UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR IT-SICHERHEIT

Estimating Input Distributions of Partial Background Knowledge for Differential Privacy Mechanisms

Abschätzen von Input-Verteilungen partieller Hintergrundwissen für Differential Privacy Mechanismen

Bachelorarbeit

im Rahmen des Studiengangs
IT-Sicherheit
der Universität zu Lübeck

vorgelegt von
Jannik Westenfeld

ausgegeben und betreut von
Prof. Dr. Esfandiar Mohammadi und B. Sc. Yara Schütt

Lübeck, den 18. November 2022

Abstract

Privacy notions like Differential Privacy can provide provable privacy guarantees for data accesses, however these guarantees come with a loss in utility. This results in a reluctance to use DP mechanisms in scenarios where precise data is a necessity. Partial Knowledge Differential Privacy tries to tackle such scenarios and provide privacy guarantees assuming a weaker attacker scenario.
This thesis analyses Partial Knowledge Differential Privacy and if it really provides a gain in utility over standard Differential Privacy notions.
To achieve this goal, we continue the research work from Arnold & Pätschke, which constructed two different algorithms. The algorithms estimate the data distribution of attacker background knowledge sets and use these estimations to calculate noise scale parameters which guarantee Passive Partial Knowledge Differential Privacy results. We show these algorithms to be correct and PPK-DP. We further evaluate the utility of them against state-of-the-art DP algorithms by comparing the calculated scales of noise needed to achieve similar results.
We achieve results that show that, under some assumptions about the data distribution of databases, the provided algorithms are competitive with standard DP results, achieving better utility for realistically sized databases while providing the same privacy guarantees against the weaker attacker scenario. Thereby we show that Partial Knowledge Differential Privacy can provide quantitative improvements for utility, but needs further research into it to guarantee support for generic data distributions.

Zusammenfassung

Differential Privacy ist eine Eigenschaft, mit welcher nachweisbar Datenschutz für Verfahren garantiert werden kann, die auf privaten Daten arbeiten. Diese garantierte Privacy kommt jedoch mit einem inhärenten Verlust der Genauigkeit in den Ergebnissen. Dies führt jedoch dazu, dass in Bereichen in denen präzise Daten erforderlich sind häufig DP-Mechanismen aufgrund ihrer Ungenauigkeit nicht verwendet werden. Partial Knowledge Differential Privacy versucht, für solche Szenarien dennoch Datenschutzgarantien unter der Annahme eines schwächeren Angreifers zu bieten.
In dieser Arbeit wird untersucht, ob mit Partial-Knowledge Differential Privacy Verfahren tatsächlich eine Nutzungsverbesserung über bekannte Differential Privacy Verfahren erzielt werden kann.
Für das erreichen dieses Ziels, führen wir die Forschungsarbeit von Arnold & Pätschke fort, in welcher zwei verschiedene Algorithmen konstruiert wurden. Die Algorithmen schätzen die Datenverteilungen von möglichen Hintergrundwissen ab und verwenden diese Schätzungen um additiven Noise zu skalieren. Die Ergebnisse können dann verwendet werden, sodass Passive Partial Knowledge Differential Privacy Garantien gelten gegenüber Angreifern, die ein schlechteres Hintergrundwissen besitzen als die Abschätzung. Wir zeigen in dieser Arbeit, dass die zwei Algorithmen korrekt funktionieren und Ausgaben liefern, die PPK-DP Eigenschaften erfüllen. Wir bewerten den Nutzen dieser Algorithmen im Vergleich zu modernen DP-Algorithmen, indem wir die jeweils berechneten Rauschskalierungen vergleichen, welche für garantierte Privacy Eigenschaften erforderlich sind.
Wir zeigen, dass die bereitgestellten Algorithmen unter bestimmten Annahmen über die Datenverteilung von Datenbanken mit modernen DP-Algorithmen konkurrieren können. Wir können die selben Privacy Garantien unter Annahme eines schwachen Angreifers erfüllen und erhalten dabei eine nachweisbare Nutzungsverbesserung. Wir zeigen damit, dass Partial Knowledge Differential Privacy quantitative Verbesserungen im Nutzens bieten kann. Es wird aber weitere Forschung benötigt, um die Unterstützung für generische Datenverteilungen zu garantieren.

Erklärung

Ich versichere an Eides statt, die vorliegende Arbeit selbstständig und nur unter Benutzung der angegebenen Hilfsmittel angefertigt zu haben.

Jannik Westenfeld
Lübeck, 09. Dezember 2022

Contents

List of cited Statements

List of proven Statements

List of Corollaries

List of Algorithms

1 Introduction

Privacy has become a more and more relevant necessity in IT settings throughout the last decade. Especially the search for provable privacy notions has been on the rise. Differential Privacy is one such established notion of modern privacy systems in IT settings. Differential Privacy guarantees are achieved by adding scaled noise assuming worst-case adversaries, hiding the influence of single data points on mechanism outputs. While ensuring privacy, adding noise results in a deterioration of utility in the results. This utility deterioration can lead to problems in sectors where precise data is needed [7, 24], often leading to a reluctance to implement DP strategies. This leads to the question of how to improve the utility of deferentially private results.
In differential privacy, it is assumed that an attacker knows all the data except for one challenge element. In most realistic settings, this over-approximates any real attacker by a wide margin. On average, adversaries will only know some subset of the accessed data. Suppose the possible background knowledge of all realistic adversaries would be known or could be approximated. In that case, additive noise can be calibrated to this realistic background knowledge instead of the over-approximation.
(*Active & Passive*) Partial Knowledge Differential Privacy, as introduced by Desfontaines, Mohammadi, Krahmer, and Basin in their 2019 paper "Differential privacy with partial knowledge" [6], tries to tackle this topic by providing a setting with verifiable models for such weak attackers. While the theoretical setting is provided, no mechanism for achieving the specified boundaries was proposed, leaving open the question if the introduced setting has real-world merit.
Building upon this, Arnold, Pätschke, Berndt, Mohammadi and Sommer in [1] developed such a mechanism for passive partial knowledge settings, but only under some conjectures. Their mechanism uses two algorithms to compute a noise scale parameter that provides a passive partial knowledge differentially private output when used as σ for additive Gaussian noise. Furthermore, as no working example of their work was provided, an evaluation of the mechanism and the possible improvement on the utility is left unresolved.

1.1 Contributions

We continue the work from Arnold, Pätschke, Berndt, Mohammadi, and Sommer in [1]. We do this by

- providing a general overview of the work done so far
- reanalysing their mechanism construction
- refactor the algorithms used in the mechanism
- resolve all necessary conjectures.

We improve on the provided privacy proofs by

- improving on the structure
- providing additional proofs for assumed ground truths

We show the constructions to suffice passive partial knowledge differential privacy.
We furthermore provide working code for the mechanism. We analyse the implementation by

- discussing problems met during implementation and their solutions
- evaluating achievable results
- testing the achieved utility against currently achievable utility using a state-of-the-art (ε, δ)-DP mechanism.

Lastly, we discuss generalisations for the usage of the mechanism.

1.2 Related Works

The topic of Partial Knowledge Differential Privacy is built upon the concepts of Differential Privacy and Noiseless Privacy. Differential Privacy as one of the standard models of modern privacy notions is a constant in privacy research. New concepts for special use cases are introduced, for example the work in [35] which analyses DP guarantees using asymmetric noise distributions, while older notions are re-evaluated with new attacker scenarios[18] or in entirely new settings [12, 25].
Noiseless privacy, and following from that partial knowledge differential privacy, however, has not received much research in recent years. Partial Differential Knowledge was recently discussed by Desfontaines in a blog post [5] and reintroduced in the book "Guide

to Differential Privacy Modifications" [26] by Pejó and Desfontaines. Besides these two occurrences no active public research has been published since the release of the original paper in 2019 [6].
Noiseless privacy, while also not the focus of most privacy related research, has been the focus of two research works in the papers "Noiseless Privacy" [11] and "Development and Analysis of Deterministic Privacy-Preserving Policies Using Non-Stochastic Information Theory" [10] by Farokhi in 2019. The papers reintroduce into the concept of noiseless privacy and iterate on the original concepts from 2014 by Bhaskar et al [3].

2 Preliminaries

Privacy can, at times, be a complex concept to grasp. To introduce the topic, we use this chapter to discuss current formal notions of privacy and their use cases. Starting with concise definitions of databases and access mechanisms, we define the environment in which privacy notions are considered. Afterwards, we define the concept of Differential Privacy and Privacy Loss as currently used privacy notions on which we will iterate in this thesis.

2.1 Basic Concepts

Before defining privacy notions used in this thesis, we first need to formally define in which context we will be working. For this, we first discuss the assumed base setting, data sets/databases.

Definition 2.1 (Data Set / Database).
Let $\mathcal{X}$ be the power set of all elements and $E \in \mathcal{X}$ a set choice of elements.
A database D is defined as an array of elements $e \in E$. We denote the ith position of a database with $D[i]$ and I as the set of all indices for a given database D.
The set of all possible databases is defined as $\mathcal{D}$.

For conformity's sake we'll be using the term database throughout the thesis.
When talking about databases, we will also need to consider a way to describe if the data has some inherent structure, if specific data points correlate with each other, or if other compositional oddities occur in it. For this, we abstract these concepts with stochastic distributions; from here on out, also called data distributions.

Definition 2.2 (Data Distribution).
Let D be a database and E a set choice of elements.
A data distribution θ describes the element composition in D, i.e. how often each $e \in E$ exists in D, as well as possible correlations of entries.
The support of the distribution is denoted as $\text{supp}(\theta)$. It describes the set of elements that have an occurrence probability greater than 0, i.e. the support describes the set of all elements e that are at least once in D.
The set of all possible data distributions θ is, furthermore, denoted as Θ.

When acting in stochastic settings, we model selections from distributions as so-called random variables. Random variables define the set of possible outputs with the occurrence probabilities of the outputs scaled to the stochastic distribution.

Definition 2.3 (Random Variables).
Random variables are described as a measurable function over a set of evaluations O with distribution θ. Distribution θ describes the probability with which each value in O is assigned to the random variable.
We denote a random variable that outputs values in set O as $RV(O)$.
The support of a random variable $\text{supp}(RV(O))$ is described as the subset of all evaluations $o \in O$ such that $o \in \text{supp}(\theta)$.

A database contains a given set of information that users can access. Working with databases, we need a way to access the data contained within them. For this, we define the concept of a query.

Definition 2.4 (Query).
Let D be a database adhering to $\theta \in \Theta$ and $I = \{1, \ldots, |D|\}$ the set of all indices.
A query q is defined as an access to database D where a user may choose a subset $J \subseteq I$ or a subset $F \subseteq E$ as input and learn either $D[i]$ for all $i \in J$ or i for all $D[i] \in F$.
Formally we write these two options as

$$q : \mathcal{D} \times I \rightarrow \{D[i] \mid i \in J\} \tag{2.1}$$

$$q : \mathcal{D} \times E \rightarrow \{i \mid D[i] \in F\} \tag{2.2}$$

In most use cases, it suffices to know how many elements in a database fulfil some condition ω, rather than learning more precise information about the database [19]. For example, it might suffice us to know that more than 10000 people live in a city rather than learning the names, addresses, etc., of these 10000 citizens.
For such scenarios, we define a special case query that looks at each entry in a database, counts how many data points equal a specified element or suffice some constraints, and returns that count. We call such a query a counting query.

Definition 2.5 (Counting Query [19]).
Let D be a database and $e \in E$ a user-chosen element.
A counting query $q : \mathcal{D} \times E \to \mathbb{N}$ takes a database D and a specified element e and returns how often that specific element is present in the database.

$$q(D, e) = |\{i \mid D[i] = e\}| \tag{2.3}$$

If e is clear from context, we omit it.

With this, we have all the necessary definitions to describe the setting we will work in throughout the thesis. Advancing over setting descriptions, we now need to dwell on more complex topics.
Before we dive into privacy notions, however, we still need one more concept introduced: noise. As we will see in the next section, most formal definitions of privacy use noise to prove their notions of privacy.

Definition 2.6 (Noise).
We define noise as a stochastic distribution θ over the real numbers with mean μ and variance σ, denoted by random variable $\mathcal{N}$. If not otherwise specified, we assume $\mu = 0$ and $\sigma = 1$. Adding noise to deterministic numeric outputs $o \in O$ transforms the output set to a random variable over the real numbers:

$$\mathcal{N} \times O \to RV(\mathbb{R}) \tag{2.4}$$

Noise can also be used over natural numbers ($\mathbb{N}$) and integers ($\mathbb{Z}$). For this, we only need to round the output to the nearest valid output.
When we provide a parameter β, we consider it to be a scaling parameter for the variance, i.e. the variance to be $\beta \cdot \sigma$. We formally write $\mathcal{N}(\beta)$.

Common types of noise we need for privacy guarantees are Laplace noise [9] and Gaussian noise [9]. Other types of noise distributions are used in practice [28, 38] but are not discussed in this thesis.
With the basic concepts down, we can tackle formal definitions of privacy.

2.2 Formal Definitions of Privacy

Privacy as a concept has been researched for the last few decades. While, at first, not a prevalent research field, with the ever-evolving field of information technology, privacy has become more and more important by the day. For example, negligence regarding personal privacy in data collection and the resulting consequences [22, 4] show that provable privacy compliance is needed more than ever.
Therefore, the raised question is how to properly provide privacy in data collection while still allowing for data usage in data-dependent fields, like the medical sector or machine learning.
First methods often tried anonymising data, as an example k-Anonymity [30, 34], but were shown time and time again to fail in realistic settings [8]. Other techniques, like access control policies [29], that provide privacy by barring access to databases entirely, break down if we want third parties to access the data for evaluation purposes and not simply store it.
Other approaches tried adding noise to outputs to veil the influence of elements, but, in most cases, it was not provable if the added noise provided any privacy guarantees [13, 9].
The biggest problem with most approaches was how to formalise the problem statement of privacy. A clear and concise model is needed, which can provide provable results even in realistic settings.
One of the first steps to such a model is to formalise private access to databases. For this, the concept of mechanisms was introduced as the standard model.

Definition 2.7 (Mechanism [9]).
Let $D \in \mathcal{D}$ be a database and $e \in E$ a challenge element.
A mechanism is a function that approximates a query result on D and e. If the used query is a counting query, a mechanism $M : \mathcal{D} \times E \to RV(\mathbb{N})$ is then further defined as a function that takes a database $D \in \mathcal{D}$ and an element $e \in E$ as inputs and maps them to the space of random variables over natural numbers $RV(\mathbb{N})$.
When adding an index β to a mechanism $\mathrm{M}_\beta(D, e)$, β is assumed a noise scale parameter and the mechanism is assumed as a counting query on e. The mechanism $\mathrm{M}_\beta(D, e)$ then works by adding noise $\mathcal{N}(\beta)$ to the counting query, resulting in an output of form:

$$M_\beta(D, e) = q(D, e) + \mathcal{N}(\beta) \tag{2.5}$$

Similarly to counting queries, if e is clear from context, we omit it.

Using mechanisms private database accesses can now be modelled. The next step is to model the probability of an attacker breaking the privacy constraints given by the access mechanism. A similar problem was discussed in the cryptographic sector a few decades before, introducing the concept of indistinguishability and the corresponding left-right game used to formalise it.

Remark 2.8 (Cryptographic Left-Right Game [20]).
In cryptography, the goal is to achieve provable security against poly-time attackers when building encryption schemes. A poly-time attacker is defined as an attacker that can make at most $O(n^k)$ calculation steps, where k is some constant. Any cipher c generated via an encryption scheme Enc(m), where m is some message, should not provide any information about m except for its length. The idea is to model the attacker in a worst-case scenario and prove that even in such a scenario, the attackers' advantage is only negligible in a security parameter ρ.
In such a worst-case scenario, the attacker can specify two arbitrary messages m_0, m_1 of equal length, one of which will be encrypted using Enc($\cdot$). As the messages are arbitrary, it can be assumed that the two provided messages give the attacker the best chance of success.
To prove the security of the encryption scheme, it is then enough to only model the two distinct cases Enc(m_0) and Enc(m_1) and the probabilities with which any attacker would answer m_0 or m_1 in them. The scheme is assumed to flip a coin with result $b \in \{0, 1\}$ to choose which message m_b to encrypt and the attackers answer to consist of a $b' \in \{0, 1\}$ describing which message $m_{b'}$ they think was encrypted. This means we only need to

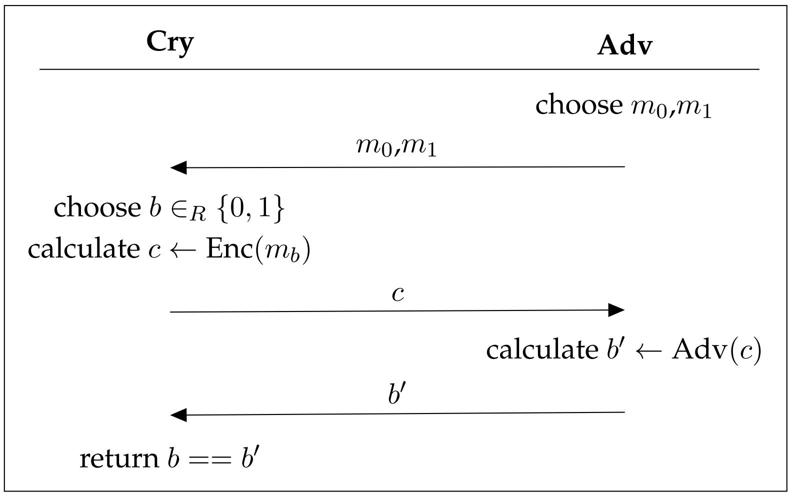


Figure 2.1: The cryptographic Left-Right Game
The security of the encryption process Enc can be modelled as a game between an adversary Adv and a cryptographer Cry. Adv sends two messages m_0, m_1 to Cry and receives the encryption of one of the two messages as a challenge c returned. The choice of the message m_b for challenge c is assumed to be perfectly random. If Adv can discern with a non-negligible advantage which message was chosen, then the encryption system is deemed not secure.

determine the four probabilities:

$$\Pr\big[b' = x | b = y\big] \text{ for } x, y \in \{0,1\} \tag{2.6}$$

Choice b is assumed to be randomly chosen in the game, meaning by randomly choosing their answer as well, all attackers' can trivially achieve a 50% win rate. If this win rate can now be upper-bounded for an attacker by an additive parameter negl(ρ) where

$$\text{negl}(\rho) < \frac{1}{2^\rho} \tag{2.7}$$

then the attackers' advantage is called negligible.
If such a bound is found to hold in a worst-case scenario, then this implies for all weaker attackers that the same bound must hold as well. The encryption scheme secure is then deemed secure. Encryption schemes that achieve this bound are called (*chosen plain-text attack*) indistinguishable.
The left-right game is visualised in Figure 2.1.

When simply transposing the left-right game to the privacy context, we get the problem that, in the case of privacy, the requirements for such a game are way too strict. The left-right game transposed would require that any two arbitrarily chosen databases cannot be distinguished from their output when queried. However, this means that all databases would be able to produce all outputs equally likely, which breaks the utility constraints we want to achieve.

The concept of neighbouring databases was introduced to loosen these requirements.

Definition 2.9 (Neighbouring databases).
Let D_0 and D_1 be two databases of equal size.
D_0 and D_1 are neighbouring if the databases differ at most in one element overall data positions, meaning there exists a permutation p for database D_1 such that

$$|\{i \mid D_0[i] \neq D_1[p_i]\}| \leq 1 \tag{2.8}$$

When two databases are neighbouring, we will denominate this with $D_0 \sim D_1$.
The influence of the one differing element on the output is called the data sensitivity and denominated with λ.

Defending only against neighbouring databases will help loosen security constraints, but more is needed to lessen the strict guarantees of the cryptographic left-right game.

Assuming the privacy game would be the cryptographic left-right game that only needs to hold for neighbouring databases instead of for arbitrary ones, we run into a problem.

Let adv(D_i,D_j) be an attackers advantage to distinguish between databases D_i and D_j.

Let us now further assume that we have databases D_0, D_1 and D_2 with $D_0 \sim D_1$ and $D_1 \sim D_2$, but $D_0 \nsim D_2$ and output $M(D_2, e) = o$. With the adapted privacy game, the advantage of any attacker to test if o was produced by $M(D_2)$ or $M(D_1)$ should be negligible, meaning adv(D_1,D_2) $=$ negl(ρ). For the test, if o was produced by $M(D_0)$ or $M(D_2)$ this must not hold, meaning adv(D_0,D_2) $>$ negl(ρ).

However, we also assumed $M(D_0) \sim M(D_1)$, meaning that the advantage between the two databases is at adv(D_0,D_1) $=$ negl(ρ).

This results in the following contradiction:

$$\begin{aligned}
\Pr[M(D_1,e)=o] &\overset{D_1\sim D_2}{=} \underbrace{\Pr[M(D_2,e)=o]}_{=\Pr[M(D_0,e)=o]+\mathrm{adv}(D_0,D_2)} - \underbrace{\mathrm{adv}(D_1,D_2)}_{=\text{negl}} && (2.9)\\
&\overset{D_2\not\sim D_0}{=} \Pr[M(D_0,e)=o] + \underbrace{\mathrm{adv}(D_0,D_2)-\text{negl}}_{=\xi>\text{negl}} && (2.10)\\
&= \Pr[M(D_0,e)=o]+\xi && (2.11)\\
&\overset{D_0\sim D_1}{>} \Pr[M(D_0,e)=o] + \underbrace{\mathrm{adv}(D_0,D_1)}_{=\text{negl}} && (2.12)\\
&= \Pr[M(D_1,e)=o] && (2.13)
\end{aligned}$$

Even though we only force neighbouring databases to be indistinguishable, this still implies that all databases need to be indistinguishable via the transitive properties shown above. The indistinguishability requirement must, therefore, also be loosened for something less constraining.
The concept of ε-indistinguishability was introduced to solve this problem.

Remark 2.10 (ε-indistinguishability).
ε-indistinguishability, similar to the negligible constraint in the cryptographic left-right game, bounds the advantage that any attacker could differentiate between two outputs. Instead of an additive term of negligible size, the allowed advantage is now bounded by a multiplicative term e^ε.
Let D_0 and D_1 be two databases, the two databases are $\varepsilon-$indistinguishable in regards to a mechanism M when the probabilities of all outputs o are bounded by

$$e^\varepsilon \cdot \Pr[M(D_0)=o] \geq \Pr[M(D_1)=o] \geq e^{-\varepsilon}\cdot \Pr[M(D_0)=o] \quad (2.14)$$

ε-indistinguishability, together with the concepts of neighbouring databases and mechanisms, build the basis for Differential Privacy.

