ARX - A Comprehensive Tool for Anonymizing Biomedical Data

Fabian Prasser[12], Florian Kohlmayer[12], Ronald Lautenschläger[1], Klaus A. Kuhn[1]
[1] Technische Universität München, München, Germany, [2] Equal contributors

Abstract

Collaboration and data sharing have become core elements of biomedical research. Especially when sensitive data from distributed sources are linked, privacy threats have to be considered. Statistical disclosure control allows the protection of sensitive data by introducing fuzziness. Reduction of data quality, however, needs to be balanced against gains in protection. Therefore, tools are needed which provide a good overview of the anonymization process to those responsible for data sharing. These tools require graphical interfaces and the use of intuitive and replicable methods. In addition, extensive testing, documentation and openness to reviews by the community are important. Existing publicly available software is limited in functionality, and often active support is lacking. We present ARX, an anonymization tool that i) implements a wide variety of privacy methods in a highly efficient manner, ii) provides an intuitive cross-platform graphical interface, iii) offers a programming interface for integration into other software systems, and iv) is well documented and actively supported.

Introduction

Collaboration and data sharing have become core elements of biomedical research; see, e.g., the joint statement on sharing research data[1] and the guidelines for access to research data of the *Organization for Economic Co-operation and Development* (OECD)[2]. For clinical data, and increasingly for genetic data as well as for combinations of these data, there is a growing understanding of related privacy threats. Disclosure of data may lead to harm for individuals, especially when different data sources are available for linkage[3]. From the legal perspective, national laws, e.g., the *Health Insurance Portability and Accountability Act* (HIPAA)[4] *Privacy Rule*, and international regulations, e.g., the *European Directive on Data Protection*[5], mandate stringent protection of personal data.

The HIPAA Privacy Rule[6] defines two basic methods for de-identifying datasets. The first requires the removal of a pre-defined set of attributes from the dataset. This procedure significantly reduces re-identification risks[7], but it can *1)* obstruct data use if the involved attributes are essential[8], and *2)* under certain circumstances not prevent re-identification[9]. The second method is "expert determination": A professional "determines that the risk is very small that the information could be used […] to identify an individual"[6]. In this context, *statistical disclosure control* (SDC) allows balancing privacy risks with data quality[8]. Examples include methods focusing on data extracts, such as *differential privacy*[10], and methods for microdata release, such as *k-anonymity*[11].

In the biomedical domain, methods for microdata release are currently preferred[12]. The primary reason for this is that they can be implemented with non-pertubative methods that preserve the truthfulness of data[12]. They have been included in guidelines of best-practices for de-identifying health data[13], and they have been successfully applied to research data[14]. Moreover, many approaches for microdata release have been developed specifically for the biomedical domain[15,16,17].

Objectives

Although anonymization is an important method for privacy protection, there is a lack of tools which are both comprehensive and readily available to informatics researchers and also to non-IT experts, e.g., researchers responsible for the sharing of data. As protection has to be balanced against losses in data utility, responsible researchers should be able to keep an overview of the anonymization process and the trade-offs chosen. This requires powerful but easy to use tools which can be integrated in research workflows. *Graphical user interfaces* (GUIs) and the option of using a wide variety of intuitive and replicable methods are needed. Tools have to offer interfaces allowing their integration into pipelines comprising further data processing modules. Moreover, extensive testing, documentation and openness to reviews by the community are of high importance. Informatics researchers who want to use or evaluate existing anonymization methods or to develop novel methods will benefit from well-documented, open-source software libraries. The lack of such a framework is illustrated by the fact that, although data anonymization has been researched for a long period of time already, only recently efforts have started to systematically evaluate and compare existing methods[18,19].

The landscape of existing tools is heterogeneous. *PARAT*[20] is the leading commercial de-identification software. It is a closed-source tool for which only limited information is available to the public. We will focus on non-commercial tools in the remainder of this article. The *UTD Anonymization Toolbox*[21] and the *Cornell Anonymization Toolkit* (CAT)[22] are research prototypes that have mainly been developed for demonstration purposes. Problems with these tools include scalability issues when handling large datasets, complex configuration requiring IT-expertise, and incomplete support of privacy criteria and methods of data transformation. *sdcMirco*[23] is a package for the *R* statistics software, which implements many primitives required for data anonymization but offers only a limited support for using them to find data transformations that are suitable for a specific context. *μ-Argus*[24] is a closed-source application that implements a broad spectrum of techniques, but it is no longer under active development. A comparison of our work with these tools is presented in the *Discussion* section.

To overcome these limitations we present ARX, a comprehensive open-source data anonymization framework that implements a simple three-step process. It provides support for all common privacy criteria, as well as for arbitrary combinations. It utilizes a well-known and highly efficient anonymization algorithm. Moreover, it implements a carefully chosen set of techniques that can handle a broad spectrum of data anonymization tasks, while being efficient, intuitive and easy to understand. Our tool features a cross-platform user interface that is oriented towards non-IT experts. Additionally, it provides a stand-alone software library with an easy-to-use public *application programming interface* (API) for integration into other systems. Our code base is extensible, well-tested and extensively documented. As such, it provides a solid basis for developing novel privacy methods.

Background and Terminology

In the context of SDC, data is organized in a tabular structure, where each *record* represents the data about one individual. Basically, *identifiers* within a dataset are modified in such a way that linkage is prevented[25]: Firstly, highly distinguishing attributes that can be used for re-identification and that are not required for analyses are removed. Examples for such *direct identifiers* include *Social Security numbers* or names. Secondly, *quasi-identifiers*, i.e., attributes which are required for analyses while being potentially identifying are *recoded* to make sure that the data fulfills well-known *privacy criteria*[25]. Examples include the age or sex of data subjects. Data recoding is typically performed with *generalization hierarchies*[11], which can be built for categorical and continuous attributes (see Figure 1). To increase the utility of resulting datasets, this method is often combined with *tuple suppression*: data records that violate the privacy criteria (so called *outliers*) are automatically removed from the dataset, while the total number of suppressed records is kept under a given threshold[11]. As a result, less generalization is required to ensure that the remaining records fulfill the privacy criteria.

