DISS. ETH NO. 27340

LOWERING THE COST OF ANONYMIZATION

A thesis submitted to attain the degree of

DOCTOR OF SCIENCES of ETH ZURICH
(Dr. sc. ETH Zurich)

presented by

DAMIEN DESFONTAINES
MSc École Normale Supérieure

born on 16.10.1992
citizen of France

accepted on the recommendation of

Prof. David Basin, examiner
Prof. Dennis Hofheinz, co-examiner
Prof. Carmela Troncoso, co-examiner
Prof. Ashwin Machanavajjhala, co-examiner

2020

Damien Desfontaines: *Lowering the cost of anonymization*, 2020.

SUPERVISOR: David Basin
LOCATION: ETH Zurich
TIME FRAME: 2016-12-01 to 2020-12-11

All views expressed in this manuscript are those of its author, not those of his employers.

[1] https://creativecommons.org/licenses/by-nc/4.0/

À Mamine

ABSTRACT

The objective of this thesis is to make it easier to understand, use, and deploy strong anonymization practices. We make progress towards this goal in three ways.

First, we make it easier to understand what anonymization means, and lower the cost of entry for new practitioners. After a historical tour of a number of possible definitions, we introduce *differential privacy*, a central concept studied throughout this thesis. We then systematize and organize the existing literature on variants and extensions of differential privacy: we categorize them in seven dimensions, compare them with each other, and list some of their main properties.

Second, we consider a natural class of weaker variants of differential privacy: definitions that assume that the attacker only has *partial knowledge* over the original data. We raise some fundamental issues with existing definitions, and we establish strong theoretical foundations to solve these issues and clearly delineate between distinct attack models. We use these foundations to improve existing results on the privacy of common aggregations and draw links between our notions and older syntactic definitions of privacy. Then, we provide strong negative results for *cardinality estimators*, a class of algorithms that cannot be made private even under very weak assumptions.

Third, we focus on practical applications. We present novel algorithms to detect reidentifiability and joinability risks of large datasets, and we describe the design of a differentially private query engine designed to be usable to non-experts. We propose multiple possible improvements to this query engine, and discuss a number of trade-offs between privacy, utility, usability, and extensibility; we then discuss operational challenges arising when rolling out anonymization at scale, from choosing parameters to providing appropriate education and guidance to engineers working on anonymization pipelines.

RÉSUMÉ

Le but de cette thèse est de simplifier
L'art d'empêcher que l'on puisse identifier
Un seul individu dans un jeu de données
Même si l'on se sert de voies insoupçonnées.

Définir ce problème est loin d'être évident :
Bon nombre d'érudits s'y sont cassé la dent.
D'abord, on retranscrit le parcours historique
Ayant déterminé la notion unique
Utilisée ici pour définition
Grâce à ses attributs et son intuition.
Nous classons et trions toutes ses variantes
En sept dimensions dociles et brillantes.

On s'attarde plus tard sur un point compliqué :
Comment modéliser l'attaquant étriqué
Ayant a priori bien trop d'incertitude
Pour accomplir son rôle avec exactitude.
Grâce à des fondements que l'on rend adéquats
Nous améliorons d'anciens résultats.
Certains sont positifs : on peut enfin comprendre
Comment voter pour l'une et pour autant prétendre
Avoir voté pour l'autre, en restant clandestins.
Certains sont négatifs : en comptant les distincts
Alors on doit choisir; soit on passe à l'échelle,
Soit on protège bien chaque info personnelle.

Les principes, c'est bien, les actions, c'est mieux.
C'est pourquoi l'on poursuit un but audacieux :
Rendre l'anonymat accessible en pratique.
Un accord adéquat de sens mathématique
Et d'innovation technique est précieux
Pour accomplir un jour ce rêve ambitieux.
Et l'outil n'est pas tout : c'est aussi nécessaire
D'aider tout un chacun pour être l'émissaire
Du maintien et respect de principes moraux.
Chaque choix peut changer un quidam en héros.

ZUSAMMENFASSUNG

Das Ziel dieser Doktorarbeit liegt in der Erleichterung des Verständnisses und der Verwendung von starken Anonymisierungstechniken, sowie der Bereitstellung von Prozeduren zur Anonymisierung. Hierbei erzielen wir auf drei verschiedene Arten Fortschritte.

Zuerst beleuchten wir die Grundlagen von Anonymisierung, um Neulingen einen einfacheren Zugang zum Fachgebiet zu ermöglichen. Nach einem geschichtlichen Überblick über ausgewählte andere Datenschutzdefinitionen wird *Differential Privacy* eingeführt, ein zentrales Konzept dieser Doktorarbeit. Der anschließende Schwerpunkt liegt auf der Systematisierung und Organisierung der wissenschaftlichen Literatur zu Varianten und Erweiterungen von Differential Privacy. Wir kategorisieren die vorhandenen Definitionsvarianten in sieben Dimensionen, vergleichen sie miteinander und führen einige ihrer Haupteigenschaften auf.

Als Zweites überprüfen wir natürliche Verallgemeinerungen der Differential Privacy: Definitionen, die einen Angreifer mit nur *teilweiser Kenntnis* der Originaldaten annehmen. Wir zeigen grundsätzliche Probleme bei diesen Definitionen auf und etablieren starke theoretische Grundlagen zur Lösung diese Schwierigkeiten. Hierbei ist unsere deutliche Abgrenzung zwischen zwei unterschiedlichen Angreifermodellen hervorzuheben. Wir nutzen diese Grundlagen, um bestehende Ergebnisse über verbreiteter Aggregationen zu verbessern, und um Verbindungen zwischen unseren Anonymisierungsegriffen und älteren syntaktischen Definitionen zu ziehen. Darauf aufbauend legen wir starke negative Ergebnisse für *Kardinalitätsschätzer* vor: Diese Klasse von Algorithmen wird selbst unter schwächsten Annahmen nicht dem Privatsphärenschutz gerecht.

Als Drittes betrachten wir praktische Anwendungen. Wir präsentieren neuartige Algorithmen zur Ermittlung von Informationszusammenführungs- und Identifizierungsrisiken in großen Datenmengen. Außerdem beschreiben wir den Entwurf eines Systems, welches Laien erlaubt, statistische Abfragen unter Wahrung der Differential Privacy auszuführen. Wir schlagen mehrere mögliche Verbesserungen für dieses System vor und besprechen eine Reihe von Kompromissen zwischen Datenschutz, Genauigkeit, Benutzerfreundlichkeit und Erweiterbarkeit; wir betrachten auch die Herausforderungen bei anonymisierten Datenveröffentlichungen von der Auswahl adäquater Datenschutzparameter bis zur angemessenen Schulung und Anleitung von EntwicklerInnen, die an Anonymisierungspipelines arbeiten.

♥

Ugh, emotions.

— Martha Wells, *Network Effect*

ACKNOWLEDGEMENTS

Pursuing a part-time PhD project, in a large company without any established process nor precedent for doing so was, unsurprisingly, not exactly straightforward. It required persistent optimism, a fair amount of perseverance, and a lot of unearned privilege. Above all, though, it required the support of a great many people, without which this whole endeavor would been entirely impossible. I cannot possibly do all of them justice in just a few words, but this certainly will not stop me from trying.

First of all, obviously, I would like to thank David Basin, my advisor. He decided to bet on me despite my part-time constraints, and having a project fairly distinct from the rest of his group: I am immensely grateful for the mentorship, trust, and independence he provided me during these four years. His encouragements to slow down when I was over-stretching myself, and his telling me with no hesitation that he would hire me full-time if things at Google got too heated, are the kind of show of support that I wish for any PhD student.

Thanks as well to Ashwin Machanavajjhala, Carmela Troncoso, and Dennis Hofheinz, who kindly accepted to be a part of my dissertation committee. I am also grateful to André Miede and Tino Wagner, whose LaTeX packages made it surprisingly painless to produce a beautiful-looking thesis.

Laura Arhire supported me from the very start, and was instrumental in my growth as an engineer; I can only dream of becoming as good a manager as she is one day. She, and Lorenzo Martignoni, gave me a level of freedom that most people only dream of: I thank them both for their trust and kindness. Peter Dickman tried to tell me that doing a part-time PhD was a bad idea, then when I still persevered, he provided me with much-needed help to collect all necessary approvals, and convinced me not to give up in the face of seemingly endless bureaucracy. His mentorship and support during the past four years are a major reason why I somehow have not been fired yet; I am very grateful for that.

To obtain necessary approvals, Michel Benard's assistance was also crucial; I am thankful that he took so much of his time to help me navigate internal processes. I am also very grateful to Cyrill Osterwalder, Niels Provos, and Gerhard Eschelbeck for being supportive of this project, and unlocking the necessary funding. Thanks to Pern Hui Chia for being the main engineering lead for the initial project and for reviewing so much of my code and drafts. Many thanks to Bryant Gipson as well for being such a fantastic cheerleader for my work.

If support from leaders made it possible to kick-start this project, support from peers is what made it sustainable and exhilarating throughout. I was very lucky to work with excellent researchers, who taught me a lot about research, writing, and teamwork: thanks to Andreas Lochbihler, Balazs Pejo, Elisabeth Krahmer, and Esfandiar Mohammadi, for being lovely people to work with and for making scientific research engaging and fun.

Speaking of research, I am also thankful to the many people who reviewed parts of our drafts, answered our emails about their work, or helped us with thorny issues: Aaron Johnson, Adam Groce, Adam Smith, Alain Forget, Aleksandra Korolova, Aleksei Triastcyn, Alex Kulesza, Antoine Amarilli, Ashwin Machanavajjhala, Borja Balle, Bryant Gipson, Chao Li, Chinmoy Mandayam, Christophe de Cannière, Kareem Amin, Christoph Dibak, Daniel Kifer, Dennis Kraft, Henk Brozius, Ilya Mironov, Iosif Pinelis, Ismaël Bouya, Jakob Dambon, Jessica Staddon, Jonathan Katz, Kunal Talwar, Lea Kissner, Lorenzo Martignoni, Marc Jeanmougin, Michael Lugo, Mikhail Berlyant, Nina Taft, Pablo Rauzy, Patricia Mirabile, Patrick Scheibe, Peter Dickman, Pierangela Samarati, Rebecca Balebako, Roberto Tauraso Sebastian Meiser, Theresia Weise, Úlfar Erlingsson, Virgile Andreani, Yu-Xiang Wang, Yusuke Kawamoto, as well as Stack Exchange users egreg, Mark, PSPACEhard, usul, and I apolologize if I missed anyone. I am also thankful to all anonymous reviewers, in particular those from IEEE S&P who decided to reject the first version of our cardinality estimators paper: their decision was the main motivator for me to come back to the drawing board and significantly strenghten the result; it was an excellent illustration of the review process working as intended and producing good outcomes.

Alessandro, Alejandro, Benjamin, Ioana, Lucas, Marius, Nevena, Paweł, Régis, Robert, Roman, Simone, Victor, and Yannis: you made the office a fun place to be, and the endless (and impressively creative!) trash talk at the foosball table is the main reason why I am looking forward for the office to fully reopen. I am also grateful to the Memecademy (yes) for making work a little bit more hilarious and many times more absurd, and for honoring me with the most treasured awards I ever received. Special hat tip to Heradon Douglas' creative genius. Thanks also to Patrick and Sven for being very pleasant officemates at the university.

Cloud DLP, the team I got to work on at the beginning of this project, was a fantastic opportunity to build and ship software for reidentification and risk analysis to the outside world; I am especially grateful to Jordanna Chord for being an inspiring leader, to Scott Ellis for helping me understand the difference between theory and practice, to Felipe Hoffa to take me to fun industry conferences and encouraging me to invest in external outreach, and to Emily Rossetti and Michael Daub for being excellent teammates.

My work in the Anonymization team has been, and still is, incredibly fun and rewarding. This is due in small part to the terrific problem space, but mostly to the exceptional people who compose it. Heartfelt thanks to Miguel Guevara, whose efforts to tackle such an uncharted problem space early on are nothing short of amazing and inspiring; much of what we do today has been made possible by your initial vision and leadership. Many thanks to Daniel Simmons-Marengo and Milinda Perera for their impossible kindness, and for their principled and rigorous approach to privacy consulting work. (Oh, and Daniel, thank you for the many excellent book recommendations, please keep them coming!) Thank you to Chao Li, Sławek Goryczka, and Pern Hui Chia for making this group survive against all odds when we were under an absurd load. I learned so much from all of you. Special thanks for the trust you put in me when you convinced me to take on the leadership hat for Anonymization consulting. It was the best and most terrible decision. I would do the same mistake again in a heartbeat.

Newer folks to the anonymization world, it is such an honor and a pleasure to work alongside you. Thank you to Janina Voigt and Yurii Sushko: you two are excellent leaders; what we have accomplished this past year is a testament to your team-building skills. Special thanks to Raine Serrano, Aishe Krishnamurthy, and Jessica Colnago: I would have completely drowned under the load if it were not for your exceptional work making anonymization consulting better for everyone. Many thanks to Christoph Dibak, David Marn, Dennis Kraft, Maria Telyatnikova, Miraç Vuslat Başaran, and Sasha Kulankhina; working with you and helping you ramp up and become anonymization experts in such a short time is probably the most impactful and rewarding thing I have done while at Google. (And Miracç, you turning Privacy on Beam from my ugly proof-of-concept to a production-ready open-source library still brings a tear to my eye when I think of it.) Thanks to Alain Forget, Vadym Doroshenko, and Xinyu Ye; I almost want to apologize to you for not having spent as much time as I would have liked working with you, but the fantastic work that you are all doing clearly shows that you do not, in fact, need my help. To all of you, I am awe-struck by your enthusiasm, eagerness to learn, technical depth, and kindness. I cannot wait to see how you grow, and what you accomplish in the future.

Google has been an exceptional place to grow, not only by improving technical and interpersonal skills, but also as a human being. Examples of extraordinary leaders abound, but I especially would like to thank Lea Kissner for being such an inspiring example of compassionate leadership, unbelievably kind and effective communication skills, and principled decision-making. Google is also where I learned the crucial importance of speaking truth to power and holding leaders accountable for their decisions. I am especially grateful to Irene Knapp and Liz Fong-Jones for giving me both the inspiration and the practical tools to organize for ethical principles. Thanks to them, I now know that you should, in fact, meet your heroes: if you choose them well, they will still somehow be as kind and inspiring in real life as you imagined they would be.

This naturally leads me to extend a wholehearted thank you to allies and accomplices of the past couple of years. Thanks to everyone making the Zürich site a fairer place to work: André Nogueira, Eli Stevens, Miriam Berger, Pedro Gonnet, Tom McAdam, Urs Zbinden, and others whom I forget or who prefer not to be named, for their involvement, advice, and/or support. Extra-special thanks to Adrien Kunysz and Raphaël Jamet for being at my side in October 2019; I would probably have exploded from all the stress of that absurd week if it were not for your support. Rebecca, thank you for the many discussions about work and life, and your stellar advice on both sides; I am lucky to be friends with you.

I benefited from an excellent work environment also thanks to my colleagues in temp, vendors, and contractors positions: cafe staff, baristas, massage therapists, cleaners, QA engineers, and many others. These folkds hold the company together, and play a critical role in building and maintaining our core products (content moderation, rating, internationalization…), yet they are still kept as second-class citizens by the company. They do not get the pay nor benefits they deserve, and the lack of recognition they get is insulting: I cannot find a single email welcoming them to the team, or wishing them well when they leave, even when we worked alongside them every single day for months. As a result, I can only list a few folks I particularly remember for their kindness: Chef Steph, Moussa Dabre, Mohammad Bibaoune, Violeta Boneva, Wanapa Inthayard. Thanks to them and to all the others.

Having two jobs, even when one has exceptional benefits, sometimes presents quite a challenge for the body and brain. I am particularly grateful to Isabelle Frank-Ziem, Johann Glessgen, and Philippe Stöckli, whose professional expertise helped me not fall apart when things got particularly tough.

The real treasure, of course, is the friends we made along the way. A million hugs to Streambed folks, the most wholesome online community there ever was; I am so grateful for all of you, and I can only hope to be as kind and strong as you lot one day. Much love towards Postcrossing enthusiasts as well: you are what makes the world absurd and beautiful. Thanks to everyone who ever sent me kind words for my blog posts or other stuff I put online, you have no idea how much it means. If you are reading these lines, go take a couple of minutes to send a few nice words to your favorite online content creators. Go on! This thesis will still be there when you come back. Speaking of which, thank you to Confused Bi-Product of a Misinformed Culture, whose mixes were the main soundtrack for my work.

Tabletop games are one of the main things that kept me sane these four years. Wilson, thank you for continuing to invite me to play even though I only have time once in a blue moon. Sam, Marco, and Praveena, thanks for the fun times rolling dice together. Cheers to the cast of Critical Role, who made me be enthusiastic about playing D&D again, and provided me with countless hours of much-needed entertainment during confinement. Many thanks to Andrei, Ioana, Lisa, Lucas, and Robert for being such a fun group to play with, and special thanks to Sylvain for being such a devious, fantastic DM.

The trick to having a social life outside of your work in Switzerland is to find weird, absurd, joyful niche communities: you end up meeting exceptionally lovely people. Maggie, thank you for the many drunken discussions about everything and anything. Gabriella, thank you for the ridiculous “five-star” experiences, for your trust, and for your great approach to life. Marco, thank you for being your beautiful, opinionated self, I aspire to be 1% as cool as you some day. Dan, thank you for your openness to try novel and scary things, and for the many fun evenings during confinement. I am lucky to have you in my chosen family.

Last but not least. Chloe, my love, thank you so much for being in my life. Thank you for the many fun moments, thank you for the constant support, thank you for the cathartic rants about thesis writing, thank you for the mind-boggling level of trust, thank you for the positive approach to relationships and life, thank you for the much-needed cuddles. Here’s to many more years of swimming up rivers, cooking delicious food, waking up lazily on Sunday mornings, drinking cocktails, and dreaming about the future.

REMERCIEMENTS

Cette thèse est dédicacée à Mamine, feu ma grand-mère paternelle. J’aspire à faire preuve d’une fraction de sa gentillesse légendaire. Nombreux sont les gens m’ayant poussé à faire une thèse, mais c’est elle qui m’a donné le gros de ma motivation et persévérance.

Je suis reconnaissant d’avoir une famille qui m’a soutenu tout du long : l’amour, c’est de continuer à manifester son soutien et affection même lorsqu’on ne comprend pas certains choix de vie ; j’ai une chance infinie à ce niveau. Papa, merci pour m’avoir donné le virus du syndicalisme, et pour continuer à me supporter même quand je déblatère ce genre d’ânerie.

Maman, merci mille fois pour l'amour débordant, les longues discussions philosophiques, les hectolitres de thé, et le regard résolument positif sur la vie. Carline et Violaine, merci pour le soutien constant et l'apport de zen lorsque j'en avais sacrément besoin. Merci à Françoise pour les conseils et l'hospitalité légendaire, merci à Hervé pour m'avoir donné le goût pour l'informatique il y a une vingtaine d'années, merci à Nelly et Margot pour être les cousines les plus adorables et enthousiasmantes de l'univers, merci à Frédi, Cécile, Papi et Mamike pour les chouettes et nombreuses fêtes et vacances en famille. Je vous aime fort !

Merci à tous mes potes de l'ENS et adjacents pour m'avoir injecté une quantité adéquate de pression sociale pour que je fasse une thèse. Vous aviez raison : c'était un projet absurde, mais quand même bien marrant. Merci aux Grotas pour les chouettes discussions politiques qui m'ont aidé à faire la part des choses et à ne pas trop devenir lobotomisé par la culture d'entreprise. Tout plein de remerciements à a3nm, Elarnon, louis, Mc, Machin, olasd, pintoch, Robin, Yap, et tous les autres gens d'IRC pour vos principes et pour les discussions mathématiques ou philosophiques aussi interminables qu'enrichissantes, et pensées spéciales pour ceux d'entre vous qui m'ont suivi dans la poudreuse en hiver !

Alex, Alma, Nicolas, et Shauna, merci pour les visites, les jeux de société et les nombreux chouettes moments. Noélie, merci pour les visites, toujours plaisantes même lorsqu'on se fait surprendre par l'apocalpyse. Pauline, merci pour l'hospitalité, les millions de cartes postales réjouissantes, et le soutien moral permanent. Manon, merci pour les jeux et la confiance qui va avec, je suis fier de t'avoir connue avant que tu ne deviennes une auteure mondialement célèbre. Juliette, merci pour la confiance et pour l'inspiration, je suis fier et chanceux de te compter parmi mes amies. Patricia, merci pour le soutien, les recommandations de lecture, et les excellents conseils sentimentaux. Laure, merci aussi pour la confiance, les vacances random et les discussions à cœur ouvert m'ont probablement évité de perdre pied plus d'une fois, merci tout plein pour être une personne aussi exceptionnellement enthousiasmante. Rien que lister tous les gens qui m'ont accompagné me met la larme à l'œil... Je ne sais pas ce que j'ai fait pour mériter d'être aussi bien entouré.

DANKSAGUNG

Sfé, Pascal und Tricia, ihr seid der Grund, warum ich mich zum ersten Mal in Zürich wie zuhause gefühlt habe. Tausend Dank für die Abendessen, Spiele une Organisation der Veranstaltungen. Olga, danke für dein Vertrauen une die vielen lustigen Abende. Danke, dass ihr alle so seltsame und augezeichnete Menschen seid.

Annerose und Bernd, vielen Dank für die herausragende Gastfreundlichkeit. Das Gefühl ein Familienmitgleid zu sein war sowohl sehr berührend als auch entscheidend, um meine Doktorarbeit fertig zu schreiben.

Schließlich, Sia, Liebe meines Lebens. Ich weiß nicht, wie ich dir nur annähernd für die ständige Liebe, Unterstützung une Vertrauen in diesen Jahren danken könnte. Du warst in den dunkelsten Momenten für mich da, und du hast die besten Zeiten tausendmal freudiger gemacht. Ich freue mich auf viele weitere Kuschelstunden, ausgefallene Abendessen, Autoreisen, Massagen, Star-Trek-Abende und Träume von der Zukunft.

CONTENTS

1 INTRODUCTION 1

2 DEFINING ANONYMIZATION 7

- 2.1 From syntactic to semantic privacy . . . 8
  - 2.1.1 *k*-anonymity . . . 9
  - 2.1.2 *k*-map . . . 14
  - 2.1.3 *l*-diversity . . . 16
  - 2.1.4 δ-presence . . . 19
  - 2.1.5 Flaws of syntactic privacy definitions . . . 22
  - 2.1.6 Differential privacy . . . 24
- 2.2 Systematizing variants & extensions of differential privacy . . . 34
  - 2.2.1 Preliminaries . . . 36
  - 2.2.2 Quantification of privacy loss (Q) . . . 40
  - 2.2.3 Neighborhood definition (N) . . . 45
  - 2.2.4 Variation of privacy loss (V) . . . 52
  - 2.2.5 Background knowledge (B) . . . 56
  - 2.2.6 Change in formalism (F) . . . 59
  - 2.2.7 Relativization of the knowledge gain (R) . . . 63
  - 2.2.8 Computational power (C) . . . 66
  - 2.2.9 Summarizing table . . . 67
  - 2.2.10 Related work . . . 80
- 2.3 Conclusion . . . 83

3 DIFFERENTIAL PRIVACY UNDER PARTIAL KNOWLEDGE 85

- 3.1 Definitional intricacies of partial knowledge . . . 86
  - 3.1.1 Related work . . . 87
  - 3.1.2 Correlated data . . . 88
  - 3.1.3 Passive vs. active attackers . . . 96
- 3.2 Opportunities: aggregations & thresholding . . . 105
  - 3.2.1 Counting queries . . . 105
  - 3.2.2 Thresholding . . . 110
  - 3.2.3 Application to *k*-anonymity . . . 115
  - 3.2.4 Composition . . . 119
- 3.3 Limits: cardinality estimation . . . 125
  - 3.3.1 Preliminaries . . . 126
  - 3.3.2 Private cardinality estimators are imprecise . . . 135
  - 3.3.3 Weakening the privacy definition . . . 144
  - 3.3.4 Mitigation strategies . . . 153
- 3.4 Conclusion . . . 155

4 FROM THEORY TO PRACTICE 157

- 4.1 Analyzing privacy risks at scale . . . 158

4.1.1 Introduction . . . 159
4.1.2 Definitions and system design . . . 162
4.1.3 Estimating reidentifiability and joinability using KHLL . . . 169
4.1.4 Experimental validation . . . 173
4.1.5 Summary and perspectives . . . 176
4.2 Building a usable differentially private query engine . . . 179
4.2.1 Introduction . . . 179
4.2.2 A simple example: histograms . . . 183
4.2.3 System model and design . . . 186
4.2.4 Accuracy . . . 197
4.2.5 Usability . . . 202
4.2.6 Robustness and testing . . . 204
4.3 Further improvements and open questions . . . 212
4.3.1 Beyond counts and sums . . . 213
4.3.2 Partition selection . . . 215
4.3.3 Improving coverage and utility . . . 226
4.3.4 Operational aspects of anonymization . . . 233
4.3.5 Other possible research directions . . . 238
5 CONCLUSION 241

BIBLIOGRAPHY 243

NOTATIONS

MATHEMATICAL NOTATIONS

$\mathbb{N}$	The set of nonnegative integers
$\mathbb{Z}$	The set of integers
$\mathbb{R}$	The set of real numbers
$\lvert E\rvert$	Cardinality of set E
$E \setminus F$	Set difference of set E and set F
$E \times F$	Cartesian product between set E and set F
$\mathbb{P}\left[A\right]$	Probability of event A
$\mathbb{P}\left[A, B\right]$	Probability of the conjunction of event A and event B
θ	A probability distribution (or θ_1, θ_2, $\dots$).
$\mathbb{P}_{X\sim\theta}\left[A(X)\right]$	Probability of event $A(X)$, when X is sampled from θ
$\mathbb{P}\left[A \mid B\right]$	Probability of event A, conditioned on event B
$\mathbb{E}\left[L\right]$	Expected value of random variable L
$\mathbb{E}_{X\sim\theta}\left[L(X)\right]$	Expected value of $L(X)$, when X is sampled from θ
$\theta_{\lvert B}$	Probability distribution θ conditioned on event B
$A(X)_{\lvert X\sim\theta,B(X)}$	Random variable $A(X)$, when X is sampled from $\theta_{\lvert B(x)}$
$f : E \to F$	Function, possibly probabilistic, from set E to set F

NOTATIONS FOR DATASETS AND RECORDS

$\mathcal{T}$	Set of possible records
$t \in \mathcal{T}$	A record (or t_1, t_2, $\dots$)
$\mathcal{D} = \mathcal{T}^*$	Set of possible datasets (sequences of records)
$D \in \mathcal{D}$	A dataset (or D_1, D_2, $\hat{D}$, $\dots$)
$\lvert D\rvert \in \mathbb{N}$	The number of records in dataset D (typically denoted n)
$D(i)$	The i-th record of dataset D ($i \leq \lvert D\rvert$)
D_{-i}	Dataset D with its i-th record removed

Some of these notations are slightly modified when we introduce the concept of record ownership in Chapter 4, see the “Datasets with user identifiers” notation table on page xxi.

DIFFERENTIAL PRIVACY AND ITS VARIANTS

$\mathcal{O}$	Set of possible outputs of a privacy mechanism
$O \in \mathcal{O}$	An output of the privacy mechanism
$S \subseteq \mathcal{O}$	A subset of outputs of a privacy mechanism
$\mathcal{M} : \mathcal{D} \rightarrow \mathcal{O}$	A privacy mechanism
$\approx_{\varepsilon}$	ε-indistinguishability (see Definition 5)
$\approx_{\varepsilon,\delta}$	(ε, δ)-indistinguishability (see Definition 14)
$\mathcal{L}_{\mathcal{M}(D_1)/\mathcal{M}(D_2)}$	Privacy loss random variable (PLRV) between $\mathcal{M}(D_1)$ and $\mathcal{M}(D_2)$, often abbreviated $\mathcal{L}_{D_1/D_2}$ (see Definition 13)
$\mathcal{B}$	Set of possible partial knowledges
$B \in \mathcal{B}$	A value of the partial knowledge (or B_1, B_2, $\hat{B}$, $\dots$)
Θ	Family of probability distributions on $\mathcal{D}$ or $\mathcal{D} \times \mathcal{B}$
$\theta \in \Theta$	A probability distribution on $\mathcal{D}$ or $\mathcal{D} \times \mathcal{B}$, also used as an abbreviation for $D \sim \theta$ or $(D, B) \sim \theta$, e.g. in $\mathbb{P}_{\theta}[\dots]$ or $\mathcal{M}(D)_{\|\theta}$
π	A probability distribution on $\mathcal{T}$
$\hat{D}$	A specific dataset, also used as an abbreviation for the event $D = \hat{D}$
$\hat{B}$	A specific value of the partial knowledge, also used as an abbreviation for the event $B = \hat{B}$
Sim	A function from $\mathcal{D}$ to $\mathcal{O}$ called a simulator (see Definition 50)
$D_{i \rightarrow t'}$	Dataset D with its i-th record replaced by $b \in \mathcal{T}$
$\hat{\theta}$	A normalization of a distribution θ (see Definition 53)
$\phi_0, \phi_1, \dots$	Parameters of a normalization $\hat{\theta}$ (see Definition 53)
$\mathcal{L}^{\mathcal{M},\theta}_{i \leftarrow t / i \leftarrow t'}(O, \hat{B})$	Privacy loss random variable (PLRV) of an output O given partial knowledge $\hat{B}$, when distinguishing between $D(i) = t$ or $D(i) = t'$ (see Definition 56)
$m(O, B)$	Abbreviation for $\max\left(0, 1 - e^{\varepsilon - \mathcal{L}^{\mathcal{M},\theta}_{i \leftarrow t / i \leftarrow t'}(O,B)}\right)$
$\text{APK}_{i,t,t',\hat{B}}$	Abbreviation for $\underset{\theta_{\|D(i)=t,\hat{B}}, O \sim \mathcal{M}(D)}{\mathbb{E}}\left[m\left(O, \hat{B}\right)\right]$
$\text{PPK}_{i,t,t'}$	Abbreviation for $\underset{\theta_{\|D(i)=a}, O \sim \mathcal{M}(D)}{\mathbb{E}}\left[m(O, B)\right]$
$\varphi^{i}_{B_1,B_2}$	Bijective mapping between B_1 and B_2 for record i (see Definition 59)
$\text{Dep}^{\mathcal{M}_1,\mathcal{M}_2,\theta}_{i,t,t'}$	Dependency of $\mathcal{M}_2$ on $\mathcal{M}_1$ to distinguish $D(i) = t$ and $D(i) = t'$ (see Definition 62)

CARDINALITY ESTIMATORS

$E \subseteq \mathcal{T}$	A set of records (or $E_1, E_2, \dots$)
$\mathcal{S}$	Set of possible sketches
$S \in \mathcal{S}$	A sketch (or $S', S_1, S_2, \dots$)
$\mathfrak{S}$	Set of probability distributions over $\mathcal{S}$
$\sigma \in \mathfrak{S}$	A probability distribution over sketches (or σ')
$S_\varnothing \in \mathcal{S}$	The empty sketch
$\text{add}\,(t, S)$	Sketch, or distribution of sketches, obtained by adding record t to a sketch S
$\text{estimate}\,(S)$	Estimated number of distinct records added to a sketch S
$S_E \in \mathcal{S}$	Sketch obtained by adding all records of set E into $S_\varnothing$
$\sigma_E \in \mathcal{S}$	Distribution of sketches obtained by adding all records of set E into $S_\varnothing$
$\text{merge}\,(S, S')$	Sketch obtained by merging sketches S and S'
$\text{merge}\,(\sigma, \sigma')$	Distribution of sketches obtained by merging distributions of sketches σ and σ' (also denoted $\sigma \cdot \sigma'$)
$\mathcal{P}_n\,(\mathcal{T})$	Set of subsets of $\mathcal{T}$ of cardinality n
$\mathbb{P}_n\,[A(E)]$	Probability of event $A(E)$, when E is taken uniformly in $\mathcal{P}_n\,(\mathcal{T})$

DATASETS WITH USER IDENTIFIERS

C	A dataset column (or $C_1, C_2, \dots$)
$c \in C$	The value of a column (or $c_1, c_2, \dots$)
$\mathcal{ID}$	Set of possible user identifiers
id	A user identifier (or id_1, $\text{id}_2, \dots$)
$\mathcal{ID}\,[c_j]$	The set of user identifiers associated with a column value
h	A hash function
$\mathcal{T} = C_1 \times \dots \times C_a$	Set of possible records, composed of multiple columns
$(\text{id}, t) \in \mathcal{ID} \times \mathcal{T}$	A row, associating a user identifier with a record
$\mathcal{D}$	A dataset, formally a multiset of rows
$\lVert D_1 - D_2 \rVert$	Row-level distance between datasets D_1 and D_2 (see Definition 78)
$\lVert D_1 - D_2 \rVert_u$	User-level distance between datasets D_1 and D_2 (see Definition 78)
$f : \mathcal{D} \to \mathbb{R}^d$	A vector-valued function
$\lVert v \rVert_1$	L^1 norm of vector $v \in \mathbb{R}^d$
Δf	L^1-sensitivity of function f (see Definition 80)
$\Delta_u f$	User-level L^1-sensitivity of function f (see Definition 80)

1

INTRODUCTION

After all, a person is herself, and others. Relationships chisel the final shape of one's being. I am me, and you.

— N.K. Jemisin, *The Fifth Season*

Alice sends an email to her friends and family to organize her birthday party. Bob explains his symptoms to his doctor, who takes notes on her computer. Camille requests directions to a restaurant on their favorite navigation app. Dede puts on a fitness tracker and goes out for a run around the neighborhood. Ember goes for a hike, and takes a few pictures of the view from up the hill.

All these activities will generate a trail of data. Nowadays, few are the endeavors that do not leave a digital trace on a person's devices, or on some remote server farm. This historically new phenomenon appeared and drastically expanded over the last few decades. Before that, only a few large industries and governments were automating computing and data processing. What brought about this extensive change? Improvements in hardware? A few core innovations, like the World Wide Web? Market forces?

Rather than looking for causes, one could look for *incentives*. Who benefits most from the collection, processing, and sharing of this data? Technology companies like to argue that their customers—or *users*—are the primary beneficiaries of their services. At first glance, the argument appears reasonable: many Internet services are free, and people use them because they find some value in them. Organizing one's pictures, looking up information for a school assignment, staying in touch with loved ones, watching TV shows at any time, are all examples of benefits that people get by sharing their personal data with online services. Even people who are otherwise reluctant to technological change can usually find parts of their lives to be made easier or more convenient thanks to technological innovations of the 21st century.

Of course, despite what their founders like to pretend, tech companies are not providing these services out of the goodness of their heart. Even if they did, building, running, and maintaining large-scale online services is expensive. So what are tech companies getting out of the exchange? How do they transform all the data they collect into profit? And how fair is that exchange?

One of the common mechanisms to extract profit out of personal data is personalized advertising. The massive scale and complexity of the online advertising industry is difficult to grasp, but the basic principle of personalized ads is relatively simple. Consider a fitness tracking app, in which users record their performance when sports training. A sportswear company launching a new line of triathlon clothes might pay the app provider to advertise their product to users who recently recorded their performance in running, cycling and swimming.

In this example, an individual user might find the exchange reasonable and fair. They get the benefit of using the app for free, at the "cost" of seeing ads that are relevant to their interests. The app company earns money for each *impression*—when an ad is shown to users—and *conversion*—when a user clicks on an ad and ends up buying the product. The company revenue scales roughly linearly with the number of users: twice as many users translate to twice as many impressions and conversions.

This individual view, however, is very narrow: it only considers each piece of personal data in isolation. It neglects a crucial aspect: individual data has additional value in large quantities, as large datasets are more than the sum of their parts. An app developer might collect data to measure what features of their app are most popular among its users. Ad tech companies can run experiments and measure which kind of ads are most likely to lead to profitable conversions. Aggregate data can also be used to do market research, enabling marketers to discover consumer trends early and act on them before their competitors.

Such analyses are only possible if data from sufficiently many people are included in the dataset: the value of a dataset increases more than linearly with its size. This creates feedback loops, especially with the recent advances in machine learning. The more training data companies have, the better their machine learning models perform, and the more useful their products become, which encourages more people to use them, which in turn increases their data and revenue. In parallel, as their user base grows, their advertising business becomes more efficient and profitable, giving a further advantage to large, established tech companies.

Seen under this lens, this phenomenon seems hardly fair. Why do individuals only get the benefits from the processing of their individual data, while large companies profit from the added, "compound" benefits of large datasets? Public relations departments of tech companies typically have two answers to this objection.

First, they point out that data collection can help improve the product itself, which benefits users. Examples abound. A search engine can get better over time, as its algorithm gets updated to improve accuracy or user satisfaction metrics. Data collection from the users of a navigation app enables more accurate traffic predictions, which in turns enables faster and smarter routing. A music app can aggregate the data it collects to improve its recommendation algorithm, which can help users discover their next favorite artist. To summarize the argument: sometimes, the incentives of tech companies and of their users are aligned, in which case large-scale data collection and processing can have an overall positive impact. Notice that this does not answer the *fairness* objection. If a company uses my data for five distinct purposes, and I only benefit from one of them, it is better than nothing, but the exchange still does not seem very fair.

This leads to the tech industry's second answer to this objection: the development of initiatives focusing on using data "for good". For instance, giving external researchers access to data for scientific research, forecasting the spread of diseases, developing early warning systems for natural disasters, etc. The idea behind these projects is to "give back" to the community: using the compound benefits of aggregated data for a cause that serves everyone. The scale of these efforts, however, is relatively small compared to other areas of business for tech companies.

For some efforts—especially when they require building entire systems from scratch, like early warning systems for earthquakes—this is not surprising. The incentives of a capitalist system do not exactly encourage companies to spend more time on doing social good than on pursuing their core business, or other profitable endeavors.

Some other efforts, though, seem to require little effort, and can potentially lead to significant positive outcomes for society and for the company's public image. One major example is sharing data with external researchers. At first glance, it seems like this practice benefits all the actors involved. Academics get access to valuable data to study, allowing them to publish papers in prestigious venues and boost their careers. Society benefits from scientific progress brought about by this work. And the company can reap public image benefits for free: all the hard analysis work, requiring valuable time and expertise, is outsourced to researchers. Even better, the process improves relations with prominent academics, who are useful allies—*key opinion formers*, in public relations jargon—to have when trying to influence regulators or polish one's media coverage.

So why is this kind of partnerships not more common? The main reason is *risk*. Sharing data externally, even under non-disclosure agreements and with some security measures in place, increases the chances that it will be leaked to other third-parties, or used for unplanned ill-intentioned purposes. The best example of this phenomenon might be the Cambridge Analytica scandal. Aleksandr Kogan, who developed a Facebook app that collected the data of millions of users, was a Cambridge researcher and a Facebook consultant. The collected data was then used to target ads to individual voters and influence elections.

Concerns about privacy risks are therefore an understandable reason, as well as a convenient excuse, for companies to avoid sharing data externally. One of the ways to solve this problem, and re-align incentives, is *anonymization*. Anonymizing personal data means altering it to minimize or eliminate some of the privacy risks it carries. Of course, anonymization alone is pointless: redacting a dataset entirely is a very easy and effective way to remove all the risk associated with it, but it would also destroy its value completely. The core challenge of anonymization is thus to preserve some useful properties of data, while still limiting risk. In the example above, where a company wants to share data with researchers, the goal is to retain the statistical properties of the data that enable rigorous scientific analysis.

This is the fundamental challenge that this thesis focuses on. How to anonymize a dataset in a way that provides strong privacy guarantees, while still preserving its usefulness? The problems that this work investigates all derive from this core question. Which concepts and techniques already exist in the scientific literature? What are their limitations and the main obstacles to their adoption? And how can we best address them?

Today, anonymization is largely seen as something extremely complicated to achieve. People might have heard of the numerous anonymization failures of the past 21st century, and concluded that it is simply impossible to correctly anonymize data. Others might have heard of more recent, stronger definitions of anonymity like differential privacy—a core concept examined and used throughout this thesis—but dismiss it as purely theoretic, and too difficult to be used in practice. The goal of this work is to dispel these misconceptions, and make it easier and cheaper to safely anonymize data.

We start this ambitious goal in Chapter 2, by trying to determine what it means for data to be sufficiently anonymized. This is not a simple endeavor, as more than 200 variants and extensions of differential privacy have been introduced in the last decade. We systematize and categorize these notions, to provide a unified and simplified view of this field (Section 2.2). Then, in Chapter 3, we focus on one family of such variants: definitions that, in contrast to the original formulation of differential privacy, assume that the attacker only has partial knowledge about the input dataset. We identify and solve definitional problems in existing variants (Section 3.1), we present new and improved results on the privacy of common mechanisms (Section 3.2), and we prove impossibility results for the important problem of cardinality estimation (Section 3.3). Finally, in Chapter 4, we look at the main obstacles to the widespread use of anonymization techniques for practical problems. We propose scalable sketching algorithms for estimating reidentifiability and joinability risk (Section 4.1), as well as a framework to implement differential privacy as part of a query engine (Section 4.2). We then propose a number of possible improvements to such a system, and discuss some operational challenges raised by the use of differential privacy in practical scenarios (Section 4.3).

PERSONAL CONTRIBUTIONS

Almost all this thesis is the result of work done in collaboration with other researchers, engineers, and other colleagues. It was an utmost privilege to work with so many exceptional people. Much of the credit goes to them, although I am solely responsible for all the mistakes that might still be present. Large portions of this work have also been published in some other form in other venues. This section gives more context on existing publications and gives credit where credit is due.

The historical overview of syntactic definitions of privacy, and the introduction of differential privacy present in Section 2.1 merely presents existing work. They are based on a series of blog posts published on my personal website [98].

The systematization of knowledge on variants and extensions of differential privacy exposed in Section 2.2 was done in collaboration with Balazs Pejo. We contributed equally to this work, whose short version was published as [102]. The long version is available on arXiv [101], and is currently under review.

The exploration on differential privacy with partial knowledge presented in Sections 3.1 and 3.2 is the result of a collaboration with Elisabeth Krahmer and Esfandiar Mohammadi, supervised by David Basin. I developed and wrote most of the original paper, but the core insights behind the concept acceptable normalizations emerged from productive discussions with Esfandiar, who also pushed for and came up with most of the composition results. Elisabeth proved the results on Θ-reducibility, and implemented the experiments in these sections; she and David also found critical flaws in initial versions of many results. The corresponding paper is available on arXiv [100] and is currently under review.

The work on cardinality estimators presented in Section 3.3 and published in [99] was done in collaboration with David Basin and Andreas Lochbihler. I defined the original problem, which originated from a Google privacy review, came up with the core results, and wrote most of the original paper. However, this could not have happened without Andreas and David, and I am grateful for their patience and mentorship: the clear definition of the attacker model is theirs, they found multiple critical flaws in earlier versions of the proofs, and I learned how to write scientific papers thanks to the many rounds of reviews they performed on my initial drafts.

The original design and implementation of KHyperLogLog is largely due to Pern Hui Chia. The work written up in Section 4.1 is the result of a collaboration with Wei-Yen Day, Miguel

Guevara, Chao Li, Milinda Perera, Daniel Simmons-Marengo, Qiushi Wang, and myself; it was published as [76]. I contributed to the production implementation of the system, helped with thorny design issues, re-implemented it in SQL, ran the experimental validation, and published the experimental code as open-source software. I also helped write the paper and convince various internal stakeholders that this was worth publishing.

The differentially private SQL engine from Section 4.2 was designed and built by Bryant Gipson, William Lam, Daniel Simmons-Marengo, Royce J Wilson, and Celia Zhang; since the publication of the corresponding paper as [388], many more people have contributed to it. My personal contribution consists mostly in reviewing the design of various elements, and turning the core engineering contribution into a scientific paper by framing the contributions, organizing them into a coherent story, defining the experimental setup, and performing many rounds of reviews.

The work on partition selection in Section 4.3.2 was done in collaboration with Bryant Gipson and James Voss. I came up with the original insight, write-up and experiments, Bryant did the initial math for the closed-form formula, and James implemented it and properly wrote it up, fixing a number of subtle problems in the process.

I wrote the initial proof of concept of Privacy on Beam, mentioned in Section 4.3.3.1; Miraç Vuslat Başaran, Christoph Dibak, and Maria Telyatnikova turned it into a production-grade, open-source library, with contributions from Skye Berghel, Manushree Vijayvergiya, and James Voss. Privacy on Beam is the Go version of a tool previously designed and implemented in C++ by Kareem Amin, Jenny Gillenwater, Alex Kulesza, Andrés Muñoz Medina, and Sergei Vassilvitski.

2

DEFINING ANONYMIZATION

Embrace diversity.
Unite—
Or be divided,
robbed,
ruled,
killed
By those who see you as prey.
Embrace diversity
Or be destroyed.

— Octavia Butler, *Parable of the Sower*

The first step towards solving any problem is defining precisely what we are trying to achieve. In the case of anonymization, this is a surprisingly difficult endeavor. Informally, the objective is simple: anonymized data should not reveal sensitive information about individuals. This informal goal is simple and intuitive, the difficulty is to convert it into a formal statement.

First, what kind of information counts as sensitive? Different people will have different expectations and opinions about what they would prefer to keep private about themselves. In some cases, there might be an obvious answer: for example, in a healthcare dataset, diagnostic information is sensitive. But in many other examples, it is not so obvious. Information that might seem benign to most people might reveal private facts about individuals: for example, revealing one's gender in an old dataset might out a transgender person to their colleagues.

One solution to this difficulty is to focus on the "about individuals" part of the informal definition above. The reasoning goes as follows. If it is impossible to associate any information with a specific individual, then that person is protected. Their anonymity is preserved.

Organizations, when communicating about how they use anonymization to protect people's privacy, typically focus on this aspect. The US Census Bureau states that it is "not permitted to publicly release [one's] responses in any way that could identify you" [199]; the relevant statute specifies that it may not "make any publication whereby data furnished by any particular [. . .] individual [. . .] can be identified" [313]. Similarly, some companies state that anonymized data "cannot be associated with any one individual" [198].

Note the use of "could" or "can" in these definitions. They limit what can *potentially* be inferred from the data. This is in stark contrast with the misleading claim, all too common among companies and research organizations, that some data they publish is anonymized because it does not *immediately* reveal people's identities [93, 94, 331]: there is a large difference between making reidentification vaguely non-trivial and making it impossible. This misuse of terminology leads to the frequent misconception that anonymization is simply impossible [59, 287].

So, how to formalize the impossibility of reidentifying individuals? Over the past few decades, a large number of definitions has been proposed. These data privacy definitions fall into two broad categories. The first approach is to try to define a criterion that applies to the data itself. A dataset that passes this criterion is considered "anonymized enough", and the problem is then to explore the different ways to transform data so the end result satisfies the criterion. In this chapter, we will introduce four such definitions, and explain the shortcomings of each, and of this general approach.

A second possibility is to consider anonymization as a property of the *process*, rather than its outputs. This is one of the core insights of *differential privacy*, a definition that has encountered remarkable success since its introduction: a quick online search shows that in 2020, more than 3000 scientific papers mentioning differential privacy have been published. We will introduce this notion, explain what advantages it provides, and detail how its guarantees can be interpreted.

This fundamental idea of a process-based definition of anonymization has proved very fruitful. In the decade following the introduction of differential privacy, a large number of variants and extensions of the original definition have been proposed. In the third and main part of this chapter, we propose a taxonomy of these variants and extensions. We list all the definitions based on differential privacy we could find in a systematic literature survey, and partition them into seven categories, depending on which aspect of the original definition is modified. We also establish a partial ordering of relative strength and expressibility between all these notions. Furthermore, we list which of these definitions satisfy some of the same desirable properties that differential privacy provides.

2.1 FROM SYNTACTIC TO SEMANTIC PRIVACY

Before differential privacy was introduced, a number of privacy researchers tried to find a good definition for anonymized data, and develop algorithms to modify a dataset so it satisfies a given definition. In this section, we list four of those definitions. We start with k-anonymity, an influential notion that was the starting point for an active area of research. We then present l-diversity, k-map, and δ-presence.

Although these definitions are no longer widely used in anonymization research, making a short tour of these notions is valuable. Their differences help understanding the various types of privacy risks from data releases, and natural approaches to mitigating these risks. Their history illustrates the difficulty of coming up with a satisfying definition of anonymized data, making it easier to appreciate the insights behind differential privacy. Finally, we will revisit some of these notions in later chapters of this thesis, where we show that some of them can still be relevant and useful.

Throughout this thesis, we denote by $\mathcal{D}$ the space of possible datasets. Each dataset $D \in \mathcal{D}$ is a list of *records* with values in an arbitrary set $\mathcal{T}$. We denote by $n = |D|$ the number of records in D. Unless specified otherwise, each individual in a dataset D is associated to a single record. We number each individual $1, \ldots, n$, the corresponding record is denoted by $D(i)$. Typically, $\mathcal{T}$ can be the set of integers $\mathbb{N}$, the set of real numbers $\mathbb{R}$, some finite set of possible *categories*, the set of strings, or a combination of the above. These notations, and others used in this section, are summarized on page xix.

2.1.1 *k-anonymity*

In 1997, the Group Insurance Commission (GIC), a Massachusetts health insurance organization, compiled and published a database of hospital visits by state employees [355]. The dataset was then shared widely with researchers, to encourage scientific research. Of course, there were privacy considerations with publishing people's health records: "in accordance with the best de-identification practice of the time", all directly identifying information was removed: name, phone number, full address, social security number, etc.

As you can probably expect, this story does not end well. Some demographic information was left in the database, to allow researchers to run statistical analyses: ZIP code, date of birth, and gender were all part of the data. William Weld, then governor of Massachusetts, "assured the public that GIC had protected patient privacy". Latanya Sweeney, then a PhD student at MIT, did a back-of-the-envelope calculation, and started suspecting that this apparently innocuous information could be, in fact, very identifying. To prove her point, she decided to try and reidentify Weld's records from the "anonymized" database [14].

With just $20, Sweeney bought the public voter records from Massachusetts, which had both full identifiers (names, addresses) and demographic data (ZIP code and date of birth), and contained the governor's information. Only one record in the hospital dataset matched the governor's demographic data, and thus, Sweeney was able to know which prescriptions and visits in the data were his. She sent him that information by mail, showing theatrically that their anonymization process was not as solid as it should have been.

Several factors made this attack possible.

1. The hospital data contained demographic information that could be used to distinguish between different records.
2. A secondary database was available to figure out the demographic information about the target.
3. The target was in both datasets.
4. And the demographic information of the target (ZIP code, date of birth, and gender) was unique within both datasets: only one record had the demographic values of the governor.

At first glance, all factors appear to be necessary for the attack to work. So, which ones can we afford to remove, while making sure that the data can be used for data analysis tasks? Let us go through all options.

1. We could remove all demographic information from the data, or even all information that might be linked to a person using auxiliary sources. Unfortunately, this would also severely hinder the utility of the data: correlations based on age, gender, and geographic info are very useful to researchers!

2. Society probably should do something about the existence of public (or commercially available) data sources that can be used in reidentification attacks. For example, regulations might outlaw the publication or sale of such datasets, making them harder to come by. However, this is a complex issue, on which individual data owners have little power.

3. Again, there is not much we can do about this aspect. We have no way to modify the secondary (public) dataset. We could decrease the probability that a random target is in our dataset by sub-sampling it, but all people in the sample would still be at risk.

4. The uniqueness of the governor's demographic information is a central aspect of the reidentification attack. Maybe, rather than suppressing all demographic values—which would render the data useless—we could find a less drastic of making those less unique.

This last suggestion is the fundamental intuition behind *k-anonymity*: everyone must be hidden in a crowd of at most k people. More precisely, each each combination of demographic values must map to at least k different people. Of course, demographic information is not the only type of data that can be used in such an attack. Any personal information that the attacker knows about an individual can be used. Thus, the first step is to define a list of *quasi-identifiers*: database columns that we assume the attacker knows. To that end, we model the space of possible records as $\mathcal{T} = Q_1 \times \cdots \times Q_a \times C_1 \times \cdots C_b$: each Q_i is a quasi-identifier (e.g. age or ZIP code), while C_i columns are assumed to be unknown by the attacker.

Definition 1 (k-anonymity [337, 354])**.** *Assume $\mathcal{T} = Q_1 \times \cdots \times Q_a \times C_1 \times \cdots C_b$. A database D is k-anonymous if for every possible combination of quasi-identifier values $(q_1, \ldots, q_a) \in Q_1 \times \cdots \times Q_a$, this combination is present in at least k records in the dataset, or in none.*

For example, assuming that ZIP code and age are both quasi-identifiers, the dataset in Table 2.1 is 2-anonymous, while the dataset in Table 2.2 is not.

ZIP code	age
4217	34
4217	34
1742	77
1742	77
4217	34

Table 2.1: A 2-anonymous dataset.

Note that we need every *combination* of values to appear at least k Thus, even if each individual value of each column appears 2 times in the dataset of Table 2.3, it is not 2-anonymous either.

The intuition is that when a dataset is k-anonymous for a sufficiently large k, the last requirement for a successful reidentification attack is broken. An attacker might find out the

ZIP code	age
4217	34
1742	77
174**3**	77
4217	34

TABLE 2.2: A dataset that is not 2-anonymous: ZIP code 1743 is unique.

ZIP code	age
4217	34
1742	34
4217	77
1742	77

TABLE 2.3: A dataset that is not 2-anonymous: all combinations of ZIP code and age are unique.

demographic information of their target using a secondary database, but then this demographic information will be linked to k different individuals, so it will be impossible to know which one is their info. The privacy guarantee gets stronger with increasing k values: the larger k, the bigger the crowd in which each individual is “hidden”.

2.1.1.1 *Policy questions raised by k-anonymity*

Using this definition in practice requires answering a number of questions. First, what types of data are reidentifying? How should one decide which columns in a dataset are assumed to be known by the attacker, and should be classified as quasi-identifiers? As we mentioned, ZIP codes, age, or gender are all good candidates for reidentification attacks: they are public or easily findable information that is also often found in sensitive datasets, especially medical ones.

In general however, there is no universal criterion for what constitutes of quasi-identifier. This choice depends on the attack model. Timestamps, medical conditions, physical characteristics, behavioral information can all be considered potentially reidentifying in some cases. In other cases, it can seem reasonable to assume that someone trying to attack a dataset would not have easy access to these values.

The second question is about the choice of k. Which value is appropriate? Again, it is difficult to say. In the healthcare world, when medical data is shared with a small number of people (typically for research purposes), k is often chosen between 5 and 15 [137]. To the best of our knowledge, there is no official law or regulation which suggests a specific value. Some universities, companies or other organizations have official guidelines, but the vast majority do not.

This choice is difficult to make. To pick a principled value for k, we would need to understand what is the link between the parameter value, and the risk of a privacy incident happening. But if k-anonymity is relatively easy to understand, estimating this risk quantitatively is not straightforward. It is difficult to measure how much more "privacy protection" is offered by larger values of k. Furthermore, the level of risk depends on many additional parameters: how valuable the data is, how bad would a privacy incident be, how many people have access to the data, etc.

2.1.1.2 *k-anonymity in practice*

To use k-anonymity in practice, one has to decide how to modify the initial dataset to create a k-anonymous version of it. Two basic techniques are typically used as building blocks for more complex algorithms: *generalization* and *suppression* [246, 247, 338, 353].

Generalization is the process of making a quasi-identifier value less precise, so that records with different values are transformed (or *generalized*) into records that share the same values. Consider the records in Table 2.4.

ZIP code	age
4217	34
4217	39
1742	75
1691	77

Table 2.4: A dataset prior to generalization

The numerical values of these records can be transformed into *numerical ranges*, so that the resulting table verifies 2-anonymity. This is demonstrated in Table 2.5.

ZIP code	age
4217	30–39
4217	30–39
1000–1999	75–79
1000–1999	75–79

Table 2.5: Dataset from Table 2.4, after generalization

Generalization makes data more imprecise to satisfy privacy requirements, but still allow useful data analysis to be done. In our example, changing precise ages into age ranges might be enough to analyze whether a disease affects younger or older people disproportionately.

Transforming a numerical value into a range is one of the most typical ways of performing generalization. Other ways include removing a value entirely (e.g. transforming a gender value into "gender unknown"), or using a generalization hierarchy (e.g. transforming an ICD-10 diagnosis code into a truncated code, or into the corresponding block [204]).

In the previous example, our records had relatively "close" demographic values, which allowed generalization to keep reasonably accurate information while still ensuring 2-anonymity. What if an additional record was added to Table 2.4, with ZIP code 9755 and age 13? The other four records can be grouped in two pairs as above, but that new record is an outlier. Grouping it with one of the pairs above would mean having very large ranges of values (age between 10 and 39, or ZIP code being completely removed), which would significantly reduce the utility of the resulting data.

Suppression can be used in this case: we could simply remove such outlier values, and only use generalization on records that can be. Using both generalization and suppression on this example could lead to the same 2-anonymous table as before, on Table 2.5. When suppression is used, there are usually strictly less records in the transformed table than in the original. On large datasets, allowing a small percentage of suppressed records typically allows the result to be k-anonymous without requiring too much generalization.

How to use generalization and suppression to reach k-anonymity, while maximizing the utility of the output data? Finding the *optimal* strategy is NP-hard for many reasonable definitions of optimality [282], but a large number of approximation algorithms have been proposed in the literature [168], and implemented as open-source software [20, 339, 363, 371].

2.1.1.3 *Limits of k-anonymity*

k-anonymity is simple to understand, and it seems natural to assume that reidentification attacks are well mitigated when a dataset is made k-anonymous. However, it only mitigates this particular kind of attack, and is difficult to use in some contexts. We present three such limitations, each of which is a major motivation for alternative definitions we introduce next: k-map, l-diversity, and δ-presence.

First, k-anonymity is difficult to use for smaller datasets, such as ones with 100 records or less. For many of these datasets, especially in the context of a healthcare study, the attacker might not know that their target is in the dataset in the first place. As such, the "search space" for reidentification is larger than the dataset: if a single user has certain characteristics in the dataset, but many more people share these characteristics outside of the dataset, the attack might still fail.

Second, we implicitly assumed that the attacker only wanted to select a target, and link a specific record with this target. In Sweeney's original attack, the privacy risk came from the additional sensitive data that was obtained after reidentification: medical records associated with the target. If a quasi-identifier combination is always associated with the same sensitive value in a dataset, then the attacker can learn this value even without reidentifying the record.

Third, for some datasets, being in the dataset constitutes sensitive information in the first place: for example, everyone participating in a clinical trial for a specific drug might share the same sensitive diagnostic.

k-anonymity falls short of modeling the three scenarios above. We elaborate on these issues in the following three sections: each section introduces a definition that attempts to solve one of these issues. k-anonymity has more problems than those listed above, we come back to these when introducing differential privacy.

2.1.2 *k-map*

Suppose a researcher conducted a survey about human sexual behavior in the US. The answers to that survey are obviously revealing sensitive information about individuals. Assume that the data collected is small-scale: only 40 subjects volunteered for the study. Afterwards, the researcher wants to share this data with other researchers. Looking at the columns in the dataset, they make the assumption that ZIP code and age are likely to be used in reidentification attacks. Could k-anonymity be used to share it in a safe way?

It will likely prove difficult to do without losing a lot of valuable information. For $k = 10$, a rather small value, we might need buckets like 20–49 for age. Those would obviously not provide a lot of statistical value. This large loss in utility was to be expected: with so few people in your database, it is necessary to bundle together very different age values.

Do we really need k-anonymity? Who are the attackers, in this scenario? The researchers with whom you share the data, and possibly unknown parties if the data ever leaks, might not have background information about who is in the dataset. Thus, the attacker must not only distinguish between different records, but they need to actually find the *real identity* of a record based on its information. This attacker is significantly weaker than the one implicitly assumed in the k-anonymity model!

ZIP code	age
85535	79
60629	42

TABLE 2.6: Two sample rows from a hypothetical clinical study.

Consider two different rows in this hypothetical database, listed in Table 2.6. At first glance, the “amount of information” for these two individuals seems to be the same. Taking a look at the values, however, tells a different story. ZIP code 85535 corresponds to the small town of Eden, Arizona. Approximately 20 people live in this ZIP code: it is likely that only one person in this ZIP code is exactly 79 years old. ZIP code 60629, however, corresponds to a part of the Chicago metropolitan area. More than 100,000 people live there: probably more than 1,000 of them are 42 years old.

Therefore, it seems like the first row is very easy to reidentify, but that we do not have enough information to reidentify the second row. k-anonymity does not capture this consideration: both rows are unique in the dataset. A variant of k-anonymity is needed for this case: k-map.

Just like k-anonymity, k-map requires us to determine which columns of the database are quasi-identifiers, and be used by the attacker to reidentify their target. This information alone, however, is not enough to compute k-map. In the example above, we assumed that the attacker does not know whether their target is in the dataset. So what are they comparing a given record with? This comparison is done across all other individuals sharing the same values in a larger, sometimes implicit, dataset. For the previous example, this could be "everybody living in the US", assuming the attacker has no prior information on who could potentially be in this dataset. We call this larger table the *reidentification dataset*, and denote it by $\mathcal{U} \subset Q_1 \times \cdot \times Q_a$.

Once the choice of quasi-identifiers and reidentification dataset is done, the definition is straightforward: it is similar to k-anonymity, except we count the number of records in the larger dataset $\mathcal{U}$.

Definition 2 (k-map [352]). *Assume $\mathcal{T} = Q_1 \times \cdots \times Q_a \times C_1 \times \cdots C_b$. A database D satisfies k-map for a reidentification dataset $\mathcal{U}$ if for every possible combination of quasi-identifier values $(q_1, \ldots, q_a) \in Q_1 \times \cdots \times Q_a$ present in D, this combination is also present in at least k records in $\mathcal{U}$.*

In our example, this corresponds to counting the number of people in the US who share the quasi-identifier values of each row in the dataset. For our sample dataset in Table 2.6, the values of the first row matched only one person in the US, so this dataset does not satisfy k-map for any value of $k \geq 2$. How to obtain a larger k? We can use techniques similar to the ones mentioned previously, in Section 2.1.1.2, and generalize some of these quasi-identifiers. An example is given in Table 2.7.

ZIP code	age
85000–85999	79
60629	42

TABLE 2.7: A dataset satisfying k-map for a large k.

ZIP codes between 85000 and 85999 include the entire city of Phoenix, Arizona. More than 36,000 people between 75 and 84 years old live in Phoenix [315], so there are probably more than 1,000 people who match the quasi-identifiers values of the first row. We saw earlier that the second row also matched 1,000+ people. Thus, this generalized dataset satisfies 1000-map.

2.1.2.1 *Policy questions raised by k-map*

It is easy to see that k-map raises the same difficult policy questions as k-anonymity, mentioned in Section 2.1.1.1: it is difficult to choose k, and it might not be obvious to choose which columns are considered quasi-identifiers. In addition, k-map requires choosing the reidentification dataset $\mathcal{U}$, which is another difficult choice. Our choice to take the entire US population in the example above is very optimistic; it implicitly assumed that the attacker had no additional information about their target whatsoever. With such a reidentification dataset, not much generalization is needed to make this dataset k-map.

In many cases, it would make sense to assume that the attacker could use a smaller reidentification database. For example, if the survey was advertised in a specific online community, the attacker could make the reasonable assumption that all survey participants are also members of this community. This would allow the attacker to restrict their search, and decrease the number of possible options for each row. As the reidentification dataset gets smaller, so does the k in k-map. The extreme would be to consider that the dataset D itself is used for reidentification: in this case, k-map is the same as k-anonymity.

2.1.2.2 *k-map in practice*

The reliance of k-map on modeling a reidentification dataset makes it significantly more difficult than k-anonymity to use in practice. In general, the person anonymizing the data does not have access to such a reidentification dataset, nor can they know which alternative dataset might be used. Procuring all available commercial datasets is generally impossible. Thus, transforming a dataset so the output satisfies k-map seems quite difficult in practice. In some situation however, the problem is tractable.

For example, suppose that instead of releasing all the data, we release a random sample of the original dataset. In this case, it is reasonable to assume that the attacker does not know which of the original records were sampled, so we can compute the k-map value based on the original dataset.

Another option is to assume that the data D is a *representative* (or *unbiased*) sample of a larger dataset. This is a reasonable approximation to make if people in the dataset were selected uniformly at random across a larger population, as is common in scientific studies. In this case, it is possible to compute an estimate of the k-map value for the data using statistical methods, either with purely statistical methods [137], or by approximating the reidentification dataset using publicly available data [82].

Finally, for small-case studies, a motivated data owner could try and do the job of an attacker "by hand": go through each record and try to map it to a real person, similarly to what we did in the toy example of Table 2.6. The result of such an analysis is essentially an estimate of the k-map value. This last option is approximative, and obviously not scalable.

2.1.3 *l-diversity*

Consider voter data, commonly available in most US states. This data is public by law, and contains full names. One could point at an arbitrary record and claim that we "reidentified" this individual. This "attack" always succeeds, but is not particularly interesting. In Sweeney's reidentification attack detailed in Section 2.1.1, the privacy issue was not only about finding the individual associated with a given record, but rather, about retrieving *sensitive* information linked with the record: diagnostics and drug prescriptions. The leak of this sensitive information associated to a specific individual was the real issue, more than the reidentification itself.

Are there cases where it is possible to find out sensitive information about someone, without reidentifying their exact record? Let us look at an example where we explicitly add sensitive information in the dataset. Each row of Table 2.8 contains a direct identifier (name), two quasi-identifiers (ZIP code and age), and one *sensitive property* (diagnostic).

name	ZIP code	age	diagnostic
Alice	4217	34	Common cold
Bob	4212	39	n/a
Camille	4732	39	HIV
Dede	4743	23	HIV

TABLE 2.8: A sample dataset with a sensitive column.

The sensitive property is not a quasi-identifier: it is the main information that we are trying to keep secret from the attacker, so it is reasonable to assume that the attacker does not know about it *a priori*. Let us use generalization to make this dataset 2-anonymous. The result can be found in Table 2.9: all combinations of ZIP code and age appear twice.

ZIP code	age	diagnostic
4210–4219	30–39	Common cold
4210–4219	30–39	n/a
4700–4799	20–39	HIV
4700–4799	20–39	HIV

TABLE 2.9: Dataset from Table 2.8, after generalization to satisfy 2-anonymity.

Now, assume the attacker wants to find Camille's diagnostic. They know that Camille has ZIP code 4732 and age 23. They can deduce that Camille's record is the third or the fourth, but cannot know which. However, it does not matter: both records have the same diagnostic. So the attacker can find out that Camille's diagnostic is HIV, even without reidentifying Camille's record! k-anonymity was not enough to protect the sensitive information.

To fix this, an alternative definition was proposed in [265]: l-diversity. It does not only mandate that each individual is indistinguishable from sufficiently many others, but also that within each group, there are multiple options for the sensitive value. In this context, the space of possible records is modeled by $\mathcal{T} = Q_1 \times \cdots \times Q_a \times C_1 \times \cdots C_b \times \mathcal{S}$; where $\mathcal{S}$ is an arbitrary set of possible sensitive values.

Definition 3 (l-diversity [265])**.** *Assume $\mathcal{T} = Q_1 \times \cdots \times Q_a \times C_1 \times \cdots C_b \times \mathcal{S}$. A database D is l-diverse if for every possible combination of quasi-identifier values $(q_1, \dots, q_a) \in Q_1 \times \cdots \times Q_a$ present in the dataset, there at least l distinct values $s_1, \dots, s_l \in \mathcal{S}$ associated with $(q_1, \dots, q_a)$ in D.*

Of course, l-diversity implies l-anonymity. In Table 2.10, we use generalization to make the dataset in Table 2.8 2-diverse. Consider our previous attacker, targeting Camille (third row). Like before, the attacker is unable to know which record in the 2-diverse dataset corresponds to Camille. Besides, they also cannot know whether Camille was healthy, or has been diagnosed with HIV. The sensitive value stays private.

ZIP code	age	diagnostic
4000–4999	20–34	Common cold
4000–4999	35–39	n/a
4000–4999	35–39	HIV
4000–4999	20–34	HIV

TABLE 2.10: Dataset from Table 2.8, after generalization to satisfy 2-diversity.

2.1.3.1 *l-diversity in practice*

Like k-anonymity, l-diversity is easy to compute, once the quasi-identifiers and the sensitive value have been chosen. The basic building blocks to transform a dataset into a l-diverse one are the same: generalization and suppression. Finding the best strategy is also done using trial-and-error heuristics. The approach used for k-anonymity is straightforward to adapt to l-diversity.

Since l-diversity is strictly stronger than k-anonymity (for $k = l$), it might seem like a natural replacement, always beneficial for privacy. However, the utility cost of l-diversity is usually much more significant than when using k-anonymity: some studies found that applying 3-diversity can be worse than using 100-anonymity, for a typical classification task [54]. This is one of the reasons why it is hardly ever used in practice.

2.1.3.2 *Limits of l-diversity*

Unsurprisingly, l-diversity shares similar difficulties than k-anonymity: choosing l is difficult, as is determining which columns are quasi-identifiers or which ones are sensitive. Even if we set aside these issues, another question looms: does l-diversity really protect the sensitive information? Consider the dataset in Table 2.11. Like the dataset from Table 2.10, it is 2-diverse. However, an attacker targeting the third row can still discover that their target has either Hepatitis B, or HIV. Some uncertainty is preserved, but the attacker still gains very sensitive information.

Another way l-diversity can fail is if by providing *probabilistic* information gain to the attacker. Consider the dataset in Table 2.12. An attacker trying to gain information on one of the rows could not know *for sure* what their target's diagnostic is, but they can significantly increase their suspicion that they have HIV. This is especially the case if we consider the background knowledge that the attacker might have on the sensitive value: for example, if the diagnostic of the first record was a condition that the attacker knows their target does not have.

How to protect against this type of probabilistic information gain? Requiring that sensitive attributes are diverse is not enough, we would need to also require that the *distribution* of

ZIP code	age	diagnostic
4000–4999	20–34	Common cold
4000–4999	35–39	Hepatitis B
4000–4999	35–39	HIV
4000–4999	20–34	HIV

Table 2.11: A different 2-diverse dataset that reveals sensitive information.

ZIP code	age	diagnostic
4000–4999	20–34	Common cold
4000–4999	20–34	HIV
4000–4999	20–34	HIV
4000–4999	20–34	HIV
4000–4999	20–34	HIV
4000–4999	20–34	HIV
4000–4999	20–34	HIV
4000–4999	20–34	HIV
4000–4999	20–34	HIV

Table 2.12: Another 2-diverse dataset that leaks probabilistic information.

sensitive values is roughly the same that the rest of the data. If 40% of the records are "healthy" in the overall data, then each bucket should also have roughly 40% of "healthy" records. This way, the attacker's knowledge cannot change too much from the baseline. This is the core idea behind another definition: t-closeness [251]. This definition is stricter than l-diversity, and unsurprisingly, comes with even greater utility loss. We do not formally introduce it here.

2.1.4 *δ-presence*

The core intuition behind k-map was that we assumed the attacker did not know who was in the dataset. Let us go back to this idea, with a slightly different scenario. Instead of a survey about sensitive information, like human sexual behavior, consider a clinical trial for a drug treating a particular condition, like HIV. The goal is still the same: safely share the data with other people.

These two settings look similar at first glance, but there is a crucial difference. Which information is sensitive, exactly? For the survey, the *answers* of each participant were sensitive, as they revealed intimate details. For the clinical study however, *being* in the dataset is the sensitive information: everyone in the clinical trial has been diagnosed with HIV. If an attacker finds out that their target has taken part in the study, they learn that their target suffers from this disease.

So, what does it change in practice? Let us assume that the dataset contains the records in Table 2.13. Following the same reasoning as for *k*-map, we can research the demographics of the population of ZIP code 85535. Table 2.14 contains the number of people living in this ZIP code, depending on age range.

ZIP code	age
85535	10
85535	12
85535	13
85535	13
85535	16
85535	43

TABLE 2.13: A sample dataset of participants to a clinical trial.

age	population
10–19	5
20–29	5
30–39	10
40–49	10
50–59	20
60+	15

TABLE 2.14: Hypothetical population of ZIP code 85535.

We could transform this part of this dataset to have it satisfy *k*-map. A possible strategy is listed in Table 2.15.

Is this strategy sufficient to successfully hide who participated in the survey? An attacker could know that there are 5 people aged between 10 and 19 in ZIP code 85535. Then, by looking at our de-identified dataset, the attacker can figure out that *all of them are part of the dataset*. Thus, they all have been diagnosed with HIV. Similarly to the discussion in Section 2.1.3, the attacker learned something sensitive about individuals, without reidentifying any record.

How to define a notion of anonymization that protects against this attack? Recall what we measured for our previous privacy definitions? For each combination of quasi-identifiers, we counted the number of records associated with this combination in the dataset (for *k*-

ZIP code	age
85535	10–19
85535	10–19
85535	10–19
85535	10–19
85535	10–19
85535	40–49

TABLE 2.15: A generalized version of the dataset from Table 2.13 which satisfies 5-map.

anonymity), or in a larger population (for k-map). In the previous example, a privacy issue arose because for some combination of quasi-identifier, these numbers were equal. To detect this, we instead compute and impose a bound on the *ratio* between those two numbers. This is the core intuition of δ-presence.

Definition 4 (δ-presence [297])**.** *Assume $\mathcal{T} = Q_1 \times \cdots \times Q_a \times C_1 \times \cdots C_b$. A database D satisfies δ-presence for a reidentification dataset $\mathcal{U}$ if for every possible combination of quasi-identifier values $(q_1, \ldots, q_a) \in Q_1 \times \cdots \times Q_a$, if this combination is present in k records in D and K records in $\mathcal{U}$, then $k/K < \delta$.*

The lower δ is, the stronger the definition becomes. The example above had $\delta = 1$: the ratio for the records with ZIP code 85535 and age range 10–19 is $5/5 = 1$. If $\delta = 0.95$, then similarly to the example in Section 2.1.3.2, the attacker might still get significant probabilistic information. The original definition was introduced with an additional *lower* bound on this ratio, in addition to the upper bound. This captures the intuition that the attacker should also not be able to find out that their target is *not* in the dataset. This latter concern is less frequent in practice, so we omitted it from the definition above for simplicity.

To make our dataset satisfy δ-presence for a lower δ, we could generalize the data further, as demonstrated in Table 2.16. This table satisfies 0.25-presence: the ratio for the records with age range 10–39 becomes $5/(5 + 5 + 10) = 0.25$, while ratio for the record with age 40–49 is still 0.1.

2.1.4.1 *δ-presence in practice*

δ-presence suffers from the same policy and applicability difficulties as k-map, listed in Sections 2.1.2.1 and 2.1.2.2: without access to the reidentification dataset $\mathcal{U}$, it is impossible to compute δ-presence exactly. Approximations are also possible, like the one proposed in [298], but they are difficult to use in practice: they require a statistical model of the reidentification dataset, and the proposed approximation algorithm is computationally costly.

ZIP code	age
85535	30–39
85535	30–39
85535	30–39
85535	30–39
85535	30–39
85535	40–49

TABLE 2.16: A generalized version of the dataset from Table 2.13 which satisfies 0.25-presence.

2.1.5 *Flaws of syntactic privacy definitions*

The four definitions mentioned in the previous section all seem to have serious limitations. Many of these attempts at defining anonymization feel like "patches" applied to prior notions to fix the most obvious flaws. However, it looks like any of these patches only creates more difficult questions, and does not suppress privacy issues entirely. Choosing when to apply each definition is also difficult: the definitions protect different aspects, so one has to think about what kind of attacker might want to attack the data before choosing the relevant definition. This choice itself is a difficult policy decision, and choosing the wrong attack model can have devastating consequences.

Perhaps more importantly, these definitions all share some fundamental issues. First, they all break down if the attacker has unexpected background knowledge. If the choice of quasi-identifier columns is wrong, and the attacker knows some auxiliary information that we did not expect, and all guarantees are lost. This background knowledge issue can also happen if the attacker has some information about the original database that we did not expect. For example, in l-diversity, if the attacker knows the information some other users in the dataset, they might reduce the number of possible options for the sensitive value. This problem is especially worrying if the attacker can potentially *influence* the data: k-anonymity, for example, is pointless if the attacker can simply add arbitrarily many elements to the dataset to have a specific quasi-identifier combination pass the threshold. This last attack vector might not be a major concern for medical data, but is realistic for e.g. online services, where a single individual could easily create fake accounts.

Second, they implicitly assume that the data release happens in isolation: the original data will not be published in any other way, nor will related data from different datasets. Controlling that a specific dataset is only released once might be feasible in theory. However, it is near-impossible to predict what might happen in the future, and invalidate a risk assessment made at a certain point in time, by some third-party. For example, consider two hospitals who independently release medical information using l-diversity, exemplified in Table 2.17. An attacker knowing that their target (with ZIP code 4217 and age 32) visited *both* hospitals can compare both releases, look at which diagnostic is the same in both, and deduce that their target has HIV.

ZIP code	age	diagnostic
4000–4999	20–34	Common cold
4000–4999	35–39	Otitis
4000–4999	35–39	HIV
4000–4999	20–34	HIV

ZIP code	age	diagnostic
4000–4999	35–39	HIV
4000–4999	35–39	Broken leg
4000–4999	20–34	n/a
4000–4999	20–34	Flu

TABLE 2.17: Two *l*-diverse datasets.

Such attacks are not theoretical: in [158], the authors experimentally show that this scenario can plausibly lead to such disclosures. Two distinct organizations publishing related data is not the only scenario in which this problem can arise: multiple k-anonymous views of the same data can also lead to k-anonymity violations [400], and the same problem can arise for multiple publications of the same dynamic dataset over time [395].

Third, the fact that a dataset satisfies a given privacy definition does not guarantee that the data release did not leak information that we were trying to protect. Indeed, consider an "arbitrary privacy definition" Def, and pick three arbitrary datasets D_1, D_2 and D_3, that all satisfy this privacy definition. Consider the following algorithm, built by an attacker who wants to find out private information about their target Camille.

- If Camille is in the dataset and has HIV, output D_1.
- Else, if Camille is in the dataset, output D_2.
- Else, output D_3.

This algorithm *always* produces an output that satisfies the required privacy definition. It is possible to instantiate Def as k-anonymity, l-diversity, δ-presence, or any other definition applying to datasets that we could think of. But of course, the algorithm also reveals private information that all definitions we previously listed tried to protect!

This example is, of course, not realistic: nobody would actually use an algorithm like this to anonymize their data. However, it demonstrates a deep issue with this class of definitions: knowing that an algorithm always outputs a dataset that satisfies a given definition is not enough to trust this algorithm not to inadvertently leak data. This is not only a theoretical concern, either: *minimality attacks* use characteristics of anonymization algorithms that adapt their strategy to optimize the utility of the output data [246, 247, 406] to reverse-engineer the original contents of the database [389].

For all definition seen so far, it is possible to check simply by looking at the output of an algorithm, whether this output satisfies the definition. Anonymity is seen as a *syntactic* property, which is why we call these definitions *syntactic privacy definitions*. How to overcome the fundamental issues of this approach? We need to change the perspective. Instead of considering anonymization as a property of the output dataset, we need to consider it as a property of the *process*, and come up with *semantic* formalizations of anonymity, that describe what information this process can potentially leak. This is one of the core insights behind differential privacy.

2.1.6 *Differential privacy*

The flaws of syntactic definitions pointed in the previous section are impossible to fix without a change in paradigm. Trying to find a criterion for a *database* to be anonymized is always going to be flawed, so we need a different approach. One of the core insights of differential privacy, which we define in this section, is that anonymization should be a property of the *process* instead. The definition should apply to the anonymization mechanism, and not to its output.

What property do we want an anonymization mechanism to have? Intuitively, we want it to not leak private information: an attacker with access to the output of the mechanism should not learn anything about an individual. A first attempt could be to require the mechanism $\mathcal{M}$ to be such that for any databases D_1 and D_2 that differ in only one record, $\mathcal{M}(D_1) = \mathcal{M}(D_2)$.

Why would such a definition accomplish our goal? For any single record of a database D, if we change or remove that record completely, we would still obtain the same output. This means that the output does not leak anything about the individual corresponding to this record: if it did, changing the record would change the output, and an attacker might be able to notice.

This definition corresponds to a *privacy goal* set out by Dalenius. In 1977, he defined *statistical disclosure* as follows.

> If the release of a statistic S makes it possible to determine the value [of a single record] more accurately than is possible without access to S, a disclosure has taken place [...]

The goal is to prevent this from happening. Our definition satisfies this goal: the output does not leak anything about individual records. The problem, of course, is that this definition is trivial. Any database D_2 can be obtained from any database D_1 by a series of "neighboring" databases, removing and adding records one by one as needed; so for *any* databases D_1 and D_2, $\mathcal{M}(D_1) = \mathcal{M}(D_2)$. It is not only impossible to learn anything about individual records by looking at the output, but it is impossible to learn anything about *the entire database*. Allowing the mechanism $\mathcal{M}$ to be randomized would not change anything: the equality would be between probability distributions, but the result would be the same.

In that sense, Dalenius' privacy goal is fundamentally impossible to achieve. If we cannot learn *anything* about individuals, then by extension, we cannot learn anything about the database either. Thus, we have to accept leaking *some* information about individuals, and try to quantify and limit this leakage of private information. This is what ε-differential privacy does: rather than requiring that $\mathcal{M}(D_1)$ is *exactly* the same as $\mathcal{M}(D_2)$, for neighboring D_1 and D_2, it requires that $\mathcal{M}(D_1)$ is *almost* the same as $\mathcal{M}(D_2)$. This notion is formalized via the concept of *ε-indistinguishability*.

Definition 5 (ε-indistinguishability [128][1])**.** *Two random variables A and B are ε-indistinguishable, denoted $A \approx_\varepsilon B$, if for all measurable sets X of possible events:*

$$\mathbb{P}\left[A \in X\right] \leq e^{\varepsilon} \cdot \mathbb{P}\left[B \in X\right] \text{ and } \mathbb{P}\left[B \in X\right] \leq e^{\varepsilon} \cdot \mathbb{P}\left[A \in X\right].$$

Informally, A and B are ε-indistinguishable if their distributions are "close"; and the smaller ε is, the closer A and B are. This notion can then be used to define differential privacy.

Definition 6 (ε-differential privacy [122, 128])**.** *A mechanism $\mathcal{M}$ is ε-differential private (or ε-DP) if for all datasets D_1 and D_2 that differ only in one record, $\mathcal{M}(D_1) \approx_\varepsilon \mathcal{M}(D_2)$.*

Just like k in k-anonymity, this definition depends on a numeric parameter ε. The smaller the ε, the stronger the definition; if $\varepsilon = 0$, the distributions are exactly the same, and as discussed above, $\mathcal{M}$ is trivial. Setting ε is, again, a non-trivial policy decision. However, this time, we can formally quantify the privacy leakage depending on the value of ε, and make statements such as "an attacker who thinks their target is in the dataset with probability 50% can increase their level of certainty to at most 75%".

How does this work? Let us first consider a simple example, called *randomized response*.

2.1.6.1 *A simple example: randomized response*

Interestingly, randomized response was introduced in 1965 [383], 40 years before differential privacy. It works as follows. Suppose we want to do a survey to know how many people are respecting social distancing guidelines during the COVID-19 pandemic. If we naively go out and ask people whether they are, many might feel uncomfortable answering honestly, and lie to us. So we can the following mechanism. The participants no longer directly answer the question "are you respecting social distancing guidelines?". Instead, each of them will flip a coin, without showing it to us.

- On heads, the participant tells the truth (YES or NO).
- On tails, they flip a second coin. If the second coin lands on heads, they answer YES. Otherwise, they answer NO.

How is this better for survey respondents? They can now answer YES without revealing that they are doing something illegal. When someone answers NO, we cannot know their true answer for sure. They could be breaking social distancing guidelines, but they might also have answered at random. Let us compute the probabilities of each answer someone who broke social distancing guidelines.

- With probability 50%, they will say the truth and answer NO.
- With probability 50%, they will answer at random.
  - They then have another 50% chance to answer YES, so 25% chance in total.

[1] This notion is similar to the cryptographic notion of indistinguishability [169], except the cryptographic version has $\varepsilon = 0$, and an additive error term instead, which is negligible. A similar notion, $(1, \varepsilon)$*-privacy*, was defined in [70], where $(1 + \varepsilon)$ used in place of e^{ε}, and it was also called *log-ratio distance* in [182]

- Similarly, in total, they have a 25% chance to answer No.

All in all, we get a 75% chance to answer No and a 25% chance to answer Yes. For someone who respecting social distancing guidelines, the probabilities are reversed: 25% chance to answer No and 75% to answer Yes. Calling this randomization mechanism $\mathcal{M}$:

$$\begin{aligned} \mathbb{P}\left[\mathcal{M}\left(\text{Yes}\right) = \text{Yes}\right] &= 0.75 & \qquad \mathbb{P}\left[\mathcal{M}\left(\text{Yes}\right) = \text{No}\right] &= 0.25 \\ \mathbb{P}\left[\mathcal{M}\left(\text{No}\right) = \text{Yes}\right] &= 0.25 & \qquad \mathbb{P}\left[\mathcal{M}\left(\text{No}\right) = \text{No}\right] &= 0.75 \end{aligned}$$

Now, 0.75 is three times bigger than 0.25. So if we choose ε such as $e^{\varepsilon} = 3$ (which corresponds to $\varepsilon \simeq 1.1$), this process is ε-differentially private. This intuitive notion of plausible deniability translates naturally in the language of differential privacy.

Of course, with a differentially private process like this one, we are getting some *noise* into our data: some answers are going to be random. But given enough answers, with high probability, the noise will cancel itself out. Suppose that we have 1000 answers in total: 400 of them are Yes and 600 are No. About 50% of all 1000 answers are random, so we can remove 250 answers from each count. In total, we get 150 Yes answers out of 500 non-random answers, so about 30% of Yes overall.

What if we want the process to offer a higher level of privacy? Instead of having the participants answer randomly with probability 50%, we can have them answer randomly 75% of the time. Conversely, if you want less noise instead, we could have them answer randomly only 25% of the time. There is a natural trade-off between privacy (for individuals) and utility (for the data), which is present in all applications of differential privacy.

How can we now translate this intuition into a formal statement about the information gain of an attacker? A relatively simple formalization is the *Bayesian* interpretation of differential privacy, generalizing an analysis that was originally done in the context of randomized response mechanisms [144].

2.1.6.2 *Formally quantifying the attacker's information gain*

Let us consider a more generic scenario than the previous example. We have a mechanism $\mathcal{M}$ which is ε-differentially private. We run it on some database D, and release the output $\mathcal{M}(D)$ to an attacker. Then, the attacker tries to find out whether their target is in D. This attack goal is similar to the one of δ-presence (see Section 2.1.4), but we could just as easily have the attacker try to find out the sensitive value associated to their target, or any other information about a single individual.

Under differential privacy, the attacker cannot gain a lot of information about their target. This is true even if this attacker has a lot of knowledge about the dataset. Let us take the stronger attacker we can think of: they know *all the database, except their target*. This attacker has to determine which database is the real one, between two options: one with their target in it (denote it D_{in}), the other without (D_{out}).

So, in the attacker's model of the world, the actual database D can be either D_{in} or D_{out}. They might have an *initial suspicion* that their target is in the database. This suspicion is represented by a Bayesian probability $\mathbb{P}\left[D = D_{in}\right]$. This probability can be anything between 0 and 1. Say, 0.9 if the attacker's suspicion is strong, 0.01 if they think it is very unlikely, 0.5 if they have no idea. Similarly, their suspicion that their target is *not* in the dataset is also a probability, $\mathbb{P}\left[D = D_{out}\right]$. Since there are only two options, $\mathbb{P}\left[D = D_{out}\right] = 1 - \mathbb{P}\left[D = D_{in}\right]$.

Now, suppose the attacker sees that the mechanism returns output O. How much information did the attacker gain? This is captured by looking at how much their suspicion changed after seeing this output. In mathematical terms, we have to compare the initial suspicion (the *prior*) $\mathbb{P}\left[D = D_{in}\right]$ with the updated suspicion $\mathbb{P}\left[D = D_{in} \,\middle|\, \mathcal{M}(D) = O\right]$ (the *posterior*). This posterior is the attacker's model of the world after seeing O. With ε-differential privacy, the posterior is never too far from the prior, and we can quantify this phenomenon exactly, as shown in Proposition 1.

Proposition 1. *Assume D is a database chosen randomly between D_1 and D_2. Then the posterior $\mathbb{P}\left[D = D_1 \,\middle|\, \mathcal{M}(D) = O\right]$ is bounded by:*

$$\frac{\mathbb{P}\left[D = D_1\right]}{e^{\varepsilon} + \left(1 - e^{\varepsilon}\right)\mathbb{P}\left[D = D_1\right]} \leq \mathbb{P}\left[D = D_1 \,\middle|\, \mathcal{M}(D) = O\right] \leq \frac{e^{\varepsilon} \cdot \mathbb{P}\left[D = D_1\right]}{e^{\varepsilon} + \left(e^{\varepsilon} - 1\right)\mathbb{P}\left[D = D_1\right]}.$$

Proof. First, we use Bayes' rule to express the posterior as a function of the prior:

$$\begin{aligned}\mathbb{P}\left[D = D_1 \,\middle|\, \mathcal{M}(D) = O\right] &= \frac{\mathbb{P}\left[D = D_1\right] \cdot \mathbb{P}\left[\mathcal{M}(D) = O \,\middle|\, D = D_1\right]}{\mathbb{P}\left[\mathcal{M}(D) = O\right]} \\ &= \frac{\mathbb{P}\left[D = D_1\right] \cdot \mathbb{P}\left[\mathcal{M}(D_1) = O\right]}{\mathbb{P}\left[\mathcal{M}(D) = O\right]}.\end{aligned}$$

Then, we divide the posterior on D_1 and the posterior on D_2 to make the unknown $\mathbb{P}\left[\mathcal{M}(D) = O\right]$ term disappear:

$$\frac{\mathbb{P}\left[D = D_1 \,\middle|\, \mathcal{M}(D) = O\right]}{\mathbb{P}\left[D = D_2 \,\middle|\, \mathcal{M}(D) = O\right]} = \frac{\mathbb{P}\left[D = D_1\right]}{\mathbb{P}\left[D = D_2\right]} \cdot \frac{\mathbb{P}\left[\mathcal{M}(D_1) = O\right]}{\mathbb{P}\left[\mathcal{M}(D_2) = O\right]}.$$

Note that the right-most term are the same as in the definition of differential privacy, so we have:

$$e^{-\varepsilon} \leq \frac{\mathbb{P}\left[\mathcal{M}(D_1) = O\right]}{\mathbb{P}\left[\mathcal{M}(D_2) = O\right]} \leq e^{\varepsilon}.$$

Plugging this into the previous formula, we get:

$$e^{-\varepsilon}\frac{\mathbb{P}\left[D = D_1\right]}{\mathbb{P}\left[D = D_2\right]} \leq \frac{\mathbb{P}\left[D = D_1 \,\middle|\, \mathcal{M}(D) = O\right]}{\mathbb{P}\left[D = D_2 \,\middle|\, \mathcal{M}(D) = O\right]} \leq e^{\varepsilon}\frac{\mathbb{P}\left[D = D_1\right]}{\mathbb{P}\left[D = D_2\right]}. \tag{2.1}$$

Replacing $\mathbb{P}\left[D = D_2\right]$ by $1 - \mathbb{P}\left[D = D_1\right]$, doing the same for $\mathbb{P}\left[D = D_2 \,\middle|\, \mathcal{M}(D) = O\right]$, and solving for $\mathbb{P}\left[D = D_1 \,\middle|\, \mathcal{M}(D) = O\right]$, leads to the desired result. □

Note that the ratio of probabilities that appeared in the proof of Proposition 1 a simple interpretation: those are what gamblers call *betting odds*. In a sport game of team A vs. team B, betting odds of 2:1 means that according to bookies, the probability for A to win is twice as much as for B. This corresponds to probabilities of about 67% and 33%, respectively. In

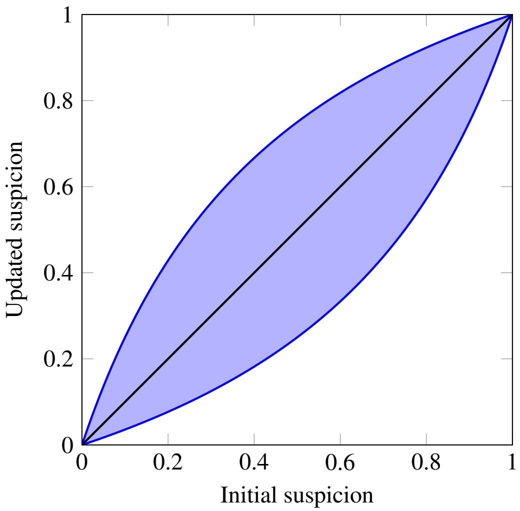


FIGURE 2.1: Prior-posterior bounds for ε-DP with $\varepsilon = \ln 3$.

particular, Equation 2.1 show that an alternative interpretation of differential privacy is that *betting odds* cannot change much after knowing the output of a DP mechanism.

For example, with $\varepsilon = \ln 3$, the upper and lower bounds for the posterior as a function of the prior are depicted in Figure 2.1. The black line is what happens if the attacker did not get their prior updated at all. The blue lines are the lower and upper bounds on the updated suspicion: it can be anywhere between the two. We can visualize the example mentioned in the previous section: for an initial suspicion of 50%, the updated suspicion is approximately between 25% and 75%.

This property can be used to convey the meaning of ε to people who are not experts. An ε of $\ln 3$ can be said to guarantee that at most, the attacker "halves their uncertainty". Similarly, $\varepsilon = \ln 2$ can "reduce the uncertainty by one third", while $\varepsilon = \ln 4$ can "reduce it by two thirds". Though not very formal, this formulation is understandable by most, and has the advantage of creating appropriate rejection reactions to large values of ε.

What does it look like for various values of ε? In Figure 2.2, we visualize the generalization of the bounds in Figure 2.2. For larger values of ε, the information gain increases fast: for example, with $\varepsilon = 5$. Then, an attacker can go from a small suspicion (say, 10%) to a very high degree of certainty (94%).

2.1.6.3 *Properties of differential privacy*

Differential privacy has several properties that make it a more convincing and convenient definition of anonymization than other definitions seen so far. First, many problems we pointed out in syntactic definitions do not apply to differential privacy. The definition protects *anything* associated with a single individual, including presence in the database and

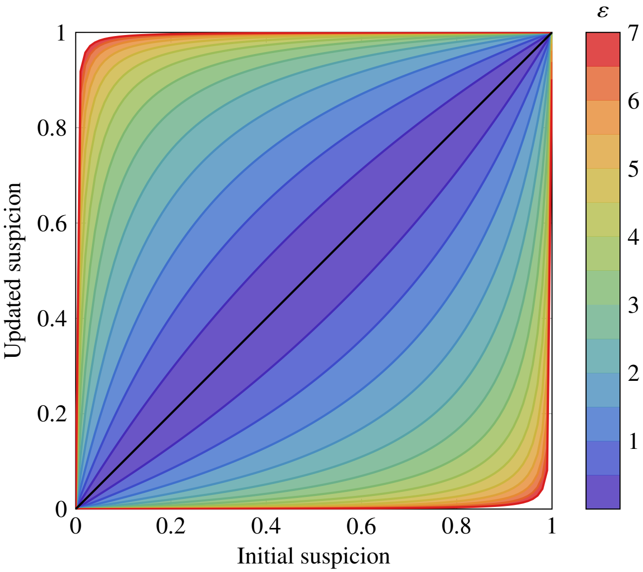


FIGURE 2.2: Prior-posterior bounds for ε-DP, for various values of ε.

sensitive information. Differential privacy also protects against an attacker that has unlimited background knowledge: D_1 and D_2 can be arbitrary neighboring databases, so the attacker could even choose all records and only have uncertainty about their target. These two aspects greatly simplify policy choices: a manual assessment of the attacker's goals and capabilities is no longer required.

As proven in the previous section, and explored in more detail in Section 2.2, differential privacy allows us to prove *semantic* properties: it limits the amount of information that an attacker can gain, no matter what attack method is used and what kind of output the mechanism generates. It does not matter what exact form the output takes: it can be a statistic, synthetic data, a machine learning model, etc.

In addition, differential privacy has a few properties that make it convenient to use. Its semantic guarantees are preserved by transformations that happen after the differential privacy stage. This property, called *post-processing*, makes it easy to reason about complex data pipelines that involve an anonymization step: everything downstream of the DP step stays DP.

Proposition 2 (Post-processing of differential privacy [30])**.** *Let $\mathcal{M}$ be an ε-DP mechanism, and let f be an arbitrary function. Then the function $D \rightarrow f\left(\mathcal{M}\left(D\right)\right)$ is also ε-DP.*

Differential privacy also satisfies *convexity*, a second property which is less crucial to practical use cases, but is another indication that DP has the kind of semantic guarantees that we expect: choosing between two DP mechanisms at random still leads to a DP mechanism.

Proposition 3 (Convexity of differential privacy [225, 226])**.** *Let $\mathcal{M}_1$ and $\mathcal{M}_2$ be two ε-DP mechanisms, and $p \in (0,1)$. The mechanism $\mathcal{M}$ defined by $\mathcal{M}(D) = \mathcal{M}_1(D)$ with probability p, and $\mathcal{M}(D) = \mathcal{M}_2(D)$ with probability $1-p$, is ε-DP.*

Differential privacy also *composes* well: combining the result of two DP mechanisms is still DP. This property, especially combined with post-processing, is how most complex DP algorithms are built: by assembling simpler mechanisms. For example, a DP algorithm for sums can be combined with a DP algorithm for counts to create a DP algorithm for average. There are several types of composition: *parallel composition*, *sequential composition*, and *adaptive composition*. We introduce the first two below.

Proposition 4 (Parallel composition [122])**.** *Let $\mathcal{M}_1$ be a ε_1-DP mechanism, and $\mathcal{M}_2$ a ε_2-DP mechanism. For any dataset D, let D_1 and D_2 be the result of an operation that separates records in two disjoint datasets. Then the mechanism $\mathcal{M}$ defined by $\mathcal{M}(D) = (\mathcal{M}_1(D_1), \mathcal{M}_2(D_2))$ is $(\max(\varepsilon_1, \varepsilon_2))$-DP.*

This property allows us to build *locally differentially private* mechanisms, in which a central server can compute global statistics without accessing the raw data from each individual. In this thesis, we mostly focus on sequential composition, which we simply call *composition*.

