Technische Universität Berlin

Chair of Database Systems and Information Management

Master's Thesis

Anonymized Access Control for Distributed Event Stores

Henri Tyl Allgöwer
Degree Program: Computer Science
Matriculation Number: 454925

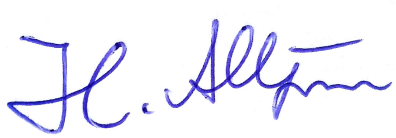
Reviewers
Prof. Dr. Volker Markl
Prof. Dr. Odej Kao

Advisor
Rudi Poepsel Lemaitre

Submission Date
30.11.2023

Hereby I declare that I wrote this thesis myself with the help of no more than the mentioned literature and auxiliary means.

Berlin, October 27, 2023



Henri Tyl Allgöwer

Zusammenfassung

Tipps zum Schreiben dieses Abschnitts finden Sie unter [16]

Abstract

The abstract should be 1-2 paragraphs. It should include:

- a statement about the problem that was addressed in the thesis,
- a specification of the solution approach taken,
- a summary of the key findings.

For additional recommendations see [16].

Acknowledgments

For recommendations on writing your Acknowledgments see [17]. Thank you to the chair at Database Systems and Information Management (DIMA)

Contents

List of Abbreviations . . . . . **x**

List of Algorithms . . . . . **xi**

1 Introduction **1**
1.1 Motivation 1
1.2 Research Challenge 1
1.3 Novelty 2
1.4 Anticipated Impact 2
1.5 Research Problem 2
1.6 Outline 3

2 Literature Review **4**
2.1 Role Based Access Control 4
2.2 Distributed Event Stores 4
2.3 Anonymization 4
2.3.1 Principles 4
2.3.2 Data Streaming 4
2.4 Related Work 4

3 Theoretical Framework **5**
3.1 Managing Different Anonymization Granularity 5
3.1.1 Use Case Example 6
3.2 Anonymization Techniques 9
3.2.1 Value Based Masking Functions 11
3.2.2 Tuple Based Masking Functions 15
3.2.3 Attribute Based Masking Functions 16
3.2.4 Table Based Anonymization Techniques 21
CASTLE 23
3.3 System Requirements 28
3.3.1 Data Stream Integration 28
3.3.2 Administration 28
3.3.3 Performance 28
3.3.4 Adaptability 28

3.3.5 Scalability . . . 28
3.3.6 Reliability . . . 28

4 Implementation . . . 29
4.1 Apache Kafka . . . 29
4.2 Apache ZooKeeper . . . 29
4.3 Anonymized Kafka . . . 29
4.3.1 Stream Manager . . . 29
4.3.2 Configuration parsing . . . 29
4.3.3 Validation . . . 29
4.3.4 Stream Config Builder . . . 29
4.3.5 Kafka Streams . . . 29
4.3.6 Anonymizers . . . 29
4.4 Test Suite . . . 29
4.4.1 Data Generator . . . 29
4.4.2 Kafka Connector . . . 29
4.4.3 Consumers . . . 29
4.5 Docker . . . 29

5 Testing and Evaluation . . . 31
5.1 Experimental Setup . . . 31
5.1.1 Experimental Design . . . 31
5.2 Parameter Optimization . . . 31
5.2.1 Interpretation of the Results . . . 31

6 Conclusion . . . 32
6.1 Future Work . . . 32

Appendix A. Further Details on the Solution Approach . . . 35

Appendix B. Extended Version of the Experimental Results . . . 36

Chapters 3-5 are the core of the thesis, whereas Chapters 1, 2, 6, and 7 provide context. The major contributions should be in Chapters 4 and 5. This structure serves as a guideline and should be customized accordingly. In particular, the generic chapter titles should be replaced with more specific ones, where appropriate (e.g., Chapter 4).

List of Figures

1 Hierarchy of masking functions 11
2 Example of suppression of an attribute. 12
3 Example of blurring of an attribute. 12
4 Example of substitution of an attribute. 13
5 Example of tokenization of an attribute. 13
6 Example of the generalization of an attribute. 14
7 Example of bucketizing of an attribute. 14
8 Example of adding noise. 14
9 Conditional substitution with a direct match condition taking the entire tuple into consideration. 16
10 Substitution with a range condition on the basis of only a singular attribute in the tuple. 16
11 Regular expression as a condition for the substition as part of a tuple based masking function. 16
12 Aggregation operations on the "age" attribute. 18
13 Arbitrary customer data where the values of the attribute "name" as well as "residency" is shuffled throughout the table. 21
14 Generalization hierarchy for the attribute *residency*. 24

List of Tables

1 Correlation in the existence of outlier[10]. 2

2 Example table of diabetes patients 6

3 Data available for the nurse staff. Note that pid, zip, insurance number and additional medical information have been suppressed as indicated by their cell's light red background. 8

4 Data available for the administration. Note that only the insurance information, medication and diagnosis are not suppressed. 9

5 K-anonymized data available for external research. The sensitive medical attribtues remains unchanged as indicated by the white cell background. Unlike the personally identifying attributes, which have been suppressed as denoted by the red cell background. The yellow cell background highlights the generalized quasi identifiable attributes. Note that the entries of the first group have unchanged values for the attributes *sex* and *ins. co.*. As all three original entries shared the same value it did not need to be generalized. . . . 10

6 Working example for table based anonymization containing a table with four attributes marked above with their corresponding category. Additionally, there is a leading column, which is not part of the data, but serves as reference. 22

7 3-anonymized version of the working example. Suppressed fields are in red cells, generalized fields in yellow. 25

8 2-diverse version of the working example. Suppressed fields are in red cells, generalized fields in yellow. 26

9 T-closeness version of the working example. Suppressed fields are in red cells, generalized fields in yellow. 27

List of Abbreviations

DIMA Database Systems and Information Management

CCPA California Consumer Privacy Act

GDPR General Data Protection Regulation

ICD International Statistical Classification of Diseases and Related Health Problems

PII Personally Identifiable Information

SSE Sum of Squared Errors

List of Algorithms

1 Pseudocode for the univariate microaggregation approach by Hansen and Mukherjee [4] 19

2 Splitting a Session[15]. 30

1 Introduction

... should include the following:

- motivation (why is this problem interesting? offer examples),
- research challenge (what is the obstacle to be overcome?),
- novelty (was this problem already solved?),
- anticipated impact (how does solving this problem impact our world?).

1.1 Motivation

The increasing popularity of data streaming in corporations highlights the imperative need for incorporating anonymization and data masking techniques in this technology. Particularly noteworthy is the extensive adoption of distributed event stores, which are scalable and fault-tolerant systems designed to capture, store, and process real-time data streams, across various sectors, including Fortune 100 companies, governments, healthcare, and transportation industries [5]. This widespread usage emphasizes the criticality of ensuring data privacy within these distributed event stores, while also highlighting the potential transformative impact of effective anonymization and data masking techniques in this domain.

1.2 Research Challenge

In the contemporary data-driven world, the growing demand for comprehensive data privacy policies is matched by increasingly stringent regulations from governments worldwide [1,2]. However, the underlying infrastructure to adequately support these policies is markedly lacking [3,4]. This disparity poses a unique challenge, particularly when considering the demands of modern database systems to maintain high performance, characterized by low latency and high throughput.

1.3 Novelty

While there exists a body of work focusing on anonymization and data masking for data streaming [6,7,8], there is a noticeable gap of research specifically targeting distributed event stores. Furthermore, although there are enterprise technologies for managing data flowing into such systems [9], there is limited literature on techniques designed for data already within. Most notably, the concept of integrating Role-Based Access Control (RBAC) within this framework, where the role assigned determines the level of anonymity accorded to the data, is a completely novel approach.

1.4 Anticipated Impact

The integration of data privacy policies with modern data stream structures, such as distributed event stores, would be a significant innovation. The introduction of Role-Based Access Control (RBAC) coupled with anonymization in distributed event stores holds the potential to contribute to more advanced, efficient, and secure data handling. By making such tools accessible and cost-effective, companies might be more inclined to prioritize and invest in user data privacy.

