FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

Anonymization for sensitive Linked Stream Data in Bio-Surveillance systems

MASTER'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Master of Science

in

Medical Informatics

by

Bernhard Ortner
Registration Number 0727931

to the Faculty of Informatics
at the Vienna University of Technology

Advisor: o.Univ.-Prof. Dipl.-Ing. Dr. A Min Tjoa
Assistance: Mag. Dr. Elmar Kiesling

Vienna, TT.MM.JJJJ

(Signature of Author)

(Signature of Advisor)

Erklärung zur Verfassung der Arbeit

Bernhard Ortner
Dietrichgasse 14B/15, 1030 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit - einschließlich Tabellen, Karten und Abbildungen -, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

(Ort, Datum)	(Unterschrift Verfasserin)

Abstract

Sharing health care data is a vital requirement in any modern health care system. Linked Stream Data (LSD) facilitates sharing by representing the data in an uniform way. Surveillance systems collect and analyze large amount of data to detect the spread of a disease. As a result of collecting massive data, such systems potentially unmask certain individuals, even in cases where the consumed data itself is properly anonymized. We address this problem by proposing a novel anonymization algorithm that ensures adaptive anonymity of data streams in two steps. First, it suppresses or generalizes individual-related data depending on the amount of available data. Then, our algorithm analyzes the long-term behavior of anonymized data and precludes inference attacks between multiple data snapshots. This minimizes the amount of lost information when data is anonymized. A result of the algorithm is that the incremental anonymization approach produces less information loss than proposed approaches in the literature.

Kurzfassung

Moderne Gesundheismonitoringsysteme erfordern die automatisierte Weitergabe von persönlichen Daten. Linked Stream Data (LSD) unterstützt diesen Vorgang, indem die Daten in ein einheitliches Format konvertiert werden. Surveillance Systeme sammeln und analysieren diese Daten, um die Verbreitung von Krankheiten zu dokumentieren und vorherzusagen. Eine Gefahr der Verarbeitungskette ist, dass einzelne Personen identifiziert werden können, sogar wenn die verarbeiteten Daten anonymisiert wurden. Um dieses Problem zu lösen, schlagen wir einen neuen Anonymisierungsalgorithmus vor, der LSD in zwei Stufen anonymisiert. Zuerst werden personenbezogene Daten auf Basis der verfügbaren Datenmenge generalisiert oder unterdrückt. Danach analysiert der Algorithmus den zeitlichen Verlauf der Daten und verhindert so Inferenzattacken zwischen zeitlich verschiedenen Daten. Das minimiert die Menge der Daten, die durch den Anonymisierungsprozess verloren geht. Ein Ergebnis unserer Arbeit ist auch, dass der inkrementelle Anonymisierungsansatz einen geringeren Informationsverlust als der derzeitige Stand der Technik aufweist.

Acronyms

LD	Linked Data
LSD	Linked Stream Data
IRI	International Resource Identifier
RDF	Resource Description Framework
ILM	Information Loss Metric
EHR	Electronic Health Record
QID	Quasi-identifier Attribute
SPARQL	SPARQL Protocol and RDF Query Language
BFS	Breadth First Search
SQL	Structured Query Language
CQ	Continuous Query
CQL	Continuous Query Language
DSMS	Data stream management system
EDBC	event driven backward chaining
CEP	Complex event processing
RDFS	RDF Schema
WHO	World Health Organization
KPI	Key Performance Indicator
UMLS	Unified Medical Language System
RODS	Real-time Outbreak and Disease Surveillance System

RLS	Recursive Least Square
WSARE	What's Strange About Recent Events
SRL	Simple rule language
MCUSUM	Multivariate cumulative sum
JS	JavaScript
CQELS	Continuous Query Evaluation over Linked Stream
LOD	Linked Open Data
IL	Information Loss
TDS	Top down specialization

List of Figures

Contents

CHAPTER 1

Introduction

Sharing health care data is a vital requirement of every modern health care system [76, 78]. A prerequisite of sharing data is collecting fragmented, distributed data in electronic health records [53, 132] and harmonizing and integrating it to represent it in an uniform way [86]. To solve these issues, the Semantic Web Health Care and Life Sciences group[1] advocates the publication of clinical and life science data on the Web in a machine processable format, so data services are able to link this data even it is hosted on various servers [16]. These services release data as „*structured data on the web and connected between different data sources, effectively allowing data in one data source to be linked to data in another data source*“ [16]. Hence the released data is called Linked Data (LD) [16]. Using LD can increase the accuracy of surveillance systems [49,57,106] by closing the gap between knowledge-level requirements and autonomous collection of data in deployed systems [42, 91]. Such systems are able to process large amounts of data by systematically collecting and analysing it to evaluate and implement public health practices [115]. They aim to detect trends, day-of-week effects and defined systematic behavior [21] to support health care decision makers or start an action to reduce the spread of diseases [42,49]. The day-of-week effect is a daily sickness value per week, „*which is defined as the number of people getting sick every day irrespective of whether they seek health care or not*“ [121]. The problem of privacy arises, when data is collected that strongly correlates with individuals data, such as day-of-week effects [16, 96]. It increases, when highly extensible data such as LD is used.

Our motivating use case is private health. In this domain, privacy has an important role due to the sensitive nature of the data [96]. Medical applications benefit from LD significantly by facilitating standardized access to data archives and frequently produce continuous data [110]. A conceptual view of a continuous medical LD system is presented in Figure 1.1. Individuals select sources of the data, e.g. wearable devices [12] or social media [31] that will be combined into a single stream. Data of the resulting stream is described in *who*, *where*, *when* and *what* dimensions [103] and analyzed by statistical analysis components.

[1] `http://www.w3.org/blog/hcls/` Accessed 29.4.2014

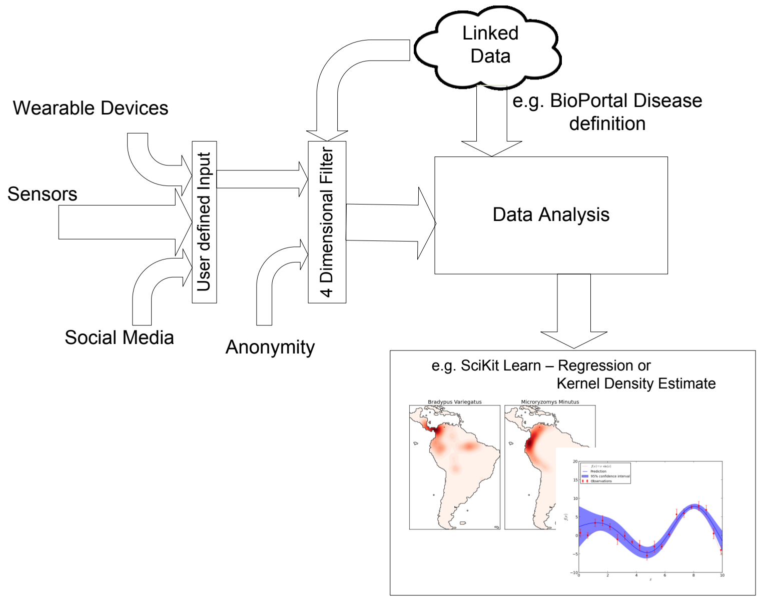


Figure 1.1: Continuous medical data processing components

1.1 Motivation

Government agencies and health care providers reduces costs by implementing and developing efficient policies. Hence the interest in individual health related data has increased, because the individual is one important trigger in the cost chain. Consequently such data is autonomously collected and analysed by surveillance system to detect cost saving potentials. If large amounts of data are processed, the individual is unmasked by combining data of various data sources [108] and therefore violates local privacy laws.
A community that benefits from this thesis is the scientific. Anonymization allows sharing and publishing scientific studies while fulfilling legal requirements. As a result, the process of searching and collecting scientific health data can be automated and will lead to new insights due to the massive amount of available data.
Furthermore, a major advantage for individuals is the improved quality of public health services, e.g., providing faster diagnosis and treatments. Additionally it is possible to improve the quality of an individual's life by predicting or inhibiting the spread of a disease. However, the individual has to trust the system to monitor his personal health status. Then the system provides hints for nutrition or advices for sports and the individual is able to adapt his life style accordingly [88]. To build a trustworthy system, the individual has to gain control over his data and adjust its level of anonymity to his specific needs [108].

1.2 Problem Description

Privacy is one of the major challenges of LD [16]. Currently developed solutions [69,101] have limitations when introducing privacy to their data schemas after its release. The data from different schemas *„can be used to re-identify by linking or matching the data to other data by looking at unique characteristics found in the released data“* [112]. Software has to guarantee that the „view“ on data returns anonymized data. To achieve this, in a single data source unique characteristics of every individual have to match at least k respondents [112,119] to avoid identifying combinations of several attributes through correlations or statistical evaluations.
The correlation problem of such data becomes even more challenging when real time data is used. Temporal data may unmask the individual through an additional dimension. The naïve approach of adding noise to data does not ensure privacy [60] and it results in noisy artifacts and the amount of usable information decreases [119]. As a result, a defined window function has to guarantee that different data sources with different privacy policies return valid data [5,112]. A window function is a function that returns data within a defined timespan [93].
A further problem of LD is that a particular query may return varying result sets, e.g. when new data is processed by a stream. This non deterministic property of LD allows to collect data and potentially unmasks individuals. To ensure privacy, each query has to return the same result set, or the result set has to be checked for constraint violations in a maximum delay of the output data [23].

1.3 Research Questions

Three constraints conclude the issues of ensuring privacy of LD [23].

- *Anonymization*: Each individual related data is anoymized and indistinguishable from other individuals [84]
- *Time*: Data has to be released after a defined timespan otherwise data is invalid or leads to invalid assumptions.
- *Utility*: Utility is maximized, depending on target application and data [84].

An opponent of anonymitzation is utility. It is defined as amount of data used for post processing and the number of records that satisfy a certain property [18,95].

Research Question 1. *How is it possible to anonymize health related Linked Stream Data considering time and utility constraints ?*

The objective of the thesis balances between anonymization and utility with a higher priority on anonymization. When the issues are balanced, can anonymization be ensured when it is scaled to multiple data sources? Consequently the second research question is formulated as:

Research Question 2. *Does redistributing of various data sources violate the constraints from research question 1 ?*

Based on the first research question, the thesis is organized into two parts. The first two chapters discuss background and related work and the third chapter describes the prototype. A conclusion and further outlook is provided at the end of the thesis.

1. **Related Work**
   A comprehensive literature research provides an overview over the state of the art. The research includes an evaluation of the servers and applied metrics to evaluate their capabilities.

2. **Implementation and Reviewing**
   A new approach to ensure privacy for LD is developed. The approach will be implemented to verify the validity of the theory. This part includes standardization of data for further analysis. Metrics that compare the original data and the anonymized data, like the Information Loss Metric (ILM) will be used.

3. **Conclusion and Demonstration**
   A generic stream data generator is used to demonstrate a „Proof of Concept " prototype. The demonstration will be kept as simple as possible, but it will include statistical tools to analyze the behavior of the system.

Excluded from the scope of this thesis is the development of a new measurement metric, developing or altering any query engines. Furthermore, developing statistical analysis methods is not within the scope.

1.4 Definition of Terms

We provide a definition of relevant general terms in the context of the thesis.
The meaning of the term „data" remains unchanged. Hence plain un-anonymized data is notated as table T, anonymized data as T' and redistributed anonymized data as T''. A tuple is referred as an entry of a set of data containing a relationship between one to several other data entries. Additionally, the term „query" refers to a request of data expressed in a structured language such as Resource Description Framework (RDF).
Without loss of generality, basic assumption is that data is fragmented over a heterogeneous IT landscape such as different servers and archives. This is a valid assumption, because current medical data is stored in medical records, representing a medication or a diagnosis [107]. Furthermore it is assumed each fragment contains n attributes, that potentially unmasks individuals [4].
An adversary is defined as user, that infers or unmasks sensitive data about an individual from any anonymized data by using linking methods, like correlation or linking two different data sets together.

1.4.1 Anonymization and Privacy

Privacy in computer science is intuitively described as *data that you keep entirely to yourself, or to a limited group of people* [41]. Privacy ensures limited access to data for a group or an

individual.
On the other hand, anonymization is the process of destroying tracks or electronic trails on the data that would lead an eavesdropper to its origins [58]. Anonymization alters or suppresses data, but anyone is able to consume anonymized data. As a result, if data is correctly anonymized a data consumer *cannot infer personal data that is processed in the anonymized data* [112].

1.4.2 Linked Data Stream

Linked Data is a set of best practices for consuming, publishing and connecting structured data on the web [16]. It relies on documents containing data in RDF format. The RDF format defines a (web) resource I as a combination of an International Resource Identifier (IRI) and data [61], blank nodes B as resources that do not have a natural IRI [74], and RDF literals L as resources that contain free text [61]. Assume there are pairwise disjoint infinite sets I, B, and L. An RDF triple is a First Order Predicate $\langle s, p, o \rangle$, such that $(s, p, o) \in (I \cup B) \times I \times (I \cup B \cup L)$ holds [16, 94].
LD is associated with a temporal dimension, it is called Linked Stream Data (LSD). Formally, when an *„RDF stream S is a bag of elements $\langle (s, p, o) : [t] \rangle$ where (s,p,o) is an RDF triple and t is a timestamp. This stream elements of LSD are represented as RDF triples with temporal annotations“* [64]. The goal of LSD is to publish stream data according to the LD principles [64, 105]. LSD streams facilitates an easy data integration and standardization process among potentially heterogeneous data collections [64].

1.4.3 Bio-Surveillance System

Bio-surveillance is defined as *„the ongoing, systematic, collection, analysis and interpretation of health data essential to the planning, implementation, and evaluation of public health practice, closely integrated with the timely dissemination of these data to those who need to know “* [47]. The definition characterizes surveillance systems with three basic properties: [47, 114]

1. The system has a clear purpose,
2. it uses standard definitions, data fields and protocols to process information, and
3. it defines domain expert organizations that are responsible for standardizing and quality assurance task of the system. Furthermore they have to interpret and disseminate received data of the system.

Compared to a decision support system, their purpose is to provide appropriate information for recommendation [81] whereas surveillance systems’ purpose is to preprocess data.

CHAPTER 2

Background

Ongoing privacy research focuses on models, algorithms or frameworks that restrict access when individual's data is released [9]. Under those circumstances, restriction skipped partial information, and established rudimentary anonymity. Currently available privacy mechanisms are dividied into three categories [95]

1. *local perturbation*: individuals trust no one except themselves and they restrict access to their data themself before they upload it to a server

2. *data publishing*: individuals trust themselves and a third party, e.g., the upload server. Third party software applies the restriction on the server.

3. *output perturbation*: plain data of individuals is kept on a trusted server and after a request to the server the output of the request is restricted accordingly.

All privacy mechanisms implicitly assume that the dataset fits into main memory. As a result, those approaches do not scale with vast amount of data that is stored in web archives such as Electronic Health Record (EHR) or web archives. Furthermore, privacy models may reveal inferences about individuals by using certain data mining techniques that exploit structural properties of the data. To preclude such inferencing, privacy alogrithms have to detect a-priori structural sensitive knowledge patterns from data before releasing it [45, 50].
Hence the collected data limits the type of analysis and the knowledge processing mechanisms. If the parts of the data are changed or released, it has to be ensured that data is still valuable to scientists [9, 45]. Taxonomy trees address this problem by using cut metrics. A cut metric specifies a plane for pruning underlying data structure (in Fig. 2.1 the dashed line). The tree is built from composition pattern that models attribute hierarchies. If an attribute is generalized it uses its generalized value instead of more concrete data value [15]. An example of a taxonomy tree with its cut metric is shown in Fig. 2.1.

A frequently used type of cut metrices is based on minimal generalization. Minimal generalization anonymity describes the minimum amount of effort to ensure a proper level of anonymization by suppressing concrete values. It terminates if all predefined privacy constraints, such as k-anonymization constraints, are fulfilled [102].

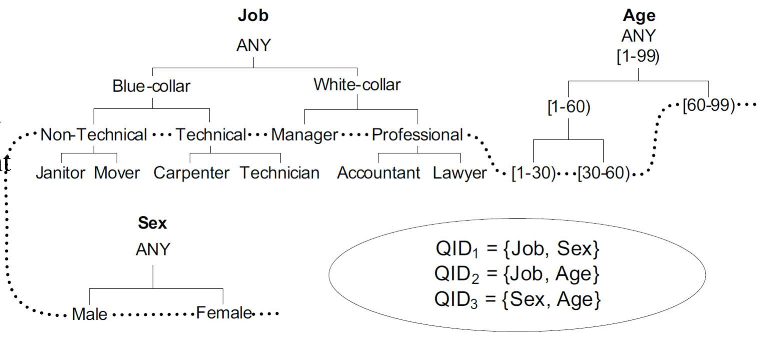


Figure 2.1: Taxonomy tree example

From a data dissemination point of view three opposing objectives have to be adressed. First, the objective of anonymization has to be solved. If highly extensible data, e.g. LD is used, current used approaches have to be reconsidered, because they are designed for static data. Extensible data is dynamically expandable by adding new links to the source LD data set [16]. A two-level solution is able to solve the problem.
Anonymization has to ensure that only proper anonymized end user data is uploaded to a dedicated server. If data is uploaded a meta refinement [45] checks window query data to identify and remove all potential individual unmasking tuples. The window is evaluated on the „query-time“.

Second, utility is considered. If key characteristics of data are altered, removed or suppressed, data is modified and has a diminished value for scientists.
Third, we balance time and space complexity of the used algorithms. Newer approaches replaced exhaustive search with alternative algorithms such as greedy search to reduce time consumption. Furthermore a major impact on the runtime have trends and events detecting operations [120].

Consequently, these problems have to be balanced, but they cannot be achieved at the same time, e.g. if no data is released, perfect privacy is achieved, but there is no utility and the algorithm has a small runtime behaviour.
On the other hand for archieving a high utility, the release of specific data is required by a consuming third party. As a consequence, anonymization potentially leaks identities of an individual and the time consumption of the selection algorithm.