Proposition 5 (Sequential composition [122])**.** *Let $\mathcal{M}_1$ be a ε_1-DP mechanism, and $\mathcal{M}_2$ a ε_2-DP mechanism. Then the mechanism $\mathcal{M}$ defined by $\mathcal{M}(D) = (\mathcal{M}_1(D), \mathcal{M}_2(D))$ is $(\varepsilon_1 + \varepsilon_2)$-DP.*

Note that the formula in Proposition 5 is not tight, especially when many mechanisms are composed. A significant body of research focuses on obtaining tighter composition bounds, either with or without knowledge of mechanisms used for composition [57, 132, 133, 213, 278, 284]. Proposition 5 still holds if $\mathcal{M}_2$ depends on the value of $\mathcal{M}_1(D)$: this variant is called *adaptive composition*. This latter property allows to quantify the gain of information over time of an attacker interacting with a differentially private query engine. Proposition 5 still holds for sequential composition; most of the tighter results also do, with some exceptions [114].

Data privacy definitions that are a property of a mechanism and not of the output dataset were already proposed in as early as 2003 [144]. However, thanks to its useful properties, differential privacy quickly became the flagship of data privacy definitions. Thousands of scientific papers have explored this notion and its variants, or introduced algorithms that satisfy it. Large organizations are also investing in differential privacy research and exploring the use of differential privacy for practical use cases: examples include the US Census Bureau [4, 159], Google [141], Apple [360], Microsoft [107], LinkedIn [224], Uber [210], etc.

2.1.6.4 *The Laplace mechanism*

Making simple statistics differentially private is relatively easy. Suppose, for example, that we have a database collecting people's eye colors, and we want to publish how many people have green eyes? Publishing the true answer is not differentially private, even if a lot of people have green eyes. Indeed, with differential privacy, we need to assume that the attacker knows almost all records, and only has uncertainty about their target. In this example, it means that they already know the eye color of everyone in the dataset, except their target. So if we output the real number k, they can compare it with the number of people with green eyes among the people they know. If it is $k-1$, then the target has green eyes. If it is k, then the target does not.

What can we do to prevent this? We can add *noise* to the real statistic: instead of returning the true answer, we generate a random number from a certain probability distribution, add this number to the original answer, and return this noisy sum. The most common probability distribution used in differential privacy is called the *Laplace distribution*. This distribution has a parameter, its *scale*, which determines how "flat" it is.

Definition 7 (Laplace distribution)**.** *The* Laplace distribution of mean 0 and of scale $b > 0$ *is the probability distribution with probability density function:*

$$\text{PDF}(x) = \frac{1}{2b}\exp\left(-\frac{|x|}{b}\right).$$

Figure 2.3 shows the probability density function of a Laplace distribution of scale $b = \frac{1}{\ln 3}$.

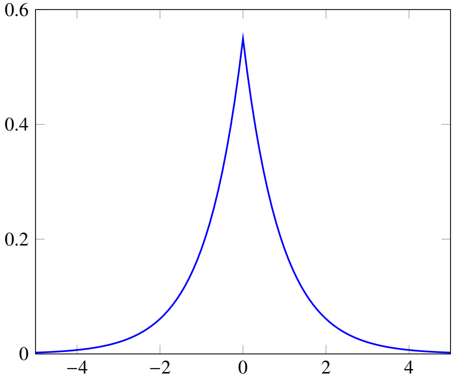


FIGURE 2.3: Laplace distribution of scale $b = \frac{1}{\ln 3}$.

To make our statistic above ε-DP, we pick a random value according to a Laplace distribution of scale $\frac{1}{\varepsilon}$, and we add it to the real value. Figure 2.4 gives a visual intuition of why it works. It shows the distribution of the number we return, depending on whether the true count is 1000 (blue line, the target does not have green eyes) or 1001 (red line, the target has green eyes).

Suppose the real number is 1001, and we returned 1003. From the attacker's perspective, what is the likelihood that the original value is 1000 vs. 1001? The hypothesis "the true

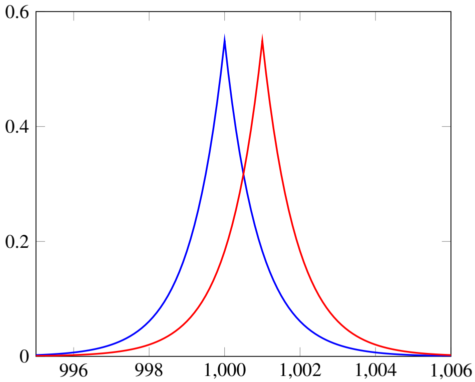


Figure 2.4: Laplace distribution of scale $b = \frac{1}{\ln 3}$ added to 1000 (blue) and 1001 (red).

answer is 1001" is a bit more likely: generating a noise of 2 is more likely than a noise of 3. How much more likely? The *ratio* of probability density functions is exactly:

$$\begin{aligned}\frac{\text{PDF}(2)}{\text{PDF}(3)} &= \frac{\frac{1}{2b}\exp\left(-\frac{|2|}{b}\right)}{\frac{1}{2b}\exp\left(-\frac{|3|}{b}\right)} \\ &= \exp\left(\left(-2+3\right)\ln 3\right) \\ &= 3.\end{aligned}$$

We can perform the same computation to compare any two events where we add a noise of k vs. $k+1$ for any k, and it is easy to check that the shape of the Laplace distribution guarantees that the ratio will always be between -3 and 3. This is exactly the guarantee we need for ε-DP, with $\varepsilon = \ln 3$: adding Laplace noise of scale $\frac{1}{\varepsilon}$ is enough to guarantee ε-DP for cases where we count unique individuals.

What if we want to compute and release more complicated statistics? For example, we might have a dataset of suggestions that your customers sent us using a feedback form, where each customer can use the form at most 5 times. We want to publish the number of suggestions we received in total, while the attacker wants to get an idea of how many suggestions their target published.

Adding noise picked from a Laplace distribution of scale $\frac{1}{\varepsilon}$ is not enough to get ε-DP. Indeed, assume that the attacker's target sent 5 suggestions, while our other 1000 customers sent one suggestion each. The influence of the target will be larger than before, as shown in Figure 2.5. The difference between the two curves is much larger than before; their ratio is at most $e^{5\varepsilon}$: using Laplace noise of scale $\frac{1}{\varepsilon}$ only gives 5ε-DP.

To fix this, we need to add more noise. How much more? It depends on the *maximum contribution* of one individual customer. If the maximum amount of suggestions is 5, we must add 5 times the amount of noise. In this example, using Laplace noise of scale $\frac{5}{\varepsilon}$ would give us ε-DP; we display this in Figure 2.6.

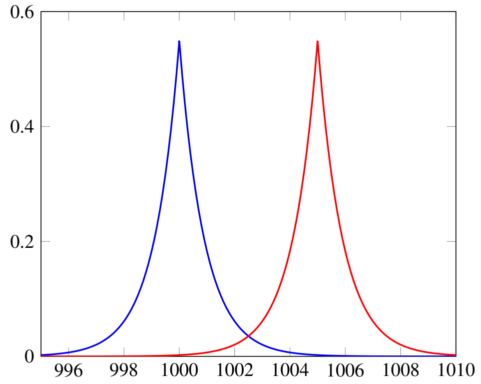


FIGURE 2.5: Laplace distribution of scale $b = \frac{1}{\ln 3}$ added to 1000 (blue) and 1005 (red).

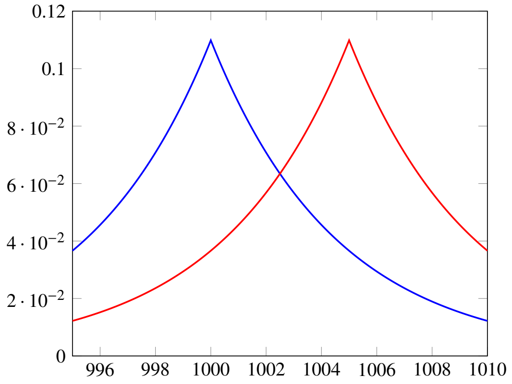


FIGURE 2.6: Laplace distribution of scale $b = \frac{5}{\ln 3}$ added to 1000 (blue) and 1005 (red).

This informal reasoning can be extended to a wider class of statistics. The maximum contribution of an individual is called the *global sensitivity*, and adding Laplace noise scaled by this global sensitivity provides differential privacy. This method for making arbitrary statistics differentially private, which we formalized below, is called the *Laplace mechanism*.

Proposition 6 (Proposition 1 in [128])**.** *Let $f : \mathcal{D} \rightarrow \mathbb{R}$ be a deterministic function. The* global sensitivity *of f is the smallest number s such that for all datasets D_1 and D_2 differing only in one record:*

$$\left| f\left(D_1\right) - f\left(D_2\right) \right| \leq s.$$

Furthermore, the mechanism $\mathcal{M}$ defined by $\mathcal{M}(D) = f(D) + X$, where X is a random variable sampled from a Laplace distribution of scale $\frac{s}{\varepsilon}$, is ε-differentially private.

Note that if we want to protect individuals, we need to be careful with the definition of a *record*: all suggestions from the same customer must be part of the same record. We come back to these considerations in Section 2.2.3. In Section 4.2, we also explain how to use this basic building block for a wide range of practical applications.

When releasing integers, instead of the Laplace mechanism, we can use a discrete distribution instead: the *geometric mechanism.*

Definition 8 (Two-sided geometric distribution)**.** *The* two-sided geometric distribution of mean 0 and of parameter $p \in (0, 1)$ *is the probability distribution such that a random variable X sampled from the distribution follows, for all $k \in \mathbb{Z}$:*

$$\mathbb{P}[X = k] = \frac{1-p}{1+p} p^{|k|}.$$

Proposition 7 ([166])**.** *Let $f : \mathcal{D} \rightarrow \mathbb{Z}$ be a deterministic function of global sensitivity $s \in \mathbb{N}$. The mechanism $\mathcal{M}$ defined by $\mathcal{M}(D) = f(D) + X$, where X is a random variable sampled from the two-sided geometric distribution of mean* 0 *and of parameter p, is* $\ln\left(\frac{1}{p}\right)$*-differentially private.*

Using the geometric distribution for integer-valued data has multiple advantages: it avoids unexpected vulnerabilities from floating-point implementations [283], returns results that are easier to understand for non-experts (which might be surprised by a non-integer result for e.g. a count of unique people), and has slightly better utility than simply rounding the result of a Laplace distribution.

2.2 SYSTEMATIZING VARIANTS & EXTENSIONS OF DIFFERENTIAL PRIVACY

Since the original introduction of differential privacy, many variants and extensions have been proposed to adapt it to different contexts or assumptions. These new definitions enable practitioners to get privacy guarantees, even in cases that the original DP definition does not cover well. This happens in a variety of scenarios: the noise mandated by DP can be too large and force the data custodian to consider a weaker alternative, the risk model might be inappropriate for certain use cases, or the context might require the data owner to make stronger statements on what information the privacy mechanism can reveal.

Figure 2.7 shows the prevalence of this phenomenon: approximately *225* different notions[2], inspired by DP, were defined in the last 15 years. As we show in Figure 2.7, this phenomenon does not seem to slow down over time. These definitions can be *extensions* or *variants* of DP. An extension encompasses the original DP notion as a special case, while a variant changes some aspect, typically to weaken or strengthen the original definition.

With so many definitions, it is difficult for new practitioners to get an overview of this research area. Many definitions have similar goals, so it is also challenging to understand which are appropriate to use in which context. These difficulties also affect experts: a number of definitions have been defined independently multiple times (often with identical meaning but different names, or identical names but different meanings). Finally, variants are often introduced without a comparison to related notions.

In this section, we attempt to solve these problems by systematizing the scientific literature on variants and extensions of differential privacy. We propose a unified and comprehensive

[2] We count all the definitions which are presented as “new” in the papers introducing them.

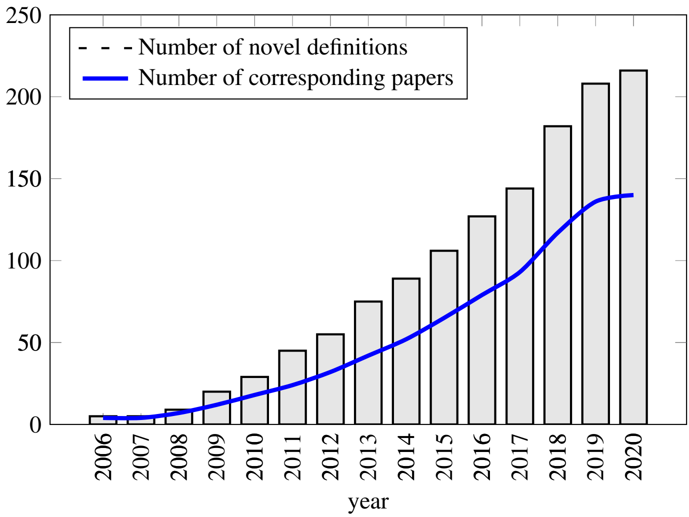


FIGURE 2.7: Accumulated number of papers which are introducing DP variants or extensions (line) and the exact number of these definitions (bar).

taxonomy of these variants and extensions, providing short explanations of the intuition, use cases and basic properties of each. By categorizing these definitions, we attempt to simplify the understanding of existing variants and extensions, and of the relations between them. We hope to make it easier for new practitioners to understand whether their use case needs an alternative definition, and if so, which existing notions are the most appropriate, and what their basic properties are.

We define seven *dimensions*: these are ways in which the original definition of DP can be modified or extended. We list variants and extensions that belong to each dimension, and we highlight representative definitions for each. Whenever possible, we compare these definitions and establish a partial ordering between the strengths of different notions. Furthermore, for each definition, we specify whether it satisfies Kifer et al.'s *privacy axioms* [225, 226] (post-processing and convexity), and whether they are composable.

This section is organized as follows.

- In Section 2.2.1, we introduce our dimensions along which DP can be modified, we present basic properties of privacy definitions, we define how definitions can relate to each other, and explain our methodology for this literature review.

- In the following 7 sections (Sections 2.2.2 to 2.2.8), we introduce our dimensions, and list and compare the corresponding definitions.

- In Section 2.2.9, we summarize the results from the previous sections into a table, showing the corresponding properties with proofs, and list the known relations.

- In Section 2.2.10, we mention related concepts and definitions which were considered out of scope for this work, and review the related literature.

2.2.1 *Preliminaries*

In this section, we introduce the notations used throughout this work, formally introduce basic properties of privacy definitions, define how definitions can relate to each other, introduce our dimensions along which DP can be modified, and present the methodology used in this literature review.

Throughout Section 2.2, we denote by $\mathcal{T}$ an arbitrary set of possible records, and by $\mathcal{D}$ the space of possible datasets; where a dataset is a finite indexed family of records. We typically use t or t' to denote records, and D, D', D_1 or D_2 to denote datasets. The indices of a dataset are typically called i and j, with $D(i)$ referring to the i-th record of a dataset D. We denote by D_{-i} the dataset D whose i-th record has been removed.

Let $\mathcal{O}$ denote an arbitrary set of possible *outputs*; outputs are typically called O, and sets of outputs called S. A *mechanism* is a function, possibly randomized, which takes a dataset as input and returns an output. Mechanisms are typically called $\mathcal{M}$, while $\mathcal{M}(D)$ is usually a random variable.

Probability distributions on $\mathcal{T}$ are called π, probability distribution on $\mathcal{D}$ are called θ, and family of probability distributions on $\mathcal{D}$ are called Θ. Given some event A, let $\mathcal{M}(D)_{|D\sim\theta,A}$ denote the random variable corresponding to the output of $\mathcal{M}(D)$, when D is drawn from a distribution θ conditioned on A. These notations, and others used throughout this section, are summarized in the tables on pages xix–xx.

2.2.1.1 *Dimensions*

Variants and extensions of differential privacy modify the original definition in various ways. To establish a comprehensive taxonomy, a natural approach is to partition them into *categories*, depending on which aspect of the definition they change. Unfortunately, this approach fails for privacy definitions, many of which modify several aspects at once, so it is impossible to have a categorization such that every definition falls neatly into only one category.

The approach we take is to define *dimensions* along which the original definition can be modified. Each variant or extension of DP can be seen as a point in a multidimensional space, where each coordinate corresponds to one possible way of changing the definition along a particular dimension. To make this representation possible, our dimensions need to satisfy the following two properties.

- *Mutual compatibility*: definitions that vary along different dimensions can be combined to form a new, meaningful definition.
- *Inner exclusivity*: definitions in the same dimension cannot be combined to form a new, meaningful definition (but they can be pairwise comparable).

In addition, each dimension should be *motivatable*: there should be an intuitive explanation of what it means to modify DP along each dimension. Moreover, each possible choice within a dimension should be similarly understandable, to allow new practitioners to determine quickly which kind of definition they should use or study, depending on their use case.

We introduce our dimensions by reformulating the guarantee offered by DP, highlighting aspects that have been modified by its variants or extensions. Each dimension is attributed a letter, and we note the dimension letter corresponding to each highlight. This formulation considers the point of view of an attacker, trying to find out some sensitive information about some input data using the output of a mechanism.

An attacker with **perfect background knowledge** (**B**) and **unbounded computation power** (**C**) is **unable** (**R**) to **distinguish** (**F**) **anything about an individual** (**N**), **uniformly across users** (**V**) even in the **worst-case scenario** (**Q**).

This informal definition of DP with the seven highlighted aspects give us seven distinct dimensions. We denote each one by a letter and summarize them in Table 2.18. Each is introduced in its corresponding section.

Dimension	Description	Typical motivations
Quantification of privacy loss	How is the privacy loss quantified across outputs?	Averaging risk, obtaining better composition properties
Neighborhood definition	Which properties are protected from the attacker?	Protecting specific values or multiple individuals
Variation of privacy loss	Can the privacy loss vary across inputs?	Modeling users with different privacy requirements
Background knowledge	How much prior knowledge does the attacker have?	Using mechanisms that add less or no noise to data
Formalism change	Which formalism describes the attacker's knowledge gain?	Exploring other intuitive notions of privacy
Relativization of knowledge gain	What is the knowledge gain relative to?	Guaranteeing privacy for correlated data
Computational power	How much computational power can the attacker use?	Combining cryptography techniques with DP

TABLE 2.18: The seven dimensions and their typical motivation.

Note that the interpretation of DP is subject to some debate. In [368], authors summarize this debate, and show that DP can be interpreted under two possible lenses: it can be seen as an *associative* property, or as a *causal* property. The difference between the two interpretations is particularly clear when one supposes that the input dataset is modeled as being generated by a probability distribution.

- In the associative view, this probability distribution is *conditioned* upon the value of one record. If the distribution has correlations, this change can affect other records as well.
- In the causal view, the dataset is first generated, and the value of one record is then *changed* before computing the result of the mechanism.

While the causal view does not require any additional assumption to capture the intuition behind DP, the associative view requires that either all records are independent in the original probability distribution (the *independence assumption*), or the adversary must know all data points except one (the *strong adversary assumption*, which we picked in the reformulation above). These considerations can have a significant impact on DP variants and extensions, either leading to distinct variants that attempt to capture the same intuition, or to the same variant being interpreted in different ways; in this work, we point it out when the distinction between the two interpretations is particularly relevant to a specific variant.

2.2.1.2 *Properties of privacy definitions*

In this section, we introduce three abstract, desirable properties of data privacy definitions. As shown in Section 2.1.6.3, these are satisfied by differential privacy itself. We will see in this work that these often, but not always, satisfied by its variants and extensions.

Two important properties of data privacy notions are called *privacy axioms*, proposed in [225, 226]. These are not axioms in a sense that they assumed to be true; rather, they are consistency checks: properties that, if not satisfied by a data privacy definition, indicate a flaw in the definition[3].

Definition 9 (Privacy axioms [225, 226])**.**

1. Post-processing[4] *(or* transformation invariance*): A privacy definition* Def *satisfies the post-processing axiom if, for any mechanism $\mathcal{M}$ satisfying* Def *and any probabilistic function f, the mechanism $D \to f(\mathcal{M}(D))$ also satisfies* Def.
2. Convexity *(or* privacy axiom of choice*): A privacy definition* Def *satisfies the convexity axiom if, for any two mechanisms $\mathcal{M}_1$ and $\mathcal{M}_2$ satisfying* Def*, the mechanism $\mathcal{M}$ defined by $\mathcal{M}(D) = \mathcal{M}_1(D)$ with probability p and $\mathcal{M}(D) = \mathcal{M}_2(D)$ with probability $1-p$ also satisfies* Def.

A third important property is, as we mentioned in Section 2.1.6.3 one of differential privacy's main strengths: *composability*. We saw that there were three types of composition; in this survey, we focus on sequential composition, which we simply call *composition*.

Definition 10 (Composability)**.** *A privacy definition* Def *with parameter α is* composable *if for any two mechanisms $\mathcal{M}_1$ and $\mathcal{M}_2$ satisfying respectively α_1-*Def *and α_2-*Def*, the mechanism $\mathcal{M}(D) = (\mathcal{M}_1(D), \mathcal{M}_2(D))$ satisfies α-*Def *for some (non-trivial) α.*

[3] The necessity of these were questioned in [202], where authors showed a natural notions of anonymity that contradict them.

[4] This definition must be slightly adapted for some variants, see for example Proposition 17 in Section 2.2.9.

We highlight the properties satisfied by each variant and extension of DP in Table 2.20 in Section 2.2.9.

2.2.1.3 *Relations between definitions*

When learning about a new data privacy notion, it is often useful to know what are the known relations between this notion and other definitions. However, definitions have parameters that often have different meanings, and whose value is not directly comparable. To capture extensions, when a definition can be seen as a special case of another, we introduce the following definition.

Definition 11 (Extensions)**.** *Let* α-Def_1 *and* β-Def_2 *be data privacy definitions. We say that* Def_1 *is* extended by Def_2*, and denote is as* $\text{Def}_1 \subset \text{Def}_2$*, if for all* α*, there is a value of* β *such that* α-Def_1 *is identical to* β-Def_2*.*

Concerning variants, to claim that a definition is stronger than another, we adopt the concept of ordering established in [88] using α and β as tuples, encoding multiple parameters. We slightly changed the original definition: the original only required the second condition to hold, which would classify any extension as a stronger variant.

Definition 12 (Relative strength of privacy definitions)**.** *Let* α-Def_1 *and* β-Def_2 *be data privacy definitions. We say that* Def_1 *is* stronger than Def_2*, and denote it* $\text{Def}_1 \succ \text{Def}_2$*, if:*

1. *for all* α*, there is a* β *such that* α-$\text{Def}_1 \implies \beta$-$\text{Def}_2$*;*

2. *for all* β*, there is an* α *such that* α-$\text{Def}_1 \implies \beta$-$\text{Def}_2$*.*

If Def_1 *is both stronger than and weaker than* Def_2*, we say that the two definitions are* equivalent*, and denote it* $\text{Def}_1 \sim \text{Def}_2$*.*

Relative strength implies a partial ordering on the space of possible definitions. On the other hand, if two definitions are equivalent, this does not mean that they are equal: they could be only equal up to a change in parameters. Both relations are reflexive and transitive; and we define the symmetric counterpart of these relations as well (i.e., $\prec$ and $\supset$). Moreover, for brevity, we combine these two concepts in a single notation: if $\text{Def}_1 \subset \text{Def}_2$ and $\text{Def}_1 \succ \text{Def}_2$, we say that Def_2 is a weaker extension of Def_1, and denote it $\text{Def}_1 \subset^{\succ} \text{Def}_2$.

A summarizing table is presented in Section 2.2.9 (Table 2.20). For each definition, we highlight its dimensions and its relation to other notions, and we also specify whether these notions satisfy the privacy axioms and the composability property (✓: yes, ✗: no, ?: currently unknown). In Section 2.2.9.1, we either provide a reference or a novel proof for each of these claims.

2.2.1.4 *Methodology*

Whether a data privacy definition fits our description is not always obvious, so we use the following criterion: the attacker's capabilities must be clearly defined, and the definition must prevent this attacker from learning about a protected property. Consequently, we do not consider:

- syntactic definitions, which are a property of the output data and not of the mechanism;
- variants of technical notions that are not data privacy properties;
- definitions whose only difference with DP is in the context and not in the formal property, like the distinction between local and central models of differential privacy.

In Section 2.2.10.1, we give a list of notions that we found during our survey, and considered to be out of scope for our work.

To find a comprehensive list of DP notions, besides the definitions we were aware of or were suggested to us by experts, we conducted a wide literature review using two research datasets: BASE [34] and Google Scholar [172]. The exact queries were run on 2020–06–01. The corresponding result counts are summarized in Table 2.19.

Query (BASE)	Hits
"differential privacy" relax year:[2000 to *]	130
"differential privacy" variant -relax year:[2000 to *]	115
a	b

Query (Google Scholar)	Hits
"differential privacy" "new notion"	224
"differential privacy" "new definition" -"new notion"	180

TABLE 2.19: Queries for the literature review.

First, we manually reviewed each paper and filtered them out until we had only papers which either contained a new definition or were applying DP in a new setting. All papers which defined a variant or extension of DP are cited in this work.

2.2.2 *Quantification of privacy loss (Q)*

The risk model associated to differential privacy is a *worst-case* property: it quantifies not only over all possible neighboring datasets but also over all possible outputs. However, in many real-life risk assessments, events with vanishingly small probability are ignored, or their risk weighted according to their probability. It is natural to consider analogous relaxations, especially since these relaxations often have better composition properties, and enable natural mechanisms like the Gaussian mechanism to be considered private [131].

Most of the definitions within this section can be expressed using the *privacy loss random variable*, first defined in [110] as the *adversary's confidence gain*, so we first introduce this concept. Roughly speaking, it measures how much information is revealed by the output of a mechanism.

Definition 13 (Privacy loss random variable [110]). *Let $\mathcal{M}$ be a mechanism, and D_1 and D_2 two datasets. The* privacy loss random variable between $\mathcal{M}(D_1)$ and $\mathcal{M}(D_2)$ *is defined as:*

$$\mathcal{L}_{\mathcal{M}(D_1)/\mathcal{M}(D_2)}(O) = \ln\left(\frac{\mathbb{P}\left[\mathcal{M}(D_1) = O\right]}{\mathbb{P}\left[\mathcal{M}(D_2) = O\right]}\right).$$

if neither $\mathbb{P}\left[\mathcal{M}(D_1) = O\right]$ nor $\mathbb{P}\left[\mathcal{M}(D_2) = O\right]$ is 0; in case only $\mathbb{P}\left[\mathcal{M}(D_2) = O\right]$ is zero then $\mathcal{L}_{\mathcal{M}(D_1)/\mathcal{M}(D_2)}(O) = \infty$, otherwise $\mathcal{L}_{\mathcal{M}(D_1)/\mathcal{M}(D_2)}(O) = -\infty$. When the mechanism is clear from context, we simply write $\mathcal{L}_{D_1/D_2}$.

For simplicity, we only consider the case where the set of possible outputs of the mechanism, $\mathcal{O}$, is countable; otherwise the PLRV can be reformulated using the density functions of $\mathcal{M}(D_1)$ and $\mathcal{M}(D_2)$ instead of specific outputs.

Differential privacy bounds the *maximum value* of $\mathcal{L}_{D_1/D_2}$. Instead of considering the maximum value, which corresponds to the worst possible output, relaxations of this section will allow a small probability of error, consider the average of the privacy loss random variable, or describe its behavior in finer ways.

2.2.2.1 *Allowing a small probability of error*

The first option, whose introduction is commonly attributed to [126], relaxes the definition of ε-indistinguishability by allowing an additional small density of probability on which the upper ε bound does not hold. This small density, denoted δ, can be used to compensate for outputs for which the privacy loss is larger than e^{ε}. This led to the definition of *approximate differential privacy*, often simply called (ε, δ)-DP. This is, by far, the most commonly used relaxation in the scientific literature.

Definition 14 ((ε, δ)-differential privacy [126]). *Two random variables A and B are (ε, δ)-*indistinguishable*, denoted $A \approx_{\varepsilon,\delta} B$, if for all measurable sets X of possible events:*

$$\mathbb{P}\left[A \in X\right] \leq e^{\varepsilon} \cdot \mathbb{P}\left[B \in X\right] + \delta \text{ and } \mathbb{P}\left[B \in X\right] \leq e^{\varepsilon} \cdot \mathbb{P}\left[A \in X\right] + \delta.$$

*A privacy mechanism $\mathcal{M}$ is (ε, δ)-*DP *(or (ε, δ)-*approximate DP*) if for any datasets D_1 and D_2 that differ only on one record, and for all $S \subseteq \mathcal{O}$, $\mathcal{M}(D_1) \approx_{\varepsilon,\delta} \mathcal{M}(D_2)$.*

This definition is equivalent with Max-KL stability [37], a special case of algorithmic stability, which requires that one change in an algorithm's inputs does not change its output "too much".

The δ in (ε, δ)-DP is sometimes explained as the probability that the privacy loss of the output is larger than e^{ε} (or, equivalently, that the ε-indistinguishability formula is satisfied). In fact, this intuition corresponds to a different definition, first introduced in [267] as *probabilistic DP*, also called *(ε, δ)-DP in distribution* in [60]. A detailed explanation of the distinction between the two definitions can be found in [277].

Definition 15 ((ε, δ)-probabilistic differential privacy [277]). *A privacy mechanism $\mathcal{M}$ is (ε, δ)-*probabilistically DP *(ProDP) if for any datasets D_1 and D_2 that differ only on one record there is a set $S_1 \subseteq \mathcal{O}$ where $\mathbb{P}\left[\mathcal{M}(D_1) \in S_1\right] \leq \delta$, such that for all measurable sets $S \subseteq \mathcal{O}$:*

$$\mathbb{P}\left[\mathcal{M}(D_1) \in S \backslash S_1\right] \leq e^{\varepsilon} \cdot \mathbb{P}\left[\mathcal{M}(D_2) \in S \backslash S_1\right].$$

It is straightforward to show that (ε, δ)-DP is stronger than (ε, δ)-ProDP (with no change in parameters); a proof of the reverse result (with parameter change) is given in [408]. Both definitions can be reformulated using the privacy loss random variable.

Proposition 8. *A mechanism $\mathcal{M}$ is:*

- *ε-DP* $\Leftrightarrow \mathbb{P}_{O\sim\mathcal{M}(D_1)}\left[\mathcal{L}_{D_1/D_2}(O) > \varepsilon\right] = 0$ *for all neighboring D_1 and D_2.*
- *(ε, δ)-DP* $\Leftrightarrow \mathbb{E}_{O\sim\mathcal{M}(D_1)}\left[\max\left(0, 1 - e^{\varepsilon - \mathcal{L}_{D_1/D_2}(O)}\right)\right] \leq \delta$ *for all neighboring D_1 and D_2.*
- *(ε, δ)-ProDP* $\Leftrightarrow \mathbb{P}_{O\sim\mathcal{M}(D_1)}\left[\mathcal{L}_{D_1/D_2}(O) > \varepsilon\right] \leq \delta$ *for all neighboring D_1 and D_2.*

Approximate and probabilistic differential privacy can be combined to form $(\varepsilon, \delta_a, \delta_p)$-*relaxed DP* (RelDP) [407], which requires (ε, δ_a)-DP with probability at least $1 - \delta_p$.

2.2.2.2 *Averaging the privacy loss*

As ε-DP corresponds to a *worst-case* risk model, it is natural to consider relaxations to allow for larger privacy loss for some outputs. It is also natural to consider *average-case* risk models: allowing larger privacy loss values only if lower values compensate it in other cases. One such relaxation is called *Kullback-Leibler privacy* [31, 88]: it considers the *arithmetic* mean of the privacy loss random variable, which measures how much information is revealed when the output of a private algorithm is observed.

Definition 16 (ε-Kullback-Leibler privacy [31, 88])**.** *A privacy mechanism $\mathcal{M}$ is ε-*Kullback-Leibler private *(KLPr) if for all D_1, D_2 differing in one record:*

$$\mathbb{E}_{O\sim\mathcal{M}(D_1)}\left[\mathcal{L}_{D_1/D_2}(O)\right] \leq \varepsilon. \tag{2.2}$$

Note that this formula can be expressed as $D_{KL}\left(\mathcal{M}(D_1)\,|\mathcal{M}(D_2)\right) \leq \varepsilon$ where D_{KL} is the Kullback-Leibler-divergence.

ε-KL privacy considers the *arithmetic* mean of the privacy loss random variable or, equivalently, the *geometric* mean of $e^{\mathcal{L}_{D_1/D_2}}$. This choice of averaging function does not attribute a lot of weight to worst-case events, where $\mathcal{L}_{D_1/D_2}$ takes high values. *Rényi DP* extends this idea by adding a parameter $\alpha \geq 1$, which allows controlling the choice of averaging function by bounding the αth momentum of the privacy loss random variable.

Definition 17 ((α, ε)-Rényi differential privacy [284])**.** *Given $\alpha > 1$, a privacy mechanism $\mathcal{M}$ is (α, ε)-*Rényi DP *(RenyiDP) if for all pairs of neighboring datasets D_1 and D_2:*

$$\mathbb{E}_{O\sim\mathcal{M}(D_1)}\left[e^{(\alpha-1)\mathcal{L}_{D_1/D_2}(O)}\right] \leq e^{(\alpha-1)\varepsilon}.$$

Note that this formula can be expressed as $D_\alpha\left(\mathcal{M}(D_1)\,|\mathcal{M}(D_2)\right) \leq \varepsilon$ where D_α is the Rényi-divergence of order α.

This definition can be naturally extended by continuity to $\alpha = 1$ (where it is equivalent to ε-KL privacy) and $\alpha = \infty$ (where it is equivalent to ε-DP). Larger values of α lead to more weight being assigned to worst-case events: (α, ε)-Rényi DP $\succ$ (α', ε)-Rényi DP iff $\alpha > \alpha'$. Besides $\alpha = 1$ and $\alpha = \infty$, Rényi DP has a simple interpretation for some values of α: $\alpha = 2$

imposes a bound on the arithmetic mean of $e^{\mathcal{L}_{D_1/D_2}}$, $\alpha = 3$ imposes it on the quadratic mean, $\alpha = 4$ on the cubic mean, etc. A related technique is the moments accountant [1] which keeps track of a bound on the moments of the privacy loss random variable during composition.

It is possible to use other divergence functions to obtain other relaxations. For example, in [380], the authors introduce two technical definitions, *binary-*$|\chi|^\alpha$ *DP* (b-$|\chi|^\alpha$ DP) and *ternary-*$|\chi|^\alpha$ *DP* (t-$|\chi|^\alpha$ DP), as part of a proof on amplification by sampling. Other examples of divergences can lead to other variants, like *ε-total variation privacy* [31] (ε-TVPr, using the total variance) and *quantum DP* [81] (QDP, using the quantum divergence).

Another possibility to average the privacy loss is to use *mutual information* to formalize the intuition that any individual record should not "give out too much information" on the output of the mechanism (or vice-versa). This is captured by *ε-mutual-information DP* (MIDP) [88], which guarantees that the mutual information between $\mathcal{M}(D)$ and $D(i)$ conditioned on D_{-i} is under a certain threshold. The bound is taken over all possible priors on D, which avoids having to reason about the attacker's background knowledge. This definition, along with KL-privacy, are technically stronger than approximate DP, but the change in parameters was criticized for not providing a strong enough guarantee [272].

Proposition 9. *For all $\varepsilon > 0$, $\delta \le 1$, and $\alpha \le 1$:*

- *ε-DP $\Longrightarrow$ $\min\{\varepsilon, \varepsilon^2\}$-KLPr (Lemma 1 in [88])*
- *ε-KLPr $\Longrightarrow$ ε-MIDP $\Rightarrow (0, \sqrt{2\varepsilon})$-DP (Lemma 1 and 2 in [88])*
- *ε-DP $\Longrightarrow (\alpha, \varepsilon)$-RényiDP $\Rightarrow \left(\varepsilon + \ln\left(\frac{\alpha-1}{\alpha}\right) - \frac{\ln(\delta)+\ln(\alpha)}{\alpha-1}, \delta\right)$-DP (Theorem 21 in [26])*

2.2.2.3 *Controlling the tail distribution of the privacy loss*

Some definitions go further than simply considering a worst-case bound on the privacy loss, or averaging it across the distribution. They try to obtain the benefits of (ε, δ)-DP with a smaller ε which holds in most cases, but control the behavior of the bad cases better than (ε, δ)-DP, which allows for catastrophic privacy loss in rare cases.

The first attempt to formalize this idea was proposed in [132], where the authors introduce *concentrated DP* (later renamed to *mean-concentrated DP* (mCoDP) in [57]). In this definition, a parameter controls the privacy loss variable globally, and another parameter allows for some outputs to have a greater privacy loss; while still requiring that the difference is smaller than a Gaussian distribution. In [57], the authors show that this definition does not satisfy the post-processing axiom, and propose another formalization of the same idea called *zero-concentrated DP* (zCoDP) [57], which requires that the privacy loss random variable is concentrated around zero.

Definition 18 ((ξ, ρ)-zero-concentrated differential privacy [57])**.** *A mechanism $\mathcal{M}$ is (ξ, ρ)-*zero-concentrated DP *if for all pairs of neighboring datasets D_1 and D_2 and all $\alpha > 1$:*

$$\mathbb{E}_{O \sim \mathcal{M}(D_1)}\left[e^{(\alpha-1)\mathcal{L}_{D_1/D_2}(O)}\right] \le e^{(\alpha-1)(\xi+\rho\alpha)}.$$

Four more variants of concentrated DP exist.

- (ξ,ρ,δ)*-approximate zero-concentrated DP* [57] (AzCoDP), which relaxes (ξ,ρ)-zCoDP by only taking the Rényi divergence on events with probability higher than $1-\delta$ instead of on the full distribution.

- (ξ,ρ,ω)*-bounded CoDP* [57] (bCoDP) relaxes (ξ,ρ)-zCoDP by requiring the inequality to hold only for $\alpha \leq \omega$.

- (ρ,ω)*-truncated CoDP* [56] (tCoDP [56]) relaxes $(0,\rho)$-zCoDP in the same way.

- (ξ,τ)*-truncated CoDP* [81] (tCoDP [81]) requires the Rényi divergence to be smaller than $\min(\xi,\alpha\tau)$ for all $\alpha \geq 1$.

The relations between these definitions and other notions in this section is well-understood. Besides the special cases (e.g., (ρ,∞)-tCoDP [56] is the same as $(0,\rho)$-zCoDP) and the relations that are a direct consequence of the definitions (e.g., (ξ,ρ)-zCoDP is the same as the condition "$(\xi+\rho\alpha)$-RénDP for all $\alpha > 0$"), we list known relations below.

Proposition 10. *For all $\varepsilon > 0$, $\delta > 0$, $\mu > 0$, $\tau > 0$, $\xi \geq 0$ and $\omega > 1$:*

- *ε-DP $\Longrightarrow \left(\frac{\varepsilon(e^{\varepsilon}-1)}{2},\varepsilon\right)$-mCoDP (Theorem 3.5 in [132])*

- *ε-DP $\Longrightarrow \left(0,\frac{\varepsilon^2}{2}\right)$-zCoDP (Lemma 8.3 in [57])*

- *ε-DP $\Longleftrightarrow (\varepsilon,0)$-zCoDP (Lemma 3.2 in [57])*

- *(μ,τ)-mCoDP $\Longrightarrow \left(\mu-\frac{\tau^2}{2},\frac{\tau^2}{2}\right)$-zCoDP (Lemma 4.2 in [57])*

- *(ξ,ρ)-zCoDP $\Longrightarrow \left(\xi+\rho, O(\sqrt{\xi+2\rho})\right)$-mCoDP (Lemma 4.3 in [57])*

- *(ξ,ρ)-zCoDP $\Longrightarrow \left(\xi+\rho+\sqrt{4\rho\ln\left(\frac{\min(1,\sqrt{\pi\rho})}{\delta}\right)},\delta\right)$-DP (Lemma 3.5 and 3.6 in [57])*

- *$\left(\xi+\sqrt{\rho\ln\frac{1}{\delta}}\right)$-DP $\Longrightarrow \left(\xi-\frac{\rho}{4}+5\sqrt[4]{\rho},\frac{\rho}{4}\right)$-zCoDP (Lemma 3.7 in [57])*

- *(ρ,ω)-tCoDP [56] $\Rightarrow (\hat{\varepsilon},\delta)$-DP, where $\hat{\varepsilon}=\rho+2\sqrt{\rho\ln\frac{1}{\rho}}$ if $\ln\frac{1}{\delta} \leq (\omega-1)^2\rho$, and $\hat{\varepsilon}=\rho\omega+\frac{\ln\frac{1}{\delta}}{\omega-1}$ otherwise (Lemma 6 in [56])*

2.2.2.4 *Extension*

Most definitions of this section can be seen as bounding the divergence between $\mathcal{M}(D_1)$ and $\mathcal{M}(D_2)$, for different possible divergence functions. In [31], the authors use this fact to generalize them and define (f,ε)*-divergence DP* (DivDP), which takes the particular divergence used as a parameter f.

Definition 19 ((f,ε)-divergence differential privacy [31])**.** *Let f be a convex function such as $f(1)=0$. A privacy mechanism $\mathcal{M}$ is* (f,ε)-divergence DP *if for all pairs of neighboring datasets D_1, D_2:*

$$\mathbb{E}_{O\sim\mathcal{M}(D_1)}\left[f\left(e^{\mathcal{L}}_{D_1/D_2}\right)\right]\leq\varepsilon.$$

An instance of this definition was presented in [118] as (f_k,ε)*-divergence DP*; which requires that $\mathbb{E}_{O\sim\mathcal{M}(D_1)}\left[\left|e^{\mathcal{L}}_{D_1/D_2}-1\right|^k\right]\leq\varepsilon^k$. This definition is mainly used to prove technical results on privacy/utility trade-offs in the local model. For any $k\leq 1$, ε-DP implies $(f_k,e^{\varepsilon}-1)$-DivDP, and when $k=2$, it is equivalent to $\left(2,\ln\left(1+\varepsilon^2\right)\right)$-RényiDP (Section 2 in [118]).

Moreover, *capacity bounded differential privacy* (CBDP) was introduced in [69], which uses H-restricted f-divergence: $D^H_f(P|Q)=\sup_{h\in H}[\mathbb{E}_{x\sim P}\left[h(x)\right]-\mathbb{E}_{x\sim Q}\left[f^*(h(x))\right]]$ where f is a divergence, H is a family of functions, and f^* is the Fenchel conjugate[5]. In other words, it requires the supremum condition to hold only for a selected set of functions (queries) instead of all possible ones. The interpretation for this definition is slightly different than other definitions in this section: H represents the possible attacks that the attacker is allowed to perform.

Finally, most definitions in this section taking two real-valued parameters can be extended to use a *family* of parameters rather than a single pair of parameters. As shown in [348] (Theorem 2) for approximate DP, probabilistic DP, and Rényi DP, finding the tightest possible family of parameters (for either definition) for a given mechanism is equivalent to specifying the behavior of its privacy loss random variable entirely.

2.2.2.5 *Multidimensional definitions*

Allowing a small probability of error δ by using the same concept as in (ε,δ)-DP is very common; many new DP definitions were proposed in the literature with such a parameter. Unless it creates a particularly notable effect, we do not mention it explicitly and present the definitions without this parameter.

Definitions in this section can be used as standalone concepts: (ε,δ)-DP is omnipresent in the literature, and the principle of averaging risk is natural enough for Rényi privacy to be used in practical settings, like posterior sampling [163] or resistance to adversarial inputs in machine learning [317]. Most variants in this section, however, are only used as technical tools to get better results on composition or privacy amplification [131, 149, 245, 380].

2.2.3 *Neighborhood definition (N)*

The original definition of differential privacy considers datasets differing in one record. Thus, the datasets can differ in two possible ways: either they have the same size and differ only on one record, or one is a copy of the other with one extra record. These two options do not protect the same thing: the former protects the *value* of the records while the latter also protects their *presence* in the data: together, they protect any property about a single individual.

[5] The Fenchel conjugate for a function f with a domain R is $f^*(x)=\sup_{y\in R}[xy-f(y)]$.

In many scenarios, it makes sense to protect a different property about their dataset, e.g., the value of a specific sensitive field, or entire groups of individuals. It is straightforward to adapt DP to protect different sensitive properties: all one has to do is change the definition of neighborhood in the original definition.

2.2.3.1 *Changing the sensitive property*

The original definition states that the ε-indistinguishability property should hold for "any datasets D_1 and D_2 that differ only on the data of one individual". Modifying the set of pairs (D_1, D_2) such that $\mathcal{M}(D_1) \approx_\varepsilon \mathcal{M}(D_2)$ is equivalent to changing the protected sensitive property.

Weaker notions

In DP, the difference between D_1 and D_2 is sometimes interpreted as "one record value is different", or "one record has been added or removed". In [227], the authors formalize these two options as *bounded DP* and *unbounded DP*. They also introduced *attribute DP* and *bit DP*, for smaller changes within the differing record.

Definition 20 ([227]). *If a privacy mechanism $\mathcal{M}$ satisfies $\mathcal{M}(D_1) \approx_\varepsilon \mathcal{M}(D_2)$ for any pair D_1, D_2, where D_1 can be obtained from D_2 by...*

- *... adding or removing one record, then $\mathcal{M}$ is ε-*unbounded DP *(uBoDP).*
- *... changing exactly one record, then $\mathcal{M}$ is ε-*bounded DP *(BoDP).*
- *... changing one attribute in a record, then $\mathcal{M}$ is ε-*attribute DP *(AttDP).*
- *... changing one bit of an attribute in a record, then $\mathcal{M}$ is ε-*bit DP *(BitDP).*

In [227], authors show that ε-unbounded DP implies 2ε-bounded DP, as changing a record can be seen as deleting it and adding a new one in its place. The original definition of ε-DP is the conjunction of ε-unbounded DP and ε-bounded DP. However, bounded DP is frequently used in the literature: *locally differentially private mechanisms*, in particular, in which each user randomizes their own data before sending it to the aggregator, typically uses bounded DP. It is often simply called differential privacy, and sometimes renamed, like in [149], where the authors call it *per-person DP*.

Variants of differential privacy that do not protect individuals, but single contributions (in the case where the same person can contribute multiple times to a dataset), are also often used in practice, especially for machine learning applications [271]. Some recent works also argue that in-between definitions are appropriate: rather than protecting a single contribution or entire users contributions, authors in [22] suggest that protecting *elements* that reveal information about users, after deduplicating or clustering contributions. For example, rather than protecting all website visits by a single user, or each visit individually, one might choose to protect the fact that a user ever visited a website (but not whether the user visited the same website once or many times). They call the corresponding definition *element-level DP* (ELDP).

Another way to relax the neighborhood definition in DP is to consider that only certain types of information are sensitive. For example, if the attacker learns that their target has cancer, this is more problematic than if they learn that their target does *not* have cancer. This idea is captured in *one-sided DP* [231]: the neighbors of a dataset D are obtained by replacing a single sensitive record with any other record (sensitive or not). The idea of sensitivity is formalized by a *policy P*, which specifies which records are sensitive. This idea cannot be captured simply by ε-indistinguishability, since one-sided DP is asymmetric.

Definition 21 ((P,ε)-one-sided differential privacy [231])**.** *Given a policy $P \subseteq \mathcal{T}$, a privacy mechanism $\mathcal{M}$ is* (P,ε)-one-sided DP *(OSDP) iff for all datasets D_1 and D_2, where D_2 has been obtained by replacing a record $t \in D_1 \cap P$ by any other record and for all $S \subseteq \mathcal{O}$:*

$$\mathbb{P}\left[\mathcal{M}(D_1) \in S\right] \leq e^{\varepsilon} \cdot \mathbb{P}\left[\mathcal{M}(D_2) \in S\right].$$

When $P = \mathcal{T}$, this is equivalent to bounded DP. Similar ideas were proposed in multiple papers:

- In [23], the authors propose *sensitive privacy*, which determines which records are sensitive based on the data itself and a *normality property N* and a graph-based definition of k-neighborhood, instead of using a data-independent determination.
- In [51], the authors introduce *anomaly-restricted DP*, which assumes that there is only one outlier in the dataset, and that this outlier should not be protected.

Stronger notions

More restrictive definitions are also possible. First, some definitions make the definition of neighborhood more explicit when a single person can contribute multiple times to a dataset; this is the case for *client/participant DP*, defined in [271]. In [123], the authors implicitly define (c,ε)*-group privacy* considers datasets that do not differ in one record, but possibly several, to protect multiple individuals. This can also be interpreted as taking correlations into account when using DP: *DP under correlation* [73] uses an extra parameter to describe the maximum number of records that the change of one individual can influence.

These two definitions are formally equivalent; but the implicit interpretation of DP behind them is different. (c,ε)-group privacy is compatible with the associative view under the strong adversary assumption (the adversary knows all records except c) or the causal view (c records are changed after the data is generated). Meanwhile, DP under correlation implicitly considers the associative view with the independence assumption; and tries to relax that assumption. This last approach was further developed via *dependent DP* [258], which uses "dependence relationships" to describe how much the variation in one record can influence the other records.

Definition 22 ((R,c,ε)-dependent differential privacy [258])**.** *A privacy mechanism $\mathcal{M}$ is* (R,c,ε)-dependent DP *(DepDP) where R is the probabilistic dependence relationship and c is the dependence size, if for any pair of datasets D_1 and D_2, where D_2 has been obtained from D_1 by changing one record and the corresponding at most $c-1$ other records according to R, $\mathcal{M}(D_1) \approx_{\varepsilon} \mathcal{M}(D_2)$.*

Note that when R is the empty relation, or when $c = 1$, this definition is equivalent to bounded DP: under the associative view of DP, this represents independence between records. Similar definitions appear in [392, 393] as *correlated DP* (CorDP), in which correlations are defined by an observation on other datasets, and in in [399] as *bayesian DP*[6] (BayDP$^{[399]}$), where the neighborhood relation is defined by an adversary having some knowledge about correlations in the data. An extension is proposed in [256] as *prior DP* (PriDP) which considers a family of adversaries instead of a single adversary.

The strongest possible variant is considered in [227], where the authors define *free lunch privacy*, in which the attacker must be unable to distinguish between any two datasets, even if they are completely different. This guarantee can be interpreted as a reformulation of Dalenius' privacy goal [91], which we mentioned in Section 2.1.6. As such, all mechanisms that satisfy free lunch privacy have a near-total lack of utility.

Definition 23 (ε-free lunch privacy [227])**.** *A privacy mechanism $\mathcal{M}$ satisfies* ε-free lunch privacy *(FLPr) if $\mathcal{M}(D_1) \approx_{\varepsilon} \mathcal{M}(D_2)$ for any pair of datasets D_1, D_2.*

2.2.3.2 *Limiting the scope of the definition*

Redefining the neighborhood property can also be used to reduce the scope of the definitions. In [349], the authors note that DP requires ε-indistinguishability of results between any pair of neighboring datasets, but in practice, the data custodian has only *one* dataset D they want to protect. Thus, they only require ε-indistinguishability between this dataset D and all its neighbors, calling the resulting definition *individual DP*. An equivalent definition was proposed in [64] as *conditioned DP*.

Definition 24 ((D, ε)-individual differential privacy [349])**.** *Given a dataset $D \in \mathcal{D}$, a privacy mechanism $\mathcal{M}$ satisfies* (D, ε)-individual DP *(IndDP) if for any dataset D' that differs in at most one record from D, $\mathcal{M}(D) \approx_{\varepsilon} \mathcal{M}(D')$.*

This definition was further restricted in [379] where besides fixing a dataset D, a record t is also fixed.

2.2.3.3 *Applying the definition to other types of input*

Many adaptations of differential privacy are simply changing the neighborhood definition to protect different types of input data than datasets. A few examples follow.

- In [75, 138, 396], the authors adopted differential privacy for locations. In [138] the authors defined *location privacy*, in which neighbors are datasets which differ in at most one record, and the two differing records are at a physical distance smaller than a given threshold. This definition also appears in [75] as *DP on r-location set*[7]. Several more location-related differential privacy variants were defined in [291]: *untrackability* (which adopts differential privacy for set of locations by protecting whether they originated from a single user or by two users), *undetectability* and *multi user untrackability*, (which extend this idea further by not assuming both sets originated from the same private data and to multiple users respectively).

[6] There are two other notions with the same name: introduced in [248, 367], we mention them in Section 2.2.4 and 2.2.5 respectively.

[7] Distinct from *DP on δ-location set* [396], which we mention in Section 2.2.4.

- In [108, 188, 316, 329, 351, 359], the authors adopt differential privacy to graph-structured data. In [188], authors present multiple alternative definitions, which protect different parts of the graph: the strongest is *node-DP*, which protects a node and all its edges; the weakest is *edge-DP*, only protects one edge; and an intermediate definition is *k-edge-DP*, which protects a total of k edges and nodes. In [359], the authors introduce *out-link privacy*, which protects all outgoing edges from a given node. In [329], the author introduces *QL-edged-labeled DP* similar to *out-link privacy*, but only protecting a predetermined subset of outgoing edges. In [316], the author introduces l_1*-weighted DP*, in which graph edges are weighted, and graphs are neighbors when the total weight of their differing edges is smaller than 1; this notion was also defined implicitly in [340]. In [351], the authors define *decentralized DP* which extends the graph neighborhood to two jumps. In [221], the authors introduce *protected DP*, which adapts DP for graphs and guarantees that no observer can learn much about the set of edges corresponding to any protected node while offering no guarantees for the other nodes. Finally, in [108] the authors introduce *seamless privacy*, which rather than protecting characteristics of a specific input graph, it ensures that certain pairs of queries on this graph return similar answers.

- In [125, 129, 130, 148, 223, 382], authors adapt differential privacy to a streaming context, where the attacker can access the mechanism's internal states. In [125, 129, 130], authors define *pan-privacy*, which comes in two variants: *event-level* pan-privacy (called *strong DP* in [382]) protects individual events, and *user-level* pan-privacy protects all events associated to a single user. In [223], the authors extend the previous idea and propose *w-event privacy*, which protects any event sequence occurring within a window of at most w timestamps. In [148, 291] this was further extended to an infinite horizon via *discounted differential privacy* (which keep assigning smaller-and-smaller weights to further-and-further events) and *everlasting privacy* (which limit the leakage of information users suffer, no matter how many executions a mechanism had), respectively.

- In [375] the authors adopt differential privacy for Random Access Memory and Private Information Retrieval. For RAM the neighborhood is defined over the sequence of logical memory requests over time; the same notion appears in [63] as *differential obliviousness* and in [11] as *oblivious DP*. The adaptation of neighborhood is similar in case of PIR; a similar notion appears in [364] as ε*-private PIR* and in [311] as ε*-DPIR*. Additionally, in [222], the authors use a similar idea to define differential privacy for outsourced database systems.

- In [211], the authors adapt differential privacy for symbolic control systems, and introduce *word-DP* and *substitution-word-DP*, protecting respectively pairs of words whose Levenshtein distance is lower than a given parameter, or whose Hamming distance is lower than a given parameter.

- In [405], the authors adapt differential privacy for text vectors, and propose *text indistinguishability*, in which the neighborhood relationship between two word vectors depends on their Euclidean distance.
- In [398] the authors defined *set operation DP*, which adopts differential privacy for set operations where $\{D_1, D_2\}$ and $\{D_1, D_2\}'$ are neighbor if either D_1 and D'_1 or D_2 and D'_2 are neighbors.
- In [242, 401], the authors define *refinement DP* and *pixel DP* respectively, which adopts the definition for images with neighbors given by some transformation or metric.
- In [201], the authors adopt DP to gossip protocols to protect the privacy of information source.
- In [307], the authors define *functional DP*, which adopts differential privacy for functions.
- In [347], the authors define *phenotypic DP*, which adopts differential privacy for genomic data, where the difference in the phenotype vector defines the neighborhood.
- In [178], the authors adapt differential privacy to recommendation systems, and define *distance-based DP*, which protects not only a given user recommendation but also all the recommendations of a given user for similar items.
- In [262], the authors adapt differential privacy to machine learning, and define *differential training privacy*, which quantifies the risk of membership inference of a record with respect to a classifier and its training data.
- In [366], the authors define *DP for bandit algorithms*, where the neighborhood notion is defined by changing any single reward in a multi-armed bandit game sequence. In [41], this definition is renamed to instantaneous DP (Def 5), and few more variants are proposed for this problem space: local DP for bandits, pan-privacy for bandits, sequential privacy for bandits, and environment privacy for bandits.

2.2.3.4 *Extensions*

It is natural to generalize the variants of this section to arbitrary neighboring relationships. One example is mentioned in [227], under the name *generic DP*[8], where the neighboring relation is entirely captured by a *relation* $\mathcal{R}$ between datasets.

Definition 25 ($(\mathcal{R},\varepsilon)$-generic differential privacy [227])**.** *Given a relation $\mathcal{R} \subseteq \mathcal{D}^2$, a privacy mechanism $\mathcal{M}$ satisfies* $(\mathcal{R},\varepsilon)$-generic DP *(GcDP [227]) if for all $(D_1, D_2) \in \mathcal{R}$, $\mathcal{M}(D_1) \approx_\varepsilon \mathcal{M}(D_2)$.*

[8] Another definition with the same name is introduced in [225, 226], we mention it in Section 2.2.4.

This definition is symmetric, but it can easily be modified to accommodate asymmetric definitions like one-sided DP.

Other definitions use different formalizations to also generalize the concept of changing the neighborhood relationship. Some (like pufferfish privacy, see Definition 35 in Section 2.2.5) use *pairs of predicates* that D_1 and D_2 must respectively satisfy to be neighbors. Others (like coupled-worlds privacy, see Definition 44 in Section 2.2.7) use *private functions* taking datasets as input, and define neighbors to be datasets that have a different output according to this private function. Others use a *distance function* between datasets, and neighbors are defined as datasets a distance lower than a given distance threshold; this is the case for *DP under a neighborhood* (DPUN), introduced in [145], *adjacent DP* (AdjDP), introduced in [235][9], *constrained DP* (ConsDP), introduced in [409] (where the distance captures a utility-related constraint), and *distributional privacy*[10] (DlPr $^{[409]}$), also introduced in [409] (with additional constraints on the neighborhood definition: neighboring datasets must be part of a fixed set and have elements in common). This distance can also be defined as the sensitivity of the mechanism, like in *sensitivity-induced DP* [335] (SIDP), or implicitly defined by a set of constraints, like what is done implicitly in [227] via *induced neighbors DP* (INDP).

One notable instantiation of generic DP is *blowfish privacy* [193]. Its major building blocks are a *policy graph* G, that specifies which pairs of domain values in $\mathcal{T}$ should not be distinguished between by an adversary; and a set of *constraints* Q that specifies the set of possible datasets that the definition protects. It was inspired by the Pufferfish framework [228] (see Definition 35 in Section 2.2.5), but the attacker is not assumed to have uncertainty over the data: instead, it models an attacker whose knowledge is a set of deterministic constraints on the data.

Definition 26 ((G, Q, ε)-blowfish privacy [187, 193])**.** *Given a policy graph $G \in \mathcal{T}^2$ and a set of constraints Q, a privacy mechanism $\mathcal{M}$ satisfies (G, ε)-*blowfish privacy *(BFPr) if for all datasets D_1 and D_2 that satisfy constraints Q and differ in only one element i such that $(D_1(i), D_2(i)) \in G$, $\mathcal{M}(D_1) \approx_\varepsilon \mathcal{M}(D_2)$.*

As noted in [193], a mechanism satisfies ε-bounded DP if and only if satisfies $(K, \mathcal{I}_n, \varepsilon)$-blowfish privacy, where K is the complete graph, and $\mathcal{I}_n$ is any datasets of size n. A particular instantiation of this idea is explored in [227] as *induced DP*, where the definition of neighbors is induced by a set of constraints.

2.2.3.5 *Multidimensional definitions*

Modifying the protected property is orthogonal to modifying the risk model implied by the quantification of privacy loss: it is straightforward to combine these two dimensions. Indeed, many definitions mentioned in this section were actually introduced with a δ parameter allowing for a small probability of error. One particularly general example is *adjacency relation divergence DP* [218], which combines an arbitrary neighborhood definition (like in generic DP) with an arbitrary divergence function (like in divergence DP).

[9] Originally simply called “differential privacy” by its authors.

[10] Another definition with the same name is introduced in [53, 333], we mention it in Section 2.2.4.

As the examples in Section 2.2.3.3 show, it is very common to change the definition of neighborhood in practical contexts to adapt what aspect of the data is protected. Further, local DP mechanisms like RAPPOR [141] implicitly use bounded DP: the participation of one individual is not secret, only the value of their record is protected. Variants that limit the scope of the definition to one particular dataset or user, however, provide few formal guarantees and do not seem to be used in practice.

2.2.4 *Variation of privacy loss (V)*

In DP, the privacy parameter ε is *uniform*: the level of protection is the same for all protected users or attributes, or equivalently, only the level of risk for the most at-risk user is considered. In practice, some users might require a higher level of protection than others or a data custodian might want to consider the level of risk across all users, rather than only considering the worst case. Some definitions take this into account by allowing the privacy loss to vary across inputs, either explicitly (by associating each user to an acceptable level of risk), or implicitly (by allowing some users to be at risk, or averaging the risk across users).

2.2.4.1 *Varying the privacy level across inputs*

In Section 2.2.3, we saw how changing the definition of the neighborhood can be used to adapt the definition of privacy and protect different aspects of the input data. However, the privacy protection in those variants is binary: either a given property is protected, or it was not. A possible option to generalize this idea further is to allow the privacy level to vary across possible inputs.

One natural example is to consider that some users might have higher privacy requirements than others, and make the ε vary according to which user differs between the two datasets. This is done in *personalized DP*, a notion first defined informally in [302], then independently in [134, 165, 212, 261]. An equivalent notion is also defined in [10] as *heterogeneous DP*, while a location-based definition is presented in [97] as *personalized location DP*.

Definition 27 (Ψ-personalized differential privacy [212])**.** *A privacy mechanism $\mathcal{M}$ provides* Ψ-personalized DP *(PerDP) if for every pair of neighboring datasets (D, D_{-i}) and for all sets of outputs $S \subseteq \mathcal{O}$:*

$$\mathbb{P}\left[\mathcal{M}\left(D_{-i}\right) \in S\right] \leq e^{\Psi(D(i))}\mathbb{P}\left[\mathcal{M}\left(D\right) \in S\right]$$

where Ψ is a privacy specification: $\Psi : \mathcal{T} \rightarrow \mathbb{R}^+$ maps the records to personal privacy preferences and $\Psi\left(D(i)\right)$ denotes the privacy preference of the i-th record.

As shown in [89, 212], ε-DP implies Ψ-PerDP, where $\Psi(t) = \varepsilon$ for all $t \in \mathcal{T}$; and if the definition is modified to become symmetric, Ψ-PerDP implies ε-DP where $\varepsilon = \max_t \Psi(t)$.

This definition can be seen as a refinement of the intuition behind one-sided DP, which separated records into sensitive and non-sensitive ones. The idea of making the privacy level vary across inputs can be generalized further, by also making the privacy level depend on the entire dataset, and not only in the differing record. This is done in [263], where the authors define *tailored DP*.

Definition 28 (Ξ-tailored differential privacy [263]). *A mechanism $\mathcal{M}$ satisfies* Ξ-tailored differential privacy *(TaiDP) for $\Xi : \mathcal{T} \times \mathcal{D} \rightarrow \mathbb{R}_0^\infty$ if for any dataset D, $\mathcal{M}(D) \approx_{\varepsilon(D(i),D)} \mathcal{M}(D_{-i})$.*

This concept can be applied to strengthen or weaken the privacy requirement for a record depending on whether they are an outlier in the dataset. In [263], the authors formalize this idea and introduce *outlier privacy*, which tailors an individual's protection level to their "outlierness". Other refinements are also introduced in [263]: *simple outlier privacy* (SOPr), *simple outlier DP* (SODP), and *staircase outlier privacy* (SCODP). A similar idea was explored in [216], which introduced *pareto DP* (ParDP): it utilizes a pareto distribution of parameters (p, r) to separate a large number of low-frequency individuals from a small number of high-frequency, and the sensitivity is calculated based on only the low-frequency individuals.

Finally, varying the privacy level across inputs also makes sense in *continuous* scenarios, where the neighborhood relationship between two datasets is not binary, but quantified. This is, for example, the case for *ε-geo-indistinguishability* [15], where two datasets D_1 and D_2 are considered *r-neighbors* if the only different record between D_1 and D_2 are at a distance r of each other, and the ε grows linearly with r.

2.2.4.2 *Randomizing the variation of privacy levels*

Varying the privacy level across inputs can also be done in a randomized way, by guaranteeing that some random fraction of users have a certain privacy level. One example is proposed in [185] as *random DP*: the authors note that rather than requiring DP to hold for any possible datasets, it is natural to only consider *realistic datasets*, and allow "edge-case" or very unrealistic datasets to not be protected. This is captured by generating the data randomly, and allowing a small proportion γ of cases to not satisfy the ε-indistinguishability property.

Definition 29 ($(\pi, \gamma, \varepsilon)$-random differential privacy [185]). *Let π be a probability distribution on $\mathcal{T}$, D_1 a dataset generated by drawing n i.i.d. elements in π, and D_2 the same dataset as D_1, except one element was changed to a new element drawn from π. A mechanism $\mathcal{M}$ is* $(\pi, \gamma, \varepsilon)$-random DP *(RanDP) if $\mathcal{M}(D_1) \approx_\varepsilon \mathcal{M}(D_2)$, with probability at least* $1 - \gamma$ on the choice of D_1 and D_2.

The exact meaning of "with probability at least $1 - \gamma$ on the choice of D_1 and D_2" can vary slightly. In [184] and in [270], the authors introduce *predictive DP* (PredDP) and *model-specific DP* respectively, which quantify over all possible choices of D_1, and picks D_2 randomly in the neighborhood of D_1. In [111], D_1 and D_2 are both taken out of a set of density larger than $1 - \gamma$, and the authors call this definition *generalized DP* (GdDP). The distribution generating the dataset is also not always assumed to be generating i.i.d. records; we denote the corresponding parameter by θ.

Random DP might look similar to probabilistic DP: in both cases, there is a small probability that the privacy loss is unbounded. On the other hand, they are very different: in random DP, this probability is computed inputs of the mechanisms (i.e., users or datasets), for probabilistic DP, it is computed across mechanism outputs. Also similarly to probabilistic DP, excluding some cases altogether creates definitional issues: random DP does not satisfy the convexity axiom (see Proposition 15 in Section 2.2.9). We postulate that using a different tool to allow some inputs to not satisfy the mechanism, similar to approximate DP or Rényi DP, could solve this problem.

Usually, data-generating distributions are used for other purposes: they typically model an adversary with partial knowledge. However, definitions in this section still compare the outputs of the mechanisms given fixed neighboring datasets: the only randomness in the indistinguishability property comes from the mechanism. By contrast, definitions of Section 2.2.5 compare the output of the mechanism on a random dataset, so the randomness comes both from the data-generating distribution and the mechanism.

2.2.4.3 *Multidimensional definitions*

As varying the privacy level or limiting the considered datasets are two distinct way of relaxing differential privacy, it is possible to combine them with the previously mentioned dimensions.

Combination with N

The definitions described in Section 2.2.3 (e.g., generic DP or blowfish privacy) have the same privacy constraint for all neighboring datasets. Thus, they cannot capture definitions that vary the privacy level across inputs. However, both ideas can be naturally captured together via distance functions. In [67], the authors introduce *$d_{\mathfrak{D}}$-privacy*, in which the function $d_{\mathfrak{D}}$ takes both datasets as input, and returns the corresponding maximal privacy loss (the ε) depending on the difference between the two datasets.

Definition 30 ($d_{\mathfrak{D}}$-privacy [67])**.** *Let $d_{\mathfrak{D}} : \mathcal{D}^2 \to \mathbb{R}_\infty$. A privacy mechanism $\mathcal{M}$ satisfies* $d_{\mathfrak{D}}$-privacy *($d_{\mathfrak{D}}$-Pr) if for all pairs of datasets D_1, D_2 and all sets of outputs $S \subseteq \mathcal{O}$:*

$$\mathbb{P}\left[\mathcal{M}(D_1) \in S\right] \le e^{d_{\mathfrak{D}}(D_1,D_2)} \cdot \mathbb{P}\left[\mathcal{M}(D_2) \in S\right].$$

When $d_{\mathfrak{D}}$ is proportional to the Hamiltonian difference between datasets, this is equivalent to ε-DP. In $d_{\mathfrak{D}}$-privacy, the $d_{\mathfrak{D}}$ function specifies both the privacy parameter *and* the definition of neighborhood: it can simply return ∞ on non-neighboring datasets, and vary the privacy level across inputs for neighboring datasets. In the original definition, the authors impose that $d_{\mathfrak{D}}$ is *symmetric*, but this condition can also be relaxed to allow $d_{\mathfrak{D}}$-privacy to extend definitions like one-sided DP.

Equivalent definitions of $d_{\mathfrak{D}}$-privacy also appeared in [138] as *l-privacy*, and in [219] as *extended DP*. Several other definitions, such as *weighted DP* [322] (WeiDP), *smooth DP* [31] (SmoDP) and *earth mover's privacy* [151] (EMDP), can be seen as particular instantiations of $d_{\mathfrak{D}}$-privacy for specific functions d measuring the distance between datasets. This is also the case for some definitions tailored for location privacy, like *geo-graph-indistinguishability* [357], which specifically applies to network graphs.

Random DP can also be combined with changing the neighborhood definition: in [396], the authors define *DP on a δ-location set*[11], for which the neighborhood is defined by a set of "plausible" locations around the true location of a user. A notable definition using the same

[11] Distinct from *DP on r-location set* [75], which we mention in Section 2.2.3.

combination of dimensions is *distributional privacy*[12], introduced in [53, 333]: it combines random DP (for a large family of distributions) and free lunch privacy.

Definition 31 ((ε, γ)-distributional privacy [53, 333])**.** *An algorithm $\mathcal{M}$ satisfies* (ε, γ)-distributional privacy *(DlPr*[53, 333]*) if for any distribution π over possible tuples, if $D_1, D_2 \in \mathcal{D}$ are picked by randomly drawing a fixed number n of elements from π without replacement, then with probability at least* $1 - \gamma$ *over the choice of D_1 and D_2, $\mathcal{M}(D_1) \approx_\varepsilon \mathcal{M}(D_2)$.*

Interestingly, this definition captures an intuition similar to the variants in Section 2.2.7: the adversary can only learn properties of the data-generating distribution, but not about particular samples (except with probability lower than γ). The authors also prove that if $\mathcal{M}$ is (ε, γ)-DlPr and $\gamma = o(\frac{1}{n^2})$, where n is the size of the dataset being generated, then $\mathcal{M}$ is also ε-DP.

Combination with Q

Different risk models, like the definitions in Section 2.2.2, are also compatible with varying the privacy parameters across inputs. For example, in [234], the author proposes *endogenous DP* (EndDP), which is a combination of (ε, δ)-DP and personalized DP. Similarly, *pseudo-metric DP* (PsDP), defined in [106], is a combination of $d_{\mathcal{D}}$-privacy and (ε, δ)-DP; while *extended divergence DP* (EDivDP), defined in [218], is a combination of $d_{\mathcal{D}}$-privacy and divergence DP.

Randomly limiting the scope of the definition can also be combined with ideas from the previous sections. For example, in [367], the authors introduce *weak Bayesian DP* (WBDP), which combines random DP and approximate DP. In [381], the authors introduce *on average KL privacy*, which uses KL-divergence as quantification metric, but only requires the property to hold for an "average dataset", like random DP; a similar notion appears in [150] as *average leave-one-out KL stability*. In [92, 367], the authors introduce *Bayesian DP*[13] (BayDP[367]) and *privacy at risk* (PAR) respectively; both definitions combine random DP with probabilistic DP, with slightly different approaches: the former quantifies over all possible datasets and changes one fixed record randomly, while the latter selects both datasets randomly, conditioned on these datasets being neighbors.

In [225], Kifer et al. go further and generalize the intuition from generic DP, introduced in Section 2.2.3, and generalize the indistinguishability condition entirely. The resulting definition is *also* called generic differential privacy.

Definition 32 ($(\mathcal{R}, M)$-generic differential privacy [225, 226])**.** *A privacy mechanism $\mathcal{M}$ satisfies* $(\mathcal{R}, M)$-generic DP *(GcDP*[225]*) if for all measurable sets $S \subseteq \mathcal{O}$ and for all $(D_1, D_2) \in \mathcal{R}$:*

$$M_{D_1,D_2}(\mathbb{P}[\mathcal{M}(D_1) \in S]) \geq \mathbb{P}[\mathcal{M}(D_2) \in S]\, M_{D_1,D_2}(\mathbb{P}[\mathcal{M}(D_1) \notin S]) \quad \geq \mathbb{P}[\mathcal{M}(D_2) \notin S]$$

where $M_{D_1,D_2} : [0,1] \to [0,1]$ *is a concave function continuous on* $(0,1)$ *such as* $M_{D_1,D_2}(1) = 1$.

[12] Another definition with the same name is introduced in [409], we mention it in Section 2.2.3.

[13] There are two other notions with the same name: introduced in [248, 399], we mention them in Section 2.2.3 and 2.2.5 respectively.

The privacy relation $\mathcal{R}$ is still the generalization of neighborhood and the privacy predicate is the generalization of the ε-indistinguishability to arbitrary functions. In particular, it can encompass all variants of described in Section 2.2.2 in addition to the ones in this section: for example, if $M_{D_1,D_2}(x) = \min\left(1, xe^{\varepsilon} + \delta, 1 - (1 - x - \delta)e^{-\varepsilon}\right)$ holds for for all D_1 and D_2, then this is equivalent to (ε,δ)-DP. This definition was an attempt at finding the most generic definition that still satisfies privacy axioms: another extension defined in the same work, *abstract DP* (AbsDP) is even more generic, but no longer satisfies the privacy axioms.

Definitions in this section are particularly used in the context of local DP[14] and in particular for applications to location privacy: various metrics have been discussed to quantify how indistinguishable different places should be to provide users of a local DP mechanism with meaningful privacy protection [68].

2.2.5 *Background knowledge (B)*

In differential privacy, the attacker is implicitly assumed to have full knowledge of the dataset: their only uncertainty is about their target. This implicit assumption is also present for the definitions of the previous dimensions: indeed, the attacker has to distinguish between two *fixed* datasets D_1 and D_2. The only source of randomness in ε-indistinguishability comes from the mechanism itself. In many cases, this assumption is unrealistic, and it is natural to consider weaker adversaries, who do not have full background knowledge. One of the main motivations to do so is to use significantly less noise in the mechanism [116].

The typical way to represent this uncertainty formally is to assume that the input data comes from a certain probability distribution (named *data evolution scenario* in [228]): the randomness of this distribution models the attacker's uncertainty. Informally, the more random this probability distribution is, the less knowledge the attacker has. However, the definition that follows depends whether DP is considered with the associative or the causal view. In the associative view, the sensitive property changes *before* the data is generated: it conditions the data-generating probability distribution. In the causal view, however, the sensitive property is only changed *after* the data is generated. The two options lead to very distinct definitions.

2.2.5.1 *Conditioning the output on the sensitive property*

Using a probability distribution to generate the input data means that the ε-indistinguishability property cannot be expressed between two fixed datasets. Instead, one natural way to express it is to condition this distribution on some sensitive property. The corresponding notion, *noiseless privacy*[15] was first introduced in [116] and formalized in [46].

Definition 33 ((Θ,ε)-noiseless privacy [46, 116])**.** *Given a family Θ of probability distribution on $\mathcal{D}$, a mechanism $\mathcal{M}$ is (Θ,ε)-*noiseless private *(NPr) if for all $\theta \in \Theta$, all i and all $t, t' \in \mathcal{T}$:*

$$\mathcal{M}(D)_{|D\sim\theta, D(i)=t} \approx_{\varepsilon} \mathcal{M}(D)_{|D\sim\theta, D(i)=t'}.$$

[14] For details, see Section 2.2.10.2.

[15] Another definition with the same name is introduced in [147], we mention it in Section 2.2.10.1.

In the original definition, the *auxiliary knowledge* of the attacker is explicitly pointed out in an additional parameter. As we show in Section 3.1.2.1 after Definition 49, in the case where there is no δ of $(\varepsilon, 0)$-DP, this syntactic add-on is not necessary, so we omitted it here.

This definition follows naturally from considering the associative view of DP with the strong adversary assumption, and attempting to relax this assumption. The exact way to model the adversary's uncertainty can be changed; for example *DP under sampling* [255], an instance of noiseless privacy, models it using random sampling.

This definition was not the first attempt at formalizing an adversary with partial background knowledge. In [264], authors define *ε-privacy*, which represents the partial knowledge as a Dirichlet distribution (instead of an arbitrary distribution), whose parameters are interpreted as characteristics of the attacker. However, this definition imposes a condition on the output, but not on the mechanism which produced the output. As such, it does not offer strong semantic guarantees like the other definitions presented in this survey.

2.2.5.2 *Removing the effect of correlations in the data*

In [36], however, the authors argue that in the presence of correlations in the data, noiseless privacy can be too strong, and make it impossible to learn global properties of the data. Indeed, if one record can have an arbitrarily large influence on the rest of the data, conditioning on the value of this record can lead to very distinguishable outputs even if the mechanism only depends on global properties of the data. To fix this problem, they propose *distributional DP* (DistDP), an alternative definition that that only conditions the data-generating distribution on one possible value of the target record, and quantifies the information leakage from the mechanism. DistDP uses a simulator, similarly to variants introduced in Section 2.2.7.

As we show later, this creates definitional issues in the presence of partial knowledge. These issues are discussed in length in Section 3.1.2, where we also introduce an alternative definition that captures the same intuition without encountering the same problems: *causal DP*.

Definition 34 ((Θ, ε)-causal differential privacy)**.** *Given a family Θ of probability distributions on $\mathcal{D}$, a mechanism $\mathcal{M}$ satisfies* (Θ, ε)-causal DP *(CausDP) if for all probability distributions $\theta \in \Theta$, for all i and all $t, t' \in \mathcal{T}$:*

$$\mathcal{M}(D)_{|D\sim\theta, D(i)=t} \approx_\varepsilon \mathcal{M}(D_{i\to t'})_{|D\sim\theta, D(i)=t} .$$

where $D_{i\to t'}$ is the dataset obtained by changing the i-th record of D into t'.

In this definition, one record is changed *after* the dataset has been generated, so it does not affect other records through dependence relationships. These dependence relationships are the only difference between noiseless privacy and causal DP: as we show in Proposition 22 in Section 3.1.2.2, when each record is independent of all others, this definition is equivalent to noiseless privacy.

2.2.5.3 *Multidimensional definitions*

Limiting the background knowledge of an attacker is orthogonal to the dimensions introduced previously: one can modify the risk model, introduce different neighborhood definitions, or even vary the privacy parameters across the protected properties along with limiting the attacker's background knowledge.

Combination with Q

Modifying the risk model while limiting the attacker's background knowledge has interesting consequences. In Section 3.1.3, we show that two options are possible: either consider the partial knowledge as additional information given to the attacker or let the attacker *influence* the partial knowledge. This distinction between an *active* and a *passive* attacker does not matter if only the worst-case scenario is considered, like in noiseless privacy. However, under different risk models, such as allowing a small probability of error, they lead to two different definitions: *active partial knowledge DP* (APKDP) and *passive partial knowledge DP* (PPKDP). Both definitions are introduced and discussed in more detail in Section 3.1.3.

To model an active attacker, APKDP quantifies over all possible values of the partial knowledge. It was first introduced in [36, 46] as *noiseless privacy*, with an additional δ parameter. We reformulate it to clarify that it implicitly assumes an active attacker. One specialization of this definition is *DP under sampling* [255] (DPuS), which mandates DP to be satisfied after a random sampling is applied to the dataset. The authors use this definition to show that applying k-anonymity to a randomly sampled dataset provides differential privacy; but this definition could also be used on its own, to model the attacker's uncertainty using a randomly sampled distribution.

PPKDP is strictly weaker: it models a passive attacker, who cannot choose their partial knowledge, and thus cannot influence the data. In this context, δ does not only apply to the output of the mechanism, but also to the value of the partial knowledge.

Causal DP can also be adapted to a risk model similar to (ε, δ)-DP: in [58], authors introduce a similar notion to causal DP as *inherit DP* (InhDP), with the small difference that the second dataset is obtained by removing one record from the first dataset, instead of replacing it; and (ε, δ)-indistinguishability is used. The authors also define *empirical DP* [58][16], which is identical, except the empirical distribution is used instead of the actual data distribution, in context where the latter is unknown. In both cases, the influence of δ on the attacker model is unclear.

Combination with N

Modifying the neighborhood definition is simpler: it is clearly orthogonal to the dimensions introduced in this section. In all definitions of this section so far, the two possibilities between which the adversary must distinguish are similar to bounded DP. This can easily be changed to choose other properties to protect from the attacker. This is done in *pufferfish privacy* [228], which extends the concept of neighboring datasets to neighboring *distributions* of datasets.

Definition 35 ($(\Theta, \Phi, \varepsilon)$-pufferfish privacy [228])**.** *Given a family of probability distributions Θ on $\mathcal{D}$, and a family of pairs of predicates Φ on datasets, a mechanism $\mathcal{M}$ verifies $(\Theta, \Phi, \varepsilon)$-*pufferfish privacy *(PFPr) if for all distributions $\theta \in \Theta$ and all pairs of predicates $(\phi_1, \phi_2) \in \Phi$:*

$$\mathcal{M}(D)_{|D\sim\theta,\phi_1(D)} \approx_\varepsilon \mathcal{M}(D)_{|D\sim\theta,\phi_2(D)}$$

[16] Another definition with the same name is introduced in [5], we mention it in Section 2.2.10.1.

Pufferfish privacy extends the concept of neighboring datasets to neighboring *distributions* of datasets; starting with a set of data-generating distributions, then conditioning them on sensitive attributes. The result compares pairs of distributions encompasses noiseless privacy, as well as other notions. For example, it captures *bayesian DP*[17] (BayDP [248]), introduced in [248], in which neighboring dataset have up to k fixed records in common and all other records are generated randomly from a distribution π.

The same idea can be formalized by comparing pairs of distributions directly. This is done in [206, 219] via *distribution privacy* (DnPr). The two formalisms are equivalent: an arbitrary pair of distributions can be seen as a single distribution, conditioned on the value of a secret parameter. Distribution privacy was instantiated in [162] via *profile-based DP* (PBDP), in which the attacker tries to distinguish between different probabilistic user profiles.

Further relaxations encompassing the introduced dimensions are *probabilistic distribution privacy* [219] (PDnPr), a combination of distribution privacy and probabilistic DP, *extended distribution privacy* [219] (EDnPr), a combination of distribution privacy and $d_{\mathfrak{D}}$-privacy, *divergence distribution privacy* [218] (DDnPr), a combination of distribution privacy, and *extended divergence distribution privacy* [218] (EDDnPr), which combines the latter two definitions. Finally, *divergence distribution privacy with auxiliary inputs* [218] considers the setting where the attacker might not know the input probability distribution perfectly.

Definitions of this section are an active area of research; a typical question is to quantify in which conditions deterministic mechanisms can provide some level of privacy. However, they are not used a lot in practice, likely because of their brittleness: if the assumptions about the attacker's limited background knowledge are wrong in practice, then the definitions do not provide any guarantee of protection.

2.2.6 *Change in formalism (F)*

The definition of differential privacy using ε-indistinguishability compares the distribution of outputs given two neighboring inputs. This is not the only way to capture the idea that a attacker should not be able to gain too much information on the dataset. Other formalisms have been proposed, which model the attacker more explicitly.

One such formalism reformulates DP in terms of hypothesis testing by limiting the type I and the type II error of the hypothesis that the output O of a mechanism originates from D_1 (instead of D_2). Other formalisms model the attacker explicitly, by formalizing their prior belief as a probability distribution over all possible datasets. This can be done in two distinct ways. Some variants consider a specific prior (or family of possible priors) of the attacker, implicitly assuming a limited background knowledge, like in Section 2.2.5. We show that these variants can be interpreted as changing the prior-posterior bounds of the attacker. Finally, rather than comparing prior and posterior, a third formalism compares two possible posteriors, quantifying over all possible priors.

[17] There are two other notions with the same name: introduced in [367, 399], we mention them in Section 2.2.3 and 2.2.4 respectively.

Definitions in this section provide a deeper understanding of the guarantees given by differential privacy, and some of them lead to tighter and simpler theorems on differential privacy, like composition or amplification results.

2.2.6.1 *Hypothesis testing*

First, differential privacy can be interpreted in terms of *hypothesis testing* [213, 385]. In this context, an adversary who wants to know whether the output O of a mechanism originates from D_1 (the *null hypothesis*) or D_2 (the *alternative hypothesis*). Calling S the rejection region, the probability of false alarm (type I error), when the null hypothesis is true but rejected, is $\mathbb{P}_{FA} = \mathbb{P}\left[\mathcal{M}(D_1) \in S\right]$. The probability of missed detection (type II error), when the null hypothesis is false but retained, is $\mathbb{P}_{FA} = \mathbb{P}\left[\mathcal{M}(D_2) \in \mathcal{O}/S\right]$.

It is possible to use these probabilities, to reformulate DP: a mechanism is ε-DP iff for all $S \subseteq \mathcal{O}$, $\mathbb{P}_{FA} + e^{\varepsilon}\mathbb{P}_{MD} \geq 1$ and $e^{\varepsilon}\mathbb{P}_{FA} + \mathbb{P}_{MD} \geq 1$; or equivalently, iff $\mathbb{P}_{FA} + \mathbb{P}_{MD} \geq \frac{2}{1+e^{\varepsilon}}$. Similarly, a mechanism is (ε, δ)-DP iff $(\mathbb{P}_{FA}, \mathbb{P}_{MD}) = \{(\alpha, \beta) \in [0,1] \times [0,1] | (1 - \alpha \leq e^{\varepsilon}\beta + \delta)\}$.

This hypothesis testing interpretation was used in [113] to define *f-differential privacy*, which avoids difficulties associated with divergence based relaxations. Specifically, its composition theorem is lossless as it provides a computationally tractable tool for analytically approximating the privacy loss. Moreover, there is a general duality between f-DP and infinite collections of (ε, δ)-DP guarantees.

Definition 36 (f-differential privacy [113])**.** *Let $f : [0,1] \to [0,1]$ be a convex, continuous, and non-increasing function such that for all $x \in [0,1]$, $f(x) \leq 1 - x$. A privacy mechanism $\mathcal{M}$ satisfies* f-differential privacy *(f-DP) if for all neighboring D_1, D_2 and all $x \in [0,1]$:*

$$\inf_{S} \{1 - \mathbb{P}\left[\mathcal{M}(D_2) \in S\right] | \mathbb{P}\left[\mathcal{M}(D_1) \in S\right] \leq x\} \geq f(x).$$

Here, S is the rejection region; and the infimum is the trade-off function between $\mathcal{M}(D_1)$ and $\mathcal{M}(D_2)$. The authors also introduce *Gaussian differential privacy* (GaussDP) as an instance of f-differential privacy, which tightly bounds from below the hardness of determining whether an individual's data was used in a computation than telling apart two shifted Gaussian distributions.

2.2.6.2 *Changing the shape of the prior-posterior bounds*

As we mentioned in Section 2.1.6.2, differential privacy can be interpreted as giving a bound on the posterior of a Bayesian attacker as a function of their prior. This is exactly the case in *indistinguishable privacy* (IndPr), an equivalent reformulation of differential privacy defined in [260]. Assuming that the attacker is trying to distinguish between options $D = D_1$ and $D = D_2$, where D_1 corresponds to the option "$t \in D$" and D_2 to "$t \notin D$", we end up with a bound on the posterior probability depending on the prior probability that we derived in Section 2.1.6.2 (Proposition 1)[18]:

$$\frac{\mathbb{P}\left[t \in D\right]}{e^{\varepsilon} + \left(1 - e^{\varepsilon}\right)\mathbb{P}\left[t \in D\right]} \leq \mathbb{P}\left[t \in D \,\middle|\, \mathcal{M}(D) = O\right] \leq \frac{e^{\varepsilon} \cdot \mathbb{P}\left[t \in D\right]}{1 + \left(e^{\varepsilon} - 1\right)\mathbb{P}\left[t \in D\right]}$$

We visualize these bounds on the top-left graph of Figure 2.8.

Some variants of differential privacy use this idea in their formalism, rather than obtaining the posterior bound as a corollary to the classical DP definition. For example, *positive membership privacy* [254] requires that the posterior does not increase too much compared to the prior. Like noiseless privacy, it assumes an attacker with limited background knowledge.

Definition 37 ((Θ, ε)-positive membership privacy [254])**.** *A privacy mechanism $\mathcal{M}$ provides* (Θ, ε)-positive membership privacy *(PMPr) if for any distribution $\theta \in \Theta$, any record $t \in \mathcal{D}$ and any $S \subseteq \mathcal{O}$:*

$$\begin{aligned}\mathbb{P}_{D \sim \theta}\left[t \in D | \mathcal{M}(D) \in S\right] &\leq e^{\varepsilon} \mathbb{P}_{D \sim \theta}\left[t \in D\right] \\ \mathbb{P}_{D \sim \theta}\left[t \notin D | \mathcal{M}(D) \in S\right] &\geq e^{-\varepsilon} \mathbb{P}_{D \sim \theta}\left[t \notin D\right].\end{aligned}$$

Note that this definition is *asymmetric*: the posterior is bounded from above, but not from below. In the same paper, the authors also define *negative membership privacy* (NMPr), which provides the symmetric lower bound, and *membership privacy*[19] (MPr), which is the conjunction of positive and negative membership privacy. They show that this definition can represent differential privacy (in its bounded and unbounded variants), as well as other definitions like *differential identifiability* [244] and *sampling DP* [253, 255], which we mention in Section 2.2.10.1. Bounding the ratio between prior and posterior by e^{ε} is also done in the context of location privacy: in [115], authors define *ε-DP location obfuscation*, which formalizes the same intuition as membership privacy.

A previous attempt at formalizing the same idea was presented in [326] as *adversarial privacy*. This definition is similar to positive membership privacy, except only the first relation is used, and there is a small additive δ as in approximate DP. We visualize the corresponding bounds on the third figure of Figure 2.8.

Definition 38 ($(\Theta, \varepsilon, \delta)$-adversarial privacy [326])**.** *An algorithm $\mathcal{M}$ is* $(\Theta, \varepsilon, \delta)$-adversarial private *(AdvPr) if for all $S \subseteq \mathcal{O}$, tuples t, and distributions $\theta \in \Theta$:*

$$\mathbb{P}_{D \sim \theta}\left[t \in D | \mathcal{M}(D) \in S\right] \leq e^{\varepsilon} \cdot \mathbb{P}_{D \sim \theta}\left[t \in D\right] + \delta.$$

Adversarial privacy (without δ) was also redefined in [391] as *information privacy*[20].

Finally, *aposteriori noiseless privacy* is a similar variant of noiseless privacy introduced in [46]; the corresponding bounds can be seen on the last figure of Figure 2.8.

[18] Note that the original formalization used in [260] was more abstract, and it used polynomially bounded adversaries, which we introduce in Section 2.2.8.

[19] Another definition with the same name is introduced in [336], we mention it in Section 2.2.1.4.

[20] Another definition with the same name is introduced in [119], we mention it later in this section.

Definition 39 ((Θ, ε)-aposteriori noiseless privacy [46]). *A mechanism $\mathcal{M}$ is said to be* (Θ, ε)-aposteriori noiseless private *(ANPr) if for all $\theta \in \Theta$, all $S \subseteq O$ and all i:*

$$D(i)_{|D\sim\theta, \mathcal{M}(D)\in S} \approx_\varepsilon D(i)_{|D\sim\theta}.$$

We visualize the prior/posterior bounds for these various definitions in Figure 2.8.

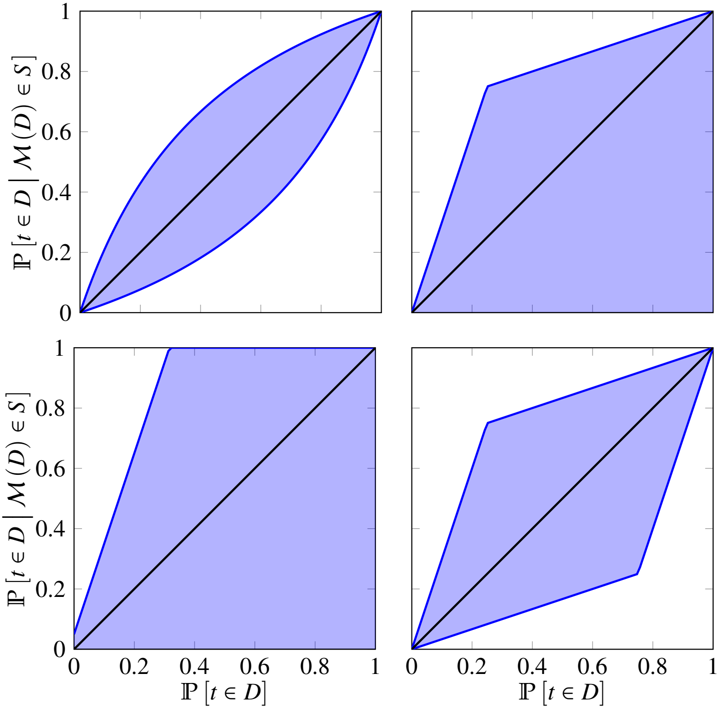


Figure 2.8: From left to right and top to bottom, using $\varepsilon = \ln 3$: posterior-prior bounds in differential privacy, positive membership privacy, adversarial privacy (with $\delta = 0.05$) and aposteriori noiseless privacy.

2.2.6.3 *Comparing two posteriors*

In [217], the authors propose an approach that captures an intuitive idea proposed by Dwork in [122]: "any conclusions drawn from the output of a private algorithm must be similar whether or not an individual's data is present in the input or not". They define *semantic privacy*: instead of comparing the posterior with the prior belief like in DP, this bounds the difference between two posterior belief distributions, depending on which dataset was secretly chosen. The distance chosen to represent the idea that those two posterior belief distributions are close is the statistical distance. One important difference between the definitions in the previous subsection is that semantic privacy quantifies over all possible priors: like in DP, the attacker is assumed to have arbitrary background knowledge.

Definition 40 (ε-semantic privacy [158, 217])**.** *A mechanism $\mathcal{M}$ is ε-*semantically private *(SemPr) if for any distribution over datasets θ, any index i, any $S \subseteq \mathcal{O}$, and any set of datasets $X \subseteq \mathcal{D}$:*

$$\left|\mathbb{P}_{D\sim\theta}\left[D \in X \;\middle|\; \mathcal{M}(D) \in S\right] - \mathbb{P}_{D\sim\theta}\left[D \in X \;\middle|\; \mathcal{M}(D_{-i}) \in S\right]\right| \leq \varepsilon.$$

A couple of other definitions also compare posteriors directly: *inferential privacy* [164] is a reformulation of noiseless privacy, and *range-bounded privacy* [121] (RBPr) requires that two different values of the PLRV are close to each other (instead of being between centered around zero like in ε-DP). It is equivalent to ε-DP up to a change in parameters, and is used as a technical tool to prove composition results.

2.2.6.4 *Multidimensional definitions*

Definitions that limit the background knowledge of the adversary explicitly formulate it as a probability distribution. As such, they are natural candidates for Bayesian reformulations. In [391], the authors introduce *identity DP*, which is an equivalent Bayesian reformulation of noiseless privacy. Another example is *inference-based causal DP*, which we define in Definition 52 in Section 3.1.2.2. It is similar to aposteriori noiseless DP, except it uses causal DP instead of noiseless DP.

Further, it is possible to consider different definitions of neighborhood. In [119], authors introduce *information privacy*[21], which can be seen as a posteriori noiseless privacy combined with free lunch privacy: rather than only considering the knowledge gain of the adversary on one particular user, it considers its knowledge gain about any possible group of values of the dataset.

Definition 41 ((Θ, ε)-information privacy [119])**.** *A mechanism $\mathcal{M}$ satisfies* (Θ, ε)-information privacy *(InfPr) if for all probability distributions $\theta \in \Theta$, all $D \in \mathcal{D}$ and all $O \in \mathcal{O}$, $D_{|D\sim\theta} \approx_\varepsilon D_{|D\sim\theta,\mathcal{M}(D)=O}$.*

The authors further prove that if Θ contains a distribution whose support is $\mathcal{D}$, then (Θ, ε)-InfPr implies 2ε-DP.

Apart from the hypothesis testing reformulations, which can be used to improve composition and amplification results, the definitions in this section mostly appear in theoretical research papers, to provide a deeper understanding of guarantees offered by DP and its alternatives. They do not seem to be used in practical applications.

2.2.7 *Relativization of the knowledge gain (R)*

A differentially private mechanism does not reveal more than a bounded amount of probabilistic information about a user. This view does not explicitly take into account other ways information can leak, like side-channel functions or knowledge about the structure of a social

[21] Another definition with the same name is introduced in [391], we mention it earlier in this section.

network. We found two approaches that attempt to include such auxiliary functions in DP variants. One possibility is to weaken DP by allowing a certain amount of leakage; another option is to explicitly forbid the mechanism to reveal more than another function, considered to be safe for release.

2.2.7.1 *Taking into account auxiliary leakage function*

In [257], authors define *bounded leakage DP*, which quantifies the privacy that is maintained by a mechanism despite bounded, additional leakage of information by some leakage function. Interestingly, this leakage function P shares the randomness of the privacy mechanism: it can, for example, capture side-channel leakage from the mechanism's execution. In the formal definition of this DP variant, the randomness is *explicit*: the privacy mechanism and the leakage takes the random bits $r \in \{0,1\}^*$ as an additional parameter.

Definition 42 ((P,ε,δ)-bounded leakage differential privacy [257])**.** *Let $P : \mathcal{D} \times \{0,1\}^*$ be a leakage function. A privacy mechanism $\mathcal{M}$ is (P,ε,δ)-*bounded leakage differentially private *(BLDP) if for all pairs of neighboring datasets D_1 and D_2, all outputs O_P of P such that $\mathbb{P}\left[P\left(D_1,r\right)=O_P\right] \neq 0$ and $\mathbb{P}\left[P\left(D_2,r\right)=O_P\right] \neq 0$, and all sets of outputs $S \subseteq \mathcal{O}$:*

$$\mathbb{P}\left[\mathcal{M}\left(D_1,r\right) \in S \;\middle|\; P\left(D_1,r\right)=O_P\right] \leq e^{\varepsilon} \cdot \mathbb{P}\left[\mathcal{M}\left(D_2,r\right) \in S \;\middle|\; P\left(D_2,r\right)=O_P\right] + \delta$$

where the randomness is taken over the random bits r.

As expected, if there is no leakage (P is a constant function), this is simply (ε,δ)-DP. The authors also show that it is closed for post-processing and composable. Furthermore, if the privacy mechanism is independent from the leakage function, it is strictly weaker than differential privacy.

