Karlsruhe Reports in Informatics 2013,13
Edited by Karlsruhe Institute of Technology,
Faculty of Informatics
ISSN 2190-4782

FACTS: A Framework for Anonymity towards Comparability, Transparency, and Sharing

Extended Version

Clemens Heidinger, Klemens Böhm, Erik Buchmann, and Kai Richter

2013

KIT – University of the State of Baden-Wuerttemberg and National Research Center of the Helmholtz Association

Fakultät für **Informatik**

FACTS: A Framework for Anonymity towards Comparability, Transparency, and Sharing
Extended Version

Clemens Heidinger, Klemens Böhm, Erik Buchmann, and Kai Richter

Karlsruhe Institute of Technology (KIT), Germany

Abstract. In past years, many anonymization schemes, anonymity notions, and anonymity measures have been proposed. When designing information systems that feature anonymity, choosing a good approach is a very important design choice. While experiments comparing such approaches are enlightening, carrying out such experiments is a complex task and is labor-intensive. To address this issue, we propose the framework FACTS for the experimental evaluation of anonymization schemes. It lets researchers implement their approaches against interfaces and other standardizations that we have devised. Users can then define benchmark suites that refer to those implementations. FACTS gives way to comparability, and it includes many useful features, e.g., easy sharing and reproduction of experiments. We evaluate FACTS (a) by specifying and executing a comprehensive benchmark suite for data publishing and (b) by means of a user study. Core results are that FACTS is useful for a broad range of scenarios, that it allows to compare approaches with ease, and that it lets users share and reproduce experiments.

Keywords: Privacy, Anonymity, Evaluation, Benchmarking

1 Introduction

In the recent past, many anonymization approaches have been proposed, i.e., anonymity notions, anonymization schemes (subsequently referred to as 'schemes'), and measures for their evaluation. It is difficult to compare them and to decide when to use which scheme. It is hard to answer important questions, e.g.:

- Given a data set, a scheme, and an attack, how much information can the attack disclose?
- Given a data set and a set of queries, which scheme maximizes query accuracy while offering, say, Differential Privacy?

We see three dimensions that must be considered when comparing approaches, namely attacks, measures and benchmarks:

Attacks. Schemes have to make assumptions on the data set to be anonymized, and on the capabilities required to break anonymization. This allows to state if and to which degree the schemes give protection. Example 1 introduces our running example. In order to ease presentation, we use well-known approaches with well-researched vulnerabilities.

Example 1: Each row in Table 1a describes one individual. The data set contains attributes (quasi-identifiers) which might allow to identify a person (“Zip”, “Age” and “Sex”). It further has a sensitive attribute (“Disease”). Table 1b shows how a k-Anonymization scheme has transformed this data. This transformation cannot shield from attacks that disclose the sensitive attribute value for an individual. This is because, for all tuples with the same values of identifying attributes, the value of the sensitive attribute is the same. However, k-Anonymization implicitly assumes that there is no correlation between quasi-identifying and sensitive attributes. The so-called homogeneity attack can exploit this to break the anonymization of Table 1b. Another scheme is required; see for instance Table 1c for an anonymization outcome with l-Diversity. ●

Table 1: Examples for Anonymization.

(a) Original data set.

Zip	**Age**	**Sex**	**Disease**
13053	28	F	Lupus
13053	29	F	Lupus
13068	21	M	AIDS
13053	23	F	AIDS

(b) Outcome with k-Anonymity ($k = 2$).

Zip	**Age**	**Sex**	**Disease**
130*	$[28, 29]$	F	Lupus
130*	$[28, 29]$	F	Lupus
130*	$[21, 23]$	*	AIDS
130*	$[21, 23]$	*	AIDS

(c) Outcome with l-Diversity ($k = l = 2$).

Zip	**Age**	**Sex**	**Disease**
130*	$[21, 28]$	*	Lupus
130*	$[23, 29]$	F	Lupus
130*	$[21, 28]$	*	AIDS
130*	$[23, 29]$	F	AIDS

[23] has shown that any scheme that preserves some utility has to rely on assumptions. Attacks in turn exploit such assumptions. This may result in new schemes to shield against them, i.e., we observe a stream of new attacks and countermeasures, for many scenarios. We say that *approaches belong to the same scenario* if they share certain basic requirements. For example, in scenario data publishing of microdata ($\mathbf{S}_{\text{PUB}}$) one releases modified data sets without any means to undo these modifications. With scenario database-as-a-service ($\mathbf{S}_{\text{DAAS}}$) in turn, a requirement is to have de-anonymization mechanisms for authorized individuals. Besides $\mathbf{S}_{\text{PUB}}$ and $\mathbf{S}_{\text{DAAS}}$ there are many more scenarios, e.g., statistical databases ($\mathbf{S}_{\text{STATS}}$) and data mining ($\mathbf{S}_{\text{MINING}}$). [13] alone describes 28 schemes of $\mathbf{S}_{\text{PUB}}$ and $\mathbf{S}_{\text{STATS}}$ proposed until the year 2009. Since then, many more schemes have been proposed, e.g., [7], [24], [26], and [37]. Differential Privacy [11] for example assumes independent database records – [24] then describes an attack exploiting dependencies between records, together with a respective new scheme. To find out if a scheme can be used in a certain real-world context, it is important to test the anonymized data against such attacks.

Measures. Besides formal proofs of anonymity and complexity analyses, quantitative measures are needed to assess the applicability of a scheme for real-world applications. An example is the probability that the anonymity of a data set can be broken if it has been anonymized with a certain scheme. Regarding performance, it is interesting to know if there is an optimal scheme that can anonymize a certain data set in reasonable time, or if a heuristic scheme is needed. Further measures consider data quality and query accuracy – we address them later in this article. However, the sheer number of schemes, attacks, and application requirements makes it hard to identify the best scheme for a given setting. Making the right choice is important to account for high-level privacy requirements, cf. [4].

Benchmarks. Schemes may be related in that they aim at the same kind of protection, e.g., against linking values of sensitive attributes to individuals. However, related schemes typically have been evaluated with different experiments. For example, [29] uses the UCI Adult data set, while the related scheme [37] uses an IPUMS census data set: One cannot compare their measurements of data quality or of query accuracy.

Example 2: Queries on non-anonymized data sets may need to be modified to be executed on the anonymization output. Query-processing techniques then must be tailored to schemes. With our running example, the query `SELECT * FROM Table 1a WHERE Age BETWEEN 22 AND 28` needs to be modified so that values of “Age” map to the generalized intervals in Table 1b. Different measures for the loss of accuracy exist. To have experiments that are comparable, not only the data must be identical, but also those modifications of the queries and the accuracy measures. ●

The three dimensions of evaluation problems described above call for a framework that supports a detailed comparison of schemes based on the requirements of real-world applications. Designing such a framework is challenging, given the wide variety of possible attacks, measures and benchmarks. Although in this paper we limit examples and discussions to $\mathbf{S}_{\text{PUB}}$ and $\mathbf{S}_{\text{DAAS}}$, we strive for a framework that also works for other scenarios, e.g., $\mathbf{S}_{\text{STATS}}$ and $\mathbf{S}_{\text{MINING}}$. In this context, the heterogeneity of scenarios is challenging. For instance, data quality is not important for $\mathbf{S}_{\text{DAAS}}$, but for $\mathbf{S}_{\text{PUB}}$, it is.

In this paper, we propose FACTS, a **F**ramework for **A**nonymity towards **C**omparability, **T**ransparency, and **S**haring. It allows to compile benchmarks together with the implementations of schemes and of attacks, data sets, query-processing techniques etc. When designing FACTS, we have devised standards for the anonymization application, namely for interfaces that researchers proposing new approaches must implement and for data they must provide. Users can then define, share, update, and execute benchmarks for anonymization that refer to the standards. FACTS addresses comparability, as Example 3 illustrates.

Example 3: An author of a query must implement two methods whose interfaces are given by FACTS: one for the query on anonymized data sets, another one for the query on the original data. In $\mathbf{S}_{\text{PUB}}$, this allows to measure the loss of query accuracy (cf. Example 2). In $\mathbf{S}_{\text{DAAS}}$, it allows to quantify the performance costs of decrypting query results and to verify that results are the same as without encryption. ●

In order to ease comparing approaches, FACT supports technical features such as the reproducibility of experiments, collaboration amongst researchers, the understandability of experimental results, and workability, i.e., the exploration of the effect of evaluation parameters. For example, FACTS supports understandability by generating and storing charts depicting experiment results together with the respective benchmark suites. This allows to link a chart to, say, the exact input data sets, parameters, and implementations, to make its computation transparent. Because new attacks keep appearing, FACTS has a central repository of benchmarks, and users can specify new benchmarks to include new attacks. The modular design of FACTS aims at composing and modifying benchmarks with ease and at reusing previous work as much as possible.

