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

Exploratory Paper

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

[10] 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 ($\mathbf{S}_{\text{MINING}}$). Differential Privacy [7] for example assumes independent database records – [11] 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. [3].

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, [12] uses the UCI Adult data set, while the related scheme [17] 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. Such requirements exist in various categories that are orthogonal to each other. At first, technical requirements must be considered, e.g., the memory footprint or the scalability of an anonymization scheme regarding the number of input tuples. Secondly, the anonymization scheme must consider the eventual use of the anonymized data. For example, if a scheme removes all data values that deviate from the average, but the use case needs to perform outlier detection, this scheme is inappropriate. The third category considers privacy preferences, attacker models and how sensitive information is represented in the data. For example, sensitive information could be materialized as set-valued data, e.g., from a shopping cart analysis, and an adversary might know typical shopping carts. Thus, FACTS must be flexible enough to implement a wide range of different schemes, attacks, and measures.

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. ●

Our evaluation is twofold. On the one hand, we have developed various benchmarks, including one $\mathbf{S}_{\text{PUB}}$ and one for $\mathbf{S}_{\text{DAAS}}$, described in a complementary technical report [9]. Here, 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.

2 Background: Terminology and Examples

We now introduce our terminology and discuss how schemes have been evaluated.

Anonymization. Our understanding of the term anonymity is broad and includes approaches such as encryption, see below. 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 [15] 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 [13]. We refer to it as $\mathcal{A}_{\text{HG}}$. l-Diversity [13] 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/implementer 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 [12] (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. ●

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.

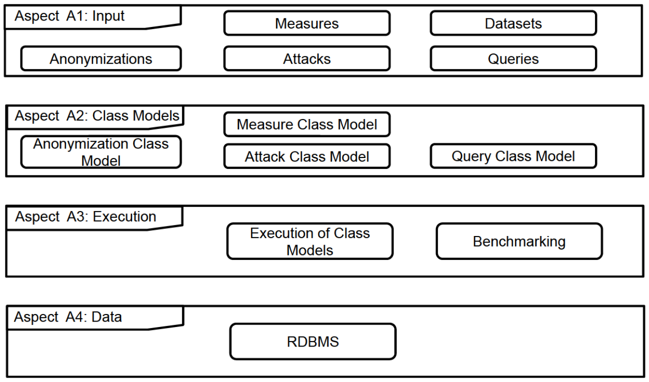

Fig. 1: FACTS – Overview.

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.

3.3 Benchmarks

We now describe how to realize benchmarks by means of the four aspects. For specifics of our implementation in Java, we refer to the documentation on the FACTS website [1].

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 $\mathit{params}(\mathit{anon})$.
3. An attack *attack*. It may have parameters, referred to as $\mathit{params}(\mathit{attack})$.
4. A set of queries Q where each $q \in Q$ may have parameters $\mathit{params}(q)$.
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.

3.4 Illustration

We now exemplarily describe how to implement an anonymity benchmark in FACTS. Our benchmark lets an adversary attack a copy of the data that has been anonymized with some scheme one wants to test. The benchmark specifies a fragment of the original data as background knowledge of the adversary, and it allows to quantify the effect of various parameters of interest. In the following, we discuss the implementation of the four aspects of FACTS.

*Aspect **A1**: Input* Listing 3 shows how to configure the input of the anonymity benchmark suite by invoking the respective methods implemented in FACTS. First, we import the input data from a file (Lines 1-3) and set up up a new experiment (Line 4-5). We further specify a scheme *anon* (Line 6) and an attack *attack* (Line 8), and link them to the experiment and the benchmark suite (Lines 7 and 9). Note that *anon* and *attack* must have been modeled in Aspect 2. Finally, we tell FACTS to use the anonymity measure *AnonDist* (Line 10). *AnonDist* implements the distance measure $\mathcal{M}_{\text{Anon-Dist}}$, as introduced in Example 5.

```
1   CSV csv = new CSV(new File(inputData));
2   Dataset input = csv.importDataset();
3   Datasets original = new Datasets().add(input);
4   Measurement 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(new AnonDist());
```

Listing 3: Configuration of anonymity benchmarks.

*Aspect **A2**: Class Models* With this aspect, we provide implementations of *anon* and *attack*, which inherit from the FACTS classes *AnonymityClassModel* and *AttackClassModel*. Listing 4 illustrates the implementation of background knowledge, which is part of *attack*. In our example, we consider the distribution of the attribute “Disease” of the original data. Thus, `backgroundKnowledge` (Line 1) has the original data set as one parameter. We use methods implemented in FACTS to count each value of the attribute named “Disease” (Lines 3-6). Another FACTS method adds the background knowledge to the framework (Line 7).

```
public void backgroundKnowledge(Dataset original, Dataset anonymized,
    Dataset preconditions, Parameters p, OperationAssembler
    knowledgeLogic)
{
 Attribute a = original.getAttribute("Disease");
 a.setAggregate(AggregateType.COUNT);
 a.setGroupByAttribute(true);
 Dataset knowledge = original.aggregateSelect(a);
 knowledgeLogic.addOperation(new NullOperation(), knowledge);
}
```

Listing 4: Statistics of original data set as background knowledge.

