and uses a trusted backup as starting point for consecutive execution of the log entries, thus verifying the witness for each transaction by using the secret initialisation vector s and the PRNG. The first encountered invalid witness provides the position of a manipulation of the log. Furthermore, the result of the verification is compared to the current database, thus being able to uncover changes done directly in the underlying file system.

5.3.3 Multiple decision makers

The basic approach is very limited with respect to the human interfaces, i.e. it only considers a single decision maker $\mathfrak{M}$ (e.g. one doctor). This approach, while reasonable for demonstrating the fundamental chaining mechanism and providing a fundamental security analysis, has a very large drawback. In a real life environments, e.g. a research lab for biomedical research, several decision makers $\mathfrak{M}_i$ will be involved for different reasons: The advice of experts on different medical fields might be needed, as well as from different experts from the same field in order to reduce errors and enhance accuracy (see Figure 5.4).

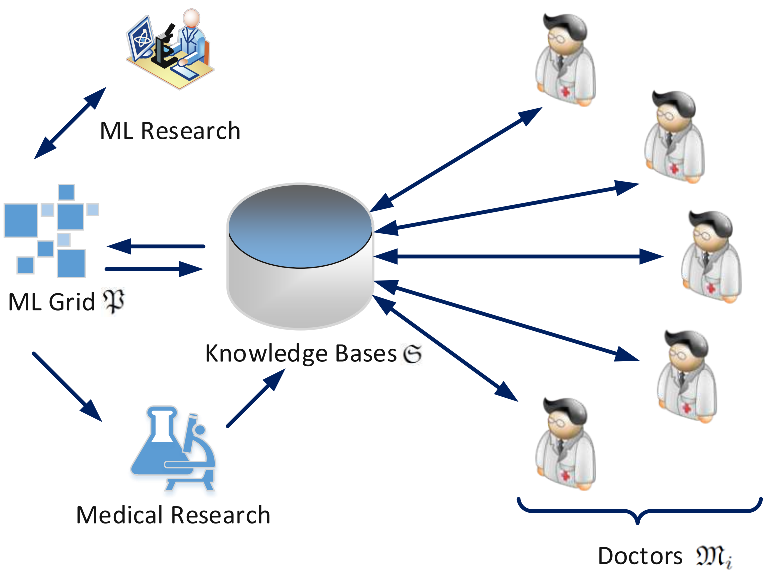

Figure 5.4: Multiple doctors in the loop.

We assume without loss of generality that all $\mathfrak{M}_i$ use the same interfaces to the database, being capable of changing the same tables, attributes and even records. This is done in order to formulate the problem of attribution of changes as general as possible and allowing the greatest amount of inferences between the decision makers, also including malicious collaboration. While secrecy of the data entered by the $\mathfrak{M}_i$ is not an issue in this architecture, vital importance is put on the aspects that no decision maker (or any combination of $\mathfrak{M}_i$) is capable of (i) hiding his/her/their changes, (ii) planting changes as if they were issued by someone else or (iii) changing the logged order of changes with respect to the application order (i.e. a transaction T_i is logged after T_j with $i < j$).

In order to implement this chaining into the architecture, the following pre-requisites are assumed:

1. All decision makers $\mathfrak{M}_i$ use different user profiles for accessing the database.
2. The attribution of the statement to the respective user issuing it can be done on the DBMS-internal level, i.e. the transaction mechanism has access to the actual user issuing the request. This is typically available for DBMSs like MySQL with InnoDB in order to provide access control. This also means that the database administrator cannot control the attribution of statements to the log.
3. The same holds true for the database replication mechanism in case of a closed-source DBMS, see Section 5.3.4.

The calculation of the chained witnesses can then be extended to account for the attribution mechanism to specific users by including the user-id $\mathfrak{M}_i$ for the i-th user into the chaining (see also Figure 5.5). The witness for transaction D_i is calculated as

$$\begin{aligned} w_i &= \mathcal{H}(w_{i-1}||D_i||t_i||\mathfrak{M}_i||r_i) \\ &= \mathcal{H}(w_{i-1}||D_i||t_i||\mathfrak{M}_i||\mathcal{R}(r_{i-1})) \end{aligned}$$

In MySQL the user-id $\mathfrak{M}_i$ can easily be included by a very simple rewrite in the logging routines: Since the InnoDB storage engine provides ACID-compliance, the user is known to the transaction mechanism, the linking is thus quite simple to implement.

5.3.4 Adaption to closed-source environments

Unfortunately for the original approach, a lot of systems typically accessed by doctors and put to work in medical research environments are not fully open source. Especially regarding high-performance database management systems, the major vendors like Teradata, IBM (DB2) and ORACLE stay closed-source. Thus, contrary to the scenario in case of using open source alternatives, changes directly to the transaction mechanism are neither feasible nor practical. In order to employ the original concept of the chained witnesses for the doctor in the loop, internal mechanisms that have a passive, read-only interface to the outside world need to be employed. As outlined in [FKKW14], the database replication mechanism is a suitable choice for supporting the approach: The task of this mechanism lies in mirroring a database instance, typically referred to as *master instance* to several so-called *slaves*, instances that are exact copies of the original

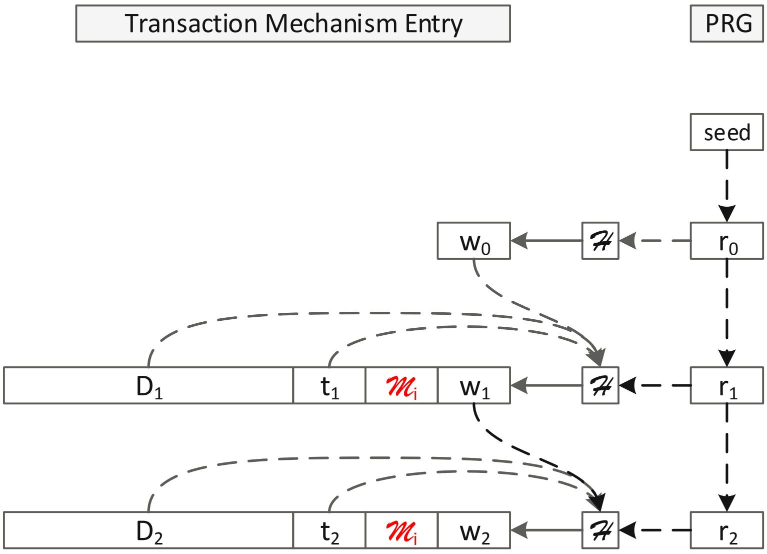

Figure 5.5: Modified chained witnesses for multiple $\mathfrak{M}_i$.

database. Typically this mechanism is employed in order to generate redundant copies for so-called *hot backups*, allowing the possibility to switch from the master instance to a slave instance transparently, e.g. in the case of defects on the master instance. Thus, the information that needs to be stored in the slave instances is typically not limited to the data sets alone, but includes vital information on the structure, large objects and even metadata like session information and timestamps. It is thus possible to adapt the chained witnesses approach to this interface, as shown in Figure 5.6.

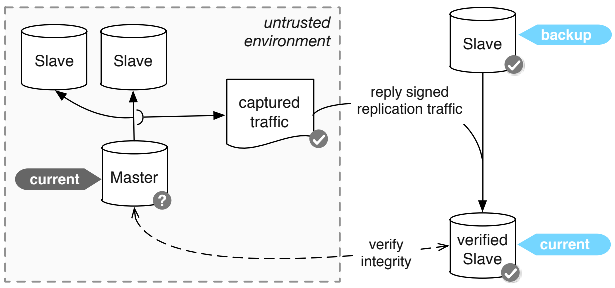

Figure 5.6: Verification with replication (taken from [FKKW14]).

In this approach, the data stream from the master instance to the slave instances is captured and the witnesses are added right before sending the stream to the slaves. This requires that the whole signing procedure is added in an extra node right on the transport layer. Since the data stream of the data replication mechanism is typically not documented for many closed-source DBMSs and might change between releases, the information is not decoded but the whole stream is simply split into blocks of a pre-defined length. These blocks are then chained using the original approach, still, due to the need to control an active slave instance for verification purposes, the trusted third party $\mathfrak{T}$ is involved far more when compared to the original approach using the transaction mechanism:

- Since the replication mechanism extracts all changes to an actual separate database instance instead of some internal mechanism stored on the same machine, attacks by the root of the master instance need to take place when the data is written, i.e. even for an attacker with root privileges it is not possible to later on alter the data that was submitted to the slave instance under the control of $\mathfrak{T}$.

- Internal transaction mechanisms are typically limited in terms of disk space that is provided for them and older entries are removed after a while. This is no problem per se in the original approach, since the chaining still needs to be valid and regular auditing of old entries allows (i) the removal of such old entries without implications to the verifiability and (ii) reduce the workload needed to be carried out in case of a check for manipulations. Using the replication mechanism allows for a much more transparent implementation, where no changes are deleted and the whole history of the original database can be rolled back.

- Since the replication mechanism is made for reliable mirroring of databases, it can be accessed very easily and without tinkering with the DBMS in place. This is not only interesting in case of closed-source approaches, but also reduces the implementation overhead in case of switching to a new version of the same DBMS. While the transaction mechanism is an internal function that is typically undocumented and can be altered between even minor version changes, the interfaces to the replication mechanism will stay much more stable and changes will be well-documented.

The main drawback of this modification compared to the original approach lies in the effort on the side of the trusted third party $\mathfrak{T}$: Instead of storing a random seed and only having to become active in case of manipulation detection or maintenance, $\mathfrak{T}$ has to operate a full instance of the DBMS with at least the same performance capabilities with respect to data insertion. Furthermore, in case of commercial platforms, this also includes additional costs for licenses, as well as trained personnel for operation. This also hinders outsourcing this task to a central authority that provides the services of a trusted third party to several different operators.