Our evaluation is twofold. On the one hand, we have developed various benchmarks, including one for $\mathbf{S}_{\text{PUB}}$ and one for $\mathbf{S}_{\text{DAAS}}$. On the other hand, we report on a user study with 19 participants that has continued for three months. The evaluation shows that FACTS addresses its objectives well, e.g., FACTS standards allow to compare approaches fairly by enforcing compliance with benchmarks. We have implemented the framework and the benchmarks in full and make everything available under a free license on our website [1]. The vision is that over time it will become common among anonymization researchers to refer to suitable benchmarks.

Paper Outline Section 2 describes our terminology, our running example in more detail, and the evaluation of approaches of the anonymization research area. Section 3 describes our framework. Section 4 describes the features and use cases. Section 5 evaluates. Section 6 concludes.

2 Background

2.1 Terminology and Examples

Anonymization. Our understanding of the term anonymity is broad and includes approaches such as encryption, see below. Three important anonymization scenarios are:

Data Publishing of Microdata ($\mathbf{S_{pub}}$) A data publisher makes anonymized data available to the public. Anonymization aims at protecting the privacy of individuals while transforming data as little as possible.

Statistical Databases ($\mathbf{S_{stats}}$) A data publisher makes anonymized results of statistical queries on data on individuals available. Anonymization aims at protecting the privacy of individuals while query results are highly accurate.

Database-as-a-Service ($\mathbf{S_{DaaS}}$) Users create an anonymized/encrypted database and let an honest-but-curious provider store it and execute queries. Users can revert anonymization and access the original data. Anonymization lets only authorized users access the original data. [31] gives an overview of privacy and security implications.

Any *scheme* takes an *original data set* as input, with *original values* in its cells. With $\mathbf{S}_{\text{STATS}}$, a set of functions that operate on the data is input as well. Schemes generate an *anonymization output*. With $\mathbf{S}_{\text{PUB}}$ and $\mathbf{S}_{\text{DAAS}}$, this output is the entire anonymized data set, with $\mathbf{S}_{\text{STATS}}$ it is anonymized query results. Any scheme seeks to protect against a certain kind of disclosure of sensitive information. The *protection model* states which information to protect. *Anonymity notions* state characteristics the output of or the information processed by schemes must have. An anonymity notion may refer to a certain protection model, i.e., any scheme compliant with the notion protects the information specified by the protection model. Adversaries execute *attacks* that try to break the protection. *Adversary models* describe the adversary, i.e., her capabilities and her background knowledge. An anonymity notion may include a reference to an adversary model: A scheme complies with the anonymity notion iff the adversary of the referenced adversary model cannot get to the information it aims to protect. Finally, *anonymity* is given if a scheme protects the information specified by the protection model against adversaries as defined by the adversary model. Thus, schemes such as pseudonymization or partitioning [2] can offer "anonymity" according to this definition.

Example 4: With $\mathbf{S}_{\text{PUB}}$, the original data sets contain quasi-identifying and sensitive attributes. An assumption is that each tuple belongs to one individual. A protection model is that any sensitive attribute value must not be linked to the respective individual. The anonymity notion k-Anonymity [35] specifies the following rule for anonymization output: For any tuple, there are at least $k-1$ other tuples with the same values for the quasi-identifying attributes. $\mathcal{S}_{\text{k-Anonymity}}$, a scheme for k-Anonymity, with k set to 2, computes the output in Table 1b. It generalizes original values, to build so-called *QI blocks*. This anonymization however cannot protect from adversary Alice who wants to disclose the disease Bob has. The adversary model is that Alice has knowledge about individuals, as follows. Alice knows that Bob is in the database, lives in a zip-code area beginning with 130, and is 21 years old. She concludes that Bob has AIDS. That is, she executes the so-called homogeneity attack [30]. We refer to it as $\mathcal{A}_{\text{HG}}$. l-Diversity [30] protects against $\mathcal{A}_{\text{HG}}$, see Table 1c. ●

Experiments. 'Approach' is our generic term for any new concept an anonymization researcher might propose. Approaches include schemes, attacks, query processing, and measures. Next to anonymity, schemes may have further goals: With $\mathbf{S}_{\text{PUB}}$, a goal is to maximize data quality for subsequent analyses. For $\mathbf{S}_{\text{STATS}}$, the data set is hidden, and the user can enter a given set of statistical queries – a goal is to maximize the accuracy of their results. For $\mathbf{S}_{\text{DAAS}}$, a goal is to maximize performance of query processing. In general, researchers strive to find schemes that are good regarding a combination of goals, as quantified by *measures*. Measures used in the literature are *anonymity*, *data quality*, *query accuracy*, and *performance*. For instance, anonymity measures quantify to which degree an adversary can break the anonymization, i.e., disclose the information specified in the protection model, and data-quality measures quantify how much the anonymized data set differs from the original one. An *experiment* to evaluate an approach has *experiment parameters*, at least an original data set and a measure.

With our terminology, a *benchmark* is a set of experiments. A *benchmark suite* bundles benchmarks with schemes that use them, and contains their parameters and runs such bundles. It yields *measure values*, i.e., values from the respective experiments as output. One benchmark may be used in several suites. We differentiate between benchmark specification and benchmark execution with suites. This is because one might have an interesting benchmark, e.g., containing a new data set, but might not have a scheme using it. In general, we see two user roles: *researchers* and *users*. Researchers are inventors/implementors of an approach. Users deploy approaches. They do not necessarily know the inner structure of the approach they use. A researcher can also be a user.

Example 5: Continuing Example 4, we illustrate how the measure of [29] (we refer to it as $\mathcal{M}_{\text{Anon-Dist}}$) quantifies the threat posed by $\mathcal{A}_{\text{HG}}$. $\mathcal{M}_{\text{Anon-Dist}}$ is the maximum distance between (a) any distribution of values of the sensitive attribute of a QI block in the anonymized data set and (b) the distribution of values of the sensitive attribute of the original data set. $\mathcal{A}_{\text{HG}}$ can conclude that an individual has a sensitive attribute value if the distance is large, as we now explain. Experiment e quantifies anonymity with $\mathcal{M}_{\text{Anon-Dist}}$:

```
e = { original data set: Table 1a, anonymity measure: $\mathcal{M}_{\text{Anon-Dist}}$ }
```

Benchmark B contains e as the only experiment, i.e., $B = \{e\}$. Benchmark suite $\mathcal{B}$ runs B for two schemes.

```
$\mathcal{B}$ = { ( experiment: $e \in B$, scheme: $\mathcal{S}_{\text{k-Anonymity}}$, parameters: {$k = 2$} ),
      ( experiment: $e \in B$, scheme: $\mathcal{S}_{\text{l-Diversity}}$, parameters: {$k = 2$, $l = 2$} ) }
```

$\mathcal{B}$ executes and generates Tables 1b and Table 1c as the anonymization output. For Table 1b, the distributions (a) are {Lupus, Lupus} and {AIDS, AIDS}. For Table 1c, the distributions (a) are {Lupus, AIDS}, both times. Distribution (b) is {Lupus, Lupus, AIDS, AIDS}. The distributions (a) for Table 1b have a greater distance to distribution (b) than the distributions (a) for Table 1c. $\mathcal{M}_{\text{Anon-Dist}}$ thus calculates a higher degree of disclosure for Table 1b. ●

2.2 Schemes and their Evaluation

In this subsection we classify work on anonymization into work on protection models, on anonymity notions, on schemes, and on measures. We focus on work that we explicitly refer to later in the paper.

Protection Models

There are several protection models, e.g., *record linkage*, *attribute linkage*, and *table linkage*: The adversary must not identify the record of an individual, associate an attribute value with an individual, or know if an individual is in the anonymization output. Another protection model is the *probabilistic* one: The anonymization output should provide the adversary with little information beyond her background knowledge. [13] systematically describes these four protection models for $\mathbf{S}_{\text{PUB}}$ and $\mathbf{S}_{\text{STATS}}$.

Anonymity Notions

Anonymity Notions for $\mathbf{S_{pub}}$ and $\mathbf{S_{stats}}$. There are many anonymity notions for $\mathbf{S}_{\text{PUB}}$ and $\mathbf{S}_{\text{STATS}}$. [13] explicitly describes 15 such notions proposed until the year 2009. Since then, many more notions have been proposed, e.g., [7] and [24].
Anonymity Notions for $\mathbf{S_{DaaS}}$. We classify the related work of $\mathbf{S}_{\text{DAAS}}$ into the four protection models of [13]. The anonymity notion introduced in [3] is privacy constraints stating which pairs of attribute values that occur together in a tuple an adversary is forbidden to learn. This belongs to protection model *attribute linkage*. The anonymity notion proposed in [20] is an indistinguishability concept for databases, Ind-ICP. An adversary must not be able to tell an anonymization of a database d apart from one of a variation of d where values of each column have been permuted. The anonymization output must satisfy this indistinguishability requirement at all times for any input database. This notion is an instance of the *probabilistic* one.