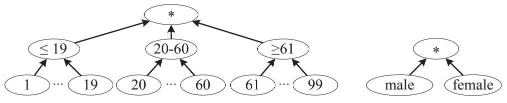


Figure 1. Hierarchies for attributes "*age*" and "*sex*"

k-Anonymity[11] is the most wide-spread privacy criterion. It ensures that each data record cannot be distinguished from at least k-1 other data records regarding the quasi-identifiers. It aims at protecting datasets against *identity disclosure*, i.e., from linking an individual to a specific data record by an attacker[26]. *ℓ-Diversity*[27] and *t-closeness*[28] aim at protecting datasets against *attribute disclosure*, where an attacker can infer information about an individual without necessarily linking it to a specific record in the dataset[26]. As an example, linkage to a set of records allows inferring information if all records share a certain attribute value. The attributes in which an attacker might be interested and which, if disclosed, could cause harm to the data subject are called *sensitive attributes*. Different variants exist for both criteria, which offer different privacy guarantees by enforcing different properties on the distributions of sensitive attribute values within a set of indistinguishable data records. *δ-Presence*[29] aims at protecting datasets against *membership disclosure*, which means that an attacker is able to determine whether or not data about an individual is contained in a dataset[26]. For an overview of further privacy criteria we refer to the work by Fung et al.[25].

Different types of algorithms can be used to transform datasets so that they fulfill a given set of privacy criteria. For the biomedical domain, the use of *globally-optimal full-domain anonymization* algorithms using *multi-dimensional global recoding* has been recommended[30]. These algorithms construct a search space, which consists of all possible combinations of generalization levels for all quasi-identifying attributes. This space of possible generalizations is then traversed to find a transformation that fulfills all privacy criteria while resulting in optimal data quality. To this end, utility is measured with *metrics* for information loss[25].

Methods

Our work aims at making data anonymization available to a wide variety of end users. We therefore decided to implement a type of algorithm that is intuitive to non-IT experts: a globally-optimal full-domain anonymization algorithm that uses multi-dimensional global recoding[17]. On the upside, such algorithms implement anonymization procedures that can easily be configured by users, e.g., by altering generalization hierarchies or choosing a suitable transformation from the solution space, and result in datasets that are well suited for biomedical analyses[30]. On the downside, the underlying coding model is strict and potentially results in low data utility. To attenuate this, our framework combines the method with local tuple suppression. It increases data quality, but can also significantly increase execution times. Consequently, our tool is also able to approximate the result in much less time. Approximated results are guaranteed to fulfill the given criteria but might not be optimal in terms of data quality, because only some transformations in the solution space are classified with absolute certainty.

In order to cover a broad spectrum of privacy problems, our tool comes with implementations of all commonly used privacy methods: k-anonymity, all variants of ℓ-diversity, two variants of t-closeness, and δ-presence. A querying interface allows selecting a research subset, which is a subset of the data records that are to be included in the final anonymized dataset. Moreover, δ-presence can be enforced on the subset, i.e., attackers that know the overall dataset can be prevented from determining whether a specific tuple is included in the subset. In contrast to previous approaches our tool is the first to support classifying the complete solution space for arbitrary combinations of privacy criteria while using generalization *and* suppression. For assessing data utility, we included a large set of metrics for information loss, including simple methods, such as *height* and *precision* as well as more sophisticated approaches, such as *discernibility*[31] and *non-uniform entropy*[30].

Data and generalization hierarchies can be imported from many different types of sources in order to provide compatibility with a wide range of data processing tools. ARX currently features interfaces for importing data from *character-separated values* (CSV) files, *MS Excel* spreadsheets and *relational database management systems* (RDBMSs), such as *PostgreSQL* or *MySQL*. Data imported into ARX is immutable and cannot be changed. ARX implements several methods that can be used for identifying data quality issues: *1)* data can be sorted, compared and analyzed regarding statistical properties, and *2)* the query interface can be used to search for records with quality issues. For handling such issues, ARX supports the automated and manual removal of records. If more complex data cleansing tasks must be performed, data can be exported to other software systems. ARX safely handles missing values by treating them similar to all other values (i.e., missing values match other missing values)[32]. This scheme allows for truthful and non-truthful handling of missing values, depending on how they are generalized, without introducing privacy issues[32].

ARX offers methods for manual and semi-automatic creation of generalization hierarchies. Semi-automatic creation is supported for all common types of attributes, such as numerical (discrete or continuous) and categorical variables. Hierarchies can be generated by grouping values based on a natural or user-defined ordering, by mapping them into user-defined or automatically created intervals or by data redaction. Hierarchies are represented in a tabular manner, which is an intuitive representation that enables compatibility with third-party applications, such as spreadsheet programs.

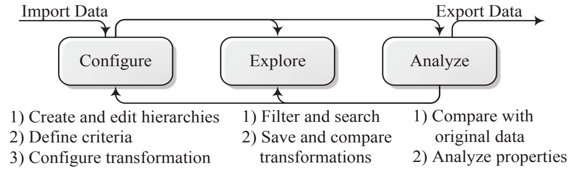

Import Data
Export Data
Configure
Explore
Analyze
1) Create and edit hierarchies
2) Define criteria
3) Configure transformation
1) Filter and search
2) Save and compare transformations
1) Compare with original data
2) Analyze properties

Figure 2. Implemented de-identification workflow

ARX supports all major aspects of data anonymization. These different aspects are combined in a multi-step process that allows users to iteratively adjust parameters, until the result matches their requirements. As depicted in Figure 2, the basic steps consist of *1)* configuring the anonymization process, *2)* exploring the solution space and *3)* analyzing the transformed data. In the configuration phase, the input data is loaded, generalization hierarchies are imported or created and all further parameters, such as privacy criteria, are specified. When the solution space has been characterized by executing the anonymization algorithm, the exploration phase allows searching the solution space for privacy-preserving data transformations that fulfill the user's requirements. To assess suitability, the analysis phase allows comparing transformed datasets to the original input dataset. Based on this analysis, further solution candidates might be considered and analyzed, or the configuration of the anonymization process might be altered. Our tool features a cross-platform graphical interface for non-IT experts. All methods implemented in ARX are accessible via the API and the GUI.