5.3.5 Thwarting information leakage

The original approach on using witnesses for securing database content as outlined in this section has one severe limitation regarding data leak detection and the detection of unsolicited information extraction, as it was solely concerned with thwarting manipulation of data, especially by high-profile attackers possessing administrator privileges. This is certainly of the utmost importance for any data driven process using sensitive (or at least valuable) information, as it guarantees the integrity of the information stored in the database to some extent. There are many foreseeable scenarios, where attacks on the integrity of data assets are an interesting factor, also including methods like specialized malware that does not need to infect the machinery on the premises, but is already implemented at the system level, thus being available at each update and running with administrator privileges (see e.g. [TBO+12] for APTs in industrial environments). Still, the original approach does not thwart any information theft, as read access to the database is not integrated into the transaction mechanisms [FHMW10a]. It is rather simple to deal with this issue in case of normal database users (as it was also the case with respect to data manipulation), as there are many logging and tracing mechanisms in place, still, these are under the direct control of the database administrator. Challenging the trustworthiness of the database administrator is fundamental for providing lasting security in such an environment: Many attacks launched against companies come from the inside [HCCY13] and a disgruntled or simply paid database administrator is surely a valid concern [CN12]. In addition, many attacks from the inside are the result of a malware infection of the user in question, ranging from simple mass spam infection to targeted attacks explicitly designed against this very system[SE13]. Thus, we need an equivalent technique regarding information retrieval comparable to the witness for read access. We will thus look into two different approaches for storing this read information, namely by

1. making read access to the database show up in the transaction mechanisms as well.
2. using database steganographic methods for storing read and write information into the database.

Both techniques can also be applied in the context of dedicated analysis platforms like the DEXHELPP research server [ZPB14], where data is specifically provided to specialized virtualized environments already containing all the analytics tools required for the task at hand.

For the first approach, i.e. letting SELECT-statements issued towards the database show up in the transaction log, we present two approaches: While the naive approach would follow the general approach on changing the logging strategy by modifying the code of the SELECT-statement in the underlying DBMS in a way it writes its content to the log, this poses a major problem: Currently in InnoDB there exists no log block structure defined for SELECT-statements and other blocks are not suitable to hold such a transaction, since this would confuse the rollback and crash-recovery mechanisms. The actual definition of the log is far more complex than just writing a single data block per transaction to the files and the definition of the transaction in question is spread over several different supporting blocks [FHMW10b]. While it is of course possible to change this and add a new block type with supporting block structure into the log, these changes would need to be implemented consistently throughout all internal database mechanisms and would require updates every time the basic InnoDB code is changed. Thus, we do not consider this strategy to be helpful, especially as it does not work automatically for closed source DBMSs, as this would mean changing the replication log functions as well. But, a simpler way of introducing the content of SELECT-statements into the transaction and replication logs can be defined straight forward: We could change the function that handles the SELECT-statement just a bit in order for it to carry out an additional INSERT into a pre-defined table, thus writing the content of the SELECT-statement to the database. This would result in automatic writing of the statement into the transaction log. Of course it is possible for the database administrator to change the content of this table, as it offers no additional protection over any other table in the database, thus the table itself does not have much value in manipulation detection, but since an INSERT-statement was triggered, the relevant part of the data is also inserted into the chaining of the witnesses. Since the data in this table is of no relevance, it can be deleted right after the INSERT is finished. The major issue with this technique lies in the vast amount of additional information inserted into the log files, thus exceeding the typically reserved space quite soon. Since the files holding the log information are structured like ring storage [FHMW10b], this would result in the first parts to be overwritten. While this does not pose a problem per se for the witness algorithm, it vastly increases the efforts required for manipulation detection. The structure of the additional table that is required to hold the SELECT information is extremely simple: It just requires a varchar-field that holds the statement.

The other approach for detecting data extraction/theft that we want to present in this section relies on the concept of database steganography as described in Section 4.2.3. The basic idea is to generate a suitable amount of database slack space right in the beginning at the setup of the database by providing an additional DELETE-routine that, in addition to deleting information like the original DELETE-function, also generates slack space by saving the deleted space from the garbage collection, much like the fpgenerate function in Section 4.6.2. In order to thwart file carving we will spread the slack space over a larger region of the database files by generating a large amount of data throughout the setup phase of the database and delete it, most of it by using the standard deletion route, and some with the help of the slack space generating deletion command. The deletion will not be done right after the insertion, but at random time in the INSERT/DELETE cycle, thus, the database will be filled up at first and then random space will be allocated to the slack space, thus spreading it over large parts of the underlying database files.

The structure of the slack space looks as follows: The space starts at a predefined address that acts like the garbage collector by pointing to the next free address. In addition to the pointer, it contains the following data: (i) Size of the slack space, (ii) remaining free space in the slack and (iii) a pointer to the latest entry in, as well as (iv) the last address of, the slack. This information is required in order to extend the slack space when needed. Another modification to the database environment that needs to be done at compile time (like the implementation of the new DELETE method) is a way of writing information to and retrieving it from the slack space. Writing should take place at every SELECT statement, thus we define a log block as the concatenation of a timestamp, an offset, user credentials and the statement itself:

$$block := timestamp||userID||offset||statement||witness$$

The timestamp contains the exact time of the issuing of the SELECT-statement, the userID is required in order to store the user issuing the statement. It is retrieved from the internal access control system that is invoked by each SELECT-statement in order to check, whether a user possesses the credentials to actually access the information declared by the SELECT-statement. This information is retrieved inside the search routines from database internals and cannot be changed by the administrator for obvious reasons. The offset simply contains the length of the statement that needs to be stored. In order to additionally secure the data inside the database slack, we will apply the chained witnesses approach as we have done for the transaction mechanism. Before adding the statement to the database slack, the routine inside the SELECT-method calculates the required space and checks, whether the slack space is still big enough. If not, the extension routine is called, which basically creates some arbitrary random data and deletes it using the slack-generating DELETE-routine and adds the required pointers from the previously last address of the slack. In any case, the routine jumps to the latest entry in the slack and adds the information in the subsequent slack space. Afterwards, it changes the last entry address and the remaining space counter in the slack space header. In order to mitigate the issue of having to split statements over several address, we assume that at generation time each data particle generated for slack generation was larger than the longest SELECT statement. In case this does not hold true, the data needs to be split over several addresses, which does not pose any problem as the data length can be calculated using the offset, thus making perfect use of all slack space available. For reading the log inside the slack space, another routine needs to be provided. The definition is quite simple, the routine takes the address of the slack space start as an input parameter and generates the list of all SELECT-statements executed against the database. Since we know the address of the latest entry based on the respective pointer in the slack space, the enumeration can either be done forward or backward in time. For the later, it must be noted though that the witnesses cannot be verified backwards.

5.4 Evaluation

5.4.1 Attacker Models and Attack Vectors

In this section we give a description of the attacker models and attack vectors with respect to the assets of the ”doctor in the loop” approach.

Data Provider and Decision Maker: Both entities could have the interest of manipulating data on the data store in case of cover-ups, e.g. manipulating previously delivered incorrect data. The main attack vector of the decision maker lies in updating data on the data store, provided either by itself, or as result (treatment) data from the data provider. The data provider possesses the same attack vectors, but in addition, he/she might try to manipulate and/or re-execute stored procedures that operate on the data in order to cover up wrong results. In order to keep the concept as simple and strong as possible, we assume that there is no dedicated secure application controlling access to and from the database by the entities, i.e. the entities write their changes directly into the data store. This is especially important in order to be secure against SQL-injections or related attacks by default.

Data Store: The data store itself is an important entity in the overall concept, since it serves as the central data exchange platform and is thus vital for providing trust in the "doctor in the loop" concept. The database administrator controls all access to the database, including the possibility to undo logs, as well as change arbitrary data and structures. Furthermore, not only the database itself, but also the underlying file system, can be of interest for an attacker: As outlined in related work [FKS+12], file carving techniques can be used in order to retrieve or manipulate data by directly accessing the database files on the file system. In this evaluation, we thus concentrate on these two fundamental attack vectors:

- The **Database Administrator (DBA)** possesses administrator privileges on the database itself, including the ability to change logging routines and user rights, as well as read access to the underlying file system.
- The **File System Administrator (FSA)** can modify arbitrary files on the server, including the files belonging to the database, as well as the OS (system) logs. He has no access to the database query interface though.

Neither of the two attackers possesses root privileges on the respective database server, thus has no access to the RAM of the machine the database is running on and therefore cannot read the values r_i from memory or alter them during execution.

5.4.2 Security Evaluation

In this section we will analyze the respective assets that could be targeted by the attackers modeled in the previous section.

Manipulation through the database (All except FSA): Both, the data provider, as well as the decision maker could be interested in re-issuing incorrectly entered data. In case they act with their own privileges, i.e. as data provider or decision maker, every modification of data is stored in the internal logs, together with the respective timestamp of the change, making it easily detectable. In case the attacker possesses administrator privileges on the DBMS (DBA), the internal log mechanisms are under the full control of the attacker, except for the chaining: Since the attacker still does not possess root privileges on the server, it is impossible for him/her to read the value r_i from the RAM, which is then used in the generation of the witness with $\mathfrak{H}$. Since $\mathfrak{H}$ is a cryptographic hash function, when given $h := \mathfrak{H}(\mathfrak{h}')$, h' cannot be deduced realistically from h. This also holds true for the closed-source approach, the hashing is still done the same way, the main difference lies in the fact that the chained blocks are not aligned with logical borders of statements, but use a fixed block size.

Targeting stored procedures (DBA): The database administrator can execute and modify every stored procedure on any stored data set. Still, in case executions change any data in any table in the database, the changes are logged in order to retain transaction safety. Since data replication is used to provide hot backups, these changes are immediately sent to the slave nodes.

Manipulation through database files (FSA): The file system administrator could bypass all logging mechanism by manipulating data directly in the underlying database files. This includes the transaction log and other rollback mechanisms, which have to be invoked by the DBMS. Using the witnesses, these changes remain detectable, since the resulting database in the verification step will be different from the one currently on the server. Still, the attacker could insert data via the file system and remove it right before the validation, making this manipulation undetectable. As a countermeasure, the validation process should be done frequently at random times. Furthermore, we propose to use the chained witnesses approach with respect to special logs containing checksums of the database files. This attack can be omitted in case of the closed-source approach, since the attacker does not have access to the database files of the slave instance under the ownership of $\mathfrak{T}$.

Manipulation of the DBMS: The attacker could remove the chained witnesses from the source code of the DBMS and install a recompiled version. While this is possible, this action would require root privileges on the server. Furthermore, modifications on the binary could be easily detected via frequent comparison of checksums of the respective code to the originally issued version.