Schemes

Schemes for $\mathbf{S_{pub}}$ and $\mathbf{S_{stats}}$. There are numerous schemes, e.g., Mondrian and Datafly for $\mathbf{S}_{\text{PUB}}$ or ϵ-Differential Privacy by adding noise for $\mathbf{S}_{\text{STATS}}$. [13] describes these and other schemes proposed until the year 2009, 28 in total. Since then, many more schemes have been proposed, e.g., [7], [26], and [37].
Schemes for $\mathbf{S_{DaaS}}$. [17] proposes "Databases as a Service" (DaaS) where an external service stores databases and executes queries. [17] and [33] propose to use encryption to protect from adversaries. [16] proposes to execute SQL over encrypted data and to speed up query execution with index attributes. [21] and [22] propose to generate an index on encrypted databases for equality and range queries. [10] proposes to outsource a B+ tree as an index structure. Three schemes [3,9,18] fragment data – this qualifies as anonymization by our broad definition. [3] stores data fragments in two databases hosted by two providers that do not cooperate. [9] extends this to an arbitrary number of fragments. In [18] we describe a scheme tailored to a specific use case which fragments a database into four parts. In [19], a base scheme generalizes the scheme of [18] to work with any database and we further propose three enhanced schemes for better performance here.

Anonymity Measures

Anonymity Measures for $\mathbf{S_{pub}}$. Most proposals identify a new attack and then have custom measures tailored to the attack. [30] for example counts the number of tuples with a homogeneous QI block with $\mathcal{A}_{\text{HG}}$.
Anonymity Measures for $\mathbf{S_{DaaS}}$. Publications here use their own or no anonymity measure. For example, the measure of [8] raises alarm if the adversary can disclose one or more values in one row of the anonymization output. The measure quantifies the probability of the adversary guessing correctly if attribute values occur in the same row. [19] uses a variation of the measure of [8] for the threat that comes with one of its four schemes. [22] and [21] study an adversary

who knows which attribute values are mapped to which index values. With poor anonymization, and one index value mapped to exactly one attribute value, the adversary would know which attribute value the index value is for. Their anonymity measures thus are the variance and the entropy of possible original values of index values.

Data-Quality Measures

There are a few data-quality measures for $\mathbf{S}_{\text{PUB}}$ and $\mathbf{S}_{\text{STATS}}$ such as the share of values generalized. [13] describes the measures systematically. For $\mathbf{S}_{\text{DAAS}}$ there are no data-quality measures. Namely, quality is always 100 percent for authorized users and 0 percent for non-authorized ones.

Query-Accuracy Measures

One query-accuracy measure frequently used with $\mathbf{S}_{\text{PUB}}$ and $\mathbf{S}_{\text{STATS}}$ is as follows:

Definition 1 (Measure $\mathcal{M}_{\textbf{Query-Acc-Avg}}$). *Let Q be a set of queries, and let δ_{rel} be the relative error of executing a query $q \in Q$ on the original data set, compared to the anonymization output. $\mathcal{M}_{\textit{Query-Acc-Avg}}$ now is $\delta_{avg} = \frac{1}{Q} \cdot \sum_{q \in Q} \delta_{rel}$*

[13] describes this and other measures for $\mathbf{S}_{\text{PUB}}$ and $\mathbf{S}_{\text{STATS}}$.

Performance Measures

Performance measures are the canonical ones, e.g., execution times of queries, schemes, or attacks.

2.3 Comparability Problem

In this subsection we describe the status quo of comparison efforts with anonymization for our scenarios.

Regarding $\mathbf{S}_{\text{PUB}}$, the survey [13] alone lists 15 anonymity notions and 28 schemes. With $\mathbf{S}_{\text{STATS}}$, there also is much activity since *Differential Privacy* [11] has been proposed. Several schemes comply with it, see [12]. Even though Differential Privacy is a probabilistic scheme, attacks are possible, e.g., [24]. Thus, one might want to measure the success of attacks with experiments here as well. The high number of publications makes it hard to decide for a specific task which scheme is compatible with its adversary model and its protection model and is best regarding anonymity, data quality, query accuracy, or performance.

With both $\mathbf{S}_{\text{PUB}}$ and $\mathbf{S}_{\text{STATS}}$, [24] observes that many articles only give incomplete adversary models. [24] further notices that authors often do not state a protection model that they aim to address. This makes comparability hard. With $\mathbf{S}_{\text{DAAS}}$, only few schemes comply with anonymity notions that specify protection models, see Tables 2a and 2b. For the publications that do specify anonymity notions, only two notions are reused. Publications create new experiments that do not compare with related work. For example, [21] conducts extensive and insightful experiments but only compares to a baseline that retrieves the entire database with each query. We conclude that there is little comparison effort in the area.

Table 2: Comparability with $\mathbf{S}_{\mathrm{DAAS}}$.

(a) Anonymity notions.

	[3]	[20]
Record Linkage		
Attribute Linkage	✓	✓
Table Linkage		
Probabilistic Attack		✓

(b) Schemes.

	Anonymity Notion	**Anonymity Measure**
[17]	●	●
[33]	●	●
[16]	●	●
[22]	●	✓
[21]	●	✓
[10]	●	●
[3]	[3]	●
[9]	[3]	●
[18]	Ind-ICP	●
[19] (base scheme and two of three enhanced schemes)	Ind-ICP	●
[19] (one of the enhanced schemes)	●	✓

Problem 1. The research area of anonymity requires support for the evaluation of approaches in different contexts regarding different criteria and for comparing approaches to each other with ease.

3 FACTS

We now present FACTS, our framework for easy comparability for anonymization research. In this section we give an overview, describe the key concepts, and say how to implement benchmarks.

3.1 Overview

FACTS is a framework for easy comparison of anonymization approaches. A core issue when designing FACTS has been to come up with class models of approaches. Class models are our standardizations of behavior and of processes in the context of anonymization. In FACTS, researchers provide implementations of class models, by implementing them against interfaces we, the designers of FACTS, have specified. Researchers further have to comply with the standards class models specify for data generation. Users configure benchmarks and benchmark suites within FACTS that refer to these implementations. Benchmark suites bundle all data, i.e., data sets, the implementations of class models, and experiment results. FACTS stores everything in a central repository. Users can execute benchmark suites to compare the state of the art with ease. The idea is that users who are experts of an anonymization sub-domain create benchmark suites for approaches where a comparison is interesting.

3.2 Aspects

FACTS covers four aspects. First, users define benchmark suites, i.e., the specification which approaches to compare based on which data, parameter settings etc. Benchmark suites refer to implementations of class models. We have defined class models as the second aspect, and there are class models of schemes, attacks, and queries. The third aspect is that FACTS executes these class models and performs the benchmarking. FACTS stores all results and protocols of executing benchmark suites in a repository, this is the fourth aspect. See Figure 1.

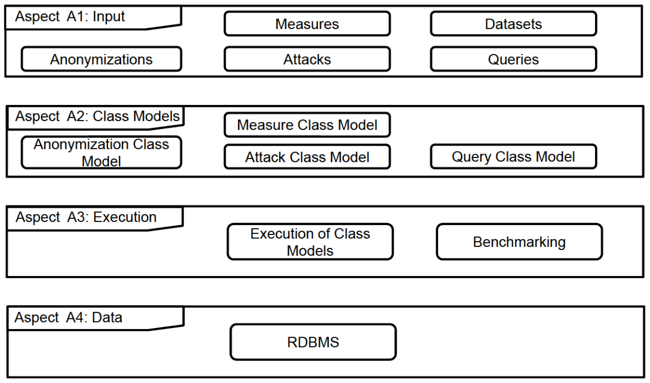

Fig. 1: FACTS – Overview.

3.3 Benchmarks

We now describe how to realize benchmarks by means of the four aspects.

Aspect A1: Input In this aspect, a user configures benchmark suites. Standardized interfaces and data representations ensure that all input plays well together, e.g., the scheme knows how to access the input data set. Benchmark suites refer to one or more experiments. An experiment has the following parameters:

1. An original data set D.
2. A scheme $anon$, possibly with parameters, referred to as $params(anon)$.
3. An attack $attack$. It may have parameters, referred to as $params(attack)$.
4. A set of queries Q where each query $q \in Q$ may have parameters $params(q)$.
5. A measure $\mathcal{M}$.

