Towards Privacy-Preserving and Regulation-Compliant Data Analysis

Lun Wang

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2022-174
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2022/EECS-2022-174.html

July 14, 2022

Towards Privacy-Preserving and Regulation-Compliant Data Analysis

by

Lun Wang

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Xiaodong Dawn Song, Chair
Associate Professor Raluca Ada Popa
Assistant Professor Jacob Noah Steinhardt

Summer 2022

Towards Privacy-Preserving and Regulation-Compliant Data Analysis

Abstract

Towards Privacy-Preserving and Regulation-Compliant Data Analysis

by

Lun Wang

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Xiaodong Dawn Song, Chair

Data analysis has become an indispensable part of today's society and is greatly impacting our everyday life. At the same time, a variety of privacy attacks are threatening data sovereignty and safety in each step of data analysis from data collection to result release. Preservation of data privacy has been studied through various lens, and can be roughly classified into two categories: the top-down approach (*e.g.* General Data Protection Regulation) attempts to design a set of universal rules to regulate sensitive data, and the bottom-up approach (*e.g.* differential privacy) targets concrete privacy challenges and solves them from an algorithmic perspective.

These two approaches, although have achieved great success separately, suffer from their intrinsic defects as well. Specifically, 1) the effective enforcement of the top-down regulations and 2) the design of the bottom-up algorithms for various applications with different trade-offs have been critical problems to solve. Fortunately, these two approaches are complementary and can become more powerful once used together. The top-down approach can be used for guidance when designing bottom-up solutions and the bottom-up methods can be leveraged to enforce the top-down regulations.

In this dissertation, the researcher presents an end-to-end framework, namely `Aegis`. `Aegis` comprises two main components, a sub-system verifying the compliance between a privacy regulation and a data analysis task, and a library of standardized privacy-preserving algorithms to implement the data analysis tasks. These two components respectively address challenges 1) and 2) mentioned above to some extent. Furthermore, by gluing the two approaches, `Aegis` magnifies their advantages and promotes a new privacy-preserving data analysis paradigm.

To my family and friends

Contents

List of Figures

List of Tables

Acknowledgments

I am really grateful to have met so many sweet people in my PhD journey. First, I would like to thank my advisor Dawn Song. I first met Dawn when I was a junior undergraduate visiting her lab. We worked together on a differential privacy project, which piqued my long-term interest in security and privacy. Dawn's support, encouragement and guidance are the most important factors that help me finish my PhD and become a well-established researcher. I can never learn enough from her profound knowledge, farsighted vision and noble work ethics.

Next, I would like to thank Professor Raluca Popa and Professor Jakob Steinhardt for serving on both my dissertation committee and qualifying exam committee, and thank Professor Joseph Gonzalez for serving on my qualifying exam committee. The dissertation would not be as good as it is without your insightful suggestions.

I am particularly grateful to have had the chance to work closely with Joseph Near in my early career. Joe is literally the first group member I met in person and has been my role model ever since. His wisdom, diligence, humbleness and sincere love of research deeply influenced me and my research path.

I also want to give a big shout-out to Kallista Bonawitz, Adria Gascon, Peter Kairouz, Jakub Konkey, Ziyu Liu, Stefano Mazzocchi, Om Thakkar, Ahbradeep Guha Thakurta and Wennan Zhu for their meticulous help during my internship at Google. I learned a lot from each of them from research to life. Fortunately, I will have the chance to work with most of them again in the near future and I cannot be more excited to start my next journey.

I am also lucky enough to have received heartwarming support from many others. I would like to thank all my collaborators, including Sylvain Bellemare, Min Du, Peng Gao, Wenbo Guo, Justin Harris, Roger Iyengar, Ruoxi Jia, Ari Juels, Sai Praneeth Karimireddy, Usmann Khan, Evgenios Kornaropoulos, Kim Laine, Yunqi Li, Xiaoyuan Liu, Andrew Low, Krishna Deepak Maram, Andrew Miller, Qi Pang, Iosif Pinelis, Robert Sim, Ni Trieu, Xian Wu, Xinyu Xing, Fan Zhang, Yupeng Zhang, Banghua Zhu and many others. I would also like to thank all my friends and colleagues at Berkeley if not mentioned above, including Xinyun Chen, Silvery Fu, Gonzalo Munilla Garrido, Narek Galstyan, Jaewan Hong, Yuncong Hu, Zhuohan Li, Jian Liu, Chenguang Wang, Guanhua Wang, Zhanghao Wu, Tiancheng Xie, Gai Yu and Jiaheng Zhang. I also want to thank all my friends outside Berkeley, which might be too long a list to fit here but thanks for all the wonderful time we have shared together.

Finally, I would like to thank my parents for their unconditional love and support.

Chapter 1

Introduction

Data analysis has become an indispensable part of today's society and is greatly impacting our everyday life. However, a variety of privacy attacks are threatening data sovereignty and safety in every step of data analysis from data collection to result release. Preservation of data privacy has been studied through various lens, but can be roughly classified into two categories: 1) the top-down approach (*e.g.* General Data Protection Regulation) attempts to design a set of universal rules to regulate sensitive data usage; 2) the bottom-up approach (*e.g.* differential privacy) targets concrete privacy issues and solves them from an algorithmic perspective. In this chapter, we introduce the two approaches respectively and discuss how the combination of the two can potentially change the landscape of privacy-preserving data analysis.

1.1 The Top-down Approach: A Regulator's View

Since the introduction of GDPR, more and more regional privacy regulations have been legislated to normalize user data usage. According to a report by the information technology & innovation foundation [145], 34 states in the United States have passed or introduced 72 privacy bills regulating the commercial collection and use of personal data. An incomplete list of today's privacy regulations includes the Health Insurance Portability and Accountability Act (*abbrev.* HIPAA, US), Family Educational Rights and Privacy Act (*abbrev.* FERPA), the Children's Online Privacy Protection Act (*abbrev.* COPPA, US), Gramm-Leach-Bliley Act (*abbrev.* GLBA, US), General Data Protection Regulation (*abbrev.* GDPR, EU), California Consumer Privacy Act (*abbrev.* CCPA, US), California Privacy Rights Act (*abbrev.* CPRA, US), Massachusetts Data Privacy Law (US), New York Privacy Act (US), Hawaii Consumer Privacy Protection Act (US), Maryland Online Consumer Protection Act (US), Digital Privacy Act (Canada), Proposal Data Protection Law (Chile, drafting), Federal Data Privacy Law (US, not yet drafting), General Data Protection Law (Brazil), ePrivacy Regulation (EU, drafting), Personal Data Protection Bill 2018 (India), Personal Information Security Specification (China), Privacy Act 1988 and amendments (Australia).

These regulations share common principles. As the pioneer of privacy regulations, GDPR proposes seven key principles of personal data control:

- *Lawfulness, fairness and transparency*: Whenever processing personal data, you shall have legitimate reason, be open about how the data is processed, and correctly handle the data without withholding any information.
- *Purpose limitation*: You shall only collect data for "specified, explicit, and legitimate purposes".
- *Data minimisation*: You shall only collect the necessary amount of data to complete your purposes.
- *Accuracy*: You are responsible for ensuring the accuracy of the collected data.
- *Storage limitation*: You shall justify data retention periods when collecting data.
- *Integrity and confidentiality (security)*: You shall maintain the integrity and confidentiality of the collected data.
- *Accountability*: You shall appropriately enforce the privacy regulations and keep a clear record of your enforcement process in case you need to prove responsibility to authorities.

Other privacy regulations share similar principles. For example, CCPA has three principles: *transparency, accountability, and control*; CPRA codifies the *purpose limitation* principle to augment CCPA.

Despite the best intentions and well-designed principles, these regulations lack concrete operational guidance. As a result, today's compliance paradigm heavily relies on human auditing, and is problematic in two aspects. First, it is an expensive process to hire and train data protection personnel and rely on manual effort to monitor compliance. The information technology & innovation foundation [145] has estimated that California's privacy law will cost $78 billion annually, with California's economy bearing $ 46 billion and the rest of the U.S. economy bearing the other $32 billion; California small businesses will bear $9 billion of in-state costs, while out-of-state small businesses face $6 billion of costs. These estimates highlight the overwhelming costs for states to create a patchwork of privacy laws and the need for an efficient privacy enforcement framework to alleviate the economic burden and minimize the impact on innovation. Second, human auditing is slow and error-prone. The inefficiency of compliance impedes the effective use of data and hinders productivity. Errors made by compliance officers can harm data subjects and result in legal liability. What makes things worse is that these regulations differ a lot in specific terms despite their similar principles and are still evolving rapidly to accommodate the ever-changing real-world requirements. This makes developing a fixed solution once and for all infeasible and we need a versatile solution to catch up the fast development of these privacy regulations.

All these factors create unprecedented needs for a *cost-efficient*, *versatile*, *verifiable*, and *secure* framework to enforce privacy regulations.

- *Cost-Efficiency*: The framework should be automatic or semi-automatic with modest human efforts to alleviate the running cost.
- *Versatility*: The framework should be applicable to most current regulations today and can accommodate possible future development.
- *Verifiability*: The enforcement process should be recorded and can be checked in the future.
- *Security*: The framework should correctly enforce the privacy regulation under reasonable threat models.

1.2 The Bottom-up Approach: A Programmer's View

On the other end of the spectrum, a variety of privacy-preserving techniques, especially customized defenses [150, 60, 164, 171], have been developed to address concrete privacy attacks such as data poisoning attacks [24, 152], model inversion attacks [50], membership inference attacks [128]. Among these techniques, differential privacy [39] is one of the most widely used ones due to its mathematical rigorousness and negligible computation overhead. Over the past decade, differential privacy has seen unprecedented prosperity and has been deployed by large service providers and public organizations including including Google [46], Apple [86], Microsoft [36], LinkedIn [77] and the US Census Bureau [135].

Differential privacy has derived many variants in the past few years including pan privacy [40], concentrated differential privacy [42], zero-concentrated differential privacy [21], Rényi differential privacy [101], f-differential privacy [38], and Gaussian differential privacy [38]. Albeit the variety of definitions, they share a similar intuition.

> *Whether or not an individual's record is in the input should not greatly influence the output.*

This intuition is initially captured in the vanilla definition of differential privacy.

Definition 1 (Approximate Differential Privacy)**.** *A randomized mechanism $\mathcal{M}$ is said to follow (ϵ, δ)-approximate differential privacy if for two neighboring datasets $\mathcal{D}$ and $\mathcal{D}'$ and arbitrary subset of $\mathcal{M}$'s output space $\mathcal{S}$,*

$$\mathbb{P}[\mathcal{M}(\mathcal{D}) \in \mathcal{S}] \leq e^{\epsilon}\mathbb{P}[\mathcal{M}(\mathcal{D}') \in \mathcal{S}] + \delta \tag{1.1}$$

If $\delta = 0$, we omit the δ part and refer to it as ϵ-differential privacy (also known as pure differential privacy).

Let's interpret the above definition step by step. First, the common definition of *neighboring datasets* are datasets differing in one record, capturing "*whether or not an individual's record is in the input*" in the intuition. Technically, the neighboring relation can be any symmetric relation. In practice, the definition also differs from case to case. We use the notion $\mathcal{D} \sim \mathcal{D}'$ to represent the neighboring relation. Second, Equation 1.1 captures "*not greatly influencing the output*" in the intuition. ϵ and δ are privacy parameters measuring the difference of the output. Specifically, ϵ bounds the multiplicative difference while δ bounds the probability of failure of the chosen ϵ. The complete interpretation of Equation 1.1 is out of scope and we refer interested readers to Dwork and Roth's well-celebrated privacy book [41].

Another important concept in differential privacy is the *sensitivity* of a deterministic mechanism $\mathcal{M}$. Intuitively, sensitivity upper bounds how much the output will change when someone's record is added to or removed from the input dataset. The most common way to achieve differential privacy is to add noise to the output and sensitivity serves as a gauge to calibrate the noise scale. There are many different sensitivity definitions, out of which the most commonly used one is ℓ_k-sensitivity.

Definition 2 (ℓ_k-sensitivity)**.** *The ℓ_k-sensitivity of a deterministic mechanism $\mathcal{M}$ is:*

$$\sup_{\mathcal{D} \sim \mathcal{D}'} \|\mathcal{M}(\mathcal{D}) - \mathcal{M}(\mathcal{D}')\|_k$$

A nice property of differential privacy is composability [21, 1, 101, 6, 158, 38, 131, 80, 172]. If several differentially private mechanisms are applied adaptively to the same dataset, then the concatenation of their outputs still satisfies differential privacy. The composed privacy parameters can be derived from each mechanism's privacy parameter.

Albeit all the advantages, most real-world deployments of differential privacy are still at the experimental stage due to two major reasons. First, most differentially private mechanisms add noise to the output so the accuracy of the output typically drops. Whenever designing a differentially private mechanism, the predominant question is always how to balance the trade-off between privacy and accuracy. Although we already have got such differentially private algorithms for several important applications such as convex optimization [70, 11, 14, 12, 23], deep learning [1, 112, 73], there are still many applications without such customized design and trivial usage of universal differentially private mechanism usually results in unacceptable accuracy. Second, the correct execution of differential privacy requires support from upstream and downstream infrastructures. For example, the privacy budget should be accurately tracked and data with an exhausted budget should be archived or discarded. However, we lack such a system to track and execute actions triggered by differential privacy in complex data processing pipelines.

1.3 1+1>2: Synergy between the Two Approaches

Despite the great success achieved separately, the two approaches suffer from their intrinsic defects. Specifically, how to enforce the privacy regulations or design and deploy differentially private algorithms remains critical problems yet to solve. Fortunately, we observe that the two approaches are complementary and can be much more effective if used together in an organized way. Privacy regulations can be used to guide the design of privacy-preserving algorithms and the corresponding ecosystems. In return, these algorithms and infrastructures can serve as powerful tools to enforce the privacy regulations.

In this dissertation, the researcher's persents an end-to-end framework, namely `Aegis`, to glue the two approaches and remake the privacy enforcement paradigm. `Aegis` comprises two main components: 1) a sub-system verifying the compliance between a privacy regulation and a data analysis task; 2) a library of standardized differentially private algorithms used to implement the data analysis tasks. The architecture of `Aegis` is shown in Figure 1.1. The

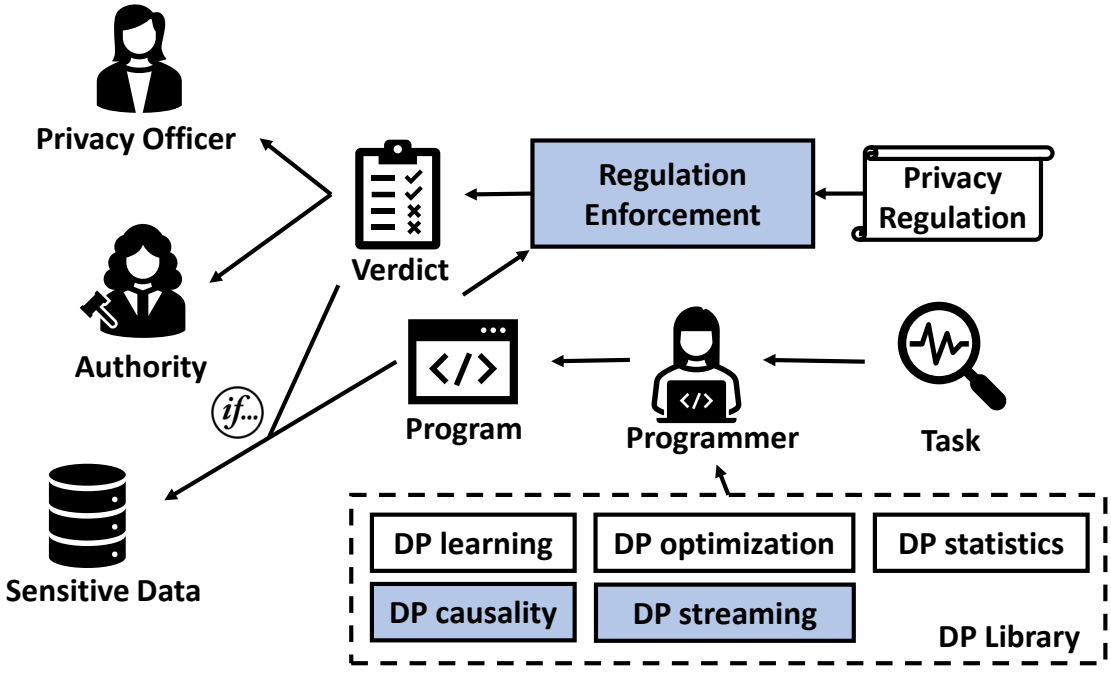


Figure 1.1: `Aegis` Architecture. The blue components are covered in this dissertation.

workflow of `Aegis` is described as follows. First, the programmer receives a data analysis task and leverages the differential privacy library to code up a program. The program, together with the privacy regulation, is fed into the regulation enforcement module which outputs a verdict including 1) a decision whether the program satisfies the regulation; 2) a system log of how the data is processed. If the decision is positive, then the program is allowed to

run on the sensitive data. The system log will be periodically audited by the privacy officer within the organization and can prove correct compliance to authorities.

As easy the workflow might look like, the correct execution hinges on two key properties. First, the regulation enforcement framework should be cost-efficient, versatile, verifiable and secure. To achieve these properties, we design a static analyzer [153] which can verify whether an analysis program satisfies a given privacy regulation. To encode a privacy regulation in a machine-readable format, we extend a policy-encoding language namely LEGALEASE [125]. The combination of these two components significantly reduces the demand for human auditing and thus greatly lowers the cost. Both the static analyzer and LEGALEASE are extensible so they can easily accommodate new changes in privacy regulations. Besides, we run the system with encrypted data and trusted execution environments for security. The system is introduced in detail in Chapter 2. Second, the differential privacy library should embrace as many applications as possible. Today most differentially private algorithms fall in the category of learning and optimization. We identify two under-explored areas: causal inference [154] and streaming [156], and develop customized differentially private algorithms for them. The algorithms are introduced in Chapter 3 and 4. Besides, the regulation enforcement module also tracks the consumed privacy budget to support correct applications of these algorithms.

In `Aegis`, we see the potential to synergize the strengths of different parties, including authorities, programmers, legal experts and privacy experts, to offer operational effectiveness in a privacy-preserving data analysis pipeline. Programmers can productively analyze sensitive data with verifiable privacy guarantees and shorter auditing time. Privacy officers and authorities can also efficiently regulate data processing without looking at the cumbersome and case-to-case details. The legislators can receive timely feedback from the system to guide future legislation and amendments. The privacy researchers can also get a better sense of the real-world demand and use it as guidance to design more practical differentially private algorithms.

Part I

Top-down Approach: Effective Enforcement of Privacy Regulations

Chapter 2

PrivGuard: Privacy Regulation Compliance Made Easier

As discussed in Chapter 1, continuous compliance with privacy regulations, such as GDPR and CCPA, has become a costly burden for companies from small-sized start-ups to business giants. The culprit is the heavy reliance on human auditing in today's compliance process, which is expensive, slow, and error-prone. To address the issue, the researchers propose PrivGuard, a novel system design that reduces human participation required and improves the productivity of the compliance process. PrivGuard is mainly comprised of two components: (1) PrivAnalyzer, a static analyzer based on abstract interpretation for partly enforcing privacy regulations, and (2) a set of components providing strong security protection on the data throughout its life cycle. To validate the effectiveness of this approach, the researchers prototype PrivGuard and integrate it into an industrial-level data governance platform. The case studies and evaluation show that PrivGuard can correctly enforce the encoded privacy policies on real-world programs with reasonable performance overhead.

2.1 Introduction

With the advent of privacy regulations such as the EU's General Data Protection Regulation (GDPR) and California Consumer Privacy Act (CCPA), unprecedented emphasis is put on the protection of user data. This is a positive development for data subjects, but presents major challenges for compliance. Today's compliance paradigm relies heavily on human auditing, and is problematic in two aspects. First, it is an expensive process to hire and train data protection personnel and rely on manual effort to monitor compliance. According to a report from Forbes [144], GDPR cost US Fortune 500 companies $7.8 Billion as of May 25th, 2018. Another report from DataGrail [143] shows that 74% of small- or mid-sized organizations spent more than $100,000 to prepare for continuous compliance with GDPR and CCPA. Second, human auditing is slow and error-prone. The inefficiency of compliance

impedes the effective use of data and hinders productivity. Errors made by compliance officers can harm data subjects and result in legal liability.

An ideal solution would enable data curators to easily ensure fine-grained compliance with minimal human participation and quickly adapt to new changes in privacy regulations. A significant amount of academic work seeks to address this challenge [111, 121, 28, 125, 94, 93, 147, 5, 133, 151]. The European ICT PrimeLife project proposes to encode regulations using Primelife Policy Language (PPL) [147] and enforce them by matching the policies with user-specified privacy preferences and triggering actions when detecting specific behaviors. A-PPL [5] extends the PPL language by adding accountability rules. These two pioneering works play important roles in the exploration of efficient policy compliance. However, as they focus on Web 2.0 applications, they provide limited support for fine-grained privacy requirement compliance in complex data analysis tasks. The SPECIAL project [133] partly inherits the design of the PPL project and, as a result, suffers from similar limitations. The closest to our work is by Sen at al. [125], which proposed a formal language (LEGALEASE) for privacy policies and a system (GROK) to enforce them. However, GROK uses heuristics to help decide whether the analysis process is compliant with a policy and thus human auditing is required to catch false-negatives. Thus, effective compliance with privacy regulations at scale remains an important challenge.

PrivGuard: Facilitating Compliance. This chapter describes a principled data analysis framework called PRIVGUARD to facilitate compliance with privacy regulations with minimal human participation. PRIVGUARD works in a five-step pipeline.

First, data protection officers (DPOs), legal experts, and domain experts collaboratively translate privacy regulations into a machine-readable policy language. The translation process is application-specific and requires domain-specific knowledge in both the application and the privacy regulation (*e.g.* mapping legal concepts to concrete fields.). The encoded policy is referred to as the *base policy*. Encoding the base policy is the step with the most human effort in PRIVGUARD's workflow.

Second, before the data is collected, the data subjects are aided by a client-side API to specify their *privacy preferences*. They can either directly accept the base policy or add their own privacy requirements. The privacy preferences are collected together with the data by the data curator.

Third, data analysts submit programs to analyze the collected data. Analysts are required to submit a corresponding *guard policy*, which is no weaker than the base policy, along with their program. Only data whose associated policy is no stronger than the guard policy can be used by the program.

Fourth, our proposed static analyzer, PRIVANALYZER, examines the analysis program to confirm its compliance with the guard policy. At the same time, the subset of the data whose privacy preferences are no stronger than the guard policy will be loaded to conduct the real analysis.

Finally, depending on the output of PRIVANALYZER, the result will be either declassified

to the analyst or guarded by the remaining unsatisfied privacy requirements (called a *residual policy*). The whole pipeline is under the close protection of cryptographic tools and trusted execution environments (TEEs) to ensure security and correctness of the execution.

Extension of Legalease: Encoding Policies. PrivGuard is designed to be compatible with many machine-readable policy languages such as [147, 5]. In this work, we instantiate our implementation with Legalease, a formal policy language from [125], because of its readability and extensibility. We extend Legalease [125] by providing new attribute types, including attributes requiring the use of privacy-enhancing technologies like differential privacy.

PrivAnalyzer: Enforcing Policies. The core component of PrivGuard is PrivAnalyzer, a static analyzer capable of checking the compliance of an analysis program with a privacy policy. PrivAnalyzer performs static analysis of the programs submitted by analysts to check their compliance with the corresponding guard policies.

In contrast to previous approaches relying on access control [160] or manual verification [125, 53, 133], PrivAnalyzer is a novel policy enforcement mechanism based on abstract interpretation [108]. PrivAnalyzer does not rely on heuristics to infer policy or program semantics, and provides provable soundness for some properties. PrivAnalyzer examines only the program and the policy (not the data), so the use of PrivAnalyzer does not reveal the content of the data it protects. Our approach works for general-purpose programming languages, including those with complex control flow, loops, and imperative features. Thus, PrivAnalyzer is able to analyze programs as written by analysts—so analysts do not need to learn a new programming language or change their workflows. We instantiate our implementation with Python, one of the most widely used programming languages for data analysis.

We implemented PrivAnalyzer in about 1400 lines of Python and integrated it in an industrial-level data governance platform to prototype PrivGuard. We evaluated the prototype experimentally on 23 open-source Python programs that perform data analytics and machine learning tasks. The selected programs leverage popular libraries like Pandas, PySpark, TensorFlow, PyTorch, Scikit-learn, and more. Our results demonstrate that PrivGuard is scalable and capable of analyzing unmodified Python programs, including programs that make extensive use of external libraries.

2.2 PrivGuard Overview

In this section, we outline the design and implementation of PrivGuard. We first walk through PrivGuard using a toy example and then introduce the system design and implementation. As last, the threat model and the security of PrivGuard are discussed.

A Toy Example

We use a toy example to demonstrate the workflow of PrivGuard, which also allows us to present the main components used. A company launches a mobile application and collects user data to help make informed business decisions. To facilitate compliance with privacy regulations, the company deploys PrivGuard to protect the collected data.

First, the DPOs, legal experts, and domain experts encode two requirements in the base policy: (1) minors' data shall not be used in any analysis; (2) any statistics on the data shall be protected using differential privacy.

Second, the privacy preferences are collected from the data subjects together with the data. Some data subjects (Group 1) trust the company and directly accept the base policy. Some (Group 2) are more cautious and want their zip codes to be redacted before analysis. The others (Group 3) do not trust the company and do not want their data to be used for purposes except for legitimate interest.

Third, a data analyst wants to survey the user age distribution. It specifies a guard policy, that besides the base policy, zip codes shall not be used in the analysis either. The analyst submits a program calculating the user age histogram to PrivGuard. She remembers to filter out all the minor information and redact the zip code field but forgets to protect the program with differential privacy.

Fourth, PrivGuard uses PrivAnalyzer to examine the privacy preferences and loads data of Group 1 and 2 into the TEE as their privacy preferences are no stricter than the guard policy. PrivGuard runs the program and saves the resulting histogram. However, after examining the program and the guard policy, PrivGuard finds that the program fails to protect the histogram with differential privacy. Thus, the histogram is encrypted, dumped to the storage layer and guarded by a *residual policy* indicating that differential privacy should be applied before the result can be declassified.

Lastly, PrivGuard outputs the *residual policy* to the analyst. The analyst, after checking the residual policy, submits a program which adds noise to the histogram to satisfy differential privacy. PrivGuard then decrypts the histogram, loads it into TEE, and executes the program to add noise to it. This time, PrivGuard finds that all the requirements in the guard policy are satisfied, so it declassifies the histogram to the analyst.

System Design

Base Policy Encoding. Encoding the base policy is the step with the most human participation in PrivGuard's workflow. The base policy should be designed collaboratively by DPOs, legal experts, and domain experts to accurately reflect the minimum requirements of the privacy regulation. Note that only one base policy is needed for each data collection and can be reused throughout all the analyses on the data. The purpose of the base policy is two-fold. First, the text version of the base policy is to be presented to data subjects as the minimum privacy standard before they opt in the data collection. Second, the data analysts need to understand the base policy before conducting analysis. If their analysis satisfies a

```
ALLOW ROLE
    Physician
ALLOW
    SCHEMA
    HealthInformation
  AND FILTER
      age < 90
  AND REDACT
      zip
```

(a) General encoding.

```
ALLOW ROLE
    Physician
ALLOW
    SCHEMA
    SerumCholestoral
  AND FILTER
      age < 90
```

(b) Concrete encoding.

Figure 2.1: Encoding of several HIPAA requirements.

stricter privacy standard, they can specify their own guard policy to take advantage of more user data.

We demonstrate the encoding process using a subset of the HIPAA safe harbor method[1] (Figure 2.1). The DPOs and legal experts first encode the regulation in a general way without considering concrete data format. As shown in Figure 2.1a, the first clause (line 1-2) allows the patient's physician to check his or her data. The second clause (lines 3-7) represents some safe harbor requirements: health information may be released if the subject is under 90 years old and the zip codes are removed. Then the DPOs and domain experts map the encoding to a concrete data collection by introducing real schemas and removing unnecessary requirements. For example, in Figure 2.1b, HealthInformation is replaced with a concrete column name in the dataset, SerumCholestoral, and the last requirement is removed as the dataset does not contain zip codes. For longer example encodings of GPDR, HIPPA, CCPA and FERPA, please refer to Appendix B.

Data & Privacy Preference Collection. Besides the base policy, the data subjects can also specify additional privacy preferences to exercise their rights to restrict processing. These privacy preferences are sent to the data curator along with the data, where they will be kept together in the storage layer. To defend against attacks during transmission and storage, the data is encrypted before sent to the data curator. The decryption key is delegated to a key manager for future decryption (Section 2.2).

A natural question is "how much expertise is needed to specify privacy preferences in LEGALEASE?" Sen et al. [125] conducted a survey targeting DPOs and found that the majority were able to correctly code policies after training. To complement their survey and better understand how much expertise is needed, we conducted an online survey targeting general users without training. The survey reveals two interesting facts: (1) there is a signif-

[1]Recent research has shown that the approach prescribed in HIPAA does not really protect the privacy of individuals. In the future, we expect that many data subjects will add a PRIVACY attribute requiring the use of a provable privacy technology like differential privacy.

icant positive correlation between the difficulty of understanding and encoding LEGALEASE policies and the user's familiarity with other programming languages; (2) most users cannot correctly understand privacy techniques such as differential privacy without training. According to these observations, we strongly recommend the users without programming experience directly accept the base policy instead of encoding their own. Although out of scope, we deem it important future direction to simplify privacy preference specification by developing more user-friendly UI and translation tools. The details of the survey are deferred to Section 2.3.

Analysis Initialization. To initialize an analysis task, the analyst needs to submit (1) the analysis program, and (2) a guard policy, to PRIVANALYZER. A guard policy should be no weaker than the base policy to satisfy the minimum privacy requirements. The stricter the guard policy is, the more data can be used for analysis.

PrivAnalyzer Analysis. After receiving the submission, PRIVANALYZER will load the privacy preferences from the storage layer and compare them with the guard policy. Only the data with preferences no stricter than the guard policy will be loaded into the TEE, decrypted using keys from the key manager, and merged to prepare for the real analysis. Meanwhile, PRIVANALYZER will (1) check that the guard policy is no weaker than the base policy and (2) then examine the guard policy and the program to generate the residual policy. To make sure the static analysis runs correctly, PRIVANALYZER is protected inside a trusted execution environment (TEE).