1.5 Research Problem

... should include the following:

- a succinct, precise, and unambiguous statement of the research problem or question to be solved,
- goals and subproblems that will be explored, including the scope of the thesis (i.e., what is in and out of scope).

Area (Million sq. miles)	Calling Code
0.29	56
0.3	90
3.8	1
0.5	51
600	9800
Pearson = 1.0	Spearman's = 0.1

Table 1: Correlation in the existence of outlier[10].

Is there also indentation here?
Hard to tell without two lines
Oh wow there is

1.6 Outline

2 Literature Review

2.1 Role Based Access Control

2.2 Distributed Event Stores

2.3 Anonymization

The concept of data anonymity plays a pivotal role in the context of access control in distributed event stores. One of the widely accepted models for preserving the privacy of data subjects in datasets is $K - Anonymity$. Introduced by Sweeney [14], the $K - Anonymity$ model posits that each data record within a dataset should be indistinguishable from at least $K - 1$ other records with respect to any set of quasi-identifiers. Mathematically, a dataset adheres to $K - Anonymity$ if, for a set of quasi-identifiers Q, each sequence of values in Q appears at least K times in the dataset.

Techniques Cryptography, Masking Functions and there is a thirsd ?

2.3.1 Principles

2.3.2 Data Streaming

One prominent example of applying anonymization techniques to streamed data is the work CASTLE [1]

Univariate Microaggregation

2.4 Related Work

... should include the following:

- definitions / technical terms,
- theoretical foundations / principles,
- descriptions of algorithms, hardware, software, and/or systems employed.

3 Theoretical Framework

3.1 Managing Different Anonymization Granularity

When planning to integrate anonymization techniques into existing systems, there are many things to consider. First one must understand the data flowing through the system. Does it include Personally Identifiable Information (PII)? Is there further sensitive data? What part of it is necessary for system maintenance? Maybe there is additional data collected for statistics. Bearing this in mind the next thought would what needs to be anonymized. For this privacy agreements with the user must be taken into account. There may be additional government regulations in place like the California Consumer Privacy Act (CCPA) in the United States of America or the General Data Protection Regulation (GDPR) in the European Union. The next course of action is deciding on specific masking functions. As was discussed in 2.3 there is a plethora to chose from. It is important to note that all forms of anonymization lead to a loss of information. While choosing Blurring or Suppression, two methods that replace attributes with a placeholder, for all critical data fields derived in the previous assessment will ensure that all privacy concerns are addressed, it will also diminish all intelligence gained from collecting this data in the first place. It is questionable if not collecting this type of data in the first place would then be the better solution as it would save storage and computing power. An alternative approach would be to invest heavily in the IT security ensuring that no intruder with malicious intent can gain access to sensitive data. Keeping in mind, however, that social engineering attacks are nowadays the most common and effective strategy (SAUCE) giving a guarantee of safety can be impossible. It also stands to reason that the more employees a company has the risk for social engineering attacks increases. Both of these radical approaches do not seem to adequately solve the problem. Fortunately, there is a way to navigate between these two extremes. An attentive observer of the company's data operations will likely notice that data can and should be restricted similarily to permissions: Only the data needed to furfill the user's duty should be accessible to the user. In addition, there are anonymization techniques, which do not lead to total information loss like the two mentioned before. Generalization for example can be employed to significantly reduce the re-identification of individuals, while simultaneously retaining some information. With data restriction and

different anonymization techniques in mind there is a middle ground to be found which maximizes security and minimizes information loss. Consider the following example:

3.1.1 Use Case Example

Hospitals depend on the collection, management and analysis of data to administer the best and most accurate care of their patients. In a modern hospital all data would be stored in a centralized hospital database. Here all data for individual patients are brought together. Table 2 shows an exemplary table of patients in the endocrinology ward of a german hospital. Note that the tuple has been shortened to enhance its readability as well as prepended with a header of the corresponding database table. (A MORE DETAILED VERSION CAN BE FOUND IN THE APPENDIX)

pid	name	zip	sex	age	ins. co.	ins. no.	diag.	gluc.	hba1c	med.
1	F. Ott	10969	M	28	TK	K15489	E10	22.1	8.74	Insulin
2	L. Lieb	34127	F	59	AOK	Y41271	E11	16.3	7.61	Metformin
3	T. Zeit	70192	M	15	TK	Z17291	E10	23.8	8.13	Insulin
4	H. Lang	80923	F	21	TK	I79435	E10	18.9	7.99	Insulin
5	J. Putz	91757	D	24	IKK	Q29751	E10	21.2	6.04	Insulin
6	I. Spies	60819	M	68	TK	J33921	E11	19.1	5.07	Metformin

Table 2: Example table of diabetes patients

The header of table 2 shows eleven attributes. First the person id (*pid*), which is the primary key for each patient as it uniquely identifies the patient in the database. Then *name*, zip code (*zip*), *sex* and *age* are included as additional personal information. This would typically also include the full address not just the zip code, contact information, height as well as weight to adapt the dosage of the medication. The subsequent two attributes include information about the patient's insurance information. Insurance companies in Germany are uniquely identified with a nine digit institutional identifier. In the case of the first entry the insurance company (*ins. co.*) has the value 101575519, which matches the identifier of the Techniker Krankenkasse (TK). Each client is then assigned a number unique to that insurance company called the insurance number (*ins. no.*). It always starts with a letter followed by digits. Finally, the datum references the

medical information. It starts with the diagnosis (*diag.*) classified according to the International Statistical Classification of Diseases and Related Health Problems (ICD). E10 being the label for Type 1 Diabetes Mellitus, E11 the label for Type 2 Diabetes Mellitus. The most important medical measurement for the treatment of this disease is the current amount of glucose (*gluc.*) in the blood. This determines the quantity of medication (*med.*) to be administered to the patient. For Type 1 Diabetes this is Insulin, for Type 2 it is Metformin. Lastly, the table includes an attribute called *hba1c*. This is the body's own three-month average of blood glucose. By means of which diabetes is diagnosed. In this case it is also symbolic for all additional diagnostic findings. Glucose and HbA1c are intentionally distinguished as separate attributes in this dataset, despite both being blood-derived metrics, due to their distinct measurement methodologies and relevance in immediate treatment contexts. Glucose can be ascertained with a single drop of blood, providing critical information for the immediate treatment. Conversely, HbA1c is derived from a complete blood count and does not require instant action.

In a hospital setting, numerous actors engage with the aforementioned dataset. The most straightforward and prominent is the doctor. She will need all data to fulfill her duties. The doctor's letter contains all personal information. The medical data is needed for diagnosis and treatment. She will also need to keep the insurance information in mind as the covered treatment options are oftentimes different for each company. Additionally, she will need to write the patient's insurance information on the rescriptions. Only the pid could be omitted, but is debatable if the overhead is worth it, considering the pid can be easily inferred with all the given information. Therefore, no anonymization to the doctor's data makes the most sense.

Supporting the doctor is the nurse staff. One of their main tasks is to monitor patients and administer medication. To accomplish this they require the diagnosis, medication and in this case the glucose data. As the HbA1c value is not relevant for the immediate treatment it can be safely ommitted. Again insurance information is necessary as nurses typically do have the liberty of administer medication according to their own judgement. This is especially important when considering how understaffed hospitals in Germany are most of the time. On the other hand the patient's personal insurance number does not play into this. As nurses also interact directly with the patients they need some basic personal information like name and sex. Pid and zip, however, are not required. Therefore, the data for the nurse staff can be anonymized as shown in Table 3 without limiting the nurses or losing valuable information.

