Karlsruhe Institute of Technology

Trading Indistinguishability-based Privacy and Utility of Complex Data

Zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

von der KIT-Fakultät für Informatik
des Karlsruher Instituts für Technologie (KIT)

genehmigte

Dissertation

von

Christine Schäler

aus Karlsruhe

Tag der mündlichen Prüfung: 15. Mai 2023
Erster Gutachter: Prof. Dr.-Ing. Klemens Böhm
Zweiter Gutachter: Prof. Dr.-Ing. Erik Buchmann

Acknowledgements

This thesis project would not have been possible without the support of many people.

First, I would thank all people who supported me professionally. I would thank my supervisor Klemens Böhm for the freedom to set my own research agenda, Erik Buchmann for reviewing my thesis and the BMBF for financial support. Additionally, I would thank my paper co-authors for their contributions and my colleagues for helpful feedback and refreshing evenings. Furthermore, I would thank Martin Schäler and Hans-Peter Schwefel for proofreading my thesis and my student assistants, especially Nico Weidmann, for their implementation support.

Finally, I would like to thank my family and friends. My parents Stefan and, in memoriam, Brigitte Tex, thank you for making the privilege to study possible for me. Martin Schäler, thank you for your endless support and pushing me towards finalizing this project. Lisa Schäler, thank you for your patience during my defense preparation and for making the journey to Karlsruhe with me.

Preface

Writing a thesis naturally raises the question, which grammatical person shall be used. Clearly, the thesis itself is authored by a single person. However, in the research community of the doctoral candidate, the convention is to use *we* (i.e., first-person plural), independent of the number of authors. We stick to this convention.

Abstract

The collection and processing of complex data, like structured data or infinite streams, facilitates novel applications. At the same time, it raises privacy requirements by the data owners. Consequently, data administrators use privacy-enhancing technologies (PETs) to sanitize the data, that are frequently based on indistinguishability-based privacy definitions. Upon engineering PETs, a well-known challenge is the privacy-utility trade-off. Although literature is aware of a couple of trade-offs, there are still combinations of involved entities, privacy definition, type of data and application, in which we miss valuable trade-offs. In this thesis, for two important groups of applications processing complex data, we study (a) which indistinguishability-based privacy and utility requirements are relevant, (b) whether existing PETs solve the trade-off sufficiently, and (c) propose novel PETs extending the state-of-the-art substantially in terms of methodology, as well as achieved privacy or utility. Overall, we provide four contributions divided into two parts. In the first part, we study applications that analyze structured data with distance-based mining algorithms. We reveal that an essential utility requirement is the preservation of the pair-wise distances of the data items. Consequently, we propose distance-preserving encryption (DPE), together with a general procedure to engineer respective PETs by leveraging existing encryption schemes. As proof of concept, we apply it to SQL log mining, useful for database performance tuning. In the second part, we study applications that monitor query results over infinite streams. To this end, w-event differential privacy is state-of-the-art. Here, PETs use mechanisms that typically add noise to query results. First, we study state-of-the-art mechanisms with respect to the utility they provide. Conducting the so far largest benchmark that fulfills requirements derived from limitations of prior experimental studies, we contribute new insights into the strengths and weaknesses of existing mechanisms. One of the most unexpected, yet explainable result, is a baseline supremacy. It states that one of the two baseline mechanisms delivers high or even the best utility. A natural follow-up question is whether baseline mechanisms already provide reasonable utility. So, second, we perform a case study from the area of electricity grid monitoring revealing two results. First, achieving reasonable utility is only possible under weak privacy requirements. Second, the utility measured with application-specific utility metrics decreases faster than the sanitization error, that is used as utility metric in most studies, suggests. As a third contribution, we propose a novel differential privacy-based privacy definition called Swellfish privacy. It allows tuning utility beyond incremental w-event mechanism design by supporting time-dependent privacy requirements. Formally, as well as by experiments, we prove that it increases utility significantly. In total, our thesis contributes substantially to the research field, and reveals directions for future research.

Zusammenfassung

Die Erfassung und Verarbeitung komplexer Daten, wie strukturierte Daten oder unendliche lange Datenströme, ermöglicht neuartige Anwendungen. Da viele dieser Daten privatheitskritische Informationen über die Eigentümer der Daten enthalten können, stellen die Dateneigentümer gleichzeitig Datenschutzanforderungen. Folglich verwenden Datenadministratoren Technologien zur Verbesserung der Privatsphäre. Diese werden *privacy-enhancing technologies* (PETs) genannt. Eine solche PET bereinigt die Daten gemäß einer Privatheitsdefinition. Da klassische Definitionen wie k-Anonymität keine Angreiferressourcen berücksichtigen, sind diese häufig Ziel von erfolgreichen Angriffen. Daher konzentriert sich die Wissenschaft derzeit auf Privatheitsdefinitionen auf Basis von Ununterscheidbarkeit. Beispiele hierfür sind kryptografische Definitionen auf Basis von kryptografischen Spielen, oder Differential Privacy (englisch für *differentielle Privatheit*). Eine bekannte Herausforderung bei der Entwicklung von PETs ist der Zielkonflikt zwischen Privatheit und Nutzbarkeit der Daten. Obwohl die Literatur einige Kompromisslösungen für diesen Zielkonflikt kennt, gibt es immer noch Kombinationen von Umgebung, Privatheitsdefiniton, Datentyp und Anwendung, bei denen uns geeignete Kompromisslösungen fehlen. In dieser Arbeit untersuchen wir auf Ununterscheidbarkeit basierende Privatheitsdefinition für zwei wichtige Gruppen von Anwendungen, die komplexe Datenbestände verarbeiten. Insbesondere untersuchen wir, (a) welche Privatheits- und Nutzbarkeitsanforderungen relevant sind, (b) ob bestehende PETs den Zielkonflikt ausreichend lösen und (c) schlagen neue PETs vor, die den Stand der Technik sowohl methodisch, als auch bezüglich der erreichten Privatheit und Nutzbarkeit wesentlich erweitern. Insgesamt liefern wir vier Beiträge, die in zwei Teile gegliedert sind.

Teil A. Distanzbasiertes Data Mining von verschlüsselten, komplexen Datenbeständen Heutzutage analysieren Dienstleister Datenbestände zum Beispiel für Unternehmen, die nicht über die Methodenkompetenz verfügen, um die Analyse selbst durchzuführen. Da Geschäftsdaten privatheitskritische Daten sind, ist es wichtig, dass der Dienstleister selbst nicht in der Lage sein darf, sensible Informationen aus den von ihm analysierten Daten abzuleiten. Da die Verschlüsselung ein gut etabliertes und allgemein akzeptiertes Werkzeug zum Schutz der Privatsphäre ist, untersuchen wir, wie die Mining-Ergebnisse für distanzbasiertes Data Mining Algorithmen bei der Verschlüsselung erhalten bleiben können. Da viele Mining-Algorithmen distanzbasiert sind, schlagen wir den Begriff der distanzerhaltenden Verschlüsselung (DPE) vor. DPE formalisiert die Nutzbarkeitsanforderungen, dass die Distanz der Datenelemente bei der Verschlüsselung beibehalten werden sollte. Dieser Begriff hat genau die richtige Strenge — wir zeigen, dass wir ihn nicht lockern können, sowohl mit formalen Argumenten als auch mit Experimenten. Das Entwerfen eines DPE-Schemas ist schwierig, da das Schema sowohl vom Datenbestand als auch vom

verwendeten spezifischen Distanzmaß abhängt. Wir schlagen ein Verfahren zur Entwicklung von DPE-Schemata für strukturierte Datenbestände vor, das als DisPE bezeichnet wird. In einer Fallstudie instanziieren wir DisPE für SQL-Logs, die beispielsweise zum Zweck der Performance-Verbesserung von Datenbanken analysiert werden. In dieser Studie entwerfen wir DPE-Schemata für alle SQL-Distanzmaße aus der wissenschaftlichen Literatur. Wir zeigen formal, dass man zu diesem Zweck eine Kombination bestehender sicherer Verschlüsselungsschemata verwenden kann, die bereits bewiesene Sicherheitslevel haben. Abschließend diskutieren wir die Verallgemeinerbarkeit unserer Ergebnisse anhand zweier weiterer Datenbestände. DPE wird in späteren Forschungsarbeiten für Sprachverschlüsselung angewendet.

Teil B. Differential Privacy zur Überwachung von Datenströmen Während Verschlüsselung ein allgemein anerkanntes Werkzeug zum Schutz der Privatheit ist, sind die Verarbeitungsmöglichkeiten verschlüsselter Daten stark begrenzt. Dies gilt insbesondere für Anwendungen, die Datenströmen kontinuierlich überwachen. Folglich konzentrieren wir uns bei der Datenstromüberwachung auf w-event Differential Privacy, den aktuellen Goldstandard in der Literatur. Wir liefern drei zusammenhängende Beiträge wie folgt:

B.i) Leistungsvergleich von Differential Privacy Mechanismen für Datenströme In der Literatur wurden verschiedene w-event Differential Privacy Mechanismen vorgeschlagen. Aus Sicht eines Datenadministrators stellt sich die Frage, welcher Mechanismus für ein PET zu verwenden ist. Bei einer umfassenden Literaturrecherche stellen wir fest, dass die Ergebnisse heutiger Studien jedoch kaum vergleichbar und teilweise in sich widersprüchlich sind. Unsere Recherchen legt jedoch nahe, dass alle bestehenden empirischen Studien zu w-event Differential Privacy Mechanismen durch ein Tupel aus vier Elementen beschrieben werden können: Mechanismen, Datenströme, Privatheitsanforderungen und Nutzbarkeitsmetriken. Für jedes Element skizzieren wir Limitationen bisheriger Studien und leiten Anforderungen ab, die die Vergleichbarkeit der Ergebnisse gewährleisten. Mit einer großangelegten empirischen Studie tragen wir neue Erkenntnisse zu den Stärken und Schwächen bestehender Mechanismen bei. Eines der unerwartetsten, aber dennoch erklärbares Ergebnis ist eine grundlegende Überlegenheit der Baselines. Diese besagt, dass basierend auf den Eigenschaften eines Datenstroms, die Verwendung einer der beiden Baseline-Mechanismen in eine gute, oder sogar die beste, Nutzbarkeit resultiert. Ein zweites wichtiges Ergebnis ist, dass der sogenannte mittlere Fehler, der die mittlere Differenz zwischen den wahren Daten und den bereinigten Daten beschreibt und häufig als Nutzbarkeitsmetrik verwendet wird, wenig über die Erhaltung der Eigenschaften von Datenströmen wie Saisonalität aussagt, die von vielen Anwendungen ausgenutzt werden. Abschließend formulieren wir Schritte für Praktiker, wie man sinnvolle Mechanismen auswählt, und Optionen für Forscher, wie man die Überlegenheit der Baselines durchbricht.

B.ii) Fallstudie: Nutzbarkeitsmetriken für die Überwachung von Stromverteilnetzen Das Ergebnis unseres Leistungsvergleichs wirft zwei Anschlussfragen auf. (1) Ist der mittlere Fehler eine angemessene Nutzbarkeitsmetrik, und (2) bieten die Baseline-Mechanismen bereits eine angemessene Nutzbarkeit. Diesen Fragen gehen wir daher anhand einer Fallstudie aus dem Bereich Stromnetzüberwachung nach. Ein Netzüberwachungssystem besteht aus mehreren Anwendungen, die Leistungsmessungen von Privatkunden verwenden, die private Informationen enthalten. In unserer Studie betrachten wir eine Reihe von Kandidaten für Nutzbarkeitsmetriken, die von Experten auf diesem Gebiet verwendet werden. Darüber hinaus spezifizieren wir realistische Privatheitsanforderungen, die den Schutz von typischen Stromnutzungsmustern formalisieren. Für unsere Experimente verwenden wir ein reales Stromnetz und Messungen, die mit einem bekannten Simulator generiert wurden. Die Studie zeigt zwei Ergebnisse: (1) Das Erreichen einer angemessenen Nutzbarkeit ist nur unter schwachen Privatheitsanforderungen möglich. (2) Der Nutzbarkeit der Netzüberwachung nimmt schneller ab, als es der mittlere Fehler vermuten lässt.

B.iii) Swellfish Privacy: Ausnutzung zeitabhängiger Privatheitsanforderungen Da unsere Fallstudie zeigt, dass es schwierig ist, einen angemessenen Nutzen unter w-event Differential Privacy zu erreichen, schlagen wir eine neuartige Differential Privacy-basierende Definition namens Swellfish Privacy vor. Mit Swellfish Privacy können Dateneigentümer feiner abgestufte Privatheitsanforderungen spezifizieren, die es ermöglichen, den Nutzen über das inkrementelle Mechanismendesign hinaus zu verbessern. Dabei stellen wir fest, dass die Privatheitsanforderungen für Datenströme im Allgemeinen zeitabhängig sind. Beispielsweise könnte eine Dateneigentümerin verheimlichen wollen, ob sie während der Mittagszeit (gesundes) Essen selbst gekocht, oder (ungesundes) bestellt hat. Es ist definitionsbedingt nicht möglich, solche Anforderungen in w-event Differential Privacy zu spezifizieren, was letztendlich zu weniger Nutzbarkeit als möglich führt. Daher schlagen wir Swellfish Privacy vor, das zeitabhängige Datenschutzanforderungen berücksichtigt. Anhand einer großen Fallstudie aus dem Bereich der Stromverbrauchsüberwachung zeigen wir, dass Swellfish-private Mechanismen bei zeitabhängigen Privatheitsanforderungen einen signifikant besseren Nutzen erzielen als aktuelle w-event Differential Privacy Mechanismen.

Insgesamt trägt unsere Dissertation wesentlich zum Forschungsfeld bei und zeigt Richtungen für zukünftige Forschung auf.

Contents

Abstract iii

Zusammenfassung v

List of Figures xiii

List of Tables xv

I. Introduction 1

1. Motivation 3

1.1. Thesis Scope 4

1.2. Challenges 5

1.3. Contributions 6

1.4. Thesis Outline 8

II. Distance-based Mining of Encrypted Structured Data 9

2. Fundamentals and Related Work 11

2.1. SQL Queries 11

2.2. Encryption 13

2.2.1. Encryption Schemes 13

2.2.2. Security of Encryption Schemes 15

3. Distance-Preserving Encryption 17

3.1. Definition and Engineering of Distance-Preserving Encryption Schemes 18

3.1.1. Definition 18

3.1.2. Inadequacy of Relaxed Notions 19

3.1.3. Distance-Preserving Encryption with DisPE 23

3.2. Case Study: Distance-Preserving Encryption of SQL Query Logs 24

3.2.1. Instantiating DisPE for SQL Query Logs 25

3.2.2. Generalization of our Study 35

3.3. Summary 37

III. Differentially Private Monitoring of Infinite Streams **39**

4. Fundamentals **41**
4.1. Notation 42
4.2. Differential Privacy 43
4.2.1. Differential Privacy for Static Databases 43
4.2.2. w-Event Differential Privacy for Streams 46
4.2.3. Utility Metrics 47

5. Benchmarking Differential Privacy Mechanisms for Streams **49**
5.1. Specific Fundamentals 50
5.1.1. Methodology of Literature Survey 51
5.1.2. The w-event Mechanism Framework 51
5.2. Literature Survey and Benchmarking Requirements 52
5.2.1. Mechanism Set $\mathbb{M}$ 53
5.2.2. Data Stream Set $\mathbb{S}$ 55
5.2.3. Privacy Requirements Set $\mathbb{P}$ 57
5.2.4. Error Metrics Set $\mathbb{E}$ 58
5.3. Benchmark Definition 58
5.3.1. Mechanism Set $\mathbb{M}$ 59
5.3.2. Data Streams Set $\mathbb{S}$ 61
5.3.3. Privacy Requirements Set $\mathbb{P}$ 64
5.3.4. Error Metrics Set $\mathbb{E}$ 64
5.4. Results 64
5.4.1. One-dimensional Artificial Streams 64
5.4.2. One-dimensional Real-World Streams 68
5.4.3. Multi-dimensional Real-World Streams 71
5.4.4. Takeaways 73
5.5. Summary 75

6. Case Study: Utility Metrics for Electricity Distribution Grid Monitoring **77**
6.1. Specific Fundamentals and Related Work 78
6.1.1. Grid Monitoring Systems 78
6.1.2. Differentially Private Grid Monitoring 82
6.2. Study Design – Grid Monitoring 83
6.2.1. Measuring Utility – Candidate Utility Metrics 83
6.2.2. Assessing Utility – Measurement Errors 84
6.2.3. Selection of Study Data 85
6.3. Study Design – Privacy 85
6.3.1. Selection of Mechanism 86
6.3.2. Assessing Privacy – Privacy Requirements 86
6.4. Results 89
6.4.1. Illustration of Grid Monitoring Results 89
6.4.2. (RQ1) Relations of Utility Metrics 91
6.4.3. (RQ2) Privacy-Utility Trade-Off 92

6.5. Summary 94

7. Swellfish Privacy: Exploiting Time-Dependent Privacy Requirements 95
7.1. Specific Fundamentals and Related Work 96
7.1.1. Typical Privacy Requirements in Different Use Cases 97
7.1.2. Policy-Driven Privacy 97
7.1.3. Further Related Work 99
7.2. Swellfish Privacy 99
7.2.1. Notion for Specifying Privacy Requirements 100
7.2.2. Neighboring Stream Prefixes 103
7.2.3. Swellfish Privacy 104
7.2.4. Generalization of Blowfish Privacy 105
7.3. Mechanism Design 106
7.3.1. Definition and Exploitation of the Effects 106
7.3.2. Baseline Mechanisms 109
7.3.3. Swellfish Mechanisms Exploiting the Effects 111
7.4. Results 114
7.4.1. Intrinsic Evaluation 114
7.4.2. Extrinsic Evaluation 119
7.4.3. Generalization of Case Study 120
7.5. Summary 121

IV. Conclusions 123

8. Summary 125

9. Outlook 127

Bibliography 129

List of Figures

1.1. Scope of this thesis. 4

2.1. Taxonomy of property-preserving encryption schemes. 14

3.1. Distribution of the distances in the reference and distorted matrices. . . . 22
3.2. Results of all experiments. 23

4.1. Continuous monitoring of query results. 42

5.1. Artificial stream vs. real-world stream. 62
5.2. δ_{MAE} values of the vary-ϵ and vary-w experiments. 65
5.3. Plots of the one-dimensional true query result streams. 68
5.4. Average MAE of mechanisms for one-dimensional real-world streams. . 69
5.5. Average MAE of mechanisms for multi-dimensional real-world streams. 72

6.1. Illustration of the two-step process of grid monitoring. 80
6.2. Identified utility metric types together with associated error metrics. . . 83
6.3. Grid monitoring results in the ground truth scenario. 89
6.4. Relations between error metrics of different types. 92
6.5. Utility for varying noise scales. 94

7.1. Relationship between secret pairs, stream policies, and policy collections. 100
7.2. Illustration of our definition of neighboring stream prefixes. 103
7.3. Results of the intrinsic evaluation – TeS effect. 117
7.4. Results of the intrinsic evaluation – TeNAT effect. 118
7.5. Results of the extrinsic evaluation. 120

List of Tables

1.1. Part structure of this thesis. 5

2.1. Important notation used in Part II. 11
2.2. An example for a database relation. 12
2.3. Concepts with respect to SQL queries. 12

3.1. Preserving the order of the pairwise distances upon encryption. 21
3.2. Overview of query distance measures. 27
3.3. Features of an SQL query. 29

4.1. Important notation used in Part III. 41

5.1. Overview of identified publications. 51
5.2. Function implementation of the baseline mechanisms. 53
5.3. Addressing of Requirements ($\mathbb{M}$-R1) to ($\mathbb{M}$-R5). 54
5.4. Addressing of Requirement ($\mathbb{S} - R2$) 56
5.5. Addressing of requirements with respect to $\mathbb{P}$ and $\mathbb{E}$. 58
5.6. Instantiation of the 4-Tuple $(\mathbb{M}, \mathbb{S}, \mathbb{P}, \mathbb{E})$. 59
5.7. Properties of the multi-dimensional streams. 71

6.1. Measured and calculated measurands. 80
6.2. Comparison of related work with respect to DP variant. 82
6.3. Overview of shares and window sizes. 88
6.4. Resulting minimum and maximum noise scales. 88
6.5. Overview of utility in all experiments. 90

7.1. Additional notation used in Chapter 7. 100
7.2. Overview of design of our Swellfish-private mechanisms. 112
7.3. Statistics of generated policy collections. 116
7.4. Mechanisms compared in the intrinsic evaluation. 117
7.5. Average MRE over all zones and Φ for different values of ϵ. 118

Part I.

Introduction

1. Motivation

The automatic processing of complex data, like structured data or infinite streams, facilitates a plurality of novel applications that improve our life significantly. One example are electricity grid monitoring applications processing power measurement streams that are essential to enable the energy transition [Dem+10; Str16; Sch+21]. This advent of data-driven applications, however, raises legitimate requirements of the data owners regarding data privacy. To tackle this, a common approach is to use privacy-enhancing technologies (PETs) [TM01] that sanitize the data to achieve privacy. To engineer a PET, a fundamental prerequisite is an implementable formal privacy definition. However, from society perspective, definitions are usually rather vague. For instance, asking Alan Furman Westin, the father of data privacy, privacy is “the claim of individuals, groups, or institutions to determine for themselves when, how, and to what extent information about them is communicated to others” [Wes67]. In the research community, this resulted in a plenty of definitions and PETs, that, in the end, are tailored to specific data types, applications and use cases [WE18]. Specifically, in the early days, researchers focus on definitions that measure properties of the sanitized data, like the number of data owners having the same attribute value. Examples are k-anonymity [Swe02] or l-diversity [Mac+07]. However, since they do not consider adversary capabilities or background knowledge, they are prone to certain attacks [Agg05; LLV07]. As a consequence, indistinguishably-based privacy definitions have been proposed. Here, an adversary should not be able to distinguish between, e.g., the true and sanitized data, with high probability. Examples for such definitions are provable security based on cryptographic games [Bel98; AL83], or statistic indistinguishably based on differential privacy [Dwo08; Mir17]. Consequently, we focus in this thesis on privacy definitions based on indistinguishably, namely, cryptographic definitions and differential privacy.

However, achieving “perfect privacy” with respect to these definitions is practically unfeasible. Specifically, perfect cryptographic security that presumes an adversary with unlimited computing capabilities [Sha49], is only achievable with *one-time* usable encryption keys having the same length as the data itself [Sha49]. The required key exchange is as hard as transmitting the data itself. Similarly, perfect differential privacy for numeric data, claiming exhaustive indistinguishability is only achievable by adding an infinite amount of noise to the data [Dwo+11], making any application using the sanitized data practically useless. Consequently, instead of aiming at perfect privacy, one aims at a trade-off between privacy and *utility*, known as the privacy-utility trade-off in literature [SRP13; AAL14; Deu+21]. Intuitively, the utility measures the usability of the sanitized data. Presuming indistinguishably-based privacy definitions, there are usually two options to instantiate the trade-off: (1) Fixing privacy requirements, and optimizing utility [SD13; GV15; BW18], or (2) fixing utility requirements, and optimizing privacy [ALN87; Pop+11; SEJ14]. Which

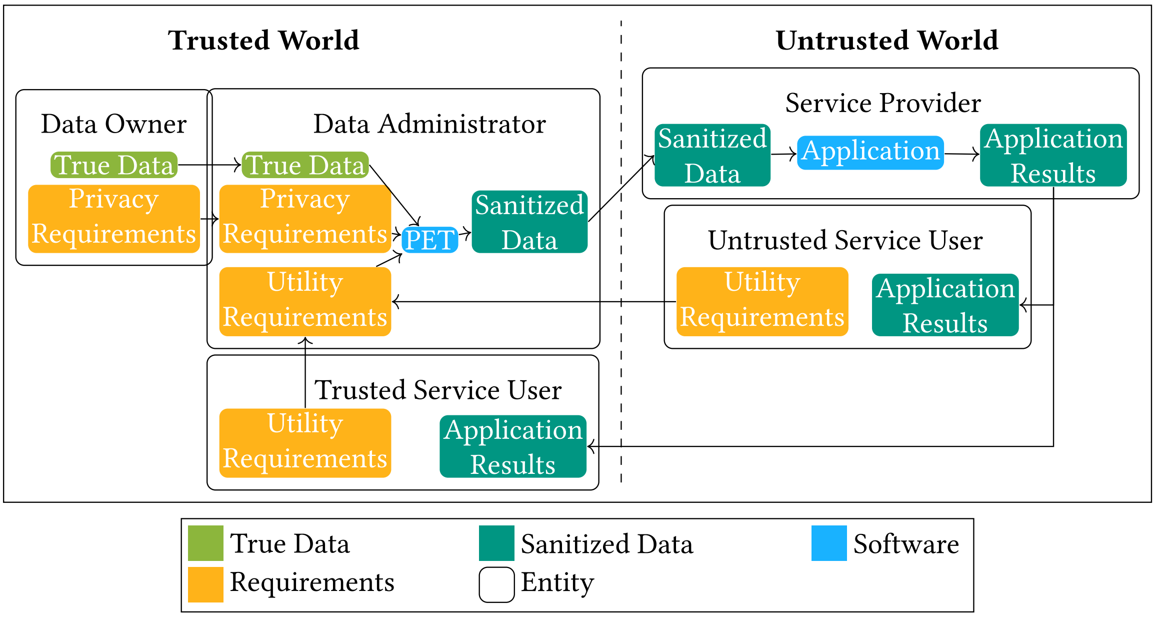


Figure 1.1.: Scope of this thesis.

option is appropriate not only depends on the scope, but also the specific privacy definition used, as we outline in the remainder.

In this thesis, we study PETs that provide valuable trade-offs between privacy and utility for indistinguishably-based privacy definitions, namely, cryptographic definitions and differential privacy. Although literature is aware of a couple of trade-offs [LBB15; EF15; SMA21], there are still combinations of scope, privacy definition, type of data and application in which we miss valuable trade-offs. Consequently, subsequently, we first state the scope of this thesis. Afterwards, we outline the open challenges with respect to this scope for both privacy definitions and different groups of application. Finally, we present our contributions that tackle these challenges.

1.1. Thesis Scope

Figure 1.1 illustrates the scope of this thesis. The dashed line separates two worlds, namely, the trusted and the untrusted world. Below, we describe both worlds.

The trusted world contains up to three entities: Data owners, data administrators and possibly service users. Data owners own the data that should be processed by an application. In line with related work on differential privacy [Pap+07; KBB15; DR+14], we name the data owned by the data owners *true data*. The data owners may have privacy requirements, stating under which prerequisites the data can be used by untrusted entities. An example for a privacy requirement in context of differential privacy is “the survey results containing my answers compared to survey results that do not contain my answers should be difficult to distinguish”. Data administrators manage the true data of the data owners. They are responsible to ensure that the data usage by service providers is in line with the privacy requirements of the data owners. In line with related work on differential

Table 1.1.: Part structure of this thesis.

Part	Service User	Indisting.-based Privacy Def.	Application Group
Part II	Trusted	Cryptographic	Distance-based data mining
Part III	Untrusted	Differential privacy	Stream monitoring

privacy [DR+14], we name this process *data sanitization.* To this end, they use a privacy-enhancing technology that sanitizes the true data, such that the sanitized data complies with the privacy requirements. The data administrator transmits the sanitized data to the untrusted world. There, service providers offer applications used by service users. To get valuable application results, the service users may have utility requirements with respect to the sanitized data. The service users can be trusted or untrusted. An example for trusted service users are the data owners itself. In this case, the data administrator can use PET that optimizes privacy for given utility requirements. A cryptographic PET is a frequent choice [ALN87; Pop+11; SEJ14]. An example for untrusted service users we find in the area of electricity grid monitoring. Here, the residential households are the data owners, but distribution grid operators are the service users. Here, to comply with law, the data administrator needs to select a PET that implements all privacy requirements, but optimizes utility. A PET based on differential privacy is a common choice here [SD13; GV15; BW18].

1.2. Challenges

In this thesis, for two important groups of applications processing complex data, we focus on solving the privacy-utility trade-off with respect to indistinguishably-based privacy definitions. These groups are applications using distance-based data mining algorithms [Def77; Est+96; PJ09; KNT00] to analyze structured data, as well as stream monitoring applications [SFY07; Zhu+00; PSW09]. For both groups, we outline the challenges below.

A. Distance-based Data Mining of Encrypted Structured Data As stated in Section 1.1, in use cases featuring trusted service providers, data administrators use PETs that optimize privacy with respect to utility requirements by using cryptographic PETs. Consequently, the first challenge is to identify the utility requirements that distance-based mining algorithms impose. Examples for potential utility requirements are preserving the exact or an approximation of the pairwise distances. Having defined a utility requirement, we need to engineer a PET that provide the best possible security while complying with the utility requirement. Engineering an encryption scheme for structured data in a way that service providers can analyze the encrypted data using distance-based mining algorithms is, however, challenging for two reasons. Namely, (i) unclear subject of encryption and (ii) distance-measure variety, that may influence each other. To illustrate, think of application analyzing SQL query logs for performance tuning reasons. Each SQL query has an inner structure. For encrypting a query, a PET would usually encrypt the whole query as a string. This results in arbitrary strings. However, data mining algorithms may request valid SQL queries as input. The latter depends certainty on the distance measure used by the service providers. Existing distance functions are conceptually different, imposing different requirements on the encryption scheme. For instance, for string distance on SQL queries, an encryption scheme that encrypts every character in the string might be suitable. In contrast, for distance measures depending on the result tuples of the queries, a fundamental requirement is the executability of the encrypted query.

B. Differentially Private Monitoring of Infinite Streams As stated in Section 1.1, when considering untrusted service providers, data administrators use PETs that optimize utility with respect to privacy requirements that are frequently based on differential privacy. This imposes the following challenges. First, in literature, various differential privacy mechanisms for streams are known. Consequently, data administrators have to select one of them. Selecting a mechanism is however challenging, because we lack consistent evaluation standards in literature. For various reasons, like usage of different data and preprocessing, studies are hardly comparable. Second, as mentioned, for a comparison of the utility of mechanisms, a utility metric is needed. Selecting a meaningful one is however challenging. In literature dominating metrics use the sanitization error that is given by the distance between true and sanitized data [BLR13; SRP13; Kel+14]. The drawback of the sanitization error is that it does not allow to draw conclusions about the influence of data sanitization of the application results. For instance, think of a sanitization mechanism that smooths a time series. While the distance between true and sanitized time series might be low, applications that detect outlier in the time series achieve far worse results on the sanitized time series. A natural alternative, that has also been considered in literature, is to use application-specific utility metrics [AS00; KFB15]. However, since individual applications feature different utility requirements on the sanitized data to achieve valuable results, the results are hard to generalize. Third, data owners have to specify privacy requirements the data administrator has to ensure. For differential private stream monitoring, the privacy requirements are expressed with two parameters: A privacy level ϵ specifying the level of indistinguishability, and the window size w specifying the maximum length of private pattern to be protected. They are fixed for the whole life-time of the stream. Selecting reasonable values for these parameters in studies is challenging. Additionally, it is not known whether PETs can achieve reasonable utility with these inflexible parameters, and how to make them more customizable to allow utility tuning by helping data owners to specify their privacy requirements more precisely.

1.3. Contributions

In this thesis, we tackle the challenges identified for each application group by studying (1) which indistinguishability-based privacy or utility requirements are relevant, (2) whether existing privacy-enhancing technologies solve the trade-off satisfactory within a case study, (3) and, if this is not the case, propose novel PETs and privacy definitions extending the state-of-the-art in terms of methodology, as well as achieved privacy or utility, substantially. Our contributions for each applications group are as follows:

A. Distance-based Data Mining of Encrypted Structured Data Nowadays, services providers offer analysis-as-a-service applications, e.g., in the cloud. Here, service providers analyze structured data relying on data mining algorithms, for example, for business organizations that do not have the methodological skills to conduct the analysis on their own. Since business data is privacy critical data, it is essential that the service provider itself must not able to infer sensitive information from the data that they analyze. Since encryption is a well-established and commonly accepted tool to preserve privacy, we investigate how to preserve the mining results for such algorithms upon encryption. Since many mining algorithms are distance-based, we propose the notion of distance-preserving encryption (DPE). DPE formalizes the utility requirement that the distance of the data items should be preserved upon encryption. This notion has just the right strictness – we show that we cannot relax it, using formal arguments as well as experiments. We propose a procedure to engineer DPE schemes, dubbed DisPE. Motivated by the fact that service users rely on the analysis of SQL query logs for, e.g., performance tuning [Sil+09; ASB17], in a case study, we instantiate DisPE for SQL query logs. In this study, we design DPE schemes for all SQL query distance measures from the scientific literature. We formally show that one can use a combination of existing secure property-preserving encryption schemes to this end. Finally, we discuss on the generalizability of our findings using two other data sets as examples. Our notation is leveraged in subsequent research and adopted to speech encryption [KLM21; Kra21; KLM22].

B. Differentially Private Monitoring of Infinite Streams While encryption is a well-established and commonly accepted tool to preserve privacy, the processing capabilities of encrypted data are highly limited. This is in particular the case for applications that monitor statistics of streaming data continuously. Consequently, for stream monitoring, we focus on w-event differential privacy, the current gold standard [Kel+14]. In this part, we provide three coherent contributions as follows:

B.i) Benchmarking Differential Privacy Mechanisms for Streams Following the proposition of w-event differential privacy, various mechanisms have been proposed. From perspective of a data administrator, this raises the question which mechanism to use. Similarly, scientific progress requires comparable empirical studies. Conducting a comprehensive literature survey, we find that the results of today's studies are hardly comparable and partially intrinsically inconsistent. However, our survey suggests that all existing empirical studies can be described by a tuple of four elements: mechanisms, streams, privacy requirements and utility metrics. For each element, we outline limitations of prior studies and derive requirements ensuring the comparability of results. Conducting a large-scale empirical study, we contribute new insights into the strengths and weaknesses of existing mechanisms. The most unexpected, yet explainable, result is a baseline supremacy. It states that one of the two baseline mechanisms is expected to deliver high or even the best utility measured by the sanitization error. A second important result is that the sanitization error, that is usually used as utility metric, reveals little about the preservation of stream properties that are usually exploited by applications. Finally, we propose steps for data administrators how to select a mechanism, and options for researchers how to break the baseline supremacy.

B.ii) Case Study: Utility Metrics for Electricity Distribution Grid Monitoring The result of our benchmark poses two follow-up questions. (1) Is the sanitization error an appropriate utility metric, and (2) do baseline mechanism provide already reasonable utility. Consequently, we tackle these questions by a case study from the area of electricity grid monitoring. A grid monitoring system consists of a plurality of applications using power measurements from residential customers containing private information. In our study, we consider a set of candidates for utility metrics, that are used by experts in the field. Additionally, we specify realistic privacy requirements that formalize the protection of power patterns. For our experiments, we use a real-world grid and measurements generated with a commonly used simulator. The study reveals two results: (1) Achieving reasonable utility is only possible under weak privacy requirements. (2) The utility of grid monitoring decreases faster than the sanitization error suggest, indicating that the utility of an application is frequently worse than literature indicates.

B.iii) Swellfish Privacy: Exploiting Time-Dependent Privacy Requirements Since our case study indicates that it is hard to achieve reasonable utility under w-event differential privacy, we propose a novel differential privacy-based privacy definition called Swellfish Privacy. With Swellfish privacy, data owners can specify finer-grained privacy requirements allowing to tune utility beyond incremental mechanism design. Towards this, we observe that privacy requirements are generally time dependent. E.g., a data owner might aim to hide whether she cooked or had (unhealthy) food delivered during lunchtime only. By design, it is not possible to specify such requirements in w-event differential privacy, ultimately resulting in less utility than possible. Therefore, we propose Swellfish privacy, which takes time-dependent privacy requirements into account. By means of a large case study from the area of power consumption monitoring, we show that, given time-dependent privacy requirements, Swellfish-private mechanisms achieve significant better utility than state-of-the art w-event differential privacy mechanisms.

1.4. Thesis Outline

The remainder of this thesis consists of three further parts. In Part II, we focus on distance-based mining of encrypted data. To this end, Chapter 2 states fundamentals regarding encryption and SQL queries. Chapter 3 proposes our novel notion of distance-preserving encryption and performs a case study on SQL query logs. Next, Part III focuses on monitoring of differentially private streams. To this end, Chapter 4 states the notation used in this part and introduces differential privacy. Chapter 5 compares existing differential privacy mechanisms for streams within a comprehensive benchmark. In Chapter 6, we conduct a case study on differentially private electricity grid monitoring. The results indicate that it is hard to achieve reasonable utility. Consequently, in Chapter 7, we propose Swellfish privacy, our novel differential privacy-based privacy definition. Respective mechanisms achieve significantly higher utility by allowing data owners to specify time-dependent privacy requirements. Finally, in Part IV, we conclude with a summary in Chapter 8 and discuss open questions for future research in Chapter 9.

Part II.

Distance-based Mining of Encrypted Structured Data

2. Fundamentals and Related Work

In the first part of the thesis, we motivated the privacy-utility trade-off and the scope of this thesis. One challenge we have discussed is how to engineer PETs that optimize privacy given utility requirements in case the data owners are the service users. In this part, we focus on how to optimize cryptographic privacy achieved with encryption for applications that are based on distance-preserving data mining algorithms. To this end, we justify the utility requirement that the PET must preserve the pair-wise distances of all data items, named *distance-preserving encryption.* The case study focuses on applications analyzing SQL query logs [Sil+09], used to tune the performance of an existing database [ASB17], or to engineer an SQL query recommendation system [AB21].

Consequently, in this chapter, we first introduce fundamentals and notation of SQL queries. Second, we state fundamentals and notation on encryption, and relate previous work on encryption to distance-preserving encryption. Table 2.1 summarizes the introduced notation and the links to the scope of this thesis.

Table 2.1.: Important notation used in Part II.

SQL query logs		
Notation	Meaning in Part II	
Rel	set of names of all relations referred to by at least one query	
Attr	set of names of all attributes referred to by at least one query	
A.Const	a constant referring to attribute $A \in Attr$	
Encryption schemes		
Notation	Meaning in Part II	Link to Thesis Scope
D	unencrypted/plain-text data set	true data
$\mathcal{D}$	set of all possible data sets	
`Gen`	key generation algorithm	
`Enc`	encryption algorithm	PET
`Enc`(D)	encrypted/cipher-text data set	sanitized data
`Dec`	decryption algorithm	

2.1. SQL Queries

Subsequently, we introduce SQL queries, and explain which kind of private information SQL queries can contain. We only explain them so far as needed for the understanding of the current thesis part, and refer to [EN00; SKS02] for further details.

Table 2.2.: An example for a database relation.

Customer				
CustomerID	FirstName	LastName	Town	Debts
1	Michael	Smith	Maryland	0
2	Michael	Wright	Orlando	1,200
3	Corinna	Wine	Vienna	3.5

Table 2.3.: Concepts with respect to SQL queries.

Concept	Example query
Arithmetic aggregate function	SELECT MAX(Debts) FROM Customer
Relation-valued aggregate function	SELECT COUNT(*) FROM Customer
Subquery	SELECT * FROM Customer WHERE EXISTS(SELECT ..))
Duplicate elimination	SELECT DISTINCT FirstName FROM Customer

SQL, also known as Structured Query Language, is a data manipulation (DML), query (DQL), definition (DDL), and control (DCL) language for relational databases. The language is standardized by ISO/IEC [Sta16], although most present-day database systems do not implement the standard in full. Generally, a relational database is a set of relations and links between relations. A relation is a table consisting of columns, also called attributes, and rows. Usually, one or more columns of a table feature as row-identifying primary key. Columns that refer to columns of other tables are called foreign keys. Table 2.2 shows an example for a relation with four attributes and three rows.

There are different syntaxes of SQL queries, depending on whether one uses SQL as data manipulation, query, definition or control language. Our case study involves all of them, except DCL, but can easily be extended to this. Most known, and therefore used for illustration in this part, are DQL queries. They query data from a database, and are also known as `SELECT` queries. Example 1 shows an example of such a `SELECT` query with a predicate in the `WHERE` clause. Table 2.3 shows example queries containing SQL constructs we refer to in this part.

Example 1 (SQL Query). *The SQL query*

SELECT FirstName FROM *Customer* WHERE Town LIKE 'Vienna'

selects the first name of all customers from Vienna.

Considering Example 1, we observe that the query also information about the underlying database. For instance, the name of the relation: *Customer.* Considering the definition of relational databases and the SQL standard, each SQL query can contain at most the following kind of information about a database: (1) a subset of the relations in the database and foreign key relationships between them, (2) a subset of attributes of the relations in (1), and (3) information about expected values of the attributes. The reason for the latter are filtering conditions, for example in `WHERE`-clauses. All three kind of information must generally not be revealed to an untrusted entity for the following reasons. First, developing a good data model, that involves the definition of relations and attributes, is challenging. Therefore, the relations and attributes are usually a company secret. Second, information about expected values of the attributes reveals information about the database rows, that contain confidential information.

To tune the performance of an existing database [ASB17], or to engineer an SQL query recommendation system [AB21], one relies on the analysis of SQL logs [Sil+09]. An SQL log D is a set of *valid* SQL queries executed over one database in a specific period of time. A query is valid iff (1) it follows the syntax used by the underlying database system and (2) refers only to relations and attributes actually existing in the underlying database. An SQL log is a list of SQL queries that have been executed over a database in a specific time period. Given an SQL log, we notate with (1) *Rel* the set of the relation names and (2) *Attr* of attribute names queried in any query in the log. Additionally, with *A.Const*, we notate a constant referring to attribute $A \in Attr$. For instance, for the query log consisting only of the query stated in Example 1, it holds: Rel = {Customer}, $Attr$ = {Customer.Town} and *Customer.Town.Const* = {Vienna}. In the remainder, $\mathcal{D}$ is the set of possible SQL logs over a database.

2.2. Encryption

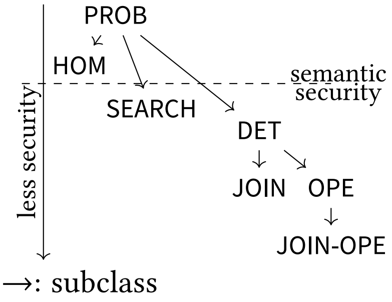
In this section, first, we introduce background knowledge on encryption schemes together with notation, and relate it to the scope of this thesis. Second, we review related work on property-preserving encryption schemes, since we leverage them to engineer distance-preserving encryption schemes.[1]

2.2.1. Encryption Schemes

In this thesis part, encryption schemes are used to sanitize the true data. Subsequently, we introduce encryption schemes as far as needed for this thesis part. For more details, we refer to [Gol09].

In order to, e.g., cluster data items, the PET must preserve the intrinsic structure of the data. Consequently, we encrypt not the whole data set as a block, but each individual item in the data set separately. In consequence, a sanitized data set is a set of sanitized data items. Definition 1 states the definition of an encryption scheme that encrypts individual data items of a data set D. Since literature on cryptography refers to true data as *unencrypted* or *plain-text data*, and to sanitized data as *encrypted* or *cipher-text data*, we stick with this notation in this part of this thesis.

[1] The following text is based on the fundamentals section in Christine Tex, Martin Schäler, and Klemens Böhm. "Towards meaningful distance-preserving encryption". In: *Proceedings of the 30th International Conference on Scientific and Statistical Database Management (SSDBM)*. ACM, 2018, pp. 1–12.



(a) Classe and their sub-classes.

Class	Stateless Schemes	Stateful Schemes
PROB	randomized AES [ST01]	-
HOM	Paillier [Pai99]	-
SEARCH	[SWP00]	-
DET	block cipher [Knu98]	-
OPE	[Bol+09]	OPES [Agr+04]

(b) Instances of classes.

Figure 2.1.: Taxonomy of property-preserving encryption schemes, inspired by [Pop+11; Li+15b].

Definition 1 (Encryption Scheme). *An encryption scheme is a Tuple (*Gen, Enc, Dec*) where*

- Gen *is a key-generation algorithm that outputs a tuple consisting of two Keys* (K_{Enc}, K_{Dec})*,*
- $\mathtt{Enc}_{K_{Enc}}$ *is an encryption algorithm that encrypts a plain-text value using Key* K_{Enc}*,*
- $\mathtt{Dec}_{K_{Dec}}$ *is a decryption algorithm that decrypts cipher-text value using Key* K_{Dec}*.*

According to Definition 1, an encryption scheme consists of three algorithms: (1) Key generation, (2) encryption and (3) decryption algorithm. The key generation algorithm generates two keys. One key is used by the encryption algorithm, and the other one by the decryption algorithm. Both keys are owned by the data administrator. He uses the same key to encrypt *all* data items in one data set. While the encryption algorithm is used for data sanitization, the decryption algorithm can revert the encryption. In the scope of this thesis, the latter is only needed if the mining results, like clusters consisting of encrypted data items, need to be decrypted to take advantage of the service. If so, the trusted service user requests the decrypted results from the data administrator. For readability, in the remainder, we abbreviate $\mathtt{Enc}_{K_{\text{Enc}}}$ with Enc and $\mathtt{Dec}_{K_{\text{Dec}}}$ with Dec. With this notation, $\mathtt{Enc}(D) = \{\mathtt{Enc}(x) \mid x \in D\}$ is the encrypted data set.

Specifications of Encryption Schemes We classify encryption schemes by the specification that they have. An example for a specification is being asymmetric or symmetric. In the following, we review the relevant specifications from literature and relate them to distance-preserving encryption.

Asymmetric vs. Symmetric In symmetric key encryption schemes, there is one key for encryption and decryption (i.e., $K_{\text{Enc}} = K_{\text{Dec}}$), while in asymmetric (or *public key*) schemes, $K_{\text{Enc}} \neq K_{\text{Dec}}$ holds. Symmetric key encryption schemes are more time efficient than asymmetric ones [AM12], but have the disadvantage that a key exchange is needed. In this part, symmetric schemes are suitable, since encryption and decryption are done by the same entity (the data administrator), i.e., no key exchange is needed. However, using a more powerful asymmetric scheme is possible as well. Our considerations in the remainder hold for both, and we do not differentiate between them.

Stateful vs. Stateless In stateful encryption schemes, the encryption algorithm has, next to the plain-text value and the encryption key, an additional input and output parameter – the state of the scheme [Bel+97]. For instance, considering stateful encryption in combination with property-preserving encryption, the state is often the set of all data items that have been encrypted so far [Agr+04]. In the most use cases, for instance in database encryption [Pop+11], the values to be encrypted are not known in advance. Stateful schemes are unpractical there. In our scenario, however, the data set to be encrypted is fixed, as the data administrator shares an already existing data set with the service provider. Therefore, we can rely on stateful schemes.

2.2.2. Security of Encryption Schemes

A PET that is based on encryption ensures privacy in the form of security. Different encryption schemes provide different levels of security, i.e., shield against different attacks. In cryptography, we differentiate between *active* and *passive* attacks. In an active attack, the adversary has access to a decryption oracle, i.e., can decrypt certain cipher-texts to some extent. A passive adversary does not have this ability. In our context, we see the service provider as a passive adversary, meaning that he must not ask the data administrator for the decryption of cipher-text values. There typically are three types of passive attacks [SK13]:

- Cipher-Text-Only Attack: The adversary has access to several cipher-texts *someone else* has selected. The adversary is successful if he can decrypt (any) randomly selected cipher-text.
- Known-Plain-Text Attack: The adversary has access to several cipher-texts *someone else* has selected and the corresponding plain-texts. He is successful if, given one cipher- and two possible plain-texts, he can determine the correct plain-text.
- Chosen-Plain-Text Attack: The adversary can ask an oracle for cipher-texts of plain-texts chosen by *himself*. He is successful if, given one challenge cipher- and two possible plain-texts, he can determine the correct plain-text. The adversary is not allowed to ask the oracle for the encryption of the challenge cipher-text.

We come back to these attacks in Section 3.2.1.1, when turning them to attacks on SQL query logs in our case study. The highest security level for passive attacks is semantic security [DH76], which is security against the chosen-plain-text attack described above.