2.2.7.2 *Borrowing concepts from zero-knowledge proofs*

When using the associative interpretation with the independence assumption, it is unclear how to adapt DP to correlated datasets like social networks: data about someone's friends might reveal sensitive information about this person. The causal interpretation of DP does not suffer from this problem, but how to adapt the associative view to such correlated contexts? Changing the definition of the neighborhood is one possibility (see Section 2.2.3.1), but it requires knowing in advance the exact impact of someone on other records. A more robust option is to impose that the information released does not contain more information than the result of some predefined algorithms on the data, without the individual in question. The method for formalizing this intuition borrows ideas from *zero-knowledge proofs* [170].

Instead of imposing that the result of the mechanism is roughly *the same* on neighboring datasets D_1 and D_2, the intuition is to impose that the result of the mechanism on D_1 can be *simulated* using only some information about D_2. The corresponding definition, called *zero-knowledge privacy* and introduced in [161], captures the idea that the mechanism does not leak more information on a given target than a certain class of aggregate metrics. This class, called *model of aggregate information* in [161], is formalized by a family of (possibly randomized) family of algorithms Agg.

Definition 43 ($(\text{Agg}, \varepsilon)$-zero-knowledge privacy [161]). *Let* Agg *be a family of (possibly randomized) algorithms* agg. *A privacy mechanism $\mathcal{M}$ is* $(\text{Agg}, \varepsilon)$-zero-knowledge private *(ZKPr) if there exists an algorithm* $\text{agg} \in \text{Agg}$ *and a simulator* Sim *such as for all datasets D and indices i, $\mathcal{M}(D) \approx_{\varepsilon} \text{Sim}(\text{agg}(D_{-i}))$.*

In [161], authors show that $(\text{Agg}, \varepsilon)$-ZKPr implies 2ε-DP for any Agg, while ε-DP implies $(\text{Agg}, \varepsilon)$-ZKPr if the identity function is in Agg. This is yet another way to formalize the intuition that differential privacy protects against attackers who have full background knowledge.

2.2.7.3 *Multidimensional definitions*

Using a simulator allows making statements of the type "this mechanism does not leak more information on a given target than a certain class of aggregate metrics". Similarly to noiseless privacy, it is possible to explicitly limit the attacker's background knowledge using a data-generating probability distribution, as well as vary the neighborhood definitions to protect other types of information than the presence and characteristics of individuals. This is done in [36] as *coupled-worlds privacy*, a generalization of distributional DP, where a family of functions priv represents the protected attribute.

Definition 44 ($(\Theta, \Gamma, \varepsilon)$-coupled-worlds privacy [36]). *Let Γ be a family of pairs of functions* $(\text{agg}, \text{priv})$. *A mechanism $\mathcal{M}$ satisfies* $(\Theta, \Gamma, \varepsilon)$-coupled-worlds privacy *(CWPr) if there is a simulator* Sim *such that for all distributions $\theta \in \Theta$, all* $(\text{agg}, \text{priv}) \in \Gamma$*, and all possible values p:*

$$\mathcal{M}(D)_{|D \sim \theta, \text{priv}(D)=p} \approx_{\varepsilon} \text{Sim}(\text{agg}(D))_{|D \sim \theta, \text{priv}(D)=p}$$

A special case of coupled-worlds privacy is also introduced in [36] as *distributional DP*, already mentioned in Section 2.2.5: each function priv captures the value of a single record, and the corresponding function agg outputs all other records.

Coupled-worlds privacy is a good example of combining variants from different dimensions: it changes several aspects of the original definition according to from **N**, **B** and **R**. Moreover, **Q** and **F** can easily be integrated with this definition by using (ε, δ)-indistinguishability with a Bayesian reformulation. This is done explicitly in *inference-based coupled-worlds privacy* [36]; the same paper also introduces inference-based distributional differential privacy (IBDDP).

Definition 45 ($(\Theta, \Gamma, \varepsilon, \delta)$-inference-based coupled-worlds privacy [36]). *Given a family Θ of probability distributions on $\mathcal{D} \times \mathcal{B}$, and a family Γ of pairs of functions* $(\text{agg}, \text{priv})$*, a mechanism $\mathcal{M}$ satisfies* $(\Theta, \Gamma, \varepsilon, \delta)$-inference-based coupled-worlds privacy *(IBCWPr) if there is a simulator* Sim *such that for all distributions $\theta \in \Theta$, and all* $(\text{agg}, \text{priv}) \in \Gamma$*:*

$$\text{priv}(D)_{|(D,B) \sim \theta, \mathcal{M}(D)=O, B=\hat{B}} \approx_{(\varepsilon,\delta)} \text{priv}(D)_{|(D,B) \sim \theta, \text{Sim}(\text{agg}(D))=O, B=\hat{B}}$$

with probability at least $1-\delta$ over the choice of O and $\hat{B}$.

Random DP was also combined with an idea similar to ZKPr: in [35], the authors introduce *typical stability*, which combines random DP with approximate DP, except that rather using

(ε, δ)-indistinguishability between two outputs of the mechanism, it uses a simulator that only knows data-generating distribution.

Definition 46 ($(\Theta, \gamma, \varepsilon, \delta)$-typical stability [35]). *Given a family Θ of probability distributions on $\mathcal{D}$, a mechanism $\mathcal{M}$ satisfies* $(\Theta, \gamma, \varepsilon, \delta)$-typical stability *(TypSt) if for all distributions $\theta \in \Theta$, there is a simulator* Sim *such that with probability at least* $1 - \gamma$ *over the choice of $D \sim \theta$, $\mathcal{M}(D) \approx_{\varepsilon,\delta} \text{Sim}(\theta)$.*

In the same paper, the authors introduce a variant of the same definition with the same name, which compares two outputs of the mechanism; this is essentially a combination between DlPr [53, 333] and approximate DP.

We did not find any evidence that the variants and extensions of this section are used outside of theoretical papers exploring the guarantees they provide.

2.2.8 *Computational power (C)*

The ε-indistinguishability property in DP is *information-theoretic*: the attacker is implicitly assumed to have infinite computing power. This is unrealistic in practice, so it is natural to consider definitions where the attacker only has polynomial computing power. Changing this assumption leads to weaker data privacy definitions. In [285], two approaches have been proposed to formalize this idea: either modeling the distinguisher explicitly as a polynomial Turing machine, either allow a mechanism not to be technically differentially private, as long as one cannot distinguish it from a truly differentially private one.

2.2.8.1 *Using a polynomial distinguisher on the output*

The attacker is not explicit in the formalization of DP based on ε-indistinguishability. It is possible change the definition to make this attacker explicit: model it as a *distinguisher*, who tries to guess whether a given output O comes from a dataset D_1 or its neighbor D_2. In doing so, it becomes straightforward to require this attacker to be computationally bounded: simply model it as a probabilistic polynomial-time Turing machine. In [285], the authors introduce IndCDP, short for *indistinguishability-based computational differential privacy*.

Definition 47 (ε_κ-IndCDP [285]). *A family $(\mathcal{M}_\kappa)_{\kappa \in \mathbb{N}}$ of privacy mechanisms $\mathcal{M}_\kappa$ provides* ε_κ-IndCDP *if there exists a negligible function* neg *such that for all non-uniform probabilistic polynomial-time Turing machines Ω (the distinguisher), all polynomials $p(\cdot)$, all sufficiently large $\kappa \in \mathbb{N}$, and all datasets $D_1, D_2 \in \mathcal{D}$ of size at most $p(\kappa)$ that differ only one one record:*

$$\mathbb{P}\left[\Omega(\mathcal{M}(D_1)) = 1\right] \leq e^{\varepsilon_\kappa} \cdot \mathbb{P}\left[\Omega(\mathcal{M}(D_2)) = 1\right] + \text{neg}(\kappa)$$

where neg *is a function that converges to zero asymptotically faster than the reciprocal of any polynomial.*

This definition can also be expressed using the authors define *differential indistinguishability*, a notion defined in [24] that adapts ε-indistinguishability to a polynomially bounded attacker.

2.2.8.2 *Using a polynomial distinguisher on the mechanism*

Another natural option is to require that the mechanism "looks like" a truly differentially private mechanism, at least to a computationally bounded distinguisher. In [285], the authors introduce SimCDP, short for *simulation-based computational differential privacy*.

Definition 48 (ε_κ-SimCDP [285])**.** *A family $(\mathcal{M}_\kappa)_{\kappa\in\mathbb{N}}$ of privacy mechanisms $\mathcal{M}_\kappa$ provides* ε_κ-SimCDP *if there exists a family $(\mathcal{M}'_\kappa)_{\kappa\in\mathbb{N}}$ of ε_κ-DP and a negligible function* neg *such that for all non-uniform probabilistic polynomial-time Turing machines Ω, all polynomials $p(\cdot)$, all sufficiently large $\kappa \in \mathbb{N}$, and all datasets $D \in \mathcal{D}$ of size at most $p(\kappa)$:*

$$\mathbb{P}\left[\Omega(\mathcal{M}(D)) = 1\right] - \mathbb{P}\left[\Omega(\mathcal{M}'(D)) = 1\right] \leq \operatorname{neg}(\kappa)$$

where neg *is a function that converges to zero asymptotically faster than the reciprocal of any polynomial.*

In [285], the authors show that ε_κ-SimCDP implies ε_κ-IndCDP.

2.2.8.3 *Multidimensional definitions*

IndCDP has been adapted to different settings, and extended to arbitrary neighborhood relationships. In *output constrained DP* (OCDP), introduced in [194], the setting is a two-party computation, each party can have its own set of privacy parameters (ε_A, ε_B, δ_A, and δ_B—the δ parameters correspond to the $\operatorname{neg}(\kappa)$ term in IndCDP), and the neighborhood relationship is determined by a function f. The authors also propose *DP for Record Linkage* (DPRL), an instance of OCDP that uses for a specific function f that captures the need to protect non-matching records during the execution of a private record linkage protocol.

Some DP variants which explicitly model an adversary with a simulator can relatively easily be adapted to model a computationally bounded adversary, simply by imposing that the simulator must be polynomial. This is done explicitly in [161], where the authors define *computational zero-knowledge privacy* (CZKPr), which could also be adapted to e.g., the two coupled-worlds privacy definitions as well.

Further, although we have not seen this done in practice in existing literature, the idea behind SimCDP can in principle be adapted to any other definition: rather than requiring that a given definition holds in an information-theoretic fashion, it should be possible to require that the mechanism "looks like" a mechanism which genuinely satisfies the definition.

Limiting the computational power of the attacker is a reasonable assumption, but for a large class of queries, it cannot provide significant benefits over classical DP in the typical client-server setting [177]. Thus, existing work using it focuses on multi-party settings [42].

2.2.9 *Summarizing table*

In this section we summarize the results from the previous 7 sections into a table where we list the known relations and show the properties with either referring to the original proof or creating a novel one.

In Table 2.20, we list the differential privacy variants and extensions introduced in this work. For each, we specify their name, parameters and where they were introduced (column 1), which dimensions they belong to (column 2), which axioms they satisfy (column 3, post-processing on the left and convexity on the right), whether they are composable (column 4) and how they relate to other differential privacy notions (column 5). We do not list definitions whose only difference is that they apply DP to other types of input, like those from Section 2.2.3.3, or geolocation-specific definitions.

The references for each claim present in Table 2.20 are listed after the table, while the proofs follow in Section 2.2.9.1.

Name & References	Dimensions[18]	Axioms		Cp.[13]	Relations
		P.P.[11]	Cv.[12]		
(ε, δ)-DP, or (ε, δ)-approximate DP [126] also known as max-KL stability [37]	**Q**	✓[2]	✓[2]	✓[8]	(ε, δ)**-DP** $\supset^{\prec}$ ε-DP
(ε, δ)-probabilistic DP [267, 277] also known as (ε, δ)-DP in distribution [60]	**Q**	✗[1]	✗[3]	✓[8]	(ε, δ)-DP $\sim$ (ε, δ)**-ProDP** $\supset^{\prec}$ ε-DP
$(\varepsilon, \delta_a, \delta_p)$-Relaxed DP [407]	**Q**	✗[1]	✗[3]	✓[8]	(ε, δ_p)-ProDP $\subset^{\succ}$ $(\varepsilon, \delta_a, \delta_p)$**-RelDP** $\supset^{\prec}$ (ε, δ_a)-DP
ε-Kullback-Leiber Pr [31, 88]	**Q**	✓[2]	✓[2]	✓[8]	(ε, δ)-DP $\prec$ ε**-KLPr** $\prec$ ε-DP
(α, ε)-Rényi DP [284]	**Q**	✓[2]	✓[2]	✓[8]	ε-KLPr $\subset^{\prec}$ (α, ε)**-RenyiDP** $\supset^{\prec}$ ε-DP
binary-$\lvert\chi\rvert^{\alpha}$ DP [380]	**Q**	?	?	?	**b-**$\lvert\chi\rvert^{\alpha}$ **DP** $\succ$ (ε, δ)-DP
tenary-$\lvert\chi\rvert^{\alpha}$ DP [380]	**Q**	?	?	?	**t-**$\lvert\chi\rvert^{\alpha}$ **DP** $\succ$ (ε, δ)-DP
ε-total variation Pr [31]	**Q**	?	?	?	ε**-TVPr** $\subset$ b-$\lvert\chi\rvert^{\alpha}$ DP
ε-quantum DP [81]	**Q**	?	?	?	
ε-mutual-information DP [88]	**Q**	✓[2]	✓[2]	✓[8]	(ε, δ)-DP $\prec$ ε**-MIDP** $\prec$ ε-KLPr
(μ, τ)-mean concentrated DP [132]	**Q**	✗[1]	?	✓[8]	(ε, δ)-DP $\prec$ (μ, τ)**-mCoDP** $\prec$ ε-DP
(ξ, ρ)-zero concentrated DP [57]	**Q**	✓[2]	✓[2]	✓[8]	(ξ, ρ)**-zCoDP** $\sim$ (μ, τ)-mCoDP
(ξ, ρ, δ)-approximate CoDP [57]	**Q**	✗[2]	?	✓[8]	(ε, δ)-DP $\succ$ (ξ, ρ, δ)**-ACoDP** $\supset^{\prec}$ (ξ, ρ)-zCoDP
(ξ, ρ, ω)-bounded CoDP [57]	**Q**	✓[2]	✓[2]	✓[8]	(ξ, ρ, ω)**-bCoDP** $\supset^{\prec}$ (ξ, ρ)-zCoDP
(η, τ)-truncated CoDP [81]	**Q**	✓[2]	✓[2]	✓[8]	(η, τ)**-tCoDP [81]** $\sim$ ε-DP
(ρ, ω)-truncated CoDP [56]	**Q**	✓[2]	✓[2]	✓[8]	(ρ, ω)**-tCoDP [56]** $\subset^{\prec}$ (ξ, ρ, ω)-bCoDP

continued

Name & References	Dimensions[18]	Axioms		Cp.[13]	Relations
		P.P.[11]	Cv.[12]		
(f,ε)-divergence DP [31]	**Q**	✓[2]	✓[2]	?	(f,ε)**-DivDP** $\supset$ most definitions in **Q**
(f_k,ε)-divergence DP [118]	**Q**	✓[2]	✓[2]	?	(f_k,ε)**-DivDP** $\subset$ (f,ε)-DivDP
(H,f,ε)-capacity bounded DP [69]	**Q**	✓[2]	✓[2]	?	(H,f,ε)**-CBDP** $\subset^{\prec}$ (f,ε)-DivDP
ε-unbounded DP [227]	**N**	✓[1]	✓[1]	✓[9]	ε-DP $\sim$ ε**-uBoDP** $\subset^{\sim}$ (c,ε)-GrDP
ε-bounded DP [227] also known as per-person DP [149]	**N**	✓[1]	✓[1]	✓[9]	ε**-BoDP** $\prec$ ε-DP
ε-attribute/bit DP [227]	**N**	✓[1]	✓[1]	✓[9]	ε**-BitDP** $\prec$ ε**-AttDP** $\prec$ ε-BoDP
ε-element DP [22]	**N**	✓[1]	✓[1]	✓[9]	ε**-ELDP** $\prec$ ε-DP
(P,ε)-one-sided DP [231]	**N**	✓[1]	✓[1]	✓[9]	(P,ε)**-OnSDP** $\supset^{\prec}$ ε-BoDP
(N,k,ε)-sensitive privacy [23]	**N**	✓[1]	✓[1]	✓[9]	(N,k,ε)**-SenPr** $\subset$ ε-(P,ε)-OnSDP
(P,ε)-anomaly-restricted DP [51]	**N**	✓[1]	✓[1]	✓[9]	(P,ε)**-ARDP** $\supset^{\prec}$ ε-DP
(c,ε)-group DP [123] also known as DP under correlation [73]	**N**	✓[1]	✓[1]	✓[9]	(c,ε)**-GrDP** $\supset^{\sim}$ ε-DP
(R,c,ε)-dependent DP [258]	**N**	✓[1]	✓[1]	✓[9]	(R,c,ε)**-DepDP** $\supset$ (c,ε)-GrDP
$(\mathcal{A},\varepsilon)$-bayesian DP [399]	**N**	✓[1]	✓[1]	✓[9]	$(\mathcal{A},\varepsilon)$**-BayDP** **[399]** $\supset$ (R,c,ε)-DepDP
$(\mathcal{A},\varepsilon)$-correlated DP [392, 393]	**N**	✓[1]	✓[1]	✓[9]	$(\mathcal{A},\varepsilon)$**-CorDP** $\subset$ $(\mathcal{A},\varepsilon)$-BayDP [399]
$(\mathcal{A},\varepsilon)$-prior DP [256]	**N**	✓[1]	✓[1]	✓[9]	$(\mathcal{A},\varepsilon)$**-PriDP** $\supset$ $(\mathcal{A},\varepsilon)$-BayDP [399]
ε-free lunch Pr [227]	**N**	✓[1]	✓[1]	✓[9]	ε**-FLPr** $\succ$ all definitions in **N**

continued

Name & References	Dimensions[18]	Axioms		Cp.[13]	Relations
		P.P.[11]	Cv.[12]		
(D,ε)-individual DP [349] also known as conditioned DP [64]	**N**	$\checkmark^{1}$	$\checkmark^{1}$	$\checkmark^{9}$	(D,ε)**-IndDP** $\prec$ ε-DP
(D,t,ε)-per-instance DP [379]	**N**	$\checkmark^{1}$	$\checkmark^{1}$	$\checkmark^{9}$	(D,t,ε)**-PIDP** $\prec$ (D,ε)-IndDP
$(\mathcal{R},\varepsilon)$-generic DP [145, 227]	**N**	$\checkmark^{1}$	$\checkmark^{1}$	$\checkmark^{9}$	$(\mathcal{R},\varepsilon)$**-GcDP** **[227]** $\supset$ most definitions in **N**
(d,Δ,ε)-constrained DP [409] also known as adjacent DP [235] and DP under a neighborhood [145]	**N**	$\checkmark^{1}$	$\checkmark^{1}$	$\checkmark^{9}$	(d,Δ,ε)**-ConsDP** $\sim$ $(\mathcal{R},\varepsilon)$-GcDP
$(d,\Delta,S_{\mathcal{D}},\varepsilon)$-distributional Pr [409]	**N**	$\checkmark^{1}$	$\checkmark^{1}$	$\checkmark^{9}$	$(d,\Delta,S_{\mathcal{D}},\varepsilon)$**-DlPr** **[409]** $\subset$ $(\mathcal{R},\varepsilon)$-GcDP
(f,ε)-sensitivity-induced DP [335]	**N**	$\checkmark^{1}$	$\checkmark^{1}$	$\checkmark^{9}$	ε**-SIDP** $\subset$ $(\mathcal{R},\varepsilon)$-GcDP
(Q,ε)-induced-neighbors DP [227]	**N**	$\checkmark^{1}$	$\checkmark^{1}$	$\checkmark^{9}$	(Q,ε)**-INDP** $\subset$ $(\mathcal{R},\varepsilon)$-GcDP
(G,Q,ε)-blowfish Pr [187, 193]	**N**	$\checkmark^{1}$	$\checkmark^{1}$	$\checkmark^{9}$	(G,Q,ε)**-BFPr** $\subset$ $(\mathcal{R},\varepsilon)$-GcDP $\supset^{\prec}$ (Q,ε)-INDP
ε-adjacency-relation div. DP [218]	**Q,N**	$\checkmark^{21}$	$\checkmark^{21}$	?	$(\mathcal{R},\varepsilon)$-GcDP [227] $\subset$ $(\mathcal{R},f,\varepsilon)$**-ARDDP** $\supset$ (f,ε)-DivDP
Ψ-personalized DP [134, 165, 212, 261, 302] also known as heterogeneous DP [10]	**V**	$\checkmark^{4}$	$\checkmark^{4}$	$\checkmark^{9}$	Ψ**-PerDP** $\supset$ ε-DP
Ξ-tailored DP [263]	**V**	$\checkmark^{4}$	$\checkmark^{4}$	$\checkmark^{9}$	Ξ**-TaiDP** $\supset$ Ψ-PerDP
$\varepsilon(\cdot)$-outlier Pr [263]	**V**	$\checkmark^{4}$	$\checkmark^{4}$	$\checkmark^{9}$	$\varepsilon(\cdot)$**-OutPr** $\subset$ Ξ-TaiDP
(k,ε)-simple-outlier Pr [263]	**V**	$\checkmark^{4}$	$\checkmark^{4}$	$\checkmark^{9}$	(k,ε)**-SOPr** $\subset$ $\varepsilon(\cdot)$-OutPr
(k,ε)-simple outlier DP [263]	**V**	$\checkmark^{4}$	$\checkmark^{4}$	$\checkmark^{9}$	ε-DP $\succ$ (k,ε)**-SODP** $\subset$ $\varepsilon(\cdot)$-OutPr
$(\bar{k},\bar{\varepsilon})$-staircase outlier DP [263]	**V**	$\checkmark^{4}$	$\checkmark^{4}$	$\checkmark^{9}$	$(\bar{k},\bar{\varepsilon})$**-SCODP** $\subset$ $\varepsilon(\cdot)$-OutPr

continued

Name & References	Dimensions[18]	Axioms		Cp.[13]	Relations
		P.P.[11]	Cv.[12]		
(ε, p, r)-Pareto DP [263]	**V**	✓[4]	✓[4]	✓[9]	(ε, p, r)-**ParDP** $\subset$ Ξ-TaiDP
$(\pi, \gamma, \varepsilon)$-random DP [185]	**V**	?	✗[5]	✓[8]	$(\pi, \gamma, \varepsilon)$-**RanDP** $\supset^{\prec}$ ε-DP
$(\pi, \gamma, \varepsilon)$-predictive DP [184] also known as model-specific DP [270]	**V**	?	?	?	$(\pi, \gamma, \varepsilon)$-RanDP $\prec$ $(\pi, \gamma, \varepsilon)$-**PredDP** $\supset^{\prec}$ ε-DP
$(\theta, \gamma, \varepsilon)$-generalized DP [111]	**V**	?	?	?	$(\theta, \gamma, \varepsilon)$-**GdDP** $\supset^{\prec}$ ε-DP
$d_{\mathcal{D}}$-Pr [67] also known as extended DP [219]	**N,V**	✓[4]	✓[4]	✓[9]	ε-DP $\subset$ $d_{\mathcal{D}}$-**Pr**
ε-weighted DP [322]	**N,V**	✓[4]	✓[4]	✓[9]	ε-**WeiDP** $\subset$ $d_{\mathcal{D}}$-Pr
$(d_{\mathcal{D}}, \varepsilon)$-smooth DP [31]	**N,V**	✓[4]	✓[4]	✓[9]	$(d_{\mathcal{D}}, \varepsilon)$-**SmoDP** $\sim$ $d_{\mathcal{D}}$-Pr
$(d_{\mathcal{D}}, \varepsilon)$-earth mover's Pr [151]	**N,V**	✓[4]	✓[4]	✓[9]	$(d_{\mathcal{D}}, \varepsilon)$-**EMDP** $\subset$ $d_{\mathcal{D}}$-Pr
$(\pi, \varepsilon, \gamma)$-DP on δ location set [396]	**N,V**	✓[4]	✓[4]	✓[9]	$(\pi, \varepsilon, \gamma)$-**LocSetDP**
(ε, γ)-distributional Pr [333, 409]	**N,V**	?	?	?	ε-FLPr $\subset$ (ε, γ)-**DlPr** [53, 333]
$(\varepsilon(\cdot), \delta(\cdot))$-endogenous DP [234]	**Q,V**	✓[4]	✓[4]	✓[9]	(ε, δ)-DP $\subset$ $(\varepsilon(\cdot), \delta(\cdot))$-**EndDP** $\supset^{\prec}$ Ψ-PerDP
$(\pi, \varepsilon, \delta)$-weak Bayesian DP [367]	**Q,V**	✓[1]	?	✓[9]	(ε, δ)-DP $\succ$ $(\pi, \varepsilon, \delta)$-**WBDP** $\prec$ $(\pi, \gamma, \varepsilon)$-RanDP
(Θ, ε)-on average KL Pr [150, 381] also known as average leave-one-out KL stability [150]	**Q,V**	✓[2][1]	?	?[16]	ε-KLPr $\succ$ (Θ, ε)-**avgKLPr** $\prec$ $(\pi, \gamma, \varepsilon)$-RanDP
$(\pi, \varepsilon, \delta)$-Bayesian DP [367]	**Q,V**	✓[1]	✗[5]	✓[9]	(ε, δ)-ProDP $\succ$ $(\pi, \varepsilon, \delta)$-**BayDP** [367] $\prec$ $(\pi, \gamma, \varepsilon)$-RanDP
$(\theta, \varepsilon, \delta)$-privacy at risk [92]	**Q,V**	✓[1]	✗[5]	✓[9]	(ε, δ)-ProDP $\succ$ $(\theta, \varepsilon, \delta)$-**PAR** $\prec$ $(\pi, \gamma, \varepsilon)$-RanDP
$(d_{\mathcal{D}}, \varepsilon, \delta)$-pseudo-metric DP [106]	**Q,N,V**	?	?	✓[8]	(ε, δ)-DP $\subset$ $(d_{\mathcal{D}}, \varepsilon, \delta)$-**PsDP** $\supset^{\prec}$ $d_{\mathcal{D}}$-Pr

continued

Name & References	Dimensions[18]	Axioms		Cp.[13]	Relations
		P.P.[11]	Cv.[12]		
(f,d,ε)-extended divergence DP [218]	**Q,N,V**	✓[4]	✓[4]	?	$d_{\mathcal{D}}$-Pr $\subset$ (f,d,ε)**-EDivDP** $\supset$ (f,ε)-Div DP
$(\mathcal{R},M)$-generic DP [225]	**Q,N,V**	✓[1]	✓[1]	?	$(\mathcal{R},M)$**-GcDP [225]** $\supset$ (ε,δ)-DP
$(\mathcal{R},q)$-abstract DP [225]	**Q,N,V**	✗[1]	✗[1]	?	$(\mathcal{R},q)$**-AbsDP** $\supset$ $(\mathcal{R},M)$-GcDP [225]
(Θ,ε)-noiseless Pr [46, 116, 164, 391]	**B**	✓[1]	✓[1]	✗[10]	ε-DP $\subset$ (Θ,ε)**-NPr**
(Θ,ε)-causal DP [100]	**B**	✓[1]	✓[1]	✗[10]	(Θ,ε)**-CausDP** $\supset^{\prec}$ ε-DP
(β,ε)-DP under sampling [255]	**Q,B**	✓[1]	✓[1]	✗[10]	(Θ,ε)-NPr $\supset$ (β,ε)**-SamDP** $\supset^{\prec}$ ε-DP
$(\Theta,\varepsilon,\delta)$-active PK DP [36, 46, 100]	**Q,B**	✓[1]	✓[1]	✗[10]	$(\Theta,\varepsilon,\delta)$**-APKDP** $\supset^{\prec}$ (Θ,ε)-NPr
$(\Theta,\varepsilon,\delta)$-passive PK DP [100]	**Q,B**	✓[1]	✓[1]	✗[10]	$(\Theta,\varepsilon,\delta)$-APKDP $\succ$ $(\Theta,\varepsilon,\delta)$**-PPKDP** $\supset^{\prec}$ (Θ,ε)-NPr
$(\Theta,\Phi,\varepsilon)$-pufferfish Pr [206, 219, 228]	**N,B**	✓[1]	✓[1]	✗[10]	(Θ,ε)-NPr $\subset$ $(\Theta,\Phi,\varepsilon)$**-PFPr** $\supset$ $(\mathcal{R},\varepsilon)$-GcDP
(π,k,ε)-Bayesian DP [248]	**N,B**	✓[1]	✓[1]	✗[10]	**BayDP [248]** $\subset$ $(\Theta,\Phi,\varepsilon)$-PFPr
$(\Theta,\varepsilon,\delta)$-distribution Pr [219]	**Q,N,B**	✓[4]	✓[4]	✗[10]	$(\Theta,\varepsilon,\delta)$**-DnPr** $\supset$ $(\Theta,\varepsilon,\delta)$-APKDP
$(\Theta,\varepsilon,\delta)$-profile-based DP [162]	**Q,N,B**	✓[4]	✓[4]	✗[10]	$(\Theta,\varepsilon,\delta)$**-PBDP** $\subset$ $(\Theta,\varepsilon,\delta)$-DnPr
$(\Theta,\varepsilon,\delta)$-probabilistic DnPr [219]	**Q,N,B**	✗[1]	✗[3]	✗[10]	ε-ProDP $\subset$ $(\Theta,\varepsilon,\delta)$**-PDnPr** $\supset$ (Θ,ε)-DnPr
(f,Θ,ε)-divergence DnPr [218]	**Q,N,B**	✓[4]	✓[4]	✗[10]	(f,ε)-DP $\subset$ (f,Θ,ε)**-DDnPr** $\supset$ (Θ,ε)-DnPr
(d,Θ,ε)-extended DnPr [219]	**N,V,B**	✓[4]	✓[4]	✗[10]	$d_{\mathcal{D}}$-Pr $\subset$ (d,Θ,ε)**-EDnPr** $\supset$ (Θ,ε)-DnPr
(d,f,Θ,ε)-ext. div. DnPr [218]	**Q,N,V,B**	✓[4]	✓[4]	✗[10]	(f,Θ,ε)-DDPr $\subset$ (d,f,Θ,ε)**-EDDnPr** $\supset$ (d,Θ,ε)-EDnPr
ε-indistinguishable Pr [260]	**F**	✓[14]	✓[14]	✓[14]	ε**-IndPr** $\sim$ ε-DP

continued

Name & References	Dimensions[18]	Axioms		Cp.[13]	Relations
		P.P.[11]	Cv.[12]		
f-DP [113]	**Q,F**	✓[1]	?	✓[1]	f**-DP** $\supset (\varepsilon,\delta)$-DP
Gaussian DP [113]	**Q,F**	✓[1]	?	✓[1]	**GaussDP** $\subset f$-Pr
(Θ,ε)-positive membership Pr [254]	**B,F**	✓[6]	✓[6]	✗[10]	(Θ,ε)**-PMPr** $\supset \varepsilon$-BoDP
(Θ,ε)-negative membership Pr [254]	**B,F**	✓[6]	✓[6]	✗[10]	(Θ,ε)**-NMPr** $\supset \varepsilon$-BoDP
(Θ,ε)-membership Pr [254]	**B,F**	✓[6]	✓[6]	✗[10]	(Θ,ε)-PMPr $\prec (\Theta,\varepsilon)$**-MPr** $\succ (\Theta,\varepsilon)$-NMPr
$(\Theta,\varepsilon,\delta)$-adversarial Pr [326, 391] also known as information privacy [391]	**Q,B,F**	✓[6]	✓[6]	✗[10]	(ε,δ)-DP $\subset (\Theta,\varepsilon,\delta)$**-AdvPr** $\prec (\Theta,\varepsilon)$-PMPr
(Θ,ε)-aposteriori noiseless Pr [46]	**B,F**	✓[6]	✓[6]	?	(Θ,ε)**-ANPr** $\sim (\Theta,\varepsilon)$-NPr
ε-semantic Pr [158, 217]	**F**	?	?	?	ε**-SemPr** $\sim \varepsilon$-DP
ε-range-bounded Pr [121]	**F**	?	?	?	ε**-RBPr** $\sim \varepsilon$-DP
(Θ,ε)-inference-based causal DP [100]	**B,F**	?	?	?	(Θ,ε)**-IBCDP** $\sim (\Theta,\varepsilon)$-CausDP
(Θ,ε)-information Pr [119]	**N,B,F**	?	?	?	(Θ,ε)**-InfPr** $\succ \varepsilon$-DP
(Agg,ε)-zero-knowledge Pr [161]	**R**	✓[1]	✓[1]	?[16]	(Agg,ε)**-ZKPr** $\succ \varepsilon$-DP
(P,ε,δ)-bounded leakage DP [257]	**Q,R**	✓[1]	✓[1]	✓[8]	(P,ε,δ)**-BLDP** $\supset (\varepsilon,\delta)$-DP
$(\Theta,\Gamma,\varepsilon)$-coupled-worlds Pr [36]	**N,B,R**	✓[1]	✓[1]	✗[4]	$(\Theta,\Gamma,\varepsilon)$**-CWPr** $\supset \varepsilon$-DP
(Θ,ε)-distributional DP [36]	**N,B,R**	✓[1]	✓[1]	✗[4]	$(\Theta,\Gamma,\varepsilon)$-CWPr $\supset (\Theta,\varepsilon)$**-DistDP** $\supset^{\prec} \varepsilon$-DP
$(\Theta,\Gamma,\varepsilon,\delta)$-inference-based CW Pr [36]	**Q,N,B,F,R**	?	?	✗[4]	$(\Theta,\Gamma,\varepsilon,\delta)$**-IBCWPr** $\succ (\Theta,\Gamma,\varepsilon)$-CWPr
$(\Theta,\Gamma,\varepsilon,\delta)$-inference-based DistDP [36]	**Q,N,B,F,R**	?	?	✗[4]	(Θ,ε)-DDP $\succ (\Theta,\Gamma,\varepsilon,\delta)$**-IBDDP** $\subset (\Theta,\Gamma,\varepsilon,\delta)$-IBCWPr

continued

Name & References	Dimensions[18]	Axioms		Cp.[13]	Relations
		P.P.[11]	Cv.[12]		
$(\Theta, \varepsilon, \gamma, \delta)$-typical stability [35]	**Q,V,R**	✓[1]	?	✓[8]	
ε_κ-SIM-computational DP [285]	**C**	✓[7]	✓[7]	✓[17]	ε_κ**-SimCDP** $\prec$ ε-DP
ε_κ-IND-computational DP [285]	**C**	✓[7]	✓[7]	✓[17]	ε_κ**-IndCDP** $\prec$ ε_κ-SimCDP
(ε, δ)-DP for Record Linkage [194]	**C**	✓[7]	✓[7]	✓[17]	(ε, δ)**-RLDP** $\subset$ (ε, δ, f)-OCDP
(ε, δ, f)-output constrained DP [194]	**N,C**	✓[7]	✓[7]	✓[17]	(ε, δ, f)**-OCDP** $\supset$ ε_κ-IndCDP
$(\mathrm{Agg}, \varepsilon)$-computational ZK Pr [161]	**R,C**	✓[7]	✓[7]	?	$(\mathrm{Agg}, \varepsilon)$**-CZKPr** $\supset^{\prec}$ $(\mathrm{Agg}, \varepsilon)$-ZKPr

TABLE 2.20: Summary of variants/extensions of DP representing the main options in each combination of dimensions.

[1] See Proposition 11.

[2] See Proposition 12.

[3] See Proposition 13.

[4] See Proposition 14.

[5] See Proposition 15.

[6] See Proposition 16.

[7] See Proposition 17.

[8] See Proposition 18.

[9] See Proposition 19.

[10] See Proposition 20.

[11] Post-processing

[12] Convexity

[13] Composition

[14] Follows directly from its equivalence with ε-DP.

[15] A modified definition was presented in [228] which is an instance of PF Pr.

[16] A proof for a restricted scenario appears in the paper introducing the definition.

[17] This claim appears in [284], its proof is in the unpublished full version.

[18] Abbreviations used for dimensions:

- **Q**: Quantification of privacy loss
- **N**: Neighborhood definition
- **V**: Variation of privacy loss
- **B**: Background knowledge
- **F**: Formalism of privacy loss
- **R**: Relativization of knowledge gain
- **C**: Computational power

2.2.9.1 *Proofs of properties*

We first list known results for variants and extensions satisfying privacy axioms, prove additional results, then we do the same for composition.

Axioms

Proposition 11.

1. *ProDP, ACoDP, and mCoDP do not satisfy the post-processing axiom [57, 277].*
2. *AbsDP satisfies neither privacy axiom, while GlDP satisfies both [225, 228]*[22].
3. *WBDP satisfies the post-processing axiom [367].*
4. *TypSt satisfies the post-processing axiom [35].*
5. *GaussDP satisfies the post-processing axiom [113].*
6. *PFPr satisfy both privacy axioms [228].*
7. *CWPr satisfy both privacy axioms* [23] *[36].*
8. *APKDP and PPKDP satisfy both privacy axioms [100].*
9. *BLDP satisfies both privacy axioms [257].*

Proposition 12. *All instantiations of DivDP satisfy both privacy axioms. In particular, approximate DP, MIDP, KLPr, RenDP, and zCoDP satisfy both axioms.*

Proof. The post-processing axiom follows directly from the monotonicity property of the f-divergence. The convexity axiom follows directly from the joint convexity property of the f-divergence. □

Proposition 13. *ProDP and ACoDP do not satisfy the convexity axiom.*

Proof. Consider the following mechanisms $\mathcal{M}_1$ and $\mathcal{M}_2$, with input and output in $\{0, 1\}$.

- $\mathcal{M}_1(0) = 0$, $\mathcal{M}_1(1) = 1$ with probability δ, and $\mathcal{M}_1(1) = 0$ with probability $1 - \delta$.
- $\mathcal{M}_2(0) = \mathcal{M}_2(1) = 1$.

Both mechanisms are $\left(\frac{1}{1-\delta}, \delta\right)$-ProDP. Now, consider the mechanism $\mathcal{M}$ which applies $\mathcal{M}_1$ with probability $1 - 2\delta$ and $\mathcal{M}_2$ with probability 2δ. $\mathcal{M}$ is a convex combination of $\mathcal{M}_1$ and $\mathcal{M}_2$, but the reader can verify that it is not $\left(\frac{1}{1-\delta}, \delta\right)$-ProDP. The result for (ξ, ρ, δ)-ACoDP is a direct corollary, since is is equivalent to (ξ, δ)-ProDP when $\rho = 0$. □

Proposition 14. *$d_{\mathfrak{D}}$-Pr satisfies both privacy axioms. Further, EDivDP also satisfies both privacy axioms.*

[22] As an immediate corollary, all definitions which combine notions in **N** and NPr also satisfy both axioms.

[23] As an immediate corollary, DistDP and ZKPr also satisfy both axioms.

Proof. The proof of Proposition 11 for PFPr (Appendix B in [228]) is a proof by case analysis on every possible protected property. The fact that ε is the same for every protected property has no influence on the proof, so we can directly adapt the proof to $d_{\mathcal{D}}$-Pr, and its combination with PFPr. Similarly, the proof can be extended to arbitrary divergence functions, like in Proposition 12. □

Proposition 15. *RanDP does not satisfy the convexity axiom.*

Proof. Let π be the uniform distribution on $\{0,1\}$, let D_1 be generated by picking 10 records according to π, and D_2 by flipping one record at random. Let $\mathcal{M}_0$ return 0 if all records are 0, and $\perp$ otherwise. Let $\mathcal{M}_1$ return 1 if all records are 1, and $\perp$ otherwise.

Note that both mechanisms are $(\pi, 2^{-9}, 0)$-RanDP. Indeed, $\mathcal{M}_0$ will only return 0 for D_1 with probability 2^{-10}, and for D_2 with probability 2^{-10} (if D_1 only has one 1, which happens with probability $10 \cdot 2^{-10}$, and this record is flipped, which happens with probability 0.1). In both cases, $\mathcal{M}_0$ will return $\perp$ for the other database; which will be a distinguishing event. Otherwise, $\mathcal{M}_0$ will return τ for both databases, so $\mathcal{M}(D_1) \approx_0 \mathcal{M}(D_2)$. The reasoning is the same for $\mathcal{M}_1$.

However, the mechanism $\mathcal{M}_{0.5}$ obtained by applying either $\mathcal{M}_0$ or $\mathcal{M}_1$ uniformly randomly does not satisfy $(\pi, 2^{-9}, 0)$-RanDP: the indistinguishability property does not hold if D_1 or D_2 have all their records set to *either* 0 or 1, which happens twice as often as either option alone. □

Proposition 16. *All variants of MPr, AdvPr, and ANPr satisfy both axioms. As a direct corollary, InfPr also satisfies both axioms.*

Proof. We prove it for AdvPr. A mechanism $\mathcal{M}$ satisfies $(\Theta, \varepsilon, \delta)$-AdvPr if for all $t \in \mathcal{T}$, $\theta \in \Theta$, and $S \subseteq \mathcal{O}$, $\mathbb{P}_{D\sim\theta}\left[t \in D \,\middle|\, \mathcal{M}(D) \in S\right] \leq e^{\varepsilon} \cdot \mathbb{P}_{D\sim\theta}\left[t \in D\right] + \delta$. We first prove that it satisfies the convexity axiom. Suppose $\mathcal{M}$ is a convex combination of $\mathcal{M}_1$ and $\mathcal{M}_2$. Simplifying $\mathbb{P}_{D\sim\theta}[\dots]$ into $\mathbb{P}[\dots]$, we have:

$$\mathbb{P}\left[t \in D \,\middle|\, \mathcal{M}(D) \in S\right] = \\ \frac{\mathbb{P}\left[t \in D \text{ and } \mathcal{M}(D) \in S \text{ and } \mathcal{M} = \mathcal{M}_1\right]}{\mathbb{P}\left[\mathcal{M}(D) \in S\right]} + \frac{\mathbb{P}\left[t \in D \text{ and } \mathcal{M}(D) \in S \text{ and } \mathcal{M} = \mathcal{M}_2\right]}{\mathbb{P}\left[\mathcal{M}(D) \in S\right]}$$

Denoting $X_i = \mathbb{P}\left[\mathcal{M}(D) \in S \text{ and } \mathcal{M} = \mathcal{M}_i\right]$ for $i \in \{1,2\}$, this gives:

$$\begin{aligned}\mathbb{P}\left[t \in D \,\middle|\, \mathcal{M}(D) \in S\right] &= \frac{X_1 \cdot \mathbb{P}\left[t \in D \,\middle|\, \mathcal{M}_1(D) \in S\right]}{X_1 + X_2} \frac{X_2 \cdot \mathbb{P}\left[t \in D \,\middle|\, \mathcal{M}_2(D) \in S\right]}{X_1 + X_2} \\ &\leq \frac{X_1\left(e^{\varepsilon} \cdot \mathbb{P}\left[t \in D\right]\right) + \delta}{X_1 + X_2} + \frac{X_2\left(e^{\varepsilon} \cdot \mathbb{P}\left[t \in D\right]\right) + \delta}{X_1 + X_2} \leq e^{\varepsilon} \cdot \mathbb{P}\left[t \in D\right] + \delta.\end{aligned}$$

The proof for the post-processing axiom is similar, summing over all possible outputs $\mathcal{M}(D)$. It is straightforward to adapt the proof to all other definitions which change the shape of the prior-posterior bounds. □

Proposition 17. *Both versions of CDP satisfy both privacy axioms; where the post-processing axiom is modified to only allow post-processing with functions computable on a probabilistic polynomial time Turing machine. As a direct corollary of Proposition 11 for CWPr, CZKPr also satisfies both privacy axioms.*

Proof. For Ind-CDP and the post-processing axiom, the proof is straightforward: if post-processing the output could break the ε-indistinguishability property, the attacker could do this on the original output and break the ε-indistinguishability property of the original definition.

For Ind-CDP and the convexity axiom, without loss of generality, we can assume that the sets of possible outputs of both mechanisms are disjoint (otherwise, this give strictly less information to the attacker). The proof is then the same as for the post-processing axiom.

For SimCDP, applying the same post-processing function to the "true" differentially private mechanism immediately leads to the result, since DP satisfies post-processing. The same reasoning holds for convexity. □

Composition

In this section, if $\mathcal{M}_1$ and $\mathcal{M}_2$ are two mechanisms, we denote $\mathcal{M}_{1+2}$ the mechanism defined by $\mathcal{M}_{1+2}(D) = (\mathcal{M}_1(D), \mathcal{M}_2(D))$.

Proposition 18 (Existing results)**.** *If $\mathcal{M}_1$ and $\mathcal{M}_2$ are respectively…*

1. *$(\varepsilon_1, \delta_1)$-DP and $(\varepsilon_2, \delta_2)$-DP, then $\mathcal{M}_{1+2}$ is $(\varepsilon_1 + \varepsilon_2, \delta_1 + \delta_2)$-DP [126].*
2. *$(\varepsilon_1, \delta_1)$-ProDP and $(\varepsilon_2, \delta_2)$-ProDP, then $\mathcal{M}_{1+2}$ is $(\varepsilon_1 + \varepsilon_2, \delta_1 + \delta_2)$-ProDP [60, 267, 277].*
3. *$\left(\varepsilon_1, \delta_{a,1}, \delta_{p,1}\right)$-RelDP and $\left(\varepsilon_2, \delta_{a,2}, \delta_{p,2}\right)$-RelDP, then $\mathcal{M}_{1+2}$ is $\left(\varepsilon_1 + \varepsilon_2, \delta_{a,1} + \delta_{a,2}, \delta_{p,1} + \delta_{p,2}\right)$-RelDP [407].*
4. *ε_1-MIDP and ε_2-MIDP, then $\mathcal{M}_{1+2}$ is $(\varepsilon_1 + \varepsilon_2)$-MIDP [88].*
5. *ε_1-KLDP and ε_2-KLDP, then $\mathcal{M}_{1+2}$ is $(\varepsilon_1 + \varepsilon_2)$-KLDP [31, 88].*
6. *$(\alpha_1, \varepsilon_1)$-RenDP and $(\alpha_2, \varepsilon_2)$-RenDP, then $\mathcal{M}_{1+2}$ is $(\max(\alpha_1, \alpha_2), \varepsilon_1 + \varepsilon_2)$-RenyiDP [284].*
7. *(μ_1, τ_1)-mCoDP and (μ_2, τ_2)-mCoDP, then $\mathcal{M}_{1+2}$ is $\left(\mu_1 + \mu_2, \sqrt{{\mu_1}^2 + {\mu_2}^2}\right)$-mCoDP [57, 132].*
8. *(ξ_1, ρ_1)-zCoDP and (ξ_2, ρ_2)-zCoDP, then $\mathcal{M}_{1+2}$ is $(\xi_1 + \xi_2, \xi_1 + \xi_2)$-zCoDP [57].*
9. *$(\xi_1, \rho_1, \delta_1)$-ACoDP and $(\xi_2, \rho_2, \delta_2)$-ACoDP, then $\mathcal{M}_{1+2}$ is $(\xi_1 + \xi_2, \xi_1 + \xi_2, \delta_1 + \delta_2 - \delta_1\delta_2)$-ACoDP [57].*
10. *$(\xi_1, \rho_1, \omega_1)$-bCoDP and $(\xi_2, \rho_2, \omega_2)$-bCoDP, then $\mathcal{M}_{1+2}$ is $(\xi_1 + \xi_2, \xi_1 + \xi_2, \min(\omega_1, \omega_2))$-bCoDP [57].*
11. *(η_1, τ_1)-CCoDP and (η_2, τ_2)-CCoDP, then $\mathcal{M}_{1+2}$ is $\left(\eta_1 + \eta_2, \frac{\eta_1 + {\eta_2}^2}{2}\right)$-CCoDP [81].*

12. *f_1-DP and f_2-DP, than then $\mathcal{M}_{1+2}$ is $f_1 \otimes f_2$-DP [113].*

13. *$(\pi, \gamma_1, \varepsilon_1)$-RanDP and $(\pi, \gamma_2, \varepsilon_2)$-RanDP, then $\mathcal{M}_{1+2}$ is $(\pi, \gamma_1 + \gamma_2, \varepsilon_1 + \varepsilon_2)$-RanDP [184].*

14. *$(\pi, \varepsilon_1, \delta_1)$-BayDP [367] and $(\pi, \varepsilon_1, \delta_1)$-BayDP [367], then $\mathcal{M}_{1+2}$ is $(\pi, \varepsilon_1 + \varepsilon_2, \delta_1 + \delta_2)$-BayDP [367]. The same result holds for WBDP as an immediate consequence of Theorem 1 in [367].*

15. *$(\Theta, \varepsilon_1, \gamma_1, \delta_1)$-TypSt and $(\Theta, \varepsilon_2, \gamma_2, \delta_2)$-TypSt, then $\mathcal{M}_{1+2}$ is $(\Theta, \varepsilon_1 + \varepsilon_2, \gamma_1 + \gamma_2, \delta_1 + \delta_2)$-TypSt [35].*

16. *$(d_{\mathfrak{D}}, \varepsilon_1, \delta_1)$-PsDP and $(d_{\mathfrak{D}}, \varepsilon_2, \delta_2)$-PsDP then $\mathcal{M}_{1+2}$ is $(d_{\mathfrak{D}}, \varepsilon_1 + \varepsilon_2, \delta_1 + \delta_2)$-PsM DP [106].*

17. *$(P_1, \varepsilon_1, \delta_1)$-BLDP and $(P_2, \varepsilon_2, \delta_2)$-BLDP, then $\mathcal{M}1 + 2$ is $(P_{1+2}, \varepsilon_1 + \varepsilon_2, \delta_1 + \delta_2)$-BLDP, where P_{1+2} is the concatenation of P_1 and P_2 (where the randomness is not shared between P_1 and P_2, nor between $\mathcal{M}_1$ and $\mathcal{M}_2$) [257].*

Proposition 19. *If $\mathcal{M}_1$ is $d^1_{\mathfrak{D}}$-private and $\mathcal{M}_2$ is $d^2_{\mathfrak{D}}$-private, then $\mathcal{M}_{1+2}$ is $d^{1+2}_{\mathfrak{D}}$-private, where $d^{1+2}_{\mathfrak{D}}(D_1, D_2) = d^1_{\mathfrak{D}}(D_1, D_2) + d^2_{\mathfrak{D}}(D_1, D_2)$.*

Proof. The proof is essentially the same as for ε-DP. $\mathcal{M}_1$'s randomness is independent from $\mathcal{M}_2$'s, so:

$$\begin{aligned}
&\mathbb{P}\left[\mathcal{M}_1(D_1) = O_1 \text{ and } \mathcal{M}_2(D_1) = O_2\right] \\
&\quad = \mathbb{P}\left[\mathcal{M}_1(D_1) = O_1\right] \cdot \mathbb{P}\left[\mathcal{M}_2(D_1) = O_2\right] \\
&\quad \leq e^{d^1_{\mathfrak{D}}(D_1,D_2)} \cdot \mathbb{P}\left[\mathcal{M}_2(D_2) = O_1\right] \cdot e^{d^2_{\mathfrak{D}}(D_1,D_2)} \cdot \mathbb{P}\left[\mathcal{M}_2(D_2) = O_2\right] \\
&\leq e^{d^{1+2}_{\mathfrak{D}}(D_1,D_2)} \cdot \mathbb{P}\left[\mathcal{M}_1(D_2) = O_1 \text{ and } \mathcal{M}_2(D_2) = O_2\right].
\end{aligned}$$

Each definition listed in Proposition 18 can also be combined with $d_{\mathfrak{D}}$-privacy, and the composition proofs can be similarly adapted. □

Proposition 20. *In general, definitions which assume limited background knowledge from the adversary do not compose.*

Proof. The proof of Proposition 19 cannot be adapted to a context in which the attacker has limited background knowledge: as the randomness partially comes from the data-generating distribution, the two probabilities are no longer independent. A typical example considers two mechanisms which answer e.g., queries "how many records satisfy property P" and "how many records satisfy property P and have an ID different from 4217": the randomness in the data might make each query private, but the combination of two queries trivially reveals something about a particular user. Variants of this proof can easily be obtained for all definitions with limited background knowledge. □

2.2.10 *Related work*

In this section, we detail our criteria for excluding particular data privacy definitions from our work, we list some relevant definitions that were excluded by the criteria presented in Section 2.2.1.4, and we list related works and existing surveys in the field of data privacy.

2.2.10.1 *Out of scope definitions*

As detailed in Section 2.2.1.4, we considered certain data privacy definitions to be out of scope for our work, even when they seem to be related to differential privacy. This section elaborates on such definitions.

Lack of semantic guarantees

Some definitions do not provide clear semantic privacy guarantees, or are only used as a tool in order to prove links between existing definitions. As such, we did not include them in our survey.

- *ε-privacy*, introduced in [264], was a first attempt at formalizing an adversary with restricted background knowledge. Its formulation does not provide a semantic guarantee, and it was superseded by noiseless privacy [46, 116] (introduced in Section 2.2.5).

- *Relaxed indistinguishability*, introduced in [326] is a relaxation of adversarial privacy that provides a plausible deniability by requiring for each tuple, that at least l tuples must exist with ε-indistinguishability. It does not provide any guarantee against Bayesian adversaries.

- *Differential identifiability*, introduced in [244], bounds the probability that a given individual's information is included in the input datasets but does not measure the *change* in probabilities between the two alternatives. As such, it does not provide any guarantee against Bayesian adversaries[24].

- *Crowd-blending privacy*, introduced in [160], combines differential privacy with k-anonymity. As it is strictly weaker than any mechanism which always returns a k-anonymous dataset, the guarantees it provides against a Bayesian adversary are unclear. It is mainly used to show that combining crowd-blending privacy with pre-sampling implies zero-knowledge privacy [160, 263].

- *Membership privacy*[25], introduced in [336], is tailored to membership inference attacks on machine learning models; the guarantees it provides are not clear.

- (k,ε)*-anonymity*, introduced in [197], first performs *k-anonymisation* on a subset of the quasi identifiers and then ε-DP on the remaining quasi-identifiers with different settings for each equivalence class of the *k-anonymous* dataset. The semantic guarantees of this definition are not made explicit.

[24] *Differential identifiability* was reformulated in [254] as an instance of membership privacy.

[25] Another definition with the same name is introduced in [254], we mention it in Section 2.2.6

- *Posteriori DP*, introduced in [378], compares two posteriors in a way similar to inferential privacy, but does not make the prior (and thus, the attacker model) explicit.
- *Noiseless privacy*[26], introduced in [147], limits the change in the number of possible outputs when one record in the dataset changes. As it does not bound the change in *probabilities* of the mechanism, it does not seem to offer clear guarantees against a Bayesian adversary.
- *Weak DP*, introduced in [382], adapts DP for streams, but it only provides a DP guarantee for the *average* of all possible mechanism outputs[27], rather than for the mechanism itself. Thus, its semantics guarantees are also unclear.
- *Error Preserving Privacy*, introduced in [90], states that the *variance* of the adversary's error when trying to guess a given user's record does not change significantly after accessing the output of the mechanism. The exact adversary model is not specified.

Variants of sensitivity

A important technical tool used when designing differentially private mechanisms is the *sensitivity* of the function that we try to compute. There are many variants to the initial concept of global sensitivity [128], including local sensitivity [303], smooth sensitivity [303], restricted sensitivity [52], empirical sensitivity [74], empirical differential privacy[28] [29] [5], recommendation-aware sensitivity [410], record and correlated sensitivity [411], dependence sensitivity [258], per-instance sensitivity [379], individual sensitivity [89], elastic sensitivity [209] and derivative sensitivity [238]. These notions only change *how* to achieve a given privacy definition (typically DP), and are not relevant to the definition itself, so we did not consider these notions in our work.

2.2.10.2 *Local model and other contexts*

In this work we focused on DP variants/extensions typically used in the *global model*, in which a central entity has access to the whole dataset. It is also possible to use DP in other contexts, without formally changing the definition. The main alternative is the *local model*, where each individual randomizes their own data before sending it to an aggregator. This model, formally introduced in [117], is used e.g., by Google [141], Apple [360], or Microsoft [107]. These models can be thought of as different ways of deploying a given privacy definition, rather than distinct definitions.

Many definitions we listed were initially presented in the local model, such as $d_{\mathfrak{D}}$-privacy [67], geo-indistinguishability [15], earth mover's Pr [151], location Pr [138], profile-based DP [162], divergence DP and smooth DP from [31], and extended DP, distribution Pr, and extended distribution Pr from [219].

Below, we list the definitions that are the same as previously listed definitions, but used in a different attacker setting; the list also includes alternatives to the local and global models.

[26] Another definition with the same name is introduced in [46, 116], we mention it in Section 2.2.5.

[27] It also assumes that some uncertainty comes from the data itself, similarly to definitions in Section 2.2.6

[28] Even though it is introduced as a variant of DP, it was later shown to be a measure of sensitivity [64].

[29] Another definition with the same name is introduced in [58], we mention it in Section 2.2.5.

- In [344], the authors introduce *distributed DP*, which corresponds to local DP, with the additional assumption that only a portion of participants are honest.
- In [220], the authors define *joint DP*, to model a game in which each player cannot learn the data from any other player, but are still allowed to observe the influence of their data on the mechanism output. In [390], authors define a slightly different version of this idea, *multiparty DP*, in which the view of each *subgroup* of players is differentially private in respect to other players inputs.
- In [50], the authors define *DP in the shuffled model*, which falls in-between the global and the local model: the local model is augmented by an anonymous channel that randomly permutes a set of user-supplied messages, and differential privacy is only required of the output of the shuffler.
- In [207], the authors define *localized information privacy*, a local version of information privacy (mentioned in Section 2.2.6).
- In [289], the authors define *utility-optimized local DP*, a local version of one-sided differential privacy (mentioned in Section 2.2.3) which additionally guarantees that if the data is considered sensitive, then a certain set of outputs is forbidden.
- In [6, 112, 301], the authors define *personalized local DP*, a local version of personalized DP (mentioned in Section 2.2.4).
- In [12], the authors define *$d_{\mathfrak{D}}$-local DP*, a local version of $d_{\mathfrak{D}}$-DP (mentioned in Section 2.2.4); this was redefined as *condensed local DP* in [180].
- In [250], the authors define *task-global DP* and *task-local DP*, which are equivalents of element-level DP (mentioned in Section 2.2.3) in a meta-learning context.

2.2.10.3 *Related work*

The relation between the main syntactic models of anonymity and DP was studied in [78], in which the authors claim that the former is designed for privacy-preserving data publishing (PPDP), while DP is more suitable for privacy preserving data mining (PPDM). We disagree with this assessment, and discuss differentially private data publishing at length in Chapter 4.

In [196], the authors classify different privacy enhancing technologies (PETs) into 7 complementary dimensions. Indistinguishability falls into the *Aim* dimension, but within this category, only *k*-anonymity and oblivious transfer are considered; differential privacy is not mentioned. In [7], the authors survey privacy concerns, measurements and privacy-preserving techniques used in online social networks and recommender systems. They classify privacy into 5 categories; DP falls into *Privacy-preserving models* along with e.g., *k*-anonymity. In [376] the authors classified 80+ privacy metrics into 8 categories based on the output of the privacy mechanism. One of their classes is *Indistinguishability*, which contains DP as well as several variants. Some variants are classified into other categories; for example Rényi DP is classified into *Uncertainty* and mutual-information DP into *Information gain/loss*. The authors list 8 differential privacy variants; our taxonomy can be seen as an extension of the contents of their work (and in particular of the *Indistinguishability* category).

In [378], authors establish connections between differential privacy (seen as the additional disclosure of an individual's information due to the release of the data), *identifiability* (seen as the posteriors of recovering the original data from the released data), and *mutual-information privacy* (which measures the average amount of information about the original dataset contained in the released data).

The appropriate selection of the privacy parameters for DP was also exhaustively studied. This problem in not trivial, and many factors can be considered: in [200], the authors used economic incentives, in [234, 243, 312], the authors looked at individual preferences, and in [237, 259], the authors took into account an adversary's capability in terms of hypothesis testing and guessing advantage respectively.

Earliest surveys focusing on DP summarize algorithms achieving DP and applications [123, 124]. The more detailed "privacy book" [131] presents an in-depth discussion about the fundamentals of DP, techniques for achieving it, and applications to query-release mechanisms, distributed computations or data streams. Other textbooks have focused on empirical performance of various algorithms [252], asymptotic upper and lower bounds for various tasks [372], or have tried to make differential privacy more approachable to non-experts [304]. Other surveys focus on the release of histograms and synthetic data with DP [189, 296].

Finally, some surveys focus on location privacy. In [266], the authors highlight privacy concerns in this context and list mechanisms with formal provable privacy guarantees; they describe several variants of differential privacy for streaming (e.g., pan-privacy) and location data (e.g., geo-indistinguishability) along with extensions such as pufferfish and blowfish privacy. In [68], the authors analyze different kinds of privacy breaches and compare metrics that have been proposed to protect location data.

2.3 CONCLUSION

In this chapter, we first listed four syntactic privacy definitions; each one a different attempt at defining anonymization based on properties of the dataset. These definitions demonstrated the various attack models that are relevant in different contexts, and the difficulty of finding a satisfying notion of anonymity. We then introduced differential privacy, a definition which changes the conceptual perspective: rather than seeing anonymization as a property of a database, it considers it as a property of the process. Thanks to this view, we can now make strong statements about the information that an attacker can get about an individual, bypassing the need for error-prone attack modeling. Differential property also comes with useful properties, such as post-processing and composition, which makes it easier for researchers and engineers to build mechanisms that satisfy it.

Differential privacy itself has been extremely successful, and the intuition behind it inspired hundreds of alternative definitions. But this also has created a situation where a newcomer to the field of anonymization can have trouble navigating it, understanding how different notions relate to each other, and choosing which notion is most appropriate for their use case. To solve this issue, we proposed a classification of DP variants and extensions using the concept of dimensions. When possible, we compared definitions from the same dimension, and we showed that definitions from the different dimensions can be combined to form new, meaningful definitions. In theory, it means that even if there were only three possible ways to change a dimension (e.g., making it weaker or stronger), this would result in $3^7 = 2187$ possible definitions: the ≈ 225 existing definitions shown in Figure 2.7 are only scratching the surface of the space of possible notions. Using these dimensions, we unified and simplified the different notions proposed in the literature. We highlighted their properties such as composability and whether they satisfy the privacy axioms by either collecting the existing results or creating new proofs, and whenever possible, we showed their relative relations to one another. We hope that this work will make the field of data privacy more organized and easier to navigate, especially for new practitioners.

3

DIFFERENTIAL PRIVACY UNDER PARTIAL KNOWLEDGE

> *However, just because something isn't surprising doesn't mean it's easy to deal with.*
>
> — Nnedi Okorafor, *Binti*

The success of differential privacy, and the many variants and extensions proposed since its introduction, suggests a solution to our initial problem. To pick a definition of privacy that makes sense in a certain context, we can start from differential privacy, and optionally modify it to better fit our use case.

If we want to modify the protected characteristic, we can change the definition of neighborhood (Section 2.2.3). If is too expensive to consider the absolute worst-case output, we can relax this by averaging the privacy loss, or accepting a very small risk (Section 2.2.2). If different people in our dataset have distinct privacy requirements, we can also model this in the definition (Section 2.2.4). In all the aforementioned cases, we can pick a variant that satisfies useful properties like post-processing or composition, and still keep the strong formal guarantees that differential privacy offers.

From a purely academic standpoint, it seems like this definitional problem is now solved. Unfortunately, this are a bit more complicated in practice. In real-world scenarios, people trying to advocate for formal privacy guarantees often encounter pushback. "Why is it necessary to opt for such a strong notion?", a skeptic might say. "This data is heavily aggregated, surely the risk is low enough. It seems fine to me. Can you find a realistic way of attacking it? Maybe then I will be convinced that stronger guarantees are necessary."

To advocate for using differential privacy, we can try explaining what happens if the definition is *not* satisfied. DP states that changing the data of a single user cannot be detected by looking at the mechanism's output. So by definition, a mechanism that is *not* differentially private allows the above situation to happen. There exist two datasets D_1 and D_2 that differ in the data of a single user, such that the difference between $\mathcal{M}(D_1)$ and $\mathcal{M}(D_2)$ is arbitrarily large, at least for some outputs. This seems bad!

However, the "attack" above is not convincing to laypeople: it feels too artificial. Surely, they say, a realistic attacker cannot craft an arbitrary pair of datasets D_1 and D_2 and observe the results. They do not control the data, nor do they have perfect knowledge about it. If the data is heavily aggregated, for example using k-anonymity with a reasonably large k, can an attacker really find out sensitive information about individuals?

In practice, aggregated data is frequently being published without noise, under the assumption that this is safe enough. The most prominent example is probably *voting*: it would be unacceptable to add noise to the tally of a democratic election, yet nobody seems to worry that publishing the exact count might somehow constitute a privacy risk. The intuition behind this hinges on the implicit assumption that the secrecy of votes makes the result *uncertain*, from the point of view of any realistic attacker.

Can this intuition be formalized? We saw in Section 2.2.5 that this idea is not new, and that multiple variants of differential privacy were introduced to capture an attacker with uncertainty over the input data. In this chapter, we focus on this specific setting. We show that modeling this scenario correctly is much more difficult than it initially appears, but that a careful analysis can formalize the above intuition, draw novel links between syntactic definitions like *k*-anonymity and differential privacy, and be used to derive negative results on problems that are impossible to solve in a privacy-preserving way.

On the theory front, we start by reviewing existing definitions formalizing the scenario where the attacker only has partial knowledge over the data. We point out fundamental flaws in existing notions in the case where there are correlations in the data, and introduce an additional criterion that is needed to fix these definitional problems. We then delineate two cases that behave differently in the presence of partial knowledge: whether the attacker can *influence* the input dataset, or whether they have a *passive* partial knowledge about the original data.

We then use this novel theory to formalize positive results in our setting of interest. We show that aggregating the data of many users does provide privacy guarantees under the partial knowledge assumption, by drawing links between this setting and amplification by shuffling. We also show that *thresholding* is an effective privacy-preserving strategy against passive attackers, and use this insight to derive new links between *k*-anonymity and differential privacy.

Finally, we show that even under this assumption of partial knowledge, some use cases cannot be solved in a privacy-preserving way. In particular, we study *cardinality estimators*, data structures that can estimate the cardinality of large datasets, and that can be *merged* while removing duplicates: this class of algorithms cannot be made private, even under extremely weak assumptions.

3.1 DEFINITIONAL INTRICACIES OF PARTIAL KNOWLEDGE

As we saw previously, differential privacy models strong attackers that can not only learn, but also influence, all but one element from the input dataset. While this strong attacker model over-approximates realistic attackers, it can also lead to overly cautious choices of noise parameters, unnecessarily deteriorating the algorithms' accuracy. Relaxing the assumptions about attackers' background knowledge and their influence on the dataset can lead to smaller noise parameters and, in turn, to more accurate results.

In some scenarios [373], one cannot exclude that the attacker might be able to influence the entire dataset. But there are many natural scenarios in which the risk of an attacker injecting a large number of data points into the dataset is negligible: censuses, phone polls, or elections are natural examples[1]. In those cases, it is appropriate to consider privacy guarantees that model weaker attackers without influence over the dataset, but with some background knowledge. While the precise estimation of an attacker's capabilities may be difficult, a

[1] This implicitly assumes the absence of malicious insiders with direct access to the collected data, but those could break privacy anyway by leaking the entire dataset, so anonymization becomes irrelevant.

more balanced privacy analysis should characterize the privacy leakage against attackers with varying degrees of background knowledge and influence over the dataset.

As we show in this section, while there is a rich body of prior work on differential privacy with partial knowledge [36, 46, 116, 228, 254, 326], it fails to account for data with correlations, and it does not make the attacker model explicit and precise. In this section, we provide a theoretical foundation for answering these questions. Our formalism solves the problems that previous definitions encounter when the data contains correlations, and it clearly delineates between attackers that only have some background knowledge and attackers that can influence the data. Our main contributions are as follows.

First, we show that existing notions of privacy under partial knowledge break down when the data contains correlations, allowing very revealing mechanisms to be mistakenly considered private. We propose a practical criterion and approach to fix this class of issues.

Second, we show that there are two distinct ways to model partial knowledge, depending on whether the attacker can only learn some properties of the data or can modify the data. We define two corresponding notions of privacy under partial knowledge: active partial knowledge, where the attacker can influence the dataset, and passive, where the attacker is unable to influence the dataset. We show that these notions have natural properties, and prove that they are equivalent for a large class of common mechanisms and assumptions over the data. Moreover, we show that the active partial knowledge assumption can be used to alleviate the challenge of precisely estimating the dataset distribution.

3.1.1 *Related work*

Among the numerous variants of differential privacy listed in Chapter 2, two main variants model adversaries with partial background knowledge, using indistinguishability: *noiseless privacy* [46, 116], and *distributional DP* [36]. This work discusses the shortcomings of distributional DP in Section 3.1.2, and Section 3.1.3 uses the formalism of noiseless privacy to define active and passive partial knowledge DP.

Other variants, which also model adversaries with partial background knowledge, are not based on indistinguishability, but directly constrain the posterior knowledge of an attacker as a function of their prior knowledge. Among those are *adversarial privacy* [326], *membership privacy* [254], and *aposteriori noiseless privacy* [46]. It is straightforwards to adapt the examples given in this chapter to show that these definitions suffer from the same flaws as noiseless privacy when data has correlations. Because of space constraints, we do not study them in detail.

Several other definitions have been proposed. *Pufferfish privacy* [228] can be seen as a generalization of noiseless privacy, and similarly, *coupled-worlds privacy* [36] (and its *inference-based* variant) generalizes distributional differential privacy: instead of protecting individual tuples, they protect arbitrary sensitive properties of the data. It is straightforward to generalize our results to the more generic frameworks.

3.1.2 *Correlated data*

When data is correlated, dependencies create problems for privacy definitions that assume an attacker with partial knowledge. To illustrate this, we recall two previously introduced definitions that model this situation differently: *noiseless privacy* (NPr) and *distributional differential privacy* (DistDP). We show that both definitions have undesirable consequences when data is correlated, and use DistDP as a starting point to solve this problem in two steps. First, we modify DistDP and introduce a new definition, *causal differential privacy* (CausDP), to prevent its most direct problems. Second, we propose a criterion that encompasses many use-cases but avoids known issues with correlated data, and makes CausDP equivalent to NPr. This allows us to cleanly define a rigorous notion of DP with partial knowledge, which allows for many practical use cases, but avoids the known issues with correlated data.

Note that there has been substantial debates about the impact of correlations on the guarantees that DP provides. The debates are summarized in [368], where the authors suggest a possible resolution: interpreting DP as a *causal* property. In this section, we continue this line of work in the context of partial knowledge. In particular, we show that modifying the definition in the same way as the "causal variants" of [368] is not sufficient to solve all issues created by the presence of correlations in the data, when the attacker only has partial knowledge.

For simplicity, in this section, we only consider the case with $\delta = 0$. We re-introduce $\delta > 0$ in Section 3.1.3.

3.1.2.1 *Existing notions*

The assumption that the attacker lacks background knowledge can be represented by considering the input data to be *noisy*. This idea was first proposed in [116], and was formalized in [46] as *noiseless privacy*. Instead of comparing two databases that differ in only one record, it uses a *probability* distribution θ, conditioned on the value of one record $D(i)$: the randomness in θ captures the attacker's uncertainty. This probability distribution generates not only a dataset D but also the attacker's partial knowledge B (with values in some space $\mathcal{B}$). For brevity, we abbreviate $\mathbb{P}_{(D,B)\sim\theta}$ as $\mathbb{P}_\theta$, abbreviate the observation $D = \hat{D}$ by $\hat{D}$, and the observation $B = \hat{B}$ by $\hat{B}$; these notations as well as others used in this paper are summarized in the notation table on page xx.

Definition 49 ((Θ, ε)-noiseless privacy [46])**.** *Given a family Θ of probability distribution on $\mathcal{D}\times\mathcal{B}$, a mechanism $\mathcal{M}$ is* (Θ, ε)-noiseless private *(NPr) if for all $\theta \in \Theta$, all $\hat{B} \in \mathcal{B}$, all indices i and all $t, t' \in \mathcal{T}$ such that $\mathbb{P}_\theta\left[\hat{B}, D(i) = t\right] \neq 0$ and $\mathbb{P}_\theta\left[\hat{B}, D(i) = t'\right] \neq 0$ (we call this condition "$\hat{B}$ is* compatible with *$D(i) = t$ and $D(i) = t'$"):*

$$\mathcal{M}(D)_{|\theta,\hat{B},D(i)=t} \approx_\varepsilon \mathcal{M}(D)_{|\theta,\hat{B},D(i)=t'} .$$

Here, the notation $\mathcal{M}(D)_{|\theta,\hat{B},D(i)=t}$ refers to the random variable defined by $\mathcal{M}(D)$, where $D \sim \theta$, conditioned on the event "$B = \hat{B}$ and $D(i) = t$".

Note even though we make the attacker's partial knowledge $\hat{B}$ explicit, we could simplify the definition to not mention it explicitly. Indeed, for each distribution $\theta \in \Theta$ with values in $\mathcal{D}\times\mathcal{B}$, we could simply take all possible background knowledges $\hat{B}$ generated by θ, and consider the family of distributions $\theta_{|\hat{B}}$, which only generate a dataset D. We use this conceptual simplification in Definition 33 and Definition 34, in Section 2.2.5. This simplification is no longer possible if we use (ε, δ)-indistinguishability with $\delta > 0$, as we show in Section 3.1.3.

The original intuition behind DP states that changing one record must not change the output too much. NPr attempts to capture this intuition for an attacker with partial knowledge, but Bassily et al. [36] argue that this definition is too strong. The following example illustrates their argument.

Example 1. *Assume θ has a global parameter μ that is either +1 or -1 with equal probabilities, and outputs n normally distributed records with mean μ and a small standard deviation. Releasing the average of the record values is not NPr: for all indices i, $\mathcal{M}(D)_{|\theta,D(i)=1}$ will be close to* 1 *and $\mathcal{M}(D)_{|\theta,D(i)=-1}$ will be close to* -1*, so the two distributions are very distinguishable. This happens even though the impact of a single record in the database is low: once μ is fixed, the random choice of i is unlikely to have a large effect on the global average.*

This example shows a definitional problem. If the attacker previously knows μ, revealing $\mathcal{M}(D)$ does not give much additional information on the target $D(i)$: an attacker with *less* initial knowledge is considered *more* powerful. We study this in detail in Section 3.1.2.3.

The authors propose an alternative definition to fix this problem: (Θ, ε)*-distributional differential privacy*. It requires that $\mathcal{M}$ can be *simulated* by another mechanism Sim, that does not have access to the sensitive property. The intuition is as follows: if $\mathcal{M}(D)$ is close to $\text{Sim}(D_{-i})$ for some simulator Sim, then $\mathcal{M}$ cannot leak "too much" about the value of $D(i)$.

Definition 50 ((Θ, ε)-distributional differential privacy [36]). *Given a family Θ of probability distributions on $\mathcal{D} \times \mathcal{B}$, a mechanism $\mathcal{M}$ satisfies (Θ, ε)*-distributional differential privacy *((Θ, ε)-DistDP) if there is a simulator* Sim *such as for all probability distributions $\theta \in \Theta$, all $\hat{B} \in \mathcal{B}$, all i, and all $t \in \mathcal{T}$ such that $\hat{B}$ is compatible with $D(i) = t$:*

$$\mathcal{M}(D)_{|\theta,\hat{B},D(i)=t} \approx_{\varepsilon} \text{Sim}(D_{-i})_{|\theta,\hat{B},D(i)=t},$$

where D_{-i} is the database D from which the record i has been removed.

The distribution θ and the mechanism $\mathcal{M} = \text{avg}$ from Example 1 satisfy this definition: the simulator can be defined as simply running $\mathcal{M}$ on D_{-i}, possibly after adding $+1$ or -1 depending on the other records.

3.1.2.2 *Distributional differential privacy under correlations*

This critique of NPr is similar to the critique of the associative view of DP in [368]. But the proposed fix has a flaw: Sim can use strong dependencies in the data to artificially satisfy the definition. In the following example, the values of different records are strongly correlated, and the simulator *cheats* by using these correlations: consequently, the identity function is considered private!

Example 2. *Let θ output n* duplicate *records: for all $i < n$, $D(2i)$ is picked from some probability distribution R, and $D(2i+1) = D(2i)$. Then the identity function Id, which simply outputs its input without any noise, is $(\{\theta\}, 0)$-DistDP! Indeed, the simulator can simply replace the missing record by its duplicate and output the entire database: $Id(D)$ is exactly the same as* $\operatorname{Sim}(D_{-i})$.

Here, the dependency relationships are "extreme", as each record is duplicated. But even when records are less strongly correlated, the problem is still present. In fact, the more dependencies are in the data, the more accurately the simulator can simulate the missing record, and the more "private" the mechanism is (since ε gets lower): a more powerful adversary, who can exploit dependencies in the data, is considered weaker by the definition. This is clearly undesirable.

How can we formalize an adversary that cannot "cheat" using dependencies in the data? We propose one possible option: using the same technique as the causal variants of DP described in [368], we simply change the target record *after* the distribution is generated.

Definition 51 ((Θ, ε)-causal differential privacy)**.** *Given a family Θ of probability distributions on $\mathcal{D} \times \mathcal{B}$, a mechanism $\mathcal{M}$ satisfies* (Θ, ε)-causal differential privacy *((Θ, ε)-CausDP) if for all probability distributions $\theta \in \Theta$, all i, all $t, t' \in \mathcal{T}$, and all $\hat{B} \in \mathcal{B}$ compatible with $D(i) = t$ and $D(i) = t'$:*

$$\mathcal{M}(D)_{|\theta,\hat{B},D(i)=t} \approx_\varepsilon \mathcal{M}(D_{i\to t'})_{|\theta,\hat{B},D(i)=t},$$

where $D_{i\to t'}$ is the database D, where the i-th record has been replaced by t'.

CausDP still captures DP's intuition: the change in one data point should not influence the output of the mechanism too much. However, the change happens *after* the influence of the dependencies in the data. This version is strictly stronger than the original version: if a mechanism $\mathcal{M}$ is (Θ, ε)-CausDP, then it is also (Θ, ε)-DistDP. Indeed, the simulator Sim can always replace the missing record with an arbitrary value b and return $\mathcal{M}(D_{i\to t'})$.

In [36], the authors also introduce an *inference-based* version of DistDP. We can as easily adapt CausDP to this different formalization.

Definition 52 ((Θ, ε)-inference-based causal differential privacy)**.** *Given a family Θ of probability distributions on $\mathcal{D} \times \mathcal{B}$, a mechanism $\mathcal{M}$ satisfies* (Θ, ε)-inference-based causal differential privacy *((Θ, ε)-IBCDP) if for all probability distributions $\theta \in \Theta$, for all indices i, all $b \in \mathcal{T}$, all $t \in \mathcal{O}$, and all $\hat{B} \in \mathcal{B}$ compatible with $\mathcal{M}(D) = t$ and $\mathcal{M}(D_{i\to t'}) = t$:*

$$D(i)_{|\theta,\hat{B},\mathcal{M}(D)=t} \approx_\varepsilon D(i)_{|\theta,\hat{B},\mathcal{M}(D_{i\to t'})=t}$$

where $D_{i\to t'}$ is the database D, where the record i has been replaced by b.

Note that this definition is equivalent to the indistinguishability-based version (Definition 51) up to a change in parameters.

Proposition 21. *(Θ, ε)-CausDP implies $(\Theta, 2\varepsilon)$-IBCDP, and (Θ, ε)-IBCDP implies $(\Theta, 2\varepsilon)$-CausDP.*

Proof. The first implication can be proven in the same way as Theorem 1 in [36], replacing $\mathrm{Sim}\left(D_{-i}\right)$ by $\mathcal{M}\left(D_{i\to t'}\right)$. For the second implication, suppose that a mechanism $\mathcal{M}$ is (Θ,ε)-IBCDP, and assume that the attacker has no background knowledge. Consider an index i, two possible record values $t,t'\in\mathcal{T}$, and one possible output value $O\in\mathcal{O}$. Bayes' rule gives us:

$$\frac{\mathbb{P}\left[\mathcal{M}\left(D_{i\to t'}\right)=O|D(i)=t\right]}{\mathbb{P}\left[\mathcal{M}\left(D\right)=O|D(i)=t\right]}=\frac{\mathbb{P}\left[D(i)=t|\mathcal{M}\left(D_{i\to t'}\right)=O\right]}{\mathbb{P}\left[D(i)=t|\mathcal{M}\left(D\right)=O\right]}\cdot\frac{\mathbb{P}\left[\mathcal{M}\left(D\right)=O\right]}{\mathbb{P}\left[\mathcal{M}\left(D_{i\to t'}\right)=O\right]}.$$

The first term is between $e^{-\varepsilon}$ and e^{ε} since $\mathcal{M}$ is (Θ,ε)-IBCDP. We only need to show that the second term is also between $e^{-\varepsilon}$ and e^{ε} to conclude the proof. Notice that when $D(i)=t'$, we have $D_{i\to t'}=D$. Thus:

$$1=\frac{\mathbb{P}\left[\mathcal{M}\left(D_{i\to t'}\right)=O|D(i)=t'\right]}{\mathbb{P}\left[\mathcal{M}\left(D\right)=O|D(i)=t'\right]}=\frac{\mathbb{P}\left[D(i)=t'|\mathcal{M}\left(D_{i\to t'}\right)=O\right]}{\mathbb{P}\left[D(i)=t'|\mathcal{M}\left(D\right)=O\right]}\cdot\frac{\mathbb{P}\left[\mathcal{M}\left(D_{i\to t'}\right)=O\right]}{\mathbb{P}\left[\mathcal{M}\left(D\right)=O\right]}.$$

Again, the first term is between $e^{-\varepsilon}$ and e^{ε} since $\mathcal{M}$ is (Θ,ε)-IBCDP. Since multiplying it with the second term gives 1, the second term is also between $e^{-\varepsilon}$ and e^{ε}. If the attacker has background knowledge, all the probabilities above are conditioned by $\hat{B}$, and the same reasoning holds.

In the more general case where the attacker does have some partial knowledge, all probabilities above are conditioned by the value of this partial knowledge, and the same reasoning holds. □

This equivalence is only true in the context of this section, where $\delta=0$; we explain later why it fails when $\delta>0$.

Does Example 1 satisfy CausDP? It depends: if b can take arbitrarily large values, $\mathrm{avg}\left(D_{i\to b}\right)$ can be arbitrarily distinguishable from $\mathrm{avg}(D)$. Otherwise, b can only have a bounded influence on the average and $\left(\{\theta\},\varepsilon\right)$-CausDP can hold for some ε. In other words, when using the fixed version of the definition, whether a given mechanism is CausDP depends on the *sensitivity* of the mechanism. This is a good thing: it suggests that it captures the same intuition as DP.

Example 1 shows that CausDP is not stronger than NPr. Is the reverse true? In Example 3, we show that this is not the case.

Example 3. *Consider the same θ as for Example 1: it depends on a global parameter, μ, which is either +1 or -1 with equal probabilities, and each of the n records is normally distributed with mean μ and a small standard deviation σ. Let $\mathcal{M}$ be the algorithm that counts* outliers*: it computes the average $\tilde{\mu}$ of all data points, and returns the number of records outside $[\tilde{\mu}-5\sigma,\tilde{\mu}+5\sigma]$. As we saw before, conditioning θ on a value of $D(i)$ is approximately equivalent to fixing μ: the number of outliers is going to be the same no matter what (0 with high probability). However, if we first condition θ on $D(i)=t$, and then change this record into t', we can choose t' so that this record becomes an outlier; and make it 1 with high probability. Thus, this mechanism is NPr but not CausDP.*

Even though Example 3 shows that NPr does not imply CausDP, it is natural to think that in many cases, if you change one data point $D(i)$ *as well as all data points correlated with it*, it will have a bigger influence on the algorithm that if you only change $D(i)$ without modifying the rest of the data. In Example 4, we show that even for a simple data dependencies and mechanisms, we can find counterexamples to this intuition.

Example 4. *Consider a probability distribution θ that outputs $2n$ records, such as for all $i < n$, $D(2i)$ is picked from some arbitrary probability distribution with values in $\mathbb{N}$, and $D(2i+1) = D(2i)$. Then the mechanism that sums all records might be in NPr, but cannot be in CausDP. Indeed, if $D \sim \theta$, then $\sum_i D(i)$ will always be even, but changing one record without modifying its duplicate can make the sum odd.*

This last example that finding a special case where NPr implies CausDP is likely difficult. There is, however, a special case where both are equivalent: the absence of dependencies in the data. If changing one record does not influence other records, then NPr and CausDP are equivalent. This result is similar to Corollary 2 in [36], but is simpler and without the change in parameters.

Proposition 22. *Let Θ be a family of probability distributions such that for all $\theta \in \Theta$ and all $\hat{B} \in \mathcal{B}$, the random variables $D(i)_{|\theta,\hat{B}}$ are mutually independent. Then a mechanism $\mathcal{M}$ is (Θ, ε)-NPr iff it is (Θ, ε)-CausDP.*

Proof. Under these conditions, $D_{-i|\theta,\hat{B},D(i)=t}$ is the same as $D_{-i|\theta,\hat{B},D(i)=t'}$; as an immediate consequence, $\mathcal{M}(D)_{|\theta,\hat{B},D(i)=t'}$ is the same as $\mathcal{M}(D_{i\to t'})_{|\theta,\hat{B},D(i)=t}$. The statement follows. □

This natural property, combined with the better behavior of CausDP in scenarios like Example 2, might seem like CausDP is a better alternative to CausDP, when one wants to capture an attacker with partial knowledge, under the causal interpretation of differential privacy. However, even with this fix, when records are not independent, CausDP is not always safe to use. We present an example from Adam Smith (personal correspondence, 2018-09-28) showing that a slightly modified version of the identity function can still be CausDP if records are strongly correlated.

Example 5. *Let θ output $3n$* triplicated *records: for all $i < n$, D_{3i} is picked from some probability distribution R, and $D(3i+1) = D(3i+2) = D(3i)$. Let $\mathcal{M}$ be a mechanism that "corrects" a modified record: if there is a record value x appearing only once, and a value y appearing only twice, then $\mathcal{M}$ changes the record x to y; then $\mathcal{M}$ always outputs the entire database. It is easy to check that $\mathcal{M}$ is $(\{\theta\}, 0)$-CausDP.*

This example is more artificial than Example 2, as the mechanism itself "cheats" to use dependencies in the data. Nonetheless, it shows that some mechanisms that leak the full database can be CausDP. Thus, using CausDP as a privacy measure of a given mechanism is dangerous if no information about the mechanism is known. We do not know whether more natural mechanisms could lead to similar counterexamples, for certain classes of probability distributions; but it is clear that simply applying the same technique as the causal variants of [368] is insufficient to solve entirely the problems with correlations under partial knowledge.

3.1.2.3 *Imposing an additional criterion on the definition*

In this section, we propose a criterion that the distribution θ must satisfy before NPr can be used. We argue that when this criterion is not satisfied, NPr is not a good measure of the privacy of a mechanism. But when it is satisfied, we obtain natural properties that are false in general: NPr and CausDP are equivalent, and attackers with more partial knowledge are stronger.

We mentioned previously that NPr was not *monotonous*: an attacker with *more* knowledge can be considered to be *less powerful*. In Example 1, an attacker A who did not know μ can *learn* μ by observing $\mathcal{M}(D)$. This increases their knowledge about $D(i)$: they now know that $D(i)$ is probably around μ, a fact previously unknown. However, an attacker B, who *already knew* μ, does not increase their knowledge as much when observing $\mathcal{M}(D)$. Example 2 and Example 5 show that DistDP and CausDP also suffer from this issue. How can we fix this problem? The informal goal is that an attacker with more background knowledge should gain more information. We must make sure that the privacy quantification ε cannot be artificially inflated by non-sensitive information learned by the attacker.

In all previous examples, the attacker's partial knowledge is strongly correlated with the sensitive information. Thus, ε does not measure the privacy loss *due to the mechanism*, but also takes into account the prior knowledge from the attacker about the sensitive attribute. Modeling the attacker's uncertainty is necessary to formalize their partial knowledge, but the only thing that should be captured by ε is the *mechanism*'s privacy leakage. To ensure this is the case, we argue that the partial knowledge must be independent from the sensitive information, and we propose a formalization that enforces this distinction between sensitive information and partial knowledge.

To this end, we propose an alternative way to model the attacker's uncertainty, and suggest to *normalize* the distribution θ to cleanly separate sensitive information and partial knowledge. We show that if such a normalization exists, then an attacker with more partial knowledge is more powerful. Thus, the existence of such a normalization is a desirable property for privacy definitions that model an attacker with partial knowledge, and we argue that it should serve as a *criterion* that must be satisfied before using such definitions, in order to get meaningful results.

How to formalize the intuition that the sensitive information should be separated from the partial knowledge? The core idea is to express θ as the output of a *generative function*, with independent random parameters. Each possible value of these parameters corresponds to a possible database.

Definition 53 (Normalization of data-generating distributions). *A* normalization *of a probability distribution θ is a family of mutually independent random variables $(\phi_0, \dots, \phi_k)$, and an injective, deterministic function $\hat{\theta}$, such that $\theta = \hat{\theta}(\phi_0, \dots, \phi_k)$.*

A normalization *de-correlates* the distribution: it splits its randomness into independent parts ϕ_i. The ϕ_i can then play distinct roles: one parameter can capture the sensitive property, while the others can model the attacker's partial knowledge. We capture this additional requirement in the following definition.

Definition 54 (Acceptable parameters)**.** *Given a distribution θ with values in $\mathcal{D} \times \mathcal{B}$, an* acceptable normalization *of θ at index i is a normalization $\hat{\theta}(\phi_0, \dots, \phi_k)$ where:*

1. *ϕ_0 entirely determines the sensitive attribute $D(i)$: there exists a function f such that for all possible values of $D(i)$, $\mathbb{P}_\theta[f(\phi_0) = D(i)] = 1$.*

2. *Some of the $\phi_1, \dots, \phi_k$ entirely determine the partial knowledge: there exists $I \subseteq \{1, \dots, k\}$ and an injective function g such that $\mathbb{P}_\theta\left[g\left((\phi_j)_{j\in I}\right) = B\right] = 1$.*

A family of distributions Θ is acceptable *if each $\theta \in \Theta$ has an acceptable normalization at all indices i.*

In practice, we can simply consider a family $(\phi_j)_{j\in I}$ for some I to be the attacker's partial knowledge, rather than using a bijection. When partial knowledge is defined using a subset of parameters, an attacker has more partial knowledge when they know more parameters.

Informally, ϕ_0 must contain enough information to retrieve the sensitive attribute, and the partial knowledge must be independent from it. How can we normalize the θ in Example 1? We cannot have one parameter for μ, and n parameters for the noise added to each record: the sensitive attribute $D(i)$ would require two parameters to express. Rather, ϕ_0 could be a pair containing *both* μ and the noise added at i, $D(i) - \mu$. The attacker's partial knowledge can be $D(j) - \mu$, for $j \neq i$, without μ. In other words, μ itself is also sensitive. What if we do not want to consider μ as sensitive? Then, ϕ_0 can be the value of the noise added to the record i, $D(i) - \mu$, and μ is another parameter that can (or not) be part of the attacker's partial knowledge.

As this example shows, this separation between the partial knowledge and the sensitive value forces us to carefully choose the sensitive value, and it prevents us from comparing scenarios where the sensitive value varies. This formalism can now be used to precisely define the relative strength of two attackers, based on their partial knowledge, and show that an attacker with more knowledge is more powerful.

Definition 55 (Relative strength of partial knowledge)**.** *Given probability distributions θ_1 and θ_2 with values in $\mathcal{D} \times \mathcal{B}$, we say that θ_1 has* more background knowledge *than θ_2 if the three following conditions are satisfied.*

1. *$D_{|\theta_1} = D_{|\theta_2}$: the only difference between the probability distributions is the partial knowledge.*

2. *For all i, there exists an acceptable normalization $\hat{\theta}(\phi_0, \dots, \phi_k)$ given i that is common to θ_1 and θ_2.*

3. *For all i, if we denote I_1 and I_2 the set of parameters that correspond to θ_1 and θ_2 respectively in this acceptable normalization, then $I_1 \supseteq I_2$.*

Given two families of probability distributions Θ_1 and Θ_2, we say that Θ_1 has more background knowledge than Θ_2 if for all $\theta_2 \in \Theta_2$, there exists $\theta_1 \in \Theta_1$ such that θ_1 has more background knowledge than θ_2.

Proposition 23. *Let Θ_1 and Θ_2 be two families of distributions such that Θ_1 has more background knowledge than Θ_2. If a mechanism $\mathcal{M}$ satisfies (Θ_1, ε)-NPr, it also satisfies (Θ_2, ε)-NPr.*

Proof. Suppose that $\mathcal{M}$ is (Θ_1, ε)-NPr. For a distribution $\theta_2 \in \Theta_2$ and an index i, let $\theta_1 \in \Theta_1$ be such that θ_1 is stronger than θ_2. By definition, there exists an acceptable normalization $\hat{\theta}(\phi_0, \ldots, \phi_k)$ common to θ_1 and θ_2. Let f be the function extracting the sensitive value from ϕ_0 in this normalization. Denoting I_1 and I_2 the set of parameters corresponding respectively to θ_1 and θ_2, as a simplification, we assume that $I_2 = \varnothing$ and $I_1 = \{1\}$; it is straightforward to adapt the proof to the more generic case. For any output O, and all values $t, t' \in \mathcal{T}$, we can decompose:

$$\begin{aligned} &\mathbb{P}\left[\mathcal{M}(D) = O \,\middle|\, f(\phi_0) = t\right] \\ &\quad = \sum_{\hat{B}} \mathbb{P}\left[\phi_1 = \hat{B} \,\middle|\, f(\phi_0) = t\right] \cdot \mathbb{P}\left[\mathcal{M}(D) = O \,\middle|\, f(\phi_0) = t, \phi_1 = \hat{B}\right]. \end{aligned}$$

The ϕ_i are independent: $\mathbb{P}\left[\phi_1 = \hat{B} \,\middle|\, f(\phi_0) = t\right]$ is the same as $\mathbb{P}\left[\phi_1 = \hat{B} \,\middle|\, f(\phi_0) = t'\right]$. Since $\mathcal{M}$ satisfies (Θ_1, ε)-NPr, we also have for all $\hat{B}$:

$$\mathbb{P}\left[\mathcal{M}(D) = O \,\middle|\, f(\phi_0) = t, \phi_1 = \hat{B}\right] \leq e^{\varepsilon} \mathbb{P}\left[\mathcal{M}(D) = O \,\middle|\, f(\phi_0) = t', \phi_1 = \hat{B}\right].$$

Thus:

$$\begin{aligned} &\mathbb{P}\left[\mathcal{M}(D) = O \,\middle|\, f(\phi_0) = t\right] \\ &\quad \leq e^{\varepsilon} \sum_{\hat{B}} \mathbb{P}\left[\phi_1 = \hat{B} \,\middle|\, f(\phi_0) = t'\right] \cdot \mathbb{P}\left[\mathcal{M}(D) = O \,\middle|\, f(\phi_0) = t', \phi_1 = \hat{B}\right] \\ &\quad \leq e^{\varepsilon} \mathbb{P}\left[\mathcal{M}(D) = O \,\middle|\, f(\phi_0) = t'\right] \end{aligned}$$

and thus, $\mathcal{M}$ satisfies (Θ_2, ε)-NPr. Adapting the proof to cases where $I_1 \subsetneq I_2$ is straightforward. □

This proposition states that, for acceptable distributions, partial background knowledge can be formalized in a reasonable and intuitive way, guaranteeing that attackers with more background knowledge are stronger. Another advantage is that when this criterion holds, NPr and CausDP are equivalent.

Proposition 24. *If Θ is an acceptable distribution, then for any ε and δ, $(\Theta, \varepsilon, \delta)$-NPr is equivalent to $(\Theta, \varepsilon, \delta)$-CausDP.*

The proof is the same as for Proposition 22. Figure 3.1 summarizes the relations between the definitions introduced in this section.

Acceptable normalizations force practitioners to define which information is considered private. If, like in DP, the sensitive information is the value of a given record, with an acceptable normalization, the attacker's partial knowledge cannot contain records correlated with the target record. This might seem overly restrictive: what if the attacker *does* know some information correlated with the target record? In this case, one must change the sensitive information to only consider the decorrelated part as sensitive.

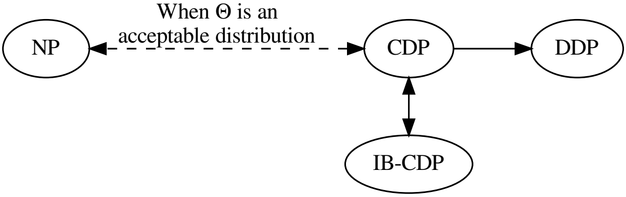


FIGURE 3.1: Relations between definitions introduced in Section 3.1.2, assuming $\delta = 0$.

To illustrate this process, consider a medical database where the diagnostic records of different patients might be correlated. To find an acceptable normalization, we must choose between two options. The first is to consider attackers that do not have information correlated with the diagnosis of a given patient: in that case, we must protect the information of multiple patients at once, similarly to *DP under correlations* (a variant of DP defined in [73]). Another option is for the sensitive property to be the diagnosis of someone *given the diagnosis of those they are correlated with*. We formally present a simpler example below, in Example 6.

Example 6. *Consider a referendum, where people vote in pairs, with some amount of correlation between pairs. More precisely, θ is a distribution that generates a database of $2n$ records according to the following process:*

- $D(2i) = 1$ *with probability p_i, and 0 with probability $1 - p_i$;*
- $D(2i+1) = D(2i)$ *with probability p_c, and $1 - D(2i)$ with probability $1 - p_c$.*

Suppose the attacker is interested in $D(1)$. There are two ways of modeling this with an acceptable normalization, depending on whether we want to allow the attacker to know $D(0)$.

- *We can pick ϕ_i to determine the value of* both *$D(2i)$ and $D(2i+1)$. This way, ϕ_0 is sufficient to know the value of the sensitive information $D(i)$, and is independent from all ϕ_i for $i > 0$.*
- *We can also pick ϕ_{2i} to be the event "$D(2i) = D(2i+1)$", and ϕ_{2i+1} to be the value of $D(2i)$. In that case, the attacker is allowed to know $D(0)$, but the sensitive value has changed: the sensitive value is whether $D(0) = D(1)$; which is independent from the actual value of $D(0)$.*

3.1.3 *Passive vs. active attackers*

In this section and the rest of this chapter, we assume that the criterion introduced in Definition 54 is satisfied: the data-generating distributions considered are always acceptable. For simplicity, we also assume that the sensitive property is the value of one record. Under these conditions, any value of the partial knowledge $\hat{B}$ is compatible with any value of the sensitive record $D(i) = t$, since these two events are independent. This also allows us to only consider the formalism of NPr; since it is equivalent with CausDP under this criterion (Proposition 24).