An important goal of our efforts is to make the anonymization framework, consisting of the graphical application and the software library, available to software developers and informatics researchers. To this end, we chose to implement it in the *Java* ecosystem, which offers one of the most popular cross-platform development environments. We chose to implement the GUI with the *Standard Widget Toolkit* (SWT) and the *JFace* library, which provide support for the native *look and feels* of the three most common platforms: *OS X*, *Windows* and

Linux/GTK. We spent extensive time on documenting our code as well as the public API and on adding a large set of examples to our code base. Our project has a high test coverage, featuring hundreds of unit tests with different configurations and a broad spectrum of input datasets, including a large set of tests for which the results were validated manually. Our website provides users with background information and extensive documentation[33].

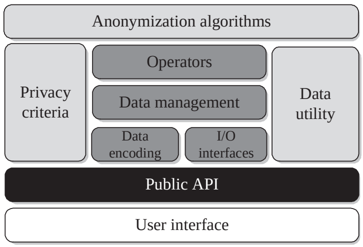


Figure 3. High-level architecture

A high-level overview of the architecture of ARX is shown in Figure 3. We carefully designed the framework, to prevent tight coupling of subsystems and ensure extensibility. The core modules (dark gray), which normally need not be modified, provide the basis for our framework. The *I/O module* provides methods for reading data from and writing data to external storage, while the *data encoding module* transforms the data into the format and memory layout required by the framework. The *data management module* deals with this internal representation and implements several optimizations (e.g., caching). Problem-specific *operators* are built on top of this representation and allow grouping of data records and computing of frequency distributions over sensitive attribute values. This provides the basis for extensible modules (light gray), which implement *privacy criteria* and metrics for measuring *data utility*. Analogously, anonymization algorithms can be plugged into the framework. Currently our tool features several variants of the *Flash* algorithm but the framework can be used to implement a large set of methods[18]. The public API is based on both the extensible and the core modules. It is also used by the graphical interface, which is completely decoupled from the internals of the framework.

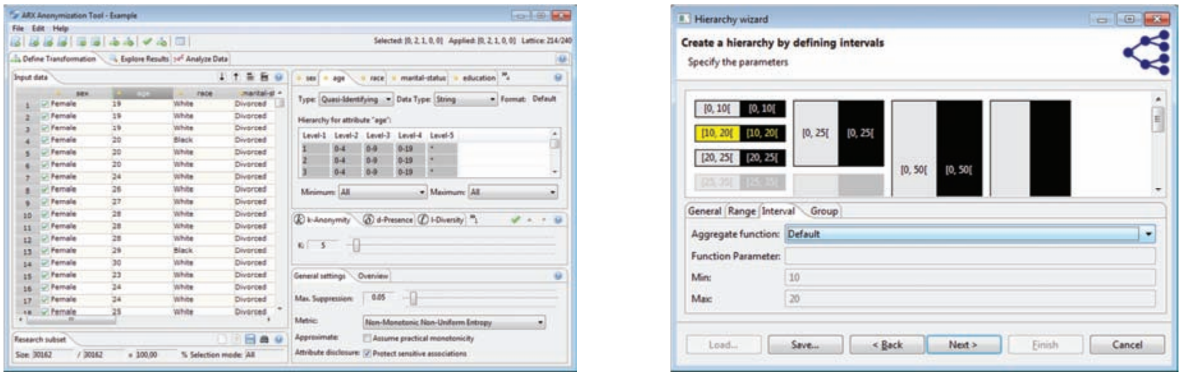
Our three-step process poses considerable challenges in terms of efficiency. Firstly, ARX automatically classifies the complete solution space to support users in finding a transformation suitable for their application scenario. Secondly, the iterative character of the process potentially requires this classification to be performed repeatedly. It is thus very important that classification can be carried out in near real-time. For this purpose, our framework features a highly efficient algorithm[17]. Moreover, instead of using existing database systems, we implemented a runtime environment that is tailored to the problem domain. In previous work we have shown that our method significantly outperforms comparable algorithms within our highly optimized framework[18].

Results

In this section, we present an overview of the graphical interface for end-users which illustrates ARX's functionality. We then address the public API for researchers and software developers. Finally, we shortly analyze the scalability of our tool in terms of execution time and memory requirements.

Importing data and configuring the de-identification process

The graphical interface of ARX is divided into three perspectives that follow the workflow outlined in the previous section. In the first perspective, a dataset can be imported and the anonymization process can be configured.



Selected: [0, 2, 1, 0, 0] Applied: [0, 2, 1, 0, 0] Lattice: 214/240

	sex	age	race	marital-st
1	☑ Female	19	White	Divorced
2	☑ Female	19	White	Divorced
3	☑ Female	19	White	Divorced
4	☑ Female	20	Black	Divorced
5	☑ Female	20	White	Divorced
6	☑ Female	20	White	Divorced
7	☑ Female	24	White	Divorced
8	☑ Female	26	White	Divorced
9	☑ Female	27	White	Divorced
10	☑ Female	28	White	Divorced
11	☑ Female	28	White	Divorced
12	☑ Female	28	White	Divorced
13	☑ Female	29	Black	Divorced
14	☑ Female	30	White	Divorced
15	☑ Female	23	White	Divorced
16	☑ Female	24	White	Divorced
17	☑ Female	24	White	Divorced
18	☑ Female	25	White	Divorced

Size: 30162 / 30162 = 100,00 % Selection mode: All

Type: Quasi-Identifying Data Type: String Format: Default

Hierarchy for attribute "age":

Level-1	Level-2	Level-3	Level-4	Level-5
1	0-4	0-9	0-19	*
2	0-4	0-9	0-19	*
3	0-4	0-9	0-19	*

