Differential Privacy Under a Constrained Dynamic Database Model

by

Alex Jacey Ligthart-Smith

ORCID: 0000-0002-3846-9061

A thesis submitted in total fulfilment for the
degree of Master of Philosophy

in the
School of Computing and Information Systems
Faculty of Engineering and Information Technology
THE UNIVERSITY OF MELBOURNE

July 2024

Abstract

The collection, storage, and use of sensitive data often requires a trade-off between individual privacy and public utility. Differential privacy (DP) formalises this trade-off, for releasing information about sensitive data, by providing provable privacy guarantees for individual dataset participants. There is a wealth of literature proposing DP algorithms, with good utility, for static datasets; however, privacy and accuracy losses accrue over multiple releases. In dynamic settings, where the dataset is growing or changing over time, the same queries are likely repeated as the dataset changes, requiring large numbers of private releases. Dynamic DP is an active area of research, with most work considering data streams, where each entry is fixed once it is added, or trajectory data, with a separate stream of updates for each individual; databases where records are updated over time, with only the latest update available for analysis, are common in practice but less thoroughly researched in the DP literature.

In this thesis, we consider a setting where the set of individuals in the database is fixed, and one individual's state is updated per unit of time. Prior to introducing our model, we present a taxonomy of the models used in key dynamic DP papers. We classify these models primarily according to their update types, and further distinguish them according to their privacy definitions. This allows us to bring together previously fragmented research into a cohesive framework.

We then introduce our fixed-size dynamic database (FSDD) model. We provide a base mechanism, τ-RQ, for repeating the same query after each update to the FSDD. Using properties of the FSDD model, we can utilise existing results from static DP to determine the optimal query frequency in the worst case, balancing the effects of random noise, added to preserve privacy, with the accrued changes to the underlying database. We extend τ-RQ to provide a mechanism to convert any static DP mechanism to the FSDD setting, and employ this to provide a mechanism for answering interactively chosen queries. Additionally, we modify existing techniques, based on the sparse vector technique (SVT), to provide DP in the FSDD setting, for use as a baseline.

We give theoretical guarantees on the privacy and accuracy of our mechanisms. Experimentally, we show that our τ-RQ mechanisms typically provide better accuracy than an SVT-based technique in the repeated query setting, and consistently provide better accuracy that the SVT-based Private Multiplicative Weights (PMW) technique in the adaptive query setting. We conclude that the characteristics of the FSDD model can be effectively harnessed to provide good utility with mechanisms that are relatively simple to implement.

Declaration of Authorship

I, Alex Jacey Ligthart-Smith, declare that this thesis titled, 'Differential Privacy Under a Constrained Dynamic Database Model' and the work presented in it are my own. I confirm that:

- The thesis comprises only my original work towards the Master of Philosophy; except where indicated in the preface;
- due acknowledgement has been made in the text to all other material used; and
- the thesis is fewer than 50,000 words in length, exclusive of tables, maps, bibliographies and appendices as approved by the Research Higher Degrees Committee.

Signed: ______________________________

Date: ______________________________

Acknowledgements

First and foremost I would like to thank my supervisors, Olya Ohrimenko and Tony Wirth. Olya's experience in the data privacy field has inspired my research and given me confidence in my own work. Tony's unwavering support has allowed me to pursue research when I otherwise might not have believed I could. Additional thanks to my Advisory Committee Chair, Alistair Moffat, for his thoughtful and pertinent advice.

Thank you to my best friend and life partner for staying by my side throughout my studies. I am so lucky to have found you, and that we have held on to each other through all these years. Thank you for feeding me and for making homes of our many houses through these years. Thank you for listening to me complain and for reminding me that I can make it through. To the rest of my chosen family, especially Joshua and Caspian, I couldn't have made it this far without you all.

Thank you to my parents: dad, for always believing in me; mum, for teaching me to hope. To my siblings: Jasmine, for being my first role model; Jaffa, for surviving with me and giving me the strength to continue. To my niblings: Jenna, for reminding me to be curious; Morgan, for reminding me to take pride in my strengths; Juli, for reminding me to laugh; and all three, for reminding me to be brave and to always be true to myself.

Finally, I would like to acknowledge that this work was made possible by the support of an Australian Government Research Training Program Scholarship, as well as additional funding from the University of Melbourne School of Computing and Information Systems and Faculty of Engineering and Information Technology.

Contents

List of Figures

List of Tables

List of Algorithms

Abbreviations

DP	Differential Privacy/Differentially Private
ε**-DP**	Pure differential privacy
(ε, δ)**-DP**	Approximate differential privacy
FSDD	Fixed-Size Dynamic Database
FSDD-T	FSDD with T-bounded updates
FSDD-∞	FSDD with infinite updates
MW	Multiplicative Weights update method
PMW	Private Multiplicative Weights algorithm
SVT	Sparse Vector Technique

Symbols and Functions

$\approx$	Neighbouring databases	(see page 10)
$\approx_{\pm}$	Unbounded neighbours	(see Definition 2.1.2)
$\approx_{\circlearrowleft}$	Bounded neighbours	(see Definition 2.1.3)
$[a..b]$	Integers from $a \in \mathbb{Z}$ to $b(> a) \in \mathbb{Z}$	$\{a, a+1, ..., b-1, b\}$
$[n]$	Integers from 1 to $n \in \mathbb{N}$	$[1..n]$
$[a]^n$	List of n copies of a	$[a, a, ..., a]$ of length n
$\{a_i\}_{i=n}^{m}$	Stream of a_is for times indexed from x to y	$\{a_x, a_{x+1}, ..., a_{y-1}, a_y\}$
$\{a_i\}$	Shorthand when stream indices clear from context	$\{a_i\}_{i=n}^{m}$
$(x)_+$	The ramp function	$\max\{0, x\}$
$d_{\pm}$	Add/delete only edit distance	(see Definition 2.1.2)
Δ_f	The ℓ_1 sensitivity of function f	(see Definition 2.1.4)
$\mathrm{Lap}\,(b)$	Laplace R.V. with scale b	$L \sim \mathrm{Laplace}(0, b)$
$\mathrm{RE}(x\|y)$	Relative Entropy from x to y	(see Definition 2.1.11)

Chapter 1

Introduction

Strong encryption can protect sensitive data while it is stored. But, what if we need to provide information about that sensitive data to untrusted parties? Statistics, machine learning models, and other information obtained from a data set can be utilised to reveal specific details of the raw data. For example, Garfinkel et al. [38] show how to reconstruct the details of individuals in a population from aggregated census data; Dinur and Nissim [18] show how to reconstruct a database from *noisy* query answers, and Cohen and Nissim [13] take this further, doing so even when the queries themselves are randomised. Data privacy has become increasingly pertinent as the collection and sharing of detailed personal data has proliferated; Statista have estimated that the volume of data created, captured, copied, and consumed worldwide increased by a factor of over 30 in the decade from 2010-20 alone [44]. While much of this data is used to increase convenience and efficiency in the public and private sectors, and our everyday lives, its propagation also increases the risk that our private information can reach the wrong hands. The effectiveness of machine learning has benefited immensely from this abundance of data, and while the training data is not directly represented in outputs, private information can still be leaked. For example, Calandrino et al. [4] are able to infer the purchasing behaviour of individuals using the recommendations given to another user, and Fredrikson et al. [36] infer individual genomic markers from a machine learning model for personalised medication dosing.

Differential privacy (DP), first introduced by Dwork et al. [24], provides a rigorous framework for provably guaranteeing the privacy of individuals' data. Specifically, differentially private mechanism outputs reveal information about a dataset while bounding the probability that an adversary can infer anything about the data of any individual contributor to the dataset. Dwork et al. [24] show that this privacy, however, must come at a cost to accuracy, and as multiple DP releases are made using the data of an individual, such as multiple queries on a single dataset, or queries on multiple datasets containing the same individual, the privacy loss increases. Total privacy loss for an individual can be bounded, by rationing a privacy budget between releases, but this comes at a cost to accuracy.

Many strategies have been developed to mitigate the effects of this privacy/accuracy tradeoff. Dwork et al. [23] give a definition of *approximate*, or (α, δ)-, differential privacy, whereby improvements to accuracy are achieved at a cost of a small probability of privacy failure. The sparse vector technique (SVT) was first conceptualised by Dwork et al. [25] to construct sanitised datasets. Using SVT, a sequence of threshold queries can be answered, with the privacy budget only diminished for the above threshold queries, maintaining a fixed accuracy bound so long as the number of above threshold queries remains below some constant. Hardt and Rothblum [42] utilised this technique in their private multiplicative weights (PMW) mechanism, where the distribution of a dataset is learned over time using the results of a sequence of queries, with both accuracy and privacy bounded for an unbounded number of queries. SVT and PMW are designed to answer many queries on a static dataset, where no records are added or changed during the query stream. But what if the underlying data set is being modified in between queries? In this thesis, we are interested specifically in dynamic databases. That is, a dataset that is shifting over time, with queries answered between updates to the database.

Differential privacy in a dynamic setting was first studied by Dwork et al. [26] and Chan et al. [9]. These works consider *bitstreams*, where a single bit of data arrives at each time. They establish important bounds on the overhead of providing dynamic DP, and introduce novel techniques that provide *pan-privacy*, a stronger model than DP that additionally provides protection against adversarial intrusions into the centrally held data. Dwork et al. [27] extended this work to consider infinite streams of categorical data, from finite data universes of arbitrary size. This categorical setting has been further studied, for example, Cummings et al. [15] provide black box mechanisms for converting static DP mechanisms to the growing data setting, as well as a PMW mechanism for growing datasets. Perrier et al. [58] and Wang et al. [66] provide mechanisms for streams of real-valued data, rather than data from a finite universe. In these growing models, data values remain fixed once they are added to the stream. This limits the ability of mechanisms to take into account the changing state or value of a single individual, such as when only the most recent data value of any individual is relevant.

Time-series and trajectory data is a widely studied area of dynamic DP, particularly relevant to the privacy of the internet of things. In this setting, each individual may update their data value at each time, and it is either the most recent value or the temporal patterns of the user updates that are of interest. This model was first studied by Rastogi and Nath [60] in a distributed data setting, and has since received wide attention. Some key works have included those of Fan and Xiong [31, 32] and Fan et al. [33, 34], who provide mechanisms for real-time aggregate monitoring, and Kellaris et al. [49], who introduce a sliding-window privacy definition for infinite streams within the time-series context.

There is less existing research in settings where only a single individual's data is updated at each time. Mir et al. [56] consider a fixed universe of individuals and a sequence of single updates to some individual's integer-valued state. They give mechanisms that provide pan-privacy, so are limited in the types of queries they can answer by the fact that the underlying data structure must itself be private. More recently, Qiu and Yi [59] consider a data model whereby individual data points can be added to and deleted from the database, but they do not consider the privacy implications of multiple entries or updates representing the same individual. We consider a model similar to that of Mir et al. [56], with a simpler and less restrictive privacy definition.

We define a dynamic database model with the following constraints: the set of users is fixed, each user is in some state from a finite data universe, and a single user updates their state at each time. For example, a sequence of scheduled reviews to the status of some group of individuals. We call this the *fixed-size dynamic database* (FSDD) model, and define privacy in terms of protecting all updates from any user. In this thesis we primarily consider a further constrained model, FSDD-T, where the maximum number of updates, T is known in advance, and use a *pure*, rather than approximate, definition of DP, i.e. ε-DP. These constraints allow us to model a database as a histogram of counts at each time t, and to bound the effect of each update by the maximum *sensitivity* of the allowed queries, and thus ration the privacy budget in advance based on the maximum number of updates. These qualities are particularly useful in allowing us to utilise existing results about the composition of differentially private mechanisms, though the use of a pure DP definition limits this advantage. We are interested in determining the extent to which we can leverage these constraints to improve the accuracy of differentially private mechanisms in the FSDD setting. We consider both theoretical accuracy bounds and empirical performance of mechanisms designed specifically for the FSDD setting, compared to more generally applicable dynamic DP mechanisms.

1.1 Contributions

Taxonomy of dynamic DP models. Prior to introducing the FSDD model, we describe the results of a survey and analysis of the data and privacy models of key dynamic DP papers. We bring together work that has been previously fragmented between various venues and research areas, both theoretical and applied. Our analysis indicates that there are three broad categories of dynamic data models present in the literature, *growing*, *time-series*, and *fully dynamic*, based on the number and type of updates allowed at each time. We create a taxonomy of the dynamic DP models in key papers, dividing them into these categories and further distinguishing them by their time-bounds and privacy models.

FSDD model definition. We define the FSDD data and privacy models, distinguishing it from the model of Mir et al. [56].

Algorithms. We provide DP mechanisms in the FSDD setting, and evaluate their performance with both theoretical bounds and experimental results. First, we consider the *repeated query* (A.K.A. *continual release*) setting, whereby a single query is applied to the data after each update. Our mechanism for privately answering numerical queries, the τ-frequency repeated query (τ-RQ), utilises the constraints of the FSDD model to improve theoretical performance over basic composition, and give a method for choosing the optimal frequency parameter τ. We prove that τ-RQ is ε-DP, with additive accuracy as given below. Proofs of theorems presented here are given in the corresponding chapters.

Theorem 3.2.3. *τ-RQ (Algorithm 6) is ε-differentially private.*

Theorem 3.2.4. *τ-RQ (Algorithm 6) is (α, β)-accurate for*

$$\alpha = \ln\left(\frac{c}{\beta}\right) \cdot \frac{c\Delta_q}{\varepsilon} + (\tau - 1)\Delta_q\,, \tag{1.1}$$

where $c = \lceil \frac{T}{\tau} \rceil$. That is, all outputs $\{a_t\}_{t=0}^{T-1}$ are within $\pm\alpha$ of the true answer to query q, on database D at time t, with probability $1-\beta$.

Our black box transformation, the τ-frequency repeated black box (τ-RBB) mechanism, generalises this method to transform any static DP mechanism to the FSDD setting, maintaining the privacy guarantee of the static mechanism (Theorem 3.2.7) and reducing a static (α, β)-accuracy guarantee as given below.

Theorem 3.2.7. *τ-RBB (Algorithm 7) is ε-differentially private.*

Theorem 3.2.8. *When instantiated with (α, β)-accurate mechanism $\mathcal{M}$, τ-RBB (Algorithm 7) is (α, β')-accurate, where*

$$\beta' = \left\lceil \frac{T}{\tau} \right\rceil \beta \,.$$

That is, all outputs $\{a_t\}_{t=0}^{T}$ are within $\pm\alpha$ of the true function f on database D at time t with probability $1-\beta'$.

Additionally, we modify SVT to apply in the FSDD setting and utilise it, within our adaptive-frequency repeated query (ARQ) mechanism, to provide DP in the repeated query FSDD setting.

Next, we consider interactive queries, where some sequence of linear queries are applied between each update. The set of possible queries is known in advance but the specific sequences are chosen adaptively. In this setting, we show that τ-RBB can be used to produce ε-differentially private histograms for answering linear queries, with accuracy given below.

Theorem 4.1.1. *τ-RH (Algorithm 10) is ε-differentially private.*

Theorem 4.1.2. *τ-RH (Algorithm 10), when limited some set of linear counting queries $\mathcal{Q} \subseteq \{0,1\}^N$, is (α, β)-accurate for*

$$\alpha = \frac{2c}{\varepsilon} \ln\left(\frac{c \cdot w_{\max}}{\beta}\right) + (\tau - 1)\,, \tag{1.2}$$

where $c = \lceil \frac{T}{\tau} \rceil$, $w_{\max} = \max_{q\in\mathcal{Q}} \sum_{i=1}^{N} w_q[i]$, where $w_q[i]$ is the weight $\in \{0,1\}$ of histogram bucket i for query q. That is, all outputs $\{a_t\}_{t=0}^{T}$ are within $\pm\alpha$ of the true answer to query q on database D_t, at time t, with probability $1-\beta$.

In addition, we modify PMW to provide DP in both the bounded and unbounded time FSDD settings: PMW-FSDD-T and PMW-FSDD-∞.

Experimental evaluation. Experimentally, we show that our τ-RQ based mechanisms provide better accuracy than SVT-based mechanisms across a broad range of scenarios. We generate synthetic data to observe the accuracy of our mechanisms, and the effectiveness of our method for optimising τ, when the distribution is relatively stable and when it is changing significantly over time. In the repeated query setting, we observe that τ-RQ and ARQ have similar accuracy for low-entropy synthetic data, while τ-RQ is significantly more accurate for high entropy data and for a real data set drawn from the US census. In the interactive query setting, we see a stark difference, with our τ-RQ mechanism outperforming PMW by a large margin across all experiments.

1.2 Outline

The remaining chapters of this thesis are organised as follows. Chapter 2 introduces relevant background on differential privacy for both static and dynamic data. Section 2.1 covers key definitions and results in static differential privacy, and describes the standard differentially private mechanisms and results used and adapted in later chapters. This includes both one-shot mechanisms and adaptive techniques for privately answering interactive sequences of queries. Section 2.2 describes existing models and techniques for differentially private analysis of dynamic data. We organise the models according to a number of characteristics, and summarise key definitions and results for each category.

In Chapter 3 we define our fixed-size dynamic database (FSDD) model, situating it within the framework of Section 2.2 and providing updated privacy definitions under this model. We then provide mechanisms for repeating a single query after each database update; the τ-RQ mechanism and ARQ, which utilises SVT. We prove the privacy of these mechanisms and provide theoretical bounds on accuracy. In Chapter 4 we provide a τ-RQ mechanism for privately answering interactive sequences of queries, and a modified PMW mechanism. We prove the privacy of these mechanisms.

In Chapter 5 we present the results of experimental analysis of the mechanisms introduced in Chapter 3 and Chapter 4. We show that our τ-RQ mechanisms demonstrate a significant accuracy advantage over both our SVT-based mechanism and PMW.

Finally, Chapter 6 brings together the theoretical and experimental results. We discuss the implications of these results as well as the limitations and future directions of the research.

Chapter 2

Background and Related Work

This chapter introduces the fundamentals of differential privacy relevant to the mechanisms and proofs that follow in the thesis, and the current landscape of differential privacy for dynamic databases as context for these mechanisms. Section 2.1 lays out definitions, and basic results and mechanisms, for differential privacy in the static setting. Section 2.2 gives existing models for differential privacy in the dynamic setting, as well as some key results and mechanisms that have inspired the work described in the rest of the thesis. The models described have various characteristics that we will draw from in developing our model, which may be considered a special case of a number of these existing models.

2.1 Differential privacy for static data

Differential privacy (DP), was first described by Dwork et al. [24], and formalised by Dwork [20], as a continuation of previous work into private statistical databases [18, 21]. It provides a rigorous mathematical definition to quantify and bound the privacy risk to individuals whose data are used to calculate answers to statistical queries. Alternative privacy definitions exist in the literature, such as k-anonymity [63] and l-diversity [54], each with their own assumptions and limitations. For details of alternative privacy definitions and their vulnerabilities, see the survey by Fung et al. [37]. Any data privacy definition is inherently limited by the fact that perfect semantic security [17, 40], whereby an attacker cannot learn anything about an individual in a database through mechanism outputs, is impossible while maintaining meaningful utility [20]. As such, differential privacy provides a probabilistic guarantee on individual privacy.

Early work in differential privacy focused on the *static* setting, where privacy is guaranteed for a single, unchanging, data set. Most definitions and theorems presented in this section are taken from the seminal textbook in differential privacy by Dwork and Roth [22], which brings together much of the early work in the field.

2.1.1 Data model

First we describe the static data model that will be used throughout, and later adapted to dynamic settings.

Consider a data universe $\mathcal{X}$, and some database $D \in \mathcal{X}^n$ such that an individual data point can be indexed as $D[i]$. If the universe is discrete and finite, its size is $|\mathcal{X}| = N$ and D can be represented as a histogram $x \in \mathbb{N}^N$. Let $[n]$ represent the list of numbers $[1..n]$. The histogram x has N buckets, each representing an element of $\mathcal{X}$, such that, for each $i \in [N]$ and $j \in [n]$, $x[i] = |\{j : D[j] = \mathcal{X}_i\}|$, the count of elements in D with value $\mathcal{X}_i$. For example, the dataset $D = [1, 3, 3, 2, 3, 1, 3, 1, 4, 3]$ drawn from universe $\mathcal{X} = 1, 2, 3, 4$ can be represented as histogram $x = [3, 1, 5, 1]$.

2.1.2 Definitions

Here we formally define differential privacy, and the utility of differentially private mechanisms, as described in key works in the literature.

Differential privacy is defined over the probability distributions of possible outputs of a randomised mechanism $\mathcal{M}$.

Definition 2.1.1 (Differential privacy, from Definition 2.4 of Dwork and Roth [22])**.** *A randomised algorithm $\mathcal{M}$ is (ε, δ)-differentially private if, for any two neighbouring databases $D \approx D'$ and any subset of possible outputs S,*

$$\Pr[\mathcal{M}(D) \in S] \leq e^{\varepsilon} \cdot \Pr[\mathcal{M}(D') \in S] + \delta\,, \tag{2.1}$$

where $\varepsilon > 0$ and $\delta \geq 0$. When $\delta = 0$, we say that $\mathcal{M}$ is ε-differentially private.

This thesis considers only ε-differential privacy, also known as *pure* differential privacy, as first introduced by Dwork et al. [24] and formalised by Dwork [20]. We give the full definition of (ε, δ)-differential privacy, first introduced by Dwork et al. [23], also known as *approximate* differential privacy when $\delta > 0$, for ease of reference.

Neighbouring databases are defined with respect to the specific privacy goal of a mechanism. There are two widely accepted definitions of neighbouring databases, for mechanisms designed to preserve the privacy of single individuals, in the static setting. In the *unbounded neighbour* model, a single record is added or deleted, while the *bounded neighbour* model allows a single record to be substituted.

Definition 2.1.2 (Unbounded neighbours). *Static databases $D \in \mathcal{X}^n$ and $D \in \mathcal{X}^{n\pm1}$ are unbounded neighbours, denoted $D \approx_{\pm} D'$ if there is some $i \in [n]$ such that $D' = D/D[i]$, or $D = D'/D'[i]$, or, the add/delete distance $d_{\pm}(D, D') = 1$. Here, the add/delete distance $d_{\pm}$ is the edit distance when only additions and deletions are allowed. That is, $d_{\pm}(D, D') = x$ if and only if D' can be derived from D by performing a sequence of x additions and deletions.*

Definition 2.1.3 (Bounded neighbours). *Static databases $D, D' \in \mathcal{X}^n$ are bounded neighbours, denoted $D \approx_{\circlearrowleft} D'$ if, for some $j \in [n]$, $D[i] = D'[i]$ for all $i \in [n] \setminus j$, or the Hamming distance $d_H(D, D') = 1$.*

In general, databases are neighbouring if $d(D, D') \leq 1$ for a given distance measure d. To prove privacy guarantees, the neighbour definition used must be taken into account. For example, converting directly from the bounded to unbounded neighbour model requires a definition of privacy where $d(D, D') > 1$, since $d_H(D, D') = 1 \implies d_{\pm}(D, D') = 2$ because changing the value of a record requires both a deletion and an addition.

Throughout the thesis we will refer to the sensitivity, Δ, shorthand for the ℓ_1-sensitivity, of queries and functions.

Definition 2.1.4 (ℓ_1-sensitivity, from Definition 2 of Dwork et al. [24]). *The ℓ_1-sensitivity, Δ_f, of a function $f : \mathcal{X}^n \to \mathbb{R}^k$, outputting a vector of length k, is:*

$$\Delta_f = \max_{D \approx D'} \|f(D) - f(D')\|_1 = \max_{D \approx D'} \left(\sum_{i=1}^{k} |(f(D))[i] - (f(D'))[i]| \right).$$

That is, the sensitivity Δ_f is the maximum ℓ_1-norm distance between the function f applied to any two neighbouring databases. For example, take a discrete universe $\mathcal{X}$ and function counting the frequency of a single value, that is $\text{count}(D, a) := |\{i : D_i = a\}|$ for some $a \in \mathcal{X}$. The ℓ_1-sensitivity $\Delta_{\text{count}} = 1$, under both the bounded and unbounded neighbour definition, since adding, deleting, or changing a single database element cannot change the frequency of a single value by more than ± 1.

We will also be interested in describing the accuracy of numeric outputs of mechanisms in terms of (α, β)-accuracy.

Definition 2.1.5 (Numeric accuracy, adapted from Definition 3.10 of Dwork and Roth [22])**.** *A mechanism $\mathcal{M}$ is (α, β)-accurate with respect to function f if:*

$$\Pr\left[|\mathcal{M}(D) - f(D)| \geq \alpha\right] \leq \beta\,.$$

That is, $\mathcal{M}$ is (α, β)-accurate if, with probability at least $1 - \beta$, an output of $\mathcal{M}$ is within $\pm\alpha$ of the output of the function, f, applied to D.

2.1.3 Key results

Now we describe key results from the static DP literature that are relevant to the thesis.

Group privacy. Consider the case where the privacy of either an individual whose data appears multiple times in a database, or for multiple correlated individuals, such as a family, must be protected. This cannot be guaranteed by the standard previous definitions, which protect only an individual record in the database. *Group privacy*, first described by Dwork et al. [24], defines differential privacy for databases where up to k correlated records appear.

Theorem 2.1.1 (Group privacy, Theorem 2.2 of Dwork and Roth [22])**.** *Let $\mathcal{M}$ be an ε-differentially private algorithm under Definition 2.1.1. Then, for a group size k, $\mathcal{M}$ is $k\varepsilon$-differentially private.*

k-neighbours are databases $D \approx^k D'$ where, for some k, $d(D, D') \leq k$. Theorem 2.1.1 allows a guarantee of ε-differential privacy for k-neighbouring databases using a mechanism that is (ε/k)-differentially private for 1-neighbouring databases.

Post-processing. Once a private output is received, an analyst or adversary may wish to apply transformations to the output. There is a simple theorem that guarantees the privacy of such transformations of the output of a differentially private mechanism $\mathcal{M}$.

Theorem 2.1.2 (Post-processing, Proposition 2.1 of Dwork and Roth [22]). *Let $\mathcal{M}$ be a randomised algorithm that is ε-differentially private. Any arbitrary randomised mapping on the output of $\mathcal{M}$ is ε-differentially private.*

That is, any transformations to a differentially private output, provided they do not require additional access to the private information, preserve differential privacy guarantees. For example, rounding the output of a differentially private mechanism does not diminish its privacy guarantee.

Composition. Now consider mechanisms that require multiple, sequential accesses to a private database. This is known as *composition.* Differential privacy is proven to be maintained under composition, with privacy loss growing linearly with the number of queries for pure DP.

Theorem 2.1.3 (Basic composition, Corollary 3.16 of Dwork and Roth [22]). *Let $\mathcal{M}_{[k]} = (\mathcal{M}_1(D), ..., \mathcal{M}_k(D))$ be a sequence of $(\varepsilon_i, \delta_i)$-differentially private randomised mechanisms, for $i \in \{1, ..., k\}$. Then $\mathcal{M}_{[k]}$ is (ε', δ')-differentially private where $\varepsilon' = \sum_{i=1}^{k} \varepsilon_i$ and $\delta' = \sum_{i=1}^{k} \delta_i$.*

Basic composition allows the composition of multiple differentially private mechanisms applied to a single database [24]. Dwork et al. [28] further show that basic composition can be extended to the *k-fold adaptive* case, where the input to each of the k mechanisms may be a different database, and the mechanisms and databases chosen adaptively. For example, if an ε_1-differentally private mechanism is performed on a census database, then an ε_2-differentially private mechanism is performed on a tax database for the same country and year, $(\varepsilon_1 + \varepsilon_2)$-differential privacy is preserved for any individual appearing in both databases.

The efficiency of composition can be improved for approximately differentially private mechanisms, that is, (ε, δ)-differentially private with $\delta > 0$ [28]. This is known as *advanced composition.* Since the mechanisms presented in this thesis preserve pure differential privacy, we do not present the advanced composition theorem here.

2.1.4 The Laplace mechanism

Using the formal definition of differential privacy, mechanisms can be designed for privately answering queries on databases, with provable privacy guarantees. This section describes some key mechanisms for answering single queries on static data, based on the first differentially private mechanism described by Dwork et al. [24], the Laplace mechanism. This mechanism provides differentially private, noisy answers to queries with real-valued answers using noise drawn from the Laplace distribution.

Definition 2.1.6 (The Laplace mechanism, from Definition 3.3 of Dwork and Roth [22])**.** *Given a function $f : \mathcal{X}^n \to \mathbb{R}^k$, the Laplace Mechanism, $\mathcal{M}_{Lap}$, is defined as:*

$$\mathcal{M}_{Lap}(D, f, \varepsilon) = f(D) + (L_1, ..., L_k)\,,$$

where each $L_i \sim \text{Laplace}(0, \Delta_f/\varepsilon)$.

The Laplace mechanism is ε-differentially private, and has accuracy given in Theorem 2.1.4.

Theorem 2.1.4 (Accuracy of the Laplace mechanism, from Theorem 3.8 of Dwork and Roth [22])**.** *The Laplace Mechanism is (α, β)-accurate for:*

$$\alpha = \frac{\Delta_f}{\varepsilon} \ln \frac{k}{\beta}\,.$$

An ε-differentially private mechanism for answering count queries can be given by:

$$\mathcal{M}_{\text{count}}(D, c) := \text{count}(D, c) + \text{Lap}\left(\varepsilon^{-1}\right)\,,$$

where $\text{Lap}\,(a)$ is shorthand for $L \sim \text{Laplace}(0, a)$. This shorthand will be used throughout. By Theorem 2.1.4, $\mathcal{M}_{\text{count}}$ is $(\varepsilon^{-1}\ln(\beta^{-1}), \beta)$-accurate. For example, with $\varepsilon := 1$, $\beta := 0.05$, $\mathcal{M}_{\text{count}}(D, a)$ will give an answer within $\pm \ln 0.05^{-1} \approx 3$ of the true answer $\text{count}(D, a)$ with at least 95% probability. There is a clear trade-off between privacy and accuracy, with a linear relationship between α and ε. This becomes particularly important when composing mechanisms, where bounding the total privacy loss limits the achievable overall accuracy.

If all counts are of interest, rather than an answer to a single counting query, the Laplace Mechanism can be used to output a synthetic histogram. Using the composition theorem, a naive method would be to release N count queries, each with privacy budget ε/N. This approach would have large noise, proportional to the size of the histogram. Better accuracy can be achieved, without loss to privacy, by using non-private information about the nature of the histogram. Since each individual data point occurs in only one histogram bucket, the number of buckets differing between neighbouring databases can be bounded, depending on the neighbour definition used.

Definition 2.1.7 (The Laplace perturbation algorithm for histograms, from Example 3.2 of Dwork and Roth [22])**.** *Given a database $D \in \mathcal{X}^n$, with corresponding histogram $x \in \mathbb{N}^N$, the Laplace perturbation algorithm, $\mathcal{M}_{LPH}$, outputs $\mathcal{M}_{LPH}(D) = \hat{x}$ where:*

$$\hat{x}[i] = x[i] + Lap\left(\frac{\Delta_x}{\varepsilon}\right).$$

The value of Δ_x depends on the privacy definition. For unbounded privacy $\Delta_x = 1$ and for bounded $\Delta_x = 2$.

For two bounded-neighbouring databases $D \approx_{\circlearrowleft} D'$ in $\mathcal{X}^n$, a single user is counted in a different histogram bucket, so at most two histogram buckets will differ, and $\Delta_x = 2$. Thus, by the properties of the Laplace mechanism, noise drawn from $\text{Lap}\,(2/\varepsilon)$ can be added to produce a noisy version ($\hat{x}$), preserving ε-differential privacy, with (α, β)-accuracy guaranteed for the count of any single bucket for $\alpha = (2/\varepsilon)\ln(\beta^{-1})$, and for any counting query over the histogram with $\alpha = (2/\varepsilon)\ln(N/\beta)$. For unbounded neighbours $D \approx_{\pm} D'$, instead $\Delta_x = 1$, so the accuracy improves to give $\alpha = \varepsilon^{-1}\ln(N/\beta)$.

The synthetic histogram produced by the Laplace perturbation algorithm can be used to answer any other queries on the histogram. For example, Dwork and Roth [22] give one implementation of a *report noisy max* algorithm that takes the largest value from the synthetic histogram to give an approximate answer to the question of which histogram bucket has the largest count.

Definition 2.1.8 (Report noisy max - Laplace, from Section 3.3 of Dwork and Roth [22])**.** *Given a database $D \in \mathcal{X}^n$, with corresponding histogram $x \in \mathbb{N}^N$, the report noisy max (Laplace) algorithm $\mathcal{M}_{LapNM}$ outputs:*

$$\mathcal{M}_{LapNM}(D) = \arg\max_i \left(x[i] + Lap\left(\frac{\Delta_x}{\varepsilon}\right)\right).$$

The value of Δ_x depends on the privacy definition. For unbounded privacy $\Delta_x = 1$ and for bounded $\Delta_x = 2$.

In practice, mechanisms using noise drawn from continuous distributions, such as the Laplace mechanism, are susceptible to side-channel attacks. Mironov [57] shows how the implementation of floating point numbers allows privacy breaches on the Laplace mechanism and propose mechanism to correct these breaches while still drawing noise from the Laplace distribution. Discrete alternatives to the Laplace mechanism, with similar theoretical privacy and accuracy guarantees, such the discrete Laplace mechanism [39] and discrete Gaussian mechanism [5], avoid these floating-point vulnerabilities but may still be vulnerable to side-channel attacks. Jin et al. [46] show that implementations of these discrete methods are vulnerable to timing attacks, and propose mitigations against their attacks. Since the focus of this thesis is in providing theoretical privacy guarantees rather than robust implementations, the algorithms presented use the continuous version of the Laplace mechanism.