Correlations in the data are not the only issue to account for when limiting the attacker's background knowledge in DP. Another important question is whether the attacker simply *receives* some partial knowledge passively, or whether the attacker can *actively influence* the data. In this section, we show how to model these two situations by adapting the notion of a privacy loss random variable to model an attacker with partial knowledge, and we explore the relationship between the two corresponding definitions. To help understand this distinction, in this section, we consider the following example.

Example 7 (Thresholding). *1000 people take part in a Yes/No-referendum. Each person votes "Yes" with some probability, independently from the others. The mechanism $\mathcal{M}$ counts the number of "Yes" votes, but only returns this if it is above 100; otherwise it returns $\perp$. The partial knowledge B contains the votes of 100 participants, and the attacker wants to know the vote of another individual $D(i)$. We will see that in case the probability that each person votes "Yes" is very small (say, 10^{-7}), the privacy of this scheme will depend on whether the attacker is passive or active.*

3.1.3.1 *Privacy loss random variable for partial knowledge*

In Section 2.2.2, we introduced the *privacy loss random variable* (Definition 13), defined as:

$$\mathcal{L}^{\mathcal{M}}_{D_1/D_2}(O) = \ln \frac{\mathbb{P}\left[\mathcal{M}(D_1) = O\right]}{\mathbb{P}\left[\mathcal{M}(D_2) = O\right]}.$$

The PLRV quantifies how much information is revealed by the output of a mechanism, and we saw in Proposition 8 that a mechanism $\mathcal{M}$ is (ε, δ)-DP iff for all neighboring databases D_1 and D_2, and all outputs O:

$$\mathop{\mathbb{E}}_{O\sim\mathcal{M}(D_1)}\left[\max\left(0, 1 - e^{\varepsilon - \mathcal{L}_{D_1/D_2}(O)}\right)\right] \le \delta.$$

Now, suppose the attacker only has partial knowledge about the data. How to adapt the definition of the PLRV to this setting? The data comes from a distribution θ, and the attacker tries to distinguish between $D(i) = t$ and $D(i) = t'$ by observing $\mathcal{M}(D)$, given partial knowledge $\hat{B}$. Since $\hat{B}$ is given to the attacker *prior* to $\mathcal{M}(D)$, we must *condition* the probabilities by $\hat{B}$.

Definition 56 (PLRV for partial knowledge). *Given a mechanism $\mathcal{M}$, a distribution θ with values in $\mathcal{D} \times \mathcal{B}$, an index i, and values $t, t' \in \mathcal{T}$, the* PLRV of an output $\hat{O} \in O$ given partial knowledge $\hat{B} \in \mathcal{B}$ *is:*

$$\mathcal{L}^{\mathcal{M},\theta}_{i\leftarrow t/i\leftarrow t'}(\hat{O}, \hat{B}) = \ln \frac{\mathbb{P}_\theta\left[\mathcal{M}(D) = \hat{O} \,\middle|\, D(i) = t, B = \hat{B}\right]}{\mathbb{P}_\theta\left[\mathcal{M}(D) = \hat{O} \,\middle|\, D(i) = t', B = \hat{B}\right]},$$

using the convention $x/0 = \infty$ for all x.

This PLRV captures the same idea as for classical DP: it quantifies the attacker's information gain. If $B = D_{-i}$, this definition is the same as the classical PLRV.

Now that we translated the concept of PLRV to account for partial knowledge, we can use it to adapt the privacy definition. The formula in Proposition 8 averages the PLRV over all possible outputs O, but the PLRV with partial knowledge has a second parameter, $\hat{B}$. How should this new parameter be handled? There are at least two reasonable possibilities.

3.1.3.2 *Active partial knowledge*

The first option is to quantify over *all* possibilities for the attacker's partial knowledge. We assume the worst: we consider the case where the attacker's partial knowledge causes the privacy to be the greatest. This models a scenario where the attacker can not only see, but also *influence* the data. If the attacker can, for example, add fake users to the database, then they can choose the values associated to these new records to maximize the chances of information gain. We therefore call this option *active* partial knowledge, short for "partial knowledge under active attacks".

Definition 57 (APKDP). *Given a family of distributions* Θ*, a mechanism* $\mathcal{M}$ *is* $(\Theta, \varepsilon, \delta)$-APKDP (Active Partial Knowledge Differential Privacy) *if for all distributions* $\theta \in \Theta$*, all indices* i*, all* $t, t' \in \mathcal{T}$*, and all* $\hat{B} \in \mathcal{B}$*:*

$$\mathop{\mathbb{E}}_{\theta_{|D(i)=t,\hat{B}},\, O\sim\mathcal{M}(D)}\left[\max\left(0, 1 - e^{\varepsilon - \mathcal{L}^{\mathcal{M},\theta}_{i\leftarrow t/i\leftarrow t'}(O,\hat{B})}\right)\right] \leq \delta,$$

or, equivalently:

$$\mathcal{M}\left(D\right)_{|\theta,D(i)=t,\hat{B}} \approx_{\varepsilon,\delta} \mathcal{M}\left(D\right)_{|\theta,D(i)=t',\hat{B}}\,.$$

The proof of this equivalence is the same as in Lemma 1 in [278], which makes it explicit that APKDP is the same as NPr in its reformulation in [36]. As shown in [368], APKDP and DP coincide whenever the attacker has full knowledge (when $B = D_{-i}$).

With APKDP, a fixed part of the distribution can be arbitrarily determined. In Example 7, this corresponds to having the attacker control some percentage of voters. Such an active attacker can simply add many fake "Yes" votes to the database, to reach the threshold of 100: this makes the thresholding pointless. $\mathcal{M}$ then becomes a simple counting query which does not provide privacy: with high probability, everybody votes "No", and the only uncertainty left is over the attacker's target.

In addition to modeling an active attacker, APKDP can also be used in scenarios where θ is unknown, but can be *approximated*. The "partial knowledge" can represent the error between the true distribution and the approximation and if APKDP is satisfied, then the privacy property also holds for the true database.

Note that in this context, explicitly modeling the background knowledge $\hat{B}$ is technically not necessary. Instead, we could simply create a new family of probability distributions Θ' by conditioning each $\theta \in \Theta$ by the value of each possible $\hat{B}$. We make this background knowledge explicit instead, so APKDP is easier to compare with PPKDP, defined in the next section.

3.1.3.3 *Passive partial knowledge*

APKDP represents situations where the attacker can modify the data. An example is an online service that publishes statistics about its use, where the attacker can interact with the service before the usage statistics are published. Now, what if the attacker cannot interact with the data? Consider e.g. researchers publishing the results of a clinical study about patients having a medical condition. A typical attacker cannot influence the clinical data, but might have some partial knowledge about other participants to the survey.

How can we model such a *passive* attacker, which has access to some information about the data, but cannot influence it? It no longer makes sense to quantify over arbitrary partial knowledge. In the same way that the reformulation of (ε, δ)-DP using the PLRV averages the PLRV over all possible outputs, we must average the PLRV over all possible values of the partial knowledge.

Definition 58 (PPKDP). *Given a family of distributions* Θ*, a mechanism* $\mathcal{M}$ *is* $(\Theta, \varepsilon, \delta)$-PPKDP (Passive Partial Knowledge Differential Privacy) *if for all distributions* $\theta \in \Theta$*, all indices* i*, and all* $t, t' \in \mathcal{T}$*:*

$$\mathop{\mathbb{E}}_{\theta_{|D(i)=t}, O \sim \mathcal{M}(D)} \left[\max\left(0, 1 - e^{\varepsilon - \mathcal{L}^{\mathcal{M},\theta}_{i \leftarrow t / i \leftarrow t'}(O,B)}\right)\right] \leq \delta.$$

In this context, δ has a similar meaning as in (ε, δ)-DP: it captures the probability that the attacker is *lucky*. In (ε, δ)-DP, it means that O allows the attacker to distinguish between D_1 and D_2 with high probability, or, equivalently, that the PLRV associated to output O is large. In (ε, δ)-PPKDP however, δ captures the probability of the attacker getting either a favorable output O, or a favorable partial knowledge B.

With PPKDP, the thresholding mechanism of Example 7 is private. Indeed, with high probability, the partial knowledge will have only “No” votes; and almost certainly, the mechanism will output $\perp$ and gives no information. We formalize this intuition in Section 3.2.2.

Note that as opposed to APKDP, PPKDP cannot be easily reformulated using (ε, δ)-indistinguishability. Since the statement $B = \hat{B}$ *conditions* both probabilities, the δ in (ε, δ)-indistinguishability only applies to the randomness of $\mathcal{M}$. To use an indistinguishability-based formulation, we would need to use δ twice, and (for example) explicitly require that (ε, δ)-indistinguishability holds with probability $1 - \delta$ over the choice of $\hat{B}$.

Remark 1. *PPKDP shares some characteristics with inference-based distributional differential privacy (IBDDP), introduced in [36]. A mechanism* $\mathcal{M}$ *satisfies IBDDP if there is a simulator* Sim *such that for all probability distributions* $\theta \in \Theta$*, and indices* i*, the statement:*

$$D(i)_{|\theta, \mathcal{M}(D) = \hat{O}, \hat{B}} \approx_{\varepsilon,\delta} D(i)_{|\theta, \mathrm{Sim}(D_{-i}) = \hat{O}, \hat{B}}$$

holds with probability $1 - \delta$ *over the choice of* $\hat{O}$ *and* $\hat{B}$*.*

Leaving aside the simulator, note that δ *is used in* two *separate parts of the definition: both over the choice of* $\hat{O}$ *and* $\hat{B}$*, and in the indistinguishability. As such, it is difficult see intuitively what* δ *corresponds to, and the interpretation based on the “probability that the attacker gets more information than* e^{ε}*” is not correct. This is one of the reasons why a*

PLRV-based formulation is more convenient: δ can simply be interpreted in the same way as in (ε, δ)-differential privacy.

Further, the strict implication between DistDP and IBDDP proven in [36] can be explained by a similar distinction between an active and a passive attacker, even if it is not made explicit in the original paper. Indeed, when $\delta > 0$, this δ applies only to the indistinguishability property of DistDP, and DistDP quantifies over all possible values of the background knowledge: the attacker is assumed to be able to choose the most favorable value of the background knowledge. In contrast, with IBDDP, δ is applied both to the indistinguishability property and *to the choice of the background knowledge; hence the attacker is implicitly assumed to get the background knowledge randomly.*

3.1.3.4 *Relation between definitions*

In this section, we formalize the relation between PPKDP and APKDP and show basic results on those definitions.

First, APKDP and PPKDP satisfy both *privacy axioms* proposed in [225], which we reproduced in Definition 9 in Section 2.2.1. These axioms express natural properties that we expect to be true for any reasonable definition of privacy.

Proposition 25. *PPKDP satisfies the post-processing axiom: if a mechanism $\mathcal{M}$ is $(\Theta, \varepsilon, \delta)$-PPKDP, then for any function f, the mechanism $\mathcal{M}'$ defined by $\mathcal{M}'(D) = f(\mathcal{M}(D))$ is also $(\Theta, \varepsilon, \delta)$-PPKDP. It also satisfies the convexity axiom: if two mechanisms $\mathcal{M}_1$ and $\mathcal{M}_2$ are both $(\Theta, \varepsilon, \delta)$-PPKDP, then the mechanism $\mathcal{M}$ that applies $\mathcal{M}_1$ with some probability p and $\mathcal{M}_2$ with probability $1 - p$ is also $(\Theta, \varepsilon, \delta)$-PPKDP.*

APKDP also satisfies these axioms.

Proof. For APKDP, the reformulation of the definition using classical (ε, δ)-indistinguishability makes the result straightforward: the proof is the same as for (ε, δ)-DP. For PPKDP, we first reformulate the definition using f-divergence. Let $f(x) = \max(0, 1 - e^{\varepsilon} x)$. Then a mechanism $\mathcal{M}$ is $(\Theta, \varepsilon, \delta)$-PPKDP iff:

$$\sum_{\hat{B}} \mathbb{P}\left[\hat{B} \mid D(i) = t\right] D_f\left(\mathcal{M}(D)_{|\theta, D(i)=t, \hat{B}} \,||\, \mathcal{M}(D)_{|\theta, D(i)=t', \hat{B}}\right) \leq \delta.$$

This view allows us to use the monotonicity and joint convexity properties of the f-divergence to immediately prove the result for PPKDP. □

We saw that (ε, δ)-APKDP bounds the probability mass of the PLRV above ε by δ, for all possible partial knowledge $\hat{B}$. By contrast, PPKDP bounds the same probability mass, *averaged* over all possible values of $\hat{B}$, weighted by their likelihood. We formalize this interpretation and use it to show that APKDP is, as expected, stronger than PPKDP. More surprisingly, we also use it to show that when $\delta = 0$, both definitions are equivalent.

Theorem 1. *Given a distribution θ, a mechanism $\mathcal{M}$, an index i, two values $t, t' \in \mathcal{T}$, an output O, a possible value of the partial knowledge $\hat{B}$, and a fixed $\varepsilon > 0$, let us denote*

$m\left(O,\hat{B}\right) = \max\left(0, 1-e^{\varepsilon-\mathcal{L}^{\mathcal{M},\theta}_{i\leftarrow t/i\leftarrow t'}(O,\hat{B})}\right)$*. The respective quantities bounded by the requirements of APKDP and PPKDP are:*

$$APK_{i,t,t',\hat{B}} = \mathop{\mathbb{E}}_{\theta_{|D(i)=t,\hat{B}},O\sim\mathcal{M}(D)}\left[m\left(O,\hat{B}\right)\right]$$

and:

$$\begin{aligned} PPK_{i,t,t'} &= \mathop{\mathbb{E}}_{\theta_{|D(i)=t},O\sim\mathcal{M}(D)}\left[m\left(O,B\right)\right] \\ &= \mathop{\mathbb{E}}_{\theta_{|D(i)=t}}\left[APK_{i,t,t',B}\right]. \end{aligned}$$

As an immediate consequence, if $\mathcal{M}$ *is* $(\Theta,\varepsilon,\delta)$*-APKDP, then it is also* $(\Theta,\varepsilon,\delta)$*-PPKDP. Further,* (Θ,ε)*-PPKDP and* (Θ,ε)*-APKDP are equivalent.*

Proof. We decompose $\text{PPK}_{i,t,t'}$ depending the value of B.

$$\begin{aligned} \text{PPK}_{i,t,t'} &= \sum_{\hat{D},\hat{B}} \mathbb{P}\left[D=\hat{D},\hat{B} \mid D(i)=t\right] \mathop{\mathbb{E}}_{O\sim\mathcal{M}(\hat{D})}\left[m\left(O,\hat{B}\right)\right] \\ &= \sum_{\hat{B}} \mathbb{P}\left[\hat{B} \mid D(i)=t\right] \sum_{\hat{D}} \mathbb{P}\left[D=\hat{D} \mid D(i)=t,\hat{B}\right] \cdot \mathop{\mathbb{E}}_{O\sim\mathcal{M}(\hat{D})}\left[m\left(O,\hat{B}\right)\right] \\ &= \sum_{\hat{B}} \mathbb{P}\left[\hat{B} \mid D(i)=t\right] \mathop{\mathbb{E}}_{\theta_{|D(i)=t,\hat{B}},O\sim\mathcal{M}(D)}\left[m\left(O,\hat{B}\right)\right] \\ &= \sum_{\hat{B}} \mathbb{P}\left[\hat{B} \mid D(i)=t\right] \text{APK}_{i,t,t',\hat{B}} \\ &= \mathop{\mathbb{E}}_{\theta_{|D(i)=t}}\left[APK_{i,t,t',B}\right]. \end{aligned}$$

For the second part of the statement, if the mechanism is (ε,δ)-APKDP, then for all i, t, t', and $\hat{B}$, $\text{APK}_{i,t,t',\hat{B}} \le \delta$, so:

$$\begin{aligned} \text{PPK}_{i,t,t'} &= \mathop{\mathbb{E}}_{\theta_{|D(i)=t}}\left[\text{APK}_{i,t,t',B}\right] \\ &\le \mathop{\mathbb{E}}_{\theta_{|D(i)=t}}\left[\delta\right] \\ &\le \delta. \end{aligned}$$

For the last part of the statement, assume that $\mathcal{M}$ is (Θ,ε)-PPKDP. Then:

$$0 = \text{PPK}_{i,t,t'} = \mathop{\mathbb{E}}_{\theta_{|D(i)=t}}\left[\text{APK}_{i,t,t',B}\right].$$

All summands are non-negative, so the sum can only be 0 if all summands are 0: for all $\hat{B}$, $\text{APK}_{i,t,t',\hat{B}} = 0$, and $\mathcal{M}$ is (Θ,ε)-APKDP. □

When $\delta > 0$, the implication is strict: the PLRV can be arbitrarily higher for certain values $\hat{B}$ of the background knowledge. Thus, quantifying over all possible values $\hat{B}$ can lead to much larger values of ε and δ than averaging over all possible values of the background knowledge: Example 7 illustrates this phenomenon. When $\delta = 0$ however, ε-APKDP and ε-PPKDP are both *worst-case* properties, like ε-DP: an attacker's ability to choose the background knowledge does not matter, since even for the passive attacker, we need to consider the worst possible output O and background knowledge B.

3.1.3.5 Θ-*reducible mechanisms*

For some mechanisms and probability distributions, active attackers are, perhaps surprisingly, not more powerful than passive attackers, even when $\delta > 0$. We introduce here a necessary condition for APKDP and PPKDP to be equivalent and we show that this condition appears in natural contexts.

Consider the example of a referendum where 2000 users take part in a vote with two options. Each user i votes "yes" with probability p_i, and "no" with probability $1 - p_i$. The mechanism $\mathcal{M}$ returns the exact tally of the vote. We assume that the attacker knows half of the votes: their partial knowledge is the vote of 1000 users. They might know that e.g. 500 of these users voted "yes", 500 voted "no", and the remaining 1000 votes are unknown. The attacker aims to get information on the vote of their target.

Does it matter, in this situation, whether the attacker is passive or active? If the attacker can choose the votes of 1000 users, they can decide that each known user will vote "yes". But changing these votes will only modify the tally in a predictable way: the attacker can remove these votes from the total tally. Intuitively, it does not matter whether these known users all vote "yes", "no", or have any other behavior known to the attacker. Without dependency relationships between users, the attacker's uncertainty solely resides in the unknown votes: a passive attacker is not weaker than an active attacker.

We generalize this intuition via the concept of Θ-*reducibility*, which captures scenarios where all possible values of background knowledge are equivalent from a privacy perspective. We then show that under this condition, APKDP is equivalent to PPKDP, and formalize the above example to show that it satisfies this condition.

Definition 59 (Θ-reducibility)**.** *A mechanism $\mathcal{M}$ is* Θ-reducible *if for all indices i, and all $B_1, B_2 \in \mathcal{B}$, there is a bijective mapping $\varphi^i_{B_1,B_2} : \mathcal{O} \to \mathcal{O}$ such that for all $t \in \mathcal{T}$ and for all $O \in \mathcal{O}$:*

$$\mathbb{P}_\theta \left[\mathcal{M}(D) = O \,\middle|\, D(i) = t, B = B_1\right] = \mathbb{P}_\theta \left[\mathcal{M}(D) = \varphi^i_{B_1,B_2}(O) \,\middle|\, D(i) = t, B = B_2\right].$$

This equivalence between possible outputs under B_1 and B_2 can be translated to an equivalence between the corresponding PLRVs: if $\mathcal{M}$ is Θ-reducible, then the global behavior of $\mathcal{L}^{\mathcal{M},\theta}_{i\leftarrow t/i\leftarrow t'}(O, B_1)_{|O\sim\mathcal{M}(D),D(i)=t}$ and of $\mathcal{L}^{\mathcal{M},\theta}_{i\leftarrow t/i\leftarrow t'}(O, B_2)_{|O\sim\mathcal{M}(D),D(i)=t}$ are the same. This equivalence between PLRVs enables us to show that Θ-reducibility implies APKDP and PPKDP are the same, even when $\delta > 0$. So there are mechanisms and background knowledge functions for which an active attacker is not stronger than a passive one.

Theorem 2. *Let Θ be a family of probability distributions, and let $\mathcal{M}$ be a Θ-reducible mechanism. Then $\mathcal{M}$ is $(\Theta, \varepsilon, \delta)$-APKDP iff it is $(\Theta, \varepsilon, \delta)$-PPKDP.*

Proof. First, we show that for all $O \in \mathcal{O}$ and all $B_1, B_2 \in \mathcal{B}$:

$$\mathcal{L}^{\mathcal{M},\theta}_{i \leftarrow t / i \leftarrow t'}(O, B_1) = \mathcal{L}^{\mathcal{M},\theta}_{i \leftarrow t / i \leftarrow t'}(\varphi_{B_1,B_2}(O), B_2).$$

This statement directly follows by unfolding the definition of $\mathcal{L}^{\mathcal{M},\theta}_{i \leftarrow t / i \leftarrow t'}(O, B_1)$ (Definition 56) and Θ-reducibility (Definition 59):

$$\begin{aligned}
\mathcal{L}^{\mathcal{M},\theta}_{i \leftarrow t / i \leftarrow t'}(O, B_1) &= \ln \frac{\mathbb{P}_\theta\left[\mathcal{M}(D) = O \,\middle|\, D(i) = t, B = B_1\right]}{\mathbb{P}_\theta\left[\mathcal{M}(D) = O \,\middle|\, D(i) = t', B = B_1\right]} \\
&= \ln \frac{\mathbb{P}_\theta\left[\mathcal{M}(D) = \varphi_{B_1,B_2}(O) \,\middle|\, D(i) = t, B = B_2\right]}{\mathbb{P}_\theta\left[\mathcal{M}(D) = \varphi_{B_1,B_2}(O) \,\middle|\, D(i) = t', B = B_2\right]} \\
&= \mathcal{L}^{\mathcal{M},\theta}_{i \leftarrow t / i \leftarrow t'}(\varphi_{B_1,B_2}(O), B_2).
\end{aligned}$$

We can now prove Theorem 2. Since $\text{APK}_{i,t,t'}$ is the expected value of $\text{PPK}_{i,t,t',B}$ (Theorem 1), it is enough to prove that for any B_1 and B_2, $\text{APK}_{i,t,t',B_1} = \text{APK}_{i,t,t',B_2}$. Recall that $m(O, B) = \max\left(0, 1 - e^{\varepsilon - \mathcal{L}^{\mathcal{M},\theta}_{i \leftarrow t / i \leftarrow t'}(O,B)}\right)$, and fix B_1 and B_2 in $\mathcal{B}$. We have:

$$\begin{aligned}
\text{APK}_{i,t,t',B_1} &= \sum_{\hat{O}} \mathbb{P}\left[\mathcal{M}(D) = \hat{O} \,\middle|\, D(i) = t, B = B_1\right] m\left(\hat{O}, B_1\right) \\
&= \sum_{\hat{O}} \mathbb{P}\left[\mathcal{M}(D) = \varphi_{B_1,B=B_2}(\hat{O}) \,\middle|\, D(i) = t, B = B_2\right] m\left(\varphi_{B_1,B_2}(\hat{O}), B_2\right)
\end{aligned}$$

using Definition 59 and the technical result above. We can then reindex the sum using the bijection $\hat{O} \to \varphi_{B_1,B_2}(\hat{O})$, and conclude:

$$\begin{aligned}
\text{APK}_{i,t,t',B_1} &= \sum_{\hat{O}} \mathbb{P}\left[\mathcal{M}(D) = \hat{O} \,\middle|\, D(i) = t, B = B_2\right] m\left(\hat{O}, B_2\right) \\
&= \text{APK}_{i,t,t',B_2}.
\end{aligned}$$

□

Θ-reducible mechanisms are fairly common, especially under the natural assumption that the attacker knows a fixed part of the dataset. We give a few examples.

Proposition 26. *Let Θ be a family of distributions in which each θ generates the record of each user independently, and assume that the background knowledge of the attacker is k fixed records of the database, for a given k. Then the following mechanisms are Θ-reducible.*

1. Counting queries*: given a predicate P, the algorithm's output is the number of records for which $P(D(i))$ is true. This a generalizes binary voting.*

2. Linear queries*: given a fixed family of weights (α_i), the mechanism returns $\sum_i \alpha_i D(i)$.*

3. Different types of means*: the arithmetic mean, the geometric mean, the harmonic mean, and the quadratic mean (assuming all $D(i)$ are positive).*

Proof. Fix $\theta \in \Theta$. For counting queries, given $B_1, B_2 \in \mathcal{B}$, define $\varphi^{count}_{B_1,B_2,i}(\mathcal{M}(D)) = \mathcal{M}(D) - \mathcal{M}(B_1) + \mathcal{M}(B_2)$. This function is linear, so injective. Call θ' the random distribution of the $n-k$ records not present in B. Then for all $a \in \mathcal{T}$ and all $O \in \mathcal{O}$:

$$\begin{aligned}
&\mathbb{P}_\theta \left[\mathcal{M}(D) = O \,\middle|\, D(i) = t, B = B_1\right] \\
&\quad = \mathbb{P}_{D' \sim \theta'} \left[\mathcal{M}(D' \cup B_1) = O \,\middle|\, (D' \cup B_1)_i = t\right] \\
&\quad = \mathbb{P}_{D' \sim \theta'} \left[\mathcal{M}(D' \cup B_2) = \varphi^{count}_{B_1,B_2,i}(O) \,\middle|\, (D' \cup B_2)_i = t\right] \\
&\quad = \mathbb{P}_\theta \left[\mathcal{M}(D) = \varphi^{count}_{B_1,B_2,i}(O) \,\middle|\, D(i) = t, B = B_2\right].
\end{aligned}$$

For linear queries, we use a similar mapping to φ, which also depends on the mapping. Let I be the indices of records present in B; then a linear query is Θ-reducible with:

$$\varphi^{linear}_{B_1,B_2,i}(\mathcal{M}(D)) = \mathcal{M}(D) + \sum_{j \in I} \alpha_j \left((B_2)_j - (B_1)_j\right).$$

This also proves the result for the arithmetic mean.

Similarly, it is easy to verify that the following φ functions show Θ-reducibility for the geometric, harmonic and quadratic mean.

$$\varphi^{geometric}_{B_1,B_2,i}(\mathcal{M}(D)) = \left(\mathcal{M}(D)^n \cdot \frac{\prod_{j \in I} (B_2)_j}{\prod_{j \in I} (B_1)_j}\right)^{1/n}$$

$$\varphi^{harmonic}_{B_1,B_2,i}(\mathcal{M}(D)) = n\left(\left(\frac{\mathcal{M}(D)}{n}\right)^{-1} + \sum_j \left(\frac{1}{(B_2)_j} - \frac{1}{(B_1)_j}\right)\right)^{-1}$$

$$\varphi^{quadratic}_{B_1,B_2,i}(\mathcal{M}(D)) = \sqrt{n \cdot \mathcal{M}(D)^2 + \sum_j \frac{(B_2)_j^2 - (B_1)_j^2}{n}}$$

The injectivity of each of these functions is clear. □

Note that in Proposition 26, it is crucial that the records in the attacker's partial knowledge are *fixed*. In general, knowing the records of k users is *not* equivalent to knowing the records of k other users. Indeed, if these records are generated with different probabilities, the randomness from the unknown records differs between both scenarios, and it is in general impossible to convert one into the other.

Remark 2. *Observation 1 in [176] claims that the partial knowledge of k records in a database of size n is the same as no partial knowledge in a database of size $n-k$. For counting queries, this holds for the same reason that counting queries are Θ-reducible: one can "remove" the k known records from the mechanism output and obtain a bijection between the cases with and without partial knowledge. Thus, the partial knowledge is irrelevant to the mechanism's privacy and can be ignored.*

This observation, however, does not hold in general: we later show in Example 7 that counting queries with thresholding *are not Θ-reducible, and in this case, the knowledge of k records in a database of size n has a very different effect than no partial knowledge in a database of size n − k.*

3.2 OPPORTUNITIES: AGGREGATIONS & THRESHOLDING

Consider a national referendum where more than 10 million people vote on a Yes/No question. Do the exact results of this referendum reveal information about individuals? It is very reasonable to assume that no realistic attacker has background knowledge of more than 99% of all votes. The remaining uncertainty of 1% of the data (100k data points) leads to a significant uncertainty that can, if properly quantified, show that no attacker can use the results of the referendum to determine how a given individual voted: the referendum results are private, even if no noise was added to them.

In the previous section, we introduced conceptual tools to rigorously analyse such scenarios. In this section, we build on these foundations to analyze such noiseless mechanisms and prove strong and intuitive results about their privacy. We also analyze the conditions necessary to use these notions in practical contexts, without making risky assumptions on their uncertainty or capabilities. Our main contributions are as follows.

First, we provide a formal account of the privacy of common kinds of queries under partial knowledge, using the tools introduced in the previous section. We show that counting queries under partial knowledge can provide privacy, with significantly lower bounds than those previously given.

Second, we show that thresholding—only returning the output if it is larger than a given threshold—can render counting queries private against passive attackers, even when there is not enough entropy in the data for the previous result to apply.

Third, we combine these results to show that under certain conditions, k-anonymity protects against a passive attacker with partial knowledge. We discuss the genericity and the limits of this result, which clearly delineates the conditions in which k-anonymity provides meaningful protection.

Finally, we look at composition, a property that is crucial to the practical applicability of privacy notions. We prove bounds for the sequential composition of noiseless mechanisms, which allows us to quantify the privacy leakage of multiple mechanisms with the same input, or of a mechanism repeated over time. We also look at nested composition: combining the uncertainty coming from the attacker's uncertainty with noise added to the result of the aggregation. We show basic properties of such mechanisms, and explain how numeric evaluation can be used to estimate the privacy loss in this context.

3.2.1 *Counting queries*

The initial motivation for limiting the attacker's background knowledge was to show that, under this assumption, some noiseless mechanisms preserve the individuals' privacy [46]. A typical example is a *counting query*, which answers the question "How many users satisfy P?"

for some property P. We can model this by a data-generating distribution θ where each record $D(i)$ is either 0 or 1 with some probability p_i, and we want to measure the privacy of the mechanism $\mathcal{M}(D) = \sum_i D(i)$. Records are assumed to be independent, and the adversary is assumed to know some portion of the records. As an immediate consequence of Theorem 2 and Proposition 26, it does not matter whether the attacker can modify, or only see, this portion of records: the values of ε and δ are identical for APKDP and PPKDP.

Furthermore, the closer p_i are to 0 or 1, the less randomness is present in the data. For extremely small or large values of p_i, the situation is very similar to one where the attacker exactly knows $D(i)$. As such, it is natural to assume that among the records that are unknown by the attacker, all p_i are between λ and $1-\lambda$, for some λ not too close to 0. This assumption can easily be communicated to non-specialists: "we assume that there are at least 1000 records that the attacker does not know, and that their level of uncertainty is at least 10% for these records."

Initial asymptotic results in this context appeared in [46] and more precise bounds were derived in [176]. In the special case where all p_i are equal to a fixed value p, Theorem 5 in [46] and Theorem 1 in [176] show that counting queries are APKDP with $\varepsilon = O\left(\sqrt{\frac{\ln(1/\delta)}{n}}\right)$ (for small δ, and increasing n) and $\delta = e^{-\Omega\left(\varepsilon^2 n\right)}$ (for small ε, and increasing n). This provides tiny values of ε and δ for moderate values of n and p. However, the assumption that all p_i are identical is unrealistic: in practice, there is no reason to assume that all users have an equal chance of satisfying P. Theorem 7 in [46] and Theorem 2 in [176] show that without this assumption, the ε obtained is still small: $\varepsilon = O\left(\sqrt{\frac{\ln(n)}{n}}\right)$, but the upper bound obtained on δ is significantly larger: $\delta = O\left(\frac{1}{\sqrt{n}}\right)$. This is more than what is typically acceptable; a common recommendation is to choose $\delta = o\left(\frac{1}{n}\right)$.

In the following theorem, we show that the exponential decrease of δ with n still holds in the general case where all p_i are different. For simplicity, we assume that the attacker has no background knowledge: because all records are independent, adding some partial knowledge has a fixed, reversible effect on the output space, similarly to Θ-reducibility. In this case, having the attacker know m records out of n is the same as having the attacker know no records among $n-m$.

Theorem 3. *Let θ be a distribution that generates n records, where $D(i)$ is the result of an independent Bernoulli trial of probability p_i. Let λ be such that for all i, $\lambda < p_i < 1-\lambda$. Let $\mathcal{M}$ be defined by $\mathcal{M}(D) = \sum_i D(i)$. Then $\mathcal{M}$ is $(\Theta, \varepsilon, \delta)$-APKDP, for any ε and δ such that:*

$$\delta \geq \mathbb{P}\left[\frac{X}{Y} \geq \varepsilon\right]$$

where X and Y are independent random variables sampled from a binomial distribution with $n-1$ trials and success probability 2λ. For a fixed $\varepsilon \leq 1$, this condition is satisfied if:

$$\varepsilon \geq \max\left(\sqrt{\frac{14\ln(1/\delta)}{\lambda(n-1)}}, \frac{27}{\lambda(n-1)}\right)$$

which gives $\delta = e^{-\Omega\left(\varepsilon^2 \lambda n\right)}$.

Proof. The proof uses existing results on privacy amplification by shuffling: in [27, 140], the authors show that adding noise independently to each data point, and then shuffling the results (hiding from the attacker which record comes from which user), provide strong DP guarantees. Even though our problem looks different, the same reasoning can be applied. First, we show that θ can be seen as applying randomized response on each record. Then, since a counting query is a symmetric boolean function, it can be composed with a shuffle of its input, which allows us to use amplification by shuffling.

Let us formalize this intuition. A Bernoulli trial of probability p_i (denoted Bernoulli (p_i)), with $\lambda < p_i < 1 - \lambda$, can be decomposed into the following process:

- Generate $b \sim$ Bernoulli (2λ).
- If $b = 0$, return $b_{v_i} \sim$ Bernoulli$\left(\frac{p_i - \lambda}{1-2\lambda}\right)$.
- If $b = 1$, return $b_{rr} \sim$ Bernoulli (0.5).

This can be seen as a *randomized response* process applied on some input b_{v_i}, itself random: $\theta = \mathcal{R}_{2\lambda}(\nu)$, where ν is a distribution generating n records, where the i-th record is generated by $b_{v_i} \sim$ Bernoulli$\left(\frac{p_i - \lambda}{1-2\lambda}\right)$, and $\mathcal{R}_{2\lambda}$ is a binary randomized response process with parameter 2λ.

Further, note that $\mathcal{M}$ can be seen as the composition between itself and a pre-shuffling phase: $\mathcal{M} = \mathcal{M} \circ \mathcal{S}$, where $\mathcal{S} : \mathcal{D} \to \mathcal{D}$ is a function that applies a random permutation to the input records. Thus, $\mathcal{M}\left(D\right)_{|D\sim\theta} = \mathcal{M}\left(\mathcal{S}\left(\mathcal{R}_{2\lambda}(D)\right)\right)_{|D\sim\nu}$. We can now apply Theorem 3.1 in [27] and its proof to show that $\mathcal{S} \circ \mathcal{R}_{2\lambda}$ is (ε, δ)-DP for ε within the constraints above (with $k = 2$ and $\gamma = 2\lambda$). By post-processing, $\mathcal{M} \circ \mathcal{S} \circ \mathcal{R}_{2\lambda}$ is also (ε, δ)-DP, which directly yields that $\mathcal{M}$ is (ε, δ)-APKDP.

Note that we omitted a small technical detail: conditioning θ on $D(i) = a$ is not identical to conditioning ν on $D(i) = a$, since no noise is added to the record i in the former case. To fix this, we need to define $\mathcal{R}_{2\lambda}$ as randomizing all records except a fixed one i. The proof of Theorem 3.1 in [27] assumes that no noise is added to the target record, so the result still holds. □

We compare this result with the previous state-of-the-art. First, we reformulate a previously known result from [176] that applies to our setting.

Proposition 27 (Theorem 3 in [176])**.** *Let θ be a distribution that generates n records, where $D(i)$ is the result of an independent Bernoulli trial of probability p_i, and let $\mathcal{M}$ be defined by $\mathcal{M}(D) = \sum_i D(i)$. Let $\mu_2 = \frac{1}{n}\sum_i p_i\left(1 - p_i\right)$ and $\mu_3 = \frac{1}{n}\sum_i p_i\left(1 - p_i\right)|1 - 2p_i|$ be respectively the average second moment and average absolute third moment of the $D(i)$. Then for any $\delta_2 \geq 1.25e^{-n\mu_2/2}$, $\mathcal{M}$ is $(\theta, \varepsilon, \delta)$-APKDP, with*

$$\varepsilon = \sqrt{\frac{2\ln(1.25/\delta_2)}{n\mu_2}} \quad \textit{and} \quad \delta = \frac{1.12\mu_3}{\sqrt{n\mu_2^3}}\left(1 + e^{\varepsilon}\right) + \delta_2.$$

Proof. For $\delta_2 = \frac{4}{5\sqrt{n}}$, this is a direct application of Theorem 3 in [176]. Changing the value δ_2 in its proof (Appendix A.2) allows us to obtain the more general formula above. This

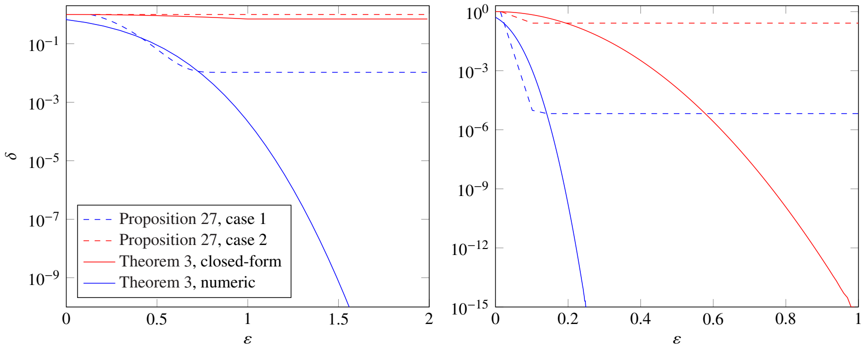


FIGURE 3.2: Comparison of (ε, δ) bounds given by Theorem 3 and Proposition 27, for $\lambda = 0.05$, $n = 100$ (left) and $n = 10{,}000$ (right).
Case 1: all but one p_i are 0.5. Case 2: the p_i are distributed uniformly over $[0.05, 0.95]$.

requires Fact 1 to be true, which is the case when $\varepsilon \leq 1$ (the authors omit this detail), or equivalently, when $\delta_2 \geq 1.25e^{-n\mu_2/2}$. □

The comparison between this result and Theorem 3 is not completely straightforward. Aside from n, Theorem 3 only depends on a global bound on the "amount of randomness" (p_i) of each user, while Proposition 27 depends on the average behavior of all users. As such, the global bound λ can be small because of one single user having a low p_i, even if all other users have a lot of variance because their p_i is close to 0.5. We therefore provide two experimental comparisons. In the first one, $p_1 = \lambda = 0.05$ and $p_i = 0.5$ for all $i > 1$. This case is designed to have the parameters of Theorem 3 underperform (as we underestimate the total amount of randomness) and those of Proposition 27 perform well. In the second one, the p_i are uniformly distributed in $[\lambda, 1 - \lambda] = [0.05, 1 - 0.05]$. In both cases, we compare the (ε, δ) graphs obtained for $n = 1000$ and $n = 100{,}000$, and present the results in Figure 3.2.

The graphs show that if we consider the smallest possible ε given by the definitions, our theorem leads to a large δ: with $\varepsilon = \Theta\left(1/\sqrt{n}\right)$, we obtain $\delta = \Theta(1)$; in contrast, Proposition 27 leads to $\delta = \Theta\left(1/\sqrt{n}\right)$. However, increasing ε to slightly larger values quickly leads to tiny values of δ, which was impossible with the previous state-of-the-art results. They also show that the closed-form bound from [27] is far from tight, as numerically computating these bounds improves them by several orders of magnitude. This leads to a natural open question: is there a better asymptotic formulation of the bounds given by amplification by shuffling for randomized response?

What is the impact of λ on the privacy guarantees? In Figure 3.3, we plot the ε obtained for $\delta = 0.01/n$ as a function of λ, for various values of n.

One natural application for this result is *voting*: in typical elections, the total tally is released without any noise. Adding noise to the election results, or not releasing them, would both be unacceptable. Thus, the results are not (ε, δ)-DP for any (ε, δ) parameters, even though publishing the tally is not perceived as a breach of privacy. The intuitive explanation for this

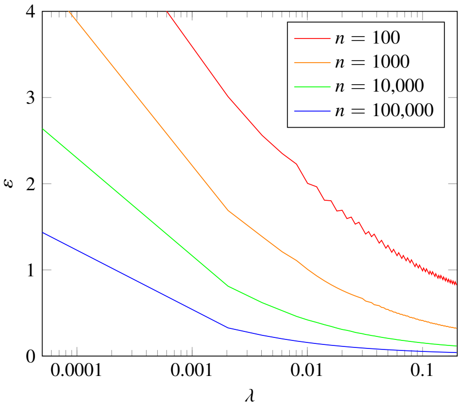


FIGURE 3.3: Comparison of ε bounds given by the numerical computation of Theorem 3, with varying λ, for various values of n and $\delta = 0.01/n$.

is that attackers are assumed not to have complete background knowledge of the secret votes. Our results confirm this intuition and quantify it. These results can easily be extended to votes between multiple candidates.

Corollary 1. *Let θ be a distribution that generates n records, where $D(i) \in \{1,\ldots,K\}$ for $K > 1$, and where $\mathbb{P}[D_i = k] = p_{i,k}$, and every record is independent from all others. Let λ be such that for all i and all k, $\lambda < p_{i,k}$. Let $\mathcal{M}$ return the histogram of all values: $\mathcal{M}(D) = (N_1,\ldots,N_K)$, where N_k is the number of records i such that $D(i) = k$. Then $\mathcal{M}$ is $(\Theta,\varepsilon,\delta)$-APKDP, for any $\varepsilon \leq 1$ and $\delta > 0$ such that:*

$$\varepsilon \geq \max\left(\sqrt{\frac{14\ln(1/\delta)}{\lambda(n-1)}}, \frac{27}{\lambda(n-1)}\right).$$

Proof. The proof is the same as for Theorem 3. With multiple options, the parameter γ of the multi-category randomized response is $\gamma = K\lambda$, which leads to the same (ε,δ) parameters. □

The results in this section apply to *individual* counting queries. This applies to scenarios like votes, but in many practical use cases, multiple queries are released. Can the results of this section be generalized to these cases? In general, noiseless mechanism do not compose. For example, fixing an individual t, queries like "How many people voted 1?" and "How many people who are not t voted 1?" can both be private on their own. However, publishing both results will reveal t's vote: the composition of both queries cannot be private. Are there special cases where noiseless counting queries can be composed?

One such case happens when each counting query contains the data of a number of *new* users, independent from users in previous counting queries. This can happen in situations where statistics are collected on actions that each user can only do once, for example, registering with an online service. In this case, we can restrict the privacy analysis of each new query to the set of independent users in its input, and use the previous results from this section: this approach is formalized and proven in Theorem 1 in [46].

What if this approach is impossible, for example if there are dependencies between each input record of each query? For example, a referendum could ask voters *multiple* questions, with correlations between the different possible answers. Another example could be app usage statistics published every day, where the data for day $d + 1$ for each user is correlated to the user's data on the previous day d. In this case, to compute the privacy loss of the first d binary queries, we can consider them as a *single* query with 2^d options. Afterwards, we can take into account the temporal correlations to compute the probabilities associated with each option, and use Corollary 1.

3.2.2 *Thresholding*

Theorem 3 gives good ε and δ parameters when there are many people who vote with "enough randomness": there is a λ such that $\lambda < p_i < 1 - \lambda$. The parameters have a dependency on λn, which in practice translates to scenarios where both options have large counts with high probability. In many practical applications, however, it is hard to know in advance whether this will be the case. Consider, for example, a mobile app gathering usage metrics on possible sequences of actions carried by users within the app. Some sequences will be very probable, and have high counts. But if there are arbitrarily many such sequences, some will be very rare: like many practical distributions, there will be a *long tail*.

To protect the data from these outlier users, a typical protection employed is *thresholding*: only return the user count associated with a sequence if it is larger than a given threshold T. What level of protection does such a technique provide? In this section, we formalize the intuition given in Example 7, and show that, assuming a passive attacker, thresholding provides protection when all voters vote with a very small or a very large probability. First, we formalize the notion of a thresholding mechanism in a simple context.

Definition 60 (Simple thresholding). *Given a database $D = (D(1), \ldots, D(n))$ with values in $\{0, 1\}$ and a threshold T, the T-*thresholding mechanism *$\mathcal{M}_T$ evaluates $\tilde{k} = \sum_i D(i)$ and returns $\perp$ if $\tilde{k} \leq T$, and k otherwise.*

Note that $\mathcal{M}_T$ only thresholds *low* counts. In many practical situations however, thresholding must be applied in both directions: it must also catch the case where, with high probability, almost all records are 1. This situation is symmetrical to thresholding low counts: without loss of generality, we can assume $T < n/2$, and the symmetric version of all results in this section hold.

Let us now show the main result of this section: if participants vote 1 with low probability, then thresholding protects against passive attackers, and in some cases also against certain active attackers. This privacy property only holds if the expected value of the count is lower than the threshold; and the level of protection depends on the ratio between the threshold and the expected value (denoted by r below).

Theorem 4. *Let θ be a distribution that returns n independent records, each of which is* 1 *with low probability: $D(i) \sim Ber(p_i)$, and $p_i < p$ for all i; moreover, let $\Theta = \{\theta\}$. Suppose that there is no partial knowledge, i.e., $|B| = 0$, and let us denote by $f(s,n,p)$ the probability that a random variable following a binomial distribution with parameters n and p has value s: $f(s,n,p) = p^s(1-p)^{n-s}\binom{n}{s}$.*

Then, if $r = \frac{p(n-1)}{(1-p)T} < 1$, $\mathcal{M}_T$ is $(\Theta,\varepsilon,\delta)$-APKDP (and thus, $(\Theta,\varepsilon,\delta)$-PPKDP), with:

$$\varepsilon = -\ln\left(1 - \frac{f(T,n-1,p)}{1-r}\right)$$
$$\delta = \frac{f(T,n-1,p)}{1-r}$$

For a large n, assuming pn is fixed, we can use the Poisson approximation and get $\delta \approx \frac{(pn)^T e^{-pn}}{(1-r)T!}$. If this quantity is small enough, $\varepsilon \approx \delta$.

If the background knowledge B is not empty, assume that the attacker knows a subset $|B|$ of records. Let $b_{\max}$ be such that $r_b = \frac{p|B|}{(1-p)b_{\max}} < 1$ and $r' = \frac{p(n-|B|-1)}{(1-p)(T-b_{\max})} < 1$. Then $\mathcal{M}_T$ is $(\Theta,\varepsilon,\delta)$-PPKDP, with:

$$\varepsilon = -\ln\left(1 - \frac{f(T-b_{\max},n-|B|-1,p)}{1-r'}\right)$$
$$\delta = \frac{f(b_{\max},|B|,p)}{1-r_b} + \frac{f(T-b_{\max},n-|B|-1,p)}{1-r'}.$$

Proof. The proof is presented in three stages.

1. First, we consider the simpler case where all p_i are equal and there is no background knowledge. This allows us to compute the PLRV exactly, and we can then split the output space into two parts. Most of its mass will be in the $\perp$ event, and we can compute it there exactly. All other events will be captured by δ.

2. Second, we extending this to non-empty partial knowledge in a similar fashion: for some $b_{\max}$, with high probability, the background knowledge will not have more than $b_{\max}$ records whose value is 1: the rest of the probability mass goes in the δ, and this allows us to use the previous idea with a new threshold $T' = T - b_{\max}$.

3. Finally, we use a coupling argument to extend this to the case where the p_i are not all the same.

First, let us compute the PLRV for $\mathcal{M}_T$ depending on the output k and the value of the background knowledge $\hat{B}$, assuming a simple distribution θ where records are i.i.d. Denote by b the number of records in $\hat{B}$ which are 1 and by $\overline{b} = |\hat{B}| - b$ the number of records that are 0. The targeted record will be called $D(t)$, and we assume it is never part of $\hat{B}$. Let θ be a distribution that returns n i.i.d records according to $D(i) \sim Ber(p)$. Then we can directly compute:

$$\mathcal{L}_{i,0,1}^{\mathcal{M}_T,\theta}\left(k,\hat{B}\right) = \ln \frac{\mathbb{P}\left[\mathcal{M}(D) = k \,\middle|\, D(t) = 0, B = \hat{B}\right]}{\mathbb{P}\left[\mathcal{M}(D) = k \,\middle|\, D(t) = 1, B = \hat{B}\right]}$$

$$= \begin{cases} \ln \frac{\sum_{s=0}^{T-b} f(s,n-|B|-1,p)}{\sum_{s=0}^{T-b-1} f(s,n-|B|-1,p)} & \text{if } k = \perp \\ \ln \frac{p(n-\overline{b}-k)}{(1-p)(k-b)} & \text{otherwise.} \end{cases}$$

Note that if $k = b$, then $\mathcal{L}_{i,0,1}^{\mathcal{M}_T,\theta}(k,B) = \infty$. The case where $k < b$ is impossible regardless of $D(i)$: there cannot be more 1s in the background knowledge than the mechanism outputs. In the case where there is no background knowledge, this becomes:

$$\mathcal{L}_{i,0,1}^{\mathcal{M}_T,\theta}(k) = \begin{cases} \ln \frac{\sum_{s=0}^{T} f(s,n-1,p)}{\sum_{s=0}^{T-1} f(s,n-1,p)} & \text{if } k = \perp \\ \ln \frac{p(n-k)}{(1-p)k} & \text{otherwise.} \end{cases}$$

This calculation allows us to bound ε and δ in the simpler case where there is no background knowledge. To do so, we need a technical lemma to bound the probability mass of the tail of the binomial distribution appearing above.

Lemma 1. *For any n, p, and m such that $m > \frac{pn}{1-p}$, we have:*

$$\sum_{s=m}^{n} f(s,n,p) < \frac{f(m,n,p)}{1 - \frac{pn}{(1-p)m}}$$

Proof. Note that for all $s \geq m$:

$$\frac{f(s+1,n,p)}{f(s,n,p)} = \frac{p}{1-p}\left(\frac{n-s}{s+1}\right) < \frac{pn}{(1-p)m}.$$

Since $m > \frac{pn}{1-p}$, this is strictly lower than 1, so the sum converges at least as fast as a geometric series, which directly gives the desired result. □

We can now start proving the main theorem; first in the case where there is no background knowledge and all p_i are equal. There are two possibilities to consider, $\mathcal{L}_{i,1,0}^{\mathcal{M}_T,\theta}$ and $\mathcal{L}_{i,0,1}^{\mathcal{M}_T,\theta}$. First, we have:

$$\begin{aligned} \mathbb{P}_\theta\left[\mathcal{M}_T(D) \neq \perp \,\middle|\, D(i) = 0\right] &< \mathbb{P}_\theta\left[\mathcal{M}_T(D) \neq \perp \,\middle|\, D(i) = 1\right] \\ &= 1 - \sum_{s=0}^{T-1} f(s,n-1,p) \\ &= \sum_{s=T}^{n-1} f(s,n-1,p) \\ &< \frac{f(T,n-1,p)}{1 - \frac{p(n-1)}{(1-p)T}} \end{aligned}$$

since $T > \frac{p(n-1)}{1-p}$, so we can use Lemma 1. Let us denote this quantity as δ. Now, we have

$$\mathcal{L}_{i,1,0}^{\mathcal{M}_T,\theta}(\perp) = \ln \frac{\sum_{s=0}^{T-1} f(s, n-1, p)}{\sum_{s=0}^{T} f(s, n-1, p)} < 0.$$

Furthermore:

$$\begin{aligned}\mathcal{L}_{i,0,1}^{\mathcal{M}_T,\theta}(\perp) &= \ln \frac{\sum_{s=0}^{T} f(s, n-1, p)}{\sum_{s=0}^{T-1} f(s, n-1, p)} \\ &< \ln\left(\frac{1}{1-\sum_{s=T}^{n-1} f(s, n-1, p)}\right) \\ &< -\ln\left(1 - \frac{f(T, n-1, p)}{1-\frac{p(n-1)}{(1-p)T}}\right).\end{aligned}$$

Thus, when the output is thresholded, the PLRV is smaller than $\varepsilon = -\ln\left(1 - \frac{f(T,n-1,p)}{1-r}\right)$, and the event “the output is not thresholded” only happens with a probability smaller than δ, which proves the initial statement in the simpler case.

Now, in the case where the background knowledge is non-empty, we must not only split the output space, but also $\mathcal{B}$ as well. Denoting b the number of “1” entries in B, there are three cases we must consider:

1. $b \geq b_{\max}$: if $b_{\max}$ is large enough, this happens with small probability, which we put in the δ term;
2. $b < b_{\max}$ and $\mathcal{M}_T(D) \neq \perp$: if $T' = T - b_{\max}$ is large enough, this happens with small probability, which we put in the δ;
3. $b < b_{\max}$ and $\mathcal{M}_T(D) = \perp$: this is the event in which most of the probability mass is concentrated on, so we bound its privacy loss to obtain ε.

The probability of the first event can be bounded by:

$$\begin{aligned}\mathbb{P}_\theta\left[b \geq b_{\max}\right] &= \sum_{s=b_{\max}}^{|B|} f(s, |B|, p) \\ &< \frac{f(b_{\max}, |B|, p)}{1-\frac{p|B|}{(1-p)b_{\max}}}\end{aligned}$$

by Lemma 1. Similarly, the probability of the second event can be bounded by:

$$\begin{aligned}\mathbb{P}_\theta\left[\mathcal{M}_T(D) \neq \perp \,\middle|\, b < b_{\max}, D(i) = 1\right] &< \sum_{s=T'}^{n-|B|-1} f(s, n-|B|-1, p) \\ &< \frac{f(T', n-|B|-1, p)}{1-\frac{p(n-|B|-1)}{(1-p)T'}}\end{aligned}$$

so we can bound δ by the sum of those two terms. Now, let us compute the privacy loss for the third case. Assuming $b < b_{\max}$, we have:

$$\mathcal{L}_{i,1,0}^{\mathcal{M}_T,\theta}(\perp, \hat{B}) = \ln \frac{\sum_{s=0}^{T-b-1} f(s, n-|B|-1, p)}{\sum_{s=0}^{T-b} f(s, n-|B|-1, p)} < 0$$

and:

$$\begin{aligned}\mathcal{L}_{i,0,1}^{\mathcal{M}_T,\theta}(\perp, \hat{B}) &= \ln \frac{\sum_{s=0}^{T-b} f(s, n-|B|-1, p)}{\sum_{s=0}^{T-b-1} f(s, n-|B|-1, p)} \\ &< \ln\left(\frac{1}{\sum_{s=0}^{T'-1} f(s, n-|B|-1, p)}\right) \\ &< -\ln\left(1 - \frac{f(T', n-|B|-1, p)}{1-r'}\right)\end{aligned}$$

Denoting this by ε, this proves the theorem in the special case where all p_i are equal to a constant p.

Now, we extend the first case (where the background knowledge is empty) to the case where all p_i are different, and $p_i < p$ for all p. Let us denote θ_p the distribution where all users vote with the same probability p. Let $g(s, n, p) = \mathbb{P}_\theta\left[\mathcal{M}_T(D) = s\right]$. By a simple coupling argument between θ and θ_p, we have for all t:

$$\begin{aligned}\sum_{s=t+1}^{n} g(s, n, p) &= \mathbb{P}_\theta\left[\mathcal{M}_T(D) > t\right] \\ &\leq \mathbb{P}_{\theta_p}\left[\mathcal{M}_T(D) > t\right] \\ &= \sum_{s=t+1}^{n} f(s, n, p).\end{aligned}$$

We can then use this fact throughout the previous proof. The application of this bound for δ is immediate, and for ε, we have:

$$\mathcal{L}_{i,0,1}^{\mathcal{M}_T,\theta}(\perp, \hat{B}) = \ln \frac{\sum_{s=0}^{T} g(s, n-1, p)}{\sum_{s=0}^{T-1} g(s, n-1, p)}.$$

We can bound the numerator by 1 and the denominator expands to:

$$\begin{aligned}\sum_{s=0}^{T-1} g(s, n-1, p) &= 1 - \sum_{s=T}^{n-1} g(s, n-1, p) \\ &\geq 1 - \sum_{s=T}^{n-1} f(s, n-1, p) \\ &= \sum_{s=0}^{T-1} f(s, n-1, p)\end{aligned}$$

so we can reuse the previous bound. The bounds translate to the case where the background knowledge is not empty by a similar argument. □

As shown in Figure 3.4, when the threshold is above the expected value, the values ε and δ given by Theorem 4 are very close. Moreover, for large n, these values are extremely small. This shows that thresholding counts constitutes a good practice, which can be used to meaningfully improve user privacy without having to know about the data distribution in advance, like in the usage statistics example at the beginning of this section.

A practitioner can apply the following reasoning: for each possible sequence of actions captured by the system collecting app usage statistics, either many users are likely to have a value of 1, in which case Theorem 3 applies and thresholding will likely not impact data utility; or the vast majority of users will have a value of 0, in which case Theorem 4 applies and thresholding will protect the rare users whose value is 1.

What if the attacker has non-zero partial knowledge, but is able to interact with the system? We saw in Example 7 that if this partial knowledge is larger than the threshold, the mechanism is not private. But if this partial knowledge is small enough, then privacy is still possible: it is equivalent to reducing the threshold for an attacker with no partial knowledge.

Proposition 28. *Let θ be the same distribution as in Theorem 4, with background knowledge of size $|B| \leq T$. Let θ' be the equivalent distribution but with $n' = n - |B|$, and no partial knowledge. Then for any ε and δ, $\mathcal{M}_T$ is $(\theta, \varepsilon, \delta)$-APKDP iff $\mathcal{M}_{T-|B|}$ is $(\{\theta'\}, \varepsilon, \delta)$-PPKDP.*

Proof. By writing down its explicit value, one can see that $\text{APK}_{i,t,t',B}$ only depends on the difference between T and the number of ones b in the background knowledge B. The same applies for the variables n and $|B|$, which appear only in the form of $n - |B|$. This shows the equivalence of $\mathcal{M}_T$ being $(\{\theta\}, \varepsilon, \delta)$-APKDP and $\mathcal{M}_{T-|B|}$ being $(\{\theta'\}, \varepsilon, \delta)$-APKDP. Since APKDP and PPKDP are the same when there is no background knowledge, the statement follows. □

3.2.3 *Application to k-anonymity*

Sections 3.2.1 and 3.2.2 formalize two intuitive phenomena under partial knowledge. First, if the attacker has a significant enough uncertain about enough people, counting queries do not leak too much information about individuals. Second, for counting queries that apply to rare enough behavior, thresholding provides meaningful protection against a passive attacker. This suggests a link to an older anonymization notion: k-anonymity. In this section, we formalize that link, and combine these two intuitions to provide a relation between k-anonymity and differential privacy under partial knowledge.

k-anonymity, which we introduce in Section 2.1.1, requires each record in a database to be indistinguishable from at least $k-1$ other records. The intuition is that blending in a large enough crowd provides protection; this intuition is close to the results of Section 3.2.1. k-anonymity is generally obtained by generalizing the data to group similar records together, then dropping the groups with less than k records. The link with the results of Section 3.2.2 is obvious.

To formalize it, we need to clarify the notion of a k-anonymity mechanism. For simplicity, we will simply assume that such a mechanism groups records by their value, and returns a truncated histogram, where all values with less than k records have been removed.

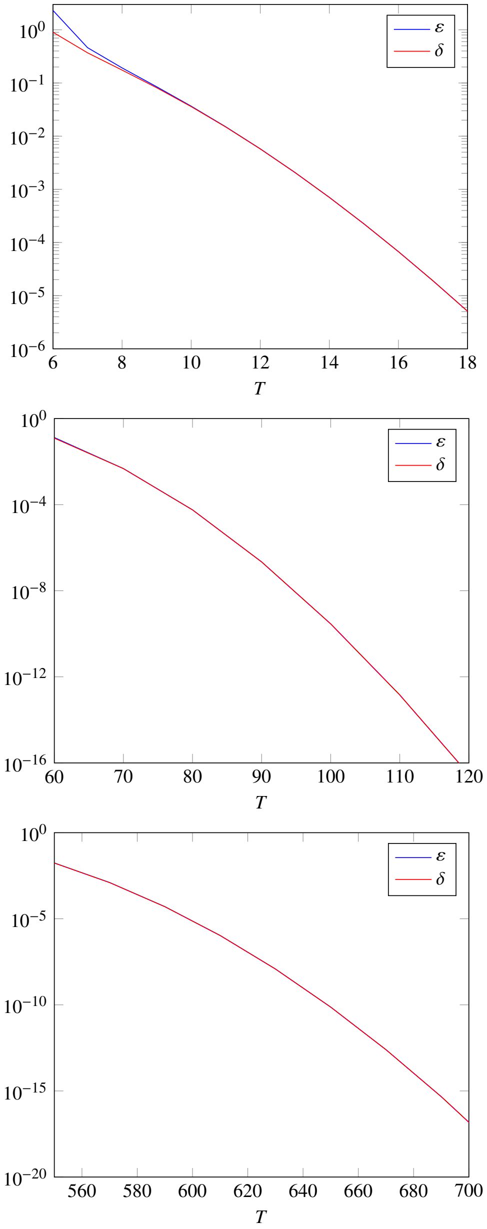


FIGURE 3.4: ϵ and δ from Theorem 4 as a function of the threshold T, where $|B| = 0$, $p = 0.5\%$, and three different values of n: $n = 1000$ (top), $n = 10{,}000$ (middle), and $n = 100{,}000$ (bottom).

Definition 61 (k-anonymity mechanism)**.** *The k-anonymity mechanism $\mathcal{M}_k$ takes a dataset in $\mathcal{D}$ as input, and returns a histogram in $(\mathbb{N} \cup \perp)^{\mathcal{T}}$. For each $t \in \mathcal{T} \cup \{\perp\}$, $\mathcal{M}_k(D)$ is defined as:*

- *for all $t \in \mathcal{T}$, $\mathcal{M}_k(D)(t) = \left|\{i \mid |D(i) = t\}\right|$ if this number is at least k (if there are less than k records with value t in D);*
- *$\mathcal{M}_k(D)(t) = \perp$ otherwise.*

If an input record is not in $\mathcal{T}$, it is ignored by $\mathcal{M}_k$.

Note that we skipped the generalization step. The results below can be easily extended to any *fixed* generalization strategy, i.e. a fixed mapping between $\mathcal{T}$ and an arbitrary space forming the support of the histogram. It is important that this strategy is fixed. If this function depends on the data, arbitrary correlations can be embedded in the output, which might leak additional information; *minimality attacks* [389] provide an example of this phenomenon.

Now, under which condition is such a mechanism private? The distribution that captures the attacker's uncertainty must be such that for all possible values $t \in \mathcal{T}$, either this value is rare enough to be thresholded with high probability, either there is sufficient randomness in the input data that releasing the exact value does not leak too much information.

In addition, we assume that it is possible for a given record to have the value $\perp$, representing their *absence* in the dataset. The count corresponding to $\perp$ are never released. We discuss later the importance of such a special value, and its practical interpretation.

Theorem 5. *Let θ be a distribution that generates n independent records in $\mathcal{T} \cup \{\perp\}$. Assume that there is a λ such that for all $t \in \mathcal{T}$:*

- *either for all indices i, $\mathbb{P}\left[D(i) = t\right] \leq \lambda$,*
- *or for all indices i, $\lambda \leq \mathbb{P}\left[D(i) = t\right] \leq 1 - \lambda$;*

furthermore, assume that for all indices i, $\lambda \leq \mathbb{P}\left[D(i) = \perp\right] \leq 1 - \lambda$, and that the attacker does not have any background knowledge.

Let T be a threshold such that $r = \frac{\lambda(n-1)}{(1-\lambda)k} < 1$. Then $\mathcal{M}_T$ is $(\{\theta\}, \varepsilon, \delta)$-APKDP for all $\delta \geq \delta_0$, where:

$$\delta_0 = \frac{2 \cdot f(T, n-1, \lambda)}{1-r}$$
$$\varepsilon = 2 \cdot \max\left(-\ln\left(1 - \frac{f(T, n-1, \lambda)}{1-r}\right), \varepsilon_c\right)$$

and ε_c is such that $\delta \geq \mathbb{P}\left[\frac{X}{Y} \geq \varepsilon_c\right]$, where X and Y are two independent random variables sampled from a binomial distribution with $n-1$ trials and success probability 2λ.

Proof. For a given index i and a possible record $t \in \mathcal{T}$, we compare the events $D(i) = t$ and $D(i) = \perp$. If we find ε and δ such that $\mathcal{M}(D)_{|D(i)=t} \approx_{\varepsilon,\delta} \mathcal{M}(D)_{|D(i)=\perp}$, then we have $\mathcal{M}(D)_{|D(i)=t} \approx_{2\varepsilon,2\varepsilon} \mathcal{M}(D)_{|D(i)=t}$ for all $t, t' \in \mathcal{T}$, which would conclude the proof immediately.

There are two options that we must consider: either for all indices i, $\mathbb{P}\left[D(i) = t\right] \leq \lambda$, either for all indices i, $\lambda \leq \mathbb{P}\left[D(i) = t\right] \leq 1 - \lambda$.

In the first case, we can reuse the analysis of Section 3.2.2: with probability $1 - \delta_0$, where $\delta_0 = \frac{f(T,n-1,\lambda)}{1-r}$, the result is thresholded, and the corresponding privacy loss is bounded by $\varepsilon_0 = -\ln\left(1 - \frac{f(T,n-1,\lambda)}{1-r}\right)$, following the same reasoning than in the proof of Theorem 4. Importantly, in this case, comparing $D(i) = t$ and $D(i) = \perp$ allows us to restrict our analysis to the value of $\mathcal{M}_T(D)(t)$: the distributions of values of $\mathcal{M}_T(D)(t')$ for all $t' \neq t$ are the same when θ is conditioned on $D(i) = t$ or $D(i) = \perp$.

In the second case, we reuse the analysis of Section 3.2.1: the distribution of $\mathcal{M}_T(D)(t)$ can be seen as the sum of records, each of whom has been randomized using a binary randomized response with parameter 2λ. Since $\mathcal{M}_T(D)(\perp)$ also follows this binary randomized response process, we can directly apply the proof of Theorem 3 with δ_0.

Combining both cases directly leads to the desired result, using the indistinguishability property between $\mathcal{M}_T(D)(t)$ and $\mathcal{M}_T(D)(\perp)$ to get one between arbitrary t and t'. □

Theorem 5 is relatively complex, and depends on a number of conditions. Let us discuss its limitations. Some of them are necessary for the result to be true, others could be overcome with a more careful analysis, at the cost of simplicity.

First, we assume that the attacker has no partial knowledge over the data. The result can easily be extended to the case where the attacker has non-zero *passive* partial knowledge of m records over the data: for the counting case, we can simply remove these m records and obtain the results with $n - m$ instead of m, and for the thresholding case, we can apply Theorem 4 directly. The discussion in Theorem 4 shows that cannot be easily extended to the case where the attacker has the ability to influence the data, unless a very small number of records can be influenced (as in Proposition 28). This captures the correct intuition that k-anonymity is vulnerable against active attackers.

Second, the choice distribution θ might seem artificial, carefully chosen so the previous results can be applied. Why would there be a value λ such that all records have a probability lower than λ of being in a fixed category, or larger than λ? The first option is reasonable: many real-life distributions are long-tailed; some types of actions, or characteristics, are simply very rare. The second option is less natural: maybe a characteristic that is common for many people is extremely rare in others, so requiring *all* records to have a high enough probability for this record seems too restrictive. However, note that this high probability captures the attacker's *uncertainty*: if the attacker knows that some records have a particularly low probability of having a certain record, it is possible to over-approximate this knowledge, and simply consider these records as known by the attacker. We can then use the previous point to still get an upper bound on the attacker's information gain.

Third, what is the meaning of the $\perp$ special case, and is it necessary for the proof of Theorem 5 to work? We use it to prove the desired indistinguishability property in the second case of the proof. Without it, it turns out that subtle problems can arise. Suppose, for example, that $\mathcal{T} = \{a, b, c\}$, and that for all i, $\mathbb{P}\left[D(i) = a\right]$ is infinitesimally small, while $\mathbb{P}\left[D(i) = b\right]$ and $\mathbb{P}\left[D(i) = c\right]$ are both close to 0.5. If the total number of records is fixed (and implicitly assumed to be known by the attacker), note that thresholding the count for a is pointless: with high probability, we can retrieve it by computing the difference between n and the counts for b and c. This phenomenon is a real vulnerability of k-anonymity when the total number of participants is known: any result showing that k-anonymity protects privacy under partial knowledge must find a way of guaranteeing that this does not happen.

Creating an artificial category $\perp$ whose count is never released solves this problem, assuming that this category has sufficient uncertainty. This hides the total number of participants and mitigate this vulnerability. Another way would be to impose that the distribution θ has multiple $t \in \mathcal{T}$ whose counts will likely be thresholded, and that these t together have enough uncertainty to hide the total count. This is also realistic in practice, given that most distributions are long-tailed, but would likely require a more complex analysis, as well as complicate the theorem statement.

Note that a link between k-anonymity and differential privacy was already introduced in [255]. We use the same notion of a k-anonymity mechanism, however, we model the attacker's partial knowledge differently. In [255], the attacker is assumed to know the value of every single record from the original dataset, but not which records have been randomly sampled from it. Arguably, the only way to satisfy that assumption in practice is to have the mechanism *actually* sample the data before applying k-anonymity. In that case, the original differential privacy definition is satisfied. By contrast, our setting assumes an attacker that has some uncertainty about the value of the records themselves; we argue that this is a much more natural way of capturing the natural assumption that the attacker has partial knowledge over the data.

How strong are the privacy parameters provided by Theorem 5 for realistic use cases? First, note that as shown by Figure 3.4, whenever the δ parameter from the thresholding operation is reasonably low, then we have $\varepsilon \approx \delta$, which is much lower than the ε values from the results on the privacy of noiseless aggregations (Section 3.2.1). Thus, to use Theorem 5 in practice, one would supposedly need to:

1. first, fix a target ε and δ that we want to obtain;
2. these privacy parameters give a range of acceptable values of λ, according to Theorem 3; we then select one such value that will serve as a boundary between "rare events" and "common events";
3. finally, calculate the threshold T based on λ and δ, according to Theorem 4.

Admittedly, the above process is not trivial to actually apply, and the constraints on λ make our second point above even more salient: not only is this choice brittle. All in all, our result is an interesting link between two important notions, and formalizes a natural intuition about the inherent privacy of simple aggregation and thresholding mechanisms, but is probably not suited for practical applications. Can it be significantly improved or extended, without adding more brittle assumptions? We leave this as an open question.

3.2.4 *Composition*

Composition theorems enable the modular analysis of complex systems and the continued usage of mechanisms over time. In this section, we study two kinds of composition. *Sequential* composition, introduced in Section 2.1.6.3, and *nested* composition, where post-processing noise is added to the result of the aggregation.

3.2.4.1 *Sequential composition*

We saw in the previous section that noiseless mechanisms could be private under partial knowledge. For such mechanisms, composition does not hold in general. We explain why dependencies between mechanisms are the root cause of composition failing, and we explain how bounding this dependencies allow us to derive usable composition results. First, we show that noiseless composition fails in general.

Example 8. *Going back to the voting example, consider the queries "How many people voted* Yes*?" and "How many people who are not X voted* Yes*?", for some individual X. As shown in Section 3.2.1, each query can be private on its own. However, publishing both results reveals X's vote: the composition of both queries is not private.*

Are there special cases where noiseless counting queries can be composed? In this section, we propose a criterion, (μ, ν)*-boundedness*, under which sequential composition does hold.

The core problem with Example 8 is that the two queries are heavily *dependent* on each other. In fact, knowing the result to the first query only leaves two options for the result of the second query: it drastically reduces an attacker's uncertainty about the second query's result. We show that this dependency between queries is the main obstacle towards a composition result and prove that mechanisms where the dependency is *bounded* (Definition 63) can actually be composed (Theorem 6).

How can we formalize the *bounded dependency* between mechanisms? A natural approach is to quantify how much the additional knowledge of the first mechanism impacts the privacy loss of the second mechanism.

Definition 62. *Given two mechanisms $\mathcal{M}_1$ and $\mathcal{M}_2$, two $t, t' \in \mathcal{T}$, two outputs O_1, O_2, a distribution θ, an index i, a possible value of the background knowledge $\hat{B}$ compatible with $D(i) = t$ and $D(i) = t'$, the* dependency of $\mathcal{M}_2$ on $\mathcal{M}_1$ to distinguish $D(i) = t$ and $D(i) = t'$ *is the function $\mathcal{O}^2 \to \mathbb{R} \cup \{-\infty, \infty\}$ defined by:*

$$Dep_{i,t,t'}^{\mathcal{M}_1,\mathcal{M}_2,\theta}\left(O_1, O_2 \middle| \hat{B}\right) = \mathcal{L}_{i,t,t'}^{\mathcal{M}_2,\theta}\left(O_2, \left(\hat{B}, \mathcal{M}_1(D) = O_1\right)\right) - \mathcal{L}_{i,t,t'}^{\mathcal{M}_2,\theta}\left(O_2, \hat{B}\right)$$

using the convention $\pm\infty - x = \pm\infty$ for all x.

Intuitively, this value quantifies the amount of additional information that $\mathcal{M}_1$ gives the attacker when analyzing the privacy loss of $\mathcal{M}_2$. In Example 8, the first term is $\pm\infty$, as knowing both the results of $\mathcal{M}_1$ and $\mathcal{M}_2$ leaks the value of $D(i)$, while the second term is typically finite. So $\text{Dep}_{i,t,t'}^{\mathcal{M}_1,\mathcal{M}_2,\theta}\left(O_1, O_2 \middle| \hat{B}\right)$ takes infinite values, which captures the fact that the two mechanisms together leak a lot of information.

Bounding this dependency can be done in the same way as using the PLRV to define differential privacy: we bound Dep by μ almost everywhere, and use a small quantity ν to capture rare events where $\text{Dep} > \mu$.

Definition 63 ((μ,ν)-bounded dependency)**.** *Given a family of distributions Θ, two mechanism $\mathcal{M}_1$ and $\mathcal{M}_2$ are (μ,ν)*-bounded dependent for Θ *if for all $\theta \in \Theta$, all indices i and records $t,t' \in \mathcal{T}$, and all $\hat{B} \in \mathcal{B}$:*

$$\mathop{\mathbb{E}}_{\substack{\theta_{|D(i)=t,\hat{B}},\\ O_1 \sim \mathcal{M}_1(D),\\ O_2 \sim \mathcal{M}_2(D)}} \left[\max\left(0, 1 - e^{\mu - Dep_{i,t,t'}^{\mathcal{M}_1,\mathcal{M}_2,\theta}\left(O_1,O_2|\hat{B}\right)}\right)\right]$$

is smaller or equal to ν.

This notion formalizes the intuition that the result of the first mechanism should not impact "too much" the result of the second mechanism. As we show in the following theorem, the dependency of $\mathcal{M}_2$ on $\mathcal{M}_1$ can be used to express the PLRV of the composed mechanism as a function of the PLRV of the two original mechanisms. As a direct consequence, we show that two (μ,ν)-bounded dependent mechanisms can be sequentially composed.

Theorem 6. *Given a distribution θ, two mechanisms $\mathcal{M}_1, \mathcal{M}_2$, an indice i, records $t,t' \in \mathcal{T}$, and $\hat{B} \in \mathcal{B}$, the PLRV of the composed mechanism $\mathcal{M}(D) := (\mathcal{M}_1(D), \mathcal{M}_2(D))$ satisfies:*

$$\mathcal{L}_{i\leftarrow t/i\leftarrow t'}^{\mathcal{M},\theta}\left(O,\hat{B}\right) = 2 \cdot Dep_{i,t,t'}^{\mathcal{M}_1,\mathcal{M}_2,\theta}\left(O_1,O_2\middle|\hat{B}\right) + \mathcal{L}_{i,t,t'}^{\mathcal{M}_1,\theta}\left(O_1,\hat{B}\right) + \mathcal{L}_{i,t,t'}^{\mathcal{M}_2,\theta}\left(O_2,\hat{B}\right).$$

As a corollary, if $\mathcal{M}_1, \mathcal{M}_2$ are (μ,ν)-bounded dependent, if $\mathcal{M}_1$ is $(\Theta,\varepsilon_1,\delta_1)$-APK, and if $\mathcal{M}_2$ is $(\Theta,\varepsilon_2,\delta_2)$-APK for Θ, then $\mathcal{M}$ is $(\Theta, 2\mu+\varepsilon_1+\varepsilon_2, \delta_1+\delta_2+\nu)$-APK.

Proof. Fix i, t, t', O and $\hat{B}$. The main statement is straightforward to prove by decomposing:

$$\begin{aligned}
\mathcal{L}_{i\leftarrow t/i\leftarrow t'}^{\mathcal{M},\theta}(O,\hat{B}) &= \ln \frac{\mathbb{P}_\theta\left[\mathcal{M}(D) = (O_1,O_2) \,\middle|\, D(i) = t, \hat{B}\right]}{\mathbb{P}_\theta\left[\mathcal{M}(D) = (O_1,O_2) \,\middle|\, D(i) = t', \hat{B}\right]} \\
&= \ln \frac{\mathbb{P}_\theta\left[\mathcal{M}_2(D) = O_2 \,\middle|\, \mathcal{M}_1(D) = O_1, D(i) = t, \hat{B}\right]}{\mathbb{P}_\theta\left[\mathcal{M}_2(D) = O_2 \,\middle|\, \mathcal{M}_1(D) = O_1, D(i) = t', \hat{B}\right]} \\
&\quad + \ln \frac{\mathbb{P}_\theta\left[\mathcal{M}_1(D) = O_1 \,\middle|\, D(i) = t, \hat{B}\right]}{\mathbb{P}_\theta\left[\mathcal{M}_1(D) = O_1 \,\middle|\, D(i) = t', \hat{B}\right]}
\end{aligned}$$

and plugging in the definition of $\text{Dep}_{i,t,t'}^{\mathcal{M}_1,\mathcal{M}_2,\theta}\left(O_1,O_2\middle|\hat{B}\right)$.

To prove the composition theorem, we have to show that, if we denote $\varepsilon = \varepsilon_1 + \varepsilon_2 + \mu$ and $\delta = \delta_1 + \delta_2 + \nu$:

$$\mathop{\mathbb{E}}_{\theta_{|D(i)=t,\hat{B}}, O \sim \mathcal{M}(D)} \left[\max\left(0, 1 - e^{\varepsilon - \mathcal{L}_{i\leftarrow t/i\leftarrow t'}^{\mathcal{M},\theta}(O,\hat{B})}\right)\right] \leq \delta.$$

Note that the function $f(x) = \max\left(0, 1 - e^x\right)$ is subadditive: for all x and y, $f(x+y) \leq f(x) + f(y)$. Indeed, if $x > 0$, then $f(x) = 0$, and it is straightforward to verify that $1 - e^{x+y} \leq 1 - e^y$. The same reasoning holds when $y > 0$. Finally, if $x \leq 0$ and $y \leq 0$, then we must show that $1 - e^{x+y} \leq 2 - e^x - e^y$, which is equivalent to $(e^x - 1)(e^y - 1) \geq 0$, which trivially holds.

We then define:

$$\begin{aligned} a &= \varepsilon_1 - \mathcal{L}_{i,t,t'}^{\mathcal{M}_1,\theta}\left(O_1, \hat{B}\right) \\ b &= \varepsilon_2 - \mathcal{L}_{i,t,t'}^{\mathcal{M}_2,\theta}\left(O_2, \hat{B}\right) \\ c &= \mu - \mathrm{Dep}_{i,t,t'}^{\mathcal{M}_1,\mathcal{M}_2,\theta}\left(O_1, O_2 \middle| \hat{B}\right) \end{aligned}$$

and we use this subadditivity property: $f(a+b+c) \leq f(a) + f(b) + f(c)$. As we have $a + b + c = \varepsilon - \mathcal{L}_{i,t,t'}^{\mathcal{M},\theta}\left(O, \hat{B}\right)$, we can directly plug this into the expression above and use the assumptions of the theorem to show that it is bounded by $\delta_1 + \delta_2 + \nu = \delta$. □

Note that the characterization of $\mathcal{L}_{i\leftarrow t/i\leftarrow t'}^{\mathcal{M},\theta}\left(O, \hat{B}\right)$ enables the use of more sophisticated composition bounds for differential privacy, such as the advanced composition theorem [131], Rényi Differential Privacy [284], or privacy buckets [278]. For simplicity, here, we only used the standard (non-tight) composition bound for DP.

A common special case directly leads to $(0,0)$-bounded dependent mechanisms: two mechanisms that work on distinct parts of a database are $(0,0)$-bounded dependent if these two parts are independent.

Proposition 29. *Let Θ be a family of distributions, and let $\mathcal{M}_1$ and $\mathcal{M}_2$ be mechanisms. Assume that for any $\theta \in \Theta$, there are functions π_1 and π_2 such that $\pi_1(D)_{|D\sim\theta}$ and $\pi_2(D)_{|D\sim\theta}$ are independent, and functions $\mathcal{M}_1'$ and $\mathcal{M}_2'$ such that $\mathcal{M}_1(D) = \mathcal{M}_1'(\pi_1(D))$ and $\mathcal{M}_2(D) = \mathcal{M}_2'(\pi_2(D))$. Then $\mathcal{M}_1$ and $\mathcal{M}_2$ are $(0,0)$-bounded dependent.*

We now present an natural example of a practical scenario where we can use these composition results.

Example 9. *Consider a regularly updated database, like usage information about an online service. Statistics q are computed from this database: for example, among registered users, how many of them used a specific feature on any given day. This count is released daily, and we want to understand how the privacy of a particular user is impacted over time.*

This can be represented by a database D where each record i is a series of binary values $(D(i))_j$, where $j = 0, 1, 2, \cdots$, and we release a series of mechanisms $\mathcal{M}_j(D) = \sum_i q(D(i)_j)$. The results of Section 3.2.1 can be used to determine the privacy of each $\mathcal{M}_j$ depending on the data-generating distribution θ. The goal is to determine the privacy of multiple queries, assuming independence between $D(i_1)$ and $D(i_2)$ for all $i_1 \neq i_2$.

The analysis of the privacy guarantees offered by this setting over time depends on θ, and on the correlations between the different values of a record. If $D(i)_{j_1}$ is independent from $D(i)_{j_2}$ for all $j_1 \neq j_2$, then the result is direct. Otherwise, we must quantify the maximum amount of correlation between $D(i)_j$ and $D(i)_{j+1}$. Quantifying this can be done using indistinguishability: we can assume, for example, that there is a $c \geq 0$ such that for all $a \in \mathcal{T}$ and all indices i and j:

$$\left(D(i)_{j+1}\right)_{|D(i)_j = t} \approx_c D(i)_j.$$

Under this assumption, it is easy to verify that mechanisms $\mathcal{M}_j$ and $\mathcal{M}_{j+1}$ are $(2c, 0)$-bounded dependent, so we can use the composition result of Theorem 6 and derive bounds on the privacy leakage over time.

This approach can be extended to other scenarios, for example if only a subset of users participate to each update, or if a referendum contains multiple questions, whose answers are correlated. Another possible scenario is if only a subset of users participate to each update. We can represent this by having $D(i)_j$ be either a categorical value (which encodes e.g. the type of interaction) or a special value $\perp$ that encodes "user i did not participate to this update". The probabilities and correlation relationships of different values associated with the same user can be set to capture different scenarios (e.g. the probability that $D(i)_j = \perp$ can be large, to capture a scenario where few users participate every round).

3.2.4.2 *Nested composition*

The results of Sections 3.2.1 to 3.2.3 show that noiseless mechanisms can be considered private, assuming some additional assumptions on the attacker's background knowledge. With enough records, even pessimistic assumptions (considering an attacker who knows a large fraction of records) can lead to very small values of ε and δ. However, one could still consider these assumptions as too brittle, and decide to add a small amount of additional noise to the mechanism to have it satisfy differential privacy in its original form.

Such mechanisms have a *double* privacy guarantee: under realistic assumptions, their privacy level is very high thanks to the attacker's uncertainty, and the additional noise provides a "worst-case" privacy level that the mechanism satisfies independently of the attacker capabilities. Without noise, we can use results like Theorem 3 to show that a given aggregation over n records is $(\Theta(|B|), \varepsilon(|B|), \delta(|B|))$-APKDP (or PPKDP), where $|B|$ is the number of records that the attacker knows. In situations like ones we have seen so far, $\varepsilon(|B|)$ and $\delta(|B|)$ can be very small when $|B|$ is close to 0, but might become unacceptably high when $|B|$ gets close to n. Adding noise can be a way to guarantee that $\varepsilon(|B|)$ and $\delta(|B|)$ never get above a certain point: when there is not enough randomness coming from the data anymore, the guarantee from post-processing noise take over. Figure 3.5 illustrates this phenomenon.

Of course, it is also natural to wonder whether the two sources of uncertainty could be combined. The privacy guarantees from Theorem 3 come from the shape of the binomial distribution, just like the shape of Laplace noise is the reason why adding it to the result of an aggregation can provide ε-DP. It seems intuitive that combining two sources of noise would have a *larger* effect.

In some cases, this effect can be numerically estimated. Given a noise distribution X added to a mechanism of sensitivity s, the PLRV can be obtained by comparing the distributions of X and $X + s$. To estimate the PLRV coming from two noise sources summed together (for example, binomial and geometric noise), we can simply compute the convolution of the corresponding two distributions, and use the result to compute the PLRV, and thus, the (ε, δ) graph. We demonstrate this approach in Figure 3.6, where we add two-sided geometric noise (see Definition 8) to a noiseless counting query.

It is natural to ask whether we could obtain generic results that quantify the combined effect of noise coming from the input data and noise added after the aggregation mechanism,

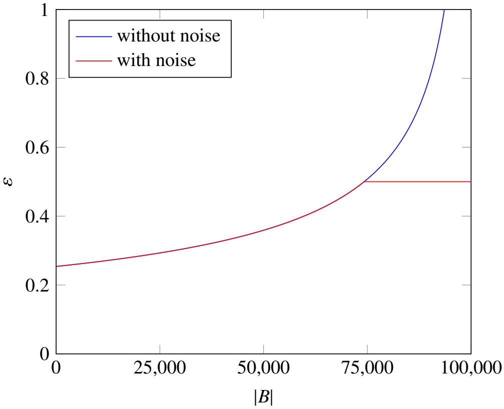


FIGURE 3.5: ε from the closed-form formula of Theorem 3 for $\delta = 10^{-10}$, $\lambda = 0.05$, and $n = 100{,}000$, as a function of the number of records known by the attacker $|B|$. We compare two scenarios: either we do not add any post-processing noise, or we add Laplace noise of scale 2 to the output.

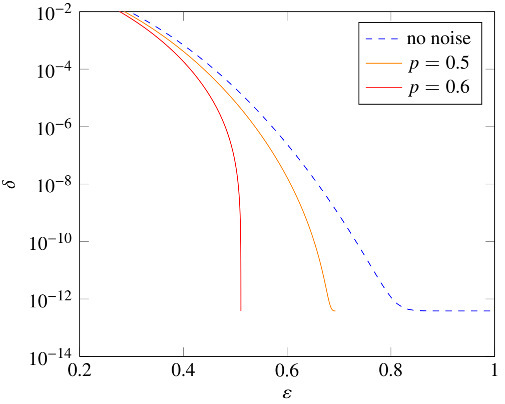


FIGURE 3.6: Numerical computation of the (ε, δ) bounds given by Theorem 3 with $n = 10{,}000$ and $\lambda = 0.05$ (in dashed blue), compared with the bounds obtained by adding two-sided geometric noise of parameter $p = 0.5$ or $p = 0.75$ (see Definition 8) and combining both probability distributions.

without numerical evaluation. In [176], the authors propose such a result, based on the fact that Gaussian distributions are closed under convolution. The noise from the input data is approximated by a Gaussian using the central limit theorem, and their Theorem 6 shows that adding Gaussian noise leads to a smaller ε. However, since the δ term comes from the central limit theorem approximation, it cannot be improved beyond $\delta = O(1/\sqrt{n})$ in general.

We could solve this by simply making the assumption that the input data unknown from the attacker *actually* follows a Gaussian distribution. Sadly, the corresponding result would be very brittle: if an attacker does not conform exactly to this approximation, then the result no longer holds. This is a major criticism of privacy definitions which assume the input data has inherent randomness [350]. The results of this paper are not so brittle, as the privacy guarantees degrade gracefully with the assumptions we make on the attacker's partial knowledge (e.g. the number of records known, or the value of p in Theorems 3 or 4).

Another approach would be choose the noise added as post-processing based on the natural noise distributions emerging from the partial knowledge assumption. For example, since the proof of Theorem 3 uses the fact that the attacker uncertainty corresponds to binomial noise, we could also add binomial noise as post-processing, since $B(n,p) + B(m,p) = B(n+m,p)$. However, this property depends on the exact value of p, which again creates a brittleness we were trying to avoid.

The question of computing the privacy loss in situations where multiple sources of randomness are combined appears in other scenarios. Amplification by sampling or amplification by shuffling are examples of such results. These two classes of results are *generic*: they do not depend on the exact mechanism used to obtain the initial (ε,δ)-DP guarantee. It is unlikely that such generic results exist when combining two arbitrary sources of noise, each of which satisfies (ε,δ)-DP.

Some other results depend on additional assumptions on the noise distribution, like amplification by iteration [149] or amplification by mixing and diffusion mechanisms [25]. These do not seem to bring significant improvements in scenarios like Theorem 3 with post-processing noise: amplification by iteration is tailored for a case where noise is added many times (not only once), while amplification by mixing and diffusion require stronger assumptions on the original noise distribution.

An generic result on the privacy guarantee of chained ε-DP mechanisms appears in [139] (Appendix B). This tight result is only valid for pure ε-DP, but the main building block holds for (ε,δ)-DP mechanisms: proving a fully generic chained composition result is equivalent to solving the special case where the input and output of both mechanisms have values in $\{0,1\}$. This result can likely be extended to the (ε,δ)-DP, although the analysis is surprisingly non-trivial, and fully generic optimality results do not necessarily mean optimality for the special case of additive noise mechanisms.

3.3 LIMITS: CARDINALITY ESTIMATION

So far, we showed that considering an uncertain attacker could lead to *positive* results about the privacy of noiseless aggregations. In this section, we show that this setting can also be used to derive *negative* results: some problems simply cannot be solved in a privacy-preserving way, even under the assumption that the attacker has complete uncertainty over the original data.

The problem we now focus on is *cardinality estimation*. Cardinality estimators, like HyperLogLog, are algorithms that produce a type of output named *sketches*. Sketches can estimate the number of distinct records in a large multiset, and can also be *merged*, ignoring duplicates across sketches. Their widespread use in privacy-sensitive contexts leads to a natural question: can they be made anonymous according to some reasonable privacy notion?

In this section, we show that the answer to this question is negative. We formulate an abstract notion of cardinality estimators, that captures their aggregation requirement: one can merge sketches without losing precision. We show that for all cardinality estimators that satisfy our aggregation requirement, sketches are almost as sensitive as raw data. Indeed, even in a setting where the attacker is assumed to have *zero* background knowledge about the input data, they can still gain significant knowledge about their target by looking at the output of the algorithm.

Our results are a natural consequence of the aggregation properties: to aggregate sketches without counting the same user twice, they must contain information about which users were previously added. The attacker can use this information, even if they do not know any other users in the sketch. Furthermore, adding noise either violates the aggregation property, or has no influence on the success of the attack. Thus, it is pointless to try and design privacy-preserving cardinality estimators: privacy and accurate aggregation are fundamentally incompatible.

This section is organized as follows.

1. In Section 3.3.1, we motivate our work with an example of how cardinality estimators can be used in practice. We formally define an abstract notion of cardinality estimators, and give a weak definition of privacy that is well-suited to how they are used in practice.

2. In Section 3.3.2, we prove that cardinality estimators cannot satisfy this privacy property while maintaining good accuracy asymptotically.

3. In Section 3.3.3, we consider even weaker versions of our initial definition. We show that our results still hold for some variants, while others notions are compatible with accurate cardinality estimation. We then highlight the consequences of our results on practical applications.

4. In Section 3.3.4, we propose and discuss various risk mitigation techniques to use cardinality estimators in practice.

3.3.1 *Preliminaries*

Many data analysis applications must count the number of distinct records in a large stream with repetitions. These applications include network monitoring [142], online analytical processing [309, 346], query processing, and database management [387]. A variety of algorithms, which we call cardinality estimators, have been developed to do this efficiently, with a small memory footprint. These algorithms include PCSA [154], LogLog [120], and

HyperLogLog [152]. They can all be parallelized and implemented using frameworks like MapReduce [95]. Indeed, their internal memory state, called a *sketch*, can be saved, and sketches from different data shards can be aggregated without information loss.

Using cardinality estimators, data owners can compute and store sketches over fine-grained data ranges, for example, daily. Data analysts or tools can then *merge* (or *aggregate*) existing sketches, which enables the subsequent estimation of the total number of distinct records over arbitrary time ranges. This can be done without re-computing the entire sketch, or even accessing the original data.

Among cardinality estimators, HyperLogLog [152] and its variant HyperLogLog++ [195] are widely used in practical data processing and analysis tasks. Implementations exist for many widely used frameworks, including Apache Spark [18], Google BigQuery [48], Microsoft SQL Server [203], and PostgreSQL [318]. The data these programs process is often sensitive. For example, they might estimate the number of distinct IP addresses that connect to a server [369], the number of distinct users that perform a particular action [21], or the number of distinct devices that appeared in a physical location [286]. To illustrate this point, we present an example of a location-based service, which we will use throughout this section.

Example 10. *A location-based service gathers data about the places visited by the service's users. For each place and day, the service stores a sketch counting the identifiers of the users who visited the place that day. This allows the service's owners to compute useful statistics. For example, a data analyst can merge the sketches corresponding to the restaurants in a neighborhood over a month, and estimate how many distinct individuals visited a given restaurant during that month. The cost of such an analysis is proportional to the number of aggregated sketches. If the queries used raw data instead, then each query would require a pass over the entire dataset, which would be much more costly.*

Note that the type of analysis to be carried out by the data analyst may not be known in advance. The data analysts should be able to aggregate arbitrarily many sketches, across arbitrary dimensions such as time or space. The relative precision of cardinality estimation should not degrade as sketches are aggregated.

Fine-grained location data is inherently sensitive: it is extremely reidentifiable [171], and knowing an individual's location can reveal private information. For example, it can reveal medical conditions (from visits to specialized clinics), financial information (from frequent visits to short-term loan shops), relationships (from regular co-presence), sexual orientation (from visits to LGBT community spaces), etc. Thus, knowing where a person has been reveals sensitive information about them.

As the above example suggests, an organization storing and processing location data should implement risk mitigation techniques, such as encryption, access controls, and access audits. The question arises: how should the sketches be protected? Could they be considered sufficiently aggregated to warrant weaker security requirements than the raw data?

To answer this question, we model a setting where a data owner stores sketches for cardinality estimation in a database. The attacker can access some of the stored sketches and any user statistics published, but not the raw data itself. This attacker model captures the *insider risk* associated with personal data collections where insiders of the service provider could gain direct access to the sketch database. In this paper, we use this insider risk scenario as the default attacker model when we refer to the attacker. In the discussion, we also consider a weaker attacker model, modeling an *external attacker* that accesses sketches via an analytics service provided by the data owner. In both cases, we assume that the attacker knows the cardinality estimator's internals.

The attacker's goal is to infer whether some user is in the raw data used to compute one of the accessible sketches. That is, they pick a *target user*, They choose a sketch built from a stream of users, and must guess whether their target is in this stream. The attacker has some *prior knowledge* of whether their target is in the stream, and examining the sketch gives them a *posterior knowledge*. The increase from prior to posterior knowledge determines their knowledge gain.

Consider Example 10. The attacker could be an employee trying to determine whether their partner visited a certain restaurant on a given day, or saw a medical practitioner. The attacker might initially have some suspicion about this (the prior knowledge), and looking at the sketches might increase this suspicion. A small, bounded difference of this suspicion might be deemed to be acceptable, but we do not want the attacker to be able to increase their knowledge too much.

Thus, our goal is to understand whether a realistic attacker could gain significant knowledge about a target by looking at a HyperLogLog sketch; or any other output from a different cardinality estimator.

3.3.1.1 *Related work*

Prior work on privacy-preserving cardinality estimators has been primarily focused on *distributed* user counting, for example to compute user statistics for anonymity networks like Tor. Each party is usually assumed to hold a set X_i of data, or a sketch built from X_i, and the goal is to compute the cardinality of $\bigcup_i X_i$, without allowing the party i to get too much information on the sets X_j with $j \neq i$. The attacker model presumes honest-but-curious adversaries.

In [369], the authors propose a noise-adding mechanism for use in such a distributed context. In [279], each party encrypts their sketch, and sends it encrypted to a tally process, which aggregates them using homomorphic encryption. In [21], the authors propose a multiparty computation protocol based on Bloom filters to estimate cardinality without the need for homomorphic encryption, and in [136], this approach is shown to be vulnerable to attacks, and a more secure variant of the protocol is presented.

Some cardinality estimation protocols who achieve differential privacy have been proposed, based either on Bloom filters [9], FM-sketches [374]. However, these sketches cannot be aggregated without significant accuracy loss, or using secure aggregation methods in a multiparty setting; this last option is explored in [77] using LogLog sketches. Another line of research focuses on counting distinct elements in a scalable way, and making only the *output* differentially private, not the sketches itself [72].

Our attacker model, based on insider risk, is fundamentally different to these models: the same party is assumed to have access to a *large number* of sketches; they must be able to aggregate them and get good estimates. We claim that cardinality estimators must necessarily make a hard choice: the sketches can be made private, or they can be aggregated without catastrophic accuracy degradation, but both properties are mutually exclusive.