The compliance enforcement actually hinges on three checks: (1) the guard policy is no weaker than the base policy; (2) only data with privacy preferences no stronger than the guard policy is used; (3) the guard policy should be satisfied before declassification. For (3), the guard policy can be satisfied either by a single program or by multiple programs applied sequentially on the data. This design endows PRIVGUARD with the ability to enforce privacy policies in multi-step analyses.

Execution & Declassification. After PRIVANALYZER finishes its analysis, the submitted program will be executed with the decrypted data inside the TEE. If the residual policy generated in the previous step is empty, then the result can be declassified to the analyst. Otherwise, the output will be encrypted again and stored in the storage layer protected by the residual policy.

Attentive readers might ask "why does PRIVGUARD not directly reject programs that fail to comply with the guard policy?" The design choice is motivated by two considerations. First, it is not always possible to get an empty residual policy when the guard policy contains ROLE or PURPOSE attributes. These attributes will be satisfied by human auditing after the real data analysis. Second, PRIVGUARD is designed to be compatible with multi-step analysis, a common case in real-world product pipelines. In multi-step analysis, it is likely that privacy requirements are satisfied in different steps.

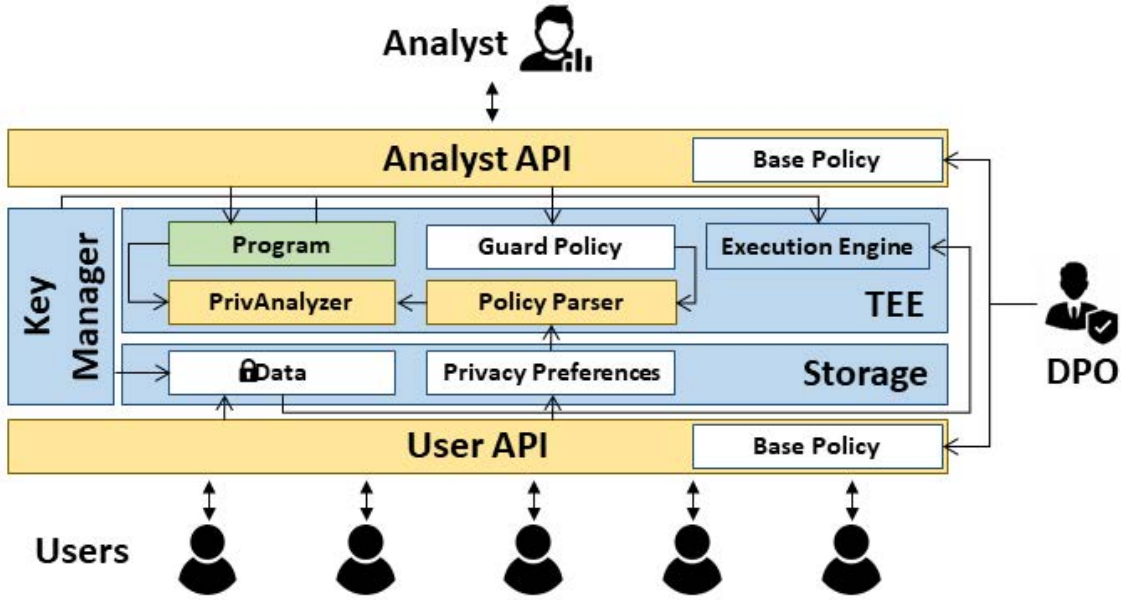


Figure 2.2: PrivGuard prototype infrastructure. White: data/policies; green: analysis programs; blue: off-the-shelf components; yellow: newly-designed components.

System Security

In this section, we present the threat model and demonstrate how to secure PrivGuard under the threat model.

Threat Model. Our setting involves four parties - (1) data subjects (*e.g.* users), (2) a data curator (*e.g.* web service providers, banks, or hospitals), (3) data analysts (*e.g.* employees of the data curator), and (4) untrusted third parties (*e.g.* external attackers). Data is collected from data subjects, managed by the data curator, and analyzed by data analysts. Both the data subjects and the data curator would like to comply with privacy regulations to either protect their own data or avoid legal or reputational risk. The data analysts, however, are honest but reckless, and might unintentionally write programs that violate privacy regulations. The only way that a data analyst can interact with the data is to submit analysis programs and check the output. The untrusted third parties might actively launch attacks to steal the data or interfere with the compliance process. A concrete example is the cloud provider which hosts a small company's service or data. We protect data confidentiality and execution integrity from third parties under the following two assumptions. First, we assume that the untrusted third parties cannot submit analysis programs to PrivAnalyzer or compromise insiders to do so. Second, we assume that the untrusted third parties fit in the threat model of the chosen TEE so that they cannot break the security guarantee of the

TEE.

Security Measure. PrivGuard takes the following measures to defend against untrusted third parties and establish a secure workflow under the above threat model. First, data is encrypted locally by the data subjects before transmitted to the data curator. The decryption key is delegated to the key manager so no one can touch the data intentionally or carelessly without asking the key manager or the data subject for the decryption key. To bind data and policy in an immutable way, the encrypted data contains a hash value of the corresponding policy. Second, all transmission channels satisfy transport layer security standards (TLS 1.3). Third, PrivAnalyzer is run inside a TEE to guarantee the integrity of the static analysis. The key manager can attest remotely to confirm that PrivAnalyzer correctly examines the program and the policies before issuing the decryption key. Fourth, data decryption and analysis program execution are protected inside the TEE as well.

Security of PrivGuard against untrusted third-parties is based on the following sources of trust. First, confidentiality and integrity of data are preserved inside TEE and TLS channels. Second, the integrity of code execution is preserved inside the TEE. Remote attestation can correctly and securely report the execution output. Third, the key manager is completely trusted such that the confidentiality of decryption keys is preserved. The design of trusted key managers is orthogonal to the focus of the section. Potential solutions include a key manager inside TEE or a decentralized key manager [95].

2.3 PrivAnalyzer: Static Analysis for Enforcing Privacy Policies

This section describes PrivAnalyzer, a static analyzer for enforcing the privacy policies tracked by PrivGuard. We first introduce the design challenges of such a static analyzer. Then we review the Legalease policy language [125], which we use to encode policies formally, then describe how to statically enforce them with both an empirical example and the formal model. We conclude this section with a usability survey of the extended Legalease language.

Background & Design Challenges

Legalease is one example of a growing body of work that has explored formal languages for encoding privacy policies [121, 28, 125, 94, 93, 151, 148, 27, 83, 53]. A complete discussion of related work appears in Section 2.5. We adopt Legalease to express PrivGuard policies due to its expressive power, formal semantics, and extensibility.

Sen et al. [125] developed a system called Grok that combines static and dynamic analyses to enforce Legalease policies. Grok constructs a data dependency graph which encodes *all* flows of sensitive data, then applies a set of inference rules to check that each

$$
\begin{array}{rcll}
attr & \in & \textit{ROLE}, \textit{SCHEMA}, \textit{PRIVACY}, \textit{FILTER}, & \\
 & & \textit{REDACT}, \textit{PURPOSE} & \\
A & \in & \text{attribute} & ::= \quad attr\ attrValue \\
C & \in & \text{policy clause} & ::= \quad A \mid A\ \textbf{AND}\ C \mid A\ \textbf{OR}\ C \\
P & \in & \text{policy} & ::= \quad (\textbf{ALLOW}\ C)^{+}
\end{array}
$$

Figure 2.3: Policy language surface syntax

node in the graph satisfies the policy. GROK combines analysis of system logs with limited static analysis to construct the graph.

The GROK approach presents two challenges. First, the approach is a heuristic: it examines syntactic properties of the program and individual executions of the program (via system logs), and thus may miss policy violations due to implicit flows [103, 170, 116, 54]. Second, the GROK approach requires making the entire dataflow graph explicit; in large systems with many data flows, constructing this graph may be intractable.

PRIVANALYZER is designed as an alternative to address both challenges. It uses static analysis based on abstract interpretation instead of GROK's heuristic analysis and avoids making the dataflow graph explicit by constructing composable *residual policies*.

Policy Syntax & Semantics

PRIVANALYZER enforces privacy policies specified in LEGALEASE [125], a framework for expressing policies using *attributes* of various types. Attributes are organized in *concept lattices* [49], which provide a partial order on attribute values. We express policies according to the grammar in Figure 2.3 (a slightly different syntax from that of LEGALEASE). A policy consists of a top-level **ALLOW** keyword followed by *clauses* separated by **AND** (for conjunction) and **OR** (for disjunction). For example, the following simple policy specifies that doctors or researchers may examine analysis results, as long as the records of minors are not used in the analysis:

ALLOW (*ROLE* Doctor **OR** *ROLE* Researcher)
AND *FILTER* age >= 18

Sen et al. [125] define the formal semantics of LEGALEASE policies using a set of inference rules and the partial ordering given by each attribute's concept lattice. We take the same approach, but use a new attribute framework based on *abstract domains* [108] instead of concept lattices. Our approach enables PRIVPOLICY to encode policies with far more expressive requirements, like row-based access control and the use of privacy-enhancing technologies as described below.

Attributes. Attributes are the basic building blocks in LEGALEASE. Sen et al. [125] describe a set of useful attributes. We extend this set with two new ones: *FILTER* encodes

row-based access control requirements, and *PRIVACY* requires the use of privacy-enhancing technologies.

Role. The *ROLE* attribute controls *who* may examine the contents of the data. Roles are organized into partially ordered hierarchies (See Appendix A). A particular individual may have many roles, and a particular role specification may represent many individuals. For example, the *doctor* role may represent doctors with many different specialties. The following policy says that only individuals with the role Oncologist may examine the data it covers:

ALLOW *ROLE* Oncologist

Schema. The *SCHEMA* attribute controls *which columns* of the data may be examined. For example, the following policy allows oncologists to examine the *age* and *condition* columns, but no others:

ALLOW *ROLE* Oncologist
AND *SCHEMA* age, condition

Privacy. The *PRIVACY* attribute controls *how* the data may be used, by requiring the use of privacy-enhancing technologies. As a representative sample of the spectrum of available mechanisms, our implementation supports the following (with easy additions): (1) De-identification (or pseudonymization); (2) Aggregation; (3) k-Anonymity [137]; (4) ℓ-diversity [92]; (5) t-closeness [87]; (6) Differential privacy [39, 41]. For example, the following policy allows oncologists to examine the *age* and *condition* columns under the protection of differential privacy with certain privacy budget:

ALLOW *ROLE* Oncologist
AND *SCHEMA* age, condition
AND *PRIVACY* DP (1.0, 1e-5)

Filter. The *FILTER* attribute allows policies to specify that certain data items must be excluded from the analysis. For example, the following policy says that oncologists may examine the age and condition of individuals over the age of 18 with differential privacy:

ALLOW *ROLE* Oncologist
AND *SCHEMA* age, condition
AND *PRIVACY* DP (1.0, 1e-5)
AND *FILTER* age $>$ 18

Redact. The *REDACT* attribute allows policies to require the partial or complete redaction of information in a column. For example, the following policy requires analysis to redact the last 3 digits of ZIP codes (e.g. by replacing them with stars). The (2 :) notation is taken from Python's slice and indicates the substring between the third character and end of the string.

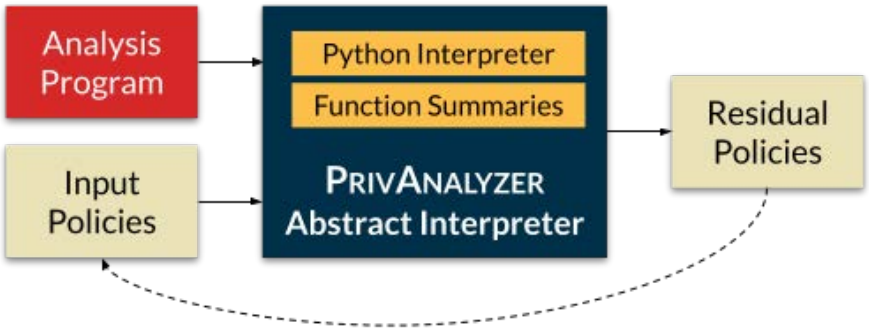


Figure 2.4: PrivAnalyzer Overview. PrivAnalyzer inputs an analysis program and policies, and produces residual policies; it can be applied repeatedly (dashed line) for multi-step analyses.

ALLOW *ROLE* Oncologist
AND *SCHEMA* age, condition
AND *PRIVACY* DP (1.0, 1e-5)
AND *FILTER* age $>$ 18
AND *REDACT* zip (2 :)

Purpose. The *PURPOSE* attribute allows the policy to restrict the purposes for which data may be analyzed. For example, the following policy allows the use of age and medical condition for public interest purposes with all the above requirements:

ALLOW *ROLE* Oncologist
AND *SCHEMA* age, condition
AND *PRIVACY* DP (1.0, 1e-5)
AND *FILTER* age $>$ 18
AND *REDACT* zip (2 :)
AND *PURPOSE* PublicInterest

PrivAnalyzer Overview

PrivAnalyzer performs its static analysis via *abstract interpretation* [108], a general framework for sound analysis of programs. Abstract interpretation works by running the program on *abstract values* instead of concrete (regular) values. Abstract values are organized into *abstract domains*: partially ordered sets of abstract values which can represent all possible concrete values in the programming language. An abstract value usually represents a specific *property* shared by the concrete values it represents. In PrivAnalyzer, abstract values are based on the abstract domains described earlier.

Our approach to static analysis is based on a novel instantiation of the abstract interpretation framework, in which we encode *policies as abstract values*. The approach is summarized

in Figure 2.4. The use of abstract interpretation allows us to construct the static analysis systematically, ensuring its correspondence with the intended semantics of attribute values.

Analyzing Python Programs. The typical approach to abstract interpretation is to build an *abstract interpreter* that computes with abstract values. For a complex, general-purpose language like Python, this approach requires significant engineering work. Rather than building an abstract interpreter from scratch, we *re-use the standard Python interpreter* to perform abstract interpretation. We embed abstract values with attached privacy policies as Python objects and define operations over abstract values as methods on these objects.

For example, the Pandas library defines operations on concrete dataframes; PrivAnalyzer defines the AbsDataFrame class for *abstract* dataframes. The AbsDataFrame class has the same interface as the Pandas DataFrame class, but its methods are redefined to compute on abstract values with attached policies. We call the redefined method a *function summary*, since it summarizes the policy-relevant behavior of the original method. For example, the Pandas indexing function __getitem__ is used for filtering, so PrivAnalyzer's function summary for this function removes satisfied *FILTER* attributes from the policy.

```
def __getitem__(self, key):
    ......
    If isinstance(key, AbsIndicatorSeries):
        # 'runFilter' removes satisfied FILTER attributes
        newPolicy = self.policy.runFilter(...)
        return Dataframe(..., newPolicy)
    ......
```

Multi-step Analyses & Residual Policies. As shown in Figure 2.4, the output of PrivAnalyzer is a *residual policy*. A residual policy is a new policy for the program's concrete output—it contains the requirements *not yet satisfied* by the analysis program. For a multi-step analysis, each step of the analysis can be fed into PrivAnalyzer as a separate analysis program, and the residual policies in the previous step become the input policies for the next step. PrivAnalyzer is *compositional*: if multiple analyses are merged together into a single analysis program, then the final residual policy PrivAnalyzer returns for the multi-step analysis will be at least as restrictive as the one for the single-step version. The use of residual policies in PrivGuard enables compositional analyses without requiring explicit construction of a global dataflow graph, addressing the challenge of Grok [125] mentioned earlier.

Handling libraries. Scaling to large programs is a major challenge for many static analyses, including abstract interpreters. Libraries often present the biggest challenge, since they tend to be large and complex; it may be impossible to analyze even a fairly small target program if the program depends on a large library. This is certainly true in our setting (Python programs for data processing), where programs typically leverage large libraries like pandas (327,007 lines of code), scikit-learn (184,144 lines of code), PyTorch (981,753

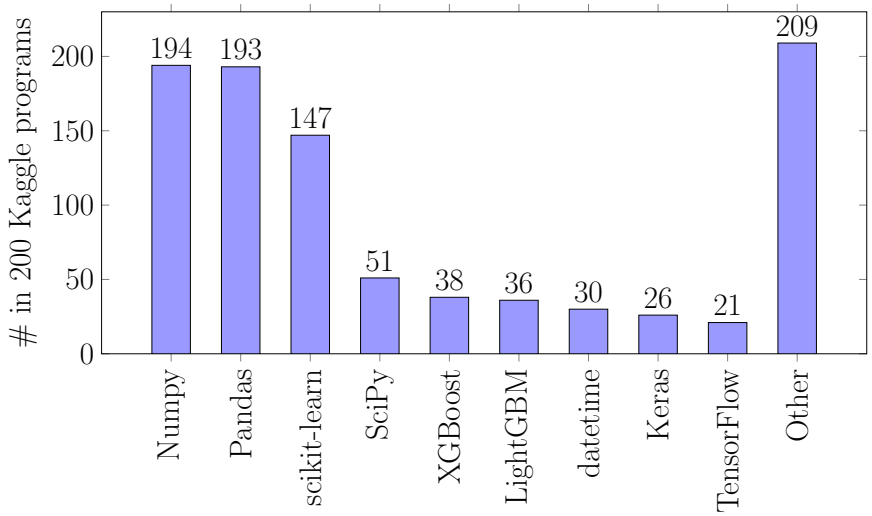


Figure 2.5: Python library frequency statistics. We summarized the top frequently used libraries.

lines of code) and Tensorflow (2,490,119 lines of code). Worse, many libraries are written in a mix of languages (e.g. Python and C/C++) for performance reasons, so analysis for each one of these languages would be needed.

Our solution is to develop *specifications* of the abstract functionality of these libraries, like the AbsDataFrame example shown earlier, in the form of function summaries. We use the function summaries during analysis instead of the concrete implementation of the library itself. This approach allows PrivGuard to enforce policies even for analysis programs that leverage extremely large libraries written in multiple languages.

Our approach for handling libraries requires a domain expert with knowledge of the library to implement its specification. In our experience, the data science community has largely agreed upon a core set of important libraries which are commonly used (e.g. NumPy, pandas, scikit-learn, etc.), so providing specifications for a small number of libraries is sufficient to handle most programs. To validate the conjecture empirically, we randomly selected 200 programs from the Kaggle platform and counted the libraries they use (Figure 2.5). The results confirmed that most data analysis programs use similar libraries. We have already implemented specifications for the most frequently used libraries (Section 2.4). Fortunately, the abstract behavior for a library function tends to be simpler than its concrete behavior. We have implemented 380 function summaries mainly for Numpy, Pandas, and scikit-learn and are actively working on adding more function summaries for various libraries.

We require correct specifications to rigorously enforce privacy policies. An illustrative example of the importance of correctly implementing specifications is the renaming function. Cunning inside attackers may want to bypass the static analysis by renaming sensitive columns. A correct specification which renames the columns in both the schema and the

privacy clauses should mitigate such attacks. To mitigate risks due to such errors, function summaries should be open-sourced so the community can help check their correctness.

Comparison with dynamic approaches. Our choice of a static analysis for PrivAnalyzer is motivated by two major advantages over dynamic approaches: (1) the ability to handle implicit data flows, and (2) the goal of adding minimal run-time overhead. The ability to detect implicit flows is a major advantage of static analysis systems [103, 170, 116, 54], including PrivAnalyzer. Unlike dynamic approaches, PrivAnalyzer cannot be defeated by complex control flow designed to obfuscate execution. For example, the data subject might specify the policy **ALLOW** *REDACT* name (1 :), which requires redacting most of the name column. An analyst might write the following program:

```
if data.name == 'Alice':
  return 1
else:
  return 2
```

This program clearly violates the policy, even though it does not return the value of data.name directly. This kind of violation is due to an *implicit flow* of the name column to the return value. A return value of 1 allows the analyst to confirm with certainty that the data subject's name is Alice. This kind of implicit flow presents a significant challenge for dynamic analyses, because dynamic analyses *only execute one branch* of each conditional, and can make no conclusions about the branch *not* taken. A dynamic analysis must either place significant restrictions on the use of sensitive values in conditionals, or allow for unsoundness due to implicit flows.

Static analyzers like PrivAnalyzer, on the other hand, can perform a worst-case analysis that inspects *both* branches. PrivAnalyzer's analysis executes both branches with the abstract interpreter and returns the *worst-case* outcome of both branches. For loops with no bound on the number of iterations, the analysis results represent the worst-case outcome, no matter how many iterations execute at runtime. This power comes at the expense of a potential lack of precision—the analysis may reject programs that are actually safe to run. Our evaluation suggests, however, that PrivAnalyzer analysis is sufficiently precise for programs that perform data analyses. Static analysis tools like PrivAnalyzer do not require the policy specification to be aware of implicit flows as it combines both types of flows in its results.

PrivAnalyzer by Example

The input to PrivAnalyzer is a single analysis program, plus all of the policies of the data files it processes. For each of the program's outputs, PrivAnalyzer produces a residual policy. After running the analysis, PrivGuard will attach each of these residual policies to the appropriate concrete output.

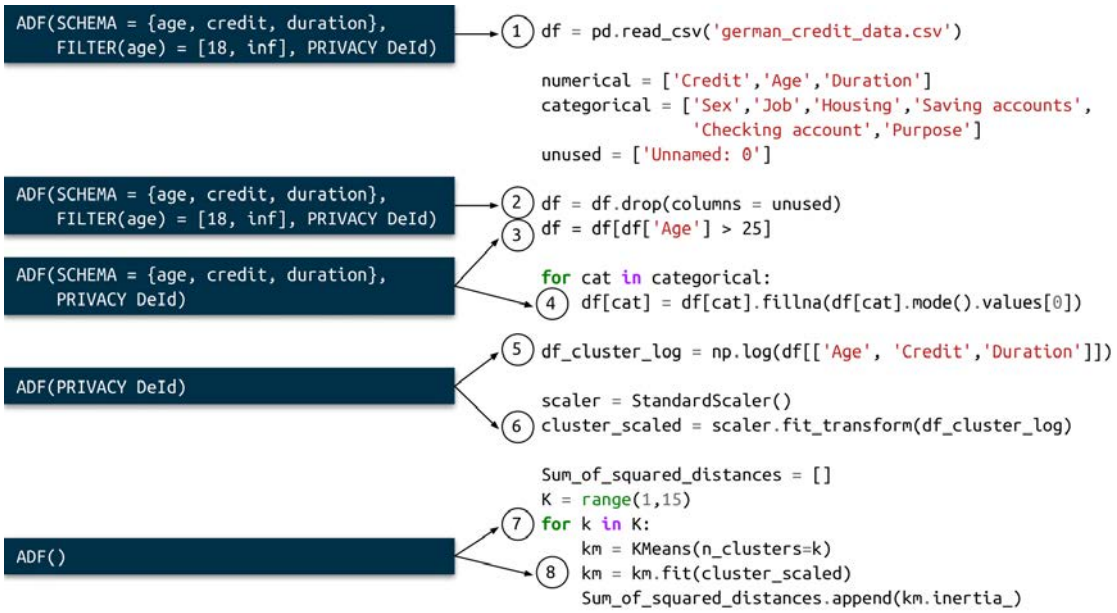


Figure 2.6: Example Abstract Interpretation with PrivAnalyzer.

PrivAnalyzer works by performing abstract interpretation, where the inputs to the program are abstract values containing representations of the associated policies. The output of this process is a set of abstract values containing representations of the residual policies.

A complete example of this process is summarized in Figure 2.6. The analysis is a Python program adapted from an open-source analysis submitted to Kaggle [32]. Important locations in the program are labeled with numbers (e.g. ①) and the associated residual policy PrivAnalyzer computes *at that program location*.

Reading Data into Abstract Values. The program begins by reading in a CSV containing the sensitive data (Fig. 2.6, ①). PrivAnalyzer's abstract interpretation library redefines `read\_csv` to return an *abstract dataframe* containing the policy associated with the data being read. At location ①, the variable `df` thus contains the full policy for the program's input. In this example, the policy allows the program to use the "age," "credit," and "duration" columns, requires filtering out the data of minors, and requires the use of de-identification.

Mixing Concrete and Abstract Values. The next part of the program defines some constants, which PrivAnalyzer represents with concrete values lacking attached policies. Then, the program drops one of the input columns (Fig. 2.6, ②); this action does not change the policy, because the columns being dropped are not sufficient to satisfy any of the policy's requirements, so the `df` variable is unchanged.

Satisfying FILTER Requirements. The next statement (Fig. 2.6, ③) performs a filtering operation. The PrivAnalyzer library redefines this operation to eliminate the appropriate *FILTER* requirements from the policy; since this filtering operation removes individuals below the age of 25, it satisfies the *FILTER* requirement in the policy, and the new value of `df` is an abstract dataframe whose policy does not have this requirement.

Handling Loops. The next part of the program contains a **for** loop (Fig. 2.6, ④). Loops are traditionally a big challenge for static analyzers. PrivAnalyzer is designed to take advantage of *loops over concrete values*, like this one, by simply executing them concretely. PrivAnalyzer can also handle loops over abstract values (described later in this section), but these were relatively rare in our case studies.

Libraries and Black-box Operations. The next pieces of code (Fig. 2.6, ⑤ and ⑥) first take the log of each feature, then scale the features. Both of these operations impact the *scale* of feature values. After these operations, it becomes impossible to satisfy policy requirements like *FILTER*, because the original data values have been lost. For lossy operations like these, which we call *black-box* operations (detailed below), PrivAnalyzer is designed to raise an alarm if value-dependent requirements (like *FILTER*) remain in the policy.

Training a Model. The final piece of code (Fig. 2.6, ⑦ and ⑧) uses the `KMeans` implementation in scikit-learn to perform clustering. We summarize this method to specify that it satisfies de-identification requirements in the policy. The result in our example is an empty residual policy, which would allow the analyst to view the results.

Challenging Language Features

We now address the approach taken in PrivAnalyzer for several challenging language features.

Conditionals. Conditionals depending on abstract values require the abstract interpreter to run both branches and compute the upper bound on both results. Since Python does not allow redefining `if` statements, we add a pre-processing step to PrivGuard that transforms conditionals by running *both* branches.

Loops. Loops are traditionally the most challenging construct for abstract interpreters to handle. Fortunately, loops in Python programs for data analytics often fall into restricted classes, like the ones in the example of Figure 2.6. Both loops in this example are over constant values—so our abstract interpreter can simply run each loop body as many times as the constant requires.

Loops over abstract values are more challenging, and the simple approach may never terminate. To address this situation, we define a *widening* operator [108] for each of the

abstract domains used in PrivAnalyzer. Widening operators force the loop to arrive at a fixpoint; in our example, widening corresponds to assuming the loop body will be executed over the whole dataframe.

Aliasing. Another challenge for abstract interpretation comes from the issue of *aliasing*, where two variables point to the same value. Sometimes, it is impossible for the analysis to determine *what* abstract value a variable references. In this case, it is also impossible to determine the outcome of the side effects on the variable.

Our approach of re-using the existing Python interpreter helps address this challenge: in PrivAnalyzer, all variable references are evaluated concretely. In most cases, references are to concrete objects, so the analysis corresponds exactly to concrete execution. In a few cases, however, this approach leads to less precise analysis. For example, if a variable is re-assigned in both branches of a conditional, PrivAnalyzer must assume the *worst-case* abstract value (i.e. with the most restrictive policy) is assigned to the variable in both cases. This approach works well in our setting, where conditionals and aliasing are both relatively rare.

Attribute Enforcement

We now describe some attribute-specific details of our compliance analysis.

Schema, Filter, and Redact. The *SCHEMA*, *FILTER*, and *REDACT* attributes can be defined formally and compliance can be checked by PrivAnalyzer. In our implementation, relevant function summaries remove the attribute from the privacy policy if the library's concrete implementation satisfies the corresponding requirement. Our summaries thus implement abstract interpretation for these functions. Note that PrivAnalyzer assumes that functions without summaries do not satisfy *any* policy requirements. PrivAnalyzer is therefore *incomplete*: some programs may be rejected (due to insufficient function summaries) despite satisfying the relevant policies.

Privacy. The *PRIVACY* attribute is also checked by PrivAnalyzer. Analysis programs can satisfy de-identification requirements by calling functions that remove identifying information (e.g. aggregating records or training machine learning models). Programs can satisfy k-Anonymity[2], ℓ-diversity[2], t-closeness or differential privacy requirements by calling specific functions that provide these properties. Our function summaries include representative implementations from the current literature: IBM Differential Privacy Library [65], K-Anonymity Library [72], and Google's Tensorflow Privacy library [141].

[2] k-Anonymity and ℓ-diversity are vulnerable to disclosure attacks as pointed out in [87]. k-Anonymity is vulnerable to homogeneity and background knowledge attacks, and ℓ-diversity suffers from skewness and similarity attacks. We strongly encourage using t-closeness or differential privacy for stronger protection. PrivGuard provides weaker approaches only for compatibility purposes.

There are two subtleties when enforcing differential privacy attributes. First, programs that satisfy differential privacy also need to track the *privacy budget* [41]. By default, PrivGuard tracks a single global cumulative privacy cost (values for ϵ and δ) for each source of submitted data, and rejects new analysis programs after the privacy cost exceeds the budget amount. PrivAnalyzer reports the privacy cost of a single analysis program, allowing PrivGuard to update the global privacy cost. A single global privacy budget may be quickly exhausted in a setting with many analysts performing different analyses. One solution to this problem is to generate differentially private synthetic data, which can be used in later analyses without further privacy cost. The High-Dimensional Matrix Mechanism (HDMM) [98] is one example of an algorithm for this purpose, used by the US Census Bureau to release differentially private data. In PrivGuard, arbitrarily many additional analyses can be performed on the output of algorithms like HDMM without using up the privacy budget. Another solution is fine-grained budgeting, at the record level (as in ProPer [44]) or a statically defined "region" level (as in UniTraX [102]). The first is more precise, but requires silently dropping records for which the privacy budget is exhausted, leading to biased results. Both approaches allow for continuous analysis of fresh data in growing databases (e.g. running a specific query workload every day on just the new data obtained that day). Second, to calculate privacy budget, PrivGuard initializes a variable to track the sensitivity of the pre-processing steps before the differentially private function. The pre-processing function summaries should manipulate the variable to specify their influence on the sensitivity. If such specification is absent in any function before the differentially private function, PrivGuard will throw a warning and recognize the differential privacy requirement as unsatisfied. We also plan to incorporate differential privacy type systems such Fuzz [114], DFuzz [51], PathC [4], HOARe^2 [10], LightDP [165], Fuzzi [166] and DUET [104].

Role. *ROLE* attributes are enforced by authentication techniques such as password, 2-factor authentication , or even biometric authentication . In addition, *ROLE* attributes are also recorded in an auditable system log described in the next paragraph, and the analysts and the data curators will be held accountable for fake identities.