In tandem with the stay and medical treatment of the patient, the administar-

pid	name	zip	sex	age	ins. co.	ins. no.	diag.	gluc.	hba1c	med.
*	F. Ott	*	M	28	TK	*	E10	22.1	*	Insulin
*	L. Lieb	*	F	59	AOK	*	E11	16.3	*	Metformin
*	T. Zeit	*	M	15	TK	*	E10	23.8	*	Insulin
*	H. Lang	*	F	21	TK	*	E10	18.9	*	Insulin
*	J. Putz	*	D	24	IKK	*	E10	21.2	*	Insulin
*	I. Spies	*	M	68	TK	*	E11	19.1	*	Metformin

Table 3: Data available for the nurse staff. Note that pid, zip, insurance number and additional medical information have been suppressed as indicated by their cell's light red background.

tion of the hospital will want to collect the money from the patient's insurance. The insurance company together with the patient's personal insurance number will suffice as identification. Administered medication will be imperative as this dictates the amount of money the hospital will get in addition to the fees for the stay. For this the diagnosis will typically have to be added as a suitable reason. No further information is required. Limiting the amount of data here is crucial as here the data is exported to a third party. Which means that additional regulations will take effect. Minimizing the data leaving the hospital minimizes security risks. With these strict rules in place the data can be adjusted as seen in Table 4.
Note at this point that an unauthorized entity, who has gained access to both the data of the nurse staff and that of the administration, would struggle to correlate the entries. The shared available data fields insurance company, diagnosis and medication are likely generic enough to not point to a singular but to many patients.

Diabetes, which afflicts over ten percent of the global population and demonstrates a rising prevalence, stands as one of the most common chronic diseases worldwide [3, 12]. Given its mostly non-lethal progression and lifetime dependency on medication, it has given rise to a substatial market. As cause, optimal treatment and cure remain subject to research, data of especially newer diabetes patients is in hot demand. To provide this data to research institutes in accordance with the regulations in place the hospital must ensure that no concrete patient can be reidentified. Here, advanced anonymization techniques such as K-Anonymization come into place. Each attribute of the data entry can be assigned to one of three categories: personally identifiable, quasi identifying and sensitive attributes. To achieve k anonymity each entry must suppress the per-

pid	name	zip	sex	age	ins. co.	ins. no.	diag.	gluc.	hba1c	med.
*	*	*	*	*	TK	K15489	E10	*	*	Insulin
*	*	*	*	*	AOK	Y41271	E11	*	*	Metformin
*	*	*	*	*	TK	Z17291	E10	*	*	Insulin
*	*	*	*	*	TK	I79435	E10	*	*	Insulin
*	*	*	*	*	IKK	Q29751	E10	*	*	Insulin
*	*	*	*	*	TK	J33921	E11	*	*	Metformin

Table 4: Data available for the administration. Note that only the insurance information, medication and diagnosis are not suppressed.

sonally identifiable attributes, while keeping the sensitive attributes untouched. Most importantly the quasi identifiable attributes of each data entry must be the same for at least k - 1 other entries of a data set. This is typically achieving with generalization of these attributes until k entries are found. In this use case the personally identifiable attributes are *pid*, *name* and *insurance number*. The quasi indentifying attributes are *zip*, *sex*, *age* and *insurance company*. The medical data comprive the sensitive attributes. A K anonymous version of this data extry is depicted in Table 5.

While the aforementioned diabetes patient use case scenario may appear unique and specific, the aspects and nuances are applicable in numerous contexts. The distinct data requirements for doctors, nurses, administration and research are anticipated to persist, albeit adapted, throughout the entire healthcare industry. It is also viable in different sectors. Imagine security levels in government matters, trade secrets and specific customer knowledge in corporations or secrecy of correspondense for the transportation industry. Distributed event stores are utilized across all of these sectors with major players relying on Apache Kafka for their day to day needs [5].

3.2 Anonymization Techniques

Data anonymization is a multifaceted process tailored to meet application requirements, user groups and their specific needs. This section first goes in depth into masking functions, the cornerstone of data anonymization. It then explores table based techniques that look at the whole dataset to ensure privacy. In this context a *datum* refers to a single piece of information e.g. a singular entry of a database

pid	name	zip	sex	age	ins. co.	ins. no.	diag.	gluc.	hba1c	med.
*	*	XXXXX	M	[10 - 70]	TK	*	E10	22.1	8.74	Insulin
*	*	XXXXX	M	[10 - 70]	TK	*	E10	23.8	8.13	Insulin
*	*	XXXXX	M	[10 - 70]	TK	*	E11	19.1	5.07	Metformin
*	*	XXXXX	{M, F, D}	[10 - 70]	ins. co	*	E11	16.3	7.61	Metformin
*	*	XXXXX	{M, F, D}	[10 - 70]	ins. co	*	E10	18.9	7.99	Insulin
*	*	XXXXX	{M, F, D}	[10 - 70]	ins. co	*	E10	21.2	6.04	Insulin

Table 5: K-anonymized data available for external research. The sensitive medical attribtues remains unchanged as indicated by the white cell background. Unlike the personally identifying attributes, which have been suppressed as denoted by the red cell background. The yellow cell background highlights the generalized quasi identifiable attributes. Note that the entries of the first group have unchanged values for the attributes *sex* and *ins. co.*. As all three original entries shared the same value it did not need to be generalized.

or any one event of a data stream. Synonymous with it also used the word *tuple*. A tuple is comprived of one or multiple *attributes* (also called *fields*). It is customary for individual tuples to be part of a bigger collection. In the static context they are collected in databases and grouped in tables. Data streams work analogous for dynamic operations. All tuples of the same database table or data stream are required to follow the same pattern. This refers to the sequence of attributes of each tuple. This pattern is fixed in a data schema associated with the table or stream. Sometimes it is appended to each datum in form of a header as seen exemplary in Table 2. As this substantially increases the size of each datum it is more common to define it once in the initialization step of the database table or data stream.
While all forms of anonymization aim to alter the underlying sensitive information, e.g. the tuple's attributes containing perhaps personally identifiying information, in a way that makes it harder if not impossible to reconstruct, they can be further categorized based on their scope of operation:

- **Value-Based** Handles one tuple at a time and replaces the values of attributes independently.
- **Tuple-Based** Operates on individual attributes of a single tuple, but considers the values of the entire tuple for the change.

- **Attribute-Based** Extends the view from one tuple to a larger collection or table of data. Evaluates the values of singular attributes of the entire set and collectively makes changes to that attribute accordingly.
- **Table-Based** Covers a table of data and perceives all attributes of each tuple. Adaptions to multiple attributes simultaneously are common. It can be argued that methods falling under this category are not masking functions but algorithms utilizing many masking functions to achieve anonymization on a table level.

To better illuminate the relationship and hierarchy of the aforementioned categories as well as provide some examples, refer to Figure 1.