2.2.2.1. Property-Preserving Encryption

Preserving properties of a data set (e.g., the order of the data items) upon encryption is a common utility requirement. Property-preserving encryption schemes are classified by the plain-text properties they preserve. For such a scheme, "perfect security" against passive attacks, i.e., semantic security, cannot be guaranteed for most properties. The reason is that the scheme intentionally leaks (at least) the property of the plain-text preserved. Since different classes preserve and leak different properties, different classes provide different security. In the following, we briefly explain different classes of property-preserving encryption schemes, to apply them to distance-preserving encryption. Figure 2.1 (a) depicts a taxonomy showing the relationships and the security levels of the various classes. The rows stand for the security levels, higher is better. For classes in the same row, because of incomparable, class-specific security notions, a security ranking is not possible.

Probabilistic Encryption (PROB). Encryption schemes are *probabilistic* if, in general, two equal values are mapped to different cipher-texts. PROB schemes without any additional assumption do not preserve any property of the data. But they feature semantic security and even security against active attacks.

Homomorphic Encryption (HOM) [FG07]. Homomorphic encryption schemes are probabilistic schemes allowing for arithmetic aggregate functions, e.g., sums, over encrypted data. As this property opens a way for active attacks, the security level is lower than for PROB schemes. However, beside probabilistic encryption, HOM is the only class that provides semantic security. The reason is that the leaked property is the homomorphism (e.g., regarding addition), but no information on the semantics of the actual values.

Searchable Encryption (SEARCH) [Bon+04; SWP00]. Searchable encryption allows for keyword search on encrypted data (e.g., LIKE '%unicorn%'). Thus, the property leaked is the occurrence of keywords in the plain-text.

Deterministic Encryption (DET). A scheme is *deterministic* if two equal values are mapped to the same cipher-text. Therefore, DET schemes allow for equality checks over encrypted data.

Order-Preserving Encryption (OPE) [Agr+04]. Order-preserving encryption schemes are deterministic and preserve the order of the data. Thus, one is able to perform range queries.

Join-Preserving Encryption (JOIN/JOIN-OPE) [Pop+11]. JOIN is a special usage mode of a DET scheme (JOIN-OPE of OPE, respectively), to allow for joins over encrypted databases. Particularly, the same encryption scheme and key encrypt the primary and foreign key.

For every encryption class, different encryption schemes exist. See Figure 2.1 (b) for the commonly used schemes. As mentioned in Section 2.2.1, we can use stateful schemes in this chapter. To our knowledge, except for the OPE-class, no meaningful stateful scheme exists.

3. Distance-Preserving Encryption

In the last chapter, we introduced the basics of encryption. One of the issues that we discussed is how to engineer PETs that sanitize data such that applications based on distance-based data mining algorithms have high utility.[1] This chapter proposes and justifies the novel notion of distance-preserving encryption, and examines how to engineer distance-preserving encryption (DPE) schemes for structured data of arbitrary complex types. For data sets consisting of items with a complex inner structure this is challenging for the following reasons that we illustrate below: (1) unclear subject of encryption and (2) distance-measure variety. We now illustrate these challenges using SQL query logs as a use case; this is also our running example in this part. Since query logs contain valuable information about data owner interests [ASB17; Ngu+15], they are a valuable resource to tune the performance of an existing database [ASB17], or to engineer an SQL query recommendation system [AB21].

Since SQL query log mining is far from trivial, it is reasonable to outsource it to a service provider or an external researcher. We now illustrate the challenges. First, SQL queries have a complex inner structure. For data items with a complex structure, it tends to be unclear what "encryption of the data items" actually means, see Example 2.

Example 2 (Unclear Subject of Encryption)**.** *Consider a set of SQL queries, i.e., an SQL query log. Which parts of the query should be encrypted? For example, one may encrypt the query string as a whole or only the tokens within the query string.*

Thus, when designing a DPE scheme, one must specify a security model tailored to the type of data considered and an encryption scheme in line with this model. Second, for any type of data, there exist different distance measures. Example 3 illustrates that distance-preserving encryption depends on the measure in use.

Example 3 (Distance-Measure Variety)**.** *For distance-based data mining over an SQL query log, one needs a distance measure. Different such measures exist. For instance, we can use a string distance measure like the Levenshtein distance or a measure that depends on the overlap of the tuples in the results of the queries. These measures are conceptually different: For example, two queries may lead to the same result even if the query string is different. Therefore, different distance-preserving encryption schemes are needed: For string distance, it*

[1] The remainder of this chapter bases on the articles Christine Tex, Martin Schäler, and Klemens Böhm. "Distance-Based Data Mining Over Encrypted Data". In: *2018 IEEE 34th International Conference on Data Engineering (ICDE)*. IEEE. 2018, pp. 1264–1267 and Christine Tex, Martin Schäler, and Klemens Böhm. "Towards meaningful distance-preserving encryption". In: *Proceedings of the 30th International Conference on Scientific and Statistical Database Management (SSDBM)*. ACM, 2018, pp. 1–12. Compared to the articles, the sections have been shortened to be less repetitive, contain minor corrections, as well as formatting and notation changes to be in line with the format and structure of this thesis.

might be reasonable to encrypt every character in the query string. In contrast, for distance measures depending on the result tuples of the queries, a fundamental requirement is the executability of the encrypted query. The reason is that the result tuples and their overlap cannot be computed otherwise.

Thus, when designing a DPE scheme, one must differentiate between the different distance measures.

Contributions and Outline In this chapter, we examine how to engineer distance-preserving encryption schemes for structured data. To this end, we split this chapter into three sections: In Section 3.1, we define distance-preserving encryption for arbitrary data sets and distance measures. Using formal arguments and experiments, we justify that our notion has just the right strictness. In addition, we specify a general procedure for designing distance-preserving encryption schemes named DisPE. This procedure describes the steps necessary to arrive at a distance-preserving encryption scheme for a certain data set. In Section 3.2, we examine which property-preserving encryption schemes introduced in the previous chapter, data administrators can apply when implementing distance-preserving encryption schemes for SQL logs. As result, we find DPE schemes for four well-known SQL query distance measures. The schemes have a higher security level than the ones CryptDB [Pop+11] would generate, i.e., shield against more attacks. Finally, by means of two further use cases, namely XQuery and relational data, we say to which extent our case-study results can be leveraged, and what remains to be done. Finally, in Section 3.3, we provide a summary of this chapter.

3.1. Definition and Engineering of Distance-Preserving Encryption Schemes

In this chapter, we examine how to design distance-preserving encryption schemes for complex-structured data. First, we define distance-preserving encryption for arbitrary data sets and distance measures. Second, we show formally and by experiments that it would not make sense to work with a notion that is less strict. Third, we specify a general procedure for designing distance-preserving encryption named DisPE. This procedure describes the steps necessary to arrive at a distance-preserving encryption scheme for a certain data set. We review well-known property-preserving encryption schemes from literature to illustrate (a) that one can apply them to instantiate our DisPE procedure (b) to assess the security of the resulting distance-preserving encryption scheme.

3.1.1. Definition

With distance-preserving encryption (DPE), the pairwise distances for the plain-text and the cipher-text data items must be the same.

Algorithm 1 k-medoids

```
1:  procedure K-MEDOIDS($d$, $k$, $D$)
2:      $M \leftarrow \forall x, y \in D$: pairwise distance $d(x, y)$
3:      select $k$ data items as medoids
4:      for $x \in D$ do
5:          associate $x$ with the closest medoid
6:      end for
7:      cost $\leftarrow \sum_{\{\text{medoid } M\}} \sum_{\{\text{all items } x \text{ associated to } M\}} d(M, x)$
8:      while cost decreases do
9:          for Medoid $M$ and Associated Item $x$ do
10:             swap the role of $M$ and $x$
11:             recompute cost
12:             if cost increased then
13:                 redo swap
14:             end if
15:         end for
16:     end while
17: end procedure
```

Definition 2 (Distance-Preserving Encryption (DPE))**.** *Let D be a data set,* d *be a distance measure and* Enc *an encryption algorithm for data items in D. Then,* Enc *is* d*-distance preserving if*

$$\forall x, y \in D : d(\mathsf{Enc}(x), \mathsf{Enc}(y)) = d(x, y).$$

Distance-preserving encryption enables distance-based data mining on encrypted data sets. This means that the mining results on cipher-text and plain-text data are the same. For instance, the same data items are assigned to clusters. To justify our definition, we explain its usefulness with the help of the k-medoids clustering algorithm [PJ09], a very common distance-based clustering algorithm. Algorithm 1 is the k-medoids clustering algorithm. It works as the k-means algorithm [HPK11], with the difference that the cluster centers always are objects occurring in the data set.

Theorem 1. *If* Enc *is* d*-distance-preserving, given the same initialization in Line 2, it holds that*

$$\mathsf{Enc}(\textit{k-medoids}(d, k, D)) = \textit{k-medoids}(d, k, \mathsf{Enc}(D)).$$

Proof. In the k-medoids algorithm stated in Algorithm 1, the only information extracted from the data set are the pairwise distances of the data items (Line 2). □

As the argumentation of Theorem 1 holds for all distance-based data mining algorithms, Theorem 1 holds for any of them.

3.1.2. Inadequacy of Relaxed Notions

The equality condition regarding all pairwise distances in Definition 2 is a strict requirement. One may ask whether an α-approximation of the distance value such as

$$|d(\mathsf{Enc}(x), \mathsf{Enc}(y)) - d(x, y)| \leq \alpha \cdot d(x, y),$$

or preserving the relative ordering such as

$$d(x, y) < d(x, z) \iff d(\mathsf{Enc}(x), \mathsf{Enc}(y)) < d(\mathsf{Enc}(x), \mathsf{Enc}(z))$$

is sufficient. This is not the case. First, we illustrate this with individual counterexamples, and second with experiments.

3.1.2.1. Counterexamples

First, to understand that using an approximate value is not sufficient, consider Example 4.

Example 4 (Approximative Distance-Values)**.** *Consider a data set $D = \{w, x, y, z\}$ where the pairwise distance between the data items is given by the following distance matrix:*

$$M = \begin{array}{c|cccc} & w & x & y & z \\ w & 0 & 0.1 & 0.11 & 0.11 \\ x & 0.1 & 0 & 0.11 & 0.11 \\ y & 0.11 & 0.11 & 0 & 0.1 \\ z & 0.11 & 0.11 & 0.1 & 0 \end{array}.$$

A k-medoids clustering with $k = 2$ results in the following clusters: $\{w, x\}$ and $\{y, z\}$. An approximation of the distances with $\alpha = 0.1$ may result in a distance matrix where the distance between all data items is given by

$$M = \begin{array}{c|cccc} & w & x & y & z \\ w & 0 & 0.11 & 0.1 & 0.1 \\ x & 0.11 & 0 & 0.1 & 0.1 \\ y & 0.1 & 0.1 & 0 & 0.11 \\ z & 0.1 & 0.1 & 0.11 & 0 \end{array}.$$

The Items w and x are not in the same cluster, as well as y and z, because their pairwise distances are higher than the other pairwise distances. Thus, the result of the clustering on encrypted data is different.

To understand why preserving the order of the pairwise distance is not sufficient, see Example 5.

Example 5 (Preserving Relative Order)**.** *Consider the example distances in Table 3.1. Again, we perform a k-medoids clustering with $k = 2$ on the plain-text data. This results in the following clusters: $\{w, x\}$ and $\{y, z\}$. Now we encrypt the data set. As we see in Table 3.1, the order of the distances is the same for the plain-text and the encrypted items. Now consider a variant of the k-medoids algorithm having a threshold for increasing the outlier robustness: If a data item x has, say, a distance ≥ 0.9 to all medoids, x is deemed an outlier and not assigned to a cluster. As we see, this is the case for item y in the cipher-text data, but not in the plain-text data, as $d(y, z)$ differs. Thus, the result of the k-medoids algorithm differs, because it outputs an outlier when performed on cipher-text data, but not on the plain-text data.*

To conclude, the pathological examples in this section indicate that ensuring the equality of the distances upon encryption is necessary to guarantee that mining results on plain-text and cipher-text data are the same. In Section 3.1.2.2, we also show this for real data with experiments.

Table 3.1.: Example of preserving the order of the pairwise distances upon encryption; distances in ascending order.

item pair	distance on plain-text data	distance on cipher-text data
(w, x)	0.1	0.1
(y, z)	0.11	0.95
(y, w)	0.5	0.96
(y, x)	0.51	0.97
(w, z)	0.52	0.98
(x, z)	0.53	0.99

3.1.2.2. Experiments

The last section shows with counterexamples that ensuring the equality of distances upon encryption is necessary to guarantee that we have the same mining results on plain-text and cipher-text data. Now, we substantiate these findings with experiments on real data. Specifically, we show that preserving the exact distances upon encryption is necessary to deploy encryption schemes independent of the specific data-mining algorithm in use. In the following, we first describe our experimental setup, then the results.

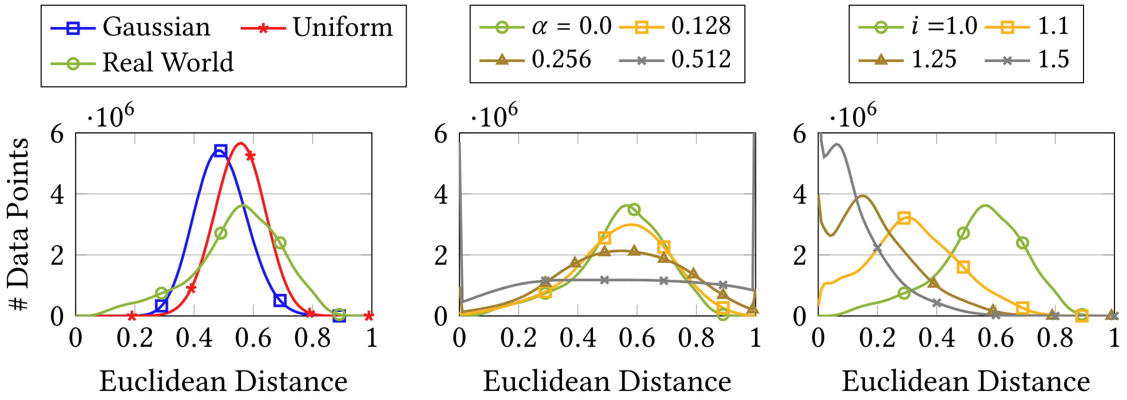
Experimental Setup We describe the experiments, the data and algorithms used.
Experiments We conduct two Experiments E1 and E2 on both alternative notions:

(E1) We quantify how much preserving an α-approximation of the distances upon encryption falsifies the results of distance-based data mining.

(E2) We quantify how much preserving the relative order of distances upon encryption falsifies the results of distance-based data mining.

Both experiments follow the following procedure: First, we determine the distance matrix of the items of our data sets. We name this matrix the *reference matrix*. Then we generate *distorted matrices* as follows: In E1, we distort the values in the reference matrix by adding noise so that we have approximations of the distances, for different approximation values α. In E2, inspired by [Won+09], we distort the values by using the i-stretching Function $s_i(v) = v^i$. Since $d(x, y) \geq 0$, the Function $s_i(d(x, y))$ increases monotonically, thus, distortion preserves the relative order of the distances. Finally, we compare the results of different distance-based mining algorithms on (a) the reference matrix and (b) distorted matrices. We always use the same parameter settings of the algorithms for (a) and (b).

Data Sets In total, we conduct both experiments on three data sets. We use two synthetic data sets – a Uniform and a Gaussian-distributed data set – as well as one real-world data set [AA96; Sch+13]. All three have dimensionality dim = 16 and consist of 10,992 data items. We normalize the distances with min/max normalization to the interval [0,1]. Figure 3.1 (a) graphs the distribution of the distances in the reference matrices. The distributions in the reference matrices are different for our three data sets. This indicates that our results are somewhat general. To get an intuition regarding the influence of the distortion in



(a) Reference Matrix (b) Distorted Matrices in E1. (c) Distorted Matrices in E2.

Figure 3.1.: Distribution of the distances in the reference matrices, and the distorted matrices for the real-world data set. The graphs share the same y-axis.

our experiments, see Figure 3.1 (b) for Experiment E1 and 3.1 (c) for Experiment E2. The distortion in E1 leads to a Uniform distribution of the distances, the one in E2 to denser distribution of the data items. The Figures 3.1 (b) and (c) only graph the distributions for the real-world data set, but the distributions of the distorted distance matrices for the Uniform and Gaussian distributed data set are similar.

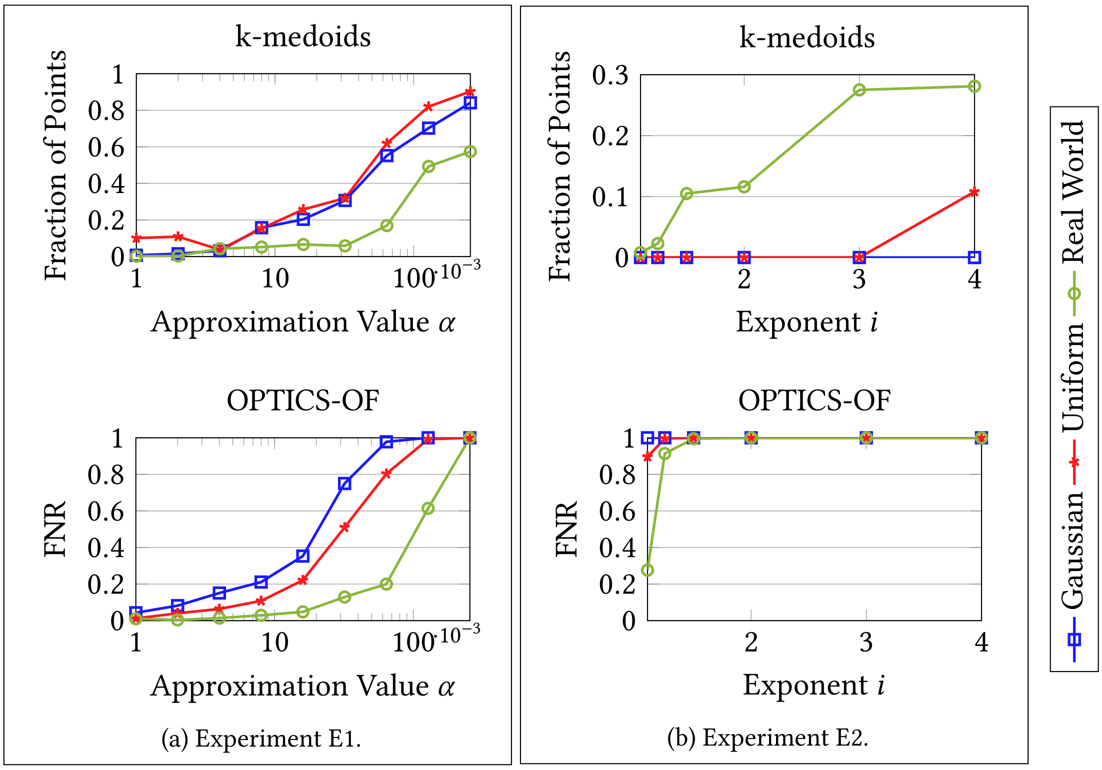
Data-Mining Algorithms We conduct the experiments with two types of distance-based data mining algorithms: clustering and outlier detection. For clustering, we use the k-medoids [PJ09] algorithm with Euclidean distance and Parameter $k = 10$. For outlier detection, we use the OPTICS-OF [Bre+99] algorithm. We select the parameters of OPTICS-OF so that they are optimal for the original data.

Results Figure 3.2 graphs the results of our experiments for all algorithms and data sets. We use the following error measures: For k-medoids algorithm, we determine the fraction of data items assigned to the wrong cluster. For OPTICS-OF algorithm, we use the false negative rate (FNR), i.e., the rate of outlier classified as inlier. – We now describe the results of both experiments, followed by a discussion.

Experiment E1 Figure 3.2 (a) graphs the results of E1. As expected, for almost all data sets and algorithms, the error increases with increasing distortion, i.e., α-values, from 0 until (nearly) 1. An error value of 1 is the highest possible error value. Thus, even for small values of α, the falsification of the clustering result is high.

Experiment E2 Figure 3.2 (b) shows the results of E2. Again, the error increases with higher distortion. Differently from E1, the error values do not exploit the full range of [0,1]. In particular, the error of k-medoids clustering remains small, i.e., ≤ 0.3. In contrast, the error with OPTICS-OF is nearly constantly 1, i.e., maximal. The reason is that stretching leads to smaller distances (cf. Figure 3.1 (c)), i.e., denser regions. The data items are moving closer together, and the outliers move inwards as well.

Discussion Preserving approximation distances or the relative ordering of the distances upon encryption falsifies the results of distance-based data mining algorithms, but to very



(a) Experiment E1.

(b) Experiment E2.

Figure 3.2.: Results of our experiments. Each sub-figure, i.e., box, encloses one experiment. For each sub-figure, the first graph shows the results for k-medoids, and the second one for OPTICS-OF. All graphs share the same legend.

different extents. While preserving approximation distances leads to high falsification for both algorithms, the falsification if preserving the relative ordering depends on the specific algorithm. This implies that, if the PET does not preserve the exact distances upon encryption, finding an appropriate encryption scheme not only depends on the distance measure, but *also on the specific algorithm in use.* This is undesirable: Organizations typically need different analyses, with different algorithms, from a service provider, and the necessity of transmitting several encrypted variants of the data would be difficult to impossible to convey. Our conclusion is that distance-preserving encryption is the only feasible option.

3.1.3. Distance-Preserving Encryption with DisPE

In the current part, we aim at finding secure distance-preserving encryption schemes for complex-structured data. To this end, we introduce the general procedure for designing distance-preserving encryption DisPE for structured data sets. We instantiate it for SQL queries in Section 3.2. Now, in turn, we remain on an abstract level. The four steps of DisPE are:

1. **Definition of the Security Model:** In the first step, one has to specify the security goals one wants to achieve with encryption, e.g., “the SQL log should not reveal information

on the content of the database". To this end, one has to (1) specify the threat model, i.e., the attacks to shield against, and (2) define a high-level encryption scheme for the type of data considered, e.g., "encrypt all constants in the query".

2. **Finding a Suitable Equivalence Notion:** Our aim is to implement the high-level encryption scheme defined in the first step so that it is distance-preserving. The distance between data items is defined for pairs of items, but encryption is done item-wise. Therefore, we introduce an intermediate notion defined for single data items: For a given distance measure, an *equivalence notion* captures the characteristics of a single data item that should be preserved upon encryption.

   Definition 3 (c-Equivalence)**.** *Let $D \in \mathcal{D}$ be a data set and* $\mathrm{c} : D \rightarrow D$ *a function (*characteristic*). In addition, let* Enc *be an encryption algorithm for data items in D. Then* Enc *ensures c-equivalence if*

   $$\forall x \in D : \mathtt{Enc}(c(x)) = c(\mathtt{Enc}(x)).$$

   In Definition 3, $\mathcal{D}$ is the set of all possible subsets of existing data items. For instance, when encrypting graph data, $\mathcal{D}$ is the set of all possible sets of graphs. Here, a Characteristic c to be preserved could be the number of incoming edges of vertices in the graph.

3. **Ensuring the Equivalence Notions:** In this step, one has to implement the high-level encryption scheme defined in Step 1 so that it ensures the equivalence notion defined in the second step. To this end, we deploy property-preserving encryption classes introduced in Section 2.2.2.1. Instead of specifying concrete schemes, we specify the corresponding encryption class, as the property to preserve is defined at class level and holds for all schemes belonging to the class. The data administrator then selects a scheme in the class (cf. Figure 2.1 (b)). In general, there are several encryption classes which can ensure an equivalence notion. We always select the appropriate encryption class according to Definition 4. As encryption-class taxonomy, we use the one from Figure 2.1.

   Definition 4 (Appropriate Encryption Class)**.** *For a given equivalence notion and encryption algorithm, an encryption class is appropriate according to an encryption-class taxonomy if*

   a) it ensures the equivalence notion and
   b) provides the highest security possible.

4. **Security Assessment:** Finally, we have to assess the security of the encryption scheme implemented. If one uses only schemes whose security is known from the literature, the security assessment is given; this is the desired case. Otherwise, a security analysis as in [BCO11] is needed.

3.2. Case Study: Distance-Preserving Encryption of SQL Query Logs

In this study, we examine which property-preserving encryption schemes one can apply when implementing distance-preserving encryption schemes for SQL logs. First, we instantiate our DisPE procedure with SQL query logs. As a result, we find DPE schemes for four well-known SQL query distance measures. The schemes have a higher security level than the ones CryptDB [Pop+11] would generate, i.e., shield against more attacks. Second, by means of two further use cases, namely XQuery and relational data, we say to which extent our results can be leveraged and what remains to be done.

3.2.1. Instantiating DisPE for SQL Query Logs

Subsequently, we instantiate our DisPE procedure with SQL query logs as our use case. After this, we explain which parts of our instantiation of DisPE one can reuse when dealing with types of data other than SQL logs.

3.2.1.1. Security Model

Next, we introduce the threat model for attacks on SQL query logs considered in this chapter and a high-level encryption scheme for SQL queries.

Threat Model Below, we (1) explain a general threat model and (2) instantiate it for SQL query logs. For the instantiation, we leverage the solution in [SK13]. In the scenario addressed here, an organization and a service provider share an encrypted SQL log. The service provider has the opportunity to perform certain attacks on the encrypted log. Our goal is to ensure security in the form of confidentiality for the underlying database. This means that we want to limit the information one can infer from the log about (1) names of relations occurring in the database, (2) names of attributes of the relations and (3) the content of the database. As stated in Section 2.2.2, one has to shield against passive attacks only. Hence, it is necessary to transform the abstract passive attacks explained in Section 2.2.2 to query logs. Literature already features this [SK13], and the transformations are as follows:

- Cipher-Text Only Attack → Query-Only Attack: In a Query-Only Attack, the adversary only has access to the encrypted query log and tries to infer the plain-text values of constants, relation names as well as attribute names[2] of a given encrypted query. For instance, the adversary has access to an encrypted query log and tries to learn information from this that helps him to decrypt a constant in a specific query.
- Known-Plain-Text Attack → Known-Query Attack: In a Known-Query Attack, the adversary has access to a number of plain-text/cipher-text query pairs, and has to distinguish between two new cipher-text queries.
- Chosen-Plain-Text Attack → Chosen-Query Attack: In a Chosen-Query Attack, the adversary has black-box access to an encryption oracle, i.e., the ability to encrypt queries, and has to distinguish between two cipher-text queries that he must not encrypt by using the oracle.

[2] Strictly speaking, the thread model in [SK13] is only defined for constants, but the model can be generalized to relation and attribute names.

Encryption of SQL Queries Intuitively, SQL queries can be encrypted in various ways, for instance by encrypting the query string as a whole. However, if we want to hide the names of relations, attributes and values of the attributes in the database only, it is sufficient to encrypt only these parts of the queries. This encryption technique for SQL queries is also known as "encryption-aware query rewriting" and has the feature that it even hides the fact that the log is encrypted [Pur+16]. While the idea of encryption-aware query rewriting has been around, it has not been studied so far how to instantiate it so that the encryption scheme for SQL queries is distance preserving.

Definition 5 (Encryption Scheme for SQL Queries). *For*

$$i \in \{Rel, Attr\} \cup \{A.Const \mid A \in Attr\},$$

let the Tuple

$$\mathtt{S}^i = (\mathtt{Gen}^i, \mathtt{Enc}^i, \mathtt{Dec}^i)$$

be an encryption scheme. An encryption scheme for SQL queries is a Tuple ($\mathtt{Gen}$*,* $\mathtt{Enc}$*,* $\mathtt{Dec}$*) where*

- $\mathtt{Gen} = (\mathtt{Gen}^{Rel}, \mathtt{Gen}^{Attr}, \{\mathtt{Gen}^{A.Const} \mid A \in Attr\})$,
- $\mathtt{Dec} = (\mathtt{Dec}^{Rel}, \mathtt{Dec}^{Attr}, \{\mathtt{Dec}^{A.Const} \mid A \in Attr\})$ *and*
- $\mathtt{Enc} = (\mathtt{Enc}^{Rel}, \mathtt{Enc}^{Attr}, \{\mathtt{Enc}^{A.Const} \mid A \in Attr\})$.

The scheme $\mathtt{S}^i$ in Definition 5 is used to encrypt $i \in \{\text{Rel}, \text{Attr}\} \cup \{A.\text{Const} \mid A \in \text{Attr}\}$, i.e., the names of relations, attributes and the constants belonging to different attributes in an SQL string.

Example 6. *For* $Q =$ *'*$\mathtt{SELECT}\ A1\ \mathtt{FROM}\ R\ \mathtt{WHERE}\ A2 > 5$*'*
the encrypted query is

$$\begin{aligned}\mathtt{Enc}(Q) = \text{'}&\mathtt{SELECT}\ \mathtt{Enc}^{Attr}(A1)\\ &\mathtt{FROM}\ \mathtt{Enc}^{Rel}(R)\\ &\mathtt{WHERE}\ \mathtt{Enc}^{Attr}(A2) > \mathtt{Enc}^{A2.Const}(5)\text{'}.\end{aligned}$$

Observe that an encryption scheme for SQL queries as in Definition 5 is not limited to SQL strings, but works on any string. We use this generalization to encrypt query results.

3.2.1.2. Suitable Equivalence Notions

In this section, we define suitable equivalence notions (cf. Definition 3) for different SQL query distance measures. We first define all notions, before we investigate how to implement encryption schemes for SQL queries that fulfill the notions in Section 3.2.1.3. By doing so, we follow the DisPE procedure. Table 3.2 gives an overview of query distance measures from literature and the core results of this and the following section. The measures follow a natural ordering: Within query-string distance, an SQL query is considered simply as a string. Query-structure distance then takes the structure of the query into account. Finally, query-result and query-access-area distance are based on the semantics, i.e., result and execution, of the query. To compute the latter two measures, it is not sufficient to only

Table 3.2.: Overview of query distance measures.

Distance Measure	Shared Information			Equivalence Notion	c	$\texttt{Enc}^{\text{Rel}}$	$\texttt{Enc}^{\text{Attr}}$	$\texttt{Enc}^{A.\text{Const}}$
	Log	Content	Domains					
Token-Based Dist.	✓	✗	✗	Token Equi.	*tokens*	DET	DET	DET
Query-Structure Dist.	✓	✗	✗	Structural Equi.	*features*	DET	DET	PROB
Query-Result Dist.	✓	✓	✗	Result Equi.	*result_tuples*	DET	DET	via CryptDB [Pop+11]
Query-Access-Area Dist.	✓	✗	✓	Access-Area Equi.	$access_A$	DET	DET	via CryptDB, excp. HOM

share the query log. For instance, to calculate query-result distance, the content of the database is needed as well (cf. Table 3.2). We discuss this issue when introducing the measures.

Token-Based Query-String Distance We now introduce token-based query-string distance and its underlying equivalence notion called token equivalence.

To define query-string distance, one interprets an SQL query as a string and uses a String-Distance Measure *string_dist* to calculate the distance, cf. Definition 6.

Definition 6 (Query-String Distance)**.** *Let $Q1$ and $Q2$ be SQL queries. The query-string distance of $Q1$ and $Q2$ is*

$$d_{String}(Q1, Q2) = string\_dist(Q1, Q2).$$

There are different types of string-distance measures [CRF+03], but not all of them are adequate for SQL queries:

Edit Distance. The edit distance $d_{\text{edit}}(s_1, s_2)$ is the minimum number of edit operations (insert, delete, substitute) needed to transform s_1 in s_2. However, intuitively, the edit distance is not appropriate for SQL strings. For example, the edit distance of two SQL queries which only differ in the ordering of the relations in the FROM-clause may be very high. Therefore, we exclude the edit distance from our further considerations.

Token-Based. Token-based distance measures divide a string into tokens (words). Thus, every Query Q is represented as a set of tokens, short *tokens*(Q), cf. Example 7. We summarized predicates of the form $A \; \Theta \; B$ to one token to avoid the intuitive issue that predicates like $A < B$ and $A > B$ results in the same token set. Then set distance measures such as the Jaccard coefficient or cosine distance (via text-to-vector) are used.

Example 7. *The token set of the Query*

$$Q = \text{'}\texttt{SELECT}A\ \texttt{FROM}\ R\ \texttt{WHERE}\ B > 5'$$

is given by

$$tokens(Q) = \{\texttt{SELECT}, A, \texttt{FROM}, R, \texttt{WHERE}, B > 5\}.$$

In the remainder of this chapter, we focus on token-based query-string distance, short token-based distance, as stated in Definition 7.

Definition 7 (Token-Based Query-String Distance)**.** *Let $Q1$, $Q2$ be SQL queries. Then the token-based query-string distance between $Q1$ and $Q2$ is*

$$d_{Token}(Q1, Q2) = 1 - \frac{|\text{tokens}(Q1) \cap \text{tokens}(Q2)|}{|\text{tokens}(Q1) \cup \text{tokens}(Q2)|}.$$

Obviously, to calculate this distance, only the queries itself (i.e., the log file) must be shared.

For token-based query-string distance, the characteristic to be preserved is the token set of the queries, i.e., $c = tokens$. Note that in Definition 8, the encryption algorithm Enc for SQL queries is used to encrypt a token set. Token-wise encryption using the corresponding scheme according to Definition 5 implements this. For instance, in Example 8, the Token A is encrypted using $\texttt{Enc}^{\text{Rel}}$.

Definition 8 (Token Equivalence)**.** *Let Q be a query. Then* Enc *ensures token equivalence for Q if the following holds:*

$$\texttt{Enc}(\text{tokens}(Q)) = \text{tokens}(\texttt{Enc}(Q)).$$

Lemma 3.2.1. *Let* Enc *be an encryption algorithm for SQL queries. If* Enc *ensures token equivalence, then it is token-based query-string distance-preserving.*

Proof. Let $Q1$ *and* $Q2$ *be SQL queries. If token equivalence holds, we get the same (encrypted) token set if we compute it on the plain-text or on cipher-text queries. Obviously, the cardinalities* $|\text{tokens}(Q1) \cap \text{tokens}(Q2)|$ *and* $|\text{tokens}(Q1) \cup \text{tokens}(Q2)|$ *remain the same. Hence,* d_{Token} *remains the same.* □

Query-Structure Distance The next distance measure for SQL queries is query-structure distance. We now state its definition and underlying equivalence notion.

In [Kho+10], the authors extract semantically important features from a query, dubbed *features*(*Q*). A feature of a query is a tuple representing a part of its structure. Table 3.3 is a complete list of features an SQL query can have and Example 8 shows an example.

Example 8. *Consider the Query Q from Example 7. The feature set of Q is given by*

$$\text{features}(Q) = \{(\texttt{SELECT}, A), (\texttt{FROM}, R), (\texttt{WHERE}, B >)\}.$$

Now, query-structure distance is defined as stated in Definition 9.

Definition 9 (Query-Structure Distance)**.** *Let* $Q1$, $Q2$ *be SQL queries. Then the query-structure distance between* $Q1$ *and* $Q2$ *is*

$$d_{Struc}(Q1, Q2) = 1 - \frac{|\text{features}(Q1) \cap \text{features}(Q2)|}{|\text{features}(Q1) \cup \text{features}(Q2)|}.$$

To calculate query-structure distance, only the queries themselves (i.e., the log file) must be shared between the organization and the service provider.

Structural equivalence ensures that there is no difference if we first compute the features of a query and then encrypt the feature set or vice versa. In Definition 10, we encrypt features of queries, which is realized as explained for token equivalence in Section 3.2.1.2.

Definition 10 (Structural Equivalence)**.** *Let Q be a query.* Enc *ensures structural equivalence for Q if the following holds:*

$$\texttt{Enc}(\text{features}(Q)) = \text{features}(\texttt{Enc}(Q)).$$

Lemma 3.2.2. *Let* Enc *be an encryption algorithm for SQL queries. If* Enc *ensures structural equivalence, then* Enc *is query-structure distance-preserving.*

Proof. Analogously to the proof of Lemma 3.2.1, as the same arguments regarding set cardinalities hold. □

Table 3.3.: Features of an SQL query [Kho+10].

Feature	Explanation
(FROM, R)	for every relation, view and relation-valued aggregate function R in the FROM-clause
(C, A)	for every attribute A in the clause C ∈ {SELECT, WHERE, GROUP BY}
(SELECT, aggr($A_1, .., A_n$)	for every aggregate function *aggr* and list of attributes $A_1, .., A_n$ in the SELECT-clause
(WHERE, $A_1 \Theta A_2$)	for every pair of attributes A_1, A_2 and operator Θ that appears in the WHERE-clause
(WHERE, $A\Theta$)	for every predicate A Θ *const* with attribute A and operator Θ in the WHERE-clause
(subquery, o)	for every subquery *subquery* that appears in WHERE-clause in combination with a set operator o ∈ {All, Any, Some, In, Exists} in a predicate

Query-Result Distance The third SQL query distance measure proposed in the literature is query-result distance. We now state its definition and underlying equivalence notion.

An SQL Query Q can be represented as the set of tuples in its result [Akb+10]. This set is given by *result_tuples*(Q). A result tuple is defined by the unique combination of attribute values. Since *result_tuples*(Q) is a set, there are no duplicates. Every query is interpreted as a query with the DISTINCT-operator. Query-result distance is defined as stated in Definition 11.

Definition 11 (Query-Result Distance)**.** *Let* $Q1$*, and* $Q2$ *be two SQL queries. Then the query-result distance between* $Q1$ *and* $Q2$ *is*

$$d_{Res}(Q1, Q2) = 1 - \frac{|\text{result\_tuples}(Q1) \cap \text{result\_tuples}(Q2)|}{|\text{result\_tuples}(Q1) \cup \text{result\_tuples}(Q2)|}.$$

It is important that the tuples in *result_tuples*(Q) depend on the state of the database. To calculate them, we must requery the database. Thus, besides the query log itself, it is necessary to share parts of the *encrypted* database as well. In particular, one needs to share the content of all attributes that are accessed by at least one query in the log. I.e., indices and constraints are not needed. In Table 3.2, we refer to the part of the database that needs to be shared as Content.

For query-result distance, the Characteristic c to be preserved is $c = \textit{result\_tuples}$.

Definition 12 (Result Equivalence)**.** *Let* Q *be a query. Then* Enc *ensures result equivalence for* Q *if*

$$\texttt{Enc}(\text{result\_tuples}(Q)) = \text{result\_tuples}(\texttt{Enc}(Q)).$$

Lemma 3.2.3. *Let* Enc *be an encryption algorithm for SQL queries. If* Enc *ensures result equivalence, it is result distance-preserving.*

Proof. Intuitively, in case result equivalence holds, the number of tuples in the two sets *result_tuples*($Q1$) ∩ result_tuples($Q2$) *and* result_tuples($Q1$) ∪ result_tuples($Q2$) *is the same for plain-text and cipher-text queries. Thus, the analogous argumentation as stated in Lemma 3.2.1 holds here as well.* □

Query-Access-Area Distance The last query-distance measure from literature is query-access-area distance, which is based on the part of the data space accessed by a query.

The access area of a Query Q is the part of the data space accessed by Q. Hence, it is a generalization of the result of a query that is independent from the current state of the database.

Definition 13 (Access Area [Ngu+15])**.** *Let Q be a query. The access area of Q,* acccess*(Q), contains all tuples that (1) might exist in at least one state allowed by the database, and (2) whose removal from at least one state would change the result of Q.*

Definition 14 (Attribute-Access Set of a Query [Ngu+15])**.** *An Attribute A is in the Attribute-Access Set* Attr_Q *of a Query Q if A occurs in the* `FROM`*-(as join attribute),* `WHERE`*-,* `GROUP BY`*-,* `HAVING`*- or* `ORDER BY`*-clause of Q.*

Definition 13 defines the access area of *one* query. With the query-access-area distance, we want to compare the access areas of *two* queries. According to [Ngu+15], this comparison takes place attribute-wise, and every attribute that is in the attribute-access set of one query is considered. Note that Definition 14 leaves aside attributes that occur only in the `SELECT`-clause of a query. Next, we compare access areas regarding the attributes in the access sets.

Definition 15 (Access Area Regarding Attribute)**.** *For a Query Q, the access area regarding an Attribute A,* $\mathrm{acccess}_A(Q)$*, is the part of the domain of A accessed by Q.*

To compare the access areas regarding attributes, we distinguish between three cases, as given in Definition 16, with an overlap-quantification Value x, which is a design parameter.

Definition 16 (Query-Access-Area Distance)**.** *Let $Q1, Q2$ be SQL queries and $Attr_{Q1,Q2}$ the set of attributes accessed by $Q1$ or $Q2$. Then the access-area distance of $Q1$ and $Q2$ is*

$$d_{AE}(Q1, Q2) = \frac{1}{|Attr_{Q1,Q2}|} \cdot \sum_{A \in Attr_{Q1,Q2}} \delta_A(Q1, Q2)$$

where

$$\delta_A(Q1, Q2) = \begin{cases} 0 \ \textit{if} \, \mathrm{acccess}_A(Q1) = \mathrm{acccess}_A(Q2) \\ x \ \textit{if} \, \mathrm{acccess}_A(Q1) \cap \mathrm{acccess}_A(Q2) \neq \emptyset \\ 1 \ \textit{otherwise} \end{cases}$$

for $x \in (0, 1)$ with a default value of 0.5.

To calculate access-area distance, in contrast to result distance, the content of the underlying database is not needed, but the constraints defining the domain of the attributes. See Table 3.2 and Example 9.

Example 9. *Consider the Queries $Q1$ and $Q2$ defined as follows:*

$$Q1 = \text{'}\texttt{SELECT * FROM}\ R\ \texttt{WHERE}\ A > 5\text{'},$$
$$Q2 = \text{'}\texttt{SELECT * FROM}\ R\ \texttt{WHERE}\ A > 3\text{'}.$$

If the domain of Attribute A in the database is given by $dom(A) = [0, 100]$, then it holds $d_{AE}(Q1, Q2) = \delta_A(Q1, Q2) = x$. In case $dom(A) = [10, 100]$, it is $d_{AE}(Q1, Q2) = \delta_A(Q1, Q2) = 0$.

With access-area equivalence, it does not matter whether we first encrypt the query and then compute its access area regarding all attributes or vice versa.

Definition 17 (Access-Area Equivalence)**.** *Let Q be a query and* $\mathrm{acccess}_Q$ *its attribute-access set. Then* Enc *ensures access-area equivalence for Q if the following holds:*

$$\forall A \in Attr_Q : \mathsf{Enc}(\mathrm{acccess}_A(Q)) = \mathrm{acccess}_A(\mathsf{Enc}(Q)).$$

Lemma 3.2.4. *Let* Enc *be an encryption algorithm for SQL queries. If* Enc *ensures access-area equivalence, then* Enc *is query-access-area distance-preserving.*

Proof. Definition 16 states that Enc *is query-access-area distance-preserving if $\delta_A(Q1, Q2) = \delta_A(Enc(Q1), Enc(Q2))$ for every Attribute A. If* Enc *ensures access-area equivalence, this equality is given, with the analogous arguments as in Lemma 3.2.1.* □

3.2.1.3. Ensuring Equivalence Notions

In this section, we show how to select the appropriate property-preserving encryption class (cf. Definition 4) to ensure all equivalence notions. To this end, we implement the encryption algorithms $\mathsf{Enc}^{A.\mathrm{Const}}$, $\mathsf{Enc}^{\mathrm{Attr}}$ and $\mathsf{Enc}^{\mathrm{Rel}}$ for each equivalence notion.

Token Equivalence For token-based string-distance-preserving encryption, we must ensure token equivalence. Lemma 3.2.5 lists the appropriate encryption classes.

Lemma 3.2.5 (Appropriate Encryption Classes for Token Equivalence)**.** *To ensure token equivalence, the following encryption classes are appropriate:*

- $\mathsf{Enc}^{A.\mathrm{Const}} = \mathsf{DET}$,
- $\mathsf{Enc}^{\mathrm{Attr}} = \mathsf{DET}$ and
- $\mathsf{Enc}^{\mathrm{Rel}} = \mathsf{DET}$.

Proof. For every encryption algorithm, according to Definition 4, we have to show that (1) the class ensures token equivalence, and that (2) we cannot use an encryption class with a higher security level. We show this as follows:

Algorithm Enc$^{A.\mathrm{Const}}$. *A constant in a query is an element of the token set of the query, as illustrated in Example 7. To facilitate token equivalence, which is an equality check of two sets, we cannot rely on the most secure probabilistic classes (i.e., PROB, SEARCH or HOM), because two constants with the same value result in different cipher-texts. Thus, the constants must be encrypted with a deterministic scheme, and* $\mathsf{Enc}^{A.\mathrm{Const}} = \mathsf{DET}$ *for every Attribute A is appropriate.*

Algorithm Enc$^{\mathrm{Attr}}$. *With analogous arguments as for* $\mathsf{Enc}^{A.\mathrm{Const}}$, $\mathsf{Enc}^{\mathrm{Attr}} = \mathsf{DET}$ *is appropriate.*

Algorithm Enc$^{\mathrm{Rel}}$. *With analogous arguments as for $Enc^{A.Const}$ and Enc^{Attr},* $\mathsf{Enc}^{\mathrm{Rel}} = \mathsf{DET}$ *is appropriate.*

Thus, we can find appropriate classes for all three algorithms. □

Structural Equivalence Lemma 3.2.6 states the appropriate encryption classes for structural equivalence.

Lemma 3.2.6 (Appropriate Encryption Classes for Structural Equivalence)**.** *To ensure structural equivalence, the following encryption classes are appropriate:*

- $\mathsf{Enc}^{A.\mathrm{Const}} = \mathsf{PROB}$,
- $\mathsf{Enc}^{\mathrm{Attr}} = \mathsf{DET}$ *and*
- $\mathsf{Enc}^{\mathrm{Rel}} = \mathsf{DET}$.

Proof. For the three algorithms, the following arguments hold:

Algorithm Enc$^{A.\mathsf{Const}}$. *As Table 3.3 shows, features of queries do not contain constants. In particular, feature extraction removes the constants from the predicates. Thus, since constants are irrelevant for feature sets of queries, the choice of* $\mathsf{Enc}^{A.Const} = \mathsf{PROB}$ *for any Attribute A is appropriate, which is the encryption class with the highest level of security.*

Algorithm Enc$^{\mathsf{Attr}}$ and Algorithm Enc$^{\mathsf{Rel}}$. *Attribute and relation names are part of features (cf. Table 3.3). Analogously to token equivalence, deterministic schemes are appropriate.*

Thus, we can find appropriate classes for all three algorithms. □

Result Equivalence A recent proposal dubbed CryptDB proposed in [Pop+11] already ensures result equivalence. A core observation in [Pop+11] is that the selection of the encryption class of $\mathsf{Enc}^{A.\mathrm{Const}}$ depends on Attribute A and its use in query predicates.

Example 10 (Ensuring Result Equivalence). *Consider the Predicate '$A = 5$'. Here, a deterministic encryption scheme for* $\mathsf{Enc}^{A.Const}$ *is sufficient. Next, consider the Predicate '$A \leq 5$'. Now, an order-preserving scheme is needed. If A is used within the* `SUM`*-aggregate function, i.e.,* `SUM`*(A),* and *in a range query, a homomorphic* and *an order-preserving scheme is needed. In this case,* $\mathsf{Enc}^{A.Const}$ *consists of two schemes. I.e., in the database, the attribute is encrypted with both encryption schemes, and during encrypting the query, the correct scheme depending on the predicate is used.*

To conclude, we can use CryptDB to find appropriate encryption classes for $\mathsf{Enc}^{A.\mathrm{Const}}$. However, CryptDB is a *database* encryption approach and supports ad-hoc queries over encrypted data. As we encrypt *fixed* SQL logs, we can modify CryptDB's approach, resulting in a higher level of security, as we describe later on. For instance, we can use stateful schemes to instantiate the classes.