Definition 2.11 (Differential Privacy [9]).
A mechanism M suffices $\varepsilon-$DP (Differential Privacy) if for any neighbouring databases D_0 and D_1, the mechanism outputs $M(D_0)$ and $M(D_1)$ are $\varepsilon-$indistinguishable, meaning for all possible outputs o it holds that

$$e^{-\varepsilon} \leq \frac{\Pr[M(D_0)=o]}{\Pr[M(D_1)=o]} \leq e^\varepsilon \quad (2.15)$$

Standard $\varepsilon-$differential privacy already brings massive improvements to provable privacy but is limited by the fact that we have hard bounds that may not always be obtainable. A mechanism may not fulfil ε-DP norms with 1-in-a-trillion odds, despite producing correct and differentially private outputs in all other cases. $\varepsilon-$differential privacy would not support such a mechanism, despite its obvious use cases.
If we also want to consider such mechanisms, we must iterate on the definition of differential privacy. A way to include a mechanism with low odds of leaking privacy is to introduce a small error bound, upper bounding the probability of privacy leaking errors. We call this error δ. To formalise this, we define $(\varepsilon, \delta)-$Differential Privacy.

Definition 2.12 ((ϵ, δ)-Differential Privacy [9]).
A mechanism M suffices $(\varepsilon, \delta)-$DP (Differential Privacy) if for any neighbouring databases D_0 and D_1, the mechanism outputs $M(D_0)$ and $M(D_1)$ are (ε, δ)-indistinguishable, meaning for all possible outputs o it holds that

$$\Pr[M(D_0) = o] \leq e^{\varepsilon} \cdot \Pr[M(D_1) = o] + \delta \tag{2.16}$$

and

$$\Pr[M(D_1) = o] \leq e^{\varepsilon} \cdot \Pr[M(D_0) = o] + \delta \tag{2.17}$$

where δ is an additional error bound.
δ can be understood as the cumulative probability of all outputs of M for which the specified $\varepsilon-$indistinguishability does not hold. As we cannot guarantee privacy in these cases, we want δ to be small.

When evaluating differential privacy results for some mechanisms, we include one additional notion; Privacy Loss. Privacy loss describes the logarithmic scale of the differential privacy term and is used to represent the DP-inequalities more concisely to simplify further analyses.

Definition 2.13 (Privacy Loss [9]).
Given two databases D_0 and D_1 as well as a mechanism M, the Privacy Loss of each output o is defined as the logarithmic ratio between occurrence probabilities of o using D_0 and D_1 respectively as input for M:

$$PL_{M,D_0,D_1}(o) := \ln\left(\frac{\Pr[M(D_0) = o]}{\Pr[M(D_1) = o]}\right) \tag{2.18}$$

One such analysis, which is also often used, is the Privacy Loss Distribution.

Definition 2.14 (Privacy Loss Distribution [9]).
Let PL_{M,D_0,D_1} be the privacy Loss random variable of a mechanism M. The distribution of all possible values in PL_{M,D_0,D_1} can be described as the following Lebesgue integral over all possible observations.

$$\omega_{M,D_0,D_1}(y) = \int_{\{o|PL_{M,D_0,D_1}(o)=y\}} \Pr[M(D_0) = o]dy. \tag{2.19}$$

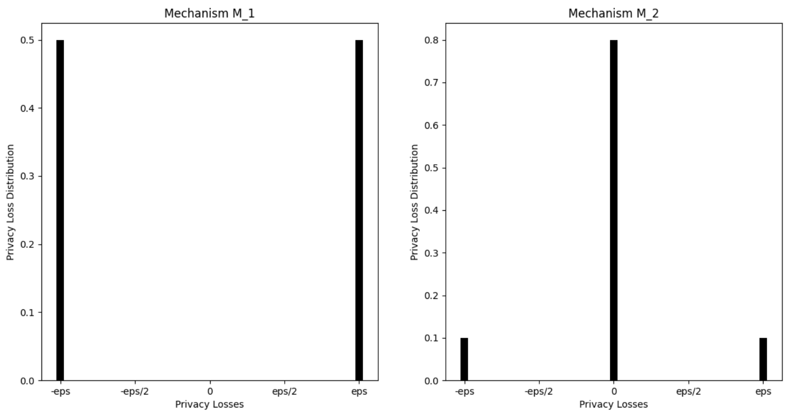


Figure 2.2: Two $\varepsilon-$DP mechanisms that share the same DP guarantees but different privacy loss distributions. While in the case of mechanism M_1, we have strict ε bounds for the privacy losses, for M_2, we can see that it is very likely that the privacy loss is 0 and only in some cases ε.

We can use privacy loss distributions to quantify and compare the usability of mechanisms. Two mechanisms may suffice the same (ε, δ)-DP parameters, but the individual privacy losses per mechanism run may vary massively. One mechanism might always produce privacy losses close to ε, while another only reaches ε in one case and stays far below the bound in all other cases. An example that showcases this possible difference between privacy loss distributions of two $\varepsilon-$DP mechanisms is provided in Figure 2.2.
With this, we introduced all necessary concepts for current privacy understanding. While we won't actively use them for the research topic of this thesis, we will use them for comparing achieved results.
In the next chapter we will iterate on the definitions of differential privacy, introducing the context of partial knowledge privacy settings and the formal definitions of active partial knowledge differential privacy and passive partial knowledge differential privacy. Furthermore, we will state the goals of this thesis in relation to these new concepts.

3 Problem Statement

In this chapter, we will introduce into the main research field of this thesis, partial knowledge differential privacy.

We will start by discussing some of the problems differential privacy faces in use cases where precise data is a necessity. We will introduce noiseless privacy a first concept to handle such problem cases and, finally, partial knowledge differential privacy as a compromise between the two extremes.

From this we will transfer to the research work by Arnold & Pätschke in [1], which builds the basis of the work in this thesis. We will introduce the general ideas achieved in their research and which statements were left open. Furthermore, we will discuss how to continue on with their research.

Lastly, we state the scope and goals of this thesis.

3.1 Setting the context

We always assume a worst-case attacker when considering privacy in differential privacy settings. In most cases, this over-approximates the actual threat any realistic attacker poses. However, this is by design. By using an over-approximation, we can guarantee specified privacy levels ($\varepsilon-$*indistinguishability with error* δ) against any real attackers.

While we can guarantee privacy constraints under these notions, using differential privacy mechanisms often results in a loss of utility from the queried database. To provide DP results, we generally add noise to query outputs, resulting in approximate results. While this introduced error can in some cases be acceptable, these inaccuracies prove to be a problem in use cases where we need precise data.

For example, medical sciences need patient data to be as precise as possible. Patient data will in a lot of cases be used to analyse disease progressions and the effectiveness of procedures. For these analyses it is crucial to remove as many points of failure. A similar problem exists in the banking sector, where inaccuracies in collected data could veil malicious transactions. In such critical use cases, this poses the problem that introduced errors through standard DP results could lead to massive cascading errors in consecutive calculations & evaluations [7, 24].

So how can we remove privacy-induced errors?

Most of the induced errors in computations come from using additive noise distributions. The results are perturbed by adding a value chosen from the noise distribution scaled to the database size and sensitivity of the data. This noise is not representative of any data we possess and, in most cases, results in the accounting error being, by design, unpredictable. Researchers tried to mitigate these errors by reducing the additive noise as much as possible without breaking guaranteed privacy bounds. Most commonly evaluated was Gaussian noise, resulting in the following first lower bound for its scale, referenced as Pure-DP.

Theorem 3.1 (Pure-DP [9]).
Let $\varepsilon > 0$ be arbitrary and λ be the sensitivity of the data.
For a parameter $c^2 > 2\ln(1.25/\delta)$ a mechanism M using Gaussian noise with scale parameter $\sigma \geq c\lambda/\varepsilon$ is (ε, δ)-differentially private.

Further research in [2] and [33] found another, even smaller, bound for Gaussian noise, improving utility even more.

Corollary 3.2 (Optimal σ for Gauss-Mechanism [2, 33]).
A Gauss mechanism $M : \mathcal{D} \times E \to RV(\mathbb{R})$ with $M(D, e) \sim \mathcal{N}(|\{i|D[i] = e\}|, \sigma^2)$ with sensitivity λ requires for a privacy loss $\delta := \delta(\varepsilon) \leq 1/4$ after c compositions

$$\sigma(\varepsilon, \delta, n) = \frac{\lambda\sqrt{c}}{\sqrt{2}\varepsilon}\left(\operatorname{erfc}^{-1}(2\delta) + \sqrt{(\operatorname{erfc}^{-1}(2\delta))^2 + \varepsilon}\right) \tag{3.1}$$

where erfc^{-1} *is the inverse of the well-understood complementary Gaussian error function.*

Plotting Pure-DP and Optimal σ calculated noise scale parameters shows that Optimal σ produces a better lower bound for the necessary noise. Some examples that visualise this behaviour can be seen in Figure 3.1.
While overall resulting in an improvement in utility over trivial DP-bounds [9], even Pure-DP and optimal σ scaled noise can impact results too much for critical use cases. Especially the noise scale being unrelated to the underlying data proves itself as being a massive problem. Uncertainty that still partially relies on underlying data distributions could increase the usability of outputs, but would loosen possible privacy guarantees.

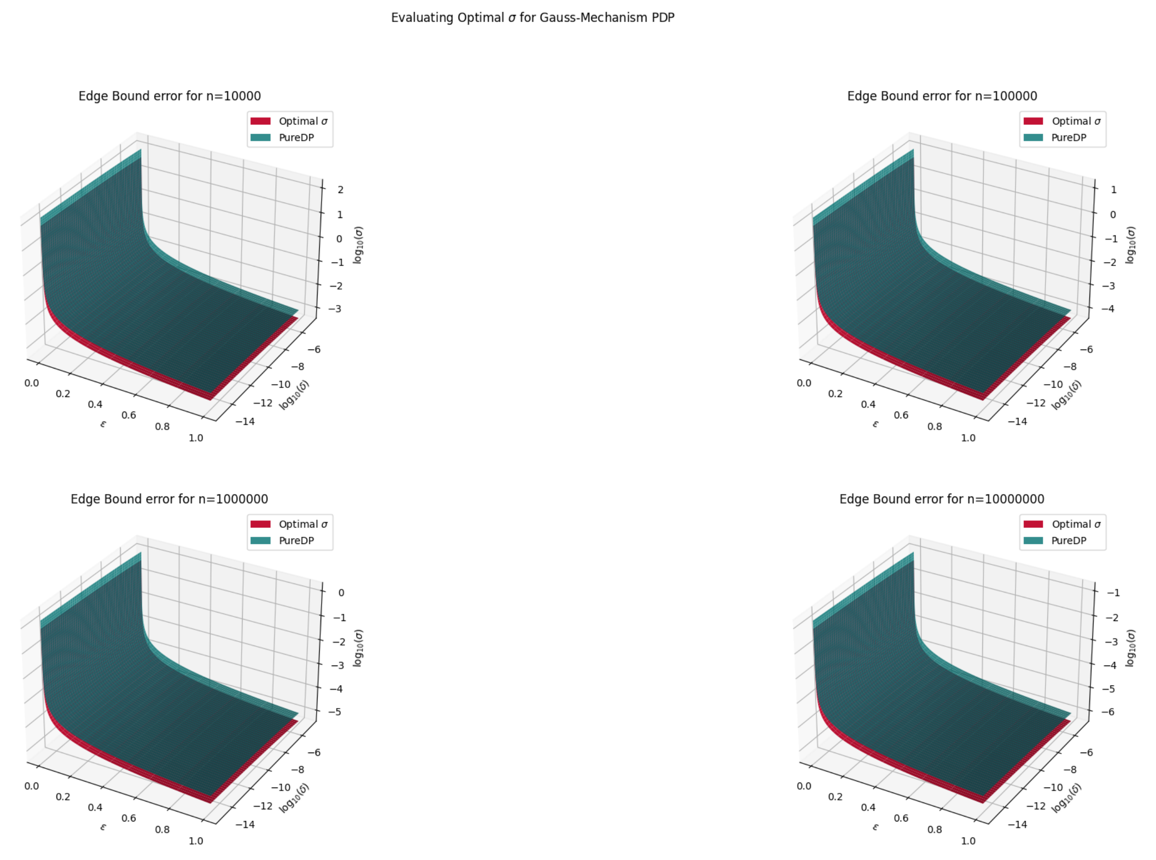


Figure 3.1: Example plottings of noise scale parameters calculated with Pure-DP and optimal σ for databases D of size $1e4$, $1e5$, $1e6$ and $1e7$, sensitivity $1/|D|$ and one composition. The results for Optimal σ construct a lower bound for all results of Pure-DP.

3.1.1 Noiseless privacy

As an interesting point of research researchers introduced the notion of noiseless and deterministic privacy [3, 11] to construct uncertainty scales based on underlying data. Instead of adding noise, inputs and databases are assumed inherently noisy or at least modifiable without additive noise to reach noisy properties.

For example, the influence of single elements in databases could be modified by some mechanism, making outputs solely reliant on data distributions while hiding the impact each element actually had on outputs.

It has been shown that noiseless privacy achieves results similar to (ε, δ)-DP for data-independent queries. The guarantees aren't based on the independence of added noise as in (ε, δ)-DP, but from inferences and assumptions about underlying data distributions and reduction of supported settings. For example, noiseless privacy as described in [3]

reduces its spectrum of outputs to Boolean functions or counting queries. Furthermore, these guarantees break down when queries are run multiple times on the same data [3, 6]. The same breakdown of privacy guarantees occurs if correlations in the original data exist that are not inherently visible. This privacy leakage is similar to problems with concepts like k-anonymity as mentioned in Section 2.2.
In the best case, however, we would want the guarantees of (ε, δ)-DP while only assuming a noiseless privacy setting, providing privacy without inherently influencing data distributions.
However, research into the topic shows that, at most, we can achieve a compromise between achieving noiseless differential privacy and accessing correlated data [6].

But how can we find such a compromise?

Data for critical use cases should, at least in theory, be stored in a comparatively safe environment. The amount of possibly leaked data should, therefore, also be limited and, in the best case, comparatively small in size. DP adversaries would then be worse than any privacy adversary encountered in realistic settings.
One idea proposed by Desfontaines et al. [6] tries to use this divergence between DP and realistic settings to gain privacy guarantees. The authors approximate possible attackers instead of always assuming worst-case scenarios. As stated above, in most cases, any attacker will only know a subset D' of the entire database D. This may be, for example, due to a data breach in the past, leaking entries in D or if a public database $\tilde{D}$ was used to create D.
Such a subset D' is assumed to be known by the attacker and classified as their background knowledge. We define such a background knowledge database D' as follows:

Definition 3.3 (Background Knowledge[1]).
Let D be a database with length n and indices $I = \{1, \dots, n\}$.
Background knowledge of a given database D is formalised as a sequence D' containing n' elements and a partial permutation $m \subset I$, which contains all indices of the known elements in regards to D. This means for a given partial permutation $m = (m_1, \dots, m_{n'})$ and $1 \leq m_k \leq n, \forall k$

$$D' = (D[m_k])_{k=1}^{n'} \tag{3.2}$$

holds.
Let now D' be a background knowledge set of length n' with data distribution θ'.
The background knowledge random variable $B_{n',n}$ describing the evaluations of D' is then instantiated with

$$B_{n',n} = \prod_{i=1}^{n'} \theta' \times Perm(n)[1 : n'] \tag{3.3}$$

where $Perm(n)[1 : n']$ describes the uniformly distributed random variable over all possible partial permutations length n' over n elements. We omit n' and n when they are clear from the context.

Generally, it is possible to quantify the information a database provider has about such open data to gauge the number of entries that any attacker could know. Assumedly they would, therefore, know the size of background knowledge D', but often not which exact data points are compromised. The size of a given background knowledge n' can be considered known to the database provider, the data distribution θ' with $D' \sim \theta'$ in most cases however not. Nonetheless, we still gain a new upper bound for the number of data points any attacker could know with $|D'|$ instead of $|D| - 1$ as assumed in standard DP notions. Furthermore, assuming a D' also implies a further subset D'' in our database D, which consists only of data points unknown to attackers. These points work in theory as described in noiseless privacy notions [3]. They perturb the output for possible attackers when those try to learn the influence of any one data point, but significantly do not influence the usability of the output for honest users [6].
We call these points non-compromised and define such a non-compromised data subset as follows:

Definition 3.4 (Non-compromised data subset).
For any database D with a sequence D' containing n' elements and a partial permutation m fixing all indices of the known elements in regards to D,

$$D'' = (D[i])_{i \notin m} \tag{3.4}$$

describes all non-compromised data points in the database D.

We can seemingly use non-compromised data to emulate noise against attackers, improving the utility of our mechanisms while still keeping privacy intact. While sounding promising, the informal structure of the concepts presented so far skim over some problems that do not seem evident at first glance. To further analyse this concept and potential problems, we formalise the concept as Partial-knowledge Differential Privacy.

3.1.2 Partial-Knowledge Differential Privacy (PK-DP)

When approximating an attacker, looking at a subset of all possible sets of background knowledge will, on average, result in non-representative approximations. Important edge cases could be ignored, and privacy would not be guaranteed. For example, consider a database containing zeros and ones pertaining to some distribution. We may assume the distribution bias to be similar in the background knowledge, but this must not be the case. However, through this the non-compromised data we use as natural noise will also be highly biased, leading to possible privacy leakage in worst-cases.
Partial-knowledge Differential Privacy notions use the formalised structure of background knowledge D' as introduced in Definition 3.3. By explicitly mentioning that D' must be from the support of all background knowledge subsets, the notions incorporate that we need to consider all possible background knowledge equivalence classes when estimating attacker. Through the introduced constraints, we remove this former inaccuracy.
As our standard models of analysing PK-DP notions, we use the privacy loss and loss distribution of mechanisms, similar to standard DP. While their use is similar, we need to adapt their definitions to the new setting, resulting in the following two re-definitions of privacy loss and privacy loss distributions under background knowledge.

Definition 3.5 (Privacy Loss Under Background Knowledge [1]).
Given a database D over the set of elements E, a background knowledge sequence D', with partial permutation $m = (m_1, \ldots, m_{n'})$ fixing all contained elements and position i with $i \notin m$ set to a or b, $a, b \in E$.
For each observation o the Privacy Loss random variable is defined as

$$PL_{M,D',m,a,b,i,bias}(o) := \left| \ln \left(\frac{\Pr_D[M(D) = o | D' = (D[m_k]_{k=1}^{n'}) \wedge D''[i] = a]}{\Pr_D[M(D) = o | D' = (D[m_k]_{k=1}^{n'}) \wedge D''[i] = b]} \right) \right| \tag{3.5}$$

assuming that indices start from 1.

Definition 3.6 (Privacy Loss Distribution Under Background Knowledge [1]).
Let $PL_{M,D',m,a,b,i,bias}(o) = y$ be the privacy loss random variable of a given mechanism M as described in Definition 3.5. Then

$$\omega_{M,D',m,a,b,i,bias}(y) = \Pr_{D \sim \theta_{|A}, \gamma \sim M(D)} [PL_{M,D',m,a,b,i,bias}(\gamma) = y] \tag{3.6}$$

describes the resulting Privacy Loss distribution.

With these two definitions, we will look at exact specifications for PK-DP.
Partial-knowledge Differential Privacy is, instead of being defined only once, divided into two different notions, Active Partial-knowledge Differential Privacy (APK-DP) and Passive Partial-knowledge Differential Privacy (PPK-DP). We need to do this to differentiate between two scenarios: one where attackers can influence the creation of the database by inserting their own data and one where they can not influence the creation.
We will first discuss APK-DP and afterwards PPK-DP.

Active Partial-knowledge Differential Privacy

APK-DP describes the setting where an attacker could actively manipulate the original database D before asking any queries. For example, an online questionnaire on chronic illnesses and their occurrence probability in a populous may be spammed with malicious inputs by an attacker during its data collection phase. In this case, we must assume that the attacker can more or less *'choose'* his preferred background knowledge D' by fine-tuning the submitted inputs to his liking.

We define active partial-knowledge differential privacy as follows:

Definition 3.7 (Active Partial Knowledge Differential Privacy (APK-DP) [6]).
Given a family of distributions Θ, a mechanism M achieves $(\Theta, \varepsilon, \delta)$-APK-DP (*Active Partial Knowledge Differential Privacy*) if for all distributions $\theta \in \Theta$, all indices i, all elements $a, b \in \operatorname{supp}(\theta')$ and all $(D', m) \in \operatorname{supp}(B)$:

$$\int_{\varepsilon}^{\infty} \omega_{M,D',m,a,b,i,bias}(y)(1 - \exp(\varepsilon - y))\,\mathrm{d}y \leq \delta. \tag{3.7}$$

holds.

It is, however, not only of interest to analyse databases where attackers can influence the data but also databases where attackers can only passively learn some of the data points. We describe this setting in the following.

Passive Partial-knowledge Differential Privacy

PPK-DP describes the setting where an attacker can only passively listen in on the original database D but not manipulate it in any way. We describe a similar example as for APK-DP. If the questionnaire on chronic illnesses is held in a controlled clinical setting, an attacker could not influence the data collection phase but could still gain background knowledge on a subset of the participants through means like social engineering. As the attacker cannot arbitrarily choose the data points he learns, his background knowledge will with high probability resemble the average case background knowledge.
We define passive partial-knowledge differential privacy as follows:

Definition 3.8 (Passive Partial Knowledge Differential Privacy (PPK-DP) [6]).
Given a family of distributions Θ, a mechanism M achieves $(\Theta, \varepsilon, \delta)$-PPK-DP (*Passive Partial Knowledge Differential Privacy*) if for all distributions $\theta \in \Theta$, all indices i and all elements $a, b \in \operatorname{supp}(\theta')$:

$$\mathbb{E}_{(D',m)\sim B}\left[\int_{\varepsilon}^{\infty} \omega_{M,D',m,a,b,i,bias}(y)(1 - \exp(\varepsilon - y))\,\mathrm{d}y\right] \leq \delta. \tag{3.8}$$

holds.

The question arises now, which of the two introduced cases should we put our research focus on?

So which notion should we further investigate?

APK-DP attackers are more powerful than PPK-DP attackers and all mechanisms sufficing APK-DP notions also suffice PPK-DP notions [6]. With this, it might seem like the optimal way to find mechanisms that achieve APK-DP and also use them in PPK-DP settings. However, as APK-DP attackers are significantly stronger, this both increases the precautions we need to uphold to keep privacy intact, reducing utility unnecessarily, as well as constructional complexity.
The research not only into APK-DP but also into PPK-DP sufficient mechanisms is an interesting research topic for utility maximisation in both settings. PPK-DP weakens the attacker significantly, at least in theory making mechanisms sufficing the notion easier to construct. It sounds more reasonable to start looking into PPK-DP mechanisms and then using the gained intuition to abstract concepts to APK-DP settings.
With these basic intuitions covered, let us look into the research work which this thesis is based on.

