NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCES
DEPARTMENT OF INFORMATICS & TELECOMMUNICATIONS

MASTER'S PROGRAM
"DATA SCIENCE AND INFORMATION TECHNOLOGIES"
SPECIALIZATION: BIG DATA AND ARTIFICIAL INTELLIGENCE

DIPLOMA THESIS

Privacy preservation in loosely-coupled, anonymized health data sources: data exploration and risk scenarios

Nikolaos E. Kapetanas

Supervisor: **Theodore Dalamagas,** Research Director, Information Management Systems Institute, Athena Research Center

ATHENS

ΟΚΤΩΒΡΙΟΣ 2022

ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΔΙΙΔΡΥΜΑΤΙΚΟ ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
"DATA SCIENCE AND INFORMATION TECHNOLOGIES (DSIT)"
ΕΙΔΙΚΟΤΗΤΑ: ΜΕΓΑΛΑ ΔΕΔΟΜΕΝΑ ΚΑΙ ΤΕΧΝΗΤΗ ΝΟΗΜΟΣΥΝΗ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Διατήρηση απορρήτου σε κατανεμημένες, ανώνυμες πηγές δεδομένων υγείας: εξερεύνηση δεδομένων και σενάρια κινδύνου

Νικόλαος Ε. Καπετανάς

Επιβλέπων: **Θεόδωρος Δαλαμάγκας,** Διευθυντής Ερευνών, Ινστιτούτο Πληροφοριακών Συστημάτων, Ερευνητικό Κέντρο Αθηνά

ΑΘΗΝΑ

ΟΚΤΏΒΡΙΟΣ 2022

DIPLOMA THESIS

Privacy preservation in loosely-coupled, anonymized health data sources: data exploration and risk scenarios

Nikolaos E. Kapetanas

A.M.: DS1190008

October 2022

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Διατήρηση απορρήτου σε κατανεμημένες, ανώνυμες πηγές δεδομένων υγείας: εξερεύνηση δεδομένων και σενάρια κινδύνου

Νικόλαος Ε. Καπετανάς

Α.Μ.: DS1190008

Οκτώβριος 2022

ABSTRACT

Protecting data in the healthcare industry is no easy feat. The healthcare data owners must balance protecting patient privacy while delivering quality patient care and meeting the strict regulatory requirements set forth by HIPAA and other regulations, such as the EU’s General Data Protection Regulation (GDPR). If the requirements are not met, hefty penalties and fines are applied. There is also a need to publish those data regularly for research purposes which will lead to better healthcare services. A critical challenge is to be able to provide effective privacy preservation to the patient's personal data. To meet this requirement, many anonymization techniques are applied like k-anonymity and l-diversity. If the published data sets are independent and contain information about the same person, then the data are still vulnerable to composition attacks. This study will adopt loosely coupled database technologies to develop a system to connect, retrieve and explore the privacy preservation of the data. The thesis will carry out the following tasks: (a) urveying state-of-the-art approaches of privacy preservation, (b) create privacy checking rules to detect composition attacks, (c) healthcare data generation and preparation, (d) designing and developing an open source, lightweight tool that connects to loosely coupled data sources and provides data exploration to already anonymized data sources for a possible breach of confidentiality.

SUBJECT AREA: Privacy Preservation

KEYWORDS: privacy, composition attacks, k-anonymity, l-diversity, quasi identifiers

ΠΕΡΙΛΗΨΗ

Η προστασία των δεδομένων στον κλάδο της υγειονομικής περίθαλψης δεν είναι εύκολη υπόθεση. Οι κάτοχοι δεδομένων υγειονομικής περίθαλψης πρέπει να εξισορροπούν την προστασία του απορρήτου των ασθενών, παρέχοντας ποιοτική περίθαλψη ασθενών καθώς και να πληρούν τις αυστηρές απαιτήσεις που ορίζονται από την HIPAA και άλλους κανονισμούς, όπως ο Γενικός Κανονισμός Προστασίας Δεδομένων (GDPR) της ΕΕ. Εάν δεν πληρούνται οι προϋποθέσεις για την δημοσιοποίηση των δεδομένων που οριζουν οι οργανισμοί, επιβάλλονται βαριές κυρώσεις και πρόστιμα. Υπάρχει επίσης ανάγκη να τα δεδομένα αυτά να δημοσιεύονται τακτικά για ερευνητικούς σκοπούς που θα οδηγήσουν σε καλύτερες υπηρεσίες υγειονομικής περίθαλψης. Μια πρόκληση είναι να μπορούμε να παρέχουμε αποτελεσματική προστασία της ιδιωτικότητας στα προσωπικά δεδομένα του ασθενούς. Για να ικανοποιηθεί αυτή η απαίτηση, εφαρμόζονται πολλές τεχνικές ανωνυμοποίησης όπως η k-anonymity και η l-diversity. Εάν τα δημοσιευμένα σύνολα δεδομένων είναι ανεξάρτητα και περιέχουν πληροφορίες για το ίδιο άτομο, τότε τα δεδομένα εξακολουθούν να είναι ευάλωτα σε composition attacks. Η μελέτη αυτή υιοθετει τεχνολογίες κατανεμημένων βάσεων δεδομένων για την ανάπτυξη ενός συστήματος, για τη σύνδεση, την ανάκτηση και τη διερεύνηση της διατήρησης της ιδιωτικότητας των δεδομένων. Η διατριβή θα εκτελέσει τα ακόλουθα: (α) τη διερεύνηση προσεγγίσεων για τη διατήρηση της ιδιωτικότητας για δεδομένα υγειονομικής περίθαλψης, (β) τη δημιουργία κανόνων ελέγχου ιδιωτικότητας για τον εντοπισμό composition attacks, (γ) τη δημιουργία και προετοιμασία δεδομένων υγειονομικής περίθαλψης, (δ) τον σχεδιασμό και ανάπτυξη ανοιχτού κώδικα λογισμικού, το οποίο συνδέεται με κατανεμημένες βάσεις δεδομένων και παρέχει εξερεύνηση των δεδομένων σε ήδη ανωνυμοποιημένες πηγές δεδομένων για πιθανή παραβίαση του απορρήτου.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Προστασία απορρήτου

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: διατήρηση απορρήτου, composition attacks, k-anonymity, l-diversity, quasi identifiers

Στη Χριστίνα

ACKNOWLEDGEMENTS

For the completion of this Thesis, the knowledge transfer and an amazing journey to the research world, I would really like to thank the supervising professor Thodori Dalamaga. I would also like to thank the Senior Researcher Manoli Terroviti for the collaboration, for his insightful comments and his valuable feedback on drafts of this work.

ΠΕΡΙΕΧΟΜΕΝΑ

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

Organizations dealing with healthcare data across the globe collect and use large amounts of data about their patients. This is a valuable asset to the organizations such as hospitals and medical centers, since this kind of data can be mined to extract a lot of insights about their patients. For example, exploring these data can throw light on patients' health issues, better drug usage and health services. This information is used by medical centers to provide value-added services, better and faster diagnosis and better medical therapies. This in turn results in higher profit and prestige for these health care centers. But these data contain patients' personal information. Personal information consists of first name, last name, identifiers like social security number, geographic and demographic information, and general sensitive information, like health issues, drug intake patterns, medical treatments and location data.

There is also a need to share these data responsibly, either for research purposes or medical reasons between healthcare centers, which means the data should be shared without revealing the identity of the patients. For example, a hospital's database could contain how many patients have reacted to a particular medical treatment or a drug. This information would be useful to a pharmaceutical company. However, these sensitive data cannot be shared or released in their original form due to legal, financial, compliance, and moral issues. For example, imagine that an insurance company could have access to medical treatment information about their clients and therefore increase the value of the insurance if a client of theirs is not doing well.

The databases that accomodate health data, usually consist of tables with (a) sensitive data, (b) personal information about patients and (c) patients visits. This information can be in separate tables or in a single table. Sensitivity comes in when the patient table is combined with a medical results table. The data in the tables contains four disjoint data sets:

1. Explicit identifiers (EI): Attributes that identify a record owner directly. These include attributes like social security number (SSN), insurance ID, and name.

2. Quasi-identifiers (QI): Attributes that include geographic and demographic information, phone numbers, and e-mail IDs. Quasi-identifiers are also defined as those attributes that are publicly available, for example, a voters database.

3. Sensitive data (SD): Attributes that contain confidential information about the record owner, such as health issues, financial status, and salary, which cannot be compromised at any cost.

4. Non Sensitive data (NSD): Data that is not sensitive for the given context.

When public sharing data, this can potentially violate individual privacy, harm the prestige and the reputation of the medical center and therefore lead to financial losses. Data leakage is becoming a major security issue. It can be either intentional or accidental exposure of sensitive information. An IDC survey (Porat et al., 2009) claims that data leakage is the number one threat, ranked higher than viruses, Trojan horses, and worms. The Natural healthcare chain Community Health Systems says that about 4.5 million pieces of “non-medical patient identification data” have been stolen on August 18, 2014 (www.infosecurity-magazine.com/news/45-million-records-stolen-from). HIPAA also published an article (https://www.hipaajournal.com/healthcare-data-breach-statistics), providing healthcare data breach statistics from October 2009 until December of 2021. Specifically, HIPAA states that those breaches have resulted in the loss, theft, exposure, or impermissible disclosure of 314,063,186 healthcare records. That equates to more than 94.63% of the 2021 population of the United States. In **Table 1** one can see the largest healthcare data breaches concerning health data providers from 2009-2021.

Table 1: Largest Healthcare Data Breaches (2009-2021) provided by HIPAA

	Name of Breached Entity	Year	Individuals Affected
1	University of California, Los Angeles Health	2015	4.500.000
2	Advocate Health and Hospitals Corporation, d/b/a Advocate Medical Group	2013	4.029.530
3	Banner Health	2016	3.620.000
4	Forefront Dermatology, S.C	2021	2.413.553
5	21st Century Oncology	2016	2.213.597

To address this issue and the privacy of an individual’s data, governments across the globe have mandated regulations that organizations have to apply. These regulations were created from entities like HIPAA in the United States, FIPPA in Canada, U.S. Declaration of Human Rights, and the EU’s Data Protection Directive. Organizations that collect and use data need to look into methods and tools to anonymize sensitive data. Those methods are referred to as anonymization techniques and are a critical piece of healthcare data publishing: it permits the sharing of data for secondary purposes with privacy assurances.

Anonymization is a set of techniques used to modify the original data in such a manner that it separates the explicit and the quasi identifiers from the sensitive data. There is a range of anonymization techniques that one can perform. As a result, an adversary will not be able to easily identify the record owner from his sensitive data. The intent is that anonymized data can be shared freely with third parties, who can perform their own analysis on the data. When anonymization is applied, the original data lose some of their utility. But this is a mandatory step. So, the balance of the data utility and data privacy should be kept in mind when performing an anonymization technique. Privacy goals are set by the data owners, and utility goals are set by data users.

However, preserving the privacy of the data is full of pitfalls. For example, simply removing explicit identifiers such as names and addresses does not guarantee to protect privacy since the remaining information (such as zip code, gender and date of birth) may still identify a person uniquely when combined with additional information (such as voter registration records). This is called composition attack. These attacks can be easily applied when dealing with loosely coupled data sources, since the data owners have applied anonymization techniques without the knowledge of other independent anonymized releases of data sets. As data publishing becomes more commonly deployed, it is increasingly difficult to keep track of all the organizations that publish anonymized datasets involving a given individual or entity and sets of data that are vulnerable to composition attacks will become increasingly difficult to use safely.

We explore some composition attack scenarios which are based on sample health care data, since it is very difficult to acquire and use real health data from third party organizations for research purposes. We assume the existence of data tables, each belonging to different data providers. They are k-anonymized and l-diversified independently and this is done from the data owners, which guarantees that the data can be published and also that no third party had access to the original data in order to anonymize them. The anonymization of the data includes three different techniques: generalization, suppression and subsampling.

However, those anonymization techniques do not guarantee that the data will not be vulnerable to composition attacks. As composition attack we refer to an attack on personal privacy which joins independent datasets in order to link personal information to a specific person. For example, a patient might have visited two hospitals for the same disease, and his personal information is in both the datasets that could be published. Those datasets are independently anonymized and distributed by the two hospitals. Although removing identifying attributes like names, SSN and e-mails, does not guarantee that the sensitive data may not be linked with other publicly available data sources. The attributes that can be efficiently used to create such links are gender, zip code, and age, which are called quasi-identifiers. This kind of attack is very difficult to identify since each party that publicly released the data, thinks the privacy preservation measures it took, are enough. To handle these difficulties, we will define some rules, that were derived from the scenarios, that can detect a possible composition attack.

This Thesis presents a framework to support data exploration for detecting early composition attacks and also supports privacy checking for already retrieved data from independent data sources. A first challenge here is to identify real life composition attack scenarios in loosely coupled data sources, concentrating on health care data. A second challenge is to generate sample anonymized health care data that are close to real data, since organizations are not willing to give any sample for research purposes. A third challenge is to produce rules that can identify possible composition attacks ad-hoc, on a set of data. A fourth challenge is to conclude on some best practices derived from the attacking scenarios that one can follow to better preserve the privacy of their data, without losing the utility of them.

The main contributions of this work can be summarized as follows:

- We identify and analyze privacy risks in various real life scenarios, that distribute, retrieve and use data from independent data sources, specifically for healthcare data.

- We define a set of rules to detect possible composition attacks early on a dataset. These rules can be applied to all tables and can help us quickly determine if data have high probability for breach of confidentiality. To apply those rules we need to also define a threshold, which is the maximum accepted risk probability. Checking rules takes O(N) time, where N is the total number of rows in the joint tables.

- We analyze composition attacking scenarios on loosely coupled data sources and we show how the set of rules for possible breach of confidentiality can be applied. The scenarios are based on data that are separated into two categories (1) non-overlapping value range quasi columns and (2) overlapping value range quasi columns.

- We further analyzed the attacking scenarios and their results to conclude some best practices one can follow when dealing with loosely coupled data sources. The best practices are mainly focused on (1) how the diversity of the quasi identifiers define the the power of the adversary, (2) the effect of the number common quasi identifiers between loosely coupled data sources, (3) the different anonymization approaches that one can follow when dealing with specific quasi identifiers categories and (4) how to choose the correct threshold.

- We have designed and developed an open source software tool, AnonymityPal, to support big data exploration for possible composition attacks and privacy checking, using state-of-the-art development frameworks, such as Presto DB, Spring Boot, Docker and Angular. The AnonymityPal supports distributed data sources that can be connected and queried via Presto DB, which is a distributed SQL query engine for running interactive analytic queries against data sources of all sizes ranging from gigabytes to petabytes. AnonymityPal offers both a query wizard and native queries, if a user needs to perform more complex queries. The main functionalities are two: the *What if scenarios* and the *Privacy check* of the data. The first service can help us determine if a dataset has high probability to be vulnerable to a composition attack. It implements the functionality of checking which rules for privacy preservation are applied in the dataset. The second service runs the k-anonymity and l-diversity algorithm in the background against the data the user retrieved from performing a query. The user can set and k, l as he wants.

Outline. In the following Chapter, we present the requisite background and related work. In Chapter 3, we present in detail the privacy preserving framework, emphasizing in the attacking scenarios and how this framework can be applied and some best practices that one can follow when dealing with loosely coupled data sources. In Chapter 4, we present the sample health data generation and setup components of AnonymityPal, software that was created as a part of our work, to help us better explore health care datasets, privacy issues and exploratory scenarios based on quasi-identifiers. We also dive into the architecture design of our software and we analyze in detail each architectural layer. In Chapter 5, we demonstrate some basic exploration scenarios in AnonymityPal. We also explore an evaluation dataset and discuss our results. Finally, in Chapter 6 the conclusions and some future ideas regarding the current work are mentioned.

2 BACKGROUND AND RELATED WORK

2.1 Overview

In this chapter, we briefly present basic concepts of privacy preserving methodologies, including k-anonymity and l-diversity, and concentrating on examples in distributed healthcare data. We also discuss related work on existing privacy preserving approaches in health data publishing and we argue about the pros and cons of these approaches.

2.2 Privacy Preserving Methodologies

In several domain areas, there is nowadays an increasing need for data sharing across multiple data sources. However, such data sharing is subject to constraints imposed by privacy of individuals or data subjects as well as data confidentiality of institutions or data providers. Preservation of privacy, specifically in health care data, has emerged as an absolute prerequisite for the exchange of confidential information between different health care institutions. Such exchange is critical for health data analysis in order to improve patient outcomes, gaining valuable insights through patient demographics analysis, and improving the quality of life.

To preserve the privacy of the data, several techniques are often required to reduce the risk of identifying sensitive information about individuals. Therefore, the data owner can first modify the data in a way to guarantee privacy, but still ensure that the data is of sufficient quality in order for the analytics to be useful and meaningful. Then, the data can be released to other parties safely. This process is called privacy-preserving data publishing.

Two well-known privacy preserving methodologies for data publishing are k-anonymity and l-diversity. Their goal is to generalize data values that themselves cannot be used to identify an individual, but, in combination with other data values (the so called quasi identifiers), there is high risk to identify an individual. Quasi identifiers aren't direct identifiers. Examples: area/zip code, date of birth.

2.2.1 K-Anonymity

The privacy preserving model of k-anonymity was defined (Samarati, 2001; SWEENEY, 2002). It is commonly described as a ‘hiding in the crowd’ approach to protect data in data sharing scenarios. For k-anonymity to be achieved there is a need to be at least k individuals in the dataset who share a set of attributes (e.g. values in different columns) that might become identifying for everyone. Thus, a k-anonymized data set has the property that each record is similar to at least another k-1 other records on the potentially identifying variables, and so it is difficult for an attacker to predict which record matches the individual.

Given a relational table T, an attribute can be characterized as:

- **Unique identifier:** an attribute, such as identity number, VAT number, that uniquely identifies an individual,
- **Quasi identifier:** an attribute that doesn’t identify an individual on its own but it can potentially identify an individual in combination with other values (e.g., from external data sources). Examples: zip code, age.
- **Sensitive attribute:** an attribute that should be protected, and whose value should not be related to an individual.

The set of all rows containing identical values for the set of quasi identifiers is referred to as an **equivalence class** for this set**.**

On **Table 2**, the *Health Condition* column is the sensitive attribute. The *Postcode*, *Age*, and *Gender* are the quasi-identifiers. The quasi-identifier attributes are those that in a combination with some background information may reveal the identity of an individual.

Table 2: Non Anonymized health records

Postcode	Age	Gender	Health Condition
13010	25	Male	Cardiovascular
13011	21	Male	Cardiovascular
13012	20	Male	Broken Leg
13012	22	Male	Broken Leg
45055	50	Male	Liver
45056	43	Male	Broken Leg
45057	45	Male	Cardiovascular
45057	45	Male	Cardiovascular
15030	45	Female	Cancer
15031	41	Female	Cancer
15032	43	Female	Cancer
45056	44	Male	Cancer

For example, suppose that a medical center has been requested to publish **Table 2**. Although the table does not explicitly include patients' names, we can have a privacy leak, given the following scenario. Assume, e.g., that Tom's personal information (Age = 20, Gender = Male, Postcode = 13012) is known to a person (i.e., the adversary) who tries to identify Tom's health condition (i.e., to breach his data privacy). Typically, when performing attacks, the adversary has some background knowledge for the person of interest that can be obtained easily from third channels like newspapers, local news etc. Assume also that the adversary knows that Tom visited this medical center earlier this month. Therefore, the adversary can easily identify that the row (13012, 20, Male, Broken Leg) is related to Tom, and, thus, Tom suffered from a broken leg.

In order to avoid cases such as the aforementioned, k-anonymity can help to prevent identity disclosure. By following the k-anonymity model, we generalize the values of quasi-identifiers so that each row becomes indistinguishable from at least k-1 other rows.

Generalization is the practice of removing identifying information that can be collected from data by reducing an attribute's specificity. The generalization of **Table 2** was done by hiding the last two numbers of the *Postcode* column and by grouping the values of the *Age* column into an age bracket.

Table 3: k-Anonymized health records, V = 4

Postcode	Age	Gender	Health Condition
130**	20-30	Male	Cardiovascular
130**	20-30	Male	Cardiovascular
130**	20-30	Male	Broken Leg
130**	20-30	Male	Broken Leg
450**	40-50	Male	Liver
450**	40-50	Male	Broken Leg
450**	40-50	Male	Cardiovascular
450**	40-50	Male	Cardiovascular
150**	40-50	Female	Cancer
150**	40-50	Female	Cancer
150**	40-50	Female	Cancer
150**	40-50	Female	Cancer

In **Table 3**, V = 4, meaning that for each equivalence class of quasi identifiers, there are at least k-1 rows(marked with different colors) with identical values in their quasi identifiers. For example, if Tom's personal information (Age = 20, Gender = Male, Postcode = 13012) is known to the adversary, she can conclude that the first four, light-blue, rows of **Table 3** could be related to Tom. But she will not be able to uncover Tom's health condition, since there are 2 different values for this sensitive attribute in the involved rows: Cardiovascular, Broken Leg.

k-Anonymity does not fully prevent leak of information in case of lack of diversity in the sensitive attribute values. Consider the following example: Tom knows Olivia, who lives just opposite the main neighborhood street, with zip code 15032. One day Olivia falls ill and is taken by ambulance to the medical center. Having seen the ambulance, Tom tries to discover what disease Olivia was diagnosed with. Tom discovers the 4-anonymous table of data published by the medical center (**Table 3**), and so he knows that one of the records in this table contains Olivia's data. Since Olivia is Tom's neighbor, he knows that Olivia is a 43-year-old American female who lives in the zip code 15032. Therefore, he can conclude that the last four, red rows of Table 2 could be related to Olivia. Since all of those patients have the same medical condition (cancer), Tom concludes that Olivia has cancer. For this reason, The L-Diversity privacy preservation model has been proposed.

2.2.2 L-Diversity

The l-diversity model (Machanavajjhala, Gehrke, Kifer and Venkitasubramaniam, 2006), combined with k-Anonymity, can provide a robust privacy preservation framework. An equivalence class satisfies l-Diversity if the probability of any row in this class is linked to a sensitive value is at most 1/l. In other words, the l parameter corresponds to the minimum number of distinct values in the sensitive attributes, within each equivalence class.

Let's consider **Table 4** The first equivalence class consists of two, light blue, rows (Age = 26-30, Gender = Male, Postcode = 130**) with two sensitive values (HIV, Cardiovascular). and the second equivalence class consists of the last four, purple, rows with three different values (HIV, Broken Arm, Cardiovascular). So, if an adversary has information about Olivia (Age = 39, Gender = Female, Postcode = 150**), he can conclude that the last four rows of **Table 4** could be related to Olivia. But he is unsure which of these records matches Olivia's, with a likelihood of 50% for Cardiovascular, 33,3% for Broken Arm and 33,3% for HIV. Thus, he is not able to clearly disclose the health condition attribute of Olivia.