Minimum: All Maximum: All

k: 5

Max. Suppression: 0.05

Metric: Non-Monotonic Non-Uniform Entropy

Approximate: □ Assume practical monotonicity

Attribute disclosure: ☑ Protect sensitive associations

Create a hierarchy by defining intervals

Specify the parameters

[0, 10[[0, 10[[10, 20[[10, 20[[20, 25[[20, 25[[25, 35[[25, 35[[0, 25[[0, 25[[0, 50[[0, 50[

Aggregate function: Default

Function Parameter:

Min: 10

Max: 20

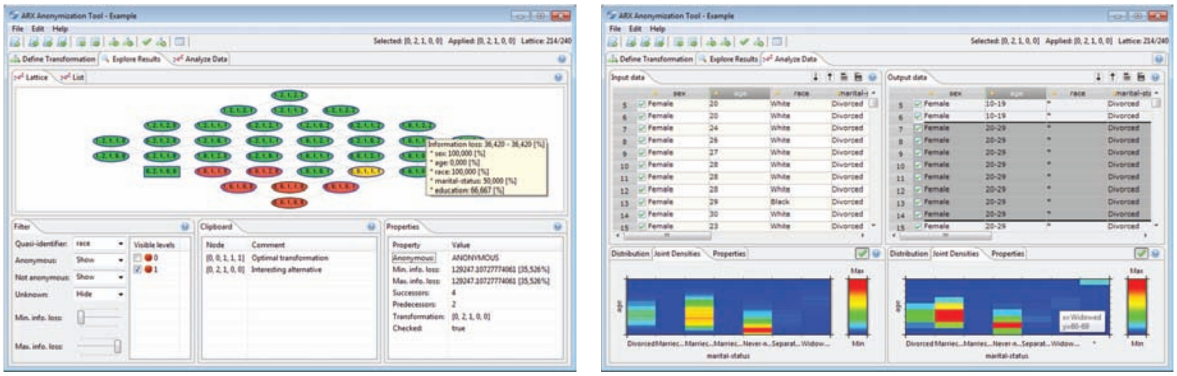
Figure 4. Interfaces for configuring the de-identification process and creating generalization hierarchies

A screenshot is presented in Figure 4. On the left-hand side, the imported dataset is displayed. Individual records can be included or excluded via checkboxes to define a research subset. An overview of the current subset is presented in the lower left area of the perspective. The header of the table showing the dataset also indicates the attribute type associated to each column in terms of a color scheme. These attribute types can be defined in the upper right area of the perspective: directly identifying attributes are removed from the dataset, quasi-identifiers are transformed by applying the provided hierarchies, while sensitive attributes are not transformed but can be used to derive t-close or ℓ-diverse transformations. *Insensitive* attributes are simply kept unchanged. The area also displays a tabular representation of the generalization hierarchies associated to the attributes. Hierarchies can be edited manually, e.g., by adding and removing generalization levels or altering labels. The lower right area of the perspective allows for defining privacy criteria and for configuring the transformation and classification process. Important aspects include selecting the metric that is to be used for assessing data utility and defining an upper bound on the number of records that can be suppressed. When all parameters have been configured, the solution space can be classified.

ARX offers several wizards for semi-automatically creating generalization hierarchies. The wizard for interval-based hierarchies is shown on the right side of Figure 4. Intervals are a natural means of generalization for variables with a ratio scale, such as integers, decimals or dates and timestamps. Each level of the hierarchy is represented by one column, with the lowest level being defined by a sequence of intervals (leftmost column). Subsequent levels can be added by grouping any given number of elements from the previous level. Any sequence of intervals or groups is automatically repeated to cover a predefined range. For values outside of this range, *top and bottom coding* can be applied. Automatically created repetitions of intervals and groups are indicated visually.

Exploring the solution space

When the solution space has been classified, the exploration perspective allows users to browse the set of all possible transformations. The aim of the perspective, a screenshot of which is presented on the left-hand side in Figure 5, is to select a set of interesting transformations for analysis.



Selected: [0, 2, 1, 0, 0] Applied: [0, 2, 1, 0, 0] Lattice: 214/240

Information loss: 36,420 - 36,420 [%]
* sex: 100,000 [%]
* age: 0,000 [%]
* race: 100,000 [%]
* marital-status: 50,000 [%]
* education: 66,667 [%]

Quasi-identifier: race
Anonymous: Show
Not anonymous: Show
Unknown: Hide
Min. info. loss:
Max. info. loss:

Visible levels
☐ ● 0
☑ ● 1

Node	Comment
[0, 0, 1, 1, 1]	Optimal transformation
[0, 2, 1, 0, 0]	Interesting alternative

Property	Value
Anonymous:	ANONYMOUS
Min. info. loss:	129247.10727774061 [35.526%]
Max. info. loss:	129247.10727774061 [35.526%]
Successors:	4
Predecessors:	2
Transformation:	[0, 2, 1, 0, 0]
Checked:	true

Input data

	sex	age	race	marital-s
5	Female	20	White	Divorced
6	Female	20	White	Divorced
7	Female	24	White	Divorced
8	Female	26	White	Divorced
9	Female	27	White	Divorced
10	Female	28	White	Divorced
11	Female	28	White	Divorced
12	Female	28	White	Divorced
13	Female	29	Black	Divorced
14	Female	30	White	Divorced
15	Female	23	White	Divorced

Output data

	sex	age	race	marital-sta
5	Female	10-19	*	Divorced
6	Female	10-19	*	Divorced
7	Female	20-29	*	Divorced
8	Female	20-29	*	Divorced
9	Female	20-29	*	Divorced
10	Female	20-29	*	Divorced
11	Female	20-29	*	Divorced
12	Female	20-29	*	Divorced
13	Female	20-29	*	Divorced
14	Female	20-29	*	Divorced
15	Female	20-29	*	Divorced

Figure 5. Interfaces for exploring the solution space and analyzing the results of data transformations

In the center of the screen, the view displays a subset of the solution space. Each node represents one transformation, which is identified by the generalization levels that it defines for the quasi-identifiers in the input dataset. Transformations are characterized by four different background colors: green denotes that the transformation results in an anonymous dataset, red denotes that the transformation does not result in an anonymous dataset, orange denotes that the transformation is the global optimum and gray denotes that the anonymity property of the transformations is unknown (only applies when approximating the result). The view supports zooming and moving the visualization of the solution space. Interesting transformations can be added to a clipboard, where they can be organized. Applying a transformation to the dataset allows exporting the resulting dataset or analyzing it in the view presented in the next section. A filter allows selecting a subset of the solution space by defining that only transformations with certain generalization levels, certain anonymity properties or with their information loss being within a defined interval should be visible. In case of an approximated result, this also includes transformations which are probably anonymous or probably non-anonymous. If such a transformation is applied, its actual anonymity property will be computed in the background and the state of the solution space will be updated.