Transforming data to achieve full anonymization, optimal utility and performance is computationally infeasible. Early balancing approaches were focused on anonymization, whereas modern focuses on optimizing the runtime behaviour. The third parameter, utility is often measured with an information loss metric, but not directly addressed.
A novel contribution of this thesis

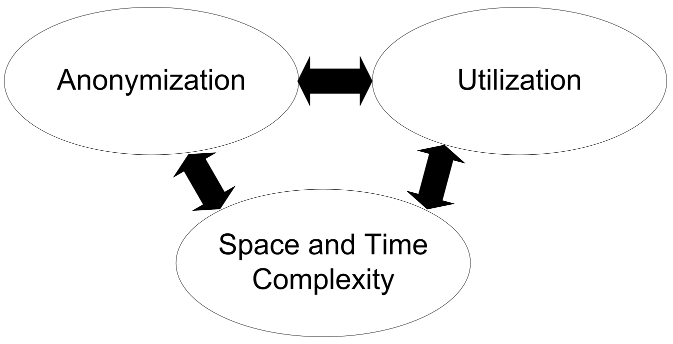


Figure 2.2: Conflicting objectives in anonymization

is balancing anonymity and utility. Furthermore performance is achieved through various approaches, like horizontal and vertical scaling, divide and conquer approaches or choosing an appropriate meta anonymization metric.
The goal of this chapter is to provide introductory knowlegde for relevant topics. First, we explain disclosure mechanisms facilitating an analyse of various anonymity approaches. Afterwards, a rudimentary description of streaming environments is provided. The section ends with basic approaches for surveillance systems.

2.1 Anonymity

This section describes background knowledge of disclosure mechanisms with the goal of providing insights of successful de-anonymization attacks.

2.1.1 Disclosure Approaches

Current access control approaches safeguards against direct disclosures, but they do not address indirect disclosures based on released data. Sweeney described disclosure as the process of releasing information derived by inference whose release was not intented [111]. Machanavajjhala *et al.* define two types of disclosure [73]:

1. Positive disclosure
   A person is able to correctly identify the value of a data fragment. In an information retrieval context, positive disclosure increases the recall ratio of data [1].
2. Negative disclosure
   Negative disclosure eliminates some possible data combinations of anonymized data. This is a major attack against anonymization.

An ideal alogrithm achieves positive and precludes negative disclosure [73]. Two different negative disclosure mechanisms are distinguished [69]:

- Membership or identity disclosure
  „*Membership or identity disclosure occurs if an individual is linked to a particular record in a released table* " [69]. When identity disclosure occurs individuals are re-identified and corresponding sensitive values are disclosed.
- (Sensitive) attribute disclosure
  Attribute disclosure occurs if new information about an individual is infered or linked, i.e. released data makes it possible to derive new knowledge about a certain characteristic of an individual.

Attribute disclosure occurs independently of membership disclosure, whereas membership disclosure always implies attribute disclosure. Before we describe possible solutions for disclosing in detail, Figure 2.3 shows an overview of current anonymization frameworks. Each framework sketched in the figure is explained in the following.

[1]retrieved relevant documents in relation to a search query - see [20]

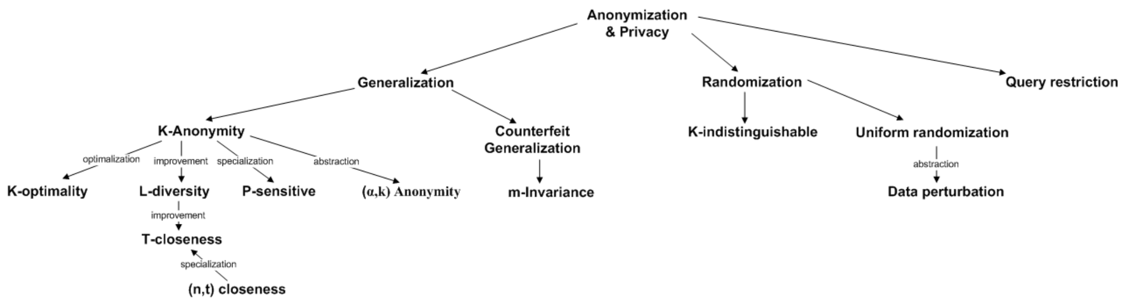


Figure 2.3: Outline of algorithmic anonymization

2.1.1.1 Query Disclosure

The combination of queries potentially leaks sensitive information. For example, if a table view [2] contains a physician to patient relation and another view contains what medication each physician has prescribed, then the combination of both is critical, because medication highly correlates with diseases [111].

2.1.2 Query Restriction

A solution for query disclosure is query restriction. It reduces the result set size of queries and controls the overlap amongst formerly used queries. The result sets of two queries are called to be separate, if the algorithm „*keeps an audit trail of queries and constantly checks for possible compromise, suppression of data cells of small size, and clustering entities into mutually exclusive atomic populations* “ [40].
Another approach to deny executing the query [129] and effectively suppresses all inferences to data [111]. A major disadvantage of this approach is that the space consumption of the audit trail increases monotonically, and that the cost of judging answerability increases exponentially [129].

2.1.3 Perturbation-based Approaches

Perturbation based approaches add noise to data depending on its evaluation of the query. This process is known as output or data perturbation. It is distinguished into two categories.
First, the probability-distribution category explores the probabilistic properties of a data set and swaps or replaces similar values of data records by a generic value that fits into the same distribution of the original data. In addition, it adds noise to the data and samples, i.e. it generates interpolation values in the resulting data set [60].
The second category is data perturbation with a fixed-data size. The perturbated attributes are selected randomly until the amount of selected attributes fits the fixed-data size. Then data is handled like in the probablity approach. This approach does not fit in the case of LD because it is limited to numerical and categorial attributes and it scales worse than the probability approach [2].

[2]A table view is a virtual table containing the result of a query

Evfimievski *et al* describe a generalized approach of data perturbation called uniform randomization. It replaces each data record in a server session with probability p by a new data record that is not originally present in a current session [40].
A major disadvantage of perturbation-based approaches is that such techniques cannot satisfy conflicting objects, like providing high quality statistics and preventing exact or partial disclosure [40].
Furthermore, Agrawal and Aggarwal showed, that an expectation-maximum algorithm restores the original distribution of data. This algorithm calculates iteratively the maximum-likelihood estimation of a selected random data sample [6]. Furthermore there are queries that cannot be executed [129] e.g. if a dataset has less data than the query will return [89].
An approach similar to data perturbation is differential or k-indistinguishable privacy. It describes an operation, e.g. adding random noise, that is applied on a certain data set to ensure privacy. If a data attribute differs in $exp(k)$ tuples from another one, differential privacy is guaranteed because the output data set is indistinguishable. The exponent is a user-set property that represents the degree of privacy. A drawback is that balancing differential privacy is difficult. On the one hand, if insufficient noise is added, data is not properly protected. On the other hand, if too much noise is added, the data gets useless [39].

2.1.4 Data modification

A different technique of ensuring privacy are data modification approaches. They provide an adequately anonymized table T' such that no privacy leak occurs. Each query is executed on table T' and returns a precise result.

2.2 Streaming Environments

When a data structure consumes continuous pieces of static data in increasing order of their temporal dimension [54, 105] the received data structure is called a data stream. The temporal dimension appears explicitly or implicitly annotated with a timestamp [64]. Such streams have a large volume and its storage is either unnecessary or impractical because of the limitations of memory. Hence they are intended to be read only once or a small number of times [52, 54, 87].
First we define Linked Stream Data (LSD). Then we describe processing mechanisms for LSD.

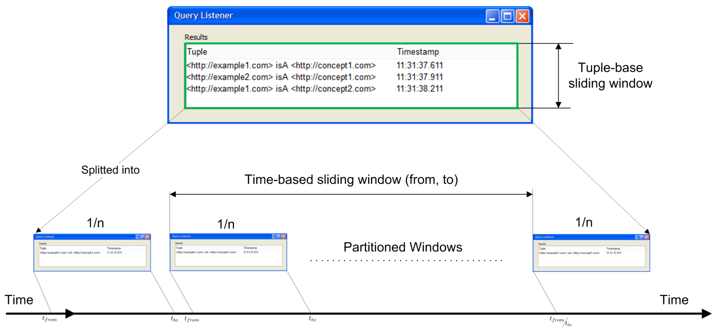
2.2.1 Linked Stream Data

RDF triples with temporal annotations form a LSD by sorting these triples after their timestamp [64]. Due to the lack of standardization, [3] there are currently two possibilities to model the time dimension of a quadruple. First, an interval models the time component and contain two date endpoints (t_{from}, t_{to}) and the quadruple is valid in the given interval.
Another approach is to model the time component as timestamp when the quadruple was recorded or received. As a result, a consumer continuously lists on new events.

[3] Current progress see `http://www.w3.org/community/rsp/`

2.2.2 Linked Stream Data processing

The result of processing LSD is a query language for continuously running a specified query. Terry *et al* defined the result of these CQs as „*set of data that would be returned if the query where executed at every instant*“ [113]. They extended the query semantic with RDF syntax and temporal semantics to cope with LSD [11, 64]. A result of ongoing research is that the output of RDF CQs are processed differently based on the chosen window approach of each LSD engine. A query window is a query associated with a time-interval function (t_{from}, t_{to}), such that a non empty set of data is evaluated. Figure 2.4 shows frequently used window approaches and distinguish between three categories.



Tuple	Timestamp
<http://example1.com> isA <http://concept1.com>	11:31:37.611
<http://example2.com> isA <http://concept1.com>	11:31:37.911
<http://example1.com> isA <http://concept2.com>	11:31:38.211

Figure 2.4: Outline of CQ windows

The CQs described by Fig. 2.4 represent the categories of list 2.2.2 [11, 52]:

1. Time-based sliding windows
   There are three approaches for processing moving sliding windows. Landmark window queries extend the query from a fixed point t in the stream to the last available tuple, i.e. it generates an aggregated result of all consumed data. A disadvantage is that the windows keep all data since t present. Gehrke *et al.* proposed that the windows are generated as pair of (count, average) from received data and updated when new data is received [48].
   Sliding window queries have two sliding timestamps t_{from} and t_{to}. When a new tuple is consumed both ends of the window are adjusted accordingly to the new tuple.
   If both of the endpoints (t_{from}, t_{to}) are fixed, the window function is called Snapshot. We do not apply snapshot windows within the thesis [48].

2. Tuple-based windows
   A tuple-based window, also called count-based or physical windows, consumes the last n tuples of an input stream.
   Another approach are logical windows and they are defined by the number of n tuples of a time interval. If less than n tuples are incoming they are extrapolated to match exactly amount of specified output tuples [11, 14].

3. Partitioned windows
   Partitioned windows split the query window into smaller sub windows and execute each of them separately. If an update of a corresponding sub window is detected, e.g. new

tuples are streamed or when a certain interval expires, the sub-window is re-evaluated by using one of the mechanisms:
The first mechanism called eager updating computes the result of the sub window when new data is available.
The second mechanism is called lazy or batch update and is computed when window results are required. When a sub window is finished it is merged with other sub windows to calculate the output of the complete window [11].

Above listed window types use window function or allow continuously monitoring of a stream engines. They achieve this goal by using functional operators that are similar to Structured Query Language (SQL) and extended by slide, tumble, resample and map operators to capture an incoming set of tuples of the stream and re-evalutate its query again. Slide defines a sliding window like tuple-based or time-based sliding window approach. Tumble is the implementation of partitioned windows and resample produces a synthetic stream by interpolating tuples of input stream. Lastly, map uses given input functions and applies them to every tuple in a stream.
Window and monitoring queries differ in the overlap the tuples of its result-set. When monitoring queries are evaluated, the stream engine deletes all data of its input and it returns the results on demand to the requesting instance [26]. Window queries, on the other hand, buffer those data and evaluate the window query. If the query is re-executed, corresponding buffers are updated and re-evaluated for window queries [68]. The amount of overlapping tuples in the window is controlled by an input parameter, i.e., setting the window size larger than the resampling time.
Continuous queries have two problems. If an input window has a large interval, e.g., an aggregation per year, all values of each input stream for a given time interval are buffered. Second, processing input tuples multiple times results in high computational costs.
The two issues are addressed by generating panes. A pane is based on tumble and an intermediate non overlapping query result that is shared between each executing window and aiming to reduce computational costs. A requirement for a pane is that the input data is dense. When a pane has less data, i.e., the window size is restricted with „group by “ to only one tuple, the pane does not reduce the number of input tuples, and therefore it does not have any benefits [10].
Whereas single stream processing mainly works with above listed approaches, queries over multiple streams are processed in different ways. Their different input models are used to generate a coherent result set. The native approach of harmonizing multiple data streams is to execute the query on each stream separately and to join all single resulting sets together by using a generalized symmetric join.

2.3 Surveillance Systems

The World Health Organization (WHO) defines a public health surveillance system as the „*continuous, systematic collection, analysis and interpretation of health-related data needed for the planning, implementation, and evaluation of public health practice*.“[4] It „*allows stakeholders to estimate the magnitude of public health problems, describe the natural history of a disease, determine the distribution and spread of illness, detect outbreaks, stimulate research, evaluate public health*

[4] `http://www.who.int/topics/public_health_surveillance/en/` Accessed 13.10.2014

practice, monitor changes in disease agents, detect changes in health practices, and facilitate planning" [109]. Two main purposes have to be fulfilled to ensure these goals: situation awareness and early event detection. Situation awareness is the ability to efficiently collect, store and process (real-time) data to respond to an action, e.g. an outbreak of an disease. Early event detection finds events at the earliest possible time to start a predefined action, e.g., to limit spread of a disease [44].
Due to their complexity events, such as identifying early indicators for a spread of a disease, are defined by domain experts due to their complexity [114].

2.3.1 Analytic design of surveillance systems

A surveillance system evaluates user defined Key Performance Indicators (KPIs). A KPI provides information on the current state of a predefined phenomenom, e.g., incidence ratio of a disease [124]. A domain expert decides which phenomena are measured and which parameters are tracked [75]. Based on the selection, corresponding systems address one or more of the following purposes: (a) capture an early case of disease, (b) find a pattern of disease in the context of a wide exposure, (c) track a proxy syndrome during an outbreak to measure the geographic and temporal spread of a disease [109]. An improvement of detecting health related phenomenon of various data sources is deterministic linkage. This technique links two data sets based on a join attribute. There are three approaches to find candidate for such join attributes.

1. Simple deterministic
   This method relies on the availability of a unique personal identifier that is used as join attribute.

2. Sequential deterministic
   In this setting, linked variables are placed in a hierarchy of decreasing or increasing value. Variable pairs are found by using a heuristic of the close related term of this hierarchy.

3. Probablistic
   Probablistic linkage is used when multi dimensional data is processed. It calculates a score for each possible matching or mismatching pair that represents the likelihood estimator of combination. The pair with the highest matching and lowest mismatching is used for matching two data sets.

A problem of matching is the validation of matched links, because matching multiple attributes is a multi-level process requires to fulfill several constraints and depends on the chosen approach for link joining [57].
Hence, deterministic linkage is a supporting task in the data collection process to improve the evidence of a surveillance system.

2.3.1.1 Surveillance System Architecture

Each surveillance system has a manager component to facitlitate the exploration and the functional investigation of anomalies to reduce false positive values and get a deeper insight into the data [75, 130]. The manager component consumes data produced by previous steps, such as data collection and data management, and detects abnormal patterns predict an arising trend of outliers that indicates a spread of a disease. A spread is detected, if a cluster of undiagnosed disease attribute data is detected [130]. The main components dividies the surveillance system into data collection and standardization module and analyse components. Figure 2.5 gives a bief overview of them, and how they are related.

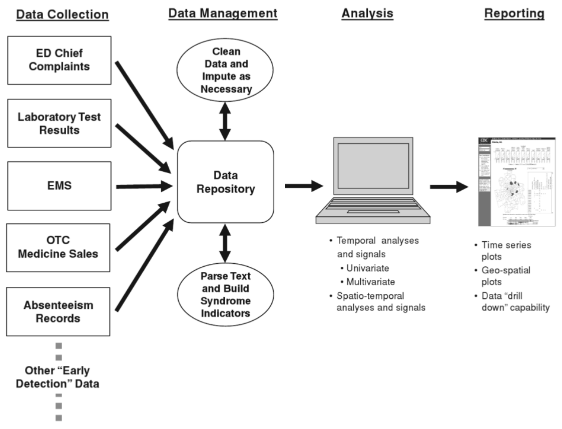


Figure 2.5: Architecture of Surveillance System [44]

First, data collection and standardization module separates into data collection and data management [19]. It processes related realtime data after an appropriate check, i.e., if data consumption is allowed with respect to legal or privacy terms. When a system is allowed to collect data, it uses an array of aberration-detection methods to identify increasing patterns above a pre-determined threshold [56, 75]. The data management component contains tasks that provide a standardized view on collected data. If data is accessed from a sub-system, e.g. from laboratory data or patient data archives, each archive used frequently provides its own data encoding and has to be standardized and cleaned [44]. Heterogeneously encoded data is integrated and standardized by using converters and transformations, such as Unified Medical Language System (UMLS). UMLS provides a cross reference among different coding and semantic structures for defining relations among clinical entities [75].
Second, the analysis component facilitates an interactive visualization of collected data, e.g., on a map, as timeseries chart or as table and combination of them. Surveillance system visualize data on a map based geographic entities and coloring them based on the incidence ratio.

2.3.2 Bio Surveillance

Fricker [44] defines bio surveillance as an abstract term for a surveillance system that measures biological phenomenon. Such systems are separated by their aim into human or epidemiological, agricultural and animal surveillance. A human surveillance system measures reactions to medical interventions and how such decisions influence a prevalence of a disease in a population. Furthermore a human surveillance system tracks medical interventions, environmental changes, medical diagnoses and results from labratory test. Figure 2.6 shows a partition of bio surveillance systems, into sub parts.

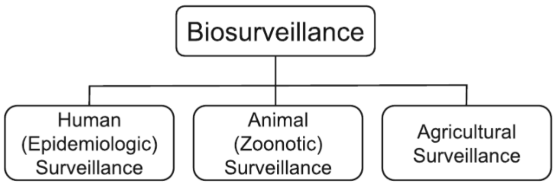

Biosurveillance
Human (Epidemiologic) Surveillance
Animal (Zoonotic) Surveillance
Agricultural Surveillance

Figure 2.6: Types of Bio Surveillance [44]

In the literature, syndromic surveillance as part of a human surveillance is a well studied system. This type of surveillance system is designed to detect diseases and outbreaks at an early stage before public health authorities notice them. Syndromic surveillance achieves this goal by using less specific real-time data and health-related data than human surveillance. [75],e.g. reports based on suspected or confirmed disease cases [37, 44]. On the other hand syndromic surveillance will be built upon highly specific KPIs designed for uncertain data.

2.3.3 Human Surveillance System

In this section several different surveillance systems are described. Each system implements the component differently. All listed system are licensed under an open source license that allow the redistribution and modification of the system. Out of scope from the evaluation are properitary systems, such as HealthMap[5], BioSense[6] and EpiSpider[7] due to licensing issues.