Purpose. The *PURPOSE* attribute is inherently informal. Thus, we take an accountability-based approach to compliance checking for purposes. Analysts can specify their purposes when submitting the analysis program, and may specify an invalid purpose unintentionally or maliciously. These purposes will be used by PrivAnalyzer to satisfy *PURPOSE* requirements. PrivGuard produces an audit log recording analysts, analysis programs, and claimed purposes. Thus, all the analysis happening in the system can be verified after the fact, and analysts can be held legally accountable for using invalid purposes.

Formal Model of PrivAnalyzer

In this section, we formally present technique for proving soundness of our static analysis. We use the filter attribute as an example to demonstrate the technique for proving soundness in the context of a simple model of a programming language. As described in Section 2.3, Python is a more dynamic language than our model, and these dynamic features may represent possible side channels for malicious adversaries.

Abstract Domains

We provide the formal definition on of the abstract domains used in PrivGuard in this section. Formally, given a concrete domain $\mathcal{D}$, we define the following components:

- an *abstract domain* $< \mathcal{D}^\sharp, \sqsubseteq >$ containing abstract values ($\mathcal{D}^\sharp$) which represent classes of abstract values, and a partial ordering ($\sqsubseteq$) on those values.
- an *abstraction function* $\alpha : \mathcal{D} \to \mathcal{D}^\sharp$ mapping concrete values to abstract values.
- a *concretization function* $\gamma : \mathcal{D}^\sharp \to \mathcal{P}(\mathcal{D})$ mapping abstract values to sets of concrete values.

For each attribute type, we define $\mathcal{D}^\sharp$, $\sqsubseteq$, α, and γ. We define each abstract domain in terms of a tabular data format approximating a Pandas dataframe (which we denote *df*).

As described earlier, some kinds of loops may cause the abstract interpreter to loop forever. To address this challenge, we adopt the standard approach of using a *widening operator* [108], denoted ∇, in place of the standard partial ordering operator $\sqsubseteq$. Unlike the partial ordering, the widening operator is guaranteed to stabilize when applied repeatedly in a loop. Finite abstract domains do not require a widening operator; for infinite domains (like the interval domain used in *FILTER* attributes), we adopt the standard widening operator used for the underlying domain (e.g. widening for intervals [108]).

Filter Attributes. We track filtering done by analysis programs using an *interval domain* [108], which is commonly used in the abstract interpretation literature. Abstract dataframes in this domain (denoted $\mathbb{D}^\sharp$) associate an interval (denoted $\mathbb{I}$) with each column c_i, and analysis results are guaranteed to lie within the interval. In addition to known intervals (i.e. (n_1, n_2)), our set of intervals includes $\top$ (i.e. the interval $(-\infty, \infty)$) and $\bot$ (i.e. the interval containing no numbers). Our interval domain works for dataframe columns containing integers or real numbers; our formalization below uses $\mathbb{R}^\infty$ to denote the set of real numbers, extended with infinity.

$$\begin{array}{lcll} i & \in & \mathbb{I}_\mathbb{R} & = & (\mathbb{R}^\infty \times \mathbb{R}^\infty) \cup \{\top, \bot\} \\ df^\sharp & \in & \mathbb{D}^\sharp & = & (c_1 : \mathbb{I}_\mathbb{R}) \times ... \times (c_n : \mathbb{I}_\mathbb{R}) \end{array}$$

$$f \in \mathsf{field} \quad m \in \mathsf{int} \quad s \in \mathsf{schema} \quad x \in \mathsf{dataframes}$$

$$\begin{array}{rcrl} \varphi & \in & \mathsf{filter} & ::= \quad c < m \mid c > m \\ e & \in & \mathsf{expr} & ::= \quad x \mid \mathrm{filter}(\varphi, e) \mid \mathrm{project}(s, e) \\ & & & \mid \quad \mathrm{redact}(c, n, n, e) \mid \mathrm{join}(e, e) \\ & & & \mid \quad \mathrm{union}(e, e) \mid \mathrm{dpCount}(\epsilon, \delta, e) \end{array}$$

Figure 2.7: Program surface syntax

$$\begin{array}{lcl} \mathtt{eval}(\rho, x) & = & \rho(x) \\ \mathtt{eval}(\rho, \mathrm{filter}(\varphi, e)) & = & \sigma_{\varphi}\mathtt{eval}(\rho, e) \\ \mathtt{eval}(\rho, \mathrm{project}(s, e)) & = & \Pi_{s}\mathtt{eval}(\rho, e) \\ \mathtt{eval}(\rho, \mathrm{redact}(c, n_1, n_2, e)) & = & \{c : \mathsf{stars}(s, n_1, n_2) \mid \\ & & \quad c : s \in \mathtt{eval}(\rho, e)\} \\ \mathtt{eval}(\rho, \mathrm{join}(e_1, e_2)) & = & \mathtt{eval}(\rho, e_1) \bowtie \mathtt{eval}(\rho, e_2) \\ \mathtt{eval}(\rho, \mathrm{union}(e_1, e_2)) & = & \mathtt{eval}(\rho, e_1) \cup \mathtt{eval}(\rho, e_2) \end{array}$$

Figure 2.8: Concrete interpreter for language in Figure 2.7.

For ease of presentation, and without loss of generality, we define α and γ in terms of dataframes with a single column c. We denote values in the column c by $df.c$.

$$\begin{array}{lcl} c : \bot & \sqsubseteq & \mathbb{D}^{\sharp} \\ \mathbb{D}^{\sharp} & \sqsubseteq & c : \top \\ c : (n_1, n_2) & \sqsubseteq & c : (n_3, n_4) \text{ if } n_1 \geq n_3 \wedge n_2 \leq n_4 \\ \alpha(df) & = & c : (\min(df.c), \max(df.c)) \\ \gamma(c : \top) & = & \mathbb{D} \\ \gamma(c : \bot) & = & \{\} \\ \gamma(c : (n_1, n_2)) & = & \{df \mid \forall v \in df.c.\ n_1 \leq v \leq n_2\} \end{array}$$

Soundness

A sound analysis by abstract interpretation requires defining the following:

- A *programming language* of expressions $e \in \mathsf{Expr}$. We define a simple language for dataframes, inspired by Pandas, in Figure 2.7.
- A *concrete interpreter* $\mathtt{eval} : \mathsf{Env} \times \mathsf{Expr} \to \mathcal{D}$ specifying the semantics of the programming language on concrete values. We define the concrete interpreter for our simple language in Figure 2.8.

$$
\begin{array}{lcl}
\texttt{eval}^\sharp(\rho, x) & = & \rho(x) \\
\texttt{eval}^\sharp(\rho, \text{filter}(\varphi, e)) & = & \texttt{eval}^\sharp(\rho, e) - \mathsf{interval}(\varphi) \\
\texttt{eval}^\sharp(\rho, \text{project}(s, e)) & = & \texttt{eval}^\sharp(\rho, e) \\
\texttt{eval}^\sharp(\rho, \text{redact}(c, n_1, n_2, e)) & = & \texttt{eval}^\sharp(\rho, e) \\
\texttt{eval}^\sharp(\rho, \text{join}(e_1, e_2)) & = & \texttt{eval}^\sharp(\rho, e_1) \sqcup \texttt{eval}^\sharp(\rho, e_2) \\
\texttt{eval}^\sharp(\rho, \text{union}(e_1, e_2)) & = & \texttt{eval}^\sharp(\rho, e_1) \sqcup \texttt{eval}^\sharp(\rho, e_2)
\end{array}
$$

$$
\mathsf{interval}(c < m) = c : (-\infty, m)
$$
$$
\mathsf{interval}(c > m) = c : (m, \infty)
$$

$$
c : (l_1, u_1) - c : (l_2, u_2) = c : (l_3, u_3)
$$
where $l_3 = -\infty$ when $l_1 \leq l_2$, and l_1 otherwise
$u_3 = \infty$ when $u_1 \geq u_2$, and u_1 otherwise

Figure 2.9: Abstract interpreter for *FILTER* attributes

- An *abstract interpreter* $\texttt{eval}^\sharp : \mathsf{Env}^\sharp \times \mathsf{Expr} \to \mathcal{D}^\sharp$ specifying the semantics of the programming language on abstract values. An example for *FILTER* attributes appears in Figure 2.9.

The concrete interpreter `eval` computes the concrete result of a program e in the context of an environment mapping variables to concrete values. The abstract interpreter $\texttt{eval}^\sharp$ computes an *output policy* of a program e in the context of an abstract environment mapping variables to their policies. The program satisfies its input policies if it has at least one empty clause (*i.e.* a satisfied clause) in its output policy.

The *soundness* property for the abstract interpreter says that the concrete result of evaluating a program e is contained in the class of values represented by the result of evaluating the same program using the abstract interpreter.

Theorem 1 (Soundness)**.** *For all environments ρ and expressions e, the abstract interpreter $\texttt{eval}^\sharp$ is a sound approximation of the concrete interpreter* `eval`*:*

$$
\texttt{eval}(\rho, e) \in \gamma(\texttt{eval}^\sharp(\alpha(\rho), e))
$$

where the abstract environment $\alpha(\rho)$ is obtained by abstracting each value in the concrete environment ρ (i.e. $\alpha(\rho)(x) = \alpha(\rho(x))$).

Soundness can be proven for each abstract domain separately. In each case, the proof of soundness proceeds by induction on e, with case analysis on the kind of abstract value returned by uses of $\texttt{eval}^\sharp$ on subterms.

Soundness for Filter Attributes. We present the abstract interpreter for the filter abstract domain in Figure 2.9. The interesting case of this interpreter is the one for filter expressions, which converts the filter predicate φ to an interval and returns an abstract

value derived from the meet of this interval and the recursive call. We prove the soundness of the interpreter as following.

Proof of soundness. By induction on e. We consider the (representative) case where $e = \mathsf{filter}(\varphi, e')$. By the inductive hypothesis, we have that $\texttt{eval}(\rho, e') \in \gamma(\texttt{eval}^\sharp(\alpha(\rho), e'))$. Let $\mathsf{interval}(\varphi) = (n_1, n_2)$. We want to show (by definition of `eval` and $\texttt{eval}^\sharp$:

$$
\begin{aligned}
&\texttt{eval}(\rho, \text{filter}(\varphi, e')) \in \gamma(\texttt{eval}^\sharp(\alpha(\rho), \text{filter}(\varphi, e'))) \\
&\iff \sigma_\varphi \texttt{eval}(\rho, e') \in \gamma(\texttt{eval}^\sharp(\alpha(\rho), e') - \mathsf{interval}(\varphi)) \\
&\iff \sigma_{n_1 \le c \le n_2} \texttt{eval}(\rho, e') \in \gamma(c : (n_1, n_2) \sqcap \texttt{eval}^\sharp(\alpha(\rho), e')) \\
&\iff \sigma_{n_1 \le c \le n_2} \texttt{eval}(\rho, e') \in \gamma(c : (\max(n_1, n_3), \min(n_2, n_4))) \\
&\iff \sigma_{n_1 \le c \le n_2} \texttt{eval}(\rho, e') \in \\
&\quad \{df \mid \forall v \in df.c.\ \max(n_1, n_3) \le v \le \min(n_2, n_4)\}
\end{aligned}
$$

which holds by the inductive hypothesis and semantics of selection in relational algebra. □

Usability Survey

To complement the survey in [125] targeting privacy champions, we conducted an online survey targeting general users to obtain a preliminary understanding of how far expertise is needed to understand or encode privacy preferences. The survey is granted IRB exemption by Office for Protection of Human Subjects under category 2 of the Federal and/or UC Berkeley requirements.

We recruit 30 participants in total among which 7 has no background in programming (Group A), 2 has programmed in one language (Group B), 15 has programmed in two languages or more (Group C), and 6 self-identify as experts in programming language (Group D). The survey is comprised of 8 questions in total. The first three are about understanding privacy policies and the latter five are to choose the correct option from 4 possible choices. Group A makes 38.1% (8/21) mistakes when understanding and 31.4% (11/35) mistakes when selecting. Group B makes 16.7% (1/6) mistakes when understanding and 20.0% (2/10) mistakes when selecting. Group C makes 15.6% (7/45) mistakes when understanding and 17.3% (13/75) mistakes when selecting. Group D makes 11.1% (2/18) mistakes when understanding and 13.3% (4/30) mistakes when selecting. Besides, each question has a different focus. For example, Question 2 focuses on understanding *ROLE* and *PURPOSE* attributes. The details of the survey results are included in Table 2.1.

We observe several interesting facts in the survey results. First, there is a big gap in accuracy between Group A and B. This indicates that it might not be trivial for users without programming experience to specify their privacy preferences in LEGALEASE directly. We deem it an important future direction to simplify this process for Group A users using more user-friendly API or machine-learning-based translation tools. Besides, this also shows that any programming experience is helpful in understanding and encoding in LEGALEASE. Second, there is no obvious accuracy gap between Group B and Group C, and Group D has better accuracy than them. Third, it is hard for all groups to answer questions related

to *PRIVACY* attributes. The hardness stems from the hardness in understanding privacy techniques such as differential privacy.

#Question	Type	Group A	Group B	Group C	Group D
1	*SCHEMA*, *FILTER*, *REDACT*	50%	0.0%	11.7%	0.0%
2	*ROLE*, *PURPOSE*	66.7%	25.0%	11.7%	0.0%
3	nested clauses	16.7%	25.0%	23.5%	25.0%
4	*SCHEMA*, *PRIVACY*	66.7%	25.0%	52.9%	50.0%
5	*SCHEMA*, *ROLE*	33.3%	25.0%	58.8%	0.0%
6	*SCHEMA*, *PURPOSE*	83.3%	0.0%	11.7%	0.0%
7	*SCHEMA*, *FILTER*	0.0%	0.0%	0.0%	0.0%
8	*SCHEMA*, *REDACT*	0.0%	25.0%	11.7%	25.0%

Table 2.1: Survey results in detail. #Question refers to the ID of the question.

Ethical Considerations. The survey was posted as a public questionnaire on Twitter and Wechat with informed consent. The participants opted in the survey voluntarily. In order to fully respect the participants' privacy, we do not collect any personal identifiable information from them. Only the answers to the questionnaire are collected.

2.4 Evaluation

The evaluation is designed to demonstrate that (1) PrivAnalyzer supports commonly-used libraries for data analytics and can analyze real-world programs, and (2) PrivGuard is lightweight and scalable. To demonstrate, we (1) test PrivAnalyzer using 23 real-world analysis programs drawn from the Kaggle contest platform, and (2) measure the overhead of PrivGuard using a subset of these programs. The results show that PrivGuard can correctly enforce PrivPolicy policies on these programs with reasonable performance overhead.

Experiment Setup

We implemented PrivAnalyzer in about 1400 lines of Python and integrated it in an industrial-level data governance platform, Parcel [110], to prototype PrivGuard. We instantiated our implementation with Inter Planetary File System (IPFS) for the storage layer, AES-256-GCM for the encryption algorithm, and AMD SEV for TEE.

To evaluate PrivGuard's static analysis on real-world programs, we collect analysis programs for 23 different tasks from Kaggle, one of the most well-known platforms for data analysis contests. These programs analyze sensitive data such as fraud detection [67] and

Index	Application Type	# LoC	External libraries
1	Fraud Detection[67]	157	LightGBM, NumPy, pandas, scikit-learn
2	Fraud Detection[67]	157	LightGBM, NumPy, pandas, scikit-learn
3	Merchant Recommendation[45]	199	LightGBM, NumPy, pandas, scikit-learn
4	Customer Satisfaction Prediction[118]	104	NumPy, pandas, scikit-learn, XGBoost
5	Customer Transaction Prediction[119]	89	NumPy, pandas, scikit-learn
6	Customer Transaction Prediction[119]	89	NumPy, pandas, scikit-learn
7	Bank Customer Classification[8]	276	NumPy, pandas, scikit-learn
8	Bank Customer Segmentation[9]	75	NumPy, pandas, scikit-learn
9	Credit Risk Analysis[32]	57	NumPy, pandas, sklearn
10	Bank Customer Churn Prediction[7]	169	NumPy, pandas, SciPy, scikit-learn
11	Heart Disease Causal Inference[62]	83	NumPy, pandas, SHAP, scikit-learn
12	Classify Forest Categories[30]	50	NumPy, pandas, PySpark
13	PyTorch-Simple LSTM[129]	178	NumPy, pandas, Keras, PyTorch
14	Tensorflow-Solve Titanic[142]	163	NumPy, pandas, scikit-learn, Tensorflow
15	Earthquake Prediction[84]	132	NumPy, pandas, tsfresh, librosa, pywt, SciPy
16	Display Advertising[37]	60	math
17	Fraud Detection[139]	106	NumPy, pandas, Keras, Tensorflow
18	Restaurant Revenue Prediction[115]	115	NumPy, pandas, FastAI, scikit-learn
19	NFL Analytics[107]	152	NumPy, pandas, SciPy, scikit-learn
20	NCAA Prediction[59]	561	NumPy, pandas, Pymc3
21	Home Value Prediction[173]	272	NumPy, pandas, sklearn, XGBoost
22	Malware Prediction[99]	194	NumPy, pandas
23	Web Traffic Forecasting[159]	346	NumPy, pandas

Table 2.2: Case Study Programs. # LoC = Lines of Code.

transaction prediction [119]. We selected these programs as case studies to demonstrate PrivGuard's ability to analyze real-world analysis programs and support commonly-used libraries. These case studies are chosen to be representative of the programs written by data scientists during day-to-day operations at many different kinds of organizations. We surveyed 100 kaggle programs randomly and found that approximately 85% programs are less than 300 lines of code (after removing blank lines). Correspondingly, our case studies range between 50 and 276 lines of code, total exactly 1600 lines of code and include randomly picked programs from Kaggle notebook and top-ranked programs on the contest leaderboard. As shown in Table 2.2, these programs use a variety of external libraries including widely used libraries like pandas, PySpark, Tensorflow, PyTorch, scikit-learn, and XGBoost.

As the first step of the evaluation, we use PrivAnalyzer to analyze the collected programs listed in Table 2.2. In the experiment, we manually designed an appropriate Legalease policy for each program, and then attached them to each of the datasets. For each program, we recorded both the time running on the dataset and the time for PrivAnalyzer to analyze the program. We also manually checked that the analysis result output by PrivAnalyzer was correct. All the experiments were run on a Ubuntu 18.04 LTS server with 32 AMD Opteron(TM) Processor 6212 with 512GB RAM. The results appear in Table 2.3. As the second step of the evaluation, we picked 7 case studies with open-source datasets, ran them on the PrivGuard prototype, and measured the system overhead.

Index	**Exec Time (s)**	**Analysis Time (s)**	**Overhead**	**Soundness**
1	12571.01	1.41	$1.12e-2\%$	✓
2	19951.10	3.32	$1.66e-2\%$	✓
3	16762.65	1.18	$7.04e-3\%$	✓
4	151.72	1.22	$8.04e-1\%$	✓
5	17.14	1.08	6.30%	✓
6	33.71	0.96	2.84%	✓
7	32.66	2.03	6.22%	✓
8	86.82	2.19	2.52%	✓
9	4.65	1.01	21.72%	✓
10	295.16	1.29	$4.37e-1\%$	✓
11	3.99	1.00	25.06%	✓
12	1017.83	1.01	$1.00e-1\%$	✓
13	717.58	6.84	$9.53e-1\%$	✓
14	13.33	4.78	35.86%	✓
15	217.36	2.26	1.04%	✓
16	3.60	1.20	33.33%	✓
17	5.19	1.37	26.40%	✓
18	47.12	1.57	3.33%	✓
19	202.96	1.33	$6.55e-1\%$	✓
20	59.83	1.66	2.77%	✓
21	54.44	2.55	4.68%	✓
22	51.36	1.23	2.39%	✓
23	45.58	2.45	5.37%	✓

Table 2.3: Execution Time vs. Analysis Time. The index of case studies is the same as in Table 2.2.

Results

Support for Real-World Programs. Our experiment demonstrates PrivGuard's ability to analyze the kinds of analysis programs commonly written to process data in organizations. The results in Table 2.3 show that the static analysis took just a second or two for most programs, with three outliers taking 3.32, 4.78, and 6.84 seconds. The reason for the outliers is described in the next paragraph.

As in other abstract interpretation and symbolic execution frameworks, we expect that conditionals, loops, and other control-flow constructs will have a bigger effect on analysis time

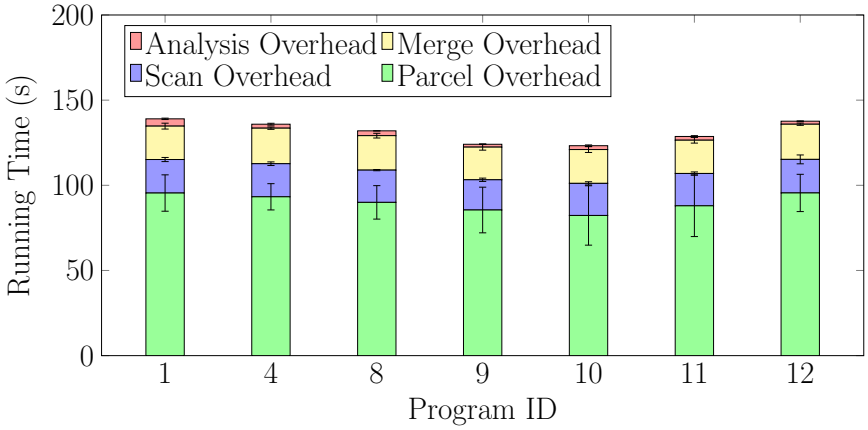


Figure 2.10: System overhead of PrivGuard prototype with one million simulated users.

than program length. Fortunately, programs for data analytics and machine learning tend not to make extensive use of these constructs, especially compared to traditional programs. Instead, they tend to use constructs provided by libraries, like the query features defined in pandas or the model construction classes provided by scikit-learn. The outliers mentioned above (case studies 2, 13, and 14) contain relatively heavy use of conditionals, and as a result, their analysis took slightly longer than the other programs. These results suggest that PrivGuard will scale to even longer programs for data analytics and machine learning, especially if those programs follow the same pattern of favoring library use over traditional control-flow constructs.

Table 2.3 reports performance overhead for all 23 case studies. The results report *analysis performance overhead*—the ratio of the time taken for static analysis to the native execution time of the original program. The results show that this overhead is negligible. For case study programs which take a significant time to run, the performance overhead of deploying PrivGuard is typically *less than 1%*. For faster-running programs, the absolute overhead is similar—just a second or two, typically—but this represents a larger relative change when the program's execution time is small. The *maximum* relative performance overhead in our experiments was about 35%, for a program taking only 13.33 seconds.

Overall Performance Overhead and Scalability. We also evaluate 7 case studies on our prototype implementation and measure the total overhead of the PrivGuard system. The results appear in Figure 2.10 and 2.11. For each case study, we synthesize one million random policies by combining possible attributes and changing parameters in the attributes to simulate one million data subjects' privacy preferences. The results show that the performance overhead for ingesting one million policies is under 150 seconds. Concretely, over half of the overhead is spent on Parcel's system overhead such as data uploading, data storage, data encryption, *etc.* Data ingestion takes about one-third of the overhead and the static

analysis only takes less than 10 seconds.

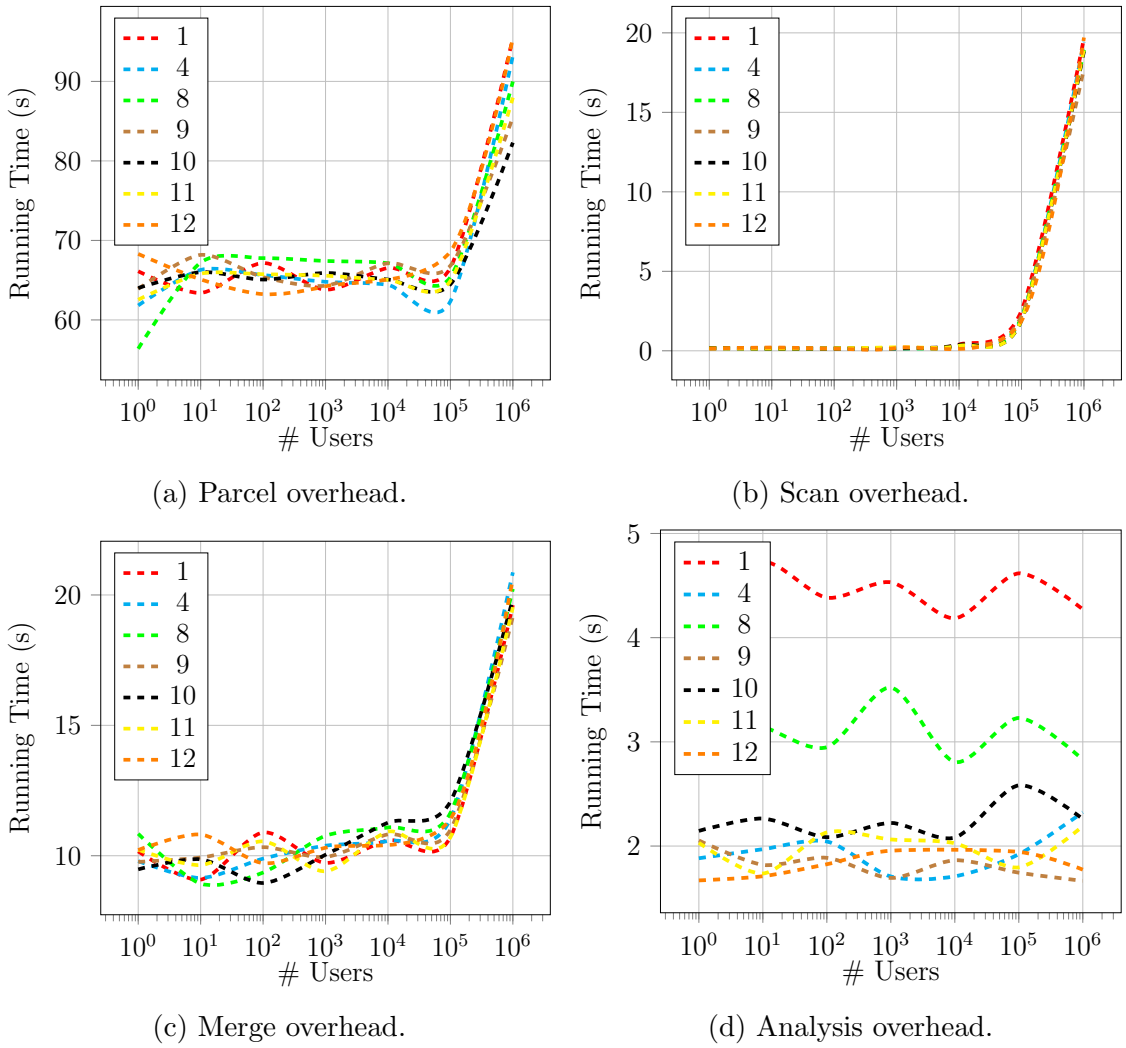
We also benchmark the overhead with different numbers of users as shown in Figure 2.11. Parcel overhead refers to the overhead incurred by the Parcel platform such as data loading or transmission. Scan overhead refers to the time spent on finding the policies no stricter than the guard policy. Merge overhead refers to the time used to merge the datasets inside TEE. Analysis overhead refers to the overhead of running PrivAnalyzer. As shown in Figure 2.11, Parcel overhead, scan overhead and merge overhead are relatively stable when the number of users is small and then scale linearly with the number of users. Note that we use the log_{10} scale to represent the x-axis. The curves are exponential but the rate scales linearly. For all experiments except the static analysis: Overhead $= \mathcal{O}(\#\text{Users}) + \mathcal{O}(1)$. Not surprisingly, the analysis overhead is almost constant for a fixed program. The results show that PrivGuard is scalable to a large number of users and datasets.

2.5 Related Work

Work related to PrivGuard falls into two categories: (1) efforts to formalize privacy policies; (2) efforts to enforce privacy policies in data processing systems; Next we briefly summarize work in these categories.

Privacy Policy Formalism. Tschantz et al. [148] use modified Markov Decision Processes to formalize purpose restrictions in privacy policies. Chowdhury [27] present a policy language based on a subset of FOTL capturing the requirements of HIPAA. Lam et al. [83] prove that for any privacy policy that conforms to patterns evident in HIPAA, there exists a finite representative hospital database that illustrates how the law applies in all possible hospitals. Gerl et al. [53] introduce LPL, an extensible Layered Privacy Language that allows to express and enforce new privacy properties such as user consent. Trabelsi et al. [147] propose the PPL sticky policy based on XACML [48] to express and process privacy policies in Web 2.0. Azraoui et al. [5] focus on the accountability side of privacy policies and extend PPL to A-PPL.

Privacy Policy Compliance Enforcement. Going beyond the formalism of privacy regulations, recent research also explores techniques to enforce formalized privacy policies. Chowdhury et al. [28] propose to use temporal model-checking for run-time monitoring of privacy policies. Sen et al. [125] introduce Grok, a data inventory for Map-Reduce-like big data systems. PODS / SOLID [94] focuses on returning control over data to its owners. In PPL policy engine [147], the policy decision point (PDP) matches the data curator's privacy policy and data subjects' privacy preferences to decide compliance. The privacy policy is enforced by the policy enforcement point. Compared with our work, the PPL policy engine provide limited support for fine-grained privacy compliance in complex data analysis tasks as its enforcement engine relies on direct trigger-to-action translation. In addition, PPL does not provide a rigorous soundness proof. Similar differences exist in its extension A-PPL [5]



(a) Parcel overhead.

(b) Scan overhead.

(c) Merge overhead.

(d) Analysis overhead.

Figure 2.11: PrivGuard overhead details. The scalability is linear (note the logarithmic scale).

and the SPECIAL project [133]. Our work provides an enforcement mechanism necessary to address these issues and can be seen as a first step towards meeting the ambitious challenge posed by Maniatis et al. [93].

2.6 Discussion

We propose PrivGuard, a framework for facilitating privacy regulation compliance. The core component is PrivAnalyzer, a static analyzer supporting compliance verification between a program and a policy. We prototype PrivGuard on Parcel, an industrial-level

data governance platform. We believe that PrivGuard has the potential to significantly reduce the cost of privacy regulation compliance.

There are also several future directions we would like to pursue for future versions of PrivGuard. First, we would like to further improve the usability of PrivGuard's API in consideration of HCI requirements so that non-experts can easily specify their own privacy preferences. Second, PrivGuard now adopts an one-shot consent strategy, which covers most current application scenarios but has several defects as pointed out in [123]. This limitation can be addressed by allowing the data curator to ask data subjects for dynamic consent after data collection, as depicted in [123].