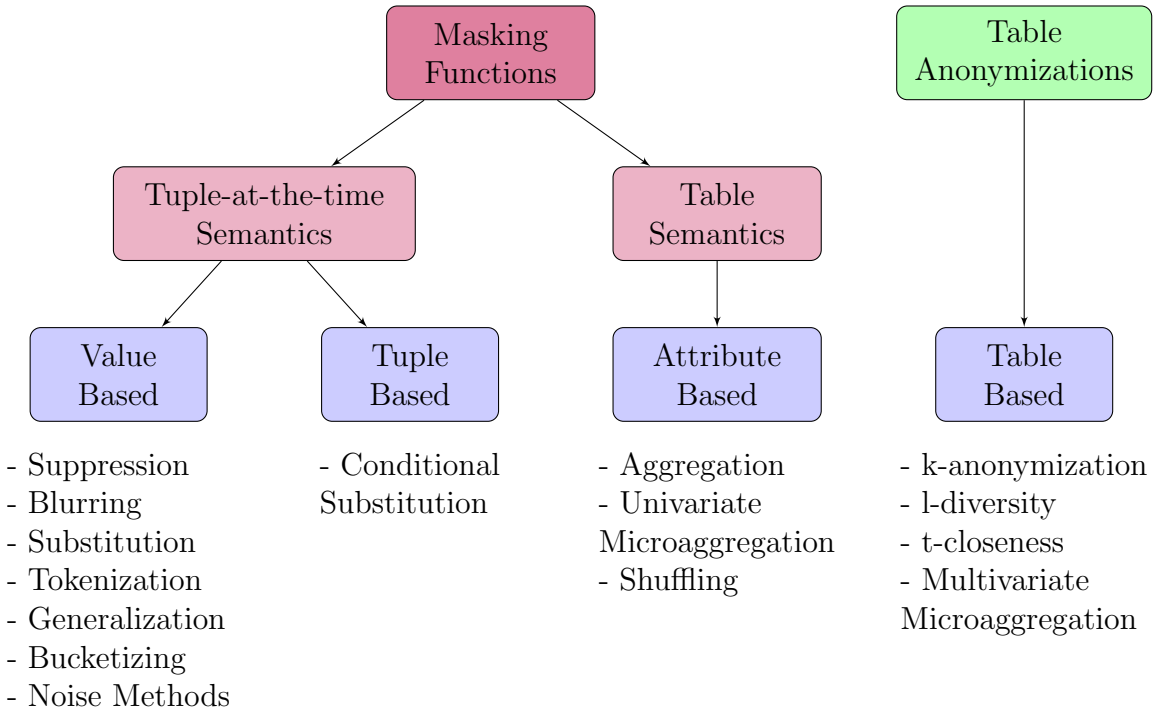


Figure 1: Hierarchy of masking functions

3.2.1 Value Based Masking Functions

Having outlined the structure and dimensions of the various masking functions, it is now time to take a closer look at the functionality and use cases of the examples given in Figure 1 starting with the value based.
Suppression aims to effectively delete the value by replacing the value of an

attribute with a meaningless character, most commonly the asterisk *. It is important to note, that actually removing the attribute from the tuple or replacing it will a null value would violate the data schema and thus negatively impact operability. The asterisk does the trick while maintaining the data schema. To work Suppression only requires a non-empty set of keys for the attributes that are supposed to be suppressed as parameter. Naturally, it leads to total information loss of the specfified fields. This can be particular useful for fields containing PII like a person's home address as exemplary shown in Figure 2.

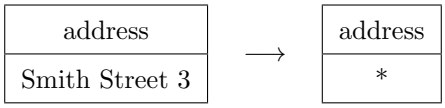


Figure 2: Example of suppression of an attribute.

A similar approach is applied in **Blurring**. Here the value of an attribute is replaced with arbitrary characters, typically Xs. It is distinguishable from Suppression in that not all characters of the value have to be replaced and even the amount of characters can remain the same. Imagine a user at the checkout of an online store that they have already purchased goods at prior to this session. Here the credit card information of that user was saved as part of the agreement from the previous session. The user is then given the option to use that credit card again, with it being specified as a sequence of blurred characters with only the last three digits in plain text as shown in Figure 3. This allows the user to double-check the card information without exposing the credit card number to the network, screen capturers or bystanders. This operation also leads to high information loss but retains some usability of the value. The parameters for Blurring include the keys to blurr as well as optionally the number of characters and whether the amount of characters is to be maintained. Note, that setting the parameters to blur all characters and reduce the amount to one is equal in funtionality to Suppression.

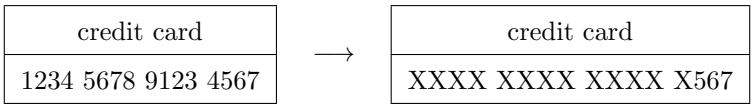


Figure 3: Example of blurring of an attribute.

Substitution replaces the value of the specified attribute with a predefined substitute. Figure 4 shows an example where a name is switched out with an

arbitrary fake name from a library. While this masking function leads to substantial information loss, it seemingly maintains the integrity of the data from an outsider's perspective. This can make the data easier to work with, while still ensuring anonymity. As parameters the keys for the attributes that are supposed to be substituted are required in tandem with the intended substitutes.

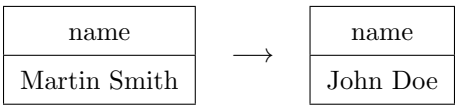


Figure 4: Example of substitution of an attribute.

An alternative approach is **Tokenization**. Here values are also substituted, but not with some arbitrary replacement. Instead, the substitute is a specific token. These tokens can be reversed to restore the original value as long as a secret is known with which the token was created. There are different approaches to achieve this. The first one coming to mind is a database mapping token to the hash of the original value. Only access to the database as well as the hashing function will yield the correct original value from the token. Another approach is to omit the database and instead use a more sophisticated cryptographic algorithm to create the token, essentially encrypting the data. While this masking function manages to preserve the information, it requires substantial overhead in form of storage for the database and/or computing power for the cryptographic algorithms. This costly masking function is usually reserved for highly sensitive data. For instance passwords as illustrated in Figure 5.

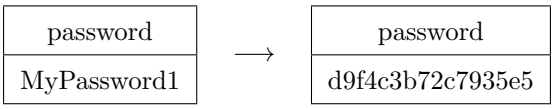


Figure 5: Example of tokenization of an attribute.

One of the most common masking functions is **Generalization**. Here, the value of an attribute is abstracted to a more general value. This effectively reduces the information, but does not remove it entirely. For example a datum with a residency field could generalize the exact location to a broader one. Instead of Berlin it would read Germany as shown in Figure 6. This could then be even further generalized to Europe and so forth. Typically, entire generalization hierarchies are provided as parameter to facilitate this. It is crucial that these hierarchies are exhaustive of

all possible arising values if no default is provided as generalization is ambiguous. These complete generalization hierarchies are required as parameters in addition to their respective attribute key.

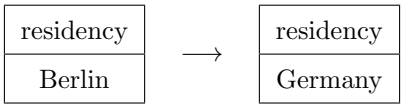


Figure 6: Example of the generalization of an attribute.

A special case of Generalization is **Bucketizing**. It functions similarly, but exclusively for numerical values. Ranges replace specific values in the tuple. Figure 7 shows an example where the age 27 is bucketized to the range [20 - 30]. Only a moderate amount of information is lost with this masking function. To deploy it requires the bucket sizes as well as the keys to the numerical attribute.

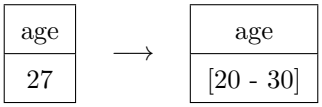


Figure 7: Example of bucketizing of an attribute.

Finally, there are the **Noise Methods**. Again, these only apply to numerical data. The idea is to modify the original values by adding noise. Typically, this noise is chosen randomly from a distribution. Utilizing the normal distribution with mean 0 it would ensure that the data over a period of time would retain its average and distribution. The standard deviation will then define how much each individual data point can diverge from the original value. The result will invalidate individual tuples but preserve the overall spread of the data in the long run. As an example Figure 8 shows noise added to two attributes of a tuple. Note that with no loss to the generality one is decreased, while the other increased.

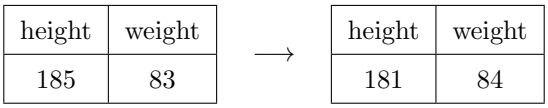


Figure 8: Example of adding noise.

3.2.2 Tuple Based Masking Functions

Extending the scope of operation from independent attributes to the tuple as a whole yields the tuple based masking functions. The most notable candidate is **Conditional Substitution**. As can be expected the change to the tuple is identical to that of the value based Substitution. The value of the attribute specified by the parameter is changed according to the given substitution dictionary. The key difference, however, is that a change only occurs if a certain condition is met. These are additionally provided as a parameter. They can be specified in a variety of different possible formats like a direct match, a numerical range or even a regular expression. Before showcasing some examples consider the following definitions:

Let T be a tuple defined as a fixed sequence of attributes a_0 to a_n, $T = (a_0, a_1, \ldots, a_n)$. Also, let $i, j \in \{0, 1, \ldots, n\}$.