2.3.3.1 Realtime Outbreak and Disease Surveillance

The Real-time Outbreak and Disease Surveillance System (RODS)[8] uses as input data chief complaints from emergency detection data and over the counter sales. This data is categorized with a bayesian classifier into predefined categories [32]. Abnormalities are detected by using different mechanisms based on the classified data. A frequently used model is linear regression, using, for instance Recursive Least Square (RLS). RLS is an iterative approach that computes expected counts for each category as well as combined counts for each geographic entities and minimizes each expected count function per iteration.
Another classifier for RODS data is What's Strange About Recent Events (WSARE). WSARE performs an exhaustive heuristic search over combinations of temporal and spatial features to detect anomalities among the density of them. If the amount of a combination is higher than the average WSARE reports an infectious disease. The outcome of both algorithms are merged togeher and displayed in user preferred views [118]. An overview of the RODS architecture is shown in Figure 2.7.

[5]Available from `http://healthmap.org/`

[6]Available from `https://biosen.se/`

[7]Available from `http://www.epispider.org/`

[8]Open Source Version can be downloaded at `http://openrods.sourceforge.net/index.php`, Accessed 25.10.2014

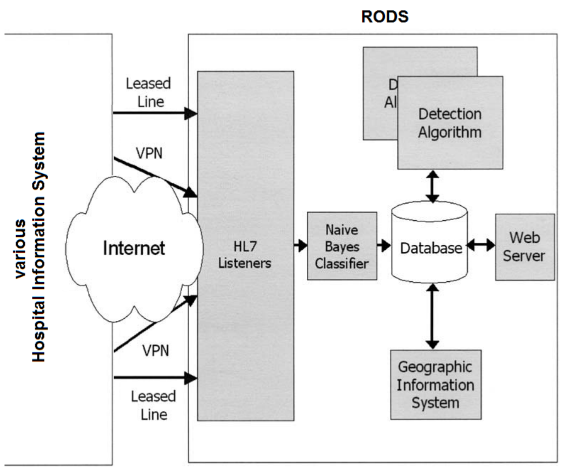


Figure 2.7: architecture of RODS components [118]

2.3.3.2 BioCaster

BioCaster [9] is an ontology-based text mining system that collects information from various news sites, like European Media Monitor Alerts (EMMA), Google, Reuters and WHO. It analyses RSS feeds from those sites by classifying and visualizing them on a map. The data management component is extended by Simple rule language (SRL) that extracts and adds additional levels of semantics to understand and disambiguate geo-temporal and linguistic terms. After this step, BioCaster maps extracted data by SRL to the BioCaster ontology [10] to avoid gaps, e.g. between a mentioned pathogen and its unmentioned disease or between occured symptoms and the disease itself [30].

2.3.3.3 BioStorm

BioStorm [11] is an ontology-based surveillance system for describing characteristics, types and relationships of data collected from various data sources, e.g. emergency departments. It consists of three types of ontologies each of them is used by system component of BioStorm.

- Data-source ontology
  Data-source ontologies correspond to customized local model of each data source. BioStorm facilitates to use any uniform solvers on consumed data sources and maps their data in real-time on the ontology. The broker's main purpose is to abstract incoming data to provide a „rdfized", heterogeneous view on integrated data. It prepares sparse distributed incoming data in a dense distribution to avoid difficulties in further data processing [92].

- Mapping ontology
  The mapping ontology transforms and maps RDF data results from the previous step into

[9] Available from `http://born.nii.ac.jp/`, Accessed 26.10.2014
[10] Available from `https://code.google.com/p/biocaster-ontology/`, Accessed: 28.10.2014
[11] Available from `http://biostorm.stanford.edu/`, Accessed: 28.10.2014

a new ontology by using „sameAs“ predicates, and runs surveillance methods on the data. These transformation includes renaming, summarization and aggregation for generating a data structure that problem solvers are able to process.

- surveillance problem-solving ontology
  This ontology models results of an evaluation of each problem solving instance. A problem solving instance is a component that conducts a surveillance task, i.e. analyzes occurences of a defined phenomenon. An RDF individual is a concrete unique instance of the model [90]. Individuals of surveillance problem solving ontology are visualized in a GUI as selectable elements.

The BioStorm controller connects outputs of the components with inputs of the next processing step. Its main purpose is to facilitate load balancing between data components and analytic components and provide a seamless integration of the modular architecture of BioStorm [34].

2.3.3.4 Limitations of Surveillance Systems

A surveillance system is useful if it generates a public health response or prevents and identifies health related problems. The criterion is evaluated by domain experts. A methodical approach is to define assessments on the impact of surveillance data on policies and interventions and lastly on the occurrence of a related health event. For re-evaluating the assessments quantitative attributes of the assessments are defined as numerical measures that run after a given time period. The evaluation of numerical measures adjust and adopt the KPIs to changed illness pattern and new priority emerged diseases [116]. If illness pattern cannot be ajusted automatically, the system has to provide appropriate feedback to its decision makers [37].
Another major problem, especially in the health domain, is the geo-encoding of an address. Current approaches are based on simple heuristics and are open to errors [27]. Furthermore, an impact on geo-code detection has scaling and zoning effect on areas of interest. Zoning is the process of creating meaningful units for an area and scaling is the process of partition or summarizing areas of interest, e.g. based on their ZIP codes [33, 131].
As a consequence, resulting addresses from patients are not necessary a good indicators for the spatial spread of a disease. A frequently used approach is to resolve the geo-location by mapping it to extensional semantics of its reference, such as the geographical ontology [75].

2.3.3.5 Data streams and Privacy

Surveillance systems often collect multiple streams of data to evaluate and monitor them for indicative disease patterns [109]. Detecting multiple streams improves sensitivity, specificity and timeliness of detection over more frequent univariate alternatives [100]. A data stream has unique attributes that are mapped and understood to generate a uniform data stream. The technique for combining multiple data streams is called data fusion. The idea is that multiple data streams are merged into a single stream.
A single stream from data fusion is analyzed with the extension of multivariate statistical techniques such as Multivariate cumulative sum (MCUSUM) based approaches to increase detection

sensitivity while limiting the number of false alarms [130]. MCUSUM method is suitable for detecting gradually increasing abnormalities within single temporal data.
Such individual abnormalities are used as QID, i.e. they provide additional dimensions such as a spatial and temporal. In that case the surveillance system maintains anonymity of individuals. Two possible solutions exists for ensuring anonymity.
First data is anonymized as it is described in Section 3.1. This process is applied before data is sent to the surveillance system. If data cannot be anonymized, its usage is restricted. Restricting data is frequently used, because a surveillance system has to effectively use data with low specificity.
If multistream data applications anonymize data, the problem of duplicate event detection arises. A single event of illness potentially shows up in several independent data streams, e.g. in a laboratory test and a chief complaint. Without a way to detect such events, duplicate events tend to increase the amount of health alerts. To avoid such events, distinct streaming data sources are chosen or the identity of an individual is replaced by a pseudo unique identifier [72].
A further problem of data fusion is that the error rate of detecting algorithm increases linearly with the amount of monitored data streams. This issue is adressed by threshold-rates for detecting an event is replaced by a bound including the amount of monitored streams. Consequently applied statistical detection methods „*may result in a severe loss of sensitivity, especially if data streams are correlated*“ [99].

CHAPTER 3

Related Work

Based on the background provided in Chapter 2, we describe closely related terms and techniques of the thesis. At the end of this chapter we provide a conlusion, summarizing keypoints and issues of literature.

3.1 Anonymity

Anonymization based approaches generalize or suppress data and depend on the distribution of its data.

3.1.1 Anonymity based Approaches

Anonymity-based approaches describe the k occurence of a record in the anonymized data set, therefore they are called k anonymity models. Generalization hierarchies and suppression techniques achieve this goal by hiding exact values of attributes of a data set. They anonymize numeric attributes, e.g. ZIP codes by replacing the last n digits with a $*$. When anonymization generalize continuous attributes, such as timestamps, they replace the concreate value by its corresponding range.
k-anonymity can always guaranteed, if the hierarchy of generalization is deep enough [4]. In the anonymization model, additional to identifying key attributes Quasi-identifier Attributes exists. A QID unmasks individuals by identifing individuals indirectly by the exclusion principle. Sweeney defines k-anonymity as:

Definition 1 (k-anonymity)**.** *Let RT*$(A_1, \ldots, A_n)$ *be a table and* QI_{RT} *be the quasi-identifier associated with it. RT is said to satisfy* k*-anonymity if and only if each sequence of values in RT[*QI_{RT}*] appears with at least* k *occurences in RT[*QI_{RT}*].*

Two categories of anonymization algorithms exist for anonymizing values of each QID. The first type, called global recording, maps QIDs to generalized or altered values, e.g. a concrete birth

date is replaced by the month of birth and its year. Local recording maps individual instance of data to a generalized or suppressed value, e.g. the day of month is replaced by a $*$.
As an illustration of these steps, we use a physician patient as an example. A valid QID for patient is the combination of $\{Sex, Zip, birthdate\}$. This tuple allows re-identification of individuals, e.g. that a patient has HIV. If patient data is correlated against publicly available data, e.g. an election register, the identities of patients become unmasked.
Every k anonymization based approach assumes that only a small number of attributes are used as QID. As a result, the k-anonymity model requires that every QID has a frequency at least of k or zero of the resulting anonymized table T'. Table 3.1 displays an example of k-anonymized patient data with $k = 2$. Each QID occurs at least two times.

Disease	Birth-Date	Sex	ZipCode
HIV	01-01-1987	Male	537**
Flu	01-01-1987	Female	537**
Heart Disease	01-01-1987	Female	537**
Broken Arm	01-01-1987	Male	537**
Brochitis	01-01-1987	Female	537**

Table 3.1: Example of anonymized patient records [66]

An improvement of k-anonymity is (α, k) anonymity. It introduces another attributes to k-anonymity that enforces the appearance of k attributes at least α times [125], i.e., each tuple has to occur between $0 \leq \alpha \leq k$ times in the anonymized table.
A disadvantage of k-anonymity is the high dependence upon dimensional structures. It anonymizes high dimensional structures, if its anonymization function ϕ is injective, i.e. assigns more than one value for each value of a record in table T [83]. Even if data is properly anonymized, the underlying algorithm tries to retain partial information about every data dimension [4]. As a consequence generalized nodes from the hierarchy describe anonymized content in an abstract way. Therefore this anonymization approach is more open to inference attacks. Another dimension that causes a problem is the time dimension. When the algorithm collects temporal data, it adds, changes or removes records constantly.
As a result, releases of anonymized data can be used for a temporal inference attack. In this scenario two data sets s_0 and s_t have the same data basis. s_0 represents an anonymized data set with time $t = 0$, and s_1 at time t. When an adversary links both data sets together, k-anonymity is compromised by infering new knowledge of changed records [112].

If more than one table is anonymized, the upper bound on execution time increases exponentially by the amount of consumed tables with $\mathcal{O}(2^n)$. A reason for exponential execution time is exhaustive search [66]. As alternative to the exhaustive search, different greedy algorithms, like Breadth First Search (BFS) can be used[1]. As a result the execution time decreased to $\mathcal{O}(nlog(n))$ [65].

[1]There were also other approaches, see [66]

In addition to the temporal problem k-anonymity has two major problems. An attacker discovers the values of sensitive attributes when there is less diversity in its data. More precisely, „*k-anonymity can create groups which leak information due to lack of diversity in the sensing attribute* " [73]. Secondly, attackers have often background knowledge, and the k-anonymity approach does not guarantee anonymity against attackers with background knowledge. An example for a background knowledge is a further table that contains additional information about a group of individual or knowledge about the density of the data.

Sensitive attribute attacks are prevented by an enhancement of k-anonymity. The **p-sensitive k-anonymity** [117] enforces that each value of a QID exisits at least distinct p times within the same data fragment. In addition the p-function is limited by the cardinality of the selected sensitive attribute. If p is strictly greater than the selected value, the p-sensitive k-anonymity property is unsatisfactory, because it assumes that each confidential attribute is uniformly distributed over its data set. If this is not the case, p-sensitive k-anonymity causes a major data utility loss [38, 117].
l-diversity [73] adresses the second problem by ensuring the most l frequent values of anonymized published tables T' have approxitmately the same occurence frequency as in T. Consequently, an adversary requires $l - 1$ data fragments of background knowledge to eliminate all possible $l - 1$ sensitive attributes for a successful disclosure. The background knowledge of an adversary is approximated by three assumptions. The attacker knows the domain of the published table T' and the fact that the table is anonymized. Furthermore, s/he has some instance-level background knowledge, i.e., s/he knows specific individuals that are anonymized in the table. Lastly, the attacker has partial knowledge about the distribution of the anonymized data. This assumptions limit the background knowledge of the attacker, because it is possible to gain information about a sensitive attribute, when the global distribution is known [69]. Machanavajjhala *et al.* proposed three heuristics for determining the most frequent records of a table [73]:

1. entropy[2] l-diversity
   Every distribution of sensitive value in each fragment of T' has at least $log(l)$ occurences. Each data fragment has at least l distinct values of a sensitive attribute. This requirement is very restrictive, i.e., when a common attribute is choosen, the information entropy is low.

2. recursive l-diversity
   When a table uses recursive l-diversity, a constant c limits the occurences of each sensitive value. If c is the lower bound for every sensitive value in a data table T', T' is said to be recursive l-diverse.

3. Distinct l-diversity
   Each sensitive attribute of a table T has l distinct attributes in the table T'. This heuristic does not prevent probabilistic inference attacks, e.g. membership disclosures, because an attacker may conclude that a record in the fragment data is very likely to have that value [69].

[2]Information Theory: Entropy describe the density of information in a piece of data [59]

When a table is anonymized with l-diversity, two issues have to be considered.
First, l-diversity addresses most frequent l attributes and less frequent attributes are published unaltered. If an adversary links the plain data to external data, anonymized individuals are unmasked [28]. Second, if data is altered to ensure a uniform density, the utility of such data is lower than in a k-anonymized table [73].

t**-closeness** addresses those problems by splitting the anonymization into two steps. First data is anonymized, e.g. applying k-anonymity or l-diversity to the data, before it is processed further. t-closeness measures and adapts iteratively the disclosure risk from the result set of l-diversity. An observation of this step is the clustering of attributes is affected by the adaption mechanism of the disclosure risk.
In the next step global background knowledge ensures t-closeness for each intra-cluster distribution. t-closeness distributes intra clusters equally to the global distribution in the table T, i.e. the distance between two distributions has a maximum threshold t [69]. The t parametrizes the tradeoff between utility and privacy, effectively limiting the amount of individual-specific information released of a table T' that an observer can learn [70, 82].
As an improvement of t-closeness (n, t)**-closeness** was proposed [70]. (n, t)-closeness enhances t-closeness by the requirement that each data cluster has at least n records. If n is equal to t, (n, t)-closeness becomes equal to t-closeness. On the other hand, if t is set to 0, every table anonymized with k-anonymity satisfies (n, t)-closeness. However, the reverse is not always true, because $(n, 0)$-closeness prohibits the creating of smaller sub-clusters. Li *et al* mentioned, that (n, t)-closeness is not tested with multiple sensing attributes [70].
Previous mentioned anonymization approaches, e.g. $k \geq 3$, are NP hard if they have to guarantee an optimal anonymization [3] [77]. **k-optimization** bypasses this complexity problem [15]. This algorithm uses a different model and assumes an implicit ordering among the QIDs and models the generalization process as a tree. The tree nodes are discrete values of the attributes from original data. Each node separates into intervals (quantitative) or a set of nodes (categorical) and forms a cluster. The trees built from clusters systematically enumerate all possible generalizations. k-optimization minimizes the generalization costs by traversing parts of the tree based on a given cost metric, i.e. minimize the amount of information loss. In the next step k-optimization prunes and dynamically re-arranges tree nodes that are not on the optimal path for keeping the anonymization tree small. A major problem of this algorithm is that data looses its integrity during the re-arrangement of the enumeration tree [15].
An open problem of anonymization is the re-publishing of anonymized data or anonymization of dynamic data. A major issue is that data snapshots differs in its anonymization [4], i.e. anonymized attributes do not change, whereas other generated attributes do. **m-invarivance** addresses this problem and introduces a new generalization model to address temporal writing changes j of a table, called counterfeited generalization [128]. The anonymized version $T^*(j)$ of table $T(j)$ is based on a partition of $T(j)$, and has the following properties [5]:

[3]Proof: Reduction from generalization/suppression to k-Dimensional Perfect Matching - see [77] for further details

[4]c.f. above described temporal inference attack

[5]adopted and simplified from Definition 2 of [128]

1. $T^*(j)$ contains a column A^g named „Group-ID “, and matching all attributes in $T(j)$ except identifier attributes A^{id}.

2. Each tuple $t \in T(j)$ has a **generalized Tuple** $t^* \in T^*(j)$ such that for each sensitive attribute A^s of $t^* = t[A^s]$ hold. Furthermore there exists an additional tuple $t[A^g]$ for each group of QID of t in $T(j)$. If a tuple of QID $t[A_i^{qi}]$ is an interval, then it is generalized as interval tuple $t^*[A_i^{qi}]$ of its QID group.

3. For each group of QID there are arbitrary numbers of **counterfeit tuples** t_c^* such that t_c^* is in the domain of A^s. Then $t^*[A_i^{qi}]$ equals the ID of its QID and is an interval subject.

4. All tuples in $T^*(j)$ with the same A^g have an identical value on every QID. These tuples form a QID group in $T^*(j)$ whose ID is the A^g value of the group.

When a publisher republishes table $T^*(j)$ he has to re-generate a new pair of $\{T^*(j), R(j)\}$ where $R(j)$ is an auxiliary table and stored separatly. $R(j)$ contains a mapping $(IDs, \text{QID}s)$ equal to amount of counterfeited tuples. A future goal is to minimize the amount of conterfeited tuples, because they do not correspond to any records in T'. The objective is to generate a pair of $\{T^*(j), R(j)\}$, that minimizes the risk of disclosure. As a consequence m-invariant produces a coherent data base view with uniform tuples. Each re-distributed data snapshot, even fragments of it, have exacly m unique records for each sensitive attribute values in T' [55, 128].
All anonymization algorithms have the problem of balancing their anonymization, utility and the algorithmic performance. The first and the second parameter is hard to balance. As a result δ-presence was introduced as a metric to measure generalization based algorithms. δ-presence defines an upper and a lower probability $(\delta_{min}, \delta_{max})$ of the quality of anonymization by comparing plain tuples of table T before anonymizing and afterwards. If δ_{min} increases and δ_{max}, decreases more information is anonymized, leading to a better privacy protection, but also results in a poorer utilization. As a result, both parameters have to be ajusted to the context of the application.