Users may omit (3) or (4) if the experiment does not make use of attacks or queries, e.g., experiments on the performance of schemes.

Aspect A2: Class Models Class models let researchers model approaches with a set of interfaces they need to implement and standardized formats of the data they need to generate. For example, there is an interface for attacks that lets researchers make background knowledge explicit, and methods accessing such background knowledge return it in a format standardized within FACTS. This for example allows authors of anonymity measures to use the knowledge. Our evaluation will show that the FACTS interfaces are on the one hand sufficiently generic and, on the other hand, specific enough to make comparisons indeed easier. Further, FACTS allows to compose complex schemes, attacks, and queries from so-called operations. Operations can be used individually, or they can be combined by means of so-called macros. Operations and macros allow to encapsulate and combine logical operations such as encryption or randomization, to reduce the necessary implementation work.

Aspect A3: Execution This aspect performs the benchmarking, with measurements. FACTS instantiates the implementations of class models of Aspect **A2** with the data of Aspect **A1**. That is, FACTS runs the schemes, attacks, and queries. Experiments are logged, including time, date of execution, and the input data set.

Aspect A4: Data This aspect stores all data, i.e., benchmark suites (**A1**), the class models and their implementations (**A2**), and the measurement results and execution logs (**A3**). FACTS stores all data sets and implementations of approaches for later runs of the same suite. This is transparent to the users; FACTS takes care of the data storage. For instance, users and researchers do not need to know the schema of the database or other internals of the framework. They do not need to concern themselves with logging or with the storage of implementations. They only have to comply with a few standardizations for data generation. One example of such a standardization is that a user has to provide a name for a benchmark suite.

Documentation. For specifics of the implementation, e.g., interfaces, we refer to the documentation on the FACTS website [1].

Implementation and User Involvement. We have implemented the framework in Java. We store the repository in an Oracle database and use Hibernate for the object-relational mappings. Users and researchers can download a runtime package (RTP) and configure it against a repository. For Aspect **A1**, users configure new benchmark suites by invoking RTP methods. For **A2**, researchers implement new approaches by extending respective Java classes. The reason why we use Java is that it is turing-complete and researchers can implement complex approaches with it: For example, with $\mathbf{S}_{\text{DAAS}}$, queries need to decrypt query results, and plain SQL instead of Java would not work. Regarding **A3**, users start the execution of benchmark suites by invoking respective methods of the RTP. Users can access benchmark results in logs with many convenience methods. Two examples are method `getOriginalDatasets` to retrieve the original data set of a benchmark, and method `getAnonymizationRuntime` to retrieve the execution time of its scheme. Another method of the RTP allows to store any information related to the first three aspects in the repository, i.e., new benchmark suites, new implementations of class models, and benchmark results. Users can search the repository with various methods, e.g., search for a benchmark suite by name.

Anonymity Benchmarks Anonymity measures quantify how well an adversary can guess the original value represented by an anonymized one. We now describe how the four aspects realize anonymity benchmarks. Benchmarks for data quality, query accuracy, or performance are similar.

Aspect A1: Input Users configure the benchmark suite with this aspect. Listing 1 has the respective method calls for FACTS initialization. FACTS provides implementations of the methods, and users just invoke them. First, users specify the input data set D, e.g., import D from a file (Lines 1-3). Next, the user sets up a new experiment with D (Line 4-5). The user must further specify the scheme *anon* (Line 6) and the attack *attack* (Line 8) and link them to the experiment and the benchmark suite (Lines 7 and 9). $\mathcal{M}$ finally is an anonymity measure (Line 10). If $\mathcal{M}$ just analyzes the anonymization output, *attack* may be a so-called null attack from our repository which does not do anything. If the threat stems from analyzing query strings or query results, the user must specify a set of queries Q in addition.

```
 1  CSV csv = new CSV(new File(inputData));
 2  Dataset input = csv.importDataset();
 3  Datasets original = new Datasets().add(input);
 4  m = benchmarkSuite.createMeasurement();
 5  m.setInputDatasets(original);
 6  AnonymityClassModel anon = new anon();
 7  m.setAnonymityClassModelImplementation(anon);
 8  AttackClassModel attack = new attack();
 9  m.setAttackClassModelImplementation(attack);
10  m.setMeasure(𝓜);
```

Listing 1: Configuration of anonymity benchmarks.

Aspect A2: Class Models A researcher implements *anon* by extending a specific class. He must specify (i) the name of the scheme, (ii) the name and type of its parameters, (iii) an informative text in natural language, and (iv) an implementation of *anon*. If researchers do not specify either of (i)-(iv), FACTS reports an error. The anonymity class model requires the result of the anonymization to be `AnonymizedDataset` objects. To fill them with content, FACTS provides a generic input method. It lets the researcher specify the original value for a certain anonymized value.

Researchers implement attack *attack* by extending another specific class. They must implement the following interfaces for preconditions of the attack, background knowledge, and for attack results.

preconditions Preconditions are rules that have to be fulfilled in order for *attack* to work. Examples are that at least 20 GB of RAM is available, or that the original data set has at least 1000 tuples.

backgroundKnowledge Background knowledge are data sets used during *attack*. Currently, some papers only describe it roughly, but no concrete data sets are given. This interface lets the researcher explicitly declare background knowledge so that it can be retrieved and used in comparisons at any later point in time. The data sets it generates can be some existing data set, or it can compute the background knowledge from the original data set. Example 6 illustrates the latter.

Example 6:

```
1  public void backgroundKnowledge(Dataset original, Dataset anonymized,
        Dataset preconditions, Parameters p, OperationAssembler
       knowledgeLogic)
2  {
3   Attribute a = original.getAttribute("tag");
4   a.setAggregate(AggregateType.COUNT);
5   a.setGroupByAttribute(true);
6   Dataset knowledge = original.aggregateSelect(a);
7   knowledgeLogic.addOperation(new NullOperation(), knowledge);
8  }
```

The definition of the interface `backgroundKnowledge` (Line 1) has the original data set as one parameter. As the implementation of the interface, the researchers use FACTS methods implemented by us to compute a count of each value of the attribute named "tag" of the original data set (Lines 3-6). Another FACTS method returns the result to the framework (Line 7). ●

attack Attack results are the guesses of the adversary which original values lie behind the cells of the anonymization output. The interface lets researchers access the background knowledge. The result of this method must consist of the same number of `Dataset` objects as there are `AnonymizedDataset` objects. In case an attribute is not important for *attack*, the implementation inserts an empty value which $\mathcal{M}$ then ignores.

With its interfaces, FACTS makes preconditions, background knowledge, and attack results available at a well-known location and available for later reference. The rationale is to give way to interesting comparability features.

Example 7: It is interesting to get the background knowledge of all attacks in a benchmark suite and compare it. Listing 2 shows the method calls necessary to get to this knowledge. FACTS provides these methods. First, a user retrieves all measurements of a benchmark suite (Line 1) and iterates over them (Line 2). Each measurement has a reference to the implementation of an attack class model. This implementation is now linked with the measurement, and one can retrieve it with method `getAttackImplementation()`. When implementing the class model, the researchers must also implement an interface that explicitly declares background knowledge. Method `getKnowledgeDatasets()` retrieves this knowledge. ●

```
1  Collection measurements = benchmarkSuite.getMeasurementsAsCollection();
2  for (Measurement m: measurements) {
3   Datasets knowledge = m.getAttackImplementation().getKnowledgeDatasets()
        ;
}
```

Listing 2: Getting all background knowledge of a benchmark suite.

Our approach that an attack outcome is one guess for each cell of the original data set is sufficiently general. With record-linkage attacks for example, the attack can store which individual described by the original data set it corresponds to, in each cell of the attack-result data set. The implementation of a measure needs to know how to interpret the values in attack-result data sets, i.e., measures and attacks have to know the specifics of each another. In other words, anonymity measures only make sense with certain attacks, but typically not all of them. By having as many cells for the guesses of the adversary as there are original values, FACTS lets the authors of measures and of attacks implement a broad range of approaches.

Aspect A3: Execution Listing 3 features the execution of a benchmark; it is a continuation of Listing 1. We, the FACTS implementors, have implemented the methods for benchmark execution, and a user just calls them. While executing the methods, FACTS refers to *anon*, *attack*, and $\mathcal{M}$. Line 1 runs the scheme *anon*, which produces the anonymized data set $anon(D)$. Next, Line 2 executes *attack*. Finally, Line 3 executes $\mathcal{M}$, which accesses the values guessed by *attack* and compares them to the original values specified during anonymization *anon*.

```
1  m.runAnonymization();
2  m.runAttack();
3  m.runMeasure();
```

Listing 3: Execution of anonymity benchmarks.