3.2 Introducing the work so far

The work of this thesis builds upon the research from Arnold & Pätschke in [1]. Their work research set out to construct a mechanism that can estimate approximately worst-case adversary distributions for background knowledge sets. With such a mechanism, such the idea, it would then be possible to upper bound the background knowledge of realistic attackers. The estimation could then be used as a bounding input to calculate additive noise reduced by the natural noise in the non-compromised rest of the data.
As we build upon their achieved goals in this thesis, we first recap the research results and discuss where we will continue.

So what results were achieved?

The research by Arnold & Pätschke resulted in two algorithms for mechanisms providing noise parameters that guarantee PPK-DP for specified (ε, δ) under a set of conditions, assumptions and, in some parts, conjectures. The assumptions are that databases and background knowledge sets follow a binomial distribution and that the size of the background knowledge is known to the defending party.
Furthermore, they first constructed an algorithm under the additional condition that the distribution of a specific attacker is known before the mechanism is executed. This is done to get a grasp on the concept of passive partial knowledge differential privacy.

The second algorithm releases this additional condition and instead approximates the distribution before calculating results. The first algorithm is used as a subroutine in the second algorithm on the estimated distribution.
Both algorithms will be introduced in detail later as a cornerstone of this thesis.

Were claims left open?

While the construction step of the algorithms was finished in [1] by Arnold & Pätschke, only one of the two algorithms has a completed privacy proof. The other algorithm has its privacy guarantees based on the following conjecture made by the researchers.

Conjecture 3.9 ($PL_{M,D',m,a,b,i,bias}$-bias correlation [1]).
Assuming all other variables M, D', m, a, b, i to be fixed, the larger the distance $|bias - 0.5|$, the higher is the privacy loss $PL_{M,D',m,a,b,i,bias}$

Due to time constraints, a proof of the conjecture was deemed out of scope for the work. However, crucial parts of the approximation and resulting privacy guarantees depend on it, meaning further research into the correctness of the statement is mentioned as a future goal. A second conjecture is also provided, but it is not necessary for the privacy proof and would only improve the efficiency of intermediary calculations in the algorithm. The conjecture will be stated, in context, later in Section 4.3.2.
Next to the conjectures, further improvements over the provided privacy proofs can be envisioned. As provided in Arnold & Pätschkes work [1], some parts of the proofs are assumed correct without reasoning why this should be the case. As an example, it is stated that all (ε, δ)-DP algorithms are also $(\varepsilon, \delta, \Theta)$-PPK-DP with no citation or intuitions given why this statement is correct. A thorough rewrite of the privacy proofs is, therefore, another point of interest.
A code implementation of the algorithms was furthermore deemed out of scope for the original research work. An example implementation and corresponding evaluation of realistically achievable privacy guarantees represent a further point of interest.
These open questions give us a reason to investigate further into the research already done. Trying to provide a complete and stable basis for the privacy proofs will be the focus of this thesis.

3.3 Scope of this thesis

As mentioned above, the scope of this thesis pertains to close the open claims of the research by Arnold & Pätschke [1]. We also improve over the existing proof basis. The thesis, thereby, focuses on three parts, finishing up the open ends in the provided privacy proofs, implementing and lastly evaluating the algorithm.

Finishing the formal proof of Algorithm 2

In finishing the formal proof for the algorithm we need to achieve a spectrum of different goals.
Firstly we need to analyse the algorithms as proposed and refactor possible errors that exist in them. Afterwards we need to look at the formal basis for the privacy proofs and, should they not be up to standard, refactor them accordingly. We also need to look for possible leaps in logic, existent in the proofs as provided in [1] and properly pad them out.
When we have a stable basis, we can then advance and either proof or disprove Conjecture 3.9. As this conjecture is a crucial cornerstone for the proof, we will formulate this as a research question.

Research Question 1 (On Conjecture 3.9).
Is Conjecture 3.9 provably correct or can we find a provable statement providing similar guarantees?

By improving on parts of the proof and also removing any conjecture from the assumptions we should then be able to provide a clean, formal privacy proof for the algorithm. We describe this as a research question as well.

Research Question 2 (On Algorithm 2).
Can we insert our results to provide a complete proof for the correctness and privacy guarantees of Algorithm 2?

We will answer both of these research questions with the work in Chapter 4.

Evaluating the work

Next to finishing the formal proof of Algorithm 2, we also want to evaluate if we only achieve theoretical improvements when using Algorithm 2, or if we can also substantiate utility improvements in a real implementation. For such an evaluation, we need to construct a working implementation of the provided algorithms. We formulate this as our next research question:

Research Question 3 (Implementing Algorithm 2).
Can we construct a correct implementation of Algorithm 2.

Similarly, we want to formulate a research question regarding the evaluation and if we can substantiate utility improvements.

Research Question 4 (Evaluating Algorithm 2).
Can we achieve $(\Theta, \varepsilon, \delta)$-PPK-DP results that are competitive with (ε, δ)-DP results, assuming the same underlying database?

We will answer these two research questions with the work in Chapter 5.

4 Method Section

This chapter introduces two algorithms from [1] that suffice the PPK-DP notion as described in Definition 3.8. The first algorithm is constructed to suffice privacy guarantees against an attacker where the algorithm knows the attackers' background knowledge set and distribution. The second algorithm then loosens constraints and suffices privacy constraints even when only the size of the attackers' background knowledge set is unknown and the exact distribution must be approximated. As stated, we will provide proofs for both algorithms and their privacy guarantees.
We start this chapter by outlining the first algorithm constructed by Arnold & Pätschke [1] with slight modifications to fix minor errors and improve readability. Afterwards, we provide a formal proof that this algorithm suffices PPK-DP. After that, we provide the second algorithm constructed by Arnold & Pätschke [1], again with modifications to fix flaws in the original design and improve readability. We then provide a general outline of the proof and state open claims we need to substantiate before formalising our outline. We then prove the open claims, reforming them into working theorems. Lastly, we close the chapter with a formal privacy proof for Algorithm 2, showing it to be
$(\Theta, \varepsilon, \kappa_1 + \kappa_2 + \kappa_4 + \sum_{i=0}^{1} \mathrm{cdf}_{Lap(0,{}^{2}/_{\varepsilon})}\left(-{}^{n\cdot\tau_i}/{}_2 + 1\right))$-PPK-DP.

Assumptions

Some assumptions were noted in [1] for the following algorithmic constructions. Most of them are in regards to the data distribution of input databases and background knowledge sets. We introduce two assumptions which we will assume throughout the entire chapter. For all databases it is assumed, that their behaviour and construction follows a binomial distribution. This implies that each single entry of a database is, therefore, independent identically distributed (i.i.d.) as a Bernoulli experiment with some bias $bias_D$.
While binomial distributions, in general, do not represent a representative model for realistic databases [14, 23], using them here allows us to construct algorithmic structures that do not inherently need to cover edge cases not intrinsic to the algorithm but its inputs. We will discuss possible modifications to assumed data distributions for more realistic models of databases in Section 6.
A further assumption will be made regarding the size of background knowledge sets. In Section 3.1 we gave reasoning for the assumption that background knowledge sizes

can be estimated. In this chapter, we will assume that used background knowledge sizes are correct, as they are needed as input values for both algorithms. Problems with this assumption will be discussed in Section 6.
With this we covered necessary assumptions and can continue on with the first algorithm.

4.1 A privacy-preserving PPK-DP algorithm for known attacker

To get a clear perspective on how PPK-DP algorithms work, we first introduce an algorithm that, assuming we know the background data distribution θ', calculates for given $\varepsilon, \delta, n, n'$ a noise scale parameter β'^*. We will show that the calculated β'^* suffices our understanding of the PPK-DP notion as presented in 3.8.
The original construction can be found in [1].

4.1.1 Algorithmic outline

When calculating β'^*, it is enough for us if we approach the optimal β and end with a slightly bigger return parameter. To efficiently achieve this while not over-engineering the process, the bisection method is used to calculate a noise quantifier that suffices for our purposes. We use the equation in the following lemma from [1] as the minimisation function in the bisection method:

Lemma 4.1 (Minimal noise scale parameter [1]).
Let $M_\beta(D) = q(D) + \mathcal{N}(\beta)$. Given a set of β values fulfilling a slightly changed PPK-DP definition where the inequality is transformed into an equation

$$\mathbb{E}_{(D',m)\sim B}\left[\int_{\varepsilon}^{\infty} \omega_{M_\beta,D',m,a,b,i,bias}(y)(1-\exp(\varepsilon - y))\,\mathrm{d}y\right] = \delta. \tag{4.1}$$

The element β^*, which minimises the utility loss, is

$$\beta^* = \operatorname*{argmin}_{\beta} \mathbb{E}_{\tilde{D}\sim\theta}[|\mathcal{N}(\beta)|]. \tag{4.2}$$

Together with the general results of privacy research that more noise gives better privacy [9], we can state that:

Corollary 4.2 ((ε, δ)-β^*-dependence).
For a fixed ε, increasing β decreases the necessary error bound δ and vice versa when considering the PPK-DP notion. If δ is also fixed to create a distinct pair (ε, δ), then the optimal noise scale parameter β^* must also be distinctly unique and continuous in δ.

The general idea follows from this; upper and lower bound β^* and then iteratively narrow these bounds while the difference between them exceeds a stated error bound. At the very least, the upper bound should be a good approximation for β^*, differing from the exact solution by at most the specified error bound. For calculations of Equation 4.1, we can use the following two lemmas, also from [1]:

Lemma 4.3 (Simplified PLD $\omega_{M,D',m,a,b,i,bias}(y)$ [1]).
Let D be a given database and let in addition to $m \in Perm(n)[1:n']$, $m_{n'+1}, \dots, m_n$ be defined such that $D[m_{n'+1}, \dots, m_n] = D''$. Then the privacy loss distribution can be even further reduced to

$$\omega_{M,D',m,a,b,i,bias}(y) = \sum_{o=0}^{n''} \binom{n-n'-1}{o-q(a)} bias^{q(D)+o-q(a)} (1-bias)^{2n-n'-1-o+q(a)-q(D)} S_{q(D'),o,y}. \tag{4.3}$$

Lemma 4.4 (Describing $S_{D',o,y}$ [1]).
For the mechanism $M_\beta = q(D) + \mathcal{N}(\beta)$ and the distribution θ, we have

$$S_{D',o,y} = \sum_{\substack{\{\gamma \in \operatorname{supp}(M_\beta(D)) | \\ PL_{M_\beta,D',m,a,b,i,bias}(\gamma)=y\}}} \Pr_{v \sim \mathcal{N}(\beta)} [v = \gamma - o - q(D')]. \tag{4.4}$$

Having introduced the algorithmic outline, we can now continue on and provide an exact description of Algorithm 1.

4.1.2 The Algorithm

We initialise two variables during the algorithm's execution to keep track of our progress in calculating β'^*. The variable lo is initialised to keep a lower bound on β'^* such that we exceed our δ bound and variable hi to keep a higher bound on β'^* such that we stay below the δ bound. We further introduce a third variable, mid, as the centre value between lo

and hi, which we use in the update steps. Using these three variables, we can establish an interval that must include the β^* as defined by Lemma 4.1.
During the initialisation we do a binary approximation over possible β'^* until we find the interval with bounds $[2^{-i}, 2^{-i+1}]$ including β^*. We then store these interval bounds in separate variables $last\_lo$ and $last\_hi$ before starting further calculations with the bisection method to keep valid bounds. We do this in case any update we make during calculations breaks the lo or hi constraints.
During the bisection method, we first check if the interval between lo and hi is smaller than our allowed error. If so, we can terminate the algorithm as we will already produce outputs we deem good enough. Otherwise, we first update the middle of our interval and then compare using it as a β against the specified δ bound.
Depending on the result, mid is either smaller than, bigger than, or equal to β^*. We can either update one of our bounds lo and hi correspondingly or output mid as β'^*.
If we updated lo or hi, we then check if our update broke them as bounds for β^* analogous to our check of mid. If not, we update our stored correct bounds with the value we just checked. We continue this as long as lo and hi are further apart than our allowed error bound.
In the end, we update mid one last time to the midpoint between the last valid bounds we encountered and then check if mid upper bounds β^*. If so, we return it; otherwise, we return the last valid upper bound we recorded. A pseudo-code version of this algorithm is provided in Algorithm 1.
Having described a generic run of Algorithm 1, we continue onward to its privacy proof.

4.1.3 Proof & Privacy Analysis

Having described Algorithm 1, we now prove its privacy guarantees. For this, we show the following lemma:

Lemma 4.5.
Algorithm 1 is $(\Theta, \varepsilon, \delta)$-PPK-DP for a given pair of ε, δ.

While a privacy proof already existed for Algorithm 1 in [1], we deemed it necessary to refactor it. This is partially due to our slight modification to its construction, as well as the fact that the original proof was kept short on purpose, but thereby skipping over some in our opinion crucial parts of it.

Algorithm 1: Noise computation for a known θ' using the bisection method.

```
Input: DB $D, D' \sim \theta', bias, n', \varepsilon, \delta, error, steps$
Result: $\beta'^*, \zeta_{error,update(steps)}$ with $\beta'^* - \beta^* \leq \zeta_{error,update(steps)}$
$lo = 0$
$hi = 0$
$mid = 0$
$init = 1$
while $\mathbb{E}_{(D',m)\sim B}\left[\int_{\varepsilon}^{\infty} \omega_{M_{init}(D),D',m,a,b,i,bias}(y)(1-\exp(\varepsilon - y))\,\mathrm{d}y\right] < \delta$ do
    $hi = init$
    $init = init/2$
    $lo = init$
end
while $\mathbb{E}_{(D',m)\sim B}\left[\int_{\varepsilon}^{\infty} \omega_{M_{init}(D),D',m,a,b,i,bias}(y)(1-\exp(\varepsilon - y))\,\mathrm{d}y\right] > \delta$ do
    $lo = init$
    $init = 2 \cdot init$
    $hi = init$
end
$mid = lo$
$last\_hi = hi$
$last\_lo = lo$
while $lo + error \leq hi$ do
    $mid = (lo + hi)/2$
    if $\mathbb{E}_{(D',m)\sim B}\left[\int_{\varepsilon}^{\infty} \omega_{M_{mid}(D),D',m,a,b,i,bias}(y)(1-\exp(\varepsilon - y))\,\mathrm{d}y\right] < \delta$ then
        $hi = mid - steps$
        if $\mathbb{E}_{(D',m)\sim B}\left[\int_{\varepsilon}^{\infty} \omega_{M_{hi}(D),D',m,a,b,i,bias}(y)(1-\exp(\varepsilon - y))\,\mathrm{d}y\right] < \delta$ then
            $last\_hi = hi$
        end
    else if $\mathbb{E}_{(D',m)\sim B}\left[\int_{\varepsilon}^{\infty} \omega_{M_{mid}(D),D',m,a,b,i,bias}(y)(1-\exp(\varepsilon - y))\,\mathrm{d}y\right] > \delta$ then
        $lo = mid + steps$
        if $\mathbb{E}_{(D',m)\sim B}\left[\int_{\varepsilon}^{\infty} \omega_{M_{lo}(D),D',m,a,b,i,bias}(y)(1-\exp(\varepsilon - y))\,\mathrm{d}y\right] > \delta$ then
            $last\_lo = lo$
        end
    else
        return $\beta'^* = mid, \zeta = 0$
    end
    update$(steps)$
end
$mid = (last\_lo + last\_hi)/2$
if $\mathbb{E}_{(D',m)\sim B}\left[\int_{\varepsilon}^{\infty} \omega_{M_{mid}(D),D',m,a,b,i,bias}(y)(1-\exp(\varepsilon - y))\,\mathrm{d}y\right] > \delta$ then
    return $\beta'^* = last\_hi, \zeta = last\_hi - last\_lo$
end
return $\beta'^* = mid, \zeta = last\_hi - last\_lo$
```

Proof (for Lemma 4.5).
Let $ppk(\beta) := \mathbb{E}_{(D',m)\sim B}\left[\int_{\varepsilon}^{\infty} \omega_{M_\beta,D',m,a,b,i,bias}(y)(1-\exp(\varepsilon - y))\,\mathrm{d}y\right]$.
We use the bisection method to calculate a maximal β'^{*} such that $ppk(\beta'^{*}) \leq \delta$ for a given δ. The bisection method only works for functions that are continuous, which we know with Corollary 4.2 to be the case for $ppk(\beta)$. We now need to check whether we always return valid values or not.
The first steps consist of looping through two while loops. This happens from line (5) to (9) and line (10) to (14) respectively. By doing this we ensure that hi and lo are valid boundaries for β'^{*}. With Corollary 4.2 we know β'^{*} to exist, meaning these two while loops will terminate, and, by design, will result in $ppk(hi) < \delta$ and $ppk(lo) > \delta$. We set $last\_hi$ and $last\_lo$ using these valid bounds afterwards and initialise mid as the midpoint value between the two bounds.
Within the algorithm a result may be returned at three distinct locations; mid in line (31), $last\_hi$ in line (37), and mid, once again, in line (39). As we know $ppk(\beta)$ to be continuous, we must return at one of the three locations. Otherwise, we would have found a position at which $ppk(\beta)$ can't be evaluated, i.e. a distinct combination of ε and β with no linked δ, creating a contradiction with Corollary 4.2.
We now evaluate the three return locations and if our results suffice $(\Theta, \varepsilon, \delta)$-PPK-DP.
When mid is returned in line (31), we beforehand checked in line (20) and (25) whether $ppk(mid) < \delta$ or $ppk(mid) > \delta$. We only enter this case if both tests failed, therefore, the only option left is that $ppk(mid) = \delta$. As $ppk(\beta)$ equals our PPK-DP equation for given ε and noise scale parameter β, and we found with mid a value exhausting its inequality to an equality with a given δ, we know this return value to suffice $(\Theta, \varepsilon, \delta)$-PPK-DP.
The next possible return is $last\_hi$ in line (37). For our algorithm to get to this point, it must have terminated the while loop from line (18) to (34) beforehand, so we must have $lo + error > hi$. Furthermore, this while loop must break at some point, as we, at the very least, half the distance between lo and hi every step. By using a further variable $steps > 0$ in each update step we more than half the distance, possibly reducing the amount of loop iterations and calculations needed. At the very most we need $\log(error)$ steps.
We only set $last\_lo$ and $last\_hi$ after first initialisation in line (28) and line (23) respectively. The check in the line beforehand guarantees that they are still valid bounds when we update them. We, therefore, know $last\_hi$ at all times after initialisation to suffice $(\Theta, \varepsilon, \delta)$-PPK-DP, as $ppk(last\_hi) < \delta$.
Another candidate for our return value is mid as it may also fulfil the PPK-DP inequality and, if so, would be a better bound than $last\_hi$. However, mid may also produce a too large δ as we do not check $ppk(mid)$ when setting it. This is also exactly the case if line (37) is reached. Returning $last\_hi$ is however, as already argued, a safe fallback option. We,

therefore, know that we suffice $(\Theta, \varepsilon, \delta)$-PPK-DP in this case.

The last return happens on the last line (39) and returns mid. At this point, we already checked in the if-statement before (line (36)) if $ppk(mid) > \delta$ holds. Since this was not the case, we have $ppk(mid) \leq \delta$. With this, we know the return value to suffice $(\Theta, \varepsilon, \delta)$-PPK-DP.

Following from the fact that all return values suffice $(\Theta, \varepsilon, \delta)$-PPK-DP and the fact that we must return at some point, we can say that Algorithm 1 is $(\Theta, \varepsilon, \delta)$-PPK-DP. $\square$

In an actual implementation ppk could be calculated using introduced Lemma 4.3 and 4.4 as a convolution of the underlying binomial distribution of D'' and used noise $\mathcal{N}(\beta)$. To analyse the run-time of Algorithm 1 independently of this calculation, we will, however, assume an oracle to compute the function ppk.

The run time of Algorithm 1 is then in $O(\log max_\beta)$ for max_β being the maximal β value such that $ppk(\beta) \leq \delta$, but $ppk(\beta - error) > \delta$. The endpoints are initialised in $O(\log max_\beta)$ (see line (5) to (14)), as we initialise them using a binary search. The while-loop is in $O\left(\log \frac{max_\beta}{error}\right)$, as we, again, use a binary search between the two endpoints, but stop when we reach an interval size of at most error.

At worst, we need as much noise as calculated by optimal Gaussian σ (see Corollary 3.2). (ε, δ)-DP describes, by design, an upper bound for needed noise scales. If our noise exceeds optimal Gaussian σ, instead of making further calculations, we can simply return the output the optimal σ as a safe result. This results in a worst-case complexity $O\left(\log \frac{\lambda\sqrt{c}}{\sqrt{2\varepsilon}}\left(\operatorname{erfc}^{-1}(2\delta) + \sqrt{(\operatorname{erfc}^{-1}(2\delta))^2 + \varepsilon}\right)\right)$ [2, 33].

We can now continue onward to Algorithm 2.

4.2 A privacy-preserving PPK-DP algorithm for approximated attacker

Now that we analysed an algorithm that, given a fixed, known background knowledge distribution, suffices $(\Theta, \varepsilon, \delta)$-PPK-DP, we want to advance over it. In general, we will not know the exact data distribution an adversary would know for a given database. Instead, we need to find a way to privately approximate such a distribution with a quasi-worst case which we can then use as an upper bound.

Arnold & Pätschke [1] provided a basis for this in their work research, providing such an algorithm but not finishing the proof for it. In this section, we will recap the work done so far by outlining the algorithm and proven notions which we need to proof its correctness and privacy guarantees.