There are scenarios where the addition of random noise can drastically reduce utility. For example, consider that potential buyers have indicated the maximum price they are willing to pay for a product, and an optimal price must be chosen; positive noise added to the optimal price may result in a price above every purchaser's maximum. In these cases, the Laplace mechanism in inappropriate, so alternative mechanisms must be used to preserve differential privacy. The exponential mechanism McSherry and Talwar [55] does this by selecting values probabilistically according to their relative utility. In this thesis we do not consider these situations, so the details of the exponential mechanism are not presented here.

2.1.5 Differentially private query sequences

Due to the degradation in privacy and/or utility required for differentially private composition, answering many queries in a differentially private manner can quickly become impractical. For certain types of queries, however, a differentially private data structure can be released that can then, thanks to the post-processing theorem (Theorem 2.1.2), be used to answer any number of queries without any additional loss to privacy. When only query answers meeting some threshold requirement are of interest, we can use the sparse vector technique (SVT) to conserve the privacy budget for those above threshold queries.

The sparse vector technique (SVT) is an algorithm which allows a sequence of interactive queries to be made on a database at a reduced cost to privacy and accuracy over simple composition. SVT is a generalisation of the 'indicator vector' introduced by Dwork et al. [25] to construct sanitised datasets, which was extended by Roth and Roughgarden [62] in their 'median mechanism' and Hardt and Rothblum [42] in their 'private multiplicative weights' (PMW) mechanism. Early formalisations of SVT as a standalone algorithm were given in lectures by Roth [61], and in the textbook by Dwork and Roth [22].

There are two standard forms of SVT in the literature. Here we will call them SVT and NumericSVT. The first provides (noisy) responses to whether or not each query answer lies above some given threshold. NumericSVT, derived from PMW [42], utilises composition theorems to give noisy numeric answers to the above threshold queries.

Algorithm 1, SVT, takes a database D, a constant c, and answers a stream of k queries $Q = \{q_i\}$ with thresholds $\Theta = \{\theta_i\}$. q_i is answered with $\top$ if a noisy answer to the query is above a noisy threshold $\hat{\theta}_i$ and $\perp$ if below, until either there have been c above threshold answers, or all k queries are answered. The surprising and powerful result here is that it is only the above threshold queries that degrade the privacy, so only their number must be bounded by c, allowing an unbounded number of below threshold queries to be answered. SVT has been shown to be ε-DP [22].

Algorithm 1 SVT (Algorithm 2 of Dwork and Roth [22])

```
Input: Database $D$, interactive query stream $\{q_i\}$, $i = 1, 2, ..., k$, sensitivity $\Delta_q$, threshold
       $\theta$, cutoff $c$, privacy parameter $\varepsilon$
Output: Stream $a \in \{\perp, \top\}^{\leq k}$
 1: $\sigma \leftarrow 2c\Delta_q/\varepsilon$                         ▷ Set noise scale function
 2: $\hat{\theta} \leftarrow \theta + \mathrm{Lap}(\sigma)$
 3: $w \leftarrow 0$
 4: for each round $i$ do
 5:     if $q_i(D) + \mathrm{Lap}(2\sigma) \geq \hat{\theta}$ then             ▷ above threshold
 6:         output $a_i \leftarrow \top$
 7:         $w \leftarrow w + 1$
 8:         $\hat{\theta} \leftarrow \theta + \mathrm{Lap}(\sigma)$           ▷ Resample threshold noise
 9:     else output $a_i \leftarrow \perp$                       ▷ below threshold
10:     if $w \geq c$ then Halt                     ▷ Exceeded hard query budget
```

Algorithm 2, NumericSVT, is very similar to SVT, except that rather than outputting $\top$ for above threshold queries, it outputs a noisy version of the numeric query answer.

This is achieved using composition: SVT is built by composing a simpler algorithm, called 'AboveThreshold' (Algorithm 1) in the Dwork and Roth book [22], that halts after a single hard query, with itself. By giving an independently noisy numeric answer after a hard query, NumericSVT is the composition of AboveThreshold and the Laplace mechanism. Throughout, we will refer to the sections of NumericSVT that are equivalent to SVT as the *decision* step, as they are what decides whether to output a numeric answer, and the noisy numeric answer itself as the *sample* step.

Algorithm 2 NumericSVT (Algorithm 3 of Dwork and Roth [22])

```
Input: Database $D$, interactive query stream $\{q_i\}$, $i = 1, 2, ..., k$ sensitivity $\Delta_q$, threshold
       $\theta$, cutoff $c$, privacy parameter $\varepsilon$
Output: Stream $a \in \{\perp, \mathbb{R}\}^{\leq k}$
 1: $\sigma_1 \leftarrow 9c\Delta_q/(4\varepsilon)$, $\sigma_2 \leftarrow 9c\Delta_q/\varepsilon$          ▷ Set noise scale function
 2: $\hat{\theta} \leftarrow \theta + \mathrm{Lap}(\sigma_1)$
 3: $w \leftarrow 0$
 4: for each round $i$ do
 5:     if $q_i(D) + \mathrm{Lap}(2\sigma_1) \geq \hat{\theta}$ then                      ▷ above threshold
 6:         output $a_i \leftarrow q_i(D) + \mathrm{Lap}(\sigma_2)$
 7:         $w \leftarrow w + 1$
 8:         $\hat{\theta} \leftarrow \theta + \mathrm{Lap}(\sigma_1)$                          ▷ Resample threshold noise
 9:     else output $a_i \leftarrow \perp$                                      ▷ below threshold
10:     if $w \geq c$ then Halt
```

A numeric accuracy guarantee for private query sequences requires a modified definition of the accuracy bound on numeric streams.

Definition 2.1.9 (Numeric accuracy of streams, from Definition 3.10 of Dwork and Roth [22])**.** *An algorithm which outputs a stream of answers $a_1, ..., a_k \in (\mathbb{R} \cup \{\perp\})*$ in response to a stream of k queries $q_1, ..., q_k$ is (α, β)-accurate if, except with probability at most β, the algorithm does not halt before q_k, and for all $a_i \in \mathbb{R}$:*

$$|q_i(D) - a_i| \leq \alpha \,,$$

and, if the outputs are in respect to a stream of thresholds $\Theta = \{\theta_i\}$, for all $a_i = \perp$:

$$q_i(D) \leq \theta_i + \alpha \,.$$

Dwork and Roth [22] show that NumericSVT is ε-DP and (α, β)-accurate for

$$\alpha = O\left(\frac{c \log\left(\frac{ck}{\beta}\right)}{\varepsilon}\right).$$

SVT, especially NumericSVT, has been incorrectly applied since some of its earliest uses. Chen and Machanavajjhala [11] proved that a specific extension of SVT not requiring a limit, c, on the number of hard queries does not satisfy differential privacy. Lyu et al. [53] further showed that early versions of NumericSVT, including those used by Hardt and Rothblum [42] and given in Roth's lecture notes [61] do not satisfy differential privacy because the noise used in the sample step is not independent of the noise used on the query answer in the decision step.

Both the SVT and NumericSVT mechanisms given in Dwork and Roth's book [22] are differentially private, however Lyu et al. [53] show that the algorithm can be made more accurate, without a loss to privacy, by not resampling the threshold noise after each query. This version is given in Algorithm 3. Lyu et al. [53] also optimise the ratio of the decision privacy budget allocated to the threshold and query ($\varepsilon_{d1} : \varepsilon_{d2}$) for NumericSVT. This requires an additional parameter, d, indicating what fraction of the privacy budget, ε, should be dedicated to the *decision* step of the algorithm. While the versions of SVT given by Lyu et al. [53] have better accuracy than those given by Dwork and Roth [22],

Algorithm 3 Optimised NumericSVT (Algorithm 7 of Lyu et al. [53])

```
Input: Database $D$, interactive query stream $Q = \{q_i\}$ and threshold stream $\Theta = \{\theta_i\}$
       for $i = 1, 2, ..., k$, maximum query sensitivity $\Delta_q$, cutoff $c$, privacy parameter $\varepsilon$
       and decision fraction $0 < d < 1$.
Output: stream $a \in \{\perp, \mathbb{R}\}^{\leq k}$
 1: $\varepsilon_d \leftarrow d\varepsilon$, $\varepsilon_s \leftarrow (1-d)\varepsilon$
 2: if every $q_i$ is monotone then $\varepsilon_{d1} \leftarrow \varepsilon_d/(1 + c^{2/3})$
 3: else $\varepsilon_{d1} \leftarrow \varepsilon_d/(1 + (2c)^{2/3})$
 4: $\varepsilon_{d2} \leftarrow \varepsilon_d - \varepsilon_{d1}$
 5: $\rho \leftarrow \text{Lap}(\Delta_q/\varepsilon_{d1})$
 6: $w \leftarrow 0$
 7: for each round $i$ do
 8:     if $q_i(D) + \text{Lap}(2c\Delta_q/\varepsilon_{d2}) \geq \theta_i + \rho$ then      ▷ above threshold
 9:         output $a_i \leftarrow q_i(D) + \text{Lap}(c\Delta_q/\varepsilon_s)$
10:         $w \leftarrow w + 1$
11:     else output $a_i \leftarrow \perp$                                        ▷ below threshold
12:     if $w \geq c$ then Halt
```

their proofs do not account for adaptive query streams. As such, we use Algorithm 2 later in this thesis for theoretical results, and Algorithm 3 for experimental comparisons.

The private multiplicative weights algorithm (PMW), introduced by Hardt and Rothblum [42], with further details in Hardt's thesis [41], is the first use of a numeric SVT. It allows a combination of the powerful results of SVT with the machine learning technique multiplicative weights (MW) to provide numeric, differentially private answers for a stream of interactive linear queries, with fixed accuracy parameters α, β. Throughout this chapter, mechanisms take an input database as a count histogram x.

Definition 2.1.10 (Linear query, as described in Chapter 4 of Dwork and Roth [22])**.** *Let D be a database drawn from a discrete universe $\mathcal{X}$ of size $|\mathcal{X}| = N$, represented by a histogram $x \in \mathbb{N}^N$. Let $w \in \mathbb{R}^N$ be a vector of weights. A linear query, q, is a weighted sum of histogram bucket counts:*

$$q(x) = w \cdot x = \sum_{i=1}^{N} w[i]x[i]\,.$$

If every $w[i] \in \{0, 1\}$ this is called a counting query.

PMW relies on releasing a public histogram to approximate the true database, and allows queries that do not differ significantly from the public histogram to be answered with no loss of privacy, using post-processing [25]. Due to some privacy leakage issues in the original algorithm, demonstrated by Lyu et al. [53], we present the version of the algorithm from Dwork and Roth [22].

Algorithm 4 performs PMW by first setting the public histogram, y, to a uniform distribution across $\mathcal{X}$ (line 3). It then makes repeated calls to NumericSVT (initialised line 4, input functions $\{q_i'\}$ given in line 6, SVT outputs a' used in lines 7, 11, 12, and 15) to check whether the (noisy) difference between the true database and the public histogram (line 6) lies above some threshold, θ, determined by the parameters (line 2). If it does, it both releases a noisy answer (lines 12 and 15) and updates the public histogram to re-weight the rows (lines 17-18) based on the result of the calls to NumericSVT. The update shifts the public histogram, y, closer to the true histogram x. Below threshold queries are answered based only on the public histogram (line 8). Since this algorithm accesses the true database only through calls to the differentially private NumericSVT, it is also differentially private, with the privacy parameters of the subroutine determined using the composition theorem.

Algorithm 4 PMW (Algorithm 6 of Dwork and Roth [22], adapted from Figure 1 of Hardt and Rothblum [42])

```
Input: Histogram $x$, with $N$ buckets, representing $n$ individuals, interactive query stream
       $Q = \{q_i\}$ where each $q_i \in \mathcal{Q}$, $i = 1, 2, ...$, cutoff $c$, privacy parameter $\varepsilon$, accuracy
       parameters $\alpha, \beta$, and weight parameter $\eta \leq 1$
Output: Stream $a = \{a_i\} \in \mathbb{R}$, public histograms $Y = \{y_i\}$
 1: $c \leftarrow 4 \log N/\alpha^2$
 2: $\theta \leftarrow (18c \log(2|\mathcal{Q}|) + \log(4c/\beta))/\varepsilon$
 3: $y_0 \leftarrow [n/N]^N$                                              ▷ synthetic database y
 4: Initialise Subroutine NumericSVT$(x, \{q'_i\}, \theta, c, \varepsilon) \to a'$      ▷ halt if subroutine halts
 5: for each round $i \leftarrow 1, 2, ...$ do
 6:     $q'_{2i-1} \leftarrow q_i(x) - q_i(y_{i-1})$, $q'_{2i} \leftarrow q_i(y_{i-1}) - q_i(x)$
 7:     if $a'_{2i-1} = \perp$ and $a'_{2i} = \perp$ then                          ▷ below threshold
 8:         output $a_i \leftarrow q_i(y_{i-1})$
 9:         $y_i \leftarrow y_{i-1}$
10:     else                                                      ▷ above threshold, hard query
11:         if $a'_{2i} = \perp$ then
12:             output $a_i \leftarrow q_i(y_{i-1}) + a'_{2i-1}$
13:             $r_i \leftarrow 1 - q_i$                             ▷ increases weight of query elements
14:         else
15:             output $a_i \leftarrow q_i(y_{i-1}) - a'_{2i}$
16:             $r_i \leftarrow q_i$
17:         for each $i \in N$ do $\hat{y}_i[i] \leftarrow y_i[i] \cdot \mathrm{e}^{-\eta r_i[i]}$                 ▷ re-weight y
18:         for each $i \in N$ do $y_i[i] \leftarrow \hat{y}_i[i] / \sum_{j=1}^{N} \hat{y}_i[j]$              ▷ normalise y
```

The accuracy of PMW relies on the relative entropy between the true and synthetic databases x and y. The relative entropy of two databases is given using the Kullback-Leibler (KL) divergence [51].

Definition 2.1.11 (Relative entropy, A.K.A KL-divergence, as given in Section 2.4 of Cummings et al. [16])**.** *The relative entropy of two databases, represented as histograms x and y over some discrete, finite domain $\mathcal{X}$ of size $|\mathcal{X}| = N$, is given by:*

$$RE(x \| y) = \sum_{i=1}^{N} x[i] \log \frac{x[i]}{y[i]} .$$

PMW halts if the number of hard queries to the NumericSVT subroutine exceeds a value c, determined by the size of the data universe $\mathcal{X}$ and the accuracy parameter α. Dwork and Roth [22] show that PMW is (α, β)-accurate, for fractional histograms, when

$$\alpha \geq O\left(n^{2/3}\left(\frac{\log N \log\left(\frac{\log(N^{1/3}n^{2/3})}{\beta}\right)}{\varepsilon}\right)^{1/3}\right).$$

If this bound is met, the probability of halting before all queries are answered is $\leq \beta$ due to the reduction in relative entropy expected from the multiplicative weights updates [42].

2.2 Differential privacy for dynamic data

So far this chapter has focused on differential privacy for some single, static, database. In practice, most data has some dynamic property, where entries are being added, removed or changed over time. In this section we describe various existing models for differential privacy over such databases, where time is broken up into discrete intervals. Our taxonomy of the various dynamic database models addressed in the differential privacy literature is original, though we have stayed as consistent as possible with accepted terminology.

The first model is a simple dynamic database, a bitstream, whereby a single bit is added to a database, represented as a bitstring, at each time t. This can be generalised to streams of categorical data, where each entry is drawn from a finite universe $\mathcal{X}$, rather than $\{0, 1\}$, we call this a *growing* database. We then move on to databases where the data cannot be represented as a string, firstly the *time-series* model, frequently used for trajectory data, each individual in the dataset takes some value from $\mathcal{X}$ at each time t. That is, there is a set of growing strings, one for each individual. Finally, we describe databases where only one individual is updated at each time, but where past entries can be deleted or over-ridden. Table 2.1 compares these properties across the key papers mentioned in this section, and in Section 3.1.3 we compare these models directly to our proposed FSDD model.

Comparing these models requires some additional terminology. First, dynamic databases may or may not have a *time bound*; if the number of updates is limited by some number T, we will call it T-bounded, otherwise we call it time-unbounded. Second, there is additional diversity in what it means for database streams to be neighbouring. Relevant neighbour definitions will be described in each of the following sections.

	Paper	**Update types**			**Time bound**		**Users per t**		**Privacy level**		
		$+$	$-$	$\circlearrowleft$	T	∞	1	$+$	E	U	w
BS	Dwork et al., 2010 [26]	✓			✓		✓		✓	✓	
	Chan et al., 2011 [9]	✓			✓	✓	✓		✓		
Growing	Dwork et al., 2010 [27]	✓				✓	✓			✓	
	Jain et al., 2012 [45]	✓			✓		✓		✓		
	Cummings et al., 2018 [15]	✓				✓	✓		✓		
	Perrier et al., 2019 [58]	✓			✓		✓		✓		
	Yıldırım et al., 2020 [69]	✓				✓	✓		✓		
	Wang et al., 2021 [66]	✓				✓	✓		✓		
	Cardoso and Rogers, 2022 [8]	✓			✓			✓	✓		
Time-series	Kellaris et al., 2014 [49]			✓		✓		✓			✓
	Fan et al., 2014 [34]			✓	✓			✓		✓	✓
	Cao and Yoshikawa, 2015 [6]			✓		✓		✓			✓
	Li et al., 2015 [52]			✓	✓	✓		✓		✓	✓
	Cao et al., 2017 [7]			✓	✓			✓	✓		
	Chen et al., 2017 [12]			✓		✓		✓			✓
FD	Mir et al., 2011 [56]			✓	✓		✓			✓	
	Qiu and Yi, 2022 [59]	✓	✓		✓	✓	✓		✓		

The models listed are categerised as bitstream (BS), growing database, time-series, and fully dynamic (FD), corresponding with subsections of this section. Update types indicate whether an update can add $(+)$, delete $(-)$, or modify the value of $(\circlearrowleft)$ an element. The number of updates is either bounded by T or unbounded. The privacy guaranteed is at *event* (E) or *user* (U) level, or, for some unbounded update cases, *sliding window* (w).

Table 2.1: Properties of dynamic databases used in various key papers mentioned in Section 2.2.

2.2.1 Differential privacy for growing databases

The bitstream model. Dwork et al. [26, 27] and Chan et al. [9] independently introduced similar early models for differential privacy in a dynamic context. In Dwork et al. [26] and Chan et al. [9], the dynamic data is a *bitstream*, modelled as a sequence of prefixes $S = S_1, S_2, ..., S_T$ for times $t = 1, 2, ..., T$, where each $S_t = 0, 1^t$. Bitstreams S, S' are neighbours if, for some time k, $S_i = S'_i$ for $i < k$ and $S_i \approx S'_i$ for $i \geq k$. In other words, the two bitstreams differ at exactly one position, k. Differential privacy over bitstreams is defined by applying Definition 2.1.1 to this new definition of neighbouring bitstreams. Figure 2.1 shows an example of two neighbouring bitstreams.

Definition 2.2.1 (Neighbouring database streams)**.** *Database streams S and S' are event-level neighbours if they may differ by the update at single time t [9, 26]; they are user-level neighbours if they may differ at multiple updates, if those updates all pertain to the same individual [26]. Where time is unbounded, unconstrained user-level privacy can* *be particularly difficult to achieve [19]. Here, a sliding-window neighbour model may be employed, whereby either event- or user-level privacy is guaranteed only for subsequences of length w of the update stream [49].*

$\mathbf{S_1}$	1						
$\mathbf{S_2}$	1	0					
...	1	0	...				
$\mathbf{S_{t-1}}$	1	0	...	1			
$\mathbf{S_t}$	1	0	...	1	0		
$\mathbf{S_{t+1}}$	1	0	...	1	0	0	
...	1	0	...	1	0	0	...

$\mathbf{S'_1}$	1						
$\mathbf{S'_2}$	1	0					
...	1	0	...				
$\mathbf{S'_{t-1}}$	1	0	...	1			
$\mathbf{S'_t}$	1	0	...	1	1		
$\mathbf{S'_{t+1}}$	1	0	...	1	1	0	
...	1	0	...	1	1	0	...

Figure 2.1: Example of a bitstream D and an event-level neighbouring bitstream D'. Yellow cells indicate the effect of updates on the bitstrings and blue cells indicate where neighbours differ.

Both Dwork et al. [26] and Chan et al. [9] provide differentially private counting algorithms under this model. Dwork et al. [26] also provide a general transformation from the static to the growing database model for differentially private, monotonically increasing functions, while Chan et al. [9] extend their counting algorithm to the time-unbounded setting. In Theorem 2.2.1 gives a general lower bound for the error differentially private counting in the growing database model, as shown by Dwork et al. [26].

Theorem 2.2.1 (Theorem 4.2 of Dwork et al. [26])**.** *Any differentially private event-level algorithm for counting on a growing database over T rounds must have error $\Omega(\log T)$.*

The algorithms and some lower bounds provided by Dwork et al. [26, 27] abide by *pan-privacy*, a stronger privacy guarantee whereby not only are outputs differentially private, but the central database itself is protected against a small fixed number of intrusions by an adversary. This limits the power of the algorithms as the database itself cannot be maintained in its true state.

Since the growing database can be modelled as a special case of all other dynamic database models, and event-level privacy is weaker than alternative privacy definitions, this bound applies to counting mechanisms across all dynamic database models presented. Later work by Dwork et al. [29] provides new mechanisms, including an optimal counting mechanism, without the pan-privacy restriction.

The strictly growing database model. The bitstream model described in Section 2.2.1 is generalised by extending the data universe, from bits, to some discrete universe $\mathcal{X}$ of size $|\mathcal{X}| = N$. We refer to this as the *strictly growing database* model, and give an example in Figure 2.2.

$\mathbf{D_1}$	7						
$\mathbf{D_2}$	7	3					
...	7	3	...				
$\mathbf{D_{t-1}}$	7	3	...	1			
$\mathbf{D_t}$	7	3	...	1	4		
$\mathbf{D_{t+1}}$	7	3	...	1	4	3	
...	7	3	...	1	4	3	...

$\mathbf{D'_1}$	7						
$\mathbf{D'_2}$	7	3					
...	7	3	...				
$\mathbf{D'_{t-1}}$	7	3	...	1			
$\mathbf{D'_t}$	7	3	...	1	7		
$\mathbf{D'_{t+1}}$	7	3	...	1	7	3	
...	7	3	...	1	7	3	...

Figure 2.2: Example of a growing database D and an event-level neighbouring database D' that differ at time t. Yellow cells indicate the effect of updates on the databases and blue cells indicate where neighbours differ.

Cummings et al. [15] use a strictly growing model, with time-unbounded updates, and provide general transformations for differentially private algorithms from the static database setting. They define a stream of databases as a sequence $S = \{D_1, D_2, ...\}$, where at initial time $t = n$, $D_n \in \mathcal{X}^n$, and at each subsequent time step $t > n$ the database grows by one element. As such, in their setting, $D_t \in \mathcal{X}^t$ at every time step, and $D_t[i] = D_{t-1}[i]$ for all $i \in [1..t-1]$. Similar models are used by Yıldırım et al. [69] and Cardoso and Rogers [8].

Under the strictly growing database model, it is assumed that each entry is independent and that entries remain static once they are added. For example, the prefixes in the bitstreams of Chan et al. [9], Dwork et al. [26] do not allow previously streamed bits to be flipped, and the growing databases of Cummings et al. [15] do not allow a record added at time (and index) t to be modified after time t. These models allow composition theorems, such as that in Theorem 2.1.1, to be applied to prove privacy guarantees. For example by setting ε_i values for each query such that their sum over all queries does not exceed some global ε. In order to describe the accuracy of algorithms in the growing database model, a new definition of numeric accuracy is required, given in Definition 2.2.2.

Definition 2.2.2 (Numeric accuracy for growing databases, from Definition 2 of Cummings et al. [15])**.** *For $\alpha_n, \alpha_{n+1}, ... > 0$, a randomised algorithm $\mathcal{M}$ is $(\{\alpha_t\}_{t\geq n}, \beta)$-accurate for query stream $Q = \{q_{t,:}\}_{t\geq n}$ if for any input data stream $X = \{x_t\}_{t\geq n}$ the algorithm outputs y such that $|q_{t,j}(x_t) - q_{t,j}(y_t)| \leq \alpha_t$ for all $q_{t,j} \in Q$ with probability at least $1 - \beta$.*

Cummings et al. [15] use an event-level privacy definition: they define database streams S, S' as neighbouring if, for some t, $D_\tau = D'_\tau$ for $\tau = 1, ..., t-1$ and $D_\tau \approx_{\circlearrowleft} D'_\tau$ for $\tau = t, t+1, ...$ Specifically, $S \approx S'$ if there is some $j = t$ such that, for all τ, $D_\tau[i] = D'_\tau[i]$

for all $i \neq j$. The authors provide a PMW algorithm for growing databases, as well as two general transformations from the static database setting to the strictly growing database setting. One of the general transformations has an accuracy guarantee that is fixed over time, and the other that improves over time. Their PMW algorithm for growing databases, PMWG, utilises the fact that the sensitivity of linear queries, relative to the size of the database, decreases as data accumulates. This allows them to guarantee privacy by scaling their privacy parameter with time, and to achieve (α, β)-accuracy, when

$$\alpha \geq O\left(\left(\frac{\log Nn \log n/\beta}{n\varepsilon}\right)^{1/3}\right),$$

which is less accurate than the static version by around a factor of $O(\log n)$.

2.2.2 Differential privacy for time-series data

The growing database models presented in previous sections allow a single record to be added at each time step. This section describes models where every individual can send an update at every time step. These models have primarily evolved around spatio-temporal data, though they can apply to any time-series data.

Definition 2.2.3 (Discrete Time Series Database)**.** *A time series database can be represented as a table with n rows, representing users, and columns $t = 1, 2, ...$ representing discrete timestamps. At each time stamp t, users may send an update to the database with some discrete value $x \in \mathcal{X}$ representing their state, or change in state, during the period $(t-1, t]$. If no update is sent by user i at time t, it is presumed that user i has maintained their previous state during the corresponding time period.*

Definition 2.2.3 broadly captures the database definitions variously described as 'time-series', 'longitudinal', and '(spatio-)temporal' in differential privacy literature. The privacy definitions vary, corresponding to event-, user-, and sliding-window-level privacy and both bounded and unbounded neighbour definitions. Figure 2.3 shows an example of a T-bounded time-series database under this definition.

Rastogi and Nath [60] first introduced differential privacy for time-series data. Kellaris et al. [49] first formalised a model similar to the one presented here, along with a new privacy definition for infinite streams, w-event level privacy, whereby event-level privacy is preserved for streams of any length, and user-level privacy is preserved within those streams for sequences of database updates up to length w. This definition degrades to event-level privacy for small w, and describes user-level privacy for finite streams of length T when $w = T$. The authors propose two algorithms to answer queries with unbounded w-event privacy for time-series databases. Cao and Yoshikawa [6] expand on w-event privacy, proposing privacy over ℓ-trajectories, defined as ℓ consecutive updates by a single user, which can occur over more than ℓ total database updates. Li et al. [52] propose algorithms releasing histograms with w-event differential privacy under the time-series model using SVT to determine publication frequency. Chen et al. [12] provide the first differentially private mechanism to allow a variety of queries to be answered over a time-series stream. Cao et al. [7] extend event-, user-, and w-event-level privacy to the scenario where an adversary has knowledge of temporal correlations within the data, which is overlooked in prior works.

Fan et al. [34] provide algorithms for aggregate statistics on web browsing behaviour. Rather than providing user- or event-level privacy, they protect browsing sessions, with each session consisting of a sequence of page views at consecutive, discrete time stamps. They extend their previous work on time-series [31–33] to provide two ε-differentially private algorithms for releasing aggregate counts for web page visits at each time stamp.

2.2.3 Differential privacy for fully dynamic data

This section describes database models that allow only a single update at each time, but updates may be an addition or deletion of an entry, or modification to an existing entry. There is limited existing literature into such *fully dynamic* models. Mir et al. [56] first proposed such a model, expanding on the growing database models of Dwork et al.

$\mathbf{D_1}$	**1**
1	2
2	1
3	2
...	
n	8

$\mathbf{D_2}$	**1**	**2**
1	2	3
2	1	4
3	2	2
...		
n	8	6

$\mathbf{D_T}$	**1**	**2**	**...**	$\mathbf{T-1}$	**T**
1	2	3	...	2	4
2	1	4	...	7	9
3	2	2	...	5	7
...					
n	8	6	...	6	3

Figure 2.3: Example of a T-bounded time-series database D, with n users, at times $t = 1, 2, ..., T$. Each table D_t represents the state of the database at time t, with colums representing timestamps and rows representing users. We can see that once a column is inserted, it remains static.

[26, 27]. Recently, Qiu and Yi [59] have introduced a more general fully dynamic database model, again drawing inspiration from the seminal works of Dwork et al. [26, 27] and Chan et al. [9]. We hereby refer to the former as the *state update* model and the latter as the *add/delete* model.

The State Update Dynamic Database Model. Definition 2.2.4 describes the database model presented by Mir et al. [56].

Definition 2.2.4 (State update dynamic database model, from Section 2.1 of Mir et al. [56])**.** *Consider a finite set of users, $\mathcal{U}$, of size n, a finite set of integer valued states $\mathcal{X} = 0, 1, ..., N$, a time bound T, and a stream S of updates. Each user $i \in \mathcal{U}$ is in state $u_{i,0} = 0$ at time $t = 0$, and at each time $t > 0$ exactly one user, j_t, updates their state. State updates are represented as a tuple indicating the user and the value of the change in state. That is, the update at time t is given by $s_t = (j_t, d_t)$ where $d \in \{-(u_{j_t,t-1}), -(u_{j_t,t-1} - 1), ..., +(N - (u_{j_t,t-1} + 1)), +(N - (u_{j_t,t-1}))\}$ so that $u_{j_t,t} = u_{j_t,t-1} + d_t \in \mathcal{X}$, and for all $i \neq j_t$, $u_{i,t} = u_{j_t,t-1}$. Thus the state of any user i at time t can be determined from S by taking $u_{i,t} = \sum_{\tau=1}^{t}\{d_\tau : s_\tau = (i, d_\tau)\}$.*

The authors further distinguish between a *partly dynamic* database, where all updates are positive integers, and a *fully dynamic* database, where updates can be positive or negative. The definition of privacy under this model is similar to definitions of bounded, user-level privacy, with some important distinctions.

S
(1, +7)
(4, +3)
(4, +2)
(2, +3)
(3, +4)
(2, +5)
(1, -2)
(3, -4)
(4, -1)
(4, +2)

D_0	0	0	0	0
D_1	7	0	0	0
D_2	7	0	0	3
D_3	7	0	0	5
D_4	7	3	0	5
D_5	7	3	4	5
D_6	7	8	4	5
D_7	5	8	4	5
D_8	5	8	0	5
D_9	5	8	0	4
D_{10}	5	8	0	6

S'
(4, +3)
(4, +2)
(2, +3)
(3, +4)
(3, +5)
(2, +5)
(3, -4)
(4, -1)
(4, +2)

D'_0	0	0	0	0
D'_1	0	0	0	3
D'_2	0	0	0	5
D'_3	0	3	0	5
D'_4	0	3	4	5
D'_5	0	3	9	5
D'_6	0	8	9	5
D'_7	0	8	5	5
D'_8	0	8	5	4
D'_9	0	8	5	6

Figure 2.4: Example of neighbouring state update streams S and S', and corresponding database streams D and D' with $n = 4$ users, where the neighbouring subsets of update times are $K = \{1, 7\}$ and $K' = \{5\}$, and neighbouring users are $(j, k) = (1, 3)$. Yellow cells indicate the effect of updates on the databases and blue cells indicate where neighbours differ.

Definition 2.2.5 (Neighbouring state update dynamic databases, from Definition 4 of Mir et al. [56])**.** *S and S' are said to be (user-level) neighbors if there exists a (multi)set of updates in S indexed by $K \subseteq [t]$ that update the same ID $j \in U$, and there exists a (multi)set of updates in S' indexed by $K' \in [t']$ that updates some $k \neq j \in U$ such that $\sum_{\tau \in K} d_\tau = \sum_{\tau \in K'} d'_\tau$ and for all other updates in S and S' indexed by $Q = [t] - K$ and $Q' = [t'] - K'$ respectively,*

$$\forall i \in \mathcal{U} \sum_{\tau \in Q} \{d_\tau : s_\tau = (i, d_\tau)\} = \sum_{\tau \in Q'} \{d'_\tau : s'_\tau = (i, d'_\tau)\}\,.$$

Definition 2.2.5 differs from previous definitions in that the order, number, and value of updates to all users may differ between neighbours, provided that each user is in the same state at time T. This is a particularly complicated privacy model that, unlike other models, has very little restriction on the number, order, and individual values of updates, but is very restrictive in terms of how neighbouring streams can differ in total value. While k and j represent different individuals, they must maintain the property that they are in the same state at time T. Figure 2.4 gives shows a simple example of neighbouring state update streams. In this example the order and value of updates not in the subsets K and K' have been maintained, though this is not required by the definition.

Beyond defining this new model, Mir et al. present three ε-pan-private sketches, as well as lower bounds for pan-privacy against a single intrusion. The algorithms presented are: *distinct count*, for partly dynamic data, estimating the number of distinct users with non-zero state; *cropped first moment*, for fully dynamic data, estimating the sum of all elements after cropping each value to some τ; and, *heavy hitters count*, for fully dynamic data, estimating the number of distinct users with a state worth at least $1/k$ of the total state, for some k. Approximation factors and lower bounds, under pan-privacy, are given for all three problems.