Analyzing transformed data

A given privacy problem can often be solved with several different transformations. Although ARX is able to automatically find a solution which is optimal regarding the selected metric for data utility, automatically choosing an appropriate solution for a given usage scenario is often difficult. The aim of this perspective, which is shown on the right-hand side in Figure 5, therefore is to support users in assessing the utility of a transformed dataset for a specific application scenario. For this purpose, it allows comparing transformed data to the original dataset.

The perspective displays the input dataset on the left and the output dataset on the right. The tables are synchronized when scrolling, allowing an easy comparison of different parts of the data. The data can be sorted according to selected attributes. It is possible to toggle between showing the whole dataset or only the research subset. Moreover, the resulting equivalence classes can be highlighted. For comparing the statistical properties of two data representations, the view also shows the frequency distributions of values of a selected attribute in both tables. Moreover, a graphical representation of contingency tables for two selected attributes is included. This feature allows for visually comparing the combined occurrence of values from two different attributes. In the example, the number of values of the attribute “marital-status” remains the same, while the number of values of the attribute “age” is reduced from 100 to 10. Additionally, some values of the attribute “age” and exactly one value of the attribute “marital-status” have been suppressed. This is reflected by the heat map on the right-hand side, which shows a slightly shifted generalization of the contingency from the input dataset.

Programmatic access via the API

All features that are accessible via the graphical interface are also accessible via the public API. However, the aim of the programming interface is to provide de-identification methods to other software systems and we note that interaction with the software library provided by ARX will often be simpler than interaction with the graphical tool. Programmatic access will usually rely on ARX’s ability to automatically determine a solution to a privacy problem.

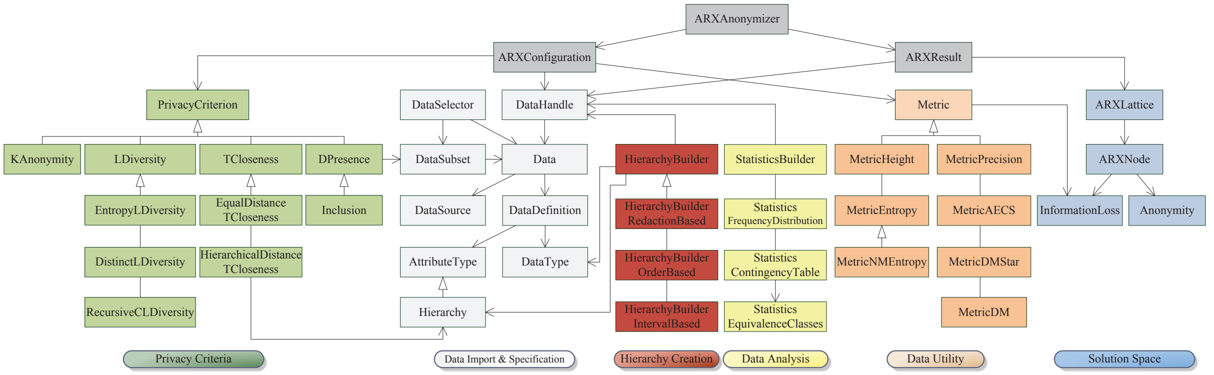


Figure 6. UML diagram of the most important classes in the public API

A *Unified Modeling Language* (UML) diagram of the most important classes of our API is shown in Figure 6. It can be seen that the API consists of a set of packages for *1) data import and specification*, *2) hierarchy generation*, *3) privacy criteria*, *4)* measuring *data utility*, *5) data analysis*, and *6)* representing the *solution space*. The classes `ARXConfiguration`, `ARXAnonymizer` and `ARXResult` provide the main interfaces to ARX’s functionalities.

Data and attribute types are provided as static variables from the `DataType` and `AttributeType` classes respectively. The class `DataHandle` allows users to interact with the data (read-only), by performing operations such as sorting, swapping rows or reading cell values. Handles can be obtained for input data, output data and research subsets of such data. Data handles representing input and derived output data are linked with each other, meaning that operations performed on one representation are transparently performed on all other representations as well. For example, sorting the input data sorts the output data analogously. The class `ARXLattice` offers several methods for exploring the solution space and for obtaining information about the properties of transformations

```
/* Load data from SQLite database*/
DataSource source = DataSource.createJDBCSource("jdbc:sqlite:test.db", "test");
source.addColumn("zipcode", DataType.STRING);
source.addColumn("age", DataType.INTEGER);
source.addColumn("diagnosis", DataType.STRING);
Data data = Data.create(source);

/* Create hierarchy with redaction*/
data.getDefinition().setAttributeType("zipcode", HierarchyBuilderRedactionBased.create(Order.RIGHT_TO_LEFT, Order.RIGHT_TO_LEFT, ' ', '*'));

/* Load hierarchies*/
data.getDefinition().setAttributeType("age", Hierarchy.create("age.csv", ';'));

/* Define sensitive attribute*/
data.getDefinition().setAttributeType("diagnosis", AttributeType.SENSITIVE_ATTRIBUTE);
```

Figure 7. Importing data and creating hierarchies with the API

(represented by the class `ARXNode`). The class `Inclusion` implements a dummy criterion that can be used to exclude tuples from the input dataset by defining a research subset.