Our privacy definition for cardinality estimators is a variant of differential privacy, presented in Definition 6 (Section 2.1.6). It uses a data-generating probability distribution to capture the attacker's uncertainty, in a similar way of variants introduced in Section 2.2.5. We explain in Section 3.3.1.4 why we need a custom privacy definition for our setup and how it relates to differential privacy and Pufferfish privacy (Definition 35, in Section 2.2.5.3). Some deterministic algorithms, which do not add noise, have been shown to preserve privacy under this assumption [36, 46, 176].

Our setting has some superficial similarities to the local differential privacy model, where the data aggregator is assumed to be the attacker. This model is often used to develop privacy-preserving systems to gather statistics [38, 40, 141, 146]. However, the setting and constraints of our work differ fundamentally from these protocols. In local differential privacy, each individual sends data to the server only once, and there is no need for deduplication or intermediate data storage. In our work, the need for intermediate sketches and unique counting leads to the impossibility result.

Finally, some work has been done on the *security* properties of HyperLogLog sketches, in particular their resistance to adversarial manipulation [330], this direction of research is largely orthogonal to the one studied in this work.

3.3.1.2 *Cardinality estimators*

In this section, we formally define cardinality estimators, and prove some of their basic properties. The notations introduced below, and used throughout this section, are summarized on page xxi.

Cardinality estimators estimate the number of distinct records in a stream. The internal state of a cardinality estimator is called a *sketch*. Given a sketch, one can estimate the number of distinct records that have been added to it (the *cardinality*).

Cardinality estimator sketches can also be aggregated: two sketches can be *merged* to produce another sketch, from which we can estimate the total number of distinct records in the given sketches. This aggregation property makes sketch computation and aggregation embarrassingly parallel. The order and the number of aggregation steps do not change the final result, so cardinality estimation can be parallelized using frameworks like MapReduce.

We now formalize the concept of a cardinality estimator. The records of the multiset are assumed to belong to a large but finite set $\mathcal{T}$.

Definition 64. *A* deterministic cardinality estimator *is a tuple* $\langle S_\varnothing, \text{add}, \text{estimate}\rangle$*, where:*

- $S_\varnothing$ *is the* empty sketch*;*
- $\text{add}\left(S, t\right)$ *is the deterministic operation that adds the record t to the sketch S and returns an updated sketch;*
- $\text{estimate}\left(S\right)$ *estimates the number of distinct records that have been added to the sketch.*

Furthermore, the add *operation must satisfy the following axioms for all sketches S and records $t, t_1, t_2 \in \mathcal{T}$.*

1. Idempotence*:* $\operatorname{add}(\operatorname{add}(S,t),t) = \operatorname{add}(S,t)$.

2. Commutativity*:* $\operatorname{add}(\operatorname{add}(S,t_1),t_2) = \operatorname{add}(\operatorname{add}(S,t_2),t_1)$.

These axioms state that add ignores duplicates and that the order in which records are added is immaterial. Ignoring duplicates is a natural requirement for cardinality estimators. Ignoring order is required for this operation to be used in frameworks like MapReduce, or open-source equivalents like Hadoop or Apache Beam. Since handling large-scale datasets typically requires using such frameworks, we consider commutativity to be a hard requirement for cardinality estimators.

We denote by $S_{t_1,\ldots,t_n}$ the sketch obtained by adding $t_1,\ldots,t_n$ successively to $S_\varnothing$, and we denote by $\mathcal{S}$ the set of all sketches that can be obtained inductively from $S_\varnothing$ by adding records from any subset of $\mathcal{T}$ in any order. Note that $\mathcal{S}$ is finite and of cardinality at most $2^{|\mathcal{T}|}$. Order and multiplicity do not influence sketches: we denote by S_E the sketch obtained by adding all records of a set E (in any order) to $S_\varnothing$.

Later, we give several examples of deterministic cardinality estimators, all of which satisfy Definition 64.

Lemma 2. $\operatorname{add}(S_{t_1,\ldots,t_n},t_i) = S_{t_1,\ldots,t_n}$ *for all* $i \leq n$.

Proof. This follows directly from the idempotence and commutativity properties. □

In practice, cardinality estimators also have a merge operation. We do not explicitly require the existence of this operation, since add's idempotence and commutativity ensure its existence as follows.

Definition 65. *To merge two sketches* S *and* S'*, choose some* $E = \{t_1,\ldots,t_n\} \subseteq \mathcal{T}$ *such that* $S' = S_E$*. We define* merge (S,S') *to be the sketch obtained after adding all records of* E *successively to* S*.*

Note that this construction is not computationally tractable in general, even though in practical scenarios, the merge operation must be fast. This efficiency requirement is not necessary for any of our results, so we do not explicitly require it either.

Lemma 3. *The* merge *operation in Definition 65 is well-defined, i.e., it does not depend on the choice of* E*. Furthermore, the merge operation is a commutative, idempotent monoid on* $\mathcal{S}$ *with* $S_\varnothing$ *as neutral record.*

Proof. Let S and S' be two sketches. If $E = \{t_1,\ldots,t_k\}$, we denote by $\operatorname{add}(S,E)$ the result of adding records of E successively to S:

$$\operatorname{add}(S,E) = \operatorname{add}(\ldots\operatorname{add}(\operatorname{add}(S,t_1),t_2)\ldots,t_k).$$

Let t_1 and t_2 be two sets such that $S_{t_1} = S_{t_2} = S'$, and let E be a set such as $S_E = S$. Then $\operatorname{add}(S,t_1) = \operatorname{add}(\operatorname{add}(S_\varnothing,E),t_1)$. Using the two properties of the add function, we get $\operatorname{add}(S,t_1) = \operatorname{add}(\operatorname{add}(S_\varnothing,t_1),E) = \operatorname{add}(S',E)$. The same reasoning leads to $\operatorname{add}(S,t_2) = \operatorname{add}(S,t_1)$: the merge function does not depend on the choice of E.

In addition, note that $\operatorname{merge}(S,S') = \operatorname{add}(S_\varnothing, E \cup t_1)$. Thus, commutativity, associativity, idempotence, and neutrality follow directly from the same properties of set union. □

Note that these properties of the merge operation are important for cardinality estimators: when aggregating different sketches, we must ensure that the result is the same no matter in which order the sketches are aggregated.

Existing cardinality estimators also satisfy efficiency requirements: they have a low memory footprint, and add and merge run in constant time. These additional properties are not needed for our results, so we omit them in our definition.

We now define *precise* cardinality estimators.

Definition 66. *Let E be a set of cardinality n taken uniformly at random in $\mathcal{T}$. The quality of a cardinality estimation algorithm is given by two metrics:*

1. *Its* bias $\mathbb{E}\left[n - \text{estimate}\left(S_E\right)\right]$
2. *Its* variance $\mathbb{V}_n = \mathbb{E}\left[\left(\mathbb{E}\left[\text{estimate}\left(S_E\right)\right] - \text{estimate}\left(S_E\right)\right)^2\right]$.

Typically, cardinality estimators used for practical applications are asymptotically unbiased: $\frac{\mathbb{E}[n - \text{estimate}(S_E)]}{n} = o(1)$. In the rest of this work, we assume that all cardinality estimators we consider are perfectly unbiased, so $\mathbb{V}_n = \mathbb{E}\left[\left(n - \text{estimate}\left(S_E\right)\right)^2\right]$. Cardinality estimators are often compared by their relative standard error (RSE), which is given by:

$$\sqrt{\mathbb{E}\left[\left(\frac{n - \text{estimate}\left(S_E\right)}{n}\right)^2\right]} = \frac{\sqrt{\mathbb{V}_n}}{n}.$$

A cardinality estimator is said to be precise *if it is asymptotically unbiased and its relative standard error is bounded by a constant. In practice, we want the relative standard error to be less than a reasonably small constant, for example less than 10%.*

Let us give a few examples of cardinality estimators, with their *memory usage* in bits ($m = \ln_2 |\mathcal{S}|$) and their relative standard error. As a first step, they all apply a hash function to the user identifiers. Conceptually, this step assigns to all users probabilistically a random bitstring of length 32 or 64 bits. Since hashing is deterministic, all occurrences of a user are mapped to the same bitstring.

- *K-Minimum Values* [29], with parameter k, maintains a list of the k smallest hashes that have been added to the sketch. With 32-bit hashes, it has a memory usage of $m = 32 \cdot k$, and its unbiased estimator has a RSE of approximately $\frac{1}{\sqrt{k}} \simeq \frac{5.66}{\sqrt{m}}$ [45].
- Probabilistic Counting with Stochastic Averaging [154] (PCSA, also known as FM-sketches) maintains a list of k *bit arrays*. When an record is added to an FM-sketch, its hash is split into two parts. The first part determines which bit array is modified. The second part determines which bit is flipped to 1, depending on the number of consecutive zeroes at the beginning. With k registers and 32-bit hashes, its memory usage is $m = 32 \cdot k$, and its RSE is approximately $\frac{0.78}{\sqrt{k}} \simeq \frac{4.41}{\sqrt{m}}$.
- LogLog [120] maintains a tuple of k registers. Like PCSA, it uses the first part of the hash to pick which register to modify. Then, each registers stores the *maximum* number of consecutive zeroes observed so far. Using $k \geq 64$ registers and a 32-bit hash, its memory usage is $m = 5 \cdot k$ bits, and its standard error is approximately $\frac{1.30}{\sqrt{k}} \simeq \frac{2.91}{\sqrt{m}}$.

- HyperLogLog [195] has the same structure and add operation as LogLog, only its estimate operation is different. Using $k \geq 16$ registers and a 64-bit hash, it has a memory usage of $m = 6 \cdot k$ bits, and its standard error is approximately $\frac{1.04}{\sqrt{k}} \simeq \frac{2.55}{\sqrt{m}}$.
- Bloom filters can also be used for cardinality estimation [310]. However, we could not find an expression of the standard error for a given memory usage in the literature.

All these cardinality estimators have null or negligible ($\ll 1$) bias. Thus, their variance is equal to their mean squared standard error. So the first four are precise, whereas we do not know if Bloom filters are.

All examples above are deterministic cardinality estimators. For them and other deterministic cardinality estimators, bias and variance only come from the randomness in the algorithm's inputs. We now define *probabilistic* cardinality estimators. Intuitively, these are algorithms that retain all the useful properties of deterministic cardinality estimators, but may flip coins during computation. We denote by $\mathfrak{S}$ the set of distributions over $\mathcal{S}$.

Definition 67. *A* probabilistic cardinality estimator *is a tuple* $\langle S_{\varnothing}, \text{add}, \text{merge}, \text{estimate}\rangle$, *where*

- $S_{\varnothing} \in \mathcal{S}$ *is the* empty sketch*;*
- add $(S, t) : \mathcal{S} \times \mathcal{T} \longrightarrow \mathfrak{S}$ *is the probabilistic operation that adds the record e to the sketch S and returns an updated sketch;*
- merge $(S_1, S_2) : \mathcal{S} \times \mathcal{S} \longrightarrow \mathfrak{S}$ *is the probabilistic operation that merges two sketches* S_1 *and* S_2*; and*
- estimate $(S) : \mathcal{S} \longrightarrow \mathbb{N}$ *estimates the number of unique records that have been added to the sketch.*

Both the add *and* merge *operations can be extended to* distributions *of sketches. For a distribution of sketches* σ *and an record e,* add (σ, t) *denotes the distribution such that:*

$$\mathbb{P}\left[\text{add}\left(\sigma, t\right) = S_0\right] = \sum_{M} \mathbb{P}\left[\sigma = S\right] \mathbb{P}\left[\text{add}\left(S, t\right) = S_0\right].$$

For two distributions of sketches σ *and* σ', merge (σ, σ') *denotes the distribution such that:*

$$\mathbb{P}\left[\text{merge}\left(\sigma, \sigma'\right) = S_0\right] = \sum_{S, S'} \mathbb{P}\left[\sigma = S\right] \mathbb{P}\left[\sigma' = S'\right] \mathbb{P}\left[\text{merge}\left(S, S'\right) = S_0\right].$$

We want probabilistic cardinality estimators to have the same high-level properties as deterministic cardinality estimators: idempotence, commutativity, and the existence of a well-behaved merge *operation. In the deterministic case, the idempotence and commutativity of the* add *operation was sufficient to show the existence of a* merge *operation with the desired properties. In the probabilistic case, this no longer holds. Instead, we require the following two properties.*

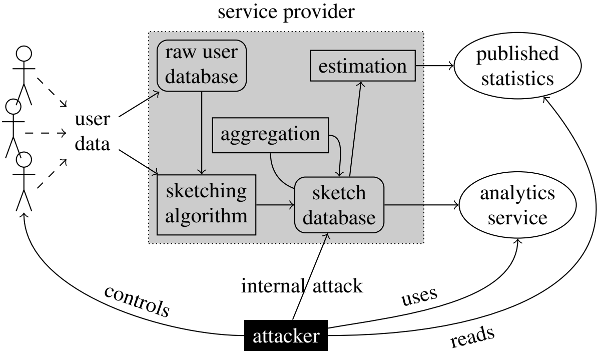


FIGURE 3.7: System and attacker model

- *For a set $E \subseteq \mathcal{T}$, let σ_E denote the sketch distribution obtained when adding records of E successively into $S_\varnothing$. The mapping from E to σ_E must be well-defined: it must be independent of the order in which we add records, and possible repetitions. This requirement encompasses both idempotence and commutativity.*

- *For two subsets E_1 and E_2 of $\mathcal{T}$, we require that:*

$$\text{merge}\left(\sigma_{E_1}, \sigma_{E_2}\right) = \sigma_{E_1 \cup E_2}.$$

These requirements encompass the results of Lemma 3.

These properties, like in the deterministic case, are very strong. They impose that an arbitrary number of sketches can be aggregated without losing accuracy during the aggregation process. This requirement is however realistic in many practical contexts, where the same sketches are used for fine-grained analysis and for large-scale cardinality estimation. If this requirement is relaxed, and the cardinality estimator is allowed to return imprecise results when merging sketches, our negative results do not hold.

For example, Tschorsch and Scheuermann proposed a cardinality estimation scheme [369] which adds noise to sketches to make them satisfy privacy guarantees in a distributed context. However, their algorithm is not a probabilistic cardinality estimator according to our definition: noisy sketches can no longer be aggregated. Indeed, [369] explains that "combining many perturbed sketches quickly drives [noise] to exceedingly high values." In our setting, aggregation is crucial, so we do not further consider their algorithm.

3.3.1.3 *Privacy for cardinality estimators*

In this section, we describe our system and attacker model, and present the privacy definition we use in our setting.

Figure 3.7 shows our system and attacker model. The service provider collects sensitive data from many users over a long time span. The raw data is stored in a database. Over shorter time periods (e.g. an hour, a day, or a week), a cardinality estimator aggregates all data into a sketch. Sketches of all time periods are stored in a sketch database. Sketches from different times are aggregated into sketches of longer time spans, which are also stored in the database. Estimators compute user statistics from the sketches in the database, which are published. The service provider may also publish the sketches via an analytics service for other parties.

The attacker knows all algorithms used (those for sketching, aggregation, and estimation, including their configuration parameters such as the hash function and the number of buckets) and has access to the published statistics and the analytics service. They control a small fraction of the users that produce user data. However, they can neither observe nor change the data of the other users. They also do not have access to the database containing the raw data.

In this work, we mainly consider an *internal attacker* who has access to the sketch database. For this internal attacker, the goal is to discover whether their target belongs to a given sketch. We then discuss how our results extend to weaker *external attackers*, which can only use the analytics service. We will see that for our main results, the attacker only requires access to one sketch. The possibility to use multiple sketches will only come up when discussing mitigations strategies in Section 3.3.4.1.

3.3.1.4 *Privacy definition*

We now present the privacy definition used in our main result. Given the system and attacker just described, our definition captures the impossibility for the attacker to gain significant positive knowledge about a given target. This privacy requirement is very *weak*: reasonable definitions used for practical algorithms would likely be stronger. Working with a weak definition *strengthens* our negative result: if a cardinality estimator satisfying our weak privacy definition cannot be precise, then this is also the case for cardinality estimators satisfying a stronger definition.

Definition 68. *A cardinality estimator satisfies* ε-sketch privacy above cardinality *N if for every $n \geq N$, $t \in \mathcal{T}$, and $S \in \mathcal{S}$, the following inequality holds:*

$$\mathbb{P}_n \left[S_E = S \mid t \in E \right] \leq e^{\varepsilon} \cdot \mathbb{P}_n \left[S_E = S \mid t \notin E \right].$$

Here, the probability $\mathbb{P}_n$ is taken over:

- *a uniformly chosen set $E \in \mathcal{P}_n(\mathcal{T})$, where $\mathcal{P}_n(\mathcal{T})$ is the set of all possible subsets $E \subseteq \mathcal{T}$ of cardinality n; and*
- *the coin flips of the algorithm, for probabilistic cardinality estimators.*

This notion is a good example of the conclusion of Section 2.2: multiple variants and extensions of differential privacy are combined to find a definition that fits exactly to a specific use case. It has three important characteristics.

1. *No background knowledge*: this definition assumes an attacker that has *total* uncertainty over the data: no record is known, and the input distribution is assumed to be completely uniform. The absence of any partial knowledge makes it a special case of definitions from Section 2.2.5; in Section 3.1.3, we saw that in this specific case, APK-DP and

PPK-DP are equivalent. In practice, a realistic attacker might have more information about the data, so a stronger privacy definition would model this prior knowledge by a larger family of probability distributions. More precisely, since the records of E are uniformly distributed in $\mathcal{T}$, the prior knowledge from the attacker $\mathbb{P}_n\left[t \in E\right]$ is exactly $|E| / |\mathcal{T}|$. A realistic attacker would likely have a larger prior knowledge about their target. However, any reasonable definition of privacy would also include the case where the attacker does not have more information on their target than on other users and, as such, would be stronger than ϵ-sketch privacy.

2. *Asymmetry*: we only consider the *positive* information gain by the attacker. Following a reasoning to the prior-posterior bound property of DP shown in Section 2.1.6.2, ε-sketch privacy imposes an *upper bound* on the probability that notion of knowledge gain is similar to the one in $t \in E$ given the observation S, but no lower bound. In other words, the attacker is allowed to deduce with absolute certainty that $t \notin E$. In practice, both positive *and* negative information gains may present a privacy risk. In our running example (see Example 10), deducing that a user did *not* spend the night at their place of residence could be problematic.

3. *Minimum cardinality*: we only require a bound on the information gain for cardinalities larger than a parameter N. In practice, N could represent a threshold over which it is considered safe to publish sketches or to relax data protection requirements. Choosing a small N (like $N = 10$) strengthens the privacy definition, while choosing a large N (like $N = 500$) limits the utility of the data, as many smaller sketches cannot be published.

The first characteristic is a variant of DP alongside our dimension **B** (Section 2.2.5), while the former two are reducing the scope of the definition in a similar way than variants from dimension **N** (Section 2.2.3). Later, in Section 3.3.3, we consider even weaker privation notions, by combining our definition with variants from dimensions **Q** (Section 2.2.2) and **V** (autorefsec:v).

ε-sketch privacy is trivially strictly weaker than differential privacy, and can almost be expressed as an instance of pufferfish privacy (Definition 35), except its asymmetry. It is immediate to show that ε-sketch privacy satisfies both privacy axioms (Definition 9).

We emphasize again that these characteristics, which result in a very weak definition, make our notion of privacy well-suited to proving *negative results*. If satisfying our definition is impossible for an accurate cardinality estimator, then a stronger definition would similarly be impossible to satisfy. For example, any reasonable choice of distributions used to represent the prior knowledge of the attacker would include the uniform distribution.

3.3.2 *Private cardinality estimators are imprecise*

Let us return to our privacy problem: someone with access to a sketch wants to know whether a given individual belongs to the aggregated individuals in the sketch. Formally, given a target t and a sketch S_E, the attacker must guess whether $t \in E$ with high probability. In Section 3.3.2.1, we explain how the attacker can use a simple test to gain significant information

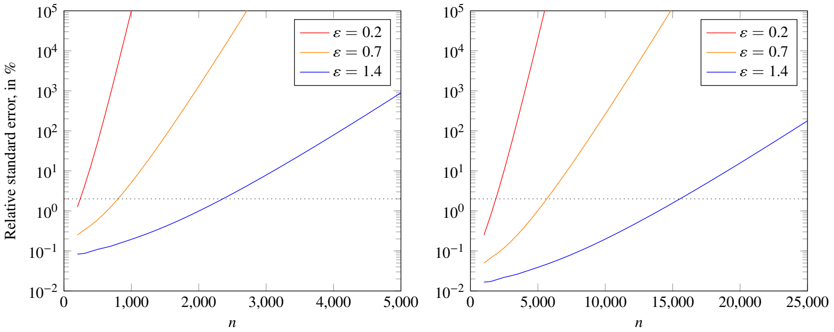


FIGURE 3.8: Minimum standard error for a cardinality estimator with ε-sketch privacy above cardinality 100 (left) and 500 (right). The dotted line is the relative standard error of HyperLogLog with standard parameters.

if the cardinality estimator is deterministic. Then, in Section 3.3.2.2, we reformulate the main technical lemma in probabilistic terms, and prove an equivalent theorem for probabilistic cardinality estimators.

3.3.2.1 *Deterministic case*

Given a target t and a sketch S_E, the attacker can perform the following simple attack to guess whether $t \in E$. They can try to add the target t to the sketch S_E, and observe whether the sketch changes. In other words, they check whether $\text{add}(S_E, t) = S_E$. If the sketch changes, this means with certainty that $t \notin E$. Thus, Bayes' law indicates that if $\text{add}(S_E, t) = S_E$, then the probability of $t \in E$ cannot decrease.

How large is this increase? Intuitively, it depends on how likely it is that adding an record to a sketch does not change it *if the record has not previously been added to the sketch*. Formally, it depends on $\mathbb{P}\left[\text{add}(S_E, t) = S_E \,\middle|\, t \notin E\right]$.

- If $\mathbb{P}\left[\text{add}(S_E, t) = S_E \,\middle|\, t \notin E\right]$ is close to 0, for example if the sketch is just a list of all records seen so far, then observing that $\text{add}(S_E, t) = S_E$ will lead the attacker to believe with high probability that $t \in E$.
- If $\mathbb{P}\left[\text{add}(S_E, t) = S_E \,\middle|\, t \notin E\right]$ is close to 1, it means that adding an record to a sketch often does not change it. The previous attack does not reveal much information. But then, it also means that many records are ignored when they are added to the sketch, that is, the sketch does not change when adding the record. Intuitively, the accuracy of an estimator based solely on a sketch that ignores many records cannot be very good.

We formalize this intuition in the following theorem.

Theorem 7. *An unbiased deterministic cardinality estimator that satisfies ε-sketch privacy above cardinality N is not precise. Namely, its variance is at least $\frac{1-c^k}{c^k}(n - k \cdot N)$, for any $n \leq N$ and $k \leq \frac{n}{N}$, where $c = 1 - e^{-\varepsilon}$*

Proof. The proof is comprised of three steps, following the intuition previously given.

1. We show that a sketch S_E, computed from a random set E with an ε-sketch private estimator above cardinality N, will ignore many records after N (Lemma 4).

2. We prove that if a cardinality estimator ignores a certain ratio of records after adding $n = N$ records, then it will ignore an even larger ratio of records as n increases (Lemma 5).

3. We conclude by proving that an unbiased cardinality estimator that ignores many records must have a large variance (Lemma 6).

The theorem follows directly from these lemmas. □

Note that if we were using differential privacy, this result would be trivial: no deterministic algorithm can ever be differentially private. However, this is not so obvious for our definition of privacy: prior work [36, 46, 176] as well as the results in Section 3.2 show that when the attacker is assumed to have some uncertainty about the data, even deterministic algorithms can satisfy the corresponding definition of privacy.

Figure 3.8 shows plots of the lower bound on the standard error of a cardinality estimator with ε-sketch privacy at two cardinalities (100 and 500). It shows that the standard error increases exponentially with the number of records added to the sketch. This demonstrates that even if we require the privacy property for a large value of N (500) and a relatively large ε, the standard error of a cardinality estimator will become unreasonably large after 20,000 records.

Lemma 4. *Let $t \in \mathcal{T}$. A deterministic cardinality estimator with ε-sketch privacy above cardinality N satisfies $\mathbb{P}_n\left[\text{add}\left(S_E, t\right) = S_E \;\middle|\; t \notin E\right] \geq e^{-\varepsilon}$ for $n \geq N$.*

Proof. We first prove that such an estimator also satisfies

$$\mathbb{P}_n\left[\text{add}\left(S_E, t\right) = S_E \;\middle|\; t \in E\right] \leq e^{\varepsilon} \cdot \mathbb{P}_n\left[\text{add}\left(S_E, t\right) = S_E \;\middle|\; t \notin E\right].$$

We decompose the left-hand side of the inequality over all possible values of S_E which that $\text{add}\left(S_E, t\right) = S_E$. If we abbreviate this set $\mathcal{I}_t = \left\{M \;\middle|\; \text{add}\left(S, t\right) = S\right\}$, we have:

$$\begin{aligned}\mathbb{P}_n\left[\text{add}\left(S_E, t\right) = S_E \;\middle|\; t \in E\right] &= \sum\nolimits_{S \in \mathcal{I}_t} \mathbb{P}_n\left[S_E = S \mid t \in E\right] \\ &\leq e^{\varepsilon} \cdot \sum\nolimits_{S \in \mathcal{I}_t} \mathbb{P}_n\left[S_E = S \mid t \notin E\right] \\ &\leq e^{\varepsilon} \cdot \mathbb{P}_n\left[\text{add}\left(S_E, t\right) = S_E \;\middle|\; t \notin E\right],\end{aligned}$$

where the first inequality is obtained directly from the definition of ε-sketch privacy.

Now, Lemma 2 gives $\mathbb{P}_{\text{add}(S_E,t)=S_E}\left[t \in E\right] = 1$, and finally:

$$\mathbb{P}_n\left[\text{add}\left(S_E, t\right) = S_E \;\middle|\; t \notin E\right] \geq e^{-\varepsilon}.$$

□

Lemma 5. *Let $t \in \mathcal{T}$. Suppose a deterministic cardinality estimator satisfies*

$$\mathbb{P}_n\left[\text{add}\left(S_E, t\right) = S_E \,\middle|\, t \notin E\right] \geq p$$

for any $n \geq N$. Then for any integer $k \geq 1$, it also satisfies $\mathbb{P}_n\left[\text{add}\left(S_E, t\right) = S_E \,\middle|\, t \notin E\right] \geq 1 - (1-p)^k$, for $n \geq k \cdot N$.

Proof. First, note that if $F \subseteq E$, and $\text{add}\left(S_F, t\right) = S_F$, then $\text{add}\left(S_E, t\right) = S_E$. This is a direct consequence of Lemma 3: $S_E = \text{merge}\left(S_{E \backslash F}, S_F\right)$, so:

$$\begin{aligned}\text{add}\left(S_E, t\right) &= \text{merge}\left(S_{E\backslash F}, \text{add}\left(S_F, t\right)\right) \\ &= \text{merge}\left(S_{E\backslash F}, S_F\right) \\ &= S_E\end{aligned}$$

We show next that when $n \geq k \cdot N$, generating a set $E \in \mathcal{P}_n\left(\mathcal{T}\right)$ uniformly randomly can be seen as generating k *independent* sets in $\mathcal{P}_N\left(\mathcal{T}\right)$, then merging them. Indeed, generating such a set can be done by as follows:

1. For each $i \in \{1, \ldots, k\}$, generate a set $E_i \subseteq \mathcal{P}_N\left(\mathcal{T}\right)$ uniformly randomly, then take the union of all these sets: $E_\cup = \bigcup_i E_i$.

2. If some records appear in multiple E_i, the total cardinality might be lower than n, so we need add records uniformly at random to reach the desired cardinality: calculate $d = |E_\cup|$, then generate a set $E' \in \mathcal{P}_{n-d}\left(\mathcal{T} \backslash E_\cup\right)$ uniformly randomly.

3. Add the missing items to complete the set: $E = E_\cup \cup E'$.

Step 1 ensures that we used k *independent* sets of cardinality N to generate E, and step 2 and 3 ensure that E has exactly n records.

Intuitively, each time we generate a set E_i of cardinality N uniformly at random in $\mathcal{T}$, we have *one chance* that t will be ignored by E_i (and thus by E). So t can be ignored by S_E with a certain probability because it was ignored by S_{E_1}. Similarly, it can also be ignored because of S_{E_2}, etc. Since the choice of E_i is independent of the choice of records in $\bigcup_{j \neq i} E_j$, we can rewrite:

$$\begin{aligned}\mathbb{P}_n\left[\text{add}\left(S_E, t\right) \neq S_E \,\middle|\, t \notin E\right] &\leq \prod_{i=1}^{k} \mathbb{P}_n\left[\text{add}\left(S_{E_i}, t\right) \neq S_{E_i} \,\middle|\, t \notin E\right] \\ &\leq \prod_{i=1}^{k}\left(1 - \mathbb{P}_n\left[\text{add}\left(S_{E_i}, t\right) = S_{E_i} \,\middle|\, t \notin e_i\right]\right) \\ &\leq (1-p)^k\end{aligned}$$

using the hypothesis of the lemma. Thus:

$$\mathbb{P}_n\left[\text{add}\left(S_E, t\right) = S_E \,\middle|\, t \notin E\right] \geq 1 - (1-p)^k.$$

□

Lemma 6. *Suppose a deterministic cardinality estimator satisfies:*

$$\mathbb{P}_n\left[\text{add}\left(S_E, t\right) = S_E \;\middle|\; t \notin E\right] \geq 1 - p$$

for any $n \geq N$ and all t. Then its variance for $n \geq N$ is at least $\frac{1-p}{p}\left(n - N\right)$.

Proof. The proof's intuition is as follows. The hypothesis of the lemma requires that the cardinality estimator, on average, *ignores* a proportion $1 - p$ of new records added to a sketch (once N records have been added): the sketch is not changed when a new record is added. The best thing that the cardinality estimator can do, then, is to store all records that it does not ignore, count the number of unique records among these, and multiply this number by $1/p$ to correct for the records ignored. It is well-known that estimating the size k of a set based on the size of a uniform sample of sampling ratio p has a variance of $\frac{1-p}{p}k$. Hence, our cardinality estimator has a variance of at least $\frac{1-p}{p}\left(n - N\right)$.

Formalizing this idea requires some additional technical steps, and the proof is decomposed into three steps. In Step 1, we fix the first N records added to the sketch, and we bound the variance of the estimator that has these records as initial input. In Step 2, we explicitly compute the probability for an record t to be ignored by S_E, first when t is fixed, then when E is fixed. We then use these results in Step 3 to average the bound over all possible choices for the N records in E, which gives us an overall bound.

Step 1: Bound the estimator variance.

Let $E \in \mathcal{P}_N\left(\mathcal{T}\right)$. Let $X_E = \left\{t \in \mathcal{T} \;\middle|\; \text{add}\left(S_E, t\right) \neq S_E\right\}$ be the set of records that are *not* ignored by S_E. Let $p_E = \frac{|X_E|}{|\mathcal{T}|}$, or equivalently, let $p_E = \mathbb{P}_t\left[\text{add}\left(S_E, t\right) \neq S_E \;\middle|\; t \notin E\right]$, where $\mathbb{P}_t$ is the distribution that picks t uniformly randomly in $\mathcal{T}$.

X_E can be seen as the *sampling set* of the estimator with E as initial input, while p_E can be seen as its *sampling fraction*: all other records are discarded, so the estimator only has access to records in E and X_E to compute its estimate.

The optimal estimator to minimize variance in this context simply counts the exact number of distinct records in the sample (remembering each one), and divides this number by p_E to estimate the total number of distinct records.[2]

What is the variance $\mathbb{V}_{n|E}$ of this optimal estimator? Suppose we added $n - N$ records after reaching the sampling part. The number of records in the sample is a random variable with variance $p_E\left(1 - p_E\right)\left(n - N\right)$. Dividing this random variable by p_E gives a variance of $\frac{1-p_E}{p_E} \cdot \left(n - N\right)$. Thus:

$$\mathbb{V}_{n|E} \geq \frac{1 - p_E}{p_E} \cdot \left(n - N\right).$$

Since the first N records are chosen uniformly at random, the variance of the overall estimator is bounded by their average. If we denote by $\mathcal{P}_N\left(\mathcal{T}\right)$ the set of all possible subsets $E \subseteq \mathcal{T}$ of cardinality N, we have:

$$\mathbb{V}_n \geq \text{avg}_{E \in \mathcal{P}_N(\mathcal{T})} \frac{1 - p_E}{p_E}\left(n - N\right) \tag{3.1}$$

[2] The optimality of an estimator under these constraints is proven e.g. in [62].

where avg stands for the average.

Step 2: Intermediary results.

Fix $E \in \mathcal{P}_N(\mathcal{T})$. Denoting by 1_X the function whose value is 1 if X is satisfied and 0 otherwise,

$$\begin{aligned}\mathbb{P}_t\left[\text{add}\left(S_E,t\right) \neq S_E \,\middle|\, t \notin E\right] &= \frac{\mathbb{P}_t\left[\text{add}\left(S_E,t\right) \neq S_E, t \notin E\right]}{\mathbb{P}_t\left[t \notin E\right]} \\ &= \frac{|\mathcal{T}|}{|\mathcal{T}|-N} \cdot \text{avg}_{t\in\mathcal{T}}\left(1_{\text{add}(S_E,t)\neq S_E}\right). \qquad (3.2)\end{aligned}$$

Indeed, $\mathbb{P}_t\left[t \notin E\right] = \frac{|\mathcal{T}|-N}{|\mathcal{T}|}$ is straightforward, and since $t \in E$ implies $\text{add}\left(S_E,t\right) = S_E$, the condition $\text{add}\left(S_E,t\right) \neq S_E, t \notin E$ can be simplified to $\text{add}\left(S_E,t\right) \neq S_E$.

Now, fix $t \in \mathcal{T}$. Then

$$\begin{aligned}\mathbb{P}_N\left[\text{add}\left(S_E,t\right) \neq S_E \,\middle|\, t \notin E\right] &= \frac{\mathbb{P}_N\left[\text{add}\left(S_E,t\right) \neq S_E, t \notin E\right]}{\mathbb{P}_N\left[t \notin E\right]} \\ &= \frac{|\mathcal{T}|}{|\mathcal{T}|-N} \cdot \text{avg}_{E\in\mathcal{P}_N(E)}\left(1_{\text{add}(S_E,t)\neq S_E}\right). \qquad (3.3)\end{aligned}$$

Indeed, we have

$$\begin{aligned}\mathbb{P}_N\left[t \notin E\right] &= \binom{|\mathcal{T}|-1}{N} \cdot \binom{|\mathcal{T}|}{N}^{-1} \\ &= \frac{(|\mathcal{T}|-1)!}{(|\mathcal{T}|-1-N)!N!} \cdot \frac{(|\mathcal{T}|-N)!N!}{|\mathcal{T}|!} = \frac{|\mathcal{T}|-N}{|\mathcal{T}|}.\end{aligned}$$

Step 3: Conclude using convexity.

We now prove that $\text{avg}_{E\in\mathcal{P}_N(\mathcal{T})}\left(p_E\right) \leq p$. Our initial hypothesis states that for all t, we have $p \geq \mathbb{P}_N\left[\text{add}\left(S_E,t\right) \neq S_E \,\middle|\, t \notin E\right]$. We can average this for every t and use (3.3) and (3.2):

$$\begin{aligned} p &\geq \text{avg}_{t\in\mathcal{T}}\left(\mathbb{P}_N\left[\text{add}\left(S_E,t\right) \neq S_E \,\middle|\, t \notin E\right]\right) \\ &\overset{(3.3)}{=} \text{avg}_{t\in\mathcal{T}}\left(\frac{|\mathcal{T}|}{|\mathcal{T}|-N} \cdot \text{avg}_{E\in\mathcal{P}_N(E)}\left(1_{\text{add}(S_E,t)\neq S_E}\right)\right) \\ &= \text{avg}_{E\in\mathcal{P}_N(\mathcal{T})}\left(\frac{|\mathcal{T}|}{|\mathcal{T}|-N} \cdot \text{avg}_{t\in\mathcal{T}}\left(1_{\text{add}(S_E,t)\neq S_E}\right)\right) \\ &\overset{(3.2)}{=} \text{avg}_{E\in\mathcal{P}_N(\mathcal{T})}\left(\mathbb{P}_t\left[\text{add}\left(S_E,t\right) \neq S_E \,\middle|\, t \notin E\right]\right) \\ &\geq \text{avg}_{E\in\mathcal{P}_N(\mathcal{T})}\left(p_E\right) \qquad (3.4)\end{aligned}$$

Now, using (3.1) and (3.4) along with the fact that the function $x \longrightarrow \frac{1-x}{x} \cdot (n-N)$ is convex and decreasing on $(0,1)$, we can conclude by Jensen's inequality that

$$\mathbb{V}_n \geq \frac{1-p}{p}\left(n-N\right).$$

□

All existing cardinality estimators satisfy our axioms and their standard error remains low even for large values of n. Theorem 7 shows, for all of them, that there are some users whose privacy loss is significant. In Section 3.3.3.2, we quantify this precisely for HyperLogLog.

3.3.2.2 *Probabilistic case*

Algorithms that add noise to their output, or more generally, are allowed to use a source of randomness, are often used in privacy contexts. As such, even though all cardinality estimators used in practical applications are deterministic, it is reasonable to hope that a probabilistic cardinality estimator could satisfy our very weak privacy definition. Unfortunately, this is not the case.

In the deterministic case, we showed that for any record t, the probability that t has an influence on a random sketch S decreases exponentially with the sketch size. Or, equivalently, the distribution of sketches of size kn that do not contain t is "almost the same" (up to a density of probability $\left(1-e^{-\varepsilon}\right)^{k}$) as the distribution of sketches of the same size, but containing t.

The following lemma establishes the same result in the probabilistic setting. Instead of reasoning about the probability that an record t is "ignored" by a sketch S, we reason about the probability that t has a *meaningful* influence on this sketch. We show that this probability decreases exponentially, even if $\mathbb{P}\left[S \neq \text{add}\left(S, t\right)\right]$ is very high.

First, we prove a technical lemma on the *structure* that the merge operation imposes on the space of sketch distributions. Then, we find an upper bound on the "meaningful influence" of an record t, when added to a random sketch of cardinality n. We then use this upper bound, characterized using the statistical distance, to show that the estimator variance is as imprecise as for the deterministic case.

Definition 69. *Let $\mathfrak{V}$ be the real vector space spanned by the family $\{\sigma_E | E \subseteq \mathcal{T}\}$ (seen as vectors of $\mathbb{R}^{\mathcal{S}}$). For any probability distributions $\sigma, \sigma' \in \mathfrak{V}$, we denote $\sigma \cdot \sigma' = \text{merge}\left(\sigma, \sigma'\right)$. We show in Lemma 7 that this notation makes sense: on $\mathfrak{V}$, we can do computations as if* merge *was a multiplicative operation.*

Lemma 7. *The* merge *operation defines a commutative and associative algebra on $\mathfrak{V}$.*

Proof. By the properties required from probabilistic cardinality estimators in Definition 67, the merge operation is commutative and associative on the family $\{\sigma_E | E \subseteq \mathcal{T}\}$. By linearity of the merge operation, these properties are preserved for any linear combination of vectors σ_E. □

Lemma 8. *Suppose a cardinality estimator satisfies ε-sketch privacy above cardinality N, and let $t \in \mathcal{T}$. Let $\sigma_{\text{out},n}$ be the distribution of sketches obtained by adding n uniformly random records of $\mathcal{T} \setminus \{t\}$ into $S_\varnothing$ (or, equivalently, $\sigma_{\text{out},n}\left(S\right) = \mathbb{P}_n\left[S_E = S | t \notin E\right]$). Then:*

$$\upsilon\left(\sigma_{\text{out},kn}, \text{add}\left(\sigma_{\text{out},kn}, t\right)\right) \leq \left(1-e^{-\varepsilon}\right)^{k}$$

where υ is the statistical distance between probability distributions.

Proof. Let $\sigma_{\text{in},n}\left(S\right)$ be the distribution of sketches obtained by adding t, then $n-1$ uniformly random records of $\mathcal{T}$ into S (or, equivalently, $\sigma_{\text{in},n}\left(S\right) = \mathbb{P}_n\left[S_E = S | t \in E\right]$). Then the definition of ε-sketch privacy gives that for every sketch S, $\sigma_{\text{out},n}\left(S\right) \geq e^{-\varepsilon}\sigma_{\text{in},n}\left(S\right)$. So we can express $\sigma_{\text{out},n}$ as the *sum* of two distributions:

$$\sigma_{\text{out},n} = e^{-\varepsilon}\sigma_{\text{in},n} + \left(1-e^{-\varepsilon}\right)\varsigma$$

for a certain distribution ς.

First, we show that $\sigma_{\text{out},kn} = \left(\sigma_{\text{out},n}\right)^k \cdot \sigma$ for a certain distribution σ. Indeed, to generate a sketch of cardinality kn that does not contain t uniformly randomly, one can use the following process.

1. Generate k random sketches of cardinality n which do not contain t, and merge them.
2. For all $E \subseteq \mathcal{T}$, denote by p_E the probability that the k sketches were generated with the records in E. There might be "collisions" between the k sketches: if several sketches were generated using the same record, $|E| < kn$. When this happens, we need to "correct" the distribution, and add additional records. Enumerating all the options, we denote $\sigma = \sum p_E \sigma^{\text{c}}_{E,nk}$, where $\sigma^{\text{c}}_{E,nk}$ is obtained by adding $nk - |E|$ uniformly random records in $\mathcal{T}\backslash E$ to $\mathcal{S}_{\varnothing}$. Thus, $\sigma_{\text{out},kn} = \left(\sigma_{\text{out},n}\right)^k \cdot \sigma$.

Note that these distributions are in $\mathfrak{V}$: we can write $\sigma_{\text{out},n} = \text{avg}_{E\in\mathcal{P}_n(\mathcal{T}),t\notin E}\sigma_E$, and similarly for $\sigma_{\text{in},n} = \text{avg}_{E\in\mathcal{P}_n(\mathcal{T})\mathcal{T},t\in E}\sigma_E$, as well as $\varsigma = \left(1-e^{-\varepsilon}\right)^{-1}\left(\sigma_{\text{out},n} - e^{-\varepsilon}\sigma_{\text{in},n}\right)$, etc. Thus:

$$\begin{aligned}
\sigma_{\text{out},kn} &= \left(\sigma_{\text{out},n}\right)^k \cdot \sigma \\
&= \left(e^{-\varepsilon}\sigma_{\text{in},n} + \left(1-e^{-\varepsilon}\right)\varsigma\right)^k \cdot \sigma \\
&= \sum_{i=0}^{k} \binom{k}{i} e^{-i\cdot\varepsilon}\left(1-e^{-\varepsilon}\right)^{k-i}\sigma^{i}_{\text{in},n} \cdot \varsigma^{k-i} \cdot \sigma.
\end{aligned}$$

Denoting $\alpha = \sum_{i=1}^{k}\binom{k}{i}e^{-i\cdot\varepsilon}\left(1-e^{-\varepsilon}\right)^{k-i}\sigma^{i-1}_{\text{in},n} \cdot \varsigma^{k-i} \cdot \sigma$ and $\beta = \varsigma^k \cdot \sigma$, this gives us:

$$\sigma_{\text{out},kn} = \alpha \cdot \sigma_{\text{in},n} + \left(1-e^{-\varepsilon}\right)^k \beta.$$

Finally, we can compute $\text{add}\left(\sigma_{\text{out},kn}, t\right)$:

$$\begin{aligned}
\text{add}\left(\sigma_{\text{out},kn}, t\right) &= \alpha \cdot \sigma_{\text{in},n} \cdot \sigma_{\{t\}} + \left(1-e^{-\varepsilon}\right)^k \beta \cdot \sigma_{\{t\}} \\
&= \alpha \cdot \sigma_{\text{in},n} + \left(1-e^{-\varepsilon}\right)^k \beta \cdot \sigma_{\{t\}}
\end{aligned}$$

Note that since $\sigma_{\text{in},n} = \text{avg}_{E\in\mathcal{P}_n(\mathcal{T}),t\in E}\sigma_E$, we have $\sigma_{\text{in},n} \cdot \sigma_{\{t\}} = \sigma_{\text{in},n}$ by idempotence, and:

$$\begin{aligned}
&\upsilon\left(\sigma_{\text{out},kn}, \text{add}\left(\sigma_{\text{out},kn}, t\right)\right) \\
&= \frac{1}{2}\left\|\sigma_{\text{out},kn} - \text{add}\left(\sigma_{\text{out},kn}, t\right)\right\|_1 \\
&= \frac{1}{2}\left\|\left(1-e^{-\varepsilon}\right)^k\beta - \left(1-e^{-\varepsilon}\right)^k\beta\cdot\sigma_{\{t\}}\right\|_1 \\
&\leq \frac{\left(1-e^{-\varepsilon}\right)^k}{2}\left(\|\beta\|_1 + \left\|\beta\cdot\sigma_{\{t\}}\right\|_1\right) \\
&\leq \left(1-e^{-\varepsilon}\right)^k.
\end{aligned}$$

□

Lemma 8 is the probabilistic equivalent of Lemmas 4 and 5. Now, we state the equivalent of Lemma 6, and explain why its intuition still holds in the probabilistic case.

Lemma 9. *Suppose that a cardinality estimator satisfies, for any $n \geq N$ and all t:*

$$\upsilon\left(\sigma_{\text{out},n}, \text{add}\left(\sigma_{\text{out},n}, t\right)\right) \leq p.$$

Then its variance for $n \geq N$ is at least $\frac{1-p}{p}\left(n-N\right)$.

Proof. The condition "$\upsilon\left(\sigma_{\text{out},n}, \text{add}\left(\sigma_{\text{out},n}, t\right)\right) \leq p$" is equivalent to the condition of Lemma 6: with probability $\left(1-p\right)$, the cardinality estimator "ignores" when a new record t is added to a sketch. Just like in Lemma 6's proof, we can convert this constraint into estimating the size of a set based on a sampling set. The best known estimator for this problem is deterministic, so allowing the cardinality estimator to be probabilistic does not help improving the optimal variance.

The same result than in Lemma 6 follows. □

Lemmas 8 and 9 together immediately lead to the equivalent of Theorem 7 in the probabilistic case.

Theorem 8. *An unbiased probabilistic cardinality estimator that satisfies ε-sketch privacy above cardinality N is not precise. Namely, its variance is at least $\frac{1-c^k}{c^k}\left(n - k\cdot N\right)$, for any $n \leq N$ and $k \leq \frac{n}{N}$, where $c = 1-e^{-\varepsilon}$*

Somewhat surprisingly, allowing the algorithm to add noise to the data seems to be pointless from a privacy perspective. Indeed, given the same privacy guarantee, the lower bound on accuracy is the same for deterministic and probabilistic cardinality estimators. This suggests that the constraints of these algorithms (idempotence and commutativity) require them to somehow keep a trace of who was added to the sketch (at least for some users), which is fundamentally incompatible with even weak notions of privacy.

3.3.3 *Weakening the privacy definition*

Our main result is negative: no cardinality estimator satisfying our privacy definition can maintain a good accuracy. Thus, it is natural to wonder whether our privacy definition is too strict, and if the result still holds for weaker variants.

In this section, we consider two classes of weaker variants of our privacy definition: allowing the privacy loss to be larger than ε on some inputs (relaxing the definition alongside dimension **Q**, in Section 2.2.2), and allowing some proportion of users to not be protected (relaxing the definition alongside dimension **V**, in Section 2.2.4).

First, we show that for the former class of variants, our negative result still holds: allowing a small probability of bad outcomes does not help. Second, we show that even though our negative result does not hold for the latter class of variants, cardinality estimators used in practice still leak a lot of information, even when we average the privacy loss across users.

3.3.3.1 *Allowing larger values of the privacy loss*

ε-sketch privacy provides a bound on how much information the attacker can gain in the worst case. As exemplified by the DP variants alongside dimension **Q**, in Section 2.2.2, it is natural to relax this requirement

A first natural relaxation is to accept a small probability of failure, similarly to (ε,δ)-DP (Definition 14 in Section 2.2.2): requiring a bound on the information gain in *most cases*, and accept a potentially unbounded information gain with low probability.

Definition 70. *A cardinality estimator satisfies* (ε,δ)-sketch privacy above cardinality N *if for every* $\mathcal{S}_O \subseteq \mathcal{S}$, $n \geq N$, *and* $t \in \mathcal{T}$,

$$\mathbb{P}_n\left[S_E \in \mathcal{S}_O \mid t \in E\right] \leq e^{\varepsilon} \cdot \mathbb{P}_n\left[S_E \in \mathcal{S}_O \mid t \notin E\right] + \delta.$$

Unfortunately, our negative result still holds for this variant of the definition. Indeed, we show that a close variant of Lemma 4 holds, and the rest follows directly.

Lemma 10. *Let* $t \in \mathcal{T}$. *A cardinality estimator that satisfies* (ε,δ)*-probabilistic sketch privacy above cardinality N satisfies, for all* $n \geq N$:

$$\mathbb{P}_n\left[\text{add}\left(S_E,t\right) = S_E \;\middle|\; t \notin E\right] \geq \left(\frac{1}{2}-\delta\right) \cdot e^{-\varepsilon}.$$

Proof. First, we show that if a cardinality estimator verifies (ε,δ)-sketch privacy at a given cardinality, then for each target t, we can find an explicit decomposition of the possible outputs: the bound on information gain is satisfied for each possible output, except on a density δ. This is similar to the definition of *probabilistic* differential privacy (Definition 15 in Section 2.2.2), except the decomposition depends on the choice of t. We then use this decomposition to prove a variant of our negative result.

Lemma 11. *If a cardinality estimator satisfies* (ε,δ)*-sketch privacy above cardinality N, then for every* $n \geq N$ *and* $t \in \mathcal{T}$, *we can decompose the space of possible sketches* $\mathcal{S} = \mathcal{S}_1 \uplus \mathcal{S}_2$ *such that* $\mathbb{P}_n\left[S_E \in \mathcal{S}_2 \mid t \in E\right] \leq 2\delta$, *and for all* $S \in \mathcal{S}_1$:

$$\mathbb{P}_n\left[S_E = S \mid t \in E\right] \leq 2e^{\varepsilon} \cdot \mathbb{P}_n\left[S_E = S \mid t \notin E\right].$$

Proof. Suppose the cardinality estimator satisfies (ε, δ)-sketch privacy at cardinality N, and fix $n \geq N$ and $t \in \mathcal{T}$. Let $\varepsilon' = \varepsilon + \ln(2)$.

Let $\mathcal{S}_1$ be the set of outputs for which the privacy loss is higher than ε'. Formally, $\mathcal{S}_1$ is the set of sketches S that satisfy

$$\mathbb{P}_n[S_E = S \mid t \in E] > e^{\varepsilon'} \cdot \mathbb{P}_n[S_E = S \mid t \notin E].$$

We show that $\mathcal{S}_1$ has a density bounded by 2δ.

Suppose for the sake of contradiction that this set has density at least 2δ given that $t \in E$:

$$\mathbb{P}_n[S \in \mathcal{S}_1 \mid t \in E] \geq 2\delta.$$

We can sum the inequalities in $\mathcal{S}_1$ to obtain:

$$\mathbb{P}_n[S_E \in \mathcal{S}_1 \mid t \in E] > e^{\varepsilon'} \cdot \mathbb{P}_n[S_E \in \mathcal{S}_1 \mid t \notin E].$$

Averaging both inequalities, we get:

$$\mathbb{P}_n[S_E \in \mathcal{S}_1 \mid t \in E] > e^{\varepsilon} \cdot \mathbb{P}_n[S_E \in \mathcal{S}_1 \mid t \notin E] + \delta$$

since $e^{\varepsilon'} = e^{\varepsilon+\ln(2)} = 2 \cdot e^{\varepsilon}$. This contradicts the hypothesis that the cardinality estimator satisfies (ε, δ)-sketch privacy at cardinality N.

Thus, $\mathbb{P}_n[S \in \mathcal{S}_1 \mid t \in E] < 2\delta$ and by the definition of $\mathcal{S}_1$, every output $S \in \mathcal{S}_2 = \mathcal{S} \setminus \mathcal{S}_1$ verifies:

$$\mathbb{P}_n[S_E = S \mid t \in E] \leq e^{\varepsilon'} \cdot \mathbb{P}_n[S_E = S \mid t \notin E].$$

□

We can then use this decomposition to prove Lemma 10. Lemma 11 allows us to get two sets $\mathcal{S}_1$ and $\mathcal{S}_2$ such that:

- $\mathbb{P}_n[S_E \in \mathcal{S}_2 \mid t \in E] \leq 2\delta$;
- and for all $S \in \mathcal{S}_1$, $\mathbb{P}_n[S_E = S \mid t \in E] \leq 2e^{\varepsilon} \cdot \mathbb{P}_n[S_E = S \mid t \notin E]$.

We decompose $\mathbb{P}_n\left[\text{add}(S_E, t) = S_E \;\middle|\; t \in E\right]$ into:

$$\mathbb{P}_n\left[\text{add}(S_E, t) = S_E, S_E \in \mathcal{S}_1 \;\middle|\; t \in E\right] + \mathbb{P}_n\left[\text{add}(S_E, t) = S_E, S_E \in \mathcal{S}_2 \;\middle|\; t \in E\right].$$

The same reasoning as in the proof of Lemma 4 gives:

$$\begin{aligned}\mathbb{P}_n\left[\text{add}(S_E, t) = S_E, S_E \in \mathcal{S}_1 \;\middle|\; t \in E\right] &\leq 2e^{\varepsilon} \cdot \mathbb{P}_n\left[\text{add}(S_E, t) = S_E, S_E \in \mathcal{S}_1 \;\middle|\; t \notin E\right] \\ &\leq 2e^{\varepsilon} \cdot \mathbb{P}_n\left[\text{add}(S_E, t) = S_E \;\middle|\; t \notin E\right]\end{aligned}$$

and since $\mathbb{P}_n[S_E \in \mathcal{S}_2 \mid t \in E] \leq 2\delta$, we immediately have:

$$\mathbb{P}_n\left[\text{add}(S_E, t) = S_E, M \in \mathcal{S}_2 \;\middle|\; t \in E\right] \leq 2\delta.$$

We conclude that

$$\mathbb{P}_n\left[\text{add}(S_E, t) = S_E \;\middle|\; t \in E\right] \leq 2e^{\varepsilon} \cdot \mathbb{P}_n\left[\text{add}(S_E, t) = S_E \;\middle|\; t \notin E\right] + 2\delta.$$

Now, Lemma 2 gives $\mathbb{P}\left[\text{add}\left(S_E, t\right) = S_E c\right] t \in E = 1$, and finally:

$$\mathbb{P}_n\left[\text{add}\left(S_E, t\right) = S_E \;\middle|\; t \notin E\right] \geq \left(\frac{1}{2} - \delta\right) \cdot e^{-\varepsilon}.$$

□

We can then deduce a theorem similar to our negative result for our weaker privacy definition.

Theorem 9. *An unbiased cardinality estimator that satisfies (ε, δ)-sketch privacy above cardinality N has a variance at least $\frac{1-c^k}{c^k}\,(n - k \cdot N)$ for any $n \leq N$ and $k \leq \frac{n}{N}$, where $c = 1 - \left(\frac{1}{2} - \delta\right) \cdot e^{-\varepsilon}$. It is therefore not precise if $\delta < 0.5$.*

Proof. This follows from Lemmas 10, 5, and 6. □

Instead of requiring that the attacker's information gain is bounded by ε for every possible output, we could bound the *average* information gain. This is equivalent to accepting a larger privacy loss in some cases, as long as other cases have a lower privacy loss. This intuition is similar to ε-Kullback-Leiber privacy (Definition 16 in Section 2.2.2), often used in similar contexts [104, 133, 327, 328]. In our case, we adapt it to maintain the asymmetry of our original privacy definition. First, we give a formal definition the *privacy loss* of a user t given output S.

Definition 71. *Given a cardinality estimator, the* positive privacy loss of t given output S at cardinality n *is defined as*

$$\varepsilon_{n,t,M} = \max\left(\ln\left(\frac{\mathbb{P}_n\left[S_E = S \mid t \in E\right]}{\mathbb{P}_n\left[S_E = S \mid t \notin E\right]}\right), 0\right).$$

This privacy loss is never negative: this is equivalent to discarding the case where the attacker gains *negative* information. Now, we bound this average over all possible values of S_E, given $t \in E$.

Definition 72. *A cardinality estimator satisfies* ε-sketch KL privacy above cardinality N *if for every $n \geq N$ and $t \in \mathcal{T}$, we have*

$$\sum_S \mathbb{P}_n\left[S_E = S \mid t \in E\right] \cdot \varepsilon_{n,t,M} \leq \varepsilon.$$

It is easy to check that ε-sketch KL privacy above cardinality N is strictly weaker than ε-sketch privacy above cardinality N. Unfortunately, this definition is also stronger than $(\varepsilon_\delta, \delta)$-sketch privacy above cardinality N for certain values of ε and δ, and as such, Lemma 10 also applies. We prove this in the following lemma.

Lemma 12. *If a cardinality estimator satisfies ε-sketch KL privacy above cardinality N, then it also satisfies $\left(\frac{\varepsilon}{\delta}, \delta\right)$-sketch privacy above cardinality N for any $\delta > 0$.*

Proof. Let $n \geq N$ and $\delta > 0$. Suppose that a cardinality estimator does not satisfy $\left(\frac{\varepsilon}{\delta}, \delta\right)$-sketch privacy at cardinality n. Then with probability strictly larger than δ, the output does not satisfy $\frac{\varepsilon}{\delta}$-sketch privacy. Formally there is $\mathcal{S}_2 \subseteq \mathcal{S}$ such that $\mathbb{P}_n\left[S_E \in \mathcal{S}_2 \mid t \in E\right] > \delta$, and such that $\varepsilon_{n,t,M} > \frac{\varepsilon}{\delta}$ for all $S \in \mathcal{S}_2$. Since all values of $\varepsilon_{n,t,M}$ are positive, we have

$$\begin{aligned}\sum_S \mathbb{P}_n\left[S_E = S \mid t \in E\right] \cdot \varepsilon_{n,t,M} &\geq \sum_{S \in \mathcal{S}_2} \mathbb{P}_n\left[S_E = S \mid t \in E\right] \cdot \varepsilon_{n,t,M} \\ &> \frac{\varepsilon}{\delta} \sum_{S \in \mathcal{S}_2} \mathbb{P}_n\left[S_E = S \mid t \in E\right] \\ &> \varepsilon.\end{aligned}$$

Hence this cardinality estimator does not satisfy ε-sketch average privacy at cardinality n. □

This lemma leads to a similar version of the negative result.

Theorem 10. *An unbiased cardinality estimator that satisfies ε-sketch KL privacy above cardinality N has a variance at least $\frac{1-c^k}{c^k}\left(n - k \cdot N\right)$ for any $n \leq N$ and $k \leq \frac{n}{N}$, where $c = 1 - \frac{e^{-4\varepsilon}}{4}$. It is therefore not precise.*

Proof. This follows directly from Lemma 12 with $\delta = \frac{1}{4}$, and Theorem 9. □

Recall that all existing cardinality estimators satisfy our axioms and have a bounded accuracy. Thus, an immediate corollary is that for all cardinality estimators used in practice, there are some users for which the *average* privacy loss is very large.

Note that we could obtain a result similar to Theorem 9 with another notion of average, using for example the same approach as Rényi DP (Definition 17 in Section 2.2.2). We could either simply use the fact that Rényi DP with $\alpha > 1$ is strictly stronger than KL privacy, or use the same reasoning to Proposition 3 in [284] to show that such a notion of average also implies (ε, δ)-sketch privacy with even lower parameters.

3.3.3.2 *Privacy loss of individual users*

ε-sketch privacy protects all users uniformly. Similarly to the variants of DP from dimension **V**, we could relax this requirement and allow some users to have less privacy than others. Variants obtained this way would generally not be sufficiently convincing to be used in practice: one typically wants to protect *all users*, not just a majority of them. In this section, we show that even if we relax this requirement, cardinality estimators would in practice leak a significant amount of information.

Allowing unbounded privacy loss for some users

First, what happens if we allow some users to have unbounded privacy loss? We could achieve this by requiring the existence of a subset of users $T \subseteq \mathcal{T}$ of density $1 - \gamma$, such that every user in T is protected by ε-sketch privacy above cardinality N. In this case, a ratio γ of possible targets are not protected. This would be somewhat similar to random DP (Definition 29 in Section 2.2.4), except the unprotected users are always the same.

This approach only makes sense if the attacker cannot choose the target t. For our attacker model, this might be realistic: suppose that the attacker wants to target just one particular person. Since all user identifiers are hashed before being passed to the cardinality estimator, this person will be associated to a hash value that the attacker can neither predict nor influence. Thus, although the attacker picks t, the true target of the attack is $h(t)$, which the attacker cannot choose.

Unfortunately, this drastic increase in privacy risk for some users does not lead to a large increase in accuracy. Indeed, the best possible use of this ratio γ of users from an accuracy perspective would simply be to count exactly the users in a sample of sampling ratio γ.

Estimating the total cardinality based on this sample, similarly to what the optimal estimator in the proof of Lemma 6 does, leads to a variance of $\frac{1-\gamma}{\gamma} \cdot (n - N)$, which is very large if γ is reasonably small. With e.g. $\gamma \simeq 10^{-4}$, this variance is too large for counting small values of n (say, $n \simeq 1000$ and $N \simeq 100$). This is not surprising: if 99.99% of the values are ignored by the cardinality estimator, we cannot expect it to count values of n on the order of thousands. But even this value of γ is larger than the δ typically used with (ε, δ)-differential privacy, where δ is often chosen to be $o(1/n)$.

But in our running example, sketches must yield a reasonable accuracy both at small and large cardinalities, if many sketches are aggregated. This implicitly assumes that the service operates at a large scale, say with at least 10^7 users. With $\gamma = 10^{-4}$, this means that thousands of users are not covered by the privacy property. This is unacceptable for most applications.

Averaging the privacy loss across users

Instead of requiring the same ε for every user, we could require that the *average* information gain by the attacker is bounded by ε. This approach is similar to *on-average KL privacy* [381], which we mentioned in Section 2.2.4. In this section, we take the example of HyperLogLog to show that accuracy is not incompatible with this notion of average privacy, but that cardinality estimators used in practice do not preserve privacy even if we average across all users.

First, we define this notion of average information gain across users.

Definition 73. *Recall the definition of the positive privacy loss $\varepsilon_{n,t,M}$ of t given output S at cardinality n from Definition 71:*

$$\varepsilon_{n,t,M} = \max\left(\ln\left(\frac{\mathbb{P}_n\left[S_E = S \mid t \in E\right]}{\mathbb{P}_n\left[S_E = S \mid t \notin E\right]}\right), 0\right).$$

The maximum privacy loss of t at cardinality n *is defined as:* $\varepsilon_{n,t} = \max_S\left(\varepsilon_{n,t,M}\right)$*. A cardinality estimator satisfies* ε-sketch privacy on average *if we have, for all n,* $\frac{1}{|\mathcal{T}|}\sum_{t\in\mathcal{T}} \varepsilon_{n,t} \leq \varepsilon$*.*

In this definition, we accept that some users might have less privacy as long as the *average* user satisfies our initial privacy definition. Here, we average over all values of ε, but like we described at the end of Section 3.3.3.1, we could use other averaging functions, which would lead to strictly stronger definitions.

We show that HyperLogLog satisfies this definition, and we discuss the value of ε for various parameters and their significance. Intuitively, a HyperLogLog cardinality estimator puts every record in a random *bucket*, and each bucket counts the *maximum number of leading zeroes* of records added in this bucket. HyperLogLog cardinality estimators have a parameter p that determines its memory consumption, its accuracy, and, as we will see, its level of average privacy.

Definition 74. *Let h be a uniformly distributed hash function. A* HyperLogLog cardinality estimator *of parameter p is defined as follows. A sketch consists of a list of 2^p counters $C_0, \ldots, C_{2^p-1}$, all initialized to 0. When adding an record e to the sketch, we compute $h(e)$, and represent it as a binary string $x = x_1 x_2 \ldots$. Let $b(e) = \left\langle x_1 \ldots x_p \right\rangle_2$, i.e., the integer represented by the binary digits $x_1 \ldots x_p$, and $\rho\left(e\right)$ be the position of the leftmost 1-bit in $x_{p+1}x_{p+2}\ldots$. Then we update counter $C_{b(e)}$ with $C_{b(e)} \leftarrow \max\left(C_{b(e)}, \rho\left(e\right)\right)$.*

For example, suppose that $x = 10001101\ldots$ and $p = 2$, then $b\left(e\right) = \langle 10\rangle_2 = 2$, and $\rho\left(e\right) = 3$ (the position of the leftmost 1-bit in $001101\ldots$). So we must look at the value for the counter C_2 and, if $C_2 < 3$, set C_2 to 3.

Theorem 11. *Assuming a sufficiently large $|\mathcal{T}|$, a HyperLogLog cardinality estimator of parameter p satisfies ε_n-sketch privacy above cardinality N on average where for $N \geq n$,*

$$\varepsilon_n \simeq -\sum_{k\geq 1} 2^{-k} \ln\left(1-\left(1-2^{-p-k}\right)^n\right).$$

Proof. To simplify our analysis, we assume in this proof that $|\mathcal{T}|$ is very large: for all reasonable values of n, picking n records uniformly at random in $\mathcal{T}$ is the same as picking a subset of $\mathcal{T}$ of size n uniformly at random. In particular, we approximate $\mathbb{P}_n\left[S_E = S \mid t \notin E\right]$ by $\mathbb{P}_n\left[S_E = S\right]$.

First, we compute $\varepsilon_{n,t,M}$ for each S, by considering the counter values of S. We then use this to determine $\varepsilon_{n,t}$, and averaging them, deduce the desired result.

Step 1: Computing $\varepsilon_{n,t,M}$

Let $t \in \mathcal{T}$, and $S \in \mathcal{S}$ such that $\mathbb{P}_n\left[S_E = S \mid t \in E\right] > 0$. Decompose the binary string $h\left(t\right) = x$ into two parts to get $b\left(t\right)$ and $\rho\left(t\right)$. Denote by $C_0, \ldots, C_{2^p-1}$ the counters of S. Note that $S = S_E$ is characterized by two conditions:

- REACH$_E(i)$ for all $0 \leq i < 2^p$, where REACH$_E(i)$ denotes the condition that if bucket i is non-empty, then an record with this number of leading zeroes was added in this bucket. Formally, REACH$_E(i)$ iff whenever $C_i > 0$, there exists $e \in E$ such that $b(e) = i$ and $\rho(e) = C_i$.
- SAX$_E$, where SAX$_E$ denotes the condition that no record has more leading zeroes than the counter for its bucket. Formally, SAX$_E$ iff for all $e \in E$, $\rho(e) \leq C_{b(e)}$.

Now, we compute the value of $\varepsilon_{n,t,M}$, depending on the value of $C_{b(t)}$. Without loss of generality, we assume that $b(t) = 0$.

Case 1: Suppose $C_0 = \rho(t)$.

We compute the probability of observing S given $t \in E$. REACH$_E$ (0) is already satisfied by t as $C_0 = \rho\left(t\right)$. So for S_E to be equal to S, E's other records only have to satisfy REACH (i) for $i \geq 1$, and SAX$_E$:

$$\mathbb{P}_n\left[S_E = S \mid t \in E\right] = \mathbb{P}_{n-1}\left[\text{SAX}_E, \text{REACH}_E(i) \text{ for all } i \geq 1\right].$$

Next, we compute the probability of observing S given $t \notin E$. This time, all records of E are chosen randomly, so

$$\begin{aligned}\mathbb{P}_n\left[S_E = S \mid t \notin E\right] &\simeq \mathbb{P}_n\left[S_E = S\right] \\ &= \mathbb{P}_n\left[\mathrm{SAX}_E, \mathrm{REACH}_E(i) \text{ for all } i\right].\end{aligned}$$

We can decompose this condition: there is a witness $e \in E$ for $\mathrm{REACH}_E(0)$ (i.e., $b(e) = 0$ and $\rho(e) = C_0 = \rho(t)$) and all others records satisfy the same conditions as in the case $t \in E$, namely SAX_E and for all $i \geq 1$, $\mathrm{REACH}_E(i)$, since this is equivalent to $\mathrm{SAX}_{E\setminus\{e\}}$ and for all $i \geq 1$, $\mathrm{REACH}_{E\setminus\{e\}}(i)$ by the choice of e. If e is chosen uniformly in $\mathcal{T}$, since h is a uniformly distributed hash function, then the following holds.

- The probability of $b(e) = 0$ is 2^{-p}.
- The probability of $\rho(e) = \rho(t)$ is $2^{-\rho(t)}$: if we denote $h(t) = x_1 \dots x_p.x_{p+1} \dots$, then $x_{p+1} = 1$ with probability $1/2$, $x_{p+1}x_{p+2} = 01$ with probability $1/4$, etc.

Thus, the probability that an record e witnesses $\mathrm{REACH}_E(0)$ is $2^{-p-\rho(t)}$ as $x_1 \dots x_p$ are independent of $x_{p+1}x_{p+2} \dots$. Since the set E has size n, there are n possible chances that such an record is chosen: the probability that at least one record witnesses $\mathrm{REACH}_E(0)$ is $1 - \left(1 - 2^{-p-\rho(t)}\right)^n$. We can thus approximate $\mathbb{P}_n\left[S_E = S \mid t \notin E\right]$ by

$$\left(1 - \left(1 - 2^{-p-\rho(t)}\right)^n\right) \cdot \mathbb{P}_{n-1}\left[\mathrm{SAX}_E, \mathrm{REACH}_E(i) \text{ for all } i \geq 1\right]$$

and thus

$$\frac{\mathbb{P}_n\left[S_E = S \mid t \in E\right]}{\mathbb{P}_n\left[S_E = S \mid t \notin E\right]} \simeq \left(1 - \left(1 - 2^{-p-\rho(t)}\right)^n\right)^{-1}.$$

Case 2: Suppose $C_0 > \rho(t)$.

We can compute the probabilities $\mathbb{P}_n\left[S_E = S \mid t \in E\right]$ and $\mathbb{P}_n\left[S_E = S \mid t \notin E\right]$ in a similar fashion.

$$\begin{aligned}\mathbb{P}_n\left[S_E = S \mid t \in E\right] &= \mathbb{P}_{n-1}\left[\mathrm{SAX}_E, \mathrm{REACH}_E(i) \text{ for all } i\right] \\ \mathbb{P}_n\left[S_E = S \mid t \notin E\right] &\simeq \mathbb{P}_n\left[S_E = S\right] \\ &= \mathbb{P}_n\left[\mathrm{SAX}_E, \mathrm{REACH}_E(i) \text{ for all } i\right].\end{aligned}$$

Since (by hypothesis) $\mathbb{P}_n\left[S_E = S \mid t \in E\right] > 0$, there exist $n-1$ distinct records of $\mathcal{T}$ which, together, satisfy conditions SAX_E and for all i, $\mathrm{REACH}_E(i)$. This allows us to bound $\mathbb{P}_n\left[S_E = S\right]$ from below. Suppose one record e of E satisfies $b(e) = 0$ and $\rho(e) = \rho(t)$, and the $n-1$ other records in $E \setminus \{e\}$ satisfy the conditions $\mathrm{SAX}_{E\setminus\{e\}}$ and $\mathrm{REACH}_{E\setminus\{e\}}(i)$ for all i. Then E satisfies SAX_E and $\mathrm{REACH}_E(i)$ for all i. The lower bound follows as before:

$$\mathbb{P}_n\left[S_E = S \mid t \notin E\right] \gtrsim \left(1 - \left(1 - 2^{-p-\rho(t)}\right)^n\right) \cdot \mathbb{P}_{n-1}\left[\mathrm{SAX}_E, \mathrm{REACH}_E(i) \text{ for all } i\right]$$

and thus

$$\frac{\mathbb{P}_n\left[S_E = S \mid t \in E\right]}{\mathbb{P}_n\left[S_E = S \mid t \notin E\right]} \lesssim \left(1 - \left(1 - 2^{-p-\rho(t)}\right)^n\right)^{-1}.$$

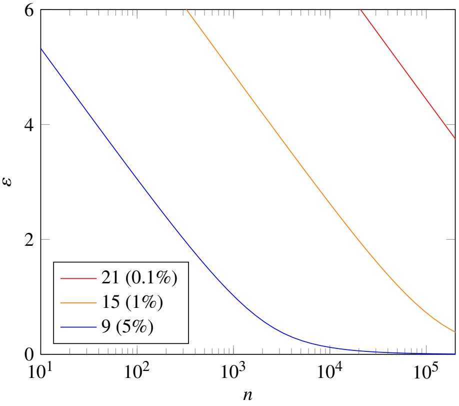


FIGURE 3.9: ε_n as a function of n, for HyperLogLog cardinality estimators of different p parameters (and their corresponding relative standard error).

We can use the results of both cases and immediately conclude that

$$\varepsilon_{n,t,M} \lesssim -\ln\left(1-\left(1-2^{-p-\rho(t)}\right)^n\right)$$

and that the equality holds if S satisfies $C_{b(t)} = \rho(t)$.

Step 2: Determining ε_n

The previous reasoning shows that the worst case happens when the counter corresponding to t's bucket contains the number of leading zeroes of t, i.e., when $\rho(t) = C_{b(t)}$:

$$\varepsilon_{n,t} = \max_S \left(\varepsilon_{n,t,M}\right) \simeq -\ln\left(1-\left(1-2^{-p-\rho(t)}\right)^n\right).$$

We can then average this value over all t. Since the hash function is uniformly distributed, $1/2$ of the values of t satisfy $\rho(t) = 1$, $1/4$ satisfy $\rho(t) = 1/4$, etc. Thus

$$\varepsilon_n \simeq -\sum_{k\geq 1} 2^{-k} \ln\left(1-\left(1-2^{-p-k}\right)^n\right).$$

□

How does this positive result fit practical use cases? Figure 3.9 plots ε_n for three different HyperLogLog cardinality estimators. It shows two important results.

First, cardinality estimators used in practice do not preserve privacy. For example, the default parameter used for production pipelines at Google and on the BigQuery service [48] is $p = 15$. For this value of p, an attacker can determine with significant accuracy whether a target was added to a sketch; not only in the worst case, but for the *average* user too. The average risk only becomes reasonable for $n \geq 10,000$, a threshold too large for most data analysis tasks.

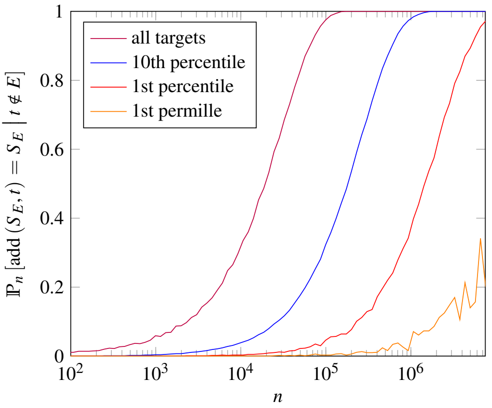


FIGURE 3.10: Simulation of $\mathbb{P}_n\left[\text{add}\left(S_E, t\right) = S_E \,\middle|\, t \notin E\right]$, for uniformly chosen values of t, using HyperLogLog sketches with parameter $p = 15$.

Second, by sacrificing some accuracy, it is possible to obtain a reasonable *average privacy*. For example, a HyperLogLog sketch for which $p = 9$ has a relative standard error of about 5%, and an ε_n of about 1 for $n = 1000$. Unfortunately, even when the average risk is acceptable, some users will still be at a higher risk: users e with a large number of leading zeroes are much more identifiable than the average. For example, if $n = 1000$, there is a 98% chance that at least one user has $\rho(e) \geq 8$. For this user, $\varepsilon_{n,t} \simeq 5$, a very high value.

Our calculations yield only an approximation of ε_n that is an upper bound on the actual privacy loss in HyperLogLog sketches. However, these alarming results can be confirmed experimentally. We simulated $\mathbb{P}_n\left[\text{add}\left(S_E, t\right) = S_E \,\middle|\, t \notin E\right]$, for uniformly random values of t, using HyperLogLog sketches with the parameter $p = 15$. For each cardinality n, we generated 10,000 different random target values, and added each one to 1,000 HyperLogLog sketches of cardinality n (generated from random values). For each target, we counted the number of sketches that ignored it.

Figure 3.10 plots some percentile values. For example, the all-targets curve (100th percentile) has a value of 33% at cardinality n = 10,000. This means that each of the 10,000 random targets was ignored by at most 33% of the 1,000 random sketches of this cardinality, i.e., $\mathbb{P}_n\left[\text{add}\left(S_E, t\right) = S_E \,\middle|\, t \notin E\right] \leq 33\%$ for all t. In other words, an attacker observes with at least 67% probability a change when adding a random target to a random sketch that did not contain it. Similarly, the 10th-percentile at n = 10,000 has a value of 3.8%. So 10% of the targets were ignored by at most 3.8% of the sketches, i.e., $\mathbb{P}_n\left[\text{add}\left(S_E, t\right) = S_E \,\middle|\, t \notin E\right] \leq 3.8\%$ for 10% of all users t. That is, for the average user t, there is a 10% chance that a sketch with 10,000 records changes with likelihood at least 96.2% when t is first added.

For small cardinalities ($n < 1,000$), adding an record that has not yet been added to the sketch will almost certainly modify the sketch: an attacker observing that a sketch does not change after adding t can deduce with near-certainty that t was added previously.

Even for larger cardinalities, there is always a constant number of people with high privacy loss. For n = 1,000, no target was ignored by more than 5.5% of the sketches. For n = 10,000, 10% of the users were ignored by at most 3.8% of the sketches. Similarly, the 1st percentile at n = 100,000 and the 1st permille at n = 1,000,000 are 4.6% and 4.5%, respectively. In summary, across all cardinalities n, at least 1,000 users t have $\mathbb{P}_n\left[\text{add}\left(S_E, t\right) = S_E \,\middle|\, t \notin E\right] \leq 0.05$. For these users, the corresponding privacy loss is $e^{\varepsilon} = \frac{1}{0.055} \simeq 18$. Concretely, if the attacker initially believes that $\mathbb{P}_n\left[t \in E\right]$ is 1%, this number grows to 15% after observing that $\text{add}\left(S_E, t\right) = S_E$. If it is initially 10%, it grows to 66%. And if it is initially 25%, it grows to 86%.

3.3.4 *Mitigation strategies*

A corollary of Theorem 7 and of our analysis of Section 3.3.3.2 is that the cardinality estimators used in practice do not preserve privacy. How can we best protect cardinality estimator sketches against insider threats,in realistic settings? Of course, classical data protection techniques are relevant: encryption, access controls, auditing of manual accesses, etc. But in addition to these best practices, cardinality estimators like HyperLogLog allow for specific risk mitigation techniques, which restrict the attacker's capabilities.

3.3.4.1 *Salting the hash function with a secret*

As explained in Section 3.3.1.2, most cardinality estimators use a hash function h as the first step of the add operation: $\text{add}\left(S, t\right)$ only depends on S and the hash value $h(t)$. This hash can be salted with a secret value. This salt can be made inaccessible to humans, with access controls restricting access to production binaries compiled from trusted code. Thus, an adversary cannot learn all the relevant parameters of the cardinality estimator and can no longer add users to sketches. Of course, to avoid a salt reconstruction attack, a cryptographic hash function must be used.

The use of a salt does not hinder the usefulness of sketches: they can still be merged (for all cardinality estimators given as examples in Section 3.3.1.2) and the cardinality can still be estimated without accuracy loss. However, if an attacker gains direct access to a sketch S with the aim of targeting a user t and does not know the secret salt, then they cannot compute $h\left(t\right)$ and therefore cannot compute $\text{add}\left(S, t\right)$. This prevents the previous obvious attack of adding t to S and observing whether the result is different.

However, this solution has two issues.

1. *Secret salt rotation*: the secret salt must be the same for all sketches as otherwise sketches cannot be merged. Indeed, if a hash function h_1 is used to create a sketch S_1 and h_2 is used to create S_2, then if $h_1\left(t\right) \neq h_2\left(t\right)$ for some t that is added to both S_1 and S_2, t will be seen as a *different user* in S_1 and S_2: the cardinality estimator no longer ignores duplicates. Good key management practices also recommend regularly rotating secret keys. In this context, changing the key requires recomputing all previously computed sketches. This requires keeping the original raw data, makes pipelines more complex, and can be computationally costly.

2. *Sketch intersection*: for most cardinality estimators given as examples in Section 3.3.1.2, it is possible for an attacker to guess $h(t)$ from a family of sketches $(S_1, \dots, S_k)$ for which the attacker *knows* that $t \in S_1$. For example, intersecting the lists stored in K-Minimum Values sketches can provide information on which hashes come from users that have been in *all* sketches. For HyperLogLog, one can use the leading zeroes in non-empty buckets to get partial information on the hash value of users who are in all sketches. Moreover, HyperLogLog++ [195] has a *sparse mode* that stores full hashes when the sketch contains a small number of values; this makes intersection attacks even easier.

   Intersection attacks are realistic, although they are significantly more complex than simply checking if $\text{add}(S, t) = S$. In our running example, sketches come from counting users across locations and time periods. If an internal attacker wants to target someone they know, they can gather information about where they went using side channels like social media posts. This gives them a series of sketches $S_1, \dots, S_k$ that they *know* their target belongs to, and from these, they can get information on $h(t)$ and use it to perform an attack equivalent to checking whether $\text{add}(S, t) = S$.

Another possible risk mitigation technique is homomorphic encryption. Each sketch could be encrypted in a way that allows sketches to be merged, and new records to be added; while ensuring that an attacker cannot do any operation without some secret key. Homomorphic encryption typically has significant overhead, so it is likely to be too costly for most use cases. Our impossibility results assume a computationally unbounded attacker; however, it is possible that an accurate sketching mechanism using homomorphic encryption could provide privacy against *polynomial-time* attackers. We leave this area of research for future work.

3.3.4.2 *Using a restricted API*

Using cardinality estimator sketches to perform data analysis tasks only requires access to two operations: merge and estimate. So a simple option is to process the sketches over an API that only allows this type of operation. One option is to provide a SQL engine on a database, and only allow SQL functions that correspond to merge and estimate over the column containing sketches. In the BigQuery SQL engine, this corresponds to allowing HLL_COUNT.MERGE and HLL_COUNT.EXTRACT functions, but not other functions over the column containing sketches [48]. Thus, the attacker cannot access the raw sketches.

Under this technique, an attacker who only has access to the API can no longer directly check whether $\text{add}(S, t) = S$. Since they do not have access to the sketch internals, they cannot perform the intersection attack described previously either. To perform the check, their easiest option is to impersonate their target within the service, interact with the service so that a sketch $S_{\{t\}}$ containing *only* their target is created in the sketch database, and compare the estimates obtained from S and $\text{merge}(S, S_{\{t\}})$. Following the intuition given in Section 3.3.2.1, if these estimates are the same, then the target is more likely to be in the dataset. How much information the attacker gets this way depends on $\mathbb{P}\left[\text{estimate}(\text{add}(S_E, t)) = \text{estimate}(S_E) \mid t \notin E\right]$. We can increase this quantity by rounding the result of the estimate operation, thus limiting the accuracy of the external attacker.

This would make the attack described in this work slightly more difficult to execute, and less efficient. However, it is likely that the attack could be adapted, for example by repeating it multiple times with additional fake records.

This risk mitigation technique can be combined with the previous one. The restricted API protects the sketches during normal use by data analysts, i.e., against external attackers. The hash salting mitigates the risk of manual access to the sketches, e.g., by internal attackers. This type of direct access is not needed for most data analysis tasks, so it can be monitored via other means.

3.4 CONCLUSION

It is natural to want to model attackers with only partial background knowledge, and to take this uncertainty into account when quantifying the privacy properties of a given mechanism. Certainly, in an ideal world where privacy would be the only concern, this would not be necessary: it is safer to always assume the most powerful adversary. However, we do not live in such a world. In practice, stakeholders frequently complain about the utility or complexity cost of differential privacy, and question whether its use is really necessary to reach a reasonable level of protection. Surely, aggregating data over many people is enough to prevent realistic adversaries from getting information about single individuals?

Ignoring those complaints, or dismissing them as entirely misguided, is an inadequate answer. Real-world adversaries do *not* have full background knowledge, and as such, it is worthwhile to improve our formal understanding of the guarantees provided by this uncertainty. This chapter provided three complementary insights about this setting.

The first insight is that modeling an attacker with partial background knowledge is more complicated and hazardous than it initially appears. As we showed in Section 3.1.2, correctly modelling such adversaries requires to carefully consider the influence of data correlations, and cleanly delineate the attacker's partial knowledge from the information that we are trying to protect. Furthermore, as we showed in Section 3.1.3, an attacker who only passively knows part of the data is a priori weaker than an attacker who can influence the data, so practical applications must also consider this dichotomy—a subtlety that does not exist for classical differential privacy. On the positive side, our work provides a solid theoretical basis for reasoning about partial knowledge in a formal, correct way; we hope that this can be of use to researchers and practitioners trying to quantify the privacy of given mechanisms under this assumptions.

The second insight is that some of these intuitions about the inherent safety of certain mechanisms under partial knowledge are correct. Results from Section 3.2.1 show that aggregating data across many users *does* preserve privacy under reasonable assumptions. Our results focus on counting, but they could easily be extended to other aggregations. Indeed, the core insight from the proof of Theorem 3 comes from drawing a link between our setting and amplification by shuffling. In essence, uncertainty about individual records can be expressed with (ε, δ)-indistinguishability, and symmetric aggregation is at least as good as random shuffling to boost privacy guarantees. This suggests that additional results and links could be drawn, and that both settings could be combined to obtain finer privacy analyses with reasonable attacker assumptions.

We also showed in Section 3.2.2 that thresholding does preserve privacy against passive attackers by providing a natural and intuitive example of the distinction between passive and active attackers with partial knowledge. Formalizing this intuition also allows us to understand under which conditions simple k-anonymity mechanisms preserve privacy against an attacker with partial knowledge. This last result depends on a number of assumptions, most of which can be linked to a corresponding weakness of k-anonymity: if one of these assumptions does not hold, then one can find an example of when k-anonymity fails to protect individual privacy. This shows one of the weaknesses of differential privacy under partial knowledge: it is difficult to obtain *robust* results that do not depend on brittle assumptions about the exact nature of the attacker's uncertainty.

Finally, in Section 3.2.4, we looked at composition properties for differential privacy with partial knowledge. By contrast to differential privacy in its original form, sequential composition does not hold in general. However, we found that quantifying the amount of correlation between the results of two queries can be used to compose privacy guarantees, and that yet again, (ε, δ)-indistinguishability is a natural formalism to do so. We also raised the interesting question of combining guarantees from partial knowledge and noise addition; a question that we largely leave open.

The third insight, from Section 3.3, is that modeling an attacker with partial knowledge can be used to obtain strong *negative* results: some practical problems are inherently impossible to solve in a privacy-preserving way, even if we assume an attacker with complete uncertainty over the input data. Our main result on cardinality estimation is an important first step in this direction, and it suggests a wide range of open follow-up questions. Can we extend it to cardinality estimators that only preserve the *precision* of the estimate operator when merging them, but not necessarily the full distribution of sketches? Can we find a finer trade-off between privacy and accuracy, allowing e.g. a small error to be incurred by each merging operation in exchange for better privacy guarantees? Are there other similar problems that cannot be solved in a privacy-preserving way without giving up strong aggregation properties, e.g. sketching algorithms used for quantiles estimation?

4

FROM THEORY TO PRACTICE

Without a plane, what was I supposed to do? Math the problem to death?

— Mary Robinette Kowal, *The Calculating Stars*

Despite the academic success of differential privacy, practical applications have largely lagged behind. Scientific papers on DP often cite a few high-profile use cases to show the relevance of this field of research, but these examples almost always come from very large companies or organizations: Google [141], Apple [360], the US Census Bureau [4, 159], Microsoft [107, 224], Uber [210], and others. Is differential privacy a luxury that only large organizations can afford?

Of course, there might be some selection bias in play in the previous observation. Researchers might think that use cases from large organizations are the best way to demonstrate practical relevance, and smaller organizations using DP might not have the incentives nor resources to publish about their anonymization practices. Still, using state-of-the-art privacy techniques makes for good press, and the cost of writing a blog post is not high. Despite this, in 2020, looking up differential privacy in a search engine still surfaces mostly academic resources and use cases from large organizations.

There are a few exceptions, most of which are fairly recent. Some startups are using differential privacy as a core part of their offering [19, 192, 241, 370], and a few efforts have been initiated to implement differential privacy libraries and publish them as open-source software [105, 173, 308, 356]. Even in those cases, the people behind those projects frequently have an academic background: only organizations that can afford to hire or train domain experts get to use differential privacy. This lag between theory and practice feels surprising: as we showed in Section 2.1.6, the basic intuition behind DP is simple, and it seems easy to generate differentially private statistics by combining simple mechanisms and using the composition properties.

What are then the obstacles that prevent a wider adoption of differential privacy? In this chapter, we present a number of results and insights from four years of working on anonymization at a large company, in a role encompassing consulting, policy, and engineering.

First, we show that certain practical problems simply cannot be solved by applying differential privacy. In particular, this is the case for *risk analysis*, where the goal is not to make data anonymous, but estimate "how risky" a given dataset is, under some risk models. Tackling this class of problems can be the first step in a company's journey towards stronger anonymization practices: before deciding to implement better techniques for data sharing and retention, data owners want to understand the risk associated with their existing practices. This setting will force us to revisit syntactic privacy definitions presented in Section 2.1, since in cases where we only have access to the *output* of a certain algorithm, without any information about the algorithm itself, we cannot know whether it is differentially private.

Second, we present a differentially private SQL engine, whose primary goal is to make it easy for people—especially non-experts—to generate differentially private statistics. Safety, scalability and usability are the three main requirements of this system. Taking our SQL engine as an example, we explain how each of these requirements can be met. Safety considerations require a thoughtful and realistic attacker model, going beyond the classical "untrusted analyst" assumption from the scientific literature. Scalability is obtained via a massively parallelizable design, which can be reproduced in other contexts than this SQL engine. To make the system usable, we design an interface that mirrors the ones data analysts are already familiar with. This requires the use of novel methods to remove roadblocks that classical differential privacy mechanism would otherwise require.

Third, we raise a number of natural open questions which emerged as we were building and rolling out this differentially private SQL engine, and partially answer them when feasible. Some of these questions are about supporting a wider range of differentially private aggregations, or about improving the utility of particularly crucial feature; in particular, we present a novel algorithm to improve the utility of partition selection primitive. Other questions are about *operational* aspects: how to choose anonymization parameters, and how to provide helpful guidance for users of differential privacy tools?

Taken together, we hope that the insights gathered in this chapter can prove helpful to current and future anonymization practitioners.

4.1 ANALYZING PRIVACY RISKS AT SCALE

We saw in Section 2.1.6 that differential privacy is a property of the mechanism, not of its output. This core insight makes this definition ill-suited for an important class of problems encountered in practice: *risk analysis*. Differential privacy can give strong guarantees about the reidentifiability (or lack thereof) of some data, *assuming the algorithm that output this data was specifically built to be differentially private*. What happens when that is not the case, when the data we are interested in has already been generated? Is there a way to quantify, or at least estimate, the reidentifiability risk?

This problem is of significant practical relevance. Large companies often have many different datasets, whose number and size grows organically with the company and over time. This raises a number of policy concerns: who should have access to what data, how long should it be retained, etc. This is one of the roles of privacy engineers: they must take inventory of existing data and make such policy decisions. Unfortunately, the need for comprehensive and consistent data governance practices is often not understood until a certain amount of organic growth has taken place, and manual efforts alone are costly and unlikely to succeed: manual reviews requiring expert judgement do not scale. Privacy engineers need a way of quantifying the risk associated with datasets, to help them prioritize which datasets to look at first, and assist them in policy efforts.

Reidentifiability risk, which we mentioned above, is one such risk. Another one is *joinability*: is there an obvious way to join two datasets that are not supposed to be joined and link the information of the same individual across their different identities? In this section, we present a possible approach to tackling these problems, and analyzing datasets to detect high reidentifiability or joinability risks. Ironically, this approach is based on a novel sketching algorithm, *KHyperLogLog* (KHLL), and relies on the same cardinality estimators that we have shown in Section 3.3 not to be private.

Precisely quantifying these risks based only on the analysis of existing datasets (without knowledge of how they have been generated) is impossible in general: our methods are therefore *heuristics*. They are meant to serve as prioritization and detection tools, to assist engineers in the difficult task of building and operationalizing a data governance program.

4.1.1 *Introduction*

Understanding and monitoring the privacy state of production systems is a crucial element of privacy engineering. Data-flow analysis enables one to know which data processing workflows are reading which data and generating which outputs. Static and dynamic code analyses help to understand what binaries do at a more granular level e.g., whether they use pre-approved APIs, whether they read or write sensitive data types, or whether they use safe defaults. Potential violations of privacy policies can be surfaced during code review sessions, before an engineer is allowed to run workflows in production systems.

However, while data flow analysis and code analysis are powerful tools, characterizing the data to assess their sensitivity (such as we propose in this chapter) can often be useful or necessary. While human reviewers often have good intuitions and context about the sensitivity of data, there are obvious limitations: humans may make mistakes, and are limited in how much they can review. An automated analysis system, on the other hand, can be accurate and scalable. Where humans would have to settle for evaluating a system or dataset once, automated systems can be re-run. This provides regression testing for privacy characteristics and enables data custodians to be confident about the properties of their systems.

Automatic evaluation becomes challenging as dataset size increases and as data becomes increasingly heterogeneous. While brute force approaches could work for smaller datasets, their runtime and memory requirements become unworkable when run on petabytes of data.

4.1.1.1 *Reidentifiability and joinability*

We identify two characteristics of datasets that are often useful during privacy impact assessments: reidentifiability and joinability, and we develop a new, scalable, automated approach to measuring them. While these terms may have different connotations, we define them below, and use them consistently throughout this work.

Reidentifiability is the potential risk that the identity of some users can be recovered in some pseudonymous datasets. A good practice is to keep such reidentifiable datasets guarded with strict access control and access audit trails. As companies collect more information to build useful services it can be difficult to manually determine when a dataset becomes reidentifiable and requires more careful handling. The ability to estimate the reidentifiability of datasets automatically and efficiently reduces the work required of data custodians to manually label the different datasets.

Joinability measures whether datasets are linkable by unexpected join keys. Sometimes it is necessary to retain multiple datasets with different ID spaces. In those cases data custodians should avoid linking the two datasets to respect the choices of users who maintain separate identities. As an example, consider a website that can be used either signed-in or signed-out. A user may choose to use the website while signed-out to separate activities from their signed-in identity. If the website operator maintains datasets about activities of both signed-in and signed-out users, it might accidentally include granular information (e.g. web browser user agent) in both datasets that could allow the signed-in and signed-out identities to be linked. In that case, we would say that the identities in the two datasets are *joinable*.

Some syntactic privacy definitions we listed in Chapter 2, like k-anonymity or k-map, suggest a possible approach to quantifying reidentifiability: for each characteristic, we could quantify the total number of people associated to the same point. Similarly, taking the simplest approach to joinability, we could say that two datasets are joinable if there exists a pair of data fields, similar in content, that can serve as a join key. To measure this similarity between fields, or measure when the values of a field are a subset of the values of another field (a notion we call *containment* later), using the Jaccard index is a popular option [55, 205].

However, managing reidentifiability and joinability risks at scale is more challenging than it appears. The naive approach requires memory proportional to the size of the dataset, which becomes extremely difficult as dataset sizes climb into the petabytes. Our experiments reveal how costly this naive approach is even for datasets in the order of gigabytes (see Section 4.1.4.1). Linear runtime and sublinear memory are necessary for large-scale data analysis.

4.1.1.2 *Contributions*

We present the KHyperLogLog (KHLL) algorithm and demonstrate how it can be used to efficiently characterize both the reidentifiability and joinability of very large datasets. Adapted from the field of cardinality estimation, KHLL produces quantitative measures for reidentifiability and joinability using only a single pass over the dataset and minimal memory. Both reidentifiability and joinability of datasets can be estimated using the compact data structures (colloquially known as “sketches”) of KHLL rather than raw data. In addition, the approach is format-agnostic, allowing it to analyze any dataset without modification. We have validated that KHLL is fast, parallelizable and accurate on both proprietary and public datasets.

KHLL can be used in a wide range of applications.

Quantitative measurement

KHLL can be used to quantitatively measure the reidentifiability risk of a dataset, or a fraction of its columns. More precisely, it can be used to calculate the uniqueness distribution of an arbitrary combination of columns with respect to a user identifier. This can inform data custodians about the sensitivity of data and its risks so they can plan a suitable and appropriate data strategy (e.g., access controls, audit trails, or stronger anonymization) at points in the life cycle of the data, including data collection, usage, sharing, retention, or deletion.

Exploring data strategies

The efficiency of KHLL provides data custodians a powerful analysis tool for exploring different data sanitization strategies. Many data analysis tasks (e.g., experimentation to improve service availability, anti-spam and fraud detection) can use a projection (view) of a high-dimensional dataset that is protected with k-anonymity. KHLL can be run on different possible combinations of columns in the dataset at once to estimate how much data will be lost as a function of the technique. Together, the data custodian and data analysts can decide how to trade off the utility of the data projection and the reidentifiability risk: which columns should be included, suppressed, generalized or made disjoint in separate datasets, and whether the data needs stronger guarantees like differential privacy.

Consider the Netflix prize dataset, which contains the movie ids and ratings given by different users at different dates (year, month and day). Analyzing the dataset using KHLL, we obtain results that mirror those of Narayanan and Shmatikov [294]. While no single column has high uniqueness (e.g., we observe that all movies included in the dataset are rated by at least 50 users), the combination of movie ratings and dates are highly unique. An efficient analysis using KHLL might have helped the Netflix team measure the reidentifiability risks, explore alternatives for treating the data, or to potentially conclude that the risk was too high to share the data externally.

Regression testing

In cases where data custodians regularly produce k-anonymous (or the like) datasets, KHLL can be further used as a regression test. KHLL analysis can be run on the output as part of the anonymization pipeline to expose any implementation bugs, or to alert on any unexpected changes to the characteristics of the input data.

Joinability assessment

KHLL can also enable efficient joinability assessment to protect user privacy. If an organization collects data about users under multiple ID spaces in different contexts (e.g., signed in vs signed out), KHLL can be used to keep the IDs separate, respecting the choice of users to conduct certain activities in certain contexts. For example, KHLL analysis can be run on two datasets of different IDs, and be used to detect data columns in the two datasets that are similar (high containment in either direction) and that are highly unique. These data columns are potential join keys that can be used to trivially link the two ID spaces. To mitigate joinability risks, engineers can choose to suppress or generalize one of the columns, or use access controls to prevent someone from using the columns to join the two identifiers. The analysis can be run periodically and attached to an alerting system that notifies engineers if joinability exceeds pre-specified limits (e.g., to quickly detect whether any newly added columns increase joinability risks). Joinability assessment is highly intractable with pairwise comparisons of raw data, but KHLL enables joinability approximation based on its compact data structures (sketches).