3.2 Streaming Environments

When anonymity-based approaches are applied on data streams, they have to address additional data streaming constraints, such as maximum output delays or memory consumption [134]. The selected query defines such parameters and passes them to the data stream engine, e.g. as query window parameters.
Current proposed streaming engines lack of standardization, therefore each engine has a different interpretation of SPARQL Protocol and RDF Query Language (SPARQL), i.e. evaluating time dependent events. We provide a list of available streaming engines and afterward we present streaming anonymization approaches.

3.2.1 Data Stream Engines

Data stream engines execute CQs and computes a result set. Current proposed streaming engines are divided into three categories. Every category has a different stream access model.

1. Data stream management system (DSMS)
   A DSMS consists of a query language, an optimizer for queries, re-scheduling of queries, a resource manager and a basis interface for end users. The output stream is generated based on one of the three patterns. *R-stream* pushes all incoming triples to the output stream. *I-stream* pushes only relevant new triples to the output stream that are not already a part of it and the output of *D-stream* processes only deleted triples. These stream types are inductively defined, i.e. the *R-stream* consists of an *I* and *D-stream*. As result, every stream produces an overhead, because the engine keeps unnecessary parts of the tuples in its memory. This overhead can be minimized by returning a sufficient approximate answer[6] [1, 79].

2. Complex event processing (CEP)
   A complex event is a temporal composition of atomic events. An atomic event is an event representing „*something that occurs, happens or changes the current state of affairs*“ [8]. Atomic events relate to other events by logical time, causality and aggregation. A logical time describes two events A and B, and the fact that A has happened before B irrespective of their timestamps.
   CEP aims to detect (complex) events by means of pattern matching techniques, i.e. filtering and extracting properties of events and mapping them to finite automates [98].

3. Rete
   The Rete „*algorithm emphasises on trading memory for performance by building comprehensive memory structures, called α and β networks for checking intra- and inter-pattern conditions over a set of objects.*“ [62]. Pattern conditions are formulated as logical rules and fired when the left-hand side,i.e., the condition part of the rule evaluates to true. A rule evaluates to true if all parts on the left side rule exists in a α network. If the rule is fired, the right side of the rule is stored as a β node and incremental propagated through the rete network. [43].

The literature distinguishes two categories of streaming engines, depending on the implementation of its architecture. „Whitebox“ architectures require an implementation of all components from a stream processing architecture. In contrast, „blackbox“ architectures focus on LSD query-rewriting and use existing engines as sub-components. Hence, blackbox systems requires a query rewriter for translating SPARQL queries to the wrapped engine's language, an orchestrator for process management and partial result aggregation and a data transformator that transforms resulting data to RDF. Two disadvantages of blackbox system are a lack of control of the processes and required overhead of query and result set transformation. Table 3.2 compares and categorizes current known approaches according to the two categories.

3.2.1.1 Streaming Sparql

Streaming Sparql is an extension of the SPARQL language to cope with CQ.A raw data stream represents data that is received by the DSMS. The logical model abstracts those data and defines

[6]see [79] Section 5 for further details

Engine name	Category	Stream Model	Comment
Streaming Sparql	blackbox	DSMS	Extension of the SPARQL language
CQELS	whitebox	DSMS	
C-Sparql	blackbox	DSMS	
SparkWave	blackbox	RETE	
E-Talis	-	CEP	black and whitebox

Table 3.2: Comparison of SPARQL engines

the query algebra and semantics of the CQ. If such query is used, it is translated into a SPARQL query by providing an accurate logical query plan. The resulting plan is executed on a native streaming processor [17]. The authors of Streaming Sparql suggest to use algebraic optimization before translating a query to a logical query plan [64].

3.2.1.2 CQELS

CQELS models streaming RDF data as flow by distinguishing between RDF streams and RDF data sets. An RDF Stream or a RDF data set is either a data flow whose leaf nodes are stored in a directed tree of operators. Query operators, such as selecting, restricting or partitioning operators, are applied recursively on corresponding nodes to produce a result stream S. These operators are reordered and swapped for improving query performance and creating equivalent data flows during the lifetime of the query. A benefit of traversing parts of queries is adopting them according to frequently occurring events of the raw RDF Stream. Hence this approach allows to take advantage of caching mechanisms to achieve a performance increase when the query is executed again.
A further improvement is using materialized views and triggers as caching and indexing mechanism for data sets. A materialized view is a temporary table that contains results of an executed query. Triggers, on the other hand notify their views when new data is available and update them respectively. If a query is executed, the result of the query is a join between its leaf nodes [63].

3.2.1.3 C-Sparql

C-Sparql converts input SPARQL queries to queries for real-time reasoners to compute a decision. Encapsulated reasoners operate on tuple-based partitional windows, i.e., they are continuously updated. C-Sparql models data as timestamp-annotated triples w.r.t. a relaxed condition that triples are monotonically non-decreasing in a stream. Consequently, triples with a timestamp can occur more than once in each registered stream.
C-Sparql queries are split by the orchestrator in query fragments and streaming fragments. The streaming part is translated into Continuous Query Language (CQL) and delegated to an external DSMS [14].
A disadvantage of such query processors is that their engine neglects updates on the RDF data [64]. Their processing power is limited to short time windows, high-frequency streams and the orchestrator has to deal with large SPARQL fragments when combining the query result [62].

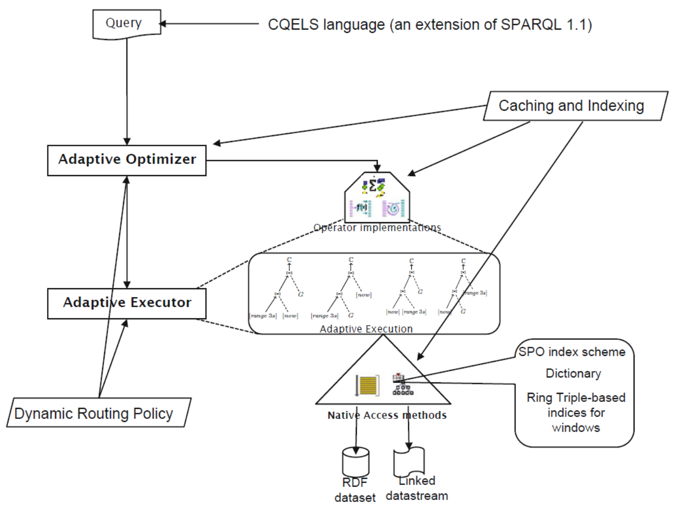


Figure 3.1: Whitebox architecture example: CQELS [63, 64]

Figure 3.2: Blackbox architecture example: C-Sparql [13]

3.2.1.4 E-Talis

E-Talis implements a rule-based language based on CEP. It is focused on detecting near real time CEP rather than posing a query and retrieving an answer. It rewrites a query into event driven backward chaining (EDBC) rules to detect incremental events and applies backward chaining afterward. An EDBC pattern is formulated as a logical formula with background knowledge, i.e., context or domain knowledge that is used for interpreting data of its stream. Complex events of EDBC are detected by filtering and matching raw streams against specific patterns [126] and pruning outdated events from memory. E-Talis has its own intermediate logical language allowing inferencing over different encoded data sources and translating all input data to this language [7, 8]. The classification of architecture depends on the point of view. Le-Phuoc *et al.* [64] classified it as blackbox architecture arguing that EP-Sparql has to preprocess streaming queries into an intermediate format before it is executed [64]. We classify it as whitebox architecture, because the E-Talis developers mentioned that EP-Sparql is the streaming Plug-in of E-Tails and the complete architecture is described in their paper [8]

3.2.1.5 Sparkwave

SparkWave performs continuous schema-enhanced patterns to match RDF data streams. SparkWave uses the $Rete$ algorithm to detect appropriate patterns. The $Rete$ algorithm stores data as logical rules and fires the rules if the left side of the rule is evaluated to true. This evaluation is stored as α nodes in $Rete$'s network. When a rule fires, the right-hand side of the rule is deduced and stored as β nodes. The algorithm fires iteratively all rules that are applicable on the new knowledge after a rule has been fired [43]. This approach of iterative knowledge deriving is extended to address RDF data processing by forming an ϵ network representing entailed triples

as layer above $Rete$'s α and β network. α nodes represents data streams and they are connected to ϵ nodes whereas β nodes functionality is extended for matching partially or completely a scope of a query window. The ϵ network bases on RDF Schema (RDFS) entailment and is responsible for finding a triple match for α and β nodes. RDFS specifies a set of entailment rules, that define the behavior of the $Rete$ reasoning.
One disadvantage of this approach is that $Rete$ does not handle static background knowledge. Furthermore, the memory consumption of Sparkwave depends heavily on the chosen $Rete$ algorithm.

3.2.2 Stream Anonymization

In streaming applications, there is a need to offer strong guarantees on the maximum allowed delay between incoming data and its anonymized output. Furthermore, the following issues have to be addressed:

1. If anonymized data is linked with other data, both data sources may contribute a QID that potentially unmask individuals.

2. Each update on anonymized data triggers a re-anonymization of the complete data set.

3. If an anonymization parameter changes, all data have to be re-anonymized or a new set or stream of anonymized data have to be generated.

4. Current proposed anonymity algorithm require that data is scanned more than once. As a result current anonymization algorithms keep the required data in memory. Consequently, when anonymizing streams, potentially infinite number of queued tuples of the corresponding stream flood the memory.

Cao *et al.* mentioned in their work [24] that traditional k-anonymization schemes rely on the assumption that each record is associated with a different person, i.e., a person occurs in the data set only once. If a person occurs in different data sets, it will be anonymized differently by k-anonymity [45].
CASTLE, a cluster based approach builds a metric space by using QIDs addresses this problem. In a metric space model each QID is modelled as a point. When CASTLE detects new incoming data it clusters triples based on weighted distance and information loss between the cluster and new detected tuples. When CASTLE classifies a new tuple, all data within the same generalized cluster is released. Furthermore, when a data cluster expires, its corresponding cluster data is immediately released [24].
The algorithm ensures anonymity in such clusters by generalizing data with a taxonomy tree [24]. This taxonomy tree is continuously updated, i.e. the generalization levels re-evaluated. As a result the taxonomy tree is continuously growing by adding new generalization nodes [67].
The clustering approach of the CASTLE algorithm has two disadvantages. First, it does not restrict range intervals of each cluster. Resulting intervals depends on QID and are not evenly distributed, e.g. a cluster has the interval [0-999] and another one [1000- 1010]. Second, cluster merging results in a re-clustering of all tuples instead of the affected ones.

An improvement of CASTLE, B-CASTLE adds a restriction on CASTLE's clustering algorithm based on the idea of balancing. A cluster balances itself based on the input data and $\alpha = \frac{\delta}{\sqrt{k+\beta}}$, whereas δ is the maximum delay of the input, β the maximum numbers of clusters and k is from k anonymity[7]. The authors of B-CASTLE addressed the second issue by adopting the merging strategy to a recursive merge with the nearest cluster until k-anonymity is fulfilled. The nearest cluster is detected by applying the correlated distance metric based on the range intervals of each cluster [122].
Instead of anonymizing data when it is generated, data can be anonymized when they are queried. In that case each sliding window is anonymized by providing a Top-Down Specialization Tree (TDS Tree). A TDS Tree consists of nodes representing an equivalence class [8] of data from its window. The size of each equivalence class is defined by k anonymization to be at least k times. When the data is updated, i.e., next data read from an input stream, the TDS Tree is updated [123, 133].

3.3 Discussion

High level anonymity, such as t-closeness is not applied to streaming data. As implication of this higher level anonymization the delay constraint of each stream has to be fulfilled and on the other hand the anonymization has to be ensured. When standardizing LSD the uncertainty of data has to be addressed, when de-referencing or expressing entities. These uncertainty depends on the chosen engine and how its own standardization operations, e.g. interval sizes and timestamp generation are implemented. As a result a combined stream of different data sources standardize and format the underlying data again to provide a unique view on them.
Furthermore analyzing several data streams is in a infancy state [46]. Currently surveillance systems consume static LD data or data streams for providing decisions. The process of getting a decision bases upon multivariate machine learning algorithms, and therefore it is error-prone, i.e., a user defines and adjusts a threshold in each recalibration of the surveillance system. Furthermore the tracked phenomenon of the system has to be adjusted, because they change over the time.

3.4 Contribution

Current proposed streaming anonymization algorithms, such as CASTLE [24] or SKY [67] applied k-anonymity on data streams. As a consequence, disclosure attacks described by Li *et al* successfully identified data from individuals in data streams [69]. We address this issue by proposing a novel approach for streaming anonymization. Our innovative approach is an iterative anonymization algorithm and consists of two steps.
First, we anonymize query windows with k-anonymity and l-diversity to ensure a basic level of anonymization from a single LSD engine. We model this anonymization step as Top down specialization (TDS) tree and initialize it from the Linked Open Data (LOD) cloud at query

[7]occurrences of k QID

[8]An equivalence class is a set of all tuples that contain same values

time. The tree forms a hierarchy of anonymization attributes and if a part of the hierarchy is unknown, our algorithm resolves the missing attributes by appropriate queries and adjusts the anonymization of the stream accordingly. The level of anonymized data depends on the cluster size and adjusts the anonymization in each iteration, i.e. if a cluster is stronger anonymized than it is necessary, the anonymization level decreases and vice versa. Consequently this step ensures an optimal anonymized data stream w.r.t. the amount of Information Loss (IL).
Furthermore, we add another step addressing temporal anonymization. This step standardizes the output of various streaming engines, i.e. it computes an union of all consumed LSD engines. After harmonizing all incoming data, our algorithm applies an improved variant of m-Invariance [128]. This anonymizes temporal changes over query windows.
As a consequence, the anonymized stream is protected against l-diversity and temporal inference attacks.

CHAPTER 4

Streaming Anonymity

Figure 4.1 outlines the implementation of the anonymization algorithms in Chapter 3. We split the architecture into a client including data generation and the streaming engine, and a server. The server incorporate the surveillance system and is described after the client. Both parts apply anonymity, i.e., the client anonymizes the streaming engines, and the server anonymizes data for the surveillance system. Furthermore, the separation between client and server represents different levels of control. The engine based anonymization represents data controlled by the user, whereas a third party, e.g., a governmental organization or a company controls the server anonymization.

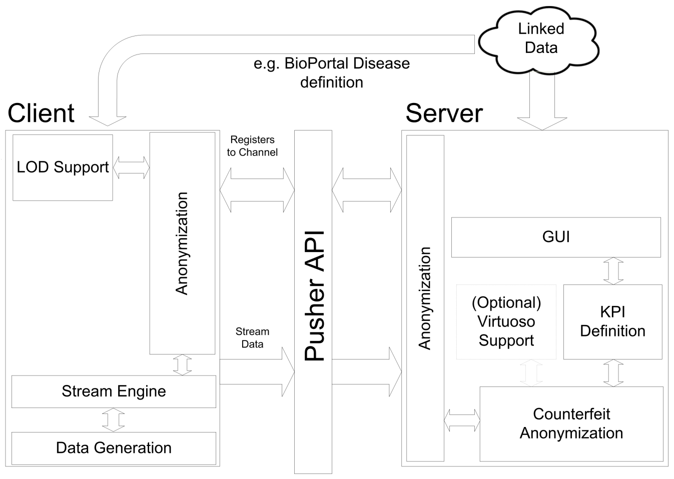


Figure 4.1: Outline of the implemented architecture

We build the data generation on a spatio-temporal model and verified it afterward. CQELS, a LSD engine, consumes and locally anonymizes, i.e. applies k-anonymity and l-diversity within a δ constraint. This constraint models the expiry date at the engine's registered streaming windows. When the δ expires, anonymized data is published on an `EventBus` to summarize `Events` of multiple streaming engines. The server anonymization consumes released `Events` and applies m-invariance for historical anonymization between multiple data snapshots. Finally a map visualizes anonymized data with the lowest ILM

4.1 Data Generation

The aim of the data generator is to generate medical data based on defined input parameters, e.g., time between data points or the amount of generated data points. The data generator produces data points continuously representing people suffering from a disease. Therefore data points are an abstract container for an occurred illness, e.g. a reported cases of influenza. Our selected use case is the influenza infection rate over a year and uses synthetic health care data representing people suffering from a Clinical Condition, i.e. Influenza per day over a year. As a result the use case defines the components of the data generator. The model consists of a temporal and a spatial model. Fricker [44] distinguished between six possible simulation types for bio-surveillance data.

- Temporal vs spatio-temporal
  Temporal models represents daily syndromic phenomenons, whereas the spatio-temporal model is designed for the temporal development of spatial input data.

- Aggregate vs individual-level data
  Aggregate simulation models generate aggregated data, such as daily counts of individuals, whereas individual-levels simulations model individual events such as people's illness.

- Completely synthetic vs partially synthetic
  Completely synthetic models use Monte Carlo algorithms to generate background disease incidence rate or outbreaks. Partially synthetic simulation models have some deterministic parameter for the simulation, e.g. getting some parameter from actual data.

The chosen model is a temporal, individual-level and partially synthetic data model. Consequently we split the dimensions „When“ and „Where“ from the data model and model them separately by using own generators. Our selected use case sets the other dimensions, such as the „What“ dimension to influenza and generates the „Who“ dimension with a static string and an increasing number. The data generation of these dimensions is valid, because our algorithm suppresses the „Who“ later and the „What“ is a sensitive attribute.

Figure 4.2 displays the ontology for LD data points.

Ontology-dependent properties (edges with an arrow) connect entities of the ontology (gray boxes), e.g., *Person* or *Spatial Thing* and links them to different ontologies. List 3 contains all used ontologies.

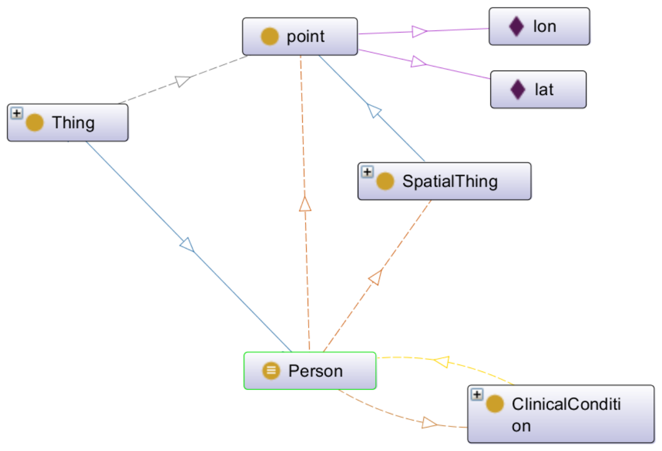


Figure 4.2: Ontology for base data generation

1. Where: SpatialThing from WGS84 Ontology.
   The WGS 84 Ontology models latitude and longitude, but other concepts like street or city are missing. When we wrote the thesis, there is an attempt to standardize these entities[1].