Part II

Bottom-up Approach: Differentially Private Data Analysis

Chapter 3

Differentially Private Causal Inference

Causal graph discovery refers to the process of discovering causal relation graphs from purely observational data. Like other statistical data, a causal graph might leak sensitive information about participants in the dataset. In this section, we present a differentially private causal graph discovery algorithm, `Priv-PC`, which improves both utility and running time compared to the state-of-the-art. The design of `Priv-PC` follows a novel paradigm called `sieve-and-examine` which uses a small amount of privacy budget to filter out "insignificant" queries, and leverages the remaining budget to obtain highly accurate answers for the "significant" queries. We also conducted the first sensitivity analysis for conditional independence tests including conditional Kendall's τ and conditional Spearman's ρ. We evaluated `Priv-PC` on 7 public datasets and compared with the state-of-the-art. The results show that `Priv-PC` achieves 10.61 to 293.87 times speedup and better utility. The implementation of `Priv-PC`, including the code used in our evaluation, is available at `https://github.com/sunblaze-ucb/Priv-PC-Differentially-Private-Causal-Graph-Discovery`.

3.1 Introduction

Causal graph discovery refers to the process of discovering causal relation graphs from purely observational data. Causal graph discovery has seen wide deployment in areas like genomics, ecology, epidemiology, space physics, clinical medicine, and neuroscience. The `PC` algorithm [134] is one of the most popular causal discovery algorithms. It is comprised of a series of independence tests like Spearman's ρ [132], Kendall's τ [76], G-test [96] or χ^2-test [97]. The algorithm starts by connecting all variables in the graph. If an independence test indicates that two variables are independent, the edge between the two variables will be removed from the causal graph. The process will continue until the edges between independent variables are totally removed.

Like other statistical data, a causal graph can leak information about participants in the dataset. For instance, Genome-Wide Association Studies involve finding causal relations between Single Nucleotide Polymorphisms (SNPs) and diseases. In this case, a causal link

between a specific SNP and a disease may indicate the participation of a minority patient. However, the problem of *effective causal graph discovery with differential privacy* remains largely unsolved.

State-of-the-art. The most straightforward solution is to perturb all the independence tests in the `PC` algorithm with calibrated noise such as Laplace or Gaussian noise [41]. However, as pointed out in [161], the numerous independence tests incur *too much noise to output meaningful causal graphs*. Even tight composition techniques based on Rényi differential privacy [1, 101, 158] cannot address the issue. The state-of-the-art solution to differentially private causal graph discovery is `EM-PC` [161], a modification of the `PC` algorithm which uses the exponential mechanism to guarantee differential privacy. Instead of perturbing each independence test with noise, `EM-PC` randomly selects how many and which edges to delete using the exponential mechanism. In this way, `EM-PC` manages to achieve a relative balance between utility and privacy. However, `EM-PC` has two severe defects. First, `EM-PC` suffers from *extremely slow computation* because: 1) many independence tests which should have been pruned have to be computed because the exponential mechanism can only deal with off-line queries; 2) the utility function used in applying the exponential mechanism is computationally intensive. In fact, the computation overhead of the utility score is so large that the implementation from the original paper [161] uses a greedy search to approximate the solution presented in the paper. It is unclear whether the differential privacy still holds given this compromise since the original sensitivity bound does not hold anymore. Second, `EM-PC` also suffers from low utility because it changes the intrinsic workflow of the `PC` algorithm. Concretely, `EM-PC` explicitly decides how many edges to delete while PC makes this decision in an on-line fashion. Thus, `EM-PC` does not converge to the `PC` algorithm and cannot achieve perfect accuracy even with substantial privacy budget.

Proposed solution. In this section, we proposed `Priv-PC`, a differentially private causal graph discovery algorithm with *much less running time* and *better result utility* compared to `EM-PC`. The design of `Priv-PC` follows a novel paradigm called `sieve-and-examine`. Intuitively, `sieve-and-examine` spends a small amount of privacy budget to filter out "insignificant" queries and answers the rest of queries carefully with substantial privacy budget. The proof that `Priv-PC` is differentially private is straightforward. The challenge is to understand why it also gives less running time and better utility.

`Sieve-and-examine`, as the name indicates, comprises two sub-processes executing alternately: `sieve` and `examine`. In the context of causal graph discovery, the `sieve` process uses sub-sampled sparse vector technique [41, 6] to filter out variable pairs unlikely to be independent with a little privacy budget. Then the `examine` process uses Laplace mechanism [41] to carefully check the remaining variable pairs and decide whether they are really independent with substantial privacy budget.

We choose sparse vector technique for its nice properties. First, sparse vector technique can answer a large number of threshold queries but only pay privacy cost for those whose

output is above the threshold[1]. Fortunately, in causal graph discovery, only a few independence tests will yield results above the threshold so with sparse vector technique, we can save much privacy cost. Second, sparse vector technique can deal with online queries, so redundant independence tests can be pruned adaptively once their target edge is removed due to a previous independence test. Thus, with sparse vector technique, we can get rid of the unnecessary independence tests in `EM-PC` and significantly accelerate private causal discovery. We propose to further accelerate the execution and reduce privacy cost by augmenting the sparse vector technique using sub-sampling without replacement [6].

However, sparse vector technique is known for its poor utility [91], which raises concern about the accuracy of `sieve-and-examine`. Actually, there exist two types of errors in `sieve-and-examine`. Type I error refers to mistakenly filtering out truly independent pairs. Type II error refers to the failure to filter out variable pairs that are not independent. To reduce the errors, we take a two-step approach. First, we suppress type I error by tweaking the threshold lower so the noise is more unlikely to flip over the output from independence to the opposite. The tweak, on the other hand, will increase the number of type II errors. Fortunately, type II errors can be corrected by the `examine` process with a high probability. Furthermore, the threshold tweak typically only increases type II errors slightly because a meaningful threshold should be far away from the clusters of both independent pairs and dependent pairs.

Independence tests in Priv-PC. The noise magnitude in `Priv-PC` grows proportionally to the sensitivity of the independence test. Thus, to obtain an appropriate noise level, we conducted rigorous sensitivity analysis for commonly used conditional independence tests including conditional Kendall's τ [79, 140] and conditional Spearman's ρ [140]. We finally chose Kendall's τ in `Priv-PC` because of its small sensitivity. It also remains an interesting open question how to integrate independence tests with infinite sensitivity such as G-test [96] or χ^2-test [97] in `Priv-PC`.

3.2 Problem Setup

In this section, we review necessary background knowledge about differential privacy and causal graph discovery.

Differentially Private Selection.

In this chapter, we mainly touch two differentially private selection mechanisms, namely exponential mechanism and sparse vector technique. Exponential mechanism is designed for differentially private selection from infinite output set. It computes an utility score for each candidate output and randomly selects from the output candidates based on probability

[1]Sparse vector technique can also only pay for queries below the threshold. For clarity, we only focus on the above-threshold queries.

derived from the utility score. The pseudo-code for exponential mechanism is shown in Algorithm 1.

Algorithm 1: Exponential Mechanism

Input: $\mathcal{D}$: dataset, O: output set, u: 1-sensitive utility function, ϵ: privacy parameters.

Function *EM*$(\mathcal{D}, O, u, \epsilon)$**:**
- Initiate U as an empty lists
- **for** $o \in O$ **do**
  - Append $u(\mathcal{D}, o)$ to U
- Randomly select o from O according to probability $\frac{\exp(\epsilon u_o/2)}{\sum_{u_i \in U} \exp(\epsilon u_i/2)}$.

Sparse vector technique is a widely used differentially private mechanism. It can answer a large number of queries while only paying privacy cost for a small portion of them. The pseudo-code for sparse vector technique is shown in Algorithm 2.

Algorithm 2: Sparse Vector Technique.

Input: D: dataset, $\{f_i\}$: 1-sensitive queries, T: threshold, c: quota of above-threshold queries, (ϵ, δ): privacy parameters.

Function *SVT*$(\mathcal{D}, \{f_i\}, T, c, \epsilon, \delta)$**:**
- **if** $\delta = 0$ **then** Let $\sigma = \frac{2c}{\epsilon}$ **else** Let $\sigma = \frac{\sqrt{32c \log \frac{1}{\delta}}}{\epsilon}$
- Let count $= 0$, $\hat{T}_{count} = T + Lap(\sigma)$
- **for** *Each query* i **do**
  - Let $\nu = Lap(2\sigma)$
  - **if** $f_i(\mathcal{D}) + \nu_i \geq \hat{T}_{count}$ **then**
    - Output $a_i = \top$
    - Let count = count + 1 Let $\hat{T}_{count} = T + Lap(\sigma)$
  - **else** Output $a_i = \perp$
  - **if** $count \geq q$ **then** Halt

Composability is an important property of differential privacy. If several mechanisms are differentially private, so is their composition. The privacy parameters of the composed mechanism can be derived using standard composition theorem like advanced composition [41] and moments accountant [1]. The sparse vector technique [41] can be viewed as a special case for composition because it can answer a large number of threshold queries while only paying privacy cost for queries above the threshold.

Causal Graph Discovery.

In statistics, causal graphs are *directed acyclic graphs* (DAGs) used to encode assumptions about the data-generating process, which are formally defined as follows.

Definition 3 (Causal Graph)**.** *A causal graph $\mathcal{G}$ is a directed acyclic graph (DAG) represented by a vertex set $V = \{v_1, v_2, \cdots, v_k\}$ and an edge set $E \subseteq V \times V$. $Adj(\mathcal{G}, v_i)$ represents the adjacent set of node v_i in graph $\mathcal{G}$. The skeleton of a DAG is the undirected version of the graph.*

Causal graph discovery refers to the process of discovering causal graphs under an observed distribution such as a dataset. The output of a causal graph discovery algorithm is a *completed, partially directed acyclic graph* (CPDAG) because the directions of some edges cannot be determined only based on the observational distribution.

There exist a variety of causal graph discovery algorithms and the `PC` algorithm is one of the most popular ones. The first step in the `PC` algorithm is to find the skeleton of the causal graph using conditional independence tests. Then the edges are directed based on some auxiliary information from the independence tests to obtain CPDAG. Because the second step does not touch the data, we only focus on the first step given the post-processing theorem [41] in differential privacy. The pseudocode of the `PC` algorithm is provided in Algorithm 3.

Algorithm 3: `PC` Algorithm.

Input: V: vertex set, $\mathcal{D}$: dataset, T: threshold

Function *PC*$(V, \mathcal{D}, T)$**:**
- $\mathcal{G}$ = complete graph on V, $ord = 0$
- **while** $\exists\ v_i$ *s.t.* $|Adj(\mathcal{G}, v_i) - v_j| \geq ord$ **do**
  - **while** $\exists$ *edge* (v_i, v_j) *s.t.* $|Adj(\mathcal{G}, v_i) - v_j| \geq ord$ *that has not been tested* **do**
    - select edge (v_i, v_j) in $\mathcal{G}$ *s.t.* $|Adj(\mathcal{G}, v_i) - v_j| \geq ord$
    - **while** $\exists\ S$ *that has not been tested* **do**
      - choose $S \subseteq Adj(\mathcal{G}, v_i) - v_j$, $|S| = ord$
      - **if** $indep\_test(ij|S) \geq T$ **then**
        - remove (v_i, v_j) from $\mathcal{G}$
        - break
  - $ord = ord + 1$
- Output $\mathcal{G}$

Conditional Independence Test.

Conditional independence test is an important building block in many causal discovery algorithms. It is used to test whether two random variables are independent conditional on another set of variables.

Definition 4 (Conditional independence test)**.** *A conditional independence test* $f : V \times V \times 2^V \times D \to \{0, 1\}$ *decides whether variable* $i \neq j \in V$ *are independent conditional on another set of variables* $k \subseteq V, i, j \notin k$*.* f *is composed of a dependence score* $s : V \times V \times 2^V \times D \to \mathbb{R}$ *and a threshold* $T \in \mathbb{R}$*.*

$$f(\mathcal{D}) = \begin{cases} 0, & s(\mathcal{D}) \leq T \\ 1, & s(\mathcal{D}) > T \end{cases}$$

, where 1 *represents "independent" and* 0 *represents "not independent".* f *is called* $|k|$*-order conditional independence test where* $|k|$ *is the size of the conditional set.*

Commonly used independence tests include Spearman's ρ, Kendall's τ, G-test and χ^2-test. Note that some independence tests like Kendall's τ output 1 when the dependence score is below the threshold. However, for clarity, we assume all the independence tests output 1 when the dependence score is above the threshold without loss of generality. Here we focus on Kendall's τ because of its small sensitivity (Section 3.3).

3.3 Priv-PC: Private Causal Graph Discovery

In this section, we proposed `Priv-PC` to effectively discover causal graphs following `sieve-and-examine` paradigm. Concretely, `Priv-PC` leverages the `sieve` process to sift out variable pairs unlikely to independent using a little privacy cost and then carefully `examine`s the remaining ones with substantial privacy budget. We first introduce `sieve-and-examine` mechanism and then demonstrate how to apply `sieve-and-examine` to the `PC` algorithm to obtain `Priv-PC`. At last, we bridge `sieve-and-examine` and `Priv-PC` by providing sensitivity analysis for Kendall's τ.

Sieve-and-examine Mechanism.

Most causal graph discovery algorithms like the `PC` algorithm need to answer many independence tests – *too many to obtain an acceptable privacy guarantee* using independent perturbation mechanisms like Laplace mechanism [41]. `EM-PC` is the first step towards reconciling the contradiction between utility and privacy in private causal discovery. However, `EM-PC` suffers from *extremely slow running time* because it additionally runs a large number of independence tests that should have been pruned. A straightforward solution is to replace the exponential mechanism [41] with the sparse vector technique [41, 91]. Sparse vector technique allows adaptive queries so unnecessary independence tests can be pruned early. Besides, the privacy cost of sparse vector technique only degrades with the number of queries above the threshold. Fortunately, only a few independence tests in causal discovery yield values above the threshold so the sparse vector technique can also save considerable privacy budget in causal discovery. However, sparse vector technique suffers from *low accuracy* as pointed out in [91], which is not acceptable in many use cases such as medical or financial analysis.

One-off sieve-and-examine. To address the issue, we propose a novel paradigm called `sieve-and-examine` which alternately executes sub-sampled sparse vector technique and output perturbation. Intuitively, the `sieve` process uses sub-sampled sparse vector technique to filter out independence tests unlikely to be above the threshold with small privacy budget. Then the left queries are `examined` carefully with substantial privacy budget using output perturbation.

For simplicity, we first introduce one-off `sieve-and-examine` shown in Algorithm 4, a simplified version of `sieve-and-examine` that halts after seeing one query above the threshold. We prove that one-off `sieve-and-examine` is ϵ-differentially private. The result can be generalized to multiple above-threshold queries using composition theorem.

Algorithm 4: One-off sieve-and-examine mechanism.

Input: $\mathcal{D}$: dataset, $\{f_i\}$: queries, T: threshold, t: threshold tweak, m: subset size, ϵ: privacy parameters, Δ: sensitivity of f

. **Function** *Sieve_and_examine*$(\mathcal{D}, \{f_i\}, T, t, m, \epsilon, \Delta)$**:**
  $\mathcal{D}' \xleftarrow{\$} \mathcal{D}, n = |\mathcal{D}|, m = |\mathcal{D}'|$;
  Let $\epsilon' = \ln(\frac{n}{m}(e^{\epsilon/2} - 1) + 1)$;
  Let $\hat{T} = T - t + Lap(\frac{2\Delta}{\epsilon'})$;
  for *Each query* i **do**
    if $f_i(\mathcal{D}') + Lap(\frac{4\Delta}{\epsilon'}) \geq \hat{T}$ **then**
      Let $k = i$;
      Break;
  if $f_k(\mathcal{D}) + Lap(\frac{2\Delta}{\epsilon}) \geq T$ **then** Output k ;
  else Output $\perp$;

Theorem 2. *Algorithm 4 is ϵ-differentially private.*

Proof Sketch. We separately prove that `sieve` and `examine` are both $\epsilon/2$-differentially private. The main body of `sieve` is a sparse vector technique with $\epsilon' = \ln(\frac{n}{m}(e^{\epsilon/2} - 1) + 1)$ privacy cost. Sub-sampling reduces the cost to $\epsilon/2$ following Theorem 9 from [6]. `Examine` process is a $\epsilon/2$-differentially private Laplace mechanism. Thus, `sieve-and-examine` is ϵ-differentially private using composition theorem. □

Result Utility. The differential privacy proof is straightforward. The challenge will be to understand when it also gives utility. Thus, we bound the probability of type I error and type II error in Algorithm 4 separately.

Theorem 3 (Error bound)**.**

- *(Type I error) Let E_1^α denotes the event that Algorithm 4 filters out $f(\mathcal{D}) \geq T + \alpha$.*

$$\mathbb{P}[E_1^\alpha] \leq \exp(-\frac{\epsilon'(\alpha + t)}{6\Delta}) - \frac{1}{4}\exp(-\frac{\epsilon'(\alpha + t)}{3\Delta})$$

- *(Type II error) Let E_2^α denotes the event that Algorithm 4 fails to filter out $f(\mathcal{D}) \leq T - \alpha$. If $\alpha \geq t$, then*

$$\mathbb{P}[E_2^\alpha] \leq \exp(-\frac{12\epsilon\alpha + \epsilon'(\alpha - t)}{6\Delta}) - \frac{1}{4}\exp(-\frac{6\epsilon\alpha + \epsilon'(\alpha - t)}{3\Delta})$$

.

Proof for Type I error. We want to upper bound the probability of E_1^α. Equally, we lower bound the probability of $\neg E_1^\alpha$ by the probability that the noise on the threshold is smaller than $\frac{1}{3}(t+\alpha)$ and the noise on the query output is smaller than $\frac{2}{3}(t+\alpha)$. Because for Laplace noise, $\mathbb{P}[x \geq w] = \exp(-w/b)$, we have

$$\mathbb{P}[\neg E_1^\alpha] \geq (1 - \frac{1}{2}\exp(-\frac{\epsilon'(\alpha + t)}{6\Delta}))^2 = 1 - \exp(-\frac{\epsilon'(\alpha + t)}{6\Delta}) + \frac{1}{4}\exp(-\frac{\epsilon'(\alpha + t)}{3\Delta})$$

. Thus,

$$\mathbb{P}[E_1^\alpha] \leq 1 - \mathbb{P}[\neg E_1^\alpha] \leq \exp(-\frac{\epsilon'(\alpha + t)}{6\Delta}) - \frac{1}{4}\exp(-\frac{\epsilon'(\alpha + t)}{3\Delta})$$

□

Proof for Type II error. If $f(D)$ is not filtered out, it needs to be missed by both sparse vector technique and the Laplace mechanism. The probability bound for being missed by the sparse vector technique is

$$\mathbb{P}[E_{svt}^\alpha] \leq \exp(-\frac{\epsilon'(\alpha - t)}{6\Delta}) - \frac{1}{4}\exp(-\frac{\epsilon'(\alpha - t)}{3\Delta})$$

following similar proof path to theorem 3. The probability being missed by the Laplace mechanism is bounded by

$$\mathbb{P}[E_{lm}^\alpha] = \exp(-\frac{2\epsilon\alpha}{\Delta})$$

. Thus,

$$\begin{aligned}\mathbb{P}[E_2^\alpha] = \mathbb{P}[E_{svt}^\alpha] \cdot \mathbb{P}[E_{lm}^\alpha] &\leq (\exp(-\frac{\epsilon'(\alpha - t)}{6\Delta}) - \frac{1}{4}\exp(-\frac{\epsilon'(\alpha - t)}{3\Delta})) \cdot \exp(-\frac{2\epsilon\alpha}{\Delta}) \\ &= \exp(-\frac{12\epsilon\alpha + \epsilon'(\alpha - t)}{6\Delta}) - \frac{1}{4}\exp(-\frac{6\epsilon\alpha + \epsilon'(\alpha - t)}{3\Delta})\end{aligned}$$

□

Intuitively, theorem 3 bounds the probability of errors conditional on the distance from the dependence score to the threshold. An interesting observation is the tweak on the threshold t decreases the probability of type I errors and increases the probability of type II errors at the same time. Because each type II error increases the privacy cost by ϵ, the

question is "*will the increment of type II errors add too much privacy cost?*" Fortunately, the answer is "no" because the increment of type II errors also depends on the distribution of dependence scores. Generally the empirical distribution of an independence score is a twin-peak curve and the threshold locates in the middle valley. In this case, the threshold tweak only slightly increases the number of type II errors because most dependence scores are far from the threshold[2].

Priv-PC Algorithm.

In this section, we demonstrate how to apply `sieve-and-examine` to `PC` algorithm to obtain `Priv-PC`. We first give an overview of `Priv-PC`. Then we discuss how to optimize the sub-sampling rate in `Priv-PC`.

The complete pseudo-code for `Priv-PC` is shown in Algorithm 5. `Priv-PC` follows the same workflow as the `PC` algorithm. It starts from a complete undirected graph (line 1) and gradually increases the order of the independence tests (line 6, 17). Within a fixed order, `Priv-PC` traverse all the variable pairs with large enough adjacent set (line 8). It selects the conditional variables from the adjacent set (line 9-10) and then executes the conditional independence test to decide whether the edge will be removed from the graph.

To achieve differential privacy, the conditional independence tests are augmented with `sieve-and-examine`. Concretely, `Priv-PC` first sub-samples a subset $\mathcal{D}'$ from $\mathcal{D}$, derives privacy parameter for the `sieve` process and tweaks the threshold (line 3-5). Then, `Priv-PC` executes the `sieve` process by adding noise to both the tweaked threshold (line 5) and the independence test (line 11). Note that the noise parameters here are different from standard `sieve-and-examine` (Algorithm 4) because the sensitivity for Kendall's τ is dependent on the dataset size (Section 18). Once an independence test on the sub-sampled dataset exceeds the threshold (line 11), the `examine` process will run the independence test again on the complete dataset with substantial privacy budget. If the result still exceeds the un-tweaked threshold (line 12), the edge is removed from the graph (line 13). Then, the sub-sampled dataset and the threshold are refreshed for the next round of `sieve-and-examine` (line 14-15).

In Algorithm 5, we require the caller of the function to explicitly give the size of the sub-sampling set. However, since the sensitivity of Kendall's τ also depends on the data size, we can actually derive an optimal sub-sampling size which adds the smallest noise under the same privacy guarantee. This requires to minimize the noise level $\frac{\sqrt{n/m}}{\ln(\frac{n}{m}(\exp(\epsilon/2)-1)+1)}$. Although there is no explicit solution for the optimization problem, we can obtain an approximate solution with numerical solver such as BFGS [109]. On the other hand, when ϵ is small, the optimal sub-sampling size is also too small to yield meaningful independence

[2]A complete explanation contains two parts. First, since most of the dependence scores are far from the threshold, the threshold tweak does not directly change the test results for most queries. Second, because the dependence scores are far from the threshold, the absolute increase of type II error probability is small. Thus, the increment of type II errors is small.

Algorithm 5: `Priv-PC` Algorithm with Kendall's τ.

Input: V: vertex set, $\mathcal{D}$: dataset, T: threshold, t: threshold tweak, m: subset size, ϵ: privacy parameter, Δ: sensitivity on the full dataset.

Function *Priv_PC*$(V, \mathcal{D}, T, t, m, \epsilon, \Delta)$**:**
- $\mathcal{G}$ = complete graph on V, $ord = 0$
- $\mathcal{D}' \xleftarrow{\$} \mathcal{D}, n = |\mathcal{D}|, m = |\mathcal{D}'|$
- Let $\epsilon' = \ln(\frac{n}{m}(e^{\epsilon/2} - 1) + 1)$
- Let $\hat{T} = T - t + Lap(\frac{2\sqrt{n}\Delta}{\sqrt{m}\epsilon'})$
- **while** $\exists$ v_i *s.t.* $|Adj(\mathcal{G}, v_i) - v_j| \geq ord$ **do**
  - **while** $\exists$ *edge* (v_i, v_j) *s.t.* $|Adj(\mathcal{G}, v_i) - v_j| \geq ord$ *that has not been tested* **do**
    - select edge (v_i, v_j) in $\mathcal{G}$ *s.t.* $|Adj(\mathcal{G}, v_i) - v_j| \geq ord$
    - **while** $\exists$ $S \subseteq Adj(\mathcal{G}, v_i) - v_j$ *that has not been tested* **do**
      - choose $S \subseteq Adj(\mathcal{G}, v_i) - v_j$, $|S| = ord$
      - **if** $\tau(ij|S) + Lap(\frac{4\sqrt{n}\Delta}{\sqrt{m}\epsilon'}) \geq \hat{T}$ **then**
        - **if** $\tau(ij|S) + Lap(\frac{2\Delta}{\epsilon}) \geq T$ **then**
          - delete (v_i, v_j) from $\mathcal{G}$
        - $\mathcal{D}' \xleftarrow{\$} \mathcal{D}, |\mathcal{D}'| = m$
        - $\hat{T} = T - t + Lap(\frac{2\sqrt{n}\Delta}{\sqrt{m}\epsilon'})$
        - break
  - $ord = ord + 1$
- Output $\mathcal{G}$, compute the total privacy cost $(\epsilon_{tot}, \delta_{tot})$ with advanced composition.

test results. Thus we take the optimal sub-sampling size by clipping the solution to range $(\frac{n}{20}, n)$.

Independence Tests in Priv-PC.

The last missing piece is the sensitivity of the conditional independence test functions. We finally choose conditional Kendall's τ for its small sensitivity. Conditional Spearman's ρ is another candidate but it can only be used on large datasets because of the large coefficient in its sensitivity.

Kendall's τ. The sensitivity of Kendall's τ is inversely proportional[3] to the training set size as pointed out in [82]. However, in our scenario, the conditional version of Kendall's τ is needed while [82] only gives the sensitivity for non-conditional Kendall's τ. In order to fill the gap, we derive the sensitivity of the conditional Kendall's τ. We first give the complete definition of Kendall's τ and its conditional version.

[3]Note that this requires the size of the dataset to be public which is a common case.

Definition 5 (Kendall's τ)**.** *Let $\{(a_1, b_1), \cdots, (a_n, b_n)\}$ denotes the observations. A pair of observation indices (i, j) are called* concordant *if $a_i > a_j$ and $b_i > b_j$. Otherwise (i, j) is called* discordant*. Kendall's τ is defined as*

$$\tau_{ij} := \frac{2|C - D|}{n(n-1)}$$

where C is the number of concordant pairs and D is the number of discordant pairs.

The first step towards complete sensitivity analysis for unconditional Kendall's τ is to extend the neighboring relation to increment.

Theorem 4. *Kendall's τ is $\frac{2}{n-1}$-sensitive.*

Proof. When the neighboring datasets are defined by replacement, the proof is done in [82]. Now we prove that the sensitivity bound generalizes to neighboring datasets defined by increment.

If we increment a dataset by one row, $|C - D|$ can increase by at most n.

$$s(\tau_{ij}) \leq \frac{|C-D|+n}{\frac{1}{2}n(n+1)} - \frac{|C-D|}{\frac{1}{2}n(n-1)} \leq \frac{|C-D|+n}{\frac{1}{2}n(n-1)} - \frac{|C-D|}{\frac{1}{2}n(n-1)} \leq \frac{2}{n-1}$$

□

Now we are ready to give the full definition of conditional Kendall's τ and derive its sensitivity.

Definition 6 (Conditional Kendall's τ)**.** *We omit the pair indices i, j and use τ_i to represent Kendall's τ in the ith block of the conditional variables. If there are k blocks in total, then conditional Kendall's τ is defined as*

$$\tau = \frac{\sum_{i=1}^{k} w_i \tau_i}{\sqrt{\sum_{j=1}^{k} w_j}}$$

where $w_i = \frac{9n_i(n_i-1)}{2(2n_i+5)}$ is the inverse of τ_i's variance.

Theorem 5. *If the conditional variables have k blocks, then conditional Kendall's τ is $\frac{c_\tau}{\sqrt{n-1}}$-sensitive, where c_τ is an explicit constant typically close to $\frac{9}{2}$.*

Proof. If the ith block contains n_i observations, then $s(\tau_i) = \frac{2}{n_i - 1}$.

Then we need to bound $\frac{w_i}{\sqrt{\sum_{j=1}^{k} w_j}}$ and its sensitivity. Assuming $\forall i \in [1, k], n_i \geq c_1$, then $c_2(n_i - 1) \leq w_i \leq \frac{9(n_i-1)}{4}$ for some explicit constants $c_2 = \frac{9c_1}{2(2c_1+5)}$. Thus

$$\frac{w_i}{\sqrt{\sum_{j=1}^{k} w_j}} \leq \frac{9(n_i - 1)}{4\sqrt{c_2(n-k)}}$$

and

$$s(\frac{w_i}{\sqrt{\sum_{j=1}^{k} w_j}}) \leq \frac{w_i'}{\sqrt{\sum_{j\neq i} w_j + w_i'}} - \frac{w_i}{\sqrt{\sum_{j=1}^{k} w_j}} \leq \frac{w_i' - w_i}{\sqrt{\sum_{j=1}^{k} w_j}} \leq \frac{9}{4\sqrt{c_2(n-k)}}$$

. Thus the complete sensitivity is bounded as follow.

$$s(\tau) \leq (\frac{w_i}{\sqrt{\sum_{j=1}^{k} w_j}} + s(\frac{w_i}{\sqrt{\sum_{j=1}^{k} w_j}}))(\tau_i + s(\tau_i)) - \frac{w_i}{\sqrt{\sum_{j=1}^{k} w_j}}\tau_i \leq \frac{27}{4\sqrt{c_2(n-k)}} + \frac{9}{2c_1\sqrt{c_2(n-k)}}$$

□

Theorem 6. *(Sensitivity of conditional Kendall's τ.) The sensitivity of conditional Kendall's τ in Definition 6 is $\frac{c_1}{\sqrt{n}}$ where n is the size of the input dataset and c_1 is an explicit constant approaching $9/2$ when the dataset size grows.*

Spearman's ρ. We also derive the sensitivity of Spearman's ρ here. We first give the complete definition of Spearman's ρ and its conditional version.

Definition 7 (Spearman's ρ)**.** *Let $\{(a_1, b_1), \cdots, (a_n, b_n)\}$ denotes the observations. If we independently sort the observations $\{a_1, \cdots, a_n\}$ and $\{b_1, \cdots, b_n\}$ in ascending order. Let d_i represent the distance between the order of a_i and b_i. Spearman's ρ is defined as*

$$\rho = |1 - \frac{6\sum_{i=1}^{n} d_i^2}{n(n^-1)}|$$

.

Kusner et al. [82] derive the sensitivity for unconditional Spearman's ρ when the neighboring relation between datasets are constrained to replacement. The first step towards complete sensitivity analysis for unconditional Spearman's ρ is to extend the neighboring relation to increment.