The Add/Delete Dynamic Database Model. Recent work by Qiu and Yi [59] extends the growing database model to allow deletions and empty updates under an event-level, unbounded privacy model. This model allows only one update per time period, however rather than changing the value of an entry, updates can either add or delete entries. Unlike the growing database models described in Section 2.2.1, updates can also be empty.

Definition 2.2.6 (Add/Delete Fully Dynamic Database Model, from Section 1.3 of Qiu and Yi [59])**.** *Let $\mathcal{X}$ be a finite state universe and $t = 1, 2, \ldots$ be a (possibly infinite) time horizon. An add/delete dynamic database is modeled by a stream of updates S with each update $s_t = (j_t, d_t) \vee \perp$ for $j_t \in \mathcal{X}$ and $d_t \in \{1, -1\}$. The number of elements in state i at time t is given by $\sum_{\tau=1}^{t}\{d_\tau : s_\tau = (i, d_\tau)\}$.*

Figure 2.5 shows examples of neighbouring streams under this model. Notice that identities of users are not required, as this model provides only event-level privacy.

S
(3, +)
(1, +)
(2, +)
(1, -)
$\perp$
(2, +)
(4, +)
(3, +)
(3, +)
(4, -)

$\mathbf{x_0}$	0	0	0	0
$\mathbf{x_1}$	0	0	1	0
$\mathbf{x_2}$	1	0	1	0
$\mathbf{x_3}$	1	1	1	0
$\mathbf{x_4}$	0	1	1	0
$\mathbf{x_5}$	0	1	1	0
$\mathbf{x_6}$	0	2	1	0
$\mathbf{x_7}$	0	2	1	1
$\mathbf{x_8}$	0	2	2	1
$\mathbf{x_9}$	0	2	3	1
$\mathbf{x_{10}}$	0	2	3	0

$\mathbf{S'}$
(3, +)
(1, +)
$\perp$
(1, -)
$\perp$
(2, +)
(4, +)
(3, +)
(3, +)
(4, -)

$\mathbf{x'_0}$	0	0	0	0
$\mathbf{x'_1}$	0	0	1	0
$\mathbf{x'_2}$	1	0	1	0
$\mathbf{x'_3}$	1	0	1	0
$\mathbf{x'_4}$	0	0	1	0
$\mathbf{x'_5}$	0	0	1	0
$\mathbf{x'_6}$	0	1	1	0
$\mathbf{x'_7}$	0	1	1	1
$\mathbf{x'_8}$	0	1	2	1
$\mathbf{x'_9}$	0	1	3	1
$\mathbf{x'_{10}}$	0	1	3	0

Figure 2.5: Example of a fully dynamic add/delete update stream S and neighbouring stream S', and corresponding histogram streams X and X' where histograms are of size $N = 4$, and where x_t is the state of the histogram at time t. Yellow cells indicate the effect of updates on the histograms and blue cells indicate where neighbours differ.

The authors provide a black box algorithm to convert any (ε, δ)-differentially private mechanism for static data, with (α, β)-accuracy for $\alpha = f_\alpha((\varepsilon, \delta), \beta)$, to a differentially private add/delete fully dynamic algorithm with accuracy

$$\alpha = O\left(\frac{\log t}{\varepsilon} + f_\alpha\left(\frac{(\varepsilon, \delta)}{\log^5 m_t}, n_t + \frac{\log^{1.5} m_t \log(1/\beta)}{\varepsilon}\right)\right),$$

where n_t is the size of the dataset at time t, and $m_t = N_t + \log\log t$ where N_t is the number of updates at time t.

While this is a flexible model that generalises many existing dynamic models, the definitions and mechanisms provided by Qiu and Yi [59] are limited to event-level, rather than the stronger user-level, privacy.

2.2.4 Alternative models

While our focus here is on a single stream of data from a discrete, finite universe of states, differential privacy for dynamic data of other forms has been explored. For example, Jain et al. [45] formalise the notion of differentially private online convex programming, where data are drawn interactively from a convex set. They give differentially private algorithms for gradient descent in these settings. Smith and Thakurta [64] continue the study of differentially private online learning, improving on the bounds given by Jain et al. [45] and expanding to the bandit setting. Perrier et al. [58] study differentially private release of statistics on real-valued data streams. They propose a mechanism for choosing a cropping threshold, when no bound on data values in known, with theoretical accuracy guarantees. Wang et al. [66] improve on the guarantees of Perrier et al. [58].

This thesis considers only the *global* model of differential privacy, whereby the database itself is considered a trusted aggregator. There is also considerable work on dynamic data in the *local* setting. The local model of differential privacy considers the case where the data aggregator is untrusted, so privacy preserving mechanisms must be applied by the user prior to sending data to a centralised database. This model was first formalised by Kasiviswanathan et al. [48], though guaranteed long before differential privacy itself had been formalised, in the randomised response mechanism of Warner [67]. Rastogi and Nath [60] and Chan et al. [10] provide differentially private mechanisms for dynamic distributed data, with no trusted aggregator. Erlingsson et al. [30] give a mechanism to improve the privacy of aggregated, locally private, data through anonymisation, where users update their data up to k times over the T time periods. Wang et al. [66] provide a local version of their global DP mechanism for real valued streams. Ye et al. [68] introduce a notion of local differential privacy for time-series data and provide three mechanisms.

Chapter 3

The Fixed Size Dynamic Database Model (FSDD)

Here we introduce a new dynamic database model for differential privacy. Our model, the fixed size dynamic database (FSDD), allows records to be modified over time, unlike the strictly growing model described in Chapter 2. It has different constraints to other dynamic database models previously described. The constraints of FSDD allow us to make assumptions about the cumulative sensitivity of database updates over time, which we use to our advantage in designing mechanisms and assessing their accuracy. In Section 3.1 we describe the FSDD model in detail, including privacy definitions and comparisons to other models.

In Section 3.2 we give several algorithms for answering the same query, with differential privacy, after each update to the database. First we give a naive algorithm as a baseline, then introduce a new algorithm, the τ-frequency repeated queries (τ-RQ) algorithm for queries with real valued answers. We then modify this to give a *black box* version that converts any static DP algorithm to an FSDD DP algorithm. Finally, we modify the sparse vector technique (SVT), using it as a subroutine to give a more sophisticated baseline than the naive algorithm.

3.1 Database and privacy model

3.1.1 Database model

Our new database model, the fixed size dynamic database (FSDD) has some fixed set of users of size n, who each start in some state in finite universe $\mathcal{X}$ at time $t = 0$. At each time $t = 1, ..., T-1$ a single user updates their state, to any state in $\mathcal{X}$, not excluding the state they were previously in. Below, we give an example of a scenario that may use such a database model, as well as a formal definition.

Example 3.1.1 (FSDD)**.** *A cohort of graduates commence a graduate employment program. Graduates have individual review meetings where their progress is assessed by the cohort manager. After each meeting, the graduate is assigned a status from:*

$$\mathcal{X} = \{P\text{: Probation}, R\text{: At Risk}, S\text{: Satisfactory}, E\text{: Exceeding Expectations}\}.$$

All graduates are in default state P at time $t = 0$. The manager wants to share how the cohort is tracking without revealing the individual status of any graduate.

Definition 3.1.1 (T-bounded fixed-size dynamic database (FSDD-T))**.** *Consider a stream of databases $S = D_0, D_1, ..., D_{T-1}$, each with n elements drawn from a discrete, finite universe $\mathcal{X}$ of size N. At time $t = 0$, initial database $D_0 \in \mathcal{X}^n$. At each subsequent time $t = 1, 2, ..., T-1$, for some $j_t \in [n]$, $D_t[j_t]$ may differ from $D_t[j_t - 1]$ and $D_t[i] = D_{t-1}[i]$ for all $i \neq j_t$. That is, at each of $T-1$ times, with T known in advance, exactly one element is updated.*

Equivalently, we can represent an FSDD stream S as an initial database D_0 and a stream of updates $U = U_1, ..., U_{T-1}$, where each $U_t = (j_t, D_t[j_t])$ for index $j_t \in [n]$ and value $D_t[j_t] \in \mathcal{X}$.

A diagram of the model described in Example 3.1.1 is shown in Figure 3.1. While this example has all individuals in the same status, P, at time 0, this is not required by Definition 3.1.1.

U
(1, R)
(4, S)
(3, R)
(1, S)
(2, S)
(4, R)
(2, E)
(3, S)
(4, S)
(1, R)

$\mathbf{D_0}$	P	P	P	P
$\mathbf{D_1}$	R	P	P	P
$\mathbf{D_2}$	R	P	P	S
$\mathbf{D_3}$	R	P	R	S
$\mathbf{D_4}$	S	P	R	S
$\mathbf{D_5}$	S	S	R	S
$\mathbf{D_6}$	S	S	R	R
$\mathbf{D_7}$	S	E	R	R
$\mathbf{D_8}$	S	E	S	R
$\mathbf{D_9}$	S	E	S	S
$\mathbf{D_{10}}$	R	E	S	S

$\mathbf{U'}$
(1, R)
(4, S)
(3, E)
(1, S)
(2, S)
(4, R)
(2, E)
(3, E)
(4, S)
(1, R)

$\mathbf{D'_0}$	P	P	P	P
$\mathbf{D'_1}$	R	P	P	P
$\mathbf{D'_2}$	R	P	P	S
$\mathbf{D'_3}$	R	P	E	S
$\mathbf{D'_4}$	S	P	E	S
$\mathbf{D'_5}$	S	S	E	S
$\mathbf{D'_6}$	S	S	E	R
$\mathbf{D'_7}$	S	E	E	R
$\mathbf{D'_8}$	S	E	E	R
$\mathbf{D'_9}$	S	E	E	S
$\mathbf{D'_{10}}$	R	E	E	S

Figure 3.1: Example of neighbouring FSDD update streams U and U' and database streams S and S', with rows representing times and columns representing individuals in the database. Databases have size $n = 4$, universe size $N = 10$, the number of updates are bounded by $T = 11$ and the databases differ at user $j = 3$. Yellow cells indicate the effect of updates on the histograms and blue cells indicate where neighbours differ.

3.1.2 Privacy model

Given that our database model has a fixed size, and we expect users to provide multiple updates, we define privacy using a *bounded, user-level* privacy model. That is, we define neighbouring database streams $S \approx S'$ as those that differ by the values of any updates by a single user, and where databases $D_t \approx_{\circlearrowleft} D'_t$ are bounded neighbours, under Definition 2.1.3 for static databases, at every time t. A formal definition is given below, and an example is given in Figure 3.1 above.

Definition 3.1.2 (FSDD Neighbours)**.** *FSDD database streams S, S' are considered neighbouring, $S \approx S'$, if both have database size n and for some $j \in [n]$, $D_t[i] = D'_t[i]$ for all $i \neq j$ at all times $t = 0, 1, ..., T-1$. That is, at all times $t \in [T-1]$, D'_t can differ from D_t only at position j.*

Equivalently, for neighbouring initial databases $D_t \approx_{\circlearrowleft} D'_t$, update streams U and U' ensure that streams S and S are neighbouring if, for some $j \in [n]$, $U'_t = U_t$ for all $U_t \in \{(j_s, \cdot) : j_s \neq j\}$ and $U'_t = (j, \cdot)$ for all $U_t \in \{(j, \cdot)\}$. That is, the index of every update is the same between neighbouring streams, and the value of each update is the same except for some index j.

When we refer to the ε-differential privacy of algorithms on FSDD streams, we do so relative to this definition of FSDD neighbours.

Definition 3.1.3 (ε-differential privacy of algorithms over FSDD streams)**.** *A randomised algorithm $\mathcal{M}$, answering a sequence of queries $\mathcal{Q}$ over an FSDD stream S, is ε-differentially private if, for any two FSDD-neighbouring (Definition 3.1.2) streams $S \approx S'$ and any subset of possible output streams Y,*

$$\Pr[\mathcal{M}(S) \in Y] \leq e^{\varepsilon} \cdot \Pr[\mathcal{M}(S) \in Y],$$

where $\varepsilon > 0$.

In designing mechanisms for differential privacy under the FSDD model we can exploit two key characteristics. The first, which is shared with most privacy models in the dynamic DP literature (with the notable exception of the state update model of Mir et al. [56]), is that at each time t, databases D_t and D'_t are neighbouring. In Figure 3.1, we see that, in any row, at most one column differs between the neighbouring database representations. This is built in to our Definition 3.1.2 of neighbouring FSDDs, and allows us to apply basic composition over time, since the k-fold adaptive composition result (see discussion of Theorem 2.1.3) guarantees that differential privacy is maintained under composition even of different databases and different mechanisms, provided each mechanism is differentially private. It is important to note that this requires that the differing updates still occur at the same times between neighbouring updates, which limits the type of privacy that can be expected when it comes to membership inference. This is a limitation shared by many dynamic differential privacy models, and, more generally, with bounded DP in that the size of the database may be exposed.

The second characteristic, of the FSDD model itself rather than the privacy model, is that only a single element is updated at each time. As such, consecutive databases D_t and D_{t+1} are also neighbouring, that is, $D_t \circlearrowleft D_{t+1}$ such that $|q(D_t) - q(D_{t+1})| \leq \Delta_q$. While this does not affect our use of basic privacy results, it allows us to bound the overall effect of updates over time to optimise accuracy.

In Section 3.2 we give a mechanism that relies only on the first characteristic, and then one that employs both, for privately answering a repeated query over time. In Chapter 4 we give mechanisms for privately answering multiple different queries after each update.

3.1.3 Comparison to existing dynamic database models

In this section, we compare the FSDD database and privacy models to those discussed in Section 2.2. We show that the database model is most similar to the state update model of Mir et al. [56], while our privacy model is much simpler. Our database model is less restrictive than growing database models (e.g. [9, 15, 26]) and more restrictive than time-series models (e.g. [6, 7, 49, 52])and the add/delete database model [59]. Table 3.1 gives an overview of key differences between these models and FSDD.

The strictly growing model. The growing database models described in Section 2.2.1 have a single entry added at each time, such that the database size is growing at a fixed rate, with no deletions or modifications to past entries. The bitstream model [9, 26, 27], where $\mathcal{X} = \{0, 1\}$, starts with an empty database at time 0, such that the size of the database at time t, $n_t = t$. This property is shared with some more general growing models, though in Cummings et al. [15], where the data universe is some discrete finite set, the database has some size n_0 at time 0, such that the database size at time t is $n_t = n_0 + t$. FSDD on the other hand has a fixed size n, but has one record updated at each time.

In their papers on pan-privacy for streaming and continual release, Dwork et al. [26, 27] give both event- and user-level privacy preserving algorithms; their requirement of pan-privacy is stronger than the user-level differential privacy we require of FSDD. In the bitstream algorithms of Chan et al. [9] and the growing database algorithms of Cummings

	Users per t	Update types	Privacy model
Strictly growing [9, 15, 26]	1	$+$	User-level pan-privacy, event-level DP.
State update [56]	1	$\circlearrowleft$	DP. Neighbours differ by user for some subset of updates, and by order of value of all updates, provided sums maintained. See Definition 2.2.5.
FSDD	1	$\circlearrowleft$	User-level DP, initial value and all updates.
Time-series [6, 7, 49, 52]	$\leq n$	$\circlearrowleft$	Various definitions, event-, user- and w-event- level DP.
Add/delete [59]	1	$+/-/\perp$	Event-level DP.

Table 3.1: Properties of dynamic databases used in various key papers mentioned in Section 2.2, where n is the number of users, and the update types are add ($+$), delete ($-$), change ($\circlearrowleft$), and empty ($\perp$).

et al. [15] only event-level privacy is guaranteed. If we were to provide an event-level privacy guarantee for FSDD, neighbouring streams would differ only at a single update, rather than all updates for a single user, and would require that $D_0 = D'_0$ for the initial databases.

The discrete time series model. FSDD may be considered a special case of the discrete time series model described in Section 2.2.2, where updates are restricted to only one user at each time. The T-bounded time-series database is often modelled, for example by Cao et al. [7], as a matrix whereby the users are represented by rows and the time periods by columns, such that the trajectory of a user over time is represented as a row vector. Kellaris et al. [49] model their infinite time-series as a stream of databases, where each database in the stream is a snapshot of the state of each user at the given time. Our model of the FSDD is like the latter, whereby we do not assume memory of previous states, or of the times of each update. This choice does not affect the privacy model, however, since our FSDD neighbour definition could apply to either.

There is a large corpus of literature considering differential privacy for time-series, and thus a variety of privacy definitions employed. For example, the four key papers we listed in Table 2.1, cover event-level [7], user-level [52], and w-event sliding window [6, 49] privacy definitions. Since w-event sliding window privacy is equivalent to user-level privacy in the T bounded setting if $w = T$, most of these are similar to our user-level privacy guarantee, whereby neighbouring streams differ on the values of a single user.

The add/delete model. The add/delete model of Qiu and Yi [59] is also more general than FSDD, allowing additions and deletions to the database, as well as empty updates. If we map our data universe $\mathcal{X}$, where $|\mathcal{X}| = N$ to the set of integers $[0..N-1]$ and restrict ourselves to the case where $D_0[i] = 0$ for all users indexed $i \in [n]$, we can model FSDD using the operations of the add/delete model. If we represent each time period of FSDD as two time periods of the add/delete model, then for each FSDD update $U_t = (j_t, D_t[j_t])$ we have a pair of add/delete updates $s_{2t-1}, = (D_t[j_t], -1)$ and $s_{2t-1}, = (D_t[j_t], -1)$ such that we remove one instance of j_t's former state and add one of its current state, so the overall set of states moves from the FSDD histogram x_{t-1} to x_t.

FSDD has user-level privacy built in to the model, which is stronger than the event-level privacy guaranteed by the add/delete model [59]. Since the add/delete database model maintains only the counts of each item (histograms) rather than the user associated with each entry, an event-level privacy guarantee is reasonable, though it would be possible to define user-level privacy for the add/delete model, as it is with the strictly growing database model.

The state update model. Finally, FSDD is very similar to the state update database model of Mir et al. [56], provided we represent the FSDD data universe as integers. The only substantial is that the state update model has that all users are in state 0 at time 0, whereas FSDD allows users to start in any state in $\mathcal{X}$. If we start with a database of 0s, we can model FSDD as the state update model by representing each update in the stream by the tuple $U'_t = (j_t, D_{t-1}[j_t] - D_t[j_t])$ showing the index of the user updates and the update as a difference between the previous state and the new state, rather than $U_t = (j_t, D_t[j_t])$.

The key difference between FSDD and the state update model is the privacy model, particularly the definition of neighbouring streams: neighbouring databases in the state update model may have the order and value of updates for all users differ between neighbours, provided their sum is equal; and rather than changing the values of a single user between neighbouring databases, it allows some subset of updates to be transferred from one user to another, though again requires the sum of these updates to be maintained. Additionally, Mir et al. [56] give pan-private algorithms, a stricter privacy guarantee than differential privacy alone. While pan-privacy is possible for FSDD streams, this requires the database itself to be stored privately, restricting the types of queries that can be made.

3.2 Repeated queries under the FSDD Model

In this section we present mechanisms for the scenario where we wish to answer the same query after each update to the database. This is a common problem in the dynamic database literature, for example, releasing counts [9, 26], histograms [6, 7, 49, 52], or heavy hitters [10, 56]. These mechanisms are often referred to as *continual release* or *continual observation* mechanisms; here, we use the term *repeated queries* for clarity of comparison with the *interactive queries* described in Chapter 4. Rather than present mechanisms for specific queries, we give mechanisms that take a query as input and release an answer to that query after each update. Section 3.2.2 describes a generalised version, where we use a static differentially private mechanism as a black box, trusting its privacy guarantees, to convert any static DP mechanism to one for FSDD.

First, in Algorithm 5, we consider a naive mechanism, NRQ, for answering repeated, numeric, queries with DP under the FSDD model. A simple application of basic composition, Algorithm 5 outputs a DP answer to the query q on the database at each time step, capped at T time-steps, using the Laplace mechanism. Its privacy follows from the privacy of the Laplace Mechanism, and we give accuracy guarantees based on the accuracy of the Laplace mechanism and the union bound.

Theorem 3.2.1. *NRQ (Algorithm 5) is ε-differentially private.*

Proof. By the privacy of the Laplace Mechanism, the answer for each round is $\left(\frac{\varepsilon}{T}\right)$-DP. Composing T $\left(\frac{\varepsilon}{T}\right)$-DP queries gives overall privacy ε, by basic composition (Theorem 2.1.3). □

Theorem 3.2.2. *NRQ (Algorithm 5) is (α, β)-accurate for*

$$\alpha = \ln\left(\frac{T}{\beta}\right) \cdot \frac{(T)\Delta_q}{\varepsilon}.$$

Proof. By the accuracy of the Laplace mechanism (Theorem 2.1.4), we have that each query has accuracy $|a_t - q(D_t)| \leq \alpha = \ln(\frac{T}{\beta}) \cdot \frac{T\Delta_q}{\varepsilon}$, with probability no less than $1 - \frac{\beta}{T}$. By the union bound, this gives (α, β)-accuracy overall. □

3.2.1 τ-frequency repeated queries on FSDD-T (τ-RQ)

NRQ (Algorithm 5) provides the same guarantees of privacy and accuracy for any arbitrary sequence of databases $D_i \in \mathcal{X}^n$. Under the FSDD-T model, databases D_t and D_{t+1} differ

Algorithm 5 NRQ: Naive repeated queries on FSDD-T

```
Input: Query function $q : \mathcal{X}^n \to \mathbb{R}$ with sensitivity $\Delta_q$. Database stream $\{D_t\}$, where
       each $D_t \in \mathcal{X}^n$ is a vector of $n$ states, each from discrete, finite universe $\mathcal{X}$, for
       times $t = 0, 1, 2, ..., T-1$. Privacy paramater $\varepsilon$.
Output: Differentially private output $\{a_t\} \in \mathbb{R}^T$
1: for each time step $t = 0, 1, 2, ..., T-1$ do
2:     output $a_t \leftarrow q(D_t) + \mathrm{Lap}\left(\frac{T\Delta_q}{\varepsilon}\right)$
```

only in the value of one element, providing us with a limit on the effect of a single update of $|q(D_t) - q(D_{t+1})| \leq \Delta_q$, where Δ_q is the ℓ_1-sensitivity of the query function q, as defined in Definition 2.1.4. Using this information, we can design algorithms with improved accuracy over composition of arbitrary database sequences.

Algorithm 6, τ-RQ provides such an improvement by querying the database only after every $1 \leq \tau \leq T$ updates, and otherwise returning the previous output. When $t \bmod \tau = 0$ we say it is a *sample round*, where we use the Laplace Mechanism, $\mathcal{M}_{\text{Lap}}$ to answer query $q : \mathcal{X}^n \to \mathbb{R}$ on database D_t with ε-differential privacy. When $t \bmod \tau > 0$ we say it is a *non-sample round*, where we output the answer from the most recent sample round, rather than taking a new sample. Equivalently, we could give no output for non-sample rounds; we provide an output at every time t for ease of comparison with SVT and PMW based algorithms presented later in the thesis. We call τ the *frequency parameter*, as it determines the distance between successive samples of the database. The total number of sample rounds is given by the *cutoff*, $c = \lceil \frac{T}{\tau} \rceil$. If $\tau = 1$, τ-RQ is equivalent to NRQ.

Consider a small-scale example, where $\tau = 4$ and $T = 10$ we would take three samples, at times $t = 0$, 4, and 8. Since we know that $|q(D_t) - q(D_{t+1})| \leq \Delta_q$, we can bound the effect of updates between samples. For example, at time $t = 6$ there have been two updates since the last sample, at time $t = 4$, so we know that $|q(D_6) - q(D_4)| \leq 2\Delta_q$, so that the additive error of output a_6, $|a_6 - q(D_6)|$ must be no more than $|a_4 - q(D_4)| + 2\Delta_q$. When choosing τ, we must balance between the error introduced by the Laplace Mechanism, which, for fixed total ε, increases as the number of samples c increases, and the error introduced by the effect of the updates between samples.

Algorithm 6 τ-RQ: τ-frequency repeated queries on FSDD-T

```
Input: Query function $q : \mathcal{X}^n \to \mathbb{R}$ with sensitivity $\Delta_q$. FSDD stream $\{D_t\}$, where each
       $D_t \in \mathcal{X}^n$ is a vector of $n$ states, each from discrete, finite universe $\mathcal{X}$, for times
       $t = 0, 1, 2, ..., T-1$, such that $|q(D_t) - q(D_{t+1})| \leq \Delta_q$. Privacy paramater $\varepsilon$.
       Frequency parameter $\tau$.
Output: Differentially private output $\{a_t\} \in \mathbb{R}^T$
1: $c \leftarrow \lceil \frac{T}{\tau} \rceil$
2: for each time step $t = 0, 1, 2, ..., T-1$ do
3:     if $t \bmod \tau = 0$ then output $a_t \leftarrow q(D_t) + \text{Lap}\left(\frac{c\Delta_q}{\varepsilon}\right)$
4:     else output $a_{t-1}$
```

Like the privacy and accuracy of NRQ, τ-RQ is ε-differentially private by the privacy of the Laplace Mechanism and basic composition.

Theorem 3.2.3 (Privacy of τ-RQ)**.** *τ-RQ (Algorithm 6) is ε-differentially private.*

Proof. This follows directly from the proof of Theorem 3.2.1, where the answer for each sample round is, instead, $\left(\frac{\varepsilon}{\lceil T/\tau \rceil}\right)$-DP. Non-sample rounds do not affect the privacy, by Theorem 2.1.2. □

The accuracy guarantee is given in two parts, first, for the sample rounds, by a union bound over the accuracy of the Laplace Mechanism, and then, for the non-sample rounds, by a bound on the cumulative effect of updates between samples.

Theorem 3.2.4 (Accuracy of τ-RQ)**.** *τ-RQ (Algorithm 6) is (α, β)-accurate for*

$$\alpha = \ln\left(\frac{c}{\beta}\right) \cdot \frac{c\Delta_q}{\varepsilon} + (\tau - 1)\Delta_q \,, \tag{3.1}$$

where $c = \left\lceil \frac{T}{\tau} \right\rceil$. That is, all outputs $\{a_t\}_{t=0}^{T-1}$ are within $\pm\alpha$ of the true answer to query q, on database D at time t, with probability $1 - \beta$.

Proof. Following the proof of Theorem 3.2.2, we have that the answer for each sample round is $\left(\alpha', \frac{\beta}{c}\right)$-accurate, where:

$$\alpha' = \ln\left(\frac{c}{\beta}\right) \cdot \frac{c\Delta_q}{\varepsilon} \,.$$

Since $|q(D_t) - q(D_{t+1})| \le \Delta_q$, we know that $|q(D_t) - q(D_{t+\tau-1})| \le (\tau - 1)\Delta_q$. Hence the accuracy for any non-sample round is at most $\alpha' + (\tau - 1)\Delta_q$.

Combining the accuracy for the sample and non-sample rounds we find that the agorithm has worst case accuracy $\alpha = \ln\left(\frac{c}{\beta}\right) \cdot \frac{c\Delta_q}{\varepsilon} + (\tau - 1)\Delta_q$. Since non-sample rounds are deterministic, we can take the union bound for β across only the sample rounds, to give $|a_t - q(D_t)| \le \alpha$ with probability no less than $1 - \beta$ overall. □

Choosing τ. We now consider how we can optimise our choice of τ to improve the accuracy of τ-RQ. Preliminary numerical analysis of theoretical bounds gives us some idea of the expected utility of our algorithms for various parameter choices. In Figure 3.2 we see how parameters ε and τ affect the theoretical accuracy bound α, for (α, β)-accuracy, as shown in Theorem 3.2.4. We plot α for all integer values of τ in $[1..T]$, and indicate the optimal choice of τ for minimising α. We see that looser privacy (larger choice of ε) improves utility (lower α), and that this compounds as the number of samples increases (smaller τ).

Figure 3.3 breaks down one of these curves into increasing and decreasing components, that is, the accuracy of the sample and non-sample rounds. The accuracy of the sample rounds improves as τ increases, since fewer queries are being made overall, decreasing the error bound of the first term, $f_1 = \alpha'$. However, as τ increases, there are more database updates between queries, so there is an increasing upper bound on the difference between the true database at a non-sample round and the database at the time of the last sample round, given by $f_2 = (\tau - 1)\Delta_q$. Here we plot f_1 and f_2 for all integer values of τ in $[2..T]$. These plots were obtained using the `matplotlib` (version 3.7.1) and `numpy` (version 1.42.2) libraries in Python 3.11, by vectorising α, f_1 and f_2, obtained using `math` library functions, across all integer choices of τ.

In Theorem 3.2.5, we give a heuristic for choosing τ by minimising the worst-case theoretical accuracy bound α.

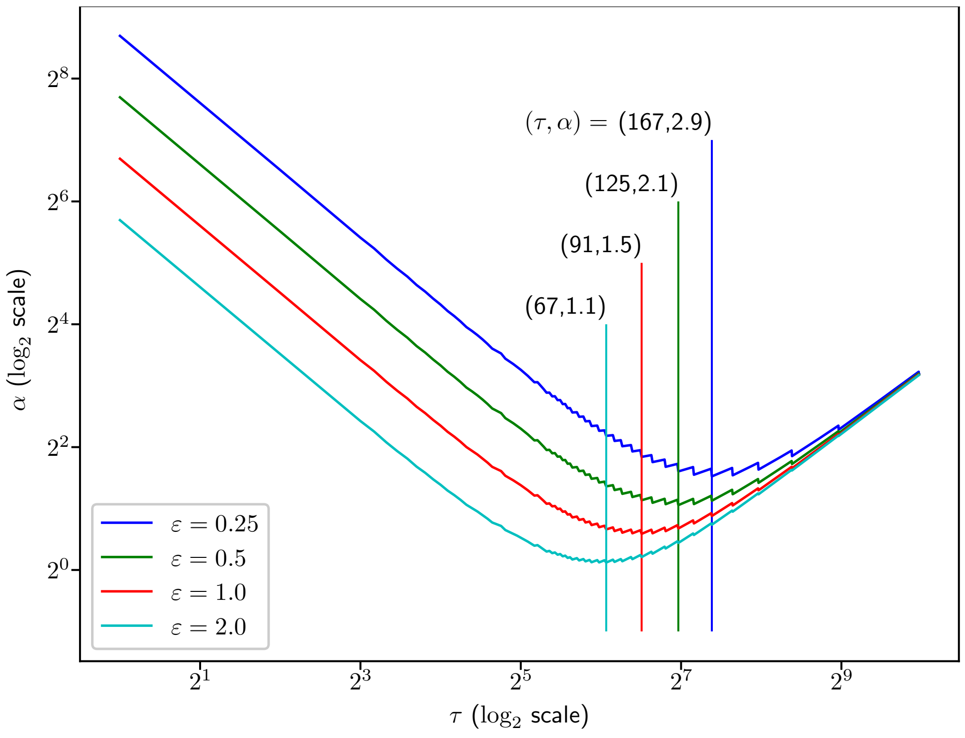

Figure 3.2: Theoretical value of α for (α, β)-privacy guarantee of τ-RQ, with parameter choices $\beta = 0.01, T = 1000, \Delta_q = 0.009$ and various choices of ε, showing minimum α values for each choice of ε. (τ, α) are marked for the minimum α for each ε.

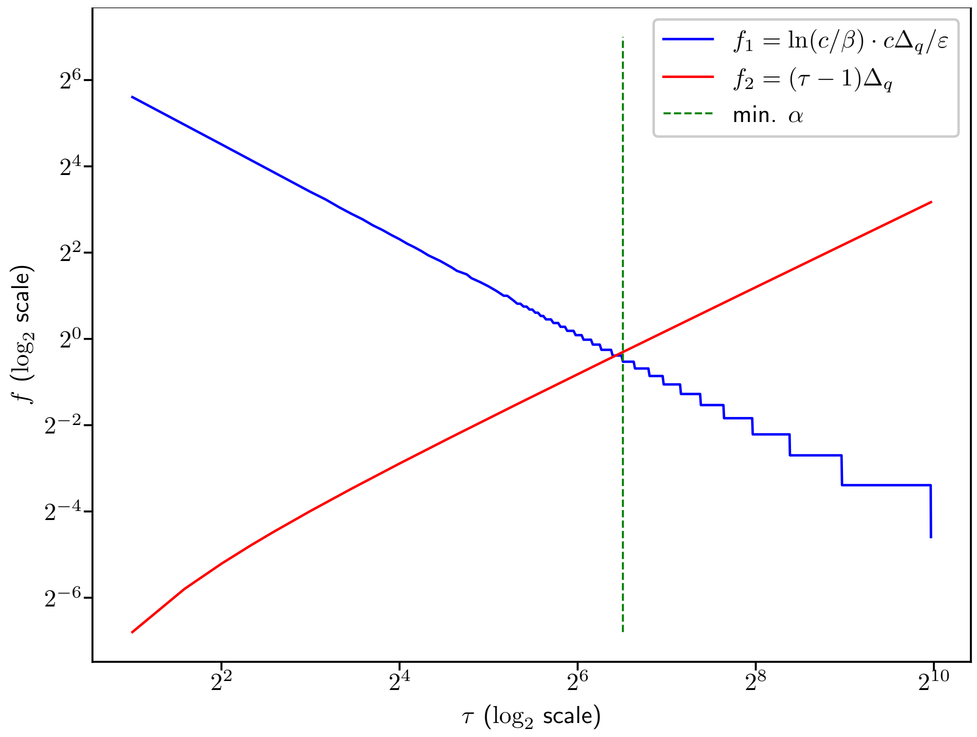

Figure 3.3: Theoretical value of α, split into decreasing and increasing components $f_1 = \ln(c/\beta) \cdot c\Delta_q/\varepsilon$ and $f_2 = (\tau - 1)\Delta_q$, where $c = \lceil \frac{T}{\tau} \rceil$, for (α, β)-privacy guarantee of τ-RQ, with parameter choices $\varepsilon = 1.0, \beta = 0.01, T = 1000, \Delta_q = 0.009$, showing the minimum α value for reference.