FACTS uses the design pattern "Inversion of Control" (IoC) to execute class models. This pattern decouples the execution of a task from its implementation: With FACTS and anonymity benchmarks, IoC ensures that class models do not contain any code for benchmarking or other run-time tasks. For example, researchers implementing *attack* do not need to ensure the execution of *anon*: It is the responsibility of FACTS to inject the proper data sets at run-time at the respective interfaces. This requires a systematic identification and implementation of the execution processes in our context. One example of an execution process is that, with attacks, FACTS first evaluates preconditions, it then computes background knowledge, before it starts the actual attack.

Aspect A4: Data This aspect stores all background knowledge and attack results, all anonymization and attack class models and their implementations, and all benchmark results. FACTS produces *anonymity logs* and *attack logs*, as follows. Logs are stored in a relational database. We have devised and implemented various convenience methods to access logs. Anonymity logs protocol the selected implementation of the anonymization class model, its parameters, and references to the input data set and to the anonymization output data set. Attack logs protocol attack class models with their parameters, anonymity measures and their values, and references to all data sets (input data set, anonymized data set, background knowledge, and attack results).

4 Features and Use Cases

In this section, we describe important features of our framework, namely comparability, reproducibility, workability, collaboration, and understandability, together with respective use cases. These use cases will form the basis of our evaluation in the next section.

Feature 1 (**Comparability**). Comparability means quantifying anonymity, data quality, query accuracy, and performance of approaches. This is to decide which approach is best for a given real-world problem.

FACTS gives way to comparability by means of benchmarks.

Use Case ($\mathbf{U}_{\text{BENCHMARK}}$). FACTS lets users define, update, and access benchmarks for anonymization. For anonymization approaches that are related, e.g., approaches that aim for the same protection model, a user creates a benchmark suite that compares them, together with attacks and queries, under a set of measures. When a researcher proposes a new attack, users can update benchmark suites to include it, or create new ones.

Feature 2 (**Reproducibility**). Reproducibility lets unbiased third parties repeat and verify experiments.

Experts in their respective scientific fields have stressed the importance of reproducibility. For example, [5] states that more research is necessary to get to good experiment tools. FACTS supports reproducibility use cases such as the following one:

Use Case ($\mathbf{U}_{\text{COMMITTEE}}$). Authors of a new scheme, attack, or query-processing technique add an implementation of their approach to the FACTS repository and to a benchmark suite. They use this benchmark suite to evaluate their technique. A respective conference committee can later retrieve the benchmark suite. The committee can rerun measurements without difficulty and award a reproducibility label.

Feature 3 (**Workability**). Workability lets one explore effects of modifications of evaluation parameters.

Workability allows to evaluate if an approach achieves good results solely because experiment parameters were chosen to its advantage. Parameter values however may be hidden in an implementation. It can be hard to identify and to vary them subsequently. FACTS addresses this:

Use Case ($\mathbf{U}_{\text{WORKABILITY}}$). Alice is developing a new approach. FACTS requires Alice to specify the parameters with interfaces she has to implement, be it for anonymization, attacks, or queries. Bob now wants to evaluate this new approach. He retrieves and changes parameters of any benchmark with the approach. To this end, he can use FACTS methods that we have already implemented. He does not need to search for parameters in the code. This lets Bob observe how parameters affect benchmark results with ease.

Feature 4 (**Collaboration**). Collaboration within the community allows for faster development of new approaches.

In publications, details such as the concrete data set, initialization or termination procedures and the values of parameters are not always given [36]. This makes it hard for researchers to build upon existing work, i.e., when implementing a new approach by reusing some of the implementation of an existing one.

Use Case ($\mathbf{U}_{\text{SHARING}}$). FACTS gives way to sharing of operations. Suppose that researchers have developed a new scheme for $\mathbf{S}_{\text{DAAS}}$ that protects against adversaries trying to find out the order of tuples. The authors search the FACTS repository and find an operation which randomizes a data set. It might have been developed for schemes of $\mathbf{S}_{\text{PUB}}$ originally.

Another use case of collaboration is to let the community assist in solving a task:

Use Case ($\mathbf{U}_{\text{ASSISTANCE}}$). A user wants to find out if her data set can be anonymized such that her quality criteria are met. The community helps her to find suitable schemes. For example, suppose that Alice wants to outsource her data to a $\mathbf{S}_{\text{DAAS}}$ provider. She wants to know if there exists an anonymization that allows executing certain queries in under one second on her data set. Alice creates a benchmark suite with her data set and queries. Other users can retrieve it and add schemes and query-processing techniques.

Feature 5 (**Understandability**). Understandability lets one grasp with ease how experimental results have been computed.

Given an experimental result such as a diagram, it can be hard to understand how exactly it has been computed, e.g., why one value is larger than another one: For example, there may be several (parametrized) schemes and attacks, operating on different background knowledge. A use case for understandability (but also for reproducibility and collaboration) is as follows:

Use Case ($\mathbf{U}_{\text{DIAGRAM}}$). Researcher Carl is developing a new scheme. He uses FACTS to implement it and creates a new benchmark suite with performance experiments. Carl wants to graph anonymization performance, to find settings where the scheme is slow. His workflow is to implement the scheme, generate the graph, and to refine the implementation. FACTS lets Carl implement this workflow. It lets Carl attach code to his benchmark suite that generates diagrams, such as the following one:

```
1  GNUPLOT gnuplot = suite.createPlot();
2  gnuplot.addPlotOption("title", "set title 'PERFORMANCE_ANO'");
3  gnuplot.addPlotOption("styleData", "set style data histogram");
4  log.exportAllAnonymizationRuntimes(suite, gnuplot);
```

FACTS features implementations of all methods of the listing, and Carl can just call them. Line 1 creates the FACTS representation of a gnuplot diagram. Lines 2 to 3 configure the plot. For example, Line 3 configures that a histogram is the graphical representation of the data. We have left out other lines of code for gnuplot configuration to ease presentation. Finally, Line 4 instantiates the gnuplot template with the anonymization run-times of Carl's benchmark suite. The result is a gnuplot file which generates a diagram.

The conventional way for Carl to realize his workflow in turn has two steps: to run measurements and to read measurement values into a file with a program that plots results. This results in more effort because Carl must initialize the steps manually for each new optimization. Further, the information which diagram belongs to which version of Carl's code might get lost if Carl is not very careful with documentation. With FACTS in turn, plots are tied to runs of a benchmark suite. This preserves the information which diagram belongs to which experiment (**Understandability**). FACTS generates plots with every run of a benchmark suite (**Reproducibility**). FACTS separates the schema of a diagram (class `GNUPLOT`) from data (class `Log`) and stores them separately in the repository, for later reuse, e.g., to share the schema of an informative graph and to use it for different approaches (**Collaboration**).

Our final use case is to simplify benchmarks for understandability:

Use Case ($\mathbf{U}_{\text{SIMPLIFY}}$). Tony has a large data set with activities of his waste-management business. He wants his business associate Silvio to access the data, but conceal it from the authorities. This is a $\mathbf{S}_{\text{DAAS}}$ scenario and requires anonymization. However, Silvio complains that certain queries are slow. Tony lets Christopher evaluate which data the problem occurs with. To this end, Christopher gradually reduces the data set size and measures query-execution times. With FACTS, he can use methods already implemented to retrieve an evaluation data set, to reduce its size, and to start measurements. Christopher observes that processing is slow if a certain client is in the data set. Tony is now able to eliminate the problem, once and for all.

Discussion FACTS is applicable for use cases of many anonymization scenarios. With the current implementation, input and output has to be relational data. Our evaluation will show that FACTS is general enough to be applicable for the very different scenarios $\mathbf{S}_{\text{PUB}}$ and $\mathbf{S}_{\text{DAAS}}$. Data from many other scenarios, e.g., association-rule mining of shopping carts, search histories, location-based services, social networks, or statistical databases, can be represented as relations. Further diversification might require slight changes to implementations and interfaces for scenario-specific use cases. Generic use cases, e.g., performance comparisons, should work as is.

5 Evaluation

We evaluate FACTS by means of an exploratory study. We declare success if FACTS allows to model state-of-the-art schemes, attacks, and measures, and if FACTS allows to execute and to compare them by means of benchmark suites. Further, we seek confirmation that the framework indeed has the features we have identified earlier.