Modification of the transaction mechanism (DBA): The authenticity of the information in the transaction mechanism/log is protected by the chained witnesses approach, so every manipulation can be detected under the given attacker model and the manipulated record can be identified. This could only be circumvented by deleting the whole log, which itself is a highly obvious manipulation pointing to the database administrator.

Combined attackers: In the above examples we split the attacker between the DBA and the FSA, still, the resilience of the approach is retained even in case the attacker possesses both privileges. This can be directly inferred from this section, since the chaining is done on DBMS level, without the involvement of either, the DBA or the FSA.

5.4.3 Limitations

The limitations of the proposed approach can mainly be attributed to limitations of the original chained witnesses approach, especially regarding the lifetime of the internal transaction logs and problems related to an attacker possessing root privileges: Since such an attacker is expected to have full control over every aspect of the system, the following additional attack vectors are available that the approach is vulnerable to:

- **Modification of the DBMS:** In case of root privileges, the attacker can simply modify the DBMS in such a way as to either remove the chained logging altogether (which is rather obvious and easy to spot), or by modify the chaining in order to give exclusive powers of manipulation: For example, the attacker could replace the hash function $\mathfrak{H}$ with a weaker version, or take control over the generation of the random numbers r_i, both allowing him/her to later on manipulate entries while still using the protection mechanism against other entities.
- **Control over Memory:** With full control over the memory of the machine the DBMS is running on, the attacker could simply record the sequence of random numbers r_i and thus manipulate older entries. While this is possible in theory, the attack possesses one issue in real implementations: The changes must be done on the transaction log by removing all current entries down to the point of manipulation and then recreating them using the sequence r_i. This can be a problem in case of running databases in case of rollbacks or crash recoveries during the manipulation process.
- **Control over Interfaces:** Especially considering the closed-source approach, with full control over all interfaces the attacker is capable of manipulating the data streams sent to the slave instances, while making no changes on the master instance.

Furthermore, the approach only works with DBMSs that actually provide transaction safety or data replication and thus provide the respective mechanisms.

More specific to the architecture provided in this paper, the main limitation lies in the assumption of independence of the different entities, which in reality may not be guaranteed due to the setup of the overall environment (e.g. a hospital running a "doctor in the loop" approach might control the doctor, the database and parts of the grid, as well as $\mathfrak{T}$). Against attacks arising from this overlap in duties and authorities, managerial countermeasures regarding organizational security must be employed, starting with strict separation of duties and reliance on an external party $\mathfrak{T}$.

CHAPTER 6

Conclusion

6.1 Summary

In this thesis we investigated several technologies in order to provide data leak protection and manipulation detection capabilities to data driven environments. Furthermore, we extended our approaches from simple processes where one data owner supplies information to several data recipients to expert in the loop systems, were an arbitrary amount of experts and AI components commit to a centralized shared database. Summarized, the main contributions of this thesis are the following:

6.1.1 Data leak detection

In Chapter 3 we provided an approach that combines anonymization with fingerprinting, or more precisely uses k-anonymization for fingerprinting, in order to detect data leaks. The approach has two major benefits, one being the capability to detect the leaking data recipient based on a single record, the second being that in case of sensitive information the anonymization step also covers the fingerprinting and no extra fingerprinting step is required. This basic approach was extended in various ways in order to incorporate colluding attackers (Section 3.4), advanced definitions of anonymity (Section 3.5), as well as advanced methods for achieving k-anonymity and derived anonymity concepts (Section 3.6). In Section 4.6, we utilized the signature log defined in Section 4.5 in order to provide a database fingerprinting mechanism that can be used in case a whole database is transferred to another user. In addition, we provided an alternative approach for the same use case based on database steganography by hiding the fingerprinting information in index slack space. Finally, we extended the chained witnesses approach proposed in previous work in order to accommodate for read access in Section 5.3.5, thus enabling the tracking of information exfiltration through the database SQL-interface by using the transaction mechanism. This is especially interesting, as the transaction log is an internal mechanism in the database out of control of the database administrator, who typically has full control over all logs. Alternatively, we used steganographic methods for providing an information store to hold all information on read-access. Both techniques can be combined. Furthermore, they can be combined with the fingerprinting techniques outlined in Chapters 3 and 4.

6.1.2 Manipulation detection

In Chapter 4 we provided a novel approach on detecting manipulation in revision secure databases possessing strictly monotonous primary keys. While seeming rather specific, these requirements are quite normal for revision secure tables. We extended the basic approach and discussed the limitations when allowing arbitrary insertion into the tree. While these limitations might be problematic for the forensic case, they provide a valuable result with respect to data deletion, as outlined in Section 4.4.3. Based on the original approach, we defined the $\mathcal{B}^+$-Tree signature allowing for forensic investigations, as well as advanced logging, backup and data transfer purposes. Regarding expert in the loop systems, we extended the chained witnesses approach from previous work in order to cater for the requirements regarding manipulation detection in interactive machine learning, including several expert parties and feeding back information from the AI components to the central database.

6.1.3 Trust for expert in the loop environments

With the promises of providing manipulation detection, as well as data leak detection, we provided a secure and trustworthy architecture for expert in the loop systems, exemplifying the approach in a doctor in the loop approach in the medical sector. With trust being a prime de-motivator for patients and doctors alike, we hope that the proposed architecture will enable the creation and establishing of platforms that will see widespread use, not only within the scientific community, but also targeting a wider audience, especially since most data driven analysis requires the incorporation of bigger data samples than currently available to many medical scientists.

In conclusion we believe that the work described in this thesis can have valuable impact on the field of data driven science and might solve issues of trust and security that are currently faced in this area. Since most parts of this work have already been published in scientific journals or at academic conferences (please refer to Section 6.2 for the list of publications), we have already gathered substantial feedback from the scientific community.

6.2 Future Work and Open Issues

In this section we discuss some open issues and possible extensions for future work in the respective areas: Our work on k-anonymity based fingerprinting could be further extended to more exotic definitions of anonymity. Furthermore, regarding l-diversity, as the approach currently only works for some specific forms of l-diversity, more work on

modifying it to be suitable for l-diversity in general would be needed. This also holds true for a practical implementation of l-diversity based fingerprinting, especially considering the problem of achieving this criterion in real-life data with correlations between the quasi identifiers and the payload.
Since our first paper in this area has already been published for several years, other research groups built upon it.

Regarding our approach for manipulation detection through $\mathcal{B}^+$-Tree forensics, this could be easily extended towards $\mathcal{B}^*$-Trees. Future work could also include the extension to $\mathcal{B}$-Trees, as well as other balanced trees currently used when storing large amounts of structured information.
Finally, regarding the expert in the loop approach, it would be interesting to apply the architecture in a real-world environment that is not focusing on research in the field of IT-Security, but is comprised of researchers outside the IT business, in order to evaluate the practicability and usefulness of the approach.
Finally, it would be interesting to put the data leak detection approaches provided in this thesis to the test in various real-life environments sporting typical user behaviour (especially under stress) and high profile attackers.

Academic & non-academic publication

Published papers

The work presented in this thesis, for the most parts, has already been published in the form of papers. The major reason for this approach lies in the inherent correction and review capabilities of the academic system. Thus, we strived to place our work in suitable journals and conferences, where the chances for good peer review are high and the level of the venue still matches the level of contribution of the respective paper. Thus, many parts of this thesis are verbatim copies from said journal and conference papers in order not to undermine the quality control provided by the academic publication process.

Peer reviewed academic journals and conferences

The following peer reviewed academic journal and conference papers were generated from content covered in this thesis:

1. **Kieseberg, P.**, Hobel, H., Schrittwieser, S., Weippl, E. and Holzinger, A., 2014. Protecting anonymity in data-driven biomedical science. In Interactive Knowledge Discovery and Data Mining in Biomedical Informatics (pp. 301-316). Springer, Berlin, Heidelberg. [KHS+14]

2. **Kieseberg, P.**, Schrittwieser, S., Mulazzani, M., Echizen, I. and Weippl, E., 2014. An algorithm for collusion-resistant anonymization and fingerprinting of sensitive microdata. Electronic Markets, 24(2), pp.113-124. [KSM+14]

3. Schrittwieser, S., **Kieseberg, P.**, Echizen, I., Wohlgemuth, S., Sonehara, N. and Weippl, E., 2011, October. An algorithm for k-anonymity-based fingerprinting. In International Workshop on Digital Watermarking (pp. 439-452). Springer, Berlin, Heidelberg. [SKE+11b]

4. Schrittwieser, S., **Kieseberg, P.**, Echizen, I., Wohlgemuth, S. and Sonehara, N., 2011, August. Using generalization patterns for fingerprinting sets of partially anonymized microdata in the course of disasters. In Availability, Reliability

and Security (ARES), 2011 Sixth International Conference on (pp. 645-649). IEEE. [SKE$^+$11a]

5. **Kieseberg, P.**, Schrittwieser, S., Mulazzani, M., Huber, M. and Weippl, E., 2011, September. Trees cannot lie: Using data structures for forensics purposes. In Intelligence and Security Informatics Conference (EISIC), 2011 European (pp. 282-285). IEEE. [KSM$^+$11b]

6. **Kieseberg, P.**, Schrittwieser, S., Morgan, L., Mulazzani, M., Huber, M. and Weippl, E., 2013. Using the structure of b+-trees for enhancing logging mechanisms of databases. Proceedings of the 13th International Conference on Information Integration and Web-based Applications & Services iiWAS, 2011 [KSM$^+$11a]

7. **Kieseberg, P.**, Schrittwieser, S., Morgan, L., Mulazzani, M., Huber, M. and Weippl, E., 2013. Using the structure of b+-trees for enhancing logging mechanisms of databases. International Journal of Web Information Systems, 9(1), pp.53-68. [KSM$^+$13]

8. **Kieseberg, P.**, Schrittwieser, S. and Weippl, E., 2018, November. Structural Limitations of B+-Tree forensics. In Proceedings of the Central European Cybersecurity Conference 2018 (p. 9). ACM. [KSW18]