Theorem 3.2.5. *The theoretical accuracy bound, α, for τ-RQ (Algorithm 6) is minimised by taking $\tau = \tau'$, given by*

$$\tau' = \arg\min_{\tau} \left(\frac{c}{\varepsilon} \ln \frac{c}{\beta} + \tau \right) , \tag{3.2}$$

where $c = \lceil \frac{T}{\tau} \rceil$.

Proof. The minimum α across all possible choices of τ, when time is T-bounded, is given by:

$$\tau' = \arg\min_{\tau \in [1,T]} \alpha \, , \tag{3.3}$$

where

$$\alpha = \ln \left(\frac{c}{\beta} \right) \cdot \frac{c\Delta_q}{\varepsilon} + (\tau - 1)\Delta_q = \Delta_q \left(\frac{c}{\varepsilon} \ln \left(\frac{c}{\beta} \right) + \tau - 1 \right) .$$

Since the sensitivity, Δ_q, can be factored out, this is equivalent to

$$\tau' = \arg\min_{\tau \in [1,T]} \left(\frac{c}{\varepsilon} \ln \frac{c}{\beta} + \tau \right) . \qquad \square$$

Due to the cutoff $c = \lceil T/\tau \rceil$ including a ceiling function, we have steps in the values of α where there is a sudden drop for the first τ resulting in some c, and then a gradual increase as the $(\tau - 1)\Delta_q$ term increases while the other term remains fixed. We can thus assume that the best choice of τ will be at one of these local minima. We call this set the *minimum distinct τs*, with shorthand MDT. Thus, a heuristic for optimising τ, is to find τ', using brute force to calculate

$$\tau' = \operatorname*{arg\,min}_{\tau \in \mathrm{MDT}} \left(\frac{c}{\varepsilon} \ln \frac{c}{\beta} + \tau \right) .$$

Since this must only be calculated once for any choice of (τ, ϵ, β), using brute force to calculate it across all $\tau \in$ MTD does not come at a significant cost to overall efficiency. Below, we give a closed form for minimising α when generalised to continuous τ.

A lower bound on accuracy. Here we show a continuous lower bound on α, under the (α, β)-accuracy definition, for τ-RQ. This bound is difficult to calculate, so is not preferable to using brute force to find the discrete minimum using Equation (3.3), but may be of independent interest.

Theorem 3.2.6. *The accuracy α of the τ-RQ mechanism (Algorithm 6), given by Theorem 3.2.4 can be bounded below by the continuous function α_c where:*

$$\alpha_c = \ln\left(\frac{T}{\tau\beta}\right) \cdot \frac{\Delta_q T}{\tau\varepsilon} + (\tau - 1)\Delta_q \,,$$

which is minimised when

$$\tau = \sqrt{\frac{T}{2\varepsilon} \cdot W\left(\frac{2e^2 \varepsilon T}{\beta^2}\right)} .$$

Proof. From Theorem 3.2.4 we have that $\alpha = \ln\left(\frac{\lceil T/\tau \rceil}{\beta}\right) \cdot \frac{\Delta_q \lceil T/\tau \rceil}{\varepsilon} + (\tau - 1)\Delta_q$. This is bounded below at each τ by the continuous version, $\alpha_c = \ln\left(\frac{T}{\tau\beta}\right) \cdot \frac{\Delta_q T}{\tau\varepsilon} + (\tau - 1)\Delta_q$. We can then find the minima of this continuous version for a lower bound that applies to both the continuous and discrete functions, by finding some $\tilde{\tau} \in \mathbb{R}$ that minimises α_c.

Taking the first derivative, we have:

$$\frac{d\alpha_c}{d\tau} = \frac{\Delta_q T}{\tau^2 \varepsilon} \left(\ln\left(\frac{\tau\beta}{T}\right) - 1 \right) + \Delta_q \,,$$

which has one positive root:

$$\frac{d\alpha_c}{d\tau} = 0 \implies \tau = \sqrt{\frac{T}{2\varepsilon} \cdot W\left(\frac{2e^2\varepsilon T}{\beta^2}\right)},$$

where $W(x)$, the Lambert W function, is the function satisfying $W(x)e^{W(x)} = z$ [14]. Taking the second derivative:

$$\frac{d^2\alpha_c}{d\tau^2} = \frac{\Delta_q T}{\tau^3\varepsilon}\left(3 - 2\ln\left(\frac{\tau\beta}{T}\right)\right),$$

and substituting $\tau' = \sqrt{\frac{T}{2\varepsilon} \cdot W\left(\frac{2e^2\varepsilon T}{\beta^2}\right)}$ we get:

$$\frac{d^2\alpha_c}{d(\tau')^2} = \frac{2\Delta_q\sqrt{2\varepsilon}}{\sqrt{T}\left(W\left(\frac{2e^2\varepsilon T}{\beta^2}\right)\right)^{\frac{3}{2}}}\left(3 - 2\ln\left(\frac{\beta}{\sqrt{2T\varepsilon}}\sqrt{W\left(\frac{2e^2\varepsilon T}{\beta^2}\right)}\right)\right). \tag{3.4}$$

We complete the proof using the following facts:

1. $\Delta_q, \varepsilon, T, \beta > 0$, by definition,
2. $\ln(W(x)) = \ln(x) - W(x)$, by Equation 3.8 of Corless et al. [14],
3. $x > 0 \implies W(x) > 0$, by definition.

By facts (1) and (2), the expression inside the brackets in Equation (3.4) simplifies to $1 + W\left(\frac{2e^2\varepsilon T}{\beta^2}\right)$, so, by (3), the whole expression is positive and thus τ' is a minimum. □

Figure 3.4 Compares the minimum (for $\tau \in \mathbb{N}$) for the discrete vs continuous accuracy bounds.

3.2.2 τ-frequency repeated black box (τ-RBB)

Several key papers in dynamic differential privacy (e.g. [15, 26]) provide mechanisms that, rather than providing differential privacy themselves, allocate portions of the privacy budget to some mechanism that provides static differential privacy, keeping track to preserve overall privacy using composition properties. We call these *black box* mechanisms, as they do not depend on the internal workings of the static mechanism called, trusting the privacy and accuracy claims of the, possibly black box, static mechanism. Here we

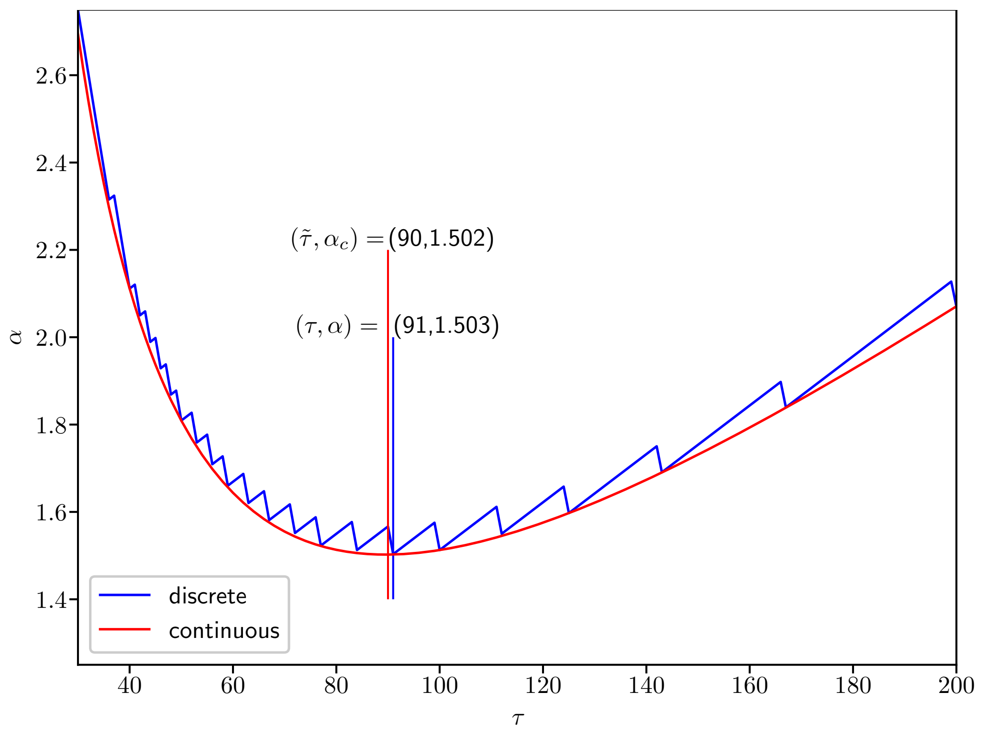

Figure 3.4: Comparison of discrete accuracy bound α and continuous α_c, with parameter choices $\beta = 0.01, T = 1000, \Delta_q = 0.009, \varepsilon = 1.0$, showing minimum α and α_c evaluated for $\tau \in \{30, 31, .., 199, 200\}$.

provide a mechanism for repeatedly calling a black box mechanism in the FSDD setting. Algorithm 7 gives a modified τ-RQ algorithm using this principle. The privacy of this algorithm follows from the privacy of the black box, combined with basic composition, and the accuracy can be determined from the accuracy of the black box using the union bound.

Algorithm 7 τ-RBB: τ-frequency repeated black box on FSDD-T

Input: Database stream $\{D_t\}$, where each $D_t \in \mathcal{X}^n$ for $t = 0, 1, 2, ..., T-1$. Privacy paramater ε. Static, ε-differentially private mechanism $\mathcal{M}(D, \varepsilon)$. Frequency parameter τ.

Output: Differentially private output $\{a_t\} \in \mathbb{R}^T$

```
1: $c \leftarrow \lceil \frac{T}{\tau} \rceil$
2: for each time step $t = 0, 1, 2, ..., T-1$ do
3:     if $t \bmod \tau = 0$ then output $a_t \leftarrow \mathcal{M}(D_t, \frac{\varepsilon}{c})$
4:     else output $a_{\lfloor \frac{t}{\tau} \rfloor}$
```

Theorem 3.2.7 (Privacy of τ-RBB). *τ-RBB (Algorithm 7) is ε-differentially private.*

Proof. This follows directly from the proof of Theorem 3.2.1, where the answer for each sample round is, instead, $\left(\frac{\varepsilon}{\lceil T/\tau \rceil}\right)$-DP. Non-sample rounds do not affect the privacy, by Theorem 2.1.2. □

Theorem 3.2.8 (Accuracy of τ-RBB)**.** *When instantiated with (α, β)-accurate mechanism $\mathcal{M}$, τ-RBB (Algorithm 7) is (α, β')-accurate, where*

$$\beta' = \left\lceil \frac{T}{\tau} \right\rceil \beta \,.$$

That is, all outputs $\{a_t\}_{t=0}^{T}$ are within $\pm\alpha$ of the true function f on database D at time t with probability $1 - \beta'$.

Proof. This follows from the union bound. ☐

3.2.3 Adaptive repeated queries on FSDD-T (ARQ)

Here we modify NumericSVT (Algorithm 3) for the FSDD-T setting, and utilise it to answer repeated queries. Similar to the DSFT algorithm for histograms, by Li et al. [52], ARQ (Algorithm 9) uses NumericSVT as a subroutine, and publishes a new sample value, a_t, only when the difference between a noisy version of true query answer $q(D_t)$ and previously published answer, a_{t-1}, exceeds a noisy version of some given threshold.

First, in Algorithm 8, we give our modified NumericSVT, NSVT-FSDD. To apply the NumericSVT algorithm to the FSDD setting (Algorithm 8), we need only to add an outer for loop, for the units of time.

Algorithm 8 NSVT-FSDD: NumericSVT for FSDD (Based on Algorithm 3 of Dwork and Roth [22])

```
Input: Database stream $S = \{D_t\}$ and query stream $Q = \{q_{t,i}\}$ for $t = 0, ..., T-1$ and
       $i = 1, ..., \ell_t$ for each $t$. Threshold $\theta$, maximum query sensitivity $\Delta_q$, cutoff $c$, and
       privacy parameter $\varepsilon$.
Output: Output stream $\mathbf{a} \in \{\perp, \mathbb{R}\}^L$, where $L = \sum_{t=0}^{T-1} \ell_t$
 1: $\rho \leftarrow \mathrm{Lap}\left(\frac{9c\Delta_q}{4\varepsilon}\right)$
 2: $w \leftarrow 0$
 3: for each time $t$ do
 4:     for each query $i$ do
 5:         $\nu_{t,i} \leftarrow \mathrm{Lap}\left(\frac{9c\Delta_q}{2\varepsilon}\right)$
 6:         if $q_{t,i}(D_t) + \nu_{t,i} \geq \theta + \rho$ then                 ▷ above threshold
 7:             output $a_{t,i} \leftarrow q_{t,i}(D_t) + \mathrm{Lap}\left(\frac{9c\Delta_q}{\varepsilon}\right)$
 8:             $w \leftarrow w + 1$
 9:             $\rho \leftarrow \mathrm{Lap}\left(\frac{9c\Delta_q}{4\varepsilon}\right)$
10:         else output $a_{t,i} \leftarrow \perp$                               ▷ below threshold
11:         if $w \geq c$ then Halt                                   ▷ Exceeded hard query budget
```

Now we show that the privacy of NSVT-FSDD follows from the privacy of NumericSVT. As we show below, the proof can be deduced from the proof of privacy of NumericSVT given by Dwork and Roth [22]. Hence, we only provide the sketch.

Lemma 3.2.9. *NSVT-FSDD is ε-differentially private.*

Proof sketch. The proof of the ε-differentially private of NumericSVT, given in Theorem 3.2.7 of Dwork and Roth [22], depends only on the sensitivity Δ_q. This sensitivity applies for any neighbouring databases. Since all pairs of databases, D_t and D'_t, for each time t, in neighbouring FSDD streams $S \approx S'$ are themselves neighbouring, substituting $D \to D_t$, $D' \to D'_t$, $f_k \to q_{t,i}$, and $\nu_k \to \nu_{t,i}$ into the proof of privacy for NumericSVT [22] results in a proof of privacy for NSVT-FSDD. □

Algorithm 9 ARQ: Adaptive frequency repeated queries on FSDD-T

```
Input: database stream $\{D_t\}$, where each $D_t \in \mathcal{X}^n$ for $t = 0, 1, 2, ..., T-1$ and $|\mathcal{X}| = N$,
       privacy paramater $\varepsilon$, query function $q : \mathcal{X}^n \to \mathbb{R}$ where $|q(D_t) - q(D_{t+1})| \leq \Delta_q$,
       threshold $\theta$, and hard query cutoff $c$.
Output: string $a \in \mathbb{R}^T$
1: $a_{-1} \leftarrow q([\frac{1}{N}]^N)$
2: Initialise Subroutine NSVT-FSDD($\{D_{t>0}\}, \{q'_i\}, \theta, \Delta_q, c, \varepsilon$ ) $\to a'$
3: for each round $t$ do
4:     $q'_{2t} \leftarrow a_{t-1} - q(D_t)$, $q'_{2t+1} \leftarrow q(D_t) - a_{t-1}$
5:     if $a'_{2t} = \perp$ and $a'_{2t+1} = \perp$ then                    ▷ below threshold
6:         $a_t \leftarrow a_{t-1}$
7:     else                                                     ▷ above threshold, hard query
8:         if $a'_{2t+1} = \perp$ then $a_t \leftarrow a_{t-1} + a'_{2t}$
9:         else $a_t \leftarrow a_{t-1} - a'_{2t+1}$
```

We begin ARQ (Algorithm 9) by taking, as an initial comparison value, a_{t-1}, the query q applied to a uniform histogram. For each database D_t, we use NumericSVT to determine whether the difference between D_t and a_{t-1}, the previous published answer, is beyond the noisy version of some threshold θ. We continue this process until we have exceeded our hard query cutoff c, or all updates have been completed. The privacy of ARQ follows from the privacy of NSVT.

Theorem 3.2.10. *ARQ (Algorithm 9) is ε-differentially private.*

Proof. The true database stream is only accessed via the NSVT-FSDD subroutine, which is ε-differentially private by Lemma 3.2.9, the proof of which depends on the privacy of NumericSVT as given by Dwork and Roth [22].

The privacy of NumericSVT given by Dwork and Roth [22] builds on the proof of a simpler algorithm, AboveThreshold, which halts after a single above threshold query is encountered. Composition of AboveThreshold with the Laplace Mechanism gives NumericSVT, such that the privacy of NumericSVT is preserved even when queries are adaptively chosen in response to above threshold queries.

Since the adaptive queries, q', generated by ARQ depend only on a fixed initial value, a_{-1} and on answers to above threshold queries provided by NSVT-FSDD, ARQ is ε-differentially private. $\square$

The optimal choice of threshold, θ, and cutoff, c, depend on several factors, including the query function, q, and the distribution of updates. Li et al. [52] provide a mechanism for adaptively choosing the threshold as the data is updated, that may be of some interest to improve the performance of this mechanism.

Chapter 4

Interactive Query Sequences

In the previous chapter, we considered mechanisms designed to answer a single, fixed query after each update to a fixed size dynamic database (FSDD). Now we consider mechanisms that allow us to answer a sequence of queries, when the queries themselves, and the number of queries between updates, are not known in advance. For example, an analyst might query a database as requests come in, without advance knowledge of which subset of possible queries will be requested within any given period. We call these *interactive* query sequences, and call the subsequence of queries made between subsequent updates a query *round*. First, in Section 4.1, we consider how the τ-RBB mechanism introduced in Section 3.2.2 can be applied to interactive linear query sequences by releasing noisy histograms, and taking answers over those histograms instead of the private database.

In Section 4.2, we consider how the private multiplicative weights (PMW) mechanism (Algorithm 4) can be modified for the FSDD setting. Similar to the work of Cummings et al. [15], we repeat the static PMW mechanism for each update, with the public histogram carrying over from the previous time period. In Section 4.2.2 we show that adding random noise to the public histogram, when the true database is updated, decreases the expected number of above threshold queries encountered by the SVT subroutine. This improves the accuracy of the expected numeric answers by increasing the portion of the privacy budget allocated to each above threshold query. Unlike the PMWG mechanism of Cummings et al. [15], under the FSDD model we cannot rely on the growth of the database to absorb the cost of repeated querying through decreasing query sensitivity, since the size of the

FSDD is fixed. As such, in an infinite update setting, we must trade off between privacy and accuracy over time.

4.1 τ-RQ for interactive query sequences

In Chapter 3 we introduced the τ-RQ mechanism for answering repeated queries in the repeated query setting. Algorithm 6 provides a mechanism for answering repeated numerical queries, which is extended in τ-RBB (Algorithm 7), to allow any differentially private mechanism, for answering a single query, from the static setting to be extended to the FSDD setting. Linear queries (see Definition 2.1.10) can be answered by applying a query function to histogram representation of a database. As such, we can answer all linear queries over a static database by releasing a histogram only once. If this released histogram is differentially private, the post-processing theorem (Theorem 2.1.2) ensures that any queries made over it cannot introduce any additional privacy leakage. In the static setting, we can use the Laplace perturbation algorithm for histograms, $\mathcal{M}_{\text{LPH}}$ (Definition 2.1.7), to release a differentially private, noisy histogram over a private database.

We extend this to the FSDD setting by using the $\mathcal{M}_{\text{LPH}}$ as our black box mechanism in τ-RBB. Algorithm 10 formalises how interactive linear queries are answered using this method.

Algorithm 10 τ-RH: τ-RQ for interactive query sequences

```
Input: Database stream $\{D_t\}$, where each $D_t \in \mathcal{X}^n$ for $t = 0, 1, ..., T-1$. Adaptive query
       stream $\{q_{t,i}\}$ where each $q_{t,i} : \mathcal{X}^n \to \mathbb{R}$, $t = 0, 1, 2, .., T-1$, $i = 1, 2, ....$ Frequency
       parameter $\tau \in [1..T]$, privacy parameter $\varepsilon \in (0, 1)$.
Output: Query answers $\{a_{t,i}\}$ each $a_{t,i} \in \mathbb{R}$. Public histograms $\{y_t\}$, where each $y_t \in \mathbb{R}^{|\mathcal{X}|}$
1: Initialise $\tau$-RBB $(\{D_t\}, \varepsilon, \mathcal{M}_{\text{LPH}}, \tau)$
2: for each time step $t = 0, 1, 2, .., T-1$ do
3:     $y_t \leftarrow$ output of $\tau$-RBB at time $t$
4:     for each query $q_{t,i}$ for $i = 1, 2, ..., l_t$ do
5:         output $a_{t,i} \leftarrow q_{t,i}(y_t)$
```

The privacy and accuracy of Algorithm 10 are derived from those of its components: τ-RBB and $\mathcal{M}_{\text{LPH}}$.

Theorem 4.1.1 (Privacy of τ-RH)**.** *τ-RH (Algorithm 10) is ε-differentially private.*

Proof. The original database stream is only accessed by the τ-RBB mechanism, which is ε-differentially private by Theorem 3.2.7 when instantiated with an ε-differentially private mechanism. The input mechanism, $\mathcal{M}_{\text{LPH}}$, is ε-differentially private by the privacy of the Laplace mechanism. All queries performed on the public histogram do not affect the privacy of the mechanism, by post-processing (Theorem 2.1.2). Thus, τ-RH is ε-differentially private. □

How we measure the accuracy of τ-RH depends on the set of possible queries. We should choose some distance measure over the histogram that corresponds to the class of possible queries, that is, the accuracy is bounded by the worst-case accuracy of the worst-case query in the query set $\mathcal{Q}$. For example, if we allow only single bucket counting queries, the accuracy would depend on the maximum error of any bucket in the histogram, while if we allow all counting queries it would be maximum sum of errors, and, if we allow linear queries with negative weights ($w \in [-1, 1]$), the maximum sum of absolute errors. In addition to these subsets of linear queries, we may also include other types of queries that can be answered on histograms, such as top-k or median.

We now give a more detailed example, for linear counting queries. We can determine the (α, β)-accuracy and thus find an optimal choice of τ as follows.

Theorem 4.1.2 (Accuracy of τ-RH for linear counting queries)**.** *τ-RH (Algorithm 10), when limited some set of linear counting queries $\mathcal{Q} \subseteq \{0,1\}^N$, is (α, β)-accurate for*

$$\alpha = \frac{2c}{\varepsilon} \ln\left(\frac{c \cdot w_{\max}}{\beta}\right) + (\tau - 1)\,, \tag{4.1}$$

where $c = \lceil \frac{T}{\tau} \rceil$, $w_{\max} = \max_{q \in \mathcal{Q}} \sum_{i=1}^{N} w_q[i]$, where $w_q[i]$ is the weight $\in \{0,1\}$ of histogram bucket i for query q. That is, all outputs $\{a_t\}_{t=0}^{T}$ are within $\pm\alpha$ of the true answer to query q on database D_t, at time t, with probability $1 - \beta$.

Proof. From the accuracy of the Laplace Mechanism (Theorem 2.1.4) and the definition of the Laplace perturbation algorithm for histograms, $\mathcal{M}_{\text{LPH}}$ (Definition 2.1.7), we have that a single bucket count over a Laplace perturbed histogram in the bounded, static setting is (α, β)-accurate with $\alpha = (2/\varepsilon)\ln(1/\beta)$, where $2/\varepsilon$ is the scale of the Laplace noise. This generalises to $\alpha = (2/\varepsilon)\ln(k/\beta)$ when taking the sum over k buckets of the histogram.

From the definition of τ-RBB (Algorithm 7) we have that the scale of the Laplace noise at each sample step is ε/c and from Theorem 3.2.8 β scales linearly with c when taken over all samples. So, for each sample step of τ-RBB in τ-RH we have that each query is

(α, β)-accurate for $\alpha = (2c/\varepsilon)\ln(k/\beta)$ where $k \le w_{\max}$ is the number of buckets included in that query. Over the c sample steps, queries made at sample step times are therefore (α, β)-accurate with $\alpha = (2c/\varepsilon)\ln(cw_{\max}/\beta)$.

Finally, between sample steps, the effect of the updates will reduce the accuracy depending on the sensitivity of the queries themselves, since each update results in a true database that is a bounded neighbour of that at the previous time step. For counting queries the sensitivity is 1 in the bounded neighbour model, since, when we move one element from one bucket to another, the sum of the buckets will increase by 1 for an element moved from a bucket not in the query to one in the query, decrease by one for the reverse, and otherwise will not change. As such, the maximum error introduced by the up to $\tau - 1$ updates between samples is $(\tau-1)\Delta_q = (\tau-1)$, so we have that the overall (α, β)-accuracy for τ-RH has $\alpha = (2c/\varepsilon)\ln(cw_{\max}/\beta) + (\tau - 1)$. □

As in Theorem 3.2.5, we can optimise τ by choosing $\tau' = \arg\min_{\tau} \alpha$. We can further improve the accuracy of our mechanism by optimising the query set. In the worst case, a linear counting query includes all buckets in the histogram, giving $w_{\max} = N$, however, if we assume that the size of the database n is non-private, which is the usual assumption for bounded DP, then we can easily discard this query to give $w_{\max} = N - 1$. Furthermore, we have that counting queries are symmetric. For example, we have that for queries $q = [1, 1, 0, 1]$ and $q' = [0, 0, 1, 0]$, and any histogram $x \in \{0, 1\}^4$, $q(x) = n - q'(x)$. As such, we can remove the larger of any two symmetric queries, answering them using post-processing on an answer to the other, such that $w_{\max} = \lfloor N/2 \rfloor$.

Thus, for the set of all optimised counting queries over a histogram of size N, we can achieve (α, β)-accuracy with $\alpha = (2c/\varepsilon)\ln(c\lfloor N/2 \rfloor/\beta) + (\tau - 1)$.

4.2 PMW-FSDD-T

Inspired by the PMWG algorithm of Cummings et al. [15], in Algorithm 12 we give a PMW algorithm for answering linear queries under the FSDD-T model, PMW-FSDD-T. Figure 4.1 shows the operations of the algorithm at a high level. In this algorithm, we instantiate a new NumericSVT instance for each update to the true database. Like the original PMW algorithm of Hardt and Rothblum [42], a public histogram, y, initially uniformly distributed, is used to answer queries. When each query arrives, the NumericSVT subroutine is used to determine whether the true answer differs significantly from the answer on y, and returns a numeric answer if it does. We call these *hard* queries. To ensure that there is no privacy leakage (see Section 3.2 of Lyu et al. [53]), each query requires two calls to SVT, one to determine if the true answer is significantly smaller than the public answer, and another to determine if it is significantly larger. If a query is determined as hard, the public histogram y is updated using the multiplicative weights (MW) update rule. Figure 4.1 gives a high level overview of PMW-FSDD-T, while Algorithms 11 and 12 show the low level details.

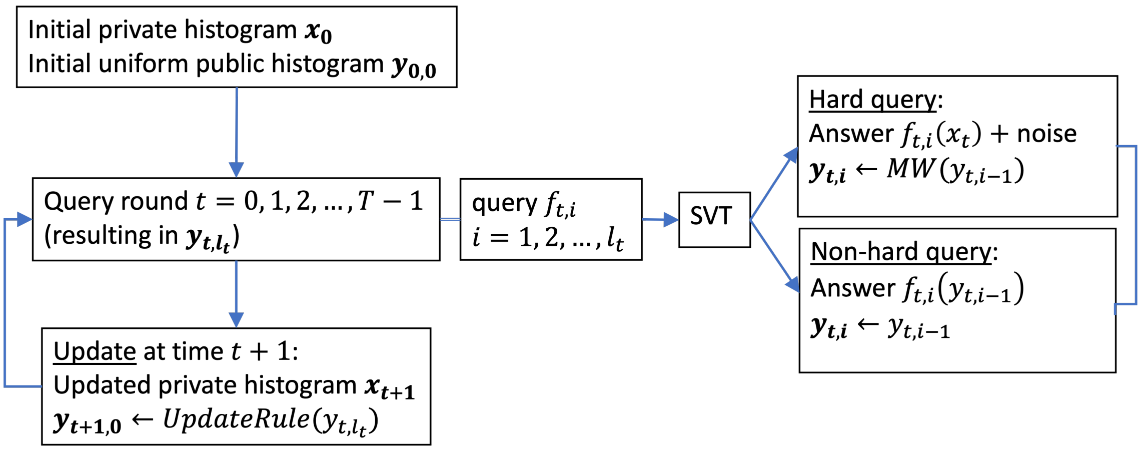
Initial private histogram x_0
Initial uniform public histogram $y_{0,0}$
Query round $t = 0, 1, 2, \dots, T-1$ (resulting in y_{t,l_t})
query $f_{t,i}$ $i = 1, 2, \dots, l_t$
SVT
Hard query: Answer $f_{t,i}(x_t)$ + noise $y_{t,i} \leftarrow MW(y_{t,i-1})$
Non-hard query: Answer $f_{t,i}(y_{t,i-1})$ $y_{t,i} \leftarrow y_{t,i-1}$
Update at time $t+1$: Updated private histogram x_{t+1} $y_{t+1,0} \leftarrow UpdateRule(y_{t,l_t})$

Figure 4.1: Flow chart showing the high level operations of PMW-FSDD-T.

Like Cummings et al. [15], we modify NumericSVT so that the hard query cutoff is managed by the PMW algorithm, rather than the SVT subroutine. We must assume that the relative entropy between the public and true histograms may increase when the database is

Algorithm 11 MSVT: Modified SVT subroutine (Based on Algorithm 6 (NSG) of Cummings et al. [16] and Algorithm 3 (NumericSparse) of Dwork and Roth [22])

Input: database D, represented as histogram x, adaptive query stream $\{\hat{q}_i\}$, $i = 1, 2, \dots$, threshold θ, privacy parameter $\hat{\varepsilon}$, cumulative cutoff $\hat{c}$.
Output: string $a \in \{\perp, \mathbb{R}\}^k$

```
1: procedure SVTSubroutine($x, \{\hat{q}_i\}, \theta, \hat{\varepsilon}, \hat{c}$)
2:     $\hat{\theta} \leftarrow \theta + \mathrm{Lap}\left(\frac{9\hat{c}}{4\hat{\varepsilon}}\right)$
3:     for each round $i$ do
4:         if $\langle \hat{q}_i, x \rangle + \mathrm{Lap}(9\hat{c}/(2\hat{\varepsilon})) \geq \hat{\theta}$ then      ▷ above threshold
5:             output $a_i \leftarrow \hat{q}_i(x) + \mathrm{Lap}(9\hat{c}/\hat{\varepsilon})$
6:             $\hat{\theta} \leftarrow \theta + \mathrm{Lap}(9\hat{c}/(4\hat{\varepsilon}))$      ▷ Resample threshold noise
7:         else output $a_i \leftarrow \perp$      ▷ below threshold
```

Algorithm 12 PMW-FSDD-T

```
Input: Fractional histogram stream $\{x_t\}$, where $|x_t| = n$ and $x_t \in \mathcal{X}$ with $|\mathcal{X}| = N$.
       Adaptive query stream $\{q_{t,i}\}$ where each $q_{t,i} \in \mathcal{Q}$, $t = 0, 1, 2, .., T-1$, $i = 1, 2, ...$.
       Threshold $\theta$, privacy parameter $\varepsilon$, accuracy parameter $\alpha$, tradeoff parameter $k$,
       and weight parameter $\eta \in (0, \frac{\alpha}{3})$. 'Outer' update function $g : \mathbb{R}^N \to \mathbb{R}^N$ and
       corresponding hard query cutoff function $\psi(\cdot)$.
Output: string $\{a_{t,i}\} \in \mathbb{R}$, public histograms $\{y_{t,i}\}$
1:  $\theta \leftarrow 2\alpha/3$
2:  $y_{0,0} \leftarrow [1/N]^N$                                   ▷ synthetic database y
3:  $\xi \leftarrow \varepsilon/(2T)$
4:  for each time step $t = 0, 1, 2, .., T-1$ do
5:      $h_t \leftarrow 0$                                         ▷ count of hard queries
6:      $c_t \leftarrow \psi(t) - \sum_{\tau=0}^{t-1} h_\tau$       ▷ max hard queries in time period
7:      Initialise SVTSubroutine$\Big(x_t, \{q'_{t,i'}\}, \theta, \xi, c_t\Big) \to \{\hat{a}_{t,i'}\}$
8:      for each query $i = 1, 2, ..., l_t$ do
9:          $q'_{t,2i-1} \leftarrow q_{t,i}(x_t) - q_{t,i}(y_{t,h_t})$, $q'_{t,2i} \leftarrow q_{t,i}(y_{t,h_t}) - q_{t,i}(x_t)$
10:         if $\hat{a}_{t,2i-1} = \perp$ and $\hat{a}_{t,2i} = \perp$ then      ▷ below threshold
11:             output $a_{t,i} \leftarrow q_{t,i}(y_{t,h_t})$
12:         else                  ▷ above threshold, hard query: perform MW update
13:             if $\hat{a}_{t,2i} = \perp$ then
14:                 output $a_{t,i} \leftarrow q_{t,i}(y_{t,h_t}) + \hat{a}_{t,2i-1}$
15:                 $r \leftarrow 1 - q_{t,i}$              ▷ increases weight of the query elements
16:             else
17:                 output $a_{t,i} \leftarrow q_{t,i}(y_{t,h_t}) - \hat{a}_{t,2i}$
18:                 $r \leftarrow q_{t,i}$
19:             for each $j \in N$ do $\hat{y}[j] \leftarrow y_{t,h_t}[j] \cdot \mathrm{e}^{-\eta r[j]}$     ▷ re-weight y
20:             $h_t \leftarrow h_t + 1$                     ▷ Increment hard query counter
21:             for each $j \in N$ do $y_{t,h_t} \leftarrow \hat{y}[j] / \sum_{k=1}^{N} \hat{y}[k]$     ▷ normalise y
22:             if $\sum_{\tau=0}^{t} h_\tau \geq \psi(t)$ then
23:                 Halt                                   ▷ Hard query budget exceeded
24:     $y_{t+1,0} \leftarrow g(y_{t,h_t})$                  ▷ according to update rule
25:     Terminate SVTSubroutine
```

updated, and calculate this potential increase in order to determine a bound on the number of above-threshold queries expected during each round, and choose privacy parameters accordingly. In PMWG, Cummings et al. [15] additionally update the public histogram y after each update to the true database, to account for the increase in entropy caused by the update. In Section 4.2.1 we consider the case where no such update is applied, and find a hard query bound that is exponential in t. In Section 4.2.2, inspired by PMWG [15], we apply uniform noise to the public histogram, and determine a hard query bound that is linear in t.