Lemma 3.2.7 (Appropriate Encryption Classes for Result Equivalence). *To ensure result equivalence, the following encryption classes are appropriate:*

- $\mathsf{Enc}^{A.\mathrm{Const}}$ as stated in CryptDB [Pop+11],
- $\mathsf{Enc}^{\mathrm{Attr}} = \mathsf{DET}$ and
- $\mathsf{Enc}^{\mathrm{Rel}} = \mathsf{DET}$.

Proof. For the three algorithms, the following arguments hold:

Algorithm Enc$^{A.\mathsf{Const}}$. *Acording to the occurrences of attribute A in predicates, as described in CryptDB [Pop+11].*

Algorithms Enc$^{\mathsf{Attr}}$ and Enc$^{\mathsf{Rel}}$. *To ensure result equivalence, one must preserve the names of relations and attributes, to ensure that query execution accesses the correct relations and attributes. Thus, a deterministic scheme is appropriate.*

Thus, we can find appropriate classes for all three algorithms. □

Query-result distance differs from the previous measures: For the previous ones, one only has to share the log. For result distance in turn, the encrypted content of the database must be shared as well. To this end, it is necessary to encrypt not only the log, but also the database. Thus, we use $\mathtt{Enc}^{A.\text{Const}}$, $\mathtt{Enc}^{\text{Attr}}$ and $\mathtt{Enc}^{\text{Rel}}$ for database encryption as well. As we see in Section 3.2.1.4, our security model captures the security of the log file *and* the database.

Differences to CryptDB We see two differences to our approach that give way to achieve a higher level of security:

- As stated in Section 2.2.1, we can use stateful or stateless encryption schemes. In turn, CryptDB cannot rely on stateful encryption schemes. For attributes encrypted with an OPE scheme, this yields a higher security level. Experimental results in [Pop+11] indicate that up to 15.4% of the attributes occurring in real-world logs are encrypted with an OPE scheme.
- In our setting, the organization does not have to encrypt/share the whole database, but only the relations/attributes accessed by queries in the log.

Discussion. In contrast to token and structural equivalence, not all SQL queries allowed by the SQL standard can be encrypted in a way that ensures result equivalence [Pop+11]. This is an inherent issue we must be aware of. For instance, queries containing arbitrary pattern-matching LIKE predicates besides keyword search, i.e., predicates that SEARCH schemes can handle, are not supported. However, the results in [Pop+11] suggest that this affects only a small share of queries occurring in a real-world log. Experiments conducted there show that, except for a query log over a calendar database containing many string operations, at most 1.2% of the attributes in the database cannot be encrypted, over five data sets.

Access-Area Equivalence In contrast to result equivalence, to ensure access-area equivalence, one has to preserve the result tuples of every query with regard to every possible state of the database. Nevertheless, we can choose nearly the same encryption classes as for result equivalence, see Lemma 3.2.8. The only special cases are attributes that occur only in the arithmetic aggregate functions SUM and AVG in the SELECT-clause, i.e., not anywhere else in a query in the log. Hence, this generally leads to a higher security level. However, as described in Section 2.2.2.1, the difference is only relevant when considering active attacks.

Lemma 3.2.8 (Appropriate Encryption Classes for Access-Area Equivalence)**.** *To ensure access-area equivalence, the following encryption classes are appropriate:*

- $\mathtt{Enc}^{A.\text{Const}} = \begin{cases} \mathtt{PROB} \text{ if } A \text{ occurs in } \mathtt{SUM/AVG} \text{ in } \mathtt{SELECT} \text{ only} \\ \text{as stated in CryptDB [Pop+11] otherwise} \end{cases}$
- $\mathtt{Enc}^{\text{Attr}} = \mathtt{DET}$ and
- $\mathtt{Enc}^{\text{Rel}} = \mathtt{DET}$.

Proof. The same arguments as in Lemma 3.2.7 apply, as the proof is based on the semantics of the operators of the relational algebra (e.g., selections) whose access space is independent from the state of the database. The only exception from Lemma 3.2.7 is the first case of $\mathsf{Enc}^{A.\mathrm{Const}}$. *The reason is as follows: For result equivalence, we must choose the* `HOM`*-class, if an arithmetic aggregate function is used in a query in the log. For access-area equivalence, we do not have to and can use (the most secure)* `PROB`*-class, because the semantics of the aggregate function in the* `SELECT`*-clause does not have any influence on the access area. This area only depends on the grouping applied. This also holds if there is no grouping. Thus, in case of an attribute to which only queries with an aggregate function in the* `SELECT`*-clause refer to, we use the* `PROB` *class.* □

As for result equivalence, we can use stateful encryption to instantiate the encryption classes. In contrast to result equivalence, we do not have to encrypt the content of the database, but only need to include the minimum and maximum values of the attribute domains of the attributes, see Example 9.

3.2.1.4. Security Assessment

As we have used known property-preserving encryption schemes from literature, their level of security is known, and we can reduce the security of the schemes to the security of the encryption schemes for SQL queries we have just specified.

Example 11 (Security Assessment)**.** *As the Figure 2.1 (b) and Table 3.2 show, we often use the* `DET`*-class for encryption and recommend the* AES *scheme to instantiate it. As mentioned in Section 2.2.2.1,* `DET` *schemes are not secure against chosen-plain-text attacks. But so far, there does not exist any successful Known-Plain-Text or Cipher-Text-Only Attack against* AES[3] *that is executable in reasonable time, so we can use the scheme here without any disadvantage.*

In particular, as shown in [SK13] and stated once more in Lemma 3.2.9, an encrypted database is as least as secure against a query log attack as our SQL query encryption schemes are against attacks on the database. In other words, if the same encryption schemes are used, an adversary cannot infer more information from the encrypted query log than from an encrypted database.

Lemma 3.2.9 (Security Assessment of Query Log Attacks [SK13])**.** *Let* $\mathcal{A} \in$ *{Cipher-Text-Only Attack, Known-Plain-Text Attack, Chosen-Plain-Text Attack}. Then a database is at least as secure against* $\mathcal{A}$ *in terms of query-log attacks as against* $\mathcal{A}$ *in terms of database attacks.*

As we have used property-preserving encryption schemes known from literature, their level of security is known. Thus, we can reduce the security of the schemes to the security of the encryption schemes for SQL queries we have just specified [SK13]. In particular, one can adapt the results from [NKW15] analyzing the security of databases encrypted with property-preserving encryption schemes. This is intended – executing a full security analysis for organizations that want to outsource data analysis is practically impossible.

To conclude, we have achieved distance-preserving encryption for every distance measure with the highest security level possible.

[3] `https://www.schneier.com/blog/archives/2012/03/can_the_nsa_bre.html` (accessed 09/2017)

3.2.2. Generalization of our Study

Now, we examine how to reuse our concepts when designing distance-preserving encryption schemes for data different from SQL logs. Our main objective is to see which parts of the proof are generally applicable, and which extra effort is necessary to apply DisPE. We illustrate our thoughts by strictly following the steps of DisPE with two types of data sets. The first one is a query log in another language, namely, XQuery [Cha+03]. The second one is data stored in a database relation, called *relational data* in the remainder.

3.2.2.1. Security Model

As mentioned, the security model definition step consists of two parts: (1) specifying the threat model, and (2) definition of a high-level encryption scheme containing all algorithms one has to implement. Regarding the first part, the justification why we consider passive adversaries only is independent of the data and only depends on the scenario. Hence, it is generally valid.

The high-level encryption highly depends on the type of data considered. However, a structured data item consists of two parts: (a) a complex type consisting of several elements, which in turn may have a complex or atomic type and is formed according to a grammar and (b) the data associated with the atomic elements. In the most use cases, it is sufficient to encrypt (b), e.g., the relation and attribute names and the constants in an SQL query, as they contain the confidential information. Examples 12 and 13 illustrate this.

Example 12 (High-Level Encryption Scheme for XQuery)**.** *An XQuery statement contains names of XML-elements and attributes as well as attribute values that are confidential. Thus, we define the high-level encryption scheme for (1) XML-element names, (2) attribute names and (3) attribute values for all attributes. As a result, we have a tuple of encryption schemes with three components, the same as for SQL queries.*

Example 13 (High-Level Encryption Scheme for Relational Data)**.** *Think of customer data stored in a table with relation and attribute names as well as attribute values that are confidential. Thus, we specify the high-level encryption scheme consisting of an encryption scheme for (1) relation names, (2) attribute names and (3) values of every attribute. As a result we have a tuple of encryption schemes with three components, very similar to the one for SQL queries.*

3.2.2.2. Suitable Equivalence Notions

An equivalence notion is always defined for a specific distance measure. However, as Examples 14 and 15 illustrate, the distance measures for SQL queries may be useful for other data sets as well in many cases, or variations are used, as Example 16 illustrates. In other words, the equivalence notions introduced earlier are meaningful in broader contexts.

Example 14 (Equivalence Notions for XQuery.)**.** *Token-based string distance, structure and result distance also are suitable distance measures for XQuery logs. Therefore, we can leverage the respective equivalence notions. Access-area distance in turn is not suitable here:*

In general, there is no underlying schema. Hence, it is not obvious at all how to generalize the notion of "access area" for XQuery.

Example 15 (Equivalence Notions for Relational Data – Same Distance Measures)**.** *For relational data, token and structure equivalence are applicable, while result and access-area equivalence are specific for query languages.*

Example 16 (Equivalence Notions for Relational Data – New Distance Measure)**.** *Suppose that one wants to use the token-based distance to analyze the data, but Attribute A should be excluded, i.e., is not part of the token set. Thus, we have a variant of the token-based distance measure and need a variant of token equivalence.*

3.2.2.3. Ensuring Equivalence Notions

We cover three cases, describing the relationship of the equivalence notions defined in Step 2 to our SQL equivalence notions, ordered by the extent of reuse options, in a decreasing manner:

a) Same Equivalence Notions If the token or structural equivalence is used, one can use the same encryption schemes as in our case study in Section 3.2. For instance, this is the case for all distance measures or equivalence notions for the customer data. For result and access-area equivalence, one cannot directly adapt the schemes for $\mathsf{Enc}^{A.\mathrm{Const}}$ for SQL queries, i.e., the CryptDB-approach, because of different execution semantics.

Example 17 (Ensuring Equivalence Notions for XQuery)**.** *A core difference between SQL and XQuery is the use of path expressions, i.e., XPath. For result equivalence, one must ensure that the evaluation of XPath location steps is the same on plain-text and on cipher-text data. As we preserve the tree structure of the XML-trees (cf. Example 12), "preserving the locations steps" only implies the usage of deterministic schemes for relation and attribute names and ensuring the correct evaluation of predicates, as for SQL.*

b) Variants of the Equivalence Notions As Example 16 illustrates, one tends to use variants of the equivalence notions introduced in our study. In this case, one can reuse parts the encryption schemes the high-level scheme consists of.

Example 18 (Ensuring Equivalence for Relational Data)**.** *Consider the equivalence notion from Example 16. It differs from token equivalence in that the distance measure does not use Attribute A. Thus, we can encrypt the attribute values of the Attribute A with a probabilistic scheme, i.e.,* $\mathsf{Enc}^{A.Const} = \mathsf{PROB}$*, and the other encryption algorithms are the same as for SQL queries.*

c) Other Equivalence Notions In case other equivalence notions are used, one must implement the encryption scheme anew. To this end, one can rely on our notions of appropriateness (Definition 4) and encryption-class taxonomy.

3.2.2.4. Security Assessment

As long as one uses well-known security schemes to ensure the equivalence notions, such as the ones in our taxonomy in Figure 2.1, the assessment of the security is known. In particular, one can adapt the reduction of Lemma 3.2.9 to other types of data. Otherwise, a full-fledged security analysis is needed, as, for instance, in [BCO11] for order-preserving encryption.

3.3. Summary

In this chapter, we study how to engineer the highest possible secure encryption schemes for structured data, such that applications that use distance-based data mining algorithms deliver the same results on encrypted as on true data. To this end, we propose the notion of distance-preserving encryption (DPE). With formal arguments, as well as experiments, we justify that preserving the exact distances upon encryption is the only feasible option. To engineer (DPE) schemes, we present the DisPE-procedure. It says how to design a DPE scheme for arbitrary data and distance measures. The procedure involves defining and ensuring equivalence notions, which capture a characteristic of data that should be preserved upon encryption. We then instantiate this procedure for SQL query logs. In this study, we find appropriate DPE schemes for all four prominent distance measures from literature. For all distance measures, we use well-known property-preserving encryption classes to implement the DPE schemes and assess their security. This is different from approaches supporting ad-hoc queries like CryptDB [Pop+11] and gives a way to higher security levels. Finally, we demonstrate how to reuse of parts of our study by the examples of XQuery and of relational data.

Part III.

Differentially Private Monitoring of Infinite Streams

4. Fundamentals

In the first part of the thesis, we motivated the privacy-utility trade-off and the scope of this thesis. One challenge we discussed is how to optimize the utility differentially private PETs in case the service users are not trusted. In this part, we focus on solutions of this challenge for PETs processing infinite streams.

This chapter introduces the fundamentals of this part. First, we introduce notation that is in line with literature and used consistently through this part. Second, we introduce differential privacy. Table 4.1 summarizes important notation that we introduce in the remainder, and the links to the scope of this thesis.

Table 4.1.: Important notation used in Part III.

Notation	Meaning in Part III	Link to Thesis Scope
Static Databases		
D	static database with n rows & cols $\mathcal{A}$	
$\mathcal{D}$	set of all static databases with cols $\mathcal{A}$	
D_t	static database at time t	true data at time t
$D_t[i]$	data of row i at time t (*event*)	
Streams		
$S = (D_1, .., D_t, ..)$	data stream	true data
$S[i] = (D_1[i], ..)$	stream reduced to row i	
$S_p = (D_1, .., D_p)$	stream prefix of length p	
$S_{J=[t,t']} = (D_t, .., D_{t'})$	sub-stream, a static snapshot of stream S	
$\pi = [\pi^1, .., \pi^T]$	sequence of T events (*pattern*)	
Queries		
$Q : D \to \mathbb{R}^{\dim}$	*dim*-dimensional query	
$Q(D)$	true query result	
$Q(S) = (Q(D_1), \cdots)$	true query result stream	
ΔQ	global sensitivity of query Q	
Privacy Mechanisms		
$\mathcal{M} : \mathcal{D}^p \to \mathcal{R}^p$	a mechanism sanitizing stream prefixes	PET
$\mathcal{M}_t : \mathcal{D} \to \mathcal{R}$	a sub-mechanism of $\mathcal{M}$ at time t	PET at time t
$r_t = \mathcal{M}_t(D_t)$	released data at time stamp t	sanitized data at time t
$R = (r_1, .., r_p) \in \mathcal{R}^p$	released stream prefix of length p	sanitized data

	Individual	$t = 1$		$t = 2$		$t = 3$		$\cdots$
		a_1	a_2	a_1	a_2	a_1	a_2	$\cdots$
Stream S	S[1]	**1**	**0**	**1**	**1**	1	1	$\cdots$
	S[2]	**0**	**0**	**0**	**1**	**0**	**1**	$\cdots$
Result $Q(D_t)$		$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$

🟦 Database 🟩 Event 🟥 Pattern

Figure 4.1.: Continuous monitoring of query results.

4.1. Notation

In this section, we introduce notation on static databases, streams and queries. Figure 4.1 illustrates these three concepts and how they relate to each other. It shows that a stream is an infinite sequence of databases. At each time stamp, the same query is executed over the database. The application uses these query results continuously.

Static Databases Let $I = \{1, .., n\}$ be a set of individuals and $\mathcal{A} = \{a_1, .., a_k\}$ a set of activities, like physical activities or appliance usages. The respective static database D is a table with column set (*cols*) $\mathcal{A}$ and row set I. In accordance with related work on differential privacy [Dwo+10; Kel+14], we use the term *individual*, which is, in most cases, equivalent to the term data owner. A database D contains the data of multiple owners. Nevertheless, there are also cases in which the whole database D, i.e., all "individuals", belong to only one data owner (see Chapter 6). A *static* database is not updated over time [DR+14]. Databases may fulfill a set of deterministic *constraints* C of arbitrary nature, ensuring the validity of the database with respect to the use case. For instance, one may constrain the attribute values of one row (intra-individual constraints), or of different rows (inter-individual) [Zhu+14; LCM16; Cao+18]. An example for the former is that the two activities *running* and *sitting* cannot both be performed at the same time stamp. As usual, we assume that constraints are public knowledge, meaning that an adversary knows them [HMD14; Zhu+14; LCM16; Cao+18].

Streams A stream $S = (D_1, D_2, .., D_t, ..)$ is an infinite sequence of static databases. For $1 \leq i \leq n$, row i corresponds to the stream $S[i] = (D_1[i], .., D_t[i], ..)$. As Figure 4.1 illustrates, each $D_t[i]$ is a use-case specific event consisting of states of activities $a_1, .., a_k$ individual i performed, that fulfills the constraints C. Additionally, a pattern $\pi = [\pi^1, .., \pi^T]$ a sequence of T events. Typically, in streams, the values of databases at different time stamps are correlated. For instance, in case an individual performs the activity *running* at $t = 1$, it is very likely that he or she performs it at $t = 2$ as well. In the context of streams, we use the notions *active window* and *stream prefix* first introduced in [Kel+14], and the term *sub-stream*, that are defined as follows. The active window of size w of time stamp t spans from time stamps $t - w$ to t. A stream prefix $S_p = (D_1, .., D_p)$ is a finite prefix of stream S having length p. A sub-stream of a stream S restricted to interval $J = [t, t']$ is a static snapshots $S_J = (D_t, .., D_{t'})$ of S.

Queries A query $Q : \mathcal{D} \to \mathbb{R}^{\text{dim}}$ maps a static database D to *dim* values being monitored. Examples are histograms queries reflecting the number of individuals per location (*dim* = number of bins), or the sums of the power consumption values of all individuals (*dim* = 1). $Q(S)$ stands for the application of Q to every static database of stream S, i.e.,

$$Q(S) = (Q(D_1), Q(D_2), \cdots).$$

4.2. Differential Privacy

Differential privacy (DP) [Dwo11] and its extensions [Kel+14; HMD14; KM14] is the current state-of-the-art privacy definition for databases and streams [MHH17]. Originally developed for static databases containing privacy-critical data about individuals, the idea is to protect every individual, while allowing to analyze the results of queries executed of a database as a whole. Therefore, it is also called *user-level privacy.* However, intuitively, the more queries the data administrator executes over the same or correlated database, the higher is the amount of noise the PET needs to add to the result of each query [DR+14]. This results in an infinite amount of noise for infinite query result streams. Consequently, for streams, PETs rely on rolling window approaches, relaxing the original definition of differential privacy. The respective definition is called w-event differential privacy.

Subsequently, we first introduce the original definition for static databases, and then w-event differential privacy for streams.

4.2.1. Differential Privacy for Static Databases

In the remainder, we first provide the formal definition of differential privacy, which specifies a condition on a so-called *mechanism* that computes the query results. Second, we state important mechanisms satisfying this definition. Third, we state properties of differential privacy that we exploit in this part.

4.2.1.1. Definition

Let $\mathcal{D}$ be a set of static databases featuring the same column set $\mathcal{A}$. As stated in Definition 4.2.1, two databases $D, D' \in \mathcal{D}$ are neighbors if one obtains one from the other by removing or adding one individual, i.e., one row.

Definition 4.2.1 (Neighboring Databases [DR+14])**.** *Let $D, D' \in \mathcal{D}$ be two databases. The databases are neighboring, short $D, D' \in \mathcal{N}$, if D differs from D' by adding, removing or changing one row.*

Differential privacy (DP) requests that the output of a mechanism should be indistinguishable, up to a factor of e^{ϵ}, for any possible existing pair of neighboring databases, as stated in Definition 4.2.2. The data administrator sets the desired *privacy level* ϵ, that is also known as *privacy budget*, given by the data owners. It usually lies between 0.1 and 1.0. A smaller value means better privacy.

Definition 4.2.2 (Differential Privacy [Dwo11])**.** *A randomized mechanism $\mathcal{M} : \mathcal{D} \rightarrow \mathcal{R}$ gives ϵ-differential privacy if for all neighboring databases D and D' and all $R \in \mathcal{R}$ holds,*

$$Pr[\mathcal{M}(D) = R] \leq e^{\epsilon} \cdot Pr[\mathcal{M}(D') = R].$$

There exist multiple extensions of this definition. Particularity important extensions are *local differential privacy* [Cor+18] and *approximate differential privacy* [DR+14] as follows. In local differential privacy, a database belongs to a single data owner, and each data owner sanitizes its own data. This is useful in scenarios without a trusted, central data administrator. If not stated otherwise, we rely on the classical, also called *global*, definition. Approximate, or (ϵ, δ)- differential privacy [DR+14], allows for an additional additive factor δ at the right-hand side of the equation in Definition 4.2.2. Since using too high (order of $\frac{1}{||D||_1}$) values for δ is risky (a mechanism releasing a few rows of D would be "private"), and even negligible values of δ the provided guarantee is weaker [DR+14], we focus on ϵ-differential privacy.

4.2.1.2. Mechanisms

Having defined differential privacy, the question is how to engineer a mechanism that satisfies this definition. In literature, there are a few mechanisms that are usually used as a building block to implement more complex mechanisms. Particularly relevant are the Laplace mechanism for numeric [DR+14], and the exponential mechanism [MT07] for categorical query results. While the primer is based on adding noise to the query results, the latter is based on sampling. In this part, we use the Laplace mechanism as a building block, as the query results considered are numeric. Consequently, we now introduce the Laplace mechanism.

The Laplace mechanism adds data-independent noise to the query results that follows a zero-mean Laplace distribution. Particular interesting is how the scale of the distribution is computed. By the definition of the distribution, the higher the scale, the higher is the expected noise. The scale depends on two factors. First, intuitively, the scale must be higher if ϵ becomes smaller. Second, the scale depends on how much the query results can vary for neighboring databases. The latter is formalized by the global sensitivity stated in Definition 4.2.3. For instance, for count queries, i.e., queries that count the number of row matching a condition, the global sensitivity is 1. The rationale is that the query result changes by at most 1, in case one row changes.

Definition 4.2.3 (Global Sensitivity [DR+14])**.** *The global sensitivity of a query Q is defined as*

$$\Delta Q := \max_{D,D' \in \mathcal{N}} ||Q(D) - Q(D')||_1.$$

With that notion, the Laplace mechanism is stated in Theorem 2.

Theorem 2 (Laplace Mechanism [DR+14]). *For a query Q, the Laplace mechanism defined as*

$$\mathcal{M}_L(Q, \mathcal{D}, \epsilon) := Q(\mathcal{D}) + Lap(\lambda)$$

with $\lambda = \frac{\Delta Q}{\epsilon}$ *provides* ϵ*-differential privacy.*

4.2.1.3. Properties

Differential privacy features three important properties that we leverage in this part. These properties are post-processing immunity, sequential and parallel composition, as well as group differential privacy.

Post-Processing Immunity Post-processing immunity relates to applications that use sanitized data. As stated in Theorem 3, an application f that uses, and potentially modifies, query results that are computed with a differentially private mechanism still satisfies differential privacy. This property differentiates differential privacy from other definitions like k-anonymity [Swe02].

Theorem 3 (Post-Processing Immunity [DR+14]). *Let* $\mathcal{M} : \mathcal{D} \longrightarrow \mathcal{R}$ *be a* ϵ*-differentially private mechanism, and* $f : \mathcal{R} \longrightarrow \mathcal{R}'$ *an arbitrary randomized mapping. Then,* $f \circ \mathcal{M}$ *is* ϵ*-differentially private.*

Sequential and Parallel Composition Composition relates to the execution of more than one mechanism on the *same* database. Since every mechanism releases information about the database, the question is whether and how the privacy level decreases. To answer this question, we differentiate between (1) running multiple mechanisms on all rows of the database, and (2) on disjoint horizontal partitions of the database. (1) is called sequential composition. As stated in Theorem 4, the ϵ sum up. (2) is called parallel composition. As stated in Theorem 5, the overall privacy level equals the worst one.

Theorem 4 (Sequential Composition [DR+14]). *For* $i \in 1, .., k$*, let* $\mathcal{M}_i$ *be an* (ϵ_i)*-differentially private mechanism. Then,* $(M_1(D), .., M_k(D))$ *is* $(\sum_{i=1}^{k} \epsilon_k)$*-differentially private.*

Theorem 5 (Parallel Composition [McS09]). *Let* $D_1, .., D_k$ *be a horizontal partitioning of* D *into partitions, such that each row of* D *is in maximally one partition. Further, for* $i \in 1, .., k$*, let* $\mathcal{M}_i$ *be a* (ϵ_i)*-differentially private mechanism. Then,* $(M_1(D_1), .., M_k(D_k))$ *is* $(\max_{i=1}^{k} \epsilon_k)$*-differentially private.*

Group Differential Privacy Group differential privacy is a generalization of Definition 4.2.2. Here, neighboring databases differ not only in one, but in $k > 1$ rows. For instance, due to correlation between rows. Theorem 6 states that in this case, the guarantee decreases to $k \cdot \epsilon$. Consequently, in order to protect, e.g., $k = 2$ rows, the data administrator has to double the noise to achieve differential privacy.

Theorem 6 (Group Differential Privacy [DR+14]). *Let $\mathcal{M}$ be a ϵ-differentially private mechanism. Then, it is $(k \cdot \epsilon)$-differentially private for groups of size k. That is, let D, D' be two databases that differ in maximum k rows. Then, it holds*

$$Pr[\mathcal{M}(D) = R] \leq e^{k \cdot \epsilon} \cdot Pr[\mathcal{M}(D') = R].$$

Related work generalizes this concept further to arbitrary neighborhood definitions [Cha+13; HMD14; KM14].

4.2.2. w-Event Differential Privacy for Streams

For streams, the w-event differential privacy (DP) is the currently most advanced privacy definition for streams [Kel+14]. Subsequently, we outline w-event differential privacy, by stating its definition and how to design mechanisms.

4.2.2.1. Definition

Similar to static differential privacy, w-event DP is based on a neighborhood definition. Since w-event DP focuses on streams, it leverages the notion of neighboring databases to define neighboring stream prefixes. Specifically, according to Definition 18, two stream prefixes are w-neighbors, if (1) the databases collected at each time stamp are pairwise the same or neighbors, and (2) all neighboring databases fit in a window of size w. In contrast to Definition 4.2.1, for constructing neighboring databases, only adding and removing of rows is allowed, but not the change of a row [Kel+14]. For $w \rightarrow \infty$, this definition of neighboring stream prefixes degenerates to the definition of neighboring databases from static DP. For $w = 1$, it degenerates to event-level DP [Dwo+10].

Definition 18 (w-Neighboring Stream Prefixes [Kel+14]). *Let w be a positive integer, and $t, t_1, t_2 \leq p$ three time stamps. Two stream prefixes S_p, S'_p are* w-neighboring, *if*

1. *for each D_t, D'_t with $D_t \neq D'_t$, it holds that D_t, D'_t are neighboring*
2. *for each $D_{t_1}, D_{t_2}, D'_{t_1}, D'_{t_2}$ with $t_1 < t_2$, $D_{t_1} \neq D'_{t_1}$ and $D_{t_2} \neq D'_{t_2}$, it holds that $t_2 - t_1 + 1 \leq w$.*

With that, w-event differential privacy is given if the query results of all w-neighboring stream prefixes are hard to distinguish, see Definition 19.

Definition 19 (w-Event ϵ-Differential Privacy [Kel+14]). *Let $\mathcal{M} : \mathcal{D}^p \rightarrow \mathcal{R}^p$ be a randomized mechanism that takes as input a stream prefix of arbitrary size. We say that $\mathcal{M}$ satisfies* w-event ϵ-differential privacy *if for all $R \in Range(\mathcal{M})$, all w-neighboring stream prefixes S_p, S'_p, and all p, it holds that*

$$Pr[\mathcal{M}(S_p) = R] \leq e^{\epsilon} \cdot Pr[\mathcal{M}(S'_p) = R].$$

4.2.2.2. Mechanisms

To design mechanisms, Theorem 7 is frequently used [Wan+16b; Kel+14]. It states that a PET can implement w-event DP by using independent DP mechanisms for each static databases D_t at time stamp t. For instance, the PET can use the Laplace mechanisms for releasing each $Q(D_t)$, as long as it ensures that the budget spend by w consecutive mechanisms does not exceed ϵ. That is, we can distribute the total budget spend per time stamp in each rolling window. To illustrate, one important w-event DP mechanism is the Uniform mechanism [Kel+14] that adds Laplace noise with scale $\lambda(\Delta Q, w, \epsilon) = \frac{\Delta Q \cdot w}{\epsilon}$ to the query results at each time stamp.

Theorem 7 (Composition [Kel+14]). *Let $\mathcal{M}$ be a mechanism processing a stream prefix $S_p = (D_1, .., D_p)$, and outputting a transcript $R = (r_1, .., r_p)$. If we can decompose $\mathcal{M}$ into p sub-mechanisms $\mathcal{M}_1, .., \mathcal{M}_p$, s.t. $\mathcal{M}_t(D_t) = r_t$, each $\mathcal{M}_t$ has independent randomness and achieves ϵ_t- differential privacy, then, $\mathcal{M}$ satisfies w-event differential privacy if*

$$\forall t \in [w, p] : \sum_{k=t-w+1}^{t} \epsilon_k \leq \epsilon.$$

4.2.3. Utility Metrics

In this part, we aim at engineering mechanisms that optimize utility. To measure the achieved utility, metrics are needed.

Generally, there exist two types of utility metrics: First, the sanitization error that is given by the distance between the released and the query result stream [FX14; Kel+14; Li+15a; Wan+16a; Wan+19; CY15]. Second, utility metrics based on the accuracy of application results [KBB15; Eib+18]. While the former might not provide enough informative value to assess the utility in a specific application, the latter might be not be generic enough. In this part, we use both types of utility metrics.

To quantify the sanitization error, the mean absolute error (MAE), and mean relative error (MRE) are commonly used [FX14; Kel+14; Li+15a; Wan+16a; Wan+19; CY15]. Additionally, the root mean squared error (RMSE) is used rarely [EL18]. The mean absolute error is defined by

$$\text{MAE}(Q(S_p), R) = \frac{1}{p} \sum_{t=1}^{p} |Q(D_t) - r_t|,$$

and the root mean squared error by

$$\text{RMSE}(Q(S_p), R) = \sqrt{\frac{1}{p} \sum_{t=1}^{p} (Q(D_t) - r_t)^2}.$$

Similar, for $\gamma \geq 0$, the mean relative error is defined by

$$\text{MRE}(Q(S_p), R) = \frac{1}{p} \sum_{t=1}^{p} \frac{|Q(D_t) - r_t|}{\max\{Q(D_t), \gamma\}}.$$

Here, γ is a sanity bound to mitigate the effect of small query results. In this part, for streams with $Q(D_t) > 0$ for all $t \in [1, p]$, if not stated otherwise, we use $\gamma = 0$. Additionally, we omit the input parameters of the metrics, if they are clear from the context.

5. Benchmarking Differential Privacy Mechanisms for Streams

In the previous chapter, we introduced the fundamentals of w-event differential privacy (DP).[1] Literature proposes various mechanisms that data administrators can use to sanitize streams to achieve w-event differential privacy [Che+17; Li+15a; Che+16; Nie+16; Wan+16c; Wan+16b]. All of them have in common that they sanitize the stream by adding noise to the query results. Consequently, the design goal of such mechanisms is to minimize the error the mechanism introduces. That is, one aims at providing high data utility. There are mechanisms specifically designed at exploiting certain stream properties, such as sparse streams [Wan+16b]. However, for most mechanisms there is little knowledge for what stream properties they provide high utility. Consequently, a logical follow-up question for data administrators is: What mechanism is expected to deliver suitable utility? A similar question arises for researchers aiming at proposing novel mechanisms to address shortcomings of existing ones. Since a theoretical examination mostly targets at the worst case, to determine the expected utility, they need to rely on empirical studies. For conducting empirical studies on static data, like relational databases [CSJ15], established benchmarks exist [Hay+16]. However, for streams, these guidelines hardly help. The reason is that w-event mechanisms work significantly different using, e.g., rolling window techniques to keep track of the available budget for each window of size w. This means that there is no generally accepted way of benchmarking w-event mechanisms. Instead, we find that existing studies deviate significantly regarding *all* relevant elements of a study, like data streams and competitor mechanisms, making the results hard to compare as we sketch below. This, in final consequence, limits practical application as well as it delays scientific progress.

Limitations of Previous Studies Conducting research on w-event differential privacy, we observe limitations in prior studies referring to all relevant elements of a study: Data streams, mechanisms used, and interpretation of the computed errors indicating mechanism utility. Specifically, most studies focus on a small set of real-world streams [Che+17; Li+15a; Che+16; Nie+16; Wan+16c; Wan+16b]. Studies that use artificial data to study the influence of relevant data properties are only known for static [Hay+16] and finite time series [FXS13]. Concerning real-world streams, we observe that not all streams are publicly

[1] The remainder of this chapter bases on the article Christine Schäler, Thomas Hütter, and Martin Schäler. *Benchmarking the Utility of w-event Differential Privacy Mechanisms – When Baselines Become Mighty Competitors.* Tech. rep. Karlsruhe Institute of Technology, KIT Scientific Working Papers 194, 2022. Compared to the article, the sections have been shortened to be less repetitive, contain minor corrections, as well as formatting and notation changes to be in line with the format and structure of this thesis. After thesis submission, a revised version of the article was accepted at the VLDB conference [SHS23].

available. Furthermore, in case they are available, mostly a preprocessing is necessary in order to use them in the study. However, the preprocessing differs highly among the publications using the same streams [Paw+19; Che+19]. Furthermore, analyzing the available streams indicates that, especially multi-dimensional streams, are often sparse. That is, they mainly contain zero values. Thus, publishing the same value all the time, which one of the baseline mechanisms in essence does [Kel+14], yields good utility at first sight according to the error metrics most commonly used. Next, quantifying the benefit the data administrator can achieve if using, e.g., the latest w-event mechanism compared to a baseline mechanism is virtually impossible, since many studies do not compare to both baselines. Therefore, it is hard to decide whether an easy-implementable baseline mechanism suffices for the current use case, or whether a sophisticated mechanism is needed. Moreover, state-of-the-art mechanisms are highly complex and subtle differences in the implementation or initialization parameters can have a significant effect on mechanism utility. Unfortunately, for most of them, no implementation is publicly available.

Contributions and Outline Motivated by the above illustrated limitations of previous experimental studies, we present the following three contributions: First, our literature analysis reveals the insight that all existing w-event mechanisms follow the same abstract framework simplifying comparison at qualitative level. We introduce this framework in Section 5.1.2. Second, in Section 5.2, we identify requirements of a benchmark. The result of a comprehensive literature survey suggests that all existing empirical studies of w-event mechanisms can be described by a tuple of four elements: mechanisms, streams, privacy requirements, and utility metrics. For each element, we outline limitations of prior studies. Based on that, we propose and justify requirements on the elements, ensuring the comparability of results. Third, Section 5.3, we show how to instantiate the four-tuple forming a comprehensive benchmark satisfying the requirements defined before. Fourth, in Section 5.4, we conduct, the so far largest empirical study comprising 252,000 single experiments that reveals new insights into the strengths and weaknesses of existing mechanisms. Analyzing the results yields three main insights as follows. (1) We systematically analyzed which stream properties have a high influence on mechanism utility. We observe that the amplitude, but not the period length, is decisive. (2) We observe an unexpected baseline supremacy. That is, for every combination of stream and privacy requirements, one out of two baseline mechanisms is among the mechanisms with the highest utility. (3) We observe that so-called data-adaptive sampling does not yield a utility improvement if the amplitudes of the stream is large. Third, considering the benchmark results, we discuss valuable findings for practitioners, i.e., data owners and administrators, as well as for the research community. They not only help data administrators to select mechanisms for their data, but also reveal research directions for future work. Finally, we provide a summary of this chapter.

5.1. Specific Fundamentals

In this section, we first outline the methodology of our literature survey. Second, we present the mechanism framework that all identified w-event mechanisms follow.

Table 5.1.: Overview of identified publications.

BA, BD, FAST$_w$ [Kel+14]	PeGaSuS [Che+17]
Retroactive Grouping [CSJ15]	Local DP [EL18]
DSAT$_w$ [Li+15a]	STBD [Liu+18]
SecWeb [Wan+16c]	DPS [Eza+19]
G-event [Che+16]	AdaPub [Wan+19]
RGP [Nie+16]	DADP [Wan+18]
RescueDP [Wan+16b; Wan+16a]	ToPS [Wan+21]
Re-DPoctor [Zha+17]	LPD-IDS [Ren+22]

5.1.1. Methodology of Literature Survey

We identify limitations of previous studies by means of a comprehensive survey analyzing the available w-event differential privacy literature. Our methodology to identify all relevant publications is as follows. We search through all publications originating from, i.e., citing, the original work of w-event differential privacy [Kel+14]. Additionally, we scanned the proceedings of the VLDB and ACM SIGMOD conference from the years 2020 to 2022, as well as and ACM CCS conference from the years 2020 to 2021, separately to not miss the most recent research results in the field. We included all publications that (i) perform an experimental evaluation on *streams* (i.e., not only finite time series) and (ii) are published at notable peer-reviewed conferences or journals. This also includes publications focusing on event-level differential privacy or own privacy definitions, that generalize event-level differential privacy or w-event differential privacy. In total, we included 17 publications listed in Table 5.1, whereby [Wan+16a] is a journal extension of [Wan+16b]. Consequently, in the remainder, we condense these two publications into one publication, and therefore speak about 16 publications.

5.1.2. The w-event Mechanism Framework

For all identified w-event differential privacy mechanisms of our literature study, the individual sub-mechanisms $\mathcal{M}_t$ follow one abstract framework shown in Algorithm 2. To ease up comparison of mechanisms and experimental results, we introduce this framework.

A sub-mechanism has four inputs[2]: The privacy requirements ϵ and w, the database D_t that shall be queried, as well as the last time stamp l where a sub-mechanism released a *sanitized* query result. The latter suggests that not at every time stamp the current query result is sanitized. Instead, previously released ones can be released multiple times. The output of the sub-mechanism is the released query result r_t. Intrinsically, the sub-mechanism implements four functions that are described below. For illustration, Table 5.2 provides example implementations of these functions.

[2] Note, individual mechanisms may use additional input parameters into $\mathcal{M}_t$ or the functions, but the idea remains the same.

Algorithm 2 w-event Mechanism Framework

```
function $\mathcal{M}_t(\epsilon, w, D_t, l)$
    if isSamplingPoint($\epsilon, w, D_t, l$) then
        $\epsilon_t \leftarrow$ budgetAllocation($\epsilon, w, D_t, l$)
        $p_t \leftarrow$ pertubation($\epsilon_t, \Delta Q, D_t$)
        $r_t \leftarrow$ filtering($p_t$)                    ▷ sanitized query result
        $l \leftarrow t$
    else $r_t \leftarrow r_l$                                ▷ approximation
    end if
    return $r_t$
end function
```

isSamplingPoint-Function. At each time stamp, the mechanism decides whether to *sample* the time stamp. Sampling means to sanitize the current query result, spending a portion of the privacy budget ϵ to perturb the query result. Then, $\mathcal{M}_t$ releases this sanitized query result. The alternative to sampling is called *approximation.* Here, the mechanism approximates the current query result with the one(s) sanitized last at time stamp l. The rationale for approximation is to save budget in case the query results change only marginally over time.

budgetAllocation-Function. In case the mechanism decides to sample, this function determines and allocates the share of privacy budget used for perturbation. Here, the strategies used among the mechanisms differ highly.

Perturbation-Function. In this function, firstly, the mechanism calculates the true query result. Secondly, it perturbs the query result using the allocated budget. To this end, all identified mechanisms leverage the Laplace mechanism.

Filtering-Function. The post-processing immunity of differential privacy stated in Theorem 3 allows to modify the perturbed query results p_t in an arbitrary way without spending budget or loosing the privacy guarantee, as long as no private information computed on D_t is used. Consequently, sub-mechanisms take advantage of this property within the filtering function to increase utility. A straight-forward filtering function is to truncate the perturbed query result, such that it fits in the domain Range(Q) of the query Q. For instance, for count queries, Range(Q) contains all non-negative integers. During truncating, the mechanism takes the perturbed query result p_t and releases $max(0, \text{round}(p_t))$, where round is a function that rounds a floating point number to the next integer.

5.2. Literature Survey and Benchmarking Requirements

In this section, we state and justify requirements on the elements each empirical study on w-event mechanisms consists of. As a result of our survey, we describe a w-event differential privacy benchmark with a 4-tuple $(\mathbb{M}, \mathbb{S}, \mathbb{P}, \mathbb{E})$:

Table 5.2.: Computation of the return values for all functions of the w-event mechanism framework of the baseline mechanisms Uniform and Sample [Kel+14].

Function	Uniform	Sample
isSamplingPoint	true	**if** w%t=0 **then** true **else** false
budgetAllocation	$\epsilon_t \leftarrow \frac{\epsilon}{w}$	$\epsilon_t \leftarrow \epsilon$
pertubation	$p_t \leftarrow Q(D_t) + \text{Lap}(\frac{\Delta Q}{\epsilon_t})$	
filtering	p_t	

- $\mathbb{M}$ is a set of mechanisms compared,
- $\mathbb{S}$ is a set of streams (i.e., data sets),
- $\mathbb{P}$ is a set of privacy requirements, i.e., (w,ϵ)-tuples,
- $\mathbb{E}$ is a set of (error) metrics to quantify mechanism utility.

In the remainder of this section, we describe the elements in more detail. We furthermore sketch the results of our literature study, illustrating how differently these elements are instantiated in related work. This makes comparing multiple empirical studies an almost futile endeavor.

5.2.1. Mechanism Set $\mathbb{M}$

Below, we state five requirements ($\mathbb{M}$-R1) to ($\mathbb{M}$-R5) that the mechanism set $\mathbb{M}$ needs to fulfill. Furthermore, we state how well these requirements are addressed in previous work proposing or comparing mechanisms. The latter is summarized in Table 5.3.

5.2.1.1. ($\mathbb{M}$-R1) Proofing the Desired Privacy Definition

Upon selecting a mechanism, the most fundamental requirement is that the mechanism provides the desired privacy guarantee. In our case, this is the w-event differential privacy definition. To this end, two options exist: (1) The publication leverages the definition directly. Then the authors need to proof that this definition is satisfied. (2) The publication proposes a new guarantee, e.g., a generalization of w-event differential privacy. Then, the authors need to prove that the proposed mechanism satisfies this new guarantee, and state how their mechanism can be parameterized such that it fulfills w-event differential privacy. Though this appears to be self-evident, our survey reveals (cf. Table 5.3) that there are mechanism propositions without a privacy proof.

5.2.1.2. ($\mathbb{M}$-R2) Inclusion of Baseline Mechanisms Uniform and Sample

In the original publication introducing w-event differential privacy [Kel+14], the authors propose two baseline algorithms: Uniform and Sample. The proposition is based on the observation that any mechanism introduces two different types of errors into the stream, namely, the perturbation and approximation error. As we outline below, for each baseline, one of them is dominant. Specifically, the perturbation error occurs in the Perturbation function that perturbs $Q(D_t)$ by adding noise. The perturbation error thus is the difference

Table 5.3.: Analyzed publications and addressing of Requirements (𝕄-R1) to (𝕄-R5) (✓yes, ✗no, ✓partially). Note, (𝕄-R4) and (𝕄-R5) are not applicable.

Reference	Privacy Definition	(𝕄-R1) Proof	(𝕄-R2) Baselines	(𝕄-R3) Sources
BA, BD, FAST$_w$ [Kel+14]	w-event	✓	Sample, Uniform	✓[a]
Retroactive Grouping [CSJ15]	event-level	✓	Uniform	✗
DSAT$_w$ [Li+15a]	w-event	✓	none[b]	✗
SecWeb [Wan+16c]	w-event	✓	Uniform	✗
G-event [Che+16]	w-event	✗	Sample	✗
RGP [Nie+16]	w-event	✓	Uniform	✗
RescueDP [Wan+16b; Wan+16a]	w-event	✓	none	✗
Re-DPoctor [Zha+17]	w-day event	✗[c]	none	✗
PeGaSuS [Che+17]	event-level	✓	Uniform	✗
Local DP [EL18]	local w-event	✓	none	✗
STBD [Liu+18]	(w,n)-DP	✓	Uniform	✗
DPS [Eza+19]	local w-event	✗[d]	Uniform	✗
AdaPub [Wan+19]	w-event	✓	none	✗
DADP [Wan+18]	distr. w-event	✓	none	✗
ToPS [Wan+21]	event-level	✓	none	✗
LPD-IDS [Ren+22]	local w-event	✓	Sample, Uniform	✗

[a] FAST used for FAST$_w$: http://www.mathcs.emory.edu/ lxiong/aims/FAST/

[b] Only the user-level mechanism is compared to Uniform.

[c] Relationship to w-event DP missing.

[d] Privacy proof missing.

of the true query result $Q(D_t)$ to the perturbed one p_t. The approximation error occurs when a mechanism decides to not sample and approximates the current $Q(D_t)$ with the last released sanitized result r_l. It is quantified by the difference of the true query result $Q(D_t)$ to the last released one r_l. Especially if query result fluctuation is small, the approximation error is also small.

Regarding the baseline mechanisms, the mechanism Uniform samples every time stamp by allocating $\epsilon_t = \frac{\epsilon}{w}$ budget for perturbation. Therefore, it only introduces perturbation error. By contrast, the mechanism Sample samples a new query result every w^{th} time stamp only, and approximates the query results at the remaining time stamps. Thus, it has the total budget, i.e., $\epsilon_t = \epsilon$, for perturbation. Therefore, its error is dominated by the approximation error. As a result, we suggest including *both* baseline mechanisms, as they allow studying the dominant error type and help quantifying the improvement of a newly proposed mechanism. However, our literature study reveals that 5 out of of 16 publications do not include any of these baselines, while 7 compare to only one baseline.

5.2.1.3. ($\mathbb{M}$-R3) Availability of Mechanism Implementations

Most mechanisms proposed in literature are intrinsically complex. For instance, for deciding when to sample multiple mechanism rely on a *proportional–integral–derivative (PID) controller* [FXS13; Wan+16b; Li+15a; Bel15] or *Kalman filter* [FXS13; Wan+16b; WB95]. At the same time, aiming at validating experimental results, we find that, for a couple of mechanisms, subtle differences in the implementation or initialization parameters can have a significant effect on mechanism utility. An example is whether the mechanism rounds the query result in the Filtering function to the query domain or not. Therefore, we advocate to make all used implementations available online. However, our literature survey reveals that for only one out of 16 publications, access to an implementation of one mechanism is given. Moreover, for the mechanism used in the original w-event publication [Kel+14], we got access to re-implemented sources by contacting the authors, which highly helped to validate our own mechanism implementations.

5.2.1.4. ($\mathbb{M}$-R4) Private Parameter Determination

Generally, all parameters of any mechanism that are computed on the true stream need to be computed in a private way [Hay+16]. An example is the number of rows in the stream, which is private information. None of our surveyed publications addresses this requirement explicitly, even though not all mechanisms sanitize these parameters. However, verifying this requirement without having access to the concrete mechanism implementation ($\mathbb{M}$-R3) is practically impossible. In our benchmark later on, we identified that three out of 10 w-event mechanism do not fulfill this requirement. To solve this issue, [Hay+16] suggests to use *mechanism repair functions*. We follow this proposal.

5.2.1.5. ($\mathbb{M}$-R5) Homogeneity of Background Knowledge

Most mechanisms use components, like PID controllers [Bel15], that have parameters as well. Background knowledge of the domain is required to set them optimally. However, in the benchmark, for comparability, it is important to use them consistently in all mechanisms [Hay+16]. For instance, all mechanisms using a PID controller use it with the same parameters for the same streams. For the surveyed publications, this does not apply.

5.2.2. Data Stream Set $\mathbb{S}$

Ideally, an empirical comparison consists of two parts. Firstly, researchers conduct a sequence of micro benchmarks on artificial data to study how stream properties may affect mechanism utility. Secondly, they use a canon of real-world streams reflecting real-world use cases.