Theorem 7. *Spearman's ρ is $\frac{30}{n}$-sensitive.*

Proof. When the neighboring datasets are defined by replacement, the proof is done in [82]. Now we prove that the sensitivity bound generalizes to neighboring datasets defined by increment. And we denote the incremented observation with (a_{n+1}, b_{n+1}). First, $\forall i \neq n+1$, d_i changes at most 2. Thus $d_i^2 - (d_i - 2)^2 \leq 4(d_i - 1) \leq 4(m-2)$, because d_i is smaller than $m-1$. Besides, d_{n+1} is at most m. Therefore, the sensitivity of ρ si bounded by

$$s(\rho) \leq \frac{30m(m-1)}{m(m^2-1)} \leq \frac{30}{m}$$

□

Now we introduce the definition of conditional spearman's ρ and derive its sensitivity.

Definition 8 (Conditional Spearman's ρ)**.** *We omit the pair indices i, j and use ρ_i to represent Spearman's ρ in the ith block of the conditional variables. If there are k blocks in total, then conditional Spearman's ρ is defined as*

$$\rho = \frac{\sum_{i=1}^{k} w_i \rho_i}{\sqrt{\sum_{j=1}^{k} w_j}}$$

where $w_i = n_i - 1$.

Theorem 8. *Conditional Spearman's ρ is $\frac{c_\rho \sqrt{k}}{\sqrt{n-k}}$-sensitive, where c_ρ is an explicit constant typically close to* 31*.*

Proof. If the ith block contains n_i observations, then $s(\rho_i) = \frac{30}{n_i}$.

Then we need to bound $\frac{w_i}{\sqrt{\sum_{j=1}^{k} w_j}}$ and its sensitivity.

$$\frac{w_i}{\sqrt{\sum_{j=1}^{k} w_j^2}} \leq \frac{n_i - 1}{\sqrt{n-k}}$$

and

$$s(\frac{w_i}{\sqrt{\sum_{j=1}^{k} w_j}}) \leq \frac{w_i'}{\sqrt{\sum_{j \neq i} w_j + w_i'}} - \frac{w_i}{\sqrt{\sum_{j=1}^{k} w_j}} \leq \frac{w_i' - w_i}{\sqrt{\sum_{j=1}^{k} w_j}} \leq \frac{1}{\sqrt{n-k}}$$

. Thus the complete sensitivity is bounded as follow.

$$s(\rho) \leq (\frac{w_i}{\sqrt{\sum_{j=1}^{k} w_j}} + s(\frac{w_i}{\sqrt{\sum_{j=1}^{k} w_j}}))(\rho_i + s(\rho_i)) - \frac{w_i}{\sqrt{\sum_{j=1}^{k} w_j}} \rho_i \leq \frac{31}{\sqrt{n-k}} + \frac{30}{c_1 \sqrt{n-k}}$$

□

3.4 Evaluation

In this section, we evaluate the effectiveness of `Priv-PC` by answering the following two questions. 1) How accurate is the result of `Priv-PC`? 2) How much running time does `Priv-PC` save?

Experiment Setup.

In order to answer the above questions, we selected 7 datasets. The detailed information about the datasets is shown in Table 3.1.

Dataset	# Features	# Samples	# Edges	Type
Earthquake [78]	5	100K	4	Binary
Cancer [78]	5	100K	4	Binary
Asia [85]	8	100K	10	Binary
Survey [124]	6	100K	6	Discrete
Alarm [15]	37	100K	46	Discrete
Sachs [117]	11	100K	17	Discrete
Child [16]	20	100K	25	Discrete

Table 3.1: Datasets used in the evaluation.

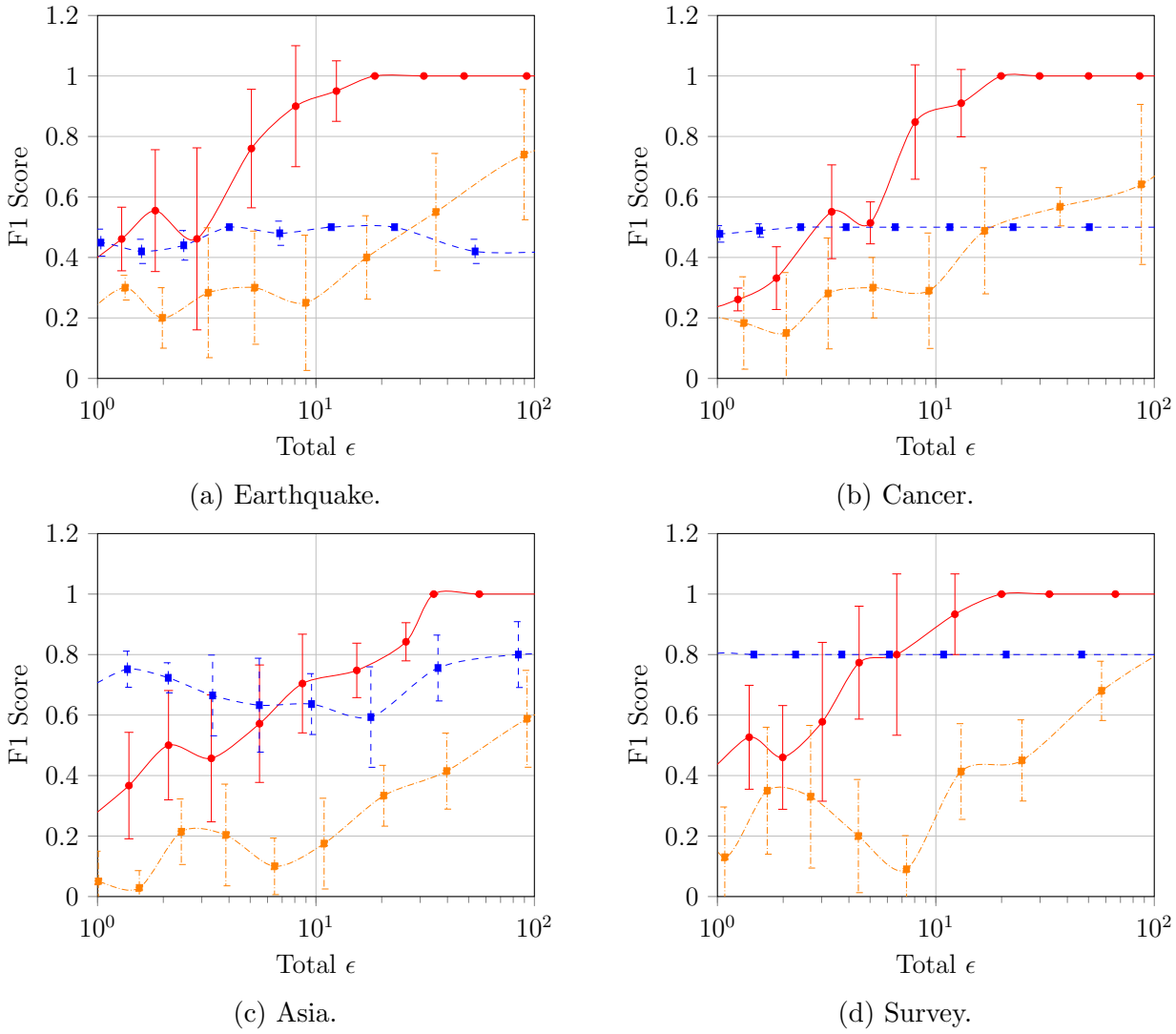
To directly compare `EM-PC` and `Priv-PC`, we ran the two algorithms on the datasets with 21 different privacy parameters and presented the results with accumulated privacy cost between 1 and 100. Furthermore, to demonstrate the utility improvement due to `sieve-and-examine`, we also directly applied sparse vector technique to `PC` algorithm (`SVT-PC`) and evaluated it under the same setting. For each privacy parameter, we ran the three algorithms for 5 times and recorded the mean and standard deviation of the utility of the output graph and the running time. We fix $\delta = 1e\text{-}3$ for both `EM-PC` and `Priv-PC` across all the experiments. Utility is measured in terms of $F1$-score[4]. All the experiments were run on a Ubuntu18.04 LTS server with 32 AMD Opteron(TM) Processor 6212 with 512GB RAM.

Utility

In the evaluation, `Priv-PC` achieves better utility than `EM-PC` when the privacy budget is reasonably large as shown in Figure 3.1. `Priv-PC` always converges to perfect accuracy when privacy cost grows while `EM-PC` does not. The reason is that `Priv-PC` converges to `PC` when privacy cost grows but `EM-PC` does not because it contains a unique sub-routine to explicitly decide the number of edges to delete. The sub-routine intrinsically inhibits the accuracy of `EM-PC`. On the other hand, `EM-PC` achieves better utility under small privacy budget because the exponential mechanism has better utility than the sparse vector technique under small privacy budget as pointed out in [91].

Compared with `SVT-PC`, `Priv-PC` always achieves much better utility in the medium privacy region (Figure 3.1). The improvement should be attributed to `sieve-and-examine` because it effectively suppresses type I and type II errors in sparse vector technique 3.3. Second, because the sensitivity of Kendall's τ is inversely proportional to the size of the input dataset, the noise is typically small when the dataset is large. Thus, the noise does not severely harm the utility while preserving rigorous privacy.

[4]If a causal graph discovery outputs $\mathcal{G} = (V, E)$ and the ground truth is $\mathcal{G}' = (V, E')$. Then $F1$-score is defined as: $F1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision}+\text{Recall}}, \text{Precision} = \frac{|E \cap E'|}{|E|}, \text{Recall} = \frac{|E \cap E'|}{|E'|}$



(a) Earthquake. (b) Cancer.

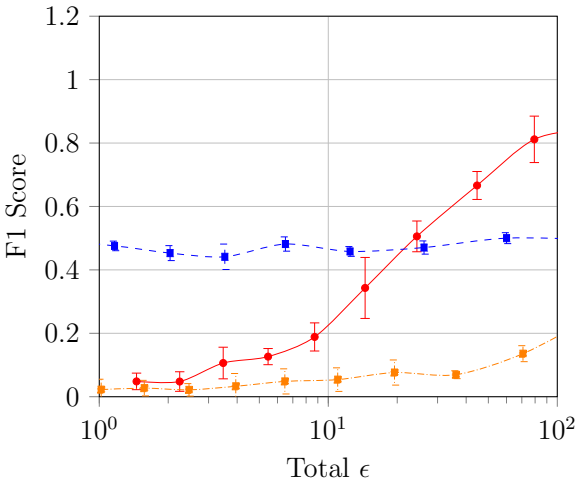
(c) Asia. (d) Survey.

Figure 3.1: $F1$-Score vs. Privacy Budget.

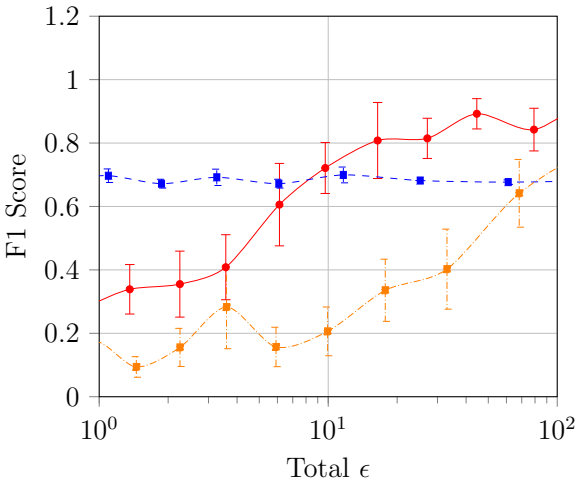
Running Time

`Priv-PC` achieves 10.61 to 32.85 times speedup on small graphs and 74.13 to 293.87 times speedup on larger graphs compared with `EM-PC` as shown in Table 3.2. The improvement is due to two reasons. First, `Priv-PC` can deal with online queries while `EM-PC` cannot. Thus, if an edge is removed due to a previous independence test, later tests on the same edge can be skipped to avoid extra computation overhead. Second, in the `sieve` process, `Priv-PC` only runs independence tests on a subset of the dataset which further accelerates the process. This also explains why `Priv-PC` sometimes runs faster than `SVT-PC`.

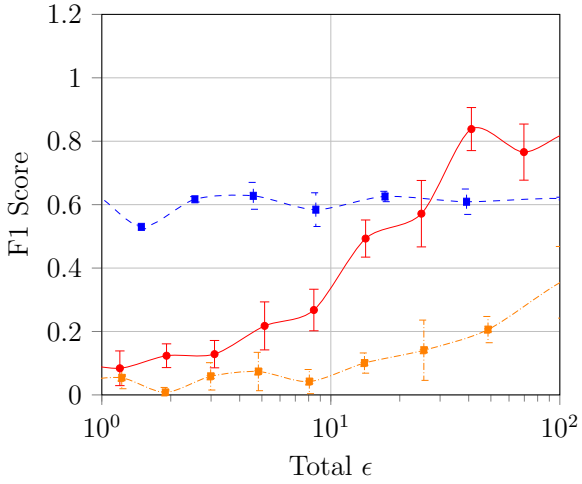
To better understand how the two factors contribute to the speedup, we run `Priv-PC`



(e) Alarm.



(f) Sachs.



(g) Child.

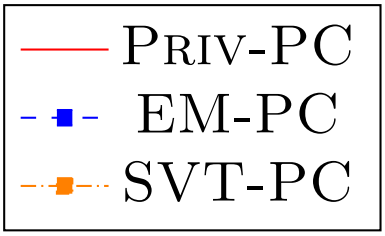


Figure 3.1: $F1$-Score vs. Privacy Budget. (Continued)

without sub-sampling under the same setting and include the results in Table 3.2. The results show that on small graphs, the first factor provides 5.97 to 27.41 times speedup and sub-sampling provides 1.20 to 2.40 times speedup; on larger graphs, the first factor provides 33.62 to 131.41 times speedup and sub-sampling provides 2.20 to 4.38 times speedup.

To better illustrate the source of the speedup, we measure the number of independence tests conducted in `EM-PC` and `Priv-PC` as shown in Table 3.3. The results show that `Priv-PC` saves 34.4% to 56.0% independence tests on small graphs and 84.3% to 86.8% on larger graphs compared to `EM-PC`.

Average Running Time	EM-PC	SVT	Priv-PC	Priv-PC w/o sub-sampling
Earthquake [78]	176.04s	3.38s	6.62s	11.01s
Cancer [78]	64.62s	2.94s	6.09s	10.83s
Asia [85]	531.80s	10.06s	16.19s	19.40s
Survey [124]	68.13s	1.21s	2.13s	5.12s
Alarm [15]	10601.33s	71.23s	143.01s	315.32s
Sachs [117]	4858.42s	4.29s	16.65s	72.98s
Child [16]	25140.67s	32.85s	85.55s	191.31s

Table 3.2: Running time when privacy budget for each `sieve-and-examine` is 1.

#IDP tests	`Priv-PC`	`EM-PC`
Asia	95	216
Cancer	37	57
Earthquake	40	61
Survey	29	38
Alarm	1843	12979
Sachs	165	1224
Child	1162	7393

Table 3.3: The number of independence tests in `Priv-PC` and `EM-PC`.

3.5 Related Work

Causal inference has a long history and there are several excellent overviews [113, 56] of this area. In this section, we briefly introduce the related works in the two most relevant sub-areas: causal discovery based on graph models and private causal inference.

Causal discovery based on graph models can be roughly classified into two categories. The first category is constraint-based causal discovery. The `PC` algorithm [134] is the most well-known algorithm in this category. It traverses all the edges and adjacent conditional sets in the causal graph and removes the edge if the conditional independence test indicates that the edge connects two independent variables. An important variation of the `PC` algorithm is the Fast Causal Inference (FCI) [134], which tolerates latent confounders. The Greedy Equivalence Search (GES) [25] is another widely-used algorithm in this category which starts with an empty graph and gradually adds edges. The second category is based on functional causal models (FCM). A FCM represents the effect Y as a function of the direct causes X and some noise: $Y = f(X, \epsilon; \theta)$. In Linear, Non-Gaussian and Acyclic Model (LiNGAM) [127], f is linear, and only one of ϵ and X can be Gaussian. In Post-Nonlinear Model (PNL) [167, 168], $Y = f_2(f_1(X) + \epsilon)$. Additive Noise Model (ANM) [66] further constrains the post-nonlinear transformation of PNL model.

Private causal discovery has a relatively short history. In 2013, Johnson *et al.* [71] studied differentially private Genome-Wide Association Studies (GWAS). They used Laplace mechanism and exponential mechanism to build specific queries of interest in GWAS. In 2015, Kusner *et al.* [82] analyzed the sensitivity of several commonly used dependence scores on training and testing datasets, and then applied Laplace mechanism to the ANM model. Xu *et al.* [161] proposed to apply exponential mechanism to the `PC` algorithm. Another line of work focuses on private bayesian inference including [35, 63, 17]. Their pioneering works are inspiring but lack novelty from differential privacy side because they all directly leverage off-the-shelf differentially private mechanisms without any modification.

3.6 Discussion

`Priv-PC` takes an important step towards practical differentially private causal discovery with high accuracy and short running time. We also performed an empirical study to demonstrate the advantages compared with the state-of-the-art.

At the same time, we observe many challenges in differentially private causal discovery that existing techniques are not capable of handling. For example, it is unclear how to reconcile independence tests with infinite sensitivity such as G-test and χ^2-test; it is unclear how to handle data type beyond categorical data like numerical data since `PC` algorithm only handles discrete data. We consider all these problems as important future work in the research agenda toward solving the private causal discovery problem.

Chapter 4

Differentially Private Frequency Moments Estimation in Streaming

We prove that $\mathbb{F}_p$ sketch, a well-celebrated streaming algorithm for frequency moments estimation, is differentially private as is when $p \in (0, 1]$. $\mathbb{F}_p$ sketch uses only polylogarithmic space, exponentially better than existing DP baselines and only worse than the optimal non-private baseline by a logarithmic factor. The evaluation shows that $\mathbb{F}_p$ sketch can achieve reasonable accuracy with differential privacy guarantee.

4.1 Introduction

Counting is one of the most fundamental operations in almost every area of computer science. It typically refers to estimating the cardinality (the 0^{th} frequency moment) of a given set. However, counting can actually refer to the process of estimating a broader class of statistics, namely p^{th} frequency moment, denoted F_p. Frequency moments estimation is at the core of various important statistical problems. F_1 is used for data mining [31] and hypothesis tests [69]. F_2 has applications in calculating Gini index [90, 55] and surprise index [58], training random forests [19], numerical linear algebra [29, 120] and network anomaly detection [81, 146]. Fractional frequency moments are used in Shannon entropy estimation [61, 169] and image decomposition [52].

Non-private frequency moments estimation is systematically studied in the data streaming model [3, 22, 146, 47, 68, 89, 75, 105, 106, 74]. This model assumes extremely limited storage such as network routers. The optimal non-private algorithm [74] uses only polylogarithmic space to maintain frequency moments. In the present work, we inherit the low space complexity requirement for the versatility of the algorithm.

The data being counted sometimes contains sensitive information. For example, to calculate Gini index, the data should contain pairs of ID and income. Frequency moments of such data, if published, might leak sensitive information. To mitigate, the gold standard of differential privacy (DP) should be applied. Special cases of DP frequency moments estima-

tion such as $p = 0, 1, 2$ are well-studied in a wide spectrum of works [26, 130, 18, 126, 149, 26, 20, 100].

In the present work, we make the first customized effort towards DP estimation of fractional frequency moments, *i.e.* $p \in (0, 1]$ with low space complexity. We show that a well-known streaming algorithm, namely $\mathbb{F}_p$ sketch [68], preserves differential privacy as is. With its small space complexity, $\mathbb{F}_p$ sketch elegantly solves the trilemma between efficiency, accuracy, and privacy.

Problem Formulation. We use bold lowercase letters to denote vectors (*e.g.* $\mathbf{a}, \mathbf{b}, \mathbf{c}$) and bold uppercase letters to denote matrices (*e.g.* $\mathbf{A}, \mathbf{B}, \mathbf{C}$). $\{1, \cdots, n\}$ is denoted by $[n]$.

Let $\mathcal{S} = \{(k_1, v_1), \cdots, (k_n, v_n)\}$ $(n \geq 1)$ be a stream of key-value pairs where $k_i \in [m]$ $(m \geq 2), v_i \in [M]$ $(M \geq 1)$. We would like to design a randomized mechanism $\mathcal{M}$ that estimates the p^{th} frequency moment:

$$F_p(\mathcal{S}) = \sum_{k=1}^{m} (\sum_{i=1}^{n} \mathbb{I}(k_i = k) v_i)^p$$

for $p \in (0, 1]$ where $\mathbb{I}$ is an indicator function returning 1 if $k = k_i$ and 0 otherwise. Oftentimes, n, m is large (*e.g.* IP streams on routers) so $\mathcal{M}$ should take polylogarithmic space in terms of n, m.

Proof Intuition. We summarize the intuition behind the proof that $\mathbb{F}_p$ sketch is differentially private when $p \in (0, 1]$. Recall that when proving DP for traditional mechanisms such as the Gaussian mechanism, the core is to upper-bound the ratio $\frac{P(x)}{Q(x)}$ where $P(x)$ and $Q(x)$ are the probability density functions of outputs when the inputs are neighboring datasets. In the proof of Gaussian mechanism, $P(x)$ and $Q(x)$ can be viewed as a horizontal translation of each other and the distance between their mean values is the sensitivity of the output.

For $\mathbb{F}_p$, however, neighboring inputs do not translate the output distribution but instead change its scale. For example, when $p = 2$, $P(x)$ and $Q(x)$ are Gaussian distributions with the same mean and different variance. Inspired by the analogy to Gaussian mechanism, we need to address the below two questions to prove differential privacy for $\mathbb{F}_p$ sketches.

Q1. *How to bound the difference between the scales of $P(x)$ and $Q(x)$?*

Q2. *How to bound the ratio between the density functions of $P(x)$ and $Q(x)$?*

To answer Q1, we propose a new sensitivity definition called *pure multiplicative sensitivity*. Pure multiplicative sensitivity depicts the maximal multiplicative change in the output when the inputs are neighboring datasets. We analyze frequency moments estimation and find that its pure multiplicative sensitivity is approximately $\max\{2^{2p-2}, 2^{2-2p}\}$ when $p \in (0, 1]$ and $n \gg M$.

To answer Q2, we first analyze the special case of $p = 1$. When $p = 1$, $\frac{P(x)}{Q(x)}$ is rigorously upper-bounded and thus $\mathbb{F}_1$ sketch preserves ϵ-DP. By analogy, we conjecture that $\mathbb{F}_p, p \in$

$(0, 1]$ also satisfies similar properties, which is doubly confirmed by the numerically simulated plots in Figure 4.2. The conjecture is formally proved in Theorem 11.

4.2 Differentially Private Fractional Frequency Moments Estimation

In this section, we first revisit $\mathbb{F}_p$ sketch and then prove the differential privacy guarantee for $\mathbb{F}_p$ sketch step by step. Different from most differential privacy analyses based on additive sensitivity, our proof depends on a variant of the multiplicative sensitivity [43] called *pure multiplicative sensitivity*. We give the first analysis of pure multiplicative sensitivity for p-th frequency moments. Then we motivate the differential privacy proof using a special case when $p = 1$. Finally we proceed to the general proof that $\mathbb{F}_p$ sketch preserves differential privacy. The main challenge stems from the fact that the density functions of p-stable distributions have no close-form expressions when $p \in (0, 1)$.

Revisiting $\mathbb{F}_p$ Sketch

For completeness, we revisit the well-celebrated $\mathbb{F}_p$ sketch by [68] (also known as stable projection or compressed counting). We first introduce p-stable distribution, the basic building block in $\mathbb{F}_p$ sketch. Then we review how to construct and query $\mathbb{F}_p$ sketch using stable distributions.

Definition 9 (p-stable distribution)**.** *A random variable X follows a β-skewed p-stable distribution if its characteristic function is*

$$\phi_X(t) = \exp(-\zeta|t|^p(1 - \sqrt{-1}\beta \operatorname{sgn}(t) \tan(\frac{\pi p}{2}))$$

where $-1 \leq \beta \leq 1$ is the skewness parameter, $\zeta > 0$ is the scale parameter to the p^{th} power.

In this section, we focus on stable distributions with $\beta = 0$, namely symmetric stable distributions. We denote a symmetric p-stable distribution by $\mathcal{D}_{p,\zeta}$, and slightly abuse the notation to denote the density function as $\mathcal{D}_{p,\zeta}(x)$. Note that the density function is the inverse Fourier transform of the characteristic function.

$$\mathcal{D}_{p,\zeta}(x) = \frac{1}{2\pi} \int_{\mathbb{R}} \exp(-\sqrt{-1}tx)\phi(t)dt = \frac{1}{2\pi} \int_{\mathbb{R}} \cos(xt) \exp(-\zeta|t|^p)dt$$

If two independent random variables $X_1, X_2 \sim \mathcal{D}_{p,1}$, then $C_1X_1 + C_2X_2 \sim \mathcal{D}_{p,C_1^p+C_2^p}$. We refer to this property as *p-stability*. $\mathbb{F}_p$ sketch leverages the p-stability of these distributions to keep track of the frequency moments.

The pseudo-code for vanilla $\mathbb{F}_p$ sketch is presented in Algorithm 6. To construct, a sketch of size r is initialized to all zeros and a projection matrix $\mathbf{P}$ is sampled from $\mathcal{D}_{p,1}^{r\times m}$ (line 2).

For each incoming key-value pair (k_i, v_i), we multiply the one-hot encoding of k_i scaled by v_i with the projection matrix $\mathbf{P}$ and add it to the sketch (line 4).

$$\mathbf{a} = \sum_{i=1}^{n} \mathbf{P} \times v_i \mathbf{e}_{k_i} = \sum_{k=1}^{m} \mathbf{P} \times (\sum_{k_i=k} v_i)\mathbf{e}_{k_i} \sim \mathcal{D}^r_{p, F_p(\mathcal{S})}$$

To query the sketch, we estimate ζ from $\mathbf{a}$ using various estimators such as median, inter-quantile, geometric mean or harmonic mean as suggested by [68], [89] and [88].

Algorithm 6: $\mathbb{F}_p$ sketch.

Input : Data stream: $\mathcal{S} = \{(k_1, v_1), \cdots, (k_n, v_n)\}$
1 **Construct:**
2 | Initialize $\mathbf{a} = \{0\}^r$, $\mathbf{P} \sim \mathcal{D}^{r \times m}_{p,1}$;
3 **Update:**
4 | **for** $i \in [n]$ **do** Let e_{k_i} be the one-hot encoder of k_i, $\mathbf{a} = \mathbf{a} + \mathbf{P} \times v_i \mathbf{e}_{k_i}$;
5 **Query:**
6 | return scale_estimator($\mathbf{a}$);

Pure multiplicative sensitivity of frequency moments estimation

As we will see in the following two subsections, the differential privacy proof for $\mathbb{F}_p$ sketch depends on the pure multiplicative sensitivity of p-th frequency moments. As the first step, we give the definition of pure multiplicative differential privacy. "Pure" is to distinguish from multiplicative sensitivity as defined in [43].

Definition 10 (Pure multiplicative sensitivity)**.** *The multiplicative sensitivity of a function $\mathcal{M}$ is defined as the maximum ratio between outputs on neighboring inputs $\mathcal{S}$ and $\mathcal{S}'$.*

$$\rho_p(n) = \sup_{|\mathcal{S}|=n, |\mathcal{S}'|=n, d(\mathcal{S},\mathcal{S}')=1} \left| \frac{\mathcal{M}(\mathcal{S})}{\mathcal{M}(\mathcal{S}')} \right|$$

We might omit the subscript and argument when they are clear from the context.

The pure multiplicative sensitivity of F_p is as below. Note that the below theorem is for the cash register model. Similar sensitivity holds for the strict turnstile model as proved in Appendix C.

Theorem 9 (Multiplicative sensitivity of F_p)**.** *A mechanism $\mathcal{M}$ which calculates $F_p, p \in (0, 1]$ has pure multiplicative sensitivity upper bounded by*

$$\rho_p \leq 2^{2-2p} \Big(\frac{n - 1 + M}{n - 1 + (m-1)^{\frac{p-1}{p}}} \Big)^p$$

Proof for Theorem 9. Theorem 9 gives an upper bound on the multiplicative change when two input datasets with the same size m differ in one entry. To prove, we first consider a slightly different setting when the second dataset is generated by adding an entry to the first dataset. Then the neighboring datasets in the original setting can be generated by adding different entries to the same dataset. Thus, taking the division of the upper and lower bound of the sensitivity in the incremental setting will give an upper bound for sensitivity in the original setting.

Concretely, let $\mathbf{u} = \{u_1, \cdots, u_m\}$ where $u_i > 0, \sum_{i=1}^{m} u_i = s$, $\Delta \geq 0$. We would like to find both upper and lower bounds for the below expression.

$$\frac{\sum_{i=2}^{m} u_i^p + (u_1 + \Delta)^p}{\sum_{i=2}^{m} u_i^p + u_1^p}, \forall p \in (0, 1] \tag{4.1}$$

To bound expression (4.1), we first observe the following two inequalities (4.2) and (4.3).

$$\forall a, b, c, d > 0, a \geq b, c \geq d, \frac{a + c}{a + d} \leq \frac{b + c}{b + d}. \tag{4.2}$$

$$\forall p \in (0, 1], (\sum_{i=1}^{m} u_i)^p \leq \sum_{i=1}^{m} u_i^p \leq m^{1-p} (\sum_{i=1}^{m} u_i)^p \tag{4.3}$$

Inequality (4.2) can be proved with simple algebra. The left-hand-side of inequality (4.3) follows because $\sum_1^m u_i^p$ is concave in $(u_1, \ldots, u_n)$ in the simplex defined by the conditions $u_i \geq 0$ for all i, and $\sum_1^m u_i = s$ and hence the minimum of $\sum_1^m u_i^p$ on the simplex is attained at a vertex of the simplex. The right-hand-side of inequality (4.3) is an instance of the well-known generalized mean inequality [138] or Hölder inequality [64].