9. **Kieseberg, P.**, Schantl, J., Frühwirt, P., Weippl, E. and Holzinger, A., 2015, August. Witnesses for the doctor in the loop. In International Conference on Brain Informatics and Health (pp. 369-378). Springer, Cham. [KSF$^+$15]

10. **Kieseberg, P.**, Malle, B., Frühwirt, P., Weippl, E. and Holzinger, A., 2016. A tamper-proof audit and control system for the doctor in the loop. Brain informatics, 3(4), pp.269-279. [KMF$^+$16]

11. **Kieseberg, P.**, Schrittwieser S., Malle B., Weippl E. and Holzinger A., 2017. Das Testen von Algorithmen in sensibler datengetriebener Forschung. 41. WI-MAW-Rundbrief. [KSM$^+$17]

12. **Kieseberg, P.** and Weippl, E., 2019. Securing Information against Manipulation in the Production Systems Engineering Process. In Security and Quality in Cyber-Physical Systems Engineering, Springer 2019. [KW19]

13. **Kieseberg, P.** and Weippl, E., 2019. Fingerprinting using database steganography. In International Conference on Software Security and Assurance ICSSA 2019. [KSW19]

14. **Kieseberg, P.**, Schrittwieser S., Amiri F., Holzinger A. and Weippl E., 2021. Advanced GDPR-compliant fingerprinting with k-anonymity. In Journal of Computer Security **(submitted)**

Overview articles in edited journals

The following overview articles were generated from content covered in this thesis:

1. **Kieseberg, P.**, Weippl, E. and Schrittwieser, S., 2016. Detection of data Leaks in Collaborative data driven Research. ERCIM NEWS, (105), pp.45-46. [KWS16]
2. **Kieseberg, P.**, Weippl, E. and Holzinger, A., 2016. Trust for the doctor-in-the-loop. ERCIM news, 104(1), pp.32-33. [KWH16]
3. **Kieseberg, P.**, Weippl, E. and Schrittwieser, S., 2017. Forensics using Internal Database Structures. ERCIM NEWS, (108), pp.50-51. [KWS17]

Other dissemination activities

Other dissemination activities include an interview in a national newspaper regarding privacy aware data collection in the darknet, as well as an assortment of different talks regarding topics of this thesis at more informal (non-peer reviewed) workshops, gatherings and outreaches to the general public:

1. Talk on "Directions in Database Forensics" incorporating results from this thesis at the IT-SecX (06.11.2015) and at the Security Forum (20.04.2016).
2. Talk on "Fingerprinting for data leak detection" incorporating results from this thesis at the "Workshop: Machine Learning for Health Informatics" (30.11.2015, Graz, Austria) and at the "Workshop: Machine Learning for Biomedicine" (26.01.2016, Graz, Austria).
3. Interview: "Wenn Daten von Unschuldigen im Darknet landen". In: derStandard (print) (23.08.2017) & online (24.08.2017).
4. Talk on "Security & Reproducibility in Health System Research" incorporating results from this thesis at the DEXHELPP Workshop (12.04.2018, Vienna, Austria).
5. Poster "Big Data Analytics and Privacy Aware Machine Learning" at "Lange Nacht der Forschung 2018" (28.09.2018, Vienna, Austria).

Overview of Generative AI Tools Used

All scientific results have been reached without the use of generative AI, as is also proven by the verbatim use of the published works which were mostly published well before the availability of these tools. AI was only used for partial translation purposes in the form of the DeepL-tool in order to generate parts of the German version of the acknowledgments and the abstract based on the respective original English versions written by Peter Kieseberg without AI support.

List of Figures

3.1 Record linkage attack 20
3.2 Record linkage attack thwarted by k-anonymity 22
3.3 Generalization of quasi-identifiers. 22
3.4 Lattice diagram for generalizations. 23
3.5 Performing different generalization/suppression operations to achieve the same level of anonymity. 26
3.6 Identifying the source of data leakage based on generalization patterns. . 29
3.7 Lattice diagram after step three. 30
3.8 Lattice diagram after the first generalization path. 31
3.9 Lattice diagram after the second generalization path. 31
3.10 The hull of a generalization pattern 36
3.11 The union of the hulls of two patterns 37
3.12 The hull generated by colluding two patterns 37
3.13 Example for a collusion 39
3.14 Examples for full-domain (A) and subtree (B) generalization. 48
3.15 Examples for subtree (A) and sibling (B) generalization. 49

4.1 Structure of the resulting $\mathcal{B}$-Tree 71
4.2 Structure of the resulting $\mathcal{B}^+$-Tree 72
4.3 One and two deleted records in a $\mathcal{B}^+$-Tree leaf (modified and taken from [FKW15]) 74
4.4 A $\mathcal{B}^+$-Tree resulting from ascending ordered inserts only. 77
4.5 Forged record in $\mathcal{B}^+$-Tree with sorted inserts only. 78
4.6 Forging many records to thwart forensic analysis. 78
4.7 $\mathcal{B}^+$-Tree resulting from two different $\mathcal{B}^+$-Trees. 82
4.8 $\mathcal{B}^+$-Tree with one node with $\frac{b}{2}+2$ elements 83
4.9 Constructing a $\mathcal{B}^+$-Tree with nodes holding more than $\frac{b}{2}+1$ elements . . 84
4.10 Constructing a $\mathcal{B}^+$-Tree with nodes holding only $\frac{b}{2}$ elements 85
4.11 Two different $\mathcal{B}^+$-Trees with the same leaf signature 86
4.12 $\mathcal{B}^+$-Tree of order 4 with $S = ((1), (2, 2), (3, 3, 2, 3, 3, 2))$ 87
4.13 Reconstructed $\mathcal{B}^+$-Tree 91
4.14 An example of a minimal $\mathcal{B}^+$-Tree, $n = 50, b = 8$ 94
4.15 An example of a maximal $\mathcal{B}^+$-Tree, $n = 50, b = 8$ 94
4.16 Overview on the proposed approach 102

5.1 Construction and storage of the chained witnesses. 110
5.2 The doctor in the loop. 110
5.3 Entities and Relations. 112
5.4 Multiple doctors in the loop. 114
5.5 Modified chained witnesses for multiple $\mathfrak{M}_i$. 116
5.6 Verification with replication (taken from [FKKW14]). 116

List of Tables

3.1 Anonymized data ($k = 2$) 22
3.2 Original microdata and two anonymized versions with $k = 2$ 27
3.3 Impact of different metrics 28
3.4 Generalization (a_1, b_0). 30
3.5 Data loss and k for solutions. 31
3.6 Generalization results. 32
3.7 Original data and three generalizations. 34
3.8 Original microdata and two anonymized versions with $k = 2$ 35
3.9 All generalization levels for the chosen data set 43
3.10 Possible generalization patterns for $k \geq 2$ 44
3.11 The constructed clusters 45
3.12 The distributed data sets 45
3.13 Generalizing QI ”age” for achieving k-anonymity ($k = 2$) 49
3.14 Subtree generalization: Original data and two anonymized sets ($k = 2$) . . 52
3.15 Original data and two anonymized sets ($k = 2$) 55
3.16 Attribute Suppression: Original data and two anonymized sets ($k = 2$) . . 57
3.17 Anatomization based fingerprinting: Original data 59
3.18 Anatomization based fingerprinting: Two QIDs 59
3.19 Anatomization based fingerprinting: Two sets (QIT, ST) 60
3.20 Original data and an anonymized set ($k = 3$) 61

4.1 Practical evaluation of runtime impact. 80

Bibliography

[ADRDS+20] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-López, Daniel Molina, Richard Benjamins, et al. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information Fusion*, 58:82–115, 2020.

[AK02] Rakesh Agrawal and Jerry Kiernan. Watermarking relational databases. In *VLDB'02: Proceedings of the 28th International Conference on Very Large Databases*, pages 155–166. Elsevier, 2002.

[ÁKKP21] Ricardo Ávila, Raphaël Khoury, Richard Khoury, and Fábio Petrillo. Use of security logs for data leak detection: A systematic literature review. *Security and Communication Networks*, 2021, 2021.

[AO08] Ali Al-Haj and Ashraf Odeh. Robust and blind watermarking of relational database systems. *Journal of Computer Science*, 4(12):1024–1029, 2008.

[ARG+10] Matthias Augustin, Kristian Reich, Gerd Glaeske, Ines Schaefer, and Marc Radtke. Co-morbidity and age-related prevalence of psoriasis: analysis of health insurance data in germany. *Acta dermato-venereologica*, 90(2):147–151, 2010.

[ASKSA+20] Eesa Al Solami, Muhammad Kamran, Mohammed Saeed Alkatheiri, Fouzia Rafiq, and Ahmed S Alghamdi. Fingerprinting of relational databases for stopping the data theft. *Electronics*, 9(7):1093, 2020.

[ASM13] Sultan Alneyadi, Elankayer Sithirasenan, and Vallipuram Muthukkumarasamy. Adaptable n-gram classification model for data leakage prevention. In *2013, 7th International Conference on Signal Processing and Communication Systems (ICSPCS)*, pages 1–8. IEEE, 2013.

[BA05] R.J. Bayardo and R. Agrawal. Data privacy through optimal k-anonymization. 2005.

[BCM+13] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 387–402. Springer, 2013.

[BF08] A. Burghardt and A.J. Feldman. Using the HFS+ journal for deleted file recovery. *digital investigation*, 5:S76–S82, 2008.

[BG12] Hetal Bhavsar and Amit Ganatra. A comparative study of training algorithms for supervised machine learning. *International Journal of Soft Computing and Engineering (IJSCE)*, 2(4):2231–2307, 2012.

[BGD05] Irtaza Barlas, Antonio Ginart, and Jordan L Dorrity. Self-evolution in knowledge bases. In *IEEE Autotestcon, 2005.*, pages 325–331. IEEE, 2005.

[BHF09] Morten Breivik, Geir Hovland, and Pål Johan From. Trends in research and publication: Science 2.0 and open access. *Modeling Identification and Control*, 30(3):181–190, 2009.

[Bil05] Brent Bilodeau. Beyond the gender binary: A case study of two transgender students at a midwestern research university. *Journal of Gay & Lesbian Issues in Education*, 3(1):29–44, 2005.

[BK17] Aiswarya Baby and Hema Krishnan. A literature survey on data leak detection and prevention methods. *International Journal of Advanced Research in Computer Science*, 8(5), 2017.