($\mathbb{S}$-R1) Artificial Streams Reflecting Stream Properties The survey reveals that artificial streams are only used in [Ren+22], that uses linear, sine and log streams. However, although related work [Kel+14; Wan+19; FX14; Ren+22] indicates that mechanism performance depends on fluctuations and sparsity of the stream, literature lacks of systematical

Table 5.4.: Addressing of Requirement ($\mathbb{S} - R2$): Public availability of real-world streams used in the publications (✓yes, ✗no/removed, ✓with limitation).

Stream	Publicly avaiable	References in which used	Limitation
APASCologne	✓	[Eza+19]	-
DNS	✗	[Wan+21]	-
Fare	✓	[Wan+21]	raw data only
Flu Death	✓	[Wan+19]	different season
Flu Outpatient	✓	[FX14]	different ages & years
Foursquare	✓	[Ren+22]	-
GeoLife	✗	[Liu+18]	-
Heart rates	✗	[Zha+17]	-
Kosarak	✓	[Wan+21]	raw data only
Montreal traffic	✗	[CSJ15]	-
Nice ride	✓	[Wan+18]	-
POS	✗	[Wan+21]	-
Retail	✓	[Wan+19]	-
Rome traffic 1	✗	[Kel+14]	-
Rome traffic 2	✗	[Nie+16]	-
San Joaq./Oldenburg	✓	[Wan+16a; Wan+16b; Li+15a]	data generator only
State Flu	✓	[Wan+19]	-
TDrive	✓	[Li+15a; Ren+22]	-
Taobao	✓	[Ren+22]	requires account
Taxi Porto	✓	[Wan+16a; Wan+16b; Li+15a; Liu+18; Wan+18]	raw data only
Traffic Seattle	✗	[FX14]	-
Unemployment	✓	[FX14]	-
US census	✓	[Li+15a]	raw data only
WiFi traces 1	✗	[Che+17]	-
WiFi traces 2	✗	[EL18]	-
WorldCup	✓	[Kel+14; Wan+16c; CSJ15; Che+16; Wan+16a; Wan+16b]	raw data only

investigations which stream properties are relevant for either mechanism utility or reflecting real-world data, and how to define them. In the next section, we propose relevant properties when instancing the benchmark.

($\mathbb{S}$-R2) Publicly Available Real-World Streams with Reproducible Prepossessing Our literature survey reveals that most approaches focus on real-world streams from a specific use case, such as location monitoring. Even though multiple publications use the streams (see Table 5.4), the respective study results are not necessarily comparable. The reason is that, in most studies, the true query result stream is prepossessed and the prepossessing varies highly. We illustrate this by two examples referring to the most frequently used stream: the WorldCup stream. The underlying raw data set contains the logs of 89,997 URLs of the FIFA 1998 Soccer World Cup website. The first example results from comparing [Kel+14] with [Wan+16b]. While [Kel+14] refers to all 89,997 web pages, i.e., dimensions, [Wan+16b] samples 2,000 of them. The reported mechanism utilities in both publications indicate that the same mechanisms may have a highly different utility depending on the conducted prepossessing. The second example results from a Bachelor Thesis project [Wei19]. When trying to reproduce the results of [Kel+14], only after having access to the prepossessed query result stream kindly provided by the original authors, we find that the authors conducted an additional preprocessing normalizing the counts. Note, in both examples, we could verify the hypothesis that the deviation of study results may originate from stream preprocessing. In many other cases, this remained a hypothesis. Since we are aware that due to license issues most publications must not publish their preprocessed streams, it is particularly important that all preprocessing steps are documented [Paw+19; Che+19].

5.2.3. Privacy Requirements Set $\mathbb{P}$

With w-event differential privacy, data owners express their privacy requirements by a tuple (ϵ, w). In this tuple, ϵ is the available privacy budget and w is the window length. Regarding the range of examined privacy budgets, window sizes, and combinations of them, there is no clear consensus in the literature (see Table 5.5). At least, in each publication the authors conduct two types of experiments:

($\mathbb{P}$ - R1) Vary-ϵ The authors examine the influence of ϵ for a fixed value of w. Mostly, ϵ between 0.1 and 10 is used.

($\mathbb{P}$ - R2) Vary-w The authors examine the effect of w for a fixed value of ϵ. Researchers usually fixes $\epsilon = 1.0$.

However, for both types of experiments, there is no consensus at all regarding the window size w. The w-values even differ for the same stream. The overall tendency is that the lower border of $w > 10$ holds and the upper border usually is in the low hundreds.

Table 5.5.: Overview of analyzed publications and addressing of requirements with respect to $\mathbb{P}$ and $\mathbb{E}$ (u unknown, d dimension, n.a. not applicable, - not performed).

Reference	($\mathbb{P}$ - R1) Vary-ϵ (ϵ, w)	($\mathbb{P}$ - R2) Vary-w (w, ϵ)	($\mathbb{E}$ - R) Utility metrics
[Kel+14]	-	([40,200], 1)	MAE, MRE $\gamma = u$
[CSJ15]	([0.02,0.1], 1)	n.a.	MAE, MRE $\gamma = u$
[Li+15a]	([0.5,1], 800)	([200,1000], u)	total sum of squared error
[Wan+16c]	([0.01,1], 120)	([40,240], 1)	MAE, MRE $\gamma = 1$
[Che+16]	([0.5,1.5], u)	([40,200], u)	MAE, MRE $\gamma_d = 0.1\% \cdot \sum_{t=1}^{p} Q(D_t)[d]$
[Nie+16]	({0.5,1.0}, 1)	({10,50}, u)	MRE $\gamma = u$
[Wan+16b; Wan+16a]	([0.1,1], 200)	([40,240], 1)	MAE, MRE $\gamma_d = 0.1\% \cdot \sum_{t=1}^{p} Q(D_t)[d]$
[Zha+17]	([0.5,1.5],14)	([7,35],1)	MAE, MRE $\gamma = 0.05\% \cdot \sum_{t=1}^{p} Q(D_t)$
[Che+17]	({0.01, 0.1},1)	n.a.	MAE, true positive rate
[EL18]	({1.1,1.9},4)	([10,100],1)	MAE, RMSE
[Liu+18]	([0.2,1.0],120)	([40,200],1)	MAE
[Eza+19]	-	([0.01,1], u)	unspecified “average error”
[Wan+19]	([0.1,0.9], 100)	([40,200], 1)	MRE $\gamma_d = 1\% \cdot \sum_{t=1}^{p} Q(D_t)[d]$
[Wan+18]	([0.1,1],40)	(1.0, [20,200])	MAE, MRE $\gamma = 0.1\% \cdot \sum_{t=1}^{p} Q(D_t)$
[Wan+21]	([0.01,0.5],1)	n.a.	mean squared error
[Ren+22]	({0.5, 2.5},20)	({10, 50},1)	MRE $\gamma_d = u$, event monitoring ratio

5.2.4. Error Metrics Set $\mathbb{E}$

As Table 5.5 shows, to determine the utility of a mechanism, researchers usually quantify the error between the true and sanitized stream according to an error metric. As shown in Table 5.5, most studies use the mean absolute error (MAE) or the mean relative error (MRE), as defined in Section 4.2.3. However, there are subtle differences in the error calculation. This applies in particular to the selection of the sanity bound of the MRE. For instance, [Wan+16b] uses a data-dependent sanity bound γ, whereas [Che+16] fixes $\gamma = 1.0$ for the WorldCup stream. In three publications, the sanity bound is not stated at all, even though the used streams contain query results of 0, requiring $\gamma > 0$. Moreover, since mechanisms rely on random values, two runs of the same mechanism using the same combination of stream and privacy requirements may result in a highly different error. Consequently, [Hay+16] suggests to run each combination multiple times, and compare the average and the 0.95-quantile of the error.

5.3. Benchmark Definition

The subject of this section is the definition of the benchmark, we conduct in the next section. It is based upon the 4-tuple representation introduced in Section 5.2. Table 5.6 shows a brief overview of the selection of the elements, resulting in the so far largest empirical study, comprising 252,000 single experiments, i.e., mechanism runs. In the remainder of this section, for each of the elements, we explain how to address the requirements

Table 5.6.: Instantiation of the 4-Tuple $(\mathbb{M}, \mathbb{S}, \mathbb{P}, \mathbb{E})$.

Elem.	Instantiation
$\mathbb{M}$	(1) Baselines: Sample [Kel+14], Uniform [Kel+14]; (2) Competitors: FAST$_w$ [Kel+14], BD [Kel+14], BA [Kel+14], DSAT$_w$ [Li+15a], RescueDP [Wan+16b; Wan+16a], AdaPub [Wan+19], PeGaSuS [Che+17]
$\mathbb{S}$	(1) 20 artificial seasonal streams with dim = 1 (2) 8 real-world streams from Table 5.4: WorldCup, Taxi Porto, Flu Outpatient, TDrive, State Flu, Flu Death, Retail, and Unemployment
$\mathbb{P}$	(1) Vary-ϵ: $\epsilon \in [0.1, 1.0]$, $w = 120$ (2) Vary-w: $w \in [40, 200]$, $\epsilon = 1.0$
$\mathbb{E}$	(1) Average MAE over 100 runs (2) Average MRE with $\gamma_d = 0.1\% \cdot \sum_{t=1}^{p} Q(D_t)[d]$ for dimension d over 100 runs (3) Comparison of average error with 0.95 quantile of error

associated to each element as well as how we ensure validity and comprehensiveness of study results.

5.3.1. Mechanism Set $\mathbb{M}$

Below, we explain how to select the mechanisms for our empirical study. We specifically outline that the selection ensures comprehensiveness and validity of the intended results. To this end, we state how we address the requirements identified in Section 5.2.

5.3.1.1. ($\mathbb{M}$-R1)-($\mathbb{M}$-R2) Considered Mechanisms

We include (1) the baseline mechanisms Sample and Uniform, as well as (2) *all* mechanisms found in our literature study that either (a) support (global) w-event differential privacy directly or can be parameterized such that they achieve w-event differential privacy. We exclude mechanisms providing local or distributed w-event differential privacy, since, by the nature of the privacy definition, their utility is lower than the one of pure w-event differential privacy mechanisms [Cor+18; Wan+18; Ren+22]. We furthermore include (b) all mechanisms used as competitors in one of the publications that propose a mechanism in (a). According to Table 5.3, criterion (a) applies to FAST$_w$ [Kel+14], DSAT$_w$ [Li+15a], SecWeb [Wan+16c], RGP [Nie+16], RescueDP [Wan+16b; Wan+16a] and AdaPub [Wan+19]. Since SecWeb is a prequel of RescueDP, we do not include SecWeb in our experimental study. We additionally exclude RGP because it is only applicable to specific query streams, namely, to hierarchic location count streams. The second criterion (b) suggests including PeGaSuS as well, because [Wan+19] that proposes AdaPub uses it as competitor. However, we do not include Uniform with backwards smoothing used in the PeGasuS publication [Che+17] explicitly, as preliminary experiments revealed that it does not yield a substantial utility improvement compared to Uniform. Since PeGaSuS provides event-level DP only, we adjust it such that it provides w-event differential privacy. Inspired by the Uniform mechanism, we do this by providing a budget of $\epsilon_t = \frac{\epsilon}{w}$ per time stamp t. Analogously to the proof in [Kel+14], PeGaSuS then fulfills w-event differential privacy.

5.3.1.2. ($\mathbb{M}$-R4) Private Parameter Determination

A pivotal requirement is that all mechanism determine data-dependent parameters in a private way. As discussed in Section 5.2, we use mechanism repair functions whenever we find parameters that are not determined in a private way. Specifically, we use the following repair functions.

DSAT$_w$ Repair Function. This mechanism uses the number of rows, i.e., total count at each time stamp, in the stream. Since streams that feature a different number of rows are neighboring, this is private information. We repair DSAT$_w$ as follows: We calculate the total counts at the first time stamp, and *perturb* it by spending 10% of the privacy budget allocated for $t = 1$. To keep the privacy guarantee, we reduce the perturbation budget allocated at time stamp $t = 1$ accordingly. If the sanitized total count equals zero, the repair function uses the value 5,000 also used in the original publication [Li+15a].

BD/BA – Column Partitioning Repair Function Mechanisms BA and BD may use an optimization requiring to group the dimensions based on their correlation. Since non-coincidental correlation among dimensions is private information, it needs to be determined in a private way. This also holds despite the observation in the original publication [Kel+14] that both mechanisms are very sensitive towards this parameter. The original results indicate the number of groups should be rather small. For instance, on the WorldCup stream, they achieve the best results with 150 groups for 89,997 dimensions [Kel+14]. Consequently, we repair BD by using 0.2% of the dimensions as number of groups. We do not group in BA, because initial tests suggest no significant improvement.

5.3.1.3. ($\mathbb{M}$-R5) Homogeneity of Background Knowledge

Except Uniform, Sample, BD and BA, all mechanisms use components, like a PID controller relying on configuration parameters. If a mechanism is the only one using a specific parameter, we set it as given in the publication. If mechanisms share parameters, we set these parameters consistently in all mechanisms. Specifically, there are two parameters used in more than one mechanism: (1) The *desired sampling rate* used in FAST$_w$ and DSAT. The FAST publication [FXS13], a subroutine of FAST$_w$ [Kel+14], uses a rather high (15%) sampling rate, while the publication proposing DSAT$_w$ uses a rather small one (1%). As preliminary experiments revealed that both mechanisms tend to provide higher utility for higher rates, we use 15% as desired sampling rate in both mechanisms. (2) The parameters of the PID controller, that is used in FAST$_w$, RescueDP and DSAT$_w$. While the publications proposing RescueDP [Wan+16b; Wan+16a] and FAST [FXS13] suggest the same parameter values, the values used in [Li+15a] proposing DSAT$_w$ differ from them. However, while in DSAT$_w$, the PID controller controls the change in the sampling rate, in RescueDP and FAST$_w$, it controls the change in the sanitized values. Consequently, the operational purpose of the use of the PID controller is different. As a result, for each mechanism, we use the parameters suggested in the respective publication.

5.3.1.4. ($\mathbb{M}$-R3) Mechanism Implementation

A correct implementation of the mechanisms is a key factor to ensure result validity. We ensure validity of our results by the following four key principles: (i) favor original implementation, (ii) re-use of well-known mechanism parts, (iii) consistency checks of independent implementations, and (iv) contact original authors if necessary. Next, we explain these principles in more detail: First, in case the publication proposing a mechanism offers an implementation, we use this implementation. However, as shown in Table 5.3, this only holds for one mechanism, namely, FAST_w. Second, multiple mechanisms use the same component (e.g., the sampler), which is itself available open source. For instance, FAST_w uses a Kalman filter and PID controller. In such cases, we use this component consistently in all mechanisms. Third, all mechanisms are implemented redundantly and independently by up to three different people, eventually all leading to consistent results. Nevertheless, our results do partially highly deviate from the results in the original publications. Consequently, we contacted the original authors of [Kel+14] proposing w-event differential privacy and thankfully received implementations from them. This not only helped to ensure that all baselines and advanced mechanisms proposed in [Kel+14] are correct, but also for principle (ii) since we had more mechanism parts for re-usage.

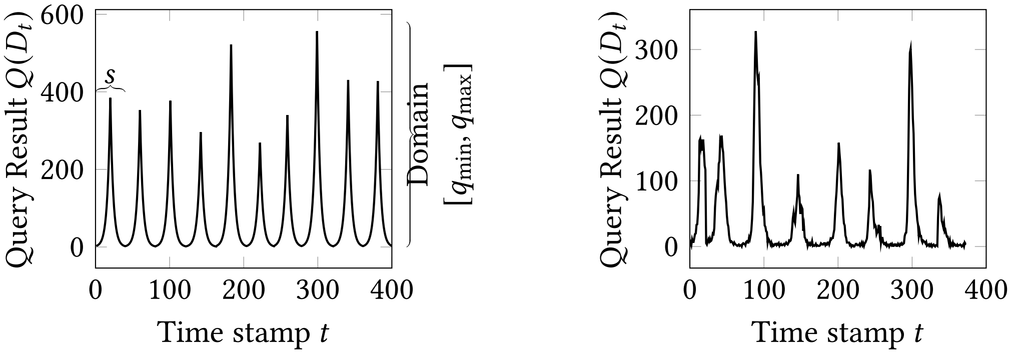
5.3.2. Data Streams Set $\mathbb{S}$

Concerning the data streams, we follow the requirements from Section 5.2 as follows: Firstly, to account for ($\mathbb{S}$-R1), we conduct a series of micro benchmarks with artificial data. Secondly, to account for ($\mathbb{S}$-R2), we conduct experiments on a comprehensive set of data streams used in literature.

5.3.2.1. ($\mathbb{S}$-R1) Artificial Streams Reflecting Stream Properties

The intention behind using artificial data is to study the influence of relevant stream properties on mechanism utility in a structured way. Generating meaningful artificial data is a challenge. For streams in general, there are various properties known to have an influence on data processing. This includes their dimensionality, as well as the components seasonality, level, and trend [HA18]. However, so far, it has not been investigated systemically which properties the real-world streams used in previous studies have, and how much these properties influence the mechanism utility. In the remainder, we first state for each property, (a) which of these properties do occur in the real-world streams from Table 5.4, and (b) how we design our artificial data generator such that we can investigate each of the properties in isolation. Then, we present the generator itself.

Dimensionality The streams in Table 5.4 so far have dimensionality between 1 and 80,000. In our micro benchmark, however, we consider univariate query result per time stamp (i.e., $\text{dim} = 1$). The rational is that we aim to understand how well a mechanism



(a) Artificial stream with length $p = 400$, domain $[0, 600]$ and expected season length $s = 40$.

(b) Flu Death stream.

Figure 5.1.: Comparison of an artificially generated data stream and a real-world stream.

retains the utility of the stream using clever budget allocation, sampling, and filtering, leveraging the inertia of the stream. By using dim = 1, we intentionally exclude in the micro benchmarks the additional utility improvement some mechanisms gain by taking advantage of correlated dimensions. The reason is that introducing known correlations in multi-dimensional streams is highly challenging.

Level Most seasonal streams feature inter-seasonal downtimes, where $Q(D_t)$ is close to zero (cf. Figure 5.1 (b)). The minimum query result is usually also the most frequent one. To decouple the level from the seasonality, we quantify the level by the minimum query result $q_{\min}$. The minimum query result of the stream influences mechanism utility in case the mechanism uses the filtering technique *truncating* (cf. Section 5.1.2). For queries like Count and Histogram, truncating rounds negative perturbed query results to zero. Consequently, whenever the Laplace mechanism adds a negative amount of noise of, e.g., -10, which holds in half of the cases, the mechanism releases the true query result instead. By contrast, if $Q(D_t) = 10$, the mechanism introduces a relative error of 100%.

Seasonality We observed that most real-world streams have a seasonality, with an exponential growth and shrinking phase. The maximum query result $q_{\max}$ the stream reaches varies highly from stream to stream. The perturbation and approximation error are however clearly influenced by the length of the seasons s, and the difference between minimum and maximum query result, named amplitude. Thus, we test the mechanism utility with respect to both. Since we use $q_{\min} = 0$, for the amplitude a, generally described by $a = q_{\max} - q_{\min}$, the following holds: $a = q_{\max}$. In the micro benchmarks, we generate streams for every combination of $s \in \{40; 60; 80; 10; 120\}$ and $a = q_{\max} \in \{10; 100; 10,00; 10,000\}$ reflecting values observed in real-world streams.

Trend We do not observe a trend in the sanitized streams from Table 5.4 used so far. Therefore, we discard this property.

Algorithm 3 Data Generator

```
function GENERATESTREAM($p, s, a$)
    $t \leftarrow 1, e \leftarrow 1.5$
    while $t < p$ do                                  ▷ Each loop generates one season
        $\mathrm{sl} \leftarrow \mathcal{G}(s, 2)$                              ▷ Dice season length
        $\mathrm{val} \leftarrow \mathcal{G}(8, 2)$                  ▷ Dice season minimum, close to 0
        $Q(D_t) \leftarrow \mathrm{val}; t++$
        for $i = 1$ to $\mathrm{sl}/2$ do
            $val \leftarrow e \cdot Q(D_{t-1})$                       ▷ Exponential growth
            $Q(D_t) \leftarrow \mathrm{val}; t++$
        end for
        ...                                           ▷ Symmetric shrinking phase
    end while
    $\max \leftarrow \max\{Q(D_1), .., Q(D_{p-1})\}$
    for $i = 1$ to $p$ do
        $Q(D_i) \leftarrow Q(D_i)/\max \cdot a$                ▷ Ensure desired amplitude
    end for
    return $Q(D_1), .., Q(D_p)$                           ▷ Ensure correct length
end function
```

Data Generator. Figure 5.1 shows example data, we generated using our data generation algorithm shown in Algorithm 3. Generally, the artificial data shall be similar to one-dimensional streams used in other studies. For the depicted data, we use $p = 400$ time stamps, amplitude $a = q_{\max} = 600$, and an average season length $s = 40$. Since in reality, not all periods have exactly the same length, we dice the length of each seasonal with $\mathcal{G}(s = 40, 2)$. For the growing phase, we use an exponential growth function with $Q(D_t) = e \cdot Q(D_{t-1})$ with $e = 1.5$. The, shrinking phase is symmetric to the growing phase. Next, we also mimic inter-seasonal downtime by dicing the season minimum with $\mathcal{G}(s = 8, 2)$, i.e, some value close to zero. Since, the maximum value of the stream generated this way depends on the actual length of the season and the diced minimal values, we need to normalize it to the desired amplitude a. Finally, because the algorithm generates the stream season-wise, it may be to long. Thus, we return the stream prefix until time stamp p.

5.3.2.2. ($\mathbb{S}$-R2) Publicly Available Real-World Streams with Reproducible Prepossessing

For comprehensiveness, we use all real-world streams that are freely available and used at least once to evaluate a w-event differential privacy mechanism. According to Table 5.4, these are: WorldCup, Taxi Porto, Flu Outpatient, TDrive, State Flu, Flu Death, Retail and Unemployment. All of them use a query Q with $\Delta Q = 1$. As far as possible, we post-processed them according to one of the respective publications. To encourage comparability and reproducibility and post-processing details at our project website[3].

[3] `https://dbresearch.uni-salzburg.at/projects/dpbench/index.html`

5.3.3. Privacy Requirements Set $\mathbb{P}$

Inspired by the experimental studies from related work, we conduct the vary-ϵ and vary-w experiments, fulfilling ($\mathbb{P}$-R1) and ($\mathbb{P}$-R2). For the vary-ϵ experiment, we select a reasonably large value for parameter w, which is $w = 120$. For the range of ϵ values to test, we use, in accordance with most studies, $\epsilon \in [0.1, 1]$ with an increment of 0.2. For the vary-w experiments, we use $\epsilon = 1$ as most studies do. Furthermore, we vary $w \in [40, 200]$ with a w increment of 40, like various previous studies.

5.3.4. Error Metrics Set $\mathbb{E}$

Since the mechanisms rely on randomness, for the same combination of privacy requirements and stream, the utility can differ highly in two mechanism runs. In accordance with various studies from related work, we run each experiment 100 times, and use the average MAE and average MRE (with $\gamma_d = 0.1\% \cdot \sum_{t=1}^{p} Q(D_t)[d]$ for dimension d) to quantify the error the mechanisms introduce into the sanitized data. Besides the average error, we additionally quantify the variance of the error as suggested by [Hay+16] for static (i.e., standard) differential privacy. To this end, we measure the 0.95 quantile of the MAE and MRE reflecting a “risk averse” data owner.

5.4. Results

In this section, we present and interpret the results of our study. First, we present the results of the artificial streams aiming at understanding the influence of stream properties on the mechanism utility. Second, we show the results of one-dimensional and multi-dimensional real-world streams to confirm our results on artificial streams and reveal insights on the preservation of stream properties in the released query results. Finally, we conclude the section with a discussion of our findings giving recommendations for practitioners and future research.

5.4.1. One-dimensional Artificial Streams

With the artificial streams, we aim at understanding how the two identified stream properties seasonal period length s and amplitude a affect the mechanism utility. Specifically, we are interested in the perspective of a data administrator aiming at selecting a mechanism for a given stream and privacy requirement. Consequently, we ask two research questions:

(RQ1) Are the stream properties decisive for mechanism selection?

(RQ2) If yes: Given a seasonal period length s, an amplitude a and privacy requirements (ϵ, w), can we recommend a mechanism and/or function design?

For brevity, we subsequently focus on the mean MAE denoted with MAE to answer these questions. This is valid, because the result patters for the 0.95 quantile of MAE and the MRE are similar. To make the mechanisms MAEs comparable over all streams and privacy requirements, we consider the MAE deterioration $\delta_{\text{MAE}}(c)$ for a specific combination $c = (m \in \mathbb{M}, (s, a), (\epsilon, w) \in \mathbb{P})$ of mechanisms, stream properties, and privacy requirements.

Uniform

a/ε	10	100	1,000	10,000
0.1	120.6	62.0	16.4	2.0
0.3	102.7	25.8	6.0	1.1
0.5	88.6	26.4	3.8	1.0
0.7	73.4	21.8	2.9	1.0
0.9	64.7	19.1	2.2	1.0
1	61.3	17.4	2.1	1.0

Sample

a/ε	10	100	1,000	10,000
0.1	1.0	1.0	2.1	2.6
0.3	1.0	1.1	2.3	4.1
0.5	1.0	1.8	2.4	6.4
0.7	1.0	2.0	2.6	9.0
0.9	1.0	2.3	2.6	11.5
1	1.0	2.3	2.6	12.8

AdaPub

a/ε	10	100	1,000	10,000
0.1	150.5	77.5	20.4	2.5
0.3	127.8	32.3	7.4	1.3
0.5	110.0	33.2	4.7	1.2
0.7	91.8	27.1	3.6	1.2
0.9	81.1	24.0	2.8	1.2
1	76.4	21.7	2.6	1.2

BA

a/ε	10	100	1,000	10,000
0.1	90.2	46.1	12.3	1.7
0.3	76.3	19.1	4.5	1.0
0.5	66.5	19.8	3.0	1.0
0.7	54.7	16.1	2.3	1.1
0.9	48.5	14.1	1.8	1.1
1	45.1	12.9	1.7	1.1

BD

a/ε	10	100	1,000	10,000
0.1	393.0	199.1	49.3	6.6
0.3	311.1	77.6	18.0	4.5
0.5	277.1	84.1	12.0	4.8
0.7	247.4	66.8	9.9	6.0
0.9	205.5	61.5	7.4	6.3
1	199.9	52.0	6.7	6.9

DSAT$_w$

a/ε	10	100	1,000	10,000
0.1	89.1	47.1	12.5	2.0
0.3	81.1	21.3	5.0	1.7
0.5	70.3	22.0	3.4	2.5
0.7	62.1	17.7	2.7	3.2
0.9	54.9	15.6	2.2	4.0
1	50.3	13.2	2.0	4.5

FAST$_w$

a/ε	10	100	1,000	10,000
0.1	8.2	4.2	1.5	1.0
0.3	6.8	1.9	1.0	1.5
0.5	5.6	2.0	1.0	2.4
0.7	4.8	1.8	1.0	3.3
0.9	4.2	1.6	1.0	4.3
1	3.9	1.5	1.0	4.7

PeGaSuS

a/ε	10	100	1,000	10,000
0.1	150.3	77.3	20.3	2.5
0.3	127.6	32.4	7.5	1.3
0.5	109.8	33.0	4.7	1.3
0.7	92.0	27.0	3.6	1.2
0.9	80.7	23.9	2.8	1.3
1	76.4	21.6	2.6	1.3

RescueDP

a/ε	10	100	1,000	10,000
0.1	6.7	3.2	1.0	25.7
0.3	5.9	1.0	1.6	625.7
0.5	5.2	1.0	3.9	1,768
0.7	4.1	1.0	9.6	1,585
0.9	3.0	1.0	27.0	3,201
1	3.2	1.0	39.0	1,977

(a) Vary-ϵ experiments with $w = 120$.

Uniform

a/w	10	100	1,000	10,000
40	20.1	4.8	1.1	1.0
80	66.6	17.5	2.0	1.0
120	60.7	17.4	2.1	1.0
160	133.1	35.3	4.0	1.0
200	96.7	26.3	4.2	1.0

Sample

a/w	10	100	1,000	10,000
40	1.0	1.9	4.0	37.5
80	1.0	1.0	1.0	4.9
120	1.0	2.3	2.6	12.8
160	1.0	1.0	1.0	2.5
200	1.0	2.0	3.2	7.5

AdaPub

a/w	10	100	1,000	10,000
40	25.1	6.1	1.3	1.2
80	83.2	21.9	2.5	1.2
120	76.0	21.8	2.6	1.2
160	168.0	44.2	5.0	1.3
200	120.5	32.9	5.3	1.2

BA

a/w	10	100	1,000	10,000
40	14.7	3.7	1.0	1.3
80	49.9	13.3	1.7	1.2
120	44.6	13.1	1.7	1.1
160	99.4	26.1	3.2	1.1
200	71.9	19.4	3.3	1.0

BD

a/w	10	100	1,000	10,000
40	22.7	5.4	1.4	2.3
80	142.2	37.8	4.7	6.6
120	180.7	54.6	7.3	6.7
160	552.2	137.7	17.3	9.4
200	411.7	126.4	20.6	8.7

DSAT$_w$

a/w	10	100	1,000	10,000
40	17.6	4.3	1.8	11.6
80	58.6	14.5	2.7	10.6
120	51.1	14.8	2.1	4.4
160	103.7	28.3	3.8	4.6
200	74.4	21.6	4.0	3.3

FAST$_w$

a/w	10	100	1,000	10,000
40	2.2	1.0	1.6	14.3
80	4.9	1.9	1.1	4.9
120	4.0	1.5	1.0	4.7
160	8.4	2.6	1.1	2.5
200	6.1	1.9	1.0	2.1

PeGaSuS

a/w	10	100	1,000	10,000
40	25.3	6.1	1.3	1.2
80	83.5	21.9	2.5	1.3
120	75.3	21.7	2.6	1.2
160	167.1	44.2	5.0	1.2
200	120.3	32.8	5.3	1.2

RescueDP

a/w	10	100	1,000	10,000
40	1.4	1.0	1.2	1.6
80	3.9	1.5	9.0	32.3
120	3.0	1.0	37.1	2,572
160	6.5	1.6	281.3	$2.8{\cdot}10^5$
200	3.2	1.0	117.7	$2.0{\cdot}10^7$

(b) Vary-w experiments with $\epsilon = 1.0$.

Figure 5.2.: δ_{MAE} values of the vary-ϵ and vary-w experiments for a period length of $s = 80$.

The MAE deterioration specifies how worse the MAE of a specific combination c is, compared to the best mechanisms for this combination of stream properties and privacy requirements:

$$\delta_{\text{MAE}}(m, (s, a), (\epsilon, w)) = \frac{\text{MAE}(m, s, a, \epsilon, w)}{\min\{\text{MAE}(m', s, a, \epsilon, w) \mid m' \in \mathbb{M}\}}.$$

In Figure 5.2, we use a color gradient marking small values, i.e., relatively high utility, in green and large values in red. Subsequently, we present and discuss the results with respect to the two research questions.

5.4.1.1. (RQ1) Are the Stream Properties decisive for Mechanism Selection?

For answering this question, we investigate the influence of the stream properties amplitude a and period length s on the MAE, and compare the influence to the influence of the privacy requirements ϵ and w. Firstly, the raw MAE results (not illustrated) of the vary-ϵ and vary-w experiments indicate that for the most mechanisms, utility behaves as expected. Specifically, they show a proportional MAE increase or decrease, towards a change of the privacy requirements. That is, for instance, in case we fix a and s and double the available budget ϵ, keeping w constant, we observe that the MAE declines by roughly Factor 2. The only notable exception is RescueDP, which has issues benefiting from higher budget in case the amplitude $a > 1,000$ holds. However, RescueDP is specifically designed for publishing multi-dimensional data with small amplitudes, explaining this result.

Regarding the influence of the period length s, we observe that, in case we fix a, ϵ and w, the MAE deterioration δ_{MAE} is equivalent for each s. Consequently, the period length is not decisive. By contrast, as Figure 5.2 showing δ_{MAE} for a period length $s = 80$ illustrates, the amplitude a is highly decisive. Here, we observe that e.g. for $a = 10$ and $\epsilon = 0.1$, Sample has the lowest MAE. However, for $a = 10,000$ and $\epsilon = 1.0$, AdaPub has the lowest MAE. The general observation is that, independent of all other investigated parameters like, season length s, ϵ or w, there are mechanisms featuring high utility either for small, or for large amplitudes.

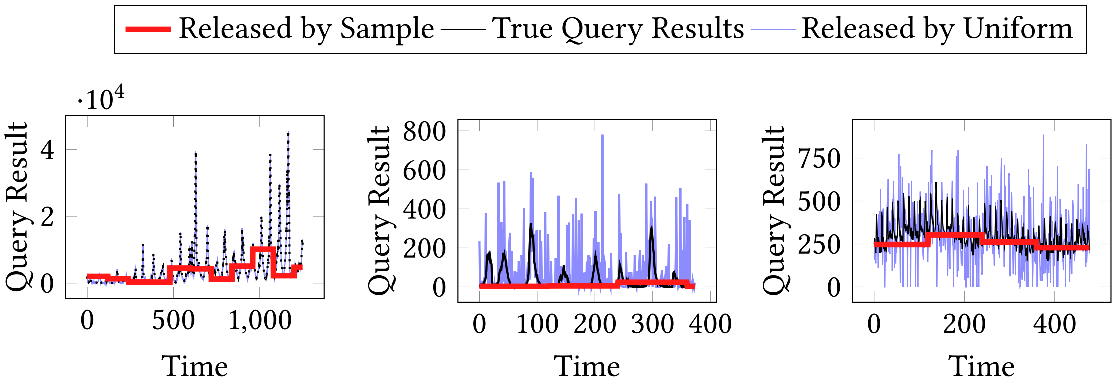
5.4.1.2. (RQ2) Can we Recommend a Mechanism for Specific Stream Properties and Privacy Requirements?

Considering Figure 5.2, we observe that for every combination of stream properties and privacy requirements, either Sample or Uniform is among the mechanisms delivering the smallest MAE. This is surprising, since it means that baseline mechanisms frequently beat sophisticated mechanisms. Subsequently, we first explain for which combinations of amplitude and privacy requirements Uniform, or Sample, respectively, deliver the smallest MAE. Secondly, we outline issues regarding hypersensitive data-adaptive sampling that sophisticated mechanisms face explaining the observed results. In the remainder of this chapter, we are particularly interested to examine whether this unexpected observation also holds for real-world streams.

Uniform Supremacy Our results suggest that mechanism Uniform is among the best mechanisms in case the amplitude a is large (i.e, $a \geq 1,000$) and the privacy requirements are not restrictive (i.e., large ϵ, small w). The larger a is, the less important is the restrictiveness. The expected MAE of Uniform is data-independent, given by MAE = $\frac{w}{\epsilon}$. Thus, in case Uniform is among the best mechanisms, this tells us little about the MAE of Uniform. Instead, the message is that the budget more sophisticated mechanisms invest, e.g., for data-adaptive sampling does not pay off, as they have an MAE exceeding $\frac{w}{\epsilon}$. In addition, for combinations in which Uniform is among the best, we expected that AdaPub and PeGaSuS consistently have a lower error than Uniform. The reason for this expectation is that they differ from Uniform only in an additional Filtering-function smoothing the perturbation noise. However, the results do not confirm our expectation. The reason that their filtering requires a fraction of the privacy budget ϵ. This investment pays off only in the downtimes between the seasons, where the query results are fairly stable. However, within the growing or shrinking phase of a season, the groups usually contain only a single time stamp and the mechanism has less budget for perturbation.

Sample Supremacy Comparing Uniform with Sample reveals that the MAE of Sample is smaller than Uniforms' if $q_{\max}$ is small and the privacy requirements are restrictive. Similar, as for Uniform, we can give a guarantee on the expected MAE. Recapitulate, for Uniform, the guarantee of the expected MAE is independent of the data. For Sample, the guarantee is independent of w, depending only on the minimum and maximum query result $[q_{\min}, q_{\max}]$, as well as ϵ. In our case, the minimum value is $q_{\min} = 0$. So, the maximum approximation error converges towards $q_{\max}$. This worst case occurs, for example, if at all time stamps that we sample, $Q(D_t) = 0$ holds, whereas, at all non-sampling time stamps, $Q(D_t) = a = q_{\max}$ holds. Moreover, the perturbation error is $\frac{1}{\epsilon}$. In sum, the MAE bound thus is $q_{\max} + \frac{1}{\epsilon}$. This means, e.g., that the bound for $a = 10$ and $\epsilon = 1$ is 11, independent of w. For Uniform, with $w = 100$, the corresponding bound is 100, i.e., Factor 10 larger. Our results furthermore reveal that we hardly observe this bound. In practice, the observed MAE of Sample is several factors smaller. The rational is that Sample has a tendency to publish the most frequent query results very accurately.

Hypersensitive Data-Adaptive Sampling The small MAEs of Sample for small amplitudes and restrictive privacy requirements suggest that one needs to minimize the perturbation error via sampling. The mechanisms BD, BA, DSAT$_w$, FAST$_w$ and RescueDP all feature data-adaptive sampling. The idea is compelling: Instead of hoping that the last release approximates the next time stamps well, the mechanism invests a fraction of the budget ϵ to monitor the stream. In case the mechanism monitors a large enough change, it releases a new query result. However, our results suggest that data-adaptive sampling does not beat sampling with a data-independent rate (conducted by Sample) consistently. Instead, they are in essence only better than Sample, in case Uniform is better than Sample. The rational is that they feature a hyper-sensitivity for small changes in the query result. Specifically, we observe the following tendency. In case the growing phase of a new season starts, such mechanisms reflect the initial changes of the query results well. In addition, they usually sample at a time point close to the peak of the first season. Thereby, they spend a



(a) Flu Outpatient stream (**Unif.**: 115; Sample: 3, 056).

(b) Flu Death stream (Unif.: 76; **Sample**: 35).

(c) Unemployment stream (Unif.: 116; **Sample**: 61).

Figure 5.3.: Plots of the one-dimensional true query result streams and the streams released by baseline mechanisms Uniform and Sample for $\epsilon = 1.0$ and $w = 120$. For each stream, the caption states the mean absolute error (MAE) of the streams released by the mechanisms. The mechanism with the lower error is highlighted in bold.

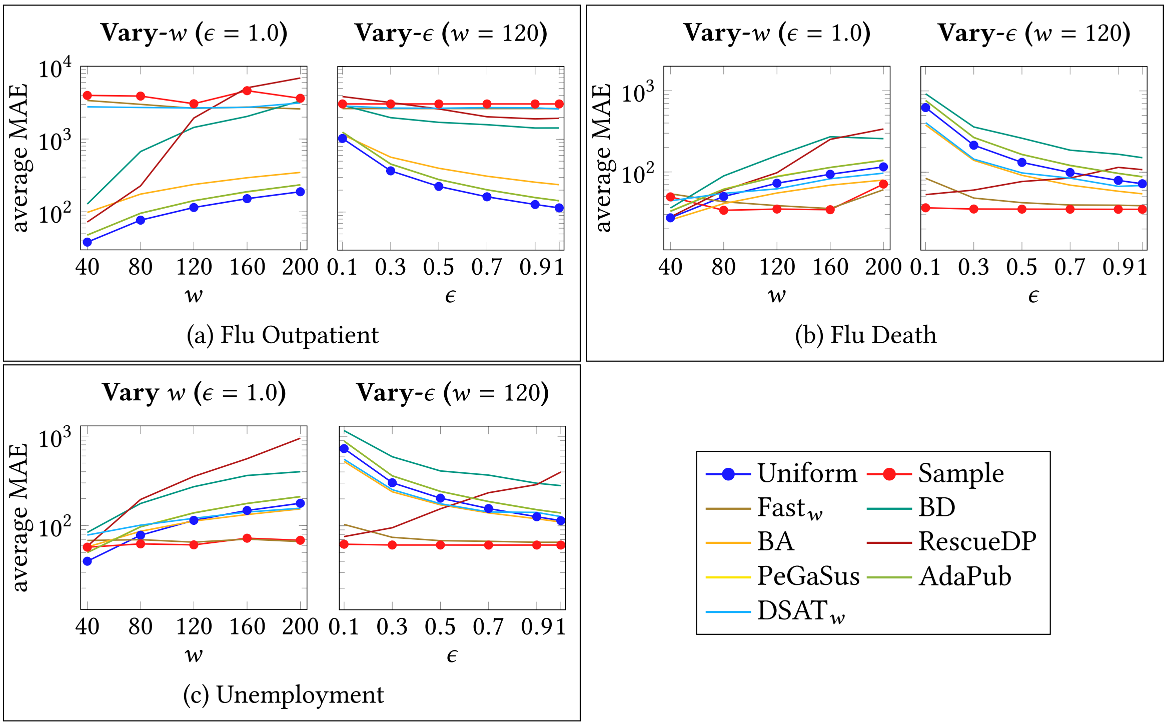
large fraction of their available budget already in the growing phase. Thus, they are much more reluctant spending budget in shrinking phase. This means they still publish large query results in the shrinking phase, incurring a high MAE. That becomes worse, in case multiple seasons fit into one window of size w. This usually holds for all commonly used window sizes w and streams.

5.4.2. One-dimensional Real-World Streams

Next, we present our evaluation results of all one-dimensional real-world streams. Thereby, we have two objectives discussed in separate subsections. Firstly, we are interested whether the results of real-world streams are consistent with the ones of the artificial streams. This holds specifically for the observed baseline supremacy. Secondly, we aim at understanding what the sanitization error (i.e., the MAE) means in context of the data streams. In a nutshell, our key results are: Firstly, the results on real-world streams are consistent with the ones obtained in the micro benchmarks. Secondly, the error metrics, though widely used, are not well-suited in the streaming setting. For example, despite Sample delivers a good MAE (and MRE), it entirely destroys the seasonality of the stream. Even constantly publishing the most frequent query results usually yields better utility than most sophisticated mechanisms have.

5.4.2.1. Confirmation of Micro Benchmark Results

In Figure 5.4, we depict the MAE for all considered mechanisms and one-dimensional real-world streams. The baseline mechanisms are marked with ∘. We discuss our key observations next. The results of the real-world streams confirm the result of the micro benchmarks as follows. Flu Death and Unemployment are two medium-amplitude streams



(a) Flu Outpatient

(b) Flu Death

(c) Unemployment

Figure 5.4.: Average MAE of w-event mechanisms for each one-dimensional real-world stream in the vary-w and vary-ϵ experiments. The legend is the same for all graphs. The baseline mechanisms are marked with ∘.

with $a < 1,000$ (cf. Figure 5.3). The Flu Outpatient stream has a large amplitude, with a maximum of about $2 \cdot 10^4$. As expected, for the large amplitude stream, Uniform has the lowest MAE. Notably, the MAE is significantly smaller than the expected MAE $\frac{w}{\epsilon}$. For instance, on the Unemployment stream where $Q(D_t)$ is frequently close to 0, the MAE is almost only half as large as expected. The reason is the truncating. Interestingly, for the Unemployment stream, we do observe a slight utility improvement by PeGaSuS towards Uniform. On the medium amplitude streams, usually, Sample has the lowest MAE. Only in case the privacy requirements are not restrictive, i.e., $\epsilon = 1$ and $w = 40$, Uniform and most other mechanism have slightly lower MAE. As in the micro benchmarks, we observe that conducting data-adaptive sampling is not superior to equidistant data-independent sampling. Finally, as in the micro benchmarks, we observe an anomalous behavior of RescueDP: For streams with large amplitude, increasing the available budget does not improve utility. Instead, it has the opposite effect.

5.4.2.2. Semantics of the Sanitization Error

So far, most studies used MAE and MRE metrics for determining mechanism utility (see Table 5.3). Considering our results on these error metrics, we observe intrinsic anomalies. For instance, the utility of Sample appears to be almost independent of the privacy requirements. Thus, we examine the semantics of the error values next. To do so, we consider common applications performed on data streams. Common objectives on streams are forecasting or change detection algorithms. For them, the preservation of the stream properties from Section 5.3.2 is highly relevant. However, there is little understanding how the MAE and MRE relate to them. To this end, we subsequently examine the sanitized query results of mechanism Sample and Uniform with respect to seasonality (i.e., period length and amplitude) and level. This way, we ensure that there is one mechanism having a low MAE for every stream due to the baseline supremacy. For the explanation of our results, we use exemplary the sanitized real-world streams shown in Figure 5.3. We verified that our explanations hold in general.

Maintaining Seasonal Growing and Shrinking of the Stream The exemplary results in Figure 5.3 reveal, that independent of the observed MAE, using mechanism Sample means that the stream entirely loses its seasonality. This also holds for streams where Sample is the best mechanism. In case the mechanism does not sample multiple times per season, one approximates an entire season with a single value for every time stamp. Thus, a small MAE of Sample suggests that the stream contains a large amount of similar query results, which the mechanism likely hits upon data-independent sampling. Simply releasing the sanitized query result at the first time stamp for *every* subsequent time stamp yields a similar utility for all streams used in studies so far (i.e., MAE and MRE).

The mechanism Uniform maintains the seasonality well, in case amplitudes are large compared to the noise introduced for sanitization. As the expected noise is $\frac{w}{\epsilon}$, one only needs to know the amplitude to decide whether Uniform delivers acceptable utility for the data administrator. However, this is not reflected by the MAE (nor MRE). For instance, in Figure 5.3 (b) the MAE is smaller than in Figure 5.3 (a). This holds despite one can clearly observe the seasonality in (a), but not in (b). That is, the MAE has no meaning for maintaining seasonality. It has a meaning how well Uniform maintains the level, as we discuss next.

Maintaining Level and Amplitude of the Stream Recapitulate, that the true level of the stream is defined by q_{min}, and the true amplitude a by $q_{\text{max}} - q_{\text{min}}$. The level and amplitude of the sanitized stream released by Uniform depend on the true level and amplitude. In case $q_{\text{min}} > \frac{w}{\epsilon}$ holds, i.e., it is higher than the expected noise of Uniform, the sanitized stream released by Uniform features the domain $[q_{\text{min}} - \frac{w}{\epsilon}, q_{\text{max}} + \frac{w}{\epsilon}]$. We observe this well in Figure 2(c). There, the measured MAE = 115.6 fairly equals the expected MAE of $\frac{w=120}{\epsilon=1} = 120$. By contrast, in Figure 2(b), q_{min} is close to 0, i.e., the minimum possible value of a count query. Thus, conducting a truncate for count queries means that the expected level change is $[\max(q_{\text{min}} - \frac{w}{\epsilon}, 0), q_{\text{max}} + \frac{w}{\epsilon}]$.

Since Sample has a low perturbation error, it does not enlarge the domain. That means, Sample does hardly publish smaller values than the original min value or larger values than the original max value. However, the sanitized streams usually miss the seasonal peaks of the true streams. Large MAE values, specifically values exceeding the MAE of Uniform, indicate that the stream contains large amplitudes, poorly reflected in the stream released by Sample. Small MAE values, in turn, indicate that are no large seasonal changes and Sample approximates small counts very accurately.