4.2.1 Algorithmic outline and necessary notions

We need a comparison value when approximating the quasi-worst-case background knowledge for a given database. For this, we use the privacy loss of possible background knowledge sets. As to not test all background knowledge biases in calculations, we privately estimate a range of supported biases beforehand, based on the database D. With the already introduced Conjecture 3.9, if proven correctly, we could then use the interval bounds as inputs, as they would upper-bound the privacy loss. By Corollary 4.2 we then also know these calculated noise scale parameters to also suffice for all cases with less privacy loss, meaning all biases in the support range.
However, with only a subset of all biases, this would still result in exponentially many computations, as we would still need to consider each possible permutation in the background knowledge. Arnold & Pätschke [1] researched this problem and proved the following two statements:

Claim 4.6 ($PL_{M,D',m,a,b,i,bias}$ only depends on $q(D')$ and not specific D' [1]).
The privacy loss $PL_{M,D',m,a,b,i,bias}$ does not directly depend on D' but on $q(D')$, that is for all $m, \tilde{m} \in Perm(n)[1 : n']$ with additionally defining $m_{n'+1}, \dots, m_n$ such that we have $D[m_{n'+1}, \dots, m_n] = D''$ for a given D and for all

- $A = D[m_1, \dots, m_{n'}] = D' \wedge D''[i] = a$ *and* $i \notin \{j_1, \dots, j_{n'}\}$,
  $B = D[m_1, \dots, m_{n'}] = D' \wedge D''[i] = b$ *and* $i \notin \{j_1, \dots, j_{n'}\}$,
- $\tilde{A} = D[\tilde{m}_1, \dots, \tilde{m}_{n'}] = \tilde{D}' \wedge \tilde{D}''[i] = a$ *and* $i \notin \{\tilde{j}_1, \dots, \tilde{j}_{n'}\}$,
  $\tilde{B} = D[\tilde{m}_1, \dots, \tilde{m}_{n'}] = \tilde{D}' \wedge \tilde{D}''[i] = b$ *and* $i \notin \{\tilde{j}_1, \dots, \tilde{j}_{n'}\}$

with $q(D') = q(\tilde{D}')$ the following holds

$$PL_{M,D',m,a,b,i,bias}(\gamma) = PL_{M,\tilde{D}',m,a,b,i,bias}(\gamma). \tag{4.5}$$

Corollary 4.7 (Privacy Loss Distribution dependencies [1]).
The function $\omega_{M,D',m,a,b,i,bias}$ depends only on the query results of the database and the challenge element and not on specific values of D' and a.

It follows from these two statements that it is enough to consider one example for each possible evaluation $q(D')$ of D' instead of all possible permutations, as all permutations will result in the same privacy loss value.

If we can, therefore, correctly and privately estimate such a quasi-worst case background knowledge $Est_{D'}$ with $q(Est_{D'})$ producing worse privacy losses than most attackers, then we could use $Est_{D'}$ in Algorithm 1 to produce a PPK-DP sufficing β-noise scale parameter. As we will only support attackers with background knowledge sets having less privacy loss than our estimation, we add the occurrence probability of worse attackers to our allowed error δ. Combining this with the approximation for the bias support range we can then also upper bound the β-noise scale parameter, not only for the bias used in our $Est_{D'}$ estimation, but for all supported biases.
For these approximations, we further use the previously introduced Lemma 4.3 and Lemma 4.4.
With this, we introduced all necessary proven statements from [1] and continue by describing the here outlined algorithm.

4.2.2 The Algorithm

We start by taking all information we have from D to start the background knowledge approximation, its size and distribution, and use it to calculate the bias of the data. Using this bias in further calculations would leak privacy; we, therefore, need to perturb it using noise.
For this, we use Laplace noise with half our privacy budget ε to not introduce cases where we cannot guarantee privacy. We denote the perturbed bias $bias_D$ and use it for further calculations.
Next, we need to approximate the boundaries for attackers which we support. We use the bounds given by the Chernoff-Hoeffdinger theorem for additive error bounds [16]. As these bounds only work on non-noised inputs, we assume $bias_D$ to be a non-noised centre value for our background knowledge bias and bound the support range around it with τ_1 and τ_2.
Using the Chernoff-Hoeffdinger theorem, we introduce errors, which we check against our given δ budget and reduce it appropriately. We also check if the Laplace noise we chose initially exceeds our supported boundaries and return a safe over-approximation should this be the case. For this the original construction uses Pure-DP (see Theorem 3.1) which we supplant with the optimal Gaussian noise (see Corollary 3.2).
Afterwards, we now have a range of biases centred around $bias_D$ which our algorithm will support. As we need to obtain an exact background knowledge database estimation for use in Algorithm 1, we now need to approximate a quasi-worst case that bounds the background knowledge privacy loss with high probability.
We construct all possible background knowledge set classes using $bias_D$ as our approximation distribution bias and calculate each class's privacy loss and occurrence probability.

Afterwards, we consider the class with the worst-case privacy loss after removing classes up to an error of κ_1 to be our approximation. The probability of the attackers real background knowledge to be worse than our estimation is thereby bounded by κ_1 and we subtract this from the δ error budget.
We now use the proven $(\Theta, \varepsilon, \delta)-$PPK-DP Algorithm 1 to construct a noise parameter on the worst-case biases allowed by our approximation. We use the rest of our privacy budget here to construct the noise.
Lastly, we check if our noise parameter is worse than what comparative (ε, δ)-DP algorithms provide and return the better parameter as our output. We again use optimal Gaussian noise (see Corollary 3.2) as our comparing value.
A pseudo-code version of this algorithm is provided in Algorithm 2.
Having outlined the algorithm, we need to take the next step and prove its correctness and privacy guarantees. For this, we first outline the general proof.

4.3 Outlining the privacy proof for Algorithm 2

Before tackling a formal proof, we first outline our general strategy. We have two steps during the algorithm where we need to guarantee our choices to be ε-indistinguishable, the Laplace noise when privately choosing $bias_D$ and the usage of Algorithm 1 as a subroutine. We also have four parts during the algorithm where we must guarantee that we do not exceed δ-error boundaries. We split our δ-error accordingly into four parts $\kappa_1, \kappa_2, \kappa_3$ and κ_4.
First, we take the knowledge about the entire database to construct an approximation for attackers as well as a specified support range $[bias_{lo}, bias_{hi}]$. We need to show that we do not base our approximation solely on our database, as this would leak privacy. We show that we do not leak privacy by using Laplace noise for estimations. We also need to prove that the probability for any attacker to be outside of our support range is bounded by the allowed error κ_2. For this, we use the well-understood Chernoff-Hoeffdinger theorem for additive errors to exclude worst-case attackers with a cumulative occurrence probability of κ_2 and show that the provided bounds meet our requirements.
Secondly, we consider the worst cases in our allowed range of biases and other errors which exist in these cases. We need to prove that we upper bound the possible error for such worst cases and have enough error budget to cover them, meaning the produced error does not exceed κ_3. Otherwise we terminate.
Thirdly, we need to choose a quasi-worst-case background knowledge satisfying our approximation, as we need a fixed background knowledge for later calculations. For this, we brute force all privacy losses of possible background knowledge sets and their respective

Algorithm 2: Noise computation for an unknown θ' distribution

```
Input: DB $D, n', \varepsilon, \delta, \kappa_1, \kappa_2, \kappa_3$ with $\kappa_1 + \kappa_2 + \kappa_3 = \delta$
Result: $\beta^* = \text{argmin}_\beta \, \mathbb{E}_{\tilde{D} \sim \theta} [|\mathcal{N}(\beta)|]$
$n = |D|$
$c = q(D)$
$bias = {}^{c}/_{n}$
$\ell = {}^{Lap(2/\varepsilon)}/_{n}$
$bias_D = bias + \ell$
$[bias_{hi}, bias_{lo}] = bias_D \pm \ell$
$\tau_1 = \max_{bias_i \in [bias_{hi}, bias_{lo}]} (x | \Pr[bias_D - \ell \leq bias_i - x/2] \leq \kappa_2/2)$    [16]
$\tau_2 = \max_{bias_i \in [bias_{hi}, bias_{lo}]} (x | \Pr[bias_D - \ell \geq bias_i + x/2] \leq \kappa_2/2)$    [16]
$\kappa_4 = \kappa_3 - \sum_{i=0}^{1} \text{cdf}_{Lap(0, 2/\varepsilon)}((-\tau_i/2 + \lambda) \cdot n) = \kappa_3 - \sum_{i=0}^{1} \text{cdf}_{Lap(0, 2/\varepsilon)}(-n \cdot \tau_i/2 + 1)$
if $|\ell| > \tau_1/2$ or $|\ell| > \tau_2/2$ or $\kappa_4 \leq 0$ then
    return $\beta^* = \frac{\sqrt{2 \cdot \log(1.26/\delta)}}{n \cdot \varepsilon/2}$        \\see Theorem 3.1
end
$base = \max(n' - (n - c), 0)$
$out = \min(n', c) - base$
$losses, prb = \underbrace{[0, \ldots, 0]}_{|\cdot| = out}$
for $i = 0$ to $out$ do
    $hits = base + i$
    $losses[i] = \left| \ln \left( \frac{\binom{n'}{hits} \cdot bias_D^{hits} \cdot (1 - bias_D)^{n' - hits}}{\binom{n'}{hits - 1} \cdot bias_D^{hits - 1} \cdot (1 - bias_D)^{n' - (hits - 1)}} \right) \right|$
end
$losses, idx = $ sort$[desc](losses)$
$idx += base$
for $i = 0$ to $out$ do
    $prb[i] = \frac{\binom{n'}{idx[i]} \binom{n - n'}{c - idx[i]}}{\binom{n}{c}}$
end
$exclude, wcr = 0$
while $exclude + prb[wcr] < \kappa_1$ do
    $exclude += prb[wcr]$
    $wcr += 1$
end
$Est_{D'} \in \{D' \sim \theta' \mid n' = |D'| \wedge wcr = q(D')\}$
$\beta = \max_{\xi_i \in \{-(\tau_1/2 + \ell), \tau_2/2 + \ell\}} ($Alg1$(D, Est_{D'}, bias_D + \xi_i, n', \varepsilon/2, \kappa_4, err, steps))$
return $\beta^* = \min \left( \beta, \frac{\sqrt{2 \cdot \log(1.26/\delta)}}{n \cdot \varepsilon/2} \right)$        \\see Theorem 3.1
```

occurrence probabilities. We need to prove that a quasi-worst-case background knowledge estimation we use upper bounds all further calculations. Furthermore, we need to prove that our choice has at least a chance of $1 - \kappa_1$ to upper bound the privacy loss of the real background knowledge.
Lastly, we use Algorithm 1 with our chosen worst-case background knowledge with the rest of our error budget, κ_4. The results' correctness follows from Algorithm 1 using a bisection method and it's privacy guarantees follow from the above-mentioned proof in Section 4.1.3. However, similar to the third part, we still need to prove that our results provide enough privacy guarantees for all our supported attackers and not only the one singular estimated case used in calculations.
If we can prove all four parts, we can add the error budgets per step as δ. The algorithm then produces noise that suffices $(\Theta, \varepsilon/2, \kappa_4)$-PPK-DP under estimations that suffice $(\Theta, \varepsilon/2, \delta - \kappa_4)$-DP. The sequential composition theorem for PPK-DP algorithms has been shown to hold in [6], meaning following with it, we can guarantee $(\Theta, \varepsilon, \delta)$-PPK-DP.
Formally, we will show the correctness of the following theorem.

Theorem 4.8.
For the family of distributions Θ, the mechanism M with its noise parameter β being computed in Algorithm 2 for all distributions $\theta \in \Theta$, all indices i, all $a, b \in \operatorname{supp}(\theta')$ satisfies the following inequality

$$\Pr_{(D',m)\sim B}\left[\int_{\varepsilon/2}^{\infty} \omega_{M_\beta,D',m,a,b,i,bias}(y)(1-\exp(\varepsilon/2 - y))\,\mathrm{d}y > \kappa_4\right] \le \kappa_1 + \kappa_2 + \left(1 - \mathrm{cdf}_{Lap(0,2/\varepsilon)}\left(\min_i\left(\tau_i/2\right) - \frac{1}{n}\right)\right). \tag{4.6}$$

Most of the proofs groundwork was already done in [1]. The notions necessary for the proof were also already provided in Section 4.2.1. However, as stated before, some notions need refactoring and necessary conjectures need to be proven before we can dive into a formal privacy proof. We first provide some updated notions that will help with our proof and afterwards discuss and prove necessary open conjectures.

4.3.1 Improved notions

In [1] Arnold & Pätschke focused on correctly modelling the background knowledge approximations and the corresponding privacy loss and privacy loss distribution. However, due to this focus, the initial privacy proof of the approximation of $bias_D$ and the supported bias interval was not thoroughly enough shown to hold bounds.

We provide the following statements with proofs to improve on the existing statements in [1], starting with a thorough analysis of the usage of the Chernoff-Hoeffdinger theorem as a subroutine.

Lemma 4.9.
Bounding the interval of possible results with $\mathbb{E}(M) - \tau_1$ and $\mathbb{E}(M) + \tau_2$ for a Laplace mechanism $M = q(D) + \ell$ using the Chernoff-Hoeffdinger theorem for additive error [16] for $\ell = Lap(0, {}^1\!/\!_\varepsilon)$ being scaled Laplace noise of at most absolute size $\min_{i\in\{1,2\}} ({}^{\tau_i}\!/\!_2)$ such that

$$\Pr\left[\mathbb{E}\left(q(\tilde{D})\right) \leq bias_j - \frac{\tau_1}{2}\right] \leq \frac{\delta}{2} \quad \wedge \quad \Pr\left[\mathbb{E}\left(q(\tilde{D})\right) \geq bias_j + \frac{\tau_2}{2}\right] \leq \frac{\delta}{2} \tag{4.7}$$

holds for $bias_j \in \{bias_{lo}, bias_{hi}\}$ with

$$\min\left(\mathbb{E}(M) - |\ell|\right) = bias_{lo} \leq \mathbb{E}(M) \leq bias_{hi} = \max\left(\mathbb{E}(M) + |\ell|\right) \tag{4.8}$$

, then M provides (ε, δ)-DP.

Proof (for Lemma 4.9). M being a Laplace mechanism with Laplace noise $\mathrm{Lap}(0, {}^1\!/\!_\varepsilon)$, we know this mechanism to be ε-DP. We now bound the allowed realm of output values with $\mathbb{E}(M) - \tau_1$ and $\mathbb{E}(M) + \tau_2$ as described by equation 4.7 and want to show that we do not exceed these bounds even with our noised input.
From the Chernoff-Hoeffdinger theorem [16], we know that $\tau_1, \tau_2 \geq 0$.
As the Chernoff-Hoeffdinger theorem [16] does not work on noised random variable inputs, we need to assume our input not to be noised and add the noise to the error we want to bound. We, therefore, assume the output of M to be a query over a secondary database $\tilde{D}$ with our perturbed random variable input as bias.
The assumptions in the lemma now give us:

$$\Pr\left[\mathbb{E}\left(q(\tilde{D})\right) \leq \mathbb{E}(M) \pm |\ell| - {}^{\tau_1}\!/\!_2\right] \overset{Eq.\ 4.8}{\iff} \Pr\left[\mathbb{E}\left(q(\tilde{D})\right) \leq bias_j - {}^{\tau_1}\!/\!_2\right] \qquad \leq \frac{\delta}{2} \tag{4.9}$$

The second bound follows analogously.

$$\Pr\left[\mathbb{E}\left(q(\tilde{D})\right) \geq \mathbb{E}(M) \pm |\ell| + {}^{\tau_2}\!/\!_2\right] \iff \Pr\left[\mathbb{E}\left(q(\tilde{D})\right) \geq bias_j + {}^{\tau_2}\!/\!_2\right] \qquad \leq \frac{\delta}{2} \tag{4.10}$$

As we bound the absolute value of $Lap(0, {}^1\!/\!_\varepsilon)$ with $\min_{i\in\{1,2\}} ({}^{\tau_i}\!/\!_2)$ we can refactor the term using this inequality. W.l.o.g we assume ${}^{\tau_1}\!/\!_2 = \min_{i\in\{1,2\}} ({}^{\tau_i}\!/\!_2)$.

It then follows that

$$\frac{\delta}{2} \geq \Pr\left[\mathbb{E}\left(q(\tilde{D})\right) \leq bias_j - {}^{\tau_1}\!/2\right] = \Pr\left[\mathbb{E}\left(q(\tilde{D})\right) \leq \mathbb{E}(M) \pm |\ell| - {}^{\tau_1}\!/2\right] \tag{4.11}$$

$$\geq \Pr\left[\mathbb{E}\left(q(\tilde{D})\right) \leq \mathbb{E}(M) - {}^{\tau_1}\!/2 - {}^{\tau_1}\!/2\right] \tag{4.12}$$

$$= \Pr\left[\mathbb{E}\left(q(\tilde{D})\right) \leq \mathbb{E}(M) - \tau_1\right] \tag{4.13}$$

and similarly for the second bound.

$$\frac{\delta}{2} \geq \Pr\left[\mathbb{E}\left(q(\tilde{D})\right) \geq bias_j + {}^{\tau_2}\!/2\right] = \Pr\left[\mathbb{E}\left(q(\tilde{D})\right) \geq \mathbb{E}(M) \pm |\ell| + {}^{\tau_2}\!/2\right] \tag{4.14}$$

$$\geq \Pr\left[\mathbb{E}\left(q(\tilde{D})\right) \geq \mathbb{E}(M) + {}^{\tau_1}\!/2 + {}^{\tau_2}\!/2\right] \tag{4.15}$$

$$\geq \Pr\left[\mathbb{E}\left(q(\tilde{D})\right) \geq \mathbb{E}(M) + {}^{\tau_2}\!/2 + {}^{\tau_2}\!/2\right] \tag{4.16}$$

$$= \Pr\left[\mathbb{E}\left(q(\tilde{D})\right) \geq \mathbb{E}(M) + \tau_2\right] \tag{4.17}$$

Equation 4.13 and Equation 4.17 now show the Chernoff-Hoeffdinger bounds evaluated at our bounding intervals $\mathbb{E}(M) - \tau_1$ and $\mathbb{E}(M) + \tau_2$. In Equation 4.12 and Equation 4.15 we used the worst-cases, either $bias_{hi}$ or $bias_{lo}$, for the used Laplace noise to achieve these bounds.
We assume $\pm{}^{\tau_1}\!/2$ as the bounds for our Laplace noise, meaning that $\ell \in [-{}^{\tau_1}\!/2, {}^{\tau_1}\!/2]$. Our worst-case inputs are then defined by $\ell = \pm{}^{\tau_1}\!/2$. This exhaust the inequalities in Equation 4.12 and 4.15 respectively for either $bias_{lo}$ or $bias_{hi}$. It can then be seen that, the probabilities of landing outside of each of these bounds are then equal to at most ${}^{\delta}\!/2$. As both cases can theoretically occur, we can upper bound the cumulative error by adding the two singular errors together.
This means our produced error is at most ${}^{\delta}\!/2 + {}^{\delta}\!/2 = \delta$. Together with the ε budget for the initial Laplace noise, we achieve (ε, δ)-DP through the induced error. □

We have shown that using the Chernoff-Hoeffdinger theorem for additive errors produces an (ε, δ)-DP result for given values. In Lemma 4.9 we assumed the Laplace mechanism to only produce results in a specified range and did not consider errors that exist, should the noise be outside of it.
From Lemma 4.9 we know that we achieve the support bounds exactly when we assume our Laplace noise values to be at most of absolute size $\min_{i\in\{1,2\}} ({}^{\tau_i}\!/2)$. Laplace noise values outside this range could, therefore, produce results outside of our support bound for biases, with some error we haven't accounted for yet.
Furthermore should our Laplace noise exhaust, but not exceed it's support size $\min_{i\in\{1,2\}} ({}^{\tau_i}\!/2)$, then another error exists.

Assume our estimated bias for the background knowledge and real bias exhaust one of the Chernoff-Hoeffdinger bounds. This would mean that the real background knowledge bias is exactly $\mathbb{E}(M(D)) - \tau_1$ or $\mathbb{E}(M(D)) + \tau_2$. However, when we now test against a neighbouring database $\tilde{D}$, this would result in $\mathbb{E}(M(\tilde{D})) = \mathbb{E}(M(D)) \pm \lambda$. The real background knowledge could in a worst-case then be $\mathbb{E}(M(\tilde{D})) - \tau_1 - \lambda$ or $\mathbb{E}(M(\tilde{D})) + \tau_2 + \lambda$, meaning outside of our allowed support range.

All values in the Laplace distribution that result in us landing outside of the support range must, therefore, be accounted for by our error. We model this by assuming the Laplace distribution to be truncated at the calculated bounds $\pm \min_{i \in \{1,2\}} (\tau_i/2)$ and consider the cumulative probabilities of us landing outside those bounds as another error.

We estimate the size of the error by the following lemma.

Lemma 4.10.
A truncated Laplace mechanism that uses noise parameter $1/\varepsilon$, with a query-sensitivity λ and that is cut at the data-dependent bounds $-\tau_1$ or τ_2 satisfies

$$\left(\varepsilon, \sum_{i=0}^{1} \mathrm{cdf}_{Lap(0,2/\varepsilon)} \left(-\tau_i + \lambda\right)\right)\text{-DP.} \tag{4.18}$$

Proof (for Lemma 4.10). We know that the Laplace mechanism using noise parameter $1/\varepsilon$ is ε-DP [9]. We now truncate the Laplace mechanism at both $-\tau_1$ to remove the interval $[-\inf, -\tau_1]$ and τ_2 to remove the interval $[\tau_2, \inf]$.

We can model a truncated Laplace mechanism using a standard implementation and the removed intervals as error δ. However, this does not cover all possible errors introduced by truncating the Laplace distribution in our specific case. As we compare against neighbouring data sets that produce outputs shifted by data sensitivity λ, we want our worst case noise to not exceed our truncation bounds, even if shifted by λ. As these bounds are data dependent, landing with added λ outside of the truncation would imply that using a neighbouring database would have resulted in different boundaries.

We, therefore, still need to consider the possibility of our bounds changing in a neighbouring database and adding it to the combined error. For this, we need to increase the interval by the data sensitivity λ to catch any edge cases as worst cases would now land exactly on the boundary instead of outside of them. We modify the intervals so that we either remove $[-\inf, -(\tau_1 - \lambda)]$ or $[\tau_2 - \lambda, \inf]$.