[BM72] R. Bayer and E.M. McCreight. Organization and maintenance of large ordered indexes. *Acta informatica*, 1(3):173–189, 1972.

[BNSE15] Thach V Bui, Thuc D Nguyen, Noboru Sonehara, and Isao Echizen. Tradeoff between the price of distributing a database and its collusion resistance based on concatenated codes. In *International Conference on Algorithms and Architectures for Parallel Processing*, pages 163–182. Springer, 2015.

[BO13] Gilles Barthe and Federico Olmedo. Beyond differential privacy: Composition theorems and relational logic for f-divergences between probabilistic programs. In *International Colloquium on Automata, Languages, and Programming*, pages 49–60. Springer, 2013.

[Bon12] Joseph Bonneau. The science of guessing: Analyzing an anonymized corpus of 70 million passwords. In *2012 IEEE Symposium on Security and Privacy*, pages 538–552, 2012.

[BOYD05] Elisa Bertino, Beng Chin Ooi, Yanjiang Yang, and Robert H Deng. Privacy and ownership preserving of outsourced medical data. In *Data Engineering, 2005. ICDE 2005. Proceedings. 21st International Conference on*, pages 521–532. IEEE, 2005.

[Bra17] Tim Bray. The javascript object notation (json) data interchange format. Technical report, 2017.

[C+09] Ann Cavoukian et al. Privacy by design: The 7 foundational principles. *Information and Privacy Commissioner of Ontario, Canada*, 5, 2009.

[Car05] B. Carrier. *File system forensic analysis*. Addison-Wesley Professional, 2005.

[Cav11] Ann Cavoukian. Privacy by design in law, policy and practice. *A white paper for regulators, decision-makers and policy-makers*, 2011.

[Cav12] Ann Cavoukian. Privacy by design [leading edge]. *IEEE Technology and Society Magazine*, 31(4):18–19, 2012.

[CD13] Nitesh V Chawla and Darcy A Davis. Bringing big data to personalized healthcare: a patient-centered framework. *Journal of general internal medicine*, 28(3):660–665, 2013.

[CGM+11] Sandy Clark, Travis Goodspeed, Perry Metzger, Zachary Wasserman, Kevin Xu, and Matt Blaze. Why (special agent) johnny (still) can't encrypt: A security analysis of the apco project 25 two-way radio system. In *USENIX Security Symposium*. Citeseer, 2011.

[CJ12] Ann Cavoukian and Jeff Jonas. *Privacy by design in the age of big data*. Information and Privacy Commissioner of Ontario, Canada, 2012.

[CJG21] Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong. Ipguard: Protecting intellectual property of deep neural networks via fingerprinting the classification boundary. In *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security*, pages 14–25, 2021.

[CMG+13] Mark J Costello, William K Michener, Mark Gahegan, Zhi-Qiang Zhang, and Philip E Bourne. Biodiversity data should be published, cited, and peer reviewed. *Trends in Ecology & Evolution*, 28(8):454–461, 2013.

[CN12] William R Claycomb and Alex Nicoll. Insider threats to cloud computing: Directions for new research challenges. In *2012 IEEE 36th Annual Computer Software and Applications Conference*, pages 387–394. IEEE, 2012.

[CNCC15] DA Clifton, KE Niehaus, P Charlton, and GW Colopy. Health informatics via machine learning for the clinical management of patients. *Yearbook of medical informatics*, 24(01):38–43, 2015.

[Com79] D. Comer. Ubiquitous B-tree. *ACM Computing Surveys (CSUR)*, 11(2):121–137, 1979.

[CRB99] Catherine Taylor Clelland, Viviana Risca, and Carter Bancroft. Hiding messages in dna microdots. *Nature*, 399(6736):533, 1999.

[CS04] Brian Carrier and Eugene H Spafford. An event-based digital forensic investigation framework. In *Digital forensic research workshop*, pages 11–13, 2004.

[CST] Mehmet U Celika, Gaurav Sharmaa, and A Murat Tekalpa. Collusion-resilient fingerprinting using random pre-warping.

[CT08] Alina Campan and Traian Marius Truta. Data and structural k-anonymity in social networks. In *International Workshop on Privacy, Security, and Trust in KDD*, pages 33–54. Springer, 2008.

[CYA12] Pern Hui Chia, Yusuke Yamamoto, and N. Asokan. Is this app safe?: a large scale study on application permissions and risk signals. In *Proceedings of the 21st international conference on World Wide Web*, pages 311–320, 2012.

[Dal86] Tore Dalenius. Finding a needle in a haystack-or identifying anonymous census record. *Journal of official statistics*, 2(3):329–336, 1986.

[DDFH+15] George Danezis, Josep Domingo-Ferrer, Marit Hansen, Jaap-Henk Hoepman, Daniel Le Metayer, Rodica Tirtea, and Stefan Schiffner. Privacy and data protection by design-from policy to engineering. *arXiv preprint arXiv:1501.03726*, 2015.

[DG09] Arti Deshpande and Jayant Gadge. New watermarking technique for relational databases. In *2009 Second International Conference on Emerging Trends in Engineering & Technology*, pages 664–669. IEEE, 2009.

[Dir95] EU Directive. Council directive 95/46/ec. *On the Protection of Individuals with Regard to the Processing of Personal Data and on the Free Movement of Such Data*, 1995.

[DJR12] Ratan Dey, Zubin Jelveh, and Keith W. Ross. Facebook users have become much more private: A large-scale study. In *2012 IEEE International Conference on Pervasive Computing and Communications Workshops*, pages 346–352, 2012.

[DSB17] Derek Doran, Sarah Schulz, and Tarek R Besold. What does explainable ai really mean? a new conceptualization of perspectives. *arXiv preprint arXiv:1710.00794*, 2017.

[DSS+00] Jana Dittmann, Peter Schmitt, Eva Saar, Joerg Schwenk, and Johannes Ueberberg. Combining digital watermarks and collusion secure fingerprints for digital images. *Journal of Electronic Imaging*, 9(4):456–467, 2000.

[Dwo11] Cynthia Dwork. Differential privacy. *Encyclopedia of Cryptography and Security*, pages 338–340, 2011.

[DWY+10] Xinbin Dai, Guodong Wang, Dong Sik Yang, Yuhong Tang, Pierre Broun, M David Marks, Lloyd W Sumner, Richard A Dixon, and Patrick Xuechun Zhao. Trichome: a comparative omics database for plant trichomes. *Plant physiology*, 152(1):44–54, 2010.

[Eck05] K. Eckstein. Forensics for advanced UNIX file systems. In *Information Assurance Workshop, 2004. Proceedings from the Fifth Annual IEEE SMC*, pages 377–385. IEEE, 2005.

[Eck10] Peter Eckersley. How unique is your web browser? In *International Symposium on Privacy Enhancing Technologies Symposium*, pages 1–18. Springer, 2010.

[EEDI+09] K. El Emam, F.K. Dankar, R. Issa, E. Jonker, D. Amyot, E. Cogo, J.P. Corriveau, M. Walker, S. Chowdhury, R. Vaillancourt, et al. A globally optimal k-anonymity method for the de-identification of health data. *Journal of the American Medical Informatics Association*, 16(5):670, 2009.

[EJ05] K. Eckstein and M. Jahnke. Data hiding in journaling file systems. In *Eingereicht beim Digital Forensic Research Workshop*. Citeseer, 2005.

[fDCP+03] Centers for Disease Control, Prevention, et al. Hipaa privacy rule and public health. guidance from cdc and the us department of health and human services. *MMWR: Morbidity and mortality weekly report*, 52(Suppl. 1):1–17, 2003.

[fECoC80] Organisation for Economic Co-operation and Development. Council. *Recommendation of the Council Concerning Guidelines Governing the Protection of Privacy and Transborder Flows of Personal Data*. OECD, 1980.

[FHMW10a] P. Fruehwirt, M. Huber, M. Mulazzani, and E.R. Weippl. InnoDB Database Forensics. In *Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on*, pages 1028–1036. IEEE, 2010.

[FHMW10b] Peter Frühwirt, Marcus Huber, Martin Mulazzani, and Edgar R Weippl. Innodb database forensics. In *2010 24th IEEE International Conference on Advanced Information Networking and Applications*, pages 1028–1036. IEEE, 2010.

[Fin08] Pete Finnigan. Oracle forensics. *OUG Scotland, DBA*, 30, 2008.

[FKKW14] Peter Frühwirt, Peter Kieseberg, Katharina Krombholz, and Edgar Weippl. Towards a forensic-aware database solution: Using a secured database replication protocol and transaction management for digital investigations. *Digital Investigation*, 11(4):336–348, 2014.

[FKS+12] Peter Fruehwirt, Peter Kieseberg, Sebastian Schrittwieser, Markus Huber, and Edgar R. Weippl. Innodb database forensics: Reconstructing data manipulation queries from redo logs. In *The Fifth International Workshop on Digital Forensics (WSDF)*, 8 2012.

[FKW15] Peter Fruhwirt, Peter Kieseberg, and Edgar Weippl. Using internal mysql/innodb b-tree index navigation for data hiding. In *IFIP International Conference on Digital Forensics*, pages 179–194. Springer, 2015.

[FOJ03] Jerry Alan Fails and Dan R Olsen Jr. Interactive machine learning. In *Proceedings of the 8th international conference on Intelligent user interfaces*, pages 39–45. ACM, 2003.

[Fro00] Julia M Fromholz. The european union data privacy directive. *Berk. Tech. LJ*, 15:461, 2000.

[FS03] V Fotopoulos and Athanassios N Skodras. Digital image watermarking: An overview. *Eurasip Newsletter*, 14(4):10–19, 2003.

[Fue17] Mayra Rosario Fuentes. Cybercrime and other threats faced by the healthcare industry. *Trend Micro*, 2017.

[FWCY10] Benjamin Fung, Ke Wang, Rui Chen, and Philip S Yu. Privacy-preserving data publishing: A survey of recent developments. *ACM Computing Surveys (CSUR)*, 42(4):14, 2010.