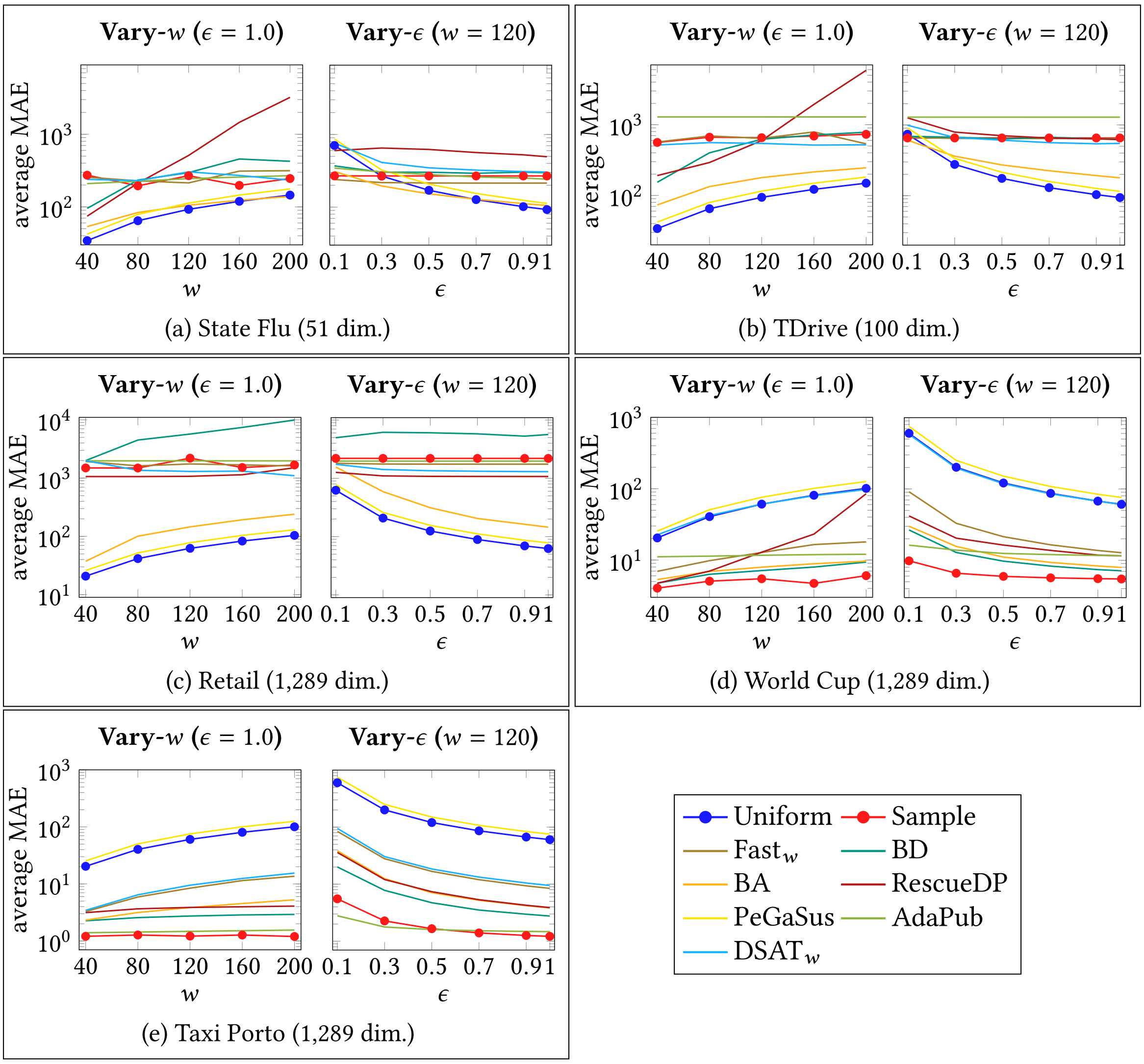
Table 5.7.: Properties of the multi-dimensional streams after preprocessing. The query result distribution states the distribution of the true query results over all sampled dimensions.

Stream S	Dimensionality	Length p	Query result distribution		
			$q_{\min}$	$q_{\max}$	90% quantile
StateFlu	51	492	0	11,452	924
TDrive	100	672	0	39,871	1,772
Retail	1,298	374	0	372,306	15,089
Taxi Porto	1,298	672	0	317	2
WorldCup	1,298	1,320	0	16,928	0

5.4.3. Multi-dimensional Real-World Streams

Next, we present our results of all multi-dimensional streams. Their properties after preprocessing, that partially included a dimension sampling, are shown in Table 5.7. The mechanism utility in Figure 5.5. Below, we aim at confirming the results we obtained on the one-dimensional streams. In addition, we focus on studying the effect of adaptive dimension grouping as an option to improve utility for multi-dimensional streams. The core idea of adaptive dimension grouping is to find a group g of dimensions that have currently a similar query result (i.e., that are correlated). This can be exploited in two ways. First, in case of BD, the sampling decision is taken per group. Then, the groups in which the query result changes more frequently are sampled more frequently than groups in which the query result is more constant. For AdaPub and RescueDP, the grouping is exploited in the perturbation function. Specifically, the mechanism perturbs the sum of the query results over all dimensions in group g, and then assigns each dimension the average of the perturbed sum. This reduces the expected perturbation error from $\frac{1}{\epsilon_t}$ to $\frac{1}{\epsilon_t \cdot |g|}$ [Wan+16a]. Hence, with increasing dimensionality, this is expected to highly reduce the perturbation error.

For the one-dimensional streams, we observed a baseline supremacy with amplitude and privacy requirements as decisive factor between Uniform and Sample, as well as hypersensitive data-adaptive sampling. Generally, the Figure 5.5 confirms together with Table 5.7 both observations.



(a) State Flu (51 dim.)

(b) TDrive (100 dim.)

(c) Retail (1,289 dim.)

(d) World Cup (1,289 dim.)

(e) Taxi Porto (1,289 dim.)

Figure 5.5.: Average MAE of w-event mechanisms for each multidimensional stream for the vary-w and vary-ϵ experiments. The streams are ordered from the smallest to the highest number of dimensions. The legend is the same for all graphs. The baseline mechanisms are marked with ∘.

On the State Flu, TDrive and Retail stream having an amplitude $> 10,000$, as expected, Uniform is among the best mechanisms. Only on the stream State Flu for small ϵ-values, a couple of other mechanism outperform Uniform. On the Wold Cup and Taxi Porto stream, Sample is among the best mechanisms. However, AdaPub has a low error, too, and even outperforms Sample for individual values of ϵ. This is interesting, because for the one-dimensional streams, AdaPub has low error iff Uniform is among the best mechanisms. This suggests that these two streams differ significantly from the first three streams. Table 5.7 reveals that both streams are sparse. Specifically, for most time stamps and dimensions, the query result is very small or even zero, i.e., minimal. We observe that AdaPubs error is for seven experiments nearly constant. For one experiment, this is the case for RescueDP, too. The rationale is that over time, the number of groups converges to 1, meaning that the mechanism releases the same query result for all dimensions. If all dimensions are in one group, the perturbation error is low, meaning that the mean error is only slightly influenced by w and ϵ. Additionally remarkable is the mean error of BD for these two streams. In the micro benchmark, and on the one-dimensional real-world streams, there is no stream for which BD is among the best mechanisms for all privacy requirements. However, BD is among the best mechanisms for WorldCup and Taxi Porto streams. Whether the rational lies in the grouping is however with our results not decidable.

5.4.4. Takeaways

The primary outcomes of our study are takeaways for practitioners, as well as for researchers. We discuss them next.

5.4.4.1. Takeaways for Practitioners

Our takeaway for practitioners forms a catalog of recommendations that aim at understanding and controlling the expected utility of differentially private stream monitoring. It contains three recommendations and targets at data owners, as well as administrators not being experts in differently privacy, but having a sophisticated background in data analysis.

Data Owners: Meaningful Window Size The data owner is responsible for selecting the privacy requirements ϵ and w. By definition of w-event differential privacy, the window size w refers to the length of the longest event-sequence the mechanism aims to protect with privacy budget ϵ. The selection of w is clearly use-case dependent. However, our literature study suggests that there is a tendency for investigating unnaturally large values of w, which causes large perturbation noise. For the seasonal data, the length of a season s may serve as a natural upper bound for w. Our specific recommendation for data owners is specifying the event-sequence the data owner aims to protect, such as the maximum length of a trajectory in location monitoring use case. This way one can determine a meaningful value for w.

Data Administrator: Meaningful Utility Metrics The data administrator is responsible for selecting a mechanism. Our results suggest that the sanitization error hardly allows conclusion whether one is still able to conduct frequent analysis tasks on streams, like forecasting or anomaly detection. So, we recommend data administrators to select mechanisms that provide high utility with respect to an application specific metric, or to understand the semantics of the sanitization error with respect to the application better.

Data Administrator: Consider the Selection of Baselines Our study indicates that, regarding data utility and predictability of the expected error, there is little a data administrator can do wrong with the Uniform or a Uniform-Sample hybrid mechanism as explained below. Specifically, if an expected error of $\frac{w}{\epsilon}$ is sufficient and the application requires query results for each time stamp, e.g., because one targets at *instant* change detection, we recommend to use Uniform. In case one does not need to detect the change instantly, one can trade time to minimize the perturbation error using a Uniform-Sample hybrid mechanism. Such mechanism samples every x^{th} time stamp releasing more accurate query results at sampling time stamps than Uniform. Specifically, it decreases the perturbation error at sampling time stamps compared to Uniform from $\frac{w}{\epsilon}$ to $\frac{w}{\epsilon \cdot k}$. In particular, in combination with selecting a meaningful value for w, this mechanism may provide sufficient utility for many applications.

5.4.4.2. Takeaways for Researchers

Our takeaway for researchers target primarily at the way how to design the four functions of the w-event mechanism framework. We discuss the functions according to their order in Algorithm 2.

isSamplingPoint-Function Currently, in case the mechanism does not sample, it approximates the current query result with the last sanitized query result. This works well at the time stamps between the seasons, when the counts remain stable. However, it yields high error in case the stream is in a growing or shrinking phase. Consequently, we propose to investigate on mechanisms that consider the seasonal nature of streams upon approximation. For instance, the mechanism could invest time and budget when starting to release a new stream, to learn a model of the stream, e.g., using machine learning in a differentially private way. The model can not only be used for the sampling decision, but also to predict whether the stream is currently in a growing or shrinking phase. If the change in the stream is not large enough that it provokes the mechanism to sample, the mechanism can correct the approximation based on the latest trend. Observe that this is orthogonal to filtering based on time-grouping, because the filter is only applied at sampled time stamps.

BudgetAllocation-Function We observe that today's mechanisms rather optimistically allocate budget trying to accurately reflect small changes in the stream. For instance, mechanism BD allocates half of its remaining budget per sampled time stamp. However, our results indicate that this yields low utility in case the stream contains large amplitudes.

Our results suggest that targeting at homogeneously distributing the budget over sampling time stamps usually results in the best utility. To this end, the mechanism may limit the number of sampling time stamps in the current active window.

Perturbation-Function Our recommendation regarding perturbation refers to mechanisms using the dimension grouping. We frequently observe that the dimensions "lump together" into few or even a single group. Then, they remain in their group even if they are not correlated anymore. Thus, we recommend computing the grouping not only on the sanitized query results and explicitly consider a functionality to ungroup dimensions not correlated anymore. Additionally, we also propose to question whether researchers should focus on dimension grouping in future work. The rationale is that using the dimension-grouping approach does only not violate privacy, in case the correlation of the dimension query results is spurious. Otherwise, correlated dimensions result from an event the data owner intends to hide that may affect multiple rows in a database D_t, and not only a single one as presumed in the original definition of differential privacy [Dwo08]. The extension of differential privacy capturing this group-differential privacy [DR+14]. It states that the increase of ΔQ entirely nullifies the benefit of dimension grouping.

Filtering-Function Our results indicates that grouping over time stamps with a grouping function that requires budget does not yield a utility improvement. Consequently, we suggest to conduct research on filtering functions that do not require budget.

Finally, in case a researcher proposes a novel mechanism, we strongly recommend conducting an empirical evaluation based on the principles, we introduced in Section 5.2. We argue this is the only way to ensure that future studies have a better comparability than most studies known so far. Most urgently, we recommend including *both* baseline mechanism.

5.5. Summary

In this chapter, we perform a large-scale literature survey revealing not only the generic w-event differential privacy mechanism framework, but also limitations of previous experimental studies comparing w-event mechanisms, preventing the comparability of the studies. These limitations prevent data administrators from selecting the best mechanism, and slows down future research. Consequently, we identify requirements on each element of a typical experimental study, and conduct a benchmark that fulfills these requirements and consists of 252,000 single experiments. We discuss the insights of our benchmark, as well as takeaways for data administrators and researchers. One insight is the unexpected baseline supremacy, saying that for each combination of mechanism and stream, one of the two baselines is among the mechanisms with the highest utility.

6. Case Study: Utility Metrics for Electricity Distribution Grid Monitoring

In the previous chapter, we observed that the baseline mechanisms Sample and Uniform provide at least competitive utility to state-of-the-art mechanisms. Consequently, in this chapter, we ask whether they provide already enough utility to provide reasonable application results. To this end, we perform a case study on the application *grid monitoring system.*[1] The background is that smart meter roll-out in Europe comes with availability of quarter-hourly power measurement streams of customers, that are the data owners considered in this study. Distribution system operators (DSOs), serving as service users in this study, aim to turn these data streams into value by using them to monitor the low-voltage grid continuously. To this end, DSOs use a plurality of automated grid analyses, like voltage and line loading analysis, that link measurements with additional data, like the grid topology from a geographic information system [Iov+21; Maa+15; Nie+17]. They either deploy the system by themselves, or do make use of a service. However, it is well-known that active as well as reactive power measurements facilitate everyone who has access to this data to inference daily habits of the customers [Mol+10; FLC17]. Consequently, customers may have privacy requirements regarding their measurements. A prominent example is hiding certain power patters like appliance usage cycles [KBB15]. To tackle these requirements, a trusted data gateway serving as data administrator, that is locally deployed at each customer connection, sanitizes the measurements with respect to the privacy requirements. After that, the gateway transmits the sanitized measurements to the DSO or service provider running the grid monitoring system.

In this chapter, we study the utility of differentially private PETs for grid monitoring. Specifically, we are interested to answers the following research questions:

(RQ1) Which utility metrics are appropriate to measure the utility of grid monitoring? For instance, is the sanitization error of the measurements, frequently used in related work on w-event differential privacy, meaningful, or do we need a separate metric for each grid analysis.

(RQ2) How does the utility of differentially private grid monitoring under reasonable privacy requirements behave? Can we give an intuition whether the utility is low or high?

[1] The remainder of this chapter bases on the article Christine Schäler and Hans-Peter Schwefel. *Studying Utility Metrics for Differentially Private Low-Voltage Grid Monitoring.* Tech. rep. Karlsruhe Institute of Technology, KIT Scientific Working Papers 193, 2022. Compared to the article, the sections have been shortened to be less repetitive, contain minor corrections, as well as formatting and notation changes to be in line with the format and structure of this thesis.

Answering these research questions is challenging for three reasons. First, to answer them, candidates for utility metrics are required. This is challenging due to the plurality of results [Pom+18] a grid monitoring system usually outputs. Second, given a utility metric, a subsequent challenge is identifying utility thresholds stating that the grid monitoring results are accurate enough for DSOs. They should be DSO-independent to support the generality of our results. Third, it is essential to study the utility for reasonable privacy requirements. Intuitively, reasonable privacy is given if realistic privacy requirements are fulfilled. However, selecting them and studying the utility with respect to a *plurality* of realistic requirements is challenging without conducting an unlimited number of experiments.

Contributions and Outline In this chapter, we provide the following three contributions. First, we identify three steps of differentially private grid monitoring, namely, (1) measurement sanitization with the PET, (2) load-flow analysis as first step in each grid monitoring system, (3) subsequent analyses like voltage and line loading analysis in Section 6.1, and candidates for utility metrics for all three in Section 6.2. For each step, the metrics are in line with the metrics used by experts for the individual steps. To give an intuition on the utility of grid monitoring results, we consider the frequent case in which measurement devices are not fully accurate, since the DSOs already accept this reduced utility. Second, in Section 6.3, we identify realistic power patterns from literature and decode them into privacy parameters. To study the utility for a plurality of them with a limited number of experiments, we leverage the baseline mechanism Uniform mechanism that unpacks multiple requirements into one scaling parameter. Third, to answer our research questions, we perform experiments on a real-world grid topology with realistic measurements in Section 6.4. With respect to the first question, our study indicates that the utility of grid monitoring decreases faster than metrics measuring the sanitization error suggest, indicating that grid monitoring specific metrics are needed to assess utility meaningfully. With respect to the second one, we observe that the utility of grid monitoring decreases faster than the sanitization error suggests, indicating that the utility of an application is frequently worse than literature indicates. Finally, we provide a summary of this chapter.

6.1. Specific Fundamentals and Related Work

In this section, we first sketch fundamentals on grid monitoring systems, resulting in requirements on the data used in our study. Then, we introduce related work on differentially private grid monitoring.

6.1.1. Grid Monitoring Systems

In this section, we first sketch fundamentals on grid topologies and measurements serving as an input into grid monitoring systems. Second, we introduce grid monitoring systems as detailed as needed to identify requirements on study data and to define utility metrics in the remainder.

6.1.1.1. Grid Topologies and Measurements

Subsequently, we sketch fundamentals on grid topologies, relevant measurands and measurement scenarios together with notation.

Grid Topology A low-voltage grid topology is given by

1. a grid topology graph $G = (N, E)$ in which the edges E are the lines, and N the nodes,
2. a function $typeN : N \longrightarrow \{\text{Trafo, customer connection box, junction box, sleeve}\}$ assigning nodes a type, and
3. a function $typeE : E \longrightarrow \mathbb{R}^4$ assigning lines quintuples of resistance (R1), reactance (X1), capacitance (C1) and ampacity ($\mathsf{I}_{\mathsf{max}}$) in ampere.

Without loss of generality, we assume that G is a tree, as this simplifies explanations and applies for most grids. A *feeder* is a single branch in the tree. Each node and line has a specific type. For nodes, it holds that the unique root of the tree is of type Trafo. It serves as connection point to the parent medium-voltage grid. The leaf nodes of the grid are customer connection boxes (CCBs) connecting residential and industrial customers with the grid. The nodes between the substation and the customer connection boxes are either junction boxes or sleeves. The type of a line specifies its electrical parameter resistance, reactance, capacitance and ampacity.

Measurands In a grid, DSOs deploy measurement devices at nodes that measure specific measurands. Typically, they are measured per phase. However, grid monitoring systems usually consider a one-phase representation of the three-phase grid [Pom+18].

Let $n \in N$ be a node, and $e \in E$ be a line. We denote with $\mathsf{V}(n, t)$ the average voltage magnitude in volts at the secondary side of node n in a time interval ending at time stamp t. Similarly, with $\mathsf{P}(n, t)$ and $\mathsf{Q}(n, t)$, we denote the total active power in kilowatts (kW) and reactive power in kilovar (kVar) injected at time t and node n into the grid. Additionally, $\mathsf{I}(e, t)$ is the average current magnitude in ampere in a time interval ending at time stamp t at the secondary side of line e. In case they are clear from the context, we omit the parameters n and e.

Measurement Scenarios The measurement scenario of a grid specifies which measurands are measured at which grid node or line. Considering real-world scenarios, we identified two measurement scenarios, namely PQ and P only, imposing different privacy requirements of a PET. Subsequently, we describe them.

Measurement Scenario PQ This is the measurement scenario stated in Table 6.1. Here, the voltages at the Trafo, as well as active and reactive power at all customer connection boxes are measured.

Measurement Scenario P only In contrast to scenario PQ, the meters at the customer connection boxes measure only active power, but not reactive power. This is a frequent setting in real-world. For instance, Smart Meters in Germany are, by default, configured accordingly [Bun21].

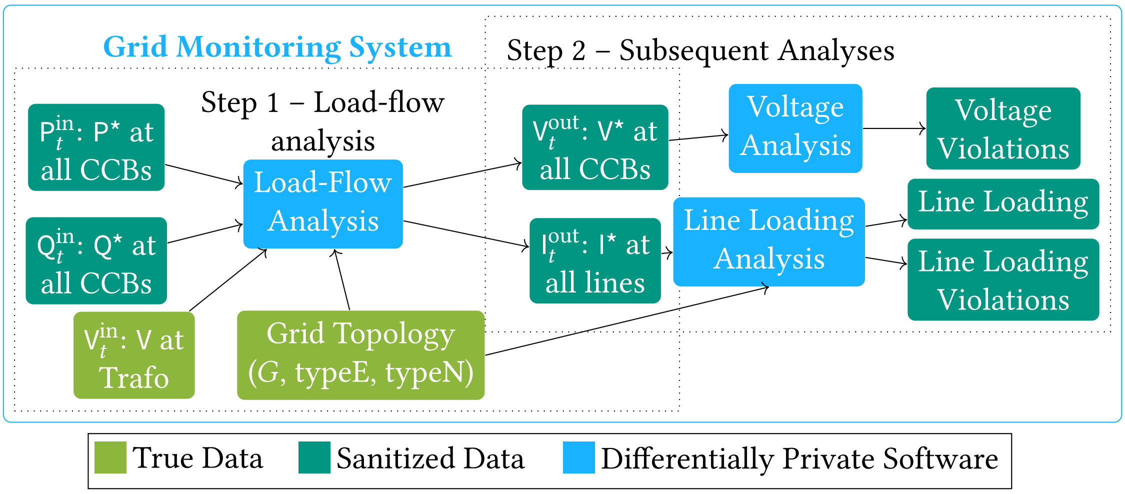

Grid Monitoring System
Step 1 – Load-flow analysis
Step 2 – Subsequent Analyses
P_t^{in}: P* at all CCBs
Q_t^{in}: Q* at all CCBs
V_t^{in}: V at Trafo
Load-Flow Analysis
Grid Topology (G, typeE, typeN)
V_t^{out}: V* at all CCBs
I_t^{out}: I* at all lines
Voltage Analysis
Line Loading Analysis
Voltage Violations
Line Loading
Line Loading Violations
🟩 True Data
🟩 Sanitized Data
🟦 Differentially Private Software

Figure 6.1.: Illustration of the two-step process of grid monitoring.

Table 6.1.: Measured measurands, and the ones calculated by load-flow analysis. Reactive power at CCBs are only measured in scenario PQ, and are replaced by pseudo-measurements in scenario P only.

Node Type	V	P	Q	I
Transformer	measured	-	-	-
Junction box	calculated	-	-	-
CCB	calculated	measured	measured (in PQ)	-
Lines	-	-	-	calculated

6.1.1.2. Grid Monitoring Systems

As illustrated in Figure 6.1, a grid monitoring system usually implements a two-step process: A load-flow analysis determining non-measured measurands followed by a plurality of subsequent grid analyses calculating system indicators. Below, we sketch both steps briefly based on [Iov+21], and state specifics relevant for ensuring the reproducibility of our results.

Step 1 – Load-flow Analysis The load-flow analysis calculates voltages at all nodes except the Trafo, and the currents at all lines. The algorithm is stated in Algorithm 4. There, based on grid topology and measurements, the algorithm obtains a set of linearized, originally non-linear, power balance equations. The unknown variables correspond to voltages at non-Trafo nodes, and the known ones to active and reactive power at the customer connection boxes. By solving this system with the iterative Newton-Raphson method [Ben66] that minimizes the mismatches in active and reactive power at customer connection boxes, the algorithm obtains the voltages of all nodes. Based on the obtained voltages, the algorithm calculates the currents. In a grid, except the power at the Trafo, the power, voltage and current measurements in one feeder are independent of the measurements in another feeder. Consequently, the load-flow analyses on different feeders

Algorithm 4 Load-flow Analysis at timestamp t [PG20]

```
function LOADFLOW_t(G, typeN, typeE, $V_t^{in}$, $P_t^{in}$, $Q_t^{in}$)
    $V_t^{init} \leftarrow V_t^{in}$
    for $c \in N$ with typeN(c) == CCB do
        $V_t(c,t) \leftarrow 0$                                  ▷ Initialization
        $V_t^{init} \leftarrow V_t^{init} \cup \{V(c,t)\}$
    end for
    $V_t^{out} \leftarrow$ Newton-Raphson(G, typeE, $V_t^{init}$, $P_t^{in}$, $Q_t^{in}$)
    $I_t^{out} \leftarrow \{\frac{|V(c,t)-V(c_j,t)|}{\text{typeE}(e_{i,j}).R1} \mid e_{i,j} \in E\}$
    return $V_t^{out}$, $I_t^{out}$
end function
```

are independent. In our study, we use the load-flow analysis that is implemented in a grid model developed in [Pom+18], and successfully validated in [Nai+21]. It is based on the MATLAB implementation in [PG20]. We stop as soon as the mismatches are smaller than 10^{-5}, or after 100 iterations otherwise. Reactive power measurements are a necessary input into the load-flow analysis. Consequently, in measurement scenario P only, so-called pseudo measurements are generated using background knowledge on the customer connection [Man+12]. Since this background knowledge is private information as well, in our study, we use for reactive power the pseudo-measurement 0 kilovar, which is a common choice in case no background knowledge is available.

Step 2 – Subsequent Grid Analyses For a low-voltage feeder, two grid analyses, namely, voltage analysis and line loading analysis are relevant [Iov+21]. Subsequently, we introduce both analyses.

Voltage Analysis By European standards [Eur19], the DSO must ensure that the voltage magnitudes fluctuates at maximum +/−10% of the nominal voltage of 400 V. To alert DSOs before they violate this hard limit, the voltage analysis verifies whether the voltages at all nodes are in range $+x\%$ (*over*voltage violation) and $-y\%$ (*under*voltage violation) of 400 V, and reports violations in case they are not. In line with industry standards, we use $x = y = 5$.

Line Loading Analysis Lines can manage a certain nominal current, that is given by their ampacity. If the actual current is higher, they overheat. The line loading analysis therefore calculates for $e \in E$ the load in percent by

$$\text{Load}(e,t) = \frac{|(\mathsf{I}(e,t)|}{\text{typeE}(e).\mathsf{I}_{\max}}$$

and alerts the DSO in case the load is above a certain limit. That way, DSOs can react before the lines overheat. A common load limit that we use is 90% [Sch+21].

Table 6.2.: Comparison related work on differential privacy for measurement data with respect to the DP variant considered.

Ref.	$w = 1$	$w \in [2, \infty)$	$w \rightarrow \infty$
[KBB15]	✓	✗	✗
[Eib+18]	✓	✗	✗
[EE17]	✗	✗	✓
[ÁC11]	✓	✓	✗
[SDT15]	✓	✗	✗
[FV18]	✓[a]	✗	✗
This chapter	✓	✓	✓

[a] Privacy potentially violated by using faithfulness value β computed on the measured data during post processing.

6.1.2. Differentially Private Grid Monitoring

In this section, we sketch related work on differentially private PETs for measurements, utility metrics for them, as well as orthogonal work on security and privacy with respect to additional data used in grid monitoring systems.

Differentially Privacy for Measurement Streams Event-and user level differential privacy has been investigated for measurement data before (see Table 6.2). However, event-level DP features only limited privacy for streams [Cao+18; ÁC11], and user-level differential privacy limited utility, already for finite time series [EE17]. w-event DP for $w \in [2, \infty)$ has proven its worth for, e.g., location streams [Wan+16a; Li+15a; Liu+18], so far, it has been only sparsely investigated for measurement data. [ÁC11] considers the concept even before the proposal of w-event DP. However, it features only a limited number of experiments with respect to w-event DP. Additionally, it considers a special mechanism called distributed differential privacy, that generally results in lower utility than other differential private mechanisms.

Measuring Utility of PETs Related work focusing on w-event DP for measurement streams assesses this error *either* by the sanitization error [Bar+14; EE17; ÁC11], *or* by the error of a specific analysis. The latter includes local energy market analysis [KBB15], specific forecasting algorithms [Eib+18], peak-load analysis [Lia+14] or state estimation [SDT15]. To the best of our knowledge, these two types of utility metrics have not been systematically related to each other before, nor intuitions on "high enough" utility are given.

Security and Privacy with Respect to Additional Data Besides privacy with respect to measurements, customers may have additional privacy requirements, like secure data transmission [Lyu+17; RN13]. Additionally, in case an untrusted service provider hosts the grid monitoring system, DSOs have to transmit the additional data, like grid topology data, needed as input for grid monitoring systems. In this context, [Nan+19; FMV20]

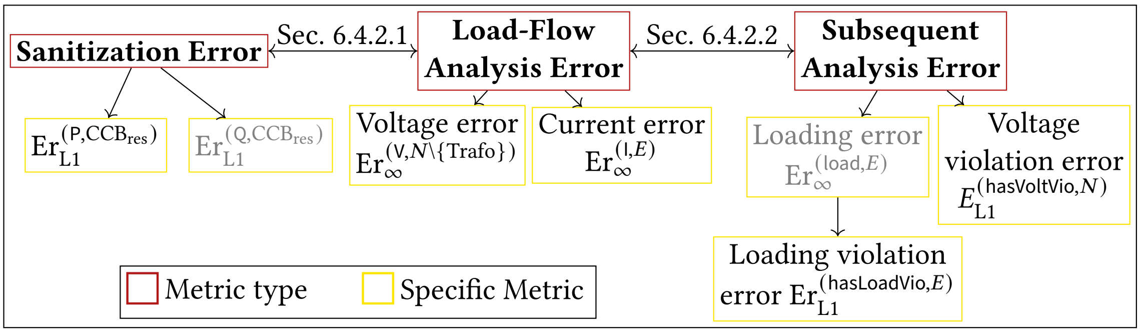

Sanitization Error
Sec. 6.4.2.1
Load-Flow Analysis Error
Sec. 6.4.2.2
Subsequent Analysis Error
$\mathrm{Er}_{\mathrm{L1}}^{(\mathsf{P},\mathrm{CCB}_{\mathrm{res}})}$
$\mathrm{Er}_{\mathrm{L1}}^{(\mathsf{Q},\mathrm{CCB}_{\mathrm{res}})}$
Voltage error $\mathrm{Er}_{\infty}^{(\mathsf{V},N\setminus\{\mathrm{Trafo}\})}$
Current error $\mathrm{Er}_{\infty}^{(\mathsf{I},E)}$
Loading error $\mathrm{Er}_{\infty}^{(\mathsf{load},E)}$
Voltage violation error $E_{\mathrm{L1}}^{(\mathsf{hasVoltVio},N)}$
Loading violation error $\mathrm{Er}_{\mathrm{L1}}^{(\mathsf{hasLoadVio},E)}$
□ Metric type
□ Specific Metric

Figure 6.2.: Identified utility metric types together with associated error metrics. The gray metrics are not considered in Section 6.4.

focus on differential privacy for grid topology data, like line parameters. However, these approaches are orthogonal to our research questions.

6.2. Study Design – Grid Monitoring

In this section, we first define candidates for utility metrics for grid monitoring. In our experiments, we investigate which of them are appropriate. Second, we state how we generate measurement errors used to give an intuition on the utility achieved by a PET. Finally, select the data for our study data based on in the identified requirements.

6.2.1. Measuring Utility – Candidate Utility Metrics

Considering Figure 6.1, we identified three types of utility metrics serving as candidates for our study shown in Figure 6.2. These are the sanitization error, load-flow analysis error as well as subsequent analysis error. In the remainder of this section, we identify the specific metrics commonly used in related work for each of the types, and propose due to lack of related work novel subsequent analysis error metrics. We state them by using the following notation: $\mathcal{E}$ is a measurand or system indicator (like line loading), and $\mathcal{X}$ is a set of lines or nodes. Measurands or system indicators *with* superscript $*$ belong to sanitized measurements or system indicators calculate by using sanitized measurements. Consequently, e.g., P and Q are the measured powers, and P^* and Q^* the sanitized ones. Additionally, p is the number of time intervals in the measurement stream prefix.

Sanitization Error Metric To measure the sanitization error, as identified in Chapter 5, related work on w-event differential privacy frequently uses the mean absolute error. For measurand $\mathcal{E}$ and set of lines or nodes $\mathcal{X}$ it is defined by

$$\mathrm{Er}_{\mathrm{L1}}^{(\mathcal{E},\mathcal{X})} = \frac{1}{p \cdot |\mathcal{X}|} \sum_{t=1}^{p} \sum_{x \in \mathcal{X}} |\mathcal{E}(x,t) - \mathcal{E}^*(x,t)|,$$

i.e., the average over the L1-norm between $\mathcal{E}$ and $\mathcal{E}^*$ at each time stamp and node or line. Since we sanitize active and reactive power of residential customers, in our study, we focus on $\mathrm{Er}_{\mathrm{L1}}^{(\mathsf{P},\mathrm{CCB}_{\mathrm{res}})}$ and $\mathrm{Er}_{\mathrm{L1}}^{(\mathsf{Q},\mathrm{CCB}_{\mathrm{res}})}$, in which $\mathrm{CCB}_{\mathrm{res}}$ is the sub-set of customer connection boxes of N belonging to residential customers.

Load-flow Analysis Error Metric The load-flow analysis calculates the voltages and currents that are subsequently used in voltage and line loading analysis. This suggests measuring the utility of load-flow analysis by the error in calculated voltages and current. To define these errors, power engineers usually rely on the maximum norm [NI20]. Consequently, to quantify the voltage and current error, we use

$$\mathrm{Er}_{\infty}^{(\mathcal{E},\mathcal{X})} = \frac{1}{p} \sum_{t=1}^{p} \max_{x \in \mathcal{X}} |\mathcal{E}(x,t) - \mathcal{E}^*(n,t)|.$$

Note that our study, measurement streams contains the measured voltages and currents at all nodes and lines, facilitating us to calculate this error. Since the load-flow analysis calculates the voltages at all nodes except the Trafo, we are interested in $\mathrm{Er}_{\infty}^{(\mathsf{V},N\setminus\{\mathrm{Trafo}\})}$. For currents, we are interested in the current error at all lines, i.e., $\mathrm{Er}_{\infty}^{(\mathsf{I},E)}$.

Subsequent Analyses Error Metrics To the best of our knowledge, there is no related work measuring the utility of voltage and line loading analysis. Consequently, we propose novel error metrics considering the benefit of the DSO of both analyses.

Voltage Analysis Let the variable $\mathsf{hasVoltVio}(n,t)$ indicate whether there is a voltage violation at node n and time stamp t. With that notation, we measure the error in the total number of voltage violations that is given by $E_{\mathrm{L1}}^{(\mathsf{hasVoltVio},N)}$.

Line Loading Analysis The line loading analysis first calculates the load of each line, and then whether the load is below the defined limit. We define an error metric for each of the two steps. First, we define loading error in percentage points (% P) by $\mathrm{Er}_{\infty}^{(\mathsf{load},E)}$. Second, let $\mathsf{hasLoadVio}(e,t)$ be the indicator variable for a line loading violation of line e at time stamp t. With that notation, we measure the error in the total number of loading violations by $\mathrm{Er}_{\mathrm{L1}}^{(\mathsf{hasLoadVio},E)}$.

6.2.2. Assessing Utility – Measurement Errors

A PET is not the only influence factor that may reduce the utility of grid monitoring system. Specifically, measurement devices are typically not completely accurate, meaning that even the analysis results on measured data are erroneous. Consequently, to assess whether the utility provided by the PET is reasonable, we propose to compare the utility of the PET with the utility that can be achieved if measurement errors are present. This utility is already accepted by DSOs. As measurement error, in line with previous work [NI20], we use measurement-dependent Gaussian noise with $\sigma = 0.01$. Specifically, let $\mathcal{G}$ be the Gaussian distribution. Then, the resulting active and reactive power measurement containing measurement errors are given by $(1 + \mathcal{G}(0,\sigma)) \cdot \mathsf{P}(t)$ and $(1 + \mathcal{G}(0,\sigma)) \cdot \mathsf{Q}(t)$.

6.2.3. Selection of Study Data

In this section, we select the grid topology and measurement data for our study. For each of them, we first state requirements we impose on the data to be used in our study based on the previous sections. Then, we state our selection. A natural overall requirement is that the grid topology and measurements match, meaning that a data set containing only measurements or only a grid topology is not appropriate.

Grid Topology We derive the following requirements on a low-voltage grid topology used in the study. First, we need the grid topology of at least one feeder of a low-voltage grid. A single feeder is also sufficient, because the results of the load-flow analysis are independent for each feeder. Second, to ensure the validity of our results, there should exist a successfully validated digital representation. Third, ideally, it should be a real-world grid to support the validity of our results. Considering these requirements, as grid topology, we use a feeder of a real-world grid topology from the Danish DSO Thy Mors Energi[2] (TME). Several studies before [IC19; Nai+21] use it as a reference grid model as well. The grid contains 25 customer connection boxes. Four of them correspond to industrial customers (e.g., a farm), the others correspond to residential customers. Additionally, the feeder contains one Trafo and 10 junction boxes.

Measurements We derive the following requirements on measurements used in the study. First, to be able to calculate utility metrics, a fully-measured measurement scenario (i.e., all measurands are measured) is needed. Second, as indicated in previous work on differential privacy for measurement data [EE17], measurement streams of one day are needed and also sufficient. Third, ideally, violations should be present in order to study the relationship between privacy requirements and violations in the voltage analysis. Considering these requirements, we use the to quarter-hourly values aggregated measurements from the undervoltage trace of 25 hours described in [Nai+21]. It features a fully-measured measurement scenario (i.e., all electrical parameters are measured) and contains undervoltage violations. The measurements are simulated with the hardware-in-the-loop simulator OPAL-RT[3], and were also used in previous studies [Nai+21]. For details regarding the simulation process, we refer to [Nai+21; IC19].

6.3. Study Design – Privacy

In this case study, we aim to protect the data of each residential customer c, meaning that we consider local differential privacy [Cor+18]. Nevertheless, the streams we consider are in line with the general stream model illustrated in Figure 4.1. Specifically, “individuals” $i \in I$ belong to generators and consumers, like appliances. The set of “activities” $\mathcal{A} = \{\mathsf{P}, \mathsf{Q}\}$ consists of the active and reactive power consumed (positive value) or generated (negative value). The query Q is the sum over active as well as reactive power. That is, $Q(D_t) = (\mathsf{P}(c,t), \mathsf{Q}(c,t))$.

[2] www.thymors.dk

[3] www.opal-rt.com

Algorithm 5 PET: Uniform Mechanism at timestamp t

```
function Uniform_t($\mathsf{P}(t), \mathsf{Q}(t), \epsilon, w, \Delta^{\mathsf{P}}, \Delta^{\mathsf{Q}}, \alpha^{\mathsf{P}}, \alpha^{\mathsf{Q}}$)
    $\epsilon^{\mathsf{P}} \longleftarrow \alpha_{\mathsf{P}} \cdot \epsilon$
    $\epsilon^{\mathsf{Q}} \longleftarrow \alpha_{\mathsf{Q}} \cdot \epsilon$
    $\lambda^{\mathsf{P}} \longleftarrow \lambda^{\mathsf{P}}(\Delta^{\mathsf{P}}, w, \alpha^{\mathsf{P}}, \epsilon) = \frac{\Delta^{\mathsf{P}} \cdot w}{\epsilon \cdot \alpha^{\mathsf{P}}}$
    $\lambda^{\mathsf{Q}} \longleftarrow \lambda^{\mathsf{Q}}(\Delta^{\mathsf{Q}}, w, \alpha^{\mathsf{Q}}, \epsilon) = \frac{\Delta^{\mathsf{Q}} \cdot w}{\epsilon \cdot \alpha^{\mathsf{Q}}}$
    return Perturbation($\mathsf{P}(t), \mathsf{Q}(t), \lambda^{\mathsf{P}}, \lambda^{\mathsf{Q}}$)
end function
function Perturbation($\mathsf{P}(t), \mathsf{Q}(t), \lambda^{\mathsf{P}}, \lambda^{\mathsf{Q}}$)
    $\mathsf{P}^*(t) \longleftarrow \mathsf{P}^*(t) + \mathrm{Lap}(\lambda^{\mathsf{P}})$        ▷ Laplace mechanism
    $\mathsf{Q}^*(t) \longleftarrow \mathsf{Q}^*(t) + \mathrm{Lap}(\lambda^{\mathsf{Q}})$
    return $\mathsf{P}^*(t), \mathsf{Q}^*(t)$
end function
```

Below, we first select the w-event mechanism used in our study based on Chapter 5. Second, we identify realistic privacy requirements from literature corresponding to appliance usages, that we use in our study.

6.3.1. Selection of Mechanism

In Chapter 5, we identified a couple of mechanisms satisfying w-event DP. For this study, we leverage the Uniform mechanism. Subsequently, we justify this selection and state details on the mechanism specific for measurement streams.

For grid monitoring purpose, it is essential to keep the seasonality of the stream. According to Chapter 5, this suggests using the Uniform mechanism. This mechanism has the additional advantage that multiple privacy requirements result in the same noise scale, allowing to investigate the utility with respect to an infinite number of privacy requirements by one PET run. The details of the Uniform mechanism used are stated in Algorithm 5. Compared to the Uniform mechanism stated in Chapter 5, it uses additional parameters and sanitizes two correlated query results. Specifically, at each time stamp t, it sanitizes the power measurements $\mathsf{P}(t)$ and reactive $\mathsf{Q}(t)$ of a customer according to given, time-invariant, privacy requirements. In addition to the typical privacy requirements ϵ and w, in the current use case, we have four additional parameters. These are the power shares Δ^{P} and Δ^{Q}, as well as the noise splitting parameters $\alpha^{\mathsf{P}}, \alpha^{\mathsf{Q}}$ satisfying $\alpha^{\mathsf{P}} + \alpha^{\mathsf{Q}} = 1$. The noise splitting parameters split the available budget ϵ between active and reactive power. This is required because active and reactive power are correlated [KOV15]. Note that in this case study, we do not apply truncate filtering. The rational is that active and reactive power can both be negative. For instance, active power is negative in case generation dominates.

6.3.2. Assessing Privacy – Privacy Requirements

To limit the number of experiments, we aim to perform experiments for different noise scales. To this end, we focus on noise scales representing appliance usages from residential customers. Consequently, we sanitize only the measurements from the residential customers. In the remainder, we select the requirements and then state the resulting noise scales used in our study.

6.3.2.1. Selection of Privacy Requirements

Below, we select the privacy requirements. Table 6.4 provides an overview of the requirements we discuss. In compliance with related work, we keep $\epsilon \in (0, 1.0]$.

Selection of Window Size w We focus on window sizes that correspond to typical appliance usage cycles. As stated in Table 6.3, we consider individual appliance usages, as well as combinations. An example for an individual appliance usage is the usage of the washing machine. According to [Got+11], it usually runs for 2 hours and 15 minutes. Considering that our measurement streams feature an interval length of 15 minutes, this corresponds to $w = \frac{2h15m}{15m} = 7$. We consider additionally combinations of appliances, because appliance usages are typically correlated. An example is that the dryer usually runs after the washing machine. To cover this, we have to select the sum of the cycle duration of both appliances. For instance, if the washing machine runs 7 time stamps, and the dryer 9 time stamps, we have to select $w = 16$. If two appliances run in parallel, we have to select the maximum of both cycle durations. For compliance with related work, we integrate the respective window size for ensuring event-level ($w = 1$) and user-level ($w = \infty$) DP in our study (see Table 6.3). For user-level DP, that is not implementable for infinite streams, we consider a so-called flattened variant in which we interpret the stream as a finite time series. Then, we choose $w = p = 100$, which is the number of time stamps of our measurement stream prefix.

Selection of Shares Δ^{P} **and** Δ^{Q} Related work uses shares according to the classical local setting that protects the existence of all producer and consumer. To calculate these shares, we rely on the highest difference in active and reactive power of two residential customers at one time stamp that can occur in our measurement trace. Specifically, for $\mathcal{E} \in \{\mathsf{P}, \mathsf{Q}\}$, the share [ÁC11] is given by

$$\Delta^{\mathcal{E}} = | \max_{c \in \mathrm{CCB}_{\mathrm{res}}, t} \mathcal{E}(t, c) - \min_{c \in \mathrm{CCB}_{\mathrm{res}}, t} \mathcal{E}(t, c)|. \tag{6.1}$$

For our study data introduced in Section 6.2.3, these shares are given by $\Delta^{\mathsf{P}} = 4.85$ kW and $\Delta^{\mathsf{Q}} = 3.37$ kVar. With respect to appliance usages, the shares must be selected in line with the power usually consumed by the appliances or appliance combination as stated in Table 6.3. Since we are not aware of any publication stating the reactive power consumed by appliances, we consider privacy requirements with respect to appliance usages only in combination with measurement scenario P only, in which reactive power is not given.

Selection of Splitting Parameters $\alpha^{\mathsf{P}}, \alpha^{\mathsf{Q}}$ In line with related work [Kel+14], in the measurement scenario PQ, we use $\alpha^{\mathsf{P}} = \alpha^{\mathsf{Q}} = 0.5$. In the measurement scenario P only, in which we do not need to hide Q, we use the full budget for sanitizing active power. This means that we use $\alpha^{\mathsf{P}} = 1$.

Table 6.3.: Overview of shares and window sizes.

Requirement	Δ^{P}	Δ^{Q}	w
Measurement Scenario PQ			
user-level DP	Eq. 6.1 ($\mathcal{E} = \mathsf{P}$)	Eq. 6.1 ($\mathcal{E} = \mathsf{Q}$)	p
w-event DP	Eq. 6.1 ($\mathcal{E} = \mathsf{P}$)	Eq. 6.1 ($\mathcal{E} = \mathsf{Q}$)	[2,p)
event-level DP	Eq. 6.1 ($\mathcal{E} = \mathsf{P}$)	Eq. 6.1 ($\mathcal{E} = \mathsf{Q}$)	1
Measurement Scenario P only			
Individual appliance i	power_share(a)	-	$\frac{\text{cycle\_duration}(i)}{15 \text{ min.}}$
i_1 followed by i_2	$\max_{i \in \{i_1, i_2\}} \{\text{power\_share}(i)\}$	-	$\sum_{i \in \{i_1, i_2\}} \frac{\text{cycle\_duration}(i)}{15 \text{ min.}}$
i_1 parallel to i_2	$\sum_{i \in \{i_1, i_2\}} \text{power\_share}(i)$	-	$\max_{i \in \{i_1, i_2\}} \{\frac{\text{cycle\_duration}(i)}{15 \text{ min.}}\}$

Table 6.4.: Privacy requirements and resulting minimum and maximum noise scales.

Name	Measrmt.	Requirement	ϵ	α^{P}	α^{Q}	λ^{P}		λ^{Q}	
	Scenario			min	max	min	max		
user	PQ	user-level DP	[0.1, 1.0]	0.5	0.5	970	9.700	674	6,740
event	PQ	event-level DP	[0.1, 1.0]	0.5	0.5	9.70	97	6.74	67.40
app	P only	indiv. appliances	[0.1, 1.0]	1.0	0.0	0.14^a	980^b	-	-
app'	P only	app reduced	[0.1, 1.0]	1.0	0.0	0.14	9.7^c	-	-

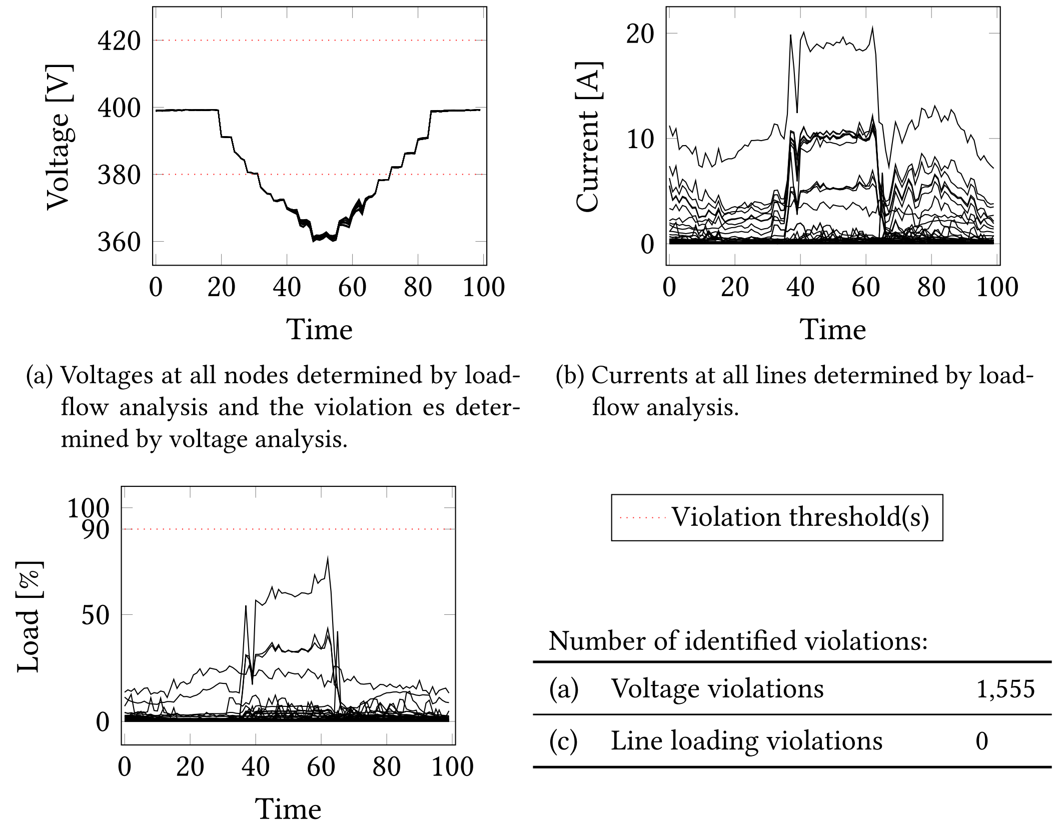
[a] Refrigerator cycle with cycle_duration= 15 min. ($w = 1$) [Got+11], power_share=140 W ($\Delta^{\mathsf{P}} = 0.14$ kW) [Got+11] and $\epsilon = 1.0$.