4.2.1 Case 1: no outer update rule

At line 24 of Algorithm 12 the public histogram is updated regardless of the results of the inner (query) loop. Here we attempt to find an appropriate hard query cutoff function $\psi(\cdot)$ if there is no such update. That is, if g is the identity function $g(y) = y$. In this case, the only updates to the public histogram are those performed in response to hard queries, using the MW update rule. In order to determine the entropy parameter, we first determine the maximum increase in entropy (KL-divergence, see Definition 2.1.11) caused by a single update to the true database. We then use this to determine the expected number of hard queries up to any time t.

We now consider how FSDD updates affect the relative entropy of the public and true histograms in this case.

Lemma 4.2.1. *The entropy increase caused by a single update to an FSDD database is bounded by:*

$$\mathrm{RE}\left(\hat{x}\|y\right) - \mathrm{RE}\left(x\|y\right) \le \tfrac{2}{n}\left(\log n + \eta c\right) \tag{4.2}$$

where c is the number of previous MW updates to the public histogram.

Proof. This proof follows the format and some techniques of the proof of Lemma 15 of Cummings et al. [16].

Let ΔN represent the set of fractional histograms equivalent to databases in $\mathcal{X}$, and let $x, \hat{x}, y \in \Delta N$, where $\hat{x}$ is a bounded neighbour of x. That is, we have some $j, k \in N$ such that $\hat{x}[j] = x[j] + \frac{1}{n}$, $\hat{x}[k] = x[k] - \frac{1}{n}$, and for all $i \neq j, k$, $\hat{x}_i = x_i$. In order to calculate a bound on the increase in entropy when an entry is modified, we must bound $\mathrm{RE}\left(\hat{x}\|y\right) - \mathrm{RE}\left(x\|y\right)$.

$$\begin{aligned}
\mathrm{RE}\left(\hat{x}\|y\right) - \mathrm{RE}\left(x\|y\right) &= \sum_{i=1}^{N}\left(\hat{x}[i]\log\frac{\hat{x}[i]}{y[i]} - x[i]\log\frac{x[i]}{y[i]}\right) \\
&= \left(\left(x[j] + \tfrac{1}{n}\right)\log\frac{x[j]+\frac{1}{n}}{y[j]} - x[j]\log\frac{x[j]}{y[j]}\right) \\
&\quad + \left(\left(x[k] - \tfrac{1}{n}\right)\log\frac{x[k]-\frac{1}{n}}{y[k]} - x[k]\log\frac{x[k]}{y[k]}\right) \\
&= \left(x[j] + \tfrac{1}{n}\right)\log\left(x[j] + \tfrac{1}{n}\right) - x[j]\log x[j] - \tfrac{1}{n}\log y[j] \\
&\quad + \left(x[k] - \tfrac{1}{n}\right)\log\left(x[k] - \tfrac{1}{n}\right) - x[k]\log x[k] + \tfrac{1}{n}\log y[k]\,.
\end{aligned} \tag{4.3}$$

For $x[i] \in [0,1]$ and $n \geq 1$, we have that $\left(x[i] \pm \frac{1}{n}\right) \log \left(x[i] \pm \frac{1}{n}\right) - x[i] \log x[i] \leq \frac{1}{n} \log n$, so

$$\mathrm{RE}\left(\hat{x} \| y\right) - \mathrm{RE}\left(x \| y\right) \leq \frac{1}{n}\left(2 \log n + \log \frac{y[k]}{y[j]}\right) . \tag{4.4}$$

Assuming the public histogram is only updated for above-threshold queries, we can bound $y[k]/y[j]$ using the fact that y is updated at most c times. We have that at time $t = 0$, all $y_0[i] = 1/n$, and, using the multiplicative weights update rule as in the original PMW algorithm [42] (see Lines 19-21 of Algorithm 12), at each subsequent update, $q > 0$,

$$y_q[i] = \frac{y_{q-1}[i] \mathrm{e}^{-\eta r_q[i]}}{\sum_{j=1}^{N} y_{q-1}[j] \mathrm{e}^{-\eta r_q[j]}} . \tag{4.5}$$

We first bound the numerator. Since $\eta > 0$ and $r_q[i] \in [0,1]$, we have

$$y_{q-1}[i] \mathrm{e}^{-\eta} \leq y_{q-1}[i] \mathrm{e}^{-\eta r_q[i]} \leq y_{q-1}[i] . \tag{4.6}$$

Next, we bound the denominator. Using the upper bound in Equation (4.6) and the fact that $y_{q-1} \in \Delta N$,

$$\mathrm{e}^{-\eta} = \mathrm{e}^{-\eta} \sum_{j=1}^{N} y_{q-1}[j] \leq \sum_{j=1}^{N} y_{q-1}[j] \mathrm{e}^{-\eta r_q[j]} \leq \sum_{j=1}^{N} y_{q-1}[j] = 1 . \tag{4.7}$$

So now we must have that, after each update,

$$y_{q-1}[i] \mathrm{e}^{-\eta} \leq y_q[i] \leq y_{q-1}[i] \mathrm{e}^{\eta} . \tag{4.8}$$

Now, since we have at most c of these updates, and we start with $y_0[i] = 1/n$, we have for all i,

$$\frac{\mathrm{e}^{-\eta c}}{N} = \frac{1}{n} \prod_{j=1}^{c} \mathrm{e}^{-\eta} \leq y[i] \leq \frac{1}{n} \prod_{j=1}^{c} \mathrm{e}^{\eta} = \frac{\mathrm{e}^{\eta c}}{N} . \tag{4.9}$$

Now, substituting $y[i]$, we have that

$$\mathrm{e}^{-2\eta c} = \frac{\mathrm{e}^{-\eta c}}{N} \frac{N}{\mathrm{e}^{\eta c}} \leq \frac{y[k]}{y[j]} \leq \frac{\mathrm{e}^{\eta c}}{N} \frac{N}{\mathrm{e}^{-\eta c}} = \mathrm{e}^{2\eta c} . \tag{4.10}$$

Substituting Equation (4.10) into Equation (4.3), we get

$$\mathrm{RE}\left(\hat{x} \| y\right) - \mathrm{RE}\left(x \| y\right) \leq \tfrac{1}{n}\left(2 \log n + \log \mathrm{e}^{2\eta c}\right) = \tfrac{2}{n}\left(\log n + \eta c\right) . \qquad \square$$

Using Lemma 4.2.1, we now bound the number of hard queries in each round (outer loop) of PMW-FSDD-T.

Theorem 4.2.2 (Hard query threshold of PMW-FSDD-T with no outer update rule)**.** *The bound on the number of hard queries expected at time t, c_t, of PMW-FSDD-T (Algorithm 12), instantiated with no outer update rule, is*

$$c_t \leq \begin{cases} \dfrac{\log N}{\eta\alpha' - \eta^2} & t = 0 \\ \dfrac{\log N}{\eta\phi}\left(\dfrac{2}{n\phi}+1\right)^{t-1} + \dfrac{\log n}{\eta}\left(\left(\dfrac{2}{n\phi}+1\right)^{t} - 1\right) & t > 0 \end{cases}$$

where $\phi := \alpha' - \eta - \dfrac{2}{n}$, *when* $n > \dfrac{2}{\alpha' - \eta}$.

Proof. This result is inspired by Corollary 16 of Cummings et al. [15].

From Theorem 4.10-4.14 of Dwork and Roth [22] we know that, in the non-private setting, $\mathrm{RE}\left(\hat{x_t}\|y_t\right) - \mathrm{RE}\left(x_t\|y_{t+1}\right) \geq \eta\alpha' - \eta^2$, where y_{t+1} is y_t updated using the multiplicative weights update rule, and $\alpha' \geq \eta$ is the privacy parameter of the SVT subroutine. That is, each time the synthetic database is updated, the relative entropy decreases by at least $\eta\alpha' - \eta^2$. We also know that the relative entropy at time $t = 0$ is bounded above by $\log N$, and we can bound the increase in entropy for each update to x using Equation (4.2).

As such, we can set an upper bound on the number of hard queries expected up to time t:

$$c_t \leq \frac{1}{\eta\alpha' - \eta^2}\left(\log N + \frac{2}{n}\sum_{i=1}^{t}\left(\log n + \eta c_i\right)\right). \tag{4.11}$$

When $t > 0$, this is equivalent to

$$\begin{aligned} c_t &\leq \frac{1}{\eta\alpha' - \eta^2}\left(\log N + \frac{2}{n}\left(t\log n + \eta c_t + \eta\sum_{i=1}^{t-1} c_i\right)\right) \\ \Rightarrow c_t\left(1 - \frac{2}{n(\alpha' - \eta)}\right) &\leq \frac{1}{\eta\alpha' - \eta^2}\left(\log N + \frac{2t\log n}{n} + \frac{2\eta}{n}\sum_{i=1}^{t-1} c_i\right), \end{aligned} \tag{4.12}$$

and when $n > \frac{2}{\alpha' - \eta}$, this resolves to

$$\Rightarrow c_t \leq \frac{1}{\eta\left(\alpha' - \eta - \frac{2}{n}\right)}\left(\log N + \frac{2t\log n}{n} + \frac{2\eta}{n}\sum_{i=1}^{t-1} c_i\right). \tag{4.13}$$

Now, let $w := \dfrac{1}{\eta\left(\alpha' - \eta - \frac{2}{n}\right)}$, and let $x := w \log N$, $y := \dfrac{2w \log n}{n}$, and $z := \dfrac{2w\eta}{n}$, then we have

$$c_t \leq x + ty + z \sum_{i=1}^{t-1} c_i \tag{4.14}$$

when the conditions of Equations (4.12) and (4.13) are met. In Lemma 4.2.3 we give a closed form for Equation (4.14).

Lemma 4.2.3. *For $t > 0$, $f \equiv \hat{f}$ where*

$$\begin{aligned} f(t) &:= x + ty + z \sum_{i=1}^{t-1} f(i) \\ \hat{f}(t) &:= x(z+1)^{t-1} + y\frac{(z+1)^t - 1}{z}. \end{aligned}$$

Proof. We prove this by induction.

First, when $t = 1$ we have $f(t) = \hat{f}(y) = x + y$.

Now, assume that $f(k) = \hat{f}(k)$, i.e. that

$$f(k) = x(z+1)^{k-1} + y\frac{(z+1)^k - 1}{z}.$$

Then we have that

$$\begin{aligned} f(k+1) &= x + (k+1)y + z\sum_{i=1}^{k} f(i) \\ &= x + ky + z\sum_{i=1}^{k-1} f(i) + y + zf(k) \\ &= (z+1)f(k) + y \\ &= (z+1)\left(x(z+1)^{k-1} + y\frac{(z+1)^k - 1}{z}\right) + y \\ &= x(z+1)^k + y\left((z+1)\frac{(z+1)^k - 1}{z} + 1\right) \\ &= x(z+1)^k + y\left(\frac{(z+1)((z+1)^k - 1) + z}{z}\right) \\ &= x(z+1)^k + y\left(\frac{(z+1)^{k+1} - 1}{z}\right) \qquad = \hat{f}(k+1). \end{aligned}$$

Since $f(1) = \hat{f}(1)$ and $f(k) = \hat{f}(k) \implies f(k+1) = \hat{f}(k+1)$, we have, by induction, that $f \equiv \hat{f}$. □

Finally we let $c_t \leq \hat{f}(t)$, by Lemma 4.2.3, and substitute back in the values of w, x, y, z to get

$$\begin{aligned} c_t &\leq x(z+1)^{k-1} + y\frac{(z+1)^k - 1}{z} \\ &= \frac{\log N}{\eta\left(\alpha' - \eta - \frac{2}{n}\right)}\left(\frac{2}{n\left(\alpha' - \eta - \frac{2}{n}\right)} + 1\right)^{t-1} + \frac{\log n}{\eta}\left(\left(\frac{2}{n\left(\alpha' - \eta - \frac{2}{n}\right)} + 1\right)^{t} - 1\right). \end{aligned} \tag{4.15}$$

From Equation (4.11), substututing $t = 0$, we have

$$c_0 \leq \frac{\log N}{\eta\alpha' - \eta^2},$$

and, from Equation (4.15), we have

$$c_t \leq \frac{\log N}{\eta\left(\alpha' - \eta - \frac{2}{n}\right)}\left(\frac{2}{n\left(\alpha' - \eta - \frac{2}{n}\right)} + 1\right)^{t-1} + \frac{\log n}{\eta}\left(\left(\frac{2}{n\left(\alpha' - \eta - \frac{2}{n}\right)} + 1\right)^{t} - 1\right),$$

when $t > 0$ and $n > \dfrac{2}{\alpha' - \eta}$. □

Applying Theorem 4.2.2, in PMW-FSDD-T for the case where there is no outer update rule applied, we can take

$$\psi(t) := \begin{cases} \dfrac{\log N}{\eta\alpha' - \eta^2} & t = 0 \\ \dfrac{\log N}{\eta\phi}\left(\dfrac{2}{n\phi} + 1\right)^{t-1} + \dfrac{\log n}{\eta}\left(\left(\dfrac{2}{n\phi} + 1\right)^{t} - 1\right) & t > 0 \end{cases} \tag{4.16}$$

where $\phi := \alpha' - \eta - \dfrac{2}{n}$, whenever $n > \dfrac{2}{\alpha' - \eta}$.

This hard query bound is exponential in t. To improve on this, we now consider an update rule that applies uniform noise to the public database, y after each update to the true database, x.

4.2.2 Case 2: uniform outer update rule

Since the version of our algorithm with no outer update rule has a bound on the expected number of hard queries that is exponential in t, we implement an additional update rule. This update is performed at line 24. The update rule used by Cummings et al. [15] in

PMWG assumes that updates to the database are drawn from a uniform distribution, so uniform noise is applied across the public histogram to offset the increase in entropy. Here we apply the same concept under our FSDD model.

Algorithm 13 Uniform public histogram update rule

```
Input: fractional histogram $y \in \mathbb{R}^N$
Output: fractional histogram $\hat{y} \in \mathbb{R}^N$
1: for each $j \in N$ do $\hat{y}[j] \leftarrow \frac{n-1}{n} y[j] + \frac{1}{nN}$
2: return $\hat{y}$
```

The uniform update rule, given in Algorithm 13, adds uniform noise such that public histogram will tend towards the uniform distribution when no multiplicative weights updates are applied between database updates. First we upper bound the increase in entropy caused by a single update to the true database. This is inspired by Lemma 15 of Cummings et al. [16], and only differs significantly in the $\frac{2}{n}\log n$ term from Equation (4.19), which is omitted by Cummings et al.. All other differences are primarily due to notation differences between the models.

Lemma 4.2.4. *Using the uniform update rule, the entropy increase caused by a single update to the database is bounded by:*

$$\mathrm{RE}\left(\hat{x}\|\hat{y}\right) - \mathrm{RE}\left(x\|y\right) \leq \frac{1}{n}\left((n+2)\log n - (n-1)\log\left(n-1\right) + \log N\right). \tag{4.17}$$

Proof. This proof follows the format and some techniques of the proof of Lemma 15 of Cummings et al. [16].

Assessing the relative entropy increase caused by a single update to the database, and setting L as the set of indices where $y_i \leq \frac{1}{N(n-1)}$ and H the set where $y_i > \frac{1}{N(n-1)}$, we have

$$\begin{aligned}\mathrm{RE}\left(\hat{x}\|\hat{y}\right) - \mathrm{RE}\left(x\|y\right) &= \sum_{i=1}^{N}\left(\hat{x}_i \log\frac{\hat{x}_i}{\hat{y}_i} - x_i \log\frac{x_i}{y_i}\right)\\ &= \sum_{i=1}^{N}\left(\hat{x}_i\log\hat{x}_i - x_i\log x_i\right) + \sum_{i\in\{L\cup H\}}\left(\hat{x}_i\log\frac{1}{\hat{y}_i} - x_i\log\frac{1}{y_i}\right).\end{aligned} \tag{4.18}$$

From Equation (4.4) we have that

$$\sum_{i=1}^{N}\left(\hat{x}_i\log\hat{x}_i - x_i\log x_i\right) \leq \tfrac{2}{n}\log n\,. \tag{4.19}$$

To bound the terms including $\hat{y}, y$ we will use the fact that $\hat{y}_i = \frac{n-1}{n} y_i + \frac{1}{nN}$. Taking the set L, where $y_i \leq \frac{1}{N(n-1)}$, we use $\hat{y}_i \geq \frac{1}{nN}$ to find

$$\begin{aligned}\sum_{i\in L}\left(\hat{x}_i \log\frac{1}{\hat{y}_i} - x_i \log\frac{1}{y_i}\right) &\leq \sum_{i\in L}\left(\hat{x}_i \log(nN) - x_i \log((n-1)N)\right)\\ &= \sum_{i\in L}(\hat{x}_i - x_i)\log((n-1)N) + \sum_{i\in L}\hat{x}_i \log\tfrac{n}{n-1}\\ &\leq \log((n-1)N)\sum_{i\in L}(\hat{x}_i - x_i)_+ + \log\tfrac{n}{n-1}\sum_{i\in L}\hat{x}_i\,.\end{aligned} \tag{4.20}$$

Taking the set H, where $y_i > \frac{1}{N(n-1)}$, we can use $\hat{y}_i \geq \frac{n-1}{n} y_i$ to find

$$\begin{aligned}\sum_{i\in H}\left(\hat{x}_i \log\frac{1}{\hat{y}_i} - x_i \log\frac{1}{y_i}\right) &< \sum_{i\in H}\left(\hat{x}_i \log\frac{1}{\frac{n-1}{n}y_i} - x_i \log\frac{1}{y_i}\right)\\ &= \sum_{i\in H}\left((\hat{x}_i - x_i)\log\frac{1}{y_i} + \hat{x}_i \log\tfrac{n}{n-1}\right)\\ &\leq \sum_{i\in H}(\hat{x}_i - x_i)_+\log\frac{1}{y_i} + \sum_{i\in H}\hat{x}_i \log\tfrac{n}{n-1}\\ &\leq \log((n-1)N)\sum_{i\in H}(\hat{x}_i - x_i)_+ + \log\tfrac{n}{n-1}\sum_{i\in H}\hat{x}_i\,.\end{aligned} \tag{4.21}$$

Combining Equations 4.19, 4.20, and 4.21 we find

$$\begin{aligned}\mathrm{RE}(\hat{x}\|\hat{y}) - \mathrm{RE}(x\|y) &\leq \tfrac{2}{n}\log n + \sum_{i=1}^{N}\left(\hat{x}_i \log\tfrac{n}{n-1} + (\hat{x}_i - x_i)_+\log((n-1)N)\right)\\ &= \tfrac{2}{n}\log n + \log\tfrac{n}{n-1} + \tfrac{1}{n}\log((n-1)N)\\ &= \tfrac{1}{n}\left((n+2)\log n - (n-1)\log(n-1) + \log N\right)\,. \quad \square\end{aligned} \tag{4.22}$$

Similar to Equations (4.11) and (4.13), we can set an upper bound on the number of hard queries expected up to time t. Since our entropy bound does not include c_t, we are able to find a closed form hard query threshold using the uniform update rule, given in Theorem 4.2.5

Theorem 4.2.5 (Hard query threshold of PMW-FSDD-T with uniform update)**.** *The bound on the number of hard queries expected at time t, c_t, of PMW-FSDD-T (Algorithm 12), instantiated with the uniform update rule (Algorithm 13), is*

$$c_t \leq \frac{1}{\eta\alpha' - \eta^2}\left(\log N + \frac{t}{n}\left((n+2)\log n - (n-1)\log(n-1) + \log N\right)\right)\,.$$

Proof. This result is inspired by Corollary 16 of Cummings et al. [15].

From Theorem 4.10-4.14 of Dwork and Roth [22] we know that, in the non-private setting, $\mathrm{RE}\left(\hat{x_t}\|y_t\right) - \mathrm{RE}\left(x_t\|y_{t+1}\right) \geq \eta\alpha' - \eta^2$, where y_{t+1} is y_t updated using the multiplicative weights update rule, and $\alpha' \geq \eta$ is the privacy parameter of the SVT subroutine. That is, each time the synthetic database is updated, the relative entropy decreases by at least $\eta\alpha' - \eta^2$. We also know that the relative entropy at time $t = 0$ is bounded above by $\log N$, and we can bound the increase in entropy for each update to x using Lemma 4.2.4.

$$\begin{aligned} c_t &\leq \frac{1}{\eta\alpha' - \eta^2}\left(\log N + \sum_{\tau=1}^{t}\frac{1}{n}\left((n+2)\log n - (n-1)\log(n-1) + \log N\right)\right) \\ &= \frac{1}{\eta\alpha' - \eta^2}\left(\log N + \frac{t}{n}\left((n+2)\log n - (n-1)\log(n-1) + \log N\right)\right). \end{aligned}$$ □

Thus, in this case, we can take

$$\psi(t) := \frac{1}{\eta\alpha' - \eta^2}\left(\log N + \frac{t}{n}\left((n+2)\log n - (n-1)\log(n-1) + \log N\right)\right).$$

This hard query bound is linear in t, a significant improvement over Theorem 4.2.2.

4.2.3 Privacy guarantees

Now we consider the privacy of Algorithm 12, PMW-FSDD-T. Since the outer update rule is a post-processing step, the privacy guarantee does not depend on the update rule used. In order to show that PMW-FSDD-T is differentially private we must first guarantee the differential privacy of the SVT subroutine. The privacy of MSVT follows from the privacy of the NumericSVT (Algorithm 2).

Lemma 4.2.6 (Privacy of Modified SVT subroutine)**.** *The modified SVT subroutine, MSVT (Algorithm 11), when used in conjunction with PMW-FSDD-T (Algorithm 12), is ξ-differentially private.*

Proof. The modified SVT subroutine at round t is equivalent to the NumericSparse algorithm of Dwork and Roth [22] with an unbounded neighbour definition. We set $x = x_t, \{q_i\} = \{\hat{q}'_{t,i}\}, \theta = 2\alpha/3, \hat{\varepsilon} = \varepsilon/(2T), c = c_t$, and enforce the hard query threshold within PMW-FSDD-T itself, as in the NSG subroutine of PMWG by Cummings et al. [16].

By the privacy of NumericSparse, and converting from unbounded to bounded DP (see discussion of Definition 2.1.3), the modified SVT subroutine at round t must therefore be (ε/T)-differentially private. In general, where $\hat{\varepsilon} = 2\xi$, we have that the subroutine is $\hat{\varepsilon}$-differentially private. □

With the privacy of the subroutine established, we apply basic composition to prove the privacy of PMW-FSDD-T, since the true database is not accessed outside the subroutine.

Theorem 4.2.7 (Privacy of PMW-FSDD-T)**.** *PMW-FSDD-T (Algorithm 12) is ε-differentially private.*

Proof. By Lemma 4.2.6 we have that the modified SVT subroutine for round t is 2ξ-differentially private. Since the true histogram x is only queried directly via the SVT subroutine, and the SVT subroutine is re-initialised for each round t, we have, by basic composition (Theorem 2.1.3) that PMW-FSDD-T is $(\sum_{t=0}^{T} \xi_t)$-differentially private. By definition we have:

$$\sum_{t=0}^{T-1} 2\xi = \sum_{t=0}^{T-1} \frac{\varepsilon}{T} = \varepsilon\,.$$ □

4.3 PMW-FSDD-∞

Now we consider the infinite update setting, whereby there is no limit, T, on the number of time periods. As mentioned previously, we cannot rely, like Cummings et al. [15] did in their PMWG mechanism, on the decreasing sensitivity of queries on a database increasing in size. Instead, we allocate our privacy budget so that the total privacy allocation does not exceed ε. We do this by calculating the privacy parameter for each round using a convergent series. This means that the accuracy is likely to decrease significantly over time.

For our convergent series, we choose the geometric series. To implement this, we remove Line 3 of Algorithm 12, instead calculating a fresh privacy parameter $\xi_t \leftarrow \frac{\varepsilon}{2k} \cdot \left(\frac{k-1}{k}\right)^t$, for some $k > 1$ inside the outer loop. ξ_t is passed as the privacy parameter to the MSVT instance for time t. PMW-FSDD-∞ is otherwise identical to PMW-FSDD-T. This creates a geometrically decreasing series of ξs such that $\sum_{t=0}^{\infty} \xi_t$ converges to ε. Thus, the accuracy of the SVT subroutine decreases over time, with the rate of this decrease dependent on the choice of k. Below, we prove the privacy of this version, however any convergent series could be used in place of the geometric series with similar privacy results.

Lemma 4.3.1 (Privacy of Modified SVT subroutine in the infinite update setting)**.** *The modified SVT subroutine, Algorithm 11, when used in conjunction with PMW-FSDD-∞, is $\hat{\varepsilon}$-differentially private.*

Proof. The modified SVT subroutine at round t is equivalent to NumericSVT with an unbounded neighbour definition, with $x = x_t, \{f_i\} = \{\hat{f}'_{t,i}\}, \theta = \frac{2\alpha}{3}, \varepsilon = \frac{\varepsilon}{2k} \cdot \left(\frac{k-1}{k}\right)^t, c = c_t, \delta = 0$ with the hard query threshold enforced by PMW-FSDD.

By the privacy of NumericSVT, and using group privacy to convert from unbounded to bounded DP, the modified SVT subroutine at round t must therefore be $(\frac{\varepsilon}{k} \cdot \left(\frac{k-1}{k}\right)^t, 0)$-differentially private. In general, where $\hat{\varepsilon} = 2\xi_t$ for some t, we have that the subroutine is $\hat{\varepsilon}$-differentially private. □

Theorem 4.3.2 (Privacy of PMW-FSDD-∞)**.** *PMW-FSDD-∞ is ε-differentially private.*

Proof. By the privacy of MSVT we have that the modified SVT subroutine for round t is ξ_t-differentially private. Since the true database x is only queried directly via the SVT subroutine, and the SVT subroutine is re-initialised for each round t, we have, by basic composition that PMW-FSDD is $(\sum_{t=1}^{\infty} \xi_t)$-differentially private. By definition we have:

$$\sum_{t=1}^{\infty} 2\xi_t = \sum_{t=1}^{\infty} \left(\frac{\varepsilon}{k} \cdot \left(\frac{k-1}{k}\right)^t\right)$$

for some $k > 1$, and by the geometric series we have:

$$\sum_{t=1}^{\infty} \left(\frac{\varepsilon}{k} \cdot \left(\frac{k-1}{k}\right)^t\right) = \frac{\varepsilon}{k} \cdot \frac{1}{1 - \frac{k-1}{k}} = \varepsilon.$$ □

In this chapter we have outlined how the τ-RBB mechanism can be utilised to answer interactive queries, and provided an alternative in a modification of the PMWG algorithm of Cummings et al. [15] to the FSDD setting. We have provided proof of the privacy of these mechanisms, as well as a bound on the (α, β)-accuracy of τ-RH for the set of linear counting queries. In Section 5.3 we compare the accuracy of these methods empirically.

Chapter 5

Experimental Evaluation

In this chapter, we present the results of experimental evaluation of the algorithms introduced in Chapters 3 and 4. We show that, in the FSDD context, our τ-RQ mechanisms often provide better accuracy than adaptive alternatives such as the sparse vector technique (SVT) and the private multiplicative weights technique (PMW), while maintaining the same privacy guarantees.

In Section 5.2 we compare τ-RQ to an SVT-based mechanism, ARQ, for FSDD data. We first present analysis over a variety of synthetic data sets and then compare those to a real data set. We find that τ-RQ consistently provides better accuracy than ARQ for data streams that shift in distribution over time, while ARQ sometimes performs better than τ-RQ when data remains uniformly distributed. In Section 5.3 we evaluate τ-RBB against PMW for mean and histogram queries. We find that τ-RBB performs significantly better than PMW across all experiments.

5.1 Experiment setup

Before presenting experimental results, we first describe the protocols, data sets, and queries used throughout.

5.1.1 Protocols

All data synthesis and experiments are performed on either macOS with a 2.3 GHz Quad-Core Intel Core i7 CPU, Intel Iris Plus Graphics 1536 MB GPU and 16 GB memory, or on a Linux virtual machine running Ubuntu with an AMD Epyc 7763 64-core processor (using 4 cores) and 16 GB memory. All mechanisms are implemented using Python 3.11.4, with all randomness generated using methods of the NumPy 1.24.2 Random Generator pseudo-random number generator.

Unless otherwise specified, each experiment is repeated 100 times, each with unique random seeds for each trial, and the same seed sequence used for each experiment, for replicability. For the repeated query streams, we use two lists of 100 random seeds, one for random noise generation and one for data stream generation. The two lists are initially generated from arbitrary, system-generated random seeds, and used consistently throughout. For interactive query streams, both the data and queries are generated once for each parameter set, from arbitrary, system-generated random seeds, and noise is generated from a single list of 100 random seeds. In all experiments we are interested in the absolute additive error of our differentially private mechanisms, relative to the non-private query answers.

There are some limitations to the experiments completed. First, the scope is limited by time and resource constraints. Our experiments involving the ARQ and PMW mechanisms do not include update streams larger than $T = 2000$, and use limited sets of parameters, due to their run-time. The real data available for use in experiments was limited as we do not have access to datasets matching the FSDD parameters. Second, the code used to run these experiments is not suitable for deployment. Pseudo-random noise has been sampled using a non-private library. While this is suitable to examine the noise magnitude of mechanism that are private in theory, it is vulnerable to side-channel attacks, such as those described by Mironov [57].

5.1.2 Synthetic data

We now describe synthetic data used in experiments. Since all of our chosen queries can be performed on a histogram of counts, we represent each database as a histogram. This improves the efficiency of storing the database stream and performing queries. An initial database is generated for time $t = 0$, then an update applied according to some update rule for each time $t \in [1..T-1]$. A *data stream* is this sequence of T histograms, including the initial histogram and the histograms resulting from each update.

We have three synthetic data stream types, with varying degrees and types of randomness, chosen to demonstrate the behaviour of our mechanisms in a variety of scenarios. A *uniform* data stream is a low entropy, randomly generated set of updates, starting with a (near-)perfectly uniform histogram at time 0. A *binomial* data stream is shifting from a binomial distribution with a small to a large success probability as it is updated, giving it a relatively large expected update effect for most query types. A *sharp-shift* data stream is deterministic, and contrived such that the update effect, for most query types, is very small during the first half of updates and very large during the second half. Except where synthetic data streams are generated to match the parameters of real data, all streams have $N = 10$ histogram buckets.

Uniform data stream. We use uniformly random data as an example of data that behaves furthest from what we would expect to cause worst-case error, since the cumulative effect of updates is negligible in expectation.

Algorithm 14 Uniform synthetic data stream generator

```
Input: Time bound $T$, database size $n$, data universe size $N$
Output: List of histograms $x$ indexed $0, ..., T-1$
 1: if $n$ mod $N = 0$ then                          ▷ can be uniformly bucketed
 2:     $x_0 \leftarrow N$ buckets of size $n/N$
 3: else
 4:     $r \leftarrow n$ mod $N$                        ▷ number to be randomly distributed
 5:     $x_0 \leftarrow N$ buckets of size $(n-r)/N$
 6:     for $i \leftarrow 1, ..., r$ do
 7:         Increment random bucket of $x_0$
 8: for $t \leftarrow 1, ..., T-1$ do
 9:     $x_t \leftarrow x_{t-1}$
10:     Decrement a random, positive valued, bucket of $x_t$
11:     Increment a random bucket of $x_t$
```

The initial uniform database is generated as a histogram. If possible, each bucket contains exactly the same count n/N; otherwise, where n mod $N = r$, each bucket has an initial count $(n-r)/N$, with the remaining r records distributed between buckets uniformly at random. The updates are generated uniformly at random: one random, non-zero bucket has one record removed, and one bucket has one record added, to simulate a state change for a single individual. It is possible for the record to move to the same bucket it came from during a single update, resulting in no change to the histogram. For each random update seed, both the initial histogram x_0 and the updated histograms $x_1, ..., x_N$ are generated from the same seed during computation. Algorithm 14 shows the process used to generate the histograms and Figure 5.1 plots example histograms over time, demonstrating how the updates affect the data.