In the remainder of this section, we will present an example for deriving an optimally anonymized transformation from a dataset loaded from an RDBMS. The process of importing data and of loading as well as creating generalization hierarchies is outlined in Figure 7. Firstly, a `DataSource` is created, which encapsulates all information required to access a database as well as the schematic properties of the data to be imported. The dataset is loaded by creating an instance of the class `Data`. Secondly, a generalization hierarchy for the attribute "zipcode" is created automatically by applying redaction. Finally, attribute types are specified. Quasi-identifiers are defined by associating a hierarchy, which is loaded from a CSV file for the attribute "age".

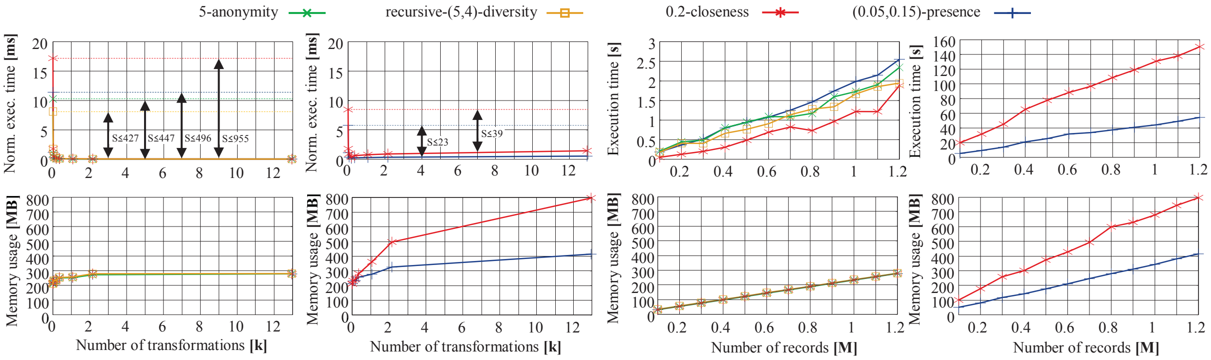
```
/* Configure the anonymization process*/
ARXConfiguration config = ARXConfiguration.create();
config.addCriterion(new KAnonymity(5));
config.addCriterion(new HierarchicalDistanceTCloseness("diagnosis", 0.6d, Hierarchy.create("diagnosis.csv", ';')));
config.setMaxOutliers(0d);
config.setMetric(Metric.createEntropyMetric());

/* Perform classification of the solution space*/
ARXAnonymizer anonymizer = new ARXAnonymizer();
ARXResult result = anonymizer.anonymize(data, config);

/* Write result of applying the optimal transformation*/
result.getOutput(result.getGlobalOptimum()).save("output.csv", ';');
```

Figure 8. Anonymizing and exporting data with the API

In Figure 8, the process of defining privacy requirements, configuring the transformation process, classifying the solution space, applying the globally-optimal transformation to the dataset and writing the result to a CSV file is sketched. The example configuration features 5-anonymity and 0.6-closeness for the sensitive attribute "diagnosis". The distance between distributions of the sensitive attribute is computed using a generalization hierarchy[28]. Tuple suppression is disabled and information loss is measured with non-uniform entropy[30].

Scalability of ARX

As already highlighted in the previous section, it is crucial that the process of classifying the search space is highly efficient. Most optimal anonymization algorithms are built on the assumption of monotonicity, meaning that generalizations of anonymous datasets are also anonymous and that specializations of non-anonymous datasets are also non-anonymous. This is, e.g., not true for ℓ-diversity and t-closeness combined with tuple suppression. ARX is the first tool to support such configurations, too, thus offering full support for tuple suppression. Our algorithm dynamically leverages any form of monotonicity. It can already prune parts of the search space if only some privacy criteria from a combination of several criteria are monotonic or if only the metric for assessing data utility is monotonic. In the context of utility metrics, monotonicity means that data quality decreases monotonically with generalization. This, again, is not always the case when tuple suppression is utilized.



(a) Scalability for monotonic configurations depending on the size of the search space

(b) Scalability for non-monotonic configurations depending on the size of the search space

(c) Scalability for monotonic configurations depending on the number of records

(d) Scalability for non-monotonic configurations depending on the number of records

Figure 9. Scalability of ARX for the *Integrated Health Interview Series* (IHIS) dataset

For our experiments we used a dataset from the *Integrated Health Interview Series* (IHIS)[34]. We have chosen this dataset, as it is publicly available and large, containing records about nearly 1.2 million individuals. From its attributes, we have chosen eight quasi-identifiers (QIs) and one sensitive attribute[18], resulting in a solution space consisting of 12,960 transformations. To assess scalability, we evaluated our tool with two different types of privacy problems: scenarios with 0% suppression in which criteria and metrics are monotonic (leading to best-case performance) and scenarios with 5% suppression in which neither criteria nor metrics are monotonic (resulting in worst-case performance). Additionally, we scaled the size of the dataset and the size of the solution space. The latter has been realized by incrementally increasing the number of QIs. In this case it has to be considered that adding a QI to a dataset multiplies the size of the solution space by the number of levels in the associated hierarchy and leads to the need to transform one additional column. To exclude this effect and derive comparable numbers for different scales, we report a *normalized execution time*, which is defined as the execution time divided by the number of transformations in the solution space and divided by the number of QIs.