First, let's upper bound expression (4.1). According to inequality (4.2) and (4.3),

$$\begin{aligned}
\frac{\sum_{i=2}^{m} u_i^p + (u_1 + \Delta)^p}{\sum_{i=2}^{m} u_i^p + u_1^p} &\overset{(2)+(3)}{\leq} \frac{(\sum_{i=2}^{m} u_i)^p + (u_1 + \Delta)^p}{(\sum_{i=2}^{m} u_i)^p + u_1^p} \\
&= \frac{(s - u_1)^p + (u_1 + \Delta)^p}{(s - u_1)^p + u_1^p} \\
&\overset{(3)}{\leq} 2^{1-p} (1 + \frac{\Delta}{s})^p
\end{aligned}$$

Similarly, to lower bound expression (4.1),

$$\begin{aligned}\frac{\sum_{i=2}^m u_i^p + (u_1+\Delta)^p}{\sum_{i=2}^m u_i^p + u_1^p} &\overset{(2)+(3)}{\geq} \frac{(m-1)^{1-p}(\sum_{i=2}^m u_i)^p + (u_1+\Delta)^p}{(m-1)^{1-p}(\sum_{i=2}^m u_i)^p + u_1^p} \\ &= \frac{(s-u_1)^p + ((m-1)^{\frac{p-1}{p}}(u_1+\Delta))^p}{(s-u_1)^p + ((m-1)^{\frac{p-1}{p}}u_1)^p} \\ &\overset{(3)}{\geq} 2^{p-1}\Big(\frac{((m-1)^{\frac{p-1}{p}}-1)u_1 + s + (m-1)^{\frac{p-1}{p}}\Delta}{((m-1)^{\frac{p-1}{p}}-1)u_1 + s}\Big)^p \\ &\geq 2^{p-1}(1+\frac{(m-1)^{\frac{p-1}{p}}\Delta}{s})^p\end{aligned}$$

Taking the division between the supremum and the infimum, we get

$$\rho_p \leq 2^{2-2p}(\frac{s+M}{s+(m-1)^{\frac{p-1}{p}}})^p \leq 2^{2-2p}(\frac{n-1+M}{n-1+(m-1)^{\frac{p-1}{p}}})^p \qquad \square$$

In a typical streaming model where m is large and $n \gg M$, $\rho_p \lessapprox 2^{2-2p} \leq 4$. To get a better sense of how ρ changes with p, we plot several curves with different hyper-parameters in Figure 4.1. Note that the pure multiplicative sensitivity only depends on n, m, M and p which are public information.

Differentially Private $\mathbb{F}_1$ Sketch

Instead of directly diving into the complete analysis, we first motivate the analysis with the special case of $p = 1$. In this case, the symmetric 1^{st}-stable distribution is the well-known Cauchy distribution: $\mathcal{D}_{1,\zeta}(x) = \frac{1}{\pi} \cdot \frac{\zeta}{\zeta^2+x^2}$, and thus the analyses are significantly simplified. The main purpose of this section is to pave the way for the proof of general $\mathbb{F}_p$ sketch.

Theorem 10 (ϵ-DP for $\mathbb{F}_1$ sketch)**.** *Let ρ_1 represent the multiplicative sensitivity of the first frequency moments. When the size of the sketch $r = 1$, $\mathbb{F}_1$ is $\ln \rho_1$-differentially private.*

Proof for Theorem 10. $\frac{\mathcal{D}_{1,F_1}(x)}{\mathcal{D}_{1,\rho_1 F_1}(x)} = \frac{\rho_1^2 F_1^2 + x^2}{\rho_1(F_1^2+x^2)}$ is a decreasing function of x when $x \in (0,\infty)$ because its derivative $\frac{2(\rho_1-\rho_1^3)F_1^2 x}{\rho_1^2(F_1^2+x^2)^2}$ is non-positive. Thus,

$$\frac{1}{\rho_1} = \frac{\mathcal{D}_{1,F_1}(\infty)}{\mathcal{D}_{1,\rho_1 F_1}(\infty)} \leq \frac{\mathcal{D}_{1,F_1}(x)}{\mathcal{D}_{1,\rho_1 F_1}(x)} \leq \frac{\mathcal{D}_{1,F_1}(0)}{\mathcal{D}_{1,\rho_1 F_1}(0)} = \rho_1$$

Then, for any data stream $\mathcal{S}$ and arbitrary measurable subset s,

$$\begin{aligned}\mathbb{P}[\mathbb{F}_1(\mathcal{S}) \in s] &= \int_{x\in s} \mathcal{D}_{1,F_1(\mathcal{S})}(x)dx = \int_{x\in s} \frac{\mathcal{D}_{1,F_1(\mathcal{S})}(x)}{\mathcal{D}_{1,F_1(\mathcal{S}')}(x)} \mathcal{D}_{1,F_1(\mathcal{S}')}(x)dx \\ &\leq \int_{x\in s} \rho_1 \mathcal{D}_{1,F_1(\mathcal{S}')}(x)dx = e^{\ln\rho_1}\mathbb{P}[\mathbb{F}_1(\mathcal{S}') \in s] \qquad \square\end{aligned}$$

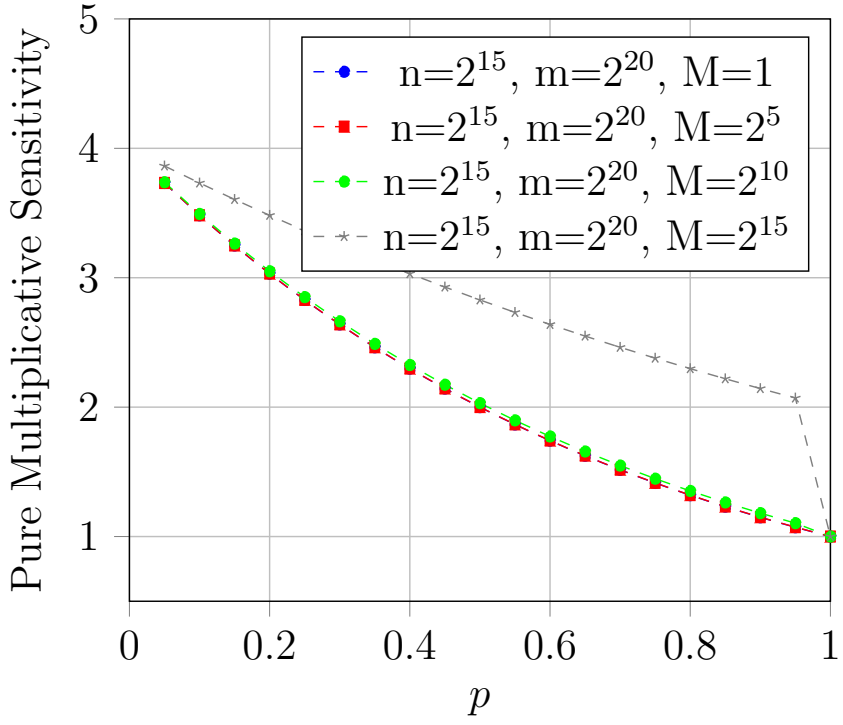


Figure 4.1: Pure multiplicative sensitivity.

Differentially Private $\mathbb{F}_p$ Sketch, $p \in (0, 1]$

The example of $\mathbb{F}_1$ being ϵ-DP indicates the possibility that $\mathbb{F}_p$ might have similar property when $p \in (0, 1]$. To validate, we plot the curves for different values of ps as shown in Figure 4.2. From the figure we can tell that when $p \in (0, 1]$, the ratio $\frac{\mathcal{D}_{p,1}(x)}{\mathcal{D}_{p,2}(x)}$ seems to be well-bounded and preserve ϵ-DP.

We now prove the conjecture as formalized in Theorem 11.

Theorem 11 (ϵ-DP for Algorithm 6)**.** *Let ρ_p represent the multiplicative sensitivity of the p-th frequency moments. When $r = 1$ and $p \in (0, 1]$, $\mathbb{F}_p$ sketch (Algorithm 6) is $\frac{1}{p} \ln \rho_p$-differentially private.*

.

Proof for Theorem 11. To prove Theorem 11, we prove the following inequality.

$$\rho_p^{-\frac{1}{p}} < \rho_p^{-1} \leq \frac{\mathcal{D}_{p,F_p}(x)}{\mathcal{D}_{p,\rho_p F_p}(x)} \leq \rho_p^{\frac{1}{p}}$$

We first prove the right-hand-side of the inequality. Observe that $\zeta^{-\frac{1}{p}} \mathcal{D}_{p,1}(\zeta^{-\frac{1}{p}} x) = \frac{\zeta^{-\frac{1}{p}}}{2\pi} \int_{\mathbb{R}} \cos(\zeta^{-\frac{1}{p}} xt) \exp(-|t|^p) dt \overset{\star}{=} \frac{1}{2\pi} \int_{\mathbb{R}} \cos(xt) \exp(-\zeta|t|^p) dt = \mathcal{D}_{p,\zeta}(x)$ where $\star$ substitutes t

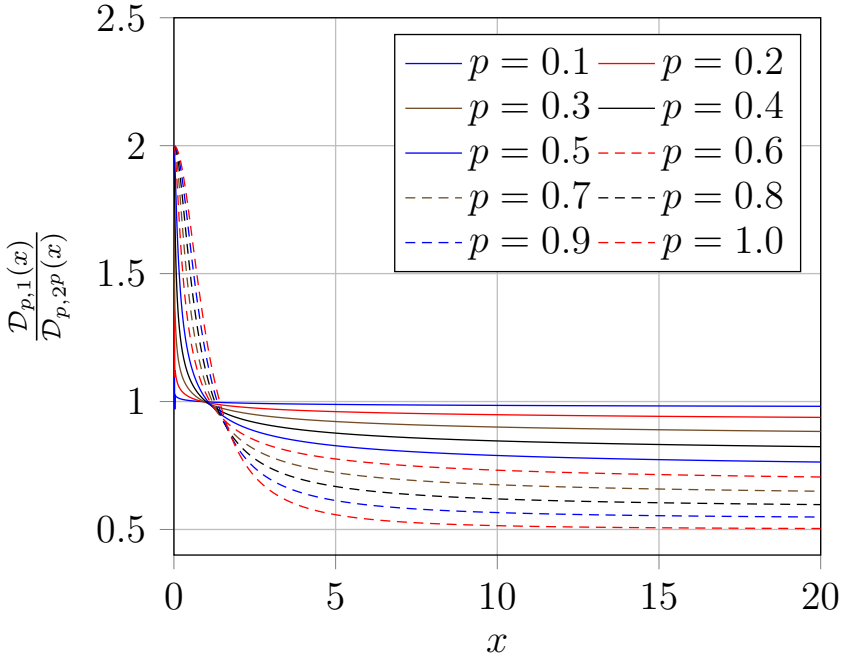


Figure 4.2: The curves of $\frac{\mathcal{D}_{p,1}(x)}{\mathcal{D}_{p,2^p}(x)}$ with different values of $p \in (0, 1]$ on $\mathbb{R}^+$. The negative half is symmetric.

with $\zeta^{\frac{1}{p}} t$ using integration by substitution. Thus,

$$\frac{\mathcal{D}_{p,F_p}(x)}{\mathcal{D}_{p,\rho_p F_p}(x)} = \rho_p^{\frac{1}{p}} \frac{\mathcal{D}_{p,1}(F_p^{-\frac{1}{p}} x)}{\mathcal{D}_{p,1}((\rho_p F_p)^{-\frac{1}{p}} x)} \leq \rho_p^{\frac{1}{p}} \frac{\mathcal{D}_{p,F_p}(0)}{\mathcal{D}_{p,\rho_p F_p}(0)} = \rho_p^{\frac{1}{p}}$$

as $\mathcal{D}_{p,1}$ is increasing on $(-\infty, 0]$ and decreasing on $[0, \infty)$, and $\rho_p \geq 1$.

To prove the left-hand-side of the inequality, we reorganize it into the format of a Fourier transform.

$$\int_0^\infty (\rho_p \exp(-F_p t^p) - \exp(-\rho_p F_p t^p)) \cos(tx) dt \geq 0$$

It suffices to show that

$$h(\rho) = \int_0^\infty \frac{\exp(-\rho F_p t^p)}{\rho} \cos(tx) dt$$

is decreasing. Taking the first derivative of h, we have

$$\frac{\partial h}{\partial \rho} = -\frac{1}{\rho^2} \int_0^\infty g(t) \cos(tx) dt \text{ , where } g(t) = \exp(-\rho F_p t^p)(\rho F_p t^p + 1)$$

According to Pólya criterion [57], it suffices to show that g is positive definite. We first observe that the function $0 \leq u \mapsto (1 + u^{1/2}) e^{-u^{1/2}}$ is the Laplace transform [122] of the

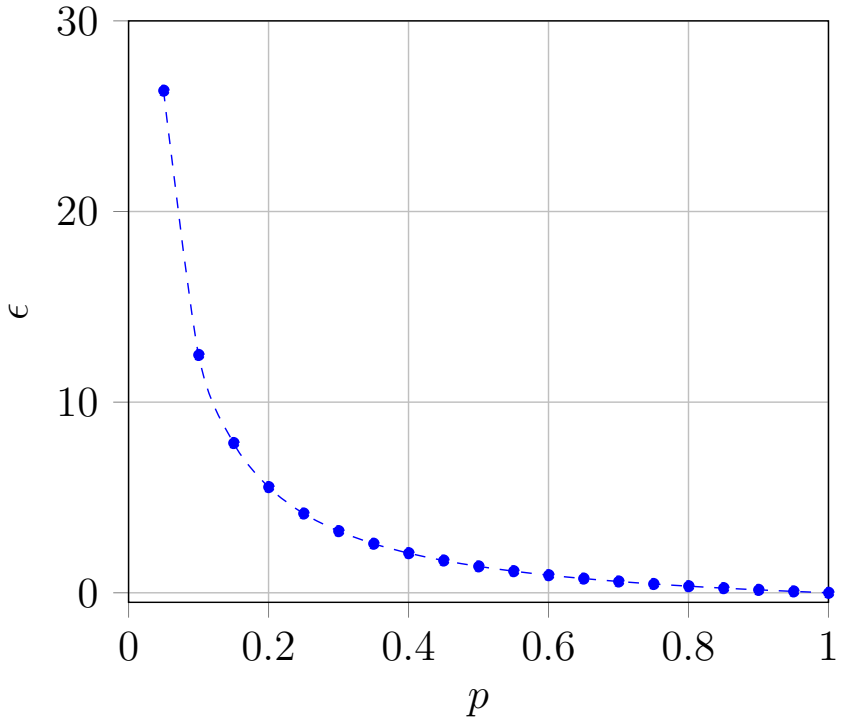


Figure 4.3: Privacy budget ϵ vs. p. $n = 2^{15}, m = 2^{20}, M = 2^4$.

positive function $0 < t \mapsto \frac{e^{-1/(4t)}}{4\sqrt{\pi}\, t^{5/2}}$ (the proof is deferred to the end) and hence a mixture of exponential functions $0 \leq u \mapsto e^{-cu}$ with $c > 0$. Thus with variable substitution, the function $s \mapsto (1 + |s|^p)e^{-|s|^p}$ is a mixture of functions $s \mapsto e^{-c|s|^{2p}}$ with $c > 0$, which are positive definite for any $p \in (0, 1]$ as they are characteristic functions of stable distributions.

The last step is to prove the function $F(u) = (1 + u^{1/2})e^{-u^{1/2}}, u \geq 0$ is the Laplace transform of $f(t) = \frac{e^{-1/(4t)}}{4\sqrt{\pi}\, t^{5/2}}, t > 0$:

$$F(u) = \int_0^\infty f(t)e^{-ut}dt, u \geq 0$$

. Let $R(u) = F(u)$ and $L(u) = \int_0^\infty f(t)e^{-ut}dt$. We observe that $\lim_{x\to\infty} L(x) = \lim_{x\to\infty} R(x) = \lim_{x\to\infty} L'(x) = \lim_{x\to\infty} R'(x) = 0$ so it is enough to show that $L''(u) = R''(u)$. After a simple rescaling, it is enough to show

$$J(a) := \int_0^\infty \exp\Big\{-\frac{1}{t} - at\Big\}\frac{dt}{2\sqrt{t}} = \frac{\sqrt{\pi}}{2}\frac{e^{-2\sqrt{a}}}{\sqrt{a}}$$

where $a > 0$ as both sides do not contain linear terms. Using substitutions $t = u^2$ and then $u = 1/(x\sqrt{a})$, we get

$$J(a) = \int_0^\infty \exp\Big\{ -\frac{1}{u^2} - au^2 \Big\}\, du = K(a)/\sqrt{a},$$

where

$$K(a) := \int_0^\infty \exp\Big\{ -ax^2 - \frac{1}{x^2} \Big\}\, \frac{dx}{x^2}.$$

Note that $K'(a) = -J(a)$ and $K(a) = J(a)\sqrt{a}$. So, we get the differential equation

$$J'(a) = -\Big(\frac{1}{\sqrt{a}} + \frac{1}{2a}\Big) J(a),$$

whose general solution is given by

$$J(a) = \frac{c}{\sqrt{a}}\, e^{-2\sqrt{a}}$$

for a constant c. To determine c, note that

$$K(a) = J(a)\sqrt{a} = \int_0^\infty \exp\Big\{ -\frac{1}{u^2} - au^2 \Big\}\, du\, \sqrt{a} = \int_0^\infty \exp\Big\{ -\frac{a}{y^2} - y^2 \Big\}\, dy$$

and

$$c = K(0+) = \int_0^\infty \exp\{-y^2\}\, dy = \frac{\sqrt{\pi}}{2}.$$

□

Privacy Amplification by Sub-sampling

The last step of Algorithm 6 estimates ζ given samples from the stable distributions. There are many candidate estimators such as the geometric estimator and the harmonic estimator [89, 88]. These estimators typically, as suggested in [89], require at least $r \geq 50$ samples to give an accurate estimation of ζ. However, the privacy parameter ϵ grows linearly with r with trivial composition [39], which might result in too weak privacy protection.

To address, we follow the standard approach, amplifying privacy using sub-sampling. Different from Algorithm 6, each input has probability q to be inserted into each dimension of $\mathbf{a}$, as presented in Algorithm 7. If we take $q = \frac{1}{r}$, then the privacy parameters in Theorem 11 hold as is. The proof is a simple application of the composition theorems [39] and privacy amplification [6].

Theorem 12 (ϵ-DP for Algorithm 7)**.** *Let ρ_p represent the multiplicative sensitivity of the p-th frequency moments. When $p \in (0, 1]$, $\mathbb{F}_p$ sketch with sub-sampling rate q is $\frac{qr}{p} \ln \rho_p$-differentially private.*

Algorithm 7: $\mathbb{F}_p$ sketch with sub-sampling. The only change appears in line 3-4 and 7, corresponding to line 3 and 5 in Algorithm 6. Bernoulli(q) refers to Bernoulli distribution with success probability q.

Input : Data stream: $\mathcal{S} = \{(k_1, v_1), \cdots, (k_n, v_n)\}$
1 **Construction**
2 Initialize $\mathbf{a} = \{0\}^r$, $P \sim \mathcal{D}_{p,1}^{r\times m}$;
3 **for** $i \in [n]$ **do**
4 $b \sim$ Bernoulli(q);
5 Let e_{k_i} be the one-hot encoder of k_i, $\mathbf{a} = \mathbf{a} + \mathbf{P} \times bv_i\mathbf{e}_{k_i}$
6 **Query**
7 return scale_estimator($\mathbf{a}$)$/q^p$;

Utility of Algorithm 7

We depict the accuracy of a F_p estimator with a pair of parameters (γ, η).

Definition 11 ((γ,η)-Accuracy)**.** *A randomized algorithm $\mathcal{M}$ is said to be (γ,η)-accurate if*

$$(1-\gamma)F_p(\mathcal{S}) \le \mathcal{M}(\mathcal{S}) \le (1+\gamma)F_p(\mathcal{S}) \quad w.p. \quad 1-\eta$$

Algorithm 7 satisfies the following utility guarantee. The space complexity is only worse than the optimal non-private algorithm [74] by a logarithmic factor. The accuracy bound is also a worst-case bound and the performance in practice is typically much better (Section 4.3).

Theorem 13 (Utility of Algorithm 7)**.** *$\forall p \in (0,1]$ and $\forall \gamma, \eta \in (0,1)$, Algorithm 7 is $(\gamma + \sqrt{\frac{q^p - q^{2p}}{\lambda}}, \eta + \lambda)$-accurate if $r = \mathcal{O}\big(\gamma^{-2}\log(\frac{1}{\eta})\big)$. In this case, Algorithm 7 uses $\mathcal{O}\big(\gamma^{-2}\log(mM/(\gamma\eta))\log(\frac{1}{\eta})\big)$ bits.*

Proof. Let $\mathcal{SA}_q(\cdot)$ represent the sub-sampling process and $\mathbb{F}_p^r$ represent a $\mathbb{F}_p$ sketch with length r. Then Algorithm 7 can be represented as $\mathbb{F}_p^r \circ \mathcal{SA}_q$ where $\circ$ represents composition of mechanisms.

First, we need the accuracy of $\mathbb{F}_p$ sketch. According to Theorem 4 of [68], if we fix the sub-sampled items,

$$\mathbb{P}[|\mathbb{F}_p^{\mathcal{O}\left(\gamma^{-2}\log(\frac{1}{\eta})\right)} \circ \mathcal{SA}_q(\mathcal{S}) - F_p \circ \mathcal{SA}_q(\mathcal{S})| \le \gamma F_p \circ \mathcal{SA}_q(\mathcal{S})] \ge 1-\eta$$

Second, we need the accuracy of the sub-sampling process. The expectation and variance of the sub-sampling process is as follow.

$$\mathbb{E}[F_p \circ \mathcal{SA}(\mathcal{S})] = q^p F_p(\mathcal{S}) \tag{4.4}$$

$$\begin{aligned}\mathbb{V}[F_p \circ \mathcal{SA}(\mathcal{S})] &= \mathbb{E}[(F_p \circ \mathcal{SA}(\mathcal{S}))^2] - \mathbb{E}^2[F_p \circ \mathcal{SA}(\mathcal{S})] \\ &\leq F_p(\mathcal{S}) \times \mathbb{E}[F_p \circ \mathcal{SA}(\mathcal{S})] - q^{2p}F_p^2(\mathcal{S}) = (q^p - q^2 p)F_p(\mathcal{S})\end{aligned} \tag{4.5}$$

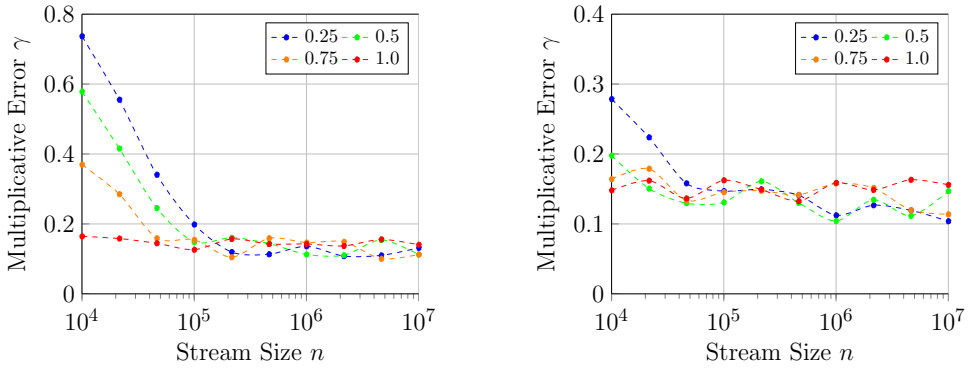
According to Chebyshev's inequality,

$$\mathbb{P}[|F_p \circ \mathcal{SA}(\mathcal{S}) - q^p F_p(\mathcal{S})| \leq \sqrt{\frac{q^p - q^{2p}}{\lambda}} F_p(\mathcal{S})] \geq 1 - \lambda \tag{4.6}$$

Combining (4.4), (4.5) and (4.6) we get Theorem 13. □

4.3 Evaluation

Evaluation Setup



(a) Uniformly Distributed Stream. (b) Binomially Distributed Stream.

Figure 4.4: Results on Synthetic Data.

As we would like to empirically understand $\mathbb{F}_p$ sketch's trade-off between space, error and privacy, we evaluate $\mathbb{F}_p$ with $p \in \{0.25, 0.5, 0.75, 1\}$ using synthetic streams of different sizes and distributions. We also evaluate $\mathbb{F}_p$ with $p \in \{0.05, 0.1, \cdots, 0.95, 1\}$ on real-world data. All the experiments were run on a Ubuntu18.04 LTS server with 32 AMD Opteron(TM) Processor 6212 with 512GB RAM.

Synthetic Data. We first evaluate $\mathbb{F}_p$ sketches using synthetic data. We synthesize two kinds of data: the key domain is either uniformly or binomially distributed. The value domain is $\{1\}$ by default. The size of the key domain is 1000.

Real-world Data. We also evaluate $\mathbb{F}_p$ sketches using real-world application usage data [163] collected by TalkingData SDK. There are more than 30 million events in this dataset, each representing one access to the TalkingData SDK. We view the event type as the key and the value is set to 1 by default.

Evaluation Results

In this section, we present the evaluation results. To avoid the influence of outliers, we report the median and interquartile of 100 runs for each data point except for the real-data evaluation. For all the evaluation, the sketch size r is 50 as suggested in [89]. The sub-sampling rate in all the experiments is 0.02.

Synthetic Data. The evaluation results on synthetic data are presented in Figure 4.4. For uniformly distributed data, we observe that as the stream size increases, the multiplicative error decreases. We conjecture the reason to be the effect of sub-sampling. Concretely, each bin in the value domain has to get enough samples to approximate the behavior of the true distribution. On the other hand, when the data is binomially distributed, the multiplicative error is relatively stable with small fluctuation. We conjecture the reason is that as binomial distribution is more concentrated, the sample complexity is smaller than uniform distribution. Besides, for uniformly distributed data, ps close to 0 have relatively large errors while the errors when p is close to 1 are small. The reason is that the further p is from 1, the larger the influence of sub-sampling.

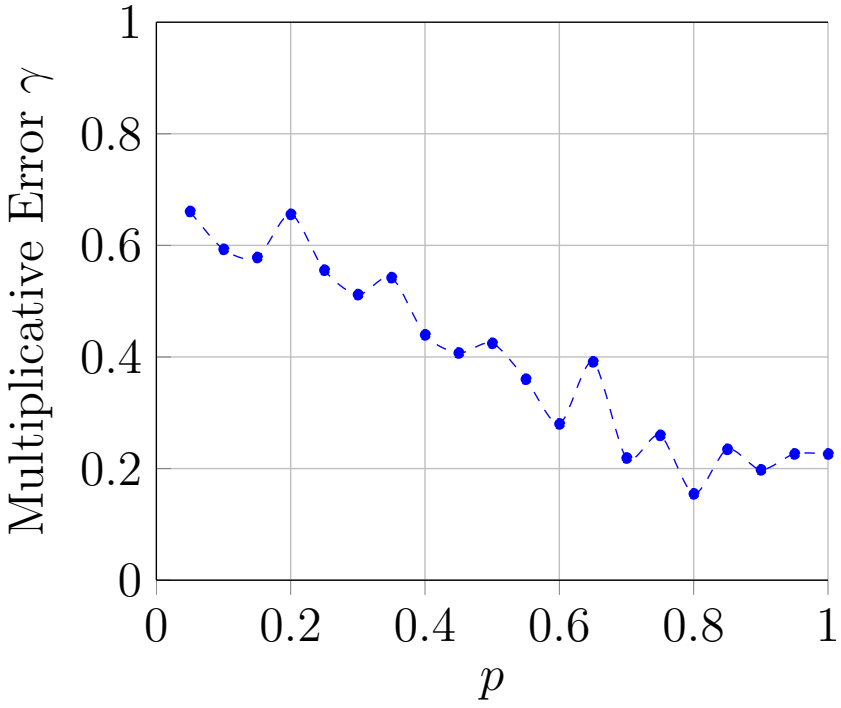


Figure 4.5: Results on Real-world Data.

Real-world Data. The evaluation results for real-world data are presented in Figure 4.5. We sampled 100,0000 data points from the dataset and the key has a domain of size 1488095. Each data point is the median of 5 runs. We observe that the further p is from 1, the higher the multiplicative error. This conforms with our observation in the evaluation on synthetic data.

4.4 Related Work

Frequency moments estimation is thoroughly studied in the data streaming model. Alon *et al.* [3] proposed the first space-efficient algorithm for estimating p^{th} frequency moments when p is integer. Indyk [68] extended the use case from integer moments to fractional moments using stable distributions. A line of following works improve Indyk's algorithm in various aspects such as space complexity [75, 105], time complexity [106, 74] or accuracy [89, 88].

Several special cases in private frequency moments estimation such as $p = 0, 1, 2$ were also well studied. Choi *et al.* [26], Smith *et al.* [130] and Dickens *et al.* [34] studied differentially private F_0 estimation, also known as cardinality estimation. They separately proved that the Flajolet-Martin sketch is differentially private as is. Several independent works [18, 126, 149, 26, 20] studied the differential privacy guarantee in the special case $p = 2$ under the name of Johnson-Lindenstrauss projection.

On the other hand, there is barely any prior work focusing on differentially private fractional frequency moments estimation. Differentially private distribution estimation algorithms [2, 162, 13, 136, 157] can be used to provide a differentially private estimation of fractional frequency moments. However, they are overkill as their outputs contain much more information than the queried fractional frequency moment. They only provide sub-optimal privacy-utility trade-off and are exponentially worse in terms of space complexity.

Datar *et al.* [33] considered a similar (but not the same) mathematical problem to the present work when designing a locality-sensitive hashing scheme. However, their analysis focuses on the simple cases when $p = 1$ and $p = 2$ and totally depends on numerical analysis for $p \in (0, 1)$.

4.5 Discussion

This chapter takes an important step towards narrowing the gap of space complexity between private and non-private frequency moments estimation algorithms. We prove that $\mathbb{F}_p$ is differentially private as is when $p \in (0, 1]$ and thus give the first differentially private frequency estimation protocol with polylogarithmic space complexity.