[Gar09] Simson L Garfinkel. Automating disk forensic processing with sleuthkit, xml and python. In *2009 Fourth International IEEE Workshop on Systematic Approaches to Digital Forensic Engineering*, pages 73–84. IEEE, 2009.

[GBC06] Annarita Giani, Vincent H Berk, and George V Cybenko. Data exfiltration and covert channels. In *Sensors, and Command, Control, Communications, and Intelligence (C3I) Technologies for Homeland Security and Homeland Defense V*, volume 6201, page 620103. International Society for Optics and Photonics, 2006.

[GEK09] Mouhcine Guennoun and Khalil El-Khatib. Securing medical data in smart homes. In *2009 IEEE International Workshop on Medical Measurements and Applications*, pages 104–107. IEEE, 2009.

[GKH15] Dominic Girardi, Josef Kueng, and Andreas Holzinger. A domain-expert centered process model for knowledge discovery in medical research: putting the expert-in-the-loop. In *International Conference on Brain Informatics and Health*, pages 389–398. Springer, 2015.

[GKKM07] Gabriel Ghinita, Panagiotis Karras, Panos Kalnis, and Nikos Mamoulis. Fast data anonymization with low information loss. In *Proceedings of the 33rd international conference on Very large data bases*, pages 758–769, 2007.

[Gro11] David Gross-AMBLARD. Query-preserving watermarking of relational databases and xml documents. *ACM Transactions on Database Systems*, 36(1):3, 2011.

[Gun17] David Gunning. Explainable artificial intelligence (xai). *Defense Advanced Research Projects Agency (DARPA), nd Web*, 2017.

[HCCY13] Carly L Huth, David W Chadwick, William R Claycomb, and Ilsun You. Guest editorial: A brief overview of data leakage and insider threats. *Information Systems Frontiers*, 15(1):1–4, 2013.

[HDJ14] Andreas Holzinger, Matthias Dehmer, and Igor Jurisica. Knowledge discovery and interactive data mining in bioinformatics-state-of-the-art, future challenges and research directions. *BMC bioinformatics*, 15(6):I1, 2014.

[HK99] F. Hartung and M. Kutter. Multimedia watermarking techniques. *Proceedings of the IEEE*, 87(7):1079–1107, 1999.

[HMK+17] Andreas Holzinger, Bernd Malle, Peter Kieseberg, Peter M Roth, Heimo Müller, Robert Reihs, and Kurt Zatloukal. Machine learning and knowledge extraction in digital pathology needs an integrative approach. In *Towards Integrative Machine Learning and Knowledge Extraction*, pages 13–50. Springer, 2017.

[Hoe14] Jaap-Henk Hoepman. Privacy design strategies. In *IFIP International Information Security Conference*, pages 446–459. Springer, 2014.

[Hol14] Andreas Holzinger. *Biomedical informatics: discovering knowledge in big data*. Springer, 2014.

[HRFN12] Johannes Heurix, Antonio Rella, Stefan Fenz, and Thomas Neubauer. Automated transformation of semi-structured text elements. In *AMCIS*, 2012.

[HRFN13] Johannes Heurix, Antonio Rella, Stefan Fenz, and Thomas Neubauer. A rule-based transformation system for converting semi-structured medical documents. *Health and Technology*, 3(1):51–63, 2013.

[HSKW13] Heidelinde Hobel, Sebastian Schrittwieser, Peter Kieseberg, and Edgar Weippl. Anonymity and pseudonymity in data-driven science. 2013.

[IDC15] Taya Irizarry, Annette DeVito Dabbs, and Christine R Curran. Patient portals and patient engagement: a state of the science review. *Journal of medical Internet research*, 17(6):e148, 2015.

[Iye02] Vijay S Iyengar. Transforming data to satisfy privacy constraints. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 279–288. ACM, 2002.

[Jac09] Adam Jacobs. The pathologies of big data. *ACM Queue*, 52(8):36–44, 2009.

[JDJ01] Neil F Johnson, Zoran Duric, and Sushil Jajodia. *Information Hiding: Steganography and Watermarking-Attacks and Countermeasures: Steganography and Watermarking: Attacks and Countermeasures*, volume 1. Springer Science & Business Media, 2001.

[JJ98] Neil F Johnson and Sushil Jajodia. Exploring steganography: Seeing the unseen. *Computer*, 31(2):26–34, 1998.

[JJB12] Peter B Jensen, Lars J Jensen, and Søren Brunak. Mining electronic health records: towards better research applications and clinical care. *Nature Reviews Genetics*, 13(6):395–405, 2012.

[Joh12] Jeremy Leighton John. *Digital forensics and preservation.* Citeseer, 2012.

[KB10] P. Koruga and M. Bača. Analysis of B-tree data structure and its usage in computer forensics. In *Central European Conference on Information and Intelligent Systems*, 2010.

[KCGD06] Karen Kent, Suzanne Chevalier, Tim Grance, and Hung Dang. Guide to integrating forensic techniques into incident response. *NIST Special Publication*, 10(14):800–86, 2006.

[KHS+14] Peter Kieseberg, Heidelinde Hobel, Sebastian Schrittwieser, Edgar Weippl, and Andreas Holzinger. Protecting anonymity in data-driven biomedical science. In *Interactive Knowledge Discovery and Data Mining in Biomedical Informatics*, pages 301–316. Springer Berlin Heidelberg, 2014.

[KKH19] Peter Kieseberg, Lukas Daniel Klausner, and Andreas Holzinger. Distortion in real-world analytic processes. *ERCIM News*, 118, 7 2019.

[Klo95] Willi Klosgen. Anonymization techniques for knowledge discovery in databases. In *First International Conference on Knowledge Discovery and Data Mining (KDD-95), Montreal, Quebec, Canada, AAAI Press, Menlo Park, California*, pages 186–191, 1995.

[KLOT12] Wesley T Kerr, Edward P Lau, Gwen E Owens, and Aaron Trefler. The future of medical diagnostics: large digitized databases. *The Yale journal of biology and medicine*, 85(3):363, 2012.

[KMF+16] Peter Kieseberg, Bernd Malle, Peter Frühwirt, Edgar R. Weippl, and Andreas Holzinger. A tamper-proof audit and control system for the doctor in the loop. *Brain Informatics*, 3(4):269–279, 2016.

[KNJ+12] Clete A Kushida, Deborah A Nichols, Rik Jadrnicek, Ric Miller, James K Walsh, and Kara Griffin. Strategies for de-identification and anonymization of electronic health record data for use in multicenter research studies. *Medical care*, 50:S82–S101, 2012.

[KPE+12a] Florian Kohlmayer, Fabian Prasser, Claudia Eckert, Alfons Kemper, and Klaus A Kuhn. Flash: efficient, stable and optimal k-anonymity. In *2012 International Conference on Privacy, Security, Risk and Trust and 2012 International Confernece on Social Computing*, pages 708–717. IEEE, 2012.

[KPE+12b] Florian Kohlmayer, Fabian Prasser, Claudia Eckert, Alfons Kemper, and Klaus A Kuhn. Highly efficient optimal k-anonymity for biomedical datasets. In *2012 25th IEEE International Symposium on Computer-Based Medical Systems (CBMS)*, pages 1–6. IEEE, 2012.

[KSF+15] Peter Kieseberg, Johannes Schantl, Peter Frühwirt, Edgar Weippl, and Andreas Holzinger. Witnesses for the doctor in the loop. In *Brain Informatics and Health*, pages 369–378. Springer, 2015.

[KSM+11a] Peter Kieseberg, Sebastian Schrittwieser, Lorcan Morgan, Martin Mulazzani, Markus Huber, and Edgar R. Weippl. Using the structure of b+-trees for enhancing logging mechanisms of databases. In *Proceedings of the 13th International Conference on Information Integration and Web-based Applications and Services*, pages 301–304, 2011.

[KSM+11b] Peter Kieseberg, Sebastian Schrittwieser, Martin Mulazzani, Markus Huber, and Edgar Weippl. Trees cannot lie: Using data structures for forensics purposes. In *Intelligence and Security Informatics Conference (EISIC), 2011 European*, pages 282–285. IEEE, 2011.

[KSM+13] Peter Kieseberg, Sebastian Schrittwieser, Lorcan Morgan, Martin Mulazzani, Markus Huber, and Edgar Weippl. Using the structure of b+-trees for enhancing logging mechanisms of databases. *International Journal of Web Information Systems*, 9(1):53–68, 2013.

[KSM+14] Peter Kieseberg, Sebastian Schrittwieser, Martin Mulazzani, Isao Echizen, and Edgar R. Weippl. An algorithm for collusion-resistant anonymization and fingerprinting of sensitive microdata. *Electronic Markets*, 24(2):113–124, 2014.

[KSM+17] Peter Kieseberg, Sebastian Schrittwieser, Bernd Malle, Edgar R. Weippl, and Andreas Holzinger. Das testen von algorithmen in sensibler datengetriebener forschung. *Rundbrief des Fachausschusses Management der Anwendungsentwicklung und -wartung (WI-MAW)*, 4 2017.

[KSW18] Peter Kieseberg, Sebastian Schrittwieser, and Edgar R. Weippl. Structural limitations of b+-tree forensics. In *Proceedings of the Central European Cybersecurity Conference 2018 on*, 2018.

[KSW19] Peter Kieseberg, Sebastian Schrittwieser, and Edgar Weippl. Fingerprinting using database steganography. In *2019 International Conference on Software Security and Assurance (ICSSA)*, pages 16–20. IEEE, 2019.

[KW19] Peter Kieseberg and Edgar Weippl. Securing information against manipulation in the production systems engineering process. In *Security and Quality in Cyber-Physical Systems Engineering*, pages 335–356. Springer, 2019.

[KWH16] Peter Kieseberg, Edgar R. Weippl, and Andreas Holzinger. Trust for the "doctor in the loop". *ERCIM News*, 1 2016.

[KWS16] Peter Kieseberg, Edgar R. Weippl, and Sebastian Schrittwieser. Detection of data leaks in collaborative data driven research. *Ercim News*, 2016, 2016.

[KWS17] Peter Kieseberg, Edgar R. Weippl, and Sebastian Schrittwieser. Forensics using internal database structures. *Ercim News*, 2017, 2017.

[Laf07] J. Lafaye. An analysis of database watermarking security. In *Third International Symposium on Information Assurance and Security*, pages 462–467, 2007.