Periodic KHLL-based joinability analyses have enabled us to uncover multiple logging mistakes that we were able to quickly resolve. One instance was the exact position of the volume slider on a media player, which was mistakenly stored as a 64-bit floating-point number. Such a high entropy value would potentially increase the joinability risk between signed-in and signed-out identifiers. We were able to mitigate the risk by greatly reducing the precision of the value we logged. In other cases, we have mitigated joinability risks by dropping certain columns entirely, or by ensuring that the access control lists of both datasets are disjoint.

Miscellaneous

If data custodians label their datasets with information about the semantics of certain columns, KHLL can be used to propagate labels through the system and find inconsistencies. If two columns have a high containment score (in either direction), they are likely to share the same semantics. If one of the columns is labelled but the other is not, then the label can be copied to the second column, and if the two columns have different labels then engineers can be alerted that one of the labels is likely incorrect. The scalability of KHLL means that it can be used to propagate labels across large datasets, and that the label correctness can be repeatedly checked by re-running analysis periodically.

Although not a primary purpose, an additional side effect of making a powerful analysis tool available to different roles in an organization is the increased awareness of anonymization and user privacy. Data custodians, engineers and analysts can discuss the analysis results with each other, gain a better understanding of reidentifiability risks when working with user data, and understand why further anonymization may be necessary.

For all of these use cases one needs to keep in mind the estimation errors of KHLL (see Section 4.1.3.1 and Figure 4.1.3.2). It is possible that KHLL may underestimate reidentifiability or joinability risks (e.g., KHLL might miss values that are unique to a single user). In general, data custodians could use KHLL to estimate risks and impacts on data utility when exploring an appropriate data protection and anonymization strategy, but then use exact counting to execute the strategy. While the joinability analysis using KHLL might be sensitive to data formats and transformations, the efficiency of KHLL makes it the best regression test for data joinability that we are aware of.

Organization

The rest of this section is organized as follows. We start by describing the design goals and challenges in Section 4.1.2. We then provide some background on cardinality estimation and present our KHLL algorithm in Section 4.1.2.4. Next, we describe the use of KHLL for reidentifiability and joinability analysis in Section 4.1.3. We evaluate the performance and accuracy of KHLL empirically in Section 4.1.4, and we summarize our work and discuss related and future work in Section 4.1.5.

4.1.2 *Definitions and system design*

In this section, we introduce definitions used throughout this section, present the scalability requirements of our system, recall some useful building blocks from cardinality estimation, and finally present our KHyperLogLog (KHLL) algorithm.

4.1.2.1 *Definitions*

In this section, we define how we quantify reidentifiability and joinability. These metrics are defined on individual columns and can be directly extended to any combinations of columns.

Recall that our main goal is to design an efficient approach for quantifying the reidentifiability and joinability risks of large datasets. Specifically, it should help mitigate the risk of mistakes by engineers (e.g., adding additional columns to datasets without realizing they are overly unique or pose joinability risks) particularly as complex production systems evolve. We assume that the users of our system want to measure the privacy risks of their datasets, and so we do not defend against malicious users attempting to misuse our algorithm to under-report the risks of their data.

Reidentifiability via uniqueness distribution

We consider datasets $D \in \mathcal{D}$ as a sequence of *rows*, where each row is a combination between a *user identifier* (or *user ID*) $\text{id} \in \mathcal{ID}$ and multiple *columns* $C_1, C_2, \ldots$ We typically only consider a single column C at a time: we index all possible user IDs $\mathcal{ID} = \{\text{id}_i\}$ and values of this column $C = \left\{c_j\right\}$, and we associate a dataset D with the set of pairs $\left\{(\text{id}_i, c_j)\right\} \in \mathcal{ID} \times C$ present in D. Furthermore, for a specific value c_j, we denote by $\mathcal{ID}\left[c_j\right] = \left\{\text{id}_i \mid (\text{id}_i, c_j) \in D\right\}$ the set of user IDs associated with this value in D. These notations, and the others introduced throughout this section, are summarized on page xxi.

Definition 75. *The* uniqueness of a column value c_j with respect to $\mathcal{ID}$ *is given by the number of unique IDs associated with* c_j *i.e.,* $\left|\mathcal{ID}\left[c_j\right]\right|$.

Definition 76. *The* uniqueness distribution of C with respect to $\mathcal{ID}$ *is estimated by the histogram of the uniqueness of individual values in* C *in dataset* D.

This definition is slightly different from k-anonymity (see Section 2.1.1). We saw that k-anonymity counts the minimum number k of unique individuals (here, IDs) associated with any value in C. Here, we keep the *entire* distribution of k values: this way, we can compute the fraction of values in C with high reidentifiability risk, and thus the potential impact to the data when one would like to protect the data with k-anonymity or its variants.

In practice, the uniqueness distribution can be skewed. A few values in C might appear with high frequency while other values may pose high reidentifiability risks as they associate with only a small number of user IDs.

As an example, imagine a log that contains the User Agent (UA) of users who visit a site. UA is an HTTP header column which describes the application type, operating system, software vendor and software version of an HTTP request. To gauge the reidentifiability of UAs, one can estimate the uniqueness distribution by counting the number of unique IDs that individual UA strings associate with. We expect a high percentage of raw UA strings to be associated with only one or a few user IDs and thus reidentifying [135].

Joinability via containment

Let C_1 and C_2 represent the sets of values of two columns in datasets D_1 and D_2 respectively. Let $|C_1|$ denote the number of unique values in C_1, i.e., the cardinality of C_1, and $C_1 \cap C_2$

denote the set of values in both C_1 and C_2 (the intersection). We measure the joinability of D_1 and D_2 through C_1 and C_2 using the containment metric.

Definition 77. *The* containment of C_1 in C_2 *is the ratio between the number of unique values of the intersection of* C_1 *and* C_2*, and the number of unique values in* C_1 *i.e.,* $|C_1 \cap C_2| \,/\, |C_1|$.

Note that containment is similar to the Jaccard index [205], but it is asymmetric. Unlike the Jaccard index which computes the ratio between the number of unique values in the intersection of C_1 and C_2, and the union of C_1 and C_2, containment uses the number of unique values in either C_1 or C_2 as the denominator. This difference is important when C_1 and C_2 differ in size. Indeed, imagine one dataset that contains a small subset of the users from a larger dataset. The Jaccard index will always be small, and would not report joinability risk even when all values of C_1 are contained in C_2.

4.1.2.2 *Scalability requirements*

While both uniqueness distribution and containment are easy to compute on small datasets, the computation will need to scale to handle very large datasets. In addition to hosting user content for digital services, organizations collect data for providing physical services (e.g., healthcare and location-based services), improving user experience and service availability, and anti-spam and fraud detection purposes. The scale of data can be huge, both in terms of the number of data columns and rows, and the number of databases.

It would be a Herculean task for human reviewers to manually oversee all product iterations and changes to data strategies. In a similar fashion, individual well-intentioned groups of engineers also find it hard to keep up with the increasingly large number of policy and regulatory requirements.

An ideal system for measuring reidentifiability and joinability that is scalable will need to use efficient and parallelizable algorithms. Also, as increasingly heterogeneous data is collected and used it will need an approach agnostic to data types and formats to handle datasets generated by different engineering teams.

4.1.2.3 *Cardinality estimation basics*

In this section, we re-introduce two cardinality estimation techniques mentioned in Section 3.3. Recall that cardinality estimation is a technique to efficiently approximate the number of distinct elements in a multiset [45, 152, 195], typically using a small amount of memory. These algorithms use compact data structures, colloquially known as “sketches”, which can summarize certain observable properties of the elements in the analyzed dataset. Sketching algorithms have also been proposed to compute other statistics such as quantiles [214, 268] and frequent values [65, 84, 281].

In addition to being memory efficient, cardinality estimation sketches support additional operations such as merging (set union). Large-scale datasets are typically stored in multiple machines (“shards”), as the entire dataset would not fit in a single machine. In such situations, one can compute the cardinality of the entire dataset using a two-step approach: compute the sketches of individual data shards, then merge all the sketches generated in step 1.

In this section, we extend two cardinality estimation algorithms named K-Minimum-Values (KMV) [45] and HyperLogLog (HLL) [152] to build KHLL.

K-Minimum-Values (KMV)

As implied by the name, KMV estimates the cardinality of a set by keeping the K smallest hash values of its elements. The intuition behind KMV is as follows. Suppose there exists a hash function that uniformly maps input values to its hash space. Note that this hash function does not need to be a cryptographic hash function, and one-wayness is not required (i.e., it does not matter if the hash function can be reversed in polynomial time). If one computes the hash of each element in the analyzed dataset, one can expect those hashes to be evenly distributed across the hash space. Then, one can estimate the cardinality of the analyzed dataset by computing the density of the hashes (i.e., the average distance between any two consecutive hashes) and dividing the hash space by the density. Since storing all the hashes can incur a significant storage cost, one can store only the K smallest hash values and extrapolate the density of the entire hash space.

As a concrete example, say there is a hash function whose outputs are evenly distributed in the range $[1, 1{,}000{,}000]$. If $K = 100$, and the Kth smallest hash value is 1000, we can compute the density by simply dividing the Kth smallest hash value by K, i.e., density $= 1000/100 = 10$. Extrapolating to the range of $[1, 1{,}000{,}000]$, with the uniformity assumption but without bias correction, one can roughly estimate the number of unique values as $1{,}000{,}000/10 = 100{,}000$.

Computing set union using KMV sketches is straightforward. Given two KMV sketches, S_1 and S_2, one can find the KMV sketch of the union of the two datasets by combining the two sketches and retaining only the K smallest hashes.

KMV sketches are efficient to produce. It requires a single pass over the dataset, but only a space complexity of $O(K)$, as it consists of K unique hash values of fixed length. The cardinality estimated by a KMV sketch has a relative standard error of $\frac{1}{\sqrt{K}}$ with the assumption that the hash space is large enough to keep hash collisions to a minimum. As a concrete example, with $K = 1024$ and using a 64-bit uniformly distributed hashing function, one can estimate the cardinality with a relative standard error of 3% and KMV sketch size of 8 KB.

HyperLogLog (HLL)

Instead of keeping the K smallest hash values, HLL further reduces the space requirement by tracking the maximum number of trailing zeros of the hash values. The maximum number of trailing zeros increases as more unique values are added to HLL given the uniformity assumption of the hash function.

From the hash of an incoming value, HLL uses the first P bits to determine the bucket number, and uses the remaining bits to count the number of trailing zeros. HLL keeps track of the maximum number of trailing zeros at each of the $M = 2^P$ buckets. After processing all values in the analyzed dataset, HLL estimates the cardinality of each bucket as 2^{m_i}, where m_i is the maximum number of trailing zeros seen in bucket i. Finally, HLL estimates the cardinality of the analyzed dataset by combining the cardinalities of individual buckets by taking the harmonic mean.

HLL sketches are also efficient to compute (i.e., using a single pass over the analyzed dataset) and provide cardinality estimates with a relative standard error of $\frac{1.04}{\sqrt{M}}$. Moreover, the space complexity of a HLL sketch is $O(M)$ since it consists of M counts of trailing zeros. As a concrete example, with M=1024 and using a 64-bit uniformly distributed hashing function, one can estimate the cardinality with a relative standard error of 3% and HLL sketch size of 768 B.

Heule et al. showed that HLL does not provide a good estimate for low cardinalities and proposed HLL++ [195] to accommodate such datasets. HLL++ maintains two different modes of sketches. When the cardinality is low, it remains in the sparse representation mode, which keeps almost the entire hash values. When the list of hash values kept grows, HLL++ switches to the conventional HLL mode which has a fixed memory footprint. The sparse representation allows HLL++ to use linear counting for estimating small cardinalities with negligible error while also keeping the sketch size small.

4.1.2.4 *KHyperLogLog (KHLL)*

While cardinality estimators are helpful, they are limited in many ways for reidentifiability and joinability analysis. While cardinality estimates can be used to estimate the average uniqueness when the total unique IDs in the dataset is known, they do not estimate the uniqueness distribution. The average alone can be misleading: the uniqueness distribution can be skewed in practice. The uniqueness distribution is also useful to inform about various data strategies, for example the feasibility of suppressing or generalizing a fraction of the unique values. The distribution could not be naively estimated as we could not assume the datasets to be structured in a way that every single row corresponds to a single user.

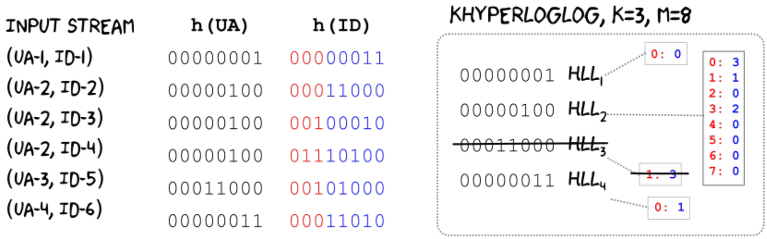
In this section, we present KHyperLogLog (KHLL), which builds on KMV and HLL to estimate uniqueness distribution and containment with a single pass over the dataset and low memory requirements. The core insight is to use a *two-level* data structure for analyzing tuples of column and ID values. KHLL contains K HLL sketches corresponding to K smallest hashes of column values. This is approximately equivalent to taking a uniform random sampling of size K over the column values. Each hash of column value comes with a corresponding HLL sketch, containing the hashes of IDs associated with the corresponding column value.

Consider a stream of pairs $(\text{id}_i, c_j) \in \mathcal{ID} \times C$ and a hash function h. KHLL processes the incoming tuples as follows:

1. Calculate $h(c_j)$ and $h(\text{id}_i)$.

2. If $h(c_j)$ is already present in the KHLL sketch, add $h(\text{id}_i)$ to the corresponding HLL sketch.

3. Else, if $h(c_j)$ is among the K smallest hashes:
    a) If there are more than K entries, purge the entry containing the largest hash of C.
    b) Add a new entry containing $h(c_j)$ and a HLL sketch with $h(\text{id}_i)$.



INPUT STREAM	h(UA)	h(ID)
(UA-1, ID-1)	00000001	00000011
(UA-2, ID-2)	00000100	00011000
(UA-2, ID-3)	00000100	00100010
(UA-2, ID-4)	00000100	01110100
(UA-3, ID-5)	00011000	00101000
(UA-4, ID-6)	00000011	00011010

FIGURE 4.1: A stream of User Agent (UA) and ID tuples processed by an example KHLL sketches with $K = 3$ and $M = 8$. When the tuple (UA-4, ID-6) is added, the entry with the largest $h(\text{UA}) = \texttt{00011000}$ and its companion HLL sketch is purged to give way to $h(\text{UA-4})$ and a new HLL. Notice that the sketches HLL_1 and HLL_4 are in sparse representation, while HLL_2 is in the conventional table form.

4. Else, do nothing.

As a specific example, consider a stream of User Agent (UA) and ID value pairs. Further, consider an 8-bit hash function and a KHLL sketch of $K = 3$ and $M = 8$. The KHLL sketch contains at most 3 entries representing the 3 smallest values of $h(\text{UA})$ in the first level, each with a HLL sketch in the second level which has at most 8 counting buckets. For example, when the KHLL sketch processes the tuple (UA-4, ID-6) which hashes to (00000011, 00011010) as shown in Figure 4.1, the entry with with the largest $h(\text{UA}) = 00011000$ and its companion HLL sketch is purged to give way to $h(\text{UA-4})$ and a new HLL.

The memory signature of a KHLL sketch depends on the parameters K and M as well as the uniqueness distribution of the data. Similarly to HLL++ [195], we design the HLL sketches in KHLL to start in the sparse representation mode which keeps a sparse list of the ID hash values. Once this representation exceeds the fixed size of a conventional HLL, it is converted to a normal representation with M counting buckets. Using a 64-bit hash function, individual counting buckets require less than a byte to count the maximum number of trailing zeros in the ID hash values.

Recall that $\mathcal{ID}[c_j] = \left\{\text{id}_i \mid (\text{id}_i, c_j) \in D\right\}$ is the set of user IDs associated with a given column value c_j in dataset D. The memory needed for a KHLL sketch considering both the sparse and conventional mode is thus $\min\left(8\left|\mathcal{ID}[c_j]\right|, M\right)$ in bytes. Since the KMV approximates a K size uniform random sample over column values, the expected memory usage for the entire KHLL will be roughly K times the average HLL size i.e., $\frac{K}{|C|} \cdot \sum_i \min\left(8\left|\mathcal{ID}[c_j]\right|, M\right)$.

This means that the memory usage of KHLL, while never above a strict upper bound, will be higher for datasets with low uniqueness in which most column values are associated with large user ID sets. Alternatively, when most column values correspond to only a few unique user IDs, the memory signature will be much smaller as the HLL sketches will be in sparse representations.

Note that KHLL does not dictate how the data is structured. To process a table of data, we simply read each row in the table, extract the ID value and the values of columns (or

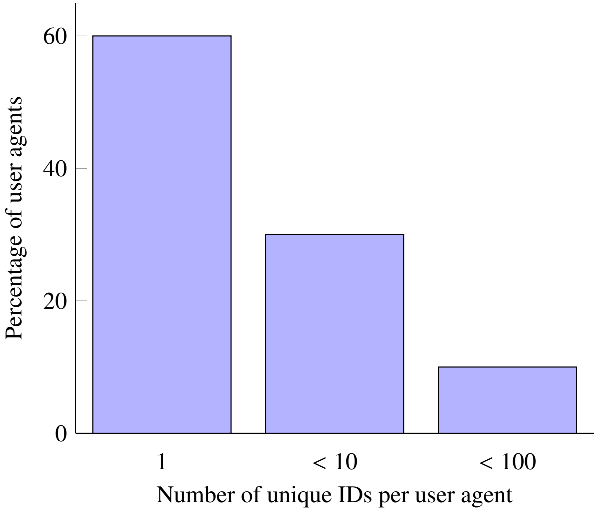


FIGURE 4.2: Example uniqueness histogram (fake data, for illustration purposes only). We expect User Agent (UA) to have a uniqueness distribution where a majority of UA strings are associated with only one or a few unique IDs.

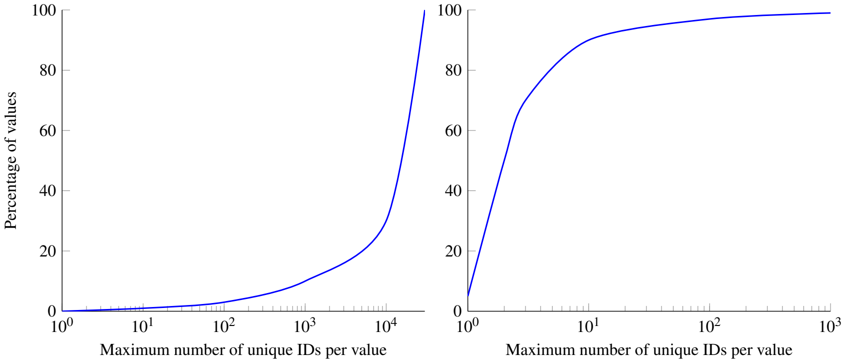


FIGURE 4.3: Two possible shapes for cumulative uniqueness distributions (fake data, for illustration purposes only). The left has low uniqueness, while the right contains values that are highly unique.

combinations of columns) of interest, and ingest the tuples of ID and column values into the corresponding KHLL sketches. This allows for tables that contain arbitrarily large number of columns, and even for tables where data about the same user can be repeated across multiple table rows.

During the design and implementation of KHLL, we considered an alternative two-level KMV-based data structure. The high level idea is to store a KMV sketch of values, like KHLL, but to also use KMV in the second level instead of HLL. We named this *K2MV* given the two-level data structure of minimum hash values. To improve the performance, we implemented this as a single table storing all hash values instead of in multiple KMV stores. The size of the table is governed by parameters K_1, K_2, corresponding to the number of hashes stored at each level. We used an amortized strategy, where hash values are stored in sorted order, and where new hash values are written to a buffer which is periodically merged into the sorted list. We ran experiments and concluded that KHLL was more memory-efficient and suitable for our needs.

4.1.3 *Estimating reidentifiability and joinability using KHLL*

In this section, we explain how to use KHLL, presented in Section 4.1.2.4, to estimate reidentifiability and joinability, as defined in Section 4.1.2.1.

4.1.3.1 *Reidentifiability*

From a KHLL sketch of $(\mathcal{C}, \mathcal{ID})$, one can estimate both the cardinality of the column and the number of unique IDs associated with individual column values i.e., the uniqueness distribution. The latter allows us to efficiently estimate the proportion of column values that are reidentifying as well as statistics such as the min, max, and median uniqueness.

Evaluating the trade-off between data loss and reidentifiability

One can plot the uniqueness distribution to visualize the percentage of column values or IDs meeting some k-anonymity thresholds. Figure 4.2 is an example histogram of how many column values are associated with each number of unique IDs. Tweaked slightly, Figure 4.3 plots the cumulative percentage of values not meeting varying k-anonymity thresholds. A "more anonymous" dataset exhibits the curve on the left as most of the column values are expected to be associated with a large number of IDs. Conversely, a highly reidentifying dataset will exhibit the curve on the right.

The cumulative distribution is particularly useful as it estimates how much data will be lost when applying a k-anonymity threshold. This allows one to determine a threshold that preserves some data utility while ensuring a reasonable privacy protection, especially when other risk mitigation measures are in place such as access control, audit trails, limited data lifetime or noising of released statistics. We can see that for the left curve, one can choose a high threshold with low data loss, while on the right even a moderate threshold will result in dropping a large portion of the dataset.

Given the efficiency of KHLL analysis, one could set up periodic analyses to assure that the uniqueness distribution does not change over time, to monitor, for example, that no more than X% of values should have less than k-anonymity.

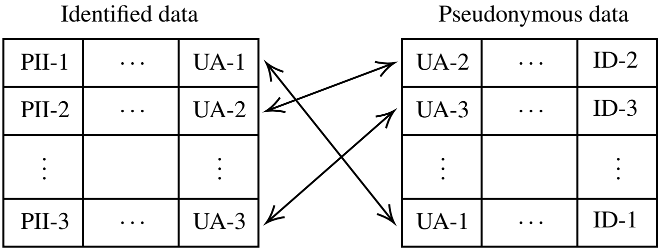
In addition to analyzing the uniqueness distribution of individual columns, KHLL can be used to analyze any combinations of columns, including a complete row. This can be done, for example, by simply concatenating the values of multiple columns, and treating the combination as a new column. The reidentifiability risk will grow as the number of dimensions increases. For example with movie recommendations, the combination of movie name, rating and date of recommendation can be highly unique [294].

Limitations and mitigations

Using a KHLL sketching algorithm with $K = 2048$ and 512 HLL buckets would give us an estimated error rate of 2% for the value cardinality and about 4% error rate for ID cardinalities. A higher error rate of ID cardinality estimates is tolerable given that we are more concerned about column values that associate with low number of IDs. In those cases, the algorithm will use HLL++ in sparse representation mode, which gives good estimates with minimal errors. Note that the trade-off between accuracy and efficiency is configurable.

Meanwhile, as a KHLL sketch effectively maintains K uniform random samples of column values, the estimated distribution does come with sampling bias. Specifically, it is possible that the estimated distribution may miss some outlier column values that associate with a large or small number of IDs. One mitigation is to run multiple analyses using different hash functions (with random seeds) to reduce the chance of outliers being consistently left out of the analysis.

4.1.3.2 *Joinability*



Identified data

PII-1	···	UA-1
PII-2	···	UA-2
⋮		⋮
PII-3	···	UA-3

Pseudonymous data

UA-2	···	ID-2
UA-3	···	ID-3
⋮		⋮
UA-1	···	ID-1

FIGURE 4.4: Example illustration of joinability between Personally Identifiable Information (PII) and pseudonymous IDs, with raw User Agent (UA) strings being the join key.

Estimating the joinability of two datasets through a pair of columns say C_1 and C_2 from their KHLL sketches is also straightforward. Recall that containment of C_1 in C_2 is given by $|C_1 \cap C_2| \,/\, |C_2|$. To compute this, we need only the cardinality of C_1 and C_2, plus the cardinality of their intersection.

Using the inclusion-exclusion principle, we estimate $|C_1 \cap C_2| = |C_1| + |C_2| - |C_1 \cup C_2|$. The union of C_1 and C_2 can be easily estimated by merging the KHLL sketch of C_1 and that of C_2. An alternative approach for computing the set intersection is by identifying the hash values that exist in both the KHLL sketches of C_1 and C_2 and computing the minmax of the K smallest hashes on both sides. This would allow us to quantify the error rate of individual set intersections directly, but the error rate will vary as the minmax of the hashes will vary for different pairs of columns. We prefer the simpler inclusion-exclusion-based approach.

In addition to determining the joinability of columns, KHLL sketches can provide insights into the potential joinability of identities in two datasets. For example, pseudonymous IDs in one dataset could be reidentified if the dataset is joinable with another dataset containing Personally Identifiable Information (PII), and if the join keys are associated with one pseudonymous ID and PII respectively (see Figure 4.4).

Specifically, given a pair of sketches of two columns C_1 and C_2 in datasets D_1 and D_2 respectively, we could estimate:

- whether C_1 is highly contained in C_2 or vice-vers;
- whether C_1 uniquely identifies $\mathcal{ID}_1$ in D_1;
- whether C_2 uniquely identifies $\mathcal{ID}_2$ in D_2.

KHLL allows us to estimate all the above conditions. Specifically, the first level of KHLL which is essentially a KMV sketch can be used to estimate the cardinality of C_1 and C_2, the cardinality of the intersection, and thus containment. Meanwhile, the second level of KHLL, consisting of HLL sketches, gives the uniqueness distribution and thus the ratio of uniquely identifying values easily.

Practical considerations for joinability analysis

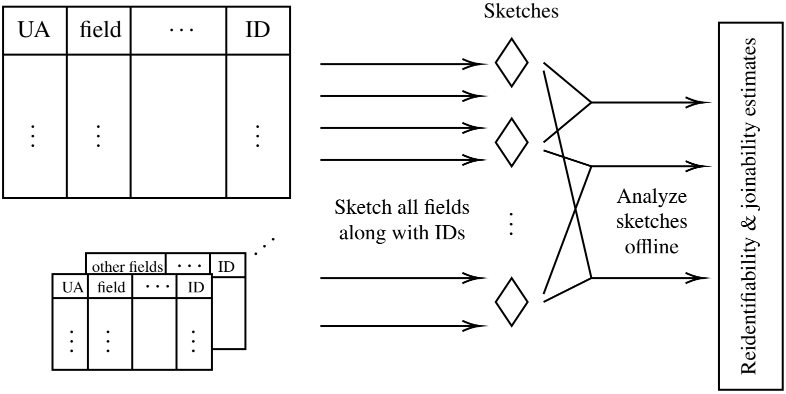

UA
field
...
ID
other fields
...
ID
UA
field
...
ID
Sketches
Sketch all fields along with IDs
Analyze sketches offline
Reidentifiability & joinability estimates

FIGURE 4.5: Two-step approach of reidentifiability and joinability analysis: (i) distributed scanners read various datasets to produce a KHLL sketch for every $C \times \mathcal{ID}$ tuple, (ii) various stats (including pairwise containment of datasets) are computed offline based on the sketches (rather than by comparing the raw datasets).

Estimating the joinability of large datasets is a hard problem. Naively estimating the pairwise joinability of datasets involves a quadratic number of full-table scans. The number of scans needed can increase quickly, especially for large datasets with many data columns.

However, as shown in Figure 4.5, KHLL allows us to estimate joinability from the sketches alone. This is a huge saving: it allows us to scan each dataset only once, and then pairwise compare the sketches rather than original datasets.

The sketching process can be agnostic to the underlying data when the schema of the datasets are well defined. For example when using protocol buffer messages [323] we can analyze new datasets without any need to configure information about columns, especially when the semantic types of data columns are properly annotated [341].

The sketching process can also be distributed. The respective data owners do not need to grant a central service access to the raw data, but simply to agree on the sketching method and to upload the sketches to a central repository. The sketches containing the hash values of potentially sensitive data including user IDs should still be treated with care such as by limiting the access and ensuring a short lifetime.

Limitations and mitigations

Using sketches for joinability analysis comes with some risks, both of reporting columns are potential join keys even though they do not present an actual risk of joinability, and of missing potential join keys.

First, note that the containment metric is agnostic to the semantics of the underlying data. Specifically, containment (or Jaccard) does not distinguish between columns that use a similar range of values but are semantically different. As an example, we could falsely determine `port_number` and `seconds_till_midnight` columns to be joinable since they both have an extensive overlap in the integer range of $[0, 86400)$: this can induce *false positives*. The rate of false positives can be mitigated by requiring a large cardinality threshold on the potential join keys.

Second, *false negatives* are also possible, and can arise for multiple reasons. For example, the containment metric will fail to detect similar columns that have been encoded differently (e.g., base64 versus raw string) or have undergone some slight transformations (e.g., a microsecond timestamp versus the coarsened millisecond version). This is a hard problem in practice. The system could potentially support some common transformations or encodings when the semantic type of a data column is known, but there is no way to handle all possibilities.

The containment metric can also be unreliable when set sizes are highly skewed. When the expected error rate of a set is larger than the cardinality of a much smaller set, the estimate for the set intersection computed using the inclusion- exclusion principle will be unreliable. One could potentially complement the containment metric with some other similarity scores like the similarity between the frequency distribution of the potential join keys.

Importantly, while KHLL can evaluate the pairwise joinability of datasets based on individual columns, estimating the joinability of datasets through arbitrary combinations of columns remains practically infeasible given that intractable number of potential column combinations. One could however systematically approach this by testing the joinability between combinations of high-risk columns, for example, involving only those that have high uniqueness.

Finally, pairwise joinability analysis does not readily detect multi-hop joinability. For example, when a dataset D_1 is joinable with dataset D_2, and D_2 is joinable with dataset D_3 through two different pairs of join keys, we will not detect that D_1 is joinable to D_3.

Such multi-hop joinability analysis could be similarly estimated using clustering and graph traversal algorithms such as label propagation [412].

4.1.4 *Experimental validation*

In this section, we validate the efficiency and accuracy of KHLL experimentally. Efficiency is validated in a production environment, so the experimental results provided are as realistic as possible. To provide reproducible validation results about accuracy, we have implemented a version of the KHLL algorithm in BigQuery Standard SQL, and simulated the computation of uniqueness distribution and joinability using publicly available datasets. The code for the experiments can likely be adapted for other SQL engines, and is shared on GitHub [143].

4.1.4.1 *Efficiency*

The KHLL algorithm and the metrics we estimate with it have been implemented in a proprietary production environment in Go using the MapReduce programming model [95]. For each column (or specified combinations of columns) in the dataset, the MapReduce outputs a KHLL sketch in one pass.

To reason about efficiency, we implemented two naive MapReduce algorithms. The first one, Count Exact (CE), computes the exact variants of the metrics that KHLL estimates. The second one, Count Exact Single (CES), computes the same set of metrics exactly, but analyzes only one single column during a given MapReduce run.

We designed the CE MapReduce to output, for each column, a dictionary of column values to ID sets (allowing the computation of various statistics that a KHLL sketch approximates). One would expect that emitting the entire column-value-to-ID-set dictionary would result in substantial memory overhead. The CES MapReduce is a more realistic simplification of CE which outputs the tuples of column values and ID sets of only one single specific column.

The test dataset on which we ran our performance analyses represents the data-flow graph of various production operations at Google. Each row in this dataset represents a node in the data-flow graph and has about 50 data columns describing the properties of the operations as well as the input and output data. One of the data columns specifies the globally unique name of the machine where the operation is run or controlled. We used this machine name as the ID column in our analyses. Note that this dataset does not contain any user data and is not privacy sensitive as these are not necessary for performance measurements.

We ran KHLL, CE and CES on several subsets of the test dataset in a shared computational cluster at Google. These analyses were provided computational resources at a priority class that is typically used for batch jobs. Measuring the performance metrics of jobs in a shared computational cluster is not straightforward since any given machine can host multiple unrelated jobs with varying priority classes that can consume machine resources in unpredictable ways. So we focused on the performance metrics which a typical customer of a commercial computational cluster (e.g., Amazon EC2, Google GCE) would pay for.

Table 4.1 shows the performance metrics of the MapReduce runs. As one can see, KHLL is consistently more efficient than CE across various metrics. Performance differs by 1 or 2 orders of magnitude even for the relatively small datasets in our experiment. In fact, the CE

MapReduce type	Input size	Algorithm	CPU usage (vCPUs)	RAM usage (GBs)	Peak RAM (GB)	Output size (GB)	Runtime (s)
All columns	1 GB	CE	4.01e+3	1.14e+4	9.34e+0	1.04e+0	5.83e+2
		KHLL	9.78e+2	2.08e+3	9.45e-1	1.60e-3	1.43e+2
	100 GB	CE	3.53e+5	3.40e+6	1.10e+2	2.64e+0	1.93e+4
		KHLL	6.25e+4	3.11e+4	2.00e+0	3.46e-3	2.63e+2
	1 TB	CE	(n.a.)	(n.a.)	(n.a.)	(n.a.)	(n.a.)
		KHLL	9.92e+5	2.37e+6	2.52e+0	3.50e-3	1.13e+4
Single column	1 GB	CES	7.23e+2	4.76e+3	8.07e-1	1.57e-2	5.76e+2
	100 GB	CES	4.47e+4	1.30e+5	1.79e+0	2.35e-1	1.10e+3

Table 4.1: Performance metrics of KHLL and exact counting algorithms. We configured KHLL to have K=2048 and M=1024. 1 GBs = 1 GB of RAM used for 1 second. Virtual CPU (vCPU) is a platform-neutral measurement for CPU resources. 1 vCPUs = 1 vCPU used for 1 second. The CE MapReduce for analyzing all data columns in a 1 TB dataset was excessively expensive and was halted.

MapReduce jobs for analyzing all data columns in an 1 TB dataset became too expensive to be completed in the shared computational cluster. Interestingly, per our test dataset, it is even more memory efficient (though slightly slower) to compute the KHLL sketches of all data columns in a single MapReduce run, than to analyze a single data column using CES. This performance disparity is critical in practice, as it is the difference between an analysis that is feasible to run, and one that is too slow to be worthwhile.

In various production setups at Google, KHLL scales to analyze hundreds of datasets, each containing potentially trillions of rows and tens of thousands of data columns measuring petabytes in sizes, to produce millions of sketches in each run.

4.1.4.2 *Accuracy of uniqueness distribution estimation*

We measured the accuracy of the estimated uniqueness distribution using three publicly available datasets. The first two are taken from the Netflix Prize dataset which was released in 2006 and shown to be reidentifying for a significant fraction of the users [294]. We estimate the uniqueness distribution of (a) movies, and (b) tuples of movie and date. Note that we do not consider the entire list of movies (or respectively all pairs of movie and date) associated with individual pseudo-identifiers. We could analyze this easily but it will be less interesting for our validation purposes, as most of the values will be unique. The third dataset is the 2010 US Census [96] through which we count the number of distinct individuals associated with a given (ZIP code, age) tuple. The corresponding uniqueness distribution gives an indication of the reidentifiability of these quasi-identifiers within the US population. As we will see, the three datasets present different uniqueness profiles, allowing us to test the accuracy of KHLL estimation in different situations.

We simulate the KHLL algorithm, with parameters $K = 2048$ and $M = 1024$. We learned from our production settings that $K = 2048$ gives a good trade-off between precision and memory usage. $M = 1048$ was chosen as the smallest possible parameter of the HLL++ library available in BigQuery Standard SQL. We use $M = 512$ in our production pipelines, which gives a comparable degree of precision in ID cardinality estimation (4% vs. 3%). As

HLL++ counts elements exactly at lower cardinalities, this has a negligible influence on our estimations. For each dataset, we compare the KHLL-based estimate to the true distribution, which is computed exactly (without approximation) using standard SQL operators.

Figure 4.6a plots the cumulative uniqueness distribution of movies in the Netflix Prize dataset. It allows an analyst to quickly answer the question: how much data do we lose if we only release the movies which have been rated by more than k users, for all possible values of k. The uniqueness of movies is low: the median number of associated users per movie is larger than 500. Figure 4.6b plots the cumulative uniqueness distribution of the tuples of movie and date. The typical uniqueness of this setting is high: over 80% of the (movie, date) tuples are associated with 10 or less unique IDs.

Figure 4.6c shows the cumulative uniqueness distribution of tuples (ZIP code, age) in the US census. Each individual in the census dataset appears in only a single row, different from the case with Netflix datasets where ratings from the same user exist on several separate records. The uniqueness of (ZIP code, age) tuples is variable: a significant portion of possible values is associated to only a few individuals, but many of the (ZIP code, age) tuples associate with larger than 100 individuals.

Across all three experiments, we observe that the estimate of uniqueness distribution using KHLL is accurate.

4.1.4.3 *Accuracy of joinability and containment estimation*

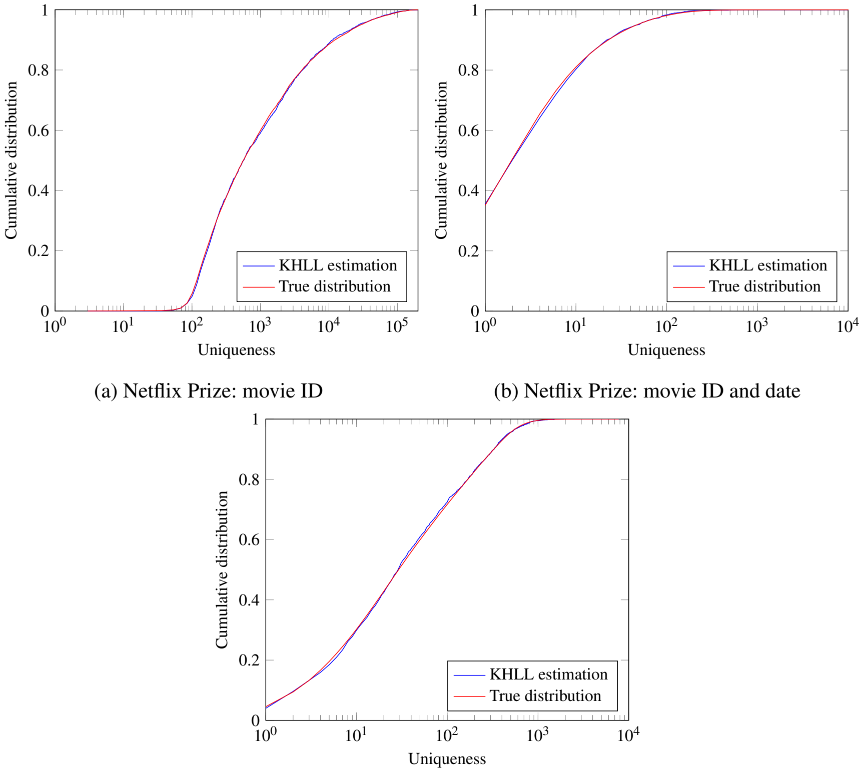
As described in Section 4.1.3.2, joinability is most interesting from the privacy perspective when a pseudonymous ID space becomes joinable with PII. Specifically if columns C_1 and C_2 are joinable, and that C_1 uniquely identifies a pseudonymous ID space while C_2 uniquely identifies PII. Three conditions are important for estimating such risk: the ratio of C_1 values uniquely identifying $\mathcal{ID}_1$, the ratio of C_2 values uniquely identifying $\mathcal{ID}_2$, and the containment of C_1 in C_2 (or containment of C_2 in C_1).

The estimate of ratio of column values uniquely identifying an ID can be seen as an estimator of the parameter p based on an observation of a binomial distribution of parameters K and p. It is well-known that a binomial distribution of parameters K and p has a variance of $p(1-p)K$, so the estimator which divides the result of the distribution by K has a variance of $p(1-p)/K$, or a standard distribution of $\sqrt{p(1-p)/K}$. Therefore, we focus our experiments on estimating the containment metric, as defined in Definition 77.

Using $K = 2048$ hashes, and assuming C_1 and C_2 have the same cardinality, the estimate of containment falls within ±5% of the true value over 90% of the time, and always stays within 10% of the true value. This is true regardless of the cardinality of the intersection of C_1 and C_2. Figure 4.7 shows the median as well as the 5th and 95th percentiles of the containment estimation, for cardinalities of 10,000 and 10,000,000.

When C_1 and C_2 have different cardinalities, however, precision can suffer. Figure 4.8 shows the median as well as the 5th and 95th percentiles of the estimation of $|C_1 \cap C_2| / |C_1|$, where true value is fixed at 50%, $|C_1| = 100{,}000$, and $|C_2|$ varies between 50,000 and 2,000,000 (so, the cardinality ratio $|C_2| / |C_1|$ ranges from 0.5 to 20).

We can observe that the larger the cardinality ratio gets, the worse the precision becomes. This is expected: since we compute $|C_1 \cap C_2|$ using the inclusion-exclusion principle, and



(a) Netflix Prize: movie ID

(b) Netflix Prize: movie ID and date

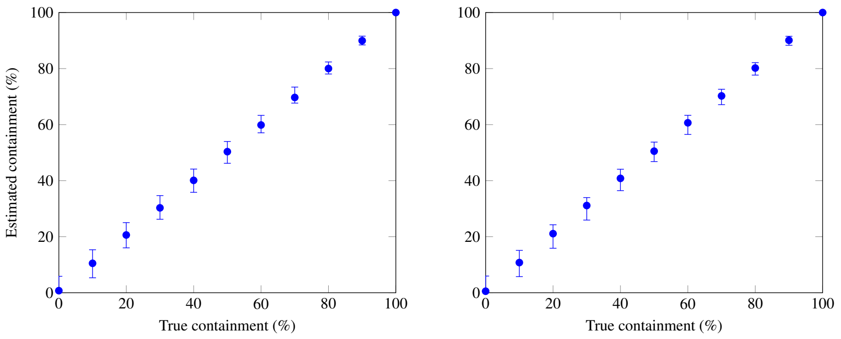
(c) US Census: ZIP code and age

FIGURE 4.6: Estimation of uniqueness distribution in different datasets using KHLL as compared to true distribution (computed exactly without approximation).

the error of the estimation is proportional to the cardinality estimated, the estimation error of $|C_1 \cap C_2|$ should be roughly proportional to $\max\left(|C_1|, |C_2|\right)$. Since the value of $|C_1 \cap C_2|$ is roughly proportional to $\min\left(|C_1|, |C_2|\right)$, the error ratio of the containment estimation will grow linearly with the cardinality ratio. This is what we observe in practice.

4.1.5 *Summary and perspectives*

The scale of data and systems in large organizations demands an efficient approach for reidentifiability and joinability analysis. KHyperLogLog (KHLL) innovates on approximate counting techniques to estimate the uniqueness distribution and pairwise containment of very large databases at scale.



(a) Low cardinality: 10,000

(b) High cardinality: 10,000,000

Figure 4.7: Estimation of containment with fixed cardinalities. The bars indicate the 95th and 5th percentiles of the estimates.

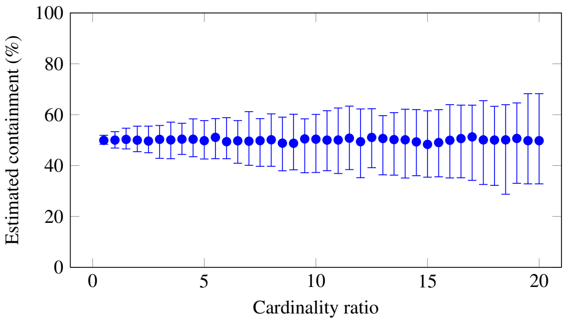


Figure 4.8: Estimation of containment with varying cardinality ratio. The true containment is 50%. The bars indicate the 95th and 5th percentiles of the estimates.

The efficiency and scalability of risk analysis using KHLL represent a practical and useful tool for large organizations in protecting user privacy. It provides an objective, quantifiable and replicable measure of reidentifability of datasets. The KHLL algorithm also presents a novel and practical approach for tackling the joinability risks of datasets and ID spaces. The efficiency of KHLL further enables periodic analyses of complex production systems that evolve over time.

4.1.5.1 *Related work*

Using the frequency of (combinations of) column values as a proxy to measure reidentifiability is not new: this is the core intuition behind k-anonymity, which we introduced in Section 2.1.1. Rather than just estimating k-anonymity, the KHLL algorithm estimates the entire uniqueness distribution, which is useful for evaluating the impact to data loss with k-anonymization. The reidentifiability and joinability risks as estimated using KHLL can serve to help determine a suitable anonymization strategy, particularly when considering the different contexts and use cases of anonymization.

The problem of gathering metadata and organizing datasets in large organizations has been described on several occasions. In [183], the authors detail a search engine for datasets, which gathers metadata at dataset level; while in [342], the authors explain how to detect and propagate column-level semantic annotations using assisted labeling and data-flow analysis. The tools we develop to automatically detect joinability of datasets could be used for a similar purpose. Inconsistent semantic annotations between data columns with high similarity scores (containment or Jaccard) can be automatically detected with the correct annotations propagated accordingly.

Cardinality estimators have been the subject of a significant body of research [45, 152, 195, 394]. The techniques we propose are largely independent of the particular algorithm chosen to approximate ID cardinality. Beyond privacy, estimating value distribution is also essential to database query optimization and data stream processing. Most research in this domain focuses on sketching values of high frequency (i.e. statistical heavy hitters or top-k queries). The closest analogue of KHLL was presented by Cormode et al. [86]; it combines the Count-Min sketch [85] and the LogLog sketch [153] for value distribution estimation. However, the Count-Min algorithm biases towards values of high frequency, which is not helpful for evaluating the impact of k-anonymity given the typical choices of k are much smaller than the frequency of heavy hitters.

MinHash [55] and SimHash [66] are two popular algorithms for estimating the Jaccard similarity of datasets. The KHLL algorithm leverages the K Minimum Values in the first level of the two-level sketch for estimating Jaccard and containment scores, using a similar $\log n$-space memory footprint. A possible improvement might be to adapt from HyperMinHash [402], capable of estimating Jaccard using only $\log \log n$-space. Yet, given that the bulk of memory usage by KHLL actually comes from the second level of the sketch for estimating the uniqueness distribution, we have not explored the feasibility of adapting KHLL to have a HyperMinHash-like data structure in the first level.

Finally, despite the extensive research for detecting data similarity, we have not seen any prior work tackling the problem of automatically detecting possible joinability between different ID spaces across datasets.

4.1.5.2 *Future work*

While KHyperLogLog is memory efficient, it still requires a linear pass over the data, which in practice is the main performance bottleneck. Techniques to produce sketches suitable for joinability analysis without scanning the entire dataset would be helpful, for example by sampling input records. It would also be interesting to pursue further innovative uses of cardinality estimation techniques in privacy enhancing technologies, for example for automatically adding semantic labels to datasets, detecting violations of retention policies, or for suggesting effective anonymization strategies.

4.2 BUILDING A USABLE DIFFERENTIALLY PRIVATE QUERY ENGINE

In the previous section, we saw that risk analysis use cases, where datasets are generated via an unknown process and we are trying to quantify their risk, cannot be easily tackled with mechanism-centric definitions like differential privacy. This is only a partial answer to the original question of this chapter: for all cases where we *can* control the data generation process, why is differential privacy not more widely used? Since its introduction, differentially private variants have been proposed for most common data analysis tasks: from simple aggregations, like counts [128], sums, or medians [127, 303], to more complex analyses, like k-means clustering [303] or empirical risk minimization [71]. Yet, real-world deployments of such algorithms seem to be few and far between.

In this section, we argue that one of the main reasons for this state of affairs is *usability*. On paper, differential privacy is relatively simple, but in practice, few implementations are readily available, and few people will go through the trouble of implementing an algorithm from a scientific paper by themselves. The lack of existing implementations is partially explained by a misalignment of incentives: academics get more recognition and career progress out of scientific papers than usable open-source software. But this is only a partial explanation: it is also surprisingly difficult to convert algorithms from the scientific literature into a usable, scalable, and solid implementation.

The potential issues are numerous. Basic primitives such as noise addition are often modeled using continuous numbers with infinite precision; and translating them naively to finite computers using floating-point numbers can lead to significant vulnerabilities. Many scientific papers make assumptions that are unrealistic in many practical use cases, like the implicit premise that each user contributes a single data point to the input dataset, that the aggregations happen along partitions that are known in advance, or that the input dataset can fit entirely in memory. Production applications also require a high degree of trust in the implementation, and it is non-trivial to adapt software reliability techniques like unit testing to randomized algorithms. Further, a tool for differential privacy must be usable by non-experts, and this requires to put significant thought into interface design.

In this section, we propose a generic and scalable framework to perform differentially private aggregations on databases, and use it as an example to introduce possible approaches to tackle the problems listed above. This framework, which we express as an operator in relational algebra, provides differential privacy even when each individual can each be associated with arbitrarily many records. To validate this system, we test the utility of typical queries on industry benchmarks, and verify its correctness with a stochastic testing framework. We highlight the benefits provided by such a query engine and the pitfalls encountered in the course of its deployment. Finally, we publish one of its implementations as open-source software.

4.2.1 *Introduction*

Query engines are a major analysis tool for data scientists, and one of the most common ways for analysts to write queries is with Structured Query Language (SQL). As a result, multiple query engines have been developed to enable data analysis while enforcing DP [42, 209, 233, 273], and all of them use a SQL-like syntax.

However, as we discuss in Section 4.2.2, these differentially private query engines make some implicit assumptions, notably that each individual in the underlying dataset is associated with at most one dataset record. This does not hold in many real-world datasets, so the privacy guarantee offered by these systems is weaker than advertised for those datasets. To overcome this limitation, we introduce a generic mechanism for bounding user contributions to a large class of differentially private aggregate functions. We then propose a design for a SQL engine using these contribution bounding mechanisms to enforce DP, even when a given individual can be associated with arbitrarily many records or when the query contains joins.

Our work goes beyond this design and accompanying analysis: we also describe the implementation of these mechanisms as part of a SQL engine, and the challenges encountered in the process. We describe the testing framework we use to increase our level of trust in the system's robustness. To aid in reproducing of our work and encourage wider adoption of differential privacy, we release core components of the system, as well as a distinct implementation of this framework, as open-source software.

4.2.1.1 *Requirements and contributions*

To be useful for non-expert analysts, a differentially private SQL engine must satisfy at least the following requirements.

- It must make realistic assumptions about the data, specifically allowing multiple records to be associated with an individual user.
- It must support typical data analysis operations, such as counts, sums, means, percentiles, etc.
- It must provide analysts with information about the accuracy of the queries returned by the engine, and uphold clear privacy guarantees.
- It must provide a way to test the integrity of the engine and validate the engine's privacy claims.

In this work, we present a differentially private SQL engine that satisfies these requirements. Our contributions are as follows.

- We detail how we use the concept of *row ownership* to enforce the original meaning of differential privacy: the output of the analysis does not reveal anything about a single *individual*. In our engine, multiple rows can be associated with the same "owner" (hereafter referred to as a *user*, although the owner could also be a group), and the differential privacy property is enforced at the user level.
- We implement common aggregations (counts, sums, medians, etc.), arbitrary per-record transforms, and joins on the row owner column as part of our engine. To do so, we provide a method of bounding query sensitivity and stability across transforms and joins, and a mechanism to enforce row ownership throughout the query transformation.

- We detail some of the usability challenges that arise when trying to use such a system in production and increase its adoption. In particular, we explain how we communicate the accuracy impact of differential privacy to analysts, and we experimentally verify that the noise levels are acceptable in typical conditions. We also propose an algorithm for automatic sensitivity determination.

- We present a testing framework to help verify that ε-DP aggregation functions are correctly implemented, and can be used to detect software regressions that break the privacy guarantees.

We hope that this work, and the associated open-source release, can increase the appropriate adoption of differential privacy by providing a usable system based on popular tools used by data analysts.

4.2.1.2 *Related work*

Multiple differentially private query engines have been proposed in the literature. In this work, we mainly compare our system to two existing differentially private query engines: PINQ [273] and Flex [209]. Our work differs in two major ways from these engines: we support the common case where a single user is associated with multiple rows, and we support arbitrary `GROUP BY` statements.

Another line of research focuses on building frameworks to define differentially private algorithms: examples include are Airavat [334], Ektelo [404] and OpenDP's programming framework [156]. These are building blocks that help write correct differentially private algorithms, but require significant changes in how programs are written, and we argue that they cannot be used as is without prior expertise on differential privacy.

In these systems, a single organization is assumed to hold all the raw data. Query engines can also be used in other contexts: differential privacy can be used in concert with secure multiparty computation techniques to enable join queries between datasets held by different organizations, systems such as DJoin [293] and Shrinkwrap [42] tackle this specific use case.

A significant amount of research focuses on improving the accuracy of query results while still maintaining differential privacy. In this work, for clarity, we keep the description of our system conceptually simple, and explicitly do not make use of techniques like smooth sensitivity [303], tight privacy budget computation methods [213, 278], variants of the differential privacy definition (Section 2.2), adjustment of noise levels to a pre-specified set of queries [249], or generation of differentially private synthetic data to answer arbitrarily many queries afterwards [49, 232, 233]. We revisit certain design choices and outline possible improvements later, in Section 4.3.

The testing framework we introduce in Section 4.2.6.3 is similar to recent work in verification for differential privacy [47, 109, 167], and was developed independently. Other approaches use semantic analysis, possibly augmented with automated proving techniques, to certify that algorithms are differentially private [32, 33, 292].

Our work is not the first to use noise and thresholding to preserve privacy: this method was originally proposed in [174, 230] in the specific context of releasing search logs with (ε, δ)-DP; our work can be seen as an extension and generalization of this insight. Diffix [155] is another system using similar primitives; however, it does not provide any formal privacy guarantee, and has been shown to be vulnerable to attacks [79, 80, 157], so a meaningful comparison with our work is not feasible. In Section 4.2.4, we provide a comparison of query accuracy between our work, PINQ, and Flex.

4.2.1.3 *Preliminaries*

We introduce here the definitions and notations used throughout this section, which are also summarized on page xxi. Because a significant part of this work specifically focuses on the case where a single user contributes multiple records to the dataset, we no longer consider a dataset as a sequence of records in $\mathcal{T}$ but as a sequence of *rows*, where each row is a pair $(\text{id}, t) \in \mathcal{ID} \times \mathcal{T}$: each row is a record associated with a specific user.

Definition 78 (Distance between datasets). *We denote* row-level change *the addition or removal of a single row from a dataset, and* user-level change *the addition or removal of all rows associated with a user. Given two datasets D_1 and D_2, we denote $\|D_1 - D_2\|$ the minimum number of row-level changes necessary to transform D_1 into D_2, and $\|D_1 - D_2\|_u$ the minimum number of user-level changes necessary to transform D_1 into D_2; we call them* row-level distance *and* user-level distance *respectively.*

In the original definition of (ε, δ)-differential privacy, each user is implicitly assumed to contribute a single record. Since we want to consider the case where this is not true, we define two variants of differential privacy to make this distinction explicit.

Definition 79 ((ε, δ)-row-level and user-level differential privacy). *A randomized mechanism $\mathcal{M}$ satisfies* row-level (ε, δ)-DP *(respectively* row-level (ε, δ)-DP*) if for all pairs of datasets $D_1, D_2 \in \mathcal{D}$ that satisfy $\|D_1 - D_2\| = 1$ (respectively $\|D_1 - D_2\|_u = 1$), $\mathcal{M}(D_1) \approx_{\varepsilon,\delta} \mathcal{M}(D_2)$.*

As previously, ε-DP is an alias for $(\varepsilon, 0)$-DP, and $\approx_{\varepsilon,\delta}$ denotes (ε, δ)-indistinguishability (see Definition 14).

Note that this notion is technically *unbounded differential privacy*, defined in [227] and mentioned in Section 2.2.3.1: we only allow neighboring datasets to differ in a row (or all rows associated with a user) that has been added or removed, not changed. Up to a change in parameters, it is equivalent to the classical definitions, but we found that this choice significantly simplifies the analysis and the implementation of differential privacy tooling.

Finally, let us define L^1-sensitivity, defined on functions that take a dataset as input and return a vector in $\mathbb{R}^d$, for some integer d.

Definition 80 (L^1-Sensitivity). *Let $\|\cdot\|_1$ be the L^1-norm on $\mathbb{R}^d$. The* global L^1-sensitivity Δf *of a function $f : \mathcal{D} \to \mathbb{R}^d$ is the smallest number such that:*

$$\left\|f(D_1) - f(D_2)\right\|_1 \leq \Delta f$$

for all D_1 and D_2 such that $\|D_1 - D_2\| = 1$. The user-global L^1-sensitivity $\Delta_u f$ *of f is the smallest number such that the above holds for all D_1 and D_2 such that $\|D_1 - D_2\|_u = 1$.*

4.2.2 *A simple example: histograms*

Before describing the technical details of our system, let us first give an intuition of how it works using a simple example: histogram queries. Consider a simple dataset that logs accesses to a given website. An analyst wants to know which browser agents are most commonly used among users visiting the page. A typical query to do so is presented in Listing 4.1.

```
SELECT
  browser_agent,
  COUNT(*) AS visits
FROM access_logs
GROUP BY browser_agent
```

Listing 4.1: Simple histogram query

How would one make this simple operation ε-differentially private? One naive approach is to add Laplace noise of scale $1/\varepsilon$ to each count, as described in Section 2.1.6.4. Unfortunately, this solution suffers from several shortcomings.

4.2.2.1 *First pitfall: multiple contributions within a partition*

It might look like the naive approach correctly hides the existence of individual *records* from the dataset: each record of the access log will only influence one of the returned counts by at most 1, and it is well known [128] that this basic mechanism provides ε-DP. However, it fails to protect the existence of individual *users*: the same user could have visited the example page many times with a particular browser agent, and therefore could have contributed an arbitrarily large number of rows to `visits` for a particular `GROUP BY` partition, violating our assumption that the query sensitivity is 1.

In PINQ and Flex, the differential privacy definition explicitly considers records as the privacy unit. Because we instead want to protect the full contribution of users, we need to explicitly include a notion of a user in our system design. In this work, we do this via the notion of a *user identifier*, or *user ID*.

Because Listing 4.1 has unbounded sensitivity, adding noise to counts is not enough to enforce differential privacy; we need to *bound user contribution to each partition*. This can be addressed is by counting distinct users, which has a user-global sensitivity of 1, instead of counting rows. Although this modifies query semantics, we chose this approach to keep the example simple. We present the modified query in Listing 4.2, which assumes that the column containing user ID information is named `uid`.

```
SELECT
  browser_agent,
  COUNT(DISTINCT uid) + Laplace(1/ε)
FROM access_logs
GROUP BY browser_agent
```

Listing 4.2: Per-partition contribution bounding

In other contexts, it might make more sense to allow a user to contribute more than once to a partition. For example, we might want to count up to five visits from each distinct user with each distinct browser agent. In this case, we would need to further modify the query to allow multiple contributions and increase sensitivity to match the maximum number of contributions.

4.2.2.2 *Second pitfall: leaking partitions*

Even if we bound the contribution of a user to each partition and adapt noise levels accordingly, the query is still not ε-DP. Indeed, suppose that the attacker is trying to distinguish between two datasets differing in only one record, but this record is a unique browser agent BA_{unique}: this browser agent does not appear in D_1, but appears once in D_2. Then, irrespective of the value of the noisy counts, the GROUP BY partitions themselves are enough to distinguish between the two datasets simply by looking at the output: BA_{unique} will appear in the query output for D_2 but not for D_1.

A simple solution to this problem was proposed in [230]: the idea is to drop from the results all partitions associated with a noisy count lower than a certain threshold τ. τ is chosen independently of the data, and the resulting process is (ε, δ)-DP with $\delta > 0$. We call this mechanism *τ-thresholding*. With a sufficiently high τ, the output rows with partitions present in D_2 but not D_1 (and vice-versa) will be dropped with high probability, making the partitions indistinguishable to an attacker. A longer discussion on the relation between ε, δ and τ can be found in Section 4.2.3.6. This approach is represented in SQL in Listing 4.3.

```sql
SELECT
  browser_agent,
  COUNT(DISTINCT uid) + Laplace(1/ε) AS c
FROM access_logs
GROUP BY browser_agent
HAVING c >= τ
```

Listing 4.3: GROUP BY filtering

PINQ and Flex handle this issue by requiring the analyst to enumerate all partitions to use in a GROUP BY operation, and return noisy counts for only and all such partitions[1]. This enforces ε-DP but impairs usability: the range of possible values is often large (potentially the set of all strings) and difficult to enumerate, especially if the analyst cannot look at the raw data.

Some data synthesis algorithms have been proposed to release ε-DP histograms over an unbounded set of partitions [275], but those seem to be limited to datasets subject to hierarchical decomposition. Our approach is simpler and more generic, at some cost in the privacy guarantee.

[1] The open-source implementation of Flex [208], however, does not appear to implement this requirement.

4.2.2.3 *Third pitfall: contributions to multiple partitions*

Finally, we must consider the possibility of a user contributing to multiple partitions in our query. Imagine a user visiting the example page with many different browsers, each with a different browser agent. Such a user could potentially contribute a value of 1 to each partition's count, changing the sensitivity of the query to be the number of partitions, which is unbounded!

Because both PINQ and Flex consider records as the privacy unit, this is not an issue for their privacy models. So long as they are only used on datasets where that requirement holds true, and where the sensitivity and stability impact of joins (and related operations) are carefully considered, they will provide adequate DP guarantees. However as shown in [276], these conditions are not always true.

Instead of adding strict requirements on the nature of the underlying dataset and on how joins are used, we introduce a novel mechanism for *bounding user contribution across partitions*. Concretely, we first choose a number κ, and for each user, we randomly keep the contributions to κ partitions for this user, dropping contributions to other partitions. This operation allows us to bound the global sensitivity of the aggregation: each user can then influence at most unique κ counts, and we can adapt the noise level added to each count, by using Laplace noise of scale κ/ε.

The final version of our query is shown in Listing 4.4. It uses a non-standard variant of the SQL `TABLESAMPLE` operator, which supports partitioning and reservoir sampling, to represent the mechanism we introduced. This final version satisfies (ε, δ)-differential privacy for well-chosen parameters.

```
SELECT
  browser_agent,
  COUNT(DISTINCT uid) + Laplace(κ/ε) AS c
FROM (
  SELECT browser_agent, uid
  FROM access_logs
  GROUP BY browser_agent, uid
)
TABLESAMPLE RESERVOIR (κ ROWS PARTITION BY uid)
GROUP BY browser_agent
HAVING c >= τ
```

LISTING 4.4: An (ε, δ)-DP query

In the remainder of this section, we formalize this approach, and adapt it to a larger set of operations. In particular, we extend it to arbitrary aggregations with bounded sensitivity, and we explain how to make this model compatible with joins.

4.2.3 *System model and design*

In this section, we present an overview of the design of our framework, detail the building blocks necessary to implement it, and formally define it as an operator in relational algebra. We finish by providing a proof that the entire process provides user-level differential privacy.

4.2.3.1 *Overview*

As suggested in Section 4.2.1.3, we assume that there is a special column of the input dataset that specifies which user owns each row. The system is agnostic to the semantics of this special column. In principle, it can be any unit of privacy that we need to protect: a device identifier, an organization, or even a unique row ID if we want to protect rows and not users. For simplicity of notation we assume that this special column is a user identifier. Users may own multiple rows in each input table, and each row must be owned by exactly one user. Our model guarantees (ε, δ)-DP with respect to each user, as defined in Definition 79.

Our DP query engine uses the underlying SQL engine to track user ID metadata across tables, and invokes a DP query rewriter when our anonymization query syntax is used on tables containing user data and any applicable permission checks succeed. Listing 4.5 provides an example of a SQL query accepted by our system.

```
SELECT WITH ANONYMIZATION
  T1.partition_key,
  ANON_SUM(T2.val, 0, 1)
FROM Table1 T1 JOIN Table2 T2 USING(uid)
GROUP BY T1.partition_key
```

Listing 4.5: Anonymization query example

The query rewriter decomposes such queries into two stages, one before and one after our introduced DP aggregation operator, invoked with `SELECT WITH ANONYMIZATION`.

The first stage begins by validating that all table subqueries inside the DP operator's `FROM` clause enforce unique user ownership of intermediate rows, which is informally described in Section 4.2.3.2. Next, for each row in the subquery result relation, our operator partitions all input rows by the vector of user-specified `GROUP BY` keys and the user identifier, and applies an intermediate non-private SQL aggregation to each group. For example, for `ANON_SUM`, we use `SUM` as a partial aggregation. Adjusting query semantics in this way is necessary to ensure that, in the next stage, the cross-user aggregations of each partition receive only one input row per user, this is described in Section 4.2.3.3.

For the second stage, we sample a fixed number of these partially aggregated rows for each user to limit user contribution *across partitions*, which is described in Section 4.2.3.4. We then compute a cross-user DP aggregation across all users contributing to each `GROUP BY` partition, limiting user contribution *within partitions*; this is described in Section 4.2.3.5. Finally, we randomly drop certain partitions that contain too few users, to guarantee that the list of published partitions does not break differential privacy guarantees, a process described in Section 4.2.3.6.

The full process is formalized in Section 4.2.3.7, where we define our operator in the language of relational algebra.

4.2.3.2 *Allowed subqueries*

SQL queries can, in principle, contain subqueries that perform arbitrary transformations on the input tables. However, for our framework to guarantee differential privacy, we need to limit this capability: we need the concept of row ownership to stay relevant throughout the computation, and be able to determine which row of the subquery output belongs to which user in the input dataset.

Thus, the relational operators present in the subquery must not create any intermediate objects of shared ownership: no intermediate row may be derived from rows owned by different users. A naive application of certain relational operators violate this requirement. For example, for the aggregation operator, rows owned by distinct users may be aggregated into the same partition. Then the resulting row from that partition will be owned by all users whose data are contributed to the group. For the naive join operator, two rows from distinct users may be joined together, creating a row owned by both users.

This restriction limits our system since some queries cannot be run. We observed that in practice, most use-cases can be fit within these constraints, and leave extensions to a wider class of queries for future work. This might require a different model than the one presented here, but changing query semantics will always be necessary for queries with unbounded sensitivity.

Assume that the column containing user identifier is named `uid`. We address the shared ownership issue by restricting each operator composing the subquery such that, for each row in that operator's output relation, that row is derived only from rows in the input relation that have matching `uid` attributes. We enforce this rule for aggregate operators by requiring that the analyst additionally groups-by the input relation's `uid` attribute. For join operators, we require that the analyst adds a `USING(uid)` clause (or equivalent) to the join condition. Additionally, each operator of T must propagate `uid` from the input relation(s) to the output relation. In queries where this is unambiguous (i.e., the analyst does not refer to `uid` in the query), we can automatically propagate `uid`.

A formalization of this approach, which list alternatives for each basic operator, is listed in Table 4.3, in Section 4.2.3.7. They are enforced recursively during the query rewrite for each operator that composes the subquery.

4.2.3.3 *Per-user aggregation*

First, we aggregate the data of each user per-partition, using the non-private version of the aggregator in the original query. For example, for `ANON_SUM`, we use the `SUM` SQL operator to sum all contributions of each user in each partition.

This operation guarantees that in the cross-user aggregation stage, each user contributes at most once to each aggregation. This does come at the cost of introducing another semantic change to the original query. For example, for `ANON_AVG`, we end up computing the average of per-user averages, which could be completely different from the real average of all values. It is easy to create examples of data distributions in which this difference has a large impact; however, we observed that it does not seem to introduce significant bias in practical applications.

This step also limits the type of aggregations that can be performed within our framework: for example, we cannot easily implement `ANON_COUNT(DISTINCT col)` when `col` is *different* from `uid` as the per-user aggregation would not contain enough information to combine the contributions of multiple users (as `col` values need to be de-duplicated to preserve semantics). We come back to these considerations in Section 4.3.1.

4.2.3.4 *Stability bounding*

Next, to bound the total privacy budget of the computation, we need to ensure that each user contributes to a fixed number of partitions in the output row. To this end, we adapt the notion of query stability from [273].

Definition 81 (Global stability)**.** *Let T be a function $T : \mathcal{D} \to \mathcal{D}$. We say that T is c-*stable *if for all $D_1, D_2 \in \mathcal{D}$:*

$$\left\|T(D_1) - T(D_2)\right\| \leq c \left\|D_1 - D_2\right\|.$$

As an immediate consequence, if a user id owns k rows in D, and T is a c-stable transformation, then there are at most $k \cdot c$ rows derived from rows owned by id in $T(D)$.

Our privacy model requires the input to the cross-user aggregation to have constant stability. Simple SQL operators have a stability of one: for example, each record in an input relation can only be projected onto one record. An addition or a deletion can only affect one record in a projection, so projections have a stability of one. The same logic applies for selection.

Other operators, such as joins, have unbounded stability because source records can be multiplied. Adding or removing a source record can affect an unbounded number of output records. When SQL operators are sequentially composed, we multiply the stability of each operator to yield the entire query's overall stability. Thus, even a subquery accepted by the checks of Section 4.2.3.2 might not be c-stable for any c.

To fix this problem, we can compose an unbounded transform T with a stability-bounding transform $\mathcal{T}_\kappa$ to yield a composite κ-stable transform:

$$\left\|\mathcal{T}_\kappa(T(D_1)) - \mathcal{T}_\kappa(T(D_2))\right\| \leq \kappa \left\|D_1 - D_2\right\|.$$

To define $\mathcal{T}_\kappa$, we use partitioned-by-user reservoir sampling with a per-partition reservoir size of κ, which has a stability of κ. Reservoir sampling was chosen for its simplicity, and because it guarantees a strict bound on the contribution of one user. Non-random sampling (e.g. taking the first κ elements, or using systematic sampling) risks introducing bias in the data, depending on the relative position of records in the dataset. Simple random sampling does not guarantee contribution bounding, and all types of sampling with replacement can also introduce bias in the data.

Joins appear frequently in queries [209], so it is imperative to support them in order for an engine to be practical. Since joins have unbounded stability, a stability bounding mechanism is necessary to provide global stability privacy guarantees on queries with joins. We can thus support a well bounded, full range of join operators.

4.2.3.5 *Bounded-contribution aggregation*

Let us now introduce the set of supported ε-DP statistical aggregates with bounded contribution. These functions are applied after the per-user, per-partition aggregation step discussed in Section 4.2.3.3, and after the stability bounding operator. Importantly, at this step, we assume that each user's contributions to a single partition have been aggregated to a single input row; this property is enforced by the query rewriter.

`COUNT(DISTINCT uid)` is the simplest example of an aggregation with bounded contribution: it counts unique users, so adding or subtracting a user will change the count by no more than 1. For more complex aggregation functions, we must determine how much a user can contribute to the result and add appropriately scaled noise. A naive solution without limits on the value of each row leads to unbounded contribution by a single user. For example, a `SUM` which can take any real as input has an unbounded L^1-sensitivity (see Definition 80).

To address this, each ε-DP function accepts an additional pair of lower and upper limit parameters L and U used to clamp (i.e., *bound*) each input. For example, consider the anonymized sum function `ANON_SUM(col, L, U)`.

Let $\text{sum}_L^U : \mathcal{D} \rightarrow \mathbb{R}$ be the function that transforms each of its inputs x into $x' = \max(\min(x, U), L)$, and then sums all x'. The global sensitivity for this bounded sum function is:

$$\Delta\text{sum}_L^U = \max\left(|L|, |U|\right)$$

and thus, `ANON_SUM` can be defined by using this function and then adding noise scaled by this sensitivity. A special case of `ANON_SUM`, widely used in practice, is `ANON_COUNT`, which is equivalent to summing 1 for each record present in the dataset.

Once the sensitivity is bounded, noise is added to internal states using the well-known Laplace mechanism (see Section 2.1.6.4) before a differentially private version of the aggregate result is released.

For `ANON_AVG`, we use the Algorithm 2.4 from [252] (Section 2.5.5): we take the quotient of a noisy sum (bounded as in `ANON_SUM`, and scaled by sensitivity $|U - L|/2$) and a noisy count. `ANON_VAR` is similarly derived; we use the same algorithm to compute a bounded mean and square it, and to compute a mean of bounded squares. `ANON_STDDEV` is implemented as the square root of `ANON_VAR`.

`ANON_NTILE`, which returns a given percentile of values given as input, is currently based on a Bayesian binary search algorithm [44, 215], and can be used to define `ANON_MIN`, `ANON_MAX`, and `ANON_MEDIAN`. The upper and lower bounds are only used to restrict the search space and do not affect sensitivity. At each iteration of the internal binary search, we count the number of values below and above the target value, add noise to these two counts, and determine the next step of this binary search based on these counts. Importantly, this requires to keep all elements in memory, which does not scale to very large input data. A simple solution, which we implemented, is to randomly sample input values once we reach a memory limit. Further research is currently underway to find better options.

For the rest of this section, we assume that contribution bounds are specified as literals in each query to simplify our presentation. Setting bounds requires some prior knowledge about the input set. For instance, to average a column of ages, the lower bound could be reasonably set to 0 and the upper bound to 120. To enhance usability in the case where there is no such prior knowledge, we also introduce a mechanism for automatically inferring contribution bounds, described in more detail in Section 4.2.5.1.

Table 4.2 lists our suite of aggregate functions. For each function that can be implemented using other functions, we give the equivalent formulation. Note that the equivalent formulation is given using SQL for simplicity, but we do not always implement it via SQL rewriting; we sometimes implement them as standalone low-level aggregators and only reuse the code for other aggregators. We come back to these implementation considerations in Section 4.3.1.

Function	Equivalent formulation
`ANON_SUM(col, L, U)`	
`ANON_COUNT(col, U)`	`ANON_SUM(1, 0, U)`[1]
`ANON_COUNT(DISTINCT uid)`	`ANON_COUNT(uid, 1)`
`ANON_AVG(col, L, U)`	[2]
`ANON_VAR(col, L, U)`	`ANON_SUM(col*col, 0, MAX(L*L,U*U)) - ANON_SUM(col, L, U)^2`[3]
`ANON_STDDEV(col, L, U)`	`SQRT(ANON_VAR(col, L, U))`
`ANON_NTILE(col, ntile, L, U)`	[4]
`ANON_MIN(col, L, U)`	`ANON_NTILE(col, 0, L, U)`
`ANON_MAX(col, L, U)`	`ANON_NTILE(col, 1, L, U)`
`ANON_MEDIAN(col, L, U)`	`ANON_NTILE(col, 0.5, L, U)`

[1] Technically, `ANON_SUM(IF col IS NOT NULL THEN 1 ELSE 0, 0, U)`.

[2] This could be formulated as `ANON_SUM(col, L, U) / ANON_COUNT(col, U')` for a well-chosen `U'`. However, the choice of `U'` makes this option undesirable, so we implement `ANON_AVG` separately instead; see the extended discussion in Section 4.3.1.1.

[3] Note that the per-user aggregation step for `ANON_VAR` is *not* `VAR`; we give more details about this case in Section 4.3.1.2.

[4] We discuss this aggregation in more detail in Section 4.3.1.3.

Table 4.2: ε-DP aggregate functions

4.2.3.6 *Minimum user threshold*

In this section, we outline a technique proposed in [230] to prevent the presence of arbitrary partitions (`GROUP BY` keys) in a query from violating the privacy predicate: the τ-*thresholding* mechanism. For example, consider the naive implementation in Listing 4.6.

```
SELECT col1, ANON_SUM(col2, L, U)
FROM Table
GROUP BY col1
```

Listing 4.6: Leaking GROUP BY keys

Suppose that for the value `col1` $= c$, only one user id contributed to the sum. Then querying without data from u would reveal the absence of the output row corresponding to group `col1` $= c$. It would be revealed with certainty that user u has value c for `col1`.

To prevent this, for each result aggregation row, we calculate an ε-DP count of unique contributing users. If that count is less than some *minimum user threshold* τ, the result row is discarded. τ is chosen by our model based on ε, δ, and κ parameter values. In the example above, the output row for group `col1` $= c$ would only appear in the result if noise added to the total user count (1) would result in a value larger than τ: this only happens with fixed probability, which we can control by carefully chosing τ.

The choice of τ depends on ε, δ, and κ;

Theorem 12. *Let $\varepsilon, \delta, \kappa > 0$. Define*

$$\tau = 1 - \frac{\kappa \ln\left(2 - 2(1-\delta)^{1/\kappa}\right)}{\varepsilon}.$$

If an SQL query:

- *guarantees that each user appears in at most κ partitions;*
- *for each partition, computes and releases an ε/κ-DP noisy count of the number of contributing users using the Laplace mechanism;*
- *does not release anything for empty partitions;*
- *and only releases counts greater or equal than τ;*

then it provides user-level (ε, δ)-DP.

Proof. First, note that that if a dataset D contains a single row, then the probability that an ε-DP noisy count of the number of rows in D will yield at least τ for any $\tau \geq 1$ is

$$\rho_\tau = \frac{1}{2} e^{-(\tau-1)\varepsilon}.$$

Indeed, this is a special case of Section 5.2 in [230]: the noisy count is distributed as the Laplace distribution centered at the true count of 1 with scale parameter $1/\varepsilon$. Evaluating the CDF at τ yields ρ_τ.

Let us now prove the main statement. Let $\mathcal{M}$ be the SQL operation described in the theorem. Consider any pair of datasets D_1, D_2 such that $\|D_1 - D_2\|_u = 1$ and such that the set of non-empty groups from D_1 is the same as that for D_2; call this set G. The ε/κ-DP noisy count will get invoked for all groups in G for both datasets, and the τ-thresholding applied. For all groups not in G, no row will be released for both datasets. A change in the user may affect a maximum of κ output counts, each of which is ε/κ-DP, so by the composition property of differential privacy (see Proposition 5), we have:

$$\mathbb{P}\left[\mathcal{M}(D_1) \in S\right] \leq e^{\varepsilon} \mathbb{P}\left[\mathcal{M}(D_2) \in S\right]$$

Next, consider an empty dataset D_3. Let U be the set of all outputs and let E be the output set containing only the output "no result rows are produced". Then we have $\mathbb{P}\left[\mathcal{M}(D_3) \in U \setminus E\right] =$

0. It remains to show that for dataset D_4 containing a single user, $\mathbb{P}\left[\mathcal{M}\left(D_4\right) \in U \setminus E\right] \le \delta$. The ε-DP count is computed by counting the number of rows, and then adding Laplace noise with scale κ/ε. Dataset D_4 contains values for a maximum of κ groups; it is only possible to produce an output row for those groups. For each groups, the probability that the noisy count will be at least τ is $\rho_\tau = \frac{1}{2}e^{-\frac{(\tau-1)\varepsilon}{\kappa}}$, by the initial observation. Then:

$$\mathbb{P}\left[\mathcal{M}\left(D_4\right) \in E\right] = 1 - \left(\rho_\tau\right)^\kappa$$

so we need to satisfy:

$$\mathbb{P}\left[\mathcal{M}\left(D_4\right) \in U \setminus E\right] = \left(\rho_\tau\right)^\kappa \le \delta.$$

Solving for τ in the expression $\left(\rho_\tau\right)^\kappa \le \delta$:

$$\tau \ge 1 - \frac{\kappa \ln(2 - 2(1-\delta)^{1/\kappa})}{\varepsilon}.$$

Lastly, consider SQL engine operator f and datasets D_5, D_6 such that D_6 is D_5 with the addition of a single user. It remains to consider the case where D_5 and D_6 do not have the same set of non-empty groups. Since they differ by one user, D_6 may have a maximum of κ additional non-empty groups, each containing 1 unique user; call this set of groups G. Call the set of the remaining groups G'. Split the rows of D_5 into two datasets: the rows that correspond to groups in G and G', respectively. Call the rows of D_6 that correspond to groups G as D_6^0 and the rows that correspond to groups G' as D_6^1. Note that D_5 only contains rows corresponding to groups G'. Now, split the operator f into two operators f_0 and f_1: f_0 is f with an added filter that only outputs result rows corresponding groups in G; f_1 is the same for G'.

We have decomposed our problem into the previous two cases. The system of f_0, D_5, and D_6^0 is the case where the pair of datasets have the same set of non-empty groups, G'. The system of f_1, the empty dataset, and D_6^1 is the case where the non-empty dataset contains a single user. Each system satisfies the DP predicate separately. Since they operate on a partition of all groups in D_5 and D_6, the two systems satisfy the DP predicate when recombined into f, D_5, and D_6.

We have shown that for two datasets differing by a single user, the DP predicate is satisfied. Thus, we have shown that our SQL query provides user-level (ε, δ)-DP. □

Note that this operation must be taken into account as part of privacy budget computation. If the only operation that the user requested was `ANON_COUNT(DISTINCT uid)`, then we can simply post-process the results of this noisy aggregation to threshold the output. Otherwise, we have to add this aggregation alongside the requested ones in the query rewriter, and split the budget between all these operations.

4.2.3.7 *Query semantics*

In this section, we formally define our (ε, δ)-DP relational operator, denoted by χ, using some of the conventional operators in relational algebra. This is simply a reformulation than the process described in Section 4.2.3 so far.

Operator	Basic form	Required variant
Projection	$\Pi_a(R)$	$\Pi_{\text{uid},a}(R)$
Selection	$\sigma_\varphi(R)$	$\sigma_\varphi(R)$
Aggregation	${}_g\mathcal{G}_a(R)$	${}_{\text{uid},g}\mathcal{G}_a(R)$
Join	$R \underset{\text{cond}}{\bowtie_\theta} S$	$R \underset{\text{cond}\wedge R.\text{uid}=S.\text{uid}}{\bowtie_\theta} S$

TABLE 4.3: Allowed table subquery operators

First, let us quickly recall standard operators in relational algebra, used to describe the transformations on datasets that occur in a query.

- $\Pi_s(R)$: project columns s from R.
- $\sigma_\varphi(R)$: select from R satisfying the predicate φ.
- ${}_g\mathcal{G}_a(R)$: group on the cross products of distinct keys in the columns in g; apply the aggregations in a to each group.
- $R \underset{\text{cond}}{\bowtie_\theta} S$: take the cross product of rows in R and S, and select only the rows that satisfy the predicate cond.

Let s (select-list), g (group-list), and a (aggregate-list) denote the attribute names $s_1, \dots, s_i$, $g_1, \dots, g_j$, and $a_1, \dots, a_k$, respectively. Assume a is restricted to only contain the ε-DP aggregate function calls discussed in Section 4.2.3.5. Our introduced operator, ${}_g\chi_a$, can be interpreted as an anonymized grouping and aggregation operator with similar semantics to ${}_g\mathcal{G}_a$, and supports queries in the general form given in Listing 4.7.

```
SELECT WITH ANONYMIZATION s, a
FROM T(R)
GROUP BY g
```

LISTING 4.7: Generic form of the queries supported by our relational operator χ

First, χ verifies row ownership constraints on T: it must not contain any additional analyst-specified operators which create intermediate objects of shared ownership, as defined in Section 4.2.3.2. Formally, χ imposes that all operators present in this subquery are *allowed*, as defined by Table 4.3.

If R is an input table where each row is owned by exactly one user, and T satisfies those constraints, then this guarantees that $T(R)$ also has such a user identifier `uid` in its schema. We can then use $T(R)$ as the input to our proposed operator χ, and define our query Q:

$$Q = \Pi_s({}_g\chi_a(T(R)))$$

The query rewriter discussed in Section 4.2.3.1 transforms a query containing χ into a query that only contains SQL primitives, minimizing the number of changes to the underlying SQL engine. We define the following in order to express our rewriting operation.

- Let #uid denote the number of user IDs present in a given partition. In SQL, it would be expressed as `COUNT(DISTINCT uid)`. We use it in the condition $\#\text{uid} \geq \tau$, which captures the τ-thresholding mechanism introduced in Section 4.2.2.2 and discussed in more detail in Section 4.2.3.6.

- Let a' be the corresponding non-ε-DP partial aggregation functions of a. For example, if a is `ANON_SUM`, a' would be `SUM`.

- Let ${}_m\mathcal{T}_n$ behave like a reservoir-sampling SQL `TABLESAMPLE` operator where m is the grouping list and n is the number of samples per group. In other words, the operation ${}_m\mathcal{T}_n(R)$ partitions R by columns m. For each partition, it randomly samples up to n rows. We use $\mathcal{T}$ as this *stability-bounding operator*, discussed in Section 4.2.3.4.

When the query rewriter is invoked on the following relational expression Q containing χ:

$$Q = \Pi_s({}_g\chi_a(T(R)))$$

it returns the modified expression, with χ expanded:

$$\begin{aligned} U &= \Pi_{\text{uid},g,a'}({}_{\text{uid}}\mathcal{T}_\kappa({}_{\text{uid},g}\mathcal{G}_{a'}(T(R)))) \\ S &= \Pi_s\sigma_{\#\text{uid}\geq\tau}({}_g\mathcal{G}_{\#\text{uid},a}(U)) \end{aligned}$$

Effectively, the rewriter splits Q into the two-stage aggregation U and S.

- U groups the output of $T(R)$ by the key vector (uid, g), applying the partial aggregation functions a' to each group. κ rows are then sampled for each user. The first aggregation enforces that there is only one row per user per partition during the next aggregation step.

- S performs a second, differentially private, aggregation over the output of U. This aggregation groups only by the keys in g and applies the ε-DP aggregation functions $(f(\text{uid}), a)$. These functions assume that every user contributes at most one input row. A filter operator is applied last to suppress any rows with too few contributing users.

Together, this implements the process described in the previous parts of Section 4.2.3.

Let us illustrate this process with a concrete example. Consider the following query. It assumes we have an Employee table and an Order table, which both have a uid field, and it is counting the number of orders made by employees in each department.

```sql
SELECT WITH ANONYMIZATION
  dept, ANON_COUNT(*, 5) AS c
FROM Employee E JOIN Order O USING(uid)
GROUP BY dept
```

Listing 4.8: Anonymization query containing a simple join.

Recall that `ANON_COUNT(*, U)` is equivalent to `ANON_SUM(col, 0, U)` in the cross-user aggregation step. We can express this query in relational algebra using our DP operator χ:

$$Q = \Pi_{\text{dept},c}\left({}_{\text{dept}}\chi_{\text{ANON\_COUNT(*, 5) AS } c}\left(E \underset{E.\text{uid}=O.\text{uid}}{\bowtie_\theta} O\right)\right)$$

Let us expand χ into the two-stage aggregation, U and S, and let us call T the initial table subquery. Our query can then be written as:

$$\begin{aligned}
T &= \Pi_{\text{dept,uid},c_2}\left({}_{\text{dept,uid}}\mathcal{G}_{\text{COUNT(*) AS } c_2}\left(E \underset{E.\text{uid}=O.\text{uid}}{\bowtie_\theta} O\right)\right)\\
U &= \Pi_{\text{dept},uid,c_2}\left({}_{\text{uid}}\mathcal{T}_\kappa(U)\right)\\
S &= \Pi_{\text{dept},c}\sigma_{c_3\geq\tau}\left({}_{\text{dept}}\mathcal{G}_{\text{ANON\_SUM}(c',0,5)\text{ AS } c,\text{ANON\_COUNT(*) AS } c_3}(T)\right)
\end{aligned}$$

Note that we add an additional user counting ε-DP function, aliased as c_3, which is compared to our threshold parameter, τ, to ensure that the grouping does not violate the (ε, δ)-DP predicate, to be discussed in Section 4.2.3.6. In this case c_3 is the number of unique users in each department: as each user contributes at most 1 to each partition, we can simply use `ANON_COUNT(*)` to compute this count of unique users.

In Figure 4.9, we illustrate the workflow with example tables `Employee` and `Order`, $\tau = 2$, and $\kappa = 1$.

4.2.3.8 *User-level differential privacy*

In this section, we show that our engine satisfies user-level (ε, δ)-DP, as defined in Section 4.2.1.3.

Suppose that a query $\mathcal{M}$ has aggregate function list $a = a_1, \ldots, a_N$, and grouping list $g = g_1, \ldots, g_J$. Let the privacy parameter for aggregation function a_i be ε_i.

Consider any set of rows owned by a single user in the input relation of our anonymization operator, ${}_g\chi_a$. We first partition and aggregate these rows by the key vector (uid, g), before sampling up to κ rows for each partition by uid.

The result for a group j is reported if the τ-thresholding predicate defined in Section 4.2.3.6 is satisfied. Computing and reporting the comparison count for this predicate is ε'_j-DP. For $D_1, D_2 \in \mathcal{D}$, such that they differ by a user's data for a *single group* j, consider each aggregation function a_i as applied to group j. By composition theorems and Theorem 12, we provide $(\varepsilon'_j + \sum \varepsilon_i, \delta_j)$ user-level (ε, δ)-DP for that row.

However, there are many groups in a given query. Due to our stability bounding mechanism, a single user can contribute to up to κ groups. The κ output rows corresponding to these groups can be thought of as the result of κ sequentially composed queries. Let $D_1, D_2 \in \mathcal{D}$ such that $\|D_1 - D_2\|_u = 1$. Let ε be the sum of the κ greatest elements in the set $\left\{\varepsilon'_j + \sum \varepsilon_i\right\}_{j=1,\ldots,J}$. Again, by composition, we conclude that for any output set S, and some $\delta > 0$, we have

$$\mathbb{P}\left[\mathcal{M}(D_1) \in S\right] \leq e^{\varepsilon}\mathbb{P}\left[\mathcal{M}(D_2) \in S\right] + \delta$$

This shows that given engine-defined parameters ε, δ, and κ, it is possible to set privacy parameters for individual ε-DP functions to satisfy the query (ε, δ)-DP predicate. For example,

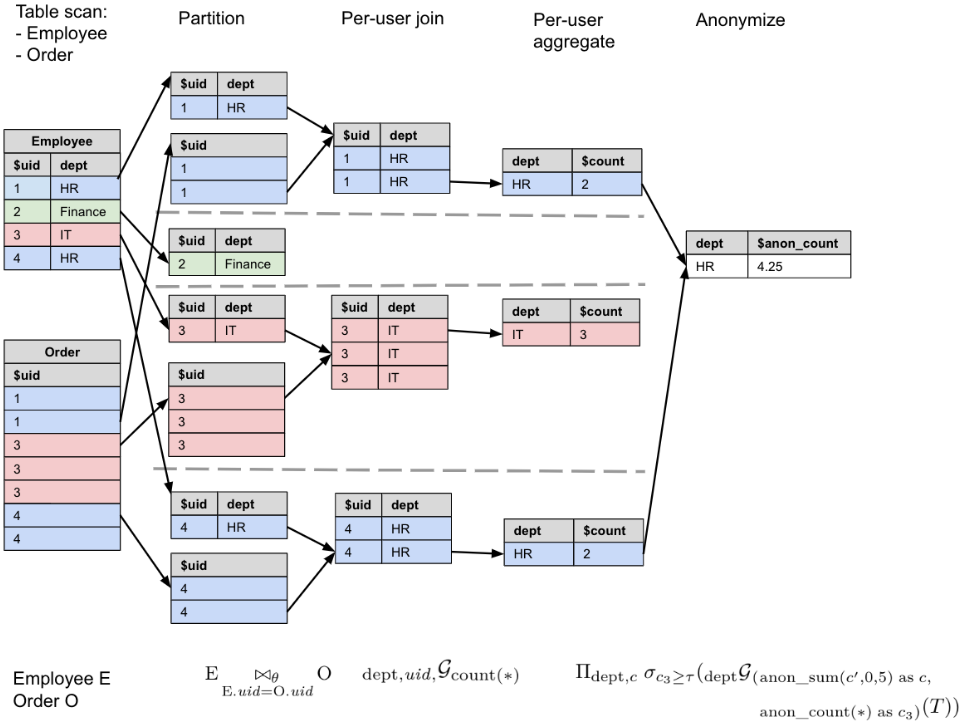


Figure 4.9: Example workflow of anonymized query. Note that in the last step, the IT department gets dropped from τ-thresholding.

we can set ε'_j and ε_i to $\frac{\varepsilon}{\kappa(N+1)}$ for all i and j. δ is enforced by the derived parameter τ as discussed in Section 4.2.3.6. Both user-privacy parameters ε and δ can therefore be bounded to be arbitrarily small by analysts and data owners.

Note that the method we use to guarantee user-level differential privacy can be interpreted as similar to row-level group privacy: after the per-user aggregation step, we use the composition theorem to provide group privacy for a group of size κ. Alone, row-level group privacy does not provide user-level privacy, but in combination to user-level contribution bounding, this property is sufficient to obtain the desired property.

4.2.4 *Accuracy*

In this section, we explore the accuracy of our system by running numerical experiments and provide analytical reasoning about the relationship between accuracy and various parameters.

4.2.4.1 *Experimental accuracy*

We assess accuracy experimentally using TPC-H [87], an industry standard SQL benchmark. The TPC-H benchmarks contains a dataset schema and queries that are similar to those used by analysts of personal data at real-world organizations. In addition, the queries contain interesting features such as joins and a variety of aggregations. We generate a TPC-H dataset with the default scale factor of 1. We treat suppliers or customers as "users", as appropriate. Our metric for accuracy is median relative error, the same one used in [209]; a smaller median relative error corresponds to higher utility.

We compute the median relative error of 1,000,000 runs for our model over TPC-H Query 1 using three different aggregation functions and $\varepsilon = 0.1$ in Table 4.4. We compare our results to 1,000,000 runs of Flex over the same query, and 10,000 runs (due to performance considerations) of PINQ over the same query. To present a fair comparison, we disabled τ-thresholding and compared only one result record to remove the need for κ stability bounding. In addition, each run of the experiment was performed using a function of fixed sensitivity, controlled by supplying the function with a lower bound of 0 and an upper bound of the value in the Δ_uQ1 column. The bounds were fixed to minimize accuracy loss from contribution clamping.

For our experiments with PINQ and Flex, we also set sensitivity to our previously determined Δ_uQ1 listed in Table 4.4. The results are close to our model's results, but because neither PINQ nor Flex can *enforce* contribution bounds for datasets with multiple contributions per user, incorrectly set sensitivity can result in query results that are *not* differentially private. Such incorrectly set bounds can be seen in experiments in Johnson et al. [209] and McSherry's analysis [276], and in the last row of Table 4.4, where PINQ and Flex report errors far below what are required to satisfy the ε-DP predicate.

With correctly set sensitivity bounds our model's results are comparable to PINQ's results for count and average. Implementation differences in our median function mean that our error is lower by a factor of 2. Both PINQ and our model outperform Flex's result for count by around an order of magnitude. We do not report errors for average and median for Flex because Flex does not support those functions.

Function	Δ_uQ1	Our model	PINQ	Flex
ε-COUNT(*)	373	0.00175	0.00179	0.0197
ε-AVG(l_extendedprice)	100000	0.00181	0.00181	[1]
ε-MEDIAN(l_extendedprice)	100000	0.00189	0.00349	[1]
ε-COUNT(*)	1[2]	0.993	4.727×10^{-6}	2.70×10^{-5}

[1] Unsupported functions

[2] Intentionally incorrect sensitivity to demonstrate contribution bounding

Table 4.4: TPC-H Query 1 errors comparison with others ($\varepsilon = 0.1$)

Query	Δ_uQ	κ	Experimental error	τ-thresholding rate	δ
Q4	5	5	0.0339	5.40×10^{-5}	6.78×10^{-7}
Q13	1	1	0.00677	0.309	6.78×10^{-7}
Q16	1	5	11.3[1]	0.9999606	2.07×10^{-4}
Q21	1	1	1.60[1]	0.999796	2.07×10^{-4}

[1] Results uninterpretable due to high levels of τ-thresholding

Table 4.5: Selected TPC-H join query results ($\varepsilon = 0.1$)

We also ran our system over a selection of TPC-H queries containing joins, the experimental results are presented in Table 4.5. Similarly, we report the median relative error of 1,000,000 runs for each query using $\varepsilon = 0.1$. We report the impact of τ-thresholding (the ratio of suppressed records), suggesting that our model is (ε, δ)-DP. δ was set with $\delta = n^{-\varepsilon \ln n}$, where n is the number of distinct users in the underlying dataset: either customers or suppliers, depending on the query.

Q4 represents how our system behaves when very little τ-thresholding occurs. Q16 and Q21 demonstrate the opposite, both queries exhibit a very large error that differs from the theoretical error due to most partitions being removed by the threshold because of their small user count. Indeed, this is by design: as partition user counts approach 1, the ratio of τ-thresholding approaches $(1 - \delta)^{1/\kappa}$. Finally, Q13 represents a more typical result, a query containing mostly medium user count partitions with a long tail of lower count partitions. A moderate amount of τ-thresholding occurs which increases error compared to Q4, but the results are still quite accurate.

4.2.4.2 *Impact of parameters on utility*

In this section we explore the relationship between utility and various parameters, which must be adjusted to balance privacy and utility [13].

Impact of ε

First, let us discuss the effect of ε on the accuracy of the query results. Note that ε is inversely proportional to the Laplace scale parameter used by differentially private aggregators to add noise. Hence, an increase in ε leads to a decrease in utility.

Proposition 30. *The median error from Laplace noise of scale Δ_u/ε is $\frac{\ln(2)\Delta_u}{\varepsilon}$.*

Proof. Let x be the median noise. By symmetry, we have:

$$\begin{aligned}\frac{1}{4} &= \mathrm{CDF}_{\mathrm{Laplace}}(x) - \mathrm{CDF}_{\mathrm{Laplace}}(0) \\ &= \left(1 - \frac{1}{2}e^{\frac{-x\varepsilon}{\Delta_u}}\right) - \frac{1}{2}\end{aligned}$$

and thus:

$$x_{\mathrm{count}} = \frac{\ln(2)\Delta_u}{\varepsilon}.$$

□

Similarly, the effect of the number of aggregations is straightforward. When a single query contains N aggregations, the privacy budget is split equally among them: each aggregation will satisfy (ε/N)-differential privacy, so the median error degrades inversely with the number of aggregations.

Changing the value of ε also impacts partition selection: recall that the threshold τ was defined in Theorem 12 as

$$\tau = 1 - \frac{\kappa \ln\left(2 - 2(1-\delta)^{1/\kappa}\right)}{\varepsilon}.$$

In particular, if all other parameters are fixed, $\tau = \Theta\left(\frac{1}{\varepsilon}\right)$: multiplying ε by two means that we will be able to keep partitions with approximately half as many users.

Impact of δ

What is the effect of varying δ for a fixed ε? This causes the threshold τ to change, which changes the number of records dropped due to thresholding. Similarly, changing κ modifies the number of records dropped due to contribution bounding. We first perform experiments on TPC-H Query 13 with $\varepsilon = .1$ and varying δ to quantify the impact on partitions returned; Figure 4.10 displays the results. The figure shows that as δ increases exponentially, the proportion of partitions thresholded decreases somewhat linearly.