[b] Space heating with cycle_duration= 210 min. ($w = 14$) [Got+11], power_share=7,000 W ($\Delta^{\mathsf{P}} = 7$ kW) [Got+11] and $\epsilon = 0.1$.

[c] Maximum noise scale from event-level DP.

6.3.2.2. Resulting Noise Scales

Table 6.4 states the resulting minimum and maximum noise scales per measurement scenario. For the measurement scenario P only, we only consider hiding individual appliance usages explicitly, as the maximum noise scale in this case is already similar to the minimum one for achieving user-level DP measurement scenario PQ. In case we do not achieve reasonable utility for user-level nor event-level DP, we consider reduced noise scales, limiting the maximum noise scale in scenario P only to the minimum noise scale needed to achieve event-level DP in scenario PQ. Note that we cover combinations of appliances implicitly, since the noise scales lie in the range between user-level privacy and consideration of individual appliances.



(a) Voltages at all nodes determined by load-flow analysis and the violation es determined by voltage analysis.

(b) Currents at all lines determined by load-flow analysis.

(c) Line loads at all lines determined by line-loading analysis.

Number of identified violations:

(a)	Voltage violations	1,555
(c)	Line loading violations	0

(d) Sums of violations identified by voltage and line loading analysis.

Figure 6.3.: Grid monitoring results in the ground truth measurement scenario PQ. In (a), each curve represents one node. In (b) and (c), each curve represents one line.

6.4. Results

In this section, we present and discuss the results of our study regarding our two research questions. Before that, we present grid monitoring results in the ground truth scenario to give an intuition on grid monitoring results in general. Second, we answer our first research question (RQ1) by relating the utility metrics of different types identified in Section 6.2.2 to each other. Finally, we state the results with respect to our second research question (RQ2) using the appropriate utility metrics identified. Note, as our experiments depend on random numbers, in line with the benchmark requirements proposed in Chapter 5, we execute each experiment multiple times, and report the average errors. As preliminary experiments revealed that the average error converges after 10 runs, we stick to this number of runs.

6.4.1. Illustration of Grid Monitoring Results

Below, we present grid monitoring results in the measurement scenario PQ to give an intuition on grid monitoring results in general. The grid monitoring results in scenario

Table 6.5.: Overview of utility in all experiments. Error ranges relate to parameter variations. Non-integer violation numbers relate to reported mean errors over 10 experiment repetitions.

Scenario	$\mathrm{Er}_{\infty}^{(\mathsf{V},N\setminus\{\mathrm{Trafo}\})}$	$\mathrm{Er}_{L1}^{(\mathsf{hasVoltVio},N)}$	$\mathrm{Er}_{\infty}^{(\mathsf{I},E)}$	$\mathrm{Er}_{\infty}^{(\mathsf{load},E)}$	$\mathrm{Er}_{L1}^{(\mathsf{hasLoadVio},E)}$
Measurement Scenario with P and Q measured					
$\mathrm{SC}^{\mathsf{PQ}}$	———————————— Ground Truth ————————————				
$\mathrm{SC}_{\sigma=0.01}^{\mathsf{PQ}}$	0.013 V	0.2	0.13 A	0.24 %P	0
$\mathrm{SC}_{\mathrm{event}}^{\mathsf{PQ}}$	[25; 17·10^3] V	[299; 1,756]	[113; 9 ·10^6] A	[201; 7 ·10^6] %P	[1; 3]·10^3
Measurement Scenario with P measured only					
$\mathrm{SC}^{\mathsf{P\ only}}$	0.23 V	13	2.64 A	1.55 %P	0
$\mathrm{SC}_{\sigma=0.01}^{\mathsf{P\ only}}$	0.24 V	4.8	4.0 A	3.1 %P	0
$\mathrm{SC}_{\mathrm{app'}}^{\mathsf{P\ only}}$	[0.4; 24.4] V	[17; 236.9]	[3.3; 95.4] A	[1; 245.5] %P	[0; 751]

serve as ground truth for all subsequent experiments. Additionally, we state the errors in measurement scenario P only compared to scenario PQ, and give an intuition on the sensitivity of the metrics. They serve as a lower bound on the errors we can achieve with a PET in scenario P only.

Figure 6.3 shows the results of the load-flow analysis as well as the subsequent analyses. These are the voltages and currents calculated by the load-flow analysis, as well as the resulting line loadings, and voltage respectively line loading violations determined by subsequent analyses. We observe that the voltages at all nodes are between 360 and 400 V, which is expected. Additionally, we see the voltage drop that was provoked during the simulation of the undervoltage trace. In total, we have 1,555 undervoltage violations: All 25 customer connections and 10 junction boxes have at either 44 or 45 time stamps voltages below 380 V. The line currents are between close to zero and 23 A. The highest currents occur for the lines near the Trafo when the voltages in the nodes drop. This is expected, as the measurements are simulated by an online modification of the *secondary* voltages at the transformer [Iov+21]. Consequently, the loads of the lines increase during the voltage drop as well. However, the line load violation limit (90%) is never exceeded, meaning that no line loading violations are present. Using pseudo-measurements for reactive power as in scenario P only reduces utility. Below, beside the pure error numbers, we give a first intuition on the high sensitivity of the subsequent analyses error metrics. Table 6.5, Line $\mathrm{SC}^{\mathsf{P\ only}}$, shows the errors with respect to all defined metrics in the measurement scenario P only. We observe a small voltage $\mathrm{Er}_{\infty}^{(\mathsf{V},N\setminus\{\mathrm{Trafo}\})}$ and current $\mathrm{Er}_{\infty}^{(I,E)}$ error. Additionally, we observe an error in the number of voltage violations, but not in the line loading violations. The rationale for the former is that, as Figure 6.3 (a) reveals, many voltage values are close to the undervoltage violation threshold. Consequently, even a small error in the voltages results in a difference in the number of violations. Inversely, a high current error is required to achieve a difference in the number of loading violations.

6.4.2. (RQ1) Relations of Utility Metrics

The DSO is interested in the results of the subsequent analyses. However, in literature, the utility of mechanisms with respect to sanitization error are known. This raises Question (RQ1) asking how the sanitization, or the error of load-flow analysis as an intermediate step, relate to the subsequent analysis error. Considering the increasing number of subsequent analyses that exist, using the sanitization error or load-flow analysis error would be preferred if appropriate. Consequently, as illustrated in Figure 6.2, we subsequently relate these three types of utility metrics to each other. Ideally, they have a pairwise linear relation. In this case, the metrics are would be interchangeable, and all are appropriate to measure the utility of grid monitoring. To this end, we consider measurement scenario P only, because we have a higher variety of privacy requirements identified than for PQ.

The key results indicate the following. First, the relations between the metric types are highly non-linear. Second, the subsequent analysis errors are more measurement data dependent than the load-flow analysis errors. Consequently, we propose to use the load-flow analysis error in future work.

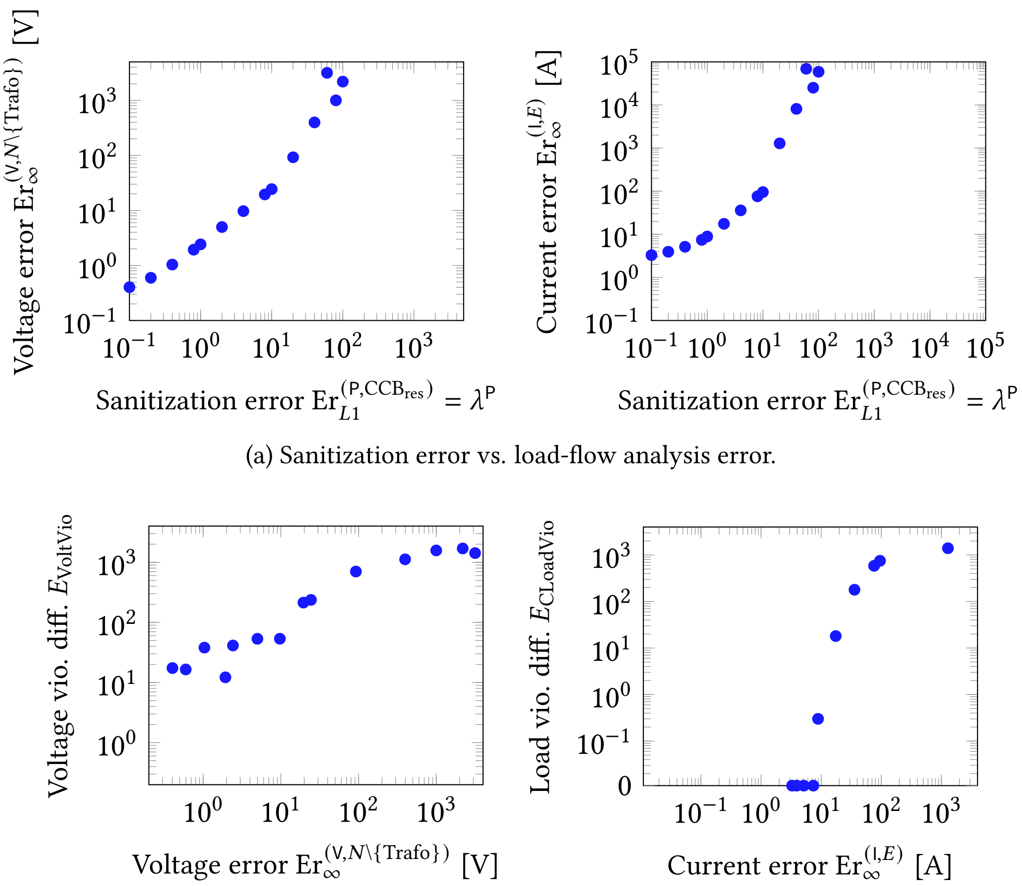
6.4.2.1. Sanitization Error vs. Load-flow Analysis Error

Below, we relate the sanitization error to both error metrics used to measure the load-flow analysis error, namely, the voltage and the current error. For the Uniform mechanism, it holds that the sanitization error with respect to active power converges towards λ^{P} [Kel+14]. Consequently, we consider λ^{P} as the sanitization error. Figure 6.4 (a) reveals that the load-flow analysis errors increase, as the sanitization error increases. However, not in a linear way. Both, the voltage and current error increase faster than the sanitization error. This means that the higher the sanitization error is, the less meaningful is it to assess the utility of grid monitoring. This applies especially to the current error increasing even faster than the voltage error. We consequently propose to use not only the sanitization error to assess the utility of newly proposed w-event DP mechanisms, but to use an analysis-specific metric in addition.

6.4.2.2. Load-flow Analysis Error vs. Subsequent Analysis Error

Knowing that the sanitization error is not appropriate, it remains the question whether the load-flow analysis error is the appropriate, or whether subsequent analysis metrics should be used. The reason is that subsequent analyses process the outputs of the load-flow analysis before the results are useful for the DSO.

Consequently, we now discuss the relationship of the voltage and current error (output of load-flow analysis) on the difference in the voltage and line loading violations (output of subsequent analyses). They are illustrated in Figure 6.4 (b). Generally, we observe that the relationship is again non-linear. In contrast to Figure 6.4 (a), the relationship is even more complex. Specifically, regarding the voltage violations, for voltage errors < 10, the subsequent analyses errors change only slightly. The reason is that for many time stamps and nodes, the voltages are next to the violation limit (see Figure 6.3). Regarding the line loading violations, we observe that even for a current error of 7 A, the violations are still correctly identified. The reason is that the loads in the undervoltage trace are far below



(a) Sanitization error vs. load-flow analysis error.

(b) Load-flow analysis error vs. subsequent analysis error.

Figure 6.4.: Relations between error metrics of different types.

the violation thresholds (see Figure 6.3). This indicates that errors of subsequent analyses are highly sensitive with respect to measurements and chosen thresholds. This imposes, however, the question of how useful they are for DSO-independent utility assessments at all. As a consequence, we propose to use load-flow analysis error metrics as a compromise.

6.4.3. (RQ2) Privacy-Utility Trade-Off

Subsequently, we first assess reasonable utility by determining the utility if measurement errors are present. Second, we compare this utility with the utility achieved by the PET. To this end, we focus, but not limit ourselves, to load-flow analysis utility metrics, as the previous section suggests. Table 6.5 gives an overview of the results discussed in this section. The scenarios are notated with $\mathrm{SC}_{\mathrm{Noise}}^{\mathrm{Measurement\ Scenario}}$. The superscript states the measurement scenario $\in \{\mathsf{PQ}, \mathsf{P\ only}\}$. The subscript states which noise is introduced into the measurements, if any. In this context, $\sigma = 0.01$ stands for noise relating to measurement errors. Additionally, for noise resulting from a PET, the names are in line with Table 6.4. Note that both dimensions, i.e., measurement scenario and introduced noise, are orthogonal to each other. All errors are computed by comparing the analysis results with the ground truth $\mathsf{SC}^{\mathsf{PQ}}$. Key outcome is that it is hard to achieve reasonable utility while keeping reasonable privacy.

6.4.3.1. Utility in the Presence of Measurement Errors

In this experiment, for each measurement scenario, namely, PQ and P only, we determine the utility if we inject measurement errors. We only use this utility to give an intuition on the utility achieved by a PET in the remainder. The utility in scenario P only is stated in Table 6.5, Lines $\mathsf{SC}^{\mathsf{PQ}}_{\sigma=0.01}$ and $\mathsf{SC}^{\mathsf{P\ only}}_{\sigma=0.01}$. For the measurement scenario PQ, we observe that the measurement errors cause only a small error in voltages and currents. Additionally, they do not affect the number of line loading violation. Specifically, the voltage, current and loading errors are a magnitude smaller than the errors in $\mathsf{SC}^{\mathsf{P\ only}}$. For the measurement scenario P only, we compare the error in $\mathsf{SC}^{\mathsf{P\ only}}$ and $\mathsf{SC}^{\mathsf{P\ only}}_{\sigma=0.01}$, since the former is a lower bound of the latter. It reveals that the measurement errors have only a marginal impact on the voltage error, but nearly double the current and loading error. Interestingly, the difference in the number of voltage violation error *decreases*, which is unexpected. However, the reason is that the voltages calculated deviate more upwards to the ground truth, because there are more imbalances in the measurements than in $\mathsf{SC}^{\mathsf{P\ only}}$. Consequently, less undervoltage violations are present.

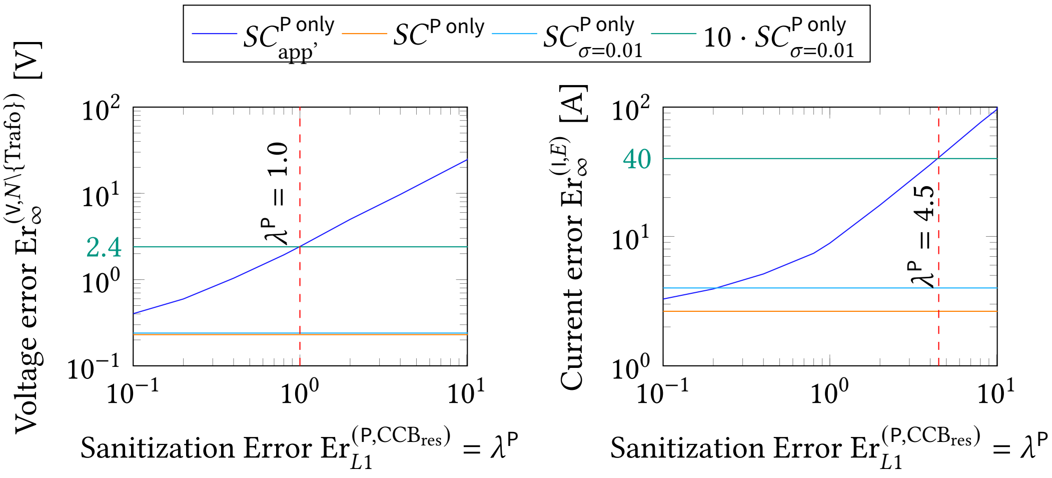
6.4.3.2. Intuition on the Utility of the PET

Subsequently, we compare the utility with the utility that can be achieved if measurement errors are present.

Measurement Scenario PQ For measurement scenario PQ, we first considered event-level differential privacy, since we expect higher utility than from a user-level differentially private PET. The noise scales used in our experiment are compliant with Table 6.4, Line 2. In Table 6.5, the Line $\mathsf{SC}^{\mathsf{PQ}}_{\mathrm{event}}$ shows the resulting errors. The lower numbers apply to $\epsilon = 1.0$, and the higher numbers to $\epsilon = 0.1$. We observe that already for $\epsilon = 1.0$ inducing the lowest privacy guarantee, the voltage and current errors are three orders of magnitudes higher than for the measurement error scenario $\mathsf{SC}^{\mathsf{PQ}}_{\sigma=0.01}$. Additionally, we observe *over*-voltage violations, that are not in line with the undervoltage trace (not visible in the table). As a result, we assess the errors are too high to achieve reasonable utility in this scenario.

Measurement Scenario P only The maximum noise scale Table 6.4 for hiding individual appliances is with $\lambda^{\mathsf{P}} = 980$ two orders of magnitudes larger than the minimum noise scale for event-level DP, that already does not yield reasonable utility. Consequently, in our experiments, we consider the reduced setting and limit the upper bound to $\lambda^{\mathsf{P}} = 9.7$, hiding the total power at one time stamp only.

To investigate whether the PET achieves reasonable utility, we now compare the error in $\mathsf{SC}^{\mathsf{P\ only}}_{\mathrm{app}}$, with the ones in $\mathsf{SC}^{\mathsf{P\ only}}_{\sigma=0.01}$. Figure 6.5 shows both. We observe that *all* considered privacy requirements yield a higher voltage error than in $\mathsf{SC}^{\mathsf{P\ only}}_{\sigma=0.01}$ by far. This means that the usage of a PET yields – even for low privacy requirements – worse utility than the utility resulting from measurement errors. However, if a higher error is acceptable



(a) Voltage error for varying noise scales.

(b) Current error for varying noise scales.

Figure 6.5.: Utility for varying noise scales.

for the DSO, the figures are useful to derive the achievable privacy requirements for a predefined error value, and vice versa. For example, as illustrated in Figure 6.5 (a), if a voltage or current error that is one order of magnitude higher than the errors in $\text{SC}^{\text{P only}}_{\sigma=0.01}$ is still acceptable, $\lambda^{\text{P}} = \min\{1.0, 4.5\} = 1.0$ is the maximum possible noise scale, that in turn corresponds to, e.g., the privacy requirement "protecting one refrigerator cycle with $\epsilon = 0.14$". To sum up, the results suggest that it is hard to achieve reasonable privacy and utility, because the utility with respect to *all* considered utility metrics for weak privacy requirements is already low.

6.5. Summary

In this chapter, we study the utility of differentially private grid monitoring. Specifically, we ask (1) which utility metrics are appropriate and (2) and how utility of a PET relates to utility under measurement errors. To this end, we identify candidates for utility metrics for all three steps of differentially private grid monitoring. To define reasonable privacy, we use privacy requirements relating to appliance usages given in literature. Based on these definitions, we perform a case study on a real-world grid and realistic measurements. With respect to the first question, we observe that the utility of grid monitoring decreases faster than the sanitization error, that is frequently used in related work on differentially privacy as utility metric, suggests. With respect to the second question, the study indicates that already under weak privacy requirements, the utility is worse than under measurement errors.

7. Swellfish Privacy: Exploiting Time-Dependent Privacy Requirements

The results in the previous chapters indicate that w-event mechanism do not provide reasonable utility. Consequently, in this chapter, we ask how to tune utility beyond incremental w-event mechanism design. The running example is the monitoring of the active power consumption in a local area.[1] Towards this, we observe that w-event DP is not customizable enough to consider the versatile nature of every-day human life, limiting the achievable data utility. Put simply, streams have a time domain, and privacy requirements of data owners may change frequently over time [MVW09]. We call this *time-dependent relevance.* However, today it is not possible to add new or to delete no longer required patterns from being protected. Instead, upon mechanism initialization time, one needs to estimate the worst combination of patterns to protect. This has two fundamental drawbacks. First, one needs to hope that one indeed found the worst case. Second, one accepts that the mechanism adds high noise to the query result at every time stamp, even if the worst case is currently not relevant. We illustrate the potential for improving utility with the following examples.

Example 1 (Time-Dependent Relevance)**.** *Suppose that a smart meter records the power consumption[2] every minute, and that one aims to monitor the `Sum` of power consumed in a city. Now think of an data owner with two privacy requirements: First, he wants to hide whether he took a shower in the morning. Second, he wants to hide whether he cooked at lunch time or had (unhealthy) food delivered. He does not aim to hide anything else.*

In the example, the data owner aims to hide two different appliance usages, i.e., shares of his power consumption, during two different time periods of the day. We call such time periods *relevance intervals*, and an appliance usage together with a relevance interval a *privacy requirement.* Time-dependent relevance of privacy requirements has two effects, *time-variant share* (TeS) and *number of affected time stamps* (TeNAT). They allow to tune data utility beyond designing new mechanisms for existing privacy definitions.

Example 2 (TeS: Time-Variant Share)**.** *The amount of power consumed to warm up water for showering, and thus to be hidden in the morning, is larger than the one needed for cooking at lunch time.*

[1] The remainder of this chapter bases on the article Christine Tex, Martin Schäler, and Klemens Böhm. "Swellfish privacy: Supporting time-dependent relevance for continuous differential privacy". In: *Information Systems* 109 (2022), p. 102079. Compared to the article, the sections have been shortened to be less repetitive, contain minor corrections, as well as formatting and notation changes to be in line with the format and structure of this thesis.

[2] In this chapter, with *power consumption*, we refer to positive active power. We assume that residential customers do not have generators.

Example 3 (TeNAT: Time-Variant Number of Affected Time Stamps)**.** *Taking a shower lasts 10 minutes at best, while cooking takes, say, at least an hour.*

Example 2 reveals that the share of the power consumption to be hidden is smaller at lunch time than in the morning. Furthermore, there is even nothing to hide in the evening and afternoon. This is the *time-variant share* effect. Example 3 indicates that there are fewer power-consumption values monitored that are affected by the respective appliance usage in the morning than at lunch time. This is the *time-variant number of affected time stamps* effect.

To use w-event DP, as shown in Section 6, the data administrators have to instantiate these two parameters with the worst share and number of affected results over all time stamps. This means that w-event DP cannot exploit time-dependent relevance by design.

Contributions and Outline In Section 7.1, we introduce fundamentals and related work. After that, first, in Section 7.2, we define Swellfish privacy, including the concepts of stream policies and policy collections. A stream policy enables an data owner to specify a privacy requirement. A policy collection allows to dynamically add or remove several privacy requirements, possibly with different relevance intervals, from being protected. We define Swellfish privacy by using these concepts, including an iterative definition of neighboring streams to allow for concurrent privacy requirements. Additionally, we prove that Swellfish privacy is a generalization of Blowfish privacy [HMD14] to the streaming setting. Second, in Section 7.3, to design Swellfish-private mechanisms exploiting the effects, we propose two tools. Each one corresponds to one effect. The first tool, the TeS sensitivity, allows designing mechanisms that protect the current relevant share of the query result. The second one, the TeNAT composition theorem, allows designing mechanisms that protect the current number of affected time stamps, that are generally not consecutive in time. With these tools, we adjust existing, and design new mechanisms featuring different sampling and budget allocation techniques. Third, in Section 7.4, we design and perform a realistic case study from the domain of power consumption monitoring. We evaluate the utility of exploiting each effect in isolation and compare our mechanisms to the state-of-the-art. The study reveals that exploiting the effects improves data utility up to three orders of magnitude, and that our mechanisms feature the best utility. Additionally, we show how to estimate the strength of the effects for arbitrary use cases. Finally, we provide a summary of this chapter.

7.1. Specific Fundamentals and Related Work

In this section, we introduce fundamentals specific to this chapter. To this end, first, we introduce and illustrate time-variant privacy requirements in multiple use cases. Second, we briefly introduce policy-based privacy definitions for static databases, that Swellfish privacy generalizes, and state why they cannot exploit time-dependent relevance of privacy requirements. Third, we state the latter for further related privacy definitions based on differential privacy.

7.1.1. Typical Privacy Requirements in Different Use Cases

We use three use cases, namely, physical activity [YH10; SWC17], power consumption [Nai+21; Eib+18; Tex+18], and location monitoring [FX14; Wan+19]. In each of them, one continuously monitors query results over a stream $S = (D_1, D_2, .., D_t, ..)$. Here, each $D_t[i]$ is a use-case specific event consisting of states of activities $a_1, .., a_k$ that individual i (i.e., a data owner) performed (see Figure 4.1). However, use cases differ in whether individuals can perform concurrent activities.

Location Monitoring As an individual is typically only at one location at a time, in this use case, the event $D_t[i]$ is a label stating at which location $\mathrm{loc}_x \in \{a_1, .., a_k\}$ individual i is at time t. One typically monitors histograms over S stating how many individuals are at which location at each time stamp. A typical privacy requirement in this use case is to hide whether an individual performed one or another trajectory $[\mathrm{loc}_x, \mathrm{loc}_y, .., \mathrm{loc}_z]$ during a specific time interval. For instance, whether one went jogging in the city park or to the bar, on Friday evening.

Physical Activity Monitoring In this use case, the event $D_t[i]$ is a vector stating which physical activities $\{a_1, .., a_k\}$, like *running* or *eating*, individual i performs at time t (see Figure 4.1). As generally, in contrast to the first use case, one individual can perform several physical activities at a time, like *sitting* and *eating*, it records a vector rather than only activity labels. However, it is, e.g., not possible to sit and run at the same time. One typically monitors histograms over S stating how many individuals perform which activity at each time stamp. In this use case, a typical privacy requirement is hiding whether one or another sequence of vectors was performed during a specific time interval. For instance, whether one sat and ate (cake), or went for a walk, in the afternoon.

Power Consumption Monitoring In this use case, the event $D_t[i]$ is a real number stating how much power an individual household i consumed at time t. The power consumed at time t is the sum of the power consumed by the appliances $\{a_1, .., a_k\}$ that run at time t. Generally, more than one appliance runs at a time. An application typically monitors sums over S stating the total power consumed, e.g., in a city, at each time stamp. In this use case, a typical privacy requirement is to hide whether an individual used specific appliances during a specific time interval (see Chapter 6). For instance, whether he cooked by using the stove and oven at lunch time.

7.1.2. Policy-Driven Privacy

Since DP often is too strong causing low data utility, policy-driven privacy definitions [KM14; HMD14] for static databases (i.e., not for streams) are proposed. However, as the time dimension is missing, they do not feature time-dependent relevance. Policy-driven privacy definitions generalize DP, allowing individuals to specify their privacy requirements with one policy. To define Swellfish privacy, one of its building blocks relies on the same idea, but features multiple policies associated with time intervals to account for time-dependent relevance. To prepare this, we now review discriminative secret pairs [KM14; HMD14] and privacy policies [HMD14]. Further, we introduce Blowfish privacy [HMD14], which is generalized by Swellfish privacy.

7.1.2.1. Discriminative Secret Pairs

A *secret* is a statement of arbitrary nature on the values, e.g., events, in a database, which can or cannot be true for a specific database. A *discriminative secret pair* (s, s') is a tuple of two secrets s and s' that are mutually exclusive, i.e., cannot both be true. It describes properties of databases an adversary should not be able to distinguish between. An example is (”Christine consumes 1kW at time t”, ”Christine consumes 5kW at time t”). For more examples, we refer to [HMD14; KM14]. Particularly relevant for our case study are sets of discriminative secret pairs defined by a distance measure and threshold [HMD14]: If the domain of the database is associated with a distance measure d, such as the Manhattan distance, a *distance-based set of discriminative secret pairs* based on threshold θ

$$P^{d,\theta} = \{(s^i(x), s^i(y)) | d(x, y) \leq \theta\}$$

formalizes that an adversary should not be able to distinguish between secrets $s^i(x), s^i(y)$ about data values x, y of individual i that are close to each other. To illustrate, an adversary should, say, not be able to distinguish between events differing by at most θ activities. Or, they should not be able to infer whether individual i used an appliance that consumes $\leq \theta$ kW.

7.1.2.2. Privacy Policies

A set of discriminative secret pairs P and activities $\mathcal{A}$ are two of three inputs of a privacy policy $\phi = (\mathcal{A}, C, P)$. The third input C is a set of database constraints serving as auxiliary knowledge (see Section 7.1.1).

7.1.2.3. Blowfish Privacy

Blowfish privacy is satisfied if an adversary cannot distinguish between query results computed on two databases that are neighboring with respect to a privacy policy, see Definition 7.1.1. In consequence, the policy-specific sensitivity is generally lower than the global one needed in the DP definition. This in turn means that the mechanism needs less noise to achieve the desired privacy level.

Definition 7.1.1 (Neighboring databases w.r.t. a policy [HMD14]). *Let $\phi = (\mathcal{A}, C, P)$ be a policy and D, D' two databases. Further, let $\mathcal{T}(D, D') \subseteq P$ be the set of discriminative secret pairs (s, s') so that secret s is true in D, and secret s' is true in D', and $\Delta(D, D') = D_1 \backslash D_2 \cup D_2 \backslash D_1$. Then D and D' are neighbors, i.e., $D, D' \in \mathcal{N}(\phi)$, if*

1. *$\mathcal{T}(D, D') \neq \emptyset$*
2. *they both fulfill the constraints in C,*
3. *there is no database D'' fulfilling C so that*
   - *$\mathcal{T}(D, D'') \subset \mathcal{T}(D, D')$, or*
   - *$\mathcal{T}(D, D'') = \mathcal{T}(D, D')$ and $\Delta(D'', D) \subset \Delta(D', D)$.*

The intuition of Definition 7.1.1 is that Blowfish privacy holds, if an adversary cannot distinguish between query results computed on two databases that differ in at least a single, but *arbitrary*, discriminative secret pair (s, s') (item 1.). For instance, in the database D, the secret s is true, but s' is not. Furthermore, in D' the opposite holds. In addition, the definition demands that all constraints are satisfied in D and D' (item 2.). The latter has no consequences for the location monitoring use case, as one can only be in one location at the same time. By contrast, it affects the activity monitoring use case. In this use case, there may be constraints as follows: in case secret s is true, a third secret s'' is true as well. For example, if one is running, i.e, $s = true$, secret s'' indicating whether the current speed is larger than 0 is true as well. This increases the sensitivity, which in turn increases the required noise. Thus, the definition additionally requests that the set of secrets D and D' differ in is *minimal* (item 3).

7.1.3. Further Related Work

To tune utility, related work either bounds the share in the static setting, or the number of affected time stamps. Using Swellfish privacy, we combine both in a time-variant manner.

Bounding Share Bounding the share has been done before in the static setting only. The smoothed sensitivity [NRS07] is a smooth upper bound on the local sensitivity. We show that Swellfish privacy is a generalization of Blowfish [HMD14] to streams. Following results from [HMD14], in the absence of constraints, this also holds for Pufferfish privacy [KM14; KBB15]. [Cao+20] focuses on releasing single trajectories under policies. Metric-based privacy [Cha+13] features distance-based policies. However, all these approaches are defined for the static setting and do not take time-variance into account.

Bounding Number of Affected Time Stamps Bounding the number of time stamps one has to protect has been done before in the streaming setting. However, event-level DP [Dwo+10] hides only one event, and w-event DP [Kel+14] hides all possible patterns anywhere in the stream. Both definitions are time-invariant. Much work exists on designing new w-event mechanisms. The latest approaches exploit special features of the streams, such as small query results [Wan+19]. Many of them use sampling [Wan+19; Li+15a] and filtering [Wan+19; Che+17] techniques. We can show that Swellfish privacy inherits post-processing immunity from the DP definition, and therefore filtering techniques can be used. However, they are orthogonal to exploiting time-variance.

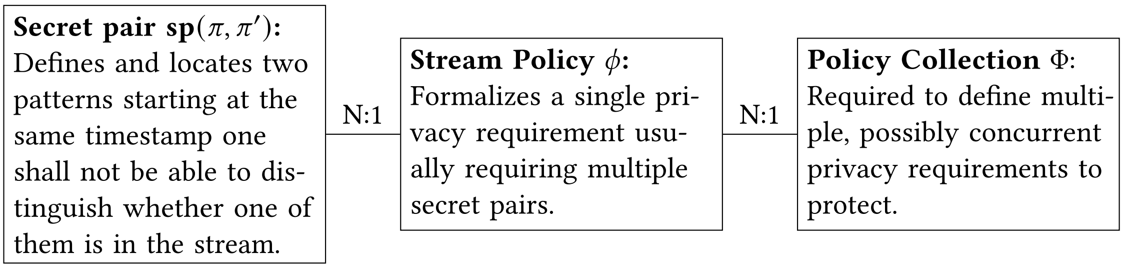
7.2. Swellfish Privacy

In this section, we propose Swellfish privacy. It is a privacy definition for differentially private continuous monitoring of infinite streams that takes time-dependent relevance into account. We use the power consumption use case as a running example, as we use it in our case study as well.

This section is structured as follows: In Section 7.2.1, we first propose our notion allowing individuals to specify multiple privacy requirements. In Section 7.2.2, we define

Table 7.1.: Additional notation used in Chapter 7.

Notation	Meaning
Secret pair $sp_j(\pi, \pi') = (s_j(\pi), s_j(\pi'))$	two secrets starting at time j one should not be able to distinguish between
Stream policy $\phi = (\mathcal{A}, C, J, P)$	formalizes one privacy requirement featuring relevance interval J
Set of secret pairs P	contains one secret pair $sp_j(\pi_q, \pi'_q)$ for each equal-length pattern pair (π_q, π'_q) and possible start time stamp j in J
Policy collection $\Phi = \{\phi_1, .., \phi_{\|\Phi\|}\}$	set of stream policies, formalizes multiple privacy requirements



Secret pair sp(π, π'): Defines and locates two patterns starting at the same timestamp one shall not be able to distinguish whether one of them is in the stream.

Stream Policy ϕ: Formalizes a single privacy requirement usually requiring multiple secret pairs.

Policy Collection Φ: Required to define multiple, possibly concurrent privacy requirements to protect.

Figure 7.1.: Relationship between secret pairs, stream policies, and policy collections. 'N:1' is a many-to-one relationship.

neighboring stream prefixes, such that the definition copes with concurrent privacy requirements. Section 7.2.3 contains our definition of Swellfish privacy. In Section 7.2.4, we show that Swellfish privacy generalizes Blowfish privacy. Table 7.1 summarizes the notation we introduce in this section.

7.2.1. Notion for Specifying Privacy Requirements

We now propose two concepts our notion is built on: stream policies and policy collections. First, stream policies are a deployment of general policies [HMD14; KM14] (see Section 7.1.2) for the streaming setting. With them, an individual can formalize a *single* privacy requirement referring to one relevance interval. Second, to allow for *multiple*, possibly concurrent, privacy requirements, we propose policy collections consisting of multiple stream policies (see Figure 7.1). Generally, there are no privacy requirements publicly available that can be used, e.g., in a case study. Therefore, to illustrate our concepts in this section and prepare our case study, we outline how we instantiate these concepts in our case study.

7.2.1.1. Stream Policies

A *stream policy* allows to define a single privacy requirement. To this end, a stream policy consists of pairs of secrets. We adopt the general idea of indistinguishable secrets from

Blowfish, but must additionally consider that secrets have a temporal dimension. Therefore, a secret $s_j^i(\pi)$ states that individual i performs pattern π in the time interval $[j, j+T]$. Here, j is called the *start time*, and $T := |\pi|$ is the length of the pattern. To illustrate, consider the pattern

$$\pi = \begin{bmatrix} a_1 = 1 & a_1 = 0 \\ a_2 = 1 & a_2 = 1 \end{bmatrix}$$

of length $T = 2$. The columns represent the events and the rows the activities configured in the event. Further, consider the example stream in Figure 4.1. There, the secret $s_{j=2}^{i=2}(\pi)$ is true, as the above pattern starts at time $j = 2$. In contrast, the secret $s_1^2(\pi)$ is *not* true. In the remainder, in line with related work [HMD14], we assume that all considered secrets belong to the same individual i, and we omit the superscript. Now, a discriminative stream secret pair $\mathrm{sp}_j(\pi, \pi')$ is a pair of two secrets, referring to different equal-length patterns π, π', but the *same* start time j ensuring that the secrets are mutually exclusive. For readability, in the remainder, *secret pair*, abbreviates *discriminative stream secret pair*.

Definition 7.2.1 (Pattern Pairs and Discriminative Stream Secret Pairs)**.** *A pattern pair (π, π') refers to two equal-length patterns $\pi \neq \pi'$ one should not be able to distinguish between. In addition, a discriminative stream secret pair $sp_j(\pi, \pi') := (s_j(\pi), s_j(\pi'))$ is a tuple of two secrets referring to a specific pattern pair, both starting at time j.*

A single secret pair may not be enough to specify a privacy requirement. Imagine the privacy requirement is to hide whether one walks one of the three trajectories π, π', or π'' at *some* time during J. Formally, this means the following. First, one requests indistinguishability of three pattern pairs, namely (π, π'), (π, π''), and (π', π''). Second, for each of these pattern pairs, one needs a secret pair for *every* start time stamp in J. This leads to Definition 7.2.2.

Definition 7.2.2 (Stream Policy)**.** *A stream policy $\phi = (\mathcal{A}, C, J, P)$ is a four-tuple containing a set of activities $\mathcal{A}$, secret pairs P, a relevance interval J, and deterministic constraints C s.t.*

$$P = \{sp_j(\pi_q, \pi_q') | \; \forall 1 \leq q \leq L \; \forall j \in J^{T_q}\}$$

and all patterns π_q, π_q' having length T_q fulfill C. Here, $J^{T_q} \subseteq J$ is the interval that begins at the same time stamp as J, but ends $T_q - 1$ time stamps earlier.

For numeric streams, like in the power consumption use case, typical sets of secret pairs are distance-based (see Section 7.1.2), and all pattern pairs contained have the same length. Below, in Example 4 we give an intuition on the introduced notation explaining how we generate stream policies for the evaluation in the remainder.

Example 4 (Case Study – Intuition of Stream Policies and their Generation)**.** *Recapitulate, in the power consumption use case one aims at hiding appliance usages. Therefore, each possible appliance usage an individual wants to hide in a certain period J becomes a stream policy. The appliance usage is visible as share of the power consumption in the stream. In this use case, we rely on the simplification of using distance-based secret pairs meaning that we do not need to specify two mutually exclusive secrets, but only their distance threshold θ. For* *instance,* $\phi^1 = (\mathcal{A}, C, J_1 = [3,6], P_{T=2}^{d,\theta=2.2})$ *states that we aim at hiding an appliance usage having an energy consumption of* 2.2*, lasting for two timestamps anywhere in the relevance interval* $J_1 = [3,6]$.

Similarly, as observed in Chapter 6, literature reveals that there are no freely available stream policies. However, we identified in in Chapter 6 the generator [Got+11] simulating realistic usage patterns of appliances in residential households, i.e., potential secrets. Thus, we use this generator to generate stream policies. These generated appliance usages do not necessarily need to occur in the streams. However, in case they do occur, they must not be visible in the query result, i.e., in the aggregated energy consumption.

In the evaluation, the stream policy $\phi = (\mathcal{A}, C, J, P)$ *per appliance usage is generated as follows: The set* $\mathcal{A}$ *of activities is given by the appliances in the household, and we have no constraints (*$C = \emptyset$*). We set* $P = P_T^{d,\theta}$*, where threshold* θ *is the power an appliance consumes, and* T *is the length of the respective appliance usage pattern. The relevance interval* J *is* $4 \cdot T$ *time stamps long and is located around the usage time scheduled by the generator.*

7.2.1.2. Policy Collections

A policy collection is a set of stream policies with possibly different relevance intervals. If relevance intervals of two stream policies overlap, we say that the underlying privacy requirements are *concurrent.*

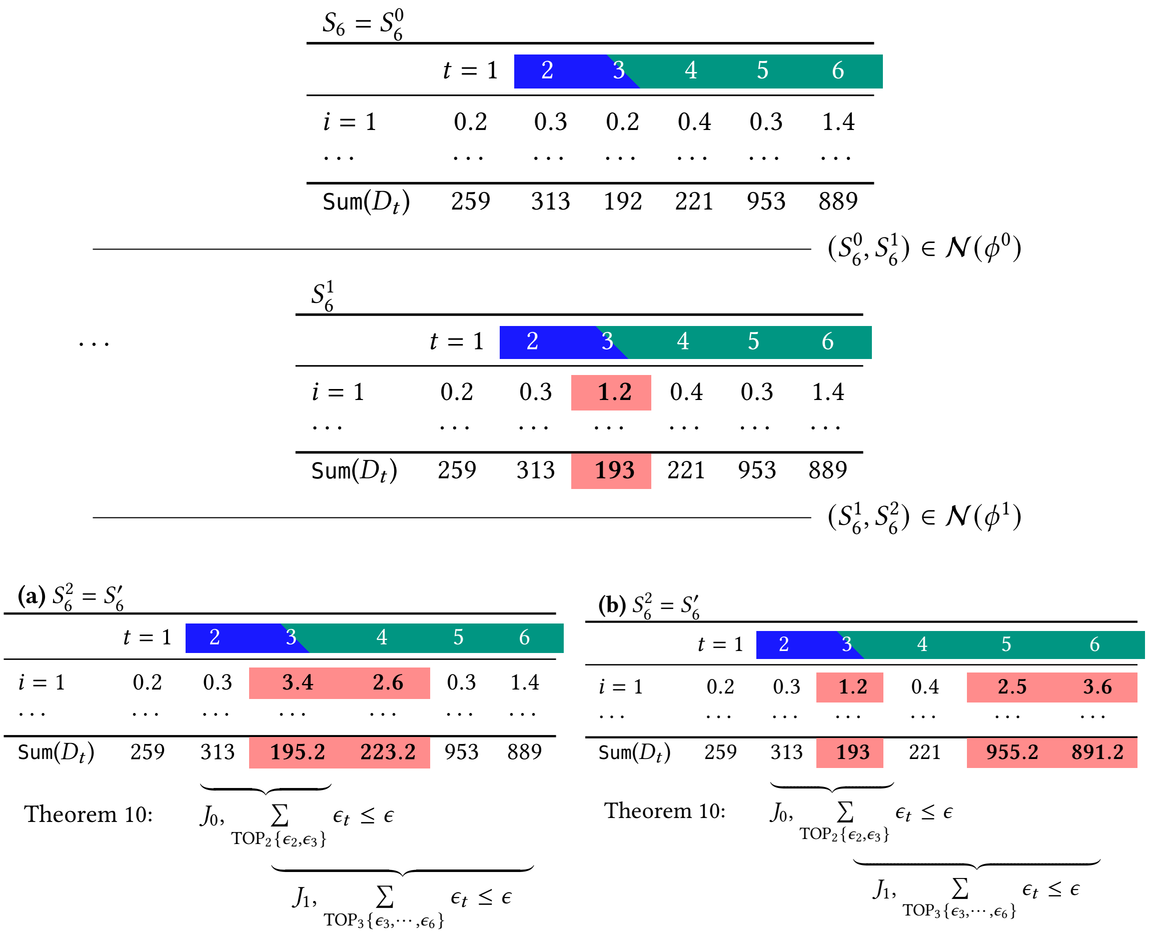
Definition 7.2.3 (Policy Collection)**.** *A policy collection* Φ *is a set of stream policies* $\{\phi_1, \cdots, \phi_{|\Phi|}\}$ *such that the policies contain the same activities* $\mathcal{A}$ *and constraints* C.

The individuals may modify a policy collection, e.g., by adding new stream policies dynamically, as long as the relevance interval of the modified stream policy has not started yet. It is also possible to have the same stream policy w.r.t. different individuals multiple times in a collection. In the latter, our privacy definition behaves like group DP [DR+14].

Example 5 (Case Study - Policy Collection)**.** *By using the generator from [Got+11], we generate several households. For each of them, we generate one policy collection. The more stream policies are contained in a policy collection, the more challenging it is to fulfill the policy collection. In our case study, we aim to challenge our mechanisms. Consequently, in each policy collection,* each *single appliance-usage scheduled by the generator becomes one separate stream policy. Multiple concatenated appliance-usages are considered as one usage. Concurrent used appliances usage (e.g., stove and oven) become concurrent privacy requirements.*

In the remainder, we use the notion of relevant stream policies and time stamps. In context of this notion, it is particularly important to observe that at a time stamp t, or – more generally – during a relevance interval J, not only one, but multiple stream policies can be relevant.

Definition 7.2.4 (Relevance of Stream Policies and Time Stamps)**.** *For a time stamp* t*, a stream policy* ϕ *is* relevant *if* $t \in J$*. Similarly, time stamp* t *is relevant if there is at least one relevant policy at time* t*. For a policy collection* Φ*, the set* $\Phi_t \subseteq \Phi$ *contains all stream policies from* Φ *that are relevant at time* t*.*



$S_6 = S_6^0$

	$t = 1$	2	3	4	5	6
$i = 1$	0.2	0.3	0.2	0.4	0.3	1.4
...	...	...	...	...	...	...
`Sum`(D_t)	259	313	192	221	953	889

$(S_6^0, S_6^1) \in \mathcal{N}(\phi^0)$

...

S_6^1

	$t = 1$	2	3	4	5	6
$i = 1$	0.2	0.3	**1.2**	0.4	0.3	1.4
...	...	...	...	...	...	...
`Sum`(D_t)	259	313	**193**	221	953	889

$(S_6^1, S_6^2) \in \mathcal{N}(\phi^1)$

(a) $S_6^2 = S_6'$

	$t = 1$	2	3	4	5	6
$i = 1$	0.2	0.3	**3.4**	**2.6**	0.3	1.4
...	...	...	...	...	...	...
`Sum`(D_t)	259	313	**195.2**	**223.2**	953	889

Theorem 10: $J_0, \sum_{\mathrm{TOP}_2\{\epsilon_2,\epsilon_3\}} \epsilon_t \leq \epsilon$

$J_1, \sum_{\mathrm{TOP}_3\{\epsilon_3,\cdots,\epsilon_6\}} \epsilon_t \leq \epsilon$

(b) $S_6^2 = S_6'$

	$t = 1$	2	3	4	5	6
$i = 1$	0.2	0.3	**1.2**	0.4	**2.5**	**3.6**
...	...	...	...	...	...	...
`Sum`(D_t)	259	313	**193**	221	**955.2**	**891.2**

Theorem 10: $J_0, \sum_{\mathrm{TOP}_2\{\epsilon_2,\epsilon_3\}} \epsilon_t \leq \epsilon$

$J_1, \sum_{\mathrm{TOP}_3\{\epsilon_3,\cdots,\epsilon_6\}} \epsilon_t \leq \epsilon$

Figure 7.2.: Illustration of our iterative definition of neighboring stream prefixes with respect to the following policy collections $\Phi = \{\phi^0, \phi^1\}$ containing the stream policies $\phi^0 = (\mathcal{A}, C, J_0 = [2,3], P_{T=1}^{d,\theta=1})$ and $\phi^1 = (\mathcal{A}, C, J_1 = [3,6], P_{T=2}^{d,\theta=2.2})$. The blue time stamps correspond to J_0, the green ones to J_1, and red values are values that changed in any iteration step.