[Lan01] Marc Langheinrich. Privacy by design—principles of privacy-aware ubiquitous systems. In *International conference on Ubiquitous Computing*, pages 273–291. Springer, 2001.

[LC11] Mary E. Ludloff and Terence Craig. *Privacy and Big Data*. 2011.

[LDR05] Kristen LeFevre, David J DeWitt, and Raghu Ramakrishnan. Incognito: Efficient full-domain k-anonymity. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 49–60, 2005.

[LDR06] Kristen LeFevre, David J DeWitt, and Raghu Ramakrishnan. Mondrian multidimensional k-anonymity. In *22nd International conference on data engineering (ICDE'06)*, pages 25–25. IEEE, 2006.

[LHMM18] Yunlong Lu, Xiaohong Huang, Yan Ma, and Maode Ma. A weighted context graph model for fast data leak detection. In *2018 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE, 2018.

[LLV07] N. Li, T. Li, and S. Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and l-diversity. In *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on*, pages 106–115. IEEE, 2007.

[LSC+12] Insup Lee, Oleg Sokolsky, Sanjian Chen, John Hatcliff, Eunkyoung Jee, BaekGyu Kim, Andrew King, Margaret Mullen-Fortino, Soojin Park, Alexander Roederer, et al. Challenges and research directions in medical cyber–physical systems. *Proceedings of the IEEE*, 100(1):75–90, 2012.

[LSL00] G.C. Langelaar, I. Setyawan, and R.L. Lagendijk. Watermarking digital image and video data. a state-of-the-art overview. *IEEE Signal Processing Magazine*, 17(5):20–46, 2000.

[LTX08] Jiexing Li, Yufei Tao, and Xiaokui Xiao. Preservation of proximity privacy in publishing numerical sensitive data. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 473–486. ACM, 2008.

[Lup08] Dennis Lupiana. Development of a framework to leverage knowledge management systems to improve security awareness. 2008.

[LWDS04] Siyuan Liu, Shuhong Wang, Robert H Deng, and Weizhong Shao. A block oriented fingerprinting scheme in relational database. In *International Conference on Information Security and Cryptology*, pages 455–466. Springer, 2004.

[LX09] Qun Li and Honglin Xu. Research on the backup mechanism of oracle database. In *2009 International Conference on Environmental Science and Information Application Technology*, volume 2, pages 423–426. IEEE, 2009.

[Mal05] Bradley A Malin. Protecting genomic sequence anonymity with generalization lattices. *Methods of information in medicine*, 44(05):687–692, 2005.

[Mil12] Dominic Milano. Content control: Digital watermarking and fingerprinting. *White Paper, Rhozet, a business unit of Harmonic Inc., http://www. rhozet. com/whitepapers/Fingerprinting—Watermarking. pdf, Last accessed May,* 30, 2012.

[MKGV07] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):3, 2007.

[MKH17] Bernd Malle, Peter Kieseberg, and Andreas Holzinger. Do not disturb? classifier behavior on perturbed datasets. In *International Cross-Domain Conference for Machine Learning and Knowledge Extraction*, pages 155–173. Springer, 2017.

[MKWH16] Bernd Malle, Peter Kieseberg, Edgar Weippl, and Andreas Holzinger. The right to be forgotten: towards machine learning on perturbed knowledge bases. In *International Conference on Availability, Reliability, and Security*, pages 251–266. Springer, 2016.

[MT11] G Murthy and R Srinivas M Tech. Achieving multidimensional k-anonymity by a greedy approach. In *International Conference on Web Services Computing (ICWSC)*, 2011.

[MW] M. Mulazzani and E. Weippl. Aktuelle Herausforderungen in der Datenbankforensik.

[MX07] Rajeev Motwani and Ying Xu. Efficient algorithms for masking and finding quasi-identifiers. In *Proceedings of the Conference on Very Large Data Bases (VLDB)*, pages 83–93, 2007.

[NC09] Jasvir Nagra and Christian Collberg. *Surreptitious software: obfuscation, watermarking, and tamperproofing for software protection*. Pearson Education, 2009.

[NH11] Thomas Neubauer and Johannes Heurix. A methodology for the pseudonymization of medical data. *International journal of medical informatics*, 80(3):190–204, 2011.

[NHW+10] Kim M Nazi, Timothy P Hogan, Todd H Wagner, D Keith McInnes, Bridget M Smith, David Haggstrom, Neale R Chumbler, Allen L Gifford, Kathleen G Charters, Jason J Saleem, et al. Embracing a health services research perspective on personal health records: lessons learned from the va my healthevet system. *Journal of general internal medicine*, 25(1):62–67, 2010.

[Opd10] John Douglas JD Opdyke. A unified approach to algorithms generating unrestricted and restricted integer compositions and integer partitions. *Journal of Mathematical Modelling and Algorithms*, 9(1):53–97, 2010.

[PAB06] Marcus Peinado, Rajasekhar Abburi, and Jeffrey RC Bell. Structural of digital rights management (drm) system, April 4 2006. US Patent 7,024,393.

[Pet11] Fabien AP Petitcolas. Kerckhoffs' principle. *Encyclopedia of cryptography and security*, pages 675–675, 2011.

[PK01] Andreas Pfitzmann and Marit Köhntopp. Anonymity, unobservability, and pseudonymity—a proposal for terminology. In *Designing privacy enhancing technologies*, pages 1–9. Springer, 2001.

[PM09] Anandabrata Pal and Nasir Memon. The evolution of file carving. *IEEE signal processing magazine*, 26(2):59–71, 2009.

[PO12] Heloise Pieterse and Martin Olivier. Data hiding techniques for database environments. In *IFIP International Conference on Digital Forensics*, pages 289–301. Springer, 2012.

[PS17] C Mercy Praba and India Dr G Satyavathy. A technical review on data leakage detection and prevention approaches. *Journal of Network Communications and Emerging Technologies (JNCET)*, 2017.

[QYJjYs06] Zhu Qin, Yang Ying, LE Jia-jin, and LUO Yi-shu. Watermark based copyright protection of outsourced database. In *Database Engineering and Applications Symposium, 2006. IDEAS'06. 10th International*, pages 301–308. IEEE, 2006.

[Ran05] Ebrahim Randeree. Secure health knowledge: Balancing security, privacy and access. *AMCIS 2005 Proceedings*, page 287, 2005.

[RBM13] Ohad Rodeh, Josef Bacik, and Chris Mason. Btrfs: The linux b-tree filesystem. *ACM Transactions on Storage (TOS)*, 9(3):9, 2013.

[RBR+00] Patrick Ruch, Robert H Baud, Anne-Marie Rassinoux, Pierrette Bouillon, and Gilbert Robert. Medical document anonymization with a semantic lexicon. In *Proceedings of the AMIA Symposium*, page 729. American Medical Informatics Association, 2000.

[RBR+11] Andrew G. Rundle, Michael D.M. Bader, Catherine A. Richards, Kathryn M. Neckerman, and Julien O. Teitler. Using google street view to audit neighborhood environments. *American Journal of Preventive Medicine*, 40(1):94–100, 2011.

[Reg16] General Data Protection Regulation. Regulation (eu) 2016/679 of the european parliament and of the council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46. *Official Journal of the European Union (OJ)*, 59(1-88):294, 2016.

[Rei04] Hans Reiser. Reiserfs, 2004.

[RF04] Richard Russon and Yuval Fledel. Ntfs documentation, 2004.

[RKS11] Preeti Raman, Hilmi Güneş Kayacık, and Anil Somayaji. Understanding data leak prevention. In *6th Annual Symposium on Information Assurance (ASIA'11)*, page 27. Citeseer, 2011.

[RLW18] Sandra Domenique Ringmann, Hanno Langweg, and Marcel Waldvogel. Requirements for legally compliant software based on the gdpr. In *OTM Confederated International Conferences On the Move to Meaningful Internet System*, pages 258–276. Springer, 2018.

[RP17] John Mark Michael Rumbold and Barbara Pierscionek. The effect of the general data protection regulation on medical research. *Journal of medical Internet research*, 19(2), 2017.

[RR98] Wayne E Relyea and Suzanne E Ronca. Real-time communications fraud monitoring system, January 6 1998. US Patent 5,706,338.

[S+02] L. Sweeney et al. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty Fuzziness and Knowledge Based Systems*, 10(5):557–570, 2002.

[S+07] Yutaka Sasaki et al. The truth of the f-measure. *Teach Tutor mater*, 1(5):1–5, 2007.

[Sam01] P. Samarati. Protecting respondents' identities in microdata release. *IEEE Transactions on Knowledge and Data Engineering*, pages 1010–1027, 2001.

[SAP04] R. Sion, M. Atallah, and Sunil Prabhakar. Rights protection for relational data. *IEEE Transactions on Knowledge and Data Engineering*, 16(12):1509–1525, 2004.

[Sar02] Paul Sarbanes. Sarbanes-oxley act of 2002. In *The Public Company Accounting Reform and Investor Protection Act. Washington DC: US Congress*, 2002.

[SASS11] Jyoti Soni, Ujma Ansari, Dipesh Sharma, and Sunita Soni. Predictive data mining for medical diagnosis: An overview of heart disease prediction. *International Journal of Computer Applications*, 17(8):43–48, 2011.

[SCFY96] Ravi S Sandhu, Edward J Coyne, Hal L Feinstein, and Charles E Youman. Role-based access control models. *Computer*, 29(2):38–47, 1996.

[Sch10] Peter Schaar. Privacy by design. *Identity in the Information Society*, 3(2):267–274, 2010.

[SCNP10] Stefan Siersdorfer, Sergiu Chelaru, Wolfgang Nejdl, and Jose San Pedro. How useful are your comments?: analyzing and predicting youtube comments and comment ratings. In *Proceedings of the 19th international conference on World wide web*, pages 891–900, 2010.

[SCS+08] Adam Silberstein, Brian F Cooper, Utkarsh Srivastava, Erik Vee, Ramana Yerneni, and Raghu Ramakrishnan. Efficient bulk insertion into a distributed ordered table. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 765–778. ACM, 2008.

[SE13] Aditya K Sood and Richard J Enbody. Targeted cyberattacks: a superset of advanced persistent threats. *IEEE security & privacy*, 11(1):54–61, 2013.