The evaluation of our framework is twofold. First, we have devised a benchmark suite $\mathcal{B}_{\text{pub}}$ to compare schemes of the scenario $\mathbf{S}_{\text{PUB}}$ systematically. We have evaluated the use cases $\mathbf{U}_{\text{BENCHMARK}}$ and $\mathbf{U}_{\text{DIAGRAM}}$ with it. Second, we have conducted a user study to evaluate how well FACTS realizes reproducibility and collaboration. We reenact the use cases $\mathbf{U}_{\text{ASSISTANCE}}$ and $\mathbf{U}_{\text{COMMITTEE}}$ with this study. The study has also given us a further instance of $\mathbf{U}_{\text{BENCHMARK}}$ and a benchmark suite, $\mathcal{B}_{\text{DaaS}}$, for scenario $\mathbf{S}_{\text{DAAS}}$. We stress however that our main contribution is not one specific benchmark suite but the idea of a *framework* to build and share such suites. Our website [1] holds supplementary material of the evaluation.

For $\mathcal{B}_{\text{pub}}$ and $\mathcal{B}_{\text{DaaS}}$, we first describe our evaluation setup, followed by results.

5.1 $\mathcal{B}_{\textbf{pub}}$: Benchmark for Scenario $\mathbf{S}_{\textbf{pub}}$

In this subsection we present a benchmark suite for $\mathbf{S}_{\text{PUB}}$. It tests well-known anonymity notions and schemes against data-quality, query-accuracy, and performance measures. It further tests the success of two attacks and measures anonymity.

Evaluation Setup The benchmark suite consists of the following data set, schemes, attacks, queries, and measures:

Dataset Our data set is the UCI adult data set[1] from a census in the US in 1994. Many studies on anonymization use it, e.g., [25], [27], or [30]. We use the attributes *age*, *workclass*, *education*, and *sex* as the quasi-identifying attributes and *occupation* as the sensitive attribute.

Schemes Our schemes are Datafly [34] and Incognito [25]. We run the schemes such that they comply with the anonymity notions k-Anonymity [35], l-Diversity [30], and t-Closeness [27]. Table 3 lists the respective parameter values. Parameter values are the same as in the evaluations of [27] and [30].

Table 3: Parameters for the schemes.

Scheme	**Parameters**
k-Anonymity	$k \in \{2, 4, 6, 8\}$
l-Diversity	$l \in \{2, 4, 6, 8\}$
t-Closeness	$t \in \{0.15, 0.20\}$

We run each scheme once per anonymity notion and parameter. We use the implementations of UT Dallas[2] and integrate them into FACTS with the adapter design pattern [14].

Attacks Our attacks are the homogeneity attack $\mathcal{A}_{\text{HG}}$ [27] and the functional-dependency attack $\mathcal{A}_{\text{FD}}$ [37]. With the latter, anonymity has not been measured before to our knowledge, only data quality and performance. Thus, this should give way to new insights. We have implemented both attacks against

[1] `http://archive.ics.uci.edu/ml/datasets/Adult`
[2] `http://cs.utdallas.edu/dspl/cgi-bin/toolbox/`

the FACTS attack class model. We have executed each attack on each anonymization output.

With both attacks, one kind of background knowledge is the information which exact values of quasi-identifying attributes individuals have. As far as we know, all existing studies in $\mathbf{S}_{\text{PUB}}$ assume that an adversary has this information available for all individuals. Experiments have shown that the homogeneity attack is effective with this assumption [30]. Here, we relax it and let $\mathcal{A}_{\text{HG}}$ only know the values of quasi-identifying attributes of every 100th individual. With the more recent attack $\mathcal{A}_{\text{FD}}$, we do not loosen up this assumption because $\mathcal{A}_{\text{FD}}$ was not tested with experiments at all yet. Here, however, we use other functional dependencies than previously tested: Our functional dependency is *(workclass, education, sex)* $\rightarrow$ *occupation*. [37] uses some quite unrealistic functional dependencies such as *race* $\rightarrow$ *marital-status*.

Queries Our set of queries is the same as with the evaluations of [28,29,38].

Measures Our measure for anonymity is the threat to disclose the sensitive attribute value of an individual. We refer to this as $\mathcal{M}_{\text{Anon-Generic}}$. We have implemented two instances of this measure. The first one ($\mathcal{M}_{\text{Anon-Generic}}$ with *avg*) computes the share of disclosed tuples over the total number of tuples. The second one ($\mathcal{M}_{\text{Anon-Generic}}$ with *max*) yields 1 if there has been one disclosure. For example, if $\mathcal{A}_{\text{HG}}$ finds one QI block with just one value for the sensitive attribute, $\mathcal{M}_{\text{Anon-Generic}}$ with *max* returns 1. We apply the two measures on each anonymization output for both attacks $\mathcal{A}_{\text{HG}}$ and $\mathcal{A}_{\text{FD}}$. Our measure for data quality is the *Global Certainty Penalty* measure [15] that quantifies the information loss with generalization. Our measure for query accuracy is $\mathcal{M}_{\text{Query-Acc-Avg}}$. Our measure for performance compares the absolute run-times (i) of each scheme and (ii) of each attack on the various anonymized data sets.

We have executed the benchmark suite on a machine with two octo-core CPUs with 2.9 GHz per core and 24 GB RAM.

Results

Comparability: Anonymity. It is interesting to learn which scheme can best withstand adversaries $\mathcal{A}_{\text{HG}}$ and $\mathcal{A}_{\text{FD}}$. Our measurements show that Incognito with $l = 2$ offers best anonymity against $\mathcal{A}_{\text{FD}}$. $\mathcal{M}_{\text{Anon-Generic}}$ with *max* is 0. This is because $\mathcal{A}_{\text{FD}}$ cannot use functional dependencies to link the respective sensitive attribute value to any individual. However, with all other parameter values tested, we observe disclosure. We can thus confirm vulnerability of schemes of $\mathbf{S}_{\text{PUB}}$ initialized with many parameters. With $\mathcal{A}_{\text{HG}}$ in turn, we do not observe any disclosure for any parameter values. We for our part conclude that homogeneity attacks are not as dramatic a threat with realistic background knowledge.

Understandability: Linking results to their experiment. Benchmark suite $\mathcal{B}_{\text{pub}}$ includes code that generates many diagrams to test use case $\mathbf{U}_{\text{DIAGRAM}}$. It illustrates the computation of measure values. FACTS has drawn the diagrams to reduce our evaluation effort. Figure 2 is one example. For this paper only, we slightly modified the respective FACTS-generated diagram to make it more compact. It makes a step towards understandability too: The figure has an identifier which allows us to find it within $\mathcal{B}_{\mathrm{pub}}$ and to link the experiment parameters to it.

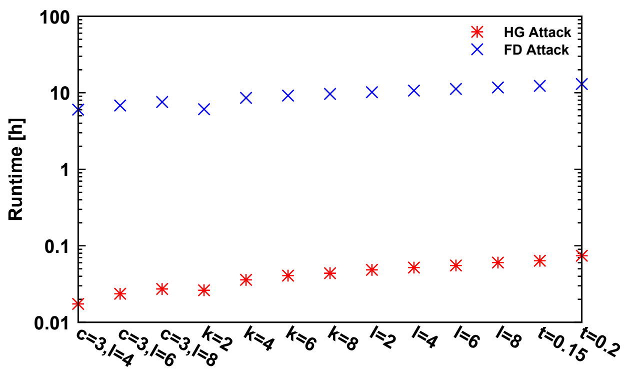

Fig. 2: Performance of attacks $\mathcal{A}_{\mathrm{HG}}$ and $\mathcal{A}_{\mathrm{FD}}$ against the output of several Datafly schemes.

5.2 $\mathcal{B}_{\mathrm{DaaS}}$: User Tasks for Anonymization, Query Processing, and Attacks

In this subsection we present a user study to evaluate how well FACTS realizes reproducibility and collaboration.
We now describe the user study that was realized as an instance of use case $\mathbf{U}_{\text{ASSISTANCE}}$ and has led to the development of benchmark suite $\mathcal{B}_{\mathrm{DaaS}}$. We have specified three tasks with $\mathbf{U}_{\text{ASSISTANCE}}$: (1) Anonymize specific data sets in a $\mathbf{S}_{\text{DAAS}}$ scenario and produce anonymization output, (2) develop query-processing techniques for each anonymization (cf. Example 2), and (3) attack the anonymization output of other study participants. In a user experiment, we have let participants solve these tasks. After the solutions were handed in, we have verified their reproducibility ($\mathbf{U}_{\text{COMMITTEE}}$). We have further compared the different solutions with performance and anonymity measures ($\mathbf{U}_{\text{BENCHMARK}}$).