At the same time, we observe several open challenges. First, the proof does not easily extend to $p \in (1, 2)$. Figure 4.6 exhibits the complexity of monocity of $\frac{\mathcal{D}_{p,1}(x)}{\mathcal{D}_{p,2^p}(x)}$ when $p \in (1, 2)$. The most complex curve when $p = 1.99$ is composed of three monotonic parts in the figure. Hence, an interesting next step is to fully understand the monotonicity pattern

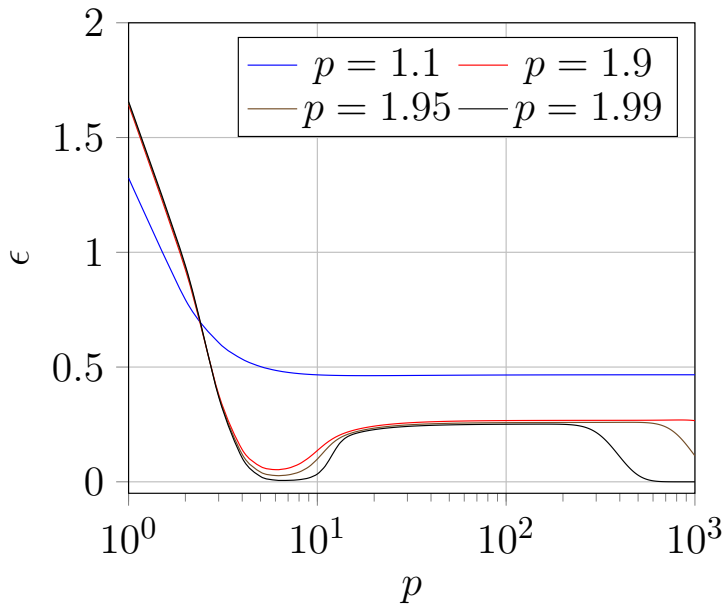


Figure 4.6: The curves of $\frac{\mathcal{D}_{p,1}(x)}{\mathcal{D}_{p,2^p}(x)}$ with different values of $p \in (1,2)$ on $\mathbb{R}^+$. The negative half is symmetric. The x-axis is log-scale to highlight the complex monotone trends.

of the ratio curve when $p \in (1,2)$ and get corresponding privacy parameters. Second, the space complexity of Algorithm 7 is still worse than the optimal non-private algorithm by a factor of $\log(m)$. It is interesting to check whether the optimal algorithm [74] also preserves differential privacy.

Chapter 5

Conclusion

In this dissertation, we present `Aegis`, a privacy-preserving and regulation-compliant data analysis framework. `Aegis` synergizes the top-down and bottom-up efforts towards a practical and cost-efficient privacy-preserving data analysis pipeline. `Aegis` comprises two main components: 1) a privacy regulation enforcement framework; and 2) a library embracing various differentially private functionalities.

In chapter 2, we present PrivGuard, our design for the privacy regulation enforcement framework. PrivGuard integrates a policy-encoding language, a static analyzer and a set of security measures to support regulation compliance verification. The verification process is cost-efficient, versatile, verifiable and secure under external attacks.

For the differential privacy library, we identify two under-explored areas: causal inference and streaming and develop differentially private algorithms for them. In chapter 3, we study the problem of causal graph discovery. We refine the well-known `PC` algorithm to obtain `Priv-PC`, which beats prior works with its outstanding privacy-utility trade-off. In chapter 4, we study a sketching algorithm for frequency moments estimation namely $\mathbb{F}_p$ sketch and prove that it is differentially private as is.

5.1 Future Work

We envision that `Aegis` has the potential to reshape today's privacy-preserving paradigm. However, there are still several grand challenges towards deploying `Aegis` in the real world: (1) the limitation of the threat model of the regulation enforcement framework; (2) the differential privacy library only deals with centralized data which contradicts the trend of decentralization; 3) other bottom-up techniques are not supported.

Dynamic Analysis for Privacy Regulation Enforcement

In PrivGuard, the threat model highlights external attacks but ignores possible internal risks. However, as has been proved many times in the past, most systems are destroyed

from inside. However, because of the dynamic property of Python, it is nearly impossible to use pure static analysis to prevent arbitrary code execution attacks launched by internal personnel. To patch the loophole, we propose to augment PrivGuard with dynamic analysis. Concretely, we plan to look into the Python interpreter, find all the function entrances that might lead to arbitrary code execution, and instrument a code snippet there to prevent suspicious behaviors.

Differentially Private Federated Learning and Analytics

Decentralization is one of the leading trends today in the world of which decentralized data analysis is an essential part. Among all the decentralized data analysis proposals, federated learning is one of the most promising ones because of its low computation cost and excellent performance in practice. How to augment federated learning with differential privacy has received much attention in the past few years and still has a long way to go. Besides learning, how to run other analytics in a federated manner is another important direction to pursue.

A Broader Class of Privacy-Preserving Techniques

Another important direction to pursue is to support a broader class of privacy-preserving techniques beyond differential privacy. For example, federated learning and multiparty-computation are another two promising approaches for privacy-preserving data analysis. The researcher has also conducted research in these directions [155, 171, 95] and plan to integrate them into `Aegis`.

Bibliography

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. "Deep learning with differential privacy". In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. 2016, pp. 308–318.

[2] Gergely Acs, Claude Castelluccia, and Rui Chen. "Differentially private histogram publishing through lossy compression". In: *2012 IEEE 12th International Conference on Data Mining*. IEEE. 2012, pp. 1–10.

[3] Noga Alon, Yossi Matias, and Mario Szegedy. "The space complexity of approximating the frequency moments". In: *Journal of Computer and system sciences* 58.1 (1999), pp. 137–147.

[4] Arthur Azevedo de Amorim, Marco Gaboardi, Justin Hsu, and Shinya Katsumata. "Metric semantics for probabilistic relational reasoning". In: *CoRR, abs/1807.05091* (2018).

[5] Monir Azraoui, Kaoutar Elkhiyaoui, Melek Önen, Karin Bernsmed, Anderson Santana De Oliveira, and Jakub Sendor. "A-PPL: an accountability policy language". In: *Data privacy management, autonomous spontaneous security, and security assurance*. Springer, 2014, pp. 319–326.

[6] Borja Balle, Gilles Barthe, and Marco Gaboardi. "Privacy amplification by subsampling: Tight analyses via couplings and divergences". In: *Advances in Neural Information Processing Systems*. 2018, pp. 6277–6287.

[7] *Bank Customer Churn Predition*. `https://www.kaggle.com/bandiang2/prediction-of-customer-churn-at-a-bank`. Online; accessed 9 January 2020. 2020.

[8] *Bank Customer Classification Random Forest*. `https://www.kaggle.com/taronzakaryan/bank-customer-classification-random-forest`. Online; accessed 9 January 2020. 2020.

[9] *Bank Customer Segmentation*. `https://www.kaggle.com/paulinan/bank-customer-segmentation`. Online; accessed 9 January 2020. 2020.

[10] Gilles Barthe, Marco Gaboardi, Emilio Jesús Gallego Arias, Justin Hsu, Aaron Roth, and Pierre-Yves Strub. "Higher-order approximate relational refinement types for mechanism design and differential privacy". In: *ACM SIGPLAN Notices* 50.1 (2015), pp. 55–68.

[11] Raef Bassily, Vitaly Feldman, Cristóbal Guzmán, and Kunal Talwar. "Stability of stochastic gradient descent on nonsmooth convex losses". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 4381–4391.

[12] Raef Bassily, Vitaly Feldman, Kunal Talwar, and Abhradeep Guha Thakurta. "Private stochastic convex optimization with optimal rates". In: *Advances in Neural Information Processing Systems* 32 (2019).

[13] Raef Bassily and Adam Smith. "Local, private, efficient protocols for succinct histograms". In: *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*. 2015, pp. 127–135.

[14] Raef Bassily, Adam Smith, and Abhradeep Thakurta. "Private empirical risk minimization: Efficient algorithms and tight error bounds". In: *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*. IEEE. 2014, pp. 464–473.

[15] Ingo A Beinlich, Henri Jacques Suermondt, R Martin Chavez, and Gregory F Cooper. "The ALARM monitoring system: A case study with two probabilistic inference techniques for belief networks". In: *AIME 89*. Springer, 1989, pp. 247–256.

[16] JO Berger. *On the development of reference priors, Bayesian Statistics 4 (JM Bernardo, JO Berger, AP Dawid and AFM Smith, eds.)* 1992.

[17] Garrett Bernstein and Daniel R Sheldon. "Differentially private bayesian inference for exponential families". In: *Advances in Neural Information Processing Systems*. 2018, pp. 2919–2929.

[18] Jeremiah Blocki, Avrim Blum, Anupam Datta, and Or Sheffet. "The johnson-lindenstrauss transform itself preserves differential privacy". In: *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*. IEEE. 2012, pp. 410–419.

[19] Leo Breiman. "Random forests". In: *Machine learning* 45.1 (2001), pp. 5–32.

[20] Zhiqi Bu, Sivakanth Gopi, Janardhan Kulkarni, Yin Tat Lee, Judy Hanwen Shen, and Uthaipon Tantipongpipat. "Fast and Memory Efficient Differentially Private-SGD via JL Projections". In: *arXiv preprint arXiv:2102.03013* (2021).

[21] Mark Bun and Thomas Steinke. "Concentrated differential privacy: Simplifications, extensions, and lower bounds". In: *Theory of Cryptography Conference*. Springer. 2016, pp. 635–658.

[22] Moses Charikar, Kevin Chen, and Martin Farach-Colton. "Finding frequent items in data streams". In: *International Colloquium on Automata, Languages, and Programming*. Springer. 2002, pp. 693–703.

[23] Kamalika Chaudhuri, Claire Monteleoni, and Anand D Sarwate. "Differentially private empirical risk minimization." In: *Journal of Machine Learning Research* 12.3 (2011).

[24] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Xiaodong Song. "Targeted Backdoor Attacks on Deep Learning Systems Using Data Poisoning". In: *ArXiv* abs/1712.05526 (2017).

[25] David Maxwell Chickering. "Optimal structure identification with greedy search". In: *Journal of machine learning research* 3.Nov (2002), pp. 507–554.

[26] Seung Geol Choi, Dana Dachman-Soled, Mukul Kulkarni, and Arkady Yerukhimovich. "Differentially-private multi-party sketching for large-scale statistics". In: *Proceedings on Privacy Enhancing Technologies* 2020.3 (2020), pp. 153–174.

[27] Omar Chowdhury, Andreas Gampe, Jianwei Niu, Jeffery von Ronne, Jared Bennatt, Anupam Datta, Limin Jia, and William H Winsborough. "Privacy promises that can be kept: a policy analysis method with application to the HIPAA privacy rule". In: *Proceedings of the 18th ACM symposium on Access control models and technologies*. ACM. 2013, pp. 3–14.

[28] Omar Chowdhury, Limin Jia, Deepak Garg, and Anupam Datta. "Temporal mode-checking for runtime monitoring of privacy policies". In: *International Conference on Computer Aided Verification*. Springer. 2014, pp. 131–149.

[29] Kenneth L Clarkson and David P Woodruff. "Numerical linear algebra in the streaming model". In: *Proceedings of the forty-first annual ACM symposium on Theory of computing*. 2009, pp. 205–214.

[30] *Classify Forest Categories*. `https://www.kaggle.com/suyashgulati/using-pyspark-randomforest-crossvalidatn-gridsrch`. Online; accessed 9 January 2020. 2020.

[31] Graham Cormode, S Muthukrishnan, and Irina Rozenbaum. "Summarizing and mining inverse distributions on data streams via dynamic inverse sampling". In: *VLDB*. Vol. 5. 2005, pp. 25–36.

[32] *Credit Risk Analysis*. `https://www.kaggle.com/damaradiprabowo/clustering-german-credit-data`. Online; accessed 9 January 2020. 2020.

[33] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S Mirrokni. "Locality-sensitive hashing scheme based on p-stable distributions". In: *Proceedings of the twentieth annual symposium on Computational geometry*. 2004, pp. 253–262.

[34] Charlie Dickens, Justin Thaler, and Daniel Ting. "(Nearly) All Cardinality Estimators Are Differentially Private". In: *arXiv preprint arXiv:2203.15400* (2022).

[35] Christos Dimitrakakis, Blaine Nelson, Aikaterini Mitrokotsa, and Benjamin IP Rubinstein. "Robust and private Bayesian inference". In: *International Conference on Algorithmic Learning Theory*. Springer. 2014, pp. 291–305.

[36] Bolin Ding, Janardhan Kulkarni, and Sergey Yekhanin. "Collecting telemetry data privately". In: *Advances in Neural Information Processing Systems* 30 (2017).

[37] *Display Advertising Challenge*. `https://www.kaggle.com/c/criteo-display-ad-challenge/discussion/10322`. Online; accessed 9 January 2020. 2020.

[38] Jinshuo Dong, Aaron Roth, and Weijie J Su. "Gaussian differential privacy". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* (2021).

[39] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. "Calibrating noise to sensitivity in private data analysis". In: *Theory of cryptography conference*. Springer. 2006, pp. 265–284.

[40] Cynthia Dwork, Moni Naor, Toniann Pitassi, Guy N Rothblum, and Sergey Yekhanin. "Pan-Private Streaming Algorithms." In: *ics*. 2010, pp. 66–80.

[41] Cynthia Dwork, Aaron Roth, et al. "The algorithmic foundations of differential privacy". In: *Foundations and Trends® in Theoretical Computer Science* 9.3–4 (2014), pp. 211–407.

[42] Cynthia Dwork and Guy N Rothblum. "Concentrated differential privacy". In: *arXiv preprint arXiv:1603.01887* (2016).

[43] Cynthia Dwork, Weijie Su, and Li Zhang. "Private false discovery rate control". In: *arXiv preprint arXiv:1511.03803* (2015).

[44] Hamid Ebadi, David Sands, and Gerardo Schneider. "Differential privacy: Now it's getting personal". In: *Acm Sigplan Notices* 50.1 (2015), pp. 69–81.

[45] *Elo Merchant Category Recommendation*. `https://www.kaggle.com/c/elo-merchant-category-recommendation`. Online; accessed 9 January 2020. 2020.

[46] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. "Rappor: Randomized aggregatable privacy-preserving ordinal response". In: *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*. 2014, pp. 1054–1067.

[47] Joan Feigenbaum, Sampath Kannan, Martin J Strauss, and Mahesh Viswanathan. "An approximate L 1-difference algorithm for massive data streams". In: *SIAM Journal on Computing* 32.1 (2002), pp. 131–151.

[48] David Ferraiolo, Ramaswamy Chandramouli, Rick Kuhn, and Vincent Hu. "Extensible access control markup language (XACML) and next generation access control (NGAC)". In: *Proceedings of the 2016 ACM International Workshop on Attribute Based Access Control*. 2016, pp. 13–24.

[49] *Formal concept analysis*. `https://en.wikipedia.org/wiki/Formal_concept_analysis`. Online; accessed 30 May 2019. 2019.

[50] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. "Model Inversion Attacks that Exploit Confidence Information and Basic Countermeasures". In: *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-16, 2015*. Ed. by Indrajit Ray, Ninghui Li, and Christopher Kruegel. ACM, 2015, pp. 1322–1333. DOI: 10.1145/2810103.2813677. URL: https://doi.org/10.1145/2810103.2813677.

[51] Marco Gaboardi, Andreas Haeberlen, Justin Hsu, Arjun Narayan, and Benjamin C Pierce. "Linear dependent types for differential privacy". In: *Proceedings of the 40th annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages*. 2013, pp. 357–370.

[52] Davi Geiger, Tyng-Luh Liu, and Michael J Donahue. "Sparse representations for image decompositions". In: *International Journal of Computer Vision* 33.2 (1999), pp. 139–156.

[53] Armin Gerl, Nadia Bennani, Harald Kosch, and Lionel Brunie. "LPL, Towards a GDPR-Compliant Privacy Language: Formal Definition and Usage". In: *Transactions on Large-Scale Data-and Knowledge-Centered Systems XXXVII*. Springer, 2018, pp. 41–80.

[54] Daniel B Giffin, Amit Levy, Deian Stefan, David Terei, David Mazières, John C Mitchell, and Alejandro Russo. "Hails: Protecting data privacy in untrusted web applications". In: *Presented as part of the 10th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 12)*. 2012, pp. 47–60.

[55] Corrado Gini. "Variabilità e mutabilità". In: *Reprinted in Memorie di metodologica statistica (Ed. Pizetti E* (1912).

[56] Clark Glymour, Kun Zhang, and Peter Spirtes. "Review of causal discovery methods based on graphical models". In: *Frontiers in Genetics* 10 (2019).

[57] Tilmann Gneiting. "Criteria of PÃglya type for radial positive definite functions". In: *Proceedings of the American Mathematical Society* 129.8 (2001), pp. 2309–2318.

[58] IJ Good. "C332. Surprise indexes and p-values". In: (1989).

[59] *Google Cloud & NCAA ML Competition 2019-Women's*. https://www.kaggle.com/c/womens-machine-learning-competition-2019/discussion/90156. Online; accessed 9 January 2020. 2020.

[60] Wenbo Guo, Lun Wang, Yan Xu, Xinyu Xing, Min Du, and Dawn Song. "Towards inspecting and eliminating trojan backdoors in deep neural networks". In: *2020 IEEE International Conference on Data Mining (ICDM)*. IEEE. 2020, pp. 162–171.

[61] Nicholas JA Harvey, Jelani Nelson, and Krzysztof Onak. "Sketching and streaming entropy via approximation theory". In: *2008 49th Annual IEEE Symposium on Foundations of Computer Science*. IEEE. 2008, pp. 489–498.

[62] *Heart Disease Causal Inference*. `https://www.kaggle.com/tentotheminus9/what-causes-heart-disease-explaining-the-model`. Online; accessed 9 January 2020. 2020.

[63] Mikko Heikkilä, Eemil Lagerspetz, Samuel Kaski, Kana Shimizu, Sasu Tarkoma, and Antti Honkela. "Differentially private Bayesian learning on distributed data". In: *Advances in neural information processing systems*. 2017, pp. 3226–3235.

[64] Otto Hölder. "Ueber einen mittelwerthabsatz". In: *Nachrichten von der Königl. Gesellschaft der Wissenschaften und der Georg-Augusts-Universität zu Göttingen* 1889 (1889), pp. 38–47.

[65] Naoise Holohan, Stefano Braghin, Pól Mac Aonghusa, and Killian Levacher. "Diffprivlib: The IBM Differential Privacy Library". In: *arXiv preprint arXiv:1907.02444* (2019).

[66] Patrik O Hoyer, Dominik Janzing, Joris M Mooij, Jonas Peters, and Bernhard Schölkopf. "Nonlinear causal discovery with additive noise models". In: *Advances in neural information processing systems*. 2009, pp. 689–696.

[67] *IEEE CIS Fraud Detection*. `https://www.kaggle.com/c/ieee-fraud-detection`. Online; accessed 9 January 2020. 2020.

[68] Piotr Indyk. "Stable distributions, pseudorandom generators, embeddings, and data stream computation". In: *Journal of the ACM (JACM)* 53.3 (2006), pp. 307–323.

[69] Piotr Indyk and Andrew McGregor. "Declaring independence via the sketching of sketches." In: *SODA*. Vol. 8. 2008, pp. 737–745.

[70] Roger Iyengar, Joseph P Near, Dawn Song, Om Thakkar, Abhradeep Thakurta, and Lun Wang. "Towards practical differentially private convex optimization". In: *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2019, pp. 299–316.

[71] Aaron Johnson and Vitaly Shmatikov. "Privacy-preserving data exploration in genome-wide association studies". In: *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2013, pp. 1079–1087.

[72] *K-Anonymity Library*. `https://github.com/KENNN/k-anonymity`. Online; accessed 02 May 2020. 2020.

[73] Peter Kairouz, Brendan McMahan, Shuang Song, Om Thakkar, Abhradeep Thakurta, and Zheng Xu. "Practical and private (deep) learning without sampling or shuffling". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 5213–5225.

[74] Daniel M Kane, Jelani Nelson, Ely Porat, and David P Woodruff. "Fast moment estimation in data streams in optimal space". In: *Proceedings of the forty-third annual ACM symposium on Theory of computing*. 2011, pp. 745–754.

[75] Daniel M Kane, Jelani Nelson, and David P Woodruff. "On the exact space complexity of sketching and streaming small norms". In: *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*. SIAM. 2010, pp. 1161–1178.

[76] Maurice G Kendall. "A new measure of rank correlation". In: *Biometrika* 30.1/2 (1938), pp. 81–93.

[77] Krishnaram Kenthapadi and Thanh TL Tran. "Pripearl: A framework for privacy-preserving analytics and reporting at linkedin". In: *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. 2018, pp. 2183–2191.

[78] Kevin B Korb and Ann E Nicholson. *Bayesian artificial intelligence*. CRC press, 2010.

[79] Edward L Korn. "Kendall's tau with a blocking variable". In: *Biometrics* (1984), pp. 209–214.

[80] Antti Koskela, Joonas Jälkö, and Antti Honkela. "Computing tight differential privacy guarantees using fft". In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2020, pp. 2560–2569.

[81] Balachander Krishnamurthy, Subhabrata Sen, Yin Zhang, and Yan Chen. "Sketch-based change detection: Methods, evaluation, and applications". In: *Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*. 2003, pp. 234–247.

[82] Matt J Kusner, Yu Sun, Karthik Sridharan, and Kilian Q Weinberger. "Private causal inference". In: *arXiv preprint arXiv:1512.05469* (2015).

[83] Peifung E Lam, John C Mitchell, Andre Scedrov, Sharada Sundaram, and Frank Wang. "Declarative privacy policy: finite models and attribute-based encryption". In: *Proceedings of the 2nd ACM SIGHIT International Health Informatics Symposium*. ACM. 2012, pp. 323–332.

[84] *LANL Earthquake Prediction*. `https://www.kaggle.com/c/LANL-Earthquake-Prediction/discussion/94390`. Online; accessed 9 January 2020. 2020.

[85] Steffen L Lauritzen and David J Spiegelhalter. "Local computations with probabilities on graphical structures and their application to expert systems". In: *Journal of the Royal Statistical Society: Series B (Methodological)* 50.2 (1988), pp. 157–194.

[86] *Learning with privacy at scale*. 2017.

[87] Ninghui Li, Tiancheng Li, and Suresh Venkatasubramanian. "t-closeness: Privacy beyond k-anonymity and l-diversity". In: *2007 IEEE 23rd International Conference on Data Engineering*. IEEE. 2007, pp. 106–115.

[88] Ping Li. "Compressed counting". In: *Proceedings of the twentieth annual ACM-SIAM symposium on Discrete algorithms*. SIAM. 2009, pp. 412–421.

[89] Ping Li. "Estimators and tail bounds for dimension reduction in ℓ_α $(0 < \alpha \leq 2)$ using stable random projections". In: *Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*. 2008, pp. 10–19.

[90] Max O Lorenz. "Methods of measuring the concentration of wealth". In: *Publications of the American statistical association* 9.70 (1905), pp. 209–219.

[91] Min Lyu, Dong Su, and Ninghui Li. "Understanding the sparse vector technique for differential privacy". In: *arXiv preprint arXiv:1603.01699* (2016).

[92] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. "l-diversity: Privacy beyond k-anonymity". In: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 1.1 (2007), 3–es.

[93] Petros Maniatis, Devdatta Akhawe, Kevin R Fall, Elaine Shi, and Dawn Song. "Do You Know Where Your Data Are? Secure Data Capsules for Deployable Data Protection." In: *HotOS*. Vol. 7. 2011, pp. 193–205.

[94] Essam Mansour, Andrei Vlad Sambra, Sandro Hawke, Maged Zereba, Sarven Capadisli, Abdurrahman Ghanem, Ashraf Aboulnaga, and Tim Berners-Lee. "A demonstration of the solid platform for social web applications". In: *Proceedings of the 25th International Conference Companion on World Wide Web*. International World Wide Web Conferences Steering Committee. 2016, pp. 223–226.

[95] Sai Krishna Deepak Maram, Fan Zhang, Lun Wang, Andrew Low, Yupeng Zhang, Ari Juels, and Dawn Song. "Churp: Dynamic-committee proactive secret sharing". In: *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 2019, pp. 2369–2386.

[96] John H McDonald. *Handbook of biological statistics*. Vol. 2. sparky house publishing Baltimore, MD, 2009.

[97] Mary L McHugh. "The chi-square test of independence". In: *Biochemia medica: Biochemia medica* 23.2 (2013), pp. 143–149.

[98] Ryan McKenna, Gerome Miklau, Michael Hay, and Ashwin Machanavajjhala. "Optimizing error of high-dimensional statistical queries under differential privacy". In: *Proceedings of the VLDB Endowment* 11.10 (2018), pp. 1206–1219.

[99] *Microsoft Malware Prediction*. `https://www.kaggle.com/shrutimechlearn/large-data-loading-trick-with-ms-malware-data`. Online; accessed 9 January 2020. 2020.

[100] Darakhshan Mir, Shan Muthukrishnan, Aleksandar Nikolov, and Rebecca N Wright. "Pan-private algorithms via statistics on sketches". In: *Proceedings of the thirtieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. 2011, pp. 37–48.

[101] Ilya Mironov. "Rényi differential privacy". In: *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*. IEEE. 2017, pp. 263–275.

[102] Reinhard Munz, Fabienne Eigner, Matteo Maffei, Paul Francis, and Deepak Garg. "UniTraX: protecting data privacy with discoverable biases". In: *International Conference on Principles of Security and Trust*. Springer, Cham. 2018, pp. 278–299.

[103] Andrew C Myers. "JFlow: Practical mostly-static information flow control". In: *Proceedings of the 26th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*. 1999, pp. 228–241.

[104] Joseph P Near, David Darais, Chike Abuah, Tim Stevens, Pranav Gaddamadugu, Lun Wang, Neel Somani, Mu Zhang, Nikhil Sharma, Alex Shan, et al. "Duet: an expressive higher-order language and linear type system for statically enforcing differential privacy". In: *Proceedings of the ACM on Programming Languages* 3.OOPSLA (2019), pp. 1–30.

[105] Jelani Nelson and David P Woodruff. "A near-optimal algorithm for L1-difference". In: *arXiv preprint arXiv:0904.2027* (2009).

[106] Jelani Nelson and David P Woodruff. "Fast manhattan sketches in data streams". In: *Proceedings of the twenty-ninth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. 2010, pp. 99–110.

[107] *NFL Punt Analytics Competition*. `https://www.kaggle.com/c/NFL-Punt-Analytics-Competition/discussion/78041#latest-663557`. Online; accessed 9 January 2020. 2020.

[108] Flemming Nielson, Hanne R Nielson, and Chris Hankin. *Principles of program analysis*. Springer, 2015.

[109] Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

[110] *Oasis Labs Parcel SDK*. `https://www.oasislabs.com/parcelsdk`. Accessed: 2021-02-02.

[111] Julian A Padget and Wamberto W Vasconcelos. "Fine-grained access control via policy-carrying data". In: *ACM Transactions on Internet Technology (TOIT)* 18.3 (2018), pp. 1–24.

[112] Nicolas Papernot, Shuang Song, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Úlfar Erlingsson. "Scalable private learning with pate". In: *arXiv preprint arXiv:1802.08908* (2018).

[113] Judea Pearl et al. "Causal inference in statistics: An overview". In: *Statistics surveys* 3 (2009), pp. 96–146.

[114] Jason Reed and Benjamin C Pierce. "Distance makes the types grow stronger: A calculus for differential privacy". In: *Proceedings of the 15th ACM SIGPLAN international conference on Functional programming*. 2010, pp. 157–168.

[115] *Restaurant Revenue Prediction*. `https://www.kaggle.com/jquesadar/restaurant-revenue-1st-place-solution`. Online; accessed 9 January 2020. 2020.

[116] Andrei Sabelfeld and Alejandro Russo. "From dynamic to static and back: Riding the roller coaster of information-flow control research". In: *International Andrei Ershov Memorial Conference on Perspectives of System Informatics*. Springer. 2009, pp. 352–365.

[117] Karen Sachs, Omar Perez, Dana Pe'er, Douglas A Lauffenburger, and Garry P Nolan. "Causal protein-signaling networks derived from multiparameter single-cell data". In: *Science* 308.5721 (2005), pp. 523–529.

[118] *Santander Customer Satisfaction*. `https://www.kaggle.com/c/santander-customer-satisfaction`. Online; accessed 9 January 2020. 2020.

[119] *Santander Customer Transaction Prediction*. `https://www.kaggle.com/c/santander-customer-transaction-prediction`. Online; accessed 9 January 2020. 2020.

[120] Tamas Sarlos. "Improved approximation algorithms for large matrices via random projections". In: *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*. IEEE. 2006, pp. 143–152.

[121] Stefan Saroiu, Alec Wolman, and Sharad Agarwal. "Policy-carrying data: A privacy abstraction for attaching terms of service to mobile data". In: *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications*. 2015, pp. 129–134.

[122] Joel L Schiff. *The Laplace transform: theory and applications*. Springer Science & Business Media, 1999.

[123] Eva Schlehahn, Patrick Murmann, Farzaneh Karegar, and Simone Fischer-Hübner. "Opportunities and Challenges of Dynamic Consent in Commercial Big Data Analytics". In: *IFIP International Summer School on Privacy and Identity Management*. Springer. 2019, pp. 29–44.

[124] Marco Scutari and Jean-Baptiste Denis. *Bayesian networks: with examples in R*. CRC press, 2014.

[125] Shayak Sen, Saikat Guha, Anupam Datta, Sriram K Rajamani, Janice Tsai, and Jeannette M Wing. "Bootstrapping privacy compliance in big data systems". In: *2014 IEEE Symposium on Security and Privacy*. IEEE. 2014, pp. 327–342.

[126] Or Sheffet. "Differentially private ordinary least squares". In: *International Conference on Machine Learning*. PMLR. 2017, pp. 3105–3114.

[127] Shohei Shimizu, Takanori Inazumi, Yasuhiro Sogawa, Aapo Hyvärinen, Yoshinobu Kawahara, Takashi Washio, Patrik O Hoyer, and Kenneth Bollen. "DirectLiNGAM: A direct method for learning a linear non-Gaussian structural equation model". In: *Journal of Machine Learning Research* 12.Apr (2011), pp. 1225–1248.

[128] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. "Membership Inference Attacks Against Machine Learning Models". In: *2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017*. IEEE Computer Society, 2017, pp. 3–18. DOI: 10.1109/SP.2017.41. URL: https://doi.org/10.1109/SP.2017.41.

[129] *Simple LSTM - PyTorch version*. https://www.kaggle.com/bminixhofer/simple-lstm-pytorch-version. Online; accessed 9 January 2020. 2020.

[130] Adam Smith, Shuang Song, and Abhradeep Thakurta. "The Flajolet-Martin Sketch Itself Preserves Differential Privacy: Private Counting with Minimal Space". In: *Advances in Neural Information Processing Systems* 33 (2020).

[131] David M Sommer, Sebastian Meiser, and Esfandiar Mohammadi. "Privacy loss classes: The central limit theorem in differential privacy". In: *Proceedings on privacy enhancing technologies* 2019.2 (2019), pp. 245–269.

[132] Charles Spearman. "The proof and measurement of association between two things." In: (1961).

[133] *SPECIAL: Scalable Policy-awarE linked data arChitecture for prIvacy, trAnsparency and CompLiance*. https://specialprivacy.ercim.eu/. Accessed: 2021-05-04.

[134] Peter Spirtes, Clark N Glymour, Richard Scheines, and David Heckerman. *Causation, prediction, and search*. MIT press, 2000.