2. What: ClinicalCondition from McGill Ontology - BioPortal.
   ClinicalCondition is an abstract term for all illnesses, e.g. Influenza or cancer.

3. Who: Person from FoaF.

4.1.1 Temporal Model

We model the temporal distribution of Influenza as a sinus based function with a step outbreak function. We scale its distribution to fit Austria's Influenza infection rate [3].

$$Y_t = max(0, \lceil \mu + \gamma_t + \theta_t + \delta_t + o_t \rceil) \qquad \text{with } t = 1, 2, 3 \ldots \tag{4.1}$$

Equation 4.1 is an univariate temporal model for incidence rate [44]. This formula consists of several sub-formulas that represents a time dependent phenomenon of the modeled disease.

$$\gamma_t = A * sin(\frac{2 * \pi * t}{365}) \tag{4.2}$$

Equation 4.2 represents seasonal deviations like fall, winter, spring and summer.

$$\theta_t = B * sin(\frac{2 * \pi * (t + \tau)}{91.25}) \tag{4.3}$$

[1] `https://joinup.ec.europa.eu/asset/core_location/asset_release/core-location-vocabulary-100`, Accessed: 23.2.2015

Equation 4.3 models intra seasonal effects, i.e. in Austria people tend to get sick in January.

$$\delta_t = \delta_{(t-1)} + (\theta_t - \mu)^2 \qquad \forall \delta > 0 \text{ and } 0 \leq t \leq 7 \tag{4.4}$$

δ represents the „day of the week “effect, i.e. clustering of sick people per day.

$$o_t = A * I_t \qquad \text{with } \tau \leq t \leq \tau + \text{ duration of sickness} \tag{4.5}$$

Formula 4.5 models the outbreak function of a disease. As approximation of I_t we used the signum function. Table 4.1 summarizes default parameters of above listed equations.

parameter	description	default value	scaled value
μ	mean	16.75	
τ	starting day of the year	0	-100
σ	standard deviation	5	
A	Magnitude	6.5	20
B		3	

Table 4.1: Equation parameters for the data generator

Figure 4.3 shows the plotted Y_t in blue with scaled values.

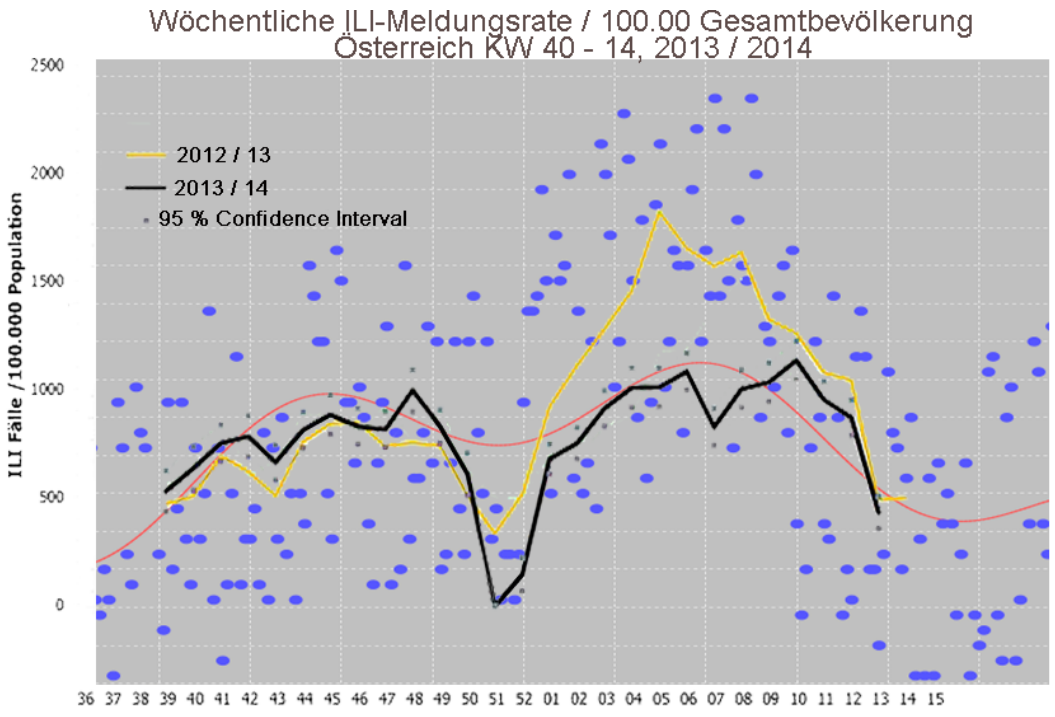


Figure 4.3: Temporal model for data generation (zoomed)

The red line represents an auxiliary line to assist the validation of the seasonal deviation of the amount of people suffering of an illness. The generator generates data points starting at the end of autumn for spreading an influenza epidemic. Figure 4.3 shows the correlation between our data generator and the Influenza data published by AGES. The yellow graph represents the influenza rate from 2012 / 2013 and the black graph from 2013 / 2014. We approximated the data from 2014 by scaling the function to the data published in their report [104].

4.1.2 Spatial Model

The spatial model scales to the amount of people that live in Austria's districts. We retrieve the population per district data from DBPedia and the ratio for selecting a district is defined by $\frac{P_D}{P_A}$. P_D represents the population of a district, e.g., Gänserndorf, whereas P_A represents the amount of people that live in Austria. When a roll [2] is higher than the selection ratio, our data-generator chooses the corresponding district [3].

The method overrepresented Vienna, Graz and Linz because approximately 2.4 million citizens

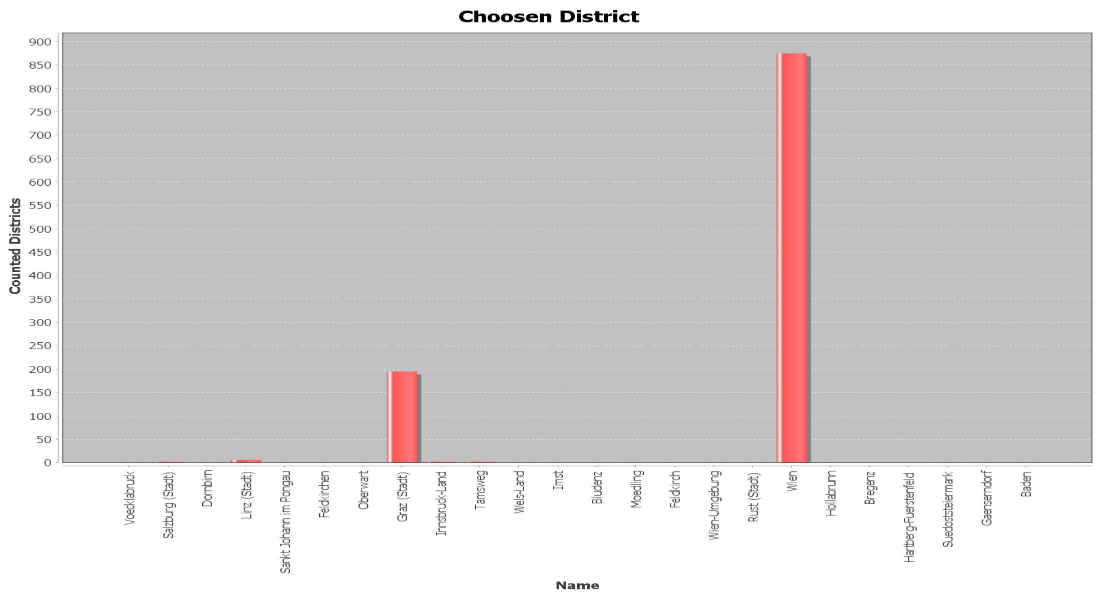


Figure 4.4: Spatial model for data generation

live in those cities (see Figure 4.4). As a result those three areas clustered all incidence cases. We use a projection[4] $f_n = r * g(n) = \frac{1}{n}$ with $0 \leq r \leq 1$ in a intermediate step for using an uniform probability of $\frac{1}{n}$ for each district. Then we sorted the probability of each district into an interval from $(0 \ldots 1)$. When the roll is in between the intervals, the generator selects the

[2]A roll represents a call of Java's Math.random()

[3]Due to performance purposes this values are cached as a static class

[4]We thank Marcus Hudec, a Professor of the Faculty of Computer Science - Data Analytics and Computing, for proposing this projection.

corresponding district and multiplies the number of incidence cases by the r ratio of the district.

4.1.3 Verification

We verified and scaled the model with data from official sources, e.g. Statistik Austria, Diagnose Netzwerk Influenza Österreich, Agentur für Gesundheit und Ernährungssicherheit and Medizinische Universität Wien of the year 2013. Table 4.1 lists the scaled values for the temporal model as „scaled value“.

4.2 Stream Engine Anonymization

The anonymization of a query window from an engine detects disclosures of QIDs within a query window and the application of local anonymization w.r.t. the maximum output delay δ of the query window. We chose anonymity-based approaches because target attribute swapping changes the interpretation of swapped data in a tremendous way, i.e., different attribute combinations have different characteristic illnesses [51, 97].

4.2.1 Streaming Engine

We selected CQELS as streaming engine because it has a lightweight whitebox architecture. In the initialize-phase, our algorithm registers a query listener with the sliding window query 4.1.

```
PREFIX wgs84: <http://www.w3.org/2003/01/geo/wgs84_pos#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT ?sex ?birthday ?phen ?lat ?lng
WHERE{
STREAM <http://ldlab.tuwien.ac.at> [RANGE 10s SLIDE 5s] {
?person wgs84:location ?point;
        foaf:birthday ?birthday;
        foaf:gender ?sex;
        <http://isis.tuwien.ac.at/surveillance/People#suffers> ?cond.
?cond a ?phen.
?point a wgs84:Point;
        wgs84:long ?lng;
        wgs84:lat ?lat.
}}
```

Listing 4.1: CQELS Listener Query

The query listener retrieves non sensible attributes such as sex, birth date, ... from the engine. Our algorithm maps the result set of the listener to a `Datapoint`, an object-oriented representation of the four `Dimensions` and dumps the RDF data. Each `Dimension` is null by default, representing that it is unset. A `Dimension` consists of several attributes, e.g. the *Where*-dimension has attributes named latitude, longitude, street, city, district, country. If an attribute is unset, our algorithm retrieves it from the LOD cloud.

4.2.2 Local Anonymization

Local anonymization is an iterative approach consisting of a sequence of steps illustrated in Figure 4.5. The flow chart is a simplified version [5] and each step of local anonymization is visualized as a black box. The diagram illustrates a single iteration of the anonymization algorithm before releasing the anonymized data.

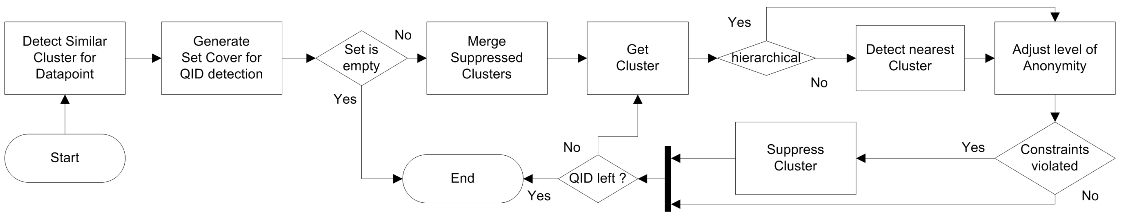


Figure 4.5: Flow Diagram for local anonymization

Algorithm 4.1 is the corresponding pseudo-code to the flow of Figure 4.5. It provides a more detailled view on the anonymization of the dimensions. A sub-routine detects the type of the dimension.
A dimension has several attributes categorized as strict hierarchy or non-hierarchy attributes. If an attribute is categorized as a strict hierarchy, a taxonomy tree is implicitly modeled by the hierarchy itself. Step 1 initializes these hierarchies and resolves them by exploiting the property that abstracter terms, like countries, have lesser instances than concrete terms, e.g. street [85].
On the other hand non-hierarchical attributes are not logical ordered. As a consequence the algorithm anonymizes each attribute separately by replacing the least significant digit of the string representation of the `Dimension` with an asterisk. When our algorithm applies a further step the next non asterisk least significant digit is replaced.
When Algorithm 4.1 processes a new `Datapoint`, a similar cluster is discovered by resolving its dimensions with Algorithm 4.2. A cluster is a collection of similar dimensions of multiple data points identified by an abstract key of the dimension and corresponds to a specific scheme of the anonymization hierarchy. We measure this similarity between the cluster and the `Datapoint` with the Levenshtein distance [29]. If the anonymity of a clusters changes, e.g. by splitting clusters, a new value of the anonymization hierarchy identifies the new cluster. The anonymity of cluster depends on the available amount of data points of a given dimensions, i.e. if k anonymity and l diversity is fulfilled anonymity decreases and vice versa.
Furthermore, anonymity depends on the combination of the dimensions, i.e., combining more than one dimension forms QIDs. Detecting QIDs is NP-hard because of the combinatorial complexity of the QID. Motwani and Xu addressed this problem by a reduction of the QID

[5] All data caching, setting local variables, and generating key calls are skipped

```
Data: Cluster, Datapoint
Result: anonymized Datapoints of a Streaming Engine
getLODHierarchies;
// Iterative steps
searchCluster(Dimension);
if Cluster == null then
    generateCluster();
    Cluster.suppress = true;
end
Cluster.addDatapoint();
set = generateSetCover();
for $s \in set$ do
    if $s$ = Where-Dim then
        if Cluster.size < k then
            $supCl$ = getSuppressedClusters();
            mergeClusters($supCl$);
        end
        // replaces the geo-Information with a more coarse one
        anonymizeCluster();
    end
    if $s$ = Who-Dim then
        if Cluster.size < k then
            $supCl$ = getSuppressedClusters();
            mergeClusters($supCl$);
        end
        // replaces the lsb with a *
        anonymizeCluster();
    end
    if $s$ = What-Dim then
        if Cluster.size < k then
            $supCl$ = getSuppressedClusters();
            mergeClusters($supCl$);
        end
        anonymizeCluster();
    end
end
if Cluster.isExpired then
    releaseClusters;
end
```

Algorithm 4.1: local Anonymization

```
Data: Dimension
Result: key of Cluster
Cluster = getCluster($dim$);
if $cl$ = $null$ then
    generateCluster();
end
if $dim$ != $hierarchical$ then
    cl = getNearestCluster($dim$);
    removeDim(cl);
    if $dim$ != $Number$ then
        dim2 = moveAsterisk($dim$);
    end
    else
        dim2 = generalizeInterval($dim$);
    end
    add(cl, dim2);
end
else
    Cluster.addDimension($dim$);
end
return dim2;
```

Algorithm 4.2: Cluster classification

finding problem to a $minSetCover$ problem[6]. The disadvantage of this method is that they always return the founded set of min cardinality of QID [80] whereas we have the necessity for all sets of QIDs. Therefore we relaxed the min condition of $SetCover$ and returned all potential combinations of QIDs. We tested this setting by using a generic input data table from their paper, i.e. a table with age, sex and state information. We scaled it to one record per second, representing a one minute query window of CQELS.

Figure 4.6 displays the detected columns per QID over 10.000 repetitions. The graph of the figure is cropped at window size 30 because the combinations of columns increases exponentially and therefore the amount of QIDs decreases. Increasing the δ, e.g., to half an hour has no effect on the amount of QID[7].

Algorithm 4.3 detects the combined dimensions of a QID and anonymizes them by using generalization. Additional potential outliers of all clusters, i.e., clusters that violates anonymization constraints are suppressed.

When the δ expires, all clusters are removed and the un-suppressed clusters are published on the Publish and Subscribe event bus.

[6]A mathematical proof is provided at their appendix

[7]Tested up to half an hour with 100 repetitions

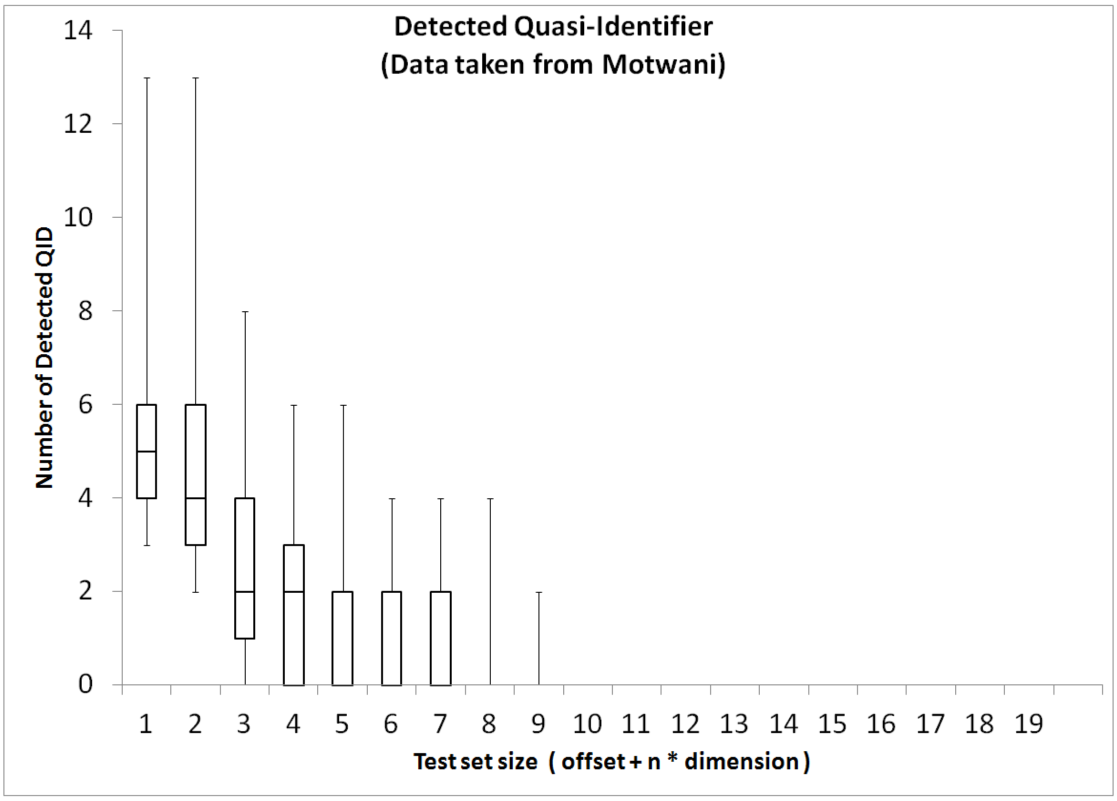


Figure 4.6: Number of detected QIDs with 10000 repetitions

4.2.3 Anonymity Experiments

In this section we describe the anonymity experiments made with our prototype. Our main goal was to detect the best parameters for streaming anonymization w.r.t. to the amount of resulting IL. First, we defined the ILM that is used in our tests to measure the amount of lost information. An information is lost, if and only if, a concrete value is replaced by an abstract or suppressed. Then we defined our executed tests. Independent of the test, we used the data generator isolated from the streaming engine as input data stream to minimize side effects, such as the initialization of engine. Furthermore, before we ran a test, we deleted and re-initialized all files to start from a defined state.