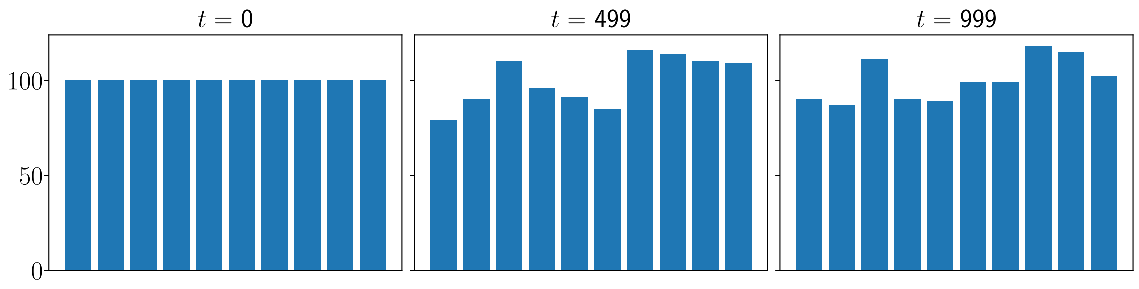

Figure 5.1: Histograms over time for uniform data for a single random seed. Parameter choices are $N = 10, T = 1000, n = 1000$.

Binomial data stream. We generate a data stream shifting from one binomial distribution to another as an example of a randomised update distribution with a large expected cumulative effect over time. This simulates a population trend shifting from one median state to another. The initial histogram is generated from a binomial distribution with success probability $p_0 = 0.2$. To generate an update, the *from* bucket is selected such that each individual has equal probability of being selected, by weighting bucket probabilities by their counts. The *to* bucket is chosen randomly according to a binomial distribution with success probability $p_u = 0.8$. This process is described in Algorithm 15. The initial histogram for each set of parameters is generated once from an arbitrary, system-generated random seed and fixed across all experiments. The updates are generated using the update seed list. The binomial distribution is simulated using the numpy.random.Generator.binomial function with parameters $(N - 1, p)$.

Algorithm 15 Binomial synthetic update generator

```
Input: Time bound $T$, database size $n$, data universe size $N$, initial histogram $x_0 \sim$ Binomial$(N-1, 0.2)$
Output: List of updated histograms $x$ indexed $1, ..., T-1$
1: for $t \leftarrow 1, ..., T-1$ do
2:     $x_t \leftarrow x_{t-1}$
3:     Choose a bucket randomly, with probability proportional to the bucket counts
4:     Select a random index $i$ from $1, ..., N$ according to Binomial$(N-1, 0.8)$
5:     Increment $x_t[j]$
```

For all updates, the new state of the updated individual is likely to be a high value, as it is drawn from a binomial distribution with a high probability of success. Since the initial data is drawn from a binomial distribution with a low probability of success, the individuals updated in early updates are likely to start in a state with a low value, and thus the effect of the update, for queries correlated to the sum of the data values (such as mean queries, and the count of a high-valued bucket), will be large. As the updates continue, the data distribution stabilises toward the binomial distribution with a high probability of success, and the expected update effect reduces. Figure 5.2 plots example histograms over time, demonstrating these update effects.

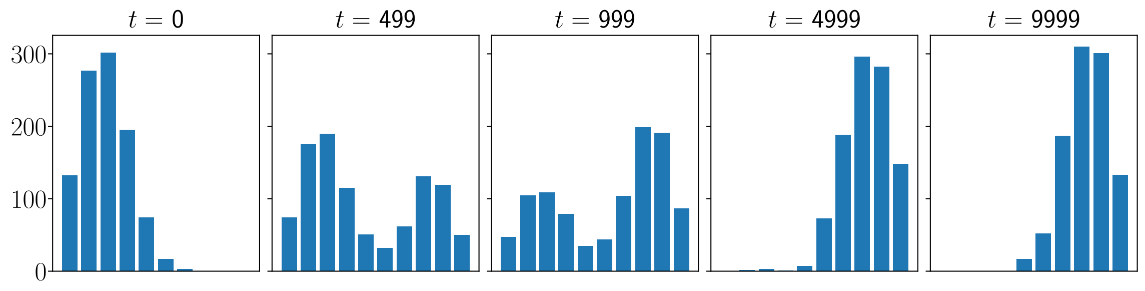

Figure 5.2: Histograms over time for binomial synthetic data for a single random seed. Parameter choices are $N = 10, T = 10000, n = 1000$.

Sharp-shift data stream. The sharp-shift data stream, generated according to Algorithm 16, is contrived to produce small update effects during the first half of updates and large update effects during the second half of updates, such that the best choice of sample frequency is dramatically different at different times. Figure 5.3 plots histograms over time, demonstrating this effect. This stream is deterministic, with all n records in bucket 1 at time 0, shifting to bucket 2 for the first half of the updates and then to bucket N for the second half of updates, such that the update effect, for most query types, is small during the first half of update and large during the second half.

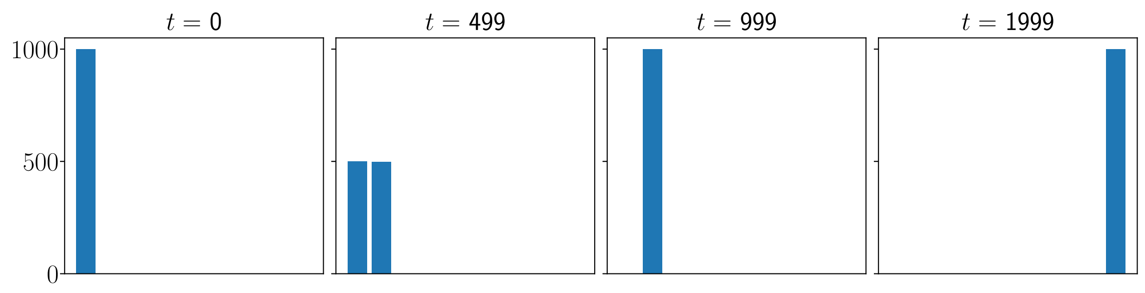

Figure 5.3: Histograms over time for sharp-shift synthetic data for a single random seed. Parameter choices are $N = 10, T = 2000, n = 1000$.

Algorithm 16 Sharp-shift synthetic data stream generator

```
Input: Database size $n$, data universe size $N$, time bound $T \leq 2n$
Output: List of histograms $H$ indexed $0, ..., T-1$
1: $x_0[1] \leftarrow n$, $x_0[2, ..., N] \leftarrow 0$
2: for $t \leftarrow 1, ..., \lfloor (T-1)/2 \rfloor$ do
3:     $x_t \leftarrow x_{t-1}$
4:     Decrement $x_t[1]$
5:     Increment $x_t[2]$
6: for $t \leftarrow \lceil T/2 \rceil, ..., T-1$ do
7:     $x_t \leftarrow x_{t-1}$
8:     if $x_t[1] > 0$ then Decrement $x_t[1]$
9:     else Decrement $x_t[2]$
10:    Increment $x_t[N]$
```

5.1.3 Real data

The Adult data set, obtained from the UCI Machine Learning Repository [1], is taken from the 1994 US census and contains 32,561 records, each with 15 attributes. Table 5.1 describes the subsets of the Adult data set used in the experiments. We use the Age attribute in experiments, and split it, according to the Income attribute, into an initial set (where income $\leq$ 50K) and an update set (where income $>$ 50K). The number of updates is limited by the size of the update set, so that $T = 7841$. At each time $t \in [1..T]$ an element in the database is selected, uniformly at random, and replaced with an element drawn, uniformly at random, from the update set. Figure 5.4 shows how this data stream changes over time. This method of generating a dynamic data stream from a static set of census data is inspired by a similar technique used by Li et al. [52] to generate time-series data.

Table 5.1: Attributes of the Adult data used in experiments

Attribute	type	range	notes
Age	integer	[17..90]	Technically categorical (ordinal), where the '90' category includes all individuals aged $\geq$ 90.
Income	categorical (ordinal)	{$\leq$50K, $>$50K}	set sizes 24,720 and 7,841 respectively

[1]`https://archive.ics.uci.edu/dataset/2`, DOI: 10.24432/C5XW20

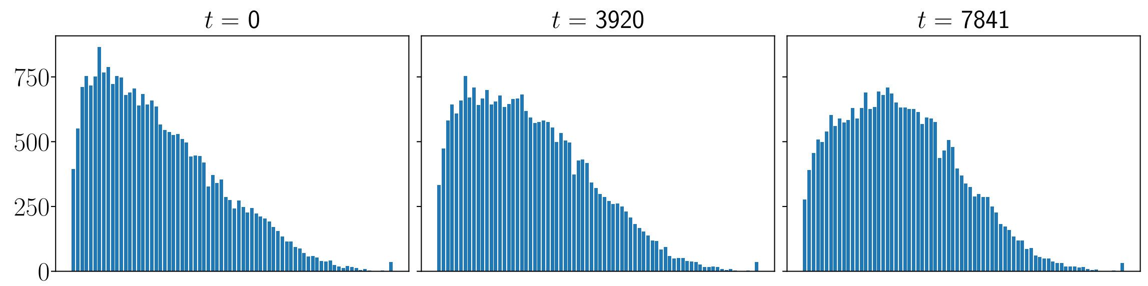

Figure 5.4: Histograms over time for Adult (Age) data stream for a single random seed.

5.1.4 Queries

Now we describe the queries used in our experiments. In the repeated query setting, where a single query is repeated after each update, we select three common queries. In the interactive setting, we must also generate streams of queries over time. For simplicity, and to align with the design of PMW, all interactive queries are linear counting queries.

Repeated queries. For repeated query experiments, three queries are used in experimental analysis. For a summary statistic, we choose the *mean query*, which calculates the mean over real-valued buckets of a histogram. For a targeted query, we choose a *count query*, counting the number of database elements in the final bucket of the histogram. Finally, for a bad case in terms of the privacy/accuracy trade-off, we choose the *histogram query*, returning the entire histogram. Table 5.2 summarises the true (non-noisy) definitions of these queries, as well as the distance measures used in ARQ calculations and the sensitivity used in noise scales.

Table 5.2: Queries used in repeated query experiments, for database $D = D_1, ..., D_n$ or count histogram $x = \{x[1], ..., x[N]\}$ where $x[i] = |\{j : D[j] = \mathcal{X}[i]\}|$ over data universe $\mathcal{X} = \{\mathcal{X}[1], ..., \mathcal{X}[N]\}$. Where $\mathcal{X}$ is ordinal, we assume it indexed in ascending order.

Query name	Database function $q(D)$	Histogram function $q(x)$	Distance measure $d(q(D), q(D'))$	Sensitivity Δ_q
mean	$\frac{1}{n}\sum_{i=1}^{n} D[i]$	$\frac{1}{n}\sum_{i=1}^{N} x[i] \cdot \mathcal{X}[i]$	$\lvert q(D) - q(D')\rvert$	$\frac{\mathcal{X}[N]-\mathcal{X}[1]}{n}$
count	$\lvert\{i : D[i] = \mathcal{X}[N]\}\rvert$	$x[N]$	$\lvert q(D) - q(D')\rvert$	1
histogram	$\{x[1], ..., x[N]\}$	x	$\sum_{i=1}^{N} \lvert x[i] - x'[i]\rvert$	2

Where each individual in the database takes a value in $[1..N]$ at each time $t \in [0..T-1]$, Figure 5.5 shows how the mean value across all individuals changes over time for the three synthetic data streams, where $N = 10$, time bound $T = 2000$, and number of individuals $n = 1000$. The distribution of histograms and bucket counts over time are shown in Figures 5.1 to 5.4

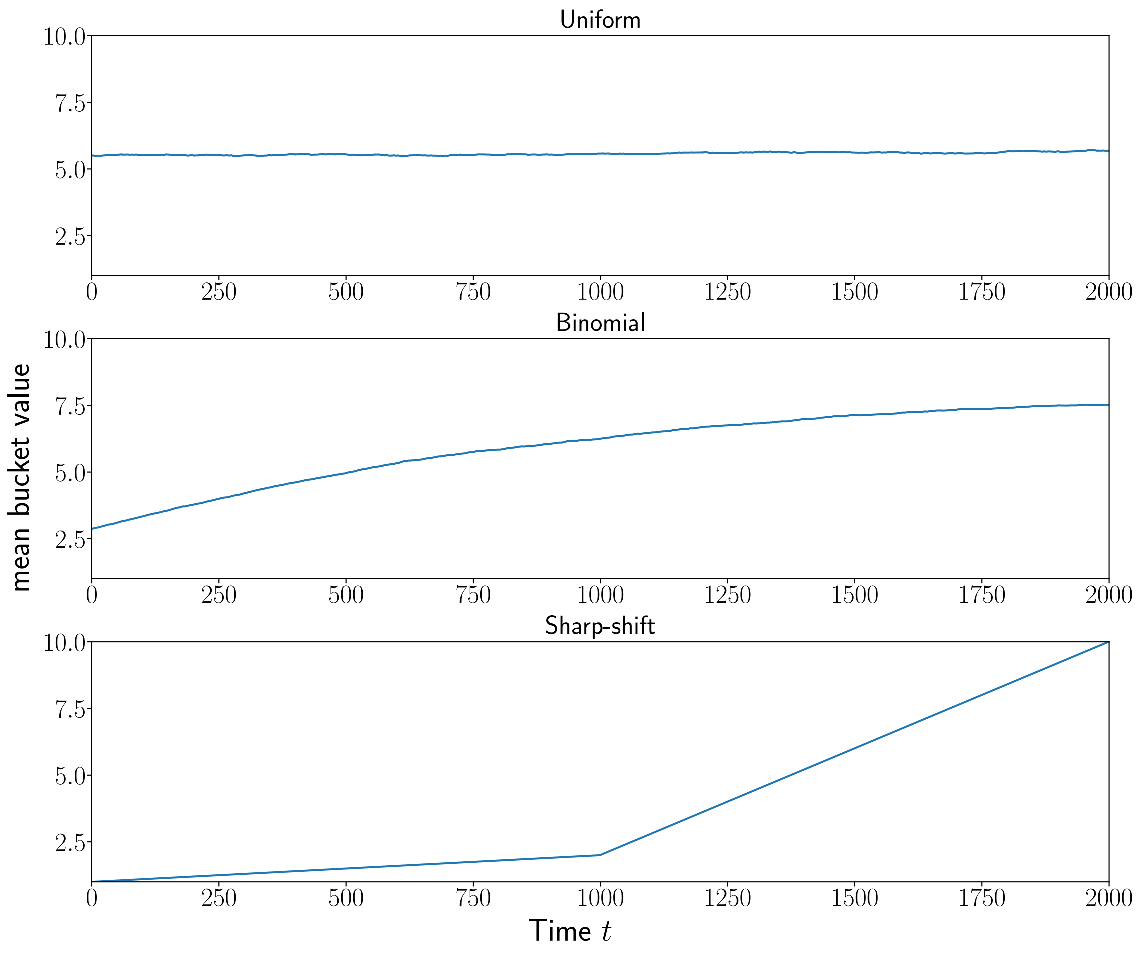

Figure 5.5: Mean individual state value (in $[1, N]$) over time for uniform, binomial, and sharp-shift synthetic data streams for a single random seed. Parameter choices are $N = 10$ histogram buckets, $T = 2000$ time periods, and $n = 1000$ individuals.

Interactive queries. For interactive query experiments, we must have a set of possible queries, and some ordered subset of those queries asked between updates. We have chosen five sets of possible linear queries, of various sizes. The largest is the set of all linear counting queries, followed by range queries, and finally single bucket counts. The sets of counting and range queries can be optimised, removing the queries that count all buckets and no buckets, as well as symmetric queries. For example, for a histogram of size $N = 3$, the query [011] can be answered by subtracting the answer to query [100] from n or 1 for histograms and normalised histograms respectively. Table 5.3 described the details of these query sets.

Algorithm 17 shows how we generate synthetic interactive query streams. The number and order of queries for each time t are chosen uniformly at random. For each experiment, where the query set is $\mathcal{Q}$, some $k \leq |\mathcal{Q}|$ is chosen to limit the number of queries that

Table 5.3: Query sets used in interactive query experiments.

Query set $\mathcal{Q}$	Description	$\lvert\mathcal{Q}\rvert$ when $N = 10$
All count	All linear queries with weight in $\{0, 1\}$, excluding the all 0 and all 1 queries	$2^N - 2 = 1022$
Optimised count	All count queries with larger of symmetric pairs removed	$(2^N - 2)/2 = 511$
All range	All range queries, excluding the all 0 and all 1 queries	54
Optimised range	All range queries with larger of symmetric pairs removed	45
Single count	Single bucket counts	$N = 10$

may be asked in each time period. For each time period, the number of queries is chosen uniformly from $[0..k]$ and the queries are chosen uniformly at random from $\mathcal{Q}$, with no query repeated in a single time period. Query streams are generated in advance, with one query stream generated for each set of parameters $(N, T, \mathcal{Q}, k)$, with the same stream used for all experiments with the same query parameters.

Algorithm 17 Interactive stream generator

```
Input: Query set $\mathcal{Q}$, query limit $k \leq |\mathcal{Q}|$, time bound $T$
Output: Query stream $Q_0, ..., Q_{T-1}$, where each $Q_i$ is some ordered subset of $\mathcal{Q}$ and
        $|Q_i| \leq k$
1: for $t \leftarrow 0, ..., T-1$ do
2:     $m \in [0..k]$ chosen uniformly at random
3:     for $i \leftarrow 1, ..., m$ do
4:         $Q_t[i] \in \mathcal{Q}/Q_t[: i-1]$ chosen uniformly at random
```

Real query streams. We use the Kosarak click-stream dataset[2] as an interactive query stream. The original dataset contains 990,002 sequences of integers between 1 and 41,270, representing sequences of website clicks from a Hungarian news portal. We transform these integers to 16-bit binary strings, used as linear count queries on our synthetic data streams as described in Section 5.1.2, with $N = 16$.

For experiments where time bound T is less than the total number of sequences in the Kosarak dataset, we select the first T sequences as our query stream. Table 5.4 shows the median and maximum sequence length, and how frequently each of the 16 histogram buckets are included in a linear count query. We see that the median sequence lengths and buckets per query remain stable at around 3 and 5, respectively, while the maximums

[2]`http://fimi.uantwerpen.be/data/kosarak.dat`

increase as we include more sequences. In the case of sequence lengths, this is explained by long sequences of queries being relatively rare.

Table 5.4: Sequence lengths and buckets per linear count query of Kosarak query streams of size $T = \{100, 1000, 2000, 990002\}$

T	Sequence lengths			Buckets per query		
	Med	Mean	Max	Med	Mean	Max
100	3	7.75	182	4	3.83	8
1000	3	8.58	430	5	4.52	11
2000	3	8.39	430	5	4.62	12
990,002	3	8.10	2498	5	4.82	15

Figure 5.6 shows how frequently each of the histogram buckets is included in a query, as a fraction of all queries. Note that the fractions do not sum to 1 because multiple buckets can appear in a single query. We see that higher bucket values occur much more frequently than lower value buckets. This is due to large integers being rare in the original dataset, and explains why queries containing large numbers of buckets do not often appear.

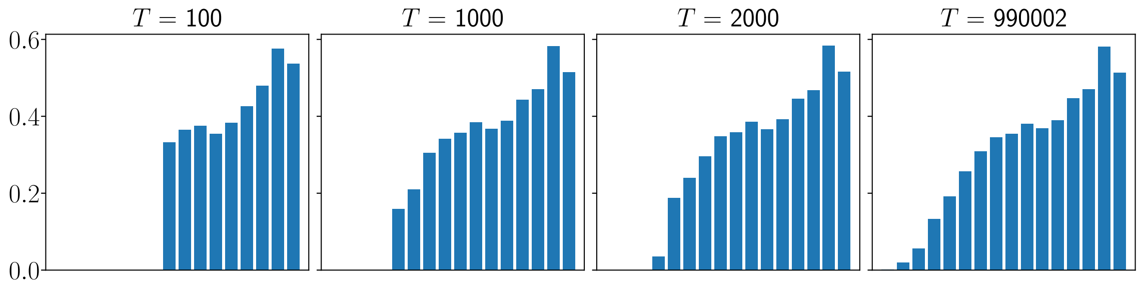

Figure 5.6: Frequency of histogram buckets in kosarak queries, as proportion of all queries, for various sized subsets and all queries.

5.2 Repeated query experiments

We now show the findings of experiments in the repeated query setting. In Sections 5.2.1 and 5.2.2 we show experimental analysis of τ-RQ and τ-RBB, on the synthetic data described in Section 5.1.2 and queries described in Section 5.1.4. First, we give an example of how the best choice of τ-RQ differs between the theory and experiments, then we compare τ-RQ and ARQ. We find that τ-RQ has lower error in most experiments, but has similar error to ARQ for the uniform data stream. In Section 5.2.3 we repeat the comparison experiments using a real data set as described in Section 5.1.3.

Throughout, DP mean and count queries are performed using τ-RQ and DP histogram queries using τ-RBB instantiated with the Laplace mechanism for histograms described in Definition 2.1.7.

5.2.1 Experimental results vs. theoretical bounds

Before comparing τ-RQ to an ARQ baseline, we examine how it performs on a synthetic data set relative to the theoretical error bound shown in Theorem 3.2.4.

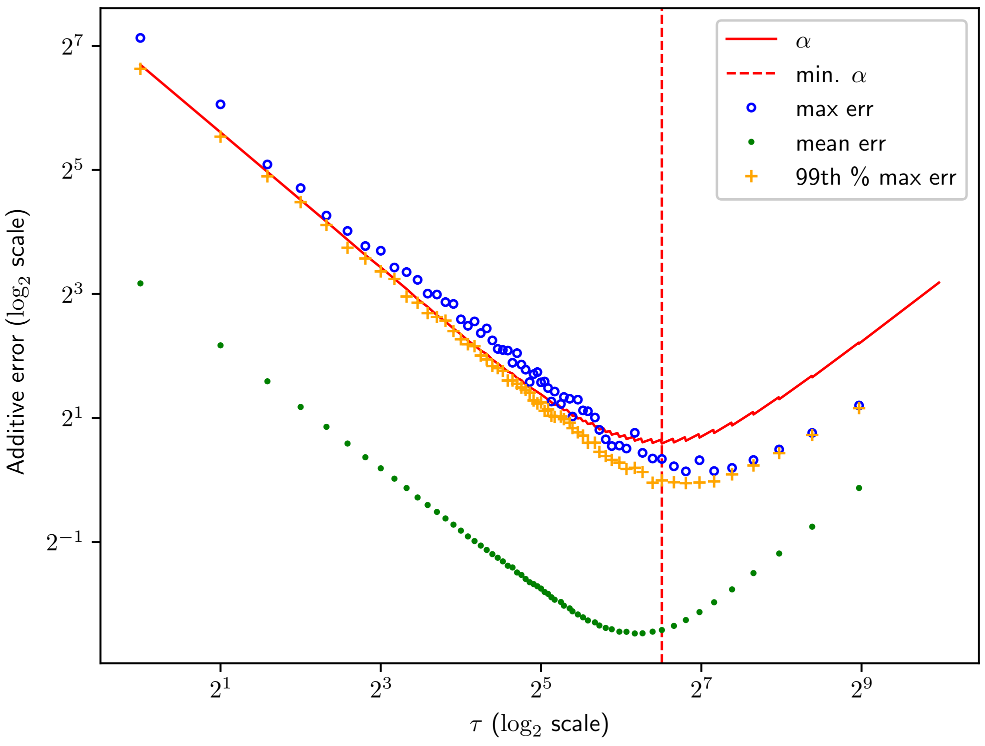

Figure 5.7: Additive error of τRQ-FSDD experiment outputs for the mean function, for various choices of τ, plotted against theoretical accuracy bound α. Parameter choices are $\beta = 0.01, \varepsilon = 1.0, N = 10, T = 1000, n = 1000$. Experiment outputs show the mean of mean errors, max of max errors, and 99th percentile of max errors across 1000 trials for each sample frequency parameter τ.

Figure 5.7 shows the results of experiments running τ-RQ, on the binomial data stream, for the mean query against various choices of $\tau \in [1..T]$. Plots are shown for privacy parameter $\varepsilon = 1.0$ and database size $n = 1000$. The theoretical upper bound on error α is shown for comparison. By choosing $\beta = 0.01$ we expect that one out of the 100 trials may have a maximum error exceeding α.

As in Figure 3.4 we see that, at first, error decreases as τ increases. With low τ we have higher cutoff $c = \lceil \frac{T}{\tau} \rceil$, and thus larger Laplacian noise. Once τ becomes significantly large we see that the effect of updates begins to dominate, and error begins increasing. In this example, the synthetic data does not have the worst case effect of Δ_q for every update, so we see that as the update effect dominates, the experimental results begin to diverge from the theoretical bound.

5.2.2 Results on synthetic data streams

In this section we show the results of both τ-RQ and ARQ across the synthetic data streams and queries described in Section 5.1.2. We first show results for a fixed set of parameters for all datasets and queries, then the results of varying parameters ε, T, and n. We find that τ-RQ generally performs better than ARQ for the binomial and sharp-shift streams, and that both methods have similar accuracy for the uniform stream. A high-level discussion of the results is provided in Section 5.4.

We implement ARQ as in Algorithm 9, with the exception that we use an optimised NumericSVT, as in Algorithm 3, as the subroutine. While we are not able to prove the privacy of the optimised NumericSVT for adaptive queries, it has better accuracy than standard NumericSVT, providing a more competitive experimental baseline. ARQ experiments are run using the same random seeds as in τ-RQ for each of the 100 trials. Since we do not have a general method for choosing the hard query cutoff c and the decision fraction d for ARQ, we run τ-RQ for all choices of τ in the *minimum distinct* τ set described in Section 3.2, and ARQ for equidistant choices of c, such that the number of choices is close to the number of choices of τ, and with $d \in 0.25, 0.5, 0.75$. Thus, we are comparing multiple instantiations of both τ-RQ and ARQ, and presenting only the results of the instantiation with the smallest error for each mechanism and measure.

We give three measures of absolute error: *median of medians* gives the median of the set containing the minimum median error for each of the 100 trials; *median of means* gives the median of the set containing the minimum mean error for each of the 100 trials; *max* gives the median of the set containing the minimum maximum error for each of the 100 trials. For each trial, the *minimum* of each of the measures is taken across each choice of τ for τ-RQ and all combinations of c and d for ARQ, where the measure itself is across all T time periods in that trial.

Figure 5.8 shows, at a high level, the process and outputs of the experiment simulations.

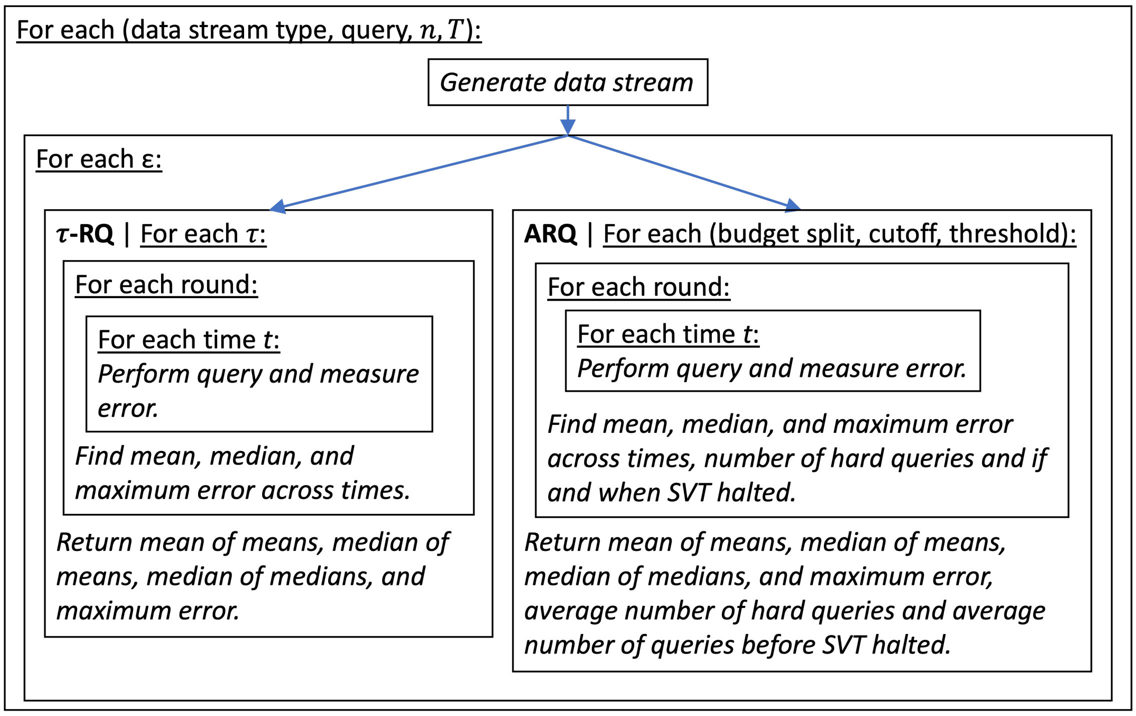
For each (data stream type, query, n, T):
Generate data stream
For each ε:
τ-RQ | For each τ:
For each round:
For each time t:
Perform query and measure error.
Find mean, median, and maximum error across times.
Return mean of means, median of means, median of medians, and maximum error.
ARQ | For each (budget split, cutoff, threshold):
For each round:
For each time t:
Perform query and measure error.
Find mean, median, and maximum error across times, number of hard queries and if and when SVT halted.
Return mean of means, median of means, median of medians, and maximum error, average number of hard queries and average number of queries before SVT halted.

Figure 5.8: High level process of repeated query experiments.

Figure 5.9 compares results across all data streams and query types. As expected, based on how the synthetic data sets are constructed (see Section 5.1.2), we find that the uniform stream tends to have the lowest absolute error, while the sharp-shift stream has the highest. We see that τ-RQ outperforms ARQ in most of these experiments, with some notable exceptions.

Influence of data stream. For the uniform data stream we see that τ-RQ and ARQ are often close, with ARQ performing better than τ-RQ for histogram queries for the two central measures of error. We observed that, for the uniform data stream, the best choice of cutoff for ARQ was frequently $c = 1$. This indicates that the initial uniform answer released by ARQ is closer to the true answer than the noisy answers released by τ-RQ, even for large τ.

For the count query we have that the median of medians does not follow the general trends for the sharp-shift data stream. This is because of how the stream is constructed, with the first half of updates having very low effect and the second half having high effect.

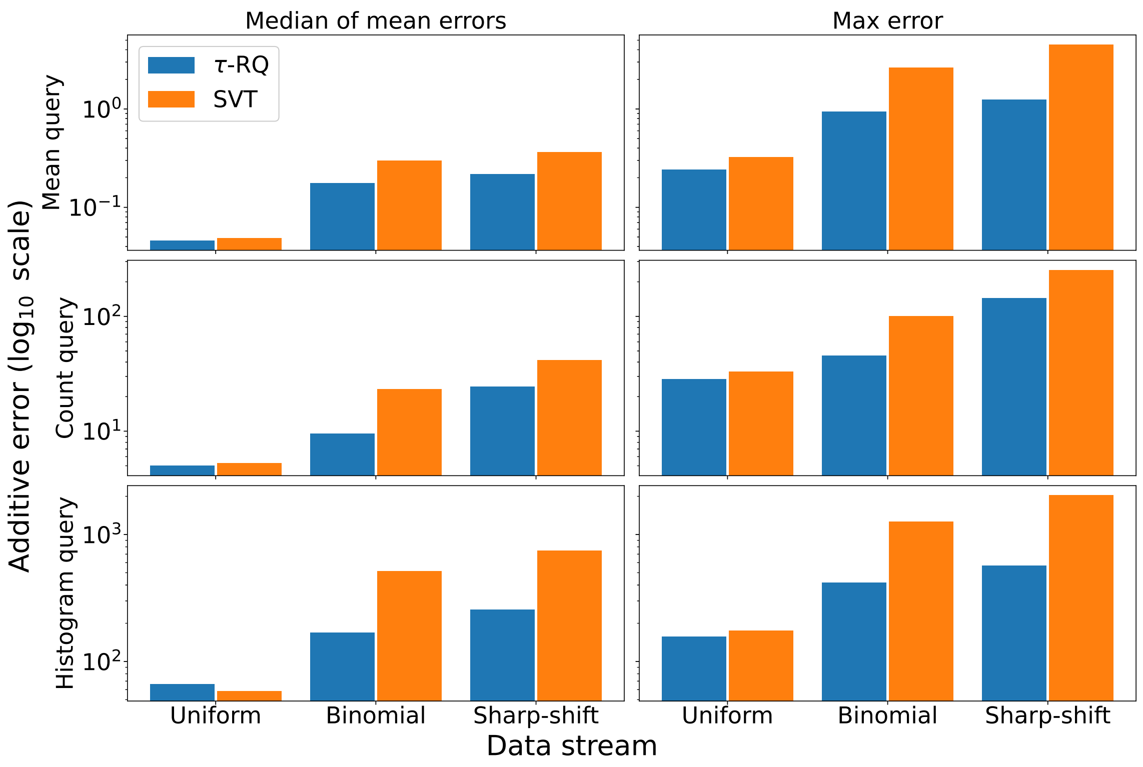

Figure 5.9: Comparison of τ-RQ and ARQ across multiple data/update distributions (uniform, binomial, and sharp-shift) and query types (mean, histogram, and count), with parameters $\varepsilon = 1.0, T = 1000, n = 1000$ fixed. Additive error is taken first over all updates in a trial, then over all trials.