Evaluation Setup. Our experiment consists of three phases where users solve different tasks with FACTS. After the three phases were completed, we handed out a user survey regarding FACTS. It is available on our website [1]. We designed the survey with care so as to not enforce positive results with the way of asking questions. Likert-scale questions did not follow patterns, i.e., positive answers have been sometimes to the left, sometimes to the right. Further, our participants answered the survey anonymously, and they knew that we could not trace negative answers back to them. We now describe the tasks, followed by a description of the participants, and incentives. Our three tasks are:

Task 1 Folksonomies [32] let users annotate digital objects with free-text labels. For example, with Last.fm, users annotate music, with Flickr photos. Folksonomies contain data that is sensitive regarding privacy. A user study [6] confirms that users see a significant benefit in being able to control who is allowed to see which data. Schemes let users only access data when the data creators have given the respective authorization. Thus, the first task is to develop schemes for CiteULike folksonomies of varying size.

Task 2 Users issue queries against folksonomies for various reasons, e.g., personal organization or communication with other users. We have identified seven types of common folksonomy queries [19]. For example, one type of query is "retrieve all tags applied to a specific object". $\mathcal{B}_{\text{DaaS}}$ includes parameters suitable for each of these seven query types for each CiteULike folksonomy data set. To continue the example, $\mathcal{B}_{\text{DaaS}}$ computes the most frequent object as one of the query parameters for each data set. This is because the most frequent object results in a large query result and thus a long query-processing time. This is an interesting extreme case that should be included in a meaningful benchmark. Thus, the second task is to develop fast processing techniques for each query type given and its parameters.

Task 3 The frequency of attribute values in folksonomies follows a power-law distribution. With improper anonymization, this leaves room for statistical attacks [8]. $\mathcal{B}_{\text{DaaS}}$ specifies as the adversary model someone with statistical background knowledge. $\mathcal{B}_{\text{DaaS}}$ computes this knowledge from the original data sets and makes the frequency of values of each attribute of the original data set available to an adversary. Thus, the third task of $\mathcal{B}_{\text{DaaS}}$ is develop attacks against the schemes developed in Task 1, given this adversary model.

Participants. We have let 19 students of computer science with a database background solve the tasks. We divided the students into four groups where three groups had five members and one group had four members. We instructed them in the fundamentals of (i) database anonymization, (ii) query processing on anonymized data, and (iii) statistical attacks. To test their understanding regarding (i) to (iii), we issued assignments to them. Two of originally 21 students did not pass them, and we did not let them participate in the subsequent evaluation.

Incentives. The participants joined the experiment as part of a practical course. Their main incentive for participation was to pass the course. To do so, participants had to earn points. Completion of the three tasks (i)-(iii) had earned them points. We had issued bonus points if participants committed their implementations of FACTS class models to the repository, or if they had developed and shared FACTS operations. Bonus points were not required to pass the course.

Results

Comparability with $\mathcal{B}_{DaaS}$ One outcome of the study has been the FACTS benchmark suite $\mathcal{B}_{\mathrm{DaaS}}$. We have imported the data set, queries, and adversary model (along with the data representing statistical background knowledge) from a previous research project of ours [19] into FACTS. $\mathcal{B}_{\mathrm{DaaS}}$ thus allows us to compare the approaches of students in an evaluation setup actually used in research. We have observed that FACTS allows us, the conductors of the study, to compare approaches with ease. We justify this claim in different ways. (1) The final result of queries on the anonymization output does always equal that of the queries on the respective original data set, for all approaches by different participants. (2) The same set of queries executes for each approach. In the past years, we had lectured this practical course without FACTS. There have been many comparison tasks that were cumbersome without the standardizations. Participants had submitted query-processing techniques that returned fewer result tuples, and they had used other query parameters than what we had specified. With FACTS, (1) its benchmarking checks correctness of results, and (2) always runs the same queries.

Reproducibility with $\mathcal{B}_{DaaS}$ We evaluate reproducibility by letting participants upload solutions and then letting them rerun them.

Our first indicator for reproducibility is if participants are able to execute approaches without errors. We say that schemes are without error if they produce an anonymization output. We say that query-execution techniques are without error if they terminate, and if they compute the correct result for all queries. We say that attacks are without error if they write their guesses for original values for each anonymized cell in the proper place for FACTS, and anonymity measures compute. A scheme writing only zeros to all cells would thus be error-free, but query execution based on it would fail. To evaluate if participants were able to reproduce the results of approaches by other participants, we have asked respective questions in the survey about the total number of schemes, queries, and attacks that participants had executed, and for how many of them participants have observed no errors. By means of answers to these questions, we have calculated the share of error-free executions, cf. Table 4. Our apriori expectations have been that the values are close to our measurements. The numbers reflect that one group has had errors with queries and attacks with our benchmark runs. The values calculated with the survey are lower, but relatively close to ours. We speculate that users not updating implementations from the FACTS repository and reporting results for non-final versions of solutions are responsible for the difference. Further, not all groups had executed every approach of every other group: It is possible that participants had executed the three error-free groups less often than the one with errors. All in all, we conclude from these observations that FACTS allows users to run approaches from the FACTS repository without difficulty and that FACTS standardizations allow to observe implementation errors that would be in the way of (fair) comparisons and reproducibility otherwise.

Our second indicator for reproducibility is if measurement values from several experiment runs on varying platforms lead to similar results. To do this com-

Table 4: Reproducibility: Error-Free Executions.

Approach	**Study Answers**	**Our Measurements**
Schemes	85 %	100 %
Query-Processing Techniques	68 %	75 %
Attacks	61 %	75 %

parison, we could rely on our execution of $\mathcal{B}_{\text{DaaS}}$ and the executions of $\mathcal{B}_{\text{DaaS}}$ by each group. We did observe similar results. For example, all performance measurements have had Group 4 as the fastest before Group 1 and Group 3 and have reported errors for Group 2. Results are not identical however because execution times depend on the computational power of clients.

We state that there is reproducibility with three of four groups ($\mathbf{U}_{\text{COMMITTEE}}$) because we were able to execute all of their approaches without error, and our measurement results were similar to theirs. We thus see strong indications that FACTS does allow for reproducibility.

Collaboration with $\mathcal{B}_{DaaS}$ We have evaluated collaboration by means of questions in the survey about operations. Question **Q11** was about the number of operations developed: 10 participants answered that their group had developed one or more operation, while 7 answered that their group had not developed any operation. Further, we have asked participants with Question **Q12** if they think that their operations are generic, i.e., how many of them can be used with other approaches in their opinion. We calculated an average share of generic operations of 55 percent (answer for **Q12** divided by answer for **Q11**, ignoring zero answers). Participants agreed that the concept of sharing operations within a community can save development time: To the respective question the mean value for answers was 3.84 on a Likert scale of 5 points (1 = strongly disagree, 5 = strongly agree). Further, participants agreed that operations are a good way to structure code: To the respective question of the survey the mean value was 3.79. These results indicate that users deem operations helpful for anonymization scenarios. Participants did not disagree to the statement of Question **Q15** that *schemes, queries, and attacks are so complex that sharing operations does not make sense because they are not applicable to more than one approach.* The mean value for answers to this question was 3.13. Some users stated in free-text comments that they were not aware of how operations might help with their complex tasks. Our conclusions is that we possibly need better documentation of how operations can help: For example, each group did use encryption and could have used the respective operation in the repository.

5.3 Current Restrictions

In this subsection, we state problems that we have encountered during evaluation or that we have learned from study participants. An example for a feature we see missing looking at other anonymization scenarios than $\mathbf{S}_{\text{PUB}}$ and $\mathbf{S}_{\text{DAAS}}$ is to have interfaces giving access to the history of query results within attacks. This would be needed to evaluate schemes referring to $\mathbf{S}_{\text{STATS}}$. We see making FACTS fit for $\mathbf{S}_{\text{STATS}}$ part of our future work. Regarding implementation, turning FACTS into a cloud-based service is future work. This and a physical platform to run experiments would give way to better usability. This is because it would free users from downloading a run-time package, and they could issue service calls instead. Finally, we tried to make it easy to integrate implementations of existing approaches ("black boxes") into FACTS. This works when existing approaches fulfill some requirements which we discuss now. Regarding anonymization, an important property is that FACTS stores the original values for each cell of the anonymization output. The relation is important for certain experiments regarding, say, data quality. External code often does not store this relation. Then it cannot be integrated into FACTS without modifying it. Regarding measures that rely on extra information produced during anonymization, one would need to adapt the framework as well. An example are measures with $\mathbf{S}_{\text{PUB}}$ that analyze the generalization hierarchy – the less generic the data, the better. For $\mathbf{S}_{\text{PUB}}$ we have developed a `DataPublishingDataset` which allows to store hierarchy information, and we have developed respective measures. We plan to develop a more general solution for storing and accessing extra information. Further, an ontology for comparability, e.g., which measure works with which attack, is future work as well. Right now we let users take care of compatibility and assume that users only bundle approaches which are compatible in benchmark suites.