Impact of κ

Next, we analyze the effect of κ for a specific artificial query. Consider the following query *after* rewriting.

```
SELECT ANON_COUNT(*)
FROM (SELECT uid, ROW_NUMBER() as rn
      FROM Table
```

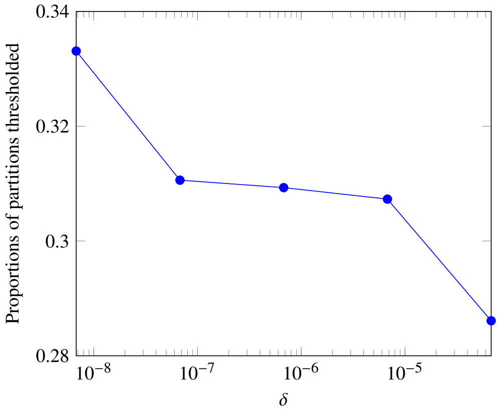


FIGURE 4.10: Partition thresholding rates on Q13 induced by various δ.

```
      GROUP BY uid, rn)
TABLESAMPLE RESERVOIR
    (κ ROWS PARTITION BY uid)
```

LISTING 4.9: Anonymization query after rewriting, showing the effect of κ.

Suppose that there are n users in the database. Suppose each user appears in a number of partitions distributed according to a fixed probability distribution θ, and call X_i the (random) number of partitions in which user i appears. Then the proportion p_κ of partitions dropped due to reservoir sampling is:

$$p_\kappa = \frac{\sum_{i=1}^{n} \max\left(X_i - \kappa, 0\right)}{\sum_{i=1}^{n} X_i}.$$

Figure 4.11 shows the effect of κ on p_κ with $n = 1000$ and various distributions θ. All distributions have similar behavior as κ increases, but the median percent error declines at different speeds based on distribution shape.

Finally, we analyze the effect of clamping on accuracy using model input distributions. Since clamping occurs at the ε-DP aggregation level, we focus on input sets that have at most one row per user.

Consider finding `ANON_AVG(col, L, U)`, where there are n values of `col` uniformly distributed on $[a, b]$. If the input distribution is symmetric, symmetric clamping will not create bias, so we clamp only one end: consider clamping bounds L and U such that $L = a$ and $a < U < b$. We analyze expected error since median error is noisier when running experiments, and the behavior of both metrics are similar.

We plot the impact of the upper clamp bound on total expected error for uniform `col` with $(a, b) = (50, 150)$ in Figure 4.12, using $L = -200$, $n = 100$, and various values of ε. The optimal point on each curve is marked with a circle. To maximize accuracy, overestimating

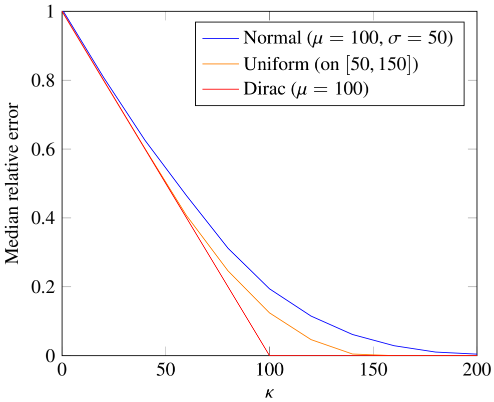


FIGURE 4.11: Median relative error induced by κ for various distributions centered at 100.

the spread of the input set must be balanced with restricting the sensitivity. Analysis with the other aggregation functions yields similar results.

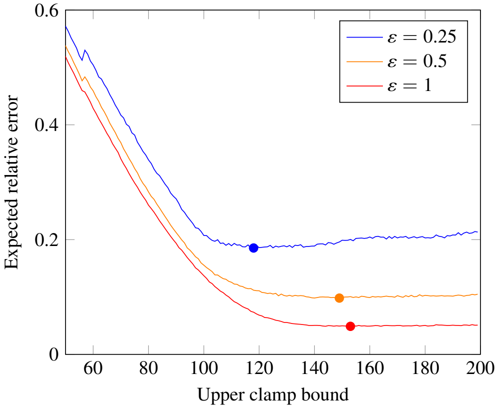


FIGURE 4.12: Expected relative error depending on the upper clamp bound for a uniform distribution on $[50, 150]$, using a lower clamp of -200, $n = 100$, and various values of ε. We highlight the optimal point on the curve, with the least expected error.

Note that overestimating the spread of the input set creates less error than underestimating it; the curves in Figure 4.12 do not grow very fast after the optimal point. This can be explained analytically: the unclamped expected mean of `col` is $\frac{a+b}{2}$; for `col` clamped between L and U, it is:

$$\frac{(2aU - a^2 - U^2)}{2(b-a)}$$

so the expected error of the clamped mean is

$$\frac{(b-U)^2}{2(b-a)}.$$

Compare the expected error to the median noise added by `ANON_AVG(col, L, U)`, which is approximately $\frac{\ln(2)(U-L)}{n\varepsilon}$. The clamping error grows quadratically with $b-U$, while the noise error only grows linearly with $U-L$. The behavior for other distributions is similar.

4.2.5 *Usability*

Designing a differentially private query engine for non-expert use requires a number of considerations beside the design and implementation of the system described in the previous section. In this section, we focus on *usability*: what makes such a system difficult to use by analysts, and how can we mitigate these concerns.

4.2.5.1 *Automatic bounds determination*

One major difference between standard SQL queries and queries using our differentially private aggregation operator is the presence of bounds: e.g., when using `ANON_SUM`, an analyst must specify the lower and upper bound for each sum input. This differs from standard workflows, and more importantly, it requires prior knowledge of the data that an analyst might not have.

To remove this hurdle, we designed an aggregation function which can be sequentially composed with our previously introduced ε-DP functions to automatically compute bounds that minimize accuracy loss. Call this function `APPROX_BOUNDS(col)`.

Without contribution bounds, the domain of an ε-DP function in our model spans $\mathbb{R}$. Finding differentially private extrema over such a range is difficult. Fortunately, we can leverage two observations. First, inputs to this function are represented on a machine using finite precision; typically 64-bit integers or floating point numbers. Second, bounds do not need to be close to the real extrema: clipping a small fraction of data will usually not have a large influence on the aggregated results, and might even have a positive influence by removing outliers.

Consider an `ANON_SUM` operation operating on 64-bit unsigned integers where bounds are not provided. We divide the privacy budget in two: the first half to be used to infer approximate bounds using `APPROX_BOUNDS(col)`; the second half to be used to calculate the noisy sum as usual. We must spend privacy budget to choose bounds in a data-dependent way.

In the `APPROX_BOUNDS(col)` function, we instantiate a 64-bin logarithmic histogram of base 2, and, for each input i, increment the $\lceil \log_2 i \rceil$th bin. If we have a total budget of ε for this operation, we add Laplace noise of scale $1/\varepsilon$ to the count in each bin, as is standard for differentially private histograms. Then, to find the approximate maximum of the input values, we select the most significant bin whose count exceeds some threshold K. This threshold depends on two parameters: the number of histogram bins B, and the probability of a false positive p of an empty bin exceeding the threshold.

The probability that a given bin produces a count above x if its true count is zero is $\frac{1}{2}e^{-x\varepsilon}$. Suppose we are looking for the most significant bin with a count greater than K. In the `APPROX_BOUNDS(col)` function, we iterate through the histogram bins, from most to least significant, until we find one exceeding t. In the worst case, the desired bin is the *least* significant bin. This means $B - 1$ bins with exact counts of 0 must not have noisy counts exceeding K. Thus, the probability that there was not a false positive in this worst case is:

$$p = (1 - \frac{e^{-K\varepsilon}}{2})^{B-1}.$$

Solving for K, we obtain:

$$K = \frac{1}{\varepsilon} \ln\left(2 - 2p^{\frac{1}{B-1}}\right).$$

For example, $B = 64$ for unsigned integers. When setting p, the trade-offs of clipping distribution tails, false positive risk, and algorithm failure due to no bin count exceeding K must all be considered. Values on the order of $(1 - 10^{-9})$ for p can be suitable, depending on ε and the size of the input dataset.

The approximate minimum bound can similarly be found by searching for the least significant bin with count exceeding t. We generalize this for signed numbers by adding additional negative-signed bins, and for floating point numbers by adding bins for negative powers of 2.

Note that there are two possible ways to use this algorithm to automatically generate reasonable bounds: we can either use once over the entire dataset, and use the same bounds in every partition, or use it in each partition separately. There are pros and cons to both approaches. Using it over the entire dataset means that more data is being used, so a smaller fraction of the ε can be used for this operation. Further, for aggregations like sums and counts, it means that the noise magnitude used in each partition will be the same, which makes the output data easier to reason about. On the other hand, using it over each partition separately is better suited for use cases where the magnitude of values varies significantly between partitions.

4.2.5.2 *Providing a measure of accuracy*

A ubiquitous challenge for a DP interface is the fact that acceptable accuracy loss depends on the data characteristics, as well as on the problem at hand. We address this by giving analysts a variety of utility loss measures. These utility loss measures can be used in two ways, depending on the scenario in which the differentially private query engine is used.

- If the engine is used to generate differentially private data according to a given strategy, by someone who has access to the raw data, the analyst can try various values of parameters to determine which give the best utility for their use case.

- If the engine is used to allow an untrusted analyst to send queries to a dataset to which they have no access, we can still return a subset of utility metrics without breaking differential privacy guarantees.

There is a profound difference between these two scenarios. Some measures of data loss are useful in both. Since all parameters (ε, δ, κ, etc.) are public, the method and noise parameters used in the anonymization process are not sensitive, and can be returned to the engine client. We could give as much detail as we could; however, in practice, we have found that confidence intervals (CI) are the best way to communicate uncertainty about the results. The CI can be calculated from each function's contribution bounds and share of ε. Importantly, it does not account for the effect of clamping or thresholding. Note that for some aggregations, it might be difficult to compute a CI. For example, averages need to combine the confidence intervals from two sources of noise (the sum and the count), and we do not support CI for quantiles.

Some measures are more useful in the former scenario, if the analyst has access to the raw data. A particularly useful metric is *partition loss*: how many partitions with low user counts are removed by τ-thresholding. We can simply return the percentage of lost partitions due to τ-thresholding, possibly as a percentage of all partitions. It would be useful to return this metric in the untrusted use case, but making it differentially private is not straightforward: a single user could add an unbounded amount of partitions, so the sensitivity of this metric is unbounded; and clamping it would defeat the point. This metric is important because it is typically actionable: coarsening the partitions (e.g. aggregating the data daily instead of hourly) can drastically limit partition loss.

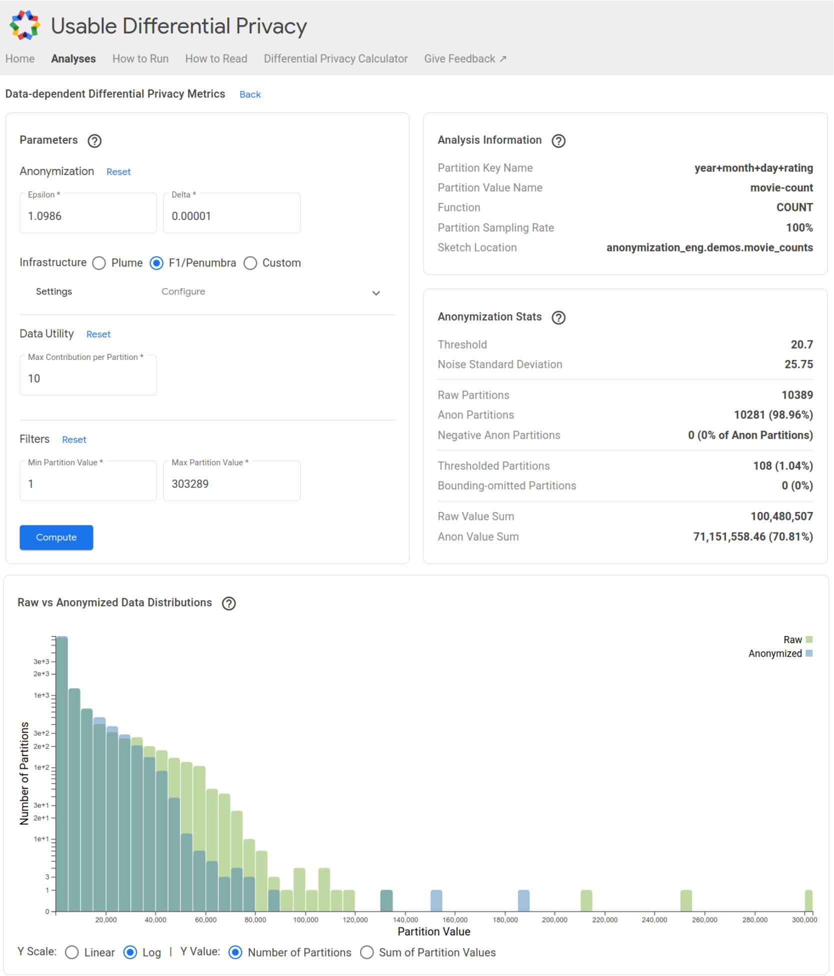
Finally, it is also useful to surface the data loss due to clamping: how many values are clamped, and what is their magnitude. In the "trusted analyst" use case, we can visualize both by simply comparing the private histogram with the non-private histogram, an example is given in Figure 4.13. In the untrusted use case, if we use automatic bounds determination (Section 4.2.5.1), the log-scale histogram gives an approximate fraction of inputs exceeding the chosen bounds; since this is private, it could also be returned to the analyst.

To represent privacy, a techniques and perspectives have been proposed in the literature [200, 234, 243, 290, 304]. In practice, we rely on the prior-posterior explanations and visualizations like those presented in Section 2.1.6.2 to provide an intuitive idea of the guarantees offered by different privacy parameters. This choice is often deferred to privacy experts, we discuss possible approaches to tackling this challenge in Section 4.3.4.1.

4.2.6 *Robustness and testing*

Writing bug-free software is notoriously hard. This is particularly problematic when bugs can potentially break the privacy guarantee that a tool is supposed to provide. A well-known example is the attack on the naive implementation of the Laplace mechanism on floating-point numbers [283]: the noisy version of such numbers can leak information on their raw value via their least significant bits. Fixing known vulnerabilities is one thing; but how can we increase our degree of confidence that the software used for important production use-cases correctly enforces its advertised privacy guarantees?

General software development good practices are obviously part of the solution: keeping the design simple and modular is one example, requiring code reviews before any code change is another. In this section, we focus on three types of methods to detect bugs and increase our overall level of confidence in our tooling: manual testing, unit testing, and stochastic testing.



Usable Differential Privacy

Home | **Analyses** | How to Run | How to Read | Differential Privacy Calculator | Give Feedback ↗

Data-dependent Differential Privacy Metrics Back

Parameters ?

Anonymization Reset

Epsilon *: 1.0986

Delta *: 0.00001

Infrastructure ○ Plume ● F1/Penumbra ○ Custom

Settings Configure

Data Utility Reset

Max Contribution per Partition *: 10

Filters Reset

Min Partition Value *: 1

Max Partition Value *: 303289

Compute

Analysis Information ?

Partition Key Name	**year+month+day+rating**
Partition Value Name	**movie-count**
Function	**COUNT**
Partition Sampling Rate	**100%**
Sketch Location	**anonymization_eng.demos.movie_counts**

Anonymization Stats ?

Threshold	**20.7**
Noise Standard Deviation	**25.75**
Raw Partitions	**10389**
Anon Partitions	**10281 (98.96%)**
Negative Anon Partitions	**0 (0% of Anon Partitions)**
Thresholded Partitions	**108 (1.04%)**
Bounding-omitted Partitions	**0 (0%)**
Raw Value Sum	**100,480,507**
Anon Value Sum	**71,151,558.46 (70.81%)**

Raw vs Anonymized Data Distributions ?

Y Scale: ○ Linear ● Log | Y Value: ● Number of Partitions ○ Sum of Partition Values

FIGURE 4.13: Screenshots of an internal tool showing the kind of metrics that can be surfaced to a trusted data custodian to assist them in tweaking various parameters. A more detailed description of this tool can be found in [314].

4.2.6.1 *Manual testing*

We audited the code manually and found some implementation issues. Some of these issues have previously been explored in the literature, notably regarding the consequences of using a floating-point representation with Laplace noise. Some of them, however, do not appear to have been previously mentioned in the literature, and are good examples of what can go wrong when writing secure anonymization implementations.

One of these comes from another detail of floating-point representation: special `NaN` ("not a number") values. These special values represent undefined numbers, like the result of $0/0$. Importantly, arithmetic operations including a `NaN` are always `NaN`, and comparisons between any `NaN` and other numbers are always `False`. This can be exploited by an attacker, for example using a query like `ANON_SUM(IF uid=4217 THEN 0/0 ELSE 0)`. The `NaN` value will survive naive contribution bounding (bounds checks like `if(value > upper_bound)` will return `False`), and the overall sum will return `NaN` iff the condition was verified. We suspect that similar issues might arise with the use of special `infinity` values, although we have not found them in our system (such values are correctly clamped).

Another possible failure mode is overflows and underflows. If sufficiently many large 64-bit integer values are added, the resulting number can be close to the maximum possible 64-bit integer. A single element added can make the result overflow, and the result might be a very small negative integer. If noise addition is implemented in a way that implicitly converts the input into a floating-point number, the resulting distributions of noisy outputs are very distinguishable: one will be almost surely positive, the other negative.

From these two examples, we found that a larger class of issues can appear whenever the user can abuse a branching condition to fail if an arbitrary condition is satisfied (by example, by throwing a runtime error or crashing the engine). Thus, in a completely untrusted environment, the engine should catch all errors and silently ignore them, to avoid leaking information in the same way; and it should be hardened against crashes. We do not think that we can completely mitigate this problem, and silently catching all errors severely impedes usability. Thus, is it a good idea to add additional risk mitigation techniques, like query logging and auditing.

Interestingly, fixing the floating-point issue in [283] can lead to a *different* issue when using the Laplace mechanism in τ-thresholding. The secure version of Laplace mechanism requires rounding the result to the nearest r, where r is the smallest power of 2 larger than $1/\varepsilon$. If the τ-thresholding check is implemented as `if (noisy_count >=`τ`)`, then a noisy count of e.g. 38.1 can be rounded up to e.g. 40 (with $r = 4$). If the threshold τ is 39, and the δ calculation is based on a theoretical Laplace distribution, then a noisy count of 38.1 should not pass the threshold, but will: this leads to underestimating the true δ. This can be fixed by using a non-rounded version of the Laplace mechanism for thresholding only; as the numerical output is never displayed to the user, attacks described in [283] do not apply.

4.2.6.2 *Unit testing*

A standard practice in software engineering is *unit testing*: testing a program by verifying that given a certain input, the program outputs the expected answer. Of course, for differential privacy software, the situation is more complicated: since DP requires adding random noise to data, we cannot simply test for output equality. Of course, this does not mean that we should give up on unit testing altogether. Two natural, complementary approaches are possible. We believe that using them both is necessary to write solid, production-ready code.

The first approach is to write traditional unit tests for the parts of the system that do *not* depend on random noise. For example, the cross-partition contribution bounding presented in Section 4.2.3.4 can relatively easily be put under test: create a fake input dataset and query where each user contributes to many rows, and verify that after the stability-bounding operator with parameter κ, each user contributes at most to κ rows. Similarly, the parts of the system that do *more* than simply adding noise can be tested by removing the noise entirely; either by *mocking* the noise generator, or by providing privacy parameters large enough for the noise to be negligible: this is partly how we test operations like `ANON_SUM`, which clamp their output between specified bounds.

The second approach is to write tests that take the noise into account, by allowing the output to be between certain bounds: for example, we can test that adding noise to 42 leads to a number between 22 and 62. The *tolerance* of these tests needs to be carefully calibrated depending on the parameters used in the output, and the desired *flakiness* of the unit tests: what percentage of false positives (the test fails only even though the code is correct, purely because of improbable noise values) is acceptable. Even with a very low flakiness level (we routinely use values lower than 10^{-20}), such tests can detect common programming mistakes, especially for complex operations. Examples of such tolerance calculations can be found in the documentation of our open-source library [365].

This approach based on tolerance levels is useful, but since we typically want to limit the flakiness probability to very small values, it only provides a partial guarantee that. Furthermore, if these simple approaches can check a number of properties of the software under test, they fail at detecting differential privacy violations, which is the most crucial property that we want to guarantee.

4.2.6.3 *Stochastic testing*

While the operations used in the engine are theoretically proven to be differentially private, it is crucial to verify that these operations are implemented correctly. Unit tests can find logical bugs, but it is also important to test the differential property itself. Of course, since the number of possible inputs is unbounded, and differential privacy is a property of probability distributions, it is impossible to exhaustively and perfectly check that the engine provides differential privacy. Thus, we fall back to *stochastic testing* and try to explore the space of datasets as efficiently as possible. This does not give us a guarantee that an algorithm passing the test is differentially private, but it is a good mechanism to detect violations.

Note that we focus on testing DP *primitives* (aggregation functions) in isolation, which allows us to restrict the scope of the tests to row-level DP. We then use classical unit testing to independently test contribution bounding. The approach could, in principle, be extended to end-to-end user-level DP testing, although we expect that doing so naively would be difficult for scalability reasons. We leave this extension as future work.

Our testing system contains four components: dataset generation, search procedure to find dataset pairs, output generation, and predicate verification.

Input generation

First, what datasets should we be generating? All DP aggregation functions are scale-invariant, so without loss of generality, we can consider only datasets with values in a unit range $[-r, r]$. Of course, we cannot enumerate all possible datasets $[-r, r]^S$, where S is the size of the dataset. Instead, we try to generate a *diverse* set of datasets. We use the Halton sequence [186] to do so. As an example, Figure 4.14 plots datasets of size 2 generated by a Halton sequence. Unlike uniform random sampling, Halton sequences ensure that datasets are evenly distributed and not clustered together.

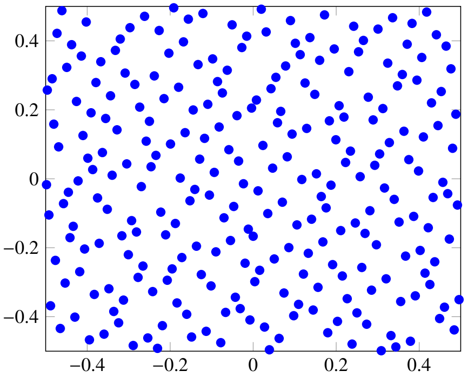


FIGURE 4.14: 256 points in $[-0.5, 0.5]^2$ generated with Halton sequences of base 2 and 3.

A dataset is a set of records: we consider its power set, and find dataset pairs by recursively removing records. This procedure is shown in Figure 4.15.

DP predicate test

Once we have pairs of adjacent datasets, we describe how we test each pair (D_1, D_2). The goal is to check that for all possible outputs S of the mechanism $\mathcal{M}$:

$$\mathbb{P}\left[\mathcal{M}(D_1) \in S\right] \leq e^{\varepsilon}\mathbb{P}\left[\mathcal{M}(D_2) \in S\right] + \delta$$

By repeatedly evaluating $\mathcal{M}$ on each dataset, we estimate the density of these probability distributions. We then use a simple method to compare these distributions: histograms.

We illustrate this procedure in Figure 4.16a and Figure 4.16b. The upper curves (in orange) are the upper DP bound, created by multiplying the probability estimate of each bin for dataset D_1 by e^{ε} and adding δ. The lower curve (in blue) is the unmodified probability estimate of D_2. In Figure 4.16a, all blue buckets are less than the upper DP bound: that the DP predicate is not violated. In Figure 4.16b, some buckets exceed this upper bound: the DP predicate is been violated. For symmetry, we also repeat this check with D_1 swapped with D_2.

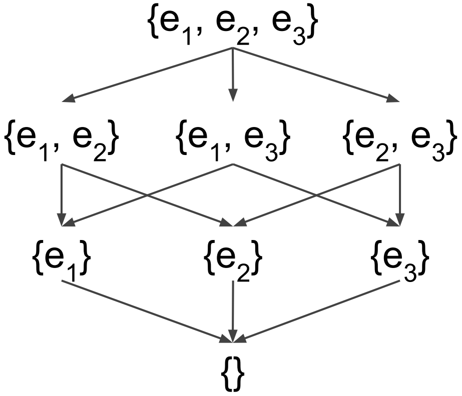


FIGURE 4.15: Dataset search graph for a dataset $\{e_1, e_2, e_3\}$

It is sufficient to terminate once we find a single pair of datasets which violate the predicate. However, since the histogram is subject to sampling error, a correctly implemented algorithm can fail this test with non-zero probability. To address this, we relax our test by using confidence intervals as bounds [384]. We can also parameterize the tester with a parameter α that tolerates a percentage of failing buckets per histogram comparison.

Full algorithm

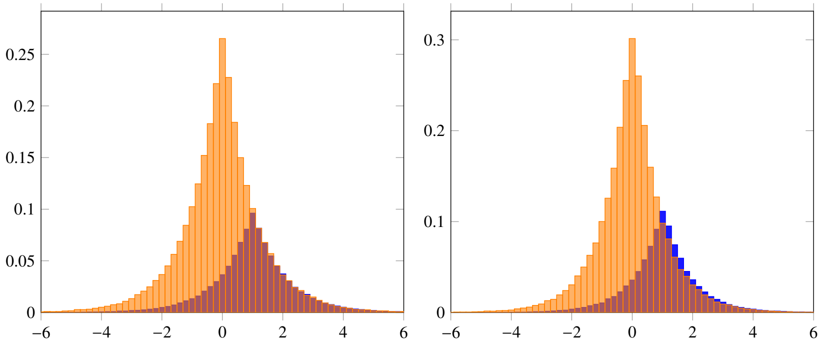
The overall approach is an algorithm that iterates over datasets and performs a DFS on each of the dataset search graphs, where each edge is a DP predicate test. We present it in algorithm 1. For simplicity, we abstract away the generation of datasets by including it as an input parameter here, which can be assumed to be generated by the Halton sequence, as we described previously. We also do not include the confidence intervals or α parameter described earlier for dealing with the approximation errors. It is also possible to adaptively choose a histogram bin width, but we put an input parameter K here. The general idea is a depth-first search procedure that iterates over edges of the dataset search graph.

Our actual implementation includes all of the above omissions, including an efficient implementation of the search procedure that caches samples of datasets already generated.

With the above procedure, we were able to detect that an algorithm was implemented incorrectly, violating DP. When we first implemented ANON_AVG, we used the Algorithm 2.3 from [252]: we used our ANON_SUM implementation to compute the noisy sum and then divided it by the un-noised count. Our first version of ANON_SUM used a Laplace distribution with scale $\frac{|U-L|}{\varepsilon}$, where U and L are the upper and lower clamping bounds, which is the correct bound when used as a component of ANON_AVG. However, this was *not* correct for noisy sum in the case when adjacent datasets differ by the presence of a row. We updated the scale to $\frac{\max(|U|,|L|)}{\varepsilon}$, as maximum change in this case is the largest magnitude. This change

```
Input: A random mechanism $\mathcal{M}$, privacy parameters $(\varepsilon, \delta)$, datasets $\mathbf{D}$, number of
        samples $N$, number of histogram buckets $K$
Output: Decision on whether we found a differential privacy violation for $\mathcal{M}$
foreach $D_r \in \mathbf{D}$ do
    $S \leftarrow$ {root node $D_r$} // Initialize search stack
    while $S \neq \emptyset$ do
        $A \leftarrow pop(S)$
        $S \leftarrow S \cup \{succ(A)\}$
        foreach $B \in succ(A)$ do
            // Generate samples
            $X_A \leftarrow \{x_A^{(i)} \sim \mathcal{M}(A) \mid i = 1, \ldots, N\}$
            $X_B \leftarrow \{x_B^{(i)} \sim \mathcal{M}(B) \mid i = 1, \ldots, N\}$
            // Determine histogram buckets
            $H_{\min}, H_{\max} \leftarrow \min(X_A \cup X_B), \max(X_A \cup X_B)$
            $h \leftarrow \frac{H_{\max} - H_{\min}}{K}$
            $\mathbf{B} \leftarrow \{B_k = [H_{\min} + (k-1) \cdot h, H_{\min} + k \cdot h] \mid k = 1, \ldots, K\}$
            foreach $B_k \in \mathbf{B}$ do
                // Check DP condition using approx. densities over $B_k$
                if $\left|\left\{i \mid x_A^{(i)} \in B_k\right\}\right| > e^{\varepsilon} \cdot \left|\left\{i \mid x_B^{(i)} \in B_k\right\}\right| + \delta$ then
                    return $\mathcal{M}$ is likely not differentially private
                end
            end
        end
    end
end
return No likely violation has been found
```

Algorithm 1: DP Stochastic Test



(a) Passing test (b) Failing test

Figure 4.16: Histogram examples for DP testing, given one pair of datasets

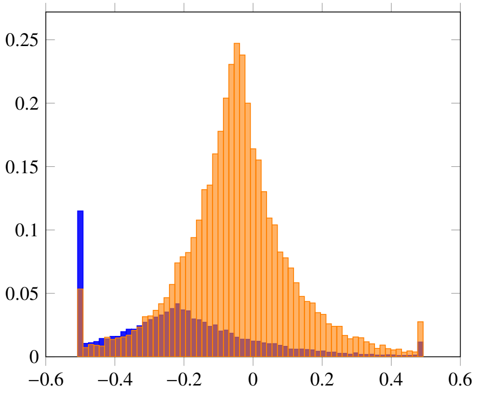


Figure 4.17: Example histogram comparison for Algorithm 2.3 from [252] with incorrect noise.

created a regression in DP guarantee for ANON_AVG, which was detected by the stochastic tester.

Figure 4.17 shows a pair of datasets where the stochastic tester detected a violation of the DP predicate. We can clearly see that several buckets violate the predicate. Once the stochastic tester alerted us to the error we quickly modified ANON_AVG to use the correct sensitivity[2].

This simple approach can be extended in multiple ways. First, rather than *only* using randomly-generated databases as input to the stochastic tester, we can also add manual test cases, which we expect to be "edge cases", like with extreme record values or empty databases. Second, the differential privacy test presented above can be improved using the approach

[2] In a dramatic turn of events, Alex Kulesza later found that Algorithm 2.3 from [252] is not, in fact, differentially private, even with this sensitivity fix. Consider a database D_1 with a single record 0, and D_2 with two values 0 and 1. Use $[0,1]$ as a clamping range. On input D_1, the density of the output is $\frac{\varepsilon}{2}e^{-\varepsilon}$; while on input D_2, it is $\varepsilon e^{-\varepsilon}$. Thus, the ratio between the two is 2, *independently of* ε: this is incorrect for values of ε lower than $\ln 2$.

presented in [167], which also provides formal bounds on the probability of errors. Third, differential privacy is not the only property that can be experimentally tested in a similar fashion: we can also test that the noise distribution corresponds to the one we expect, by writing a clean-room implementation of each noise distribution, and comparing both via *closeness tests* [103].

4.3 FURTHER IMPROVEMENTS AND OPEN QUESTIONS

In the previous section, we presented a generic system to answer SQL queries with user-level differential privacy. This system is remarkably simple: it simply analyzes what are the possible ways in which a typical SQL query can break differential privacy, and proposes simple building blocks to overcome these obstacles. Importantly, it solves most use cases we see in practice: counts and sums are enough to cover the vast majority of aggregation queries that analysts and engineers typically need to run on sensitive data. For these typical use cases, our simple system performs well, scales to arbitrary data size, and can be implemented and tested in a robust, production-ready fashion. We hope that this simplicity, and the open-source publication of the implementation, will encourage further research and engineering contributions in this direction, and increase the adoption of differential privacy.

In this section, we list possible improvements and extensions to our framework. We raise many questions and provide only a few answers, most of which are partial: the road to build a general-purpose, usable, scalable, and robust differentially private query engine is long and many aspects of this problem are still largely unexplored.

First, in Section 4.3.1, we discuss how to fit other kinds of aggregations within our framework. For some of them, this task is not trivial, and requires to make the computation model more generic, which presents interesting design challenges. Further, some of these primitives are simply not well-studied in the existing literature, especially under strict scalability requirements, and under the assumption that each user can contribute multiple data points.

Second, in Section 4.3.2, we come back to the problem of partition selection, introduced in Section 4.2.2.2 and solved using Laplace noise and thresholding in Section 4.2.3.6. We introduce an optimal partition selection primitive in the common special case where each user contributes to a single partition only, and we discuss its extension to other cases.

Third, in Section 4.3.3, we discuss other possible utility and usability improvements: we present Privacy on Beam, an open-source implementation of the framework described in Section 4.2 making slightly different design trade-offs, and we explain how Gaussian noise can be used to boost utility when a single user can contribute to many partitions.

Fourth, in Section 4.3.4, we focus on operational questions that arise when rolling out differential privacy tooling in practice, and discuss policy topics such as parameter choice or privacy budget tracking.

Finally, in Section 4.3.5, we take a step back and discuss other possible directions for future research on differentially private query engines.

4.3.1 *Beyond counts and sums*

Differentially private counts and sums are enough to solve a large class of use cases requiring anonymized data aggregation. However, analysts sometimes need more complex aggregations, and some of them raise interesting questions. These questions fall into two categories: how to fit these aggregations in the two-step aggregation model of our framework (per-user aggregation first, cross-user aggregation second) in a usable fashion, and how to implement the related primitives in a scalable fashion.

4.3.1.1 *Average*

Averages are a first simple example. At first glance, they seem to be nothing more than a sum divided by a count. However, to compute them this way, the user would need to specify *three* parameters: a lower and upper bound for the sum, and an upper bound of the amount of contributions. Thus, they would need to know not only the range of typical values, but also the number of values that each user typically contributes. This is a lot to ask of an analyst who simply wants to know the average of a column. We could use automatic bounds determination for both operations, but the privacy budget cost would then be particularly high.

To compute averages, an alternative option is to first compute a non-private per-user average, and then to average the results across all users in a private way. This changes the semantics of the operation: we are now computing an average of averages, instead of an average of all values. If there is a correlation between the distribution of values and the number of contributions (for example, each user contributes either few low values, or many large values), this might introduce bias. This change of semantics might seem shocking at first glance, but note that this is not limited to averages: clamping, or contribution bounding, also change the semantics of a query.

In practice, averaging averages does not create significant quality issues. Thus, in scenarios where usability is most important, like the SQL implementation of our framework, we use this method. For implementations of the framework that tend to be used by engineers with more knowledge about their data and with the ability to run exploratory analyses, we opt for the mechanism that performs a global sum and divides it by a global count.

Note that in that case, the output of the per-user aggregation is *different* than the overall result. Each per-user aggregation must return a partial sum and a partial count, while the cross-user aggregation will only return an average value. Thus, we have to modify our framework to remove the simplifying assumption that the per-user aggregation is simply the non-private version of the cross-user aggregation.

4.3.1.2 *Variance*

The variance operation has the same basic problem as the average operation, although the right decision is even more clear-cut: computing a per-user variance, and taking the variance of the result, is obviously not a valid way to compute the total variance between all values. In fact, there is no obvious way to compute a per-user partial aggregation that would allow to compute the variance of values. The only option is to preserve the original values, and for the per-user "aggregation" step to be only a *sampling* step, that selects up to a fixed number of values associated to each user. This underscores the importance of extending our framework to make sure that the per-user aggregation step can be arbitrarily different from the cross-user DP aggregation.

4.3.1.3 *Quantiles*

Quantiles present interesting challenges. First, they raise the same question as averages: is it appropriate to use the non-private version of the aggregation as a per-user aggregation step? For some values of quantiles, the answer is clearly positive: for example, to compute the minimum value of a dataset, we can take the per-user minimum, and run the differentially private minimum on this set. This limits the contribution per user to 1, which is good for utility, and does not change the semantics of the aggregation.

However, the situation is different for non-extremum quantiles. For medians, taking the per-user median value and computing the median of those can introduce the same kind of bias as for averages. Compute near-extremum quantile values (like the 1st and 99th percentile) per-user is also highly non trivial if each user only contributes a handful of values. Thus, like for averages, it might be a good idea to simply sample a fixed number of records as a per-user "aggregation" step; although deciding when to make this choice is not obvious.

Second, privately computing quantiles in a scalable way does not seem to be a solved problem. Most common options implicitly assume that all values can fit in memory, which is unsuited to large-scale datasets and massively parallel computation frameworks. This is the case for Bayesian binary search [44, 215], for smooth sensitivity [303], and for the exponential mechanism [127]. Existing methods have been proposed in the literature for the non-private case, like KLL [214], but those do not provide privacy guarantees, and naively adding noise to their result does not provide good utility, since their global sensitivity is unbounded. A naive option is to randomly sample values when there are too many values to fit them all in memory, but this option likely provides very suboptimal accuracy.

Third, all the methods that we could find in the literature on differentially private quantiles also implicitly assume that each user contributes at most one value. They can be adapted to the case where each user contributes multiple values by bounding the total number of values contributed by each user, and using composition theorems, but there is no reason why this naive strategy would provide optimal or even good utility.

Finally, many practical use cases for quantiles actually require to compute *multiple* quantiles for the same series of values. For example, it is common to compute latency at the 50th, 95th and 99th percentile [236]. When naively using the private methods mentioned above, each such percentile must be computed on a separate privacy budget, which is likely far from optimal. We leave it to future work to improve the state-of-the-art of differentially private quantiles.

4.3.1.4 *Counting distinct values*

Counting distinct values corresponds to the following operation.

```
SELECT
```

```
  website,
  COUNT(DISTINCT browser_agent)
FROM access_logs
GROUP BY website
```

Listing 4.10: Count distinct operation

This is easy to do in a scalable way if we want to count unique user identifiers, but if we want to count distinct *values* (here, browser agents), it presents an interesting scalability challenge. If the total number of distinct values is small enough to keep all of them in memory, the problem is trivial: simply add Laplace noise to the exact count, scaled by the maximum number of contributions for any single user. However, if linear memory usage is not an option because of scalability requirements, we need a smarter approach.

Many non-private sketching algorithms have been proposed for this problem, and a number of them are listed in in Section 3.3.1.2. How to make their output differentially private? Note that this is a fundamentally different problem than the one tackled in Section 3.3: we do not want to protect the sketches themselves, but only their output. A noisy variant of Bloom filters was proposed for a related problem in [9], and in [77], authors show that LogLog [120] achieves differential privacy "for free" if we assume the hashes used in the algorithm are computed after adding a random salt to the input values. This method can serve as a first approach to implement this important primitive.

In the generic case where each user contributes multiple values to the initial dataset, another natural problem emerges: how to *choose* the values that each user contributes in an optimal way? For example, assume each user contributes two values: one fixed value a which is the same across all users, and a value b_i that is different for each user. If the contribution bound is fixed to 1, choosing a value at random between a and b_i for each user is very suboptimal: we should, instead, try to pick *rare* values to increment the total count. This raises a problem similar to the one tackled in [179]: the choice for each user should depend on the choice for other users, but this obviously makes it trickier to calculate sensitivity. We are not aware of any research on this specific point; we leave this as an open research question.

4.3.2 *Partition selection*

Recall the running example of Section 4.2.2.

```
SELECT
  browser_agent,
  COUNT(*) AS visits
FROM access_logs
GROUP BY browser_agent
```

Listing 4.11: Simple histogram query

One of the pitfalls of making such a query differentially private, identified in Section 4.2.2.2, is to select which *partitions* (here, browser agents) will be present in the output. In Section 4.2.2.2 and Section 4.2.3.6, we reused an insight from [230], and used Laplace-based thresholding to avoid this pitfall: we essentially count unique users associated with each partition, add Laplace noise to each count, and keep only the partitions whose counts are above a fixed threshold. The scale of the noise and the threshold value determine ε and δ.

In this section, we explore possible improvements to this partition selection method. We start by discussing prior work in more detail (Section 4.3.2.1) and introducing definitions (Section 4.3.2.2). Then, we present a partition selection mechanism for the case where each user contributes to one partition, prove its optimality (Section 4.3.2.3), and experimentally compare it to existing methods (Section 4.3.2.4). We then discuss possible extensions to cases where each user contributes to multiple partitions as well as implementation considerations (Section 4.3.2.5).

4.3.2.1 *Prior work and contributions*

Even though Laplace thresholding was introduced in 2009 in [230], the specific primitive of partition selection did not much attention until [179], where the authors call the generic problem *differentially private set union*. Each user is associated with one or several partitions, and the goal is to release as many partitions as possible while making sure that the output is differentially private.

In [179], the main use case is word and n-gram discovery in Natural Language Processing: data used in training models must not leak private information about individuals. In this context, each user potentially contributes to many elements; the sensitivity of the mechanism can be high. The authors propose two strategies applicable in this context. First, they use a *weighted* histogram so that if a user contributes to fewer elements than the maximum sensitivity, these elements can add more weight to the histogram count. Second, they introduce *policies* that determine which elements to add to the histogram depending on which histogram counts are already above the threshold. These strategies obtain significant utility improvements over the simple Laplace-based strategy.

In this work, in contrast to [179], we focus on the *low-sensitivity* use case: each user contributes to exactly one partition. This different setting is common in data analysis: when the `GROUP BY` operation partitions the set of users in distinct partitions, each user contributes exactly one element to the set union. Choosing the contributions of each user is therefore not relevant; the only question is to optimize the probability of releasing each element in the final result. For this specific problem, we introduce an optimal approach, which maximizes this probability.

4.3.2.2 *Definitions*

Throughout most of this work, we will assume that each user contributes to only one partition; and the goal is to release as many partitions as possible. In that case, each partition can be considered independently, so the problem is simple to model. Each partition has a certain number of users associated with it, and the only question is: with which probability do we release this partition? Thus, a strategy for partition selection is simply a function associating the number of users in a partition with the probability of keeping the partition.

Definition 82 (Partition selection primitive)**.** *A* partition selection primitive *is a function* $\pi : \mathbb{N} \rightarrow [0,1]$ *such that* $\pi(0) = 0$*. The corresponding* partition selection strategy ρ_π *counts the number n of users in each partition, and releases this partition with probability* $\pi(n)$*.*

Formally, we say that a partition selection primitive is (ε, δ)*-differentially private if the corresponding partition selection strategy* $\rho_\pi : \mathbb{N} \rightarrow \{\text{drop}, \text{keep}\}$*, defined by:*

$$\rho_\pi(n) = \begin{cases} \text{drop} & \text{with probability } 1 - \pi(n) \\ \text{keep} & \text{with probability } \pi(n) \end{cases}$$

is (ε, δ)*-differentially private.*

Note that partitions associated with no users are not present at all in the input data, so the probability of releasing them must be 0: the definition requires $\pi(0) = 0$.

4.3.2.3 *Main result*

Let us define an (ε, δ)-DP partition selection primitive π_{opt} and prove that the corresponding partition selection strategy is optimal. In this context, optimal means that it maximizes the probability of releasing a partition with n users, for all n.

Definition 83 (Optimal partition selection primitive)**.** *A partition selection primitive* π_{opt} *is* optimal for (ε, δ)-DP *if it is* (ε, δ)*-DP, and if for all* (ε, δ)*-DP partition selection primitives* π *and all* $n \in \mathbb{N}$*:*

$$\pi(n) \leq \pi_{\text{opt}}(n).$$

We introduce our main result, then we prove it in two steps: we first prove that the optimal partition selection primitive can be obtained recursively, then derive the closed-form formula of our main result from the recurrence relation.

Theorem 13 (General solution for π_{opt})**.** *Let* $\varepsilon > 0$ *and* $\delta \in (0,1)$*. Defining:*

$$n_1 = 1 + \left\lfloor \frac{1}{\varepsilon} \ln\left(\frac{e^\varepsilon + 2\delta - 1}{(e^\varepsilon + 1)\delta} \right) \right\rfloor,$$
$$n_2 = n_1 + \left\lfloor \frac{1}{\varepsilon} \ln\left(1 + \frac{e^\varepsilon - 1}{\delta} \left(1 - \pi_{\text{opt}}(n_1)\right) \right) \right\rfloor,$$

and $m = n - n_1$*, the partition selection primitive* π_{opt} *defined by:*

$$\pi_{\text{opt}}(n) = \begin{cases} \frac{e^{n\varepsilon}-1}{e^\varepsilon-1} \cdot \delta & \text{if } n \leq n_1 \\ \left(1 - e^{-m\varepsilon}\right)\left(1 + \frac{\delta}{e^\varepsilon-1}\right) + e^{-m\varepsilon}\pi_{\text{opt}}(n_1) & \text{if } n > n_1 \text{ and } n \leq n_2 \\ 1 & \text{otherwise} \end{cases}$$

is optimal for (ε, δ)*-DP.*

These formulas assume $\varepsilon > 0$ and $\delta > 0$. We also cover the special cases where $\varepsilon = 0$ or $\delta = 0$.

Theorem 14 (Special cases for π_{opt})**.**

1. *If $\delta = 0$, partition selection is impossible: the optimal partition selection primitive π_{opt} for $(\varepsilon, 0)$-DP is defined by $\pi_{\text{opt}}(n) = 0$ for all n.*

2. *If $\varepsilon = 0$, the optimal partition selection primitive π_{opt} for $(0, \delta)$-DP is defined by $\pi_{\text{opt}}(n) = \min(1, n\delta)$ for all n.*

The rest of this section is a proof of Theorem 13.

Recursive construction

How do we construct a partition selection primitive π so that the partition is output with the highest possible probability under the constraint that π is (ε, δ)-DP? Using the definition of differential privacy, the following inequalities must hold for all $n \in \mathbb{N}$.

$$\pi(n+1) \le e^{\varepsilon}\pi(n) + \delta \tag{4.1}$$

$$\pi(n) \le e^{\varepsilon}\pi(n+1) + \delta \tag{4.2}$$

$$(1 - \pi(n+1)) \le e^{\varepsilon}(1 - \pi(n)) + \delta \tag{4.3}$$

$$(1 - \pi(n)) \le e^{\varepsilon}(1 - \pi(n+1)) + \delta. \tag{4.4}$$

These inequalities are not only necessary, but also *sufficient* for π to be DP. Thus, the optimal partition selection primitive can be constructed by recurrence, maximizing each value while still satisfying the inequalities above. As we will show, only inequalities (4.1) and (4.4) above need be included in the recurrence relationship. The latter can be rearranged as:

$$\pi_{\text{opt}}(n+1) \le 1 - e^{-\varepsilon}(1 - \pi_{\text{opt}}(n) - \delta)$$

which leads to the following recursive formulation for π_{opt}.

Lemma 13 (Recursive solution for π_{opt})**.** *Given $\delta \in [0, 1]$ and $\varepsilon \ge 0$, π_{opt} satisfies the following recurrence relationship: $\pi_{\text{opt}}(0) = 0$, and for all $n > 0$:*

$$\pi_{\text{opt}}(n) = \min\left(e^{\varepsilon}\pi_{\text{opt}}(n-1) + \delta, 1 - e^{-\varepsilon}(1 - \pi_{\text{opt}}(n-1) - \delta), 1\right) \tag{4.5}$$

Proof. Let π_0 be defined by recurrence as above; we will prove that $\pi_0 = \pi_{\text{opt}}$.

First, let us show that π_0 is monotonic. Fix $n \in \mathbb{N}$. It suffices to show for each argument of the min function in (4.5) is larger than $\pi_0(n)$.

First argument: since $\varepsilon \ge 0$ implies $e^{\varepsilon} \ge 1$ and $\delta \ge 0$, we have $e^{\varepsilon}\pi_0(n) + \delta \ge \pi_0(n)$.

Second argument: we have

$$\begin{aligned} 1 - e^{-\varepsilon}(1 - \pi_0(n) - \delta) &= 1 - e^{-\varepsilon}(1 - \pi_0(n)) + e^{-\varepsilon}\delta \\ &\ge 1 - (1 - \pi_0(n)) \\ &= \pi_0(n) \end{aligned}$$

using that $1 - \pi_0(n) \ge 0$ since $\pi_0(n) \le 1$ by (4.5).

Third argument: this is immediate given (4.5) and the fact that $\pi_0(0) = 0$.

It follows that $\pi_0(n+1) \geq \pi_0(n)$.

Because π_0 is monotonic, it immediately satisfies inequalities (4.2) and (4.3), and inequalities (4.1) and (4.4) are satisfied by definition.

Since π_0 satisfies all four inequalities above, it is (ε, δ)-DP. Its optimality follows immediately by recurrence: for each $n+1$, if $\pi(n+1) > \pi_{\text{opt}}(n+1)$, it cannot be (ε, δ)-DP, as one of the inequalities above is not satisfied: π_0 is the fastest-growing DP partition selection strategy, and therefore equal to π_{opt}. □

The special cases for π_{opt} in Theorem 14 can be immediately derived from Lemma 13: the rest of this section focuses on proving the general form in Theorem 13.

Derivation of the closed-form solution

Let us now show that the closed-form solution of Theorem 13 can be derived from the recursive solution in Lemma 13. First, we show that there is a crossover point n_1, below which only the first term of the recurrence relation matters, and after which only the second term matters (until $\pi_{\text{opt}}(n)$ reaches 1).

Lemma 14. *Assume $\varepsilon > 0$ and $\delta > 0$. There are crossover points $n_1, n_2 \in \mathbb{N}$ such that $0 < n_1 \leq n_2$ and:*

$$\pi_{\text{opt}}(n) = \begin{cases} 0 & \text{if } n = 0 \\ \pi_{\text{opt}}(n-1)\, e^{\varepsilon} + \delta & \text{if } n > 0 \text{ and } n \leq n_1 \\ 1 - e^{-\varepsilon}\left(1 - \pi_{\text{opt}}(n-1) - \delta\right) & \text{if } n > n_1 \text{ and } n \leq n_2 \\ 1 & \text{otherwise.} \end{cases} \tag{4.6}$$

Proof. We consider the arguments in the min statement in (4.5), substituting x for $\pi_{\text{opt}}(n)$:

$$\begin{aligned} \alpha_1(x) &= e^{\varepsilon} x + \delta \\ \alpha_2(x) &= 1 - e^{-\varepsilon}(1 - x - \delta) \\ \alpha_3(x) &= 1 \end{aligned}$$

This substitution allows us to work directly in the space of probabilities instead of restricting ourselves to the sequence $(\pi_{\text{opt}}(n))_{n=0}^{\infty}$. Taking the first derivative of these functions yields:

$$\begin{aligned} \alpha_1'(x) &= e^{\varepsilon} \\ \alpha_2'(x) &= e^{-\varepsilon} \\ \alpha_3'(x) &= 0 \end{aligned}$$

Since the derivative of $\alpha_1(x) - \alpha_2(x)$ is $e^{\varepsilon} - e^{-\varepsilon} > 0$, there exists at most one crossover point x_1 such that $\alpha_1(x) < \alpha_2(x)$ for all $x < x_1$, $\alpha_2(x_1) = \alpha_1(x_1)$, and $\alpha_1(x) > \alpha_2(x)$ for all $x > x_1$. Setting $\alpha_1(x) = \alpha_2(x)$ and solving for x yields:

$$e^{\varepsilon} x + \delta = 1 - e^{-\varepsilon}(1 - x - \delta)$$

which leads to:

$$e^{\varepsilon}x - e^{-\varepsilon}x = 1 - \delta - e^{-\varepsilon}(1 - x - \delta)$$

and finally:

$$x_1 = (1 - \delta) \cdot \frac{1 - e^{-\varepsilon}}{e^{\varepsilon} - e^{-\varepsilon}}.$$

Since the derivative of $\alpha_2(x) - \alpha_3(x)$ is $e^{-\varepsilon} > 0$, there exists at most one crossover point x_2 such that $\alpha_2(x) < \alpha_3(x)$ for all $x < x_2$, $\alpha_2(x_2) = \alpha_3(x_2)$, and $\alpha_2(x) > \alpha_3(x)$ for all $x > x_2$. Setting $\alpha_2(x) = \alpha_3(x)$ and solving for x yields:

$$x_2 = 1 - \delta.$$

From the formulas for x_1 and x_2, it is immediate that $0 < x_1 < x_2 < 1$. As such, the interval $[0, 1]$ can be divided into three non-empty intervals:

1. On $[0, x_1]$, $\alpha_1(x)$ is the active argument of $\min(\alpha_1(x), \alpha_2(x), \alpha_3(x))$.
2. On $[x_1, x_2]$, $\alpha_2(x)$ is the active argument of $\min(\alpha_1(x), \alpha_2(x), \alpha_3(x))$.
3. On $[x_2, 1]$, $\alpha_3(x)$ is the active argument of $\min(\alpha_1(x), \alpha_2(x), \alpha_3(x))$.

The existence of the crossover points is not enough to prove the lemma: we must also show that these points are reached in a finite number of steps. For all $n \geq 1$ such that $\pi_{\text{opt}}(n) \neq 1$, we have:

$$\begin{aligned}
&\pi_{\text{opt}}(n) - \pi_{\text{opt}}(n-1) \\
&\qquad = \min\left(e^{\varepsilon}\pi_{\text{opt}}(n-1) + \delta, 1 - e^{-\varepsilon}\left(1 - \pi_{\text{opt}}(n-1) - \delta\right)\right) - \pi_{\text{opt}}(n-1) \\
&\qquad \geq \min\left(\delta, \left(1 - e^{-\varepsilon}\right)\left(1 - \pi_{\text{opt}}(n-1)\right) + e^{-\varepsilon}\delta\right) \\
&\qquad \geq e^{-\varepsilon}\delta.
\end{aligned}$$

Since $\pi_{\text{opt}}(n) - \pi_{\text{opt}}(n-1)$ is bounded from below by a strictly positive constant $e^{-\varepsilon}\delta$, the sequence achieves the maximal probability 1 for finite n. □

This allows us to derive the closed-form solution for $n < n_1$ and for $n_1 \leq n < n_2$ stated in Theorem 13.

Lemma 15. *Assume $\varepsilon > 0$ and $\delta \leq 0$. If $n \leq n_1$, then $\pi_{\text{opt}}(n) = \frac{e^{n\varepsilon}-1}{e^{\varepsilon}-1} \cdot \delta$. If $n_1 \leq n < n_2$, then denoting $m = n - n_1$:*

$$\pi_{\text{opt}}(n) = \left(1 - e^{-m\varepsilon}\right)\left(1 + \frac{\delta}{e^{\varepsilon} - 1}\right) + e^{-m\varepsilon}\pi_{\text{opt}}(n_1).$$

Proof. For $n < n_1$, expanding the recurrence relation yields:

$$\begin{aligned}
\pi_{\text{opt}}(n) &= \pi_{\text{opt}}(n-1)\, e^{\varepsilon} + \delta \\
&= \delta \sum_{k=0}^{n-1} e^{k\varepsilon} \\
&= \frac{e^{n\varepsilon} - 1}{e^{\varepsilon} - 1} \cdot \delta.
\end{aligned}$$

For $n_1 \leq n < n_2$, denoting $m = n - n_1$, expanding the recurrence relation yields:

$$\begin{aligned}\pi_{\text{opt}}(n) &= 1 - e^{-\varepsilon}\left(1 - \pi_{\text{opt}}(n-1) - \delta\right) \\ &= \left(1 - e^{-\varepsilon} + \delta e^{-\varepsilon}\right)\sum_{k=0}^{m-1} e^{-k\varepsilon} + e^{-m\varepsilon}\pi_{\text{opt}}(n_1) \\ &= \left(1 - e^{-\varepsilon} + \delta e^{-\varepsilon}\right)\frac{1 - e^{-m\varepsilon}}{1 - e^{-\varepsilon}} + e^{-m\varepsilon}\pi_{\text{opt}}(n_1) \\ &= \left(1 - e^{-m\varepsilon}\right)\left(1 + \frac{\delta}{e^{\varepsilon} - 1}\right) + e^{-m\varepsilon}\pi_{\text{opt}}(n_1).\end{aligned}$$

□

We can now find a closed-form solution for n_1 and for n_2.

Lemma 16. *The first crossover point n_1 is:*

$$n_1 = 1 + \left\lfloor \frac{1}{\varepsilon}\ln\left(\frac{e^{\varepsilon} + 2\delta - 1}{\delta(e^{\varepsilon} + 1)\delta}\right)\right\rfloor$$

Proof. Using the formula for x_1 in the proof of Lemma 14, we see that $\pi_{\text{opt}}(n-1) \leq x_1$ whenever:

$$\frac{e^{(n-1)\varepsilon} - 1}{e^{\varepsilon} - 1} \cdot \delta \leq \frac{1 - \delta}{e^{\varepsilon} + 1}.$$

Rearranging terms, we can rewrite this inequality as:

$$\begin{aligned}n &\leq 1 + \frac{1}{\varepsilon}\ln\left[\frac{(1-\delta)(e^{\varepsilon} - 1)}{\delta(e^{\varepsilon} + 1)} + 1\right] \\ &= 1 + \frac{1}{\varepsilon}\ln\left[\frac{(1-\delta)(e^{\varepsilon} - 1) + \delta(e^{\varepsilon} + 1)}{\delta(e^{\varepsilon} + 1)}\right] \\ &= 1 + \frac{1}{\varepsilon}\ln\left[\frac{e^{\varepsilon} + 2\delta - 1}{\delta(e^{\varepsilon} + 1)}\right].\end{aligned}$$

Since n is an integer, the supremum value defining n_1 is achieved by taking the floor of the right-hand side of this inequality, which concludes the proof. □

Lemma 17. *The second crossover point n_2 is:*

$$n_2 = n_1 + \left\lfloor \frac{1}{\varepsilon}\ln\left(1 + \frac{e^{\varepsilon} - 1}{\delta}\left(1 - \pi_{\text{opt}}(n_1)\right)\right)\right\rfloor$$

Proof. We want to find the maximal m such that:

$$\left(1 - e^{-m\varepsilon}\right)\left(1 + \frac{\delta}{e^{\varepsilon} - 1}\right) + e^{-m\varepsilon}\pi_{\text{opt}}(n_1) \leq 1.$$

We can rewrite this condition into:

$$-e^{-m\varepsilon}\left(1 + \frac{\delta}{e^{\varepsilon} - 1} - \pi_{\text{opt}}(n_1)\right) \leq \frac{-\delta}{e^{\varepsilon} - 1}$$

which leads to:

$$\begin{aligned} e^{m\varepsilon} &\leq \frac{e^{\varepsilon}-1}{\delta}\left(1+\frac{\delta}{e^{\varepsilon}-1}-\pi_{\text{opt}}\left(n_1\right)\right) \\ &\leq 1+\frac{e^{\varepsilon}-1}{\delta}\left(1-\pi_{\text{opt}}\left(n_1\right)\right) \end{aligned}$$

and finally:

$$m \leq \frac{1}{\varepsilon}\ln\left(1+\frac{e^{\varepsilon}-1}{\delta}\left(1-\pi_{\text{opt}}\left(n_1\right)\right)\right)$$

since m must be an integer, we take the floor of the right-hand side of this inequality to obtain the result. □

4.3.2.4 *Numerical validation*

Theorem 13 shows that the optimal partition selection primitive π_{opt} outperforms all other options. How does it compare with our previous strategy of adding Laplace noise and thresholding the result, described in Section 4.2.3.6? For simplicity, we recall this strategy in the simpler case where the sensitivity is one.

Definition 84 (Laplace-based partition selection [230])**.** *We denote by* Lap(b) *a random variable sampled from a Laplace distribution of mean* 0 *and of scale b. The following partition selection strategy* ρ_{Lap}*, called* Laplace-based partition selection*, is* (ε, δ)*-differentially private:*

$$\rho_{\text{Lap}}(n) = \begin{cases} \text{drop} & \text{if } n + \text{Lap}\left(\frac{1}{\varepsilon}\right) < 1 - \frac{\ln(2\delta)}{\varepsilon} \\ \text{keep} & \text{otherwise.} \end{cases}$$

We denote by π_{Lap} *the corresponding partition selection primitive:*

$$\pi_{\text{Lap}}(n) = \mathbb{P}\left[\rho_{\text{Lap}}(n) = \text{keep}\right].$$

As expected, using the optimal partition selection primitive translates to a larger probability of releasing a partition with the same user. As shown in Figure 4.18, the difference is especially large in the high-privacy regime.

To better understand the dependency on ε and δ, we also compare the *midpoint* obtained for both partition selection strategies ρ: the number n for which the probability of releasing a partition with n users is 0.5. For Laplace-based partition selection, this n is simply the threshold. As Figure 4.19 shows, the gains are especially substantial when ε is small, and not significant for $\varepsilon > 1$. Figure 4.20 shows the dependency on δ: for a fixed ε, there is a *constant* interval between the midpoints of both strategies. Thus, the relative gains are larger for a larger δ, since the midpoint is also smaller.

4.3.2.5 *Discussion*

The approach presented here is both easy to implement and efficient. The random decision for a given partition takes a constant time to compute, thanks to the closed-form formula in

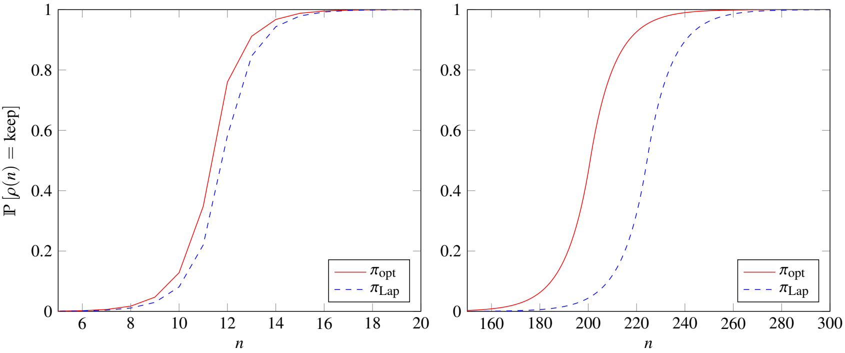


FIGURE 4.18: Probability of releasing a partition depending on the number of unique users, comparing Laplace-based partition selection with π_{opt}. On the left, $\varepsilon = 1$ and $\delta = 10^{-5}$; on the right, $\varepsilon = 0.1$ and $\delta = 10^{-10}$.

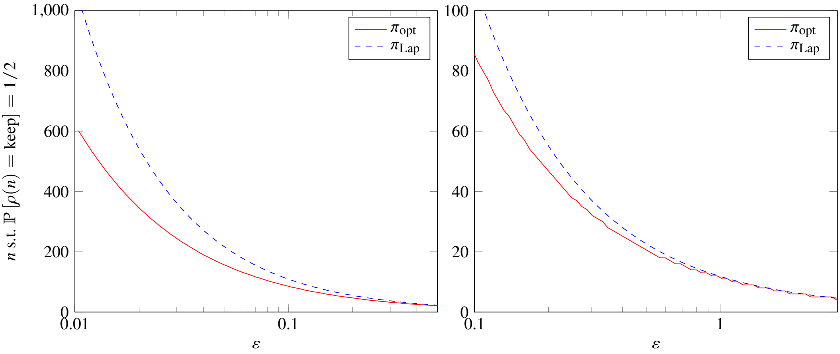


FIGURE 4.19: Comparison of the mid-point of the partition selection strategy ρ as a function of ε, for $\delta = 10^{-5}$.

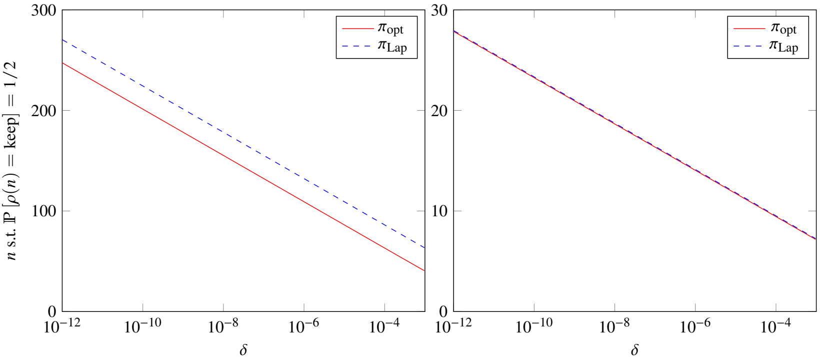


FIGURE 4.20: Comparison of the mid-point of the partition selection strategy ρ as a function of δ, for $\varepsilon = 0.1$ (left) and $\varepsilon = 1$ (right).

Theorem 13. Counting the number of unique users per partition can be done in one pass over the data and is massively parallelizable. Furthermore, since there is a relatively small value N such that the probability of keeping a partition with $n \geq N$ users is 1, the counting process can be interrupted as soon as a partition reaches N users. This keeps memory usage low (in $O(N)$) without requiring approximate count-distinct algorithms like HyperLogLog, for which a more complex sensitivity analysis would be needed.

Our approach could, in principle, be extended to cases where each user can contribute to $\Delta > 1$ partitions. Following the intuition of Lemma 13, we could list a set of recursive equations defining $\pi_{\text{opt}}(n)$ as a function of $\pi_{\text{opt}}(k)$ for $k < n$. However, this recursive formulation is much more complex when Δ is large, for multiple reasons.

1. With Δ partitions to consider at the same time, the set of possible outcomes has cardinality 2^{Δ}: keep or drop for each partition.

2. To compute $\pi_{\text{opt}}(n)$ by recurrence, we must consider a large set of possible neighboring databases: each of the other $\Delta - 1$ contributions of the user can have anywhere from 0 to $n - 1$ users. Separately considering all those $(\Delta - 1)^n$ possibilities quickly becomes intractable.

3. For each of these possibilities, the probability can be expressed as a polynomial of degree Δ in the values of $\pi_{\text{opt}}(k)$, for $k < n$. Solving these also becomes intractable as Δ increases.

The above approach might be workable for $\Delta = 2$ or even $\Delta = 3$, but the complexity cost is likely too high to be worth implementing. Furthermore, the recurrence-based proof of optimality of π_{opt} only holds assuming that each user contributes to *exactly* Δ partitions in the original dataset, so strategies based on selecting *which* partitions to contribute to, or *weighing* the partition of each user if there are fewer than Δ, cannot bring additional benefits. This case is relatively frequent for $\Delta = 1$, but rarely happens for larger values of Δ.

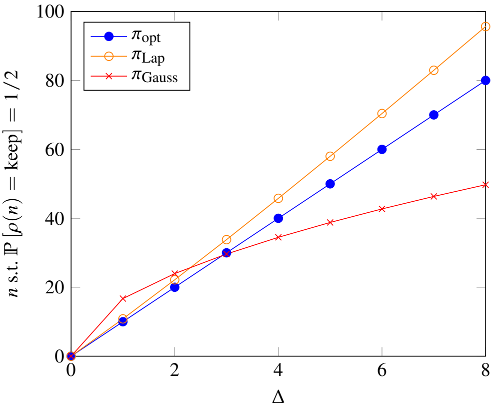


FIGURE 4.21: Comparison of our method with Laplace-based and Gaussian-based thresholding, with $\varepsilon = 1.1$ and $\delta = 10^{-5}$. For our method and Laplace-based thresholding, we split the privacy budget in Δ equal parts. For Gaussian-based thresholding, we use the formula in Theorem 16 (Section 4.3.3.2) to add noise, and split δ between noise addition and thresholding in a way that minimizes the threshold.

Thus, this work leaves two obvious open questions. Is it possible to extend to overcome the problems described above, and extend our optimal approach to larger sensitivities in a simple and efficient manner? Furthermore, is it possible to combine this primitive with existing approaches to differentially private set union [179], like weighted histograms or policy-based strategies?

In the meantime, can we simply use this primitive for the low-sensitivity use case, and adopt the approach from [179] when each user can contribute to multiple partitions? If scalability is a hard requirement, then the answer is not straightforward.

- The policy-based approaches described in [179] require the values of each user one after the other, and compare them with the histogram previously built from all previous users. The linearity prevents us from implementing the algorithm in a massively parallel fashion, and the full histogram must fit in memory. Both are significant obstacles to scalability; designing alternative algorithms with better scalability properties is left as an open question.

- However, part of their core insights can still be used. For most values of ε and δ, Gaussian noise gives better results for $\Delta > 3$ than naively splitting the budget across contributions and using our partition selection primitive, as shown in Figure 4.21. Furthermore, for users who contribute fewer values than Δ, giving these contributions more weight in the histogram is also possible.

4.3.3 *Improving coverage and utility*

In Section 4.2, we presented a framework for computing differentially private statistics, and we explained how it could be implemented as an extension of a general-purpose SQL engine. SQL has major usability advantages: the ability to write SQL queries is more common than the ability to write code in a traditional programming language. The declarative aspect of SQL also gives us a large freedom to rewrite or optimize the way queries are run, and semantic analysis is much easier to perform on SQL queries rather than in a Turing-complete programming language.

However, more powerful programming languages and computation models also come with a number of distinct advantages. In this section, we quickly present an alternative implementation of our framework on *Apache Beam* [17], then we illustrate the possible benefits that can be gained by such an alternative.

4.3.3.1 *Privacy on Beam*

SQL is the tool of choice of many data analysts, but many engineers who build pipelines for large-scale systems opt for more powerful computation frameworks, like Apache Beam [17], an evolution of prior technologies like MapReduce [95]. Apache Beam is a programming model in which a client can describe a data processing pipeline, and run it in a massively parallel fashion without having to implement parallelization themselves, or worry about technical details like redundancy.

Apache Beam has client libraries written in Java, Go and Python. They allow clients to specify complex data transformations that would be difficult to express in SQL, either because of syntactic inelegance (e.g. chaining a large number of search & replace operations on a string) or lack of language support (e.g. computations that require loops, or calling an external service). Such examples are very common for complex data pipelines, which explains the popularity of frameworks like Apache Beam in organizations running large-scale computational tasks. Furthermore, even though frameworks like Apache Beam have a declarative aspect (high-level API functions do not provide interface guarantees on their execution plan), it is often easier to understand which *actual* computation tasks are executed when running a pipeline, allowing engineers to finely optimize their performance if needed.

Migration costs are one of the main obstacles to the wide-scale adoption of privacy technologies: nobody likes to hear that they will need to change their practices and infrastructure entirely to add a layer of privacy protection, and asking people to do so is usually a losing battle. This is especially the case if their use case requires using a more powerful framework than the one we are trying to push them towards. Thus, it is crucial to implement differentially privacy features in the tools that people are already using. This is a core reason why we invested in adapting the model we presented in Section 4.2 to frameworks like Apache Beam. We have done so and published an open-source implementation in Go, available at [320]. In this section, we discuss the advantages of using a more powerful framework, and the challenges that came with this project.

Overview

Let us give an example, freely inspired from the public Codelab for Privacy on Beam [83]. The following Apache Beam pipeline, using the Go SDK, assumes that the input data is a collection of structs of type `Visit`, which encapsulate information about the visit of individuals to a given shop.

```
type Visit struct {
  VisitorID  string
  Day        time.Date
  EurosSpent int
}
```

Suppose we want to compute the total euros spent by weekday. A traditional Apache Beam pipeline would look like the following, assuming that the initial collection is stored in a variable named `input`.

```
func extractDayAndSpend(v Visit) (time.Date, int) {
  return v.Day, v.EurosSpent
}

// Initialize the pipeline and its scope
p := beam.NewPipeline()
s := p.Root()
input := readInput(s, "path/to/file")

// Extract day and spend from each Visit
extracted := beam.ParDo(s, extractDayAndSpend, input)
// Sum the spend of each user per day
totalSpent := stats.SumPerKey(s, extracted)

// Determine where to results and execute the pipeline
writeOutput(s, output)
runner.Execute(context.Background(), p)
```

In the code above, `input` is a `PCollection`<`Visit`>: it is conceptually similar to an array, except elements are not directly accessible, as a `PCollection` can represent a very large collection stored across multiple machines. Then, `extracted` and `totalSpent` are `PCollections` with (key, value) type <`time.Date,int`>.

Now, to make this pipeline differentially private, we need to first encapsulate the input into a `PrivatePCollection`, which will implicitly track user identifiers.

```
// Encapsulate the input
epsilon, delta := 1.0, 1e-10
privacySpec := pbeam.NewPrivacySpect(epsilon, delta)
pcol := pbeam.MakePrivateFromStruct(
  s, input, privacySpec, "VisitorID"
)

// Extract day and spend from each Visit
extracted := pbeam.ParDo(s, extractDayAndSpend, pcol)
// Sum the spend of each user per day
sumParams := pbeam.SumParams{
  MaxPartitionsContributed: 5,
  MinValue: 0,
  MaxValue: 100,
}
totalSpent := pbeam.SumPerKey(s, extracted, sumParams)
```

The `MaxPartitionsContributed`, `MinValue`, and `MaxValue` parameters are conceptually the same as the κ, `L` and `U` introduced in Section 4.2: how many different partitions a single user can contribute to, and how many euros can they contribute at most in a single partition.

Interface design

A SQL query is conceptually easy to understand: there is a clear input and output, and it is relatively easy to check that it satisfies all the constraints of our framework (see Section 4.2.3.7). By contrast, an Apache Beam pipeline can have multiple outputs, and contains transformations with arbitrarily complex logic implemented in a Turing-complete language. Thus, it is much more difficult to automatically analyze its semantics, or to use rewriting operations like in SQL. This makes the adaptation of our framework difficult.

We solve this problem by introducing `PrivatePCollection`. The high-level intuition is that a `PrivatePCollection` *encapsulates* a `PCollection`, and provides a simple yet powerful interface guarantee: all `PCollection`s that are output from this `PrivatePCollection` are differentially private, with the (ε, δ) parameters and user identifier given during initialization. Not only does this cleanly delineates between the private and the non-private part, but it also prevents library users from mistakenly copying data that has not been made differentially private yet.

This interface is also easy to understand for programmers already familiar with Apache Beam: `PrivatePCollection` can be used just like `PCollection`, with additional constraints to ensure privacy guarantees. Ideally, an operation is allowed by the interface if and only if it is compatible with our privacy model.

Behind the scenes, a `PrivatePCollection`<`Visit`> is actually a `PCollection` of type <`userID,Visit`>: we keep track of the user identifier associated with each record, and this user identifier is then used for all privacy-sensitive operations. This plays the same role as the subquery constraints from Section 4.2.3.2, which also guarantee that each record belongs to a single user. Importantly, per-user transformations are still allowed: using `pbeam.ParDo`, the equivalent of `beam.ParDo`, a client can use arbitrary logic to transform a record, including filtering records out and creating multiple records from a single one: the outputs of a `pbeam.ParDo` transformation on a record owned by a given user ID will also be owned by the same user ID.

Improved expressibility

The power of a programming language like Go is convenient to clients of the library, as it enables them to implement complex transformations. It also presents an opportunity for us to surface more options to the client, so they can fine-tune the behavior of their differential privacy pipeline and optimize the utility of the data they generate.

We briefly touched on an example in Section 4.3.1.1: in the SQL implementation of our framework, we compute averages as the average of per-user averages, for usability reasons: many aggregations have the same two options (lower and upper bound), so having a third option for averages would be confusing and error-prone, as SQL options are specified as unnamed arguments to the corresponding function. When using Apache Beam, this problem is somewhat mitigated: we can use structs with named fields as options for each aggregation, which makes the code easier to read.

The possibilities offered by a powerful framework like Apache Beam do not stop there. In the SQL version of our framework, a single value of κ was used within all aggregations of a query, and the privacy budget was shared equally across aggregations. Again, usability is a main reason for these technical choices: surfacing too many options is awkward and confusing. But these choices are not optimal in general, and Apache Beam offers more customization options so pipeline owners can tweak these values to optimize the utility of their data more finely.

Finally, in our SQL system, the partition selection primitive was silently added next to the other aggregations present in a query. By contrast, in Privacy on Beam, generating a list of partitions can be a standalone primitive, and differentially private aggregations can use a fixed list of partitions. This is not only useful to manually set the parameters used within this primitive (exact mechanism, privacy budget, number of partitions each user can contribute to), but also allows an analyst to manually specify a list of partitions that is not data-dependent, and skip thresholding altogether.

4.3.3.2 *Gaussian noise*

The Laplace mechanism is, by far, the most common way of building differentially private algorithms. Its simplicity makes it particularly easy to reason about and use in a framework like ours: it provides perfect ε-DP (with $\delta = 0$) for a single metric, it scales linearly with the sensitivity of each aggregation, and with the maximum number of partitions that a single user can contribute to. This last fact is a direct application from the basic composition theorem (Proposition 5, Section 2.1.6.3), which is known not to be tight [133, 213]. In particular, in [213], authors prove the following composition theorem, which is tight in general.

Theorem 15 (Theorem 3.3 in [213])**.** *For any* $\varepsilon > 0$ *and* $\delta \in [0, 1]$*, the sequential composition of k independent* (ε, δ)*-DP mechanisms satisfies* (ε', δ')*-DP, with:*

$$\begin{aligned}\varepsilon' &= (k - 2i)\varepsilon \\ \delta' &= 1 - (1 - \delta)^k (1 - \delta_i)\end{aligned}$$

for all integers $0 \leq i \leq \lfloor k/2 \rfloor$*, where:*

$$\delta_i = \frac{\sum_{l=0}^{i-1} \binom{k}{l} \left(e^{(k-l)\varepsilon} - e^{(k-2i+l)\varepsilon}\right)}{(1 + e^{\varepsilon})^k}.$$

Thus, a natural option for improving utility is to use this result to split the privacy budget between different aggregations, and between the different contributions of a single user across partitions in each aggregation. However, this first idea is less attractive than it initially appears. First, it only brings significant utility gains for large values of k (larger than 20), which limits its usefulness in practice, as the maximum number of contributions a single user contributes to is often lower. Second, it is very tricky to implement: one needs to reverse the formula above to split a given privacy budget, and we very quickly encounter floating-point issues when doing so.

Note that this tight composition theorem is *generic*: it holds for *any* (ε, δ)-DP mechanism. But in our framework, we really only care about the noise mechanism used in practice. If we can find a better composition result for Laplace noise, or even change the type of noise we are using, we might get stronger results and pay a smaller complexity cost.

The Gaussian mechanism turns out to be a strong candidate for such an optimization: instead of using noise drawn from a Laplace distribution, the Gaussian mechanism adds Gaussian noise to the result of the original query. We first introduce this mechanism, then discuss its implementation, and finish by pointing out a few natural open questions.

Definition

The Gaussian mechanism was first proposed in [131], and is a fundamental building block to differentially private machine learning [1, 39]. Let us first introduce the concepts necessary to define this mechanism formally.

Definition 85 (Gaussian distribution)**.** *The* Gaussian distribution of mean μ and of variance $\sigma^2 > 0$ *is the probability distribution with probability density function:*

$$\text{PDF}(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{x - \mu}{\sigma}\right)^2\right).$$

Using the Gaussian distribution provides benefits besides the utility gains we will discuss shortly. This distribution plays a central in many scientific theories and practices, so most data analysts and engineers are already familiar with it and its basic properties. It also has a much more "predictable" behavior, with tails decaying extremely fast: the probability that a random variable sampled from a Gaussian distribution of mean 0 and variance σ^2 ends up outside $[-5\sigma, 5\sigma]$ is less than one in a million.

Crucially, the Gaussian mechanism depends on a different definition of sensitivity than the Laplace mechanism: the L_2 sensitivity.

Definition 86 (L_2 sensitivity). *Let $f : \mathcal{D} \rightarrow \mathbb{R}^d$ be a deterministic function. The L_2* sensitivity *of f is the smallest number Δ_2 such that for all datasets D_1 and D_2 differing only in one record:*

$$\left\|f\left(D_1\right) - f\left(D_2\right)\right\|_2 \leq \Delta_2$$

where $\|\cdot\|_2$ is the L_2 norm.

We can now introduce a tight analytical result from [28], that quantifies the variance of the Gaussian mechanism necessary and sufficient to obtain differential privacy based on the L_2-sensitivity of a function.

Theorem 16 (Theorem 8 in [28]). *Let $f : \mathcal{D} \rightarrow \mathbb{R}^d$ be a deterministic function with finite L_2 sensitivity Δ_2. The mechanism $\mathcal{M}$ defined by $\mathcal{M}(D) = f(D) + \boldsymbol{X}$, where $\boldsymbol{X}$ is a random vector of $\mathbb{R}^d$ where each coordinate is independently sampled from a Gaussian distribution of mean 0 and with variance σ^2, is (ε, δ)-differentially private iff:*

$$\mathrm{CDF}_{\mathrm{Gaussian}}\left(\frac{\Delta_2}{2\sigma} - \frac{\varepsilon\sigma}{\Delta_2}\right) - e^{\varepsilon}\mathrm{CDF}_{\mathrm{Gaussian}}\left(-\frac{\Delta_2}{2\sigma} - \frac{\varepsilon\sigma}{\Delta_2}\right) \leq \delta$$

where $\mathrm{CDF}_{\mathrm{Gaussian}}$ is the cumulative distribution function of a Gaussian distribution of mean 0 and variance 1.

This formula looks complex, but it is straightforward to implement a binary-search-based algorithm to find out the lowest possible σ that gives (ε, δ)-DP given the L_2 sensitivity Δ_2.

Importantly, this formula is invariant if σ and Δ_2 are scaled by the same number: the standard deviation of the noise required for (ε, δ)-DP is a linear function of Δ_2. This insight is the main reason why Gaussian noise is particularly well-suited to contexts where each user can contribute to many partitions. Indeed, recall that the L_2 norm of a vector $\mathbf{v} = (v_1, \ldots, v_d)$ is defined by:

$$\sqrt{\sum_{i=1}^{d} v_i^2}.$$

Each partition corresponds to a particular dimension of $\mathbb{R}^d$: to compute Δ_2, if a user can contribute to at most k partitions, all but k of these coordinates are going to be 0. If the maximum change to each partition is s, then $\Delta_2 \leq s\sqrt{k}$: the magnitude of the noise scales with the *square root* of the number of partitions contributed to. With typical values of (ε, δ), this makes Gaussian noise a better choice for values of k on the order of 10 or larger.

Implementation

Implementing Gaussian noise properly presents a few challenges. First, naively implementing this continuous probability distribution on finite computers will also introduce vulnerabilities. This fact is well-known for Laplace noise [283], but a quick analysis of common random number generation libraries used in various languages shows that the problem would arise for Gaussian noise as well if implemented naively: to generate 64-bit floating-point numbers, Java uses the Marsaglia polar method [269] with only 53 bits of randomness [299, 300], Python uses the same method [306, 325], Go uses the Ziggurat method, which only uses 32 bits of randomness 99% of the time [305], etc. Thus, the noisy values will necessarily be *sparse* within the space of all possible 64-bit floating-point values, which will create the same vulnerability than the one described in [283]. It is therefore crucial to focus some attention to this problem, and come up with robust implementations that are proven to uphold (ε, δ)-DP guarantees. A handful of solutions have recently been proposed [61, 321, 361], so we assume that this problem can be solved.

Another challenge is that we do not want to have Gaussian noise as the *only* option in our framework: some applications require pure ε-DP, and for low values of k, using Laplace noise is much better for utility, even for relatively large values of δ. So we need to design a system in which we can easily switch one type of noise for another. However, Laplace noise and Gaussian noise use different sensitivities, which makes this non-trivial. With Laplace noise, we could simply use the standard composition theorem to split the privacy budget across different partitions, and consider each partition independently. With Gaussian noise however, we have to have a high-level overview of what each aggregation is doing in each partition, we cannot simply rely on splitting the value of ε in equal parts.

A natural way to solve this design question is to pass the number of partitions that each user contributes to as a parameter to each aggregator, and pass the value of ε associated to the entire aggregation. This way, each aggregator can compute the appropriate value of the noise. Note that yet another subtlety appears here: Theorem 16 assumes that there is a *uniform* bound on the per-partition contribution of individual users, but this is not always the case in practice, for example if we use approximate bounds determination (Section 4.2.5.1) per-partition. Luckily, it is relatively straightforward to extend this result in the case where a user can contribute more in some partitions than others.

Proposition 31 (Generalized analytical Gaussian mechanism). *Let $f : \mathcal{D} \to \mathbb{R}^d$ be a deterministic function such that for all databases D_1 and D_2 differing in a single record, and for all $i \in \{1, 2, \ldots, d\}$:*

$$\left| f_i(D_1) - f_i(D_2) \right| < \Delta_i$$

where $f_i(D)$ denotes the i-th coordinate of $f(D)$. The mechanism $\mathcal{M}$ defined by $\mathcal{M}(D) = f(D) + \boldsymbol{X}$, where $\boldsymbol{X}$ is a random vector of $\mathbb{R}^d$ where the i-th coordinate is independently sampled from a Gaussian distribution of mean 0 and with variance σ_i^2, is (ε, δ)-differentially private if for all i:

$$\mathrm{CDF}_{\mathrm{Gaussian}}\left(\frac{\Delta_i\sqrt{d}}{2\sigma_i} - \frac{\varepsilon\sigma_i}{\Delta_i\sqrt{d}}\right) - e^{\varepsilon}\mathrm{CDF}_{\mathrm{Gaussian}}\left(-\frac{\Delta_i\sqrt{d}}{2\sigma_i} - \frac{\varepsilon\sigma_i}{\Delta_i\sqrt{d}}\right) \leq \delta.$$

Proof. Note that this mechanism is equivalent to computing $f(D)$, dividing each coordinate $f_i(D)$ by Δ_i, adding i.i.d. Gaussian noise according to Theorem 16 with $\Delta_2 = \sqrt{d}$, then multiplying each coordinate by Δ_i.

Once reformulated this way, the result is almost immediate: the first rescaling gives a mechanism of L_2 sensitivity $\sqrt{d}$, the noise addition step guarantees (ε, δ)-differential privacy according to Theorem 16, and the result follows directly by post-processing. □

Finally, note that the core insight behind using Gaussian noise to improve the utility of differentially private queries is to consider the release of multiple metrics *together*, as if they were a single query outputting a single output in $\mathbb{R}^d$, instead of considering them separately. Doing this for a single histogram is natural. An additional possible extension is to make the same reasoning *across entirely different queries*. Consider, for example, the following queries.

```
SELECT
  browser_agent,
  COUNT(DISTINCT uid)
FROM access_logs
GROUP BY browser_agent
```

Listing 4.12: Query counting users, grouped by browser agent

```
SELECT
  hour_of_day,
  COUNT(DISTINCT uid)
FROM access_logs
GROUP BY hour_of_day
```

Listing 4.13: Query counting users, grouped by hour of day

Both queries have a per-partition sensitivity of 1, and we might choose different cross-partition contribution bounds for the two queries (say, κ_1 and κ_2). To make the result of both queries (ε, δ)-differentially private using Gaussian noise, it is natural to split the budget between the two. But for the same reason as before, it is more optimal to consider them as a *single* query, where each user can contribute to $\kappa_1 + \kappa_2$ partitions: this allows us to scale the noise by $\sqrt{\kappa_1 + \kappa_2}$ instead of $\sqrt{\kappa_1} + \sqrt{\kappa_2}$. This leads to a natural trade-off between utility and design best practices: such optimizations require a simultaneous view of all queries and the way each of them is implemented. This goes against typical good practices in design, which require compartmentalization of a system into simple parts.

4.3.4 *Operational aspects of anonymization*

Technical contributions can only get us so far. Building the right tooling is only half the battle: rolling out this tooling in practical scenarios presents an array of *operational* challenges. In this section, we discuss two such challenges, which we encountered during our efforts to scale the number of uses of differential privacy in a large tech company.

The first challenge is about *anonymization parameters*: how to pick values of ε, δ, or even the unit of privacy in a given setting? The second is about *policies*: how to best help engineers understand the guarantees provided by our tooling, and correctly apply differential privacy principles to their pipelines? In both cases, we only provide partial answers to these difficult questions, and these answers are largely unscientific. We still hope that they can be a valuable contribution to a wider conversation between anonymization researchers and practitioners about these challenges.

4.3.4.1 *Setting anonymization parameters*

As we mentioned in Section 2.1, choosing reasonable anonymization parameters for a given use case has always been a tricky problem. Differential privacy gives a clearer semantic view to its parameters than past definitions, allowing to express the meaning of ε using Bayesian inference (Section 2.1.6.2) or hypothesis testing (Section 2.2.6.1). These guarantees are stronger than past attempts at quantifying the risk of e.g. certain choices of k for k-anonymity [137], but do not resolve this issue entirely: choosing parameters remains difficult.

The first question around parameters has to be: what are protecting exactly? Do we assume that each user contributes exactly one record to the dataset, like is generally done in the literature? Do we protect all contributions from each user, like what we argued in Section 4.2? But the question is deeper: what *unit of privacy* should we protect? As the list of variants in Section 2.2.3 shows, there is nothing universal about this choice.

Protecting individuals might be *too weak*: for example, if statistics are collected about families or households, protecting *entire households* might make more sense. Using device identifiers, or account identifiers, can be one way of implementing this goal, for example when everyone in a given household interacts with the same voice assistant device. But protecting individuals can also be *too strong* to give reasonable utility: Facebook's URL dataset [280], for example, protects single *actions* taken on the website (e.g. a single user viewing a single post or sharing a specific URL).

The authors justify this choice by explaining that this allows them to "add significantly less noise compared with [user-level] differential privacy". This policy view is not entirely incompatible with user-level DP: from action-level DP, the authors derive user-level guarantees that depend on the number of actions each user has taken. They guarantee a given user-level ε of 0.45 *for 99% of users*, and derive the parameters for action-level DP accordingly. The 1% of users who contribute more have a lower degree of protection. This two-step approach is certainly worth considering, although it is not without risks: in [280], adding inactive users to the dataset would implicitly make the privacy guarantees worse for everyone, as more users would end up in the 1% of particularly active users.

In other releases, the privacy unit has a time component: Google's Community Mobility Reports [8] and Symptom Search Trends [43] protect users' contributions *in a single day*, while LinkedIn's Audience Engagements API [332] protects the contributions of each user during each month. When releasing new data regularly for an unbounded period of time, having a temporal dimension in the unit of privacy is a requirement; this is sometimes called "renewing the privacy budget" [358]. Doing so means that no hard limits can be given on the total privacy loss for each user throughout the unbounded period.

How risky is such a choice, then? The few practical examples of reconstruction attacks [2, 397] (where the attacker tries to find out full original data records) or membership inference attacks [191, 288, 295, 324, 345] (where the attacker tries to guess whether a data record was part of the original dataset) do not seem to be immediately applicable to such long-lived DP publications. Since correlations in the data are often exploited in attacks [324, 397], maybe the tooling introduced in Chapter 3 can be leveraged to quantify the intuition that some data releases present a sufficiently low level of risk, even without formal guarantees? Bridging the gap between strong formal guarantees, like user-level DP with a small ε, and the feasibility of practical attacks, remains an open question.

Choosing ε is also a challenging question. We found that the Bayesian interpretation of differential privacy (Section 2.1.6.2) can be very useful to help people understand the impact of ε on the level of protection. Concrete examples like randomized response (Section 2.1.6.1) also help. Perhaps provocatively, we argue that when trying to roll out differential privacy at scale, looking for a perfect way to determine ε is not a good use of resources; rather, choosing a default value with a reasonable order of magnitude is a better strategy. Having a fixed default value as the starting point for discussions saves a lot of time and effort: use cases whose utility constraints are compatible with the default value can simply use it, limiting the resources spent on these lower-risk projects. Other uses cases, for which the default value does not seem to work, typically fall into three categories.

First, the use case might be fundamentally incompatible with *any* form of anonymization, e.g. if the data is high-dimensional and cannot be aggregated further. Other options must then be explored: a slightly larger ε will not help.

Second, the anonymization strategy might need to be optimized. For example, there is no need to aggregate metrics at a fine-grained temporal granularity (say, every minute), even though larger granularities (say, hourly or daily metrics) would be enough to solve the original problem. Then, adapting privacy parameters is not the right answer: modifying the aggregation process itself is the correct approach. This can also happen when many aggregations seem to be necessary, but all of them be derived from a smaller subset of aggregations that can anonymized with the default value of ε.

Third, there are projects for which small variations in ε values make a difference in feasibility. In this case, finer-grained analyses are needed, and one can add certain conditions under which the default value can be relaxed, depending on e.g. data sensitivity, exposure, or retention times. Importantly, a reasonable default value makes these discussions *rare*: time is better spent working towards improving tooling, usability, and utility of the vast majority of use cases, rather than discussing parameters for specific projects.

Finally, many practical use cases use (ε,δ)-DP with a non-zero δ. What are reasonable values for this parameter? A common suggestion is to use $\delta = o(1/n)$, where n is the number of users in the dataset: a mechanism that releases a uniformly random record is (ε,δ)-DP for $\varepsilon = 0$ and $\delta = 1/n$, since two datasets that differ in a single record would output this record only with probability $1/n$, and some other record with the exact same probability of $1/n$ each. To avoid classifying this obviously bad mechanism as private, δ should be significantly smaller than $1/n$.

This reasoning, however, only applies to this specific mechanism, which would presumably not be used in practice. Practical (ε,δ)-DP mechanisms do not release a record at random. For example, the δ from Laplace-based partition selection (Section 4.3.2.4) only applies to partitions with a single element in them, and the δ from Gaussian noise (Section 4.3.3.2) only means that the privacy loss can be higher than ε, but not significantly with a large probability. Further, worse mechanisms (like releasing the entire dataset with probability δ) can also be (ε,δ)-DP, but not seem reasonable for *any* non-zero value of δ [274].

Thus, we argue that δ should be chosen *differently* depending on the exact mechanism. Vanishingly low values (say, 10^{-30}) are necessary for some rare use cases so a catastrophic event is guaranteed to never happen in practice, but larger values make sense in other scenarios: Facebook's URL dataset, for example, uses 10^{-5} for the δ associated with Gaussian noise [280]. The δ could also conceivably depend on the data itself: for example, for Laplace-based partition selection, since the risk of the δ only depends on the number of partitions with a single user, we could first estimate the number of such partitions, and select the δ accordingly.

This suggests a way to simplify the interface of differentially private tooling, by only surfacing ε to tool users, and have δ be chosen automatically. A default can be chosen for some algorithms, while for others, more information could be asked of the analyst (e.g. an order of magnitude of the number of partitions with a single user), or determined automatically in a data-dependent way, possibly using a small fraction of the ε budget.

4.3.4.2 *Attack models and policies*

Rolling out strong anonymization practices to a large number of use cases takes more than building tooling and picking reasonable parameter values. It is also essential to provide education and operational guidance to users of this tooling: even when the interface is simplified as much as possible, differential privacy tooling is still easy to apply incorrectly. In this section, we focus on a particular use case: long-lived anonymization pipelines, in the trusted curator model.

Note that this scenario is different from what is assumed in much of the DP literature, where an analyst is given differentially private query access to a private database. Relying solely on DP for this last use case presents immense challenges: such a system must track privacy budget and rate-limit queries, possibly split the budget across analysts, handle of query errors in a way that does not reveal information but still allows well-meaning analysts to debug queries, or prevent side-channel attacks [16, 181]. LinkedIn's Audience Engagements API [332] is perhaps the only public system actually operating under this model; it uses very large privacy budgets and a fairly restricted class of allowed queries.

Instead, we focus on a scenario in which analysts have some level of access to the original, raw data (presumably, with a number of security measures in place) and are trying to publish or share some anonymized statistics over this data. This is the model used in data releases such as the 2020 US Census [3] or Facebook's URL dataset [280]. In addition, we assume that this release is long-lived: the raw dataset grows over time, and new data is regularly published; this is the case for e.g. Google's Community Mobility Reports [8] or Symptom Search Trends dataset [43].

In this scenario, the people running differentially private algorithms write one or several non-adversarial queries, make sure that they provide differential privacy guarantees with a fixed privacy budget, and release the output of those queries. Mistakes can still happen: for example, floating-point attacks from [283] can cause unexpected privacy loss even without ill intentions from the person running the tool. Perhaps more importantly, honest mistakes can *also* weaken or break privacy guarantees: users must be supported with education and operational guidance to make sure they understand how to use the tooling correctly.

Let us take a simple example. If a user repetitively runs a differentially private pipeline, and no budget tracking mechanism is implemented across runs, the total privacy loss is unbounded, and so differential privacy no longer provides any formal guarantee. Preventing even simple mistakes like these from happening is more complicated than it appears. Tracking privacy budget in the tooling itself, and setting a cap at how much one can query the data, causes significant usability issues for legitimate users. Analysts trying to define which anonymization strategy to use for a specific data release often need a lot of experimentation, incompatible with such hard limits.

Instead, analysts must understand the concept of a privacy budget: the collection of many individual outputs of anonymization pipelines is "less anonymized" than each in isolation. The data we publish cannot be easily recomputed and re-published: otherwise, the initial privacy budget computations will need to be adjusted. This is far from natural to non-experts. No other privacy or security property works this way: the union of encrypted data is still encrypted, adding data to an access-controlled directory does not change its access level, etc.

This underscores the need for policies that specifically address data generated for experimental purposes. Although such data might technically qualify as anonymized, good practices should still be followed, such as limiting access controls lists to those who need access and setting short retention periods.

This aspect also has important implications on planning, testing, validation, and anomaly detection practices: finding out *before publication* that some data is erroneous is crucial, and since manual validation does not scale, this requires significant investment into automated testing. Further, it is also a good practice to plan for extra privacy budget in case something does need to be recomputed: technical changes, like underlying inference algorithms being improved over time, can unexpectedly impact the utility of the published metrics, and testing is always imperfect. Techniques like *scaling factors* [8] can help reduce the privacy cost of such events, but not avoid it entirely.

Finally, the need to experiment and to validate the output data also requires a somewhat relaxed view of what goes in the privacy budget. The very idea of doing experiments on the raw data to develop a final anonymization strategy might sound shocking to theorists! With a extremely principled approach to differential privacy, *every* decision we make based on the data will influence what eventually gets published (and when!), and should be counted as part of the privacy budget. So if multiple anonymization strategies are compared, and one ends up being chosen, all of them should count towards the privacy budget, or the choice itself should be done in a differentially private manner. Similarly, since the result of automated testing ends up being visible to a potential attacker (who could notice if data is held back), testing should itself be differentially private, and count towards the budget.

We argue that such a view is entirely incompatible with practical requirements. Nobody will agree to publishing noisy data without first making sure that the result is usable, which typically requires many rounds of trial-and-error. Good testing involves not relying on any single piece of infrastructure or logic to validate the data: testing the accuracy of differentially private data by using noisy data, when raw data is available, is nonsensical to data analysts who require strong guarantees on data quality.

Simply not counting this towards the privacy budget feels somewhat uncomfortable. As DP mechanisms get more and more complex to optimize privacy/accuracy trade-offs, they also get more hyperparameters, which can be fine-tuned to maximize the mechanism utility on certain data. Running a very large number of simulations to find the best value for these parameters seems unwise: the hyperparameter values themselves might leak private data…

One compromise is to count *automatic* tuning of parameters towards privacy budget calculations, while *manual* experimentation should not. For example, the data-dependent method for choosing δ introduced at the end of Section 4.3.4.1 would not be counted towards the total privacy budget if the user manually provides an order of magnitude of the number of small partitions in the data, but would consume some ε if this determination is done automatically (and, presumably, more precisely).