With these two introduced errors in mind, we consider the two truncation cases. When the distribution is cut at $-\tau_1$, then the resulting error is the cumulative probability of all possible removed results. We can write this error as:

$$\int_{x=-\infty}^{-\tau_1+\lambda} \mathrm{pdf}_{Lap(0,1/\varepsilon)}(x)dx = \mathrm{cdf}_{Lap(0,1/\varepsilon)}(-\tau_1+\lambda) \tag{4.19}$$

If the distribution would only be cut at $-\tau_1$, then it would satisfy $(\varepsilon, \mathrm{cdf}_{Lap(0,1/\varepsilon)}(-\tau_1+\lambda))$-DP.
Analogously when we cut the distribution at τ_2, then we can write the error as:

$$\int_{x=\tau_1-\lambda}^{\infty} \mathrm{pdf}_{Lap(0,1/\varepsilon)}(x)dx = 1 - \mathrm{cdf}_{Lap(0,1/\varepsilon)}(\tau_2-\lambda) = \mathrm{cdf}_{Lap(0,1/\varepsilon)}(-\tau_2+\lambda) \tag{4.20}$$

If the distribution would only be cut at τ_2, then it would satisfy $(\varepsilon, \mathrm{cdf}_{Lap(0,1/\varepsilon)}(-\tau_2+\lambda))$-DP.
In the considered case, we now not only cut the distribution at $-\tau_1$, but also at τ_2. As $\tau_1, \tau_2 > 0$, we know the two intervals to not intersect. The combined probability to land in either of the two cases can thereby be calculated by simply adding the two errors together.

$$\delta = \mathrm{cdf}_{Lap(0,1/\varepsilon)}(-\tau_1+\lambda) + \mathrm{cdf}_{Lap(0,1/\varepsilon)}(-\tau_2+\lambda) \tag{4.21}$$

$$= \sum_{i=0}^{1} \mathrm{cdf}_{Lap(0,1/\varepsilon)}(-\tau_i+\lambda) \tag{4.22}$$

□

With this we have shown that our subroutines for the *bias* approximation satisfy an (ε, δ)-DP bound, but not necessarily a $(\Theta, \epsilon, \delta)$-PPK-DP bound. Therefore, we also need to show the following.

Lemma 4.11.
All mechanisms that are (ϵ, δ)-DP are also $(\Theta, \epsilon, \delta)$-PPK-DP.

Proof (for Lemma 4.11). Let M be an (ϵ, δ)-DP mechanism.
By definition, we know that definition 2.12 holds for M assuming any two neighbouring databases D_0, D_1. Furthermore, for any background knowledge D' drawn from $\theta' \in \Theta$, this D' must be a subset of both D_0 and D_1 as the challenge element is unknown to attackers. We now rewrite the D' in relation to D_1 as described in definition 3.3:

$$D' = D_1[m_k]_{k=1}^{n'} \tag{4.23}$$

with n' being the size of the background-knowledge set. Furthermore, we write the rest of D_1 as

$$D'' = D_1 \backslash D' \tag{4.24}$$

following from definition 3.4. The following equation then holds:

$$D_1 = D' \cup D'' \tag{4.25}$$

As an adversary never knows the challenge element $D[i]$ in an (ϵ, δ)-DP scenario, we know that D'' must at least hold this element but could hold more. A worst case scenario for the analysis of $(\Theta, \epsilon, \delta)$-PPK-DP would be exactly the case that the background-knowledge set D' contains all elements, except the challenge element.
In all other cases, we could always reduce the analysis to this case and would get a valid result as we would ascribe adversaries more background knowledge than they have.
The worst case for $(\Theta, \epsilon, \delta)$-PPK-DP can, therefore, assuming $i \notin Perm(n)[1 : n-1]$ and $D[i]$ to be the challenge element, be described as:

$$D' = D_1[m_k]_{k=1}^{n-1} \tag{4.26}$$

D'' now only holds the challenge element, while all other positions are assumed to be known by the attacker. D'' only consisting of one challenge element describes exactly an (ϵ, δ)-DP scenario and our mechanism M can be used to achieve a sufficing result. The worst-case $(\Theta, \epsilon, \delta)$-PPK-DP scenario can be solved using an (ε, δ)-DP mechanism.
As a mechanism that suffices in the worst-case scenario will always suffice in all other cases, it follows that an (ϵ, δ)-DP mechanism will also suffice in all $(\epsilon, \delta, \Theta)$-PPK-DP cases and is thereby also $(\epsilon, \delta, \Theta)$-PPK-DP. □

With this, we can now venture onward from improving in [1] assumed notions and tackle the main territory of this section, answering research question 1. Can we prove or disprove Conjecture 3.9?

4.3.2 Extant Conjectures

In [1] the achieved results base their privacy claims on the correctness of the in Section 3.2 already introduced Conjecture 3.9. Using the conjecture it would be an implication that we can upper-bound the privacy loss of background knowledge sets by their biases. This is used in Algorithm 2 in line (31) when Algorithm 1 is called to guarantee that the returned results also upper bound the necessary noise.

However, this is not the only conjecture made by Arnold & Pätschke in [1]. The following conjecture is also made.

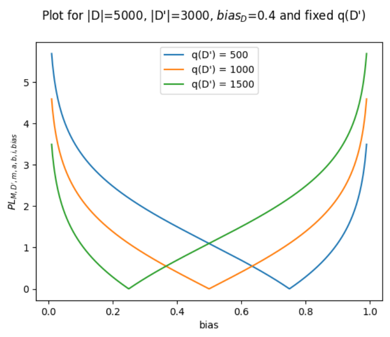
Conjecture 4.12 (Monotonically decreasing Privacy Loss Distribution [1]).
The function $\omega_{M,D',m,a,b,i,bias}(y)$ is monotonically decreasing in y.

While Conjecture 4.12 holding is not necessary for the algorithm to work correctly [1], it would imply that we could estimate deltas more efficiently than needing to calculate the entire privacy loss distribution. However, as its contribution is only run-time improving, we consider the conjecture to be out of scope for this thesis.
The question now remains, if we can prove or disprove Conjecture 3.9. Through analysing Conjecture 3.9, we come to the following conclusion:

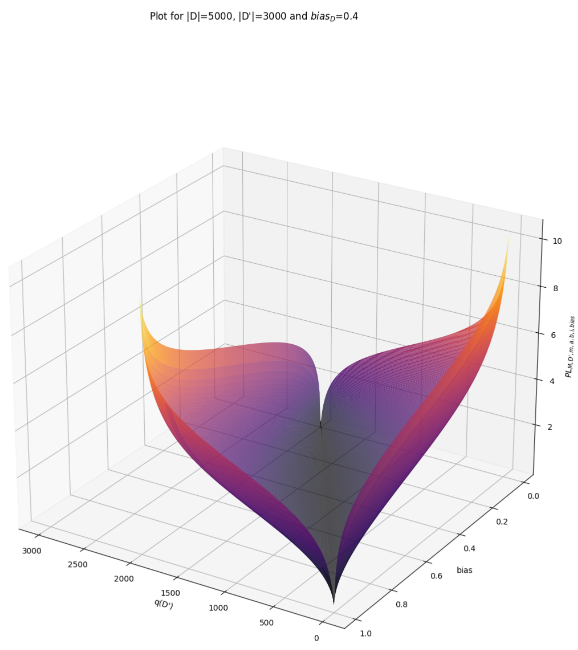
Claim 4.13.
Conjecture 3.9 is false.

Proof (for Claim 4.13). One simple way to disprove Conjecture 3.9 is to show one counter-example where the privacy loss $PL_{M,D',m,a,b,i,bias}$ is not growing with larger distance $|bias - 0.5|$ while all other variables are fixed.
We give such an example with $|D| = 5000$, $|D'| = 3000$, $bias_D = 0.4$, plotted in Figure 4.1a evaluated at three different values for $q(D')$. While the conjecture holds for $q(D') = 1000$, we can see that for $q(D') \in \{500, 1500\}$ it does not prove to hold. Instead the privacy loss appears to grow with larger distance $|bias - 0.75|$ for $q(D') = 500$ and $|bias - 0.25|$ for $q(D') = 1500$. We will see in the next section that this is indeed the case.
Instead of growing with a larger distance $|bias - 0.5|$, the privacy loss is minimising in some linear function dependent on $q(D')$. We will discuss the linear function in $q(D')$ and what it describes in Section 4.3.3. □

As we can see, Conjecture 3.9 does not hold, breaking all privacy guarantees that are based on it. It is, therefore, necessary to fix the conjecture before we are able to prove that Algorithm 2 is private. We need a theorem that provides similar guarantees for the bounds to finish the proof.
Plotting more of the privacy loss domain $PL_{M,D',m,a,b,i,bias}$ gives an intuition on how to advance. As a starting point, we plotted a representative subset of all possible $q(D')$ and $bias$ combinations for $|D| = 5000$, $|D'| = 3000$ and $bias_D = 0.4$. The resulting plane can be seen in Figure 4.1b.



(a) Plot of small example cases of $q(D')$ for the privacy losses $PL_{M,D',m,a,b,i,bias}$ with $|D| = 5000$, $|D'| = 3000$ and $bias_D = 0.8$ fixed and varying $bias$. The privacy loss minimises dependent on the value $q(D')$ instead of 0.5 as conjectured.



(b) Plot for the privacy loss plane $PL_{M,D',m,a,b,i,bias}$ with $|D| = 5000$, $|D'| = 3000$ and $bias_D = 0.4$ fixed in regards to varying $q(D')$ and $bias$. The privacy loss seems to minimise in a linear function dependent on $q(D')$.

Figure 4.1: Example plottings for $PL_{M,D',m,a,b,i,bias}$, showing that Conjecture 4.12 does not hold by providing an example where we can observe the privacy loss behaving antithetical to the conjecture.

Starting with the intuition from this plotting, our next goal is to fix the disproved conjecture.

4.3.3 Fixing Conjecture 3.9

As shown in the last section, the Conjecture 3.9 as introduced in [1] fails to hold. Furthermore, a critical section of our proof for algorithm 2 initially needs this conjecture to hold. We now face the problem to fix the proof by first fixing conjecture 3.9 and then showing that our fixed version gives the exact properties needed.
To fix the conjecture, we try and see how the privacy loss $PL_{M,D',m,a,b,i,bias}$ is influenced by the used $bias$ for its calculations. Figure 4.1 provides the intuition that the $bias$ seems to be inversely influenced by $q(D')$ in regards to increases in privacy loss. While at first seemingly counter-intuitive, we need to consider that the privacy loss is also dependent on the non-compromised data subset D''. D'' is exactly comprised of all elements of D

not in the background knowledge D'. We can make an informed guess that the observed inverse influence might be traceable to D''.

Based on this idea, we formulate the following theorem.

Lemma 4.14.
Let $bias_{sec}$ be the expected value of a fixed D'' as defined by Definition 3.4 and $bias$ the expected value of the background knowledge D'.
Assume that all variables besides $bias$ are fixed in $PL_{M,D',m,a,b,i,bias}$. With increasing, absolute distance between the two biases $bias$ and $bias_{sec}$ $|bias - bias_{sec}|$, the privacy loss $PL_{M,D',m,a,b,i,bias}$ also increases.

Proof (for Lemma 4.14). For a function to increase in an absolute distance equates to the function being monotonically decreasing for any values smaller than the comparative value and similarly monotonically increasing for any bigger values. The privacy loss $PL_{M,D',m,a,b,i,bias}$ must then be monotonically decreasing in background knowledge biases $bias \in [0, bias_{sec})$. Similarly, the privacy loss must increase monotonically in interval $bias \in (bias_{sec}, 1]$.

To show this functional behaviour, we need to analyse the first partial derivative of the Privacy Loss in the direction of $bias$. With the Privacy Loss as defined in Definition 3.5, we, therefore, need to determine:

$$\frac{\partial PL_{M,D',m,a,b,i,bias}(o)}{\partial bias} = \frac{\partial \left| \ln \left(\frac{Pr_D\left[M(D)=o | D[m_j]_{j=1}^{n'}=D' \wedge D''[i]=a\right]}{Pr_D\left[M(D)=o | D[m_j]_{j=1}^{n'}=D' \wedge D''[i]=b\right]} \right) \right|}{\partial bias} \tag{4.27}$$

To efficiently analyse this equation, we take a look at the innermost ratio of probabilities. The used probability fixes the background knowledge and challenge element, only keeping the non-compromised data points variable. We can use this to refactor the term as follows:

$$\frac{Pr_D\left[M(D)=o | D[m_j]_{j=1}^{n'}=D' \wedge D''[i]=a\right]}{Pr_D\left[M(D)=o | D[m_j]_{j=1}^{n'}=D' \wedge D''[i]=b\right]} \overset{[1]}{=} \frac{\binom{n-n'-1}{o-q(D')} bias^{o-q(D')} (1-bias)^{n-n'-1-(o-q(D'))}}{\binom{n-n'-1}{o-q(D')-1} bias^{o-q(D')-1} (1-bias)^{n-n'-1-(o-q(D')-1)}} \tag{4.28}$$

We simplify the term by grouping $bias$ and $(1 - bias)$ terms.

$$= \frac{\binom{n-n'-1}{o-q(D')} bias^{o-q(D')-(o-q(D')-1)}}{\binom{n-n'-1}{o-q(D')-1} (1-bias)^{n-n'-1-(o-q(D')-1)-(n-n'-1-(o-q(D')))}} \tag{4.29}$$

$$= \frac{\binom{n-n'-1}{o-q(D')} bias}{\binom{n-n'-1}{o-q(D')-1} (1 - bias)} \tag{4.30}$$

We now also break down the binomial coefficients, rewriting them in their factorial form

$$= \frac{\frac{(n-n')!}{(o-q(D'))!(n-n'-1-(o-q(D'))!)}}{\frac{(n-n')!}{(o-q(D')-1)!(n-n'-1-(o-q(D')-1))!}} \cdot \frac{bias}{1 - bias} \tag{4.31}$$

and continue simplifying the term by breaking down the factorials as much as possible.

$$= \underbrace{\frac{n - n' - o + q(D')}{o - q(D')}}_{=const} \cdot \frac{bias}{(1 - bias)} \tag{4.32}$$

The first term in Equation 4.32 does not rely on the only variable term $bias$. This means, all values are fixed and the result of it is a constant value in regards to further calculations. We denote the term with $const$. With this, we can now insert Equation 4.32 into the original Equation 4.27, thereby simplifying the term.

$$\frac{\partial PL_{M,D',m,a,b,i,bias}(o)}{\partial bias} = \frac{\partial \left| \ln \left(const \cdot \frac{bias}{(1-bias)} \right) \right|}{\partial bias} \tag{4.33}$$

Now we can start calculating the partial derivative via recursive usage of the chain rule.

$$\frac{\partial \left| \ln \left(const \cdot \frac{bias}{(1-bias)} \right) \right|}{\partial bias} = \frac{\partial}{\partial bias} \left(\sqrt{\left(\ln \left(const \cdot \frac{bias}{1 - bias} \right) \right)^2} \right) \tag{4.34}$$

$$= \frac{1}{2} \cdot \frac{1}{\left| \ln\left(const \cdot \frac{bias}{1-bias} \right) \right|} \cdot 2 \ln\left(const \cdot \frac{bias}{1-bias} \right) \cdot \frac{1}{const \cdot \frac{bias}{1-bias}} \cdot \frac{1}{(1-bias)^2} \tag{4.35}$$

$$= \frac{\ln \left(const \cdot \frac{bias}{1-bias} \right)}{\left| \ln \left(const \cdot \frac{bias}{1-bias} \right) \right|} \cdot \frac{1}{const \cdot bias \cdot (1 - bias)} \tag{4.36}$$

We now need to different cases in the partial derivative. Depending on the result of $\ln\left(const \cdot \frac{bias}{1-bias}\right)$ the first fraction of Equation 4.36 either evaluates as 1 or -1. The evaluation depends on if the term results in a positive or negative value, in the negative case resulting in a change of sign for the entire Equation 4.36. As the natural logarithm is monotonically increasing in its input, it suffices to analyse $const \cdot \frac{bias}{1-bias}$ and if it evaluates to a value smaller or bigger than 1.
Case 1: $0 < const \cdot \frac{bias}{1-bias} < 1$

$$\frac{\ln\left(const \cdot \frac{bias}{1-bias}\right)}{\left|\ln\left(const \cdot \frac{bias}{1-bias}\right)\right|} \cdot \frac{1}{const \cdot bias \cdot (1-bias)} = -1 \cdot \frac{1}{const \cdot bias \cdot (1-bias)} \tag{4.37}$$

In case 1, the first derivative of the privacy loss always evaluates to a negative value. For the interval of biases that fulfil this condition, we know the privacy loss to be monotonically decreasing.
Case 2: $const \cdot \frac{bias}{1-bias} > 1$

$$\frac{\ln\left(const \cdot \frac{bias}{1-bias}\right)}{\left|\ln\left(const \cdot \frac{bias}{1-bias}\right)\right|} \cdot \frac{1}{const \cdot bias \cdot (1-bias)} = 1 \cdot \frac{1}{const \cdot bias \cdot (1-bias)} \tag{4.38}$$

In case 2, the first derivative of the privacy loss always evaluates to a positive value. Similarly to case 1, we can infer that the interval of biases that fulfil this condition will increase the privacy loss monotonically.
Lastly, we also need to evaluate which value is the boundary value between the two cases. For this we can solve $const \cdot \frac{bias}{1-bias} = 1$ for the value $bias$.
Case 3: $const \cdot \frac{bias}{1-bias} = 1$

$$const \cdot \frac{bias}{1-bias} = 1 \tag{4.39}$$

$$\frac{n-n'-o+q(D')}{o-q(D')} \cdot \frac{bias}{1-bias} = 1 \tag{4.40}$$

$$(n-n'-o+q(D')) \cdot bias = (o-q(D')) \cdot (1-bias) \tag{4.41}$$

$$(-bias) \cdot (o-q(D')) + bias \cdot (n-n') = (-bias) \cdot (o-q(D')) + o - q(D') \tag{4.42}$$

$$bias \cdot (n-n') = o - q(D') \tag{4.43}$$

$$bias = \frac{o-q(D')}{n-n'} \tag{4.44}$$

We can now rewrite this equation as follows:

$$bias = \frac{o-q(D')}{n-n'} = \frac{q(D)-q(D')}{|D|-|D'|} \overset{Def.\ 3.4}{=} \frac{q(D'')}{|D''|} = bias_{sec} \tag{4.45}$$

The boundary value between the privacy loss monotonically de- and increasing is $bias_{sec}$. For any bias smaller than $bias_{sec}$, the privacy loss is, therefore, monotonically decreasing and, analogously, for any $bias$ bigger than $bias_{sec}$, monotonically increasing.
Stated differently, the Privacy Loss $PL_{M,D',m,a,b,i,bias}(o)$ is monotonically decreasing for biases $bias$ in the interval $[0, bias_{sec})$ and monotonically increasing for biases in the interval $(bias_{sec}, 1]$. Thereby it follows that the privacy loss $PL_{M,D',m,a,b,i,bias}(o)$ monotonically increases in $|bias - bias_{sec}|$ □

$bias_{sec}$ is inherently dependent on $bias$ for fixed $bias_D$, as can be followed from Definition 3.4. Therefore, we can not use the results of Lemma 4.14 unmodified, as we would otherwise test background knowledge approximations against the same approximation and not a fixed point of reference. We will discuss how to fix this problem in detail. For this, we restate the dependency between $bias_{sec}$, $bias$ and $bias_D$ in the following corollary.

Corollary 4.15.
Let D be a database adhering to a distribution with expected value $bias_D$, D' be a background knowledge with expected value bias and D'' the non-compromised data with $bias_{sec}$, then it follows that:

$$bias_D = \frac{|D'|}{|D|} bias + \frac{|D''|}{|D|} bias_{sec} \tag{4.46}$$

Proof (for Corollary 4.15). By Definition 2.2, we know that

$$bias_D = \frac{q(D)}{|D|} \tag{4.47}$$

holds. Furthermore, from Definition 3.4 we know that D' and D'' split both D and $q(D)$ into two disjoint subsets. We can now rewrite the term in Equation 4.47 as follows

$$= \frac{q(D')}{|D|} + \frac{q(D'')}{|D|} \tag{4.48}$$

$$= \frac{|D'| \cdot q(D')}{|D'| \cdot |D|} + \frac{|D''| \cdot q(D'')}{|D''| \cdot |D|} \tag{4.49}$$

$$= \frac{|D'|}{|D|} bias + \frac{|D''|}{|D|} bias_{sec} \tag{4.50}$$

□

Corollary 4.15 restates that $bias_{sec}$ is inherently linked to $bias$. Approximating one of the two values results in the other value also only being approximated. As stated before, we can not use $bias_{sec}$ as the comparative value in Algorithm 1, as we would test the quality of the approximation against the approximation itself.

However, an interesting fact can be learned from this. As we can construct $bias_D$ from $bias$ and $bias_{sec}$, it follows that:

Corollary 4.16.
When Corollary 4.15 holds, then if follows for $bias_D$, $bias$ and $bias_{sec}$ that

$$\min(bias, bias_{sec}) \leq bias_D \leq \max(bias, bias_{sec}) \tag{4.51}$$

Using this corollary, we can now rewrite Lemma 4.14 to the following theorem.

Theorem 4.17.
Let $bias$ be the expected value of the background-knowledge D' and $bias_D$ the expected value of the entire database D.
Assume that all variables besides bias are fixed in $PL_{M,D',m,a,b,i,bias}$. With increasing, absolute distance $|bias - bias_D|$, the privacy loss $PL_{M,D',m,a,b,i,bias}$ also increases.

Proof (for Theorem 4.17).
From Lemma 4.14 we know that the privacy loss $PL_{M,D',m,a,b,i,bias}(o)$ is monotonically increasing in $|(bias - bias_{sec})|$, reaching its minimum at $bias = bias_{sec}$. From Corollary 4.15, we furthermore know $bias_D$ to be the weighted average of $bias$ and $bias_{sec}$. This means for fixed $bias_D$ that both of the values are inherently linked and for the particular case $|bias - bias_{sec}| = 0$ it follows from Corollary 4.15 that:

$$bias = bias_{sec} = bias_D \tag{4.52}$$

In the considered setting we assume $bias_D$ to be fixed. Only $bias$ is variable in the setting, however $bias_{sec}$ will also vary as modifying $bias$ without changing $bias_D$ forces changes in $bias_{sec}$. This, again, follows directly from Corollary 4.15. It follows that increasing the difference $|bias - bias_{sec}|$ will also increase the differences $|bias - bias_D|$ and $|bias_D - bias_{sec}|$. It follows thereby that

$$\text{Lemma } 4.14 \quad \Leftrightarrow \quad \text{Theorem } 4.17 \tag{4.53}$$

$\square$

The combination of the so-far provided proofs gives us another insight on the privacy loss random variable behaviour. Increasing the absolute difference between two values while keeping their weighted average fixed also increases the absolute distance between the values and the average. We can create the following corollary from Theorem 4.17 and Lemma 4.16.