Table 4: l-diversified table, l = 2

Postcode	Age	Gender	Health Condition
130**	26-30	Male	Cardiovascular
130**	26-30	Male	HIV
150**	36-40	Female	Cardiovascular
150**	36-40	Female	Broken Arm
150**	36-40	Female	HIV
150**	36-40	Female	Cardiovascular

2.3 Related Work

2.3.1 Privacy Preserving Approaches in Health Data

The privacy preservation of health data publishing and the anonymization of individual databases have been extensively studied in recent years. The main contributions are algorithms that transform a dataset to meet a privacy principle such as k-anonymity, l-diversity etc. Some others have developed and contributed in anonymization software tools that mitigate linking attacks.

In the area of the distributed databases, there are a number of potential approaches one may apply to enable privacy preservation. The two basic approaches are:

- Collect data from local data sources, and anonymize them globally to be published,
- Require data to be anonymized locally, collect and integrate the anonymized data to be published.

Both approaches hide challenges due to limited space and computational resources of the systems offering those privacy preserving services, security reasons and possible issues that may occur if the data owners don't trust the third party to access their data.

One approach that is proposed (Vardalachakis et al., 2019) is to collect the data from each data provider and then perform data anonymization independently as shown in **Fig. 1**. Data recipients can then query the individual anonymized databases or an integrated view of them. The main disadvantage of this approach is that the data are anonymized after getting access to them and hence in a case of security compromise of this tool, the un-anonymized dataset can be accessed from the attacker. Also, the corresponding amount of space that is needed to anonymize the data via this approach may exceed the computational limitations of many systems or can be costly to acquire the space and the computational power to anonymize the data from many sources.

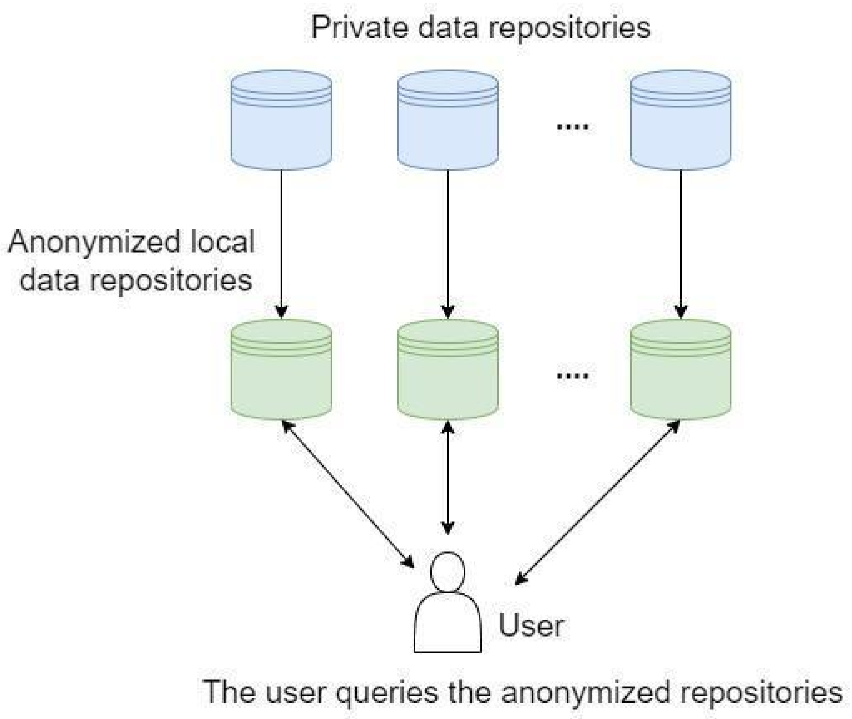

Private data repositories
....
Anonymized local data repositories
....
User
The user queries the anonymized repositories

Figure 1: ShinyAnonymizer

An alternative approach can be a centralized database (Fung, Wang, Chen and Yu, 2010), known more like data warehousing, assuming there is a third party that can be trusted by each of the data owners as shown in **Fig. 2** In this scenario, data owners send their data to the trusted third party where data integration and anonymization are performed. Clients then can query the centralized database. However, finding such a trusted third party is not always possible. Another drawback of this approach is that a possible compromise of the server by hackers could lead to a complete privacy loss for all the participating parties and data subjects.

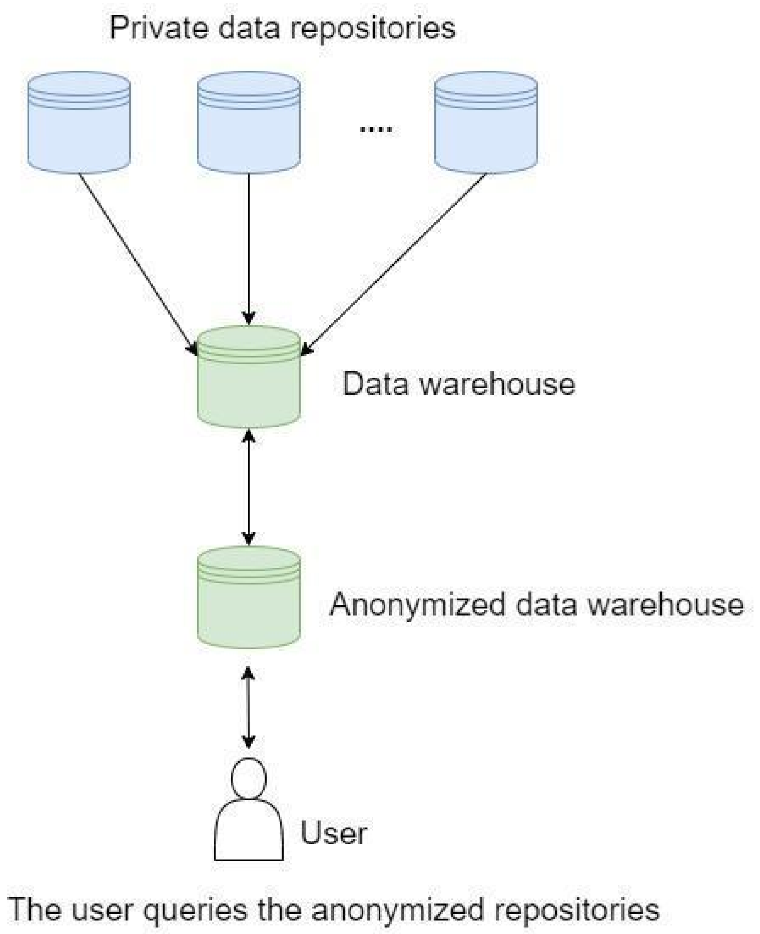

Private data repositories
....
Data warehouse
Anonymized data warehouse
User
The user queries the anonymized repositories

Figure 2: Centralized database anonymization approach

Moreover, as mentioned in the previous approach, the anonymization in a centralized way with so much data hides many challenges like handling limited storage, time to anonymize etc.

Another approach shown in **Fig. 3** is the creation of a virtual database (Jurczyk and Xiong, 2009) that collects through secure distributed protocols the health care data residing at individual databases, anonymizing them locally and publishing them to a server that can be queried. One drawback of this approach is the heavy infrastructure that is needed to apply the secure distributed communication protocol, due to the fact that the nodes have to use additional protocols in each step of computation. Therefore, an issue can be that the infrastructure can be costly in the era of big data, since there is a need for generalized tree construction. A generalized tree is a tree in which any given node can have any number of children (no fixed value of children). One of the most basic steps in the data anonymization process is constructing a generalized tree and since each node can have an unknown number of children, this tends to be harder to implement. The simplest method is sequentially exploring all of the Big Data. Another simple method is using a Hadoop-based generalized tree construction (Wang, Fung and Yu, 2006; Fung, Wang and Yu, 2007). However, both methods require too much time, and the tree construction is affected by the existence of fields containing errors or missing values. Moreover, this distributed anonymization protocol is expected to be run offline on an infrequent basis and so live analytics are not possible to be available to the user.

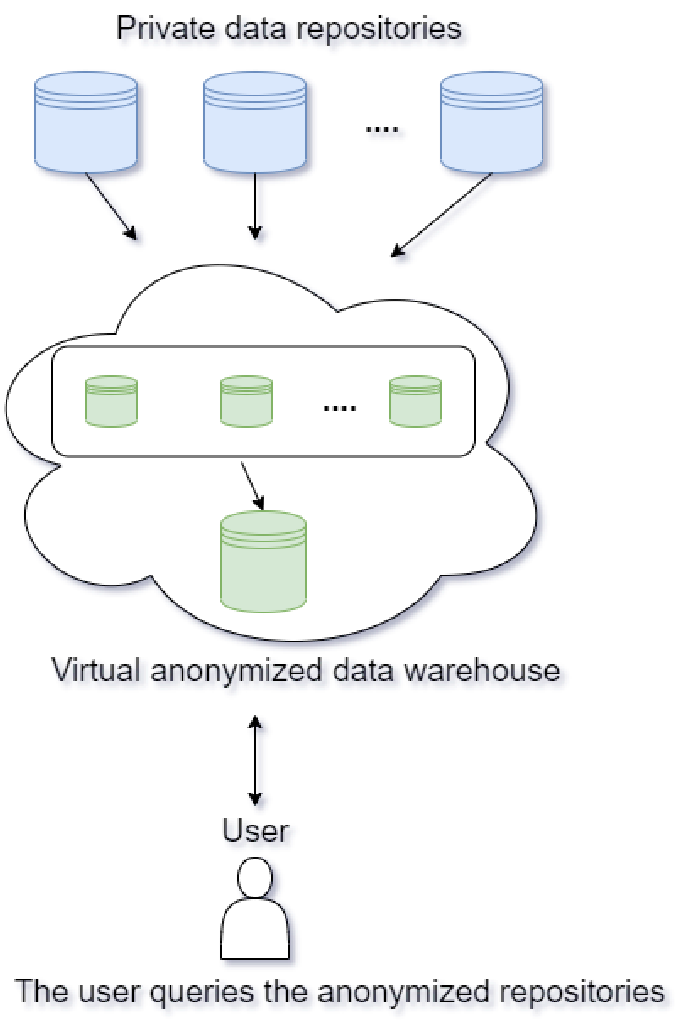

Private data repositories
....
....
Virtual anonymized data warehouse
User
The user queries the anonymized repositories

Figure 3: Virtual data warehouse approach

Another related solution that was proposed (Xue et al., 2011) is a system which consists of a set of sources, a central data collector, several data analysts, and of a central recovery authority. **Fig. 4** illustrates a high-level overview of this system. Specifically, the sources model a client-application that runs at the institutes which generate the data to be collected. Every medical institute provides its data to its corresponding source application as a set of (non-anonymized) data records. The data collector gathers this data from the sources, and makes it available to data analysts which perform the desired statistical data analysis. The system ensures that personal identifying data is replaced with pseudonyms once it leaves the sources, and that only the recovery authority is capable of linking a pseudonym to its respective identity.

This solution heavily relies on the existence of a PKI. PKI is a public key infrastructure that allows the system authentication for each data source, the data collector, the recovery authority, and the data analysts. It is assumed that every source, the central data collector, the recovery authority, and every external data analyst have a private and public key-pair for encryption and signing. The network communicating is over SSL/TLS connections between the client and server authentication.

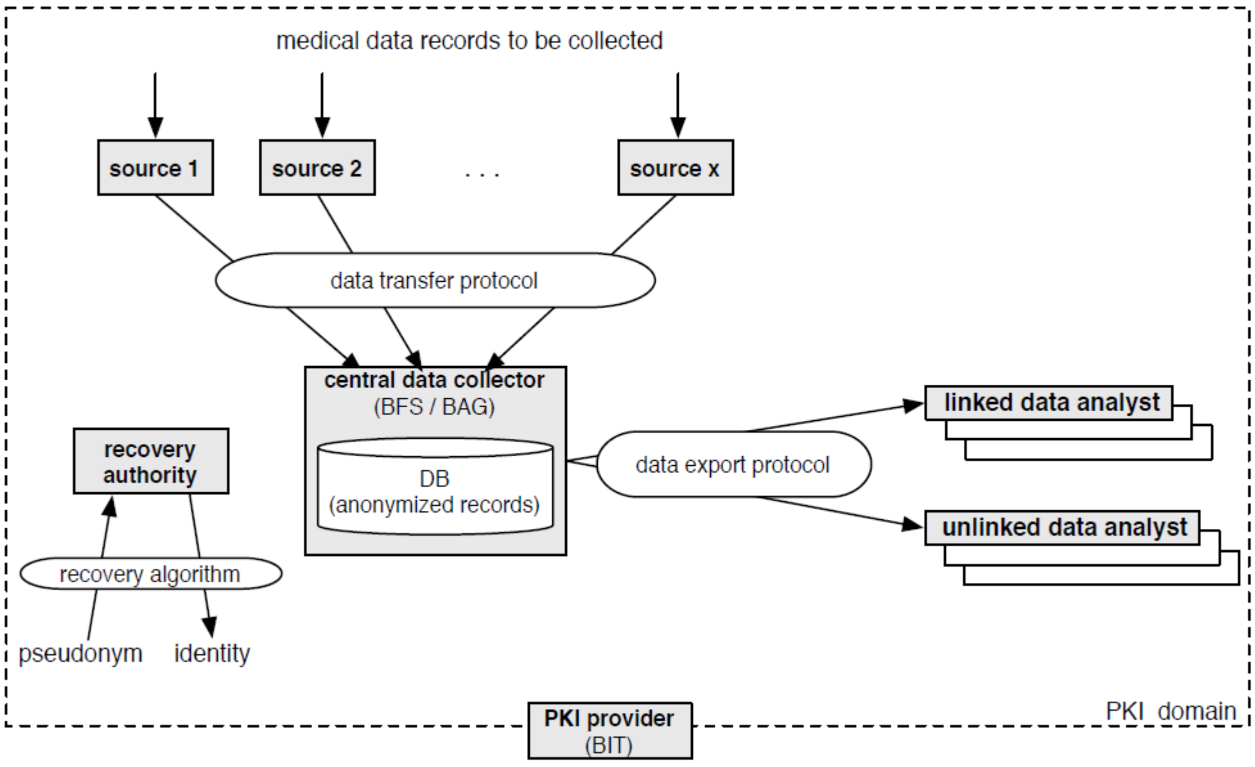


Figure 4: Overview of the System Model

For the aforementioned solution, the calculated number of anonymized patient data that will be collected per year from all hospitals, health insurance funds and care providers is assumed to be around 5 million, each being treated for 6 diseases. Every treatment of a disease generates 100 records on average. This means that every year 3 billion records have to be transmitted to and stored in the database. This already shows that the data has to be transferred regularly and not only once a year. The regular data transfer and the anonymization procedure create an overhead to the overall system. Also the data owners should trust their non-anonymized data to the central data collector which is not always possible.

There are some works focused on data anonymization of distributed data (Jiang and Clifton, 2006; Barhamgi, Benslimane, Ghedira and Gancarski, 2011) that studied the problem of anonymizing data vertically partitioned at multiple data providers without disclosing data from one site to the other.

A more advanced approach was made through the DataSHIELD project, which is a series of R libraries. DataSHIELD is a distributed infrastructure, and provides a novel technological solution that can circumvent some of the most basic challenges in facilitating the access of researchers and other health care professionals to individual level data. It follows the client-server approach, intended to enable the non-disclosive co-analysis of distributed sensitive research data(Gaye et al., 2014). The DataSHIELD relies on REST interfaces establishing the connection between the DataShield client and a DataShield server (OPAL server: opaldoc.obiba.org) installed at each institute, which receives the analysis command and executes it. The main objective of DataSHIELD is to secure data integration and retain the data where they are but run analytical queries as if they were combined in one database.

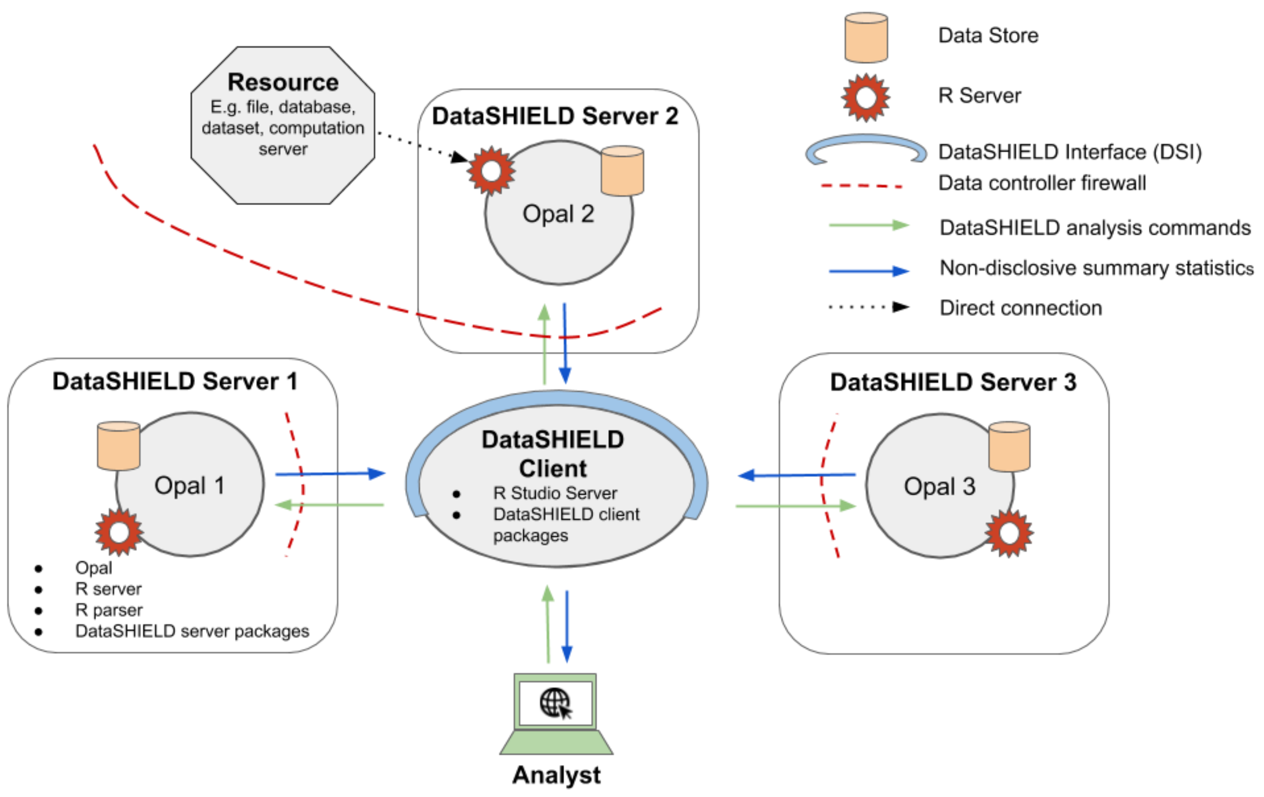


Figure 5: DataSHIELD architecture

The main drawback of DataSHIELD, as shown in **Fig. 5**, is that it is an inflexible solution with respect to other popular programming languages like R (**programming language dependency**). Moreover, the DataSHIELD packages are deployed to an OPAL server, which restricts the analysis to a predefined data source executing the R commands. In other words, the third party owners should have access to an OPAL server which has access to their health care data. On the one hand it offers secure connection but on the other hand creates infrastructure overhead.

In our work, we have designed and developed a lightweight tool, aiming to establish real time connection with the loosely coupled data sources, providing ad hoc exploratory visualization of data, and enabling the (re)use of distributed healthcare data, while data owners stay in control of their own data. As loosely coupled, we define the data sources that are on different databases and do not have any direct connection between them like a foreign key. The main assumption of our approach is that the data are already anonymized in their original corresponding location, no previously required technical knowledge is needed for the non-expert users about anonymization and also no risk of the original data exposure or unwanted data manipulation in case of a server compromise. The analytical tasks should visit the data sources via a big data query engine and execute the tasks.

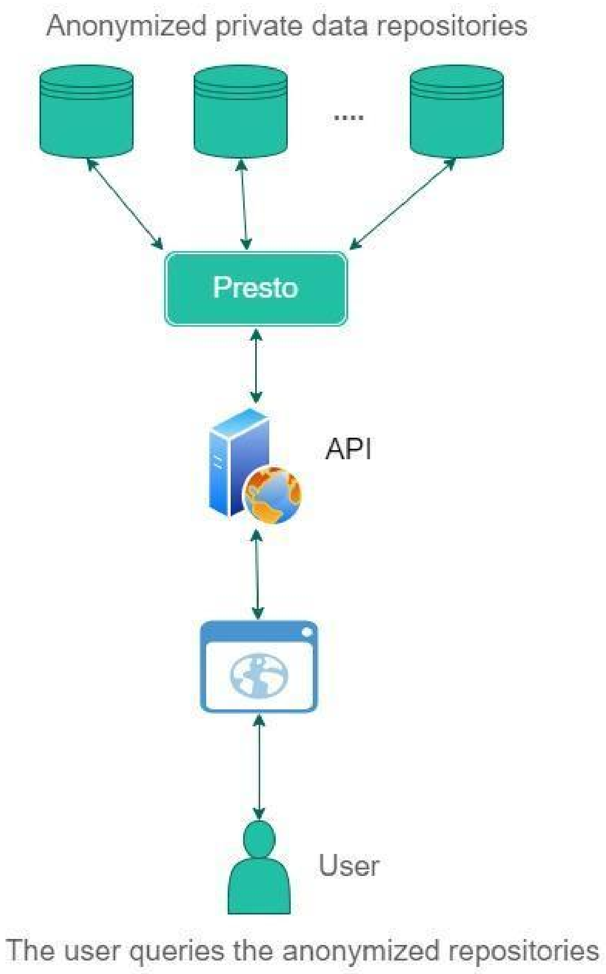

Anonymized private data repositories
....
Presto
API
User
The user queries the anonymized repositories

Figure 6: AnonymityPal approach

Fig. 6. gives a high-level overview of the architecture components, which will be discussed in the next sections.

Table 5 presents an overview of all close related approaches to our work. For each approach, we consider the following aspect:

- direct access to anonymized data,
- anonymization functionalities after collecting the third party data,
- infrastructure efficiency,
- alarms for possible privacy leaks on data access,
- security in case of a server compromise.

As shown on **Table 5**, none of the approaches satisfies all the criteria, since all the solutions hide challenges. For example, the approaches that apply anonymization in a centralized way like a data warehouse, a virtual database or an application's database, is likely to be a complex system, given the number of clients it communicates with, and the amount of data it collects. In practice, such a system is harder to secure. We believe that having trust for a central data collector to collect and anonymize is a major drawback of those systems.