Further, a masking function for conditional substitution mf_{cs} can be defined based on a condition c evaluated on the attribute a_i, with a substitute s for the attribute a_j:

$$mf_{cs}(i, c, j, s) = \begin{cases} (a_0, \ldots, a_{j-1}, s, a_{j+1}, \ldots, a_n) & \text{if } c \text{ matches on } a_i \\ T & \text{else} \end{cases}$$

Note, a match on c can take on different forms. An example of a conditional substitution masking function for a direct match on the value of one of the attributes is shown in Figure 9. It shows an excerpt of a database table with three entries. Remember that all tuples of a single database table share the same data scheme. For better understanding the header with the attribute's labels is shown in the first row of each table in the figure. It shows that the data in this table has two attributes, *rank* and *salary*. The masking function $mf_{cs}(0,'Manager', 1,'*')$ is applied to every tuple in the table. It can be read as: if the attribute with index 0 matches on 'Manager', substitute the attribute with index 1 with '*'. Following this logic only one entry in the output was changed, as there was just one match.

Another example is shown in Figure 10. The condition in this case is a range. Naturally, it can only be applied to numerical attributes since there is a clear ordering of values. In the example the database table shows two attributes: *name* and *age*. The masking function employed is $mf_{cs}(1, [0, 18], 1,'minor')$. Subsequently, the output shows two changes in the table as two entries match the condition. Note that this time the condition is evaluated on the same attribute as the substitution e.g. $i = j$.

Finally, Figure 11 shows $mf_{cs}(0,'@example\backslash.com', 1, 0)$ being used as masking function. The attribute *Points* for all entries with the domain 'example.com' in

rank	salary
Worker	62 000
Assistant	45 000
Manager	135 000

$\xrightarrow{mf_{cs}(0,'Manager',1,'*')}$

rank	salary
Worker	62 000
Assistant	45 000
Manager	*

Figure 9: Conditional substitution with a direct match condition taking the entire tuple into consideration.

name	age
John	45
Frederik	7
Samatha	15

$\xrightarrow{mf_{cs}(1,[0,18],1,'minor')}$

name	age
John	45
Frederik	minor
Samatha	minor

Figure 10: Substitution with a range condition on the basis of only a singular attribute in the tuple.

the *Email* attribute, determined through the regular expression $'@example\backslash.com'$ is set to 0.

Email	Points
user1@example.com	150
service@mail.org	325
john@example.com	25

$\xrightarrow{mf_{cs}(0,'@example\backslash.com',1,0)}$

Email	Points
user1@example.com	0
service@mail.org	325
john@example.com	0

Figure 11: Regular expression as a condition for the substition as part of a tuple based masking function.

3.2.3 Attribute Based Masking Functions

Following the hierarchy of masking functions depicted in Figure 1 this concludes the masking functions with Tuple-at-the-time semantics. Next, the focus shifts towards operations on tables, more specifically on collections of tuples. Before, the masking functions were applied to each tuple individually. The focus is to

add more context to the anonymization and in return diminishing the information loss caused by anonymizing altogether. Also, it allows the better handling of outliers. Returning to the generalization example from Figure 6, where the residency 'Berlin' was generalized to 'Germany'. If in an entire table there is only one german resident, it would be easy for an unauthorized party, who has gained access to this table, to reidentify the individual from the output. All the masking operation would have done is cost resources and lead to information loss within the system. This is of course not desirable. Fortunately, edge cases like these are unlikely for data with low spread or high density. And even in data sets with high spread and likelihood of outliers, situations like the aforementioned can be easily avoided with knowledge of the data and the correct choice of masking function for the appropriate edge cases e.g. Suppression. An approach that more complex anonymization techniques like the ones discussed in Section 2.3.2 employ inherently. The subsequent subsection will go into more depth on this as well. For the moment, it is crucial to understand that more input can lead to less information loss and more anonymization reliability. It does, however, come at higher computing cost as the findings in Chapter 5 show. More input in the context of attribute based masking functions refers to more values from multiple tuples for the same attribute. Whithin the scope of this thesis it is essential to highlight the particular interest in data streams being the foundation of distributed event stores as opposed to static databases or bounded datasets. Data streams inherently pose additional challenges due to their unbounded nature. In Section 2.3.2 this has been addressed already and windowing was introduced as a means of discretizing data streams. In this context 'collections of tuples', 'table' and 'window' all refer to this same discretization: a finite number of tuples as working base obtained through windowing operations. It is worth mentioning that the parameters for windowing like window size can be included in the optimization of the anonymization itself. This particular nuance is looked into in Section 5.2

Building on this understanding, **Aggregation** as an attribute based masking function leverages this principle. In essence, aggregation combines multiple values into one single value. This in turn will represent the attribute for all input tuples. This effectively reduces the information of an individual entry but preserves the underlying trend. By merging data it significantly reduces the reidentification probability of a single individual through this attribute. There is a variety of aggregation techniques, but they all have one thing in common. They work only on numerical values. Figure 12 shows an example. The table contains six tuples, each tuple only has one attribute, *age*. The right side of the arrow illustrates the result. Note that the amount of tuples remains unchanged, but now all tuples have the same value for the attribute. Depending on the aggregation type, which is written underneath each column, the values have been combined. The first column shows

the sum, the second the median, the third the average, the fourth the maximum value found, the fifth the minimum value found, the sixth the number of input tuples and the last the most frequent value in the input table.

age
23
45
26
32
26
27

$\longrightarrow$

age						
180	32	30	45	23	6	26
180	32	30	45	23	6	26
180	32	30	45	23	6	26
180	32	30	45	23	6	26
180	32	30	45	23	6	26
180	32	30	45	23	6	26
Sum	Median	Average	Max	Min	Count	Mode

Figure 12: Aggregation operations on the "age" attribute.

A special form of aggregation is **Univariate Microaggregation**. This masking function specifically optimizes the goal of minimizing information loss. The method accomplishes this by clustering similar data. The aggregation is then only applied on these groups. This approach yields a table with greater variance in data than what is observed with standard aggregation. Also, outliers have less effect on the overall table. Of course the clustering of data will require additional resources. One approach to univariate microaggregation is offered by Hansen and Mukherjee [4]. They formulate the univariate microaggregation problem as a table consisting of n entries to be divided into subgroups with at least k observations with minimized spread of the original data within the subgroups. The authors transform the univariate microaggregation problem into an equivalent graph problem. They then prove that this graph problem can be solved using a shortest-path algorithm in polynomial time. The authors commence by collecting all n values for the given attribute and sorting them in ascending order to form a vector V with $|V| = n$. Each V_i corresponds with an original value from the table, now at position $i, i \in [1, n]$. They go on to construct a directed weighted graph $G_{k,n}$. The nodes are labeled with i and added with a source node with label 0. The graph has a directed edge $e(i, j)$ from each node i to node j if $i + k \leq j \leq i + 2k$. The edge $e(i, j)$ is then associated with the group of values, called the 'corresponding group', in V with index h where $i \leq h \leq j$. The edge $e(i, j)$ is then assigned a weight equal to the Sum of Squared Errors (SSE) of the corresponding group. Finally, they prove that the shortest path in $G_{k,n}$ is the optimal solution of the

univariate microaggregation problem. Algorithm 1 shows pseudocode for their proposed solution.