Corollary 4.18. *For any fixed $M, D', m, a, b, i, k, bias_D$ and biases $bias, bias'$ that satisfy the equation*

$$PL_{M,D',m,a,b,i,bias}(o) \leq PL_{M,D',m,a,b,i,bias'}(o) \tag{4.54}$$

the biases $bias$ and $bias'$ also satisfy the equation:

$$|bias - bias_D| \leq |bias' - bias_D| \tag{4.55}$$

Proof (for Corollary 4.18). From Theorem 4.17 we know that for fixed values M, D', m, a, b, i, k and $bias_D$ the privacy loss is monotonically increasing in increasing absolute distance $|bias - bias_D|$.
$bias$ and $bias'$ now satisfy for fixed $M, D', m, a, b, i, k, bias_D$, that the privacy loss under $bias$ is smaller than or equal to the privacy loss under $bias'$ (Equation 4.54). However, as can easily be deduced, this can only be the case if also Equation 4.55 holds. □

With Theorem 4.17 and Corollary 4.18, we now have similar constraints as given by the false conjecture 3.9. With $bias_D$, we can use a value known to us as an optimum for $bias$, and $bias_{sec}$, and show that biases varying from it will result in increasing privacy losses.
The next step is to apply our constructed theorem (Theorem 4.17) and corollary (Corollary 4.18) to show that we can estimate a worst-case noise scale parameter.

4.3.4 Applying Theorem 4.17 and Corollary 4.18

When applying Theorem 4.17 and Corollary 4.18, while applicable to the entire range $[0, 1]$, we only want to look at a comparatively small range around $bias_D$.
As established, the background knowledge we want to approximate is assumed to be chosen randomly from all background knowledge sets. Starkly varying from $bias_D$ could, therefore, leak more privacy than we could estimate when assuming random background knowledge choices.
To bound our support range of biases, we introduce the concept of $(\tau_1, \tau_2), \kappa_2$-representative databases and $(\tau_1, \tau_2), \kappa_2$-closeness.

Definition 4.19 ((τ_1, τ_2)-close bias).
A bias $bias$ is (τ_1, τ_2)-close for some values $\tau_1, \tau_2 > 0$ to another bias $bias'$ if the two biases fulfil the following inequality:

$$bias' - \tau_1 \leq bias \leq bias' + \tau_2 \tag{4.56}$$

Definition 4.20 ($(\tau_1, \tau_2), \kappa_2$-representative database).
A database D' is $(\tau_1, \tau_2), \kappa_2$*-representative* for some values $\tau_1, \tau_2, \kappa_2 > 0$ to another database D if its bias $bias$ is (τ_1, τ_2)-close to the bias $bias_D$ of D with at least $1 - \kappa_2$ probability:

$$\Pr[bias_D - \tau_1 \leq bias \leq bias_D + \tau_2] \geq 1 - \kappa_2 \tag{4.57}$$

In Corollary 4.18 we stated that the privacy loss is inherently smaller for one bias $bias$ compared to $bias'$ if $bias'$ varies more from $bias_D$.
We can now adapt Corollary 4.18 to work with $(\tau_1, \tau_2), \kappa_2$-close databases and, therefore, (τ_1, τ_2)-close biases. We only limit the maximal divergence between $bias$ and $bias_D$, thereby providing two bounding biases for the maximal divergence in each direction from $bias_D$. We formulate the conversion to the new setting in the following lemma.

Lemma 4.21. *If database D' is $(\tau_1, \tau_2), \kappa_2$-representative to database D, then for all biases $bias$ such that $bias_D - \tau_1$ and $bias_D + \tau_2$ are bounding values for $bias$, meaning $bias_D - \tau_1 \leq bias \leq bias_D + \tau_2$, the inequality*

$$PL_{M,D',m,a,b,i,bias}(o) \leq \max(PL_{M,D',m,a,b,i,bias_D - \tau_1}(o), PL_{M,D',m,a,b,i,bias_D + \tau_2}(o)) \tag{4.58}$$

holds.

Proof (for Lemma 4.21). Following Corollary 4.18, we can simplify our proof to only consider the biases used for the respective Privacy Losses. It, therefore, suffices to show that the following equation needs to hold at all times:

$$|bias - bias_D| \leq \max\left(|bias_D - \tau_1 - bias_D|, |bias_D + \tau_2 - bias_D|\right) \tag{4.59}$$

We split the proof into three distinct cases to simplify the analysis. The three cases are with respect to the $bias$ as follows, the $bias$ is lower than $bias_D$, the $bias$ is equal to $bias_D$, and the $bias$ is bigger than $bias_D$.

1. $bias < bias_D$
   If the $bias$ is less than $bias_D$, then we can rewrite the left side of Equation 4.59 as follows:

   $$|bias - bias_D| = bias_D - bias \tag{4.60}$$

   It is implied by D' being $(\tau_1, \tau_2), \kappa_2-$representative to D and the thereby assumed $bias$ bounds that

   $$bias_D - \tau_1 \leq bias \tag{4.61}$$

   From this, we can follow

   $$\begin{aligned} bias_D - bias &\leq bias_D - (bias_D - \tau_1) && (4.62) \\ &\leq \tau_1 && (4.63) \end{aligned}$$

   τ_1 must be greater or equal to 0. This follows directly from Definition 4.19.
   Inserting this into Equation 4.59, we now know that at least one of the two parameters from the max function on the right side of the equation evaluates to a value at least as big as the left side. Our maximum, therefore, also needs to evaluate to this or a bigger value.
   The lemma, therefore, holds for biases smaller than $bias_D$.

2. $bias = bias_D$
   If $bias$ is equal to $bias_D$, the left side of Equation 4.59 evaluates as

   $$|bias - bias_D| = bias_D - bias_D = 0 \tag{4.64}$$

   As both of the inputs to the maximum function of the equations are absolute values and upper- and lower-bound $bias$, we know that both of them must equate to a value bigger or equal to zero.
   The lemma, therefore, holds if $bias$ equals $bias_D$.

3. $bias > bias_D$
   If our bias is bigger than $bias_D$, then we can rewrite the left side of Equation 4.59 as

$$|bias - bias_D| = bias - bias_D \tag{4.65}$$

   Evidently we know from the fact that D' is $(\tau_1, \tau_2), \kappa_2-$representative to D that

$$bias_D + \tau_2 \geq bias \tag{4.66}$$

   From this, we can follow

$$\begin{aligned} bias - bias_D &\leq (bias_D + \tau_2) - bias_D && (4.67) \\ &\leq \tau_2 && (4.68) \end{aligned}$$

   Inserting this result into Equation 4.59, we now know that at least one of the two parameters from the max function on the right side of the equation evaluates to a value at least as big as the left side. Our maximum, therefore, also needs to evaluate to this or a bigger value.
   The lemma, therefore, holds for biases bigger than $bias_D$.

The lemma holds in all distinct cases, and these cases also represent all possible values for $bias$, it follows that Lemma 4.21 holds. □

We have shown that the Privacy Loss can be upper bounded for databases D' that are $(\tau_1, \tau_2), \kappa_2$-representative to some other databases D. Moreover, both $bias_D$ and τ_1, τ_2 can be learned without knowing anything but the size of D'. The only thing remaining is to show, that we upper bound the probability of real background knowledge sets breaking privacy guarantees using estimations created with Lemma 4.21. We need to test if the δ of the approximation is at least as big as the δ produced by the real background knowledge. If that is the case, then we can use Corollary 4.2 to imply, that the calculated noise parameter with the estimated background knowledge will also suffice for the real background knowledge. We proof that the estimation background knowledge produces a worse δ with the following theorem:

Theorem 4.22. *If database D' is $(\tau_1, \tau_2), \kappa_2$-representative to another database D, then for all biases $bias$ such that*
$bias_D - \tau_1 \leq bias \leq bias_D + \tau_2$

$$\begin{aligned}\delta_{real}(\varepsilon) :=& \underset{o \sim M(D)}{\mathbb{E}}[1 - \exp\big(\varepsilon - PL_{M,D',m,a,b,i,bias}(o)\big)] \\ \leq& \max(\underset{o \sim M(D)}{\mathbb{E}}[1 - \exp\big(\varepsilon - PL_{M,D',m,a,b,i,bias_D - \tau_1}(o)\big)], \\ & \underset{o \sim M(D)}{\mathbb{E}}[1 - \exp\big(\varepsilon - PL_{M,D',m,a,b,i,bias_D + \tau_2}(o)\big)]) \\ :=& \delta_{approximated}(\varepsilon) \end{aligned} \tag{4.69}$$

Proof (for Theorem 4.22). The expected value $\mathbb{E}$ describes the arithmetic means of its input. If we want to compare two expected values, $\mathbb{E}(X)$ and $\mathbb{E}(X')$ and we know, that for all values in X and X′ the inequality $X \leq X'$ holds, that the arithmetic means of X will also be smaller than that of X′.
For our inequality in Equation 4.69, we can, therefore, simplify the term to consider the inputs of $\mathbb{E}$ instead of the expected values themselves. We can then check if the inputs fulfil the inequality for all observations o as this would imply the same inequality in the expected values. We, therefore, need to show that the probability of a privacy loss being bigger than ε is higher in one the bounding biases $bias_D - \tau_1$ or $bias_D + \tau_2$ than real background knowledge bias $bias$.

$\forall o$:

$$\begin{aligned} 1 - \exp\big(\varepsilon - PL_{M,D',m,a,b,i,bias}(o)\big) \leq& \max(1 - \exp\big(\varepsilon - PL_{M,D',m,a,b,i,bias_D - (\tau_1/2 + \ell)}(o)\big) \\ & , 1 - \exp\big(\varepsilon - PL_{M,D',m,a,b,i,bias_D + \tau_2/2 + \ell}\big)(o)\big) \end{aligned} \tag{4.70}$$

For simplicity's sake, we substitute the privacy loss $PL_{M,D',m,a,b,i,bias}(o)$ with $x(o)$, privacy loss $PL_{M,D',m,a,b,i,bias_D - (\tau_1/2 + \ell)}(o)$ with $x'(o)$ and privacy loss $PL_{M,D',m,a,b,i,bias_D + \tau_2/2 + \ell})(o)$ with $x''(o)$.
Using the constructed substitutions, we can write the left side of Equation 4.69 as follows:

$\forall o$:

$$1 - \exp\left(\varepsilon - \underbrace{x(o)}_{\leq \max(x'(o), x''(o))}\right) \tag{4.71}$$

With Lemma 4.21 we can upper bound $x(o)$ with the maximum of $x'(o)$ and $x''(o)$. From this follows:

$$1 - \exp\left(\underbrace{\varepsilon - x(o)}_{\geq \max(\varepsilon - x'(o), \varepsilon - x''(o))}\right) \tag{4.72}$$

Similarly, we can now lower bound the term $\varepsilon - x(o)$. We continue on with:

$$1 - \underbrace{\exp(\varepsilon - x(o))}_{\geq \max(\exp(\varepsilon - x'(o)), \exp(\varepsilon - x''(o)))} \tag{4.73}$$

As the exponential function is monotonically increasing in its input, lower bounding its input will also lower bound its output. We can continue the estimations one final time, resulting in:

$$\underbrace{1 - \exp(\varepsilon - x(o))}_{\leq \max(1 - \exp(\varepsilon - x'(o)), 1 - \exp(\varepsilon - x''(o)))} \tag{4.74}$$

We can now write Equation 4.74 as an inequality instead of an estimation. This results in

$$1 - \exp(\varepsilon - x(o)) \leq \max\left(1 - \exp\big(\varepsilon - x'(o)\big), 1 - \exp\big(\varepsilon - x''(o)\big)\right) \tag{4.75}$$

which, if we reverse the substitution for $x(o)$, $x'(o)$ and $x''(o)$, is exactly Equation 4.70. With this we have proven that the inequality holds for all o. $\square$

With this, we introduced all notions necessary to proof Theorem 4.8 and show Algorithm 2 to be $(\Theta, \varepsilon, \delta)$-PPK-DP.

4.4 A full privacy guarantees proof for Algorithm 2

In this section, we are finally able to provide a privacy proof for Algorithm 2. The exact guarantees we need to show have already been specified in Theorem 4.8 earlier in the chapter. We set all discussed and proven notions together, showing that Algorithm 2 suffices $(\Theta, \varepsilon, \kappa_1 + \kappa_2 + \kappa_4 + \sum_{i=0}^{1} \operatorname{cdf}_{Lap(0, 2/\varepsilon)}(-n \cdot \tau_i/2 + 1))$-PPK-DP.

Proof (for Theorem 4.8). We want to prove the correctness and privacy guarantees of Algorithm 2. For this, we split the algorithm into four distinct steps.

1. In the first step, we show that we privately choose an interval for supported biases around $bias_D$ for databases $D \sim \theta$ with privacy guarantees $\varepsilon/2$ and error of at most
$$\kappa_2 + \sum_{i=0}^{1} \operatorname{cdf}_{Lap(0, 2/\varepsilon)}(-n \cdot \tau_i/2 + 1) \tag{4.76}$$
2. Afterwards, in the second step, we show that we choose a background knowledge estimation $Est_{D'}$ such that all worse background knowledge sets $\widetilde{Est}_{D'}$ appear cumulatively with a probability of at most κ_1.
3. In the third step, we then show, assuming the first two steps, that we calculate a noise parameter β such that the probability of privacy leakage is bounded by κ_4 for all considered biases $bias$ and background knowledge sets D'.

Should these three steps hold, we can then use the sequential composition theory for differential privacy to upper-bound the resulting error of consecutively executing all three steps with the following:

$$\begin{aligned}\Pr_{(D',m)\sim B}&\left[\int_{\varepsilon/2}^{\infty} \omega_{M_\beta, D', m, a, b, i, bias}(y)(1 - \exp(\varepsilon/2 - y))\,\mathrm{d}y > \kappa_4\right] \\ &\leq \kappa_1 + \kappa_2 + \sum_{i=0}^{1} \operatorname{cdf}_{Lap(0, 2/\varepsilon)}(-n \cdot \tau_i/2 + 1)\,.\end{aligned} \tag{4.77}$$

4. Lastly, we still need to show that, should we exceed any of our error bounds at any point, we still provide a valid output by over-approximating β.

We start with the first step.

1. We choose $bias_D$ randomly from a Laplace noise distribution centred around the expected value of D and scaled by $2/\varepsilon$. We know such a Laplace mechanism to in-

herently be $\varepsilon/2$-DP[9]. Afterwards, we use the Chernoff-Hoeffdinger theorem for additive error bounds to estimate the interval of supported biases around $bias_D$ with $bias_D - \tau_1$ and $bias_D + \tau_2$.

From Lemma 4.9 and Lemma 4.10 we know the introduced error δ_1 of this to be bounded by

$$\delta_1 \leq \kappa_2 + \sum_{i=0}^{1} \mathrm{cdf}_{Lap(0,2/\varepsilon)}\left(-n\cdot\tau_i/2 + 1\right) \tag{4.78}$$

Step one, therefore, is private and works correctly.

We continue with the second step.

2. We choose $Est_{D'}$ by the following strategy:

    i.) Firstly, estimate for each possible background knowledge $q(\tilde{D})$ in the range $[0, |D'|]$ the privacy loss using $bias_D$.
    Following from Claim 4.6, we know the privacy losses are only influenced by the query result of background knowledge sets and not their permutation of positions.

    ii.) Secondly, order the $q(D')$ descending by their respective privacy loss and calculate the occurrence probability of each $q(\tilde{D})$ using the hypergeometric distribution.
    The hypergeometric distribution describes choosing a specified subset with specified data distribution from another distribution. By definition of the hypergeometric distribution, we know the calculated occurrence probabilities to be correct.

    iii.) Thirdly, exclude worst-case $q(\tilde{D})$ in regards to the privacy loss up to a maximum cumulative occurrence probability of κ_1.
    As we, at most, exclude query classes up to a cumulative probability of κ_1, the thereby introduced error is bounded by it.

    iv.) Lastly, choose the worst remaining $q(\tilde{D})$ in regards to the privacy loss as our approximation of $q(D')$ and choose a random permutation of elements sufficing the chosen query result as $Est_{D'}$.
    We know that $Est_{D'}$ has the worst privacy loss of all $q(\tilde{D})$ that we have not excluded, thereby being a quasi-worst case approximation of D'.

We now consider all background knowledge sets $\tilde{Est}_{D'}$ that are worse than $Est_{D'}$ in regards to their privacy loss. Following directly from the described strategy, we set $Est_{D'}$ to be the quasi-worst-case background knowledge with an allowed error of κ_1. This means, we excluded worse background knowledge estimations up to a cumulative occurrence probability of κ_1 and chose $Est_{D'}$ as the worst remanding background knowledge afterwards. All possible $\tilde{Est}_{D'}$ we could consider must already be excluded by the strategy design.

With that, we can conclude that step two works correctly with probability $1 - \kappa_1$.

We continue with the third step.

3. In the third step, we use the interval bounds $-\tau_1$ and τ_2 as well as the privately chosen $bias_D$ from step 1 and $Est_{D'}$ from step 2 and use them as inputs for Algorithm 1. With the two interval bounds, we then call Algorithm 1 twice as a subroutine, resulting in the two calls $\mathbf{Alg1}(D, Est_{D'}, bias_D - \tau_1, |D'|, {}^{\varepsilon}/_{2}, \kappa_4, err, steps)$ and $\mathbf{Alg1}(D, Est_{D'}, bias_D + \tau_2, |D'|, {}^{\varepsilon}/_{2}, \kappa_4, err, steps)$ that respectively return β_1 and β_2.

   From our privacy proof of Algorithm 1 in Lemma 4.5, we know these two calls to be $(\Theta, \varepsilon, \delta)$-PPK-DP, meaning they each fulfil their respective version of the following two equations. Furthermore, we know with Theorem 4.22 that β_1 and β_2 are lower bounds for necessary noise when defending against attackers in the support range.

   Let $\beta = \max(\beta_1, \beta_2)$ be the worse result of the two outputs.

   The first call now fulfils

$$\mathbb{E}_{(Est_{D'},m)\sim B}\left[\int_{\varepsilon/2}^{\infty} \omega_{M_\beta, Est_{D'}, m, a, b, i, bias_D - \tau_1}(y)(1 - \exp({}^{\varepsilon}/_{2} - y))\, \mathrm{d}y\right] \leq \kappa_4 \tag{4.79}$$

   and the second call fulfils

$$\mathbb{E}_{(Est_{D'},m)\sim B}\left[\int_{\varepsilon/2}^{\infty} \omega_{M_\beta, Est_{D'}, m, a, b, i, bias_D + \tau_2}(y)(1 - \exp({}^{\varepsilon}/_{2} - y))\, \mathrm{d}y\right] \leq \kappa_4. \tag{4.80}$$

   The probability that we get inputs that do not achieve these bounds is, at most, the cumulative errors from step 1 and step 2, as we could have only created erroneous inputs if we landed in one of their respective error cases.
   We can, therefore, write for the first call

$$\begin{aligned}\Pr_{(Est_{D'},m)\sim B}&\left[\int_{\varepsilon/2}^{\infty} \omega_{M_\beta, Est_{D'}, m, a, b, i, bias_D - \tau_1}(y)(1 - \exp({}^{\varepsilon}/_{2} - y))\, \mathrm{d}y > \kappa_4\right] \\ &\leq \kappa_1 + \kappa_2 + \sum_{i=0}^{1} \mathrm{cdf}_{Lap(0, {}^{2}/_{\varepsilon})}\left(-{}^{n \cdot \tau_i}/_{2} + 1\right) \end{aligned} \tag{4.81}$$

and

$$\Pr_{(Est_{D'},m)\sim B}\left[\int_{\varepsilon/2}^{\infty}\omega_{M_\beta,Est_{D'},m,a,b,i,bias_D+\tau_2}(y)(1-\exp(\varepsilon/2-y))\,\mathrm{d}y>\kappa_4\right]$$
$$\leq \kappa_1+\kappa_2+\sum_{i=0}^{1}\operatorname{cdf}_{Lap(0,2/\varepsilon)}\left(-n\cdot\tau_i/2+1\right) \tag{4.82}$$

for the second call. As the probability that the privacy leakage is bigger than κ_4 is upper bounded by the errors of steps 1 and 2, it must conversely follow that the privacy leakage is smaller than κ_4 in all cases that provide valid outputs in step 1 and 2.

We now combine the three steps.
From the correctness of step 1, we know, if we did not exceed error bounds δ_1, that the background knowledge $bias$ is $(\tau_1,\tau_2),\kappa_2$-close to $bias_D$ and that, together with the correctness of step 2, D' must also be $(\tau_1,\tau_2),\kappa_2$-representative to $Est_{D'}$, if we did not exceed error bound κ_1. With Theorem 4.22 and step 3, it then follows that:

$$\Pr_{(D',m)\sim B}\left[\int_{\varepsilon/2}^{\infty}\omega_{M_\beta,D',m,a,b,i,bias}(y)(1-\exp(\varepsilon/2-y))\,\mathrm{d}y>\kappa_4\right]$$
$$\leq \max\Bigg(\Pr_{(Est_{D'},m)\sim B}\left[\int_{\varepsilon/2}^{\infty}\omega_{M_\beta,Est_{D'},m,a,b,i,bias_D-\tau_1}(y)(1-\exp(\varepsilon/2-y))\,\mathrm{d}y>\kappa_4\right],$$
$$\Pr_{(Est_{D'},m)\sim B}\left[\int_{\varepsilon/2}^{\infty}\omega_{M_\beta,Est_{D'},m,a,b,i,bias_D+\tau_2}(y)(1-\exp(\varepsilon/2-y))\,\mathrm{d}y>\kappa_4\right]\Bigg) \tag{4.83}$$
$$\leq \kappa_1+\kappa_2+\sum_{i=0}^{1}\operatorname{cdf}_{Lap(0,2/\varepsilon)}\left(-n\cdot\tau_i/2+1\right) \tag{4.84}$$

We lastly need to check that at all possible times we return valid $(\Theta,\varepsilon,\kappa_1+\kappa_2+\kappa_4+\sum_{i=0}^{1}\operatorname{cdf}_{Lap(0,2/\varepsilon)}\left(-n\cdot\tau_i/2+1\right))$-results.

4. When we do not exceed any error bounds throughout the algorithm, then we return the maximum β from step 3. From Equation 4.84 we know this return value to suffice a given $(\Theta,\varepsilon,\delta)$-PPK-DP bound.

   Beside this return we only have one other return in line (11). We enter this return should we exceed given error bounds after step 1. In this case we return the (ε,δ)-DP noise scale parameter calculated with Pure-DP. With Lemma 4.11 we know this return value to also suffice a given $(\Theta,\varepsilon,\delta)$-PPK-DP bound.

   We can not return at any other point in the algorithm and all possible return values suffice $(\Theta, \varepsilon, \kappa_1 + \kappa_2 + \kappa_4 + \sum_{i=0}^{1} \mathrm{cdf}_{Lap(0,2/\varepsilon)}(-n \cdot \tau_i/2 + 1))$.