[135] *Statistical Safeguards*. https://www.census.gov/about/policies/privacy/statistical_safeguards.html#:~:text=The%202020%20Census%20will%20use,threats%20in%20today's%20digital%20world.. Online; accessed 03 June 2022. 2022.

[136] Ananda Theertha Suresh. "Differentially private anonymized histograms". In: *arXiv preprint arXiv:1910.03553* (2019).

[137] Latanya Sweeney. "k-anonymity: A model for protecting privacy". In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 10.05 (2002), pp. 557–570.

[138] Stanislav Sykora. "Mathematical Means and Averages: Basic Properties". PhD thesis. Ed. S. Sykora, 2009.

[139] *TalkingData AdTracking Fraud Detection Challenge*. https://www.kaggle.com/c/talkingdata-adtracking-fraud-detection/discussion/56283. Online; accessed 9 January 2020. 2020.

[140] Jeremy MG Taylor. "Kendall's and Spearman's correlation coefficients in the presence of a blocking variable". In: *Biometrics* (1987), pp. 409–416.

[141] *TensorFlow Privacy*. https://github.com/tensorflow/privacy. Online; accessed 03 May 2020. 2020.

[142] *Tensorflow-Deep Learning to Solve Titanic*. https://www.kaggle.com/linxinzhe/tensorflow-deep-learning-to-solve-titanic. Online; accessed 9 January 2020. 2020.

[143] *The Age of Privacy: The Cost of Continuous Compliance*. https://datagrail.io/downloads/GDPR-CCPA-cost-report.pdf. Online; accessed 30 July 2020. 2020.

[144] *The GDPR Racket: Who's Making Money From This $9bn Business Shakedown*. https://www.forbes.com/sites/oliversmith/2018/05/02/the-gdpr-racket-whos-making-money-from-this-9bn-business-shakedown/#54c0702034a2. Online; accessed 30 July 2020. 2020.

[145] *The Looming Cost of a Patchwork of State Privacy Laws*. https://itif.org/publications/2022/01/24/looming-cost-patchwork-state-privacy-laws#:~:text=For%20example%2C%20ITIF%20has%20estimated,state%20small%20businesses%20face%20%246. Online; accessed 03 June 2022. 2022.

[146] Mikkel Thorup and Yin Zhang. "Tabulation based 4-universal hashing with applications to second moment estimation." In: *SODA*. Vol. 4. 2004, pp. 615–624.

[147] Slim Trabelsi, Jakub Sendor, and Stefanie Reinicke. "Ppl: Primelife privacy policy engine". In: *2011 IEEE International Symposium on Policies for Distributed Systems and Networks*. IEEE. 2011, pp. 184–185.

[148] Michael Carl Tschantz, Anupam Datta, and Jeannette M Wing. "Formalizing and enforcing purpose restrictions in privacy policies". In: *2012 IEEE Symposium on Security and Privacy*. IEEE. 2012, pp. 176–190.

[149] Jalaj Upadhyay. "Differentially private linear algebra in the streaming model". In: *arXiv preprint arXiv:1409.5414* (2014).

[150] Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y. Zhao. "Neural Cleanse: Identifying and Mitigating Backdoor Attacks in Neural Networks". In: *2019 IEEE Symposium on Security and Privacy (SP)* (2019), pp. 707–723.

[151] Frank Wang, Ronny Ko, and James Mickens. "Riverbed: enforcing user-defined privacy constraints in distributed web services". In: *16th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 19)*. 2019, pp. 615–630.

[152] Lun Wang, Zaynah Javed, Xian Wu, Wenbo Guo, Xinyu Xing, and Dawn Song. "Backdoorl: Backdoor attack against competitive reinforcement learning". In: *arXiv preprint arXiv:2105.00579* (2021).

[153] Lun Wang, Usmann Khan, Joseph Near, Qi Pang, Jithendaraa Subramanian, Neel Somani, Peng Gao, Andrew Low, and Dawn Song. "PRIVGUARD: Privacy Regulation Compliance Made Easier". In: *31th {USENIX} Security Symposium ({USENIX} Security 22)* (2022).

[154] Lun Wang, Qi Pang, and Dawn Song. "Towards practical differentially private causal graph discovery". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 5516–5526.

[155] Lun Wang, Qi Pang, Shuai Wang, and Dawn Song. "FED-chi2: Privacy Preserving Federated Correlation Test". In: *arXiv preprint arXiv:2105.14618* (2021).

[156] Lun Wang, Iosif Pinelis, and Dawn Song. "Differentially Private Fractional Frequency Moments Estimation with Polylogarithmic Space". In: *The Tenth International Conference on Learning Representations* (2022).

[157] Tianhao Wang, Milan Lopuhaä-Zwakenberg, Zitao Li, Boris Skoric, and Ninghui Li. "Locally differentially private frequency estimation with consistency". In: *arXiv preprint arXiv:1905.08320* (2019).

[158] Yu-Xiang Wang, Borja Balle, and Shiva Prasad Kasiviswanathan. "Subsampled rényi differential privacy and analytical moments accountant". In: *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR. 2019, pp. 1226–1235.

[159] *Web traffic time series forecasting*. `https://www.kaggle.com/muonneutrino/wikipedia-traffic-data-exploration`. Online; accessed 9 January 2020. 2020.

[160] *What Is IAM?* `https://docs.aws.amazon.com/IAM/latest/UserGuide/introduction.html`. Online; accessed 29 April 2020. 2020.

[161] Depeng Xu, Shuhan Yuan, and Xintao Wu. "Differential privacy preserving causal graph discovery". In: *2017 IEEE Symposium on Privacy-Aware Computing (PAC)*. IEEE. 2017, pp. 60–71.

[162] Jia Xu, Zhenjie Zhang, Xiaokui Xiao, Yin Yang, Ge Yu, and Marianne Winslett. "Differentially private histogram publication". In: *The VLDB Journal* 22.6 (2013), pp. 797–822.

[163] Qingqing Ye, Haibo Hu, Xiaofeng Meng, and Huadi Zheng. "PrivKV: Key-value data collection with local differential privacy". In: *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2019, pp. 317–331.

[164] Dong Yin, Yudong Chen, Kannan Ramchandran, and Peter L. Bartlett. "Byzantine-Robust Distributed Learning: Towards Optimal Statistical Rates". In: *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*. Ed. by Jennifer G. Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, 2018, pp. 5636–5645. URL: `http://proceedings.mlr.press/v80/yin18a.html`.

[165] Danfeng Zhang and Daniel Kifer. "LightDP: Towards automating differential privacy proofs". In: *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages*. 2017, pp. 888–901.

[166] Hengchu Zhang, Edo Roth, Andreas Haeberlen, Benjamin C Pierce, and Aaron Roth. "Fuzzi: A three-level logic for differential privacy". In: *Proceedings of the ACM on Programming Languages* 3.ICFP (2019), pp. 1–28.

[167] Kun Zhang and Lai-Wan Chan. "Extensions of ICA for causality discovery in the hong kong stock market". In: *International Conference on Neural Information Processing*. Springer. 2006, pp. 400–409.

[168] Kun Zhang, Heng Peng, Laiwan Chan, and Aapo Hyvärinen. "ICA with sparse connections: Revisited". In: *International Conference on Independent Component Analysis and Signal Separation*. Springer. 2009, pp. 195–202.

[169] Haiquan Zhao, Ashwin Lall, Mitsunori Ogihara, Oliver Spatscheck, Jia Wang, and Jun Xu. "A data streaming algorithm for estimating entropies of od flows". In: *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*. 2007, pp. 279–290.

[170] Lantian Zheng and Andrew C Myers. "Dynamic security labels and static information flow control". In: *International Journal of Information Security* 6.2-3 (2007), pp. 67–84.

[171] Banghua Zhu, Lun Wang, Qi Pang, Shuai Wang, Jiantao Jiao, Dawn Song, and Michael I Jordan. "Byzantine-Robust Federated Learning with Optimal Statistical Rates and Privacy Guarantees". In: *arXiv preprint arXiv:2205.11765* (2022).

[172] Yuqing Zhu, Jinshuo Dong, and Yu-Xiang Wang. "Optimal accounting of differential privacy via characteristic function". In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2022, pp. 4782–4817.

[173] *Zillow Prize: Zillow's Home Value Prediction (Zestimate)*. `https://www.kaggle.com/sudalairajkumar/simple-exploration-notebook-zillow-prize`. Online; accessed 9 January 2020. 2020.

Appendix A

Role Lattices in Privacy Regulations

The incomplete role lattices for GDPR, HIPAA, FERPA and CCPA are shown in Figure A.1, A.2, A.3 and A.4.

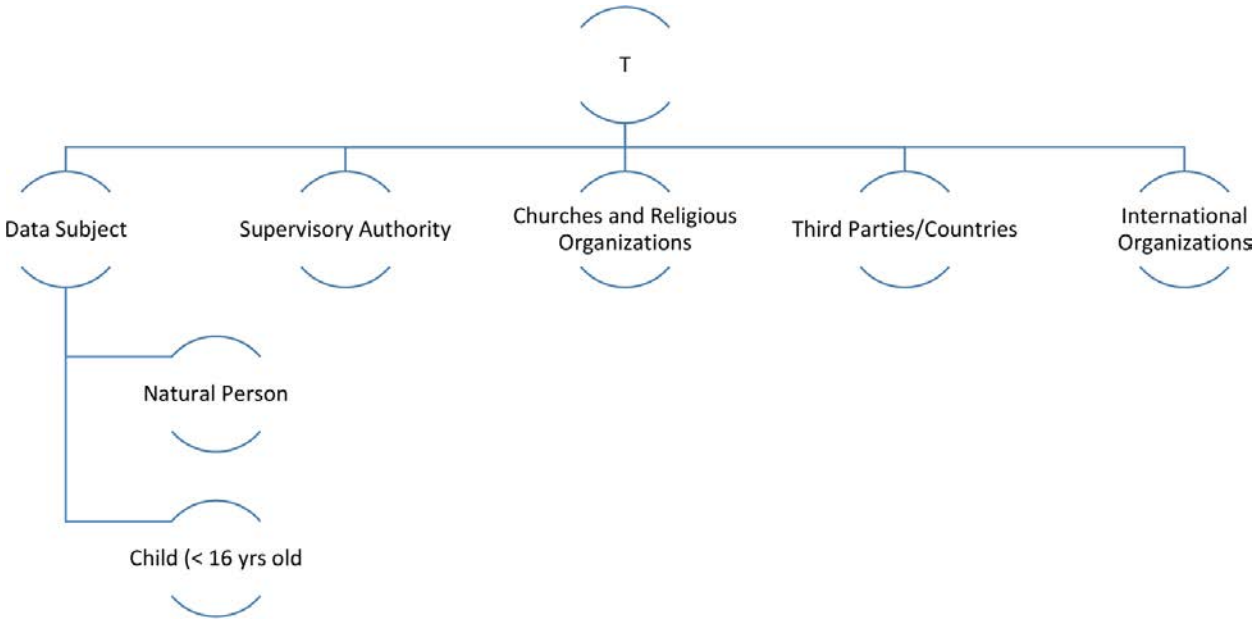


Figure A.1: Role Lattice of GDPR.

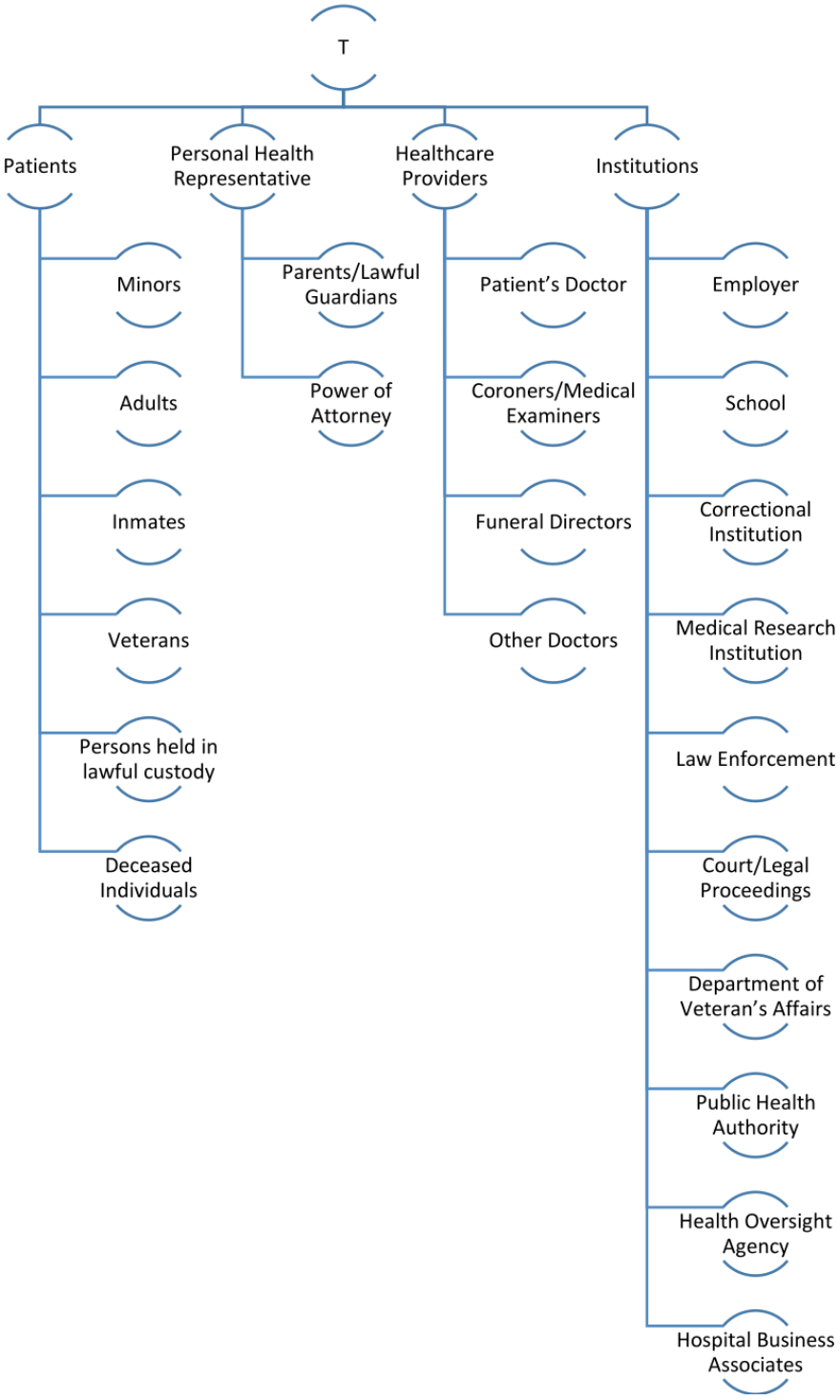


Figure A.2: Role Lattice of HIPAA.

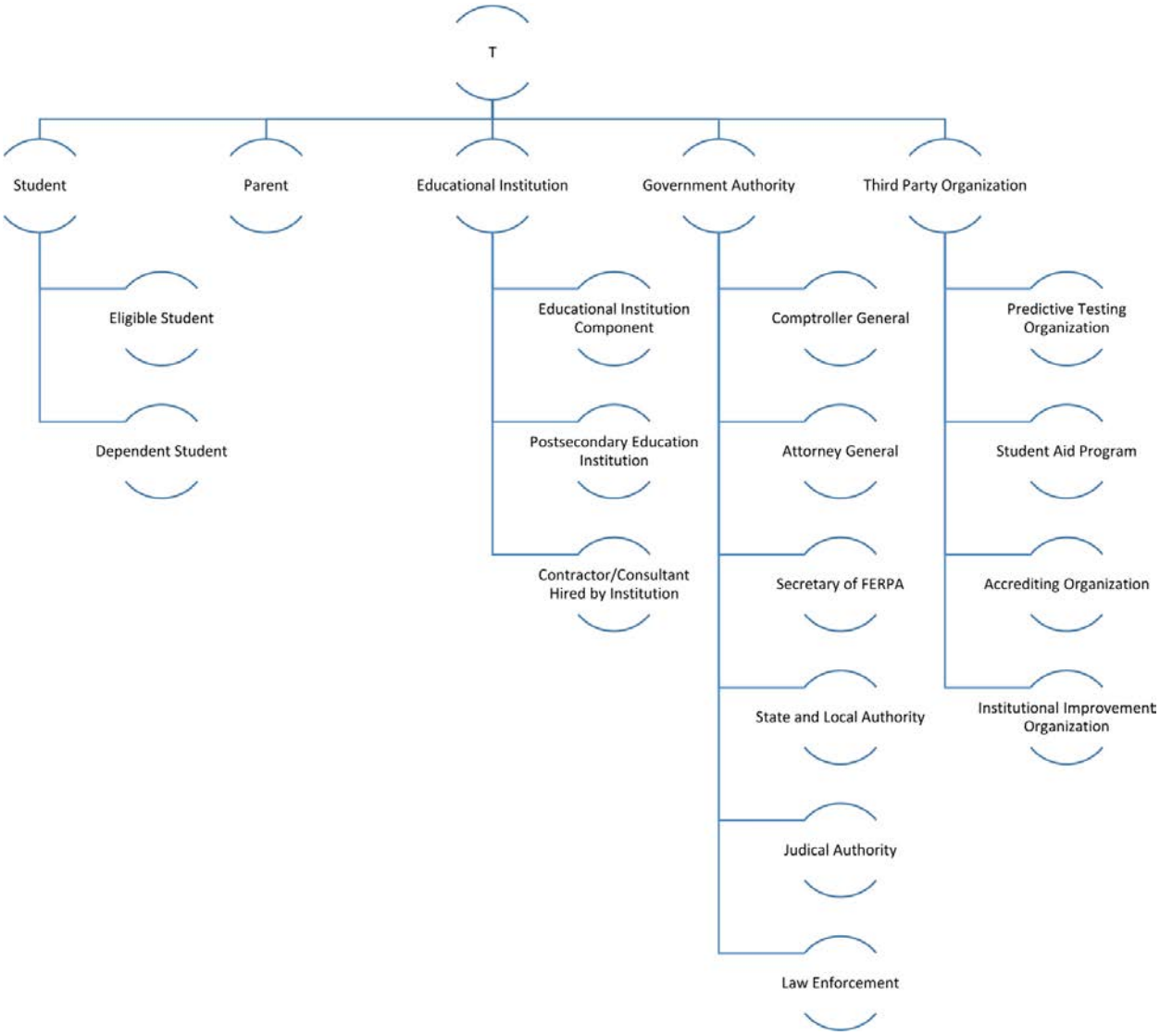


Figure A.3: Role Lattice of FERPA.

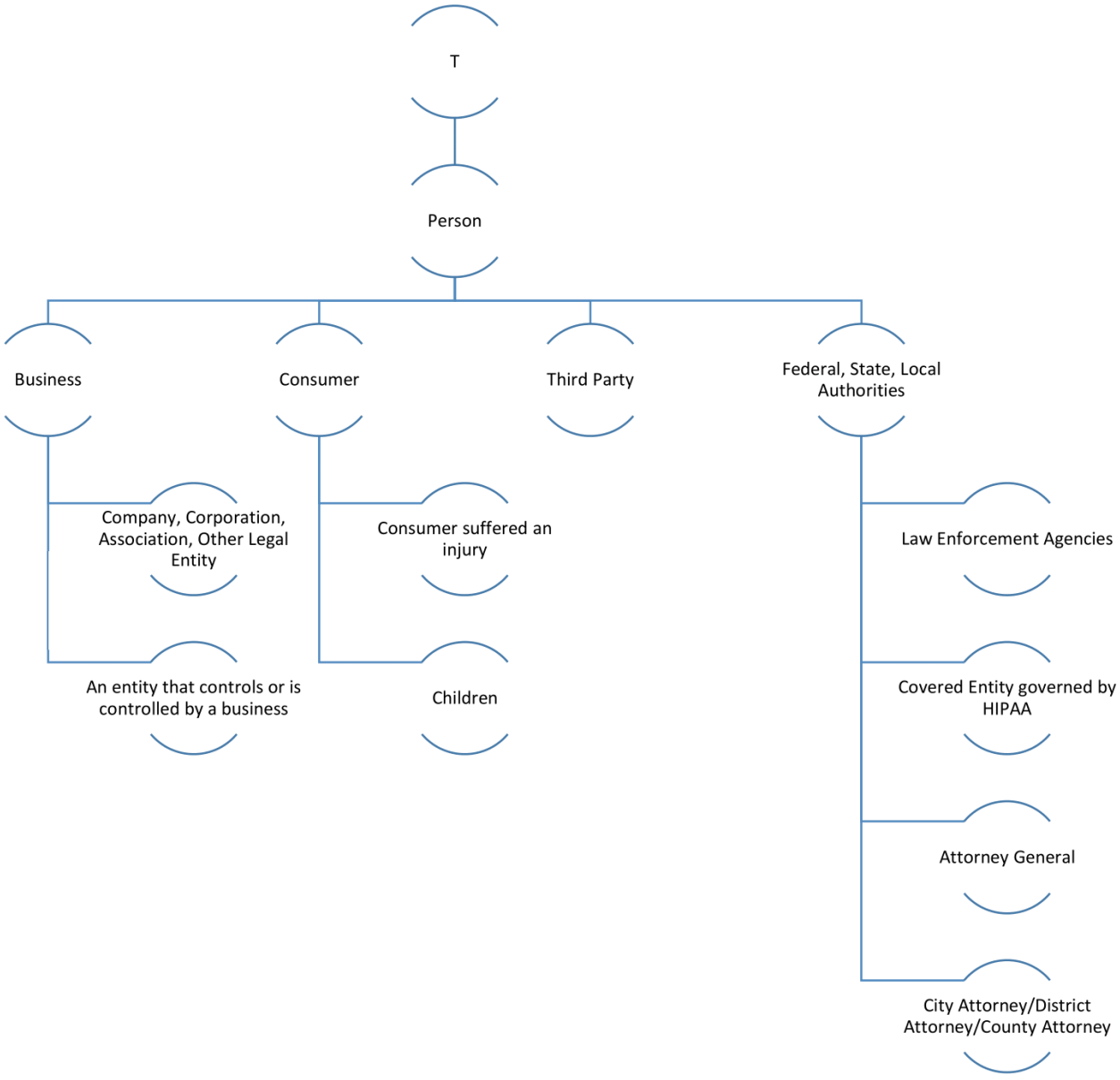


Figure A.4: Role Lattice of CCPA.

Appendix B

Legalease Encodings of Privacy Regulations

The longer LEGALEASE encodings of HIPAA, FERPA and CCPA are shown in Figure B.2, B.3 and B.4.

```
1  ALLOW SCHEMA PersonalInformation
2    AND FILTER ConsentObtained = 1
3    AND PRIVACY De—Identify
4
5  ALLOW SCHEMA PersonalInformation
6    AND ROLE UserAffiliatedOrganization
7
8  ALLOW SCHEMA PersonalInformation
9    AND ROLE SupervisoryAuthority OR ROLE HealthcareOrganization
10   AND PURPOSE PublicInterest LegalObligation PublicHealth
11
12 ALLOW SCHEMA PersonalInformation
13   AND ROLE LegalAuthority
14   AND PURPOSE PublicInterest ForJudicialPurposes
```

Figure B.1: Formal encoding of several GDPR requirements.

```
1  ALLOW SCHEMA PersonalInformation
2    AND FILTER age < 90
3    AND REDACT zip
4
5  ALLOW SCHEMA ProtectedHealthInformation
6    AND ROLE PersonalHealthRepresentative
7
8  ALLOW SCHEMA ProtectedHealthInformation
9    AND (ROLE HealthcareProvider OR ROLE Institution)
10   AND FILTER ConsentObtained = 1
11   AND PURPOSE Treatment Payment HealthCareOperations
12
13 ALLOW SCHEMA ProtectedHealthInformation
14   AND ROLE HealthcareProvider
15   AND PURPOSE Treatment Payment HealthCareOperations
16
17 ALLOW SCHEMA ProtectedHealthInformation
18   AND (ROLE HealthcareProvider OR ROLE Institution)
19   AND PURPOSE FraudDetection
20
21 ALLOW SCHEMA ProtectedHealthInformation
22   AND ROLE HIPAA_Secretary
23   AND PURPOSE HIPAACompliance
24
25 ALLOW SCHEMA ProtectedHealthInformation
26   AND (ROLE CoveredEntity OR ROLE HospitalBusinessAssociate)
27   AND PURPOSE PublicHealth Research
28
29 ALLOW SCHEMA ProtectedHealthInformation
30   AND FILTER YearsDead >= 50
31
32 ALLOW SCHEMA ProtectedHealthInformation
33   AND (ROLE LawEnforcement OR ROLE JudicialAuthority)
34   AND PURPOSE CrimeInvestigation
35   AND FILTER ConsentObtained = 1
36
37 ALLOW SCHEMA ProtectedHealthInformation
38   AND FILTER ConsentObtained = 1
39   AND PURPOSE Marketing Sale
```

Figure B.2: Formal encoding of a subset of HIPAA.

```
1  ALLOW SCHEMA EducationRecords
2    AND ROLE Parent
3
4  ALLOW SCHEMA EducationRecords
5    AND ROLE UserInstitutions
6
7  ALLOW SCHEMA EducationRecords
8    AND FILTER ConsentObtained = 1
9
10 ALLOW SCHEMA EducationRecords
11   AND ROLE (EducationOfficial EducationalInstitution
12             EducationalRepresentative )
13   AND PURPOSE Education
14
15 ALLOW SCHEMA EducationRecords
16   AND ROLE (ComptrollerGeneral AttorneyGeneral Secretary
17             StateLocalAuthority )
18
19 ALLOW SCHEMA EducationRecords
20   AND PURPOSE FinancialAid
21
22 ALLOW SCHEMA EducationRecords
23   AND ROLE (StateLocalAuthority)
24
25 ALLOW SCHEMA EducationRecords
26   AND ROLE Institution
27   AND PURPOSE (ConductEducationStudy)
28
29 ALLOW SCHEMA EducationRecords
30   AND ROLE AccreditingOrganization
31   AND PURPOSE AccreditingFunction
32
33 ALLOW SCHEMA EducationRecords
34   AND PURPOSE JudicialOrder Subpoena
35
36 ALLOW SCHEMA EducationRecords
37   AND PURPOSE PublicInterest PublicHealth
38
39 ALLOW SCHEMA DirectoryInformation
40
41 ALLOW SCHEMA EducationRecords
42   AND ROLE LawEnforcement JudicialAuthority
43   AND PURPOSE CrimeInvestigation
```

Figure B.3: Formal encoding of a subset of FERPA.

```
1  ALLOW SCHEMA PersonalInformation
2    AND FILTER ConsentObtained = 1
3
4  ALLOW SCHEMA PersonalInformation
5    AND ROLE LawEnforcementAgency
6    AND PURPOSE CrimeInvestigation
7
8  ALLOW SCHEMA PersonalInformation
9    AND ROLE PersonCoveredByEvidentaryPrivilegeUnderCALaw
10   AND PURPOSE PrivilegedCommunication
11
12 ALLOW SCHEMA ProtectedHealthInformation
13   AND ROLE CoveredEntityByFDHH
14   AND PURPOSE HIPAA
15
16 ALLOW SCHEMA PersonalInformation
17   AND PURPOSE ConsumerReport
18
19 ALLOW SCHEMA PersonalInformation
20   AND (ROLE AttorneyGeneral
21        OR ROLE DistrictAttorney
22        OR ROLE CountyCounsel)
```

Figure B.4: Formal encoding of several CCPA requirements.

Appendix C

Pure Multiplicative Sensitivity in Strict Turnstile Model

In this appendix, we derive the pure multiplicative sensitivity of F_p in the strict turnstile model. In the strict turnstile model, for a key-value stream $\mathcal{S} = \{(k_1, v_1), \cdots, (k_n, v_n)\}$ $(n \geq 1)$ where $k_i \in [m]$ $(m \geq 2), v_i \in \{-M, \cdots, M\}$ $(M \geq 1)$, the sum of vs of the same key should always be non-negative:

$$\sum_{i=1}^{n} \mathbb{I}(k_i = k) v_i \geq 0$$

Besides, for the utility of the result, we need to assume that $M < n - 1$.

Theorem 14 (Multiplicative sensitivity of F_p in strict turnstile model)**.** *A mechanism $\mathcal{M} < n-1$ which calculates $F_p, p \in (0, 1]$ in the strict turnstile model when has pure multiplicative sensitivity upper bounded by*

$$\rho_p^{st} \leq 2^{2-2p}(1 + \frac{2M}{n-1-M})^p$$

Proof for Theorem 14. An upper bound for the sensitivity of F_p in the strict turnstile model can be derived by taking the division of the upper and lower bound in the incremental setting following the same logic as the proof for Theorem 9. The upper bound is the same as in the proof of Theorem 14 so we only need to calculate the lower bound of the following expression.

First, we observe the following two inequalities.

$$\forall a, b, d > 0, c \geq 0, a \leq b, c \leq d, \frac{a+c}{a+d} \leq \frac{b+c}{b+d}. \tag{C.1}$$

$$\frac{\sum_{i=2}^{m} u_i^p + (u_1 - \Delta)^p}{\sum_{i=2}^{m} u_i^p + u_1^p} \overset{(3)+(7)}{\geq} \frac{(\sum_{i=2}^{m} u_i)^p + (u_1 - \Delta)^p}{(\sum_{i=2}^{m} u_i)^p + u_1^p}$$
$$= \frac{(s - u_1)^p + (u_1 - \Delta)^p}{(s - u_1)^p + u_1^p}$$
$$\overset{(3)}{\geq} 2^{p-1}(1 - \frac{\Delta}{s})^p$$

Taking the division between the supremum and the infimum, we get

$$\rho_p^{\text{st}} \leq 2^{2-2p}(1 + \frac{2\Delta}{s - \Delta})^p \leq 2^{2-2p}(1 + \frac{2M}{n - 1 - M})^p \qquad \square$$

As shown in Figure C.1, when m is the same, the sensitivity is very close to the sensitivity in the cash register model if $M \ll n$.

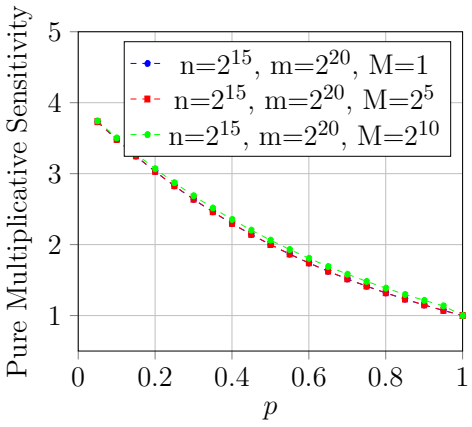


Figure C.1: Pure multiplicative sensitivity in the Strict Turnstile Model.