Algorithm 1: Pseudocode for the univariate microaggregation approach by Hansen and Mukherjee [4]

Parameters: T: Table to be anonymized
a: Attribute in T's Schema subject to microaggregation
k: Minimum observations per group
Output : Shortest path in the form of a list of edges
$V \leftarrow$ Values of column a in T sorted in ascending order
$n \leftarrow |V|$
Initialize $G_{k,n}$ as an empty graph
Add a source node s to $G_{k,n}$ with label 0
for i *from* 0 *to* n **do**
  for j *from* $i + k$ *to* $\min(i + 2k - 1, n)$ **do**
    Compute the corresponding group CG from $V[i : j]$
    Compute the mean $\bar{x}$ of CG
    Compute the SSE for CG
    Add a directed edge $e(i, j)$ to $G_{k,n}$ with weight SSE
  end
end
Apply Dijkstra's algorithm on $G_{k,n}$ starting from s
return *The list of edges forming the shortest path in* $G_{k,n}$

Let us break this down. Remember that the core idea is to minimize information loss. Less spread in a group that is subject to aggregation is key here. Therefore, ordering the elements before grouping neighbors ensures that for any two elements within a group there does not exist another element in another group that lies between them. The question now is where to start and end each group. They then go on to construct a graph by adding nodes for each value in V. Intuitively, these are all possible starting positions of subgroups. The only constraint is that each group must at least contain k observations. The compund inequality $i+k \leq j \leq i+2k$ for the creation of edges enforces this constraint and additionally limits the number of elements within a group to $2k-1$. This is important as larger groups lead to more information loss. However, limiting the group size to exactly k is not feasible, as n is not required to be a multiple of k. If $n \mod k \neq 0$ at least one group must contain more than k elements. Now each node i is connected via an edge $e(i, j)$ to all nodes j that are at least k and at most $2k - 1$ elements away from it. The corresponding groups to each edge $e(i, j)$ are then evaluated for their closeness to each other. This is done by calculating the within squared error SSE. First the

mean $\bar{x}$ of the values in the corresponding group cg is calculated. Then for each value x_i in cg the squared error sqe_i is calculated, $sqe_i = (x_i - \bar{x})^2$. The SSE for a corresponding group cg with h values is then equal to the sum of sqe_i for all x_i in cg: SSE $= \sum_{i=1}^{h} sqe_i$. With the directed weighted graph $G_{k,n}$ defined and an artificial source node labeled with 0 connected to the first k to $2k-1$ nodes properly weighted, a shortest path algorithm is applied. In Algorithm 1 Dijkstra [2] is used. Finally, the optimal solution is the grouping according to the corresponding groups of all the edges of the shortest path.

Shuffling is another attribute based masking function. The technique involves rearranging the values of the specified attribute. The value of the attribute for an individual tuple is exchanged with that of another tuple in the same table. Thus, from an observing standpoint, the overall data actually remains exactly the same. No information is lost. In addition, reidentification on basis of the value of this attribute is no longer possible as the likelihood of the value belonging to the original tuple is slim. Consequently, shuffling diminishes the direct applicability or interpretability of the changed tuple. It becomes impossible to maintain valid statistical correlations among multiple attributes within a tuple. Also, users of the data can no longer rely on the veracity of individual data in the table. The negative impact on statistical value and tuple autheticity intensifies with increasing data spread. Mixing outlying values into otherwise inlying tuples can prove to be detrimental when performing data analysis. Taking into account Shuffling's strengts and weaknesses a primary area of application is with personally identifying attributes. In this context, outliers are nonexistent, and ideally, statistics on the basis of these attributes are rare or even absent. Consider the example shown in Figure 13. It depicts a table consisting of five tuples with three attributes *name*, *residency* and *purchase* respectively. Shuffling is separately applied to each of the two attributes containing PII, *name* and *residency*. Note, the names do not correspond with the residencies anymore, this is due to them being shuffled independently. Also observe that the residency of the second tuple is unchanged. When shuffling each element is switched with an element at a random index in the set. Therefore, it can occur that an element is actually shuffled with itself. The implementation itself can then be enhanced with a seed to effectively determine the randomness if that is to be desired. The overall result in this example shows that no person can be reidentified from a purchase, while the overall data volume and variety has been maintained. On a statistical standpoint no harm has been done either as the primary interest does not lie in personally indentifiable attributes. It is worth noting that while names are shuffled, they still persist in the dataset. In situations requiring extreme confidentiality, this might not provide sufficient anonymization. Another risk emerges if another database table with shared

attributes exist; such tables could be used to cross-reference tuples, potentially nullifying the effectiveness of this masking function.

name	residency	purchase
Jane	Hillside	Laptop
Alice	Cityburg	Teapot
John	Lakefront	Desk
Eve	Townsville	Camera
Bob	Villagetop	Bookshelf

$\longrightarrow$

name	residency	purchase
John	Townsville	Laptop
Jane	Cityburg	Teapot
Alice	Villagetop	Desk
Bob	Hillside	Camera
Eve	Lakefront	Bookshelf

Figure 13: Arbitrary customer data where the values of the attribute "name" as well as "residency" is shuffled throughout the table.

3.2.4 Table Based Anonymization Techniques

Extending the scope of operation from individual attributes within a larger collection of tuples to the entirety of the tuple with all its attributes, it is in this context the concept of table based anonymization techniques emerges. It is essential at this point to remark that these table based anonymization techniques rely on a fundamentally different premise than masking functions. While masking functions main aim is to obscure or change pieces of data, table based anonymization techniques operate on a holistic approach. The individual datum is only regarded as part of the larger set, which in turn is to be remolded to fit into predefined data privacy policies. To elaborate on this difference further think of the underlying methodology of the masking functions thus far. They all intend on masking sensitive information be it in form of an attribute or tuple, they make specific changes to it to ensure that individual values are no longer recognizable. They suppress, obscure and generalize the original values. The main intent here is to mask or hide the sentive information, essentially camouflaging it to deter any reidentification. On the other hand table based anonymization techniques do not target individual data points. They use more sophisticated methods of analyzing the data set as a whole and transforming the entire table ensuring that it aligns with established privacy standards. To facilitate the comprehension of these concepts, consider the Table 6 as a working example throughout this subsection. It depicts a table with five columns. In typical relational databases, attributes can be categorized into three main types: Personally Identifiable Information (PII), Quasi-identifiers, and Sensitive attributes. Personally Identifiable Information (PII) is defined as any

information that can be used to explicitly identify an individual. In this example the *name* attribute serves this purpose. Quasi-identifiers are attributes that can be used to implicitly identify an individual when correlated with other information. In the current example, the attributes *residency* and *age* function as quasi-identifiers. Sensitive attributes contain information collected about an individual but must be protected from being traced back to the individual once anonymized. In this table, *diagnosis* serves as a sensitive attribute. Finally, the example contains a nameless leading column, marked as 'Row Index'. While not part of the table's data, it serves as a reference to facilitate the reader's understanding, especially as rows will be rearranged and anonymized subsequently.

Row Index	PII	Quasi-Identifier		Sensitive
	name	residency	age	diagnosis
1	Ahmed	Berlin	27	Covid
2	John	Glasgow	45	Asthma
3	Thomas	Munich	32	Covid
4	Anna	Madrid	59	Diabetes
5	Winston	Manchester	20	Covid
6	Kim	Stuttgart	54	Covid
7	Miguel	Barcelona	34	Asthma
8	Farah	London	41	Diabetes
9	Jane	Bilbao	70	Diabetes

Table 6: Working example for table based anonymization containing a table with four attributes marked above with their corresponding category. Additionally, there is a leading column, which is not part of the data, but serves as reference.

The most notable concept is ***k*-anonymity**. In the definition by Sweeney from 2002 [14] k-anonymity demands that any one entry in a table must be indistinguishable to $k-1$ others in regard to its quasi-identifiers. Table 7 shows a 3-anonymized version of the working example. The entries were grouped into sizes of $k = 3$ according to their similarity in quasi-identifiers. These then were generalized to encompass all values in a group making them indistinguishable. The *name* was suppressed, while the *diagnosis* remained unchanged. Now no individual can be reidentified given this table. Over the years a number of algorithms have emerged

to efficiently transform a set of data to conform to k-anonymity. Notabely, this includes Datafly by Sweeney herself [13] as well as Mondrian [7] and Incognito [6] by LeFevre et al. Each of these algorithms has its own computational complexity, quality of results, and applicability to different kinds of data. Datafly for example is more suited for categorical data, while Mondrian is more optimized for numerical data whereas Incognito focuses on the optimization for computational efficiency. As this thesis's focal point is dynamic data in the form of data streams another algorithm emerges as the most relevant - 'CASTLE: a δ-constrained scheme for k_s-anonymizing data streams' by Cao et al. [1].