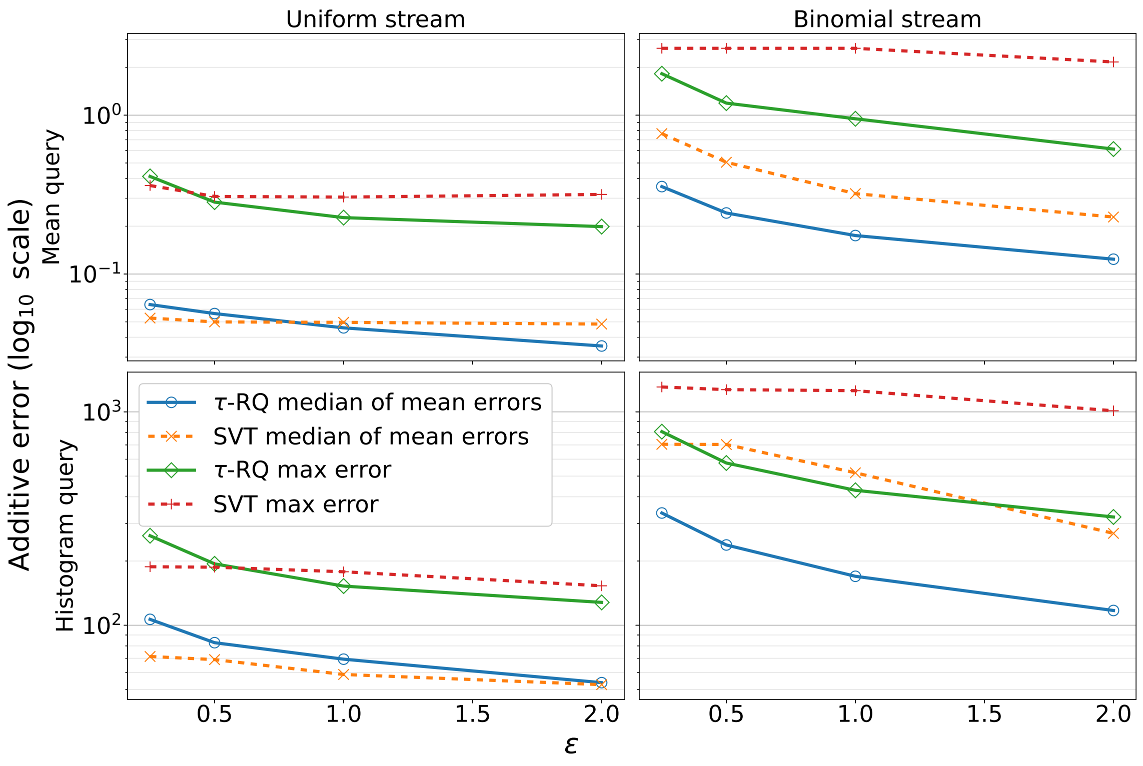

Figure 5.10: Comparison of τ-RQ and ARQ, for mean and histogram queries on uniform and binomial distributions, across a range of privacy parameters $\varepsilon \in \{0.25, 0.5, 1.0, 2.0\}$, and fixed update rounds $T = 1000$ and database size $n = 1000$.

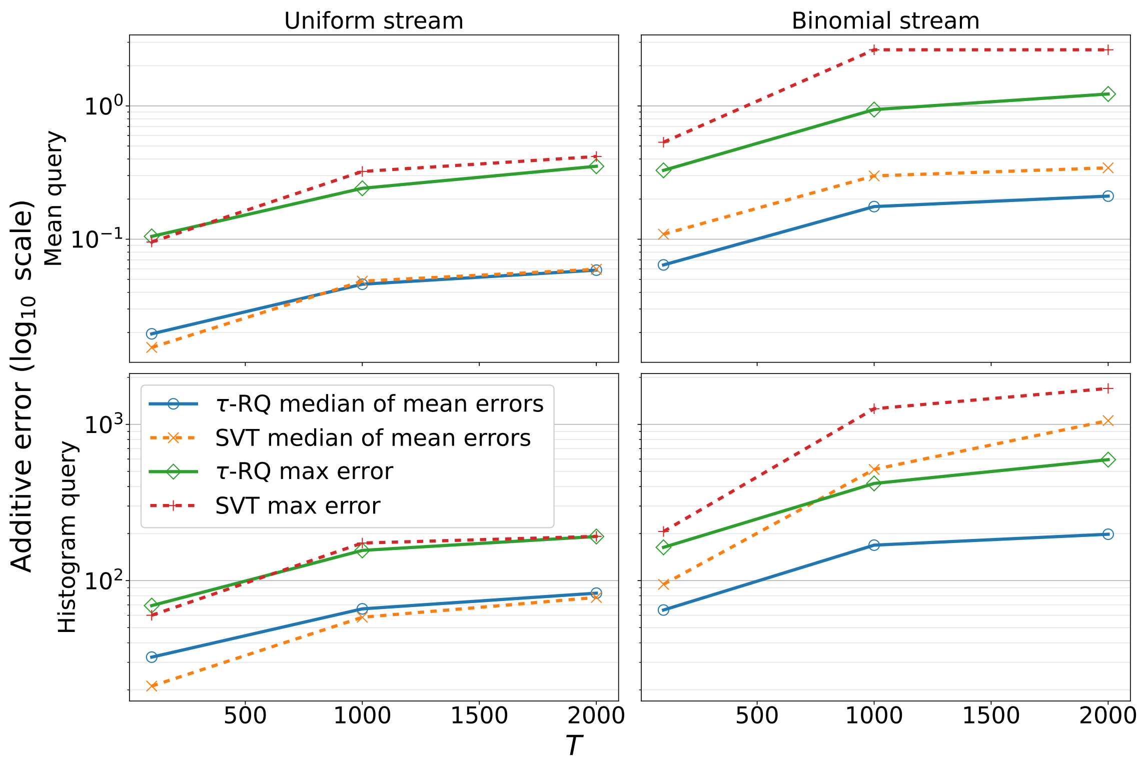

Figure 5.11: Comparison of τ-RQ and ARQ, for mean and histogram queries on uniform and binomial distributions, across a range of update round counts $T \in \{100, 1000, 2000\}$, and fixed database size $n = 1000$ and privacy parameter $\varepsilon = 1.0$.

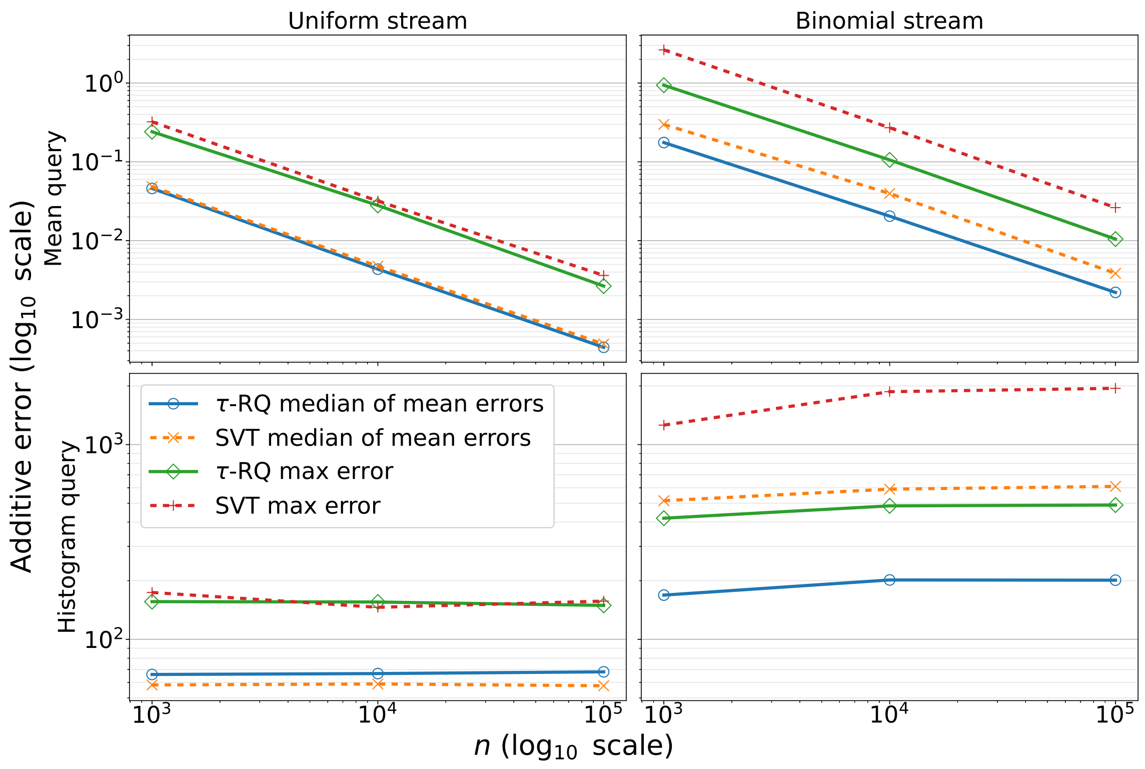

Figure 5.12: Comparison of τ-RQ and ARQ, for mean and histogram queries on uniform and binomial distributions, across a range of database sizes $n \in \{10^3, 10^4, 10^5\}$, and fixed update rounds $T = 1000$ and privacy paramter $\varepsilon = 1.0$.

The median is thus taken from the low effect updates. ARQ performs better in this case because it is choosing a low c and investing all its privacy budget in the first sample.

Influence of privacy parameter. Figure 5.10 shows results of varying ε within {0.25, 0.5, 1.0, 2.0}. As expected, we see that the error decreases as ε increases. On the uniform data stream, we see ARQ generally outperforming τ-RQ for very small ε. This is, again, is because the best choice of cutoff is usually $c = 1$, so ARQ not be releasing noisy answers at all in these cases, since when ε is small, noise is larger.

Influence of time bound. Figure 5.11 shows results of varying T. We choose T in the set $\{100, 1000, 2000\}$ primarily to show variance while being able to complete experiments in a tractable time. As expected, we see that error increases as the number of updates increases.

Influence of database size. Figure 5.12 shows results of varying n. We choose $n = 1000$ as our minimum as it allows us to demonstrate the sharp-shift stream to its full effect for $T = 2000$. We then increase twice, in powers of 10, so that we can see the effect of a large population on query types that have sensitivity dependent on n, and of the relationship between T and n in terms of the maximum effect on the data distribution of updates.

For the histogram query, sensitivity is not tied to the database size, so we do not see a dramatic change in error as we vary n. For the mean query, sensitivity decreases as n increases, so we see error trending down. The relationship between τ-RQ and ARQ are relatively consistent across choices of n.

5.2.3 Results on real data with synthesised updates

In this section we present the results of the same experiments as in Section 5.2.2, this time including the Adult data stream described in Section 5.1.3.

Figure 5.13 shows results of experiments comparing the error rates of τ-RQ and ARQ on the Adult (Age) dataset and the synthetic uniform and binomial data streams. To match the properties of the real data set, we choose $n = 24720$, $T = 7841$, and $\mathcal{X} = [17..90]$ such

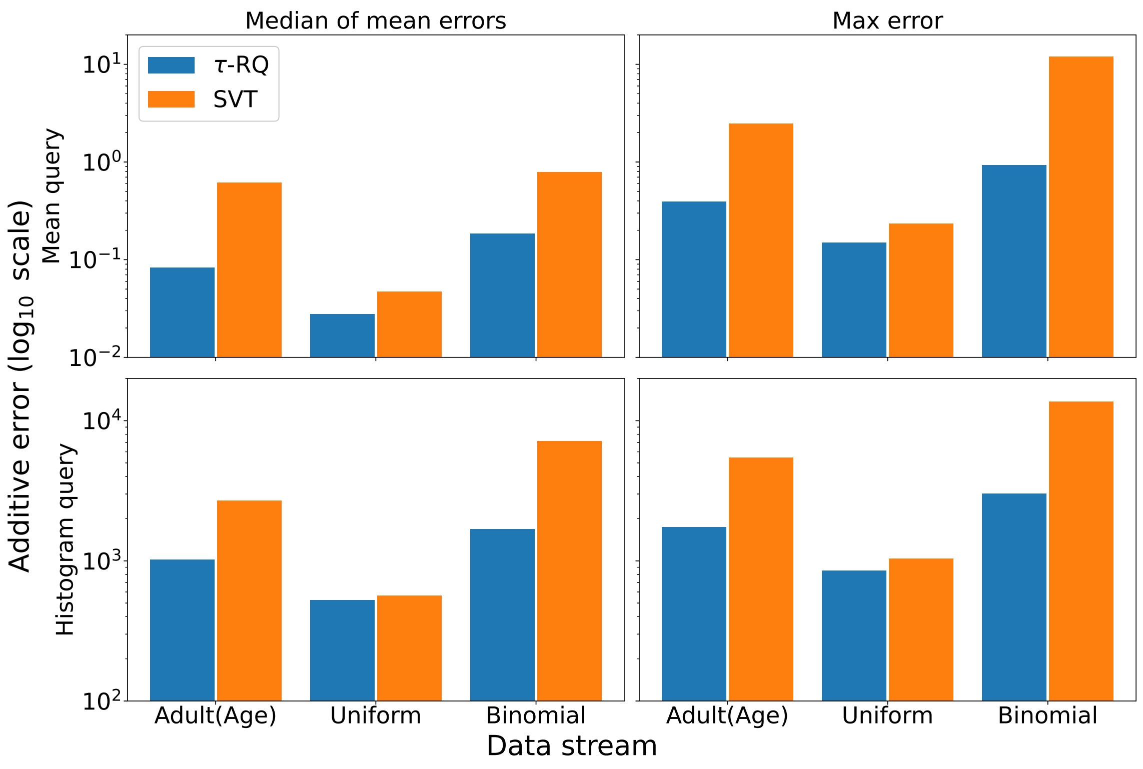

Figure 5.13: Comparison of τ-RQ and ARQ across multiple data/update distributions and query types, with paramaters taken from the Adult (Age) dataset.

that $N = 74$. We do not assess the count query in this case, since it was optimised in the synthetic data experiments to catch the maximum effect of the sharp-shift stream, and a large effect from the binomial stream.

We see that the results follow the trends of the synthetic data streams, with error falling between those for the uniform and binomial data. Error values in general are larger than in previous experiments due to the higher time bound, T and number of histogram buckets, N.

5.3 Interactive query experiments

We now present the results of experiments in the interactive setting, comparing the mechanisms described in Chapter 4. In Section 5.3.1 we show results of experiments with both synthetic data and synthetic query streams. In Section 5.3.2 we show results of experiments with synthetic data streams and a real query stream. A high-level discussion of the results is provided in Section 5.4. In these experiments we again find that our τ-RQ mechanism achieves lower additive error than the comparison mechanism; in this case, PMW-FSDD-T.

5.3.1 Results on synthetic data and query streams

In this section, we present the results of interactive query experiments for the synthetic data streams described in Section 5.1.2, with synthetic query streams described in Section 5.1.4. In this section we compare PMW-FSDD-T to the interactive version of τ-RQ described in Section 4.1, τ-RH. We instantiate τ-RH with τ chosen according to Equation (3.3) with α chosen as given in Theorem 4.1.2. We see that τ-RH has smaller additive error than PMW-FSDD-T across all experiments.

Influence of data and query streams. Figure 5.14 shows the median, across 100 trials, of the mean, across all queries, of additive error for various query streams; for all synthetic data streams, with default parameters for privacy, time bound, and number of individuals. Figure 5.15 similarly shows the maximum additive error. We see that τ-RH has smaller median of mean additive error across all experiments. The choice of data stream has a smaller effect on error than in the repeated query experiments, but follows a similar pattern, where the uniform data stream has the smallest error and the sharp-shift stream the largest. The choice of PMW threshold parameter α has varying effects for different query sets and query limits k, and is relatively unaffected by the data stream.

It is interesting to note that PMW-FSDD-T performs better for larger query sets. We expect that this is because there are more similar queries that can be asked for non-optimised data sets, so that they are more likely to closely match the public histogram. Relative to τ-RH this relationship is more dramatic, so further experimentation with larger query sets may reveal an improvement in the error of PMW-FSDD-T over τ-RH.

Influence of varying parameters. Figure 5.16 shows the effect of varying the data stream parameters for privacy, database size, and time bound. Interestingly, we see that varying the privacy parameter ε has very little effect on the error of PMW-FSDD-T. This requires further investigation. The choice of the threshold parameter α in PMW-FSDD-T has varying effect when the database size or time bound are changed. Further investigation

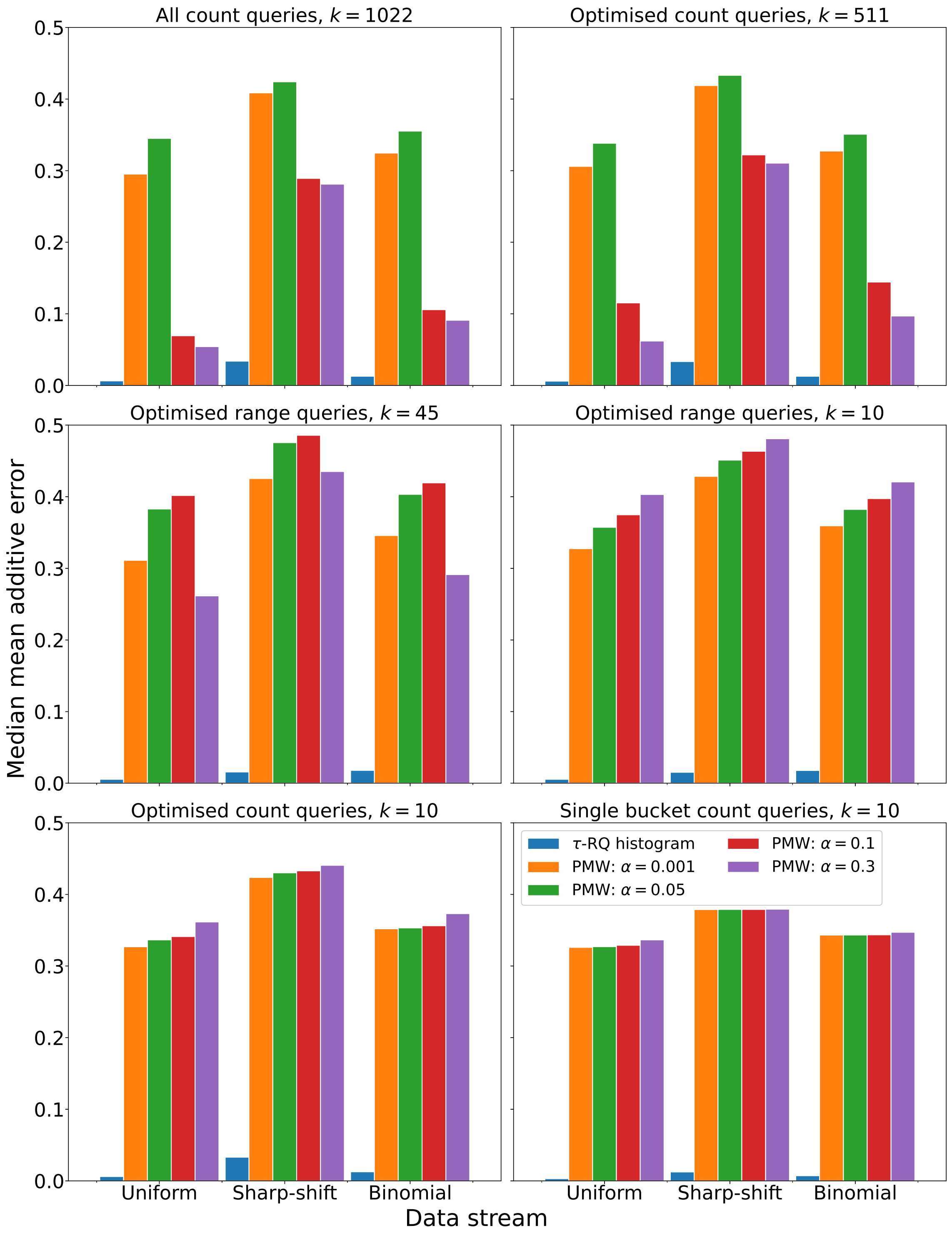

Figure 5.14: Comparison of additive error (median over trials of mean over queries in each trial) for τ-RH and PMW-FSDD-T for synthetic data streams and various linear query sets. The queries in each round are chosen uniformly at random from the query set, with the number of queries chosen uniformly in $[0..k]$. All experiments have privacy parameter $\varepsilon = 1.0$, time bound $T = 1000$ and number of individuals $n = 1000$.

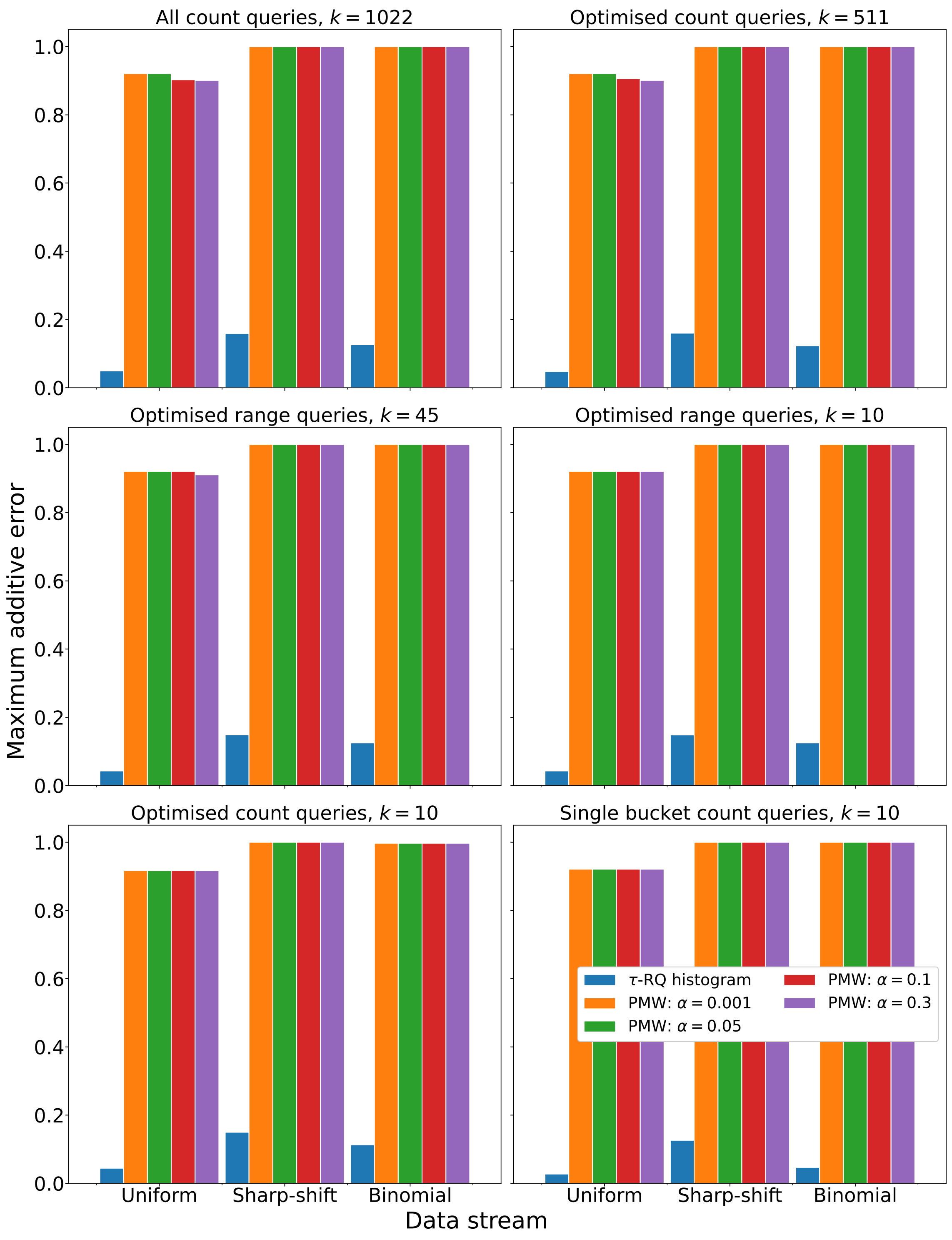

Figure 5.15: Comparison of additive error (maximum over all queries in all trials) for τ-RH and PMW-FSDD-T for synthetic data streams and various linear query sets. The queries in each round are chosen uniformly at random from the query set, with the number of queries chosen uniformly in $[0..k]$. All experiments have privacy parameter $\varepsilon = 1.0$, time bound $T = 1000$ and number of individuals $n = 1000$.

into the choice of the α parameter is needed, including trying more choices, to determine explanations of its effect over all experiments.

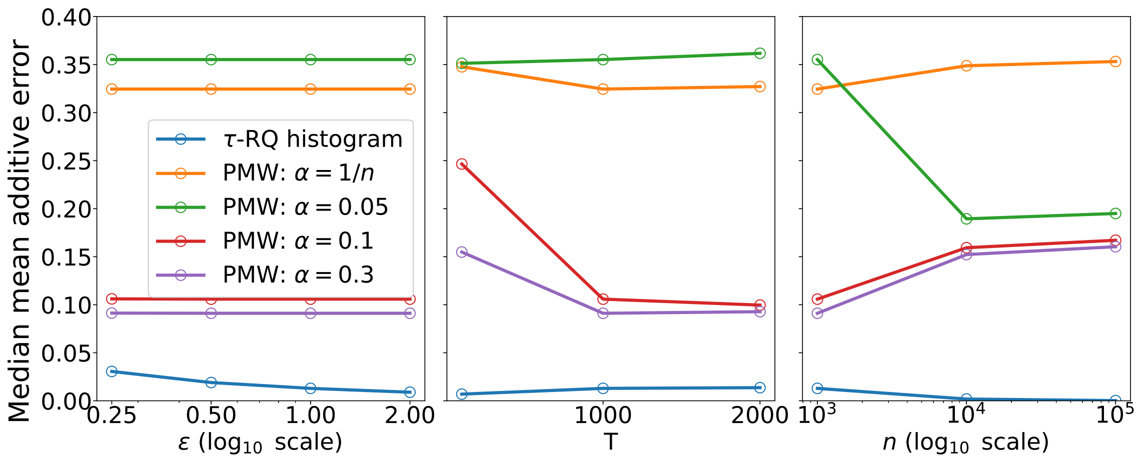

Figure 5.16: Comparison of additive error (median across all trials of mean across all queries in each trial) for τ-RH and PMW-FSDD-T for the binomial data stream with optimised count queries. The queries in each round are chosen uniformly at random from the query set, with the number of queries in each round chosed uniformly in [0..1022]. Default parameter values are $\varepsilon = 1.0$, $T = 1000$ and $n = 1000$.

5.3.2 Results on real query stream

In this section we present the results of repeating the experiments described in Section 5.3.1 using queries taken from the Kosarak query stream described in Section 5.1.4. For these experiments we increase the default parameter for the time bound to $T = 2000$, to allow more of the Kosarak query sequences to be included.

Figure 5.17 shows the median mean error of our query streams with Kosarak queries, with synthetic query streams shown for comparison. We see that the error of PMW-FSDD-T for this real query stream is higher, for most choices of α, than the synthetic streams. The error of τ-RH on the other hand is similar to that for range queries, and significantly less than PMW-FSDD-T across all data streams.

Influence of data stream. In Figure 5.17 we see that τ-RH exhibits the familiar behaviour of error increasing from uniform to binomial to sharp-shift distributions. PMW-FSDD-T is fairly consistent across distributions, but shows a similar pattern.

Influence varying parameters. Figure 5.18 shows the effect of varying the data stream parameters for privacy, database size, and time bound, for the binomial data stream.

When varying epsilon we see a similar pattern to Figure 5.16, with ε having little effect on the error of PMW-FSDD-T, while τ-RH displays the expected behaviour of decreasing error with increasing ε. The effects of varying n and T, are also similar to the synthetic experiments. τ-RH displays decreasing error with increasing dataset size and increasing error with increasing time bound. The effects of these parameters for PMW-FSDD-T vary between different choices of PMW parameter α.

5.4 Discussion

Our τ-RQ mechanisms have shown smaller additive error than the adaptive comparison algorithms, the SVT-based ARQ and PMW-FSDD-T, across most experiments and all

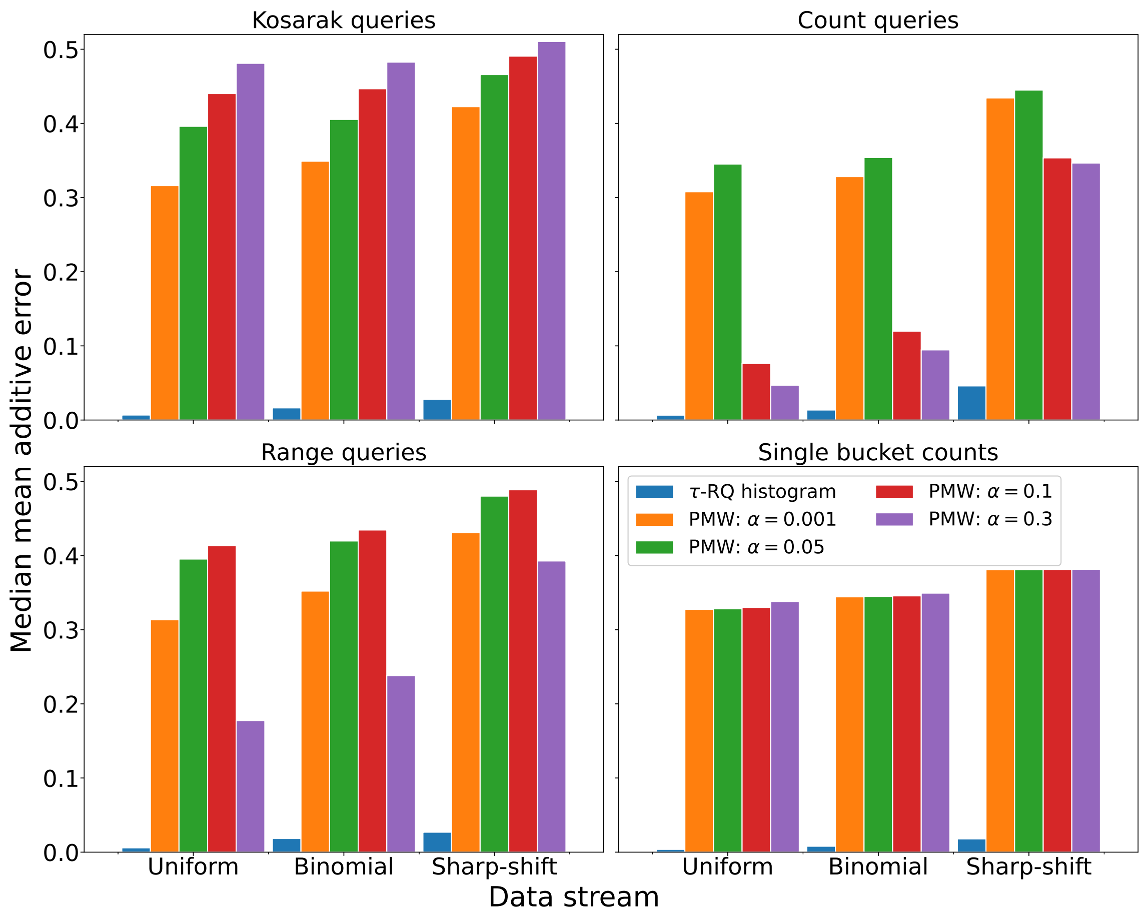

Figure 5.17: Comparison of additive error (median across all trials of mean across all queries in each trial) of the Kosarak query stream to the count query stream (optimised, $k = 511$), range query stream (optimised, $k = 45$), and single bucket count query stream ($k = 10$), for τ-RH and PMW-FSDD-T across multiple data stream types (uniform, binomial, and sharp-shift). All experiments have privacy parameter $\varepsilon = 1.0$, time bound $T = 2000$ and number of individuals $n = 1000$.

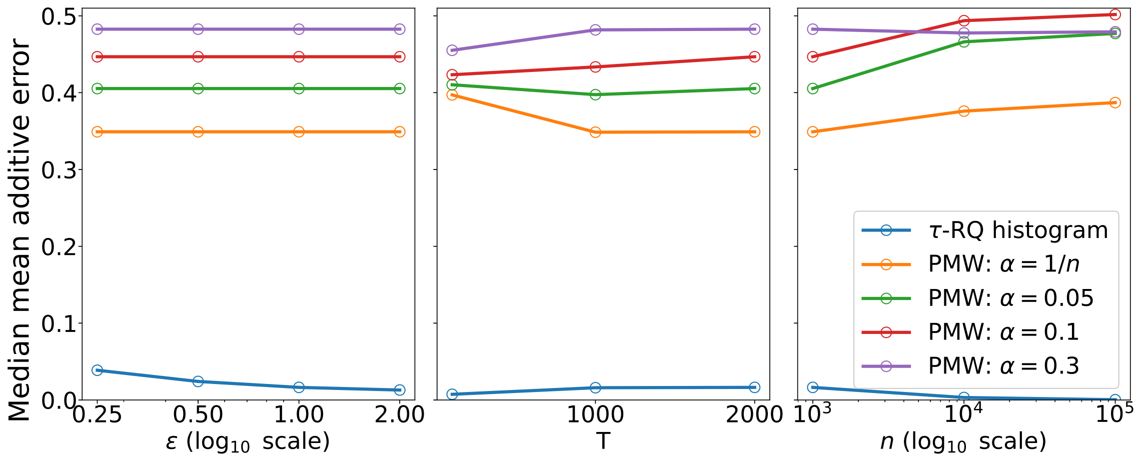

Figure 5.18: Comparison of additive error (median across all trials of mean across all queries in each trial) for τ-RH and PMW-FSDD-T for the binomial data stream with Kosarak queries. Default parameter values are $\varepsilon = 1.0$, $T = 2000$ and $n = 1000$.

measures. This is a strong indication that, under the constraints of the FSDD setting, such adaptive mechanisms do not provide the improvement over basic composition that they promise to. The experimental results presented in Sections 5.2.2 and 5.2.3 indicate that τ-RQ is likely a better choice than ARQ in the FSDD-T repeated query setting. Though ARQ had smaller error in some experiments for uniformly distributed data streams, this was not reflected in the results on the real data stream that had only a small shift in the data distribution over time.