With all steps covered, it follows that when our algorithm returns, it does so in the worst case sufficing Equation 4.84. □

With this we finished the privacy proof of Algorithm 2. We have shown that only assuming proven statements, that we can calculate a
$(\Theta, \varepsilon, \kappa_1 + \kappa_2 + \kappa_4 + \sum_{i=0}^{1} \mathrm{cdf}_{Lap(0,2/\varepsilon)}(-n \cdot \tau_i/2 + 1))$-PPK-DP noise scale parameter.
We can now use the proven Theorem 4.8 and state furthermore.

Corollary 4.23.
For the family of distributions Θ, the mechanism M based on Algorithm 2 is $(\Theta, \varepsilon, \delta)$-PPK-DP, as we have:

$$\delta = \kappa_1 + \kappa_2 + \kappa_3 \tag{4.85}$$

$$= \kappa_1 + \kappa_2 + \kappa_4 + \sum_{i=0}^{1} \mathrm{cdf}_{Lap(0,2/\varepsilon)}(-n \cdot \tau_i/2 + 1)) \tag{4.86}$$

We can estimate the complexity of a mechanism M as given in Corollary 4.23 with $O(g \log g) \cdot O(n)$.
The code from line (1) to line (11) can be calculated in constant time. The lines (13) to (30) then need linear time, as we need to estimate privacy losses and occurrence probabilities for $O(|D|)$ background knowledge sets. Only the calls to Algorithm 1 remain.
The limiting factor for its calculation are the PPK-DP terms which we use to evaluate noise scale parameters. The paper [33] showed that the privacy loss distribution can be calculated using so-called privacy buckets in $O(g \log g)$ with g being the granularity of the privacy buckets used to display the distribution. We assume that in our scenario a value of $g = 100,000$ will suffice for our needs.
Setting the run-times together, we land in $O(g \log g) \cdot O(n)$ for one database access with the described mechanism.
With this we finished all privacy proofs and continue to the implementation and evaluation of Algorithm 2.

5 Evaluation

Having shown that outputs from Algorithm 2 suffice PPK-DP privacy constraints, we now want to evaluate the provided utility. In Chapter 3 we mentioned that PPK-DP is a compromise between noiseless privacy and standard (ε, δ)-DP, conjectured to improve utility against the latter. We can quantify utility improvement by assessing the calculated noise scale parameters. Smaller scaled noise will produce less uncertainty, meaning a smaller noise scale will improve utility. Comparing some state-of-the-art (ε, δ)-DP mechanisms and their utility against the results of Algorithm 2 could give us an indication if we actually improve utility.

To determine if we improve utility, we first present an implementation of a mechanism that uses Algorithm 2 for calculating its noise scale parameter. We use this implementation as the basis for our evaluation. We then provide an evaluation sets that we will use in our experiments, as well as the general setup used. Lastly, we evaluate the achieved results and compare them against a mechanism using optimal σ as described in Corollary 3.2 for calculating its noise scale parameter.

5.1 Implementing a mechanism using Algorithm 2

When discussing the implementation of Algorithm 2 we need to consider both the environment we want to use for the implementation, as well as problems that could arise with our choice. For this, we split this section into three parts.

Firstly, we discuss the programming environment we use, why we chose it, and what difficulties we expect by making this choice. Afterwards, we describe the implementation of subroutines used in Algorithm 2 and how we implemented them. Lastly, we discuss how we combine the subroutines as well as code sections not covered up to this point.

Important code sections can be found in the Appendix and will be referenced when applicable. The entire source code can be found in the projects Git repository [37].

Environment

During the execution of Algorithm 2 we need a variety of mathematical functions for probability calculations on data distributions. Especially the background knowledge approximations and privacy loss distribution calculations we make during intermediary steps

need fast and precise calculations. We want to choose an environment that, at best, provides good maintained libraries for these necessary calculations, the possibility to easily visualise results, as well as an easily abstractable syntax.
Under these constraints, we have chosen Python as our programming environment.
While standard Python doesn't provide meaningful support for our requirements, we can easily fix these by using tried and tested third party libraries such as NumPy [15] and SciPy [36] for probability calculations and data handling. Furthermore, we can use Matplotlib [17] as another tried and tested third party library to visualise results.
There are no libraries on the same level as NumPy, SciPy and Matplotlib for privacy loss and differential privacy calculations. However, a comparatively well-researched open-source implementation exists by David Sommer [32]. The implementation was created for his with Sebastian Meiser and Esfandiar Mohammadi co-authored 2018 paper "Privacy Loss classes" [33] and has been used and referenced in further papers since then [27, 21].
With this we fulfil two of our criteria, good maintained libraries and easily visualisations, with the last one being fulfilled by the general high-level form of Python code. It is well-understood that Python allows for easy adaptation into other programming languages [31]. Furthermore, as Python is a script-based programming language, it is also possible to use it as an intermediary step between lower level database queries and final outputs [31].
With this we handled our environment and can continue onward to implementations of used subroutines.

Subroutines

Algorithm 2 uses multiple subroutines for its calculations. Algorithm 1 and its respective subroutine for PPK-DP-calculations in line (31) are the most obvious ones, but don't represent the full set of used subroutines. The additive bounds given by the Chernoff-Hoeffdinger theorem from line (7) to (8), the privacy loss calculations in line (18) and the hypergeometric distribution calls in line (23) should, due to their implementation complexity, be modelled as subroutines as well.
Before discussing the implementation of Algorithm 2, it is favourable to discuss these subroutines first.

Chernoff-Hoeffdinger theorem for additive error bounds

The Chernoff-Hoeffdinger theorem states, that the probability that the expected value over n independent and identically distributed random variables to exceed or fall below some divergence err can be upper bound [16]. In our case, we provide the probability we do

not want to exceed and want to calculate the related err.
While there is a strict 1-to-1 relation between err and the resulting probability, calculating the error out of the probability is not an easy task, due to the calculation terms structure. However, for our purpose it is enough to estimate the error with some slight variance. It is comparatively easy to achieve an estimation using the bisection method, similar to Algorithm 1. We, therefore, use it to estimate err up to some small chosen variance.

Hypergeometric distribution

For the occurrence probabilities of background knowledge sets we use the hypergeometric distribution for calculations. We need to calculate these probabilities for all possible background knowledge sets, meaning the amount of times we execute this term grows in the size of the database.
While one execution with the standard implementation of SciPy is relatively fast, the problem we encounter is that we need to access the distribution values for all possible inputs. With the SciPy implementation this results in the distribution being calculated in its entirety each and every time, which, with growing size, adds up in run-time. In the best-case scenario we would want to make only one costly step, pre-calculating necessary factorials, and only make small updates when switching between different background knowledge set calculations.
For this we need to refactor the term, which we do as follows.

Lemma 5.1.
Line 23 of Algorithm 2 can be rewritten as

$$prb[i] = e^{\ln(n'!)+\ln((n-n')!)+\ln(c!)+\ln((n-c)!)-\ln(n!)-\ln(idx[i]!)-\ln((c-idx[i])!)-\ln((n'-idx[i])!)-\ln((n-n'-(c-idx[i]))!)} \tag{5.1}$$

Proof. Line 23 of Algorithm 2 describes a hypergeometric distribution. We can, therefore refactor the term as follows:

$$
\begin{aligned}
prb[i] &= \frac{\binom{n'}{idx[i]}\binom{n-n'}{c-idx[i]}}{\binom{n}{c}} && (5.2)\\
&= \frac{\frac{n'!}{idx[i]!(n'-idx[i])!} \cdot \frac{(n-n')!}{(c-idx[i])!(n-n'-c+idx[i])!}}{\frac{n!}{c!(n-c)!}} && (5.3)\\
&= \frac{n'!}{idx[i]!(n'-idx[i])!} \cdot \frac{(n-n')!}{(c-idx[i])!(n-n'-c+idx[i])!} \cdot \frac{c!(n-c)!}{n!} && (5.4)\\
&= e^{\ln\left(\frac{n'!}{idx[i]!(n'-idx[i])!} \cdot \frac{(n-n')!}{(c-idx[i])!(n-n'-c+idx[i])!} \cdot \frac{c!(n-c)!}{n!}\right)} && (5.5)\\
&= e^{\ln\left(\frac{n'!}{idx[i]!(n'-idx[i])!}\right)+\ln\left(\frac{(n-n')!}{(c-idx[i])!(n-n'-c+idx[i])!}\right)+\ln\left(\frac{c!(n-c)!}{n!}\right)} && (5.6)\\
&= e^{\ln(n'!)-\ln(idx[i]!(n'-idx[i])!)+\ln((n-n')!)-\ln((c-idx[i])!(n-n'-c+idx[i])!)+\ln(c!(n-c)!)-\ln(n!)} && (5.7)\\
&= e^{\ln\left(n'!\right)+\ln\left((n-n')!\right)+\ln(c!)+\ln((n-c)!)-\ln(n!)-\ln(idx[i]!)-\ln((c-idx[i])!)-\ln\left((n'-idx[i])!\right)-\ln\left((n-n'-(c-idx[i]))!\right)} && (5.8)
\end{aligned}
$$

□

The additive parts $\ln(n'!)$, $\ln((n-n')!)$, $\ln(c!)$, $\ln((n-c)!)$ and $\ln(n!)$ do not depend on the round variable i in the loop surrounding line (23) and can be pre-calculated and stored once before the loop to further improve calculation times. For the necessary updates of the four other parts of the exponential we formulate the following corollary.

Corollary 5.2.
Given two indices i, j so that $idx[i] + 1 = idx[j]$.
The following four equations hold:

$$
\begin{aligned}
\ln(idx[j]!) &= \ln(idx[i]!) + \ln(idx[j]) && (5.9)\\
\ln((c-idx[j])!) &= \ln((c-idx[i])!) - \ln(c-idx[i]) && (5.10)\\
\ln\left((n'-idx[j])!\right) &= \ln\left((n'-idx[i])!\right) - \ln\left(n'-idx[i]\right) && (5.11)\\
\ln\left((n-n'-(c-idx[j]))!\right) &= \ln\left((n-n'-(c-idx[i]))!\right) + \ln\left(n-n'-(c-idx[j])\right) && (5.12)
\end{aligned}
$$

This can be used to further improve calculation times by stepping through an ascending-sorted idx rather than the normal idx, using the four equations as update steps to pre-calculated values.

Proof. We assume that i, j so that $idx[i] + 1 = idx[j]$. We now show the equivalence of Equation 5.9 as an example.

$$\begin{aligned}
\ln(idx[j]!) &= \ln((idx[i] + 1)!) && (5.13)\\
&= \ln(idx[i]! \cdot (idx[i] + 1)) && (5.14)\\
&= \ln(idx[i]!) + \ln(idx[i] + 1) && (5.15)\\
&= \ln(idx[i]!) + \ln(idx[j]) && (5.16)
\end{aligned}$$

The three other equations follow analogously using logarithm rules. □

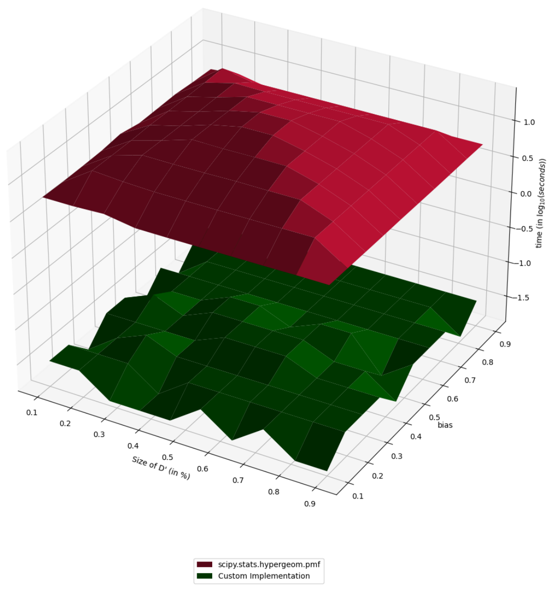
We can use Lemma 5.1 and Corollary 5.2 to calculate the hypergeometric distribution results for a subset choice of $x + 1$ elements when we know the partial natural logarithms for x elements by simply adding 4 terms and then calculating an exponential term. This is assuming all other values are fixed and only the amount of elements chosen changes.
The possible background knowledge sets are fixed in size, similarly is the original database D and it's bias. Algorithm 2 provides exactly the case described in the paragraph above. An example plotting of the resulting improvement in run-time has been plotted in Figure 5.1. Figure 5.1a plots run-times for databases of size 1e4, Figure 5.1b for databases of size 2e4 and Figure 5.1c for databases of size 5e4.
Scipy's costly rebuilding of the underlying factorials in the hypergeometric distribution results in massive run-times even for comparatively small database sizes. With the custom implementation however, we improve these run-times by factors of above 100.

Privacy Losses of possible Background Knowledge

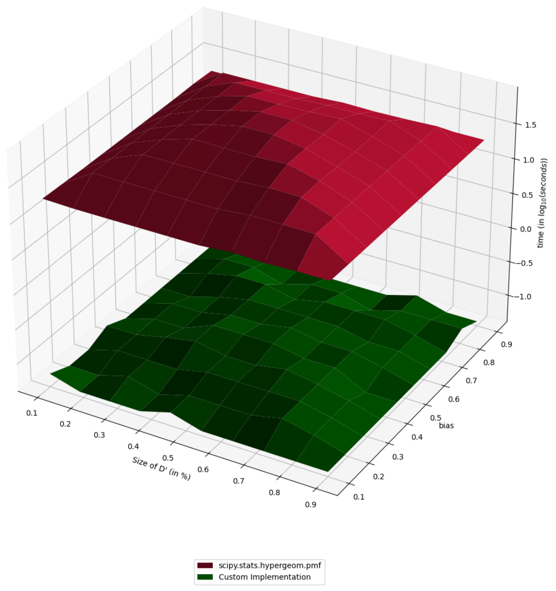
As a measurement to compare background knowledge sets we use the privacy losses of the respective background knowledge. We want to make the privacy loss calculations as simple as possible. We can achieve this by refactoring the privacy loss term as described in the following lemma.

Lemma 5.3.
Line 18 of Algorithm 2 can be rewritten as

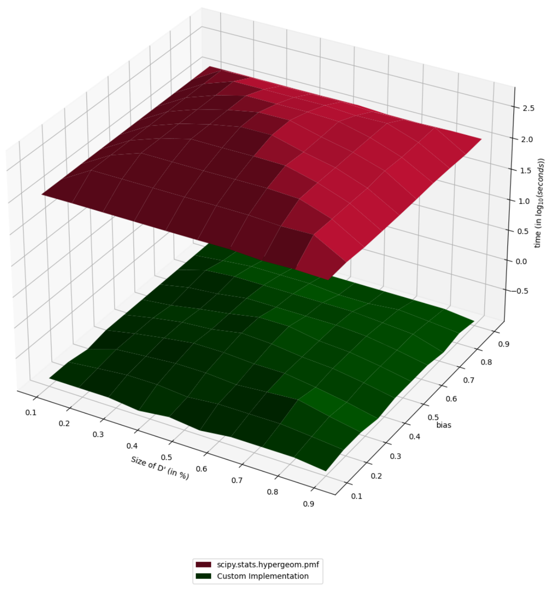
$$losses[i] = \ln(n - (hits - 1)) + \ln(bias_D) - \ln(hits) - \ln(1 - bias_D) \qquad (5.17)$$



(a) Run-time in log_{10} seconds to calculate all occurrence probabilities of background knowledge sets for different background knowledge sizes and bias values for databases with size $|D| = 10000$.



(b) Run-time in log_{10} seconds to calculate all occurrence probabilities of background knowledge sets for different background knowledge sizes and bias values for databases with size $|D| = 20000$.



(c) Run-time in log_{10} seconds to calculate all occurrence probabilities of background knowledge sets for different background knowledge sizes and bias values for databases with size $|D| = 50000$.

Figure 5.1: Run-time disparities between scipy.stats.hypergeom.pmf [36] and the described implementation from Lemma 5.1 with update steps as described in Corollary 5.2.

Proof. We can proof the correctness of our refactored line of code, if we can find a line of equivalency transformations that start with our original line and end with the one provided in the Lemma.

$$losses[i] = \ln\left(\frac{\binom{n'}{hits} \cdot bias_D^{hits} \cdot (1 - bias_D)^{n'-hits}}{\binom{n'}{hits-1} \cdot bias_D^{hits-1} \cdot (1 - bias_D)^{n'-(hits-1)}}\right) \tag{5.18}$$

We can now simplify the fraction inside the natural logarithm by removing exponents as much as possible.

$$= \ln\left(\frac{\binom{n'}{hits} \cdot bias_D}{\binom{n'}{hits-1} \cdot (1 - bias_D)}\right) \tag{5.19}$$

We can now rewrite the binomial coefficient using it's defined fraction form.

$$= \ln\left(\frac{\frac{n'!}{hits! \cdot (n'-hits)!} \cdot bias_D}{\frac{n'!}{(hits-1)! \cdot (n'-(hits-1))!} \cdot (1 - bias_D)}\right) \tag{5.20}$$

We simplify again by removing the common factorial $n'!$

$$= \ln\left(\frac{(hits-1)! \cdot (n'-(hits-1))! \cdot bias_D}{hits! \cdot (n'-hits)! \cdot (1 - bias_D)}\right) \tag{5.21}$$

and simplify the fraction further by also removing parts common in both numerator and denominator.

$$= \ln\left(\frac{(n'-(hits-1)) \cdot bias_D}{hits \cdot (1 - bias_D)}\right) \tag{5.22}$$

Lastly we use logarithm laws to rewrite our current logarithmic fraction as sums of logarithms.

$$= \ln\big((n'-(hits-1)) \cdot bias_D\big) - \ln(hits \cdot (1 - bias_D)) \tag{5.23}$$

$$losses[i] = \ln(n-(hits-1)) + \ln(bias_D) - \ln(hits) - \ln(1 - bias_D) \tag{5.24}$$

□

Algorithm 1

Algorithm 1 is used in the last step of Algorithm 2 and therefore needs to be implemented as a subroutine. As Algorithm 1 uses a bisection method, we can use the pseudo-code from Section 4.1.2 as our basis and get the algorithm running almost instantly.

The only difficulty are the comparisons made over the PPK-DP inequality in lines (5), (10), (20), (23), (25), (28) and (36). For these, we need to discuss how we handle privacy loss distribution calculations in the implementation.

Privacy Loss Distributions

All calculations regarding privacy loss distributions have been made using the open source implementation from David Sommer [32]. By specifying two distributions of neighbouring databases, the resolution of the privacy loss distribution and an allowed error bound, the implementation calculates upper bounds for δ values under a given ε. The implementation needs $O(g\log(g))$ time for its calculations, where g describes the resolution of the privacy loss distribution.
In our case, we provide the distributions of two neighbouring binomial distributions over the non-compromised data subset D'', each convoluted with scaled Gaussian noise. For the convolution we use for small D'' ($|D''| \leq 20000$) *numpy.convolve* [15] and for big databases ($|D''| > 20000$) *scipy.signal.convolve* [36].
Scipy's implementation of the convolution uses Fast-Fourier transformations (FFT), which improve the convolution run-time from $O(|D''|^2)$ to $O(|D''| \cdot \log(|D''|))$. However using FFT results in an underlying noise χ of size $\chi \sim 1e-14$. While normally negligible, for our calculations this results in an extreme over-approximation for noise scale parameters, as they can reach values as low as $1e-20$ and below. For cases where the FFT-introduced noise outweighs our calculations we try and remove it using an approximation of the expected noise scale parameter development.
The source code for Algorithm 1 and the parameters used in the privacy loss distributions can be found in the projects Git repository [37].

The final implementation

The final implementation of Algorithm 2 now combines all mentioned subroutines and combines them as described in the pseudo code provided in Section 4.2.
The source code for the implementation can be found in the projects Git repository [37].

5.2 Evaluation Set

The output of Algorithm 2 is dependent on multiple different factors, namely the size of database D, privacy guarantees ε, maximum allowed error δ, the data distribution of D $bias_D$ and the background knowledge size $|D'|$.
For a thorough analysis of the utility we want to test against all possible influences. Therefore, we need to test for all mentioned parameters.

For size $|D|$ we chose a set of databases from sizes $1e4$ to $1e7$. We believe to cover most realistic sized databases with these choices.
For ε parameters we looked at ranges normally used in academia. We chose values in the range $[1, 0.001]$. During evaluation it proved to be easier to calculate resulting δ parameters for specified noise scale parameters β instead of the other way around. As we, however, split δ during the execution of the algorithm into error bounds $\kappa_1, \kappa_2, \kappa_3$ and κ_4 we thus needed to estimate all applicable errors individually and add them together. We will mention how we approached this in the Experimental Setup 5.3.
For $bias_D$ we chose evenly spaced values from the interval $[0.01, 0.99]$ to get an impression on its influence on results. We similarly chose background knowledge sizes $|D'|$ by specifying the size in relation to $|D|$. We chose background knowledge sizes evenly spaced from relative sizes in interval $[0.01, 0.99]$.

5.3 Experimental Setup

For the experimental setup we first evaluated each subroutine individually. This was due to time constraints and calculation run times exceeding computation capabilities.
We evaluated each subroutine with (ε, δ) guarantees and added them as described in the Algorithm 2 together.
We evaluated each subroutine on similar sets as described in Section 5.2 as to be able to fairly add errors and Epsilon guarantees together.

5.4 Results

We discuss the results achieved ordered by subroutine.

Chernoff Hoeffdinger bounds

We provide visual results for the Chernoff Hoeffdinger bounds in Figure 5.2. We can see, that the ε-bounds we achieve for specified κ_2, so that the Laplace mechanism used to approximate $bias_D$ is supported is inherently linked to database sizes and the bias of the underlying database.