Table 5: Solutions Overview

Solution	Direct Access To Anonymized Data	Applies anonymization after collecting the third party data	Lightweight - Cost efficient	Privacy Issues Detection	Secure in case of Server Compromise
Vardalachakis et al., 2019, Fig. 1	✖	✔	✔	✖	✖
Fung, Wang, Chen and Yu, 2010, Fig. 2	✖	✔	✖	✖	✖
Jurczyk and Xiong, 2009, Fig. 3	✖	✔	✖	✖	✔
Xue et al., 2011 Fig. 4	✖	✔	✖	✖	✔
DataSHIELD, Fig. 5	✔	✖	✖	✖	✔
Anonymity Pal, Fig. 6	✔	✖	✔	✔	✔

3 A PRIVACY PRESERVING FRAMEWORK IN LOOSELY COUPLED, ANONYMIZED HEALTH DATA SOURCES

3.1 Overview

In this chapter, we briefly present the basic concept of plausible attacks in loosely coupled anonymized data sources. Specifically, (a) we define a set of rules to check for privacy breach in such distributed settings, (b) we present a method that exploits these rules to decide whether privacy preservation holds, and (c) we analyze different scenarios of possible privacy violations and their probability of occurrence based on datasets characteristics.

3.2 Privacy Scenario Assumptions

The scenarios we are going to explore are based on generated sample healthcare data, since it was very difficult to acquire and use real health data from third party organizations for research purposes. We created two tables, each of one belonging to a different data owner like a hospital or medical center. The personal privacy in the tables is ensured by privacy-preserving data publishing methods and anonymization of the data at the time of publication. K-anonymization and l-diversification is applied to these tables separately. This guarantees to the data owners that if they publish their data, it would be difficult for an adversary to link data to a specific person. Also it guarantees that no third party will have access to the original data and be responsible to apply a set of anonymization techniques, which may hide security concerns. The anonymization of the data includes three different techniques:

- *Generalization*
  Reducing the precision of a field. For example, the age can be generalized to a broder time interval. For example, a 35-year-old person would have a value of 30-40 age range in the respective column. Generalization maintains the truthfulness of the data.

- *Suppression*
  Replacing a value in a data set with an asterisk value to indicate a missing value. For example, in a birth registry, a 60-year-old person would have a high probability of being unique. To protect this person we would suppress the age value with an "*".

- Subsampling
  Releasing only a random sample of the data set rather than the whole data set. For example, 50% of the data may be released instead of all the records.

However, those guarantees should not be enough to the data owners. An attack on personal privacy which uses independent datasets is called a composition attack. For example, a patient might have visited two hospitals for the same disease, and his information is independently anonymized and distributed by the two hospitals. Although removing identifying attributes like names, SSN and e-mails, does not guarantee that the sensitive data may not be linked with other publicly available data sources. The attributes that can be efficiently used to create such links are gender, zip code, and age, which are called quasi-identifiers.
This kind of attack is very difficult to be identified since each party that publicly released the data, thinks the privacy preservation measures it took were enough. Also, one cannot be sure if some of the data in their published sample dataset exist on another independent dataset as well. To handle these issues, we will introduce some rules that, if followed, can detect a possible composition attack. We also contribute with a software platform,

AnonymityPal, which offers privacy preservation services, by adopting the specified rules and by offering a basic privacy check via k-anonymity and l-diversity. That can help reduce the possibility of a composition attack when one is trying to combine information from independent data sources.

3.3 Composition attack on loosely coupled data sources

A distributed setting of loosely coupled data sources is vulnerable to the so-called composition attacks. Next, we give an example of such an attack.

Consider, for example, **Table 6** and **Table 7**, both 3-diversified, which publish anonymized patient datasets. Different colors are used for the equivalence classes defined by the attributes *Postcode, Age, Gender* and *Postcode, Age* of **Tables 6** and **7**, respectively.

Table 6: l-diversified table, l = 3

Postcode	Age	Gender	Health Condition
156**	26-30	Female	Cardiovascular
156**	26-30	Female	HIV
156**	26-30	Female	Broken Arm
156**	26-30	Female	Broken Arm
156**	36-40	Male	Cancer
156**	36-40	Male	Cardiovascular
156**	36-40	Male	Flu
156**	36-40	Male	Flu

Table 7: l-diversified table, l = 3

Postcode	Age	Nationality	Health Condition
156**	26-30	*	Eye Infection
156**	26-30	*	HIV
156**	26-30	*	Tuberculosis
156**	26-30	*	HIV
156**	41-45	*	Tuberculosis
156**	41-45	*	Cardiovascular
156**	41-45	*	Flu

156**	41-45	*	Flu

Assume that:

1. The attributes *Postcode*, *Age, Gender* are quasi-identifiers, while the *Health Condition* is the sensitive attribute.

2. The values of *Postcode, Age* and *Nationality* attributes have been generalized to satisfy the 3-diversity anonymization requirement for both tables, while *Gender* and *Health Condition* contain the raw (original) values.

3. The adversary has some background knowledge, meaning that he knows some quasi-identifier values of a particular person, for example, Olivia who is 30 years old, lives in an area with *Postcode* 15670, and she has recently visited both hospitals for the same health reason.

If the adversary performs a (left) join on both tables on the attributes (*Postcode, Age, Health Condition),* she will get the following records as a result:

156**	26-30	*	HIV

156**	26-30	*	HIV

She observes that the value of the sensitive attribute is the same (i.e., HIV) in both records. Based on this, she can conclude that Olivia has been infected by HIV, although she is not able to conclude which particular record corresponds to Olivia. To deal with such privacy leak problems, we next define a set of rules that detect composition attacks.

3.4 Definition of Privacy Checking Rules to Detect Composition Attacks

Let R be anonymized relational tables, k-anonymized and l-diversified. Then, consider:

- *RT*: the result table that occurs from joining and filtering the *R* tables on quasi identifiers and sensitive attribute
- *N*: the number of records in *RT*
- *V*: the number of distinct values of the sensitive attribute in *RT*
- *P*: Probability of occurrence for the dominant sensitive value (i.e., highest frequent value of the sensitive attribute)
- *Nv*: the number of records having a distinct value *v* of the sensitive attribute in *N*

For example, in **Table 6**, *N* = 8 and *V*= 5 (Cardiovascular, HIV, Broken Arm, Cancer, Flu), *Nv = 2* for Broken Arm, *Nv = 2* for Flu. Also P is 25% for Broken Arm and Flu, respectively. We next define the following privacy checking rules:

Rule 1: If *(N == 1) OR ((N >= 2) AND (V = 1))*, then *P = 1*, and, thus, we have a privacy leak, i.e., we can uniquely relate a sensitive value to the person in request.

Rule 2: If *(N >= 2) AND (V >= 2)*, then P = Nv / N, i.e., we can relate a sensitive value v to the person in request with probability P.

Consider the example in Section 3.2 where *RT* involves two records with the same sensitive value resulting from joining **Table 6** and **7**. In that example, *Rule 1* applies since *N == 2*, *V = 1,* and, thus, *P = 1*.

Consider now another *RT* that involves the following records:

Postcode	Age	Gender	Health Condition
132**	26-30	Female	Cardiovascular
132**	26-30	Female	Eye Infection
132**	26-30	Female	Broken Leg
132**	26-30	Female	Broken Leg

Here, *Rule 2* applies, since *N = 4* and *V = 3* (Cardiovascular, Eye Infection, Broken Leg), and so, we can relate the sensitive value Broken Leg to the person in request with probability P=50%. Similarly, P=25% for Cardiovascular, and P=25% for Eye Infection.

Typically, we can define a risk threshold *T* for *P*, such that in every case where *P>T* then we accept that there is high privacy risk and have a breach of confidentiality. We take the highest risk when *P=1 (Rule 1)*.

We next show how those rules can be used to decide whether privacy preservation holds.

Method Privacy checking (RT, N, V, T)

Input:
RT: the results table that occurred after joining and filtering the R tables,
N: the number of records in RT,
V: the number of distinct values of the sensitive attribute in *RT*,
T: the risk threshold
Output: FALSE, if there is high privacy risk

```
1: if N == 1 then
2:     return FALSE.
3: if N > = 2 and V == 1 then
4:     return FALSE.
5: for each value v of each equivalence sensitive class in RT do
6:     P = Nv / N
7:     if P >= T then
8:         return FALSE.
9: return TRUE.
```

Checking rules takes O(N) time, where N is the number of rows returned from the query performed by the attacker, because in the worst case the method should scan the whole results dataset to identify the unique sensitive values.

Note that in the example of Olivia, we consider cases where a patient has visited both medical centers for the same reason. In case this does not hold (i.e.,a patient has visited both medical centers but for different reasons), we should not include the sensitive attribute in the joint condition. The above rules, though, still hold.

3.5 Privacy Preserving Scenarios

Consider two tables that store patient information with the following attributes: *ZipCode, Gender, Nationality, Age, Marital Status, Blood Type* and *Health Condition*. The first six attributes, namely *ZipCode*, *Gender*, *Nationality*, *Age*, *Marital Status* and *Blood Type* are the quasi-identifiers. The *Health Condition* is the sensitive attribute. Assume that each row in the table refers to a patient, and that each patient has at most one row in each of the tables. Both tables have the column *ZipCode* in common.

Two scenarios are considered, depending on whether the quasi-identifier columns to be joined have overlapping value ranges or not. Note that we consider two identical value ranges to be non-overlapping. For example, two age ranges [10, 30] and [15, 35] are overlapping, while [10, 30] and [40, 50], or [10, 30] and [10, 30] are not.

3.5.1 Non-overlapping Value Range Quasi Columns

We consider 4 cases.

Case 1

Consider **Table 8** and **Table 9** which are anonymized and contain data segments from two hospitals, both including the same person's (e.g., Tom) health records. Assume that Tom's personal information (Marital Status = Married, Gender = Male, Zip Code = 13012) is known to the adversary. The adversary also knows that Tom visited two hospitals for medication. We will next demonstrate how we can have a breach of confidentiality:

1. Join left on the *ZipCode* and *Health Condition* columns from **Table 8** and **9.**
2. Filter the join results, selecting records based on "where ZipCode = 130** and Marital_Status = Married". Results are presented in **Table 10**.
3. Check the privacy checking rules for a breach of confidentiality: *Rule 1* matches our case since *N* = 1 and *V* = 1 (Diabetes), so *P* = 1. So, one we can uniquely relate Diabetes to the person in request (Tom).

Table 8: Anonymized data segment of Hospital A, l = 2, V = 2

Quasi-Identifiers			Sensitive Attribute
ZipCode	Age	Marital Status	Health Condition
130**	**20-30**	**Single**	**Cardiovascular**
130**	**20-30**	**Married**	**HIV**
130**	**20-30**	**Single**	**Diabetes**

130**	**20-30**	**Married**	**Diabetes**
150**	**30-40**	**Single**	**Broken Arm**
150**	**30-40**	**Single**	**Broken Pelvis**
150**	**60-70**	**Married**	**Broken Leg**
150**	**60-70**	**Married**	**Broken Arm**
160**	**50-60**	**Married**	**Eye Disease**
160**	**50-60**	**Married**	**Cardiovascular**
160**	**50-60**	**Married**	**Broken Arm**
450**	**40-50**	**Single**	**Cardiovascular**
450**	**40-50**	**Single**	**Diabetes**
450**	**40-50**	**Single**	**HIV**
771**	**30-40**	**Single**	**Diabetes**
771**	**30-40**	**Single**	**Cancer**
771**	**30-40**	**Single**	**HIV**

Table 9: Anonymized data segment of Hospital B, l = 2, V = 2

Quasi-Identifiers				**Sensitive Attribute**
ZipCode	Nationality	Gender	Blood Type	Health Condition
130**	**European**	**Male**	**A**	**Cancer**
130**	**European**	**Male**	**A**	**Diabetes**
130**	**European**	**Male**	**A**	**Cardiovascular**
150**	**European**	**Male**	**B**	**Broken Arm**
150**	**European**	**Male**	**B**	**HIV**
150**	**European**	**Female**	**AB**	**Cancer**
150**	**European**	**Female**	**AB**	**Diabetes**
160**	**European**	**Female**	**O**	**Cardiovascular**
160**	**European**	**Female**	**O**	**HIV**
160**	**European**	**Female**	**A**	**Cardiovascular**
160**	**European**	**Female**	**A**	**Broken Arm**
450**	**Asian**	**Female**	**A**	**Broken Arm**
450**	**Asian**	**Female**	**A**	**Broken Leg**

771**	**American**	**Male**	**O**	**Diabetes**
771**	**American**	**Male**	**O**	**Cancer**
771**	**American**	**Male**	**O**	**Cancer**

Table 10: Join of Tables 8 and 9 - query results

Quasi-Identifiers						**Sensitive Attribute**
ZipCode	Marital Status	Gender	Age	Nationality	Blood Type	Health Condition
130**	**Married**	**Male**	**20-30**	**European**	**A**	**Diabetes**

Case 2

If we take the above example but instead of Tom, the adversary knows Peter, who are single, results are:

130**	**Single**	**Male**	**20-30**	**European**	**A**	**Diabetes**
130**	**Single**	**Male**	**20-30**	**European**	**A**	**Cardiovascular**

In this case, checking privacy checking rules, *Rule 2* holds, with *N* = 2, *V* = 2 (Diabetes and Cardiovascular). Then, he can relate the sensitive attributes to Peter with 1/V = 50%. Depending on the risk threshold *T* for P, we may decide whether there is high privacy risk to have a breach of confidentiality.

Case 3

Consider **Table 9** and **Table 11**, which contain data segments from two hospitals. The adversary knows that a person visited both hospitals for medication and some personal information (Nationality = European) about him. The adversary will join (left) on the common column, *Health Condition,* and filter the Nationality column (quasi identifier) with 'European' as value (since he knows that Tom has the same value in both tables).

Table 11: Anonymized data segment of Hospital A, I = 2, V = 2

Quasi-Identifiers			**Sensitive Attribute**
Gender	Age	Marital Status	Health Condition
Male	**20-30**	**Single**	**Cardiovascular**
Female	**20-30**	**Married**	**HIV**
Male	**20-30**	**Single**	**Diabetes**
Female	**20-30**	**Married**	**Diabetes**

Female	**30-40**	**Single**	**Broken Arm**
Female	**30-40**	**Single**	**Broken Pelvis**
Male	**60-70**	**Married**	**Broken Leg**
Male	**60-70**	**Married**	**Broken Arm**
Male	**50-60**	**Married**	**Eye Disease**
Male	**50-60**	**Married**	**Cardiovascular**
Male	**50-60**	**Married**	**Broken Arm**
Male	**40-50**	**Single**	**Cardiovascular**
Male	**40-50**	**Single**	**Diabetes**
Male	**40-50**	**Single**	**HIV**
Female	**30-40**	**Single**	**Diabetes**
Female	**30-40**	**Single**	**Cancer**
Female	**30-40**	**Single**	**HIV**

Results are presented in **Table 12**. The results do not satisfy the k-anonymity, l-diversity(k, l < 2), since we have distinct records (e.g., 3rd row). Different colors are used for rows defined by the same sensitive value.

Table 12: Intersection of Tables 8 and 10 - query results

Marital Status	Age	Gender	Blood Type	ZipCode	Health Condition
Single	**20-30**	**Male**	**A**	**130****	**Diabetes**
Single	**20-30**	**Female**	**AB**	**150****	**Diabetes**
Married	**20-30**	**Male**	**A**	**130****	**Diabetes**
Married	**20-30**	**Female**	**AB**	**150****	**Diabetes**
Single	**20-30**	**Male**	**A**	**130****	**Cardiovascular**
Single	**20-30**	**Female**	**A**	**160****	**Cardiovascular**
Single	**20-30**	**Female**	**O**	**160****	**Cardiovascular**
Married	**50-60**	**Male**	**A**	**130****	**Cardiovascular**
Married	**50-60**	**Female**	**A**	**160****	**Cardiovascular**
Married	**50-60**	**Female**	**O**	**160****	**Cardiovascular**
Married	**20-30**	**Female**	**O**	**160****	**HIV**
Married	**20-30**	**Male**	**B**	**150****	**HIV**

Married	**50-60**	**Male**	**B**	**150****	**Broken Arm**
Married	**50-60**	**Female**	**A**	**160****	**Broken Arm**
Married	**60-70**	**Male**	**B**	**150****	**Broken Arm**
Married	**60-70**	**Female**	**A**	**160****	**Broken Arm**

Based on the privacy checking rules, *Rule 2* holds with *N* = 16 and *V* = 4. We, then, calculate the probability of occurrence for the dominant sensitive value (i.e., highest frequent value of the sensitive attribute) P for all distinct values if the sensitive attribute:

Table 13: Calculated P for Diabetes, HIV, Cardiovascular and Broken Arm

	Diabetes	**HIV**	**Cardiovascular**	**Broken Arm**
P	4/16 = 25%	2/16 = 12.5%	6/16 = 37.5%	4/15 = 25%

The dominant sensitive value (highest frequent value of the sensitive attribute) is Cardiovascular. Depending on the risk threshold *T* for *P*, we may decide whether there is high privacy risk to have a breach of confidentiality.

Note that if the adversary needs to increase the probability of finding out the reason Tom visited both hospitals, he should find more info about him, i.e., find values for more quasi identifiers. In the above example, if he had the extra information of the person's age (*Age* = 60-70), the matching rule would be *Rule 1* since it would have been *N* = 1 and *V* = 1 (Broken Arm), and so *P* = 1.

Married	**60-70**	**Male**	**B**	**150****	**Broken Arm**

Case 4

Consider **Table 8** and **Table 9**, as well as Tom's personal information (*Marital Status = Married, Nationality = European*), which are known to the adversary. It is also known that Tom visited two hospitals for a different medical reason. Thus, the adversary will not join left on the sensitive column. In the same way as in previous cases, the adversary follows the same steps with the only difference being that he is looking for records that have different values for the sensitive attributes of **Tables 8** and **9**.

The adversary joins (left) **Table 8** with **9** on the *Nationality* column. Next, he filters the joint results with a WHERE statement pointing to *Nationality = European* and with *Marital Status = Single*. From the results, the adversary tries to identify the rows from Table A that have different sensitive attributes from Table B. The final results are presented in **Table 14**.

Table 14: Query Results

Age	Health Condition Table A	Health Condition Table B	Gender	Blood Type	ZipCode
20-30	**Diabetes**	**Cancer**	**M**	**A**	**130****
20-30	**Diabetes**	**Cardiovascular**	**M**	**A**	**130****
20-30	**Diabetes**	**Broken Arm**	**M**	**B**	**150****
20-30	**Diabetes**	**HIV**	**M**	**B**	**150****
20-30	**Diabetes**	**Cancer**	**F**	**AB**	**150****
20-30	**Diabetes**	**Cardiovascular**	**F**	**A**	**160****
20-30	**Diabetes**	**Broken Arm**	**F**	**A**	**160****
20-30	**Diabetes**	**Cardiovascular**	**F**	**O**	**160****
20-30	**Diabetes**	**HIV**	**F**	**O**	**160****
20-30	**Cardiovascular**	**Cancer**	**M**	**A**	**130****
20-30	**Cardiovascular**	**Diabetes**	**M**	**A**	**130****
20-30	**Cardiovascular**	**Broken Arm**	**M**	**B**	**150****
20-30	**Cardiovascular**	**HIV**	**M**	**B**	**150****
20-30	**Cardiovascular**	**Cancer**	**F**	**AB**	**150****
20-30	**Cardiovascular**	**Diabetes**	**F**	**AB**	**150****
20-30	**Cardiovascular**	**Broken Arm**	**F**	**A**	**160****
20-30	**Cardiovascular**	**HIV**	**F**	**O**	**160****

Next, we will explore the following three scenarios: (a) investigating the results with the aim to identify the individual by ignoring the *Health Condition B*, (b) investigating the results with the aim to identify the individual by ignoring the *Health Condition A*, and (c) investigating the results with the aim to identify the individual, keeping both *Health Condition A* and *Health Condition B.*

Considering case (a), the adversary can check the *Health Condition of Table A*, and see that two values exist, Diabetes and Cardiovascular. So, he knows that the patient visited the first hospital either for Diabetes or Cardiovascular.

Based on privacy checking rules, *Rule 2* holds, with *N = 17* and *V = 2*. Thus, the probability of occurrence for the dominant sensitive value (i.e., highest frequent value of the sensitive attribute) P for all distinct values of the sensitive attribute is calculated as follows:

Table 15: Calculated P for Diabetes, and Cardiovascular

	Diabetes	**Cardiovascular**
P	9/17 = 53%	8/17 = 47%

The dominant sensitive value is *Diabetes, with* P = 53%.

Considering case (b), the adversary investigates the *Health Condition* of Table B, ignoring *Health Condition* of Table A. Based on the privacy checking rules, he concludes that *Rule 2* holds, with *N = 17* and *V = 5*. Then, the probability of occurrence for the dominant sensitive value P for all distinct values of the sensitive attribute is calculated as follows:

Table 16: Calculated P for Diabetes, HIV, Cardiovascular, Broken Arm and Cancer

	Diabetes	**HIV**	**Cardiovascular**	**Broken Arm**	**Cancer**
P	2/17 =11.5%	4/17 = 23.5%	3/17 = 18%	4/17 = 23.5%	4/17 = 23.5%

It is important here to note that the dominant sensitive values are HIV*,* Broken Arm and Cancer and thus, it is difficult to uniquely relate one to Tom.

Consider now the case (c) the adversary can see the results as multi-value attributes *Health Condition A - Health Condition B*. This means that the adversary is trying to identify which pairs of health conditions are the most likely to relate to Tom. Considering the presented results of **Table 14** and the privacy checking rules, Rule 2 holds with *N = 17* and *V = 7* (since there are 7 pairs of different health issues).

Table 17: Calculated P for pairs of diseases

	Diabetes - Cancer	**Diabetes - Cardiovascular**	**Diabetes - Broken Arm**	**Diabetes - HIV**	**Cardiovascular - Cancer**	**Cardiovascular - Broken Arm**	**Cardiovascular - HIV**
P	2/17 = 12%	5/17 = 29%	2/17 = 12%	2/17 = 12%	2/17 = 12%	2/17 = 12%	2/17 = 12%

The dominant sensitive (multi)value is Diabetes - Cardiovascular*, with* P = 29%, which is relatively low to be considered as a possible breach of confidentiality.

3.5.2 Overlapping Value Range Quasi Columns

We consider 2 cases for investigation:

Case 1

Consider **Tables 18** and **19**, which contain data segments from two hospitals. They have the *Lower* and *Upper Age* columns in common but the age grouping is different, and there are overlaps. Consider that the adversary knows Olivia (Age = 29, Marital Status = Married), which visited both hospitals for the same reason.

The adversary first joins left from **Table 18** to **19** on the sensitive attribute, the Health Condition, since the reason for visiting hospital A and B is the same. Then, he filters the results with a "where" statement as "between Lower_Age and Upper_Age". For example the where statement is like:

WHERE (AGE_VALUE BETWEEN TableA.LOWER_AGE AND TableA.UPPER_AGE, 29)
AND (AGE_VALUE BETWEEN TableB.LOWER_AGE AND TableB.UPPER_AGE)

In this example as AGE_VALUE we will use the 29 (Olivia's Age).