7.2.2. Neighboring Stream Prefixes

The secret pairs in a stream policy define a set of stream prefixes containing them. We must provide indistinguishability for every possibly existing pair of neighboring stream prefixes. Basically, two stream prefixes are neighbors if they are neighbors with respect to a *single* policy. This concept is similar to Blowfish's concepts of neighboring static databases. However, with Swellfish privacy, stream prefixes can also be neighbors with respect to *several* policies. In the worst case, neighboring streams differ in one secret pair from *each* stream policy in a policy collection. Generally, not all combinations of secret pairs from different stream policies result in neighboring streams. This is, e.g., the case if two secrets from different policies cannot be true at the same time. But, in case they can be true, we argue that the individual intends to hide the respective secret pair *combination*, i.e., the concatenation of secrets from different policies. Secret pair combinations may result in additional secret pairs to be hidden, as Example 6 illustrates.

Example 6 (Location Monitoring Use Case – Combining Secret Pairs)**.** *Suppose that the secret pairs $sp_{j_1=1}(\pi_1, \pi_1')$ and $sp_{j_2=3}(\pi_2, \pi_2')$ belong to different stream policies. Further, let $\pi_1' = [loc_1, loc_2, loc_3]$ and $\pi_2 = [loc_3, loc_4]$ be given. In these patterns, the last location in π_1' is equal to the first location in π_2. This means that the secrets $s_{j_1=1}(\pi_1')$ and $s_{j_2=3}(\pi_2)$ can be true at the same time. This in turn means that the individual intends to hide the specified secret pairs, as well as additional ones, in particular, $(s_1(\pi_1), s_1(\pi_{combi}))$ with $\pi_{combi} = [loc_1, loc_2, \pi_2']$.*

The following definition of neighboring stream prefixes ensures that there is a pair of neighboring stream prefixes for every possible combination of secret pairs.

Definition 7.2.5 (Neighboring Stream Prefixes w.r.t. a Policy Collection)**.** *Let Φ be a policy collection. The stream prefixes S_p, S_p' are neighbors with respect to Φ, i.e., $S_p, S_p' \in \mathcal{N}(\Phi)$ iff there exists*

1. *a sequence $[\phi^0, .., \phi^k]$ of stream policies from Φ containing each stream policy at most once,*
2. *a sequence $[S_p^0 = S_p, .., S_p^l, .., S_p^{k+1} = S_p']$ of stream prefixes, starting with S_p and ending with S_p',*

such that for all $0 \leq l \leq k$ it holds that $(S_p^l, S_p^{l+1}) \in \mathcal{N}(\phi^l)$.

Example 7 (Neighboring Stream Prefixes without Constraints)**.** *Consider Figure 7.2. It contains a policy collection, together with stream prefixes. Let $C = \emptyset$. Assuming that the stream policies belong to individual $i = 1$, it shows that $S_6, S_6' \in \mathcal{N}(\Phi)$ for Case (a) and (b). For instance, consider Case (a). The stream prefixes S_6^0 and $S_5^1 6$ are neighbors with respect to policy ϕ^0, as they differ by a pattern of length $T = 1$ consuming $\theta = 1$ kW at time stamp $t = 3 \in J_0$. Similarly, S_6^1 and S_6^2 are neighbors with respect to policy ϕ^1.*

Example 7 assumes that the policy collection does not feature any constraints. But recall that policy collections feature constraints, and that stream prefixes are neighbors w.r.t. a single policy if they differ in the minimal number of secret pairs. We now explain the effects of intra- and inter-individual constraints on neighboring databases on the basis of Example 7. First, as the events in the secrets already fulfill the intra-individual constraints, Example 7 would not change if C contains such constraints only. However, second, if C contains intra-individual constraints, neighboring streams may additionally differ in the data of individuals that are different from $i = 1$. Formally, this is because we require subsequent streams to be neighbors w.r.t. a policy as defined in Definition 7.1.1.

7.2.3. Swellfish Privacy

Based on the notion of neighboring stream prefixes, we now define Swellfish privacy.

Definition 7.2.6 ((ϵ, Φ)-Swellfish Privacy)**.** *Let $\mathcal{M}$ be a randomized mechanism that has as input a stream prefix S_p of arbitrary size, Φ be a policy collection, $S_p, S'_p \in \mathcal{N}(\Phi)$ and $\epsilon > 0$. The mechanism $\mathcal{M}$ gives (ϵ, Φ)-Swellfish privacy iff*

$$Pr[\mathcal{M}(S_J) = R] \leq e^{\epsilon} \cdot Pr[\mathcal{M}(S'_J) = R]$$

for every relevance interval J of a stream policy $\phi \in \Phi$.

It says that one has ϵ budget for each relevance interval J. This also holds if multiple policies are relevant at a time stamp $t \in J$. As a result, one must consider two aspects: (1) The share of the query result to be protected, as well as (2) the number of time stamps one distributes the budget between during J depends on *all* policies that are relevant at any time stamp in J.

7.2.4. Generalization of Blowfish Privacy

Before we discuss the relationship to other privacy definitions in the remainder, we highlight that Swellfish privacy generalizes Blowfish privacy.

While Blowfish privacy is defined for the static setting and allows for one policy only, Swellfish privacy is defined for the streaming setting and allows for policy collections. However, reduced to the static setting, Swellfish privacy is equivalent to Blowfish privacy. To prove this, we show (1) that one can implement (ϵ, ϕ_B)-Blowfish privacy with Swellfish privacy, and (2) vice versa. To this end, we consider a definition of Swellfish privacy restricted to a static database given in Definition 7.2.7.

Definition 7.2.7 (Static (ϵ_t, Φ_t)-Swellfish privacy)**.** *Let $\mathcal{M}_t$ be a randomized mechanism that has as input a static database D_t. The mechanism $\mathcal{M}_t$ gives (ϵ_t, Φ_t)-Swellfish privacy iff for all neighboring databases $(D_t, D'_t) \in \mathcal{N}(\Phi_t)$, and all $R \in Range(\mathcal{M})$, the following holds:*

$$Pr[\mathcal{M}_t(D_t) = R] \leq e^{\epsilon} \cdot Pr[\mathcal{M}_t(D'_t)) = R].$$

We now show both directions separately.

Swellfish ⟶ Blowfish Privacy Let $\phi_B = (\mathcal{A}, P, C)$ be a Blowfish policy, and $\phi_S = (\mathcal{A}, P, C, J)$ with $J = [t, t]$ be a stream policy. Both policies contain the same activities, constraints and set of secret pairs. The relevance interval of ϕ_S is restricted to the time stamp t, i.e., to one time stamp only. Then, providing (ϵ_t, Φ_S)-Swellfish privacy for the policy collection $\Phi_S = \{\phi_S\}$ is equivalent to providing (ϵ_t, ϕ_B)-Blowfish privacy, as $\Phi_t = \Phi_S$. As result, a Swellfish private mechanism provides Blowfish privacy.

Blowfish ⟶ Swellfish Privacy Let Φ_S be a policy collection. W.l.o.g., assume that at time t, the two stream policies $\Phi_{S,t} = \{\phi, \phi^*\}$ are relevant. Let $s_t(\pi)$, $s^*_t(\pi^*)$ be any two secrets from ϕ, ϕ^*, that can be simultaneously true at time t. For each such two secrets, we construct a new secret covering both of them, i.e., both are true. Then, we replace the secrets $s_t(\pi)$ and $s^*_t(\pi^*)$ with this new secret in all stream secret pairs. That way, we obtain a new set of secret pairs P_B and respective policy $\phi_B = (\mathcal{A}, P_B, C, J = [t, t])$. For this policy ϕ_B, it holds that Swellfish neighbors $(D_t, D^*_t) \in \mathcal{N}(\Phi_{S,t})$ are also Blowfish neighbors, i.e., $(D_t, D^*_t) \in \mathcal{N}(\phi_B)$.

7.3. Mechanism Design

After defining Swellfish privacy, we now design mechanisms providing it. First, in Section 7.3.1, by formally defining the effects featured in the introduction, we derive two tools, each one allowing to exploit one of the effects. Second, in Section 7.3.2, we propose baseline mechanisms that cannot exploit the effects and discuss how to improve them by using our proposed tools. Third, based on this discussion, in Section 7.3.3, we propose a framework for mechanism design and three concrete mechanisms. In contrast to the baseline, the latter ones exploit the TeS and TeNAT effect.

7.3.1. Definition and Exploitation of the Effects

To allow for the continuous publishing of query results, our aim is to design mechanisms that consist of independent (ϵ_t, Φ_t)-Swellfish private sub-mechanisms $\mathcal{M}_1, .., \mathcal{M}_p$, where $\mathcal{M}_t$ outputs the private query result at time t. In this section, we formally define the TeS and TeNAT effect, and derive tools to exploit these effects. To this end, the first tool to exploit the TeS effect is the TeS sensitivity, used to implement a sub-mechanism. The second tool is the TeNAT composition theorem, used to determine budget ϵ_t for each sub-mechanism.

7.3.1.1. TeS Effect and Sensitivity

To ensure DP, one typically adds Laplace noise to the query result that is proportional to the sensitivity of the query. In the Swellfish framework, the sensitivity depends on the share of the query result to be hidden and therefore tends to be different for each time stamp t. The share for a specific t depends on *all* policies that are relevant at t. Formally, it is defined by the *TeS sensitivity* of a query as given in Definition 7.3.1.

Definition 7.3.1 (TeS Sensitivity). *Let Φ be a policy collection. For a query $Q : D \longrightarrow \mathbb{R}^{dim}$, the TeS sensitivity at time t is*

$$\Delta_Q^t(\Phi) = \max_{(S_p, S'_p) \in \mathcal{N}(\Phi)} ||Q(D_t) - Q(D'_t)||_1.$$

The TeS effect is defined by the fraction of the TeS sensitivity compared to the global sensitivity, as given in Definition 7.3.2.

Example 8 (TeS Sensitivity). *Consider again the policy collection from Figure 7.2. There, at time $t = 3$, in the worst case, neighboring stream prefixes differ by $|\texttt{Sum}(D_t) - \texttt{Sum}(D'_t)| = 3.2$ kW. Namely, both stream policies ϕ^0 and ϕ^1 featuring thresholds 1.0 and 2.2 are relevant. In contrast, at $t = 4$, the TeS sensitivity is 2.2, as only policy ϕ^1 is relevant. At $t = 1$, the TeS sensitivity is 0, as no policy is relevant.*

Definition 7.3.2 (TeS effect). *Given a policy collection, the time-variant share effect is defined by*

$$\varnothing_t \frac{\Delta_Q^t}{\Delta Q},$$

where $\varnothing_t$ is the average over time stamps $t \in [1, p]$.

Theorem 8 states that adding noise that is proportional to the TeS sensitivity at time t yields (ϵ_t, Φ_t)-Swellfish privacy for a specific time stamp t. Thus, one exploits this effect by using the TeS sensitivity for adding noise.

Theorem 8. *Let t be a time stamp. The mechanism $\mathcal{M}_t$ given by*

$$\mathcal{M}_t(D_t) = Q(D_t) + Lap(\frac{\Delta_Q^t(\Phi)}{\epsilon})$$

provides (ϵ_t, Φ_t)-Swellfish privacy at time t.

Proof. As Swellfish privacy is equivalent to Blowfish privacy in the static setting, this holds due to Theorem 5.1 from [HMD14]. □

Generally, calculating the exact TeS sensitivity is challenging, because one iteration step in our definition of neighboring streams can influence previous and subsequent steps in different ways. But we can give an upper bound formalized in Theorem 9. The intuition is that in the worst case, all policy-specific sensitivities of all relevant policies add up. This upper bound is tight if we have policies that do not share secret pairs and do not feature inter-individual constraints. However, e.g., if one has a single policy collection containing two policies containing the same secret pairs, this upper bound is not tight: In this case, the TeS sensitivity is given by the maximum of the policy-specific sensitivities. This holds, e.g., for the power consumption use case.

Theorem 9. *The TeS sensitivity is bounded by*

$$\Delta_Q^t(\Phi) \in [\max_{\phi \in \Phi_t} \Delta_Q^t(\phi), \sum_{\phi \in \Phi_t} \Delta_Q^t(\phi)].$$

Proof. First, neighboring stream prefixes differ in at least one stream policy. Therefore, $\Delta_Q^t(\Phi) \geq \max_{\phi \in \Phi_t} \Delta_Q^t(\phi)$. Second, neighboring stream prefixes differ mostly in all policies from Φ_t. Therefore, $\Delta_Q^t(\Phi) \leq \sum_{\phi \in \Phi_t} \Delta_Q^t(\phi)$. □

7.3.1.2. TeNAT Effect and Composition Theorem

Two stream policies generally feature different patterns. For exploiting the TeNAT effect, it is important that the patterns usually have different length. Intuitively, one needs less noise to hide shorter patterns than longer ones. Put simply, the TeNAT effect measures the fraction of the maximum pattern length of currently relevant policies, compared to the maximum pattern length of all policies, i.e., w. However, it is not as simple as that, as it does not only come down to pattern length, but also to overlaps of relevance intervals. This is captured in our definition of the number of policy-affected time stamps $\delta(J)$ below. Unlike patterns, the policy-affected time stamps are generally not consecutive in J. Consequently, to exploit the TeNAT effect, one needs a composition theorem that takes this into account.

Algorithm 6 Bounding $\delta(J)$

```
function CalcDeltaJ($\Phi, \phi = (A, C, J, P)$)
    $\delta(J) \leftarrow \max\{|\pi| \mid sp_J(\pi, \pi') \in P\}$
    for $\phi^* = (A, C, J^*, P^*) \in \Phi_J$ do
        if $\phi^* \neq \phi$ then
            $o \leftarrow |J \cap J^*|$
            $T_{\max} \leftarrow \max\{|\pi| \mid sp_J(\pi, \pi') \in P^*\}$
            $\delta(J)$ += $\min\{o, T_{\max}\}$
        end if
    end for
    if $\delta(J) > |J|$ then
        $\delta(J) \leftarrow |J|$
    end ifreturn $\delta(J)$
end function
```

Definition 7.3.3 (TeNAT effect). *Given a policy collection Φ, let w be the maximum pattern length of all policies contained in Φ, and $\delta(J)$ the maximum number of policy-affected time stamps. Then, the time-variant number of affected time stamps effect is*

$$\varnothing_t \frac{\max_{\phi \in \Phi_t} \delta_\phi(J)}{w},$$

where $\varnothing_t$ is the average over time stamps $t \in [1, p]$.

Defining and Determining Policy-Affected Time Stamps Given two specific neighboring stream prefixes, we refer to a time stamp t that is affected by any policy as policy-affected time stamp.

Definition 7.3.4 (Policy-Affected Time Stamp). *Let Φ be a policy collection, and $S_p, S'_p \in \mathcal{N}(\Phi)$. A time stamp t is a* policy-affected time stamp *iff $D_t \neq D'_t$.*

For instance, in Figure 7.2, in both cases, time stamp $t = 3$ is a policy-affected time stamp, but $t = 2$ is not. Nevertheless, different neighboring streams may feature different policy-affected time stamps. For instance, $t = 4$ in Case (a) in Figure 7.2 is a policy-affected time stamp, but not in Case (b). Thus, to provide indistinguishability for all possible neighboring stream prefixes, one must protect the *maximum* number of policy-affected time stamps over all neighboring stream prefixes. For a given relevance interval J of a policy Φ, we abbreviate this number with $\delta(J)$, i.e.,

$$\delta(J) = \max_{S_p, S'_p \in \mathcal{N}(\Phi)} |\{t \mid D_t \neq D'_t, t \in J\}|.$$

If during J only one policy is relevant, $\delta(J)$ is the length of the longest pattern in the policy. Otherwise, determining the exact number is challenging, as it varies for different pairs of neighboring streams. However, one can determine an upper bound on $\delta(J)$ as stated in Algorithm 6. The intuition is as follows. First, one initiates $\delta(J)$ by the length T of longest pattern π in J. Second, one needs to consider all overlapping stream policies ϕ^* being relevant for at least at one time stamp $\in J$. For each of them, we add to $\delta(J)$ the share of longest pattern in J', which can range into J. It is given by $\min\{|J \cap J^*|, \max\{|\pi| \mid sp_j(\pi, \pi') \in P^*\}\}$. If multiple policies have the same relevance interval J, $\delta(J)$ is equal regardless which policy is used as input of Algorithm 6, as the policies 'see' each other (Line 3).

Protecting all Possible Policy-Affected Time Stamps With this, we arrive at Theorem 10, the TeNAT composition theorem. The intuition is as follows. For each relevance interval J, one must protect $\delta(J)$ time stamps in this interval. In case many stream policies overlap with each other, the policy-affected time stamps are distributed arbitrarily over J. Therefore, the theorem presumes that they can lie *anywhere* in the interval. However, as one knows that one has to protect only $\delta(J)$ time stamps, it is sufficient to ensure that the sum of the budgets consumed at any $\delta(J)$ time stamps does not exceed ϵ.

Theorem 10 (TeNAT Composition). *Let $\mathcal{M}$ be a mechanism having as input a stream prefix $S_p = (D_1, .., D_p)$, and outputting $R = (r_1, .., r_p)$. Assume that we can decompose $\mathcal{M}$ into p sub-mechanisms $\mathcal{M}_1, .., \mathcal{M}_p$, s.t. $\mathcal{M}_t(D_t) = r_t$, where each $\mathcal{M}_t$ has independent randomness and achieves (ϵ_t, Φ_t)- Swellfish privacy. Then $\mathcal{M}$ satisfies (ϵ, Φ)-Swellfish privacy if*

$$\forall \phi \in \Phi : \sum_{\epsilon_t \in \mathcal{X}} \epsilon_t \leq \epsilon, \tag{7.1}$$

where $\mathcal{X} = \ TOP_{\delta(J)}\{\epsilon_t | t \in J\}$ is the set of the highest $\delta(J)$ budgets spent during J.

Proof. Let S_J and S'_J neighboring snapshots differing in at most $\delta(J)$ time stamps, and $\mathbb{P}_{\delta(J)}(J)$ be the set containing all sets from the power set of a relevance interval J of size $\delta(J)$. As (1) all mechanisms use independent randomness, (2) are (ϵ_t, Φ_t)-Swellfish private, and (3) Eq. 7.1 holds, we have

$$\begin{aligned}
\frac{Pr[\mathcal{M}(S_J) = R]}{Pr[\mathcal{M}(S'_J) = R]} &\overset{(1)}{=} \max_{\mathcal{Y} \in \mathbb{P}_{\delta(J)}(J) \cap [1,p]} \prod_{t \in \mathcal{Y}} \frac{Pr[\mathcal{M}_t(D_t) = r_t]}{Pr[\mathcal{M}_t(D'_t) = r_t]} \\
&\overset{(2)}{\leq} \max_{\mathcal{Y} \in \mathbb{P}_{\delta(J)}(J) \cap [1,p]} \prod_{t \in \mathcal{Y}} e^{\epsilon_t} \\
&= \max_{\mathcal{Y} \in \mathbb{P}_{\delta(J)}(J) \cap [1,p]} \exp(\sum_{t \in \mathcal{Y}} \epsilon_t) \\
&= \exp(\sum_{\epsilon_t \in \mathcal{X}} \epsilon_t) \overset{(3)}{\leq} \exp(\epsilon),
\end{aligned}$$

which proofs the claim. □

7.3.2. Baseline Mechanisms

In this section, we design mechanisms satisfying Swellfish privacy. To this end, we first propose a baseline. Then, we discuss how we can do better than the baseline in terms of utility by exploiting the effects with the tools just introduced.

7.3.2.1. w-Event DP Baseline for Policy Collections without Constraints

Swellfish privacy differs from w-event DP in the specification of privacy policies and constraints (see Section 7.1.3). However, in the absence of constraints, a $(w = \max_{(\mathcal{A},C,J,P)\in\Phi} |J|)$-event private mechanism satisfies Swellfish privacy.

Theorem 11. *Let Φ be a policy collection without constraints, $\epsilon > 0$ and let $\mathcal{M}$ be a w-event DP mechanism using global sensitivity. If $w \geq \max_{(\mathcal{A},C,J,P)\in\Phi} |J|$, then $\mathcal{M}$ provides (ϵ, Φ)-Swellfish privacy.*

Proof. Let $\phi = (\mathcal{A}, C, J, P) \in \Phi$. Then, $|J| \leq |J'| = w$ where J' is the relevance interval that corresponds to

$$(\mathcal{A}, C, J, P) = \operatorname*{arg\,max}_{(\mathcal{A},C,J,P)\in\Phi} |J|.$$

Let S_J, S'_J be sub-streams of the stream prefixes S_p, S'_p, which differ during interval J only. Then, S_p, S'_p are neighboring stream prefixes for $w \geq |J|$. As $\mathcal{M}$ is $w = |J'|$-event differentially private, it follows that

$$\frac{Pr[\mathcal{M}(S_J) = R]}{Pr[\mathcal{M}(S'_J) = R]} = \frac{Pr[\mathcal{M}(S_p) = R]}{Pr[\mathcal{M}(S'_p) = R]} \leq \exp(\epsilon),$$

which proofs the claim. □

7.3.2.2. Improving the Baseline

Due to the TeS and TeNAT effect, we hypothesize that one can have a higher utility than with w-event DP baseline in many cases.

First, regarding the TeS effect, w-event mechanisms scale Laplace noise proportionally to the global sensitivity. Since the TeS sensitivity tends to be smaller, scaling Laplace noise with the TeS sensitivity is expected to improve data utility. In particular, this is the case for time stamps t where no policy is relevant, as one can publish the query results without perturbation there.

To obtain a Swellfish-private mechanism exploiting the TeS effect, one can take a w-event mechanism and replace the global sensitivity with the TeS sensitivity in its PERTURBATION function (see Section 5.1.2). We use such mechanisms in our experimental study to investigate the influence of the TeS effect. However, w-event mechanisms implementing adaptive ISSAMPLINGPOINT and BUDGETALLOCATION functions may presume that the sensitivity is constant over time. For such mechanisms, one has to adapt these functions as well. An example is RescueDP [Wan+16a] whose adaptive ISSAMPLINGPOINT function needs the sensitivity of the *next*, so far unknown sampling point to calculate the sampling point.

Second, regarding the TeNAT effect, the number of time stamps $\delta(J)$ to be protected in a relevance interval J might be different for each interval. So it may be much smaller than w. Therefore, distributing the budget in line with the TeNAT composition theorem that depends on $\delta(J)$ should improve data utility further. As the BUDGETALLOCATION function of a mechanism typically builds on its ISSAMPLINGPOINT function, we now

Algorithm 7 Swellfish Mechanism-Framework

```
function $\mathcal{M}_t(\epsilon_t^s, \epsilon^{op}, \Phi, l)$
    if $|\Phi_t| == 0$ then                                        ▷ no relevant policies
        return $Q(D_t)$
    else
        if isSamplingPoint$(t, D_t, r_l, \epsilon_t^s, \Phi)$ then
            $\epsilon_t^{\text{op}} \leftarrow$budgetAllocation$(t, \epsilon^{\text{op}}, \Phi)$
            $r_t \leftarrow$Perturbation$(\epsilon_t^{\text{op}}, \Delta_t^Q, D_t) = Q(D_t) + \text{Lap}(\frac{\Delta_t^Q}{\epsilon_t^{\text{op}}})$
            $l \leftarrow t$                                        ▷ $\epsilon_t = \epsilon_t^s + \epsilon_t^{\text{op}}$
        else $r_t \leftarrow r_l$                                 ▷ approximation, $\epsilon_t = \epsilon_t^s$
        end if
    end if
    return $r_t$
end function
```

propose Swellfish-private mechanisms featuring typical isSamplingPoint and appropriate budgetAllocation functions.

7.3.3. Swellfish Mechanisms Exploiting the Effects

We propose a general framework that allows designing Swellfish-private mechanisms that exploit the TeS and TeNAT effect, as well as three different mechanisms implementing this framework. They use well-known techniques like dynamic budget allocation and data-adaptive sampling.

7.3.3.1. Swellfish Mechanism-Framework

To exploit the TeS effect, a mechanism can use the TeS sensitivity for noise scaling. To design mechanisms exploiting the TeNAT effect, we propose mechanisms that consist of independent sub-mechanisms structured as in the general Algorithm 7. A sub-mechanism $\mathcal{M}_t$ following this algorithm publishes the unperturbed query results if there are no relevant policies. Otherwise, it decides whether to sample the query results according to its sampling function isSamplingPoint, possibly using some sampling budget ϵ_t^s in case the function is data-dependent. In case the mechanism decides to sample, it perturbs the query results with some output perturbation budget ϵ_t^{op} determined by budgetAllocation. It is subject of the implementation to ensure that the composition theorem is respected. In case the mechanism does not sample, it approximates the results with the last published query result. In the following, we propose three mechanisms implementing different sampling strategies. See Table 7.2 for an overview of how the functions are implemented. We now explain each proposed mechanism and state how it respects the composition theorem.

7.3.3.2. Mechanism Instances

Based on our framework, we propose three different mechanisms.

Table 7.2.: Overview of design of our Swellfish-private mechanisms.

Mechanism	`isSamplingPoint(`t`,` ϵ_t^s`)`	`budgetAllocation()`
UnicornIS	sample at timestamp t iff there does not exist $(\mathcal{A}, C, J, P) \in \Phi_t$ such that J contains a previous sampling point	return ϵ
UnicornPS	return true	see Algorithm 9 with $\epsilon_{\text{op}} = \epsilon$
Unicorn	see Algorithm 8	see Algorithm 9 with $\epsilon_{\text{op}} = 0.5 \cdot \epsilon$

Predefined-Rate Sampling with UnicornIS UnicornIntervalSample (UnicornIS) samples at most once per relevance interval. Specifically, it samples at time t in case there is no policy relevant that features a relevance interval containing any sampling point. It does not need budget for sampling, and thus uses full budget $\epsilon_t^{\text{op}} = \epsilon$ for output perturbation. Therefore, it is Swellfish-private by design. As the mechanism samples infrequently, its error is dominated by the approximation error at non-sampling time stamps.

Permanent Sampling with UnicornPS In case the query results vary significantly over time, sampling more frequently may improve utility. So, we propose the mechanism UnicornPermanentSample (UnicornPS) that samples at every time stamp. That is, isSamplingPoint always returns true, and does not consume any sampling budget. The challenge then is proper output perturbation budget allocation via budgetAllocation.

budgetAllocation-Function The overall aim is to spend in each relevance interval the *entire* budget ϵ without violating the privacy guarantee. Furthermore, we aim at distributing the budget preferably homogeneously over each relevance interval, to achieve constantly high utility over time.

To this end, at each time t with at least one relevant policy, the budgetAllocation function (see Algorithm 9) computes for each policy $\phi \in \Phi_t$ an output perturbation (op) budget $\epsilon_{\phi,t}^{\text{op}}$. Then, it takes the minimum budget over all policies in Φ_t to respect the privacy guarantee.

Computing the remaining budget per policy The computation of the policy-specific budget is an ensemble of two strategies, targeting at the fulfillment of our aim. The first strategy, named *uniform*, allocates the same budget for every time stamp t, similar as the baseline mechanism Uniform identified in Section 5.2.1. To this end, it divides the entire budget ϵ^{op} available for output perturbation for each relevance interval by the number of affected time stamps $\delta(J)$.

To motivate our second strategy, consider again Example 7 and time stamp $t = 3$. There, $\delta(J_0) = 2$, and thus, $\epsilon_{\phi^0,t,s1}^{\text{op}} = \frac{\epsilon^{\text{op}}}{2}$. However, $\delta(J_1) = 3$ and therefore $\epsilon_{\phi^1,t,s1}^{\text{op}} = \frac{\epsilon^{\text{op}}}{3} < \epsilon_{\phi^0,t,s1}^{\text{op}}$. We say ϕ^1 *dominates* ϕ^0 at time t. As result, $\epsilon_t^{\text{op}} = \frac{\epsilon^{\text{op}}}{3}$, if one uses Strategy 1 only. This would, in turn, mean that during J_0, one does not spend the whole budget, which is in contrast to our aim.

To counter this, we propose the second strategy, named *absorb-and-distribute*. The idea is to absorb the remaining budget after J_1 has ended. To this end, one calculates the remaining budget (Line 8). In case there are already $\delta(J)$ release time stamps in J_0, we replace the currently lowest budget with a higher one (Line 10). As a result, $\epsilon^{\mathrm{op}}_{\phi,t,rm}$ contains the remaining budget for this policy, which we distribute uniformly over all remaining time stamps (Line 13).

Data-Adaptive Sampling with Unicorn Sampling every time stamp might be a waste of budget if there are times where the query results vary only slightly. So we propose the mechanism Unicorn that performs data-adaptive sampling. It uses the same BUDGETALLOCATION function as UnicornPS, but implements data-adaptive sampling in ISSAMPLINGPOINT.

Algorithm 8 Dissimilarity-based sampling of Unicorn.

```
function ISSAMPLINGPOINT($t, D_t, r_l, \epsilon^{\mathrm{s}}_t, \Phi$)
    $\lambda_{t,1} \leftarrow \frac{1}{\epsilon^{\mathrm{s}}_t \cdot 0.5} \cdot \frac{\Delta^t_Q(\Phi)}{dim}$        ▷ $dim$ = number of dimensions
    $dis \leftarrow \frac{1}{n}\sum_{i=1}^{n} |r_l[i] - D_t[i]|$
    $dis \leftarrow dis + \mathrm{Lap}(\lambda_{t,1})$        ▷ dissimilarity $dis$ perturbation
    $\lambda_{t,2} \leftarrow \frac{1}{\epsilon^{\mathrm{s}}_t \cdot 0.5} \cdot \frac{\Delta^t_Q(\Phi)}{dim}$        ▷ expected noise for query result perturbation
    if $dis > \lambda_{t,2}$ then        ▷ decide whether to sample
        return true
    else
        return false
    end if
end function
```

In the literature, there are two types of data-adaptive sampling. Since PID-based sampling [FX14; Wan+19] presumes constant sensitivity, we rely on dissimilarity-based sampling [Kel+14]. Here, one samples iff the expected approximation error is higher than the expected perturbation error. To decide this, one determines the dissimilarity between the published value at the last sampling point and the current, un-perturbed, query result. To ensure privacy one must sanitize the dissimilarity computation, costing a fraction of the privacy budget. In line with related work [Kel+14], Unicorn spends half of the budget for the dissimilarity perturbation, as well as for output perturbation. However, different splits of the budget are possible. Below, we state why Unicorn fulfills (ϵ, Φ)-Swellfish privacy.

Theorem 12. *The mechanism UnicornPS and Unicorn fulfill (ϵ, Φ)-Swellfish privacy.*

Proof. Intuitively, the privacy of the Unicorn mechanism holds for the following reason: Given arbitrary $\delta(J)$ time stamps in a relevance interval, at these time stamps, the mechanism spends at most $\frac{\epsilon}{2}$ for dissimilarity perturbation, as well as for output perturbation. As UnicornPS uses the same BUDGETALLOCATION function, the following arguments prove that it spends at most ϵ budget for output perturbation at arbitrary $\delta(J)$ time stamps in a relevance interval. Formally, consider Unicorn, and let J be a relevance interval, and $\mathcal{X} = TOP_{\delta(J)}\{\epsilon_t = \epsilon^s_t + \epsilon^{op}_t | t \in J\}$. We prove that (1) $\sum_{\epsilon_t \in \mathcal{X}} \epsilon^s_t \leq \frac{\epsilon}{2}$ and (2) $\sum_{\epsilon_t \in \mathcal{X}} \epsilon^{op}_t \leq \frac{\epsilon}{2}$. Then, with (1), (2) and Theorem 10, the claim follows. Eq. (1) holds, as $\sum_{\epsilon_t \in \mathcal{X}} \epsilon^s_t \leq \sum_{\epsilon_t \in \mathcal{X}} 0.5 \cdot \frac{\epsilon^t}{\delta(J)} = \frac{\epsilon}{2}$ in the function isSamplingPoint.

Algorithm 9 Budget allocation of UnicornPS ($\epsilon^{\text{op}} = \epsilon$) and Unicorn($\epsilon^{\text{op}} = 0.5 \cdot \epsilon$).

```
1:  function BUDGETALLOCATION($t, \epsilon^{\text{op}}, \Phi$)
2:      $\epsilon_t^{\text{op}} = \infty$
3:      for $\phi \in \Phi_t$ do
4:          // Strategy s1 – uniform
5:          $\epsilon^{\text{op}}_{\phi,t,s1} \leftarrow \frac{\epsilon^{\text{op}}}{\delta(J)}$
6:          // Strategy s2 – absorb-and-distribute
7:          $\mathcal{X} \leftarrow \text{TOP}_{\delta(J)}\{\epsilon_t^{\text{op}} | t \in J\}$
8:          $\epsilon^{\text{op}}_{\phi,t,\text{rm}} \leftarrow \epsilon^{\text{op}} - \sum_{\epsilon^{\text{op}}_{t'} \in \mathcal{X}} \epsilon^{\text{op}}_{t'}$
9:          if no. sampling time stamps in $J \geq \delta(J)$ then
10:             $\epsilon^{\text{op}}_{\phi,t,\text{rm}} - = \min\{\epsilon^{\text{op}} | t' < t, t' \in J\}$
11:         end if
12:         no_rel_ts $\leftarrow J$.end $- t + 1$        ▷ Number of remaining relevant time stamps
13:         $\epsilon^{\text{op}}_{\phi,t,s2} \leftarrow \frac{\epsilon^{\text{op}}_{\text{rm},\phi}}{\text{no\_rel\_ts}}$
14:         $\epsilon^{\text{op}}_{\phi,t} \leftarrow \max\{\epsilon^{\text{op}}_{\phi,t,s1}; \epsilon^{\text{op}}_{\phi,t,s2}\}$
15:     end for
16:     $\epsilon_t^{\text{op}} = \min_{\phi \in \Phi_t}\{\epsilon^{\text{op}}_{\phi,t}\}$
17:     return $\epsilon_t$
18: end function
```

Concerning ϵ_t^{op}, we discern between the following cases: If there is only one relevant policy, and the mechanism samples every time stamp, then the mechanism uses strategy 1 for budget allocation only. In consequence, Eq. (2) holds for the same reasons as Eq. (1). Otherwise, strategy 2 distributes the saved or dominated budget over the remaining time stamps. Here, Line 7 in BUDGETALLOCATION *ensures Eq. (2).* □

7.4. Results

In this section, we firstly evaluate Swellfish privacy experimentally by means of a case study and secondly generalize the case study results for arbitrary use cases. The study contains two parts. In the intrinsic part, we examine the strength of the TeS and TeNAT effect for different parameters (e.g., stream and privacy budget). In the extrinsic evaluation, we compare the utility of our Swellfish-private mechanisms to the ones of w-event competitors from literature, including the state-of-the-art.

7.4.1. Intrinsic Evaluation

We now comprehensively examine the strength of the TeS and TeNAT effect concerning relevant parameters. The parameter space contains four dimensions: data stream, policy collection, mechanism, and privacy budget. In the first experiment, we examine how combinations of the first three dimensions influence the two effects. Then, we generalize these results for different values of ϵ. Before presenting the results, we introduce how we instantiate the dimensions indicating validity and comprehensiveness of our results. Note, as all mechanisms use random noise, in line with Chapter 5, we repeat every experiment for each dimension value combination 100 times to eliminate statistical bias. To measure utility, we use the mean relative error (MRE) defined in Section 4.2.3. Because the streams feature only query results greater than zero, we use sanity bound $\gamma = 0$. In addition, to allow for easy reproducibility, all data (including privacy policies) and implementations are available online[3].

7.4.1.1. Dimension Parameters

We instantiate the dimensions as follows.

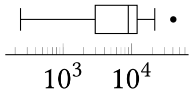
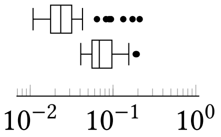
Data Streams We use 20 data streams from the GEFCom 2012 data set [HPF14]. It is one of largest energy data sets available often used for such evaluations [Eib+18]. It compromises the hourly power consumption values of the entire US over 4.5 years ($p = 152,277$), partitioned into 20 zones. For each zone, there is one stream. The data set is well suited as the streams have different average query results over time. For instance, zone 4 has the smallest (0.5 MW), while zone 18 has the highest one with 213.57 MW.

Policy Collections To evaluate the influence of policy collections, we rely on data generators creating realistic usage patterns of appliances in residential households [Got+11] as stated in Section 7.2.1. As stated there, we create the worst case of a very privacy affine user aiming at protecting every application usage. In sum, we generate 55 policy collections being a realistic number of households in a neighborhood according to a recent project[4]. In line with related work [HMD14], we assume that there are no inter-individual constraints. Table 7.3 states statistics of the 55 policy collections generated. In the first two rows, it provides the parameters for the w-event competitors. We estimate the global sensitivity by the sum of consumption if all appliances are running at the same time stamp. It is equal for all policy collections. The window size w is different per policy collection. We therefore state the distribution as a boxplot. It varies between 240 and 40,553. The latter two rows provide a data-independent indicator that show that the effects should be visible in the study. The indicator for TeS Effect is the fraction of temporal sensitivity and global sensitivity of a policy collection. The given boxplot indicates that, for a given time stamp, the temporal sensitivity is up to two orders of magnitudes smaller than the global one, indicating that TeS effect might be visible in the study results. The indicator for TeNAT effect is the fraction of number of true neighboring time stamps $\delta(J)$ to window length w, i.e., the number of neighboring time stamps assumed in w-event DP. We determine an average value over time, to respect the length of the relevance intervals either. The indicator shows that $\delta(J)$ is by orders of magnitude smaller than w.

[3] `https://github.com/chryenix/swellfish-public`

[4] www.esquire-project.de, www.net2dg.eu

Table 7.3.: Statistics of generated policy collections ($\varnothing_t$ = average over time).

Parameters w-event competitors		
Parameter		Value
Global sensitivity	ΔQ	27.57
Window size per policy collection Φ	w	
Estimation of MAE Improvement of Uniform by Effect		
Effect	Calculation	Value
TeS effect	$\varnothing_t \frac{\Delta_Q^t}{\Delta Q}$	
TeNAT effect	$\varnothing_t \frac{\max_{\phi \in \Phi_t} \delta(J)}{w}$	

Mechanisms To evaluate the TeS effect, we modify the Perturbation function of existing w-event mechanisms, leaving all other functions identical. This is not possible for every w-event mechanism (see Section 7.3.2.2), meaning that we cannot use any w-event mechanism. Table 7.4 states known mechanisms we can use. These are the mechanisms Sample, Uniform, Budget Distribution (BD) and Budget Allocation (BA) proposed in the original article on w-event DP [Kel+14] featuring different sampling or budget allocation. We use all of them and notate the corresponding mechanism using TeS sensitivity with prefix "TeS".

As the TeNAT composition theorem is not constructive, there is no general rule how to adapt the sampling and budgetAllocation function of a w-event mechanism, such that it exploits the TeNAT effect. To this end, we use w-event mechanisms where only minimal adaptions of one of these functions are needed, such that the mechanism exploits the effect. This is the case for the Uniform and Sample mechanism, as they *either* have a sampling or a budgetAllocation function. To adjust Uniform, we replace its budgetAllocation function with Strategy 1 of UnicornPS. To adjust Sample, we replace its isSamplingPoint function with the one of UnicornIS. We use the prefix "TeNAT" for the adapted mechanisms.

Privacy Budgets To evaluate the effect of different budget sizes, we conduct one experiment using the following values: 0.01, 0.1, 1.0, and 10. If not stated otherwise, we use $\epsilon = 1.0$.

7.4.1.2. Experiment 1: Influence of Stream, Policy Collection, and Mechanism

We now state and discuss the results of this experiment. In each of the following figures, the data stream dimension is reflected by the different zones on the x-axis. The zones are ordered by increasing average query result over time. The mechanisms are encoded using different colors and we use box plots displaying the distribution of the utility among

Table 7.4.: Mechanisms compared in the intrinsic evaluation.

TeS Effect				
	w-event	vs.	Swellfish	Implementation
🟦	Uniform [Kel+14]	vs.	TeSUniform	use Δ_t^Q
🟥	Sample [Kel+14]	vs.	TeSSample	
🟩	BD [Kel+14]	vs.	TeSBD	
🟨	BA [Kel+14]	vs.	TeSBA	
TeNAT Effect				
	w-event	vs.	Swellfish	Implementation
🟦	Uniform [Kel+14]	vs.	TeNATUniform	UnicornPS using ΔQ + Strategy 1 only
🟥	Sample [Kel+14]	vs.	TeNATSample	UnicornIS using ΔQ

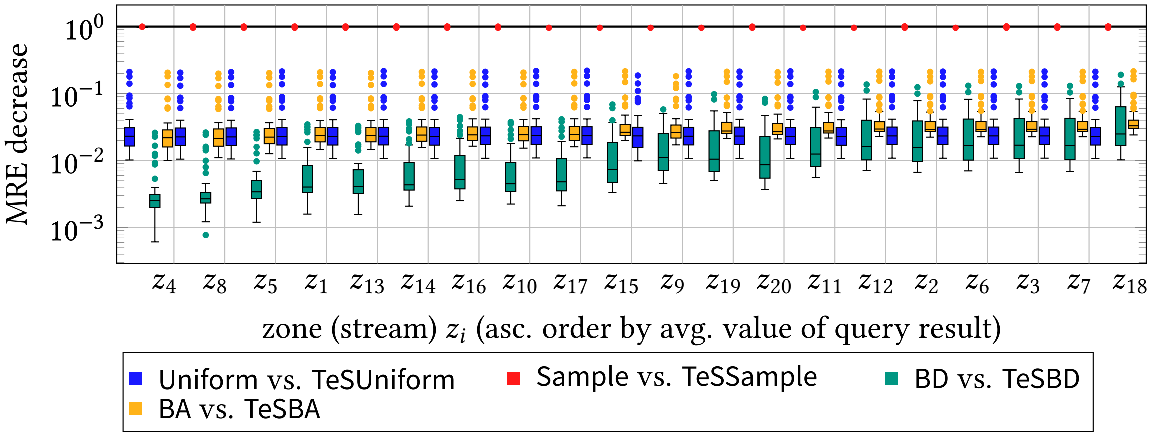


Figure 7.3.: Results of the intrinsic evaluation – TeS effect.

the privacy collections. On the y-axis, we show the increase of utility measured by the decrease of the mean relative error (MRE) , i.e., $\frac{\text{MRE Swellfish}}{\text{MRE } w\text{−event}}$. A value of 1 means that both mechanisms provide the same data utility. A smaller value indicates a decrease of the MRE caused by exploiting the corresponding effect.

TeS Effect Figure 7.3 illustrates the results regarding the TeS effect. We observe three key findings from the experiments: (1) General large error decrease, (2) no decrease if a fixed sampling rate is used, and (3) highest decrease for BD. Regarding (1), for all mechanisms except for Sample, the MRE decreases by one to three orders of magnitude. The decrease tends to become smaller for zones with higher power consumption. This is expected and is in line with previous results [Eib+18]. Regarding (2), exploiting the TeS effect does not influence the utility of Sample. The rationale is as follows. Generally, mechanisms profit from the TeS effect at sampling time stamps only. The mechanism Sample samples every w-th time stamp only. In consequence, its MRE is dominated by

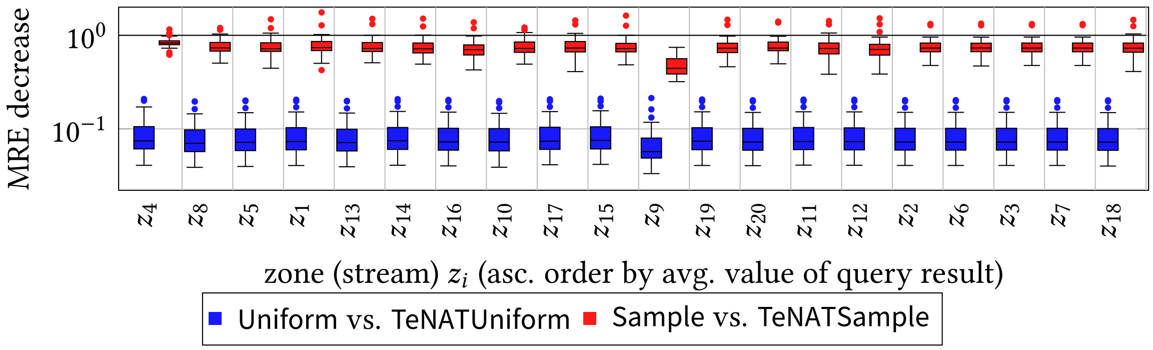


Figure 7.4.: Results of the intrinsic evaluation – TeNAT effect.

Table 7.5.: Average MRE over all zones and Φ for different values of ϵ.

	MRE			
mechanism	$\epsilon = 0.01$	$\epsilon = 0.1$	$\epsilon = 1$	$\epsilon = 10$
TeSBD	180.80	18.10	2.09	0.43
TeSBA	112.82	11.33	1.14	0.12
Unicorn	29.07	2.91	0.29	0.03

the approximation error at non-sampling time stamps, which is nearly independent from the sensitivity used. Regarding (3), BD profits most from the TeS effect. Namely, as a result of its budget-allocation strategy, BD samples at the beginning of each window only. As the windows are long, it samples infrequently, meaning that its MRE is also dominated by the approximation error. While TeSBA samples infrequently as well, it additionally publishes true query results in the meantime, if the temporal sensitivity is zero. As the mechanism does not have to approximate these time stamps, this reduces the total approximation error significantly. However, though BA has the same sampling strategy as BD, BA profits much less from the TeS effect. Namely, after sampling, due to its budget-allocation strategy, BA has to skip various time stamps where it must not publish, and thus, cannot publish true query results.

TeNAT Effect Considering Figure 7.4, we first observe an MRE decrease by an order of magnitude for the Uniform mechanism. This is expected, as it samples every time stamp with a reduced noise scale. Second, we observe a small MRE decrease for Sample for most policies. This is because TeNATSample samples more often than Sample, which generally decreases the MRE. However, it samples at most once per relevance interval, which is rare, leading to a small improvement only. Despite this general MRE decrease, there is an increase of MRE in a few cases. The reason is as follows: The sampling time stamps of TeNATSample are different from the ones of Sample. As the sampling strategy of both mechanisms is not data-adaptive, Sample may select the better sampling time stamps by chance, leading to a lower MRE.

7.4.1.3. Experiment 2: Influence of Privacy Budget

We now examine the influence of parameter ϵ using all values $\in \{0.01, 0.1, 1, 10\}$. For the data independent mechanism TeSUniform, TeNATUniform, TeSSample, and TeNATSample, by definition, the average noise and therefor the MRE is proportional to the size of ϵ. The same holds for two of our proposed mechanisms UnicornPS and UnicornIS. This is different for the data-dependent mechanisms. The respective MRE values for a varying privacy budget are stated in Table 7.5. The results reveal that increasing or decreasing the privacy budget ϵ leads to a proportional change in the corresponding MRE as well. As a consequence, we can safely use $\epsilon = 1.0$ in the extrinsic evaluation.