4.2.3.1 Information Loss Metric

The distances from generalized or suppressed data to un-anonymized data is referred as IL. An ILM is a distance metric counting those distances in the released data. Equation 4.6 defines the IL for a data point [22].

$$IL(e) = |e| * \sum \frac{Max_{N_i} - Min_{N_i}}{|N_i|} + \sum \frac{H(\Lambda(\cup_{C_j}))}{H(\mathcal{T}_{C_j})} \tag{4.6}$$

```
   Data: Cluster
 1 Cluster Data: Datapoint
 2 Datapoint Result: anonymized Cluster
 3 for Cluster ∈ allClusters do
 4     if Cluster.dimension > l then
           // l-diversity check
 5         if Cluster.size > k then
               // one step down
 6             decrementAnonymity;
 7         end
 8         else
 9             incrementAnonymity;
10         end
           // anonymize resulting Cluster
11         anonymizeCluster();
12     end
13     else
           // l-diversity check failed
14         Cluster.suppress = true;
15     end
16 end
```

Algorithm 4.3: Anonymization Algorithm

$|e|$ is the number of records of an attribute, $|N|$ represents the size of numeric domain N, $\Lambda(\cup_{C_j})$ is the subtree rooted at the lowest common ancestor of every value in $\cup_{C_j}$ and $H(\mathcal{T})$ is the height of taxonomy tree $\mathcal{T}$ [22].
The comprehensive IL summarizes all $IL(e)$ of the query window. The implemented IL is simpler because every attribute is mapped to a hierarchy. As a result the term of numeric properties was set to 0.

4.2.3.2 Anonymity Measurements

Parameters of 0 and anonymity parameters $1 \leq$ violates the anonymization constraints we defined based on research question 1. Consequently, we excluded such settings from our anonymity experiments.
In the first test, we measured the influence of the anonymization parameters. We increased one parameter and the other two were fixed, e.g. k was set to three and the rest to two. We assumed the δ in miliseconds and increased it by $iteration$ * 250. This tests aimed to find the most dominant parameter from the algorithm. The parameters of the measurements are iterated until 30, because of the observations made in the previous section (see Figure 4.6).

Figure 4.7 visualizes the runtime behavior of the local anonymization with increasing parameters

k, l and δ.

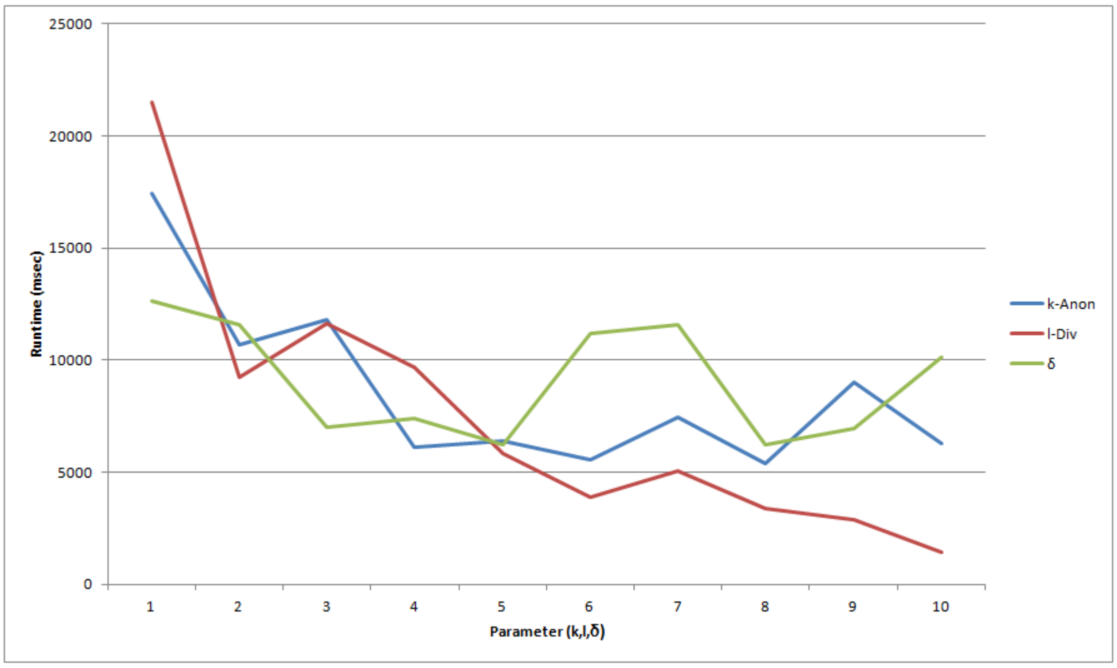


Figure 4.7: Runtime measurement of local parameters

When the k parameter is low, more time is required to compute an anonymized data set, because each dimension of a data point is located as separate cluster. When both parameters are increased the runtime and the amount of clusters decreases. As a result, the runtime and the amount of generated clusters strongly correlate (see Figure 4.8).
The window time constraint δ has the highest influence on the runtime because it forces to release all cluster when δ expires (see Figure 4.7 for an analyse of δ parameters).

In the second test, we searched the best parameter for k-anonymity.

Figure 4.8 displays the IL obtainted for increasing parameters of k. The settings for the measurement are the same of the runtime measurements. As a result, the IL increases linearly with the runtime and directly correlates with the amount of generated clusters.

4.2.3.3 Test Results

One goal of the thesis was to identify the best setting for streaming anonymization. The best setting was the combination of k, l and δ that ensures anonymity with the lowest IL.

The excerpt of the best combinations regarding IL and runtime are listed in Appendix A.1. A frequently detected pattern is that best combinations have set k to two, $l \geq 20$ and a δ of four. The resulting average IL is between 0 and 70. The ILM represents the amount of anonymization movements within a data release, e.g. when data is anonymized the number of IL is accordingly increased or decreased.

4.2.3.4 Discussion

l-diversity is an extension of k-anonymity, hence l-diversity has a lower impact on the runtime of the algorithm compared to the application of k-anonymity.
More generally, density based anonymity approaches such as l-diversity may not applicable be to streaming engines, because they enforce that the sensitive attribute occur at least l times. This is contrary to the idea of a query, i.e. selecting data that has a defined attribute.
If we compare our results with those results listed in the streaming anonymity literature [24, 67], we are able to reproduce their results.
Compared to the experiments of Cao *et al.* we increased the size of the unanonymized data, because we assumed a discrete data stream. This was a valid assumption because of the variety of the plain data stream. Consequently, our additional observation [8] is that the amount of detected QIDs exponentially decreases with the increasing size of the data when the amount of potential QID is limited by a static number (see Figure 4.6).
When we compare our approach to other approaches from the literature, our implementation has less IL than the proposed approaches. The IL of approach increases logarithmic and not linear compared to Cao and Li. However we confirm that, when an anonymization parameter is increased, the IL also increases (see Figure 4.8).

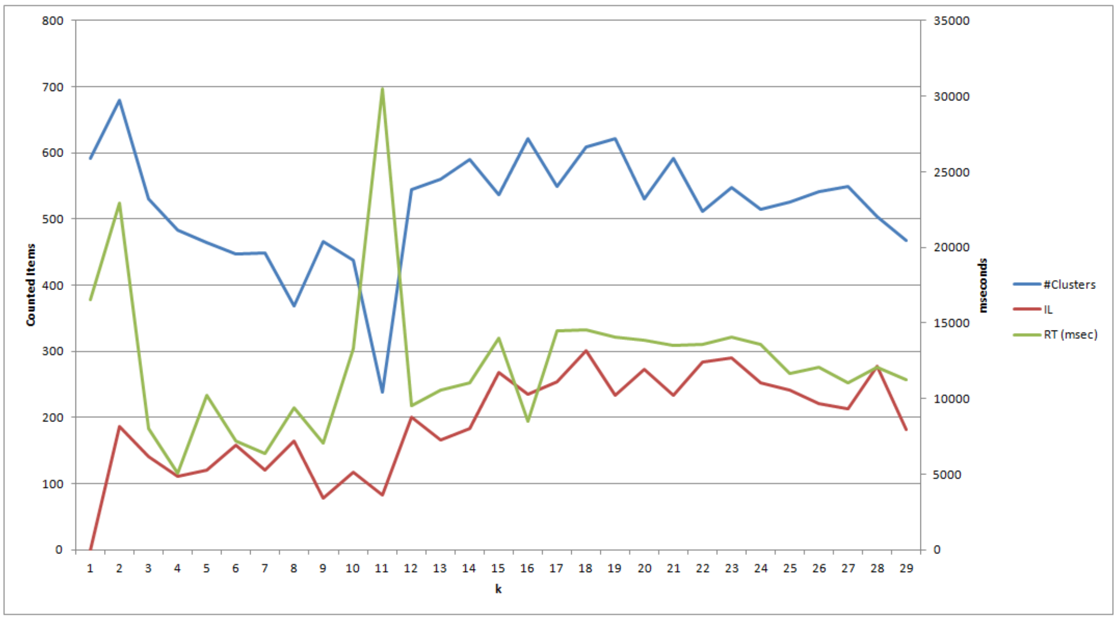


Figure 4.8: Information loss of local anonymization w.r.t. setting

When we use the data generation model specified in section 4.1, the amount of outliers is than increased. Thereupon we made the observation, that the amount of outliers have lower impact on the runtime, because when a cluster is merged we start with the abstract root node, e.g. Austria, and move down to the concrete leaf nodes. When a node has more than k children, we generate

[8]based on those made by Cao [23]

a separate sub-cluster. As a consequence, when two clusters of outliers are merged and the resulting cluster is generated in linear time.

4.3 Anonymization of Meta-Data

Our main motivation for anonymization of multiple window data is to avoid inference or distribution attacks over continuously republished data such as data streams [128].
Consequently, our algorithm uses historic anonymization to anonymize temporal changes between two snapshots and afterward it checks if the data violates any diversity constraints.
Figure 4.9 outlines the anonymization process of two snapshots of one engine [9]. Without loss of generality, a snapshot is data returned by a query window at timestamp t.

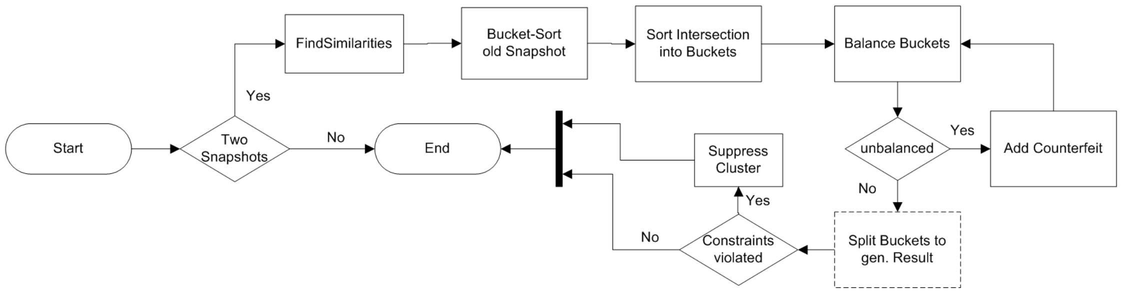


Figure 4.9: Flow Diagram for anonymizing multiple query windows

4.3.1 Historical Anonymization

Historical meta anonymization avoids changes of the structure of multiple snapshots by *exploiting the correlation between two „snapshots“* [128]. A modification of the amount of anonymized snapshot data forms an identifiable record by reducing or adding the anonymization of non sensitive attributes in two different snapshots. As a result, we use a distance metric to detect such correlations and anonymize its data thereupon.
In contrast to m-Invariance proposed by Xiao *et al.* [128], we assume that input data is already anonymized with k-anonymity, because we used m-Invariance as a further anonymization step after local anonymization, i.e. the algorithm has no information about un-anonymized data. List below summarizes all differences of our approach compared to the approach proposed by Xiao *et al.*

1. Division Phase
   Each datapoint has a signature representing the string concatenation of all non sensitive attributes sorts the datapoint into a bucket. The intersection of such buckets is stateless and minimizes the Levenshtein distance between each signature, because the same tuple of s_{t-1} forms a similar anonymization cluster in the snapshot s_t.

[9]The dotted box is optional and is used for debugging purposes

2. Balancing and Assignment Phase
   Our algorithm uses data from the intersection as counterfeit tuples instead of generating counterfeit tuples. After consuming all data of the intersection and if a bucket is not balanced, a counterfeit tuple is generated. The main goal of this step is to minimize the amount of counterfeits.

3. Split Phase
   This phase is omitted. It is replaced by collecting all datapoints of all buckets.

Our algorithm stores counterfeit tuples in a separate temporary table as bucket-Id and counterfeit-tuple pair and prints the table on the command line. Algorithm 4.4 sketches our modified version of Cao's m-Invariance.

```
Data: two Snapshots: s$_{t-1}$, $s_t$
Result: $m$-Invariance of $s_{t-1} \cup s_t$
1 inter = Intersection( $s_{t-1} \cap s_t$);
2 buckets = BucketSort($s_{t-1}$);
  // removes intersection datapoints from $s_t$
3 remove($s_t$, inter);
4 while $s_t$ not empty do
5     Bucket.addDatapoint(bucket);
6 end
7 balancedBuckets = balance(buckets);
8 return balancedBuckets;
```

Algorithm 4.4: m-Invariance Anonymization

It adds counterfeit tuples, if and only if a bucket is not eligible. A bucket is eligible if all identifying elements, e.g. „What"-dimension, contains the same amount of data points [128].

```
Data: buckets of a snapshot
Result: balanced buckets with counterfeit data
1 max $\leftarrow$ number of Datapoints;
2 for bucket $\in$ buckets do
3     if bucket.Datapoint.size $\leq$ max then
4         generateCounterfeit($bucket$);
5     end
6 end
```

Algorithm 4.5: Balancing Phase of m-Invariance

4.3.2 Anonymity Experiments

In this section we describe anonymity experiments between different data snapshots. Our motivation was to validate of our algorithm and to detect limitations of it. A further goal was to

minimize the amount of generated IL. Therfore we provide an enhanced ILM of Section 4.2.3.1 and add a penalty term for the numbers of generated counterfeits.
Then we defined our settings for the tests. The experiments are executed with continuous streamed data, i.e. static data from previous results are used as seed and streamed to our algorithm.

4.3.2.1 Global Information Loss Metric

We enhance the equation of section 4.2.3.1 by adding a term Ψ for the counterfeit tuples. Equation 4.7 defines the global IL.

$$IL(e) = |e| * \sum \frac{Max_{N_i} - Min_{N_i}}{|N_i|} + \sum \frac{H(\Lambda(\cup_{C_j}))}{H(\mathcal{T}_{C_j})} + \sum \frac{H(\Psi(\cup_{T_i}))}{|T_i|} \tag{4.7}$$

The ILM from Equation 4.6 are unchanged, i.e. the first summarizes IL ratio for numeric attributes and the second for taxonomy trees. The new term adds a ratio of the amount of used counterfeit tuples for a sensitive property combination.

4.3.2.2 Anonymity Measurements

We exclude from our measurments snapshots with one signature of anonymized data and snapshots that occured only once, because such data is excluded as trivial case of a anonymized data stream and therefore not suitable for streaming anonymization.
Our first experiment for anonymizing multiple query windows is a functional test where we reproduced the results of the paper from Xiao *et al.* The input data of this test was data from Table 1 and Table 2 and the expected returned data was Table 3 from the m-Invariance paper [128]. Our motivation for this test was to proof the validity of our approach by comparing the output of both algorithms. The next experiment scaled the m-Invarance w.r.t. to the number of input data and the number of sensitive attributes up to a maximum of 10 [10]. This test aimed to detect and measure the process of generating counterfeit tuples. The input data of the first test was used as a seed and continuously consumed by our algorithm to produce a comparable output.
We ran the experiment again with seeded data from local anonymization with further data for simulating an outbreak of illness, i.e., to instigate the generation of counterfeits by increasing number of outliers. Our last experiment was to analyize the behaviour of the algorithm with simulated data, i.e., we used data snapshots of local anonymization as seed to generate compareable snapshots. These snapshots were anonymized with our approach to analyse the behaviour of our algorithm.

4.3.2.3 Results

The first test proved the validity because it produces the same results listed in the paper [128]. The conclusion of the second test was that the amount of sensitive attributes is proportional to

[10]Motivated by the Paper from Xiao

the amount of generated counterfeits. Table A.2 in the appendix lists the precise data and the setting of this test.

The reason for this observation is that sensitive attributes defines the number of buckets because their value is used as label. When these buckets are checked for eligiblity, the algorithm adds counterfeits until the corresponding bucket is eligible. Figure 4.10 displays the number of measured counterfeits w.r.t. the total number of used tuples and the runtime. Furthermore the number of counterfeits is direct proportial to the amount of consumed tuples.

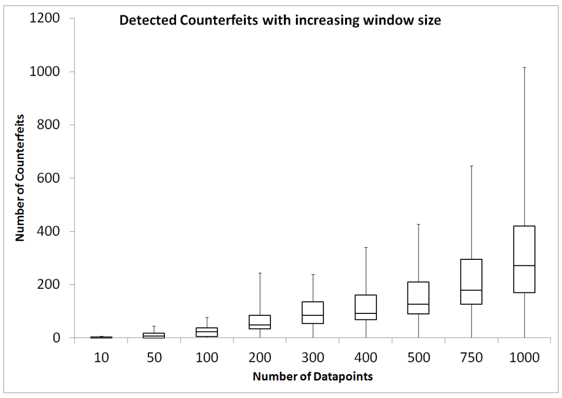


Figure 4.10: Number of generated counterfeits as a function of increasing snapshot size

4.3.2.4 Discussion

Our approach has less IL than approaches proposed in the literature, because we generate counterfeit tuple if and only if we cannot use existing data from the intersection. Therefore, the amount of generated counterfeit data is minimized because the ILM considers counterfeit as an additive term.