Table 18: Anonymized data segment of Hospital A, l = 2, V = 2

Quasi-Identifiers				Sensitive Attribute
ZipCode	Lower_Age	Upper_Age	Marital Status	Health Condition
130**	**20**	**30**	**Single**	**Cardiovascular**
130**	**20**	**30**	**Married**	**HIV**
130**	**20**	**30**	**Single**	**Diabetes**
130**	**20**	**30**	**Married**	**Diabetes**
150**	**30**	**40**	**Single**	**Broken Arm**
150**	**30**	**40**	**Single**	**Broken Pelvis**
150**	**60**	**70**	**Married**	**Broken Leg**
150**	**60**	**70**	**Married**	**Broken Arm**
160**	**50**	**60**	**Married**	**Eye Disease**
160**	**50**	**60**	**Married**	**Cardiovascular**
160**	**50**	**60**	**Married**	**Broken Arm**
450**	**40**	**50**	**Single**	**Cardiovascular**
450**	**40**	**50**	**Single**	**Diabetes**
450**	**40**	**50**	**Single**	**HIV**
771**	**30**	**40**	**Single**	**Diabetes**
771**	**30**	**40**	**Single**	**Cancer**
771**	**30**	**40**	**Single**	**HIV**

Table 19: Anonymized data segment of Hospital B, l = 2, V = 2

Quasi-Identifiers					Sensitive Attribute
Lower_Age	Upper_Age	Nationality	Gender	Blood Type	Health Condition
25	**35**	**European**	**Male**	**A**	**Cancer**
25	**35**	**European**	**Male**	**A**	**Diabetes**
25	**35**	**European**	**Male**	**A**	**Cardiovascular**
35	**45**	**European**	**Male**	**B**	**Broken Arm**
35	**45**	**European**	**Male**	**B**	**HIV**
55	**65**	**European**	**Female**	**AB**	**Cancer**
55	**65**	**European**	**Female**	**AB**	**Diabetes**
65	**75**	**European**	**Female**	**O**	**Cardiovascular**
65	**75**	**European**	**Female**	**O**	**HIV**
35	**45**	**European**	**Female**	**A**	**Cardiovascular**
35	**45**	**European**	**Female**	**A**	**Broken Arm**
45	**55**	**Asian**	**Female**	**A**	**Broken Arm**
45	**55**	**Asian**	**Female**	**A**	**Broken Leg**
25	**35**	**American**	**Male**	**O**	**Diabetes**
25	**35**	**American**	**Male**	**O**	**Cancer**
25	**35**	**American**	**Male**	**O**	**Cancer**

The joint results are presented in **Table 20**.

Table 20: Intersection of Tables 18 and 19 - query results

ZipCode	Marital Status	Lower Age Table A	Upper Age Table A	Lower Age Table B	Upper Age Table B	Nationality	Gende r	Blood Type	Health Condition
130**	**Single**	**20**	**30**	**25**	**35**	**American**	**M**	**O**	**Diabetes**
130**	**Single**	**20**	**30**	**25**	**35**	**Europeran**	**M**	**A**	**Cardiovascular**
130**	**Married**	**20**	**30**	**25**	**35**	**American**	**M**	**O**	**Diabetes**

Next we filter the results presented above with Marital Status = Married and we get one result back.

130**	**Married**	**20**	**30**	**25**	**35**	**American**	**M**	**O**	**Diabetes**

In this case, *Rule 1* holds, with *N* = 1, *V* = 1 (Diabetes), so *P* = 1. So, one we can uniquely relate Diabetes to the person in request (Olivia).

Case 2

Consider **Table 14** and **Table 17**. For every range (a,b) in *Lower* and *Upper Age* columns of **Table 21**, there is a set of (a_1,b_1), (a2, b2), (a3, b3), … in **Table 14**, where a_i<b, i=1,2,3, … and a_i >= a and b_i < b. As in *Case 1* adversary knows Olivia (Age = 29, Marital Status = Married).

Table 21: Anonymized data segment of Hospital B, I = 2, V = 2

Quasi-Identifiers					Sensitive Attribute
Lower_Age	Upper_Age	Nationality	Gender	Blood Type	Health Condition
30	**34**	**European**	**Male**	**A**	**Cancer**
30	**34**	**European**	**Male**	**A**	**Diabetes**
30	**34**	**European**	**Male**	**A**	**Cardiovascular**
25	**29**	**European**	**Male**	**B**	**Broken Arm**
25	**29**	**European**	**Male**	**B**	**HIV**
55	**59**	**European**	**Female**	**AB**	**Cancer**
55	**59**	**European**	**Female**	**AB**	**Diabetes**
60	**64**	**European**	**Female**	**O**	**Cardiovascular**
60	**64**	**European**	**Female**	**O**	**HIV**
35	**39**	**European**	**Female**	**A**	**Cardiovascular**
35	**39**	**European**	**Female**	**A**	**Broken Arm**
45	**49**	**Asian**	**Female**	**A**	**Broken Arm**
45	**49**	**Asian**	**Female**	**A**	**Broken Leg**
25	**29**	**American**	**Male**	**O**	**Diabetes**
25	**29**	**American**	**Male**	**O**	**Cancer**
25	**29**	**American**	**Male**	**O**	**Cancer**

Table 22. presents the joint result of tables **18.** and **21.** on the Health Condition, which were filtered with a "where" statement as "between Lower_Age and Upper_Age".

Table 22: Intersection of Tables 18 and 21 - query results

ZipCode	Marital Status	Lower Age Table A	Upper Age Table A	Lower Age Table B	Upper Age Table B	Nationality	Gender	Blood Type	Health Condition
130**	**Single**	**20**	**30**	**25**	**29**	**American**	**M**	**O**	**Diabetes**

130**	**Married**	**20**	**30**	**25**	**29**	**Europeran**	**M**	**B**	**HIV**
130**	**Married**	**20**	**30**	**25**	**29**	**American**	**M**	**O**	**Diabetes**

The adversary filters the results of **Table 22.** with *Marital Status* = Married, resulting in the last two rows.

130**	**Married**	**20**	**30**	**25**	**29**	**Europeran**	**M**	**B**	**HIV**
130**	**Married**	**20**	**30**	**25**	**29**	**American**	**M**	**O**	**Diabetes**

Rule *2* holds with *N* = 2, *V* = 2 (Diabetes and HIV). Then, he can relate the sensitive attributes to Olivia with *P* = 1 / *V* = ½ = 50%. Depending on the risk threshold *T* for *P*, we may decide whether there is high privacy risk to have a breach of confidentiality.

3.5.3 Best Practices

In the previous sections we analyzed different attacking scenarios and their results. From these, we can conclude some best practices to follow when dealing with loosely coupled anonymized data sources.

3.5.3.1 Quasi-identifiers value diversity - the power of the adversary

The number of quasi-identifies about which an adversary has some background information reflects the power of the adversary. In *Case 1,* the adversary knows three quasi-identifiers (Marital Status, Gender, Zip Code) and the attack resulted in high risk of having a breach of confidentiality. In *Case 2,* the adversary knows only one quasi-identifier (Nationality), and the attack resulted in low privacy risk. Note that in *Case 3,* which is an extension of *Case 2,* we show that if the adversary had background information for one more quasi-identifier, the attack could be successful with higher probability than that of Case 2.

The power of the adversary increases as he gains more knowledge for the quasi-identifier values. Consider the case where a patient suffers from chronic conditions. It would be easier for the adversary to know diagnosis values for patients with chronic conditions whose diagnoses keep repeating across visits than to know diagnosis values for patients who have a variety of different diseases. So, we expect his power to increase with the number of times a patient is visiting a medical center. Therefore, is it important to consider, as a best practice, the existence of value diversity across the quasi-identifiers for the patient visits.

3.5.3.2 The effect of the number common quasi-identifiers in loosely coupled data sources

One of the plausible reasons for the attack to be more severe in the aforementioned attack scenarios was the number of the common quasi-identifiers between the tables to join. This is because the adversary has more columns where he can join, depending on the background knowledge, and thus conclude to more specific results. One should also consider that they are cases with common quasi columns but with different value ranges. This means that an attack can be less effective if the value ranges differ a lot from one another, even if the number of common quasi columns is high. In the above experiments we have considered the scenario in which two anonymized releases of datasets contain information about overlapping age groups.

As data publishing becomes more widely accepted among organizations that would like to share data for research and collaborative purposes, it is possible that the number of anonymized releases available containing information about the same subset of people are more than just two. Specifically, when dealing with medical data, there is a high probability the quasi columns to be mostly common, since hospitals and medical centers use commonly accepted terminologies and concepts. That means that the possibility of common columns between two or more tables is high enough to let the adversary gather information about a target population and use the composition attack to deduce the sensitive attribute values.

In this case, when two or more medical centers are about to publish data that contain information for individuals that are in both of them, one should cooperate with the other in order to anonymize their data in combination, and not separately.

3.5.3.3 Different anonymization approaches on specific quasi-identifiers categories

From the cases described in the previous sections, we can derive some cases where we can use different anonymization approaches. We focus on (a) low diversity quasi-identifier values and (b) longitudinal data.

(a) We can have some quasi-identifiers that can only have low diversity in their values, like the Marital Status (Single, Married) and Gender (Male, Female). As we saw in the aforementioned cases, the knowledge of those low diversity quasi-identifiers played a crucial role to conclude the sensitive value with high percentage. To avoid that, we can replace male and female values by the value “Any gender”. In this case, the generalization plays the same role as cell suppression. The same approach can be used for the marital status column.

(b) In our scenarios, we had Table A and B, where each patient was included once in each of them. This is what makes the data cross-sectional. As cross-sectional we refer to the data that have records of many different individuals at a given time and each observation belongs to a different individual. When joining the two tables, the data are transformed to longitudinal, since the quasi-identifier values of a patient can be seen more than one time in the table (e.g one from table A and one from B). The data can also transit from cross-sectional to longitudinal when over a period of time the patients make multiple hospital visits. As we proved in our test scenarios, k-anonymity and l-diversity were not enough to protect the data. This happens because those privacy preservation methods treat a data set as cross-sectional. For example, consider that we know a patient who visited Hospital A and B for one or more times and those hospitals published their anonymized data. Joining those tables will result in having very similar records since the demographical data like gender, age and zip code are not changing. This will make it easier for the adversary to de-identify the patients’ sensitive value. Thus, trying to apply techniques for privacy preservation to a longitudinal data set using methods developed for cross-sectional data, you’ll either do too little to protect the patients properly, or you’ll need to apply strict anonymization techniques and distort the data to the point where it becomes useless.

Therefore, one should consider treating longitudinal data differently. In fact, the protection of longitudinal data is still a very difficult thing. One approach, as proposed (Tamersoy et al., 2012) is to use sequence alignment and clustering-based heuristics to anonymize longitudinal patient records. Another approach (Loukides, Gkoulalas-Divanis and Malin, 2010) that was proposed, extracts the clinical features that require protection, anonymizes each record that contains any clinical codes (e.g ICD codes), to ensure that it links to no fewer than k individuals with respect to these sets.

3.5.3.4 Choosing the right threshold

A crucial part of our explorational methodologies for deciding if there is a high probability of breach of confidentiality is to set the value of privacy risk T. So we need to discuss the practical aspects of choosing the risk threshold. The question is: what is the maximum acceptable probability of breach of confidentiality for the whole data set? In order to answer this question, we need to define the acceptable probability in every different publishing scenario.
The health data contain categorical data (e.g., diagnosis codes, procedure codes and often demographic information about the patient and the provider) as well as numeric data (e.g., patient's age, length of stay in hospital etc). Those data are (a) internally published between the medical centers and (b) publicly published. An important consideration for those cases is the law. Specific privacy laws and jurisdictions are applied in the health care sector. For example, the current health privacy regulations in Europe do not specify the acceptable risk and often rely on previous cases. For example in k-anonymity, there is a significant amount of previous defined k values that were used in different scenarios.

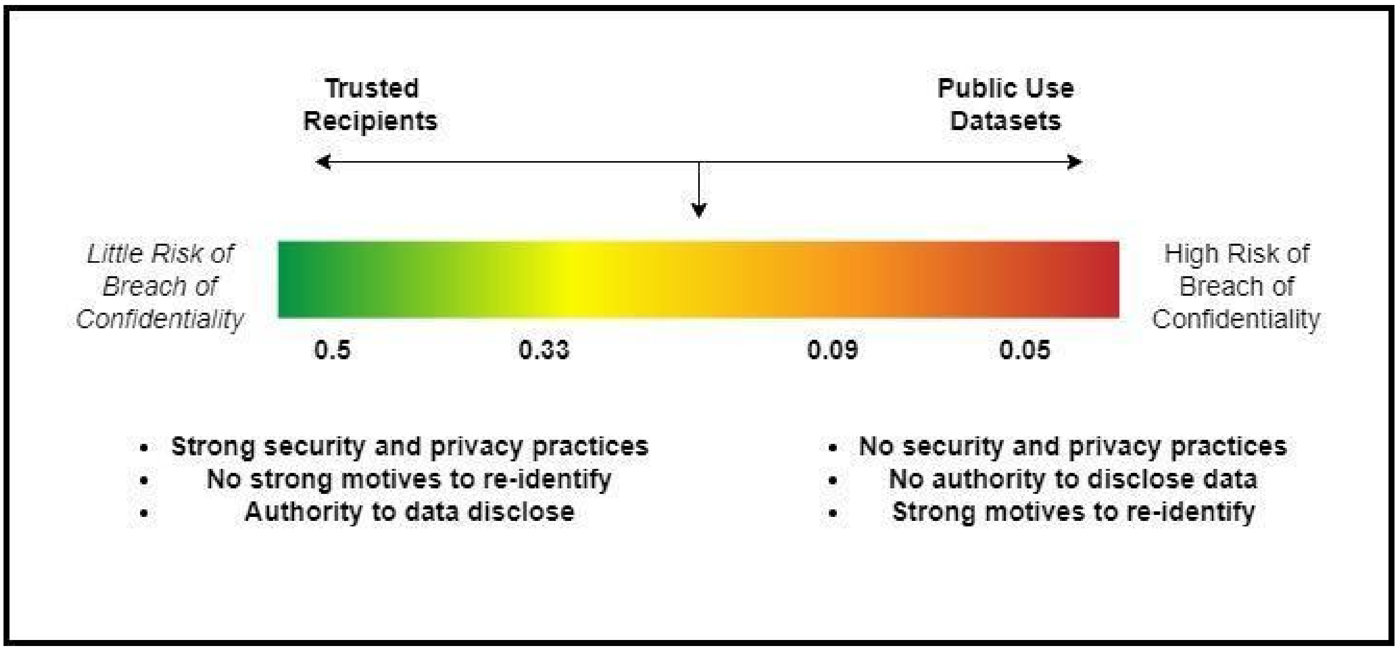
For more than two decades, data custodians, which are responsible for the aggregation, storage and use of data sets, have been releasing health data. During those periods' guidelines, regulations, policies and court cases have defined in a way what can be considered acceptable levels of risk. In health care data sets, this acceptable level of risk can be modified due to the fact that many fields are correlated or have natural constraints. For example, one treatment would often precede another or certain drugs given in combination. There are correlations among drugs and diagnoses. For example, a health care professional can determine the drugs that a patient is likely to take from his diagnosis and to some extent vice versa as well. Thus, generalization or suppression of pharmacy data may not be effective if diagnosis data are in the dataset. In those cases, both fields (e.g. drug and diagnosis) would need to be generalized or suppressed in a compatible way to decrease the probability of predicting one from the other. Note that many pieces of information in medical records can be correlated. We should also be concerned with inferences. If a column in the dataset can be inferred from other information that is not inside the dataset, then that field should be considered a quasi-identifier. Examples of inferences would be age and gender from a medical lab test result because some lab tests are only applicable on men or women often within a particular age range.

Considering the above, one can see that it is difficult to define a constant threshold that covers most cases, especially on health datasets. Every scenario is different from the other considering (a) the sensitivity of the data, (b) if the adversary is an insider or an external attacker, (c) the motives of the adversary, (d) if strong privacy measures have been applied and (e) if an authority has applied data disclosure. So how can one choose the correct threshold?

On the one hand, previous disclosures of cancer registry data have used thresholds of 5% and 20% as acceptable for public release and research use, respectively (Howe, Lake and Shen, 2006; Emam, 2008; Howe, Lake, Lehnherr and Roney, 2002), depending on the nature of the data recipient. On the other hand, the HIPAA Privacy Rule method, which aims to create data that have a low probability of breach of confidentiality, used a threshold of 50% when analyzing the acceptable probability of breach of confidentiality, during the consultations prior to issuing privacy regulations. Using a threshold value equal to 0.5 would be considered quite high by most observers. Historically, in the United States, some data custodians have used a maximum probability for breach of confidentiality equivalent to 0.33 (Private Lives and Public Policies, 1993). In Canada, the release threshold that was adopted from the Population and Public Health (PPH) Surveillance Team, which provides epidemiological support across a variety of public health service areas, was 5% (Wilkinson, Green, Nowicki and Von Schindler, 2020), as it is the industry-accepted threshold for the public release of data when the negative consequences of successful breach of confidentiality is considered to be extremely high and could result in significant potential harms and injuries to the individual.

Taking all the above into account, we provide some thresholds in **Fig. 7** that can be applied in our decision rules of detecting composition attacks. The thresholds that are proposed, are set through the empirical data that was mentioned before, and can be applied depending on the scenario (e.g. if the recipient is trusted etc.).

Figure 7: Different maximum T values


Trusted Recipients
Public Use Datasets
Little Risk of Breach of Confidentiality
High Risk of Breach of Confidentiality
0.5
0.33
0.09
0.05
• Strong security and privacy practices
• No strong motives to re-identify
• Authority to data disclose
• No security and privacy practices
• No authority to disclose data
• Strong motives to re-identify

4 PLATFORM ARCHITECTURE AND SOFTWARE DESIGN

4.1 Overview

In this chapter, we present the sample data generation and setup components. We also dive into the architecture design of our software and we analyze in detail each architecture layer.

4.2 Data and Setup

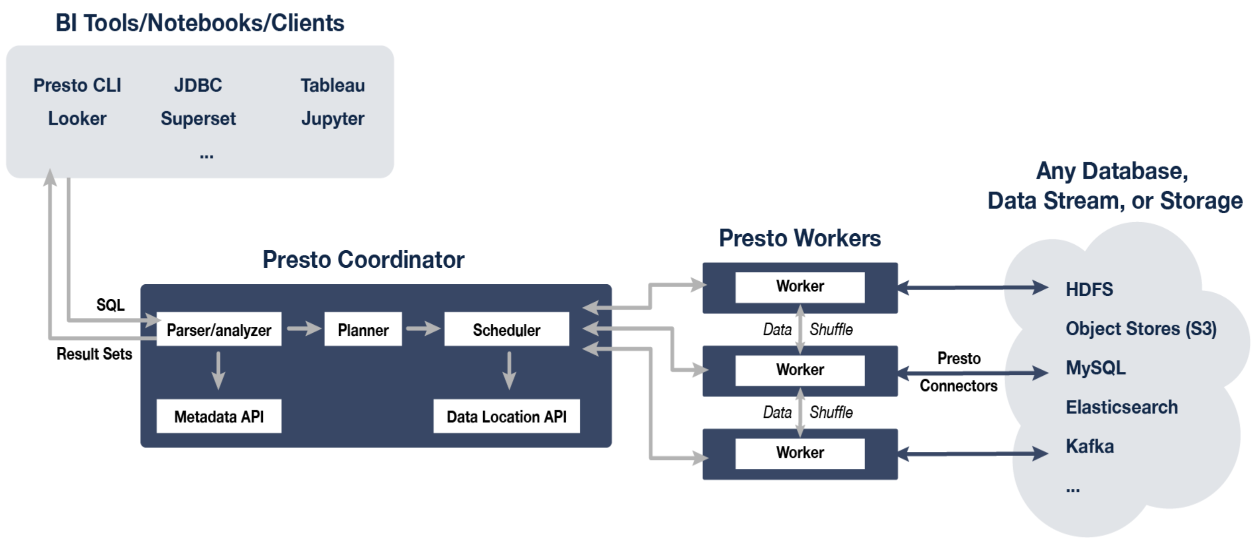
Due to the difficulty and expensines of accessing Electronic Medical Records (EMRs) due to privacy concerns and technical problems, we created and used a health data generator, which generated anonymized sql scripts with sample data for a) PostgreSQL and b) Mongodb database. The data are k-anonymized and l-diversified with the k, l = 3.

The scripts were run in the corresponding databases, when those databases were integrated through Docker containers. Docker is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another. A container image is a lightweight, standalone, executable package of software that includes everything needed to run an application.

Next to the database initialization, a docker image with PrestoDB is configured. Presto is the de facto standard for query federation that has been used for interactive queries, near-real-time data analysis, and large-scale data analysis. Presto is open source and can run interactive analytic queries against data sources of all sizes ranging from gigabytes to petabytes.

A single Presto query can combine data from multiple sources, allowing for analytics across entire organizations. It also supports pluggable connectors that provide data for queries. The requirements vary by connector. Connector examples include: Hive for HDFS or Object Stores (S3), MySQL, ElasticSearch, Cassandra, Kafka and more.

Figure 8: Presto Architecture



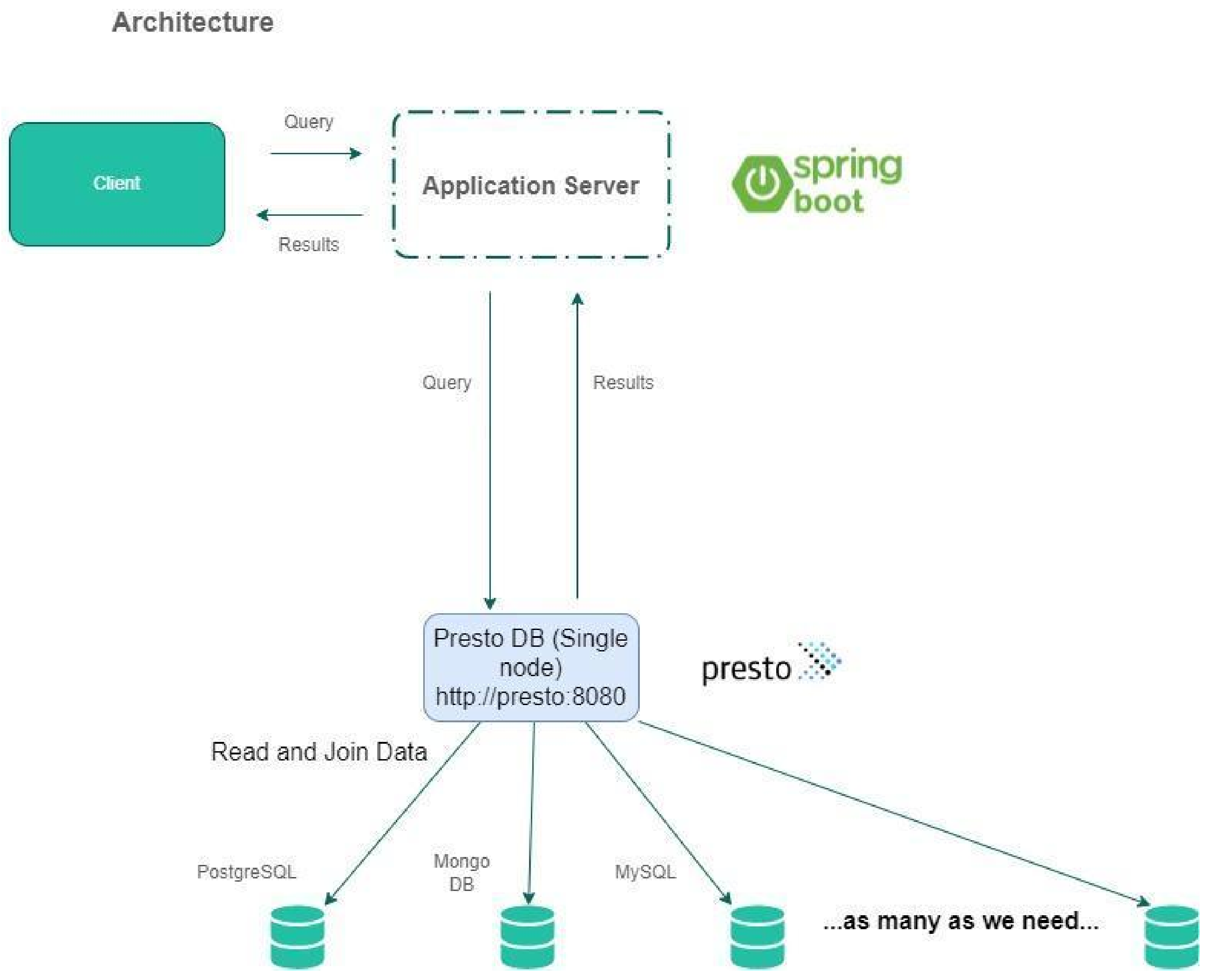
Presto is orchestrated as well in a docker-compose file with all the necessary configurations. One can configure and connect to Presto as many databases and nodes as he wants. Our Postgresql and MongoDB were connected with the PrestoDB simulating different nodes.