7.4.2. Extrinsic Evaluation

Next, we compare the utility of our proposed mechanisms to the state-of-the-art. To this end, we use the same streams, policy collections, and methodology as in the intrinsic evaluation. As mechanisms, we however compare our proposed mechanisms UnicornIS, UnicornPS and Unicorn to (1) the current state-of-the-art w-event DP mechanism RescueDP [Wan+19], and (2) the best adjustment of the w-event mechanisms used in the intrinsic evaluation, i.e., TeSBD and TeSBA. In addition, to assess the influence of sampling on mechanism utility, we add the mechanism Unicorn* into our study. It is a non-private variant of Unicorn featuring perfect sampling. In other words, it uses the budget only for query result perturbation, i.e., $\epsilon^{\text{op}} = \epsilon$, and does not perturb the dissimilarity value in the function isSamplingPoint.

Figure 7.5 shows the MRE of the competitors (upper plot) and of Swellfish-framework mechanisms (bottom plot). We observe three key findings, which we now discuss: (1) Best utility by mechanisms from Swellfish framework, (2) UnicornIS is best for publishing small query results, and (3) marginal difference in utility of Swellfish framework mechanisms performing dynamic budget allocation. Regarding (1), in general, the competitors from the literature perform worse than our mechanisms except for UnicornIS. This is expected, as they exploit only the TeS (TeSBD, TeSBA) or even none of the effects (RescueDP). Regarding (2), considering our mechanisms only, we observe that UnicornIS performs worst for all zones except for Zone 4, having the smallest average query result. For this zone, it also performs better than the w-event competitors. As, additionally, the MRE of UnicornIS is almost independent from the zone, this mechanism is suitable for continuous publishing query results that are small. Regarding (3), by comparing UnicornPS with Unicorn and its non-private variant Unicorn*, we observe small utility differences. This indicates that almost perfect sampling decisions yield only little utility improvement in the Swellfish-framework.

Summing up, exploiting the TeS and TeNAT effect improves the utility of adjusted existing mechanisms by orders of magnitude and allows to design novel high-utility Swellfish-private mechanisms with even higher utility. The results indicate that, to design a high-utility Swellfish-private mechanism, good budget allocation is more important than the sampling strategy.

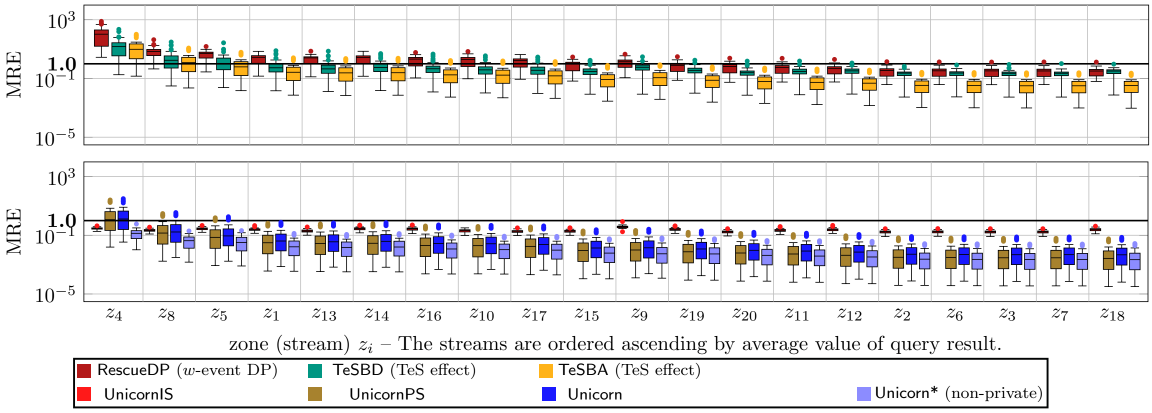


Figure 7.5.: Results of the extrinsic evaluation.

7.4.3. Generalization of Case Study

The results from the case study indicate that – in the power-consumption use case – exploiting the effects improves data utility from one to three orders of magnitude. However, it is natural to ask what improvement to expect for other use cases. Abstractly, for non-sampling mechanisms, we can estimate the mean absolute error improvement of TeS knowing the average TeS sensitivity, i.e., predict the improvement of TeSUniform over Uniform. The mean absolute error (MAE) defined in Section 4.2.3.

Theorem 13. *Given a policy collection Φ, the improvement of the MAE of TeSUniform over Uniform is given by*

$$\frac{\mathit{MAE\ TeSUniform}}{\mathit{MAE\ Uniform}} = \varnothing_t \frac{\Delta_Q^t}{\Delta Q}.$$

Proof. According to its definition, the mechanism Uniform adds Laplace noise with scale $\lambda_t^U = \frac{\Delta Q \cdot \epsilon}{w}$ to the query result at time t. Here, $w = \max_{\phi\in\Phi} \delta(J)$. Similarly, TeSUniform uses scale $\lambda_t^{TSU} = \frac{\Delta_Q^t \cdot \epsilon}{w}$. As the scales equal the expected MAE, we obtain the claim by dividing both scales. □

Similarly, with the average size of the maximum $\delta(J)$ over all time stamps, one can estimate the improvement of the TeNAT effect.

Theorem 14. *Given a policy collection Φ, the improvement of the MAE of TeNATUniform over Uniform is given by*

$$\frac{\mathit{MAE\ TeNATUniform}}{\mathit{MAE\ Uniform}} = \varnothing_t \frac{\max_{\phi\in\Phi_t} \delta(J)}{w},$$

where $w = \max_{\phi\in\Phi} \delta(J)$.

Proof. According to its definition, the mechanism Uniform adds Laplace noise with scale $\lambda_t^U = \frac{\Delta Q \cdot \epsilon}{w}$ to the query result at time t. Similarly, TeNATUniform uses scale $\lambda_t^{TIU} = \frac{\Delta_Q \cdot \epsilon}{\max_{\phi\in\Phi_t}}$. As the scales equal the expected MAE, we obtain the claim by dividing both scales. □

The respective values for the case study are given in Table 7.3. The MAE values we obtained in our study are in line with them. We now give an intuition regarding the expected TeS sensitivity and number of policy-affected time stamps we expect in the location and physical activity monitoring use case. To illustrate, the TeS effect tends to be high if the global sensitivity of the query is high, and if there are few sensitive privacy policies. In the location-monitoring use case, the global sensitivity equals 1, as there are no concurrent activities. The temporal sensitivity is either 0 (no relevant policy) or 1. Thus, one profits from the TeS effect if there are few relevant policies. However, we expect a smaller improvement than in the case study, as the difference between global and TeS sensitivity is smaller. This is different in the physical-activity-monitoring use case, as the global sensitivity is given by the number of activities an individual can perform, and we expect that an individual usually wants to hide only a few activities. Further, the TeNAT effect is high if relevance intervals are – compared to the number of policy-affected time stamps – long. An example is short trajectories to be hidden in long relevance intervals.

7.5. Summary

To tune utility beyond incremental w-event mechanism design, we propose Swellfish privacy for continuously publishing differentially private query results. It allows to dynamically add or to remove time-dependent policies a data owner aims to hide adapting to changing privacy needs. A consequence of the customizability is that Swellfish privacy offers two effects, namely time-variant share and number of affected time stamps. Appropriately exploited, both effects allow tuning data utility by reducing the noise added by privacy mechanisms significantly. Consequently, we show how to exploit these effects in privacy mechanisms by proposing two tools. The first one, namely the TeS sensitivity, allows exploiting the time-variant share effect. The second one, the TeNAT composition theorem, allows exploiting the time-variant number of affected time stamps effect. We prove the significant utility improvement by a realistic case study in the power-consumption use case, as well as theoretic proofs.

Part IV.

Conclusions

8. Summary

The collection and processing of complex data, like structured data and infinite streams, facilitates novel data-driven applications. As many service users do not have the methodological skills to conduct the analysis on their own, they outsource them to services providers. As the service providers are usually not trusted, data administrators tackle these concerns by sanitizing the data with a privacy-enhancing technology (PET). Such a PET is frequently based on an indistinguishably-based privacy definition. However, the sanitization usually implies a loss of application utility. Consequently, data administrators aim for PETs that provide a valuable trade-off between privacy and utility. For indistinguishably-based privacy definitions, depending on the use case, the trade-off is usually achieved by either (1) optimizing utility for given privacy requirements or (2) optimizing privacy for given utility requirements. In this thesis, we study privacy-utility trade-offs for indistinguishably-based privacy definitions for two important groups of applications. Specifically, we investigate (a) which indistinguishability-based privacy and utility requirements are relevant, (b) whether existing PETs solve the trade-off sufficiently, and (c) propose novel PETs extending the state-of-the-art in terms of methodology, as well as achieved privacy or utility substantially. In total, we provide four contributions divided into two parts as follows.

In Part II, that wraps our first thesis contribution, we focus on distance-based data mining applications, such as k-medoids clustering, processing structured data. Additionally, we consider instances of the thesis scope in which data owners equal service users. Therefore, we optimize privacy for given utility requirements. We reveal that to achieve reasonable application results, a fundamental utility requirement is that the pair-wise distances of the data items are preserved upon data sanitization. Since encryption is a well-accepted privacy enhancing technology, we aim for a PET that is based on encryption. To engineer distance-preserving encryption (DPE) schemes used by a PET, we present the DisPE-procedure. It states how to design a DPE scheme for arbitrary data and distance measures. The procedure involves defining and ensuring equivalence notions, which capture a characteristic of data that should be preserved upon encryption. We then instantiate this procedure for SQL query logs, which are a valuable resource for database performance tuning. In this study, we find appropriate DPE schemes for all four prominent SQL query distance measures from literature, and generalize it by the examples of XQuery and of relational data.

In Part III, we focus on monitoring applications for streaming data, such as electricity grid monitoring. Since the processing capabilities of encrypted data are highly limited, we focus on data sanitization according to the w-event differential privacy framework. Additionally, we focus on instances of the thesis scope in which data owners and service users are two different entities. This means that data owners are usually less willing to lower their privacy requirements. Therefore, we focus on optimizing utility for fixed privacy requirements. In the first contribution of this part, we compare state-of-the-art mechanisms that are used to implement a PET with respect to the utility they provide. We reveal limitations of previous studies resulting in requirements on the typical elements of an experimental study. Subsequently, we conduct the so-far largest experimental study that fulfills the requirements. Considering the results of this study, we reveal insights about the strength and weaknesses of mechanisms. From them, we derive takeaways for practitioners (data owners, data administrators) as well as researchers. One insight is the baseline supremacy, saying that one of the two baseline is for every combination of stream and privacy requirement among the mechanisms with the highest utility. A second one is that the sanitization error, that is frequently used to measure utility, is not meaningful for applications that analyze the seasonality of the stream. Consequently, in the second contribution of this part, we conduct a case study from the area of electricity grid monitoring that aims at answering the following two research questions: (1) Which utility metrics are appropriate to measure the utility of grid monitoring? (2) How does the utility of differentially private grid monitoring under reasonable privacy requirements behave? The study reveals that achieving reasonable utility is only possible under weak privacy requirements, and that the utility measured with application-specific utility metrics decrease faster than the sanitization error suggests. Considering this result, as a third contribution of this part, we propose the novel Swellfish privacy framework, that allows to tune utility beyond incremental mechanism design. It is based on the observation that privacy requirements of the data owners are usually time-dependent, while the w-event differential privacy framework assumes them to be constant. Consequently, the Swellfish framework allows data owners to specify fine-granular privacy requirements that may vary over time. We show how to exploit two resulting effects upon mechanism design. By experiments from the area of power consumption monitoring, as well as formal proves, we show that exploiting these effects yields significant utility improvement compared to w-event DP.

In total, our thesis contributes substantially to the research field, and reveals directions for future research.

9. Outlook

In this thesis, we focus on the privacy-utility trade-offs a trusted data administrator has to take when configuring a privacy-enhancing technology (PET). Such a PET sanitizes true data before transmitting it to an untrusted service provider. Understanding these trade-offs is a fundamental prerequisite to use untrusted services serving privacy critical data in practice. Our contributions focus on selected trade-offs, but there are other, still unexplored, privacy-quality trade-offs. In the following, we elaborate on open research questions that are closely related to our contributions.

Part II: Distance-preserving Encryption – Leveraging and Transfer In Chapter 3, we designed for distance-preserving encryption schemes to allow for distance-based mining of encrypted data. In our case study, we engineered such encryption schemes for various SQL query distance measures. There are three directions for future work: (1) Levering exactly the encryption schemes used in our case study to allow for additional applications processing SQL logs. (2) Using the notation for data that is different from SQL logs. (3) Finding similar notions and encryption scheme engineering procedures for applications for which distance-preserving encryption is not the right notion. With respect to (1), while distance-based data mining algorithms are quite common, there exist additional mining algorithms that do not rely on pair-wise distances only [HPK11]. Examples are association-rule mining, classification or regression algorithms. Additionally, there are algorithms specifically designed for SQL logs, like cleaning an SQL from antipatterns [ASB17] and SQL query recommendations [AB21]. In this context, particularly interesting is whether one can leverage the engineered distance-preserving encryption schemes to allow for mining data encrypted data according to other groups. For instance, we know that result equivalence for SQL queries is also useful for association-rule mining over encrypted SQL logs [Ali+11], and token equivalence is appropriate for cleaning SQL antipatterns [ASB17]. Considering points (2) and (3), there is already recent work adopted our notion to speech encryption [KLM21; Kra21; KLM22]. However, considering the wide spectrum of use cases, the question in which use cases our notion imposes interesting novel research questions.

Part III: Utility Metrics In Part II, we measured utility with respect to various metrics. These include the sanitization error, as well as application-specific metrics. Except for the metrics that measure the difference in violations that the grid monitoring systems computes, they all have in common that they consider the query results at each time stamp separately. However, this is usually not the case for typical stream applications, like change detection applications [Agg07]. Such applications analyze the progression of the stream, and rely on stream properties like seasonality. In Chapter 5, we observed, that state-of-the-art w-event mechanisms do not preserve these stream properties. Consequently, we see two open questions for future work. The first question is how to define utility metrics that measure the error in the stream properties. The second one is how to engineer mechanisms that explicitly target at preserving these properties.

Part III: Private Privacy Requirements & Constraints in Swellfish Privacy For policy-based differential privacy frameworks like Blowfish [HMD14], Pufferfish [KM14] and Swellfish privacy (Chapter 7), policy collections are presumed to be public knowledge. However, since they are detailed, individuals may deem their policy collections sensitive information, and no trusted entity is available for the aggregation. To tackle this, one possibility is to focus on the local setting, in which each data owner sanitizes its own data [Cor+18]. Second, generally, not only static databases itself feature constraints, but there also might be dependencies between static databases at different time stamps [Cao+18; ZKL22]. Such constraints may, e.g., affect non-relevant time stamps. We aim at investigating how to consider such constraints in our framework.

Part III: Instantiation of Trade-Off for Streams In Chapter 1, we identified and leveraged the two options to trade privacy and utility: Optimizing privacy given utility requirements, or vice versa. Here, either privacy *or* utility requirements can be defined, but not both. There, a few works that allow requirements on both [LBB15; EF15; SMA21]. They achieve them by solving an optimization problem. However, they are only applicable to static data [LBB15; SMA21], or do not support indistinguishability-based privacy definitions [EF15; SMA21]. An important direction for future work is consequently to engineer respective PETs for streams. One challenge is that the PET does not “see” the complete stream, making it hard to give utility guarantees. Here, probabilistic utility requirements that may additionally depend on a rolling window might be a first approach.

Overall Thesis: Theory and Practice In Chapter 1, we discussed that by law, data owners have the right to decide who has access to their data and for which purposes it is used. However, we still lack of PETs generally accepted by data owners. Therefore, researches, data administrators and service providers do not know where to focus on. For instance, whether it is worth for service providers to focus on services that feature high utility under differential privacy. One issue is the non-awareness of data owners which kind of privacy requirements they have, and which PETs could tackle them in which way. For instance, a recent study [CKR21] indicates that data owners are, in principle, more willing to share their data if a differential privacy mechanism is applied, but we lack descriptions of differential privacy for data owners, that are usually not experts in the research field. Here, educational work is needed.

Bibliography

[AA96] Fevzi Alimoglu and Ethem Alpaydin. “Methods of combining multiple classifiers based on different representations for pen-based handwritten digit recognition”. In: *Proceedings of the 5th Turkish Artificial Intelligence and Artificial Neural Networks Symposium (TAINN)*. Bogazici University Press. 1996.

[AAL14] Shahab Asoodeh, Fady Alajaji, and Tamás Linder. “Notes on information-theoretic privacy”. In: *Proceedings of the 52nd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE. 2014, pp. 1272–1278.

[AB21] Natalia Arzamasova and Klemens Böhm. “Scalable and data-aware SQL query recommendations”. In: *Information Systems* 96 (2021), p. 101646.

[ÁC11] Gergely Ács and Claude Castelluccia. “I have a dream! (differentially private smart metering)”. In: *Proceedings of the 13th International Workshop on Information Hiding (IH)*. Springer. 2011, pp. 118–132.

[Agg05] Charu C Aggarwal. “On k-anonymity and the curse of dimensionality”. In: *Proceedings of the 31st International Conference on Very large Data Bases (VLDB)*. Vol. 5. VLDB Endowment, 2005, pp. 901–909.

[Agg07] Charu C Aggarwal. *Data streams: models and algorithms*. Vol. 31. Springer, 2007.

[Agr+04] Rakesh Agrawal et al. “Order preserving encryption for numeric data”. In: *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*. ACM, 2004, pp. 563–574.

[Akb+10] Javad Akbarnejad et al. “SQL QueRIE recommendations”. In: *Proceedings of the VLDB Endowment (PVLDB)* 3.1-2 (2010), pp. 1597–1600.

[AL83] Dana Angluin and David Lichtenstein. *Provable security of cryptosysterns: A survey*. Tech. rep. TR-288. Yale University, 1983.

[Ali+11] Julien Aligon et al. “Mining preferences from OLAP query logs for proactive personalization”. In: *Proceedings the 15th International Conference on Advances in Databases and Information Systems (ADBIS)*. Springer. 2011, pp. 84–97.

[ALN87] Niv Ahituv, Yeheskel Lapid, and Seev Neumann. “Processing encrypted data”. In: *Communications of the ACM* 30.9 (1987), pp. 777–780.

[AM12] Monika Agrawal and Pradeep Mishra. “A comparative survey on symmetric key encryption techniques”. In: *International Journal on Computer Science and Engineering* 4.5 (2012), p. 877.

[AS00] Rakesh Agrawal and Ramakrishnan Srikant. “Privacy-preserving data mining”. In: *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data.* ACM, 2000, pp. 439–450.

[ASB17] Natalia Arzamasova, Martin Schäler, and Klemens Böhm. “Cleaning antipatterns in an SQL query log”. In: *IEEE Transactions on Knowledge and Data Engineering (TKDE)* 30.3 (2017), pp. 421–434.

[Bar+14] Pedro Barbosa et al. “Lightweight privacy for smart metering data by adding noise”. In: *Proceedings of the 29th Annual ACM Symposium on Applied Computing (SAC).* 2014, pp. 531–538.

[BCO11] Alexandra Boldyreva, Nathan Chenette, and Adam O’Neill. “Order-preserving encryption revisited: Improved security analysis and alternative solutions”. In: *Proceedings of the 31st Annual Cryptology Conference (CRYPTO).* Springer. 2011, pp. 578–595.

[Bel+97] Mihir Bellare et al. “A concrete security treatment of symmetric encryption”. In: *Proceedings 38th Annual Symposium on Foundations of Computer Science (FOCS).* IEEE. 1997, pp. 394–403.

[Bel15] Richard E Bellman. “Adaptive control processes”. In: *Adaptive Control Processes.* Princeton university press, 2015.

[Bel98] Mihir Bellare. “Practice-oriented provable-security”. In: *School organized by the European Educational Forum.* Springer. 1998, pp. 1–15.

[Ben66] Adi Ben-Israel. “A Newton-Raphson method for the solution of systems of equations”. In: *Journal of Mathematical analysis and applications* 15.2 (1966), pp. 243–252.

[BLR13] Avrim Blum, Katrina Ligett, and Aaron Roth. “A learning theory approach to noninteractive database privacy”. In: *Journal of the ACM (JACM)* 60.2 (2013), pp. 1–25.

[Bol+09] Alexandra Boldyreva et al. “Order-preserving symmetric encryption”. In: *Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT).* Springer. 2009, pp. 224–241.

[Bon+04] Dan Boneh et al. “Public key encryption with keyword search”. In: *Proceedings of the 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT).* Springer. 2004, pp. 506–522.

[Bre+99] Markus M Breunig et al. “Optics-of: Identifying local outliers”. In: *Proceedings of the Third European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD).* Springer. 1999, pp. 262–270.

[Bun21] Bundesamt für Sicherheit in der Informationstechnik. *Technische Richtlinie BSI TR-03109-1 Anforderungen an die Interoperabilität der Kommunikationseinheit eines intelligenten Messsystems.* Tech. rep. Bonn: BSI, 2021.

[BW18] Borja Balle and Yu-Xiang Wang. “Improving the gaussian mechanism for differential privacy: Analytical calibration and optimal denoising”. In: *Proceedings of the 35th International Conference on Machine Learning (ICML).* Proceedings of Machine Learning Research. 2018, pp. 394–403.

[Cao+18] Yang Cao et al. “Quantifying differential privacy in continuous data release under temporal correlations”. In: *IEEE Transactions on Knowledge and Data Engineering (TKDE)* 31.7 (2018), pp. 1281–1295.

[Cao+20] Yang Cao et al. “PGLP: Customizable and Rigorous Location Privacy through Policy Graph”. In: *Proceedings of the 25th European Symposium on Research in Computer Security (ESORICS).* Springer. 2020, pp. 655–676.

[Cha+03] Don Chamberlin et al. “XQuery: A query language for XML”. In: *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data.* Vol. 682. ACM, 2003, p. 50.

[Cha+13] Konstantinos Chatzikokolakis et al. “Broadening the scope of differential privacy using metrics”. In: *Proceedings of 13th International Symposium on Privacy Enhancing Technologies Symposium (PETS).* Springer. 2013, pp. 82–102.

[Che+16] Mian Cheng et al. “An event grouping approach for infinite stream with differential privacy”. In: *Proceedings of the 10th Asia-Pacific Services Computing Conference (APSCC).* Springer. 2016, pp. 106–116.

[Che+17] Yan Chen et al. “Pegasus: Data-adaptive differentially private stream processing”. In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS).* ACM. 2017, pp. 1375–1388.

[Che+19] Xiaoli Chen et al. “Open is not enough”. In: *Nature Physics* 15.2 (2019), pp. 113–119.

[CKR21] Rachel Cummings, Gabriel Kaptchuk, and Elissa M Redmiles. “’I need a better description’: An Investigation Into User Expectations For Differential Privacy”. In: *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security (CCS).* 2021, pp. 3037–3052.

[Cor+18] Graham Cormode et al. “Privacy at scale: Local differential privacy in practice”. In: *Proceedings of the 2018 ACM SIGMOD International Conference on Management of Data.* ACM, 2018, pp. 1655–1658.

[CRF+03] William W Cohen, Pradeep Ravikumar, Stephen E Fienberg, et al. “A Comparison of String Distance Metrics for Name-Matching Tasks”. In: *Proceedings of IJCAI-03 Workshop on Information Integration on the Web (IIWeb-03).* 2003, pp. 73–78.

[CSJ15] Rui Chen, Yilin Shen, and Hongxia Jin. “Private analysis of infinite data streams via retroactive grouping”. In: *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management (CIKM).* ACM. 2015, pp. 1061–1070.

[CY15] Yang Cao and Masatoshi Yoshikawa. "Differentially private real-time data release over infinite trajectory streams". In: *Proceedings of the 16th IEEE International Conference on Mobile Data Management (MDM)*. IEEE. 2015, pp. 68–73.

[Def77] Daniel Defays. "An efficient algorithm for a complete link method". In: *The Computer Journal* 20.4 (1977), pp. 364–366.

[Dem+10] Erhan Demirok et al. "Evaluation of the voltage support strategies for the low voltage grid connected PV generators". In: *Proceedings of the 2010 IEEE Energy Conversion Congress and Exposition (ECCE)*. IEEE. 2010, pp. 710–717.

[Deu+21] Daniel Deutch et al. "On optimizing the trade-off between privacy and utility in data provenance". In: *Proceedings of the 2021 ACM SIGMOD International Conference on Management of Data*. 2021, pp. 379–391.

[DH76] Whitfield Diffie and Martin Hellman. "New directions in cryptography". In: *IEEE Transactions on Information Theory* 22.6 (1976), pp. 644–654.

[DR+14] Cynthia Dwork, Aaron Roth, et al. "The Algorithmic Foundations of Differential Privacy". In: *Foundation and Trends ® in Theoretical Computer Science* 9.3-4 (2014), pp. 211–407.

[Dwo+10] Cynthia Dwork et al. "Differential privacy under continual observation". In: *Proceedings of the forty-second ACM symposium on Theory of Computing (STOC)*. 2010, pp. 715–724.

[Dwo+11] Cynthia Dwork et al. "Differential privacy—a primer for the perplexed". In: *Joint UNECE/Eurostat work session on statistical data confidentiality* 11 (2011).

[Dwo08] Cynthia Dwork. "Differential privacy: A survey of results". In: *Proceedings of the 5th International Conference on Theory and Applications of Models of Computation (TAMC)*. Springer. 2008, pp. 1–19.

[Dwo11] Cynthia Dwork. "Differential Privacy". In: *Encyclopedia of Cryptography and Security*. Springer, 2011.

[EE17] Günther Eibl and Dominik Engel. "Differential privacy for real smart metering data". In: *Computer Science-Research and Development (CSRD)* 32.1 (2017), pp. 173–182.

[EF15] Murat A Erdogdu and Nadia Fawaz. "Privacy-utility trade-off under continual observation". In: *Proceedings of the 2015 IEEE International Symposium on Information Theory (ISIT)*. IEEE. 2015, pp. 1801–1805.

[Eib+18] Günther Eibl et al. "The influence of differential privacy on short term electric load forecasting". In: *Energy Informatics* 1.1 (2018), pp. 93–113.

[EL18] Fatima Zahra Errounda and Yan Liu. "Continuous location statistics sharing algorithm with local differential privacy". In: *Proceedings of the 2018 IEEE International Conference on Big Data (Big Data)*. IEEE. 2018, pp. 5147–5152.

[EN00] Ramez Elmasri and Shamkant B Navathe. *Fundamentals of Database Systems*. Springer, 2000.

[Est+96] Martin Ester et al. "A density-based algorithm for discovering clusters in large spatial databases with noise." In: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD).* Vol. 96. 34. AAAI Press. 1996, pp. 226–231.

[Eur19] European Committee for Electrotechnical Standardization. *DIN EN50160: Voltage characteristics of electricity supplied by public distribution systems.* Berlin: Beuth Verlag, 2019.

[Eza+19] Soheila Ghane Ezabadi et al. "Differentially private streaming to untrusted edge servers in intelligent transportation system". In: *2019 18th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/13th IEEE International Conference on Big Data Science and Engineering (TrustCom/BigDataSE).* IEEE. 2019, pp. 781–786.

[FG07] Caroline Fontaine and Fabien Galand. "A survey of homomorphic encryption for nonspecialists". In: *EURASIP Journal on Information Security* (2007), pp. 1–10.

[FLC17] Jingyao Fan, Qinghua Li, and Guohong Cao. "Privacy disclosure through smart meters: Reactive power based attack and defense". In: *Proceedings of the 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN).* IEEE. 2017, pp. 13–24.

[FMV20] Ferdinando Fioretto, Terrence W K Mak, and Pascal Van Hentenryck. "Differential Privacy for Power Grid Obfuscation". In: *IEEE Transactions on Smart Grid* 11.2 (2020), pp. 1356–1366.

[FV18] Ferdinando Fioretto and Pascal Van Hentenryck. "Constrained-based differential privacy: Releasing optimal power flow benchmarks privately". In: *Proceedings of the 15th International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research (CPAIOR 2018).* Springer. 2018, pp. 215–231.

[FX14] Liyue Fan and Li Xiong. "An Adaptive Approach to Real-Time Aggregate Monitoring With Differential Privacy". In: *IEEE Transactions on Knowledge and Data Engineering (TKDE)* 26.9 (2014), pp. 2094–2106.

[FXS13] Liyue Fan, Li Xiong, and Vaidy Sunderam. "FAST: differentially private real-time aggregate monitor with filtering and adaptive sampling". In: *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data.* ACM, 2013, pp. 1065–1068.

[Gol09] Oded Goldreich. *Foundations of cryptography: volume 2, basic applications.* Cambridge University Press, 2009.

[Got+11] Sebastian Gottwalt et al. "Demand side management—A simulation of household behavior under variable prices". In: *Energy Policy* 39.12 (2011), pp. 8163–8174.

[GV15] Quan Geng and Pramod Viswanath. "The optimal noise-adding mechanism in differential privacy". In: *IEEE Transactions on Information Theory* 62.2 (2015), pp. 925–951.

[HA18] Rob J Hyndman and George Athanasopoulos. *Forecasting: principles and practice.* OTexts, 2018.

[Hay+16] Michael Hay et al. "Principled evaluation of differentially private algorithms using dpbench". In: *Proceedings of the 2016 ACM SIGMOD International Conference on Management of Data.* 2016, pp. 139–154.

[HMD14] Xi He, Ashwin Machanavajjhala, and Bolin Ding. "Blowfish privacy: Tuning privacy-utility trade-offs using policies". In: *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data.* ACM. 2014, pp. 1447–1458.

[HPF14] Tao Hong, Pierre Pinson, and Shu Fan. "Global energy forecasting competition 2012". In: *International Journal of Forecasting* 30.2 (2014), pp. 357–363.

[HPK11] Jiawei Han, Jian Pei, and Micheline Kamber. *Data mining: concepts and techniques.* Elsevier, 2011.

[IC19] Florin Iov and Catalin Ciontea. *Net2DG Deliverable D5.1 – First integrated deployment at Lab and Testbed.* Tech. rep. Net2DG, 2019.

[Iov+21] Florin Iov et al. *Net2DG Deliverable D5.3 - Final Consolidated Results.* Tech. rep. Net2DG, 2021.

[KBB15] Stephan Kessler, Erik Buchmann, and Klemens Böhm. "Deploying and evaluating pufferfish privacy for smart meter data". In: *Proceedings of the IEEE 12th Intl Conf on Ubiquitous Intelligence and Computing and IEEE 12th Intl Conf on Autonomic and Trusted Computing and IEEE 15th Intl Conf on Scalable Computing and Communications and Its Associated Workshops (UIC-ATC-ScalCom).* IEEE. 2015, pp. 229–238.

[Kel+14] Georgios Kellaris et al. "Differentially private event sequences over infinite streams". In: vol. 7. 12. VLDB Endowment, 2014, pp. 1155–1166.

[KFB15] Stephan Kessler, Christoph M Flath, and Klemens Böhm. "Allocative and strategic effects of privacy enhancement in smart grids". In: *Information Systems* 53 (2015), pp. 170–181.

[Kho+10] Nodira Khoussainova et al. "SnipSuggest: Context-aware autocompletion for SQL". In: *Proceedings of the VLDB Endowment (PVLDB)* 4.1 (2010), pp. 22–33.

[KLM21] Piotr Krasnowski, Jerome Lebrun, and Bruno Martin. "Introducing an experimental distortion-tolerant speech encryption scheme for secure voice communication". In: *arXiv preprint arXiv:2102.09809* (2021).

[KLM22] Piotr Krasnowski, Jerome Lebrun, and Bruno Martin. "A novel distortion-tolerant speech encryption scheme for secure voice communication". In: *Speech Communication* (2022).

[KM14] Daniel Kifer and Ashwin Machanavajjhala. “Pufferfish: A framework for mathematical privacy definitions”. In: *ACM Transactions on Database Systems (TODS)* 39.1 (2014), 3:1–3:36.

[KNT00] Edwin M Knorr, Raymond T Ng, and Vladimir Tucakov. “Distance-based outliers: algorithms and applications”. In: *The VLDB Journal* 8.3 (2000), pp. 237–253.

[Knu98] Lars R Knudsen. “Block Ciphers—a survey”. In: *State of the Art in Applied Cryptography*. Springer, 1998, pp. 18–48.

[KOV15] Peter Kairouz, Sewoong Oh, and Pramod Viswanath. “The composition theorem for differential privacy”. In: *Proceedings of the 32nd International conference on Machine Learning (ICML)*. Proceedings of Machine Learning Research. 2015, pp. 1376–1385.

[Kra21] Piotr Krasnowski. “Joint source-cryptographic-channel coding for real-time secure voice communications on voice channels”. PhD thesis. Université Côte d’Azur, 2021.

[LBB15] Fabian Laforet, Erik Buchmann, and Klemens Böhm. “Individual privacy constraints on time-series data”. In: *Information Systems* 54 (2015), pp. 74–91.

[LCM16] Changchang Liu, Supriyo Chakraborty, and Prateek Mittal. “Dependence Makes You Vulnberable: Differential Privacy Under Dependent Tuples”. In: *Proceedings of the Annual Network and Distributed System Security Symposium (NDSS)*. Vol. 16. The Internet Society, 2016, pp. 21–24.

[Li+15a] Haoran Li et al. “Differentially private histogram publication for dynamic datasets: an adaptive sampling approach”. In: *Proceedings of the 24th ACM international Conference on Information and Knowledge Management (CIKM)*. ACM. 2015, pp. 1001–1010.

[Li+15b] Jin Li et al. “L-EncDB: A lightweight framework for privacy-preserving data queries in cloud computing”. In: *Knowledge-Based Systems* 79 (2015), pp. 18–26.

[Lia+14] Xiaojing Liao et al. “Towards secure metering data analysis via distributed differential privacy”. In: *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*. IEEE. 2014, pp. 780–785.

[Liu+18] Xiang Liu et al. “Trajectory Privacy Protection on Spatial Streaming Data with Differential Privacy”. In: *2018 IEEE Global Communications Conference (GLOBECOM)*. IEEE. 2018, pp. 1–7.

[LLV07] Ninghui Li, Tiancheng Li, and Suresh Venkatasubramanian. “t-closeness: Privacy beyond k-anonymity and l-diversity”. In: *Proceedings of the 23rd International Conference on Data Engineering (ICDE)*. IEEE. 2007, pp. 106–115.

[Lyu+17] Lingjuan Lyu et al. “Privacy-preserving aggregation of smart metering via transformation and encryption”. In: *Proceedings of the 2017 IEEE Trustcom/BigDataSE/ICESS*. IEEE. 2017, pp. 472–479.

[Maa+15] Heiko Maaß et al. “Data processing of high-rate low-voltage distribution grid recordings for smart grid monitoring and analysis”. In: *EURASIP Journal on Advances in Signal Processing* 2015.1 (2015), pp. 1–21.

[Mac+07] Ashwin Machanavajjhala et al. “l-diversity: Privacy beyond k-anonymity”. In: *ACM Transactions on Knowledge Discovery from Data (TKDD)* (2007), pp. 106–115.

[Man+12] Efthymios Manitsas et al. “Distribution system state estimation using an artificial neural network approach for pseudo measurement modeling”. In: *IEEE Transactions on Power Systems* 27.4 (2012), pp. 1888–1896.

[McS09] Frank D McSherry. “Privacy integrated queries: an extensible platform for serving data analysis”. In: *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data.* 2009, pp. 19–30.

[MHH17] Ashwin Machanavajjhala, Xi He, and Michael Hay. “Differential privacy in the wild: A tutorial on current practices & open challenges”. In: *Proceedings of the 2017 ACM SIGMOD International Conference on Management of Data.* ACM. 2017, pp. 1727–1730.

[Mir17] Ilya Mironov. “Rényi differential privacy”. In: *Proceedings of the 30th IEEE Computer Security Foundations Symposium (CSF).* IEEE. 2017, pp. 263–275.

[Mol+10] Andrés Molina-Markham et al. “Private memoirs of a smart meter”. In: *Proceedings of the 2nd ACM workshop on embedded sensing systems for energy-efficiency in building (BuildSys).* ACM, 2010, pp. 61–66.

[MT07] Frank McSherry and Kunal Talwar. “Mechanism design via differential privacy”. In: *Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS).* IEEE. 2007, pp. 94–103.

[MVW09] Simon Moncrieff, Svetha Venkatesh, and Geoff AW West. “Dynamic privacy in public surveillance”. In: *Computer* 42.9 (2009), pp. 22–28.

[Nai+21] Karthikeyan Nainar et al. “Experimental validation and deployment of observability applications for monitoring of low-voltage distribution grids”. In: *Sensors* 21.17 (2021), p. 5770.

[Nan+19] L. Nandakumar et al. “Protecting the grid topology and user consumption patterns during state estimation in smart grids based on data obfuscation”. In: *Energy Informatics* 2.1 (2019), pp. 1–23.

[Ngu+15] Hoang Vu Nguyen et al. “Identifying User Interests within the Data Space-a Case Study with SkyServer.” In: *Proceedings of the 18th International Conference on Extending Database Technology (EDBT).* Vol. 2015. 2015, pp. 641–652.

[NI20] Karthikeyan Nainar and Florin Iov. “Smart Meter Measurement-Based State Estimation for Monitoring of Low-Voltage Distribution Grids”. In: *Energies* 13.20 (2020), p. 5367.

[Nie+16] Yiwen Nie et al. "Geospatial streams publish with differential privacy". In: *International Conference on Collaborative Computing: Networking, Applications and Worksharing*. Springer. 2016, pp. 152–164.

[Nie+17] Jimmy J Nielsen et al. "Secure real-time monitoring and management of smart distribution grid using shared cellular networks". In: *IEEE Wireless Communications* 24.2 (2017), pp. 10–17.

[NKW15] Muhammad Naveed, Seny Kamara, and Charles V Wright. "Inference attacks on property-preserving encrypted databases". In: *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (CCS)*. 2015, pp. 644–655.

[NRS07] Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. "Smooth sensitivity and sampling in private data analysis". In: *Proceedings of the thirty-ninth annual ACM Symposium on Theory of Computing (STOC)*. ACM, 2007, pp. 75–84.

[Pai99] Pascal Paillier. "Public-key cryptosystems based on composite degree residuosity classes". In: *Proceedings of the 17th International Conference on Theory and Application of Cryptographic Techniques (EUROCRYPT)*. Springer. 1999, pp. 223–238.

[Pap+07] Spiros Papadimitriou et al. "Time series compressibility and privacy". In: *Proceedings of the 33rd International Conference on Very Large Data Bases (VLDB)*. VLDB Endowment. 2007.

[Paw+19] Mateusz Pawlik et al. "A link is not enough–reproducibility of data". In: *Datenbank-Spektrum* 19.2 (2019), pp. 107–115.

[PG20] Praviraj PG. *Newton-Raphson Loadflow*. MATLAB Central File Exchange. `https://www.mathworks.com/matlabcentral/fileexchange/21059-newton-raphson-loadflow`. May 2020.

[PJ09] Hae-Sang Park and Chi-Hyuck Jun. "A simple and fast algorithm for K-medoids clustering". In: *Expert systems with applications* 36.2 (2009), pp. 3336–3341.

[Pom+18] Daniel Vázquez Pombo et al. *Net2DG Deliverable D2.1 – Algorithms for grid estimation and observability applications*. Tech. rep. Net2DG, 2018.

[Pop+11] Raluca Ada Popa et al. "CryptDB: protecting confidentiality with encrypted query processing". In: *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles (SOSP)*. ACM. 2011, pp. 85–100.

[PSW09] Dane Petersen, Jay Steele, and Joe Wilkerson. "WattBot: a residential electricity monitoring and feedback system". In: *Extended Abstracts of the 2019 Conference on Human Factors in Computing Systems (CHI)*. 2009, pp. 2847–2852.

[Pur+16] Detty Purnamasari et al. “Query Rewriting and Corpus of Semantic Similarity as Encryption Method for Documents in Indonesian Language”. In: *Proceedings of the Second International Conference on Electrical Systems, Technology and Information (ICESTI),* Springer. 2016, pp. 565–571.

[Ren+22] Xuebin Ren et al. “LDP-IDS: Local Differential Privacy for Infinite Data Streams”. In: *Proceedings of the 2022 SIGMOD International Conference on Management of Data* (2022), pp. 1064–1077.

[RN13] Sushmita Ruj and Amiya Nayak. “A decentralized security framework for data aggregation and access control in smart grids”. In: *IEEE Transactions on Smart Grid* 4.1 (2013), pp. 196–205.

[Sch+13] Martin Schäler et al. “QuEval: Beyond high-dimensional indexing à la carte”. In: *Proceedings of the VLDB Endowment (PVLDB)* 6.14 (2013), pp. 1654–1665.

[Sch+21] Christine Schäler et al. “Increased Renewable Hosting Capacity of a Real Low-Voltage Grid Based on Continuous Measurements – Results from an Actual PV Connection Request”. In: *Proceedings of the 17th European Dependable Computing Conference (EDCC) Workshops* (2021), pp. 90–98.

[SD13] Jordi Soria-Comas and Josep Domingo-Ferrer. “Optimal data-independent noise for differential privacy”. In: *Information Sciences* 250 (2013), pp. 200–214.

[SDT15] Henrik Sandberg, György Dán, and Ragnar Thobaben. “Differentially private state estimation in distribution networks with smart meters”. In: *Proceedings of the 54th IEEE Conference on Decision and Control (CDC).* IEEE. 2015, pp. 4492–4498.

[SEJ14] Bharath K Samanthula, Yousef Elmehdwi, and Wei Jiang. “K-nearest neighbor classification over semantically secure encrypted relational data”. In: *IEEE Transactions on Knowledge and Data Engineering (TKDE)* 27.5 (2014), pp. 1261–1273.

[SFY07] Yasushi Sakurai, Christos Faloutsos, and Masashi Yamamuro. “Stream monitoring under the time warping distance”. In: *Proccedings of the 23rd IEEE International Conference on Data Engineering (ICDE).* IEEE. 2007, pp. 1046–1055.

[Sha49] Claude Shannon. “Communication Theory of Secrecy Systems”. In: *The Bell System Technical Journal* 28.4 (1949), pp. 656–715.

[SHS23] Christine Schäler, Thomas Hütter, and Martin Schäler. “Benchmarking the Utility of w-event Differential Privacy Mechanisms – When Baselines Become Mighty Competitors”. In: *Proceedings of the VLDB Endowment (PVLDB)* 16.8 (2023), pp. 1830–1842.

[Sil+09] Fabrizio Silvestri et al. “Mining query logs: Turning search usage data into knowledge”. In: *Foundations and Trends® in Information Retrieval* 4.1–2 (2009), pp. 1–174.

[SK13] Tahmineh Sanamrad and Donald Kossmann. "Query log attack on encrypted databases". In: *Proceedings of the 10th VLDB Workshop on Secure Data Managemen (SDM)*. Springer, 2013, pp. 95–107.

[SKS02] Abraham Silberschatz, Henry F Korth, and Shashank Sudarshan. *Database system concepts*. Vol. 5. McGraw-Hill New York, 2002.

[SMA21] Chandra Sharma, Bishwas Mandal, and George Amariucai. "A practical approach to navigating the tradeoff between privacy and precise utility". In: *Proceedings of the 2021 IEEE International Conference on Communications (ICC)*. IEEE. 2021, pp. 1–6.

[SRP13] Lalitha Sankar, S Raj Rajagopalan, and H Vincent Poor. "Utility-privacy tradeoffs in databases: An information-theoretic approach". In: *IEEE Transactions on Information Forensics and Security* 8.6 (2013), pp. 838–852.

[ST01] National Institute of Standards and Technology. *FIPS 197 – Advanced Encryption Standard (AES)*. Tech. rep. Washington, D.C.: U.S. Department of Commerce, 2001.

[Sta16] International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC). *ISO/IEC 9075-*: Database Language SQL*. Beuth Verlag, Berlin, 2016.

[Str16] Bruce N Stram. "Key challenges to expanding renewable energy". In: *Energy Policy* 96 (2016), pp. 728–734.

[SWC17] Shuang Song, Yizhen Wang, and Kamalika Chaudhuri. "Pufferfish privacy mechanisms for correlated data". In: *Proceedings of the 2017 ACM SIGMOD International Conference on Management of Data*. ACM, 2017, pp. 1291–1306.

[Swe02] Latanya Sweeney. "k-anonymity: A model for protecting privacy". In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 10.05 (2002), pp. 557–570.

[SWP00] Dawn Xiaoding Song, David Wagner, and Adrian Perrig. "Practical techniques for searches on encrypted data". In: *Proceedings of the 2000 IEEE Symposium on Security and Privacy (S&P)*. IEEE. 2000, pp. 44–55.

[Tex+18] Christine Tex et al. "PrivEnergy: a privacy operator framework addressing individual concerns". In: *Proceedings of the Ninth International Conference on Future Energy Systems (e-Energy)*. ACM. 2018, pp. 426–428.

[TM01] Herman T Tavani and James H Moor. "Privacy protection, control of information, and privacy-enhancing technologies". In: *ACM SIGCAS Computers & Society* 31.1 (2001), pp. 6–11.

[Wan+16a] Qian Wang et al. "Real-time and spatio-temporal crowd-sourced social network data publishing with differential privacy". In: *IEEE Transactions on Dependable and Secure Computing (TDSC)* 15.4 (2016), pp. 591–606.

[Wan+16b] Qian Wang et al. “RescueDP: Real-time spatio-temporal crowd-sourced data publishing with differential privacy”. In: *Proceedings of the 35th Annual IEEE International Conference on Computer Communications (INFOCOM).* IEEE. 2016, pp. 1–9.

[Wan+16c] Qian Wang et al. “Secweb: Privacy-preserving web browsing monitoring with w-event differential privacy”. In: *Proceedings of the 12th EAI International Conference on Security and Privacy in Communication Systems (SecureComm).* Springer. 2016, pp. 454–474.

[Wan+18] Zhibo Wang et al. “Privacy-preserving crowd-sourced statistical data publishing with an untrusted server”. In: *IEEE Transactions on Mobile Computing* 18.6 (2018), pp. 1356–1367.

[Wan+19] Teng Wang et al. “Adaptive differentially private data stream publishing in spatio-temporal monitoring of IoT”. In: *Proceedings of the 38th IEEE International Performance Computing and Communications Conference (IPCCC).* IEEE. 2019, pp. 1–8.

[Wan+21] Tianhao Wang et al. “Continuous release of data streams under both centralized and local differential privacy”. In: *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security (CCS).* 2021, pp. 1237–1253.

[WB95] Greg Welch and Gary Bishop. *An introduction to the Kalman filter.* Tech. rep. Department of Computer Science, University of North Carolina at Chapel Hill, 1995.

[WE18] Isabel Wagner and David Eckhoff. “Technical privacy metrics: a systematic survey”. In: *ACM Computing Surveys (CSUR)* 51.3 (2018), pp. 1–38.

[Wei19] Nico Weidmann. “Differentially private event sequences over infinite streams”. Bachlor Thesis (advisor: Christine Tex). Karlsruhe Institute of Technology, 2019.

[Wes67] Alan F Westin. *Privacy and freedom.* New York: Atheneum, 1967.

[Won+09] Wai Kit Wong et al. “Secure kNN computation on encrypted databases”. In: *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data.* ACM, 2009, pp. 139–152.

[YH10] Che-Chang Yang and Yeh-Liang Hsu. “A review of accelerometry-based wearable motion detectors for physical activity monitoring”. In: *Sensors* 10.8 (2010), pp. 7772–7788.

[Zha+17] Jiajun Zhang et al. “Re-DPoctor: Real-time health data releasing with w-day differential privacy”. In: *Proceedings of the IEEE Global Communications Conference (GLOBECOM).* IEEE. 2017, pp. 1–6.

[Zhu+00] Zhigang Zhu et al. “VISATRAM: A real-time vision system for automatic traffic monitoring”. In: *Image and Vision Computing* 18.10 (2000), pp. 781–794.

[Zhu+14] Tianqing Zhu et al. “Correlated differential privacy: Hiding information in non-IID data set”. In: *IEEE Transactions on Information Forensics and Security* 10.2 (2014), pp. 229–242.

[ZKL22] Xueru Zhang, Mohammad Mahdi Khalili, and Mingyan Liu. “Differentially Private Real-Time Release of Sequential Data”. In: *ACM Transactions on Privacy and Security* (2022).