CASTLE

CASTLE stands for Continuously Anonymizing STreaming data via adaptive cLustEring. Streaming data differs from static databases in two key aspects: first, it is unbounded, necessitating an approach that can handle data of indeterminate size. The second distinction is that streaming data is append-only; that is, entries in a data stream are not modified or deleted once added. However, the significance of entries may diminish over time, a factor that the authors of CASTLE address by focusing on the *freshness* of the data. They do this by defining a dynamic variant of k-anonymity called k_s-anonymity. Each tuple t of a stream S consists, as always, of personally identying attributes $p_1, \ldots, p_n$, quasi-identifiers $q_1, \ldots, q_n$ and sensitive attributes $s_1, \ldots, s_n$. Additionally, it is given another attribute, the position p in S. The anonymized result of the stream S is termed S_{out}. Given a position P, the authors consider S_{out} k_s *anonymized up to* P if the collection of all tuples with $t.p \leq P$ in S_{out} is k-anonymized. To further facilitate the requirement of freshness they add a δ-constraint to their scheme. It says that any such stream S satisfied the δ-constraint if all tuples with a position less than $t.p - \delta$ are already in S_{out}. Keep in mind that there is a time difference, the processing time, between S and S_{out}. The algorithm consumes the plaintext stream S one tuple at a time and produces the anoynmized stream S_{out} in batches. The δ-constraint limits the number of tuples that are being processed by CASTLE at any one time and subsequently ensures that fresh data is produced to S_{out}.

Minimizing information loss is important and for k-anonymity the only leeway lies within the quasi-identifiers as they are generalized. Sensitive attributes remain untouched and PII is suppressed. To be able to minimize information loss, it first has to be quantified. For a numerical attribute q_i and the generalization g to the range $[l, u]$, from the domain $[L, U]$ of q_i, the corresponding information loss is defined by the authors as follows:

$$VInfoLoss(g) = \frac{u-l}{U-L}$$

The information loss for categorical attributes is calculated according to their generalization hierarchy. Figure 14 shows the generalization hierarchy of the attribute *residency* used in the working example (Table 6)

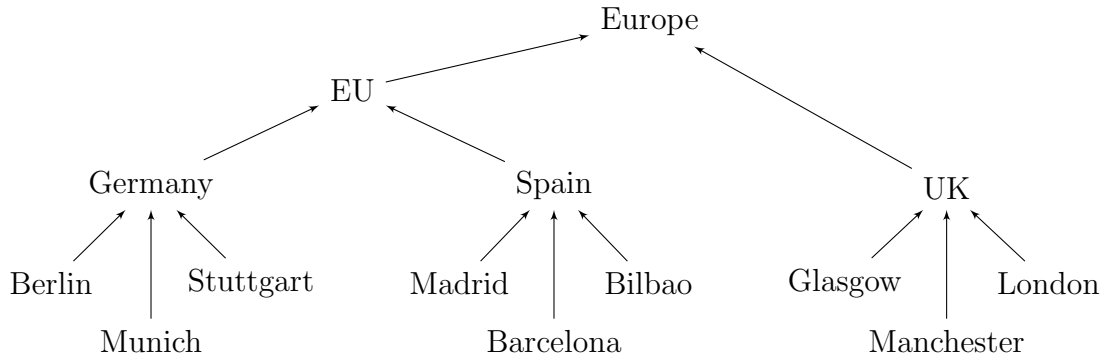


Figure 14: Generalization hierarchy for the attribute *residency*.

The information loss for the generalization g of a categorical attribute to a node v in the categorical attribute's generalization hierarchy is defined as follows:

$$VInfoLoss(g) = \frac{|S_v|-1}{|S|-1}$$

Here, $|S_v|$ is the number of leaf nodes of the subtree rooted at v and $|S_v|$ is the overall number of leaf nodes in the generalization hierarchy. No generalization, e.g. the attribute remaining the value of a leaf node in its generaliation hierarchy, would lead to $\frac{1-1}{|S|-1} = 0$ no information loss. An example of the attribute *residence* and its generalization hierarchy as it is shown in Figure 14 would be the generalization to 'Germany' leading to an information loss of $\frac{3-1}{9-1} = \frac{2}{8}$.
Overall, the information loss G due to the generalization of its n quasi-identifiers is defined as follows:

$$VInfoLoss(G) = \frac{1}{n} \sum_{i=1}^{n} VInfoLoss(g_i)$$

The CASTLE scheme revolves around the clustering of data. Each cluster encompasses a number of tuples and is defined by the common generalization of the quasi-identifiers of its members. As mentioned before the algorithm consumes one tuple t at a time from the stream. Information loss is key to the assignment of t to a cluster. For all clusters the enlargement cost is determined. It is the difference in information loss of the cluster if it generalized its quasi-identifiers to include t and the current information loss of the cluster. If t already fits whithin the generalization of the cluster, the enlargement cost is 0. The tuple is then assigned to the cluster with the minimal enlargement cost.

	name	residency	age	diagnosis
1	*	Germany	[25 - 55]	Covid
3	*	Germany	[25 - 55]	Covid
6	*	Germany	[25 - 55]	Covid
2	*	UK	[20 - 45]	Asthma
5	*	UK	[20 - 45]	Covid
8	*	UK	[20 - 45]	Diabetes
4	*	Spain	[30 - 70]	Diabetes
7	*	Spain	[30 - 70]	Asthma
9	*	Spain	[30 - 70]	Diabetes

Table 7: 3-anonymized version of the working example. Suppressed fields are in red cells, generalized fields in yellow.

While k-anonymity offers protection against identity disclosure, it is insufficient for guarding against attribute disclosure. Specifically, k-anonymity does not impose diversity on sensitive attributes within each equivalence class (i.e., a group of at least k elements indistinguishable in respect to their quasi-identifiers). As a result, an attacker can still infer the sensitive attribute of an individual with a high degree of confidence if all k records in the same equivalence class share the same sensitive value. Recall the 3-anonymized version of the example in Table 7. All entries within the first group share the same value for the sensitive attribute *diagnosis*. To mitigate this vulnerability, ***l*-diversity** is introduced as an enhancement to k-anonymity. It requires that each equivalence class contain at least l distinct values for the sensitive attributes, thereby adding a layer of protection against attribute disclosure. Concretely, Machanavajjhala et al. define the principle of l-diversity in their paper "l-Diversity: Privacy Beyond k-Anonymity" [9] as follows: "A q^*-block is l-diverse if it contains at least well-represented values for the sensitive attribute S. A table is l-diverse if every q^*-block is l-diverse." In this context a "q^*-block" refers to an equivalence class and, in its simplest form, "well-represented" is to be understood as *distinct* values according to their definition. This is also the definition used by Cao et al. in CASTLE when illuminating their extension of the algorithm proposed by them to allow for the adherence of l-diversity. In fact the extension refers only to a small change within the delta constraint logic of their algorithm as well as a renewed cluster splitting approach. Essentially, it ensures that clusters that are meant to be returned are certain to contain at least l different values for the sensitive attributes. If this is not the case

clusters are merged and subsequently bucketized in accordance to their values of the sensitive attributes. Should a bucket contain k elements it is ready to be split from the rest and output. Take a look at Table 8, it shows a 2-diverse version of the working example. The groups have been adjusted to ensure that they each conatin at least two distinct sensitive values. This has come at the cost of generalizing the *residency* attribute one more time.

	name	residency	age	diagnosis
1	*	EU	[25 - 55]	Covid
6	*	EU	[25 - 55]	Covid
7	*	EU	[25 - 55]	Asthma
2	*	UK	[20 - 45]	Asthma
5	*	UK	[20 - 45]	Covid
8	*	UK	[20 - 45]	Diabetes
3	*	EU	[30 - 70]	Covid
4	*	EU	[30 - 70]	Diabetes
9	*	EU	[30 - 70]	Diabetes

Table 8: 2-diverse version of the working example. Suppressed fields are in red cells, generalized fields in yellow.