4.3 Architecture of our Software - System design

4.3.1 Presentation Layer

Fig. 9 illustrates the high level architecture diagram for AnonymityPal, with client-server architecture that has their logic separated into layers. The client side is implemented with Angular which is an application design framework and development platform for creating efficient and sophisticated single-page apps. The application server is developed via Spring Boot, which is built on top of the core Spring framework. It is a simplified and automated version of the spring framework. AnonymityPal follows a layered architecture in which each layer communicates to other layers (above or below in hierarchical order).

Figure 9: System Architecture of AnonymityPal

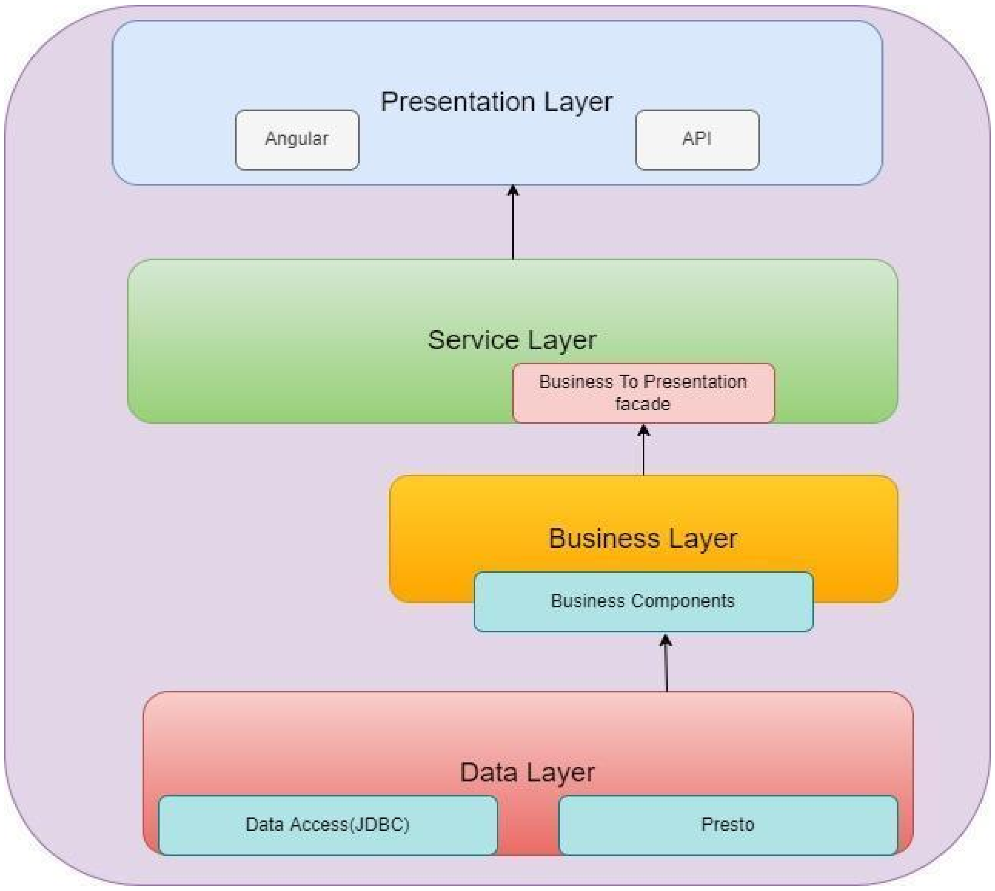


The application server consists of five layers:

1. Presentation Layer – UI components
2. Service Layer – JSON Translation
3. Business Layer – Business Logic, Validation & Authorization
4. Persistence Layer – Storage Logic
5. Database Layer – Actual Database

As shown in **Fig**. **10** the data layer communicates with the business layer, the business layer communicates with the service layer and the final results are sent to the presentation layer, where the user can interact with them.

Figure 10: Architecture Layers



The presentation layer is the top layer of the AnonymityPal architecture. It is developed and consists of Views. i.e., the front-end part of the application. It handles the HTTP requests. It is responsible for converting the JSON field's parameter to Typescript Objects and vice-versa. It passes the requests to the next layer. i.e., the service layer(or Rest API layer).

The presentation layer of AnonymityPal is developed with the latest Angular framework, following many Angular application architectures. Our goal was to design a single page application in a way to be maintainable, sustainable, easy development speed and ease of adding new features in the long run. To achieve these goals, we applied:

- ➔ Unidirectional data flow
- ➔ Abstractions between application layers - Core, Shared & Feature Modules
- ➔ Lazy Loading
- ➔ Modular design
- ➔ Dynamic Components

A hierarchical tree structure was used by implementing parent-children relationship concepts for our components. Therefore the data flow between each component with events (actions). The child component will send an action using @output and every component will accept it using @input. This is called unidirectional data flow.

There was a need to make sure our code is well-isolated and has a simple and clear dependency model. To achieve this, we used abstractions between the application layers, decoupling the presentation layer from the core layer. The core layer is responsible for encapsulating the state and behavior of an application and includes core modules and services. The abstraction layer will behave as a facade for the presentation layer. It means that it will expose an API and coordinate the communication between presentation layer components and the core of the application, since all the outside communication is happening in the presentation layer.

Since AnonymityPal was designed to be lightweight, the need for low response time and resources consumption was mandatory. To achieve that, we used Lazy Loading. Its purpose is to load the feature modules of the application on run-time. The response time, and the resources consumption are lower since the application loads the code bundle necessary at the time (in a splitted way) and when it is mandatory.

Moreover, to improve even more the performance and the loading time, we used dynamic components. This means that the location of the components in the application is not defined at build time. That means that the DOM is dynamically loading the components when we want them to. This also keeps the DOM clean.

4.3.2 Business Layer

The business layer contains all the business logic. It consists of service classes. First, it is responsible for validating the data that is provided through the service layer(or Rest API). Domain-Driven Design was used to design the business components, based on models of the underlying domain. The domain (business logic perceptive) is the center of this project and the business layers currently consists of four services:

1. ***Privacy service:*** logic to check if k-anonymity and l-diversity are satisfied
2. ***Presto service:*** logic to validate and run the sql queries on Presto
3. ***Loosely coupled sources privacy preservation service:*** logic to detect privacy preservation issues
4. ***Custom query builder service:*** logic to create the SQL command from the UI custom query builder

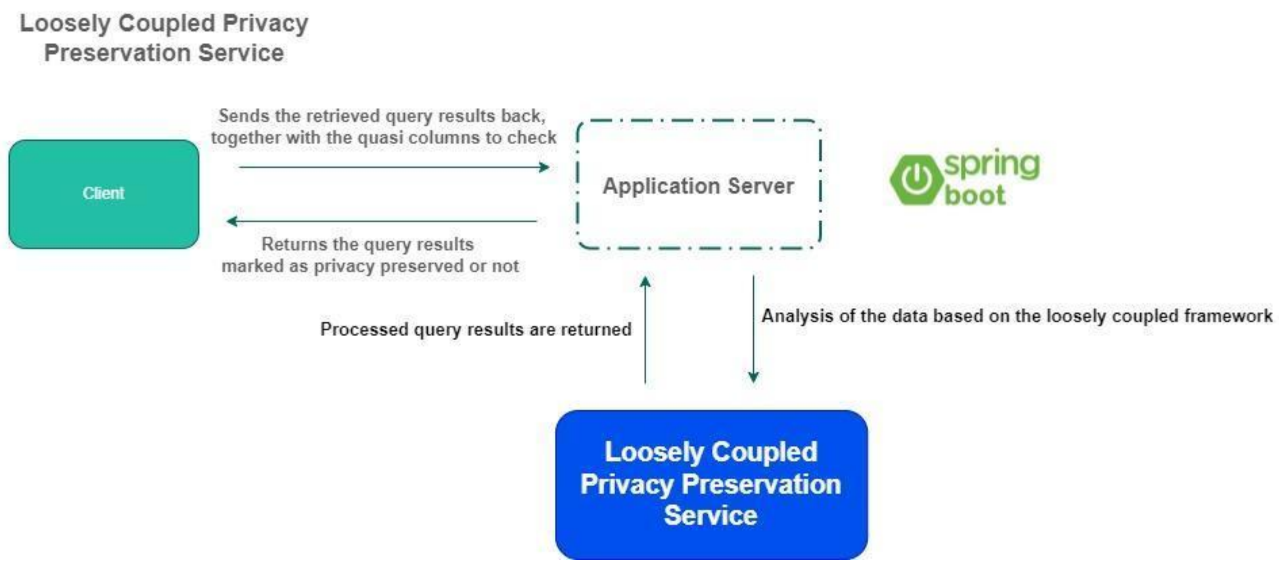
The main advantages of our design is communication, flexibility is based on object-oriented programming. So everything from the project will be an object and hence encapsulated and modular.

All the aforementioned services are between the rest layer and the persistence layer and are responsible to validate early the data that are passed through the rest layer and manipulate the data in a way that will keep the integrity of the database safe. The business layer is in control of the data structures. This means the data needed/produced by the business logic is often not in the right format for communicating to the user and/or other external APIs. Therefore Adapters are used, i.e. something that transforms the data to/from the business logic needs.

From the user interface one can investigate the degree of the privacy preservation of the returned rows. The adversary can choose the columns that may has/can find or if he has some background knowledge. The selected columns, as well as the previous returned rows, are sent to the application server and an analysis of those data is done based on the loosely coupled framework (rules defined in section 3.1).

From the business point of view the most important service in the Business layer is loosely coupled sources of privacy preservation service. This service is used to check if the data retrieved from a query keep their privacy preservation in case the adversary finds more background information for an individual. As demonstrated in **Fig. 11**, one can send previous retrieved data, the new information he acquired for an existing quasi column and the respective value, and in return the service will send back the rows marked as privacy preserved or not. This means that the service will first calculate the probability of privacy leak for the respective quasi columns and their values (sent by the user) and if the newly calculated probability of privacy preservation violates the set threshold, then the rows are marked as not privacy preserved.

Figure 11: Loosely Coupled Privacy Check


Loosely Coupled Privacy Preservation Service
Sends the retrieved query results back, together with the quasi columns to check
Client
Application Server
spring boot
Returns the query results marked as privacy preserved or not
Processed query results are returned
Analysis of the data based on the loosely coupled framework
Loosely Coupled Privacy Preservation Service

4.3.3 Persistence Layer and Database Layer

The persistence layer contains all the database storage logic. It is responsible for converting business objects to the database row and vice-versa. The main operations are read operations since the purpose of the tool is to retrieve information from the various databases. AnonymityPal was configured to handle the data that are retrieved dynamically. Since we do not know what is the underlying format of the data that are retrieved, first they are converted to strings and then the type of the value is resolved (e.g text, numeric or date).

The database layer contains all the databases such as MySQL, MongoDB, etc. This layer can contain multiple databases. It is used mainly for performing reading and joining operations in the available tables. The database layer has the responsibility to connect through various distributed databases via the Presto.

A JDBC driver has been configured to access Presto. All systems that connect to Presto with the JDBC driver must be granted access to query tables in the system.jdbc schema. When a SQL statement is to be executed, AnonymityPal is running it against one or more catalogs. Our examples include catalogs for Mongo and PostgreSQL data sources. When addressing a table in Presto, the fully-qualified table name is always rooted in a catalog. For example, a fully-qualified table name of mongodb.health_catalog_1.table would refer to the test patient data table in the health_catalog_1 schema in the Mongodb catalog. All catalogs are defined in properties files stored in the Presto configuration directory.

Schemas are a way to organize tables. Together, a catalog and schema define a set of tables that can be queried. When accessing a database such as PostgreSQL with Presto, a schema translates to the same concept in the target database. A table is a set of unordered rows which are organized into named columns with types. This is the same as in any relational database. The mapping from source data to tables is defined by the connector. When the database connections are established, the user can request data via the user interface or via APIs.

4.4 API

This layer includes multiple APIs for enabling connection to external data sources such as relational and no SQL databases. Those APIs go through Presto db in order to retrieve the data. In addition, the appropriate REST function calls and the corresponding APIs are available for programmatically retrieving data via plane SQL or via a query builder.

The respective REST functions are responsible for getting the data that were sent from the presentation layer, through the business layer, to finally reach Presto in order to run the SQL query against the corresponding catalogs. These APIs accept JSON for request payload and also send responses to JSON. JSON is the standard for transferring data. Almost every networked technology can use it: JavaScript has built-in methods to encode and decode JSON either through the Fetch API or another HTTP client. Server-side technologies have libraries that can decode JSON.

The designed endpoints are grouped to those that contain associated information. For example all the endpoints that concern presto are under the “/api/presto/” path and all endpoints that concern privacy are under “/api/privacy/”.

When errors occur, to eliminate confusion for API users, we handle errors gracefully and return HTTP response codes that indicate what kind of error occurred. This gives maintainers of the API enough information to understand the problem that’s occurred. A detailed API can be seen in **Table 23**.

Table 23: API calls and usability descriptions

HTTP request method	Rest API	Usability
GET	**/api/presto/getAvailableDbs**	**Retrieve available database names, connected to presto**
GET	**/api/presto/getAvailableDbSchemas**	**Retrieve available database schemas, connected to presto**
GET	**/api/presto/getAvailableSchemaTables**	**Retrieve available database schema tables,**

		connected to presto
GET	**/api/presto/getColumnsFromTable**	**Retrieve columns from a specific table**
GET	**/api/presto/getFilterOperations**	**Retrieve available SQL filter operations**
GET	**/api/presto/getJoinOperations**	**Retrieve available SQL join operations**
GET	**/api/queryservice/getQueryResults**	**Get native query results**
POST	**/api/queryservice/getCustomQueryResults**	**Post a custom query and retrieve the results**
POST	**/api/privacy/getQueryResultsPrivacyChecked**	**Check if the query results meet the privacy criteria**
POST	**/api/privacy/checkPrivacyPreservation**	**Check if the query results preserve their privacy if we gain knowledge for extra quasi-identifiers**

5 DEMONSTRATION SCENARIOS AND EVALUATION

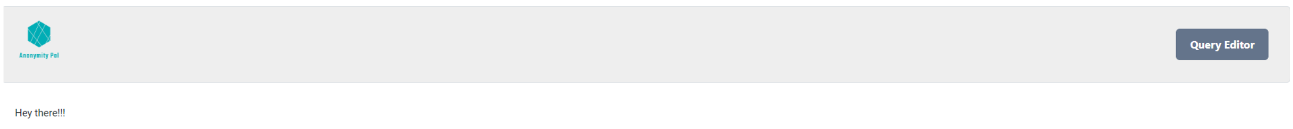
5.1 Overview

In this chapter, we demonstrate an exploration of two basic scenarios around healthcare data: (1) a scenario that has a possible breach of confidentiality and (2) one scenario without. We will then evaluate our results from the security researcher/adversary point of view.

5.2 Demonstration

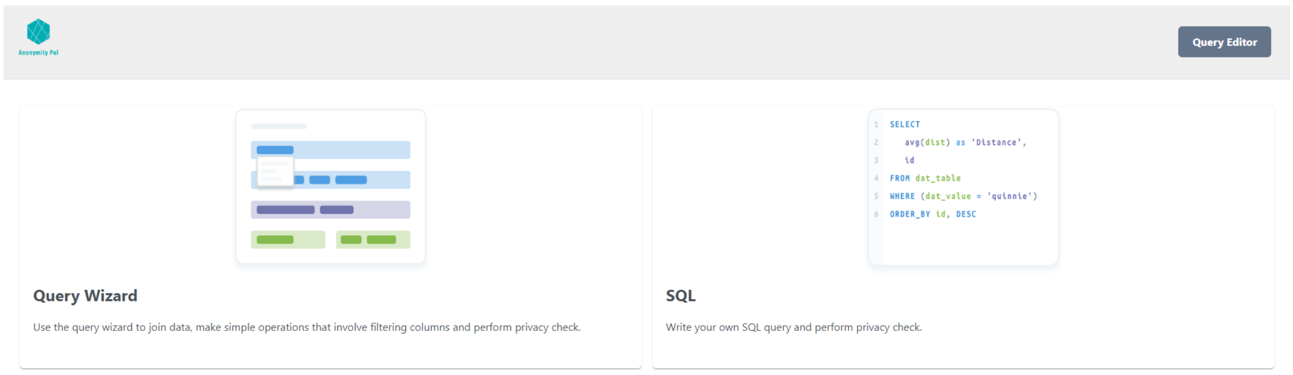
The first scenario that we will explore is the one from section 3.4 *Case 1*. We assume that the test data is loaded to the distributed databases, and Presto is configured correctly. The first step is to choose the tables that we want to explore. As shown in **Fig. 12** we click the 'Ask a Question' button, in order to choose the custom question or the native query as in **Fig. 13**.

Figure 12: Landing Page



Hey there!!!

Figure 13: Page to choose between custom or native query



Query Wizard

Use the query wizard to join data, make simple operations that involve filtering columns and perform privacy check.

```
SELECT
   avg(dist) as 'Distance',
   id
FROM dat_table
WHERE (dat_value = 'quinnie')
ORDER_BY id, DESC
```

SQL

Write your own SQL query and perform privacy check.

By choosing the **Query wizard**, we select the first database, the schema and the table of interest as shown in **Fig. 14, 15**. We chose the sample database of mongodb and next the schema of health_data_db_1 as in **Fig. 15**.

Figure 14: Selecting a database in custom query builder

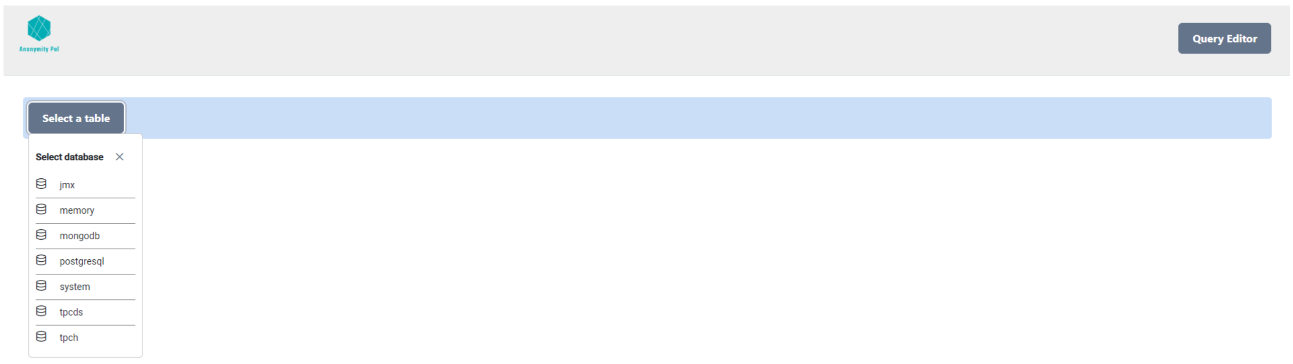
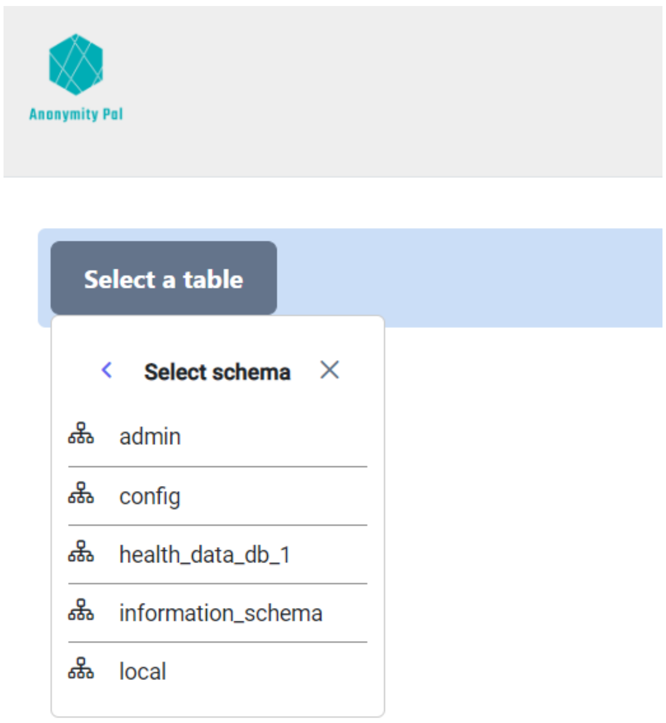
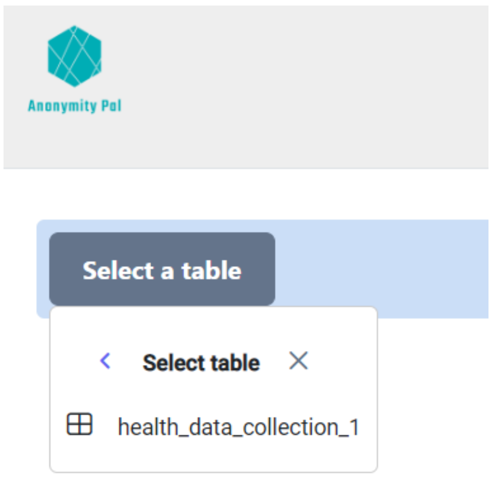


Figure 15: Selecting a database schema in custom query builder



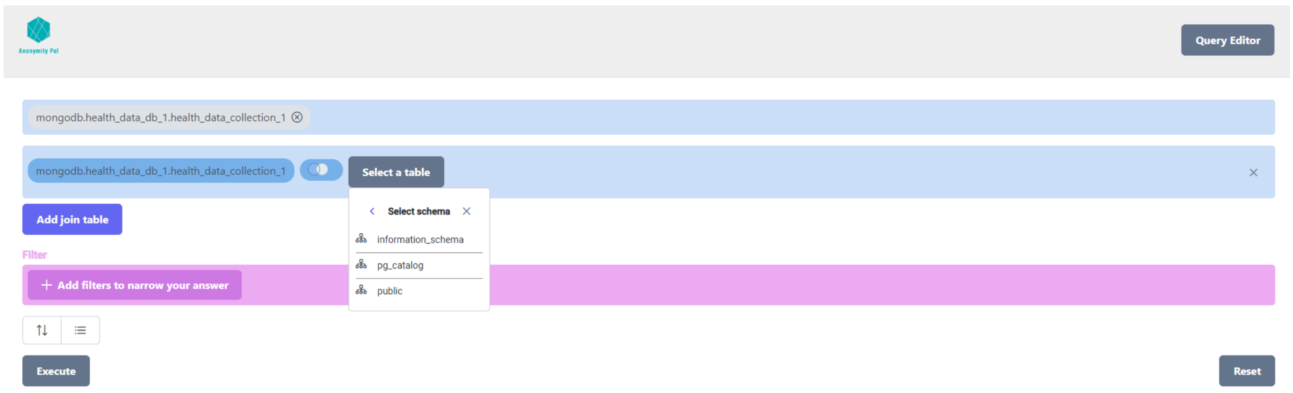
In the final step of the wizard for table selection, is to choose the actual table. As shown in **Fig. 16**, there is only one table/collection available in the sample database, which we are going to choose.