Furthermore, our approach copes with n sensitive attribute instead of being limited to one [127]. A consequence of our tests is that the number of sensitive attributes scales linearly with the number of generated counterfeits and we can confirm the observations of m-Invariance made in the literature [127, 128].

We discarded t-closeness, because this anonymization technique requires global background knowledge of the snapshot. It changes when our algorithm anonymizes a new snapshot, and as a result the background knowledge is rebalanced in every iteration. Hence the background knowledge is temporal stateless, i.e. it has no information about the previous snapshot and therefore it does not prevent de-anonymization between multiple snapshots of data streams. As a consequence current proposed solutions of streaming t-closeness does not fit into the scope of our Research Question 1 [25].

m-Invariance addresses this problem by generating a counterfeit tuple and each snapshots has uniform distribution that slowly changes w.r.t. to the time [36].

4.4 Visualization

The goal of the visualization is to display and summarize the generated data in a user friendly way by calling appropriate methods of the endpoint. Before data is retrieved the visualization has to register listeners on the corresponding channel from the endpoint.

4.4.1 Endpoint

The endpoint for registering a communication channel is programmed in Apache CXF. The data is sent chunk-wise[11] to avoid HTTP 413 errors. This error occurs when a too large list of data points is serialized.
The number of chunks is fixed at $\frac{n}{50}$ where n is the size of the result set. The data load is minimized by splitting the data into anonymization, IL and debug data. Every data is released on the channel and by default the application registers on the anonymization channel.

4.4.2 Front-End

The Front-End is programmed in JavaScript (JS). Open source libraries, such as OpenLayers and Pusher are used to display the geographical information of a datapoint. The end user is able to choose between multiple visualizations, e.g. a heatmap or pins depending on the type of information that shall be displayed.

4.4.2.1 OpenLayers

OpenLayers displays the „*Where*“dimension of the datapoint as geo-shape on a map. A geo-shape is colored if and only if the incidence rate is higher than a threshold [12] of the maximum incidence rate that is retrieved from the endpoint[13]. Figure 4.11 displays the map view colored political districts with a high incidence rate.

4.4.2.2 Pusher

Pusher allows stateless communication between the REST endpoint of meta-anonymization and the front-end. The library itself is licensed under the MIT license, but built of a commercial company. Therefore there are restrictions on the amount of messages being sent every day.

[11] Approximately 50 datapoints per chunk
[12] 20, 40, 60 and 80%
[13] `http://www.ages.at`, Accessed 21.3.2015

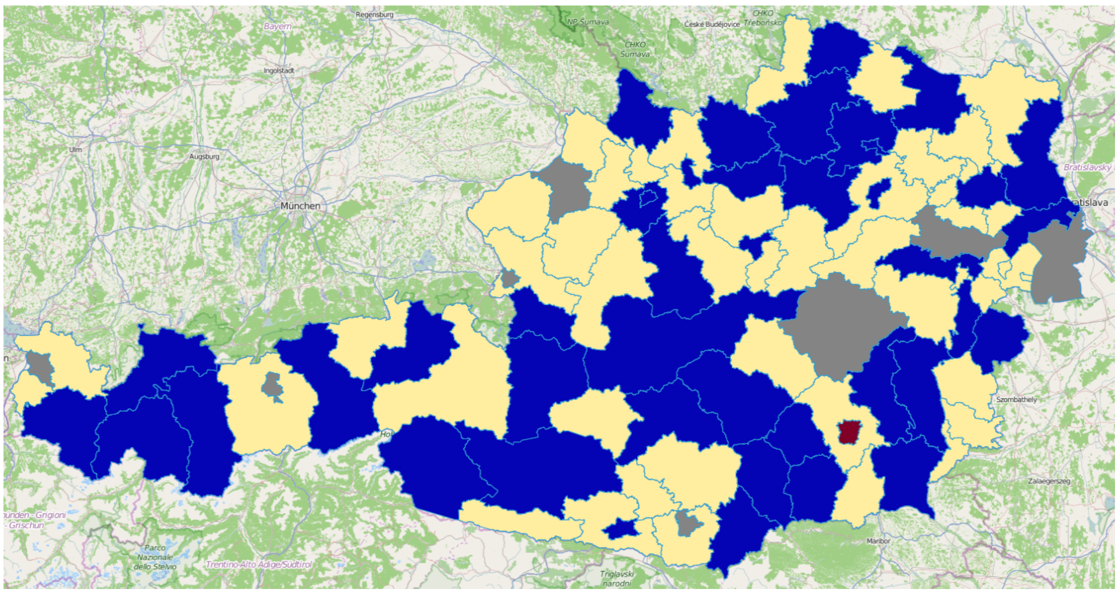


Figure 4.11: Map view of anonymized incidence rate per political district

CHAPTER 5

Conclusions and Future Research

Anonymization with an adequate utility has to be ensured before the data is released. These goals are achieved by our two level architecture.
We developed an adaptive streaming anonymization algorithm to maximize the generated utility of our architecture. This algorithm adjusted the anonymization to the consumed amount of data and to the available data at the LOD cloud. As a result, it suppressed or generalized single parts of the four dimensions from a data point. We tested this approach with a parameterized data generator to analyse its limitation w.r.t. different statistical disease models. Consequently, our streaming algorithm produced a lower IL than current solutions from the literature.
Based on the results of the streaming algorithm, we anonymized data between similar snapshots to avoid unmasking individuals that do not change within several snapshots. We tested the scalability of our approach by increasing the amount of data for a single sensitive attribute.
Our solution has the advantage that it is highly extensible and independent of the implementation. Hence it allows to add or remove engines or properties during execution. Furthermore it is implemented as a lightweight and separated the streaming anonymization from the anonymization of the snapshots.
A further research direction is to improve the model of the background knowledge of adversaries. An adversary may have absolute facts or partial related knowledge of specific groups of the demographic distribution, for instance that male persons never have ovarian cancer. In addition he may receive background information from other sources or use inference mechanisms to unmask individual by combining those data sources [71].
Another research direction is the exhaustive exploration of statistical models with LOD to minimize the model drift by periodical refinements with linked (stream) data. The idea of exhaustive exploration is to systematically explore the state-space of the model w.r.t. to defined model constraints [35].
When a model is built from a data set, the modeller may use data from Eurostat[1] or any other statistical LOD dataset. These datasets contain defined concepts that are interlinked[2] with other

[1]`http://ec.europa.eu/eurostat/de` Accessed 10.3.2015
[2]e.g. with the „sameAs " relation

datasets. As a result, the state-space of the model is unlimited, and it potentially finds solution consuming less resources or better satisfying its constraints.

APPENDIX A

A.1 Optimization of anonymization Parameters

The scope of this test is to find optimal parameter combinations for k, l and δ. The test algorithm generates 1000 datapoints and anonymizes them. After the algorithm finishes the IL, the amount of generated clusters and the runtime are measured. Measurements with $k = 1$ are excluded, because this combinations release data unanonymized. Table A.1 contains the fourty combinations w.r.t. IL[1] and runtime and is sorted by IL and Runtime descending.

[1]This property is the pivot property

k	l	δ	Number of Clusters	Information Loss	Runtime (in msec)
2	17	4	0	0,000	177
2	25	4	0	0,000	344
2	27	3	169	10,000	2173
2	23	4	170	20,000	1775
2	30	4	257	20,000	2494
2	28	2	129	25,000	781
2	28	4	71	25,000	1133
2	18	3	160	25,000	1835
2	30	2	159	30,000	1335
8	26	2	191	30,000	1544
2	13	3	375	30,000	3062
2	20	4	303	30,000	3432
2	12	4	205	35,000	2741
2	24	4	270	35,000	3344
2	25	3	133	45,000	1593
2	17	2	410	45,000	3438
2	7	3	218	45,800	2985
2	8	4	266	45,800	3203
2	29	2	199	50,000	1919
2	10	4	249	50,000	3766
5	5	1	463	50,400	3266
2	8	3	262	50,800	3034
2	7	4	263	50,800	3188
2	9	1	403	55,000	1626
2	27	1	528	60,000	1945
2	13	2	344	60,000	2297
2	14	2	402	60,000	2563
2	23	2	365	60,000	2682
2	15	3	256	60,000	3122
2	9	3	303	60,000	3374
2	26	4	297	60,000	4011
9	27	2	481	60,600	5528
2	6	2	385	61,600	2709
2	6	3	191	61,600	2935
2	6	4	241	61,600	3763
2	2	3	310	63,200	5076
2	10	1	561	65,000	2034
2	19	2	468	65,000	2823
2	14	4	225	65,000	3418
3	26	4	453	65,200	6349
5	28	4	91	65,600	1031
2	29	1	411	70,000	1511
2	7	1	587	70,000	2008

A.2 Counterfeit Data Test Result

Table A.2 displays the results from the meta-anonymization test. In this test the snapshot data is continuously increased and the amount of counterfeits are measured after each test. A test is repeated 100 times. The amount of sensitive attributes are set to four.

Datapoints	Min	1.Quartil	Median	3.Quartil	Max
10	0	0	1	3	8
50	0	0	7,5	17,25	45
100	0	5,75	24	38	77
200	2	35	49,5	84,5	245
300	37	54	85	135	240
400	4	69	92,5	162	340
500	42	90	127,5	210,5	427
750	37	127	180	295,25	646
1000	6	171	271	420,75	1016

Bibliography

[1] Daniel Abadi, Donald Carney, Ugur Cetintemel, Mitch Cherniack, Christian Convey, C Erwin, Eduardo Galvez, M Hatoun, Anurag Maskey, Alex Rasin, et al. Aurora: a data stream management system. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 666–666. ACM, 2003.

[2] Nabil R Adam and John C Worthmann. Security-control methods for statistical databases: a comparative study. *ACM Computing Surveys (CSUR)*, 21(4):515–556, 1989.

[3] AGES. Grippe - influenza. `http://www.ages.at/themen/krankheitserreger/grippe/saison-201415/`, 2015. [Online; accessed: 2015-06-08].

[4] Charu C Aggarwal. On k-anonymity and the curse of dimensionality. In *Proceedings of the 31st international conference on Very large data bases*, pages 901–909. VLDB Endowment, 2005.

[5] Charu C Aggarwal and S Yu Philip. *A general survey of privacy-preserving data mining models and algorithms*. Springer, 2008.

[6] Dakshi Agrawal and Charu C Aggarwal. On the design and quantification of privacy preserving data mining algorithms. In *Proceedings of the twentieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 247–255. ACM, 2001.

[7] Darko Anicic, Paul Fodor, Roland Stuhmer, and Nenad Stojanovic. Event-driven approach for logic-based complex event processing. In *Computational Science and Engineering, 2009. CSE'09. International Conference on*, volume 1, pages 56–63. IEEE, 2009.

[8] Darko Anicic, Sebastian Rudolph, Paul Fodor, and Nenad Stojanovic. Stream reasoning and complex event processing in etalis. *Semantic Web*, 3(4):397–407, 2012.

[9] Ajit Appari and M Eric Johnson. Information security and privacy in healthcare: current state of research. *International journal of Internet and enterprise management*, 6(4):279–314, 2010.

[10] Arvind Arasu, Brian Babcock, Shivnath Babu, John Cieslewicz, Mayur Datar, Keith Ito, Rajeev Motwani, Utkarsh Srivastava, and Jennifer Widom. Stream: The stanford data stream management system. *Book chapter*, 2004.

[11] Arvind Arasu, Shivnath Babu, and Jennifer Widom. The cql continuous query language: semantic foundations and query execution. *The VLDB JournalThe International Journal on Very Large Data Bases*, 15(2):121–142, 2006.

[12] Ron M Aryel. Physiologic and space-based sensors. *Handbook of biosurveillance*, page 387, 2006.

[13] Davide Francesco Barbieri, Daniele Braga, Stefano Ceri, and Michael Grossniklaus. An execution environment for c-sparql queries. In *Proceedings of the 13th International Conference on Extending Database Technology*, pages 441–452. ACM, 2010.

[14] Davide Francesco Barbieri, Daniele Braga, Stefano Ceri, Emanuele Della Valle, and Michael Grossniklaus. Querying rdf streams with c-sparql. *ACM SIGMOD Record*, 39(1):20–26, 2010.

[15] Roberto J Bayardo and Rakesh Agrawal. Data privacy through optimal k-anonymization. In *Data Engineering, 2005. ICDE 2005. Proceedings. 21st International Conference on*, pages 217–228. IEEE, 2005.

[16] Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked data-the story so far. *International journal on semantic web and information systems*, 5(3):1–22, 2009.

[17] Andre Bolles, Marco Grawunder, and Jonas Jacobi. *Streaming SPARQL-extending SPARQL to process data streams*. Springer, 2008.

[18] Justin Brickell and Vitaly Shmatikov. The cost of privacy: destruction of data-mining utility in anonymized data publishing. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 70–78. ACM, 2008.

[19] David L Buckeridge, H Burkom, A Moore, J Pavlin, P Cutchis, and W Hogan. Evaluation of syndromic surveillance systemsdesign of an epidemic simulation model. *Morbidity and Mortality Weekly Report*, pages 137–143, 2004.

[20] Michael K. Buckland and Fredric C. Gey. The relationship between recall and precision. *JASIS*, 45(1):12–19, 1994.

[21] Howard S Burkom, Sean Patrick Murphy, and Galit Shmueli. Automated time series forecasting for biosurveillance. *Statistics in medicine*, 26(22):4202–4218, 2007.

[22] Ji-Won Byun, Ashish Kamra, Elisa Bertino, and Ninghui Li. Efficient k-anonymization using clustering techniques. In *Advances in Databases: Concepts, Systems and Applications*, pages 188–200. Springer, 2007.

[23] Jianneng Cao, Barbara Carminati, Elena Ferrari, and K-L Tan. Castle: Continuously anonymizing data streams. *Dependable and Secure Computing, IEEE Transactions on*, 8(3):337–352, 2011.

[24] Jianneng Cao, Barbara Carminati, Elena Ferrari, and Kian Lee Tan. Castle: A delay-constrained scheme for k s-anonymizing data streams. In *Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on*, pages 1376–1378. IEEE, 2008.

[25] Jianneng Cao, Panagiotis Karras, Panos Kalnis, and Kian-Lee Tan. Sabre: a sensitive attribute bucketization and redistribution framework for t-closeness. *The VLDB Journal*, 20(1):59–81, 2011.

[26] Don Carney, Ugur Çetintemel, Mitch Cherniack, Christian Convey, Sangdon Lee, Greg Seidman, Michael Stonebraker, Nesime Tatbul, and Stan Zdonik. Monitoring streams: a new class of data management applications. In *Proceedings of the 28th international conference on Very Large Data Bases*, pages 215–226. VLDB Endowment, 2002.

[27] Hutchatai Chanlekha and Nigel Collier. A methodology to enhance spatial understanding of disease outbreak events reported in news articles. *International journal of medical informatics*, 79(4):284–296, 2010.

[28] Shuchi Chawla, Cynthia Dwork, Frank McSherry, Adam Smith, and Hoeteck Wee. Toward privacy in public databases. In *Theory of Cryptography*, pages 363–385. Springer, 2005.

[29] William Cohen, Pradeep Ravikumar, and Stephen Fienberg. A comparison of string metrics for matching names and records. In *Kdd workshop on data cleaning and object consolidation*, volume 3, pages 73–78, 2003.

[30] Nigel Collier, Son Doan, Ai Kawazoe, Reiko Matsuda Goodwin, Mike Conway, Yoshio Tateno, Quoc-Hung Ngo, Dinh Dien, Asanee Kawtrakul, Koichi Takeuchi, et al. Biocaster: detecting public health rumors with a web-based text mining system. *Bioinformatics*, 24(24):2940–2941, 2008.

[31] Nigel Collier, Nguyen Truong Son, and Ngoc Mai Nguyen. Omg u got flu? analysis of shared health messages for bio-surveillance. *J. Biomedical Semantics*, 2(S-5):S9, 2011.

[32] Mike Conway, John N Dowling, and Wendy W Chapman. Using chief complaints for syndromic surveillance: A review of chief complaint based classifiers in north america. *Journal of biomedical informatics*, 46(4):734–743, 2013.

[33] Ellen K Cromley and Sara McLafferty. *GIS and public health.* Guilford Press, 2012.

[34] Monica Crubezy, Martin O'Connor, Zachary Pincus, Mark A Musen, and David L Buckeridge. Ontology-centered syndromic surveillance for bioterrorism. *Intelligent Systems, IEEE*, 20(5):26–35, 2005.

[35] Alexandre David, Kim G Larsen, Axel Legay, Marius Mikučionis, Danny Bøgsted Poulsen, Jonas Van Vliet, and Zheng Wang. Statistical model checking for networks of priced timed automata. In *Formal Modeling and Analysis of Timed Systems*, pages 80–96. Springer, 2011.

[36] Rinku Dewri, Indrakshi Ray, and David Whitley. Query m-invariance: Preventing query disclosures in continuous location-based services. In *Mobile Data Management (MDM), 2010 Eleventh International Conference on*, pages 95–104. IEEE, 2010.

[37] R Dicker, F Coronado, D Koo, and RG Parrish. Principles of epidemiology in public health practice. *Atlanta, GA: US Department of Health and Human Services. Centers for Disease Control and Prevention (CDC)*, 2006.

[38] Josep Domingo-Ferrer and Vicenç Torra. A critique of k-anonymity and some of its enhancements. In *Availability, Reliability and Security, 2008. ARES 08. Third International Conference on*, pages 990–993. IEEE, 2008.

[39] Cynthia Dwork. Differential privacy. In *Automata, languages and programming*, pages 1–12. Springer, 2006.

[40] Alexandre Evfimievski, Ramakrishnan Srikant, Rakesh Agrawal, and Johannes Gehrke. Privacy preserving mining of association rules. *Information Systems*, 29(4):343–364, 2004.

[41] Rick Falkvinge. How does privacy differ from anonymity, and why are both important? `https://www.privateinternetaccess.com/blog/2013/10/how-does-privacy-differ-from-anonymity-and-why-are-both-important/`, October 2013. Accessed: 2.September 2014.

[42] Lee Feigenbaum, Ivan Herman, Tonya Hongsermeier, Eric Neumann, and Susie Stephens. The semantic web in action. *Scientific American*, 297(6):90–97, 2007.

[43] Charles L Forgy. Rete: A fast algorithm for the many pattern/many object pattern match problem. *Artificial intelligence*, 19(1):17–37, 1982.