In the case of testing, this criterion does not exactly work: automated testing is necessary for long-lived anonymization pipelines, and as we mentioned before, cannot realistically be done in a DP manner. How to avoid inadvertent information leakage? One mitigation strategy is to ensure that the effects of testing are *coarse* enough to not reveal significant insights. For example, publishing all metrics that pass some quality tests and holding back others seems unwise. However, having some global tests and preventing the publication of *all* metrics when those tests fail is, intuitively, very unlikely to cause any privacy issue.

These considerations must be implemented in education and guidance. This requires significant investment: one cannot simply build the right tooling and consider the problem solved. This should not be surprising: in any technological system providing security or privacy properties, honest mistakes are much more likely to happen than ill-intentioned attacks, and relying on technology alone often fails to provide the desired outcomes.

4.3.5 *Other possible research directions*

The discussion throughout Section 4.3 makes it obvious that we are only at the very beginning of the path towards building usable, general-purpose, and robust differential privacy tooling and infrastructure. We introduced many open problems, most of which were about adding additional features, or improving the utility of specific primitives. But this work also suggests a number of larger open questions and possible research directions.

For example, on the utility side, it might be possible to compensate the data loss due to contribution bounding and thresholding: we are dropping records according to a simple probability distribution, so it might be feasible to automatically correct for this sampling and rescale the results in a private way to limit the bias introduced. Optimizing the algorithms used for specific sets of queries, or caching some results to allow people to re-run queries for free, are also natural options. Results on amplification by sampling might also be used in query frameworks like ours to obtain better privacy/accuracy trade-offs.

More generally, we believe that future work on DP should consider that realistic data typically includes multiple contributions from a single user. Taking this into account is not easy: in particular, one of the main obstacles is that *asymptotic* results are then likely much more difficult to derive. If every user can contribute multiple records, what is the distribution of the number of records per user? Depending on this distribution, different algorithms may have very different behaviors and privacy/accuracy trade-offs. Comparisons with the state-of-the-art will necessarily be less straightforward, and will involve more experimental validation. Considering this question while designing differentially private algorithms is, however, of utmost importance: many real-world datasets simply do not satisfy the assumption that the input dataset only has a single record per user.

This remark about experimental validation underscores the importance of a crucial and largely unexplored area of differential privacy research: benchmarking. Nowadays, scientific papers often provide comparisons with prior work on ad hoc datasets or on synthetic data. The evaluation we presented in Section 4.2.4 is no exception. TPC-H was designed as a general-purpose performance benchmark for database systems [87]; its queries are not representative of the type of queries that are typical in situations where applying differential privacy makes sense. There are a few benchmarks today that focus on differential privacy, but they only consider specific problems like one- and two-dimensional histograms [189, 190], and they do not consider multiple contributions by the same user. We argue that a good benchmark for general-purpose differentially private query engines should have several important characteristics.

- The datasets used must be publicly available.
- The data and the queries run must be representative of real-world use cases for differentially private query engines, including aggregations besides histograms.
- The datasets and queries used must be large enough to mirror practical scalability requirements.
- The system under test must provide user-level differential privacy, even when a single user can contribute multiple records.

Building a benchmarking system that satisfies these requirements will not be easy: one of the challenges is that differential privacy is typically applied on *sensitive* data, which is typically not public. Besides the requirements listed above, defining a set of good scoring metrics is also going to be highly non-trivial. However, we believe that such a benchmark could encourage valuable research, making it a promising and worthwhile research direction.

We made a number of statements about improving *usability*, but we have been defining this concept very loosely, and our remarks where purely based on practical experience with helping engineers and data analysts use differential privacy. This anecdotal evidence, however, hardly constitutes scientific evidence that one method is better than another from a usability perspective. To get a better understanding of what helps people use differential privacy in practice, systematic qualitative and quantitative user research is certainly needed.

Finally, we hope that the discussion on operational challenges can serve as a starting point for honest and genuine discussions around best operational practices within the community of anonymization practitioners. However, reaching consensus on these questions will take more than technical expertise: questions like “how to explain anonymization best practices to non-experts” cannot be answered by a clever mathematical formula, and a multidisciplinary approach is needed. Academic work can strengthen our understanding of the link between parameter choice and attack success, or take an experimental approach to find out which policies produce the best outcomes, but an open and transparent discussion across the industry is also needed to define anonymization good practices. Perhaps standard bodies, or open-source collaborations, could be good places to host such discussions?

5

CONCLUSION

From the ground, we stand. From our ships, we live. By the stars, we hope.

— Becky Chambers, *Records of a Spaceborn Few*

Trying to lower the cost of anonymization, in 2020, can sometimes feel like a pointless academic endeavor. Is data privacy really worth focusing on, when climate change is quickly making our planet unlivable [362]? Only a global pandemic has seemed able to temporarily slow down the global warming process [239], while existing plans to tackle this enormous challenge seem both inefficient and doomed to failure [240]. Not only do we appear to be unable to mitigate this looming threat, but we also seem woefully unprepared to handle its consequences. Migration crises will likely intensify over the next few decades, and nations all over the world are already showing their systematic inability to handle them with any shred of humanity or kindness [175, 229, 343]. We urgently need to build more just societies, that acknowledge their white supremacist past, and uphold fundamental human rights for everyone, including marginalized communities. Instead, world superpowers are quickly descending into fascism [386], building technological infrastructure that empowers belligerent authoritarianism [319, 377], and getting away with committing cultural genocide [403].

Fighting off these potentially catastrophic shifts is critical to upholding every value we hold dear. But preventing our societies from succumbing to apocalyptic threats is only half the battle: we also need to actively build a fairer, kinder, more sustainable world. Systemic challenges require systemic solutions: we need to change incentives and structures so that people and organizations naturally tend towards doing the right thing for all the stakeholders they impact.

The development of privacy-preserving technologies can play a positive role in pushing some of those incentives in the right direction. As an example, encrypting Web traffic at scale is globally beneficial, and efforts that make secure protocols easier to use and deploy are making it that comparatively harder for organizations to resist calls to migrate to safer practices. Similarly we envision a world where there are fewer and fewer reasons for sharing or publishing data in an unsafe, identifiable manner, which puts individuals at risk. Such a shift would primarily benefit those who, today, pay the highest price when their personal information is leaked: marginalized groups targeted by online harassment, victims of domestic abuse, or political dissidents under authoritarian regimes.

Nonetheless, it is worth examining the implicit power dynamics at play in differential privacy, especially in its central form discussed throughout this thesis, as almost all the techniques we introduced, developed, or improved upon, assume that a central organization collects and stores raw data from individuals. Once all this personal and sensitive information has been collected, this central organization the one responsible for anonymizing it and then sharing or publishing it in a safe manner. This is a vertical model of differential privacy: it presupposes a high level of trust that the individual must put in the central organization, to take great care of their data and not abuse it. This trust might be completely misguided, as many recent high-profile data leaks or questionable uses of data tend to suggest. Further, this relationship is often not as consensual as it first seems: for instance, can you really choose not to participate to a service that all your friends use to organize social events? Finally, even organizations with good intent are subject to applicable laws and regulations, some of which can compel them to share the data from their users with authoritarian governments.

This model will likely never disappear entirely: retailers need to know who their customers are and where to ship their packages, hospitals need to have information about the patients they treat, government organizations like the US Census have a mandate to collect demographic data from their residents and use it to make decisions that impact their democratic system. Further, as we mentioned in the introduction of this thesis, and as exemplified by use cases like the Community Mobility Reports or the Search Trends symptoms dataset, such data publications can have a significantly positive social impact. Thus, we argue that improving incentives for such use cases is still worth doing.

However, these implicit power dynamics, which indirectly cause massive systemic problems, suggest that lowering the cost of central differential privacy is not enough. To minimize the negative societal impact from data hoarding, we need to incentivize organizations not to collect sensitive data in the first place. This suggests investing in alternative models, like local or distributed differential privacy. In these models, a central aggregator can learn useful information about global trends, but does not have access to the raw data: they cannot abuse it, leak it inadvertently, or provide it to governments.

Randomized response was a first example of such a system, but a number of very promising advances have been made in that direction in the past few years. Trusted shufflers or secure aggregation techniques can provide strong guarantees and high utility with a minimal level of trust. Just like any other privacy-enhancing technology, making them both useful and usable will be paramount to their success, and will unlock their positive impact globally.

We predict that rolling out these protocols at scale will raise very similar issues to the ones discussed in this thesis about central differential privacy: improving usability will be difficult but paramount, unexpected vulnerabilities will have to be identified and fixed, tricky operational challenges will appear and will need to be tackled. Collaborations across the industry, governments, nonprofit organizations and academics will be necessary for such a project to be successful. It will be difficult, frustrating, tiring, fascinating, and exhilarating—anything worth doing usually is.

BIBLIOGRAPHY

1. Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K. & Zhang, L. *Deep learning with differential privacy*. Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (2016). (Cit. on pp. 43, 230).
2. Abowd, J. M. *Staring-Down the Database Reconstruction Theorem*. Joint Statistical Meetings, Vancouver, BC (2018) (cit. on p. 234).
3. Abowd, J. M. *The US Census Bureau adopts differential privacy*. Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (2018), p. 2867 (cit. on p. 236).
4. Abowd, J. M., Andersson, F., Graham, M., Vilhuber, L. & Wu, J. *Formal Privacy Guarantees and Analytical Validity of OnTheMap Public-Use Data*. 2010. `https://ecommons.cornell.edu/handle/1813/47672` (cit. on pp. 30, 157).
5. Abowd, J. M., Schneider, M. J. & Vilhuber, L. *Differential privacy applications to Bayesian and linear mixed model estimation*. Journal of Privacy and Confidentiality (2013) (cit. on pp. 58, 81).
6. Acharya, J., Bonawitz, K., Kairouz, P., Ramage, D. & Sun, Z. *Context Aware Local Differential Privacy*. ICML 2020: 37th International Conference on Machine Learning (2020) (cit. on p. 82).
7. Aghasian, E., Garg, S. & Montgomery, J. *User's Privacy in Recommendation Systems Applying Online Social Network Data, A Survey and Taxonomy*. arXiv preprint arXiv:1806.07629 (2018) (cit. on p. 82).
8. Aktay, A., Bavadekar, S., Cossoul, G., Davis, J., Desfontaines, D., Fabrikant, A., Gabrilovich, E., Gadepalli, K., Gipson, B., Guevara, M., *et al. Google COVID-19 community mobility reports: Anonymization process description (version 1.0)*. arXiv preprint arXiv:2004.04145 (2020) (cit. on pp. 234, 236 sq.).
9. Alaggan, M., Cunche, M. & Gambs, S. *Privacy-preserving Wi-Fi analytics*. Proceedings on Privacy Enhancing Technologies vol. 2018(2), p. 4 (2018) (cit. on pp. 128, 215).
10. Alaggan, M., Gambs, S. & Kermarrec, A.-M. *Heterogeneous Differential Privacy*. Journal of Privacy and Confidentiality vol. 7(2), p. 6 (2017) (cit. on pp. 52, 71).
11. Allen, J., Ding, B., Kulkarni, J., Nori, H., Ohrimenko, O. & Yekhanin, S. *An algorithmic framework for differentially private data analysis on trusted processors*. Advances in Neural Information Processing Systems (2019) (cit. on p. 49).
12. Alvim, M., Chatzikokolakis, K., Palamidessi, C. & Pazii, A. *Local differential privacy on metric spaces: optimizing the trade-off with utility*. 2018 IEEE 31st Computer Security Foundations Symposium (CSF) (2018) (cit. on p. 82).

13. Amin, K., Kulesza, A., Munoz, A. & Vassilvtiskii, S. *Bounding User Contributions: A Bias-Variance Trade-off in Differential Privacy*. Proceedings of the 36th International Conference on Machine Learning, PMLR 97 (2019), p. 263 (cit. on p. 198).

14. Anderson, N. *"Anonymized" data really isn't—and here's why not*. Ars Technica. (2009) (cit. on p. 9).

15. Andrés, M. E., Bordenabe, N. E., Chatzikokolakis, K. & Palamidessi, C. *Geo-indistinguishability: Differential privacy for location-based systems*. Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security (2013) (cit. on pp. 53, 81).

16. Andrysco, M., Kohlbrenner, D., Mowery, K., Jhala, R., Lerner, S. & Shacham, H. *On subnormal floating point and abnormal timing*. 2015 IEEE Symposium on Security and Privacy (2015), p. 623 (cit. on p. 236).

17. *Apache Beam*. `https://beam.apache.org/` (Accessed: 2020-09-18) (cit. on p. 226).

18. *Approximate algorithms in Apache Spark: HyperLogLog and Quantiles*. `https://databricks.com/blog/2016/05/19/approximate-algorithms-in-apache-spark-hyperloglog-and-quantiles.html` (Accessed: 2020-08-14) (cit. on p. 127).

19. Arram, M. *Case Study: Differential Privacy Innovation at Bluecore*. `https://georgianpartners.com/differential-privacy-innovation-at-bluecore/` (Accessed: 2020-08-18) (cit. on p. 157).

20. *ARX Data Anonymization Tool*. `https://arx.deidentifier.org/` (Accessed: 2020-10-05) (cit. on p. 13).

21. Ashok, V. G. & Mukkamala, R. *A scalable and efficient privacy preserving global itemset support approximation using Bloom filters*. IFIP Annual Conference on Data and Applications Security and Privacy (2014) (cit. on pp. 127 sq.).

22. Asi, H., Duchi, J. & Javidbakht, O. *Element Level Differential Privacy: The Right Granularity of Privacy*. arXiv preprint arXiv:1912.04042 (2019) (cit. on pp. 46, 70).

23. Asif, H., Papakonstantinou, P. A. & Vaidya, J. *How to Accurately and Privately Identify Anomalies*. Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security (2019) (cit. on pp. 47, 70).

24. Backes, M., Kate, A., Meiser, S. & Ruffing, T. *Differential Indistinguishability for Cryptography with (Bounded) Weak Sources*. Grande Region Security and Reliability Day (GRSRD) (2014) (cit. on p. 66).

25. Balle, B., Barthe, G., Gaboardi, M. & Geumlek, J. *Privacy amplification by mixing and diffusion mechanisms*. Advances in Neural Information Processing Systems (2019), p. 13298 (cit. on p. 125).

26. Balle, B., Barthe, G., Gaboardi, M., Hsu, J. & Sato, T. *Hypothesis Testing Interpretations and Renyi Differential Privacy.* AISTATS (2019), p. 2496 (cit. on p. 43).

27. Balle, B., Bell, J., Gascón, A. & Nissim, K. *The Privacy Blanket of the Shuffle Model*. Annual International Cryptology Conference (2019), p. 638 (cit. on pp. 107 sq.).

28. Balle, B. & Wang, Y.-X. *Improving the Gaussian Mechanism for Differential Privacy: Analytical Calibration and Optimal Denoising*. International Conference on Machine Learning (2018), p. 394 (cit. on p. 231).

29. Bar-Yossef, Z., Jayram, T., Kumar, R., Sivakumar, D. & Trevisan, L. *Counting distinct elements in a data stream*. International Workshop on Randomization and Approximation Techniques in Computer Science (2002) (cit. on p. 131).

30. Barak, B., Chaudhuri, K., Dwork, C., Kale, S., McSherry, F. & Talwar, K. *Privacy, accuracy, and consistency too: a holistic solution to contingency table release*. Proceedings of the twenty-sixth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems (2007), p. 273 (cit. on p. 29).

31. Barber, R. F. & Duchi, J. C. *Privacy and statistical risk: Formalisms and minimax bounds*. arXiv preprint arXiv:1412.4451 (2014) (cit. on pp. 42–45, 54, 69 sq., 72, 78, 81).

32. Barthe, G., Espitau, T., Grégoire, B., Hsu, J., Stefanesco, L. & Strub, P.-Y. *Relational reasoning via probabilistic coupling*. Logic for Programming, Artificial Intelligence, and Reasoning (2015), p. 387 (cit. on p. 181).

33. Barthe, G., Gaboardi, M., Hsu, J. & Pierce, B. *Programming language techniques for differential privacy*. ACM SIGLOG News vol. 3(1), p. 34 (2016) (cit. on p. 181).

34. *BASE – Bielefeld Academic Search Engine*. `https://www.base-search.net/` (Accessed: 2020-10-07) (cit. on p. 40).

35. Bassily, R. & Freund, Y. *Typical stability*. arXiv preprint arXiv:1604.03336 (2016) (cit. on pp. 65 sq., 75 sq., 79).

36. Bassily, R., Groce, A., Katz, J. & Smith, A. *Coupled-worlds privacy: Exploiting adversarial uncertainty in statistical data privacy*. Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on (2013) (cit. on pp. 57 sq., 65, 73 sq., 76, 87, 89–92, 98 sqq., 129, 137).

37. Bassily, R., Nissim, K., Smith, A., Steinke, T., Stemmer, U. & Ullman, J. *Algorithmic stability for adaptive data analysis*. Proceedings of the forty-eighth annual ACM symposium on Theory of Computing (2016) (cit. on pp. 41, 69).

38. Bassily, R. & Smith, A. *Local, private, efficient protocols for succinct histograms*. Proceedings of the forty-seventh annual ACM symposium on Theory of computing (2015), p. 127 (cit. on p. 129).

39. Bassily, R., Smith, A. & Thakurta, A. *Private empirical risk minimization: Efficient algorithms and tight error bounds*. 2014 IEEE 55th Annual Symposium on Foundations of Computer Science (2014), p. 464 (cit. on p. 230).

40. Bassily, R., Stemmer, U., Thakurta, A. G., *et al. Practical locally private heavy hitters*. Advances in Neural Information Processing Systems (2017), p. 2288 (cit. on p. 129).

41. Basu, D., Dimitrakakis, C. & Tossou, A. *Differential Privacy for Multi-armed Bandits: What Is It and What Is Its Cost?* arXiv preprint arXiv:1905.12298 (2019) (cit. on p. 50).

42. Bater, J., He, X., Ehrich, W., Machanavajjhala, A. & Rogers, J. *Shrinkwrap: efficient SQL query processing in differentially private data federations*. Proceedings of the VLDB Endowment (2018) (cit. on pp. 67, 180 sq.).

43. Bavadekar, S., Dai, A., Davis, J., Desfontaines, D., Eckstein, I., Everett, K., Fabrikant, A., Flores, G., Gabrilovich, E., Gadepalli, K., Glass, S., Huang, R., Kamath, C., Kraft, D., Kumok, A., Marfatia, H., Mayer, Y., Miller, B., Pearce, A., Perera, I. M., Ramachandran, V., Raman, K., Roessler, T., Shafran, I., Shekel, T., Stanton, C., Stimes, J., Sun, M., Wellenius, G. & Zoghi, M. *Google COVID-19 Search Trends Symptoms Dataset: Anonymization Process Description (version 1.0)*. arXiv preprint arXiv:2009.01265 (2020) (cit. on pp. 234, 236).

44. Ben-Or, M. & Hassidim, A. *The Bayesian learner is optimal for noisy binary search (and pretty good for quantum as well)*. 49th Annual IEEE Symposium on Foundations of Computer Science (2008), p. 221 (cit. on pp. 189, 214).

45. Beyer, K., Haas, P. J., Reinwald, B., Sismanis, Y. & Gemulla, R. *On synopses for distinct-value estimation under multiset operations*. Proceedings of the 2007 ACM SIGMOD international conference on Management of data (2007) (cit. on pp. 131, 164 sq., 178).

46. Bhaskar, R., Bhowmick, A., Goyal, V., Laxman, S. & Thakurta, A. *Noiseless database privacy*. International Conference on the Theory and Application of Cryptology and Information Security (2011) (cit. on pp. 56, 58, 61 sq., 73 sq., 80 sq., 87 sq., 105 sq., 110, 129, 137).

47. Bichsel, B., Gehr, T., Drachsler-Cohen, D., Tsankov, P. & Vechev, M. *DP-finder: Finding differential privacy violations by sampling and optimization*. Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (2018), p. 508 (cit. on p. 181).

48. *BigQuery Technical Documentation on Functions & Operators*. `https://cloud.google.com/bigquery/docs/reference/standard-sql/functions-and-operators` (Accessed: 2020-08-18) (cit. on pp. 127, 151, 154).

49. Bindschaedler, V., Shokri, R. & Gunter, C. A. *Plausible deniability for privacy-preserving data synthesis*. Proceedings of the VLDB Endowment vol. 10(5), p. 481 (2017) (cit. on p. 181).

50. Bittau, A., Erlingsson, Ú., Maniatis, P., Mironov, I., Raghunathan, A., Lie, D., Rudominer, M., Kode, U., Tinnes, J. & Seefeld, B. *Prochlo: Strong privacy for analytics in the crowd*. Proceedings of the 26th Symposium on Operating Systems Principles (2017) (cit. on p. 82).

51. Bittner, D. M., Sarwate, A. D. & Wright, R. N. *Using Noisy Binary Search for Differentially Private Anomaly Detection*. International Symposium on Cyber Security Cryptography and Machine Learning (2018) (cit. on pp. 47, 70).

52. Blocki, J., Blum, A., Datta, A. & Sheffet, O. *Differentially private data analysis of social networks via restricted sensitivity*. Proceedings of the 4th conference on Innovations in Theoretical Computer Science (2013) (cit. on p. 81).

53. Blum, A., Ligett, K. & Roth, A. *A learning theory approach to noninteractive database privacy*. Journal of the ACM (JACM) (2013) (cit. on pp. 51, 55, 66, 72).

54. Brickell, J. & Shmatikov, V. *The cost of privacy: destruction of data-mining utility in anonymized data publishing*. Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining (2008), p. 70 (cit. on p. 18).

55. Broder, A. *On the Resemblance and Containment of Documents*. Proceedings of the Compression and Complexity of Sequences 1997 (1997), p. 21. (Cit. on pp. 160, 178).

56. Bun, M., Dwork, C., Rothblum, G. N. & Steinke, T. *Composable and versatile privacy via truncated CDP*. Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing (2018) (cit. on pp. 44, 69).

57. Bun, M. & Steinke, T. *Concentrated differential privacy: Simplifications, extensions, and lower bounds*. Theory of Cryptography Conference (2016) (cit. on pp. 30, 43 sq., 69, 76, 78).

58. Burchard, P. & Daoud, A. *Empirical Differential Privacy*. arXiv preprint arXiv:1910.12820 (2019) (cit. on pp. 58, 81).

59. Campbell-Dollaghan, K. *Sorry, your data can still be identified even if it's anonymized*. Fast Company. (2018) (cit. on p. 7).

60. Canard, S. & Olivier, B. *Differential Privacy in distribution and instance-based noise mechanisms.* IACR Cryptology ePrint Archive (2015) (cit. on pp. 41, 69, 78).

61. Canonne, C., Kamath, G. & Steinke, T. *The Discrete Gaussian for Differential Privacy*. arXiv preprint arXiv:2004.00010 (2020) (cit. on p. 232).

62. Casella, G. & Berger, R. L. *Statistical inference* (Duxbury Pacific Grove, CA, 2002) (cit. on p. 139).

63. Chan, T. H., Chung, K.-M., Maggs, B. M. & Shi, E. *Foundations of differentially oblivious algorithms*. Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms (2019) (cit. on p. 49).

64. Charest, A.-S. & Hou, Y. *On the meaning and limits of empirical differential privacy*. Journal of Privacy and Confidentiality (2016) (cit. on pp. 48, 71, 81).

65. Charikar, M., Chen, K. & Farach-Colton, M. *Finding frequent items in data streams*. International Colloquium on Automata, Languages, and Programming (2002), p. 693 (cit. on p. 164).

66. Charikar, M. S. *Similarity estimation techniques from rounding algorithms*. Proceedings of the thiry-fourth annual ACM symposium on Theory of computing (2002), p. 380 (cit. on p. 178).

67. Chatzikokolakis, K., Andrés, M. E., Bordenabe, N. E. & Palamidessi, C. *Broadening the scope of differential privacy using metrics*. International Symposium on Privacy Enhancing Technologies Symposium (2013) (cit. on pp. 54, 72, 81).

68. Chatzikokolakis, K., ElSalamouny, E., Palamidessi, C., Anna, P., *et al. Methods for Location Privacy: A comparative overview*. Foundations and Trends® in Privacy and Security (2017) (cit. on pp. 56, 83).

69. Chaudhuri, K., Imola, J. & Machanavajjhala, A. *Capacity bounded differential privacy*. Advances in Neural Information Processing Systems (2019) (cit. on pp. 45, 70).

70. Chaudhuri, K. & Mishra, N. *When random sampling preserves privacy*. Annual International Cryptology Conference (2006) (cit. on p. 25).

71. Chaudhuri, K., Monteleoni, C. & Sarwate, A. D. *Differentially private empirical risk minimization.* Journal of Machine Learning Research vol. 12(3) (2011) (cit. on p. 179).

72. Chen, L., Ghazi, B., Kumar, R. & Manurangsi, P. *On Distributed Differential Privacy and Counting Distinct Elements*. arXiv preprint arXiv:2009.09604 (2020) (cit. on p. 128).

73. Chen, R., Fung, B. C., Yu, P. S. & Desai, B. C. *Correlated network data publication via differential privacy*. The VLDB Journal—The International Journal on Very Large Data Bases (2014) (cit. on pp. 47, 70, 96).

74. Chen, S. & Zhou, S. *Recursive mechanism: towards node differential privacy and unrestricted joins*. Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data (2013) (cit. on p. 81).

75. Chen, Z., Bao, X., Ying, Z., Liu, X. & Zhong, H. *Differentially Private Location Protection with Continuous Time Stamps for VANETs*. International Conference on Algorithms and Architectures for Parallel Processing (2018) (cit. on pp. 48, 54).

76. Chia, P. H., Desfontaines, D., Perera, I. M., Simmons-Marengo, D., Li, C., Day, W.-Y., Wang, Q. & Guevara, M. *KHyperLogLog: Estimating Reidentifiability and Joinability of Large Data at Scale*. 2019 IEEE Symposium on Security and Privacy (SP) (2019), p. 350 (cit. on p. 5).

77. Choi, S. G., Dachman-Soled, D., Kulkarni, M. & Yerukhimovich, A. *Differentially-Private Multi-Party Sketching for Large-Scale Statistics*. Proceedings on Privacy Enhancing Technologies vol. 3, p. 153 (2020) (cit. on pp. 128, 215).

78. Clifton, C. & Tassa, T. *On syntactic anonymity and differential privacy*. 2013 IEEE 29th International Conference on Data Engineering Workshops (ICDEW) (2013) (cit. on p. 82).

79. Cohen, A., Nikolov, A., Schutzman, Z. & Ullman, J. *Reconstruction Attacks in Practice*. DifferentialPrivacy.org. `https://differentialprivacy.org/diffix-attack/`. 2020 (cit. on p. 182).

80. Cohen, A. & Nissim, K. *Linear Program Reconstruction in Practice*. Journal of Privacy and Confidentiality vol. 10(1) (2020) (cit. on p. 182).

81. Colisson, L. *L3 Internship report: Quantum analog of Differential Privacy in term of Rényi divergence.* (2016) (cit. on pp. 43 sq., 69, 78).

82. *Computing k-map estimates with Cloud DLP*. `https://cloud.google.com/dlp/docs/compute-risk-analysis%5C#compute-k-map` (Accessed: 2020-10-05) (cit. on p. 16).

83. *Computing Private Statistics with Privacy on Beam*. `https://codelabs.developers.google.com/codelabs/privacy-on-beam/` (Accessed: 2020-09-23) (cit. on p. 227).

84. Cormode, G. & Hadjieleftheriou, M. *Finding frequent items in data streams*. Proceedings of the VLDB Endowment vol. 1(2), p. 1530 (2008) (cit. on p. 164).

85. Cormode, G. & Muthukrishnan, S. *An improved data stream summary: the count-min sketch and its applications*. Journal of Algorithms vol. 55(1), p. 58 (2005) (cit. on p. 178).

86. Cormode, G. & Muthukrishnan, S. *Space efficient mining of multigraph streams*. Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems (2005), p. 271 (cit. on p. 178).

87. Council, T. P. P. *TPC-H benchmark specification*. `http://www.tpc.org/tpch/`. 2008 (cit. on pp. 197, 239).

88. Cuff, P. & Yu, L. *Differential privacy as a mutual information constraint*. Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (2016) (cit. on pp. 39, 42 sq., 69, 78).

89. Cummings, R. & Durfee, D. *Individual sensitivity preprocessing for data privacy*. Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms (2020) (cit. on pp. 52, 81).

90. Dai Nguyen, T., Gupta, S., Rana, S. & Venkatesh, S. *A Privacy Preserving Bayesian Optimization with High Efficiency*. Pacific-Asia Conference on Knowledge Discovery and Data Mining (2018) (cit. on p. 81).

91. Dalenius, T. *Towards a methodology for statistical disclosure control*. statistik Tidskrift (1977) (cit. on p. 48).

92. Dandekar, A., Basu, D. & Bressan, S. *Differential Privacy at Risk: Bridging Randomness and Privacy Budget.* Proceedings on Privacy Enhancing Technologies, p. 1 (2020) (cit. on pp. 55, 72).

93. De Montjoye, Y.-A., Hidalgo, C. A., Verleysen, M. & Blondel, V. D. *Unique in the crowd: The privacy bounds of human mobility*. Scientific reports vol. 3, p. 1376 (2013) (cit. on p. 7).

94. De Montjoye, Y.-A., Radaelli, L., Singh, V. K., *et al. Unique in the shopping mall: On the reidentifiability of credit card metadata*. Science vol. 347(6221), p. 536 (2015) (cit. on p. 7).

95. Dean, J. & Ghemawat, S. *MapReduce: a flexible data processing tool*. Communications of the ACM vol. 53(1), p. 72 (2010) (cit. on pp. 127, 173, 226).

96. *Decennial Census of Population and Housing*. 2010. `https://factfinder.census.gov` (cit. on p. 174).

97. Deldar, F. & Abadi, M. *PLDP-TD: Personalized-location differentially private data analysis on trajectory databases*. Pervasive and Mobile Computing (2018) (cit. on p. 52).

98. Desfontaines, D. *Personal blog*. `https://desfontain.es/privacy` (Accessed: 2020-09-22) (cit. on p. 4).

99. Desfontaines, D., Lochbihler, A. & Basin, D. *Cardinality estimators do not preserve privacy*. Proceedings on Privacy Enhancing Technologies vol. 2019(2), p. 26 (2019) (cit. on p. 4).

100. Desfontaines, D., Mohammadi, E., Krahmer, E. & Basin, D. *Differential privacy with partial knowledge*. arXiv preprint arXiv:1905.00650 (2019) (cit. on pp. 4, 73 sq., 76).

101. Desfontaines, D. & Pejó, B. *Differential Privacies: a taxonomy of differential privacy variants and extensions (long version)*. arXiv prpeprint arXiv:1906.01337 (2019) (cit. on p. 4).

102. Desfontaines, D. & Pejó, B. *SoK: Differential Privacies*. Proceedings on Privacy Enhancing Technologies vol. 2020(2) (2020) (cit. on p. 4).

103. Diakonikolas, I., Gouleakis, T., Peebles, J. & Price, E. *Collision-based Testers are Optimal for Uniformity and Closeness.* Electronic Colloquium on Computational Complexity vol. 23, p. 178 (2016) (cit. on p. 212).

104. Diaz, C., Seys, S., Claessens, J. & Preneel, B. *Towards measuring anonymity*. International Workshop on Privacy Enhancing Technologies (2002), p. 54 (cit. on p. 146).

105. *Diffprivlib: IBM's differential privacy library*. `https://github.com/IBM/differential-privacy-library` (Accessed: 2020-08-18) (cit. on p. 157).

106. Dimitrakakis, C., Nelson, B., Zhang, Z., Mitrokotsa, A. & Rubinstein, B. I. P. *Differential privacy for bayesian inference through posterior sampling*. Journal of Machine Learning Research vol. 18(1), p. 343 (2017) (cit. on pp. 55, 72, 79).

107. Ding, B., Kulkarni, J. & Yekhanin, S. *Collecting telemetry data privately*. Advances in Neural Information Processing Systems (2017) (cit. on pp. 30, 81, 157).

108. Ding, X., Wang, W., Wan, M. & Gu, M. *Seamless Privacy: Privacy-Preserving Subgraph Counting in Interactive Social Network Analysis*. Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), 2013 International Conference on (2013) (cit. on p. 49).

109. Ding, Z., Wang, Y., Wang, G., Zhang, D. & Kifer, D. *Detecting Violations of Differential Privacy*. Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (ACM, Toronto, Canada, 2018), p. 475. (Cit. on p. 181).

110. Dinur, I. & Nissim, K. *Revealing information while preserving privacy*. Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems (2003) (cit. on pp. 40 sq.).

111. Dixit, K., Jha, M., Raskhodnikova, S. & Thakurta, A. *Testing Lipschitz Property over Product Distribution and its Applications to Statistical Data Privacy*. Theory of Cryptography - Lecture Notes in Computer Science (2013) (cit. on pp. 53, 72).

112. Dobbe, R., Pu, Y., Zhu, J., Ramchandran, K. & Tomlin, C. *Customized Local Differential Privacy for Multi-Agent Distributed Optimization*. arXiv preprint arXiv:1806.06035 (2018) (cit. on p. 82).

113. Dong, J., Roth, A. & Su, J. W. *Gaussian differential privacy*. arXiv preprint arXiv:1905.02383 (2019) (cit. on pp. 60, 74, 76, 79).

114. Dong, J., Durfee, D. & Rogers, R. *Optimal Differential Privacy Composition for Exponential Mechanisms*. ICML 2020: 37th International Conference on Machine Learning (2020) (cit. on p. 30).

115. Dong, K., Guo, T., Ye, H., Li, X. & Ling, Z. *On the limitations of existing notions of location privacy*. Future Generation Computer Systems (2018) (cit. on p. 61).

116. Duan, Y. *Privacy without noise*. Proceedings of the 18th ACM conference on Information and knowledge management (2009) (cit. on pp. 56, 73, 80 sq., 87 sq.).

117. Duchi, J. C., Jordan, M. I. & Wainwright, M. J. *Local privacy and statistical minimax rates*. Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on (2013) (cit. on p. 81).

118. Duchi, J. C. & Ruan, F. *The Right Complexity Measure in Locally Private Estimation: It is not the Fisher Information*. arXiv preprint arXiv:1806.05756 (2018) (cit. on pp. 45, 70).

119. Du Pin Calmon, F. & Fawaz, N. *Privacy against statistical inference*. Communication, Control, and Computing (Allerton), 2012 50th Annual Allerton Conference on (2012) (cit. on pp. 61, 63, 74).

120. Durand, M. & Flajolet, P. *Loglog counting of large cardinalities*. European Symposium on Algorithms (2003) (cit. on pp. 126, 131, 215).

121. Durfee, D. & Rogers, R. M. *Practical Differentially Private Top-k Selection with Pay-what-you-get Composition*. Advances in Neural Information Processing Systems (2019) (cit. on pp. 63, 74).

122. Dwork, C. *Differential Privacy*. Proceedings of the 33rd international conference on Automata, Languages and Programming (2006) (cit. on pp. 25, 30, 62).

123. Dwork, C. *Differential privacy: A survey of results*. International Conference on Theory and Applications of Models of Computation (2008) (cit. on pp. 47, 70, 83).

124. Dwork, C. *The differential privacy frontier*. Theory of Cryptography Conference (2009) (cit. on p. 83).

125. Dwork, C. *Differential privacy in new settings*. Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms (2010) (cit. on p. 49).

126. Dwork, C., Kenthapadi, K., McSherry, F., Mironov, I. & Naor, M. *Our Data, Ourselves: Privacy Via Distributed Noise Generation.* Eurocrypt (2006) (cit. on pp. 41, 69, 78).

127. Dwork, C. & Lei, J. *Differential privacy and robust statistics*. Proceedings of the forty-first annual ACM symposium on Theory of computing (2009), p. 371 (cit. on pp. 179, 214).

128. Dwork, C., McSherry, F., Nissim, K. & Smith, A. *Calibrating noise to sensitivity in private data analysis*. Theory of Cryptography Conference (2006) (cit. on pp. 25, 33, 81, 179, 183).

129. Dwork, C., Naor, M., Pitassi, T. & Rothblum, G. N. *Differential privacy under continual observation*. Proceedings of the forty-second ACM symposium on Theory of computing (2010) (cit. on p. 49).

130. Dwork, C., Naor, M., Pitassi, T., Rothblum, G. N. & Yekhanin, S. *Pan-Private Streaming Algorithms.* ICS (2010), p. 66 (cit. on p. 49).

131. Dwork, C., Roth, A., *et al. The algorithmic foundations of differential privacy*. Foundations and Trends® in Theoretical Computer Science (2014) (cit. on pp. 40, 45, 83, 122, 230).

132. Dwork, C. & Rothblum, G. N. *Concentrated differential privacy*. arXiv preprint arXiv:1603.01887 (2016) (cit. on pp. 30, 43 sq., 69, 78).

133. Dwork, C., Rothblum, G. N. & Vadhan, S. *Boosting and differential privacy*. Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on (2010), p. 51 (cit. on pp. 30, 146, 229).

134. Ebadi, H., Sands, D. & Schneider, G. *Differential privacy: Now it's getting personal*. Acm Sigplan Notices (2015) (cit. on pp. 52, 71).

135. Eckersley, P. *How unique is your web browser?* International Symposium on Privacy Enhancing Technologies Symposium (2010), p. 1 (cit. on p. 163).

136. Egert, R., Fischlin, M., Gens, D., Jacob, S., Senker, M. & Tillmanns, J. *Privately computing set-union and set-intersection cardinality via Bloom filters*. Australasian Conference on Information Security and Privacy (2015) (cit. on p. 128).

137. El Emam, K. & Dankar, F. K. *Protecting privacy using k-anonymity*. Journal of the American Medical Informatics Association (2008) (cit. on pp. 11, 16, 234).

138. ElSalamouny, E. & Gambs, S. *Differential privacy models for location-based services*. Transactions on Data Privacy (2016) (cit. on pp. 48, 54, 81).

139. Erlingsson, Ú., Feldman, V., Mironov, I., Raghunathan, A., Song, S., Talwar, K. & Thakurta, A. *Encode, shuffle, analyze privacy revisited: formalizations and empirical evaluation*. arXiv preprint arXiv:2001.03618 (2020) (cit. on p. 125).

140. Erlingsson, Ú., Feldman, V., Mironov, I., Raghunathan, A., Talwar, K. & Thakurta, A. *Amplification by shuffling: From local to central differential privacy via anonymity*. Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms (2019), p. 2468 (cit. on p. 107).

141. Erlingsson, Ú., Pihur, V. & Korolova, A. *RAPPOR: Randomized aggregatable privacy-preserving ordinal response*. Proceedings of the 2014 ACM SIGSAC conference on computer and communications security (2014) (cit. on pp. 30, 52, 81, 129, 157).

142. Estan, C., Varghese, G. & Fisk, M. *Bitmap algorithms for counting active flows on high speed links*. Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement (2003) (cit. on p. 126).

143. *Evaluating KHLL accuracy with BigQuery*. `https://github.com/google/khll-paper-experiments`. (Accessed: 2020-09-08) (cit. on p. 173).

144. Evfimievski, A., Gehrke, J. & Srikant, R. *Limiting privacy breaches in privacy preserving data mining*. Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems (2003) (cit. on pp. 26, 30).

145. Fang, C. & Chang, E.-C. *Differential privacy with delta-neighbourhood for spatial and dynamic datasets*. Proceedings of the 9th ACM symposium on Information, computer and communications security (2014) (cit. on pp. 51, 71).

146. Fanti, G., Pihur, V. & Erlingsson, Ú. *Building a rappor with the unknown: Privacy-preserving learning of associations and data dictionaries*. Proceedings on Privacy Enhancing Technologies vol. 2016(3), p. 41 (2016) (cit. on p. 129).

147. Farokhi, F. *Noiseless Privacy*. arXiv preprint arXiv:1910.13027 (2019) (cit. on pp. 56, 81).

148. Farokhi, F. *Temporally Discounted Differential Privacy for Evolving Datasets on an Infinite Horizon*. 2020 ACM/IEEE 11th International Conference on Cyber-Physical Systems (ICCPS) (2020), p. 1 (cit. on p. 49).

149. Feldman, V., Mironov, I., Talwar, K. & Thakurta, A. *Privacy amplification by iteration*. 2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS) (2018) (cit. on pp. 45 sq., 70, 125).

150. Feldman, V. & Steinke, T. *Calibrating noise to variance in adaptive data analysis*. Proceedings of Machine Learning Research (2018) (cit. on pp. 55, 72).

151. Fernandes, N., Dras, M. & McIver, A. *Generalised differential privacy for text document processing*. International Conference on Principles of Security and Trust (2019) (cit. on pp. 54, 72, 81).

152. Flajolet, P., Fusy, É., Gandouet, O. & Meunier, F. *HyperLogLog: the analysis of a near-optimal cardinality estimation algorithm*. DMTCS Proceedings (2008) (cit. on pp. 127, 164 sq., 178).

153. Flajolet, P. & Martin, G. N. *Probabilistic counting*. 24th Annual Symposium on Foundations of Computer Science (sfcs 1983) (1983), p. 76 (cit. on p. 178).

154. Flajolet, P. & Martin, G. N. *Probabilistic counting algorithms for data base applications*. Journal of computer and system sciences (1985) (cit. on pp. 126, 131).

155. Francis, P., Eide, S. P. & Munz, R. *Diffix: High-utility database anonymization*. Annual Privacy Forum (2017), p. 141 (cit. on p. 181).

156. Gaboardi, M., Hay, M. & Vadhan, S. *A Programming Framework for OpenDP* (2020) (cit. on p. 181).

157. Gadotti, A., Houssiau, F., Rocher, L., Livshits, B. & De Montjoye, Y.-A. *When the signal is in the noise: exploiting diffix's sticky noise*. 28th {USENIX} Security Symposium ({USENIX} Security 19) (2019), p. 1081 (cit. on p. 182).

158. Ganta, S. R., Kasiviswanathan, S. P. & Smith, A. *Composition attacks and auxiliary information in data privacy*. Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining (2008) (cit. on pp. 23, 63, 74).

159. Garfinkel, S. L., Abowd, J. M. & Powazek, S. *Issues Encountered Deploying Differential Privacy*. Proceedings of the 2018 Workshop on Privacy in the Electronic Society (2018) (cit. on pp. 30, 157).

160. Gehrke, J., Hay, M., Lui, E. & Pass, R. Advances in Cryptology–CRYPTO 2012 (Springer, 2012) (cit. on p. 80).

161. Gehrke, J., Lui, E. & Pass, R. *Towards privacy for social networks: A zero-knowledge based definition of privacy*. Theory of Cryptography Conference (2011) (cit. on pp. 64 sq., 67, 74 sq.).

162. Geumlek, J. & Chaudhuri, K. *Profile-based Privacy for Locally Private Computations*. 2019 IEEE International Symposium on Information Theory (ISIT) (2019), p. 537 (cit. on pp. 59, 73, 81).

163. Geumlek, J., Song, S. & Chaudhuri, K. *Renyi differential privacy mechanisms for posterior sampling*. Advances in Neural Information Processing Systems (2017) (cit. on p. 45).

164. Ghosh, A. & Kleinberg, R. *Inferential Privacy Guarantees for Differentially Private Mechanisms.* 8th Innovations in Theoretical Computer Science Conference (ITCS 2017). vol. 67 (2017), p. 3 (cit. on pp. 63, 73).

165. Ghosh, A. & Roth, A. *Selling privacy at auction*. Games and Economic Behavior (2015) (cit. on pp. 52, 71).

166. Ghosh, A., Roughgarden, T. & Sundararajan, M. *Universally utility-maximizing privacy mechanisms*. SIAM Journal on Computing vol. 41(6), p. 1673 (2012) (cit. on p. 34).

167. Gilbert, A. C. & Mcmillan, A. *Property Testing For Differential Privacy*. 2018 56th Annual Allerton Conference on Communication, Control, and Computing (Allerton) (2018), p. 249 (cit. on pp. 181, 212).

168. Gkoulalas-Divanis, A., Loukides, G. & Sun, J. *Publishing data from electronic health records while preserving privacy: A survey of algorithms*. Journal of biomedical informatics vol. 50, p. 4 (2014) (cit. on p. 13).

169. Goldwasser, S. & Micali, S. *Probabilistic encryption*. Journal of computer and system sciences (1984) (cit. on p. 25).

170. Goldwasser, S., Micali, S. & Rackoff, C. *The knowledge complexity of interactive proof systems*. SIAM Journal on computing (1989) (cit. on p. 64).

171. Golle, P. & Partridge, K. *On the anonymity of home/work location pairs*. Pervasive computing (2009) (cit. on p. 127).

172. *Google Scholar*. `https://scholar.google.com/` (Accessed: 2020-10-07) (cit. on p. 40).

173. *Google's Differential Privacy library*. `https://github.com/google/differential-privacy` (Accessed: 2020-08-18) (cit. on p. 157).

174. Gotz, M., Machanavajjhala, A., Wang, G., Xiao, X. & Gehrke, J. *Publishing search logs—a comparative study of privacy guarantees*. IEEE Transactions on Knowledge and Data Engineering vol. 24(3), p. 520 (2011) (cit. on p. 181).

175. Grant, H. *"If I die, that is OK": the Calais refugees with nowhere to turn*. The Guardian. (2020) (cit. on p. 241).

176. Grining, K. & Klonowski, M. *Towards Extending Noiseless Privacy: Dependent Data and More Practical Approach*. Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security (2017), p. 546 (cit. on pp. 104, 106 sq., 125, 129, 137).

177. Groce, A., Katz, J. & Yerukhimovich, A. *Limits of computational differential privacy in the client/server setting*. Theory of Cryptography Conference (2011) (cit. on p. 67).

178. Guerraoui, R., Kermarrec, A.-M., Patra, R. & Taziki, M. *D 2 p: distance-based differential privacy in recommenders*. Proceedings of the VLDB Endowment (2015) (cit. on p. 50).

179. Gulhane, P., Gopi, S., Kulkarni, J., Shen, J. H., Shokouhi, M. & Yekhanin, S. *Differentially Private Set Union*. ICML 2020: 37th International Conference on Machine Learning (2020) (cit. on pp. 215 sq., 225).

180. Gursoy, M. E., Tamersoy, A., Truex, S., Wei, W. & Liu, L. *Secure and Utility-Aware Data Collection with Condensed Local Differential Privacy*. IEEE Transactions on Dependable and Secure Computing, p. 1 (2019) (cit. on p. 82).

181. Haeberlen, A., Pierce, B. C. & Narayan, A. *Differential Privacy Under Fire.* USENIX Security Symposium. vol. 33 (2011) (cit. on p. 236).

182. Haitner, I., Mazor, N., Shaltiel, R. & Silbak, J. *Channels of Small Log-Ratio Leakage and Characterization of Two-Party Differentially Private Computation*. Theory of Cryptography Conference (2019) (cit. on p. 25).

183. Halevy, A., Korn, F., Noy, N. F., Olston, C., Polyzotis, N., Roy, S. & Whang, S. E. *Goods: Organizing google's datasets*. Proceedings of the 2016 International Conference on Management of Data (2016), p. 795 (cit. on p. 178).

184. Hall, R. *New Statistical Applications for Differential Privacy*. PhD thesis (PhD thesis, Carnegie Mellon, 2012) (cit. on pp. 53, 72, 79).

185. Hall, R. J., Wasserman, L. A. & Rinaldo, A. *Random Differential Privacy*. Journal of Privacy and Confidentiality vol. 4(2), p. 3 (2013) (cit. on pp. 53, 72).

186. Halton, J. H. *Algorithm 247: Radical-inverse Quasi-random Point Sequence*. Commun. ACM vol. 7(12), p. 701. (1964) (cit. on p. 208).

187. Haney, S., Machanavajjhala, A. & Ding, B. *Design of policy-aware differentially private algorithms*. Proceedings of the VLDB Endowment (2015) (cit. on pp. 51, 71).

188. Hay, M., Li, C., Miklau, G. & Jensen, D. *Accurate estimation of the degree distribution of private networks*. Data Mining, 2009. ICDM'09. Ninth IEEE International Conference on (2009) (cit. on p. 49).

189. Hay, M., Machanavajjhala, A., Miklau, G., Chen, Y. & Zhang, D. *Principled evaluation of differentially private algorithms using dpbench*. Proceedings of the 2016 International Conference on Management of Data (2016), p. 139 (cit. on pp. 83, 239).

190. Hay, M., Machanavajjhala, A., Miklau, G., Chen, Y., Zhang, D. & Bissias, G. *Exploring privacy-accuracy tradeoffs using dpcomp*. Proceedings of the 2016 International Conference on Management of Data (2016), p. 2101 (cit. on p. 239).

191. Hayes, J., Melis, L., Danezis, G. & De Cristofaro, E. *LOGAN: Membership inference attacks against generative models*. Proceedings on Privacy Enhancing Technologies vol. 2019(1), p. 133 (2019) (cit. on p. 234).

192. *Hazy*. `https://hazy.com/` (Accessed: 2020-08-18) (cit. on p. 157).

193. He, X., Machanavajjhala, A. & Ding, B. *Blowfish privacy: Tuning privacy-utility tradeoffs using policies*. Proceedings of the 2014 ACM SIGMOD international conference on Management of data (2014) (cit. on pp. 51, 71).

194. He, X., Machanavajjhala, A., Flynn, C. & Srivastava, D. *Composing Differential Privacy and Secure Computation: A case study on scaling private record linkage*. Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (2017) (cit. on pp. 67, 75).

195. Heule, S., Nunkesser, M. & Hall, A. *HyperLogLog in practice: algorithmic engineering of a state of the art cardinality estimation algorithm*. Proceedings of the 16th International Conference on Extending Database Technology (2013) (cit. on pp. 127, 132, 154, 164, 166 sq., 178).

196. Heurix, J., Zimmermann, P., Neubauer, T. & Fenz, S. *A taxonomy for privacy enhancing technologies*. Computers & Security (2015) (cit. on p. 82).

197. Holohan, N., Antonatos, S., Braghin, S. & Mac Aonghusa, P. *(k,e)-Anonymity: k-Anonymity with e-Differential Privacy*. arXiv preprint arXiv:1710.01615 (2017) (cit. on p. 80).

198. *How Google Anonymizes Data*. `https://policies.google.com/technologies/anonymization` (Accessed: 2020-08-18) (cit. on p. 7).

199. *How the Census Bureau Protects Your Data*. https://2020census.gov/en/data-protection.html (Accessed: 2020-08-18) (cit. on p. 7).

200. Hsu, J., Gaboardi, M., Haeberlen, A., Khanna, S., Narayan, A., Pierce, B. C. & Roth, A. *Differential privacy: An economic method for choosing epsilon*. 2014 IEEE 27th Computer Security Foundations Symposium (2014) (cit. on pp. 83, 204).

201. Huang, Y. & Dai, H. *Quantifying Differential Privacy of Gossip Protocols in General Networks*. arXiv preprint arXiv:1905.07598 (2019) (cit. on p. 50).

202. Huber, M., Müller-Quade, J. & Nilges, T. Number Theory and Cryptography (Springer, 2013) (cit. on p. 38).

203. *HyperLogLog implementation for mssql*. https://github.com/shuvava/mssql-hll (Accessed: 2020-08-18) (cit. on p. 127).

204. *International Classification of Diseases, 10th Revision*. https://www.who.int/classifications/icd/icdonlineversions/en/ (Accessed: 2020-08-18) (cit. on p. 13).

205. Jaccard, P. *Lois de distribution florale dans la zone alpine*. vol. 38, p. 69 (1902) (cit. on pp. 160, 164).

206. Jelasity, M. & Birman, K. P. *Distributional differential privacy for large-scale smart metering*. Proceedings of the 2nd ACM workshop on Information hiding and multimedia security (2014) (cit. on pp. 59, 73).

207. Jiang, B., Li, M. & Tandon, R. *Context-aware Data Aggregation with Localized Information Privacy*. 2018 IEEE Conference on Communications and Network Security (CNS) (2018), p. 1 (cit. on p. 82).

208. Johnson, N. & Near, J. P. *Dataflow analysis & differential privacy for SQL queries*. https://github.com/uber/sql-differential-privacy (Accessed: 2019-09-04) (cit. on p. 184).

209. Johnson, N., Near, J. P. & Song, D. *Towards practical differential privacy for SQL queries*. Proceedings of the VLDB Endowment (2018) (cit. on pp. 81, 180 sq., 188, 197).

210. Johnson, N., Near, J. P., Hellerstein, J. M. & Song, D. *Chorus: Differential Privacy via Query Rewriting*. arXiv preprint arXiv:1809.07750 (2018) (cit. on pp. 30, 157).

211. Jones, A., Leahy, K. & Hale, M. *Towards differential privacy for symbolic systems*. 2019 American Control Conference (ACC) (2019) (cit. on p. 49).

212. Jorgensen, Z., Yu, T. & Cormode, G. *Conservative or liberal? personalized differential privacy*. Data Engineering (ICDE), 2015 IEEE 31st International Conference on (2015) (cit. on pp. 52, 71).

213. Kairouz, P., Oh, S. & Viswanath, P. *The composition theorem for differential privacy*. IEEE Transactions on Information Theory (2017) (cit. on pp. 30, 60, 181, 229 sq.).

214. Karnin, Z., Lang, K. & Liberty, E. *Optimal quantile approximation in streams*. 2016 ieee 57th annual symposium on foundations of computer science (focs) (2016), p. 71 (cit. on pp. 164, 214).

215. Karp, R. M. & Kleinberg, R. *Noisy binary search and its applications*. Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms (2007), p. 881 (cit. on pp. 189, 214).

216. Kartal, H. B., Liu, X. & Li, X.-B. *Differential privacy for the vast majority*. ACM Transactions on Management Information Systems (TMIS) (2019) (cit. on p. 53).

217. Kasiviswanathan, S. P. & Smith, A. *On the 'semantics' of differential privacy: A Bayesian formulation*. Journal of Privacy and Confidentiality (2014) (cit. on pp. 62 sq., 74).

218. Kawamoto, Y. & Murakami, T. *Local distribution obfuscation via probability coupling*. 2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton) (2019) (cit. on pp. 51, 55, 59, 71, 73).

219. Kawamoto, Y. & Murakami, T. *Local obfuscation mechanisms for hiding probability distributions*. European Symposium on Research in Computer Security (2019) (cit. on pp. 54, 59, 72 sq., 81).

220. Kearns, M., Pai, M., Roth, A. & Ullman, J. *Mechanism design in large games: Incentives and privacy*. Proceedings of the 5th conference on Innovations in theoretical computer science (2014) (cit. on p. 82).

221. Kearns, M., Roth, A., Wu, Z. S. & Yaroslavtsev, G. *Private algorithms for the protected in social network search*. Proceedings of the National Academy of Sciences (2016) (cit. on p. 49).

222. Kellaris, G., Kollios, G., Nissim, K. & O'Neill, A. *Accessing data while preserving privacy*. arXiv preprint arXiv:1706.01552 (2017) (cit. on p. 49).

223. Kellaris, G., Papadopoulos, S., Xiao, X. & Papadias, D. *Differentially private event sequences over infinite streams*. Proceedings of the VLDB Endowment (2014) (cit. on p. 49).

224. Kenthapadi, K. & Tran, T. T. *PriPeARL: A Framework for Privacy-Preserving Analytics and Reporting at LinkedIn*. Proceedings of the 27th ACM International Conference on Information and Knowledge Management (2018) (cit. on pp. 30, 157).

225. Kifer, D. & Lin, B.-R. *Towards an axiomatization of statistical privacy and utility*. Proceedings of the twenty-ninth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems (2010) (cit. on pp. 30, 35, 38, 50, 55, 73, 76, 100).

226. Kifer, D. & Lin, B.-R. *An axiomatic view of statistical privacy and utility*. Journal of Privacy and Confidentiality (2012) (cit. on pp. 30, 35, 38, 50, 55).

227. Kifer, D. & Machanavajjhala, A. *No free lunch in data privacy*. Proceedings of the 2011 ACM SIGMOD International Conference on Management of data (2011), p. 193 (cit. on pp. 46, 48, 50 sq., 70 sq., 182).

228. Kifer, D. & Machanavajjhala, A. *A rigorous and customizable framework for privacy*. Proceedings of the 31st ACM SIGMOD-SIGACT-SIGAI symposium on Principles of Database Systems (2012) (cit. on pp. 51, 56, 58, 73, 75 sqq., 87).

229. Kingsley, P. & Shoumali, K. *Taking Hard Line, Greece Turns Back Migrants by Abandoning Them at Sea*. The New York Times. (2020) (cit. on p. 241).

230. Korolova, A., Kenthapadi, K., Mishra, N. & Ntoulas, A. *Releasing search queries and clicks privately*. Proceedings of the 18th international conference on World wide web (2009), p. 171 (cit. on pp. 181, 184, 190 sq., 215 sq., 222).

231. Kotsogiannis, I., Doudalis, S., Haney, S., Machanavajjhala, A. & Mehrotra, S. *One-sided Differential Privacy*. 2020 IEEE 36th International Conference on Data Engineering (ICDE) (2020), p. 493 (cit. on pp. 47, 70).

232. Kotsogiannis, I., Tao, Y., He, X., Fanaeepour, M., Machanavajjhala, A., Hay, M. & Miklau, G. *PrivateSQL: a differentially private SQL query engine*. Proceedings of the VLDB Endowment vol. 12(11), p. 1371 (2019) (cit. on p. 181).

233. Kotsogiannis, I., Tao, Y., Machanavajjhala, A., Miklau, G. & Hay, M. *Architecting a Differentially Private SQL Engine*. Conference on Innovative Data Systems Research (2019) (cit. on pp. 180 sq.).

234. Krehbiel, S. *Choosing Epsilon for Privacy as a Service*. Proceedings on Privacy Enhancing Technologies (2019) (cit. on pp. 55, 72, 83, 204).

235. Krishnan, V. & Martinez, S. *A Probabilistic Framework for Moving-Horizon Estimation: Stability and Privacy Guarantees*. IEEE Transactions on Automatic Control, p. 1 (2020) (cit. on pp. 51, 71).

236. *Latency Analysis Dashboard — Apigee Edge Documentation*. `https://docs.apigee.com/api-platform/analytics/latency-analysis-dashboard` (Accessed: 2020-10-16) (cit. on p. 214).

237. Laud, P. & Pankova, A. *Interpreting Epsilon of Differential Privacy in Terms of Advantage in Guessing or Approximating Sensitive Attributes*. arXiv preprint arXiv:1911.12777 (2019) (cit. on p. 83).

238. Laud, P., Pankova, A. & Martin, P. *Achieving Differential Privacy using Methods from Calculus*. arXiv preprint arXiv:1811.06343 (2018) (cit. on p. 81).

239. Le Quéré, C., Jackson, R. B., Jones, M. W., Smith, A. J., Abernethy, S., Andrew, R. M., De-Gol, A. J., Willis, D. R., Shan, Y., Canadell, J. G., *et al. Temporary reduction in daily global CO2 emissions during the COVID-19 forced confinement*. Nature Climate Change, p. 1 (2020) (cit. on p. 241).

240. Leahy, S. *Most countries aren't hitting 2030 climate goals, and everyone will pay the price*. National Geographic. (2019) (cit. on p. 241).

241. *LeapYear*. `https://leapyear.io/` (Accessed: 2020-08-18) (cit. on p. 157).

242. Lecuyer, M., Atlidakis, V., Geambasu, R., Hsu, D. & Jana, S. *Certified robustness to adversarial examples with differential privacy*. 2019 IEEE Symposium on Security and Privacy (S&P) (2019) (cit. on p. 50).

243. Lee, J. & Clifton, C. *How much is enough? choosing ε for differential privacy*. International Conference on Information Security (2011) (cit. on pp. 83, 204).

244. Lee, J. & Clifton, C. *Differential identifiability*. Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining (2012) (cit. on pp. 61, 80).

245. Lee, J. & Kifer, D. *Concentrated differentially private gradient descent with adaptive per-iteration privacy budget*. Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (2018) (cit. on p. 45).

246. LeFevre, K., DeWitt, D. J. & Ramakrishnan, R. *Incognito: Efficient full-domain k-anonymity*. Proceedings of the 2005 ACM SIGMOD international conference on Management of data (2005) (cit. on pp. 12, 23).

247. LeFevre, K., DeWitt, D. J. & Ramakrishnan, R. *Mondrian multidimensional k-anonymity*. Data Engineering, 2006. ICDE'06. Proceedings of the 22nd International Conference on (2006) (cit. on pp. 12, 23).

248. Leung, S. & Lui, E. *Bayesian mechanism design with efficiency, privacy, and approximate truthfulness*. International Workshop on Internet and Network Economics (2012) (cit. on pp. 48, 55, 59, 73).

249. Li, C., Hay, M., Miklau, G. & Wang, Y. *A data-and workload-aware algorithm for range queries under differential privacy*. Proceedings of the VLDB Endowment vol. 7(5), p. 341 (2014) (cit. on p. 181).

250. Li, J., Khodak, M., Caldas, S. & Talwalkar, A. *Differentially Private Meta-Learning*. ICLR 2020 : Eighth International Conference on Learning Representations (2020) (cit. on p. 82).

251. Li, N., Li, T. & Venkatasubramanian, S. *t-closeness: Privacy beyond k-anonymity and l-diversity*. Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on (2007) (cit. on p. 19).

252. Li, N., Lyu, M., Su, D. & Yang, W. *Differential privacy: From theory to practice*. Synthesis Lectures on Information Security, Privacy, & Trust vol. 8(4), p. 1 (2016) (cit. on pp. 83, 189, 209, 211).

253. Li, N., Qardaji, W. & Su, D. *On sampling, anonymization, and differential privacy or, k-anonymization meets differential privacy*. Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security (2012) (cit. on p. 61).

254. Li, N., Qardaji, W., Su, D., Wu, Y. & Yang, W. *Membership privacy: a unifying framework for privacy definitions*. Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security (2013) (cit. on pp. 61, 74, 80, 87).

255. Li, N., Qardaji, W. H. & Su, D. *Provably private data anonymization: Or, k-anonymity meets differential privacy*. CoRR, abs/1101.2604 (2011) (cit. on pp. 57 sq., 61, 73, 119).

256. Li, Y., Ren, X., Yang, S. & Yang, X. *Impact of Prior Knowledge and Data Correlation on Privacy Leakage: A Unified Analysis*. IEEE Transactions on Information Forensics and Security (2019) (cit. on pp. 48, 70).

257. Ligett, K., Peale, C. & Reingold, O. *Bounded-Leakage Differential Privacy*. 1st Symposium on Foundations of Responsible Computing (FORC 2020) (2020) (cit. on pp. 64, 74, 76, 79).

258. Liu, C., Chakraborty, S. & Mittal, P. *Dependence Makes You Vulnerable: Differential Privacy Under Dependent Tuples.* NDSS (2016) (cit. on pp. 47, 70, 81).

259. Liu, C., He, X., Chanyaswad, T., Wang, S. & Mittal, P. *Investigating statistical privacy frameworks from the perspective of hypothesis testing*. Proceedings on Privacy Enhancing Technologies (2019) (cit. on p. 83).

260. Liu, J., Xiong, L. & Luo, J. *Semantic Security: Privacy Definitions Revisited.* Trans. Data Privacy (2013) (cit. on pp. 60 sq., 73).

261. Liu, Z., Wang, Y.-X. & Smola, A. *Fast differentially private matrix factorization*. Proceedings of the 9th ACM Conference on Recommender Systems (2015) (cit. on pp. 52, 71).

262. Long, Y., Bindschaedler, V. & Gunter, C. A. *Towards measuring membership privacy*. arXiv preprint arXiv:1712.09136 (2017) (cit. on p. 50).

263. Lui, E. & Pass, R. *Outlier privacy*. Theory of Cryptography Conference (2015) (cit. on pp. 52 sq., 71 sq., 80).

264. Machanavajjhala, A., Gehrke, J. & Götz, M. *Data publishing against realistic adversaries*. Proceedings of the VLDB Endowment (2009) (cit. on pp. 57, 80).

265. Machanavajjhala, A., Gehrke, J., Kifer, D. & Venkitasubramaniam, M. *l-diversity: Privacy beyond k-anonymity*. Data Engineering, 2006. ICDE'06. Proceedings of the 22nd International Conference on (2006) (cit. on p. 17).

266. Machanavajjhala, A. & He, X. Handbook of Mobile Data Privacy (Springer, 2018) (cit. on p. 83).

267. Machanavajjhala, A., Kifer, D., Abowd, J., Gehrke, J. & Vilhuber, L. *Privacy: Theory meets practice on the map*. Proceedings of the 2008 IEEE 24th International Conference on Data Engineering (2008) (cit. on pp. 41, 69, 78).

268. Manku, G. S., Rajagopalan, S. & Lindsay, B. G. *Approximate medians and other quantiles in one pass and with limited memory*. ACM SIGMOD Record vol. 27(2), p. 426 (1998) (cit. on p. 164).

269. Marsaglia, G. & Bray, T. A. *A convenient method for generating normal variables*. SIAM review vol. 6(3), p. 260 (1964) (cit. on p. 231).

270. McClure, D. R. *Relaxations of differential privacy and risk/utility evaluations of synthetic data and fidelity measures*. PhD thesis (Duke University, 2015) (cit. on pp. 53, 72).

271. McMahan, H. B., Ramage, D., Talwar, K. & Zhang, L. *Learning Differentially Private Recurrent Language Models*. ICLR 2018 : International Conference on Learning Representations 2018 (2018) (cit. on pp. 46 sq.).

272. McSherry, F. *On "Differential Privacy as a Mutual Information Constraint"*. Blog. 2017. https://github.com/frankmcsherry/blog/blob/master/posts/2017-01-26.md (cit. on p. 43).

273. McSherry, F. D. *Privacy integrated queries: an extensible platform for privacy-preserving data analysis*. Proceedings of the 2009 ACM SIGMOD International Conference on Management of data (2009), p. 19 (cit. on pp. 180 sq., 188).

274. McSherry, F. D. *How many secrets do you have?* 2017. https://github.com/frankmcsherry/blog/blob/master/posts/2017-02-08.md (Accessed: 2020-10-16) (cit. on p. 235).

275. McSherry, F. D. *Synthethic Data via Differential Privacy*. https://github.com/frankmcsherry/blog/blob/master/assets/Synth-SIGMOD.pdf (Accessed: 2019-05-28) (cit. on p. 184).

276. McSherry, F. D. *Uber's differential privacy .. probably isn't*. https://github.com/frankmcsherry/blog/blob/master/posts/2018-02-25.md (Accessed: 2019-03-22) (cit. on pp. 185, 197).

277. Meiser, S. *Approximate and Probabilistic Differential Privacy Definitions*. Cryptology ePrint Archive, Report 2018/277 (2018) (cit. on pp. 41, 69, 76, 78).

278. Meiser, S. & Mohammadi, E. *Tight on Budget? Tight Bounds for r-Fold Approximate Differential Privacy*. Proceedings of the 25th ACM Conference on Computer and Communications Security (CCS) (ACM, 2018) (cit. on pp. 30, 98, 122, 181).

279. Melis, L., Danezis, G. & Cristofaro, E. D. *Efficient Private Statistics with Succinct Sketches*. In: Proceedings of the NDSS Symposium 2016. Internet Society: San Diego, CA, USA. (2016) (2016) (cit. on p. 128).

280. Messing, S., DeGregorio, C., Hillenbrand, B., King, G., Mahanti, S., Mukerjee, Z., Nayak, C., Persily, N., State, B. & Wilkins, A. Facebook Privacy-Protected Full URLs Data Set (Harvard Dataverse, 2020). https://doi.org/10.7910/DVN/TDOAPG/DGSAMS (cit. on pp. 234, 236).

281. Metwally, A., Agrawal, D. & El Abbadi, A. *Efficient computation of frequent and top-k elements in data streams*. International conference on database theory (2005), p. 398 (cit. on p. 164).

282. Meyerson, A. & Williams, R. *On the complexity of optimal k-anonymity*. Proceedings of the twenty-third ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems (2004), p. 223 (cit. on p. 13).

283. Mironov, I. *On significance of the least significant bits for differential privacy*. Proceedings of the 2012 ACM conference on Computer and communications security (2012), p. 650 (cit. on pp. 34, 204, 206, 231 sq., 236).

284. Mironov, I. *Renyi differential privacy*. Computer Security Foundations Symposium (CSF), 2017 IEEE 30th (2017) (cit. on pp. 30, 42, 69, 75, 78, 122, 147).

285. Mironov, I., Pandey, O., Reingold, O. & Vadhan, S. Advances in Cryptology-CRYPTO 2009 (Springer, 2009) (cit. on pp. 66 sq., 75).

286. Monreale, A., Wang, W. H., Pratesi, F., Rinzivillo, S., Pedreschi, D., Andrienko, G. & Andrienko, N. Geographic Information Science at the Heart of Europe (Springer, 2013) (cit. on p. 127).

287. Morse, J. *Sorry, your 'anonymized' data probably isn't anonymous*. Mashable. (2018) (cit. on p. 7).

288. Murakami, T., Hamada, K., Kawamoto, Y. & Hatano, T. *Privacy-Preserving Multiple Tensor Factorization for Synthesizing Large-Scale Location Traces*. arXiv preprint arXiv:1911.04226 (2019) (cit. on p. 234).

289. Murakami, T. & Kawamoto, Y. *Utility-optimized local differential privacy mechanisms for distribution estimation*. 28th {USENIX} Security Symposium ({USENIX} Security 19) (2019) (cit. on p. 82).

290. Naldi, M. & D'Acquisto, G. *Differential privacy: an estimation theory-based method for choosing epsilon*. arXiv preprint arXiv:1510.00917 (2015) (cit. on p. 204).

291. Naor, M. & Vexler, N. *Can Two Walk Together: Privacy Enhancing Methods and Preventing Tracking of Users*. 1st Symposium on Foundations of Responsible Computing (FORC 2020), p. 20 (2020) (cit. on pp. 48 sq.).

292. Narayan, A., Feldman, A., Papadimitriou, A. & Haeberlen, A. *Verifiable differential privacy*. Proceedings of the Tenth European Conference on Computer Systems (2015), p. 1 (cit. on p. 181).

293. Narayan, A. & Haeberlen, A. *DJoin: differentially private join queries over distributed databases*. Presented as part of the 10th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 12) (2012), p. 149 (cit. on p. 181).

294. Narayanan, A. & Shmatikov, V. *Robust de-anonymization of large sparse datasets*. 2008 IEEE Symposium on Security and Privacy (S&P) (2008), p. 111 (cit. on pp. 161, 170, 174).

295. Nasr, M., Shokri, R. & Houmansadr, A. *Machine learning with membership privacy using adversarial regularization*. Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (2018), p. 634 (cit. on p. 234).

296. Nelson, B. & Reuben, J. *Chasing Accuracy and Privacy, and Catching Both: A Literature Survey on Differentially Private Histogram Publication*. arXiv preprint arXiv:1910.14028 (2019) (cit. on p. 83).

297. Nergiz, M. E., Atzori, M. & Clifton, C. *Hiding the presence of individuals from shared databases*. Proceedings of the 2007 ACM SIGMOD international conference on Management of data (2007) (cit. on p. 21).

298. Nergiz, M. E. & Clifton, C. *δ-presence without complete world knowledge*. IEEE Transactions on Knowledge and Data Engineering vol. 22(6), p. 868 (2009) (cit. on p. 21).

299. *nextDouble() in Random.java — OpenJDK 8 Source Code*. http://hg.openjdk.java.net/jdk8/jdk8/jdk/file/tip/src/share/classes/java/util/Random.java%5C#l531 (Accessed: 2020-10-16) (cit. on p. 231).

300. *nextGaussian() in Random.java — OpenJDK 8 Source Code*. http://hg.openjdk.java.net/jdk8/jdk8/jdk/file/tip/src/share/classes/java/util/Random.java%5C#l554 (Accessed: 2020-10-16) (cit. on p. 231).

301. Nie, Y., Yang, W., Huang, L., Xie, X., Zhao, Z. & Wang, S. *A Utility-optimized Framework for Personalized Private Histogram Estimation*. IEEE Transactions on Knowledge and Data Engineering (2018) (cit. on p. 82).

302. Niknami, N., Abadi, M. & Deldar, F. *SpatialPDP: A personalized differentially private mechanism for range counting queries over spatial databases*. Computer and Knowledge Engineering (ICCKE), 2014 4th International eConference on (2014) (cit. on pp. 52, 71).

303. Nissim, K., Raskhodnikova, S. & Smith, A. *Smooth sensitivity and sampling in private data analysis*. Proceedings of the thirty-ninth annual ACM symposium on Theory of computing (2007) (cit. on pp. 81, 179, 181, 214).

304. Nissim, K., Steinke, T., Wood, A., Altman, M., Bembenek, A., Bun, M., Gaboardi, M., O'Brien, D. R. & Vadhan, S. *Differential privacy: A primer for a non-technical audience*. Privacy Law Scholars Conf (2017) (cit. on pp. 83, 204).

305. *normal.go — Go Source Code*. https://golang.org/src/math/rand/normal.go (Accessed: 2020-10-16) (cit. on p. 232).

306. *normalvariate() in random.py — Python 3.4 Source Code*. https://github.com/python/cpython/blob/05c28b08f6e2fc8782472b026c98a3fdd61a2ba9/Lib/random.py%5C#L370 (Accessed: 2020-10-16) (cit. on p. 231).

307. Nozari, E. *Networked Dynamical Systems: Privacy, Control, and Cognition*. PhD thesis (UC San Diego, 2019) (cit. on p. 50).

308. *Open Differential Privacy*. https://opendifferentialprivacy.github.io/ (Accessed: 2020-08-18) (cit. on p. 157).

309. Padmanabhan, S., Bhattacharjee, B., Malkemus, T., Cranston, L. & Huras, M. *Multidimensional clustering: A new data layout scheme in db2*. Proceedings of the 2003 ACM SIGMOD international conference on Management of data (2003) (cit. on p. 126).

310. Papapetrou, O., Siberski, W. & Nejdl, W. *Cardinality estimation and dynamic length adaptation for Bloom filters*. Distributed and Parallel Databases (2010) (cit. on p. 132).

311. Patel, S., Persiano, G. & Yeo, K. *What Storage Access Privacy is Achievable with Small Overhead?* Proceedings of the 38th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (2019) (cit. on p. 49).

312. Pejo, B., Tang, Q. & Biczók, G. *Together or Alone: The Price of Privacy in Collaborative Learning*. Proceedings on Privacy Enhancing Technologies (2019) (cit. on p. 83).

313. Percival, K. *Federal Laws That Protect Census Confidentiality*. https://www.brennancenter.org/sites/default/files/2019-08/Report_Federal_Laws_Census_Confidentiality.pdf (Accessed: 2020-08-18) (cit. on p. 7).

314. Perera, M. & Guevara, M. *Improving Usability of Differential Privacy at Scale*. https://youtube.com/watch?v=qWWgo4Nsx9M (Accessed: 2020-10-27) (cit. on p. 205).

315. *Phoenix Arizona Population and Demographics Resources*. https://phoenix.areaconnect.com/statistics.htm (Accessed: 2020-10-05) (cit. on p. 15).

316. Pinot, R. *Minimum spanning tree release under differential privacy constraints*. arXiv preprint arXiv:1801.06423 (2018) (cit. on p. 49).

317. Pinot, R., Yger, F., Gouy-Pailler, C. & Atif, J. *A unified view on differential privacy and robustness to adversarial examples*. Workshop on Machine Learning for CyberSecurity at ECMLPKDD 2019 (2019) (cit. on p. 45).

318. *PostgreSQL extension extension adding HyperLogLog data structures as a native data type*. https://github.com/citusdata/postgresql-hll (Accessed: 2018-05-03) (cit. on p. 127).

319. Poulson, J. *Reports of a Silicon Valley/Military Divide Have Been Greatly Exaggerated*. Tech Inquiry. (2020) (cit. on p. 241).

320. *Privacy on Beam*. https://github.com/google/differential-privacy/tree/main/privacy-on-beam (Accessed: 2020-09-23) (cit. on p. 226).

321. Project, H. P. T. *Randomness and Noise*. https://github.com/opendifferentialprivacy/whitenoise-core/blob/develop/whitepapers/noise/noise.pdf (Accessed: 2020-08-18) (cit. on p. 232).

322. Proserpio, D., Goldberg, S. & McSherry, F. *Calibrating data to sensitivity in private data analysis: a platform for differentially-private analysis of weighted datasets*. Proceedings of the VLDB Endowment (2014) (cit. on pp. 54, 72).

323. *Protocol Buffers*. https://developers.google.com/protocol-buffers/ (Accessed: 2020-09-08) (cit. on p. 172).

324. Pyrgelis, A., Troncoso, C. & Cristofaro, E. D. *Knock Knock, Who's There? Membership Inference on Aggregate Location Data*. In: Proceedings of the 25th Network and Distributed System Security Symposium (NDSS 2018). Internet Society: San Diego, CA, USA. (2018) (2018) (cit. on p. 234).

325. *random() in random.py — Python 3.4 Source Code*. https://github.com/python/cpython/blob/05c28b08f6e2fc8782472b026c98a3fdd61a2ba9/Lib/random.py%5C#L649 (Accessed: 2020-10-16) (cit. on p. 231).

326. Rastogi, V., Hay, M., Miklau, G. & Suciu, D. *Relationship privacy: output perturbation for queries with joins*. Proceedings of the twenty-eighth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems (2009) (cit. on pp. 61, 74, 80, 87).

327. Rebollo-Monedero, D., Forne, J. & Domingo-Ferrer, J. *From t-closeness-like privacy to postrandomization via information theory*. IEEE Transactions on Knowledge and Data Engineering (2010) (cit. on p. 146).

328. Rebollo-Monedero, D. & Forné, J. *Optimized query forgery for private information retrieval*. IEEE Transactions on Information Theory (2010) (cit. on p. 146).

329. Reuben, J. *Towards a Differential Privacy Theory for Edge-Labeled Directed Graphs*. SICHERHEIT 2018 (2018) (cit. on p. 49).

330. Reviriego, P. & Ting, D. *Security of HyperLogLog (HLL) Cardinality Estimation: Vulnerabilities and Protection*. IEEE Communications Letters vol. 24(5), p. 976 (2020) (cit. on p. 129).

331. Rocher, L., Hendrickx, J. M. & De Montjoye, Y.-A. *Estimating the success of re-identifications in incomplete datasets using generative models*. Nature communications vol. 10(1), p. 1 (2019) (cit. on p. 7).

332. Rogers, R., Subramaniam, S., Peng, S., Durfee, D., Lee, S., Kancha, S. K., Sahay, S. & Ahammad, P. *LinkedIn's Audience Engagements API: A privacy preserving data analytics system at scale*. arXiv preprint arXiv:2002.05839 (2020) (cit. on pp. 234, 236).

333. Roth, A. *New algorithms for preserving differential privacy*. Microsoft Research (2010) (cit. on pp. 51, 55, 66, 72).

334. Roy, I., Setty, S. T., Kilzer, A., Shmatikov, V. & Witchel, E. *Airavat: Security and privacy for MapReduce.* NSDI. vol. 10 (2010), p. 297 (cit. on p. 181).

335. Rubinstein, B. I. & Aldà, F. *Pain-free random differential privacy with sensitivity sampling*. Proceedings of the 34th International Conference on Machine Learning-Volume 70 (2017) (cit. on pp. 51, 71).

336. Sablayrolles, A., Matthijs, D., Schmid, C., Ollivier, Y. & Jegou, H. *White-box vs Black-box: Bayes Optimal Strategies for Membership Inference*. ICML 2019 : Thirty-sixth International Conference on Machine Learning. vol. 97 (2019), p. 5558 (cit. on pp. 61, 80).

337. Samarati, P. *Protecting respondents identities in microdata release*. IEEE transactions on Knowledge and Data Engineering (2001) (cit. on p. 10).

338. Samarati, P. & Sweeney, L. *Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression*. Tech. rep. (technical report, SRI International, 1998) (cit. on p. 12).

339. *sdcMicro: Statistical Disclosure Control Methods for Anonymization of Data and Risk Estimation*. `https://CRAN.R-project.org/package=sdcMicro` (Accessed: 2020-10-05) (cit. on p. 13).

340. Sealfon, A. *Shortest paths and distances with differential privacy*. Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (2016) (cit. on p. 49).

341. *Semantic types of protocol buffer fields can be annotated using custom options.* `https://developers.google.com/protocol-buffers/docs/proto%5C#options` (Accessed: 2020-09-08) (cit. on p. 172).

342. Sen, S., Guha, S., Datta, A., Rajamani, S. K., Tsai, J. & Wing, J. M. *Bootstrapping privacy compliance in big data systems*. 2014 IEEE Symposium on Security and Privacy (2014), p. 327 (cit. on p. 178).

343. Serwer, A. *A Crime by Any Name*. The Atlantic. (2019) (cit. on p. 241).

344. Shi, E., Chan, H., Rieffel, E., Chow, R. & Song, D. *Privacy-preserving aggregation of time-series data*. Annual Network & Distributed System Security Symposium (NDSS) (2011) (cit. on p. 82).

345. Shokri, R., Stronati, M., Song, C. & Shmatikov, V. *Membership inference attacks against machine learning models*. 2017 IEEE Symposium on Security and Privacy (SP) (2017), p. 3 (cit. on p. 234).

346. Shukla, A., Deshpande, P., Naughton, J. F. & Ramasamy, K. *Storage estimation for multidimensional aggregates in the presence of hierarchies*. VLDB (1996) (cit. on p. 126).

347. Simmons, S., Sahinalp, C. & Berger, B. *Enabling privacy-preserving GWASs in heterogeneous human populations*. Cell systems (2016) (cit. on p. 50).

348. Sommer, D. M., Meiser, S. & Mohammadi, E. *Privacy loss classes: The central limit theorem in differential privacy*. Proceedings on Privacy Enhancing Technologies (2019) (cit. on p. 45).

349. Soria-Comas, J., Domingo-Ferrer, J., Sánchez, D. & Megías, D. *Individual differential privacy: A utility-preserving formulation of differential privacy guarantees*. IEEE Transactions on Information Forensics and Security (2017) (cit. on pp. 48, 71).

350. Steinke, T. & Ullman, J. *The Pitfalls of Average-Case Differential Privacy*. `https://differentialprivacy.org/average-case-dp/` (Accessed: 2020-08-14) (cit. on p. 125).

351. Sun, H., Xiao, X., Khalil, I., Yang, Y., Qin, Z., Wang, H. W. & Yu, T. *Analyzing Subgraph Statistics from Extended Local Views with Decentralized Differential Privacy*. Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security (2019) (cit. on p. 49).

352. Sweeney, L. *Computational disclosure control: a primer on data privacy protection*. PhD thesis (Massachusetts Institute of Technology, 2001) (cit. on p. 15).

353. Sweeney, L. *Achieving k-anonymity privacy protection using generalization and suppression*. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems (2002) (cit. on p. 12).

354. Sweeney, L. *k-anonymity: A model for protecting privacy*. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems (2002) (cit. on p. 10).

355. Sweeney, L. *Only you, your doctor, and many others may know*. Technology Science vol. 2015092903(9), p. 29 (2015) (cit. on p. 9).

356. Szymkow, B. *Roadmap to Differential Privacy for All*. https://blog.openmined.org/making-algorithms-private/ (Accessed: 2020-08-18) (cit. on p. 157).

357. Takagi, S., Cao, Y., Asano, Y. & Yoshikawa, M. *Geo-Graph-Indistinguishability: Protecting Location Privacy for LBS over Road Networks*. IFIP Annual Conference on Data and Applications Security and Privacy (2019) (cit. on p. 54).

358. Tang, J., Korolova, A., Bai, X., Wang, X. & Wang, X. *Privacy loss in apple's implementation of differential privacy on macos 10.12*. arXiv preprint arXiv:1709.02753 (2017) (cit. on p. 234).

359. Task, C. & Clifton, C. *A guide to differential privacy theory in social network analysis*. Proceedings of the 2012 International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2012) (2012) (cit. on p. 49).

360. Team, D. P. *Learning with Privacy at Scale* (2017). (Cit. on pp. 30, 81, 157).

361. Team, G. D. P. *Secure Noise Generation*. https://github.com/google/differential-privacy/blob/main/common_docs/Secure_Noise_Generation.pdf (Accessed: 2020-09-25) (cit. on p. 232).

362. *The Effects of Climate Change*. https://climate.nasa.gov/effects/ (Accessed: 2020-09-09) (cit. on p. 241).

363. *The SECRETA system*. http://users.uop.gr/~poulis/SECRETA/index.html (Accessed: 2020-10-05) (cit. on p. 13).

364. Toledo, R. R., Danezis, G. & Goldberg, I. *Lower-cost e-private information retrieval*. Proceedings on Privacy Enhancing Technologies (2016) (cit. on p. 49).

365. *Tolerance Calculation for ApproxEquals in Tests — Privacy on Beam*. https://github.com/google/differential-privacy/blob/main/privacy-on-beam/docs/Tolerance_Calculation.pdf (Accessed: 2020-10-16) (cit. on p. 207).

366. Tossou, A. C. & Dimitrakakis, C. *Algorithms for differentially private multi-armed bandits*. Thirtieth AAAI Conference on Artificial Intelligence (2016) (cit. on p. 50).

367. Triastcyn, A. & Faltings, B. *Bayesian Differential Privacy for Machine Learning*. ICML 2020: 37th International Conference on Machine Learning (2020) (cit. on pp. 48, 55, 59, 72, 76, 79).

368. Tschantz, M. C., Sen, S. & Datta, A. *SoK: Differential Privacy as a Causal Property*. 2020 IEEE Symposium on Security and Privacy (SP) (2020) (cit. on pp. 37, 88 sqq., 92, 98).

369. Tschorsch, F. & Scheuermann, B. *An algorithm for privacy-preserving distributed user statistics*. Computer Networks (2013) (cit. on pp. 127 sq., 133).

370. *Tumult Labs*. `https://www.tmlt.io/` (Accessed: 2020-08-18) (cit. on p. 157).

371. *UTD Anonymization Toolbox*. `http://cs.utdallas.edu/dspl/cgi-bin/toolbox/index.php` (Accessed: 2020-10-05) (cit. on p. 13).

372. Vadhan, S. Tutorials on the Foundations of Cryptography, p. 347 (Springer, 2017) (cit. on p. 83).

373. Venkatadri, G., Andreou, A., Liu, Y., Mislove, A., Gummadi, K. P., Loiseau, P. & Goga, O. *Privacy Risks with Facebook's PII-Based Targeting: Auditing a Data Broker's Advertising Interface* (2018), p. 89 (cit. on p. 86).

374. Von Voigt, S. N. & Tschorsch, F. *RRTxFM: Probabilistic Counting for Differentially Private Statistics*. Conference on e-Business, e-Services and e-Society (2019), p. 86 (cit. on p. 128).

375. Wagh, S., Cuff, P. & Mittal, P. *Differentially private oblivious ram*. Proceedings on Privacy Enhancing Technologies (2018) (cit. on p. 49).

376. Wagner, I. & Eckhoff, D. *Technical privacy metrics: a systematic survey*. ACM Computing Surveys (CSUR) (2018) (cit. on p. 82).

377. Waldman, P., Chapman, L. & Robertson, J. *Palantir knows everything about you*. Bloomberg. (2018) (cit. on p. 241).

378. Wang, W., Ying, L. & Zhang, J. *On the relation between identifiability, differential privacy, and mutual-information privacy*. IEEE Transactions on Information Theory (2016) (cit. on pp. 81, 83).

379. Wang, Y.-X. *Per-instance Differential Privacy and the Adaptivity of Posterior Sampling in Linear and Ridge regression*. arXiv preprint arXiv:1707.07708 (2017) (cit. on pp. 48, 71, 81).

380. Wang, Y.-X., Balle, B. & Kasiviswanathan, S. P. *Subsampled Rényi Differential Privacy and Analytical Moments Accountant*. The 22nd International Conference on Artificial Intelligence and Statistics (2019), p. 1226 (cit. on pp. 43, 45, 69).

381. Wang, Y.-X., Lei, J. & Fienberg, S. E. *On-average kl-privacy and its equivalence to generalization for max-entropy mechanisms*. International Conference on Privacy in Statistical Databases (2016) (cit. on pp. 55, 72, 148).

382. Wang, Y., Sibai, H., Mitra, S. & Dullerud, G. E. *Differential Privacy for Sequential Algorithms*. arXiv preprint arXiv:2004.00275 (2020) (cit. on pp. 49, 81).

383. Warner, S. L. *Randomized response: A survey technique for eliminating evasive answer bias*. Journal of the American Statistical Association (1965) (cit. on p. 25).

384. Wasserman, L. *All of statistics: a concise course in statistical inference* (Springer Science & Business Media, 2013) (cit. on p. 209).

385. Wasserman, L. & Zhou, S. *A statistical framework for differential privacy*. Journal of the American Statistical Association (2010) (cit. on p. 60).

386. Weinstein, A. *This is Fascism*. The New Republic. (2020) (cit. on p. 241).

387. Whang, K.-Y., Vander-Zanden, B. T. & Taylor, H. M. *A linear-time probabilistic counting algorithm for database applications*. ACM Transactions on Database Systems (TODS) (1990) (cit. on p. 126).

388. Wilson, R. J., Zhang, C. Y., Lam, W., Desfontaines, D., Simmons-Marengo, D. & Gipson, B. *Differentially Private SQL with Bounded User Contribution*. Proceedings on Privacy Enhancing Technologies vol. 2020(2) (2020) (cit. on p. 5).

389. Wong, R. C.-W., Fu, A. W.-C., Wang, K. & Pei, J. *Anonymization-based attacks in privacy-preserving data publishing*. ACM Transactions on Database Systems (TODS) vol. 34(2), p. 1 (2009) (cit. on pp. 23, 117).

390. Wu, G., He, Y., Wu, J. & Xia, X. *Inherit differential privacy in distributed setting: Multiparty randomized function computation*. Trustcom/BigDataSE/I SPA, 2016 IEEE (2016) (cit. on p. 82).

391. Wu, G., Xia, X. & He, Y. *Information Theory of Data Privacy*. arXiv preprint arXiv:1703.07474 (2017) (cit. on pp. 61, 63, 73 sq.).

392. Wu, X., Dou, W. & Ni, Q. *Game theory based privacy preserving analysis in correlated data publication*. Proceedings of the Australasian Computer Science Week Multiconference (2017) (cit. on pp. 47, 70).

393. Wu, X., Wu, T., Khan, M., Ni, Q. & Dou, W. *Game theory based correlated privacy preserving analysis in big data*. IEEE Transactions on Big Data (2017) (cit. on pp. 47, 70).

394. Xiao, Q., Zhou, Y. & Chen, S. *Better with fewer bits: Improving the performance of cardinality estimation of large data streams*. IEEE INFOCOM 2017-IEEE Conference on Computer Communications (2017), p. 1 (cit. on p. 178).

395. Xiao, X. & Tao, Y. *M-invariance: towards privacy preserving re-publication of dynamic datasets*. Proceedings of the 2007 ACM SIGMOD international conference on Management of data (2007), p. 689 (cit. on p. 23).

396. Xiao, Y. & Xiong, L. *Protecting locations with differential privacy under temporal correlations*. Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (2015) (cit. on pp. 48, 54, 72).

397. Xu, F., Tu, Z., Li, Y., Zhang, P., Fu, X. & Jin, D. *Trajectory recovery from ash: User privacy is not preserved in aggregated mobility data*. Proceedings of the 26th international conference on world wide web (2017), p. 1241 (cit. on p. 234).

398. Yan, Z., Liu, J., Li, G., Han, Z. & Qiu, S. *PrivMin: Differentially Private MinHash for Jaccard Similarity Computation*. arXiv preprint arXiv:1705.07258 (2017) (cit. on p. 50).

399. Yang, B., Sato, I. & Nakagawa, H. *Bayesian differential privacy on correlated data*. Proceedings of the 2015 ACM SIGMOD international conference on Management of Data (2015) (cit. on pp. 48, 55, 59, 70).

400. Yao, C., Wang, X. S. & Jajodia, S. *Checking for k-anonymity violation by views*. Proceedings of the 31st international conference on Very large data bases (2005), p. 910 (cit. on p. 23).

401. Ying, X., Wu, X. & Wang, Y. *On linear refinement of differential privacy-preserving query answering*. Pacific-Asia Conference on Knowledge Discovery and Data Mining (2013) (cit. on p. 50).

402. Yu, Y. W. & Weber, G. M. *HyperMinHash: MinHash in LogLog space*. arXiv preprint arXiv:1710.08436 (2017) (cit. on p. 178).

403. Zenz, A. *'Thoroughly reforming them towards a healthy heart attitude': China's political re-education campaign in Xinjiang*. Central Asian Survey vol. 38(1), p. 102 (2019) (cit. on p. 241).

404. Zhang, D., McKenna, R., Kotsogiannis, I., Hay, M., Machanavajjhala, A. & Miklau, G. *Ektelo: A framework for defining differentially-private computations*. Proceedings of the 2018 International Conference on Management of Data (2018), p. 115 (cit. on p. 181).

405. Zhang, J., Sun, J., Zhang, R., Zhang, Y. & Hu, X. *Privacy-Preserving Social Media Data Outsourcing*. IEEE INFOCOM 2018-IEEE Conference on Computer Communications (2018) (cit. on p. 50).

406. Zhang, L., Jajodia, S. & Brodsky, A. *Information disclosure under realistic assumptions: Privacy versus optimality*. Proceedings of the 14th ACM conference on Computer and communications security (2007), p. 573 (cit. on p. 23).

407. Zhang, Z., Qin, Z., Zhu, L., Jiang, W., Xu, C. & Ren, K. *Toward practical differential privacy in smart grid with capacity-limited rechargeable batteries*. 2015 (cit. on pp. 42, 69, 78).

408. Zhao, J., Wang, T., Bai, T., Lam, K.-Y., Ren, X., Yang, X., Shi, S., Liu, Y. & Yu, H. *Reviewing and improving the Gaussian mechanism for differential privacy*. arXiv preprint arXiv:1911.12060 (2019) (cit. on p. 42).

409. Zhou, S., Ligett, K. & Wasserman, L. *Differential privacy with compression*. Information Theory, 2009. ISIT 2009. IEEE International Symposium on (2009) (cit. on pp. 51, 55, 71 sq.).

410. Zhu, T., Li, G., Ren, Y., Zhou, W. & Xiong, P. *Differential privacy for neighborhood-based collaborative filtering*. Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (2013) (cit. on p. 81).

411. Zhu, T., Xiong, P., Li, G. & Zhou, W. *Correlated differential privacy: hiding information in non-IID data set*. IEEE Transactions on Information Forensics and Security (2015) (cit. on p. 81).

412. Zhu, X. & Ghahramani, Z. *Learning from labeled and unlabeled data with label propagation*. Tech. Rep., Technical Report CMU-CALD-02–107, Carnegie Mellon University (2002) (cit. on p. 173).

CURRICULUM VITAE

PERSONAL DATA

Name	Damien Desfontaines
Date of Birth	October 16, 1992
Place of Birth	Lille, France
Citizen of	France

EDUCATION

2011–2014	École Normale Supérieure, Paris, France *Final degree:* Master's degree in Computer Science
2009–2011	Classes Préparatoires MPSI/MP* Lycée Berthollet Annecy, France
2006–2009	Lycée Vaugelas Chambéry, France *Final degree:* Baccalauréat scientifique

EMPLOYMENT

2014–	Senior Software Engineer Privacy & Data Protection Office *Google Gmbh*, Zürich, Switzerland
2014	Research intern *Laboratoire Poncelet*, Moscow, Russia
2013	Research intern *University of Auckland*, Auckland, New Zealand
2012	Google Summer of Code intern *GeoGebra institute*, Linz, Austria

PUBLICATIONS

1. Aktay, A., Bavadekar, S., Cossoul, G., Davis, J., Desfontaines, D., Fabrikant, A., Gabrilovich, E., Gadepalli, K., Gipson, B., Guevara, M., *et al. Google COVID-19 community mobility reports: Anonymization process description (version 1.0)*. arXiv preprint arXiv:2004.04145 (2020).
2. Bavadekar, S., Dai, A., Davis, J., Desfontaines, D., Eckstein, I., Everett, K., Fabrikant, A., Flores, G., Gabrilovich, E., Gadepalli, K., Glass, S., Huang, R., Kamath, C., Kraft, D., Kumok, A., Marfatia, H., Mayer, Y., Miller, B., Pearce, A., Perera, I. M., Ramachandran, V., Raman, K., Roessler, T., Shafran, I., Shekel, T., Stanton, C., Stimes, J., Sun, M., Wellenius, G. & Zoghi, M. *Google COVID-19 Search Trends Symptoms Dataset: Anonymization Process Description (version 1.0)*. arXiv preprint arXiv:2009.01265 (2020).
3. Chia, P. H., Desfontaines, D., Perera, I. M., Simmons-Marengo, D., Li, C., Day, W.-Y., Wang, Q. & Guevara, M. *KHyperLogLog: Estimating Reidentifiability and Joinability of Large Data at Scale*. 2019 IEEE Symposium on Security and Privacy (SP) (2019), p. 350.
4. Desfontaines, D., Lochbihler, A. & Basin, D. *Cardinality estimators do not preserve privacy*. Proceedings on Privacy Enhancing Technologies vol. 2019(2), p. 26 (2019).
5. Desfontaines, D., Mohammadi, E., Krahmer, E. & Basin, D. *Differential privacy with partial knowledge*. arXiv preprint arXiv:1905.00650 (2019).
6. Desfontaines, D. & Pejó, B. *Differential Privacies: a taxonomy of differential privacy variants and extensions (long version)*. arXiv prpeprint arXiv:1906.01337 (2019).
7. Desfontaines, D. & Pejó, B. *SoK: Differential Privacies*. Proceedings on Privacy Enhancing Technologies vol. 2020(2) (2020).
8. Desfontaines, D., Voss, J. & Gipson, B. *Differentially private partition selection* (2020).
9. Wellenius, G. A., Vispute, S., Espinosa, V., Fabrikant, A., Tsai, T. C., Hennessy, J., Williams, B., Gadepalli, K., Boulange, A., Pearce, A., *et al. Impacts of State-Level Policies on Social Distancing in the United States Using Aggregated Mobility Data during the COVID-19 Pandemic*. arXiv preprint arXiv:2004.10172 (2020).
10. Wilson, R. J., Zhang, C. Y., Lam, W., Desfontaines, D., Simmons-Marengo, D. & Gipson, B. *Differentially Private SQL with Bounded User Contribution*. Proceedings on Privacy Enhancing Technologies vol. 2020(2) (2020).