Figure 16: Selecting a database schema in custom query builder



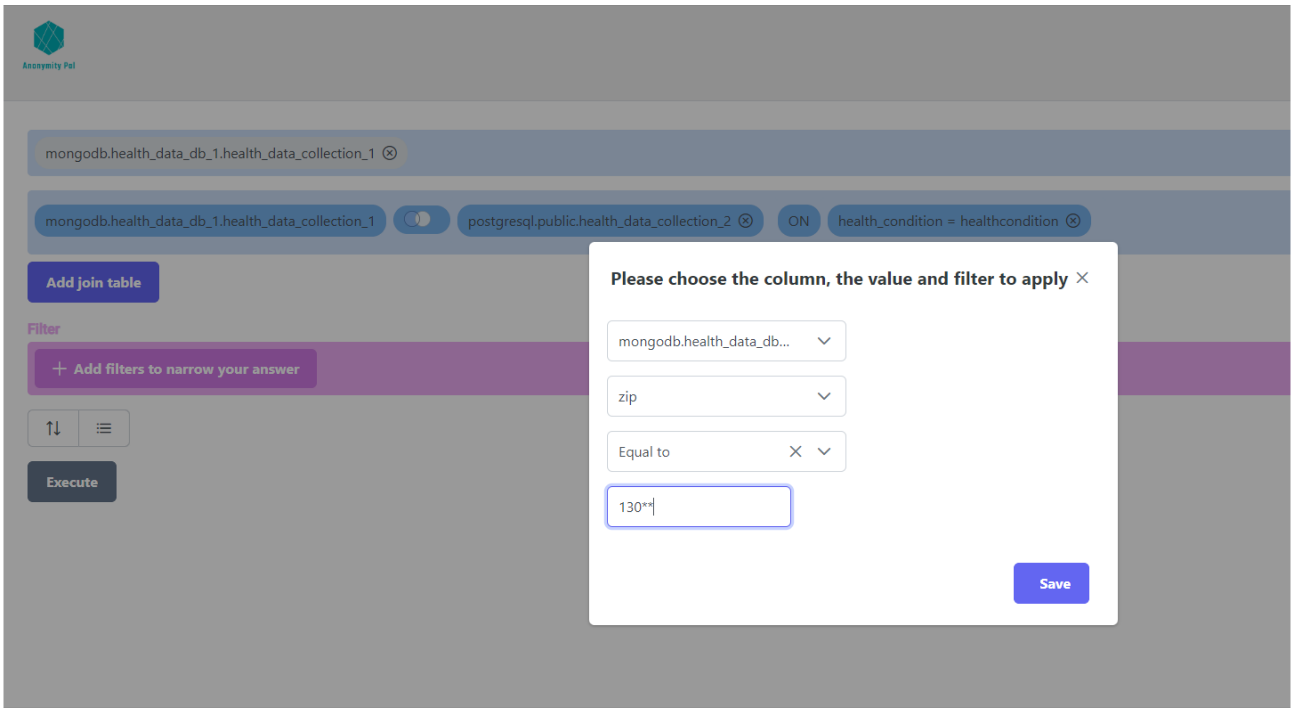
Once we select our first table, AnonymityPal will take us to the query builder. There we can see the selected table path **mongodb > health_data_db_1 > health_data_collection_1**.

Figure 17: Join sensitive columns


Anonymity Pal
Query Editor
mongodb.health_data_db_1.health_data_collection_1
mongodb.health_data_db_1.health_data_collection_1
Select a table
Select schema
information_schema
pg_catalog
public
Add join table
Filter
Add filters to narrow your answer
Execute
Reset

Next we need to perform the respective join to the second sample database (e.g postgres) on the health condition column, which is common for both tables. Then we need to filter our results with ZipCode equal to 130**.

Figure 18: Filtering the column *Zip* with 130 value**



mongodb.health_data_db_1.health_data_collection_1

mongodb.health_data_db_1.health_data_collection_1 | postgresql.public.health_data_collection_2 | ON | health_condition = healthcondition

Please choose the column, the value and filter to apply

mongodb.health_data_db...

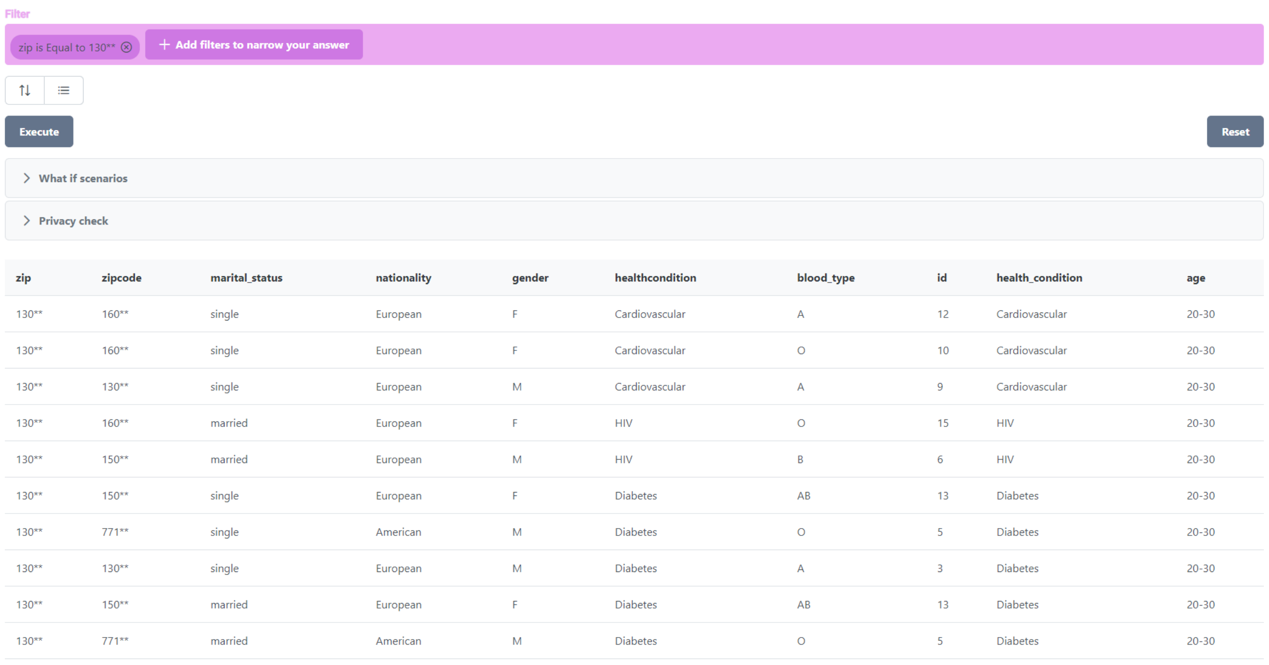
zip

Equal to

130**

In **Fig. 19** we can see the returned results after executing the SQL query that we formed with the help of the wizard query.

Figure 19: Filtering the column *Zip* with 130 value**



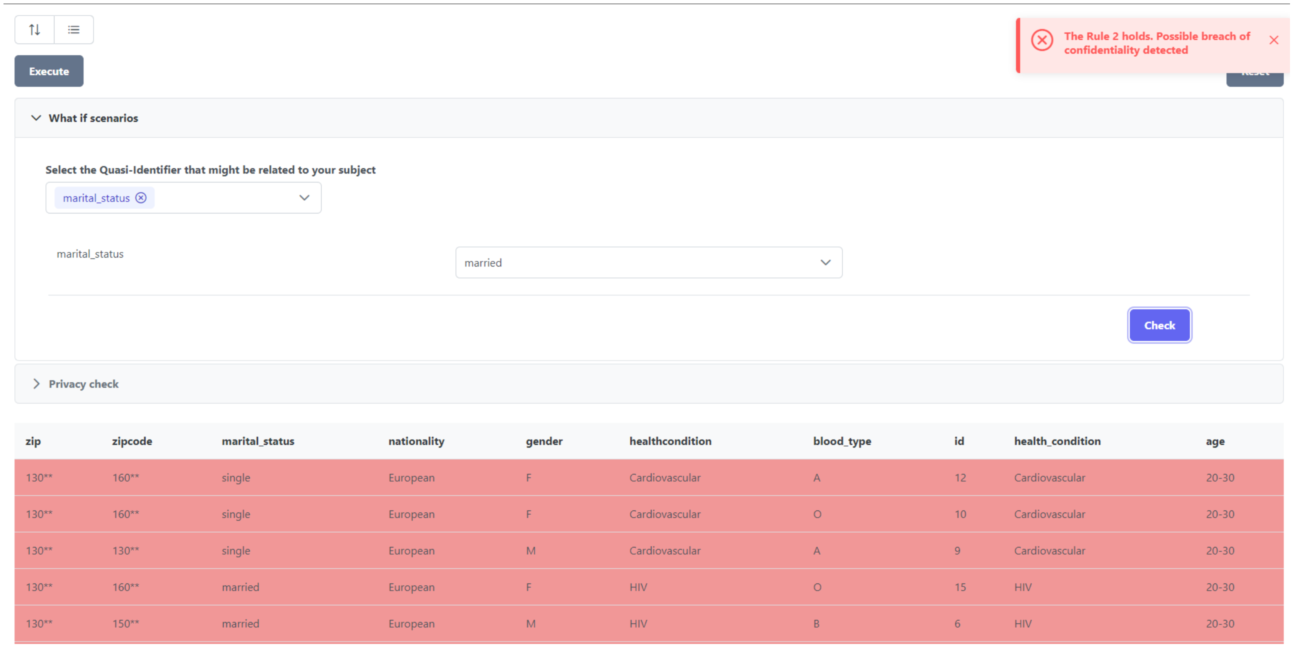
zip is Equal to 130**

zip	zipcode	marital_status	nationality	gender	healthcondition	blood_type	id	health_condition	age
130**	160**	single	European	F	Cardiovascular	A	12	Cardiovascular	20-30
130**	160**	single	European	F	Cardiovascular	O	10	Cardiovascular	20-30
130**	130**	single	European	M	Cardiovascular	A	9	Cardiovascular	20-30
130**	160**	married	European	F	HIV	O	15	HIV	20-30
130**	150**	married	European	M	HIV	B	6	HIV	20-30
130**	150**	single	European	F	Diabetes	AB	13	Diabetes	20-30
130**	771**	single	American	M	Diabetes	O	5	Diabetes	20-30
130**	130**	single	European	M	Diabetes	A	3	Diabetes	20-30
130**	150**	married	European	F	Diabetes	AB	13	Diabetes	20-30
130**	771**	married	American	M	Diabetes	O	5	Diabetes	20-30

5.2.1 What If Scenarios

Next we will use the *What If Scenarios* functionality in order to investigate if by giving a column a value that exists in the results, if that can lead to a possible breach of confidentiality. First we open the What If Scenarios tab, and we choose one of the available quasi columns of the result table, here the marital_status. Next we choose one of the available values that are present in the column, e.g. married.

Figure 20: What If Scenario to *Marital Status* column with value married


Execute
The Rule 2 holds. Possible breach of confidentiality detected
What if scenarios
Select the Quasi-Identifier that might be related to your subject
marital_status
Check
Privacy check

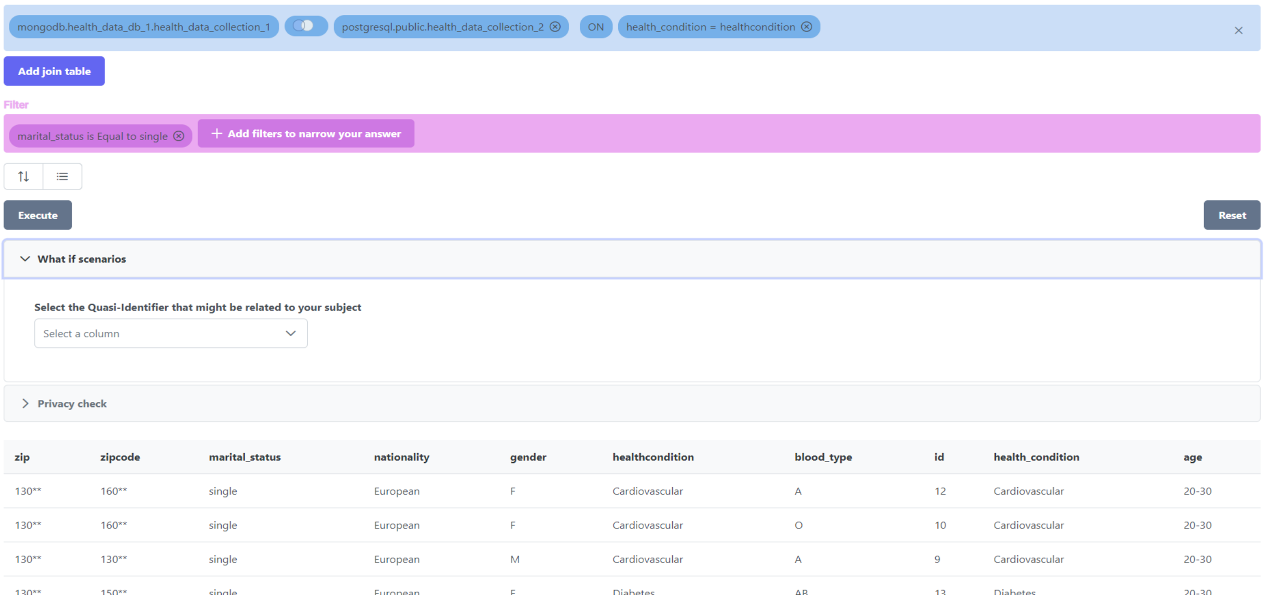
marital_status: married

zip	zipcode	marital_status	nationality	gender	healthcondition	blood_type	id	health_condition	age
130**	160**	single	European	F	Cardiovascular	A	12	Cardiovascular	20-30
130**	160**	single	European	F	Cardiovascular	O	10	Cardiovascular	20-30
130**	130**	single	European	M	Cardiovascular	A	9	Cardiovascular	20-30
130**	160**	married	European	F	HIV	O	15	HIV	20-30
130**	150**	married	European	M	HIV	B	6	HIV	20-30

In **Fig. 20** we can see that the privacy check service signals with red our retrieved results, meaning that if the adversary knew that the person he is looking for is married, he would know with high probability the health condition. Also an error message indicating that Rule 2 holds is informing the user.

Consider now a second example, with the test data of Case 1. This time we will join both tables in the sensitive attribute but we will filter the column *Marital Status* with the value ‘single’. That query will bring the results as shown in **Fig. 21**:

Figure 21: Query results



mongodb.health_data_db_1.health_data_collection_1
postgresql.public.health_data_collection_2
ON
health_condition = healthcondition
Add join table
Filter
marital_status is Equal to single
+ Add filters to narrow your answer
Execute
Reset
What if scenarios

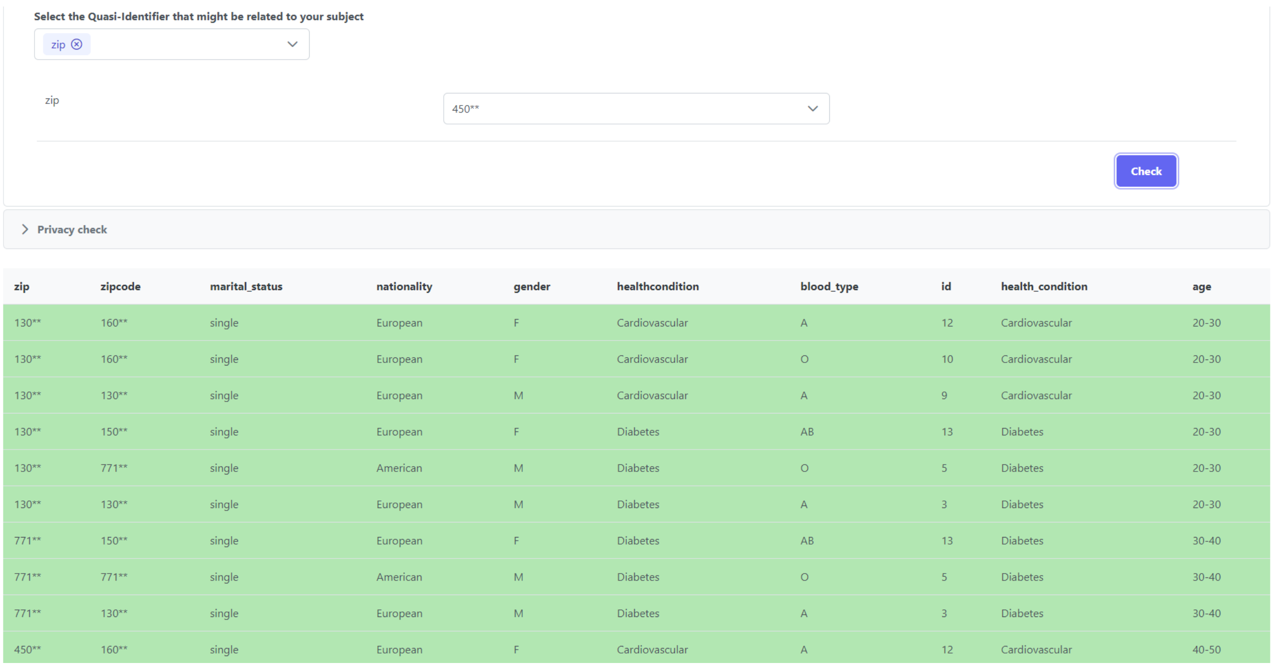
Select the Quasi-Identifier that might be related to your subject

Select a column
Privacy check

zip	zipcode	marital_status	nationality	gender	healthcondition	blood_type	id	health_condition	age
130**	160**	single	European	F	Cardiovascular	A	12	Cardiovascular	20-30
130**	160**	single	European	F	Cardiovascular	O	10	Cardiovascular	20-30
130**	130**	single	European	M	Cardiovascular	A	9	Cardiovascular	20-30
130**	150**	single	European	F	Diabetes	AB	13	Diabetes	20-30

Next we are going to investigate if some extra background information could lead us to a breach of confidentiality. The first thing we will try, is to check if we can identify the person we are looking for, if for example, we knew that he lives on zip code 450**. As we can see from **Fig. 22** the result is green, meaning that this information will not help identify the individual.

Figure 22: What If Scenario indicating no issues - *Zip* column filtered with 450**



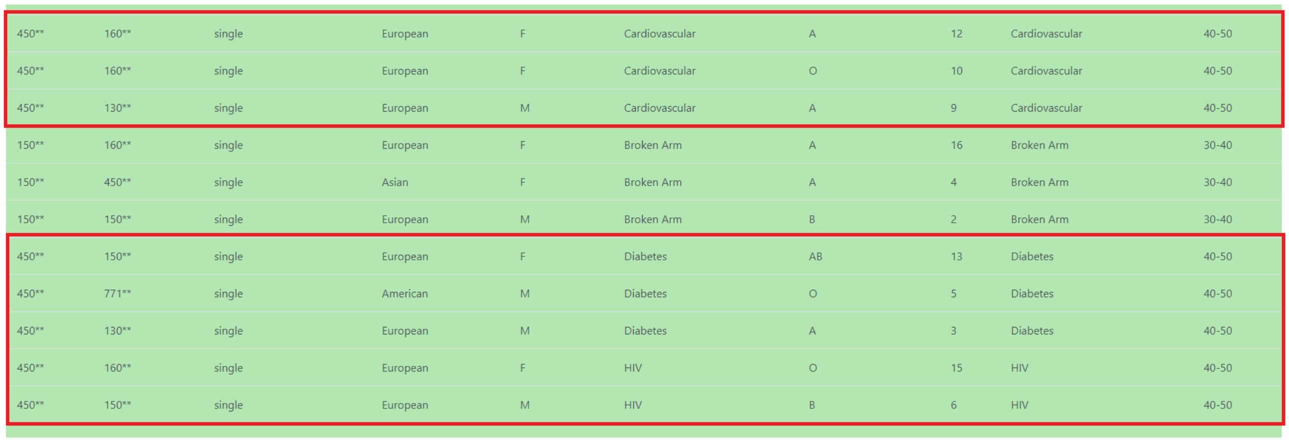
Select the Quasi-Identifier that might be related to your subject

zip: 450**

zip	zipcode	marital_status	nationality	gender	healthcondition	blood_type	id	health_condition	age
130**	160**	single	European	F	Cardiovascular	A	12	Cardiovascular	20-30
130**	160**	single	European	F	Cardiovascular	O	10	Cardiovascular	20-30
130**	130**	single	European	M	Cardiovascular	A	9	Cardiovascular	20-30
130**	150**	single	European	F	Diabetes	AB	13	Diabetes	20-30
130**	771**	single	American	M	Diabetes	O	5	Diabetes	20-30
130**	130**	single	European	M	Diabetes	A	3	Diabetes	20-30
771**	150**	single	European	F	Diabetes	AB	13	Diabetes	30-40
771**	771**	single	American	M	Diabetes	O	5	Diabetes	30-40
771**	130**	single	European	M	Diabetes	A	3	Diabetes	30-40
450**	160**	single	European	F	Cardiovascular	A	12	Cardiovascular	40-50

That can be verified also by checking the table results in the red placeholder in **Fig. 23**. The results in red are 8 and so N = 8 and V = 3 since there are 3 distinct values Cardiovascular, Diabetes and HIV.