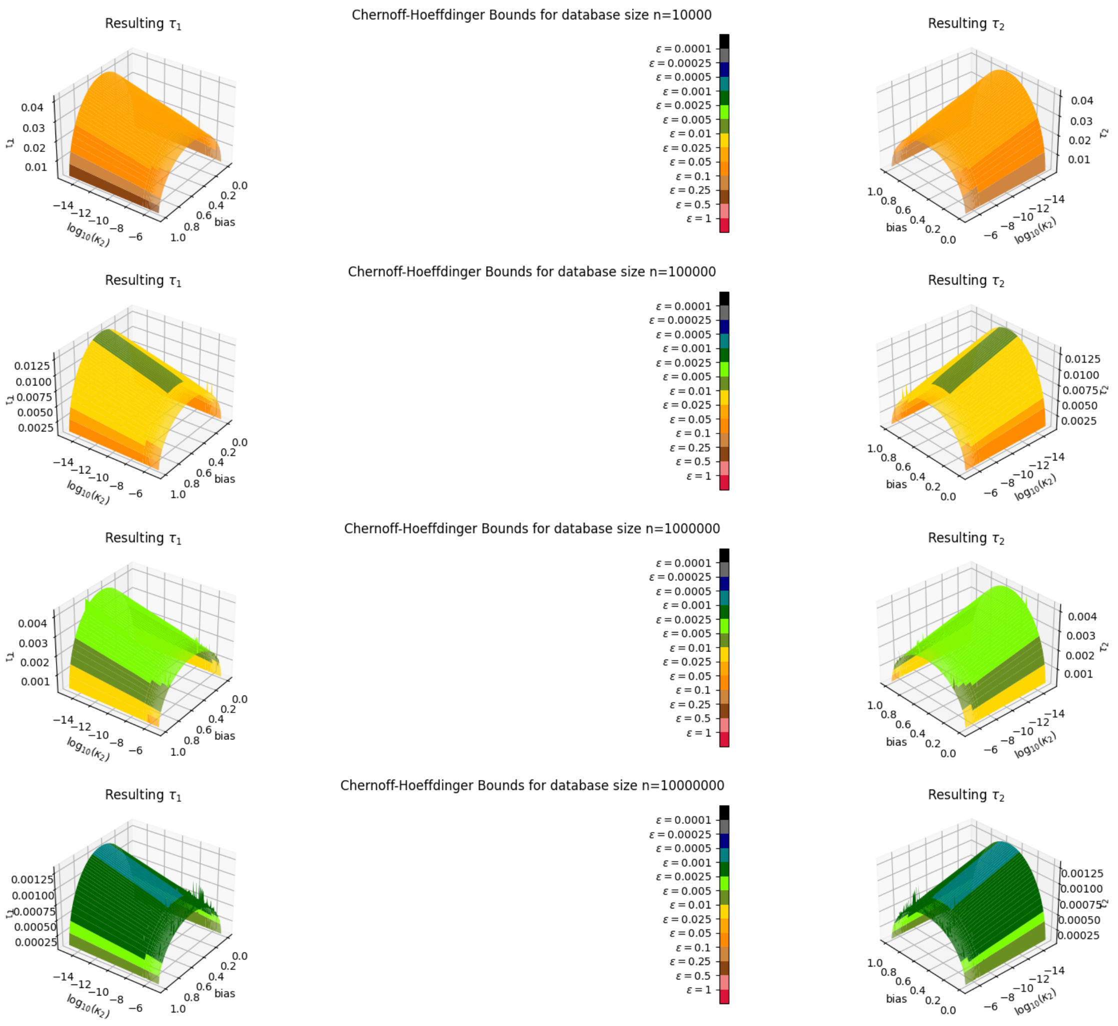
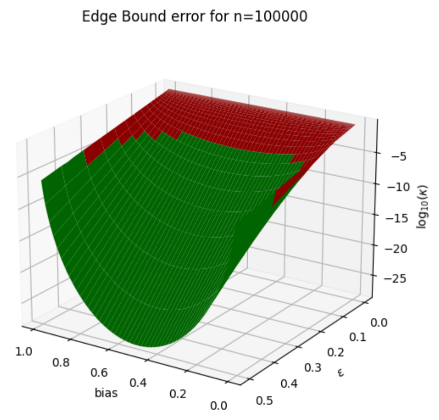


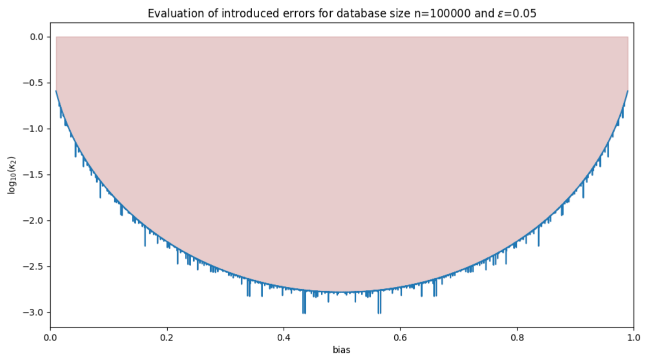
Figure 5.2: Plotting of resulting τ_1, τ_2 for Chernoff Hoeffdinger bounds calculated for a range of different database sizes, biases and sizes of κ_2. The colour depicts the supported ε, such that at least 99% of all values from $\mathrm{Lap}(0, {}^2/_\varepsilon)$ are smaller than $\min_{i\in\{1,2\}}({}^{\tau_i}/_2)$, meaning we have at most a probability of 1% to enter line 11 in Algorithm 2.

Error through Laplace truncation

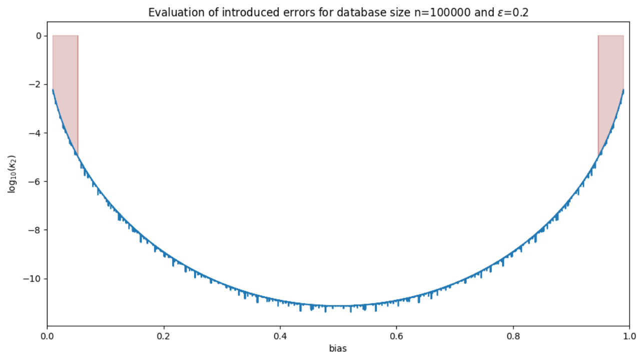
We can observe similar results to the Chernoff bounds in the Laplace truncation error. We achieve worse results for highly biased databases, supporting only comparatively big ε. The exact supported ε for example $n = 100000$ can be seen in Figure 5.3a.



(a) Plot of the δ-domain for the Laplace truncation. δ-values smaller than 1e-5 are coloured green and assumed acceptable, while δ-values bigger than 1e-5 are colored red and assumed to exceed allowed error bounds.



(b) Example plot for worst supported ε for $n = 100000$ as shown in Figure5.2. Using $\varepsilon = 0.05$ would result in the truncation of the Laplace distribution with τ_1 and τ_2 always exceeding error budgets.



(c) Follow-up plot to Figure 5.3b with $\varepsilon = 0.2$ instead of 0.05. With this, the error through truncation is smaller than the allowed error budget for over 90% of cases.

Occurrence Probabilities

Analysing the occurrence probability subroutine showed that the most prevalent influence on the amount of excluded background knowledge sets was the size of the background knowledge itself.

The most background knowledge sets in percent could be excluded for background knowledge sizes around 50% while nearing either of the two extremes would result in the percent dropping. For smaller database sizes rapidly so.

A plotting of some representative results can be found in Figure 5.4.

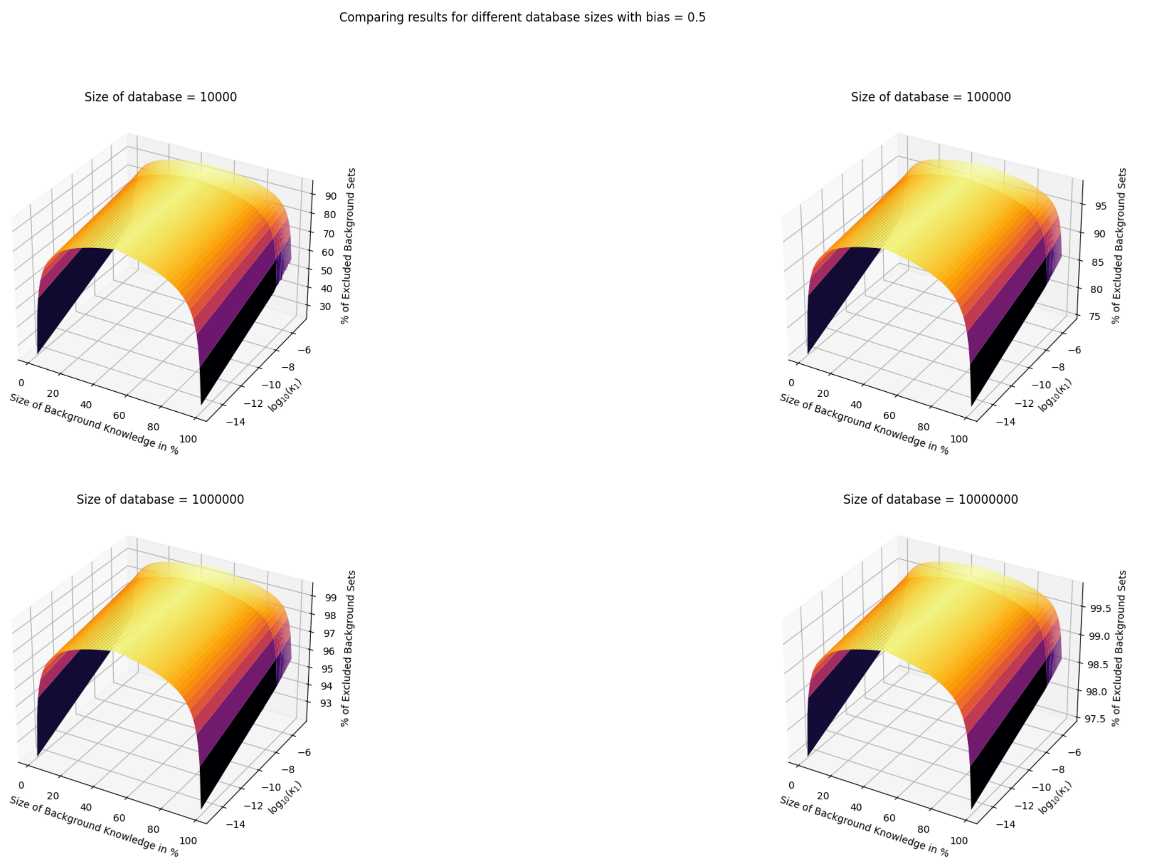


Figure 5.4: Plotting of excluded background knowledge sets up to a cumulative occurrence probability $1 - \kappa_1$. The results were plotted for $n \in [1e4, 1e5, 1e6, 1e7]$. For databases starting at 100000 entries, choosing $\kappa_1 \leq 1e - 14$ can be deemed good enough. For $n = 10000$ we should use choose $\kappa_1 \approx 1e - 6$.

Algorithm 1 and Algorithm 2

Due to time constraints we were unable to finish evaluation, leaving the results for Algorithm 1 sadly out of scope here. Educated guessing would assume that we would see similar dependencies on the bias, background knowledge size and data size.

6 Discussion/Limitations

Having finished our evaluation of results, we use this chapter to discuss possible shortcomings and limitations in our implementation. We first describe limitations due to implementation choices, afterwards through chosen parameters and lastly limitations inherent to the algorithmic structure.

6.1 Limitations

6.1.1 Implementation specific limits

We have several limitations in our implementation. Firstly, we use for the Chernoff-bound calculations a bisection method and introduce further errors when theoretically the bound should be exhaustible, calculating exact τ_1, τ_2 instead of approximations.
Furthermore, edge cases for biases are not handled well by our implementation, resulting in critical errors for $bias_D$ and $bias$ values close to 1 or 0.
Another error exists with Pythons 64-bit floats, as the 64-bit precision can result in errors for certain edge cases, over or under approximating multiple values throughout the execution.
Further complications with calculating convolutions also diminished achieving possible results.

6.1.2 Parametric limits

The worst results we achieved were on small databases with $n < 100$. At that size the errors in estimations take overhand and outweigh most gained improvements from the weakened attacker setting.
Similar breakdowns occur for highly biased distributions, with $bias$ values smaller than 0.1 and bigger than 0.9 resulting at times in unusable results. The best results can be achieved for biases that are close to 0.5, but generally speaking bias values in the interval $[0.25, 0.75]$ seem to work fine.
Furthermore, due to our construction we need to assume binomial distributions for the databases, which provides more background noise than most realistic databases. A generalisation to more realistic data distributions could then give more weight to the achieved results.

6.1.3 Algorithmic limits

Lastly we discuss algorithmic limits. At the beginning of Algorithm 2 we use Laplace noise to privately estimate $bias_D$. While this guarantees us privacy, we introduce massive errors through this into our system. The Laplace truncation that fixes the size of the noise to τ_1 and τ_2 has the biggest impact on the algorithms run.
If we could improve the estimation of $bias_D$, then we could massively improve the guarantees achievable with Algorithm 2.

With this, we discussed the most important to mention limitations on our implementation and results.

7 Conclusion

Finishing up this thesis, we recap the goals we achieved. We set out to finish up the privacy proofs of the two algorithms constructed by Arnold& Pätschke in [1]. For this, we needed to prove a remaining conjecture from the original research work, or disprove it and create similar guarantees through other means. We described this in Research Question 1. We achieved the following:

Research Answer 1. Conjecture 3.9 is false. However, we can construct a similar statement with Theorem 4.17 and Corollary 4.18, giving us similar guarantees on which we can safely base further privacy analysis.

Building on this, we then set out to prove the correctness and privacy guarantees of Algorithm 2. We formulated this as Research Question 2. We were also able to achieve this goal, which we will summarise in the following answer to our question:

Research Answer 2. It is possible to show the correctness of Algorithm 2. We can also provide privacy guarantees, namely that Algorithm 2 is $(\Theta, \varepsilon, \delta)$-PPK-DP for

$$\delta = \kappa_1 + \kappa_2 + \kappa_4 + \sum_{i=0}^{1} \mathrm{cdf}_{Lap(0,{}^{2}\!/_{\varepsilon})}\left(-{}^{n\cdot\tau_i}\!/_{2} + 1)\right) \tag{7.1}$$

With this we achieved the goals concerning formal privacy proofs in their entirety. We showed that PPK-DP algorithms can be constructed and proven to guarantee specified $(\Theta, \varepsilon, \delta)$ bounds.
However, we also wanted to implement the algorithm to not only proof PPK-DP results on paper, but also evaluate the algorithm in a test setting. We were able to correctly implement Algorithm 2 as well as all necessary subroutines. This allows us to evaluate for a noise scale parameter β with a complete input set.
We, therefore, answer our Research Question 3 with the following:

Research Answer 3. We were able to implement Algorithm 2 and are able to calculate noise scale parameters β for a given input set $(D, |D'|, \varepsilon, \delta, \kappa_1, \kappa_2, \kappa_3)$.

Lastly, we set out to compare the results from Algorithm 2 against current state-of-the-art (ε, δ)-DP algorithms. We were able to estimate plausible (ε, δ) guarantees with our implementation, however due to computational limitations we were not able to test a varied enough test set to make definitive statements in regards to the improvement on utility.
We state our results as follows:

Research Answer 4. We were able to calculate singular noise scale parameters β using our implementation. However, due to time constraints we were unable to Compare against over optimal σ 3.2.

We were not able to achieve evaluation results, however the theoretical basis has been proven to hold.

8 Future Work

We believe this work to be a strong start for further research in the field of partial knowledge differential private algorithms. In this chapter, we will summarise some points of interest for further research.

8.1 Improving implementation

The Implementation we provided in Chapter 5 currently only supports databases and noise scale parameters up to certain sizes. Databases with more than one million entries run on average in time scales of multiple hours. This is mostly due to the iterative calculations of the privacy loss distributions under differing noise scale parameters. Especially time consuming is the data distribution convolution with the binomial data distribution and Gaussian noise. We estimate that over 90% of run-time spent can be traced back to these calculations.
If it wouldn't be needed to calculate the entire privacy loss distribution, but only specific subsets of it, then the run-time could be improved. Conjecture 4.12 would provide such an estimation, but we were not able to prove it's correctness in this thesis. However, we are quite certain that the following conjecture in regards to it is true:

Conjecture 8.1. *Corollary 4.18 implies the correctness of Conjecture 4.12.*

It would be of interest to formally prove this conjecture correct to improve run-times.
Another way to improve run-time would be to abstract the convolution which is used to a more efficiently implementable operation. However this would require a deeper understanding of the PPK-DP notion and how to show its guarantees. We still regard this as an interesting point of research.

8.2 Adapt algorithm to support differing distributions

So far, the constructed algorithms only support databases which adhere to binomial distributions. However, as already mentioned in Chapter 4, we can't model most databases using a simple binomial distribution.

If it would be able to abstract the algorithm to more complex data distributions, that would provide more insight if partial knowledge differential privacy can work in realistic settings.
One such example of a more realistic database modelling can be achieved with binomial mixture distributions. Further research could focus on abstracting the constructed algorithms to binomial mixture distributions and check if PPK-DP guarantees can be achieved at all.

8.3 Adapting the algorithm to APK-DP

So far, this research only focused on passive partial knowledge differential privacy and mechanisms that suffice its notion. However, as discussed in the introduction of partial knowledge differential privacy in Section 3.1.2, active partial knowledge differential privacy is just as interesting a research topic.
With first algorithmic constructions for PPK-DP shown to guarantee privacy, further research into the question if we can also construct similar algorithms for APK-DP comes up as an interesting research topic. A first idea which research could focus on is, if the proven PPK-DP algorithm could be modified to also work in APK-DP settings.

References

[1] Paula Arnold, Anna Pätschke under the supervision of Sebastian Berndt, Esfandiar Mohammadi, and David Sommer. Estimating input distributions. Universität zu Lübeck - Institute for IT-Security, 2021.

[2] Borja Balle and Yu-Xiang Wang. Improving the gaussian mechanism for differential privacy: Analytical calibration and optimal denoising, 2018.

[3] Raghav Bhaskar, Abhishek Bhowmick, Vipul Goyal, Srivatsan Laxman, and Abhradeep Thakurta. Noiseless database privacy. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology – ASIACRYPT 2011*, pages 215–232, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[4] Long Cheng, Fang Liu, and Danfeng (Daphne) Yao. Enterprise data breach: causes, challenges, prevention, and future directions. *WIREs Data Mining and Knowledge Discovery*, 7(5):e1211, 2017.

[5] Damien Desfontaines. Research post: Differential privacy under partial knowledge. `https://desfontain.es/privacy/partial-knowledge.html`, 06 2022. Last accessed: 08.12.2022.

[6] Damien Desfontaines, Elisabeth Krahmer, and Esfandiar Mohammadi. Passive and active attackers in noiseless privacy. *CoRR*, abs/1905.00650, 2019.

[7] Josep Domingo-Ferrer, David Sánchez, and Alberto Blanco-Justicia. The limits of differential privacy (and its misuse in data release and machine learning). *Commun. ACM*, 64(7):33–35, jun 2021.

[8] Josep Domingo-Ferrer and Vicenç Torra. A critique of k-anonymity and some of its enhancements. In *2008 Third International Conference on Availability, Reliability and Security*, pages 990–993, 2008.

[9] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3–4):211–407, aug 2014.

[10] Farhad Farokhi. Development and analysis of deterministic privacy-preserving policies using non-stochastic information theory. *IEEE Transactions on Information Forensics and Security*, PP:1–1, 03 2019.

[11] Farhad Farokhi. Noiseless privacy. *CoRR*, abs/1910.13027, 2019.

[12] Natasha Fernandes, Annabelle McIver, and Carroll Morgan. The laplace mechanism has optimal utility for differential privacy over continuous queries. In *2021 36th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–12, 2021.

[13] Simone Fischer-Hübner and Amon Ott. From a formal privacy model to its implementation. In *21st National Information Systems Security Conference*, 1998.

[14] Arik Friedman and Assaf Schuster. Data mining with differential privacy. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 493–502, 2010.

[15] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

[16] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

[17] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[18] 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 *2022 IEEE Symposium on Security and Privacy (SP)*, pages 473–488, 2022.

[19] Noah Johnson, Joseph P. Near, and Dawn Song. Towards practical differential privacy for sql queries. *Proc. VLDB Endow.*, 11(5):526–539, oct 2018.

[20] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography*. Chapman and Hall/CRC Press, 2007.

[21] Antti Koskela, Joonas Jälkö, and Antti Honkela. Computing tight differential privacy guarantees using fft. In Silvia Chiappa and Roberto Calandra, editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 2560–2569. PMLR, 26–28 Aug 2020.

[22] Balachander Krishnamurthy, Konstantin Naryshkin, and Craig Wills. Privacy leakage vs. protection measures: the growing disconnect. *IEEE*, 05 2012.

[23] Ninghui Li, Wahbeh Qardaji, and Dong Su. On sampling, anonymization, and differential privacy or, k-anonymization meets differential privacy. In *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security*, pages 32–33, 2012.

[24] Ashwin Machanavajjhala, Xi He, and Michael Hay. Differential privacy in the wild: A tutorial on current practices & open challenges. In *SIGMOD '17*, SIGMOD '17, page 1727–1730, New York, NY, USA, 2017. Association for Computing Machinery.

[25] Ryan McKenna, Gerome Miklau, Michael Hay, and Ashwin Machanavajjhala. HDMM: optimizing error of high-dimensional statistical queries under differential privacy. *CoRR*, abs/2106.12118, 2021.

[26] Balázs Pejó and Damien Desfontaines. *Background Knowledge (B)*, pages 37–42. Springer International Publishing, Cham, 2022.

[27] Balazs Pejo and Damien Desfontaines. *Quantification of Privacy Loss (Q)*, pages 13–18. Springer, 01 2022.

[28] Harrison Quick. Generating poisson-distributed differentially private synthetic data. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 184(3):1093–1108, 2021.

[29] Pierangela Samarati and Sabrina Capitani de Vimercati. Access control: Policies, models, and mechanisms. In Riccardo Focardi and Roberto Gorrieri, editors, *Foundations of Security Analysis and Design*, pages 137–196, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[30] Pierangela Samarati and Latanya Sweeney. Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression. In *Technical Report SRI-CSL-98-04*. Computer Science Laboratory, SRI International, 1998.

[31] Nischal Shrestha, Titus Barik, and Chris Parnin. It's like python but: Towards supporting transfer of programming language knowledge. In *2018 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pages 177–185, 2018.

[32] David Sommer. Privacy buckets. `https://github.com/sommerda/privacybuckets`, 2020. Last accessed: 08.12.2022.

[33] David Sommer, Sebastian Meiser, and Esfandiar Mohammadi. Privacy loss classes: The central limit theorem in differential privacy. Cryptology ePrint Archive, Paper 2018/820, 2018. `https://eprint.iacr.org/2018/820`.

[34] Latanya Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05):557–570, 2002.

[35] Shun Takagi, Yang Cao, and Masatoshi Yoshikawa. Asymmetric differential privacy. *CoRR*, abs/2103.00996, 2021.

[36] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[37] Jannik Westenfeld. Algorithmic implementation. `https://git.its.uni-luebeck.de/theses/bachelor-jannik_westenfeld-ppk_dp`, 2022. Last accessed: 08.12.2022.

[38] Ying Zhao and Jinjun Chen. A survey on differential privacy for unstructured data content. *ACM Comput. Surv.*, 54(10s), sep 2022.