[Sha05] Yakov Shafranovich. Common format and mime type for comma-separated values (csv) files. 2005.

[SHK+21] Djordje Slijepčević, Maximilian Henzl, Lukas Daniel Klausner, Tobias Dam, Peter Kieseberg, and Matthias Zeppelzauer. *k*-anonymity in practice: How generalisation and suppression affect machine learning classifiers. *arXiv preprint arXiv:2102.04763*, 2021.

[SJL+05] Jin S Seo, Minho Jin, Sunil Lee, Dalwon Jang, Seungjae Lee, and Chang D Yoo. Audio fingerprinting based on normalized spectral subband centroids. In *Proceedings.(ICASSP'05). IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005.*, volume 3, pages iii–213. IEEE, 2005.

[SKE+11a] Sebastian Schrittwieser, Peter Kieseberg, Isao Echizen, Sven Wohlgemuth, and Noboru Sonehara. Using generalization patterns for fingerprinting sets of partially anonymized microdata in the course of disasters. In *2011 Sixth International Conference on Availability, Reliability and Security*, pages 645–649, 2011.

[SKE+11b] Sebastian Schrittwieser, Peter Kieseberg, Isao Echizen, Sven Wohlgemuth, Noboru Sonehara, and Edgar R. Weippl. An algorithm for k -anonymity-based fingerprinting. In *IWDW'11 Proceedings of the 10th international conference on Digital-Forensics and Watermarking*, pages 439–452, 2011.

[SKH02] Karen Su, Deepa Kundur, and Dimitrios Hatzinakos. Novel approach to collusion-resistant video watermarking. In *Security and watermarking of multimedia contents. Conference*, volume 4675, pages 491–502, 2002.

[SKH05] K. Su, D. Kundur, and D. Hatzinakos. Statistical invisibility for collusion-resistant digital video watermarking. *IEEE Transactions on Multimedia*, 7(1):43–51, 2005.

[SLW11] Xiaoxun Sun, Min Li, and Hua Wang. A family of enhanced (l, α)-diversity models for privacy preserving data publishing. *Future Generation Computer Systems*, 27(3):348–356, 2011.

[ŠM19] Tanja Šarčević and Rudolf Mayer. An evaluation on robustness and utility of fingerprinting schemes. In *International Cross-Domain Conference for Machine Learning and Knowledge Extraction*, pages 209–228. Springer, 2019.

[Smi03] Sean W Smith. Humans in the loop: Human-computer interaction and security. *Security & Privacy, IEEE*, 1(3):75–79, 2003.

[Spi02] Ray Spier. The history of the peer-review process. *TRENDS in Biotechnology*, 20(8):357–358, 2002.

[SPS07] C. Swenson, R. Phillips, and S. Shenoi. File System Journal Forensics. *Advances in Digital Forensics III*, pages 231–244, 2007.

[Ste09] Robert Steinbrook. Health care and the american recovery and reinvestment act. *New England Journal of Medicine*, 360(11):1057–1060, 2009.

[Swe02a] L. Sweeney. Achieving k-anonymity privacy protection using generalization and suppression. *International Journal of Uncertainty Fuzziness and Knowledge-Based Systems*, 10(5):571–588, 2002.

[Swe02b] L. Sweeney. Comments to the department of health and human services on "standards of privacy of individually identifiable health information". 2002.

[SWPB18] Marta Sabou, Dietmar Winkler, Peter Penzerstadler, and Stefan Biffl. Verifying conceptual domain models with human computation: A case study in software engineering. In *Sixth AAAI Conference on Human Computation and Crowdsourcing*, 2018.

[SY12a] Xiaokui Shu and Danfeng Daphne Yao. Data leak detection as a service. In *International Conference on Security and Privacy in Communication Systems*, pages 222–240. Springer, 2012.

[SY12b] Xiaokui Shu and Danfeng Daphne Yao. Data leak detection as a service: challenges and solutions. 2012.

[TBO+12] Olivier Thonnard, Leyla Bilge, Gavin O'Gorman, Seán Kiernan, and Martin Lee. Industrial espionage and targeted attacks: Understanding the characteristics of an escalating threat. In *International workshop on recent advances in intrusion detection*, pages 64–85. Springer, 2012.

[TC10] Traian Marius Truta and Alina Campan. Avoiding attribute disclosure with the (extended) p-sensitive k-anonymity model. In *Data Mining*, pages 353–373. Springer, 2010.

[TCN+14] Danan Thilakanathan, Shiping Chen, Surya Nepal, Rafael Calvo, and Leila Alem. A platform for secure monitoring and sharing of generic health data in the cloud. *Future Generation Computer Systems*, 35:102–113, 2014.

[TGH+08] H Thiele, J Glandorf, P Hufnagel, G Körting, and M Blüggel. Managing proteomics data: from generation and data warehousing to central data repository. *J Proteomics Bioinform*, 1:485–507, 2008.

[TH12] Matthew Thompson and Carl Heneghan. We need to move the debate on open clinical trial data forward. *BMJ*, 345, 2012.

[TWWL03] Wade Trappe, Min Wu, Z Jane Wang, and KJ Ray Liu. Anti-collusion fingerprinting for multimedia. *IEEE Transactions on Signal Processing*, 51(4):1069–1087, 2003.

[VKL18] Eduard Fosch Villaronga, Peter Kieseberg, and Tiffany Li. Humans forget, machines remember: Artificial intelligence and the right to be forgotten. *Computer Law & Security Review*, 34(2):304–313, 2018.

[WdW96] Leon Cornelis Roelof Johannes Willenborg and Ton de Waal. *Statistical Disclosure Control in Practice*. 1996.

[Web98] Ron A Weber. *Information systems control and audit*. Pearson Education, 1998.

[WF06] Ke Wang and Benjamin Fung. Anonymizing sequential releases. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 414–423. ACM, 2006.

[WGWW+16] Sandra Wartner, Dominic Girardi, Manuela Wiesinger-Widi, Johannes Trenkler, Raimund Kleiser, and Andreas Holzinger. Ontology-guided principal component analysis: Reaching the limits of the doctor-in-the-loop. In *International Conference on Information Technology in Bio-and Medical Informatics*, pages 22–33. Springer, 2016.

[WHT06] E. Weippl, A. Holzinger, and A. M. Tjoa. Security aspects of ubiquitous computing in health care. *Elektrotechnik Und Informationstechnik*, 123(4):156–161, 2006.

[WJA06] Merrill Warkentin, Allen Johnston, and April Adams. User interaction with healthcare information systems: Do healthcare professionals want to comply with hipaa? *AMCIS 2006 Proceedings*, page 326, 2006.

[WKR95] Robert Weibel, Stefan Keller, and Tumasch Reichenbacher. Overcoming the knowledge acquisition bottleneck in map generalization: the role of interactive systems and computational intelligence. In *International Conference on Spatial Information Theory*, pages 139–156. Springer, 1995.

[WL12] Robert West and Jure Leskovec. Human wayfinding in information networks. In *Proceedings of the 21st international conference on World Wide Web*, pages 619–628, 2012.

[WMC14] Shuang Wang, Noman Mohammed, and Rui Chen. Differentially private genome data dissemination through top-down specialization. *BMC medical informatics and decision making*, 14(1):S2, 2014.

[WRK19] Dietmar Winkler, Felix Rinker, and Peter Kieseberg. Towards a flexible and secure round-trip-engineering process for production systems engineering with agile practices. In *International Conference on Software Quality*, pages 14–30. Springer, 2019.

[WS12] Shijun Wang and Ronald M Summers. Machine learning and radiology. *Medical image analysis*, 16(5):933–951, 2012.

[WSB+12] Byron C Wallace, Kevin Small, Carla E Brodley, Joseph Lau, and Thomas A Trikalinos. Deploying an interactive machine learning system in an evidence-based practice center: abstrackr. In *Proceedings of the 2nd ACM SIGHIT International Health Informatics Symposium*, pages 819–824. ACM, 2012.

[WZA+16] Alexander M Walker, Xiaofeng Zhou, Ashwin N Ananthakrishnan, Lisa S Weiss, Rongjun Shen, Rachel E Sobel, Andrew Bate, and Robert F Reynolds. Computer-assisted expert case definition in electronic health records. *International Journal of Medical Informatics*, 86:62–70, 2016.

[XT06] Xiaokui Xiao and Yufei Tao. Anatomy: Simple and effective privacy preservation. In *Proceedings of the 32nd international conference on Very large data bases*, pages 139–150. VLDB Endowment, 2006.

[YH08] Gregory S Yovanof and George N Hazapis. Disruptive technologies, services, or business models? *Wireless Personal Communications*, 45(4):569–583, 2008.

[Zai09] Peter Zaitsev. Innodb architecture and performance optimization. *O'Reilly MySQLConference and Expo, April 2009*, 2009.

[ZB11] Hui Zang and Jean-Chrysostome Bolot. Anonymization of location data does not work: a large-scale measurement study. In *Proceedings of the 17th annual international conference on Mobile computing and networking*, pages 145–156, 2011.

[ZJWL04] Zhi-Hao Zhang, Xiao-Ming Jin, Jian-Min Wang, and De-Yi Li. Watermarking relational database using image. In *Proceedings of 2004 International Conference on Machine Learning and Cybernetics (IEEE Cat. No.04EX826)*, volume 3, pages 1739–1744, 2004.

[ZL14] Qin Zhang and Loukas Lazos. Collusion-resistant query anonymization for location-based services. In *2014 IEEE International Conference on Communications (ICC)*, pages 768–774. IEEE, 2014.

[ZO11] Hessam Zakerzadeh and Sylvia L Osborn. Faanst: fast anonymizing algorithm for numerical streaming data. In *Data privacy management and autonomous spontaneous security*, pages 36–50. Springer, 2011.

[ZPB14] G Zauner, N Popper, and F Breitenecker. State of the art research in austria: Dexhelpp-decision support for health policy and planning: Methods, models and technologies based on existing health care data. *Value in Health*, 17(7):A452, 2014.

[ZZ02] Chengqi Zhang and Shichao Zhang. *Association rule mining: models and algorithms*. Springer-Verlag, 2002.