Figure 23: Results with *Zip* column 450**



450**	160**	single	European	F	Cardiovascular	A	12	Cardiovascular	40-50
450**	160**	single	European	F	Cardiovascular	O	10	Cardiovascular	40-50
450**	130**	single	European	M	Cardiovascular	A	9	Cardiovascular	40-50
150**	160**	single	European	F	Broken Arm	A	16	Broken Arm	30-40
150**	450**	single	Asian	F	Broken Arm	A	4	Broken Arm	30-40
150**	150**	single	European	M	Broken Arm	B	2	Broken Arm	30-40
450**	150**	single	European	F	Diabetes	AB	13	Diabetes	40-50
450**	771**	single	American	M	Diabetes	O	5	Diabetes	40-50
450**	130**	single	European	M	Diabetes	A	3	Diabetes	40-50
450**	160**	single	European	F	HIV	O	15	HIV	40-50
450**	150**	single	European	M	HIV	B	6	HIV	40-50

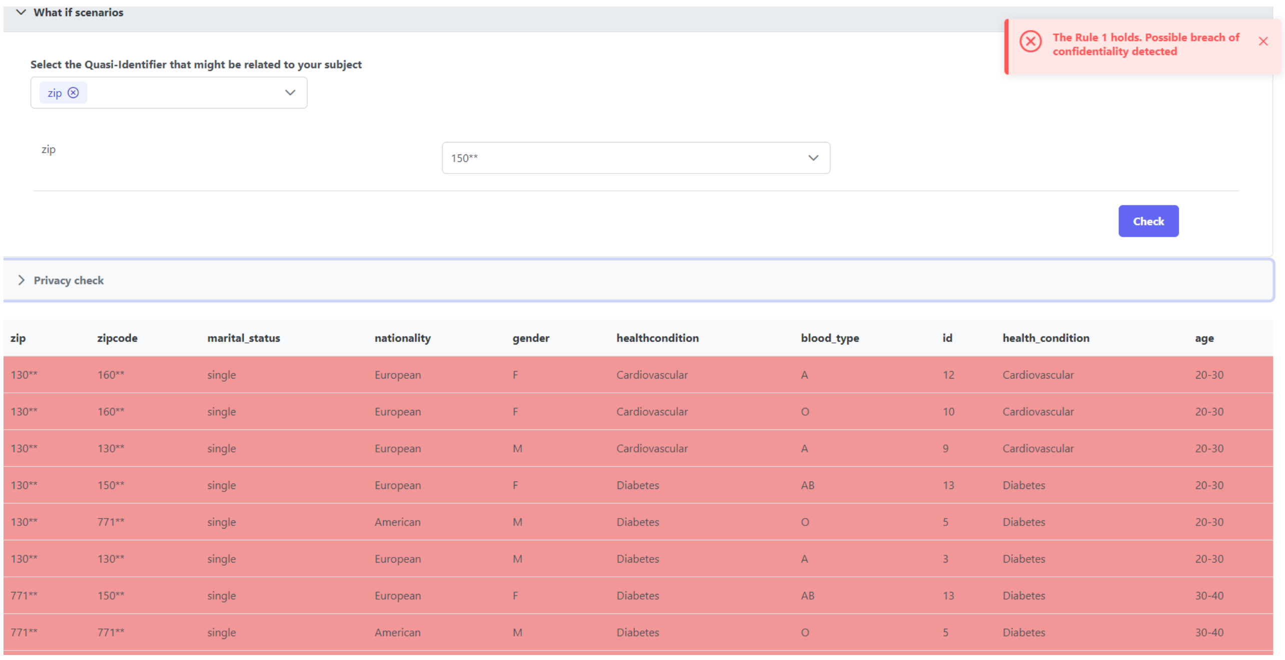
In **Table 24**. we have calculated based on the breach of confidentiality rules, probability of possible identification of an individual. Currently the threshold that we have set is 50% so none of the calculated probabilities is higher than that.

Table 24: Calculated P for Diabetes, HIV, Cardiovascular

	Diabetes	**HIV**	**Cardiovascular**
P	3/8 = 37,5%	2/8 = 25%	3/8 = 37,5%

On the other hand, if we check the zip code 150** we can see that the results got red and an error message of Rule 1 holds. This means that if we knew this information about the individual, a potential breach of confidentiality is possible.

Figure 24: What If Scenario indicating an issue - *Zip* column filtered with 150**



The Rule 1 holds. Possible breach of confidentiality detected

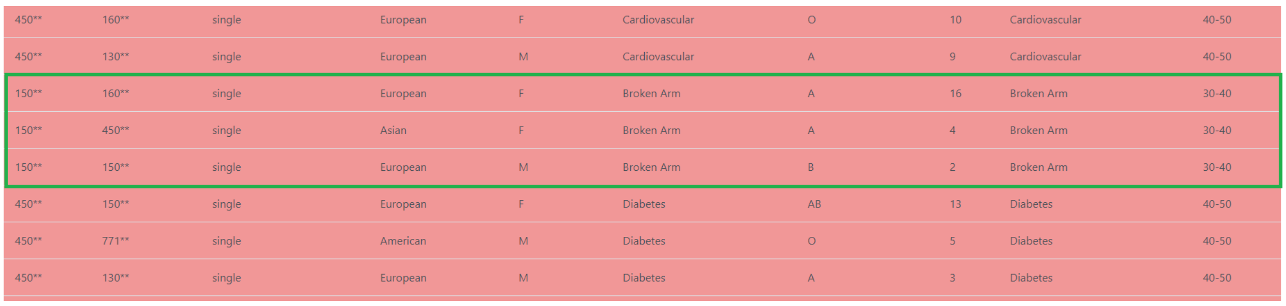
Select the Quasi-Identifier that might be related to your subject: zip

zip: 150**

zip	zipcode	marital_status	nationality	gender	healthcondition	blood_type	id	health_condition	age
130**	160**	single	European	F	Cardiovascular	A	12	Cardiovascular	20-30
130**	160**	single	European	F	Cardiovascular	O	10	Cardiovascular	20-30
130**	130**	single	European	M	Cardiovascular	A	9	Cardiovascular	20-30
130**	150**	single	European	F	Diabetes	AB	13	Diabetes	20-30
130**	771**	single	American	M	Diabetes	O	5	Diabetes	20-30
130**	130**	single	European	M	Diabetes	A	3	Diabetes	20-30
771**	150**	single	European	F	Diabetes	AB	13	Diabetes	30-40
771**	771**	single	American	M	Diabetes	O	5	Diabetes	30-40

This can be verified as well by looking at **Fig. 25**, where the record with the green placeholder is the only with zip code value 150**. Thus, one can be sure that this person has visited the hospital with a Broken Arm, since it is the only value in the sensitive column.

Figure 25: Results with zip code 150**

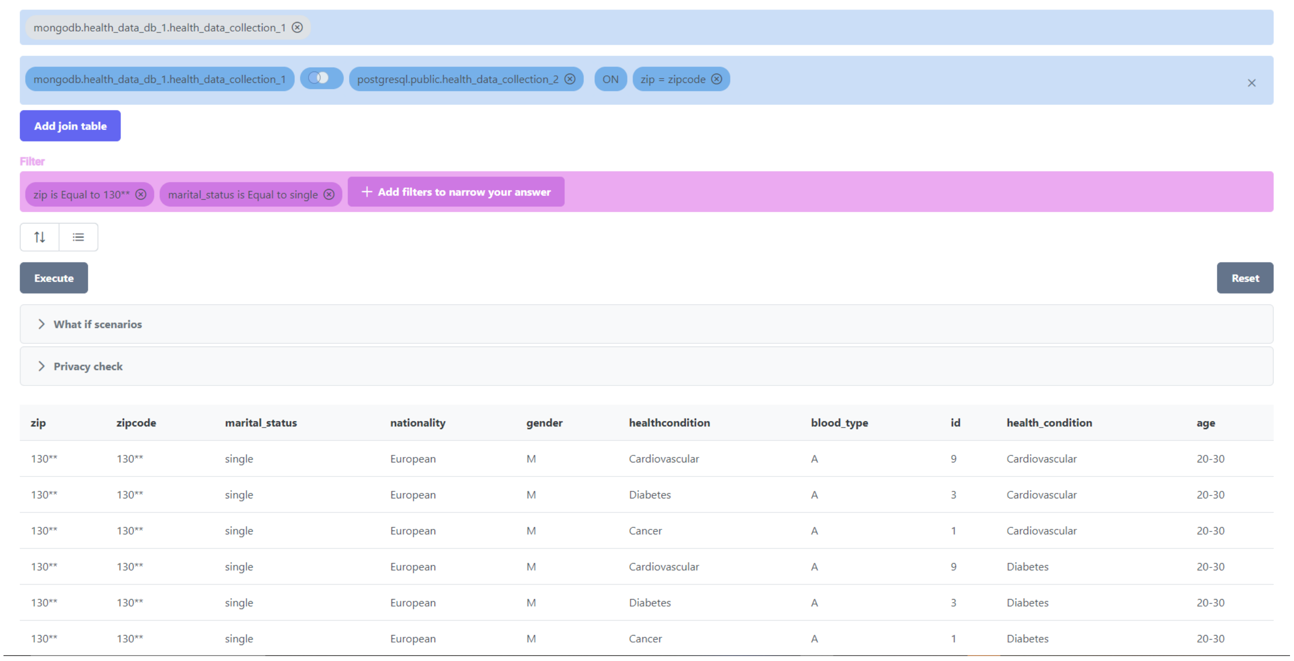


450**	160**	single	European	F	Cardiovascular	O	10	Cardiovascular	40-50
450**	130**	single	European	M	Cardiovascular	A	9	Cardiovascular	40-50
150**	160**	single	European	F	Broken Arm	A	16	Broken Arm	30-40
150**	450**	single	Asian	F	Broken Arm	A	4	Broken Arm	30-40
150**	150**	single	European	M	Broken Arm	B	2	Broken Arm	30-40
450**	150**	single	European	F	Diabetes	AB	13	Diabetes	40-50
450**	771**	single	American	M	Diabetes	O	5	Diabetes	40-50
450**	130**	single	European	M	Diabetes	A	3	Diabetes	40-50

5.2.2 Privacy Check

When retrieving some data from loosely coupled data sources, we are not sure if those results preserve the privacy of the individuals. For this reason AnonymityPal offers the Privacy check, which checks if the retrieved data preserve k-anonymity and l-diversity. The k, l are given by the user respectively. The k-anonymity and l-diversity algorithms were developed and integrated from us.

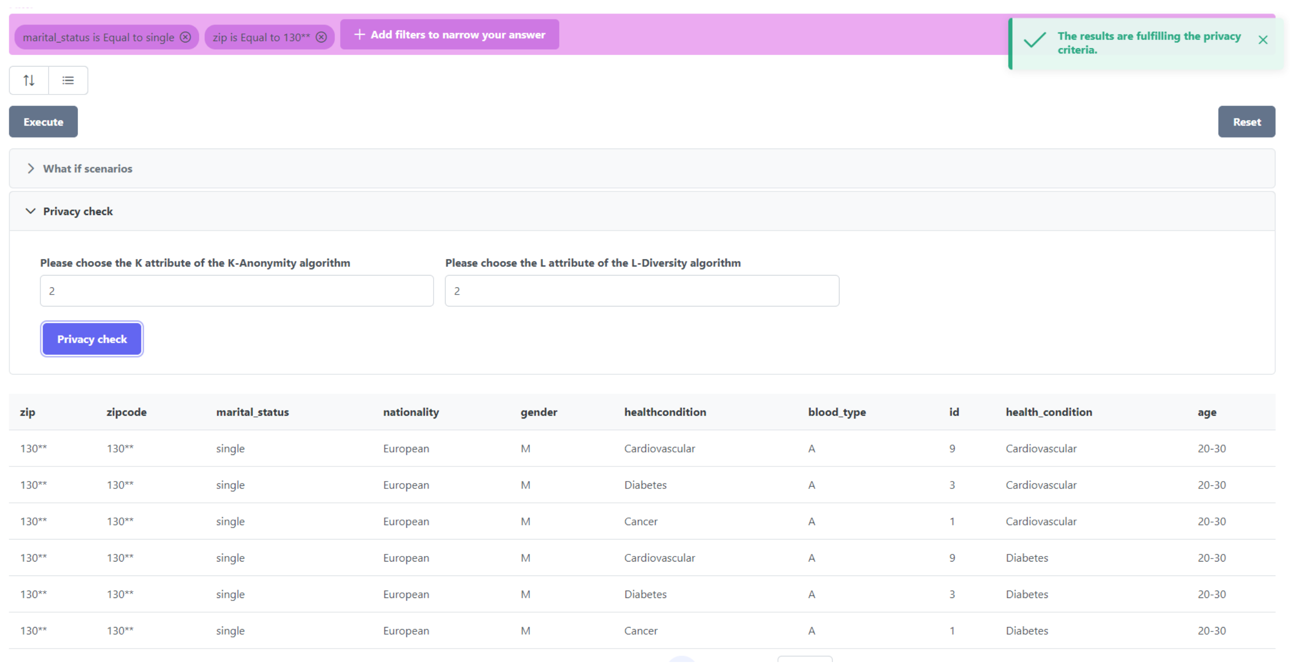
Figure 26: Query results to be checked from Privacy Check


mongodb.health_data_db_1.health_data_collection_1
mongodb.health_data_db_1.health_data_collection_1
postgresql.public.health_data_collection_2
ON
zip = zipcode
Add join table
Filter
zip is Equal to 130**
marital_status is Equal to single
+ Add filters to narrow your answer
Execute
Reset
What if scenarios
Privacy check

zip	zipcode	marital_status	nationality	gender	healthcondition	blood_type	id	health_condition	age
130**	130**	single	European	M	Cardiovascular	A	9	Cardiovascular	20-30
130**	130**	single	European	M	Diabetes	A	3	Cardiovascular	20-30
130**	130**	single	European	M	Cancer	A	1	Cardiovascular	20-30
130**	130**	single	European	M	Cardiovascular	A	9	Diabetes	20-30
130**	130**	single	European	M	Diabetes	A	3	Diabetes	20-30
130**	130**	single	European	M	Cancer	A	1	Diabetes	20-30

In **Fig. 27** we checked the privacy of our data with k,l = 2, and we got back a successful message of fulfilling the privacy criteria.

Figure 27: Privacy Check for k, l = 2 returns successful message



The results are fulfilling the privacy criteria.

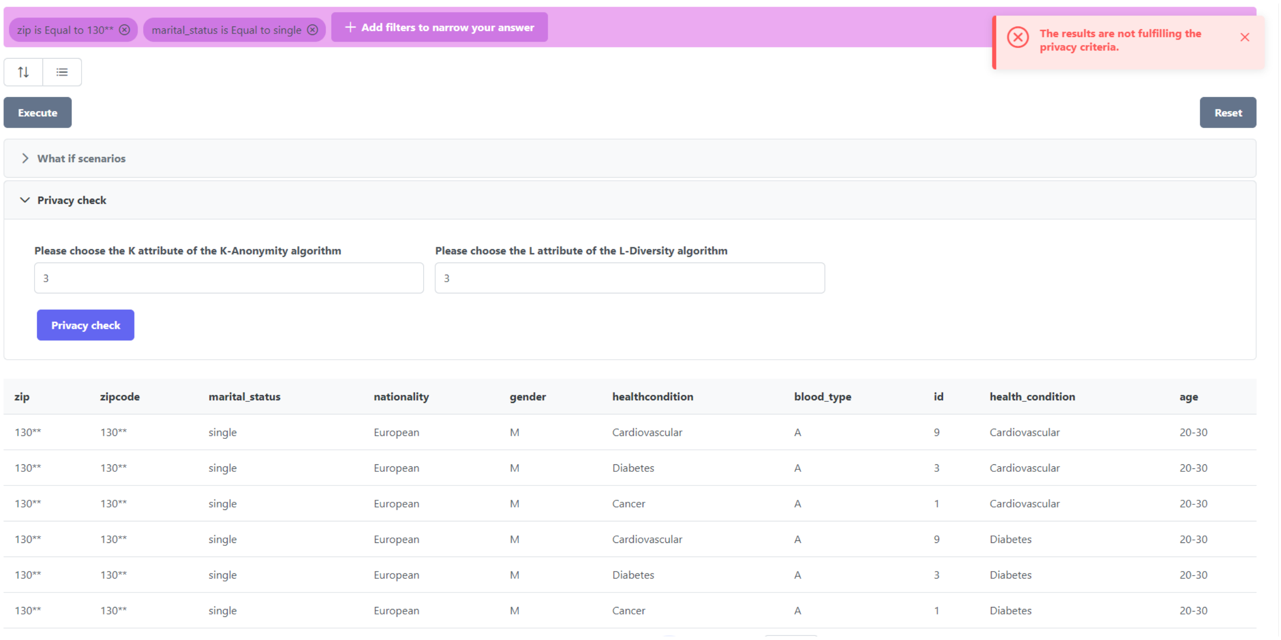
Please choose the K attribute of the K-Anonymity algorithm: 2

Please choose the L attribute of the L-Diversity algorithm: 2

zip	zipcode	marital_status	nationality	gender	healthcondition	blood_type	id	health_condition	age
130**	130**	single	European	M	Cardiovascular	A	9	Cardiovascular	20-30
130**	130**	single	European	M	Diabetes	A	3	Cardiovascular	20-30
130**	130**	single	European	M	Cancer	A	1	Cardiovascular	20-30
130**	130**	single	European	M	Cardiovascular	A	9	Diabetes	20-30
130**	130**	single	European	M	Diabetes	A	3	Diabetes	20-30
130**	130**	single	European	M	Cancer	A	1	Diabetes	20-30

In the same example if we try with k, l = 3 then an error message of not fulfilling the privacy criteria is returned as we can see in **Fig. 28**.

Figure 28: Data failed the privacy check for k, l = 2



zip is Equal to 130** ⓧ | marital_status is Equal to single ⓧ | + Add filters to narrow your answer

The results are not fulfilling the privacy criteria.

Execute | Reset

What if scenarios

Privacy check

Please choose the K attribute of the K-Anonymity algorithm: 3

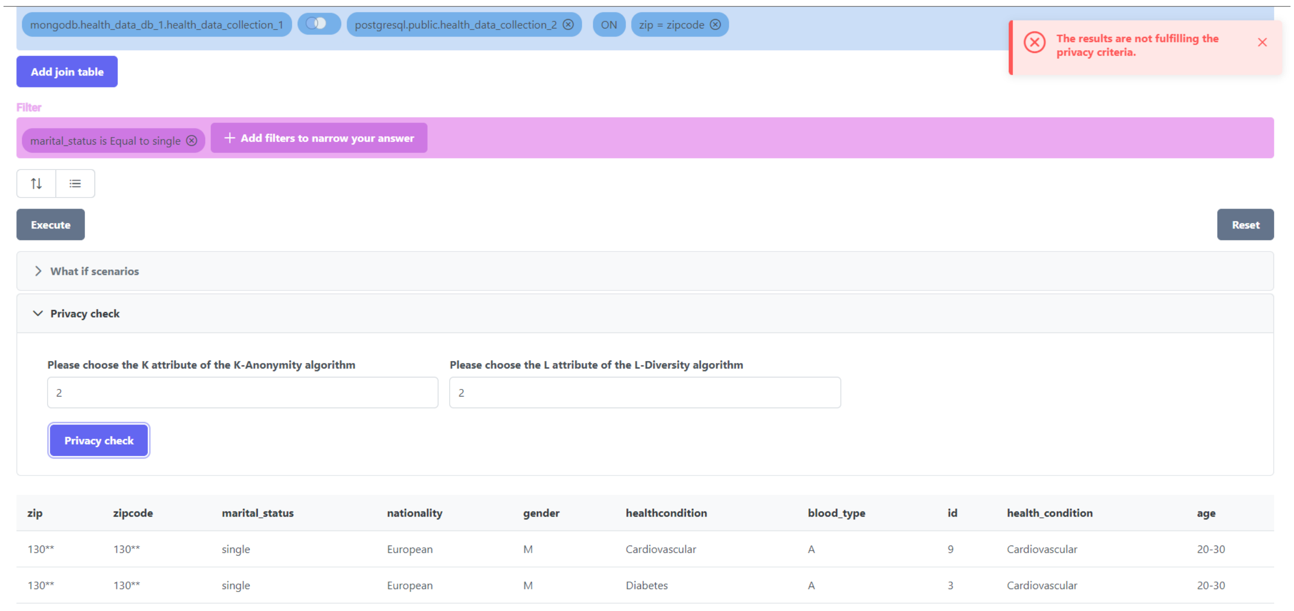
Please choose the L attribute of the L-Diversity algorithm: 3

Privacy check

zip	zipcode	marital_status	nationality	gender	healthcondition	blood_type	id	health_condition	age
130**	130**	single	European	M	Cardiovascular	A	9	Cardiovascular	20-30
130**	130**	single	European	M	Diabetes	A	3	Cardiovascular	20-30
130**	130**	single	European	M	Cancer	A	1	Cardiovascular	20-30
130**	130**	single	European	M	Cardiovascular	A	9	Diabetes	20-30
130**	130**	single	European	M	Diabetes	A	3	Diabetes	20-30
130**	130**	single	European	M	Cancer	A	1	Diabetes	20-30

In **Fig. 29**, a simpler example of privacy check is tried, which failed, and it is easier to verify the result, since only two records are present with different values.

Figure 29: Simpler example of data failure in the privacy check for k, l = 2



mongodb.health_data_db_1.health_data_collection_1 | postgresql.public.health_data_collection_2 ⓧ | ON | zip = zipcode ⓧ

The results are not fulfilling the privacy criteria.

Add join table

Filter

marital_status is Equal to single ⓧ | + Add filters to narrow your answer

Execute | Reset

What if scenarios

Privacy check

Please choose the K attribute of the K-Anonymity algorithm: 2

Please choose the L attribute of the L-Diversity algorithm: 2

Privacy check

zip	zipcode	marital_status	nationality	gender	healthcondition	blood_type	id	health_condition	age
130**	130**	single	European	M	Cardiovascular	A	9	Cardiovascular	20-30
130**	130**	single	European	M	Diabetes	A	3	Cardiovascular	20-30

5.3 Experimental Evaluation

A major concern in distributed systems is the performance of the systems that are involved and are handling the data. In this section we provide some experimental evidence to demonstrate the usefulness of the model we presented on Section 3.4. We focus on the performance of AnonymityPal as we increase the data retrieved from the distributed data sources. We present the results based on the efficiency and scalability of AnonymityPal.

Our testbed consisted of one desktop machine, which was equipped with a 3.4GHz Intel Core i5 CPU PC with 20GB of memory.

To evaluate the scalability of our system, we performed some discrete simulations on a realistic topology of healthcare data providers and patients who have records at multiple providers. Due to the sensitive nature of real world data, several attempts to obtain actual healthcare data were unsuccessful; instead we generated our own, consisting of three hospitals. Every data provider maintains information about its patients in some internal database with a proprietary schema. Each hospital provides one read-only view with patient data. Each view consists of tuples of the form (id, attr1, attr2, …) where id is a locally generated identifier for each row. Each row belongs to a specific patient.