The results are summarized in Figure 9. The normalized execution times for the monotonic case show that the optimizations implemented in ARX (including pruning based on monotonicity) lead to a speedup of up to a factor of 955 compared to the baseline execution time. The baseline is defined by the configuration with only one QI, as, in this case, almost no optimizations can be leveraged. In the non-monotonic scenario, the entire solution space must be searched. Still, ARX is able to leverage its optimized runtime environment to achieve speedups of up to a factor of 39 compared to its baseline performance. The factor between best-case execution times (~2.5s) and worst-case execution times (~150s) is roughly 60. The results of our experiments with increasing dataset size and a non-monotonic configuration show that our tool can evaluate 12,960 transformations in about 150s. This means that in an average of roughly 12ms, ARX is able *1)* to generalize and suppress values in a dataset consisting of 1.2 million records, with eight QIs each, *2)* to group the transformed records according to the QIs, *3)* to compute frequency distributions for the sensitive attribute values in each group, *4)* to check whether each frequency distribution fulfills t-closeness measured with the *Earth Mover's Distance* (EMD) based on a generalization hierarchy, and *5)* to compute the overall utility of the dataset in terms of non-uniform entropy. We note that our experiments were performed on cheap commodity hardware[18]. Performance increases further when no frequency distributions are required, e.g., for k-anonymity or δ-presence. In terms of memory requirements, our tool uses up to 300MB for monotonic and up to 800MB for non-monotonic configurations. We note that ARX implements a space-time trade-off that allows reducing its memory consumption, if required[17].

Discussion

Principal Results

We have presented a comprehensive open-source data anonymization framework. It is under active development, well-tested, and available for many platforms. We have given an overview of the core design of our system, the graphical user interface, and the public API for external software. Our tool implements a three-step data anonymization process, and it supports arbitrary combinations of privacy criteria and the use of tuple suppression for finding optimal transformations regarding t-closeness, ℓ-diversity or δ-presence.

ARX is still under development and constantly being updated with new features; we have just released version 2.2.0 which fully supports the anonymization workflow described in this paper. While this workflow and the implemented methods are motivated by usage scenarios of the biomedical domain, the tool can also handle other types of data. Through feedback from users and researchers we have learned that there is indeed a strong demand for data anonymization tools such as ARX. We constantly update our online documentation to provide answers to common questions and extend our tool to cover functionalities required by our users.

Tests and Experiences

We have successfully tested and evaluated the current version of ARX with multiple real-world benchmark datasets[18], including publicly available biomedical data, such as IHIS[34]. We plan to add a formal study with real-world use cases. From a scientific perspective, we have used our framework as a basis for several informatics research projects[17,18,35,36]. The hypothesis that our efforts have made data anonymization technologies available to a broader audience is supported by access statistics to our project website. In one year (2013/7-2014/7), our website had more than 2000 unique visitors, hundreds of whom have downloaded our tool. Visitors from India, the USA, Germany and Japan account for over 50% of all traffic.

Comparison to Prior Work

The UTD Anonymization Toolbox[21] supports three different privacy criteria (k-anonymity, ℓ-diversity and t-closeness) and uses a *SQLite* database backend. In our experiments, we encountered problems with larger datasets.

It further lacks a graphical interface and requires configuration to be performed via an XML file. It does not support combining tuple suppression with ℓ-diversity or t-closeness, which can lead to low data quality.

Table 1. Comparison to previous approaches

		UTD-AT	CAT	sdcMicro	μ-Argus	ARX
Developer Support	Open source	Yes	Yes	Yes	No	Yes
	Active	No	No	Yes	No	Yes
	Public API	No	No	Yes	No	Yes
	Extensibility	Low	Low	Low	No	High
	Cross-platform	Yes	Yes	Yes	No	Yes
	Prog. Language	Java	C++	R	C++	Java
Usability	GUI coverage	None	Full	Partial	Full	Full
	Hierarchy creation	No	No	Yes	No	Yes
	Visualization	No	Data, Risks	Data, Risks	Risks	Data, solution space
	Data sources	CSV	Proprietary	CSV, Various	CSV, Various	CSV, Excel, DBMS
	Hierarchy format	Proprietary	Proprietary	Proprietary	Proprietary	CSV
	Standalone	No	Yes	No	Yes	Yes
Anonymity Methods	Automatic solution	Yes	Yes	Partial	No	Yes
	Privacy criteria	k, ℓ, t	ℓ, t	k, ℓ	None	k, ℓ, t, δ
	Generalization	Yes	Yes	Yes	Yes	Yes
	Tuple suppression	Partial	No	Yes	Yes	Yes
	Risk assessment	No	Limited	Yes	Yes	Limited

The Cornell Anonymization Toolkit (CAT)[22] supports ℓ-diversity and t-closeness. It was developed for demonstration purposes and is no longer under active development. It lacks support for tuple suppression and requires input data to be in a tool-specific format. sdcMicro[23] is a package for the R statistics software and as such not meant to be a standalone application. It provides a graphical user interface, but only implements limited methods for automatically solving privacy problems or classifying the solution space. It supports k-anonymity and ℓ-diversity and is still being actively developed. μ-Argus[24] is a project which is no longer under active development. It is a closed-source Windows application that is not intended to act as a software library. It provides a broad spectrum of recoding techniques, including global recoding with local suppression as well as top and bottom coding and multiple methods for risk estimation. De-identification must be performed manually. SECRETA[19] is a tool that allows comparing different anonymization algorithms for relational and transactional data, but it is not available to the public. Many related tools include methods for evaluating re-identification risks, e.g., sdcMicro or μ-Argus. A simple but often used measure for the *prosecutor re-identification risk*[37] is the minimum, maximum and average size of equivalence classes, which is also available in our tool. If users need to use further risk assessment methods, data can be exported into other applications. The results of our comparison have been summarized in Table 1.

Future Work

In future work, we plan to enhance ARX with several additional features. We already implemented multiple risk estimators, e.g., the approach by Dankar et al.[38], but integrating the results into our tool will require further work. Additionally, we plan to combine our method with less restrictive coding models (e.g., local recoding) and to provide a set of typically needed hierarchies. We are also actively working on support for transactional attributes as well as methods for secure continuous data publishing. Currently, ARX focusses on methods for privacy-preserving release of microdata. In future work, we plan to integrate non-interactive variants of differential-privacy, which provide provable privacy guarantees that are independent of an attacker's background knowledge[12].