6 Conclusions

Nowadays, a broad variety of anonymization approaches exists. We observe that requirements, goals, adversary models, implementations, or evaluation parameters are publicly available only for a few of them. It is very difficult to answer which approach is best regarding anonymity, data quality, query accuracy, and performance. To deal with this situation, we have proposed a framework, FACTS, that allows to compare anonymization approaches with ease. Researchers can implement their approaches within FACTS against so-called class models. We have systematically devised interfaces of class models that ease comparing and benchmarking approaches. Besides comparability, FACTS has other useful features, e.g., to support researchers in the documentation and presentation of experiment results. Our evaluation shows that FACTS allows to define comprehensive benchmark suites for anonymization scenarios, and that it addresses user needs well. Our vision is that FACTS will give way to a higher degree of comparability within the research area.

References

1. http://facts.ipd.kit.edu/.

2. J. Abramov, A. Sturm, and P. Shoval. A Pattern Based Approach for Secure Database Design. In *Proceedings of the Advanced Information Systems Engineering Workshops*, pages 637–651, 2011.
3. G. Aggarwal, M. Bawa, P. Ganesan, H. Garcia-Molina, K. Kenthapadi, R. Motwani, U. Srivastava, D. Thomas, and Y. Xu. Two Can Keep A Secret: A Distributed Architecture for Secure Database Services. In *Proceedings of the Conference on Innovative Data Systems Research (CIDR)*, pages 186–199, 2005.
4. M. Barhamgi, D. Benslimane, S. Oulmakhzoune, N. Cuppens-Boulahia, F. Cuppens, M. Mrissa, and H. Taktak. Secure and Privacy-Preserving Execution Model for Data Services. In *Proceedings of the International Conference on Advanced Information Systems Engineering (CAiSE)*, volume 7908, pages 35–50, 2013.
5. P. Bonnet, R. Johnson, D. Koop, T. Kraska, R. Müller, D. Olteanu, P. Papotti, C. Reilly, D. Tsirogiannis, C. Yu, J. Freire, S. Manegold, D. Shasha, M. Bjø rling, W. Cao, J. Gonzalez, J. Granados, N. Hall, S. Idreos, and M. Ivanova. Repeatability and workability evaluation of SIGMOD 2011. *ACM SIGMOD Record*, 40(2):45–48, 2011.
6. T. Burghardt, E. Buchmann, J. Müller, and K. Böhm. Understanding User Preferences and Awareness: Privacy Mechanisms in Location-Based Services. In *Proceedings of the International Conference on Cooperative Information Systems (CoopIS)*, pages 304–321. Springer, 2009.
7. J. Cao and P. Karras. Publishing microdata with a robust privacy guarantee. *Proceedings of the VLDB Endowment*, 5(11):1388–1399, 2012.
8. A. Ceselli, E. Damiani, S. De Capitani Di Vimercati, S. Jajodia, S. Paraboschi, and P. Samarati. Modeling and Assessing Inference Exposure in Encrypted Databases. *ACM Transactions on Information and System Security (TISSEC)*, 8(1):119–152, 2005.
9. V. Ciriani, S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati. Combining Fragmentation and Encryption to Protect Privacy in Data Storage. *ACM Transactions on Information and System Security (TISSEC)*, 13(3):1–33, 2010.
10. S. De Capitani di Vimercati, S. Foresti, S. Paraboschi, G. Pelosi, and P. Samarati. Efficient and Private Access to Outsourced Data. In *Proceedings of the International Conference on Distributed Computing Systems (ICDCS)*, pages 710–719. IEEE, 2011.
11. C. Dwork. Differential Privacy. In *Proceedings of the International Colloquium on Automata, Languages and Programming, Part II (ICALP)*, pages 1–12. Springer, 2006.
12. C. Dwork. Differential Privacy and the Power of (Formalizing) Negative Thinking. *Principles of Security and Trust*, 7215:1–2, 2012.
13. B. C. M. Fung, K. Wang, R. Chen, and P. S. Yu. Privacy-preserving data publishing: A survey of recent developments. *ACM Computing Surveys*, 42(4):1–53, 2010.
14. E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison-Wesley, 1995.
15. G. Ghinita, P. Karras, P. Kalnis, and N. Mamoulis. Fast data anonymization with low information loss. In *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, pages 758–769. VLDB Endowment, 2007.
16. H. Hacigümüş, B. Iyer, C. Li, and S. Mehrotra. Executing SQL Over Encrypted Data in the Database-Service-Provider Model. In *Proceedings of the International Conference on Management of Data (SIGMOD)*, pages 216–227. ACM, 2002.

17. H. Hacigümüş, S. Mehrotra, and B. Iyer. Providing Database as a Service. In *Proceedings of the International Conference on Data Engineering (ICDE)*, pages 29–38, 2002.
18. C. Heidinger, E. Buchmann, M. Huber, K. Böhm, and J. Müller-Quade. Privacy-Aware Folksonomies. In *Proceedings of the European Conference on Research and Advanced Technology for Digital Libraries (ECDL)*, pages 156–167. Springer, 2010.
19. C. Heidinger et al. Efficient and secure exact-match queries in outsourced databases. *World Wide Web*, pages 1–39, 2013.
20. C. Henrich, M. Huber, C. Kempka, and R. Reussner. Secure Cloud Computing through a Separation of Duties. Technical report, Karlsruhe Institute of Technology (KIT), 2010.
21. B. Hore, S. Mehrotra, M. Canim, and M. Kantarcioglu. Secure multidimensional range queries over outsourced data. *The VLDB Journal*, 21(3):333–358, 2012.
22. B. Hore, S. Mehrotra, and G. Tsudik. A Privacy-Preserving Index for Range Queries. In *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, pages 720–731. VLDB Endowment, 2004.
23. D. Kifer and A. Machanavajjhala. No Free Lunch in Data Privacy. In *Proceedings of the International Conference on Management of Data (SIGMOD)*, pages 193–204. ACM Press, 2011.
24. D. Kifer and A. Machanavajjhala. A Rigorous and Customizable Framework for Privacy. In *Proceedings of the Symposium on Principles of Database Systems (PODS)*, pages 77–88. ACM Press, 2012.
25. K. LeFevre, D. J. DeWitt, and R. Ramakrishnan. Incognito: Efficient Full-Domain K-Anonymity. In *Proceedings of the International Conference on Management of Data (SIGMOD)*, pages 49–60. ACM, 2005.
26. C. Li and G. Miklau. An adaptive mechanism for accurate query answering under differential privacy. *Proceedings of the VLDB Endowment*, 5(6):514–525, 2012.
27. N. Li, T. Li, and S. Venkatasubramanian. t-Closeness: Privacy Beyond k-Anonymity and l-Diversity. In *Proceedings of the International Conference on Data Engineering (ICDE)*, pages 106–115. IEEE, 2007.
28. T. Li and N. Li. Injector: Mining Background Knowledge for Data Anonymization. In *Proceedings of the International Conference on Data Engineering (ICDE)*, pages 446–455. IEEE, 2008.
29. T. Li and N. Li. On the Tradeoff Between Privacy and Utility in Data Publishing. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 517–526. ACM, 2009.
30. A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam. l-Diversity: Privacy Beyond k-Anonymity. In *Proceedings of the International Conference on Data Engineering (ICDE)*, 2006.
31. V. Manousakis, C. Kalloniatis, E. Kavakli, and S. Gritzalis. Privacy in the Cloud: Bridging the Gap between Design and Implementation. In *Proceedings of the Advanced Information Systems Engineering Workshops*, pages 455–465, 2013.
32. I. Peters. *Folksonomies: Indexing and Retrieval in the Web 2.0.* Walter de Gruyter, 2009.
33. E. Shmueli, R. Waisenberg, Y. Elovici, and E. Gudes. Designing Secure Indexes for Encrypted Databases. In *Proceedings of the Conference on Data and Applications Security (DBSec)*, pages 54–68. Springer, 2005.
34. L. Sweeney. Achieving k-Anonymity Privacy Protection Using Generalization and Suppression. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems (IJUFKS)*, 10(5):571–588, 2002.

35. L. Sweeney. k-Anonymity: A Model for Protecting Privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems (IJUFKS)*, 10(5):557–570, 2002.
36. P. Vandewalle, J. Kovacevic, and M. Vetterli. Reproducible research in signal processing. *IEEE Signal Processing Magazine (SPM)*, 26(3):37–47, 2009.
37. H. Wang and R. Liu. Privacy-preserving publishing microdata with full functional dependencies. *Data & Knowledge Engineering (DKE)*, 70(3):249–268, 2011.
38. X. Xiao and Y. Tao. Anatomy: Simple and Effective Privacy Preservation. In *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, pages 139–150, 2006.