The experimental environments are as follows: we employ three generated real-life datasets, two in PostgreSQL and in one MongoDB. The first dataset in PostgreSQL has 4 quasi attributes; 3 of them are quasi identifier attributes. The fourth is the sensitive value. The same applies for the dataset in MongoDB. These two datasets consist of 20.000 patient records. The second dataset in PostgreSQL has 5 quasi attributes and 1 sensitive. It consists of 20.000 records. Each record represents one patient incident of hospital visit and each patient has visited both hospitals only once. The columns of the first dataset are *zip, age, marital_status*, and *health_condition*. *Zip* column has 5 possible values, age has 4 possible values, which are age groups, like 20-30, 40-50 etc. We consider married and single values in the attribute *marital_status.* The *health_condition* can have 13 possible values representing health diagnosis. The columns of the dataset in MongoDB are *nationality, gender, blood_type, zipCode,* and *disease*. The *nationality* column can take 5 possible values, the *gender* can be either male or female, the *blood_type* 4 values, the *zipCode* 5 possible values and the *disease* can take up to 13 possible values. The columns in the second PostgreSQL dataset are *country_of_live, wordclass, education, race,* and *health_issue*. The *health_issue* column can take 13 possible values representing health diagnosis, that are similar to the health column of the other two datasets. The *country_of_live* can take 24 different values, representing the current country the patient lives. The *wordclass* column as well as the education can take up to 8 values, and the race can have 5 values like white, black, asian etc.

In our experiments, we tried to (1) left join one of the PostgreSQL in the disease column with MongoDB on the health_condition column and add multiple filters, as one can see in **Fig. 30**, (2) run the *Privacy Check* service in the retrieved data and (3) run the *What If Scenarios* service on the retrieved data. For all three steps we took measurements of how fast the tasks were completed.

Figure 30: Custom query with join and filter

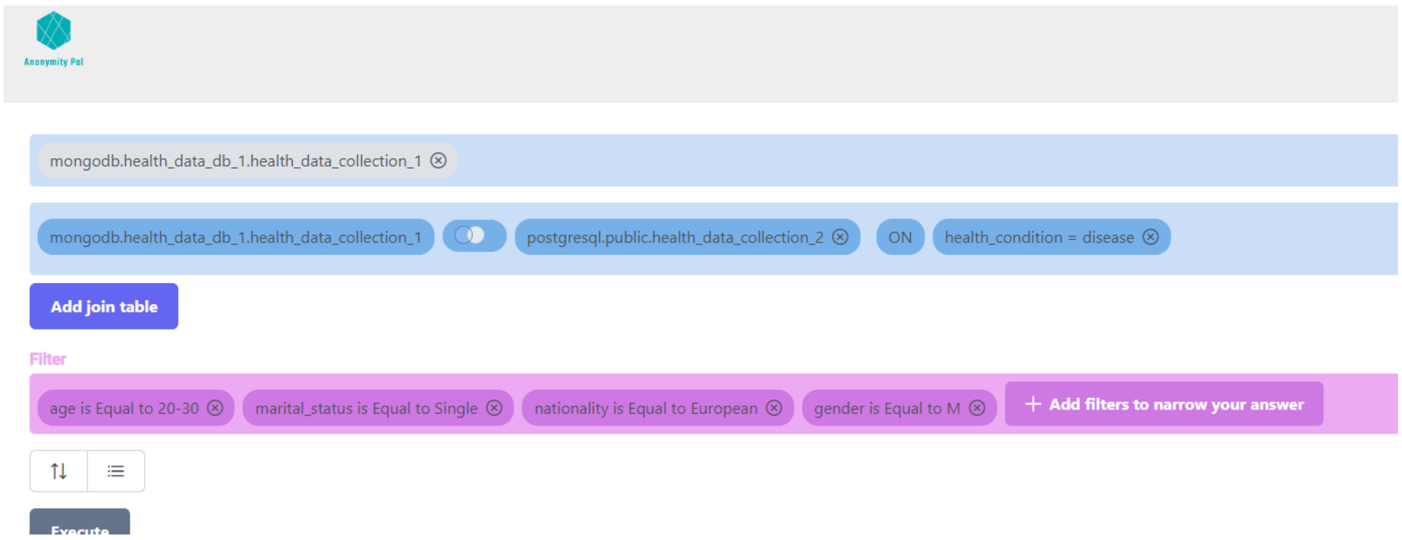

Anonymity Pal
mongodb.health_data_db_1.health_data_collection_1
mongodb.health_data_db_1.health_data_collection_1
postgresql.public.health_data_collection_2
ON
health_condition = disease
Add join table
Filter
age is Equal to 20-30
marital_status is Equal to Single
nationality is Equal to European
gender is Equal to M
+ Add filters to narrow your answer
Execute

Fig. 31 shows the summarized execution times for different data loads. We can see that AnonymityPal takes more time to retrieve data as the number of results is increasing. Here, execution times are much more strongly influenced when we join data from different data sources. More than half a million data needed around 22 seconds to be returned to the user, which is acceptable, since in general, fetching a large amount of data takes a significant amount of time. That’s because a lot of work needs to be done to move large volumes of data from the database server to the user interface. Specifically, the data has to be scanned from the disks of the data providers and loaded into the buffer pool of Presto. Then the loaded data is sent over the network, the application server will get the data in tabular fashion (e.g., the JDBC ResultSet) and the application transforms the tabular-based data into tree-based structures. Finally, the tree-based structure is transformed to JSON and sent over the network to the browser and the browser needs to load the entire JSON and use it to build the UI.

Figure 31: Execution times for different data loads

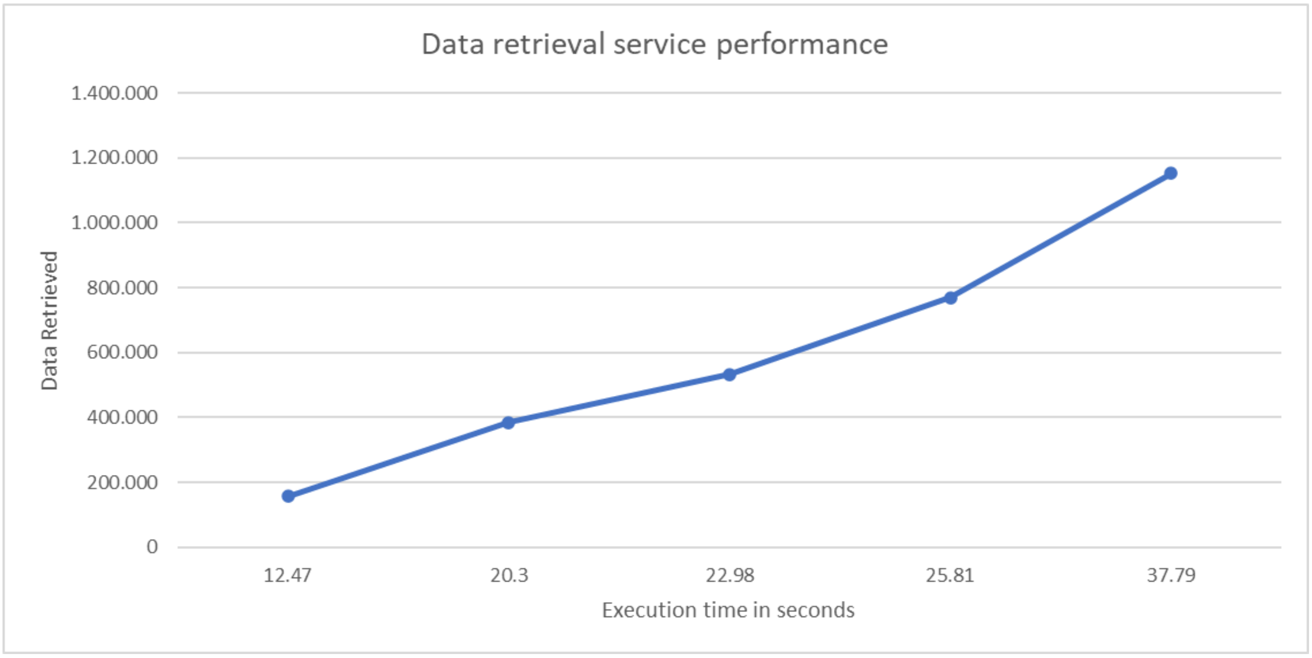


Fig. 32 shows the summarized execution times for different k,l parameters on the same data. Note that, as the k,l parameters are getting bigger, the execution time is getting smaller.

Figure 32: Execution times for different k, l parameters

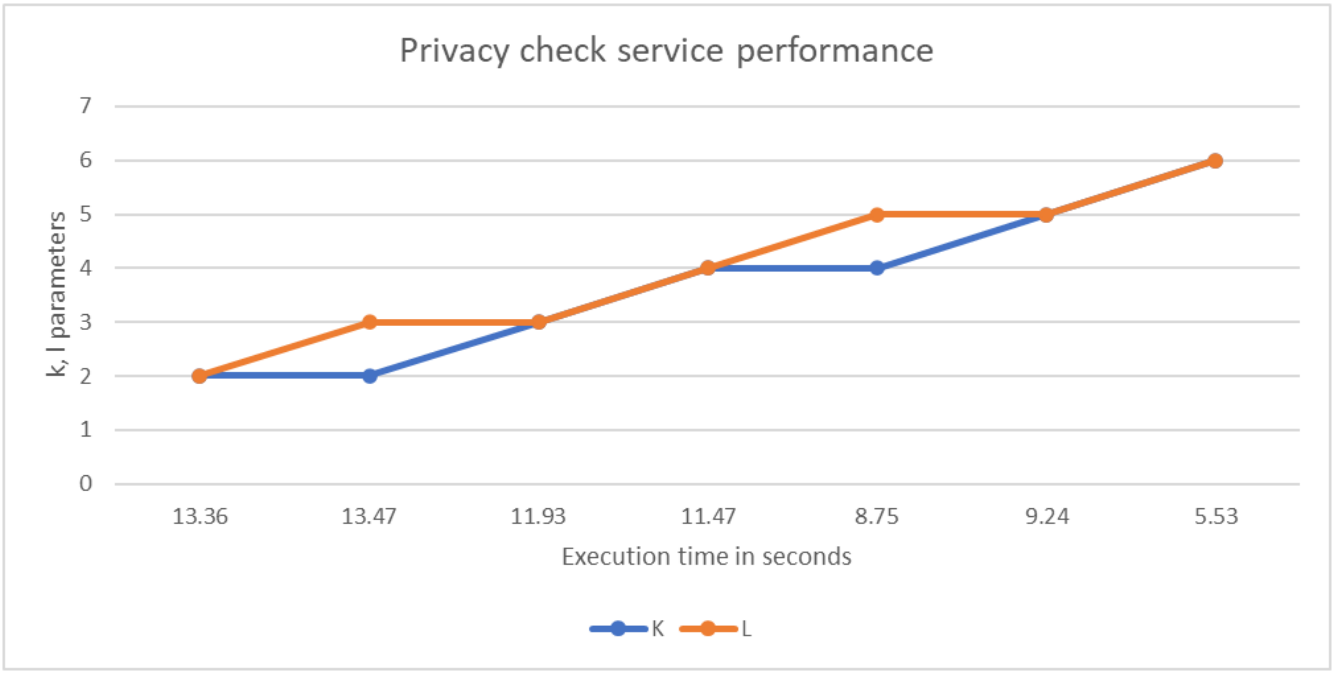
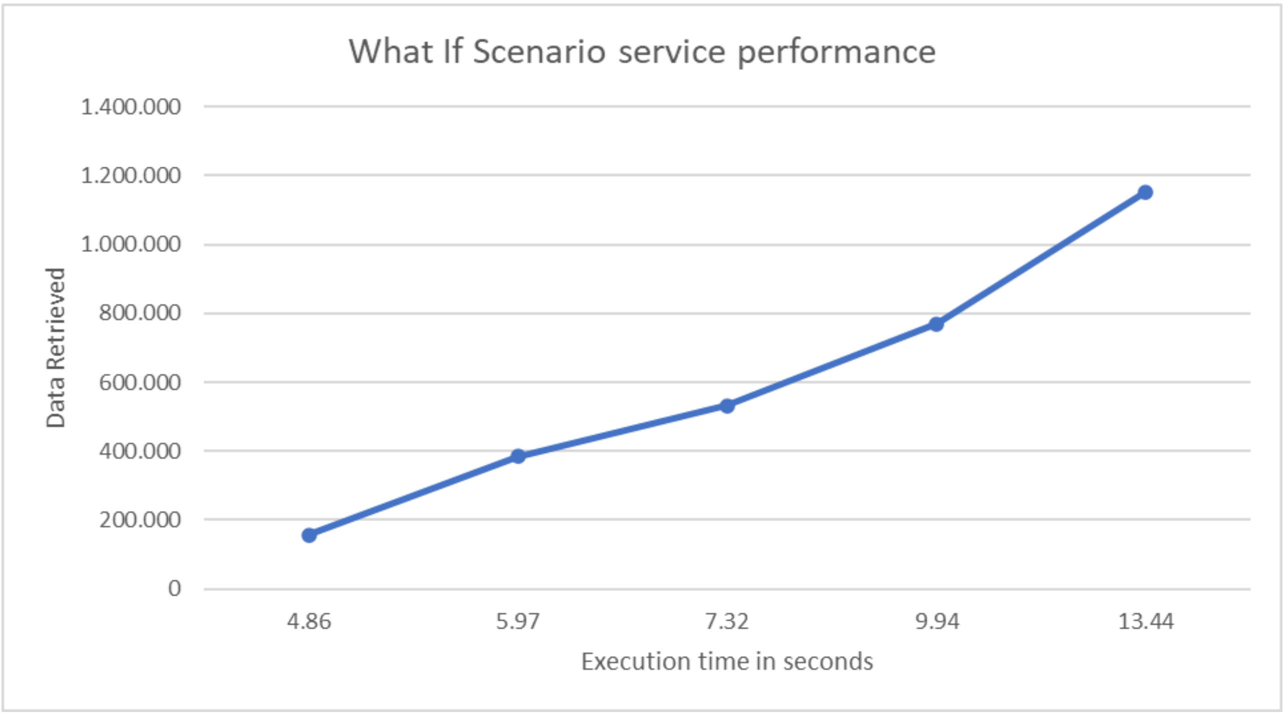


Fig. 33 shows the summarized execution times for different data loads for the *What if scenarios* service. We can see that AnonymityPal takes more time to analyze the data for possible composition attacks, as the number of data is increasing. Around 1.2 million data can be analyzed approximately in 13 seconds which makes our tool powerful.

Figure 33: What if scenario service execution times for different data loads



6 CONCLUSIONS AND FURTHER WORK

Understanding the complexity of keeping the privacy of the individual secure in healthcare data is an important aspect of data publishing. The organizations that have this kind of data should follow all the regulations that are defined in each country. Many methodologies to privacy preserve the data exist, but the road of anonymization is full of pitfalls. Composition attacks are widely used against anonymized data sources by joining these sources together and trying to exploit them. However, to develop a method that can detect early composition attacks when joining independent data sources is necessary to better preserve the privacy of the individual.

In this thesis we proposed a high-level framework, which depends on a set of rules for privacy checking. A precise formulation of the properties used in the rules, to detect composition attacks in the presence of independent anonymized data sources. Our experimental study indicates that several currently proposed anonymization methodologies, including k-anonymity and its variants, are vulnerable to composition attacks. The analysis was done on generated anonymized health care data, as close to real life scenarios. Of course, further investigation through experimental procedures is required to evaluate the integrity of these results.

We also developed an open source tool, AnonymityPal, for analyzing and comparing privacy-preserving data in sharing infrastructures for medical research. We believe that our framework makes it easier to track possible breach of confidentiality from composition attacks. It also can help us check if the data we have are k-anonymized and l-diversified. This can support the decision makers and regulatory authorities in gaining a better understanding of their data and what are the extra steps needed to protect their data. We have shown that AnonymityPal is of value, by using it to analyze data acquired by joining and filtering 2 or more independent datasources. Finally, our results also provide insights into gaps, regarding the anonymization itself as well as the current landscape of data sharing infrastructures, that may be worth exploring in the future.

The AnonymityPal utilized distributed database query engines like PrestoDB, to develop AnonymityPal and evaluate possible breach of confidentiality in the data, providing the asset of handling large amounts of relational data efficiently, accurately and with high speed. PrestoDB, which was containerized into a docker container, is quite ideal for this purpose since it provides a wide variety of query optimization algorithms that can be of great use during a complex query in a distributed network. Moreover, it can be easily accessed by many programming languages through its REST API, while it gives a more dynamic approach to the process of large-scale data analysis. Presto is open source and can run interactive analytic queries against data sources of all sizes ranging from gigabytes to petabytes. The distributed data sources which were connected to PrestoDB were three, one MongoDB and two Postgresql and were containerized into docker containers as well. This helps in the portability of the application. A Spring Boot application was used to connect to PrestoDB and also to provide our main services (1) What if scenarios and (2) Privacy check. Also it offers a query builder functionality as well as the functionality of running native queries against PrestoDB. The UI was developed with the latest Angular framework.

In this study we demonstrated some scenarios that were analyzed in detail by utilizing the UI of AnonymityPal and we proved how easy a user can detect possible issues. We also used the Privacy check functionality, which integrates the k-anonymity and l-diversity algorithms, to check if the privacy of the retrieved data is preserved.

Amongst the most important findings of this study were some best practices that were derived from the composition attacking scenarios. More specifically, we focused on (1) how the diversity of the quasi identifiers define the the power of the adversary, (2) the effect of the number common quasi identifiers between loosely coupled data sources, (3) the different anonymization approaches that one can follow when dealing with specific quasi identifiers categories and (4) how to choose the correct threshold.

Further Work

The most striking question that arises from this work is in the field of What if scenarios, a main functionality of our framework. For example, if we had some new information about a column, in general, without needing to provide a specific value to try, how much that could affect the privacy of the individual. Or is there a smart way to find information about a value if I find information about another value. For example, if the adversary finds information about all people at age 30, could this lead to finding extra information about people at age 32 ? Another interesting direction would be to study other attacking scenarios beside composition attacks, and integrate a set of early detection rules into AnonymityPal. A natural candidate for future investigation is in which depth other anonymity algorithms protect the data of the individuals in a distributed environment. This investigation needs the data to be anonymized with more complex algorithms and perform composition attacking scenarios in order to calculate the probability of breach of confidentiality.

BIBLIOGRAPHY

[1] Machanavajjhala, A., Gehrke, J., Kifer, D. and Venkitasubramaniam, M., 2006. L-diversity: privacy beyond k-anonymity. 22nd International Conference on Data Engineering (ICDE'06),.

[2] Samarati, P., 2001. Protecting respondents identities in microdata release. IEEE Transactions on Knowledge and Data Engineering, 13(6), pp.1010-1027.

[3] SWEENEY, L., 2002. k-ANONYMITY: A MODEL FOR PROTECTING PRIVACY. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 10(05), pp.557-570.

[4] Barhamgi, M., Benslimane, D., Ghedira, C. and Gancarski, A., 2011. Privacy-Preserving Data Mashup. 2011 IEEE International Conference on Advanced Information Networking and Applications,.

[5] Emam, K., 2008. Heuristics for De-identifying Health Data. IEEE Security &amp; Privacy Magazine, 6(4), pp.58-61.

[6] Fung, B., Wang, K., Chen, R. and Yu, P., 2010. Privacy-preserving data publishing. ACM Computing Surveys, 42(4), pp.1-53.

[7] Fung, B., Wang, K. and Yu, P., 2007. Anonymizing Classification Data for Privacy Preservation. *IEEE Transactions on Knowledge and Data Engineering*, 19(5), pp.711-725.

[8] Gaye, A., Marcon, Y., Isaeva, J., LaFlamme, P., Turner, A., Jones, E., Minion, J., Boyd, A., Newby, C., Nuotio, M., Wilson, R., Butters, O., Murtagh, B., Demir, I., Doiron, D., Giepmans, L., Wallace, S., Budin-Ljøsne, I., Oliver Schmidt, C., Boffetta, P., Boniol, M., Bota, M., Carter, K., deKlerk, N., Dibben, C., Francis, R., Hiekkalinna, T., Hveem, K., Kvaløy, K., Millar, S., Perry, I., Peters, A., Phillips, C., Popham, F., Raab, G., Reischl, E., Sheehan, N., Waldenberger, M., Perola, M., van den Heuvel, E., Macleod, J., Knoppers, B., Stolk, R., Fortier, I., Harris, J., Woffenbuttel, B., Murtagh, M., Ferretti, V. and Burton, P., 2014. DataSHIELD: taking the analysis to the data, not the data to the analysis. International Journal of Epidemiology, 43(6), pp.1929-1944.

[9] Howe, H., Lake, A. and Shen, T., 2006. Method to Assess Identifiability in Electronic Data Files. American Journal of Epidemiology, 165(5), pp.597-601.

[10] Jiang, W. and Clifton, C., 2006. A secure distributed framework for achieving k-anonymity. The VLDB Journal, 15(4), pp.316-333.

[11] Jurczyk, P. and Xiong, L., 2009. Distributed Anonymization: Achieving Privacy for Both Data Subjects and Data Providers. *Data and Applications Security XXIII*, pp.191-207.

[12] Loukides, G., Gkoulalas-Divanis, A. and Malin, B., 2010. Anonymization of electronic medical records for validating genome-wide association studies. Proceedings of the National Academy of Sciences, 107(17), pp.7898-7903.

[13] Porat, S., Carmeli, B., Domany, T., Drory, T., Kveler, K., Melament, A. and Nelken, H., 2009. Masking Gateway for Enterprises. Languages: From Formal to Natural, pp.177-191.

[14] 1993. Private Lives and Public Policies.

[15] Tamersoy, A., Loukides, G., Nergiz, M., Saygin, Y. and Malin, B., 2012. Anonymization of Longitudinal Electronic Medical Records. IEEE Transactions on Information Technology in Biomedicine, 16(3), pp.413-423.

[16] Vardalachakis, M., Kondylakis, H., Koumakis, L., Kouroubali, A. and Katehakis, D., 2019. ShinyAnonymizer: A Tool for Anonymizing Health Data. *Proceedings of the 5th International Conference on Information and Communication Technologies for Ageing Well and e-Health*,.

[17] Wang, K., Fung, B. and Yu, P., 2006. Handicapping attacker's confidence: an alternative to k-anonymization. *Knowledge and Information Systems*, 11(3), pp.345-368.

[18] Wilkinson, K., Green, C., Nowicki, D. and Von Schindler, C., 2020. Less than five is less than ideal: replacing the “less than 5 cell size” rule with a risk-based data disclosure protocol in a public health setting. Canadian Journal of Public Health, 111(5), pp.761-765.

[19] Xue, M., Papadimitriou, P., Raïssi, C., Kalnis, P. and Pung, H., 2011. Distributed Privacy Preserving Data Collection. *Database Systems for Advanced Applications*, pp.93-107.