[44] Ronald D Fricker. *Introduction to Statistical Methods for Biosurveillance: With an Emphasis on Syndromic Surveillance*. Cambridge University Press, 2013.

[45] Benjamin CM Fung, Ke Wang, and Philip S Yu. Anonymizing classification data for privacy preservation. *Knowledge and Data Engineering, IEEE Transactions on*, 19(5):711–725, 2007.

[46] Mohamed Medhat Gaber, Arkady Zaslavsky, and Shonali Krishnaswamy. Mining data streams: a review. *ACM Sigmod Record*, 34(2):18–26, 2005.

[47] Robert Gaynes, Chesley Richards, Jonathan Edwards, T Grace Emori, Teresa Horan, Juan Alonso-Echanove, Scott Fridkin, Rachel Lawton, Gloria Peavy, and James Tolson. Feeding back surveillance data to prevent hospital-acquired infections. *Emerging infectious diseases*, 7(2):295, 2001.

[48] Johannes Gehrke, Flip Korn, and Divesh Srivastava. On computing correlated aggregates over continual data streams. In *ACM SIGMOD Record*, volume 30, pages 13–24. ACM, 2001.

[49] Robert R German, LM Lee, JM Horan, R Milstein, C Pertowski, M Waller, et al. Updated guidelines for evaluating public health surveillance systems. *MMWR Recomm Rep*, 50(RR-13):1–35, 2001.

[50] Aris Gkoulalas-Divanis, Grigorios Loukides, and Jimeng Sun. Publishing data from electronic health records while preserving privacy: A survey of algorithms. *Journal of biomedical informatics*, 50:4–19, 2014.

[51] Mita Sanghavi Goel, Christina C Wee, Ellen P McCarthy, Roger B Davis, Quyen Ngo-Metzger, and Russell S Phillips. Racial and ethnic disparities in cancer screening. *Journal of general internal medicine*, 18(12):1028–1035, 2003.

[52] Lukasz Golab and M Tamer Özsu. Issues in data stream management. *ACM Sigmod Record*, 32(2):5–14, 2003.

[53] Jane Grimson, Gaye Stephens, Benjamin Jung, William Grimson, Damon Berry, and Sebastien Pardon. Sharing health-care records over the internet. *Internet Computing, IEEE*, 5(3):49–58, 2001.

[54] Sudipto Guha, Nina Mishra, Rajeev Motwani, and Liadan O'Callaghan. Clustering data streams. In *Foundations of computer science, 2000. proceedings. 41st annual symposium on*, pages 359–366. IEEE, 2000.

[55] Xianmang He, Yanghua Xiao, Yujia Li, Qing Wang, Wei Wang, and Baile Shi. Dynamic anonymization for marginal publication. In *Scientific and Statistical Database Management*, pages 451–460. Springer, 2011.

[56] Kelly J Henning. What is syndromic surveillance? *Morbidity and Mortality Weekly Report*, pages 7–11, 2004.

[57] Allen Herman, Brian McCarthy, Janice Bakewell, Ryk Ward, Beth Mueller, Noreen Maconochie, Anne Read, Pnina Zadka, et al. Data linkage methods used in maternally-linked birth and infant death surveillance data sets from the united states (georgia, missouri, utah and washington state), israel, norway, scotland and western australia. *Paediatric and perinatal epidemiology*, 11(S1):5–22, 1997.

[58] Cory Janssen. Techopedia dictionary; definition - what does anonymization mean? `http://www.techopedia.com/definition/28007/anonymization-data`, August 2015. Accessed: 15.August 2015.

[59] Edwin T Jaynes. Information theory and statistical mechanics. *Physical review*, 106(4):620, 1957.

[60] Hillol Kargupta, Souptik Datta, Qi Wang, and Krishnamoorthy Sivakumar. On the privacy preserving properties of random data perturbation techniques. In *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*, pages 99–106. IEEE, 2003.

[61] Michel Klein. Xml, rdf, and relatives. *IEEE Intelligent Systems*, 16(2):26–28, 2001.

[62] Srdjan Komazec, Davide Cerri, and Dieter Fensel. Sparkwave: continuous schema-enhanced pattern matching over rdf data streams. In *Proceedings of the 6th ACM International Conference on Distributed Event-Based Systems*, pages 58–68. ACM, 2012.

[63] Danh Le-Phuoc, Minh Dao-Tran, Josiane Xavier Parreira, and Manfred Hauswirth. A native and adaptive approach for unified processing of linked streams and linked data. In *The Semantic Web–ISWC 2011*, pages 370–388. Springer, 2011.

[64] Danh Le-Phuoc, Josiane Xavier Parreira, and Manfred Hauswirth. *Linked stream data processing*. Springer, 2012.

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

[66] Kristen LeFevre, David J DeWitt, and Raghu Ramakrishnan. Multidimensional k-anonymity. In *Data Engineering, 2006. ICDE'06. Proceedings of the 22nd International Conference on*, pages 25–25. IEEE, 2006.

[67] Jianzhong Li, Beng Chin Ooi, and Weiping Wang. Anonymizing streaming data for privacy protection. 2008.

[68] Jin Li, David Maier, Kristin Tufte, Vassilis Papadimos, and Peter A Tucker. No pane, no gain: efficient evaluation of sliding-window aggregates over data streams. *ACM SIGMOD Record*, 34(1):39–44, 2005.

[69] Ninghui Li, Tiancheng Li, and Suresh Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and l-diversity. In *ICDE*, volume 7, pages 106–115, 2007.

[70] Ninghui Li, Tiancheng Li, and Suresh Venkatasubramanian. Closeness: A new privacy measure for data publishing. *Knowledge and Data Engineering, IEEE Transactions on*, 22(7):943–956, 2010.

[71] Tiancheng Li and Ninghui Li. Injector: Mining background knowledge for data anonymization. In *Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on*, pages 446–455. IEEE, 2008.

[72] Joseph Lombardo, Howard Burkom, Eugene Elbert, Steven Magruder, Sheryl Happel Lewis, Wayne Loschen, James Sari, Carol Sniegoski, Richard Wojcik, and Julie Pavlin. A systems overview of the electronic surveillance system for the early notification of community-based epidemics (essence ii). *Journal of urban health*, 80(1):i32–i42, 2003.

[73] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):3, 2007.

[74] Alejandro Mallea, Marcelo Arenas, Aidan Hogan, and Axel Polleres. On blank nodes. In *The Semantic Web–ISWC 2011*, pages 421–437. Springer, 2011.

[75] Kenneth D Mandl, J Marc Overhage, Michael M Wagner, William B Lober, Paola Sebastiani, Farzad Mostashari, Julie A Pavlin, Per H Gesteland, Tracee Treadwell, Eileen Koski, et al. Implementing syndromic surveillance: a practical guide informed by the early experience. *Journal of the American Medical Informatics Association*, 11(2):141–150, 2004.

[76] Julienne Meyer. Qualitative research in health care: Using qualitative methods in health related action research. *BMJ: British Medical Journal*, 320(7228):178, 2000.

[77] Adam Meyerson and Ryan Williams. On the complexity of optimal k-anonymity. In *Proceedings of the twenty-third ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 223–228. ACM, 2004.

[78] Noman Mohammed, Benjamin Fung, Patrick CK Hung, and Cheuk-kwong Lee. Anonymizing healthcare data: a case study on the blood transfusion service. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1285–1294. ACM, 2009.

[79] Rajeev Motwani, Jennifer Widom, Arvind Arasu, Brian Babcock, Shivnath Babu, Mayur Datar, Gurmeet Manku, Chris Olston, Justin Rosenstein, and Rohit Varma. Query processing, resource management, and approximation in a data stream management system. CIDR, 2003.

[80] Rajeev Motwani and Ying Xu. Efficient algorithms for masking and finding quasi-identifiers. In *Proceedings of the Conference on Very Large Data Bases (VLDB)*, pages 83–93, 2007.

[81] Mark A Musen, Blackford Middleton, and Robert A Greenes. Clinical decision-support systems. In *Biomedical informatics*, pages 643–674. Springer, 2014.

[82] MVR NarasimhaRao, JS VenuGopalkrisna, RNV Vishnu Murthy, and Ch Raja Ramesh. Closeness: privacy measure for data publishing using multiple sensitive attributes. *Heart*, 2(2):2, 2012.

[83] M Ercan Nergiz and Chris Clifton. Thoughts on< i> k</i>-anonymization. *Data & Knowledge Engineering*, 63(3):622–645, 2007.

[84] Mehmet Ercan Nergiz, Maurizio Atzori, and Yucel Saygin. Towards trajectory anonymization: a generalization-based approach. In *Proceedings of the SIGSPATIAL ACM GIS 2008 International Workshop on Security and Privacy in GIS and LBS*, pages 52–61. ACM, 2008.

[85] Quoc-Hung Ngo, Son Doan, and Werner Winiwarter. Using wikipedia for extracting hierarchy and building geo-ontology. *International Journal of Web Information Systems*, 8(4):401–412, 2012.

[86] Timo Niemi, Turkka Näppilä, and Kalervo Järvelin. A relational data harmonization approach to xml. *Journal of Information Science*, 2009.

[87] Liadan O'callaghan, Adam Meyerson, Rajeev Motwani, Nina Mishra, and Sudipto Guha. Streaming-data algorithms for high-quality clustering. In *2013 IEEE 29th International Conference on Data Engineering (ICDE)*, pages 0685–0685. IEEE Computer Society, 2002.

[88] House of Lords. Surveillance: Citizens and the state. `http://www.publications.parliament.uk/pa/ld200809/ldselect/ldconst/18/1805.htm`, 2009. [Online; accessed: 2015-05-18].

[89] Jeff Offutt and Wuzhi Xu. Generating test cases for web services using data perturbation. *ACM SIGSOFT Software Engineering Notes*, 29(5):1–10, 2004.

[90] Martin Oconnor, Holger Knublauch, Samson Tu, Benjamin Grosof, Mike Dean, William Grosso, and Mark Musen. Supporting rule system interoperability on the semantic web with swrl. In *The Semantic Web–ISWC 2005*, pages 974–986. Springer, 2005.

[91] Martin OConnor, Ravi Shankar, Samson Tu, Csongor Nyulas, Dave Parrish, Mark Musen, and Amar Das. Using semantic web technologies for knowledge-driven querying of biomedical data. In *Artificial Intelligence in Medicine*, pages 267–276. Springer, 2007.

[92] Martin J OConnor, David L Buckeridge, Michael Choy, Monica Crubezy, Zachary Pincus, and Mark A Musen. Biostorm: a system for automated surveillance of diverse data sources. In *AMIA Annual Symposium Proceedings*, volume 2003, page 1071. American Medical Informatics Association, 2003.

[93] Kostas Patroumpas and Timos Sellis. Window specification over data streams. In *Current Trends in Database Technology–EDBT 2006*, pages 445–464. Springer, 2006.

[94] Jorge Pérez, Marcelo Arenas, and Claudio Gutierrez. Semantics and complexity of sparql. In *The Semantic Web-ISWC 2006*, pages 30–43. Springer, 2006.

[95] Vibhor Rastogi, Dan Suciu, and Sungho Hong. The boundary between privacy and utility in data publishing. In *Proceedings of the 33rd international conference on Very large data bases*, pages 531–542. VLDB Endowment, 2007.

[96] Thomas C Rindfleisch. Privacy, information technology, and health care. *Communications of the ACM*, 40(8):92–100, 1997.

[97] Andrea L Roberts, Stephen E Gilman, Joshua Breslau, Naomi Breslau, and Karestan C Koenen. Race/ethnic differences in exposure to traumatic events, development of post-traumatic stress disorder, and treatment-seeking for post-traumatic stress disorder in the united states. *Psychological medicine*, 41(01):71–83, 2011.

[98] D Robins. Complex event processing. In *Second International Workshop on Education Technology and Computer Science. Wuhan*, 2010.

[99] Henry Rolka, Howard Burkom, Gregory F Cooper, Martin Kulldorff, David Madigan, and Weng-Keen Wong. Issues in applied statistics for public health bioterrorism surveillance using multiple data streams: research needs. *Statistics in Medicine*, 26(8):1834–1856, 2007.

[100] Josep Roure, Artur Dubrawski, and Jeff Schneider. A study into detection of bio-events in multiple streams of surveillance data. In *Intelligence and Security Informatics: Biosurveillance*, pages 124–133. Springer, 2007.

[101] Owen Sacco and Alexandre Passant. A privacy preference ontology (ppo) for linked data. In *LDOW*. Citeseer, 2011.

[102] Pierangela Samarati. Protecting respondents identities in microdata release. *Knowledge and Data Engineering, IEEE Transactions on*, 13(6):1010–1027, 2001.

[103] Bill Schilit, Norman Adams, and Roy Want. Context-aware computing applications. In *Mobile Computing Systems and Applications, 1994. WMCSA 1994. First Workshop on*, pages 85–90. IEEE, 1994.

[104] Daniela Schmid. Nationale referenzzentrale für influenza - epidemiologie. `http://bmg.gv.at/cms/home/attachments/5/8/4/CH1435/CMS1410260270112/jb_influenza_2013.pdf`, 2014. [Online; accessed: 2015-06-08].

[105] Juan F Sequeda and Oscar Corcho. Linked stream data: A position paper. 2009.

[106] Galit Shmueli and Stephen E Fienberg. Current and potential statistical methods for monitoring multiple data streams for biosurveillance. In *Statistical Methods in Counterterrorism*, pages 109–140. Springer, 2006.

[107] Edward H Shortliffe and G Octo Barnett. Medical data: their acquisition, storage, and use. In *Medical Informatics*, pages 41–75. Springer, 2001.

[108] Spiros Simitis. Reviewing privacy in an information society. *University of Pennsylvania Law Review*, pages 707–746, 1987.

[109] Daniel M Sosin. Draft framework for evaluating syndromic surveillance systems. *Journal of urban health*, 80(1):i8–i13, 2003.

[110] Alex J Sutton, Keith R Abrams, David R Jones, David R Jones, Trevor A Sheldon, and Fujian Song. *Methods for meta-analysis in medical research*. J. Wiley, 2000.

[111] Latanya Sweeney. Achieving k-anonymity privacy protection using generalization and suppression. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05):571–588, 2002.

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

[113] Douglas Terry, David Goldberg, David Nichols, and Brian Oki. *Continuous queries over append-only databases*, volume 21. ACM, 1992.

[114] Steven M Teutsch and R Elliott Churchill. *Principles and practice of public health surveillance*. Oxford University Press, 2000.

[115] Stephen B Thacker. Historical development. *Principles and practice of public health surveillance*, 2:1–15, 2000.

[116] Stephen B Thacker, R Gibson Parrish, and Frederick L Trowbridge. A method for evaluating systems of epidemiological surveillance. *World health statistics quarterly. Rapport trimestriel de statistiques sanitaires mondiales*, 41(1):11–18, 1987.

[117] Traian Marius Truta and Bindu Vinay. Privacy protection: p-sensitive k-anonymity property. In *ICDE Workshops*, page 94, 2006.

[118] Fu-Chiang Tsui, Jeremy U Espino, Virginia M Dato, Per H Gesteland, Judith Hutman, and Michael M Wagner. Technical description of rods: a real-time public health surveillance system. *Journal of the American Medical Informatics Association*, 10(5):399–408, 2003.

[119] Jaideep Vaidya, Christopher W Clifton, and Yu Michael Zhu. *Privacy preserving data mining*, volume 19. Springer, 2006.

[120] Vassilios S Verykios, Elisa Bertino, Igor Nai Fovino, Loredana Parasiliti Provenza, Yucel Saygin, and Yannis Theodoridis. State-of-the-art in privacy preserving data mining. *ACM Sigmod Record*, 33(1):50–57, 2004.

[121] Michael M Wagner, Andrew W Moore, and Ron M Aryel. *Handbook of biosurveillance*. Academic Press, 2011.

[122] Pu Wang, Jianjiang Lu, Lei Zhao, and Jiwen Yang. B-castle: An efficient publishing algorithm for k-anonymizing data streams. In *Intelligent Systems (GCIS), 2010 Second WRI Global Congress on*, volume 2, pages 132–136. IEEE, 2010.

[123] Weiping Wang, Jianzhong Li, Chunyu Ai, and Yingshu Li. Privacy protection on sliding window of data streams. In *Collaborative Computing: Networking, Applications and Worksharing, 2007. CollaborateCom 2007. International Conference on*, pages 213–221. IEEE, 2007.

[124] Al Weber and Ron Thomas. Key performance indicators. *Measuring and Managing the Maintenance Function, Ivara Corporation, Burlington*, 2005.

[125] Raymond Chi-Wing Wong, Jiuyong Li, Ada Wai-Chee Fu, and Ke Wang. (α, k)-anonymity: an enhanced k-anonymity model for privacy preserving data publishing. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 754–759. ACM, 2006.

[126] Eugene Wu, Yanlei Diao, and Shariq Rizvi. High-performance complex event processing over streams. In *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, pages 407–418. ACM, 2006.

[127] Xiaokui Xiao and Yufei Tao. Anatomy: Simple and effective privacy preservation. In *Proceedings of the 32nd international conference on Very large data bases*, pages 139–150. VLDB Endowment, 2006.

[128] Xiaokui Xiao and Yufei Tao. M-invariance: towards privacy preserving re-publication of dynamic datasets. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 689–700. ACM, 2007.

[129] Xiaokui Xiao and Yufei Tao. Dynamic anonymization: accurate statistical analysis with privacy preservation. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 107–120. ACM, 2008.

[130] Ping Yan, Daniel Zeng, and Hsinchun Chen. A review of public health syndromic surveillance systems. In *Intelligence and Security Informatics*, pages 249–260. Springer, 2006.

[131] Linda J Young and Carol A Gotway. Linking spatial data from different sources: the effects of change of support. *Stochastic Environmental Research and Risk Assessment*, 21(5):589–600, 2007.

[132] Q Zeng and James J Cimino. Evaluation of a system to identify relevant patient information and its impact on clinical information retrieval. In *Proceedings of the AMIA Symposium*, page 642. American Medical Informatics Association, 1999.

[133] Junwei Zhang, Jing Yang, Jianpei Zhang, and Yongbin Yuan. Kids: k-anonymization data stream base on sliding window. In *Future Computer and Communication (ICFCC), 2010 2nd International Conference on*, volume 2, pages V2–311. IEEE, 2010.

[134] Bin Zhou, Yi Han, Jian Pei, Bin Jiang, Yufei Tao, and Yan Jia. Continuous privacy preserving publishing of data streams. In *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology*, pages 648–659. ACM, 2009.