References

1. Wellcome Trust. Sharing research data to improve public health. 2013. Available from: http://wellcome.ac.uk/About-us/Policy/Spotlight-issues/Data-sharing/Public-health-and-epidemiology/WTDV030690.htm
2. OECD. Principles and guidelines for access to research data from public funding. 2006. Available from: www.oecd.org/sti/sci-tech/38500813.pdf.
3. Heeney C, Hawkins N, de Vries J, Boddington P, Kaye J. Assessing the privacy risks of data sharing in genomics. Public Health Genomics. 2011;14(1):17-25.
4. United States Congress. Health insurance portability and accountability act of 1996. Public Law. 1996:1-349.
5. Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data. Off J Eur Communities. 1995;38(L. 281).
6. U.S. Department of Health and Human Services. Office for Civil Rights. HIPAA Administrative Simplification Regulation, 45 CFR Parts 160, 162, and 164; 2013.

7. Benitez K, Malin B. Evaluating re-identification risks with respect to the HIPAA Privacy Rule. J Am Med Inform Assoc. 2010;17(2):169-77.
8. Malin B, Benitez K, Masys D. Never too old for anonymity: a statistical standard for demographic data sharing via the HIPAA Privacy Rule. J Am Med Inform Assoc. 2011;18(1):3-10.
9. Loukides G, Denny J, Malin B. The disclosure of diagnosis codes can breach research participants' privacy. J Am Med Inform Assoc. 2010;31:1-31.
10. Dwork C. Differential privacy. Proc Int Coll Automata, Languages and Programming. 2006;1-12.
11. Samarati P, Sweeney L. Protecting respondents identities in microdata release. IEEE Trans Knowl Data Eng. 2001;13(6):1010-1027.
12. Dankar F et al. Practicing differential privacy in health care: a review. Trans Data Priv. 2013;5:35-67.
13. Health System Use Technical Advisory Committee Data De-Identification Working Group. Best practice guidelines for managing the disclosure of de-identified health information. 2010. Available from: http://www.ehealthinformation.ca/documents/de-idguidelines.pdf.
14. El Emam K, Paton D, Dankar F, Koru G. De-identifying a public use microdata file from the Canadian national discharge abstract database. BMC Med Inform Decis Mak. 2011;11(1):53.
15. Benitez K, Loukides G, Malin B. Beyond safe harbor. Proc Int Conf Health Inform. 2010;163-172.
16. Ye H, Chen ES. Attribute utility motivated k-anonymization of datasets to support the heterogeneous needs of biomedical researchers. AMIA Annu Symp Proc. 2011;1573-82.
17. Kohlmayer F, Prasser F, Eckert C, Kemper A, Kuhn KA. Flash: efficient, stable and optimal k-anonymity. Proc Int Conf Privacy, Secur Risk Trust. 2012:708-717.
18. Prasser F, Kohlmayer F, Kuhn KA. A benchmark of globally-optimal anonymization methods for biomedical data. Proc Int Symp Computer-Based Medical Systems. 2014;66-71.
19. Poulis G, Aris GD, Grigorios L, Spiros S, Christos T. SECRETA: a system for evaluating and comparing relational and transaction anonymization algorithms. Proc Int Conf Ext Database Technology. 2014;620-623.
20. About PARAT De-Identification Software [cited 04 Aug 2014]. Privacy Analytics Inc. Available from: http://www.privacyanalytics.ca/software/parat/
21. UTD Anonymization Toolbox [cited 04 Aug 2014]. UT Dallas Data Security and Privacy Lab. Available from: http://cs.utdallas.edu/dspl/cgi-bin/toolbox/
22. Cornell Anonymization Toolkit [cited 04 Aug 2014]. Cornell Database Group. Available from: http://sourceforge.net/p/anony-toolkit/
23. sdcMicro [cited 04 Aug 2014]. Data-Analysis. Available from: http://cran.r-project.org/web/packages/sdcMicro/
24. μ-Argus manual. Available from: neon.vb.cbs.nl/casc/Software/MuManual4.2.pdf
25. Fung BCM, Wang K, Fu AWC, Yu PS. Introduction to privacy-preserving data publishing: concepts and techniques. 1st ed. Chapman and Hall/CRC; 2011:376.
26. Li T, Li N, Zhang J, Molloy I. Slicing: a new approach for privacy preserving data publishing. IEEE Trans Knowl Data Eng. 2012;24(3):561-574.
27. Machanavajjhala A, Kifer D, Gehrke J. l-Diversity: privacy beyond k-anonymity. Trans Knowl Discov from Data. 2007;1(1):3.
28. Li N, Li T, Venkatasubramanian S. t-Closeness: privacy beyond k-anonymity and l-diversity. Proc Int Conf Data Eng. 2007:106-115.
29. Nergiz ME, Atzori M, Clifton C. Hiding the presence of individuals from shared databases. Proc ACM SIGMOD Int Conf Manag data. 2007:665-676
30. Emam K El, Dankar F, Issa R, Jonker E, D. A globally optimal k-anonymity method for the de-identification of health data. J Am Med Inform Assoc. 2009;16(5):670-682.
31. Bayardo RJ, Agrawal R. Data privacy through optimal k-anonymization. Proc Int Conf Data Eng. 2005:217-228.
32. Ciglic M, Eder J, Koncilia C. k-anonymity of microdata with null values. Proc Int Conf Database and Expert Sys Appl. 2014.
33. ARX - Powerful Data Anonymization [cited 04 Aug 2014]. TUM. Available from: http://arx.deidentifier.org.
34. Integrated Health Interview Series [cited 04 Aug 2014]. NHIS. Available from: http://www.ihis.us.
35. Kohlmayer F, Prasser F, Eckert C, Kemper A, Kuhn KA. Highly efficient optimal k-anonymity for biomedical datasets. Proc Int Symp Computer-Based Medical Systems. 2012:1-6.
36. Kohlmayer F, Prasser F, Eckert C, Kuhn KA. A flexible approach to distributed data anonymization. J Biomed Inform. 2014;50:62-76.
37. El Emam K, Dankar FK. Protecting privacy using k-anonymity. J Am Med Inform Assoc. 2008;15(5):627-637.
38. Dankar FK, El Emam K, Neisa A, Roffey T. Estimating the re-identification risk of clinical data sets. BMC Med Inform Decis Mak. 2012;12(1):66.