In these repeated query experiments, the results shown for τ-RQ represent the best choice of the frequency parameter τ in each experiment, and, for ARQ, the best of several choices for each of the hard query cutoff c, the split of privacy budget between the decision and sample stages, and the hard query threshold θ. In a real-world implementation, these parameters would need to be chosen in advance, possibly without any prior knowledge of the data and update distributions. While we do have a heuristic for choosing an appropriate τ parameter, this is chosen according to the worst case, and so may not give the best results for τ-RQ when the effect of updates over time is relatively small. For ARQ, there is no indication in the literature of how to best choose these parameters, so further investigation is required to complete an appropriate fixed-parameter comparison. An alternative adaptive mechanism, such as the SVT based distance-based sampling with adaptive threshold, Algorithm 2 (DSAT) of Li et al. [52], could provide a more suitable comparison where parameters must be fixed in advance.

The results presented in Section 5.3 indicate that τ-RH is a significantly better choice than PMW-FSDD for interactive queries in the FSDD-T setting, even when the data and updates are uniformly distributed. In these experiments, frequency parameter τ was fixed according to our heuristic. The design of PMW means there is only one parameter, α, that must be chosen. In each of our experiments we compared four choices of α, with the best performing of these varying based on the query set, but not the data stream. The choice of privacy parameter ε also had little effect on the accuracy of the PMW mechanism, an interesting result that warrants further investigation. The relative improvement in the accuracy of PMW for some choices of α, when the number of allowed queries are large, may indicate that PMW would demonstrate some advantage for much larger numbers of queries.

Overall, the relative simplicity of implementing the τ-RQ mechanisms and the clarity of the heuristic for choosing the τ parameter, combined with these experimental results, provide good evidence that our mechanisms are more practical and reliable than their adaptive counterparts.

Chapter 6

Conclusions and Future Directions

In this chapter, we conclude by summarising the contributions of the thesis and outlining possible directions of future work in DP for FSDD. We finish with a brief conclusion of our findings.

6.1 Summary of contributions

In Section 2.2 we present a taxonomy of dynamic DP. We separate previous work into three broad categories. The *growing database* model encompasses the earliest work in dynamic DP, and is still an active area of research. Under this model, a stream of updates arrive one-by-one, and records are fixed once added. The *time-series* model is most often used in applied research, particularly in the fields of trajectories and internet-of-things; though important theoretical work also falls under this model. Under this model, there is a stream of updates for each user, with historical records maintained in the database. Finally, there are the *fully dynamic* database models, where records can be changed after they are added to the database. This model does not receive much attention in the literature, though there are many applications where only the current state of the data is relevant, or where storage of all previous updates is unfeasible. This is where we situate our own FSDD model. As well as defining these categories and situating key works within them, we also consider more fine-grained distinctions between models, based on whether the number of updates are fixed in advance, and the privacy model.

In Chapter 3 we define our new data model, the fixed-size dynamic database (FSDD). Our model is similar to that in previous work by Mir et al. [56] in that the set of users is fixed, and a single user is updated per unit of time. Our model allows the data universe to be any finite discrete set of states, a broader scope than the finite set of natural numbers covered by their state update model. We define a privacy model that is less restrictive, offering differential privacy rather than pan-privacy; inspired by the user-level privacy definitions often employed for time-series data, it does not depend on the specific values of the updates.

We provide three mechanisms for the release of DP answers to repeated queries in the FSDD setting. Our first algorithm, τ-RQ (Algorithm 6), utilises the Laplace mechanism to release answers to numerical queries, with the frequency of query release optimally determined according to the accuracy of the Laplace mechanism and the query sensitivity. A repeated black-box transformation mechanism, τ-RBB (Algorithm 7), allows for alternative noise generation and non-numerical queries by converting any existing DP mechanism for the static setting to the FSDD setting. Finally, for comparison, we provide a sparse sector technique (SVT) based mechanism, ARQ, for the FSDD setting (Algorithm 9).

In Chapter 4 we consider interactive queries, where a set of possible queries, but not the number or order of queries between each update, is known in advance. We utilise τ-RBB, in our τ-RH (Algorithm 10), mechanism to provide public histograms for answering linear queries in this setting, and provide a private multiplicative weights mechanism, PMW-FSDD, for comparison (Algorithm 12).

In Chapter 5 we experimentally evaluate the mechanisms given in Chapters 3 and 4. We find that our τ-RQ mechanisms provide better accuracy than ARQ and PMW-FSDD in most experiments. τ-RQ demonstrates significantly better accuracy than ARQ when the data distribution is shifting over time, and similar accuracy for a uniformly distributed dataset with uniformly distributed updates. τ-RH demonstrates significantly better accuracy than PMW-FSDD across all data and update streams.

6.2 Future directions

This thesis introduces the FSDD model and provides ε-differentially private mechanisms for query release, primarily in the T-bounded update setting. This work can be extended by relaxing the privacy definition, allowing unbounded streams, or allowing multiple individuals to be updated per unit of time. Additionally, lower bounds on accuracy under the FSDD model can be investigated, and compared to existing lower bounds on other models. Since each FSDD update results in a neighbouring database, a possible line of investigation, into additional mechanisms and theoretical guarantees, is modelling FSDD streams using graphs; with vertices representing database histograms and edges representing neighbour relationships.

Additional comparison mechanisms. Due to the time constraints of a master's, thesis, our experimental comparisons are limited to ARQ and PMW-FSDD. Some additional candidate comparison mechanisms, that are not implemented, include the DSAT mechanism introduced by Li et al. [52] for repeated queries and histogram publication, the binary tree mechanism [9, 26, 65] for adaptive interval queries, and the Net/SmallDB mechanism [1, 15] for histogram publication.

Adaptive query frequency. Given that the initial choice of τ in τ-RQ is based on worst case analysis, it could instead be optimised based on suitable assumptions about the distribution of the updates. For example, τ could be increased if there is a reasonable expectation that the updates will have a similar distribution to the initial database. If suitable information is not available to inform such assumptions, inspired by the adaptive threshold of the DSAT mechanism [52], the sample frequency could be chosen adaptively based on the previous samples. For example, decreasing the sample frequency if the samples differ very little.

Approximate differential privacy. The privacy guaranteed by all mechanisms in this thesis is ε-differential privacy, or *pure* DP. As mentioned in Definition 2.1.1, there is a relaxed definition of (ε, δ)-differential privacy, or *approximate* DP, that allows a small probability of privacy failure. Under approximate DP, by the theorems of advanced and optimal composition [28, 47], composition is more efficient. Modifying our mechanisms for approximate DP would improve their accuracy, at a small cost to the robustness of the privacy guarantee.

Relaxed database model. Our database model allows only one individual to be included in each update. To make our results more applicable to real-world settings, such as under a constrained time-series model, we could consider a bound, k, on the number of individuals included in each update. Extending τ-RQ to this setting would require only re-optimising for the increased update effect. One possible line of exploration would be to compare the accuracy of such mechanisms to mechanisms for time-series data where all individuals can be updated per unit of time.

Infinite updates. Most of our mechanisms are designed under the assumption of a bound T, on the number of updates. There is much work in the literature regarding infinite streams of data (e.g., [9, 15, 59]). Our mechanisms could be readily adapted to the infinite update setting using sliding-window privacy Kellaris et al. [49]. We expect that bounded user-level ε-DP, with utility not tending towards uniformly random outputs, is not possible in the infinite update setting under FSDD due to the fixed database size, though we have not proven this.

Lower bounds. In the literature, lower bounds for the accuracy of the growing database model are well studied (e.g., [3, 26, 35, 43]). The growing database can be modelled as a special case of FSDD where there is a single user, or where each user updates only once after starting in a neutral state. As such, we cannot improve on these bounds, though it would be interesting to investigate whether matching bounds can be found for FSDD.

Graph representation. The FSDD model has the useful property that databases in the stream, S, separated by a single update, are neighbouring. That is, $D_t \approx_{\circlearrowleft} D_{t+1}$. Additionally, since data points take values from a discrete, finite universe, databases can be represented as a histogram of counts. Thus, since the number of entries is fixed, the set of possible databases in a stream can be represented as a finite graph with vertices representing histograms, and edges between neighbouring histograms. An FSDD stream can be represented as a walk on such a graph, with the cumulative effect of updates being the distance between the start vertex and current vertex.

Recent work by Boedihardjo et al. [2] introduced a *superregular random walk* for generating differentially private synthetic data. We expect that, given the graph properties of the FSDD model, such a walk could be applied. The properties of these graphs, and the walk representation of database streams, is also a promising direction for reasoning about bounds on accuracy of DP mechanisms under FSDD. A similar construction was used by Kifer and Machanavajjhala [50] to provide differential privacy for queries,n- in the case where some non-private statistics about the data have already been released, by using the shortest path in such a graph as a utility score in the exponential mechanism [55].

6.3 Conclusions

Existing scholarship into differential privacy for dynamic data focuses on data streams, where data are added over time, and time-series, where a stream of data exists for each individual. There is limited research into DP for data that is modified over time, with our literature review finding only two papers, by Mir et al. [56] and Qiu and Yi [59], considering such models. We introduce a new database and privacy model, the fixed-size dynamic database, that is similar to the database model of Mir et al. [56], with a single value modified per update, but with a broader scope and less restrictive privacy model.

By limiting updates to a single user per update, unlike the time-series models, the effect of each update on the overall distribution of the database is small. We provide mechanisms for releasing both repeated and interactive queries that sample the database at fixed intervals, and give heuristics for optimising this frequency based on the worst-case update effect. Our empirical analyses of τ-RQ, τ-RBB, and τ-RH show that these mechanisms are successful in harnessing the constraints of the FSDD model to improve accuracy over existing adaptive mechanisms, since the fixed sample intervals allow the full privacy budget to be allocated to sampling, rather than shared with a decision process.

Bibliography

[1] Avrim Blum, Katrina Ligett, and Aaron Roth. A learning theory approach to non-interactive database privacy. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing*, STOC'08, pages 609–618, New York, NY, USA, May 2008. ACM. ISBN 978-1-60558-047-0. doi: 10.1145/1374376.1374464. URL `https://doi.org/10.1145/1374376.1374464`.

[2] March Boedihardjo, Thomas Strohmer, and Roman Vershynin. Private measures, random walks, and synthetic data. *CoRR*, abs/2204.09167(arXiv:2204.09167), April 2022. doi: 10.48550/arXiv.2204.09167. URL `http://arxiv.org/abs/2204.09167`.

[3] Jean Bolot, Nadia Fawaz, S. Muthukrishnan, Aleksandar Nikolov, and Nina Taft. Private decayed predicate sums on streams. In *Proceedings of the 16th International Conference on Database Theory*, ICDT '13, pages 284–295, New York, NY, USA, March 2013. ACM. ISBN 978-1-4503-1598-2. doi: 10.1145/2448496.2448530. URL `https://doi.org/10.1145/2448496.2448530`.

[4] Joseph A. Calandrino, Ann Kilzer, Arvind Narayanan, Edward W. Felten, and Vitaly Shmatikov. "you might also like:" privacy risks of collaborative filtering. In *32nd IEEE Symposium on Security and Privacy*, SP'11, pages 231–246. IEEE, May 2011. doi: 10.1109/SP.2011.40. URL `https://doi.org/10.1109/SP.2011.40`.

[5] Clément L. Canonne, Gautam Kamath, and Thomas Steinke. The discrete gaussian for differential privacy. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020*, volume 33 of *NeurIPS'20*. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper/2020/hash/b53b3a3d6ab90ce0268229151c9bde11-Abstract.html`.

[6] Yang Cao and Masatoshi Yoshikawa. Differentially Private Real-Time Data Release over Infinite Trajectory Streams. In *16th IEEE International Conference on Mobile*

Data Management, volume 2 of *MDM'15*, pages 68–73. IEEE, June 2015. doi: 10.1109/MDM.2015.15. URL `https://doi.org/10.1109/MDM.2015.15`.

[7] Yang Cao, Masatoshi Yoshikawa, Yonghui Xiao, and Li Xiong. Quantifying Differential Privacy under Temporal Correlations. In *33rd IEEE International Conference on Data Engineering*, ICDE'17, pages 821–832. IEEE, April 2017. doi: 10.1109/ICDE.2017.132. URL `https://doi.org/10.1109/ICDE.2017.132`.

[8] Adrian Rivera Cardoso and Ryan Rogers. Differentially private histograms under continual observation: Streaming selection into the unknown. In *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, volume 151 of *Proceedings of Machine Learning Research*, pages 2397–2419. PMLR, May 2022. URL `https://proceedings.mlr.press/v151/rivera-cardoso22a.html`.

[9] T.-H. Hubert Chan, Elaine Shi, and Dawn Song. Private and Continual Release of Statistics. *ACM Transactions on Information and System Security*, 14(3):1–24, November 2011. ISSN 1094-9224, 1557-7406. doi: 10.1145/2043621.2043626. URL `https://dl.acm.org/doi/10.1145/2043621.2043626`.

[10] T.-H. Hubert Chan, Mingfei Li, Elaine Shi, and Wenchang Xu. Differentially private continual monitoring of heavy hitters from distributed streams. In *Privacy Enhancing Technologies - 12th International Symposium*, volume 7384 of *Lecture Notes in Computer Science*, pages 140–159, Berlin, Heidelberg, 2012. Springer. ISBN 978-3-642-31680-7. doi: 10.1007/978-3-642-31680-7_8. URL `https://doi.org/10.1007/978-3-642-31680-7_8`.

[11] Yan Chen and Ashwin Machanavajjhala. On the privacy properties of variants on the sparse vector technique. *CoRR*, abs/1508.07306, August 2015. URL `http://arxiv.org/abs/1508.07306`.

[12] Yan Chen, Ashwin Machanavajjhala, Michael Hay, and Gerome Miklau. Pegasus: Data-adaptive differentially private stream processing. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS'17, pages 1375–1388. ACM, 2017. doi: 10.1145/3133956.3134102. URL `https://doi.org/10.1145/3133956.3134102`.

[13] Aloni Cohen and Kobbi Nissim. Linear program reconstruction in practice. *Journal of Privacy and Confidentiality*, 10(1), January 2020. ISSN 2575-8527. doi: 10.29012/

JPC.711. URL https://journalprivacyconfidentiality.org/index.php/jpc/article/view/711.

[14] Robert M Corless, Gaston H Gonnet, David EG Hare, David J Jeffrey, and Donald E Knuth. On the lambert w function. *Advances in Computational Mathematics*, 5:329–359, 1996. doi: 10.1007/BF02124750. URL https://doi.org/10.1007/BF02124750.

[15] Rachel Cummings, Sara Krehbiel, Kevin A. Lai, and Uthaipon Tao Tantipongpipat. Differential privacy for growing databases. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems*, volume 31 of *NeurIPS'18*, pages 8878–8887. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/hash/ac27b77292582bc293a51055bfc994ee-Abstract.html.

[16] Rachel Cummings, Sara Krehbiel, Kevin A. Lai, and Uthaipon Tao Tantipongpipat. Differential privacy for growing databases. *CoRR*, abs/1803.06416, March 2018. URL http://arxiv.org/abs/1803.06416.

[17] Tore Dalenius. Towards a methodology for statistical disclosure control. *Statistik Tidskrift*, 15:429–444, 1977. URL https://hdl.handle.net/1813/111303.

[18] Irit Dinur and Kobbi Nissim. Revealing information while preserving privacy. In *Proceedings of the Twenty-Second ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, PODS'03, pages 202–210, New York, NY, USA, June 2003. ACM. ISBN 978-1-58113-670-8. doi: 10.1145/773153.773173. URL https://doi.org/10.1145/773153.773173.

[19] Wei Dong, Qiyao Luo, and Ke Yi. Continual Observation under User-level Differential Privacy. In *44th IEEE Symposium on Security and Privacy*, SP'23, pages 2190–2207. IEEE, May 2023. ISBN 978-1-66549-336-9. doi: 10.1109/SP46215.2023.10179466. URL https://www.computer.org/csdl/proceedings-article/sp/2023/933600c190/1NrbZwal6hi.

[20] Cynthia Dwork. Differential Privacy. In *Automata, Languages and Programming, 33rd International Colloquium*, volume 4052 of *Lecture Notes in Computer Science*, pages 1–12, Berlin, Heidelberg, 2006. Springer. ISBN 978-3-540-35908-1. doi: 10.1007/11787006_1. URL https://doi.org/10.1007/11787006_1.

[21] Cynthia Dwork and Kobbi Nissim. Privacy-Preserving Datamining on Vertically Partitioned Databases. In *Advances in Cryptology - CRYPTO 2004, 24th Annual International CryptologyConference*, volume 3152 of *Lecture Notes in Computer Science*, pages 528–544, Berlin, Heidelberg, 2004. Springer. ISBN 978-3-540-28628-8. doi: 10.1007/978-3-540-28628-8_32. URL `https://doi.org/10.1007/978-3-540-28628-8_32`.

[22] Cynthia Dwork and Aaron Roth. The Algorithmic Foundations of Differential Privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3-4):211–407, 2014. ISSN 1551-305X, 1551-3068. doi: 10.1561/0400000042. URL `http://www.nowpublishers.com/articles/foundations-and-trends-in-theoretical-computer-science/TCS-042`.

[23] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our Data, Ourselves: Privacy Via Distributed Noise Generation. In *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, volume 4004 of *Lecture Notes in Computer Science*, pages 486–503, Berlin, Heidelberg, 2006. Springer. ISBN 978-3-540-34546-6 978-3-540-34547-3. doi: 10.1007/11761679_29. URL `http://link.springer.com/10.1007/11761679_29`.

[24] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam D. Smith. Calibrating Noise to Sensitivity in Private Data Analysis. In *Theory of Cryptography, Third Theory of Cryptography Conference*, volume 3876 of *Lecture Notes in Computer Science*, pages 265–284, Berlin, Heidelberg, 2006. Springer. ISBN 978-3-540-32731-8 978-3-540-32732-5. doi: 10.1007/11681878_14. URL `http://link.springer.com/10.1007/11681878_14`.

[25] Cynthia Dwork, Moni Naor, Omer Reingold, Guy N. Rothblum, and Salil P. Vadhan. On the complexity of differentially private data release: Efficient algorithms and hardness results. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing*, STOC'09, pages 381–390, New York, NY, USA, May 2009. ACM. ISBN 978-1-60558-506-2. doi: 10.1145/1536414.1536467. URL `https://doi.org/10.1145/1536414.1536467`.

[26] Cynthia Dwork, Moni Naor, Toniann Pitassi, and Guy N. Rothblum. Differential privacy under continual observation. In *Proceedings of the 42nd ACM Symposium on*

Theory of Computing, STOC'10, pages 715–724, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0050-6. doi: 10.1145/1806689.1806787. URL `https://doi.org/10.1145/1806689.1806787`.

[27] Cynthia Dwork, Moni Naor, Toniann Pitassi, Sergey Yekhanin, and Guy N. Rothblum. Pan-private streaming algorithms. In *Innovations in Computer Science - ICS 2010*, pages 66–80, Tsinghua University, Beijing, China, January 2010. Tsinghua University Press. URL `http://conference.iiis.tsinghua.edu.cn/ICS2010/content/papers/6.html`.

[28] Cynthia Dwork, Guy N. Rothblum, and Salil P. Vadhan. Boosting and Differential Privacy. In *51st Annual IEEE Symposium on Foundations of Computer Science*, pages 51–60. IEEE, October 2010. doi: 10.1109/FOCS.2010.12. URL `https://doi.org/10.1109/FOCS.2010.12`.

[29] Cynthia Dwork, Moni Naor, Omer Reingold, and Guy N. Rothblum. Pure Differential Privacy for Rectangle Queries via Private Partitions. In *Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security*, volume 9453 of *Lecture Notes in Computer Science*, pages 735–751, Berlin, Heidelberg, 2015. Springer. ISBN 978-3-662-48800-3. doi: 10.1007/978-3-662-48800-3_30. URL `https://doi.org/10.1007/978-3-662-48800-3_30`.

[30] Úlfar Erlingsson, Vitaly Feldman, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Abhradeep Thakurta. Amplification by Shuffling: From Local to Central Differential Privacy via Anonymity. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA'19, pages 2468–2479. SIAM, January 2019. doi: 10.1137/1.9781611975482.151. URL `https://epubs.siam.org/doi/abs/10.1137/1.9781611975482.151`.

[31] Liyue Fan and Li Xiong. Real-time aggregate monitoring with differential privacy. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM'12, pages 2169–2173, New York, NY, USA, October 2012. ACM. ISBN 978-1-4503-1156-4. doi: 10.1145/2396761.2398595. URL `https://doi.org/10.1145/2396761.2398595`.

[32] Liyue Fan and Li Xiong. An Adaptive Approach to Real-Time Aggregate Monitoring With Differential Privacy. *IEEE Transactions on Knowledge and Data Engineering*, 26(9):2094–2106, September 2014. ISSN 1558-2191. doi: 10.1109/TKDE.2013.96. URL `https://doi.org/10.1109/TKDE.2013.96`.

[33] Liyue Fan, Li Xiong, and Vaidy S. Sunderam. FAST: Differentially private real-time aggregate monitor with filtering and adaptive sampling. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, SIGMOD'13, pages 1065–1068, New York, NY, USA, June 2013. ACM. ISBN 978-1-4503-2037-5. doi: 10.1145/2463676.2465253. URL `https://doi.org/10.1145/2463676.2465253`.

[34] Liyue Fan, Luca Bonomi, Li Xiong, and Vaidy S. Sunderam. Monitoring web browsing behavior with differential privacy. In *Proceedings of the 23rd International World Wide Web Conference*, WWW'14, pages 177–188, New York, NY, USA, April 2014. ACM. ISBN 978-1-4503-2744-2. doi: 10.1145/2566486.2568038. URL `https://doi.org/10.1145/2566486.2568038`.

[35] Hendrik Fichtenberger, Monika Henzinger, and Jalaj Upadhyay. Constant matters: Fine-grained Complexity of Differentially Private Continual Observation. *CoRR*, abs/2202.11205, April 2022. URL `http://arxiv.org/abs/2202.11205`.

[36] Matthew Fredrikson, Eric Lantz, Somesh Jha, Simon M. Lin, David Page, and Thomas Ristenpart. Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing. In *Proceedings of the 23rd USENIX Security Symposium*, USENIX Security'14, pages 17–32. USENIX Association, 2014. ISBN 978-1-931971-15-7. URL `https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/fredrikson_matthew`.

[37] Benjamin C. M. Fung, Ke Wang, Rui Chen, and Philip S. Yu. Privacy-preserving data publishing: A survey of recent developments. *ACM Computing Surveys*, 42 (4):14:1–14:53, June 2010. ISSN 0360-0300. doi: 10.1145/1749603.1749605. URL `https://doi.org/10.1145/1749603.1749605`.

[38] Simson L. Garfinkel, John M. Abowd, and Christian Martindale. Understanding database reconstruction attacks on public data. *Communications of the ACM*, 62 (3):46–53, February 2019. ISSN 0001-0782, 1557-7317. doi: 10.1145/3287287. URL `https://dl.acm.org/doi/10.1145/3287287`.

[39] Arpita Ghosh, Tim Roughgarden, and Mukund Sundararajan. Universally Utility-maximizing Privacy Mechanisms. *SIAM Journal on Computing*, 41(6):1673–1693, 2012. ISSN 00975397. doi: 10.1137/09076828X. URL https://www.proquest.com/docview/1265813038/abstract/DD76335C2ECC4911PQ/1.

[40] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, January 1984. ISSN 10902724. doi: 10.1016/0022-0000(84)90070-9.

[41] Moritz Hardt. *A Study of Privacy and Fairness in Sensitive Data Analysis.* PhD thesis, Princeton University, Princeton, NJ, 2011. URL http://arks.princeton.edu/ark:/88435/dsp01vq27zn422.

[42] Moritz Hardt and Guy N. Rothblum. A Multiplicative Weights Mechanism for Privacy-Preserving Data Analysis. In *51st Annual IEEE Symposium on Foundations of Computer Science*, FOCS'10, pages 61–70, October 2010. doi: 10.1109/FOCS.2010.85. URL https://doi.org/10.1109/FOCS.2010.85.

[43] Monika Henzinger, Jalaj Upadhyay, and Sarvagya Upadhyay. Almost tight error bounds on differentially private continual counting. In *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms*, SODA'23, pages 5003–5039. SIAM, 2023. doi: 10.1137/1.9781611977554.CH183. URL https://doi.org/10.1137/1.9781611977554.ch183.

[44] IDC and Statista. Volume of data/information created, captured, copied, and consumed worldwide from 2010 to 2020, with forecasts from 2021 to 2025 (in zettabytes), June 2021. URL https://www-statista-com.eu1.proxy.openathens.net/statistics/871513/worldwide-data-created/.

[45] Prateek Jain, Pravesh Kothari, and Abhradeep Thakurta. Differentially Private Online Learning. In *Proceedings of the 25th Annual Conference on Learning Theory*, volume 23 of *JMLR Workshop and Conference Proceedings*, pages 24.1–24.34. JMLR, June 2012. URL https://proceedings.mlr.press/v23/jain12.html.

[46] Jiankai Jin, Eleanor McMurtry, Benjamin I. P. Rubinstein, and Olga Ohrimenko. Are we there yet? timing and floating-point attacks on differential privacy systems. In *43rd IEEE Symposium on Security and Privacy*, SP'22, pages 473–488,

May 2022. doi: 10.1109/SP46214.2022.9833672. URL https://doi.org/10.1109/SP46214.2022.9833672.

[47] Peter Kairouz, Sewoong Oh, and Pramod Viswanath. The Composition Theorem for Differential Privacy. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 1376–1385. JMLR, June 2015. URL https://proceedings.mlr.press/v37/kairouz15.html.

[48] Shiva Prasad Kasiviswanathan, Homin K. Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam D. Smith. What Can We Learn Privately? In *Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science*, FOCS'08, pages 531–540, USA, October 2008. IEEE. ISBN 978-0-7695-3436-7. doi: 10.1109/FOCS.2008.27. URL https://doi.org/10.1109/FOCS.2008.27.

[49] Georgios Kellaris, Stavros Papadopoulos, Xiaokui Xiao, and Dimitris Papadias. Differentially private event sequences over infinite streams. *Proceedings of the VLDB Endowment*, 7(12):1155–1166, August 2014. ISSN 2150-8097. doi: 10.14778/2732977.2732989. URL https://doi.org/10.14778/2732977.2732989.

[50] Daniel Kifer and Ashwin Machanavajjhala. No free lunch in data privacy. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data*, SIGMOD '11, pages 193–204, New York, NY, USA, June 2011. ACM. ISBN 978-1-4503-0661-4. doi: 10.1145/1989323.1989345. URL https://doi.org/10.1145/1989323.1989345.

[51] S. Kullback and R. A. Leibler. On Information and Sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, March 1951. ISSN 0003-4851, 2168-8990. doi: 10.1214/aoms/1177729694. URL https://projecteuclid.org/journals/annals-of-mathematical-statistics/volume-22/issue-1/On-Information-and-Sufficiency/10.1214/aoms/1177729694.full.

[52] Haoran Li, Li Xiong, Xiaoqian Jiang, and Jinfei Liu. Differentially Private Histogram Publication for Dynamic Datasets: An Adaptive Sampling Approach. In *Proceedings of the 24th ACM International on Conference on Information and*

Knowledge Management, CIKM'15, pages 1001–1010, New York, NY, USA, October 2015. ACM. ISBN 978-1-4503-3794-6. doi: 10.1145/2806416.2806441. URL `https://doi.org/10.1145/2806416.2806441`.

[53] Min Lyu, Dong Su, and Ninghui Li. Understanding the sparse vector technique for differential privacy. *Proceedings of the VLDB Endowment*, 10(6):637–648, February 2017. ISSN 2150-8097. doi: 10.14778/3055330.3055331. URL `https://doi.org/10.14778/3055330.3055331`.

[54] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. ℓ-diversity: Privacy beyond k-anonymity. *ACM Transactions on Knowledge Discovery from Data*, 1(1):3, 2007. ISSN 1556-4681. doi: 10.1145/1217299.1217302. URL `https://dl.acm.org/doi/10.1145/1217299.1217302`.

[55] Frank McSherry and Kunal Talwar. Mechanism Design via Differential Privacy. In *48th Annual IEEE Symposium on Foundations of Computer Science*, FOCS'07, pages 94–103. IEEE, October 2007. doi: 10.1109/FOCS.2007.66. URL `https://doi.org/10.1109/FOCS.2007.41`.

[56] Darakhshan J. Mir, S. Muthukrishnan, Aleksandar Nikolov, and Rebecca N. Wright. Pan-private algorithms via statistics on sketches. In *Proceedings of the 30th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS'11, pages 37–48, New York, NY, USA, June 2011. ACM. ISBN 978-1-4503-0660-7. doi: 10.1145/1989284.1989290. URL `https://doi.org/10.1145/1989284.1989290`.

[57] Ilya Mironov. On significance of the least significant bits for differential privacy. In *Proceedings of the 2012 ACM SIGSAC Conference on Computer and Communications Security*, CCS '12, pages 650–661, New York, NY, USA, October 2012. ACM. ISBN 978-1-4503-1651-4. doi: 10.1145/2382196.2382264. URL `https://doi.org/10.1145/2382196.2382264`.

[58] Victor Perrier, Hassan Jameel Asghar, and Dali Kaafar. Private Continual Release of Real-Valued Data Streams. In *Proceedings of the 26th Annual Network and Distributed System Security Symposium*, NDSS'19, San Diego, CA, 2019. The Internet Society. ISBN 978-1-891562-55-6. doi: 10.14722/ndss.2019.

23535. URL https://www.ndss-symposium.org/wp-content/uploads/2019/02/ndss2019_07B-5_Perrier_paper.pdf.

[59] Yuan Qiu and Ke Yi. Differential Privacy on Dynamic Data. *CoRR*, abs/2209.01387, September 2022. doi: 10.48550/arXiv.2209.01387. URL https://arxiv.org/abs/2209.01387v2.

[60] Vibhor Rastogi and Suman Nath. Differentially private aggregation of distributed time-series with transformation and encryption. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, SIGMOD'10, pages 735–746, New York, NY, USA, June 2010. ACM. ISBN 978-1-4503-0032-2. doi: 10.1145/1807167.1807247. URL https://doi.org/10.1145/1807167.1807247.

[61] Aaron Roth. Lecture 11: The sparse vector technique. Lecture notes, CIS 800/002 The Algorithmic Foundations of Data Privacy, October 20, University of Pennsylvania, 2011. URL https://www.cis.upenn.edu/~aaroth/courses/slides/Lecture11.pdf.

[62] Aaron Roth and Tim Roughgarden. Interactive privacy via the median mechanism. In *Proceedings of the 42nd ACM Symposium on Theory of Computing*, STOC'10, pages 765–774, New York, NY, USA, June 2010. ACM. ISBN 978-1-4503-0050-6. doi: 10.1145/1806689.1806794. URL https://doi.org/10.1145/1806689.1806794.

[63] Pierangela Samarati and Latanya Sweeney. Generalizing data to provide anonymity when disclosing information. In *Proceedings of the Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, PODS'98, page 188. ACM, 1998. doi: 10.1145/275487.275508. URL https://www.eng.auburn.edu/~xqin/courses/comp7370/k-anonymity-pods98.pdf.

[64] Adam D. Smith and Abhradeep Thakurta. (Nearly) Optimal Algorithms for Private Online Learning in Full-information and Bandit Settings. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013*, volume 26 of *NeurIPS'13*. Curran Associates, Inc., 2013. URL https://proceedings.neurips.cc/paper/2013/hash/c850371fda6892fbfd1c5a5b457e5777-Abstract.html.

[65] Adam D. Smith and Jonathan Ullman. Lecture 7: The Binary Tree Mechanism. Lecture notes, Privacy in Statistics and Machine Learning, Spring, Boston University and

Northeastern University, 2021. URL https://dpcourse.github.io/2021-spring/lecnotes-web/lec-08-cdf.pdf.

[66] Tianhao Wang, Joann Qiongna Chen, Zhikun Zhang, Dong Su, Yueqiang Cheng, Zhou Li, Ninghui Li, and Somesh Jha. 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'21, pages 1237–1253, New York, NY, USA, November 2021. ACM. ISBN 978-1-4503-8454-4. doi: 10.1145/3460120.3484750. URL https://doi.org/10.1145/3460120.3484750.

[67] Stanley L. Warner. Randomized Response: A Survey Technique for Eliminating Evasive Answer Bias. *Journal of the American Statistical Association*, 60(309):63–69, March 1965. ISSN 0162-1459. doi: 10.1080/01621459.1965.10480775. URL https://www.tandfonline.com/doi/abs/10.1080/01621459.1965.10480775.

[68] Qingqing Ye, Haibo Hu, Ninghui Li, Xiaofeng Meng, Huadi Zheng, and Haotian Yan. Beyond Value Perturbation: Local Differential Privacy in the Temporal Setting. In *40th IEEE Conference on Computer Communications*, INFOCOM'21, pages 1–10. IEEE, May 2021. doi: 10.1109/INFOCOM42981.2021.9488899. URL https://doi.org/10.1109/INFOCOM42981.2021.9488899.

[69] Sinan Yıldırım, Kamer Kaya, Soner Aydın, and Hakan Buğra Erentuğ. Differentially Private Frequency Sketches for Intermittent Queries on Large Data Streams. In *2020 IEEE International Conference on Big Data*, BigData'20, pages 4083–4092. IEEE, December 2020. doi: 10.1109/BigData50022.2020.9377786. URL https://ieeexplore.ieee.org/document/9377786.