*Aspect **A3**: Execution* The third aspect is about running our anonymity benchmark (Listing 5). Line 1 runs the scheme *anon*, which produces the anonymized data set $anon(D)$. Next, Line 2 executes *attack*. Finally, Line 3 executes the anonymity measure. It 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 5: Execution of anonymity benchmarks.

*Aspect **A4**: Data* The final aspect is storing and logging of all classes, models and test data in a relational database. Since FACTS handles this internally, no additional code is required.

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, [4] 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 [16]. 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 the user perceive the impact of all input parameters of an experiment on the experimental results.

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. FACTS supports understandability by allowing the user to execute series of anonymization experiments with varying parameters, by providing logs of intermediate results that one can analyzes with data-analyitcs tools, and by providing convenience methods to generate diagrams. 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. 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 Our evaluation will show that FACTS is general enough to be applicable to the very different scenarios $\mathbf{S}_{\text{PUB}}$ and $\mathbf{S}_{\text{DAAS}}$. Furthermore, FACTS is directly applicable to many other scenarios where input and output data can be represented as relations, e.g., association-rule mining of shopping carts, search histories, location-based services, social networks, or statistical databases. Anonymization scenarios for continuously changing data, e.g., data streams or incrementally updated databases, would require to adapt our interfaces and their implementations of Aspects A1 and A4. However, generic benchmark functionality, e.g., performance measurements, 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.

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. It is based on an 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. Additionally, our technical report [9] includes a benchmark suite $\mathcal{B}_{\text{pub}}$, the use case $\mathbf{U}_{\text{DIAGRAM}}$ and an alternative instance of $\mathbf{U}_{\text{BENCHMARK}}$. Supplementary evaluation material is available on the project website [1].

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 $\mathcal{B}_{\text{DaaS}}$. 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 [14] 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 [5] 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 [8]. 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 [6]. $\mathcal{B}_{\text{DaaS}}$ 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 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.

Results

Comparability with $\mathcal{B}_{DaaS}$ One outcome of the study has been the FACTS benchmark suite $\mathcal{B}_{\text{DaaS}}$. 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 [8] into FACTS. $\mathcal{B}_{\text{DaaS}}$ 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 2. 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 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.

Table 2: Reproducibility: Error-Free Executions.

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

Our second indicator for reproducibility is if measurement values from several experiment runs on varying platforms lead to similar results. To do this comparison, 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 $\mathcal{B}_{DaaS}$ To evaluate the collaboration feature, we have asked respective questions in the survey. In a nutshell, users deem that the concept to collaborate with anonymization operations through the FACTS repository is useful. Our complementary technical report provides more details, also on other aspects of our evaluation.

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 *CAiSE Workshops*, 2011.
3. M. Barhamgi et al. Secure and Privacy-Preserving Execution Model for Data Services. In *CAiSE*, 2013.
4. P. Bonnet et al. Repeatability and workability evaluation of SIGMOD 2011. *ACM SIGMOD Record*, 40(2), 2011.
5. T. Burghardt et al. Understanding User Preferences and Awareness: Privacy Mechanisms in Location-Based Services. In *CoopIS*, 2009.
6. A. Ceselli et al. Modeling and Assessing Inference Exposure in Encrypted Databases. *TISSEC*, 8(1), 2005.
7. C. Dwork. Differential Privacy. In *ICALP*, 2006.
8. C. Heidinger et al. Efficient and secure exact-match queries in outsourced databases. *World Wide Web*, pages 1–39, 2013.
9. C. Heidinger et al. FACTS: A Framework for Anonymity towards Comparability, Transparency, and Sharing (Extended Version). Technical report, Karlsruhe Institute of Technology (KIT), 2013. http://digbib.ubka.uni-karlsruhe.de/volltexte/1000037502.
10. D. Kifer and A. Machanavajjhala. No Free Lunch in Data Privacy. In *SIGMOD*, 2011.
11. D. Kifer and A. Machanavajjhala. A Rigorous and Customizable Framework for Privacy. In *PODS*, 2012.
12. T. Li and N. Li. On the Tradeoff Between Privacy and Utility in Data Publishing. In *KDD*, 2009.
13. A. Machanavajjhala et al. l-Diversity: Privacy Beyond k-Anonymity. In *ICDE*, 2006.
14. I. Peters. *Folksonomies: Indexing and Retrieval in the Web 2.0*. Walter de Gruyter, 2009.
15. L. Sweeney. k-Anonymity: A Model for Protecting Privacy. *IJUFKS*, 10(5), 2002.
16. P. Vandewalle et al. Reproducible research in signal processing. *SPM*, 26(3), 2009.
17. H. Wang and R. Liu. Privacy-preserving publishing microdata with full functional dependencies. *DKE*, 70(3), 2011.