Adhering to the l-diversity principle, however, does not protect against attackers with background knowledge. Similarity attacks for example, which exploit the inherent characteristics and structural patterns within the data to compromise their anonymity, are still effective. In the context of the 2-diverse table presented earlier (see Table 8), the susceptibility to similarity attack becomes evident. Take, for instance, the first equivalence class characterized by residency in the EU and age between 25 and 55 years. The diagnoses within this class - Covid and Asthma - are both lung-related conditions. An attacker could easily infer that an individual belonging to this group is highly likely to be suffering from a respiratory disease. This is addressed by ***t*-Closeness** as it takes the distribution of sensitive attributes into consideration. Li et al., who have coined the term in their paper 't-Closeness: Privacy Beyond k-Anonymity and l-Diversity' [8], define the t-closeness principle as follows: "An equivalence class is said to have t-closeness if the distance between the distribution of a sensitive attribute in this class and the distribution of the attribute in the whole table is no more than a threshold t. A table is said to have t-closeness if all equivalence classes have t-closeness.". This principle serves as an enahnced metric for privacy preservation by addressing the distributional

skewness in sensitive attributes that might otherwise lead to privacy leaks. The advantage of t-Closeness stemming from the face that the overall distribution of the sensitive attributes is mimiced in each equivalence class, thereby minimizing the changes of attribute disclosure thorough background knowledge or data patterns. This principle applied to the working example is shown in Table 9. As the overall distribution of respiratory diseases in the original table (Table 6) is 2/3 the entry with row index 4 instead of 7 is exchanged within the first and third group. This leads to a preservation of the distribution at a minor cost of generalization in the *age* quasi-identifier.

	name	residency	age	diagnosis
1	*	EU	[25 - 60]	Covid
4	*	EU	[25 - 60]	Diabetes
6	*	EU	[25 - 60]	Covid
2	*	UK	[20 - 45]	Asthma
5	*	UK	[20 - 45]	Covid
8	*	UK	[20 - 45]	Diabetes
3	*	EU	[30 - 70]	Covid
7	*	EU	[30 - 70]	Asthma
9	*	EU	[30 - 70]	Diabetes

Table 9: T-closeness version of the working example. Suppressed fields are in red cells, generalized fields in yellow.

An honorable mention for the table based anonymization techniques is **Multivariate Microaggregation**. This method extends Univariate Microaggregation by simultaneously considering multiple attributes and aggregating them in a conjunctive manner. To achieve this, it groups records into clusters of at least k similar records, where k is the microaggregation factor, and subsequently replaces each record in the cluster with the cluster's centroid. Although this ensures that no individual record can be uniquely identified, it incurs a substantial computational cost. The computational intensity is such that, to date, no efficient algorithm for this problem has been identified. In fact, Oganian and Domingo-Ferrer have proven that multivariate microaggregation is an NP-hard problem [11].

3.3 System Requirements

3.3.1 Data Stream Integration

3.3.2 Administration

3.3.3 Performance

3.3.4 Adaptability

3.3.5 Scalability

3.3.6 Reliability

4 Implementation

4.1 Apache Kafka

4.2 Apache ZooKeeper

4.3 Anonymized Kafka

Class Diagram

4.3.1 Stream Manager

4.3.2 Configuration parsing

4.3.3 Validation

4.3.4 Stream Config Builder

4.3.5 Kafka Streams

4.3.6 Anonymizers

4.4 Test Suite

4.4.1 Data Generator

4.4.2 Kafka Connector

4.4.3 Consumers

4.5 Docker

Putting it all together Docker compose Network Volumes Dependencies Individual Dockerfiles
... should include the following:

- research methodology (e.g., prototype and experiments, case study, literature survey, theoretical analysis),
- derivations and descriptions of algorithms, hardware, software, and/or systems developed.

Algorithm 2: Splitting a Session[15].

Parameters:
e : Tuple to be inserted.
te (e) : Event-time of e.

S ← slice that covers te (e);
if *S starts at te (e)* **then**
  //Slice before S must be fixed.
  change the type of the slice before S to combined;
  add e to S;
else
  // S does not start at te (e).
  change tend (S) to te (e) (excluding te (e) from S);
  change type of S to flexible;
  add slice in [te (e), former tend (S)] with former type of S.
  add e to the new slice.
end

5 Testing and Evaluation

TODO

5.1 Experimental Setup

... should include the following:

- define experimental data and workload(s),
- discussion about the selection and interpretation of the evaluation metrics,
- discussion about the computing environment, including hardware, software, tools.

5.1.1 Experimental Design

5.2 Parameter Optimization

5.2.1 Interpretation of the Results

This sub-section should include

- description and an interpretation of the experimental results.
- explanation for any anomalies or any unexpected behavior.

6 Conclusion

6.1 Future Work

... should include the following:

- problem restated and a brief summary of the methodology,
- student contributions (e.g., survey, open-source software, journal publication),
- a brief summary of the findings and results,
- limitations and generalizability of the findings and results.
- lessons learned,
- recommendations for future research.

Bibliography

[1] Cao, J., Carminati, B., Ferrari, E., Tan, K.L.: Castle: A delay-constrained scheme for k-anonymizing data streams. pp. 1376–1378. IEEE (4 2008)

[2] Dijkstra, E.W.: A note on two problems in connexion with graphs. In: Edsger Wybe Dijkstra: His Life, Work, and Legacy, pp. 287–290 (2022)

[3] Federation, I.D.: Diabetes facts & figures (2023), `https://idf.org/about-diabetes/diabetes-facts-figures/`, accessed: 2023-10-01

[4] Hansen, S.L., Mukherjee, S.: A polynomial algorithm for optimal univariate microaggregation. IEEE Transactions on Knowledge and Data Engineering 15, 1043–1044 (7 2003)

[5] Kafka, A.: Organizations powered by apache kafka (2023), `https://kafka.apache.org/powered-by`, accessed: 2023-10-01

[6] LeFevre, K., DeWitt, D.J., Ramakrishnan, R.: Incognito: Efficient full-domain k-anonymity. In: Proceedings of the 2005 ACM SIGMOD international conference on Management of data. pp. 49–60. ACM (2005)

[7] LeFevre, K., DeWitt, D.J., Ramakrishnan, R.: Mondrian multidimensional k-anonymity. In: 22nd International Conference on Data Engineering (ICDE'06). pp. 25–25. IEEE (2006)

[8] Li, N., Li, T., Venkatasubramanian, S.: T-closeness: Privacy beyond k-anonymity and l-diversity. In: 2007 IEEE 23rd International Conference on Data Engineering. pp. 106–115. IEEE (2007)

[9] Machanavajjhala, A., Kifer, D., Gehrke, J., Venkitasubramaniam, M.: l-diversity: Privacy beyond k-anonymity. ACM Transactions on Knowledge Discovery from Data (TKDD) 1(1), 3 (2007)

[10] Mahdi Esmailoghli, Jorge-Arnulfo Quiané-Ruiz, Z.A.: Cocoa: Correlation coefficient-aware data augmentation (2021)

[11] Oganian, A., Domingo-Ferrer, J.: On the complexity of optimal microaggregation for statistical disclosure control. Statistical Journal of the United Nations Economic Commission for Europe 18(4), 345–353 (12 2001)

[12] Organization, W.H.: Diabetes fact sheet (2023), ttps://www.who.int/news-room/fact-sheets/detail/diabetes, accessed: 2023-10-01

[13] Sweeney, L.: Weaving technology and policy together to maintain confidentiality. In: Journal of Law, Medicine & Ethics. pp. 184–195. Wiley Online Library (1997)

[14] Sweeney, L.: K-anonymity: A model for protecting privacy. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 10(05), 557–570 (2002)

[15] Traub, J., Grulich, P.M., Cuéllar, A.R., Bress, S., Katsifodimos, A., Rabl, T., Markl, V.: Scotty: General and efficient open-source window aggregation for stream processing systems (2020), https://www.redaktion.tu-berlin.de/fileadmin/fg131/Publikation/Papers/Traub_TODS-21-Scotty_preprint.pdf

[16] Wallwork, A.: English for writing research papers, chap. Abstracts, pp. 177–245. Springer International Publishing Switzerland (2011)

[17] Wallwork, A.: English for writing research papers, p. 306. Springer International Publishing Switzerland (2011)

Appendix A. Further Details on the Solution Approach

Appendix B. Extended Version of the Experimental Results